TABLE OF CONTENTS

Contributed Articles

Association for Computing Machinery

Advancing Computing as a Science & Profession

TABLE OF CONTENTS

Special Contribution from the first workshop on SafeAI at UAI 2025

Association for Computing Machinery

Advancing Computing as a Science & Profession

Uncertain Boundaries: Multidisciplinary Approaches to Copyright Issues in Generative AI

Archer Amon[1], Zhipeng Yin[1], Zichong Wang[1], Avash Palikhe[1], Tongjia Yu[2], and Wenbin Zhang[1*]
[1] Florida International University, Miami, FL, United States
[2] Goldman Sachs, New York, NY, United States

ABSTRACT

Generative AI is becoming increasingly prevalent in creative fields, sparking urgent debates over how current copyright laws can keep pace with technological innovation. Recent controversies of AI models generating near-replicas of copyrighted material highlight the need to adapt current legal frameworks and develop technical methods to mitigate copyright infringement risks. This task requires understanding the intersection between computational concepts such as large-scale data scraping and probabilistic content generation, legal definitions of originality and fair use, and economic impacts on intellectual property (IP) rights holders. However, most existing research on copyright in AI takes a purely computer science or law-based approach, leaving a gap in coordinating these approaches that only multidisciplinary efforts can effectively address. To bridge this gap, our survey adopts a comprehensive approach synthesizing insights from law, policy, economics, and computer science. It begins by discussing the foundational goals and considerations that should be applied to copyright in generative AI, followed by methods for detecting and assessing potential violations in AI system outputs. Next, it explores various regulatory options influenced by legal, policy, and economic frameworks to manage and mitigate copyright concerns associated with generative AI and reconcile the interests of IP rights holders with that of generative AI producers. The discussion then introduces techniques to safeguard individual creative works from unauthorized replication, such as watermarking and cryptographic protections. Finally, it describes advanced training strategies designed to prevent AI models from reproducing protected content. In doing so, we highlight key opportunities for action and offer actionable strategies that creators, developers, and policymakers can use in navigating the evolving copyright landscape.

Keywords

Generative AI, Copyright law, Intellectual property, AI policy, Copyright infringement detection

1. INTRODUCTION

The growing popularity of generative AI has reignited significant concerns around intellectual property (IP) rights, especially given that many commercial AI models rely heavily on datasets freely scraped from the internet [21]. This practice has resulted in several high-profile controversies, including real-world cases such as the legal dispute involving Stability AI's Stable Diffusion model allegedly replicating artists' styles without permission [42], and lawsuits filed by artists against companies like Midjourney and DeviantArt for unauthorized use of their works [38]. Additionally, literary authors have raised objections against large language models like ChatGPT for generating content closely resembling their copyrighted texts [109, 75, 111]. Companies have responded to these incidents in various ways: some have attempted to shift blame onto users who prompt models to create potentially infringing content [80], while others invoke broad and ambiguous interpretations of "fair use" to defend their practices, further complicating accountability and enforcement [29]. This ambiguity not only weakens the legal protections available to copyright holders but also undermines public trust and the perceived integrity of generative AI systems. Although technical measures to proactively mitigate these violations exist, such as content fingerprinting, watermarking, and cryptographic methods, their adoption has been slow and inconsistent [126]. Additionally, the rapidly evolving AI technology landscape and the absence of clear legal standards exacerbate these challenges, making it difficult for stakeholders, including creators, developers, and regulators, to effectively coordinate their efforts.

Addressing these challenges requires a holistic approach that combines technical solutions with supportive policy frameworks. To navigate these complexities, this paper aims to provide a comprehensive survey of current methods for enhancing copyright compliance in generative AI, focusing on four main goals: (1) detecting copyright violations and evaluating model performance, (2) understanding how regulatory landscapes shape technical strategies for protecting individual copyrighted works from unauthorized use, (3) protecting individual copyrighted works from being used in AI systems without authorization, and (4) designing AI models in a way that prevents generation of content violating copyright. In doing so, we adopt a uniquely multidisciplinary perspective, incorporating insights from computer science, law, policy, and economics to provide a more holistic framework for addressing copyright challenges in generative AI. Within this framework, we evaluate various methods based on their effectiveness and feasibility in preventing copyright violations, while also considering their impact on the utility of generative AI models. Through this survey, we aim to provide actionable insights for the development of technical,

*Corresponding author: wenbin.zhang@fiu.edu

legal, and policy strategies, enabling creators, developers, and policymakers to navigate the complex copyright challenges introduced by generative AI.

Paper Structure. The subsequent sections of this paper are structured as follows: Section 2 provides foundational background, explaining key concepts of generative AI and the complexities of copyright law as it applies to AI-generated content. Section 3 outlines a comprehensive taxonomy categorizing multidisciplinary methods addressing copyright issues in generative AI. Section 4 presents techniques for detecting and evaluating potential copyright infringements by AI models. Section 5 reviews regulatory frameworks and policy measures designed to manage and mitigate copyright risks associated with generative AI. Sections 6 and 7 discuss technical approaches aimed at protecting copyrighted content from unauthorized AI copying, as well as advanced training methods specifically designed to prevent generative AI models from producing infringing output. Finally, Sections 8 and 9 explore available resources, ongoing challenges, and future research directions in addressing copyright concerns within the evolving landscape of generative AI.

2. BACKGROUND

2.1 Generative AI Models

Definition. Generative AI broadly refers to artificial intelligence systems capable of creating new content, such as text, images, audio, or video, by learning patterns from extensive datasets. Compared to traditional AI systems, which focus on analyzing existing data or making decisions, generative AI models produce new outputs in response to user prompts [41]. These models are typically trained on large datasets, enabling them to generate content that mimics the style and structure of the data they were trained on, although small models are also an emerging field of research [118, 46, 59, 121]. Common model types include text-to-text, text-to-image, text-to-video, and image-to-video AI, as well as multimodal systems [50, 64, 72, 122] that integrate multiple input and output forms.

Model architecture. These models are typically trained on massive amounts of data, and utilize large architectures to encode inputs into a high-dimensional latent space and use a generator model to produce varied outputs through a stochastic behavior [41]. Most generative AI systems today are built on a transformer architecture consisting of an encoder and decoder, using a multi-head self-attention mechanism to handle long-term dependencies in data by assigning higher weights to more relevant tokens [13]. Because of the large amount of resources needed, generative models are often trained through a "pre-training" paradigm, where general purpose models are later fine-tuned for specific applications, creating a more complex chain of command where issues of indirect liability are more likely to come into play [120].

2.2 Copyright Applied to Generative AI

Fair use standards Broadly speaking, copyright allows the creator of an original work to prevent others from creating and profiting from unauthorized replication, distribution, or derivation of their works [83]. When evaluating copyright violations, US law has identified four main pillars that constitute "fair use" of copyrighted material: (1) the purpose and character of the use, (2) the nature of the copyrighted work, (3) the amount and substantiality of the portion taken, and (4) the effect of the use on the potential market for the work [28]. These questions are particularly difficult to address in the context of generative AI systems as copyrighted data is often compiled into massive datasets for model training, and the impact of a particular copyrighted work on generated outputs cannot always be traced clearly.

Analyzing violations. Given the complexity of understanding fair use, analyzing potential copyright violations in generative AI often requires mixing legal and technical understandings. While there is some legal precedent for copyright cases involving search engines, web code, and APIs [85], many questions have yet to be answered in the context of AI systems. There is no clearly delineated amount for what counts as "fair use" of a copyrighted work; for example, the copyright of certain content types such as cookbooks or dictionaries is more likely to be infringed by copying even a small part, whereas this may be acceptable for larger novels [44]. AI also frequently combines both expressive and non-expressive properties [87], and analyzing compliance often requires looking at both low-level content transformations such as n-grams and verbatim copying, and higher level concepts like themes and storylines [44].

Levels of memorization. When searching for technical evidence to explain AI copyright violations, many researchers point to the phenomenon of memorization, where generative AI models reproduce near-exact copies of training data [27]. Memorization can arise due to overfitting or the underlying distribution of data [97], and leads to both direct verbatim reproduction and more subtle forms of copying. Yet despite language models frequently committing plagiarism at the paraphrase or idea-based level [56], most existing research focuses only on verbatim copying, making more subtle forms harder to assess [19]. There is thus a need to expand AI copyright research into identifying and mitigating indirect copying of protected works.

Practical challenges for the AI context. Another factor that often complicates AI copyright issues from both a legal and technical perspective is the opaque and decentralized nature of AI development. Many generative AI models are "black-box", meaning developers and researchers cannot fully understand their internal functions and trace how copyrighted content may be appearing [61, 101]. AI development today also occurs through a highly distributed supply chain [57], and lack of coordination between involved parties reduces the chance for methods to be effective at scale or resilient across later transformations of an AI model [84].

3. TAXONOMY OUTLINE

Facing dual challenges from legal and technical environments, many researchers have called for a co-evolution of technology and law so that developments in each field may support the other [44]. Considering this need for interdisciplinary work, our paper aims to combine a detailed overview of state-of-the-art technical methods for reducing copyright violations in generative AI with an analysis of the regulatory landscape which may support these methods. Specifically, we organize our analysis using a framework that maps copyright concerns across key stages of the generative AI lifecycle, integrating legal and technical work into a coherent structure. As shown in Figure 1, we categorize recent work

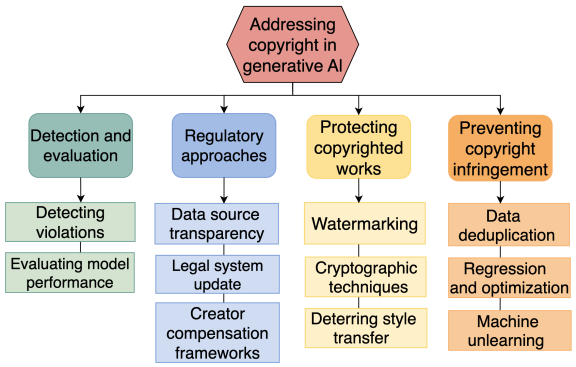


Figure 1: An overview of our proposed taxonomy.

supporting copyright compliance for generative AI into four specific focus areas, tracing copyright issues from foundational challenges to practical solutions while incorporating a diverse range of actors and development stages. We begin with **(1) detecting and assessing copyright violations**, outlining methods that identify where and how infringement occurs so that targeted solutions can be applied. Next, we discuss **(2) regulatory approaches** that can help facilitate more effective copyright protection in AI. In the following two sections, we explore technical mitigation strategies in two distinct areas: Methods for **(3) protecting copyrighted works**, which give individual creators tools to safeguard their from unauthorized use, and **(4) preventing copyright infringement**, covering model-level training strategies aimed at reducing the over likelihood of AI systems generating infringing outputs. After covering these four areas, we present resources like datasets and toolkits to support these goals, and conclude by highlighting emerging challenges in the field.

4. DETECTING AND EVALUATING VIOLATIONS

We begin by surveying methods for detecting and evaluating instances where AI models are likely to infringe on copyright. These methods are critical as they help stakeholders to identify problem sources and apply targeted remedies, assess risk and compliance, and use concrete evidence of infringement to inform legal and policy decisions. We divide the methods here into two main categories: (1) web tools for detecting unauthorized AI-enabled reproduction of copyrighted works and (2) datasets and methods for evaluating general model performance.

4.1 Detecting violations

Methods for flagging AI-enabled copyright violations draw from two primary areas of research: detecting copyright infringement and detecting AI-generated content. There is often a trade-off between the two, as methods for detecting infringement often have limited applicability once content has been altered from its original form, and AI-generated content (AIGC) detection relies on picking up special signatures from AI-generated content which may be less present in a close copy of a human creator's work.

Identifying copyright infringement. The first set of methods focuses on finding places where protected content is replicated without authorization, primarily relying on simple web tools. Reverse image search tools such as Yandex[1], Tineye[2], and Google Reverse Image Search[3], along with text search tools like Scribbr[4] and Grammarly[5] allow users to find instances where content may have been replicated. Recent models have also emerged which leverage AI to search the web for potential violations, combining BERT with DNN models to search for and flag infringing content [45]. However, many research studies on detecting copyright infringement are highly limited in scope, and may fail at detecting violations made by AI which alters the content beyond its original form [56, 19]. Special applications where the expressive and non-expressive elements of a work are closely linked, like with LLM-powered code generation, may also need unique methods for determining if a certain use counts as infringement or not [92, 119].

Identifying AI Generated Content. The next set of methods aims to detect AI generated content and trace it back to its original source to identify potential violations. A wide variety of methods have been developed to detect AI generated content across text [117, 35], image [86], video [43], and multi-modal [47, 124] content. They show that traditional methods such as logistic regression, random forest, SVM, and classifier-based methods display a fair level of accuracy at separating human from AI created content. However, not every AI generated image is one that infringes on copyright, so it is also necessary to have methods for scanning content registries to identify what images it was likely sourced from.

Fingerprinting for Comparison. Fingerprinting has been recommended for many applications to better enable identification and take-down of copyrighted material, providing a unique trace allowing digital content to be identified and attributed to its source [77]. Preetha and Bindu use a wavelet based video fingerprint to extract signatures from different images created from a video, extracting both temporal and spatial features into a compact form which can be stored in a video database and used to determine whether a query video is drawn from that database source [89]. Ning et al. develop a similar system allowing users to register content using its fingerprint rather than the original work [82]. While many different strategies for using digital fingerprints have been developed, they generally share four main characteristics: uniqueness, stability, extractability, and compactness [20]. As a result, fingerprints may provide an efficient way to identify infringing works at scale, but current research applied specifically to the context of generative AI problems is limited.

4.2 Assessing model performance

Jailbreaking methods. Several methods aim to test how easily a model can be made to generate protected content. Text-to-image models can often be prompted to generate copyrighted content even when keywords for a protected IP are replaced with a description of the image [52]. Kim et al. introduce an automatic prompt generation pipeline using LLMs to autogenerate descriptions and create revised prompts designed to induce reproduction of copyrighted content [52]. Their model can evaluate LLM copyright compli-

[1] https://yandex.com/images
[2] https://tineye.com
[3] https://google.com/images
[4] https://scribbr.com/plagiarism-checker/
[5] https://grammarly.com

ance without requiring access to any internal weights, allowing it to function on black-box systems. The prompts generated through their pipeline successfully jailbreak ChatGPT to generate copyrighted content 76% of the time, with an 11% block rate. Test cases can be designed for exploring a model's tendency to engage in specific copying behaviors, like verbatim vs indirect copying.

AI dataset probing. Similar methods aim to reduce the black-box nature of AI models by probing into whether certain copyrighted content is included in their training dataset. For this purpose, Duarte et al. introduce DE-COP, a benchmarking method designed to determine if a piece of copyrighted content is included in a training dataset [33]. Their approach is to probe an LLM with multiple-choice questions to complete a passage of a suspected target book, with options including both verbatim text and paraphrases. The LLMs that were tested showed substantially different performance on these tests depending on whether or not a book is in their training dataset, with the DE-COP method showing a 72% accuracy for detecting suspect books, compared to about 4% for previous models. However, information about whether or not something is in a model's dataset is not enough to prove a violation of copyright. This method should thus be combined with other strategies which examine model outputs to find violations.

Model-level risk quantification. To fulfill the need for a broad metric for evaluating copyright risk, Zhou et al. introduce CopyScope, a framework for quantifying infringement at the model level by using Fréchet Inception Distance (FID) to capture image similarity in the way that most closely mirrors human perception [135]. Using an ensemble based approach of trying different combinations of model sub-components, Zhou et al. use FID-Shapley values to calculate how much each component contributes to the final image's likeness to the training data, identifying which models are most likely to cause infringement issues and should be identified as targets for extra attention.

Regulatory Options for AI Copyright Issues. Copyright challenges in generative AI cannot be solved through advancements in computing alone; disciplines such as law and policy play a central role in shaping discussions around copyright and promoting specific values in design. Legal and policy standards not only work to clarify where violations are present [96], but also frequently serve as a prerequisite for implementing technical safeguards on a wide scale. For instance, several mitigation techniques depend on a well-documented data life cycle, requiring clear transparency about where data is sourced and how it is used [125], which better regulations can help coordinate. These frameworks also work to shape broader industry practices around data collection, creator consent, and compensation, fundamentally affecting what copyright issues emerge. Taking a uniquely interdisciplinary lens, this section explores key regulatory approaches that interact with AI copyright mitigation and enforcement, integrating perspectives from law, economics, and more.

4.3 Data source transparency

Voluntary measures and Industry standards. Transparency in AI training data is a foundational principle for advancing copyright protection in AI [95]. Transparency measures empower copyright holders to monitor and protect against misuse of their works, and promote better principles and processes for designing AI models. Inspired by similar "privacy-by-design" principles, several experts advocate for adopting measures that actively promote transparency throughout the AI development process [37]. Groups like the Coalition for Content Provenance and Authenticity and the Content Authenticity Initiative have frequently contributed to these discussions, developing technical standards for tracking data origins with features that allow rights holders to specify whether training is allowed [95]. However, the adoption of such measures remains largely dependent on the voluntary actions of individual companies, which limits their widespread effectiveness.

Standardized labels. Another approach calls for standardized labels to harmonize transparency efforts. These AI "nutrition labels", inspired by similar efforts in the food agency, call for disclosing information on an AI system's data sources, potential risks, and limitations [24]. While these models have been adapted to support labeling of generative AI [106], more work is necessary to explore how they can be applied to the specific context of copyright. A similar proposed tool is "data cards", or structured summaries that provide essential facts about datasets and explain the rationale behind decisions made in creating them [90]. Professional organizations and regulatory authorities could adopt policies to promote the adoption of these measures or integrate them within other documentation and reporting requirements.

AI system audits. Independent audits of AI systems may be proposed as part of a broader licensing framework or an industry standard certification separate from statutory policy. Auditing for copyright is typically done on model data sets to understand the data collection process and nature of the dataset [74]. However, these "data audits" typically focus on general industry data practices rather than holding dataset creators accountable, and are often divorced from other model-level audits, creating a disconnect between data understanding and deployment regulations [10]. Noting the fragmentation of the current AI audit landscape, Manheim et al. recommend creating AI audit standards boards keep auditing standards up to date with current advancements in AI and clarify which standards are suited for specific domains and applications [74]. This work also has a technical dimension, as decisions need to be made about how much access should be given to auditors. In this scope, Casper et al. propose expanding beyond traditional "black-box" access audits which only observe AI system outputs. To make audits more robust, they suggest including "white-box" access to information about model weights and inner workings, and "outside-the-box" access that provides information about training and deployment processes [14].

4.4 Copyright legal system updates

Considerations for legal change. As generative AI systems continue to increase their capabilities and become more widespread, existing doctrines of copyright law become increasingly unsuited for resolving disputes about AI [2, 53]. This creates a need for broader structural change, which is likely to be implemented through a gradual system of legislative action and interpreted rulings rather than a single policy. Many ideas revolve around strengthening protections for creators [40] or updating the copyright law system

to address specific questions around data scraping and ownership in the digital age [21]. In these discussions, policymakers often need to manage the trade-off between transparency and feasibility of implementation. Beyond facing criticisms of stifling innovation which may turn government support against these measures [16], creating too harsh of standards around training data may significantly limit the amount available, and limited data is more likely to reinforce biases and stereotypes [40].

Opt-out policies. Opt-out provisions aim to reconcile issues of consent and copyright by allowing creators to opt out of having their work used for AI training [127]. However, it is often unclear how they are meant to function in practice [137]. Existing opt-out methods are often difficult and seen by creators as largely a PR stunt [87], causing a need for more research to address the fragmentation of opt-out policies in the status quo [51]. Pasquale and Sun (2024) propose a mandatory opt-out mechanism requiring AI developers to remove works from their databases upon request if infringement has been documented, and take action action to prevent the infringing content from being made again [87]. While this method creates more transparency by requiring developers to properly manage their datasets and confirm upon request if a source has been used [87], it functions more as a post-hoc right to remove copyrighted content after violation has occurred, leaving a remaining need for more preventative opt-out strategies. Many rights holders may also be unaware of unauthorized AI reproductions of their work or lack information about opt-out procedures, creating barriers to establishing a system of full consent.

Content management registries. To help streamline consent management for new AI training, Balan et al. introduce a decentralized registry for content creators to assert their right to opt in or out of AI training, combining distributed ledger technology with visual fingerprinting [7]. Their prototype model prevents a scalable way to trace generative AI training data to determine consent, similar to many of the methods discussed in the protection section. They propose such a registry also be used to track information for compensating creators that opt in to AI training.

Compensation frameworks. Many new proposals have identified that reworking current compensation frameworks may help reconcile the interests of creators and AI producers. Compensation not only ensures individual creators are treated fairly, but also establishes better data provenance tracking and reduces power differentials between AI developers and creators [87] when frameworks are well designed. Compensation systems could be arranged through either new statutory [40] or existing contract-based structures [114], working with collective rights management organizations to streamline negotiating and distributing remuneration. These models generally fall within three main categories, as illustrated in Table 1.

Table 1: Compensation frameworks for AI training data.

Compensation Model	Pays Based On
Pay to Train	Percentage of training data
Pay to Train and Inspire	Contribution to generated outputs
AI Royalties	Negotiated IP partner framework

Pay-to-train compensation. The pay-to-train model involves rewarding IP holders based on the percentage of their contribution to a dataset. Such a model could use increased dataset transparency requirements to calculate payments based on the amount of copyrighted material within AI datasets and the monetary value attributed to its use for AI training. These payments could be afforded to individual creators, or distributed into collective funds to support creators. However, dataset sources are not always well-documented, and payments may be minimal for individual creators beyond famous authors or artists whose work is more likely to comprise a large portion [26].

Pay-to-train-and-inspire compensation. This model works backwards to understand which items in a model's training data likely inspired a particular output, and distribute compensation accordingly [34]. Wang et al. combine probabilistic methods with Shapley value interpretability techniques under a game theory model to establish a base framework for compensation in this way and show its viability through practical experimentation with common data sources [114]. This method works best for AI models trained on limited data with copyright split between a smaller amount of owners. In some cases, it may be more practical to estimate aggregate payoffs for copyright owners across all generated outputs rather than tracing each output back to its source for compensation.

AI royalties. The last model, AI royalties, aims to create collaborative partnerships between IP rights holders and AI companies for compensation based on the market usage and value of their systems [34]. This model could be implemented using existing contract law systems, recognizing the exclusive right of rights holders to commercial use of substantially similar copyright or trademark uses, and granting the AI company the right to use its IP to create outputs under defined use guidelines, sharing in a portion of the overall revenue generated by the system. Ducru et al. argue that this model is the best suited to mutually benefit the interests of AI creators and IP rights holders, and eliminates the need for case-by-case determinations by allowing a broader, predefined agreement that covers all outputs generated by the system [34].

5. PROTECTING COPYRIGHTED WORKS FROM AI

We now introduce technical methods for mitigating copyright infringement in generative AI, starting with methods for protecting individual copyrighted works. These methods are primarily applied at the data collection and preprocessing level, and may be used by creators to protect their works as well as developers seeking to preserve attribution and traceability of training data used. While many protection techniques are frequently used in conjunction, we divide them into three primary categories: (1) watermarking, (2) cryptographic methods, and (3) deterring style transfer.

5.1 Watermarking

Watermarking methods serve to embed an imperceptible identification layer onto images or videos to identify ownership over the content. They can be embedded directly into inputs such as images, or their intermediate representations such as encoder/decoder feature maps [32]. We divide this section into two main approaches based on the methods

used to apply the watermark: attention-based mechanisms and cryptographic methods.

Attention based mechanisms. Attention-based techniques leverage AI models' tendency to focus on specific image regions to create watermarks which remain present after a high degree of manipulation. For example, Zhang et al. introduce deep learning-based video watermarks using a custom attention mechanism designed to recover a 32-bit watermark with 99% accuracy post-transformation [128]. Similar methods can be used for image content by applying watermark algorithms based on Tchebichef moments which describe the spatial distribution of an image's intensity [36, 81, 116]. By partitioning the host image into non-overlapping blocks and calculating Tchebichef moments for each block, Ernawan and Kabir are able to prioritize watermark embedding in areas of lower visual entropy or complexity. As a result, their method demonstrates resilience against noise disruptions and JPEG2000 compression, while preserving a high peak signal-to-noise ratio (PSNR) of approximately 40 dB. While attention mechanisms like these are more resilient to various transformations, testing may be needed to ensure watermarks maintain high enough 'pattern uniformity' to be learned and reproduced by generative diffusion models [30].

Cryptography-based watermarks. In order to further increase traceability and security, several techniques for watermarking incorporate various cryptographic methods. For video content, Zheng et al. incorporate a double-layered watermarking technique combined with blockchain technology [133, 23]. The double watermarking technique embeds both robust and fragile watermarks into the video content, where the robust watermark ensures copyright protection, while the fragile watermark facilitates tamper detection and integrity verification. By integrating double watermarking with decentralization, the combined method achieved a 90% precision rate and a 95% recall rate in detecting tampered parts of watermarked videos against adversarial attacks. Sanivarapu et al. present a similarly robust digital watermarking system using cryptographic techniques to protect images from copyright infringement [99]. The system first embeds a QR code watermark using Discrete Wavelet Transform (DWT) [108]. It then layers on the transformed matrix with Singular Value Decomposition (SVD), and uses the RSA algorithm for watermark embedding, where the coefficients of DWT are modified using secret keys to embed the watermark, increasing the security and encryption of the image. While cryptographic watermarks provide better protection against specific threats like forgery and tampering, they also come with high computational overhead, and may be more perceptible than attention-based watermarks which can be optimized for processing through various generative models.

5.2 Cryptographic methods

Beyond applications in watermarks, advanced cryptographic strategies frequently aid in tracing and ownership verification of copyrighted content [31, 103]. One main type is digital signatures and hashing, which aim to authenticate content by providing a record of ownership and evidence of alterations or tampering that may be done to content, such as removing watermarks. Chain and Kuo (2013) use chaotic map transformations for generating digital signatures, using their unpredicable pattens to produce unique signatures from text for later verification [15]. By contrast, the Elliptic Curve Digital Signature Algorithm (ECDSA) used by Chandrashekhara et al. combines elliptic curve cryptography with digital signatures [17], using a SHA-256 hashing algorithm to generate a public key from a private one to authenticate the signature [5]. Commonly implemented along with hashing, blockchain methods similarly work to safeguard various works from infringement by recording transactions in cryptographically linked blocks, which are harder to tamper with than traditionally protected methods [48, 91]. The decentralized nature of blockchain enhances security by removing central points of control and creating a publicly accessible record of ownership. This can be particularly helpful for tracking data and model copyright status across each stage in the AI development life cycle, such as by integrating blockchain with contract management software for digital content [98].

5.3 Deterring style transfer

For image generation models, style transfer refers to the ability to produce the same content of a target image across a variety of artistic styles [39], which can lead to non-direct reproduction of copyrighted works. While style transfer is often explored as an intentional goal, such as for filling missing frames in an animation [12], unintentional replication of an artist's style can increase copyright risks. Adversarial layers can help protect artistic work from being copied by adding a minimally perceptible layer that distorts the ability of an AI model to recognize it as a normal image and prevents the creation of derivative works [63, 102, 134]. For example, Li et al. further develop this strategy by using a momentum-based ensemble method to enhance the ability for these protections to be generalized across AI models [62]. By altering the intermediate style representation of an image across multiple encoders and combining them through a softmax regression gradient, their method of "Neural Style Protection" can protect style transfer across both known and unknown models, while end-to-end and random noise baseline methods offer minimal protection.

6. PREVENTING COPYRIGHT INFRINGEMENT

Moving on from methods to protect individual works, this section discusses prevention methods that AI developers can use to reduce the overall risk of a model generating reproductions of copyrighted works — a significant concern for developers seeking to avoid reputation loss and legal costs [73]. These methods focus on improving the general behavior of a model, and can be useful to apply at later stages in the AI development life cycle, including fine-tuning existing models. With many current datasets lacking proper content attribution or copyright information, these solutions may serve as a short-term way to enhance copyright compliance without needing to fully construct new large-scale datasets.

6.1 Data de-duplication

De-duplication refers to the process of removing redundant data within a model's training dataset [22], and has commonly been researched for the purposes of reducing data storage space [136] and improving query performance [93]. De-duplication methods may apply fingerprints or hash values to divided data chunks in order to reduce the computational resources needed to check for duplicates in a big

data context [113]. While existing research primarily focuses on the benefits of de-duplication for structured data and predictive models, Lee et al. show that de-duplication in generative AI data can improve overall performance and reduce verbatim copying in generative AI models [58]. Particularly when combined with hashing or blockchain methods, de-duplication is effective at creating cleaner datasets for training generative AI and preventing copyright issues arising from the overuse of any particular material [79, 4]. De-duplication is especially well suited for cleaning "noisy" datasets, such as data scraped from social media [60], as it can simultaneously reduce training time while improving performance and language understanding for LLMs.

6.2 Regression and optimization

In some cases, altering modeling and optimization choices can help prevent a model from reproducing copyrighted data, as Chu et al. "copyright regression" approach demonstrates [25]. By adding an inverse term to the training objective that discourages the model from generating outputs that match copyrighted data and demonstrating mathematically that it can be applied successfully on the softmax function, Chu et al. successfully create a method which helps balance between model performance and copyright protection. However, this method relies on knowledge about which training samples are copyrighted, which is not provided for most existing LLM datasets [94]. Kim et al. highlight the difficulty of applying an end-filter to remove copyrighted content, noting that no open source models are currently able to identify and differentiate copyrighted content, particularly as many datasets lack information about copyright attribution [52]. While target datasets of copyrighted content have been created for the purposes of evaluating whether a model will produce them [52], filtering out all potential violations will require larger and more comprehensive datasets that contain clear information about attribution and copyright status of works.

6.3 Machine unlearning

Unlearning techniques. For cases where an AI model has already been trained on copyrighted data, "machine unlearning" methods can be used to remove a group of samples from its training data, allowing it to act as though it has never seen the data before. This can be especially helpful from a legal perspective, allowing rules such as the GDPR's "right to be forgotten" [18] to be applied to machine learning models. Multiple methods have been developed to selectively remove content while preserving general features a model may have learned from the content. Zhang et al. introduce two different methods for unlearning: Elastic Weight Consolidation (EWC), which adds a constraint to the model's loss function to neutralize the influence of "removed" data, and Decreasing Moment Matching (DMM), which approximates the model's knowledge as a Gaussian distribution and selectively matches moments to similarly reduce reliance on data [131].

Generative AI specific strategies. While earlier research mostly focused on unlearning for classification models, newer studies have explored unlearning for generative AI [68]. Liu et al. conduct a comprehensive survey of machine unlearning for generative models, categorizing them into two main approaches: parameter optimization, which focuses on adjusting model parameters linked with target removal data, and in-context unlearning, which alters input prompts through an API aiming to steer the model away from the "unlearned" content [69].

Knowledge entanglement. One of the largest problems identified in both method types is knowledge entanglement, where data requiring removal is often closely tied to a model's knowledge of certain topics, causing a trade-off between model performance and compliance with the unlearning goal. To solve this issue, Tang et al. introduce a three-component framework to allow models to unlearn certain data without sacrificing their expressive capabilities [110]. The three components include a Knowledge Unlearning Induction module which trains the model to forget specific sequences, a Contrastive Learning Enhancement module to maintain overall performance, and an Iterative Unlearning Refinement module to iteratively update the target data for unlearning, preventing a drastic shift to the model from ocurring. Similar strategies include adopting "un-unlearning" techniques to reintroduce unlearned data in context. This can be important to prevent the unlearned data from being recreated if introduced later as input to the system [105] or retained by association with similar concepts [52]. For example, Van Gogh's *Starry Night* may still be recreated after a model attempts to unlearn "Van Gogh" as it has high correlations with the concepts *star* and *night*. For this reason, unlearning strategies will often retain a copy of the unlearned data to serve as a reference for evaluating model outputs without being used to train the generative model [110].

7. RESOURCES

7.1 Datasets

While no comprehensive dataset of all copyrighted works has been developed [52], benchmark datasets of both AI-generated and human content serve as critical resources for testing and evaluating generative model performance on copyright issues. In this section, we present some of the most commonly used datasets for AI copyright research along with a discussion of their use potential.

Human content datasets. Repositories of human-generated content may serve as a benchmark for evaluating AI models by testing if an AI model completes a passage of protected text [56] or comparing to AI-generated content for improving detection algorithms [124, 43, 86]. Popular datasets for imagery include COCO [65, 66], Flickr30K [123], and OpenImages [55], collectively offering over 9 million images with annotations describing the objects included and overall scene. Other datasets such as the the Metropolitan Museum of Art Open Access collection [6] and WikiArt [115] focus primarily on artistic contributions, which can be helpful for exploring style transfer or searching for matches between AI generations and human-created artwork. Text datasets include OpenSubtitles [67], BookCorpus [8], the WikiText collection from wikipedia articles [78], and JSTOR's library of journal articles [49] may be similarly used to scan for potential memorization, or compare human with AI-generated content.

Combined human-AI datasets. A separate category of datasets combines human-generated with AI-generated works across various art styles and writing subjects [107, 70, 9]. These datasets are primarily valuable for training models to recognize text or image pairs, which allows for

both detection and mitigation of copyright violations [54, 100]. Some datasets in this category are designed for the specific purpose of comparing human copyrighted works to altered digital versions. Aboutalebi et al. introduce the Deepfake Art Challenge dataset, consisting of over 32,000 image pairs that are either forgeries, adversarially contaminated, or not [1]. The selected methods used to modify images include inpainting, style transfer, adversarial data poisoning, and cutmix, representing popular copyright violation types such as modifying painting styles or using partial image data.

Feature / Artifact based datasets. Beyond providing overall examples of human and AI creations, another category of datasets provides annotations over content to highlight specific 'artifacts' in visual media, which can be used for both detecting and preventing violations. These 'perceptual artifacts' capture subtle distortions or irregularities within an outputted image or video, which can be used for detecting the presence of AI in an image or "inpainting" to regenerate areas with potentially unwanted artifacts [32]. Datasets such as PAL4Inpaint [130] and PAL4VST [129] can be used to localize artifacts within images which may show copyright violations, such as a distorted logo within an image. However, more research is necessary to understand how methods designed for deepfake detection and authentication may be applicable to copyright [32].

7.2 Toolkits

Technical resources. While most methods for detecting and mitigating AI copyright violations have yet to be applied extensively outside of a limited resource context, a few tools have been created for public use by AI developers and creators looking to protect their work. Copyright Catcher, developed by Patronus AI researchers, is the first API which aims to test for potential copyright violations in LLMs [88]. Its method closely resembles those discussed in the evaluation methods subsection [52, 33], using a dataset sampled from popular books on Goodreads to test if a model will complete the beginning of a prompt given from protected text. While this is a good start, it may not capture more complex copying behavior such as paraphrasing [56], and may have limited performance for models trained on other types of media. Most recently, researchers at Imperial College London developed a "copyright traps" system where creators can include ficticious entities in their content to detect where LLMs may be using their content, capturing behavior beyond pure memorization [76, 11]. Resources have also been created to allow for better tracing and analysis of which data is typically used to train AI models. The Data Provenance Explorer, which allows practitioners to trace and filter on data provenance for the most popular open source data collections, is one such tool created for this purpose [71]. The interactive UI includes a tool to explore over 1800 popular datasets, viewing information such as their licenses, sources, and creators, along with additional visualizations about website protocols for scraping and AI training. However, this resource only focuses on text datasets, and more work is needed to establish provenance record tracing systems for other types of data.

Governance toolkits. Several frameworks and toolkits have been developed for analyzing the ethical impacts of AI algorithms, such as the AI and data protection risk toolkit developed by the UK Information Commissioner's Office [112] and the Ethics & Algorithms Toolkit developed in collaboration with the Center for Government Excellence at John Hopkins University [3]. However, no large-scale policy toolkits have been developed with the specific goal of increasing copyright compliance in AI. As legal and regulatory discussion around copyright in AI continues to evolve, one place where toolkits may play a helpful role is by enhancing public understanding of AI models in relation to copyright. Working to promote participatory approaches to AI and machine learning, Shen et al. established the Model Card Authoring Toolkit, a toolkit combining technical interfaces and collective decision-making protocols to empower community members to understand and make decisions about AI models in line with their collective values [104]. Applied to the issue of copyright, focus groups including authors, artists, and other rights holders could benefit from these tool kits to help understand and articulate their collective preferences surrounding AI models.

8. CHALLENGES AND FUTURE DIRECTIONS

As researchers continue to work to understand and improve the behavior of generative AI models, several directions offer potential for discovering new information about identifying and mitigating copyright violations in generative AI. On the side of detection and evaluation, a persistent challenge is *identifying indirect and ambiguous forms of copying* like paraphrasing or near-duplicate code generation [19, 34, 119]. More refined benchmarks and detection tools are needed to improve copyright enforcement for these cases. In a similar vein, *building copyright-specific datasets* is necessary to facilitate better research and detection of copyright issues at scale, track content ownership, and help facilitate certain technical mitigation strategies. For technical methods protecting copyrighted works, *strengthening watermark resilience* remains a major challenge, as there is often a trade-off between imperceptibility and security, and common modifications like compression or noise can easily weaken or remove existing markers [132]. On the side of preventing copyright infringement, more research is needed to understand how AI model parameters can reduce copyright violations while *retaining maximum utility* and *propagating to downstream models* which may be fine-tuned for different goals. Furthermore, there is a need to explore how domain-specific methods can be *transferrable across multiple types of content*, as many current strategies are limited in scope, which may hamper coordination and research. Lastly, each of these efforts continually adapt as the legal system around generative AI evolves. Stronger *consensus on the bounds of fair use* can help strengthen copyright enforcement and bridge the interests of IP rights holders and AI developers.

9. CONCLUSION

As generative AI systems trained on copyrighted data continue to proliferate in the absence of clear legal frameworks, a combination of technical and policy measures is necessary to prevent copyright infringement and ensure that generative AI is developed with fair use principles in mind. Creators, developers, policymakers, and other stakeholders should take action to carefully assess their current compliance with copyright standards and establish stronger sys-

tems of oversight, mitigation, and accountability for copyright harms caused by AI. In many cases, this will involve a continued dialogue between rights holders and AI providers to understand their demands and increase transparency surrounding technical design choices. Researchers should also expand efforts to detect violations, protect creative works, and improve the copyright performance of AI models to address the challenges previously mentioned. Through this integrated approach, creators, developers, and policymakers can collaborate to promote an AI ecosystem designed to support fair use and foster human creativity.

Acknowledgements

This work was supported in part by the National Science Foundation (NSF) under Grant No. 2404039.

10. REFERENCES

[1] Hossein Aboutalebi et al. *DeepfakeArt Challenge: A Benchmark Dataset for Generative AI Art Forgery and Data Poisoning Detection*. 2024. arXiv: 2306.01272.

[2] Jacob Alhadeff, Cooper Cuene, and Max Del Real. "Limits of algorithmic fair use". In: *Washington Journal of Law, Technology & Arts* 19.1 (2024), pp. 1–53.

[3] David Anderson et al. *Ethics & Algorithms Toolkit*. Center for Government Excellence at Johns Hopkins University, 2018.

[4] R. Aparna, Roopa G. Kulkarni, and Shilpa Chaudhari. "Secure Deduplication for Images using Blockchain". In: *IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. 2020.

[5] Andrew W. Appel. "Verification of a Cryptographic Primitive: SHA-256". In: *ACM Transactions on Programming Languages and Systems* (2015).

[6] The Metropolitan Museum of Art. *The Met: Open Access*.

[7] Kar Balan et al. "DECORAIT - DECentralized Opt-in/out Registry for AI Training". In: *Proceedings of the 20th ACM SIGGRAPH European Conference on Visual Media Production*. CVMP '23. 2023.

[8] John Bandy and Nicholas Vincent. 2021.

[9] Bhat, Aaditya. *GPT-wiki-intro (Revision 0e458f5)*. 2023.

[10] Abeba Birhane et al. *AI auditing: The Broken Bus on the Road to AI Accountability*. 2024.

[11] Caroline Brogan and Gemma Ralton. *Hidden Data Could Reveal if an AI Model Was Trained on Copyrighted Material*. July 2024.

[12] Qiang Cai et al. "Image neural style transfer: A review". In: *Computers and Electrical Engineering* (May 2023).

[13] Yihan Cao et al. *A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT*. 2023.

[14] Stephen Casper et al. "Black-Box Access is Insufficient for Rigorous AI Audits". In: *The 2024 ACM Conference on Fairness, Accountability, and Transparency*. FAccT '24. June 2024.

[15] Kai Chain and Wen-Chung Kuo. "A new digital signature scheme based on chaotic maps". In: *Nonlinear Dynamics* 74 (Dec. 2013).

[16] Keith Jin Deng Chan, Gleb Papyshev, and Masaru Yarime. "Balancing the Tradeoff between Regulation and Innovation for Artificial Intelligence: An Analysis of Top-down Command and Control and Bottom-up Self-Regulatory Approaches". In: *Technology in Society* (Oct. 2024).

[17] J. Chandrashekhara et al. "A COMPREHENSIVE STUDY ON DIGITAL SIGNATURE". In: *International Journal of Innovative Research in Computer Science & Technology* 9.3 (May 2021).

[18] Cheng-chi Chang. "When AI Remembers Too Much: Reinventing the Right to Be Forgotten for the Generative Age". In: *Washington Journal of Law, Technology & Arts* 19 (2024).

[19] Tong Chen et al. "CopyBench: Measuring Literal and Non-Literal Reproduction of Copyright-Protected Text in Language Model Generation". In: (2024).

[20] Wendi Chen, Wensheng Gan, and Philip S. Yu. *Digital Fingerprinting on Multimedia: A Survey*. 2024. arXiv: 2408.14155.

[21] Simon Chesterman. "Good models borrow, great models steal: intellectual property rights and generative AI". In: *Policy and Society* (2024), puae006.

[22] Nipun Chhabra and Manju Bala. "A Comparative Study of Data Deduplication Strategies". In: *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*. 2018, pp. 68–72.

[23] Sribala Vidyadhari Chinta et al. "AI-driven healthcare: Fairness in AI healthcare: A survey". In: *PLOS Digital Health* 4.5 (2025), e0000864.

[24] Kasia S. Chmielinski et al. *The Dataset Nutrition Label (2nd Gen): Leveraging Context to Mitigate Harms in Artificial Intelligence*. 2022.

[25] Timothy Chu, Zhao Song, and Chiwun Yang. *How to Protect Copyright Data in Optimization of Large Language Models?* 2023.

[26] Amanda Coelho Della Giustina. "Fair Compensation for Copyrighted Data Used in AI Training". MA thesis. Tilburg Institute for Law, Technology, and Society (TILT), 2024.

[27] A. Feder Cooper and James Grimmelmann. *The Files are in the Computer: Copyright, Memorization, and Generative AI*. 2024. arXiv: 2404.12590.

[28] Copyright Advisory Services. *Fair Use*.

[29] Thomas F Cotter. "Fair use and copyright overenforcement". In: *Iowa L. Rev.* 93 (2007), p. 1271.

[30] Yingqian Cui et al. "DiffusionShield: A Watermark for Copyright Protection Against Generative Diffusion Models". In: *NeurIPS 2023 Workshop on Diffusion Models* (2023).

[31] Saad Mohamed Darwish, Mona Mahamod Abu-Deif, and Saleh Mesbah Elkaffas. "Blockchain for video watermarking: An enhanced copyright protection approach for video forensics based on perceptual hash function". In: *PLOS ONE* (Oct. 2024), e0308451.

[32] Jingyi Deng et al. *A Survey of Defenses against AI-generated Visual Media: Detection, Disruption, and Authentication*. 2024. arXiv: 2407.10575.

[33] André V. Duarte et al. *DE-COP: Detecting Copyrighted Content in Language Models Training Data*. 2024.

[34] Pablo Ducru et al. *AI Royalties – an IP Framework to Compensate Artists & IP Holders for AI-Generated Content*. 2024. arXiv: 2406.11857.

[35] Ahmed M. Elkhatat, Khaled Elsaid, and Saeed Almeer. "Evaluating the efficacy of AI content detection tools in differentiating between human and AI-generated text". In: *International Journal for Educational Integrity* (Sept. 2023).

[36] Ferda Ernawan and Muhammad Nomani Kabir. "An Improved Watermarking Technique for Copyright Protection Based on Tchebichef Moments". In: *IEEE Access* 7 (2019), pp. 151985–152003.

[37] Heike Felzmann et al. "Towards Transparency by Design for Artificial Intelligence". In: *Science and Engineering Ethics* (Nov. 2020), pp. 3333–3361.

[38] Giancarlo Frosio. "Generative AI in court". In: *Recreating Creativity, Reinventing Inventiveness*. Routledge, 2024, pp. 3–44.

[39] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. "Image Style Transfer Using Convolutional Neural Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2414–2423.

[40] Christophe Geiger and Vincenzo Iaia. "The forgotten creator: Towards a statutory remuneration right for machine learning of generative AI". In: *Computer Law & Security Review* (Apr. 2024), p. 105925.

[41] Roberto Gozalo-Brizuela and Eduardo C. Garrido-Merchan. *ChatGPT is not all you need. A State of the Art Review of large Generative AI models*. 2023.

[42] Carol Mullins Hayes. "Generative artificial intelligence and copyright: Both sides of the black box". In: *Available at SSRN 4517799* (2023).

[43] Peisong He et al. *Exposing AI-generated Videos: A Benchmark Dataset and a Local-and-Global Temporal Defect Based Detection Method*. 2024. arXiv: 2405.04133.

[44] Peter Henderson et al. "Foundation Models and Fair Use". In: *Journal of Machine Learning Research* 24.400 (2023).

[45] Aldo Hernandez-Suarez et al. "Methodological Approach for Identifying Websites with Infringing Content via Text Transformers and Dense Neural Networks". In: *Future Internet* (Dec. 2023).

[46] Cheng-Yu Hsieh et al. *Distilling Step-by-Step! Outperforming Larger Language Models with Less Training Data and Smaller Model Sizes*. 2023.

[47] Liting Huang et al. *RU-AI: A Large Multimodal Dataset for Machine Generated Content Detection*. 2024. arXiv: 2406.04906.

[48] Tao Jiang et al. "Research on the Application of Blockchain in Copyright Protection". In: *2020 International Conference on Culture-oriented Science & Technology (ICCST)*. 2020, pp. 616–621.

[49] JSTOR. *Text-mining support*.

[50] Joohoon Kang and Youngjoo Yi. "Beyond ChatGPT: Multimodal generative AI for L2 writers". In: *Journal of Second Language Writing* (Dec. 2023), p. 101070.

[51] Paul Keller and Zuzanna Warso. *Defining Best Practices for Opting Out of ML Training*. Tech. rep. Open Future, Sept. 2023.

[52] Minseon Kim et al. *Automatic Jailbreaking of the Text-to-Image Generative AI Systems*. 2024. arXiv: 2405.16567.

[53] John T. Kivus. "Generative AI and Copyright Law: A Misalignment That Could Lead to the Privatization of Copyright Enforcement". In: *North Carolina Journal of Law & Technology* 25.3 (Apr. 2024), pp. 447–494.

[54] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Communications of the ACM* (May 2012).

[55] Alina Kuznetsova et al. "The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale". In: *CoRR* abs/1811.00982 (2018).

[56] Jooyoung Lee et al. "Do Language Models Plagiarize?" In: *WWW '23: Proceedings of the ACM Web Conference*. Apr. 2023.

[57] Katherine Lee, A. Feder Cooper, and James Grimmelmann. *Talkin' 'Bout AI Generation: Copyright and the Generative-AI Supply Chain*. 2024.

[58] Katherine Lee et al. *Deduplicating Training Data Makes Language Models Better*. 2022.

[59] Liunian Harold Li et al. *Symbolic Chain-of-Thought Distillation: Small Models Can Also "Think" Step-by-Step*. 2024.

[60] Xianming Li and Jing Li. *Generative Deduplication For Socia Media Data Selection*. 2024.

[61] Yangzi Li. "Does black-box machine learning shift the US fair use doctrine?" In: *Journal of Intellectual Property Law & Practice* (Sept. 2021), pp. 1175–1185.

[62] Yaxin Li et al. "Neural Style Protection: Counteracting Unauthorized Neural Style Transfer". In: *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024.

[63] Chumeng Liang et al. "Adversarial Example Does Good: Preventing Painting Imitation from Diffusion Models via Adversarial Examples". In: (2023).

[64] Paul Pu Liang, Amir Zadeh, and Louis-Philippe Morency. "Foundations & Trends in Multimodal Machine Learning: Principles, Challenges, and Open Questions". In: *ACM Comput. Surv.* (June 2024).

[65] Tsung-Yi Lin et al. *Common Objects in Context*.

[66] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: (2015). arXiv: 1405.0312.

[67] Pierre Lison and Jörg Tiedemann. May 2021.

[68] Sijia Liu et al. *Rethinking Machine Unlearning for Large Language Models*. 2024. arXiv: 2402.08787.

[69] Zheyuan Liu et al. *Machine Unlearning in Generative AI: A Survey*. 2024. arXiv: 2407.20516.

[70] Vijini Liyanage, Davide Buscaldi, and Adeline Nazarenko. "A Benchmark Corpus for the Detection of Automatically Generated Text in Academic Publications". In: *13th Conference on Language Resources and Evaluation (LREC 2022)*. 2022, pp. 4692–4700.

[71] Shayne Longpre et al. *The Data Provenance Initiative: A Large Scale Audit of Dataset Licensing & Attribution in AI*. 2023.

[72] Ming Y. Lu et al. "A multimodal generative AI copilot for human pathology". In: *Nature* (June 2024), pp. 466–473.

[73] Nicola Lucchi. "ChatGPT: A Case Study on Copyright Challenges for Generative Artificial Intelligence Systems". In: *European Journal of Risk Regulation* (Aug. 2023), pp. 1–23.

[74] David Manheim et al. *The Necessity of AI Audit Standards Boards*. 2024.

[75] Gary Marcus and Reid Southen. *Generative AI Has a Visual Plagiarism Problem - IEEE Spectrum*. Jan. 2024.

[76] Matthieu Meeus et al. *Copyright Traps for Large Language Models*. 2024. arXiv: 2402.09363.

[77] David Megías, Minoru Kuribayashi, and Amna Qureshi. "Survey on Decentralized Fingerprinting Solutions: Copyright Protection through Piracy Tracing". In: *Computers* 9.2 (Apr. 2020), pp. 1–19.

[78] Stephen Merity et al. *WikiText-103 Dataset*. Sept. 2016.

[79] Dutch T. Meyer and William J. Bolosky. "A study of practical deduplication". In: *ACM Transactions on Storage* 7.4 (Jan. 2012), pp. 1–20.

[80] Midjourney. *Midjourney Terms of Service*. Feb. 2023.

[81] R. Mukundan, S.H. Ong, and P.A. Lee. "Image analysis by Tchebichef moments". In: *IEEE Transactions on Image Processing* 10.9 (2001), pp. 1357–1364.

[82] Bowen Ning et al. "Research and Development of Copyright Registration and Monitoring System Based on Digital Watermarking and Fingerprint Technology". In: *2021 International Conference on Culture-oriented Science & Technology (ICCST)*. 2021, pp. 354–358.

[83] U.S. Copyright Office. *Definitions (FAQ) — U.S. Copyright Office*. 2008.

[84] Paul Ohm. "Focusing on Fine-Tuning: Understanding the Four Pathways for Shaping Generative AI". In: *SSRN Electronic Journal* (2024).

[85] David W. Opderbeck. "Copyright in AI Training Data: A Human-Centered Approach". In: *SSRN Electronic Journal* (2024).

[86] Daeeol Park, Hyunsik Na, and Daeseon Choi. "Performance Comparison and Visualization of AI-Generated-Image Detection Methods". In: *IEEE Access* 12 (2024), pp. 62609–62627.

[87] Frank A. Pasquale and Haochen Sun. "Consent and Compensation: Resolving Generative AI's Copyright Crisis". In: *SSRN Electronic Journal* (2024).

[88] Patronus AI. *Introducing CopyrightCatcher, the First Copyright Detection API for LLMs*. Mar. 2024.

[89] S. Preetha and V. R. Bindu. "A Wavelet Optimized Video Copy Detection Using Content Fingerprinting". In: *Journal of Signal Processing Systems* ().

[90] Mahima Pushkarna, Andrew Zaldivar, and Oddur Kjartansson. "Data Cards: Purposeful and Transparent Dataset Documentation for Responsible AI". In: *2022 ACM Conference on Fairness, Accountability, and Transparency*. FAccT '22. June 2022.

[91] Amna Qureshi and David Megías Jiménez. "Blockchain-Based Multimedia Content Protection: Review and Open Challenges". In: *Applied Sciences* (2021).

[92] Homayoon Rafatijo and Dennis Crouch. "When Is Software Code Copyrightable? Is Its Unauthorized Copying Excusable as a Fair Use?" In: *Preview of United States Supreme Court Cases* (2020), p. 14.

[93] B. Rasina Begum and P. Chitra. "SEEDDUP: A Three-Tier SEcurE Data DedUPlication Architecture-Based Storage and Retrieval for Cross-Domains Over Cloud". In: *IETE Journal of Research* (Feb. 2021).

[94] Jie Ren et al. "Copyright Protection in Generative AI: A Technical Perspective". In: *arXiv* (2024).

[95] Daniel Rodriguez Maffioli. "Copyright in Generative AI training: Balancing Fair Use through Standardization and Transparency". In: *SSRN Electronic Journal* (2023).

[96] Eleonora Rosati. "Infringing AI: Liability for AI-generated outputs under international, EU, and UK copyright law". In: (2024).

[97] Brendan Leigh Ross et al. *A Geometric Framework for Understanding Memorization in Generative Models*. 2024. arXiv: 2411.00113.

[98] Yilin Sai et al. *Is Your AI Truly Yours? Leveraging Blockchain for Copyrights, Provenance, and Lineage*. 2024. arXiv: 2404.06077.

[99] Prasanth Vaidya Sanivarapu et al. "Digital Watermarking System for Copyright Protection and Authentication of Images Using Cryptographic Techniques". In: *Applied Sciences* (Aug. 2022), pp. 1–13.

[100] Giordano Scarciotti and Alessandro Astolfi. "Data-driven model reduction by moment matching for linear and nonlinear systems". In: *Automatica* (2017).

[101] Johannes Schneider. "Explainable Generative AI (GenXAI): a survey, conceptualization, and research agenda". In: *Artificial Intelligence Review* ().

[102] Shawn Shan et al. "Glaze: protecting artists from style mimicry by text-to-image models". In: *Proceedings of the 32nd USENIX Conference on Security Symposium*. SEC '23. Anaheim, CA, USA: USENIX Association, 2023. ISBN: 978-1-939133-37-3.

[103] Sujie Shao et al. "WFB: watermarking-based copyright protection framework for federated learning model via blockchain". In: *Scientific Reports* (Aug. 2024).

[104] Hong Shen et al. "The Model Card Authoring Toolkit: Toward Community-centered, Deliberation-driven AI Design". In: *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*. FAccT '22. 2022.

[105] Ilia Shumailov et al. *UnUnlearning: Unlearning is not sufficient for content regulation in advanced generative AI*. 2024.

[106] Meixue Si et al. *A Solution toward Transparent and Practical AI Regulation: Privacy Nutrition Labels for Open-source Generative AI-based Applications*. 2024. arXiv: 2407.15407.

[107] Ravidu Silva and Jordan J. Bird. *AI-ArtBench*. 2023.

[108] Athanassios Skodras. *Discrete Wavelet Transform: an Introduction*. Feb. 2003, pp. 1–26.

[109] Jonathan Stempel. *NY Times sues openai, Microsoft for infringing copyrighted works ...* Dec. 2023.

[110] Haoyu Tang et al. *Learn while Unlearn: An Iterative Unlearning Framework for Generative Language Models*. 2024. arXiv: 2407.20271.

[111] Stuart A. Thompson. "We Asked A.I. to Create the Joker. It Generated a Copyrighted Image." In: *The New York Times* (Jan. 2024).

[112] UK Information Commissioner's Office. *AI and Data Protection Risk Toolkit*. May 2024.

[113] K. Vijayalakshmi and V. Jayalakshmi. "Analysis on data deduplication techniques of storage of big data in cloud". In: *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*. 2021.

[114] Jiachen T. Wang et al. *An Economic Solution to Copyright Challenges of Generative AI*. 2024. arXiv: 2404.13964.

[115] WikiArt. Dec. 2022.

[116] Bin Xiao, Jian-Feng Ma, and Jiang-Tao Cui. "Radial Tchebichef moment invariants for image recognition". In: *Journal of Visual Communication and Image Representation* 2 (Feb. 2012).

[117] Yaqi Xie et al. *MUGC: Machine Generated versus User Generated Content Detection*. 2024.

[118] Canwen Xu et al. *Small Models are Valuable Plug-ins for Large Language Models*. 2023.

[119] Weiwei Xu et al. *A First Look at License Compliance Capability of LLMs in Code Generation*. 2024.

[120] Rui-Jie Yew. "Break It 'Til You Make It: An Exploration of the Ramifications of Copyright Liability Under a Pre-training Paradigm of AI Development". In: *Proceedings of the Symposium on Computer Science and Law*. 2024, pp. 64–72.

[121] Zhipeng Yin et al. "AMCR: A Framework for Assessing and Mitigating Copyright Risks in Generative Models". In: *ECAI 2025*.

[122] Zhipeng Yin et al. "Uncertain Boundaries: A Tutorial on Copyright Challenges and Cross-Disciplinary Solutions for Generative AI". In: *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 2025.

[123] Peter Young et al. "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions". In: *Transactions of the Association for Computational Linguistics* (2014), pp. 67–78.

[124] Xiaomin Yu et al. *Fake Artificial Intelligence Generated Contents (FAIGC): A Survey of Theories, Detection Methods, and Opportunities*. 2024.

[125] Dawen Zhang et al. "Privacy and Copyright Protection in Generative AI: A Lifecycle Perspective". In: *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI*. 2024.

[126] Haoqi Zhang. "Research on Legal Risks and Responses of Generative Artificial Intelligence". In: *Mod. L. Rsch.* 5 (2024), p. 57.

[127] Hongjiao Zhang and Yahong Li. "Opt-Out Implied Licenses in Copyright Law: From Search Engines to GPAI Models". In: *GRUR International* (July 2024).

[128] Kevin A. Zhang et al. "Robust Invisible Video Watermarking with Attention." In: (2019).

[129] Lingzhi Zhang et al. *Perceptual Artifacts Localization for Image Synthesis Tasks*. 2023. arXiv: 2310.05590.

[130] Lingzhi Zhang et al. *Perceptual Artifacts Localization for Inpainting*. 2022. arXiv: 2208.03357.

[131] Yongjing Zhang et al. "Machine Unlearning by Reversing the Continual Learning". In: *Applied Sciences* 13 (Aug. 2023), pp. 1–10.

[132] Xuandong Zhao et al. *Invisible Image Watermarks Are Provably Removable Using Generative AI*. 2024. arXiv: 2306.01953.

[133] Jingjing Zheng et al. "A Novel Video Copyright Protection Scheme Based on Blockchain and Double Watermarking". In: (Dec. 2021). Ed. by Honghao Gao.

[134] Haonan Zhong et al. "Copyright Protection and Accountability of Generative AI: Attack, Watermarking and Attribution". In: *Companion Proceedings of the ACM Web Conference 2023*. WWW '23. Apr. 2023.

[135] Junlei Zhou et al. *CopyScope: Model-level Copyright Infringement Quantification in the Diffusion Workflow*. 2023.

[136] Lixi Zhou et al. "Serving deep learning models with deduplication from relational databases". In: *Proceedings of the VLDB Endowment* (June 2022), pp. 2230–2243.

[137] Gina Maria Ziaja. "The text and data mining opt-out in Article 4(3) CDSMD: Adequate veto right for rightholders or a suffocating blanket for European artificial intelligence innovations?" In: *Journal of Intellectual Property Law & Practice* (Feb. 2024), pp. 453–459.

Mapping Deep Learning to Blockchain Security: A Survey

Vanessa Ortiz[†,*], Alexandria Bates[‡,*], Gaby G. Dagher[♮], Jun Zhuang[♮], and Tim Andersen[♮]
[†]Montclair State University, [‡]The Pennsylvania State University
[♮]Boise State University
[†]vegavegav1@montclair.edu,[‡]atb5640@psu.edu,
[♮]{gabydagher, junzhuang, tandersen}@boisestate.edu

ABSTRACT

This survey explores how specific artificial neural networks (such as GNNs, CNNs, RNNs, LSTMs, GANs, Transformers, MLPs, and hybrid models) have been applied to secure blockchain systems. Blockchain technology continues to expand across various fields due to its decentralized, tamper-resistant qualities. However, several concerns have been raised since they are more vulnerable to real-time threats and attacks. Deep learning, a branch of artificial intelligence (AI), presents itself as a promising solution for enhancing blockchain security due to its ability to learn from complex datasets. We categorize current research by neural network type, the application domain of the model, applied blockchain layers, security/privacy, and attack/defense, and identify existing patterns, challenges, and research gaps. Several conclusions include data scarcity issues, latency, and limited deployment across all blockchain layers (such as the consensus layer and hardware layer). After studying and comparing several existing surveys, we provide a unique contribution by explaining deep learning techniques specifically tailored for blockchain security, which enables us to highlight numerous opportunities for future researchers to address current limitations, such as scalability and privacy.

1. INTRODUCTION

Blockchain technology is currently quite prominent in both academic and industrial fields, such as healthcare [53; 40; 67], financial services [51; 69], smart contracts [13; 41; 32], and many other areas. These areas exploit its unique strengths, such as data immutability and its tamper-resistant nature [4; 15; 9]. Despite the wide range of strengths offered by blockchain technology, these systems are still susceptible to security threats and anomaly attacks [12; 32]. As blockchain technology continues to scale up in size and become more complex, security systems struggle to meet real-time demands [8; 25]. Real-time threat detection is necessary for these systems to accurately handle threats and anomalies that attempt to compromise sensitive data [25; 38], which blockchain technology is struggling to match.

Deep learning, a subset of machine learning, is an emerging technology that could potentially address the previously mentioned issues with blockchain systems [33; 26]. Its ability to learn from large-scale data and adapt to distinctive environments could properly address issues found with blockchain security, allowing new paths for researchers hoping to improve blockchain security and privacy. Various neural network architectures have demonstrated capabilities in strengthening blockchain systems, such as Graph Neural Networks (GNNs) [25; 14; 32; 30; 12; 42; 57; 33; 54; 11; 56], Convolutional Neural Networks (CNNs) [26; 69; 7; 31; 5], Long Short-Term Memory Networks (LSTMs) [4; 39; 13; 50; 41; 37; 9], and transformers [40; 8; 1; 16; 36]. Models that utilize neural networks can provide contributions towards security (e.g., anomaly detection and threat mitigation) and privacy (e.g., decentralized data protection) [39; 57]. The functionality of deep learning for security and privacy heavily depends on model type, data availability, and the applied blockchain layer(s). Therefore, we clearly classify and analyze each paper that discusses deep learning for blockchain security (DL4BCS) to properly demonstrate which characteristics existing research has. This survey will also state the neural network(s) utilized in an article, along with what blockchain layer(s) the system employs. We also go in-depth on whether a study features a system with build specifications meant to defend from threats, or to attack threats. Finally, we clearly state an application domain for each paper, showcasing how each researcher would apply their system in real-world conditions.

To explore the intersection of deep learning for blockchain security (DL4BCS), this survey paper explores current research that focuses on the potential this field could provide. Our goal is to address the following research questions:

RQ1. How has deep learning been used to address existing blockchain-related security concerns?

RQ2. How do different types of neural networks contribute to blockchain security?

RQ3. What are the main similarities and gaps found in Deep Learning for Blockchain Security between different neural networks?

RQ4. What potential research directions could we provide for DL4BCS-related studies?

Based on these questions, there is a clear focus on how deep learning can impact blockchain security, which provides a specific scope for our research. This survey will provide an in-depth analysis and categorization of current research that discusses this specific topic. Any papers that discuss alternative, yet similar topics, such as large language models (LLMs) for blockchain, or machine learning for blockchain,

[*]First two authors contributed equally to this work.

Table 1: **Overview of Existing Related Surveys.** Based on our stated search criteria in Section 3, we found and compared related surveys about blockchain technology and machine learning from various perspectives, such as whether the survey is machine learning for blockchain or vice versa (with a specific focus on LLMs or deep learning). We also categorize whether a paper discusses security, architecture-based analysis, and future research. This table provides a visualization of whether (i) papers focus on using artificial intelligence for blockchain or the other way around, (ii) a survey discusses LLMs or deep learning, (iii) a paper addresses a model that applies to security, (iv) a paper provides definitions and specifics for various neural networks, and (v) a paper provides researchers several potential research directions for DL4BCS based on existing research. We denote ●, ◐, and ○ as a full, partial, and no discussion of the corresponding items.

Source	**ML for BC**		**BC for ML**		**Security**	**Architecture-based** *Analysis*	**Future Research Directions**
	LLM	*DL**	*LLM*	*DL**			
Ressi et al. 2024 [48]	◐	●	○	◐	●	○	●
He et al. 2024 [27]	●	○	○	○	●	○	●
Zhang et al. 2020 [73]	○	●	○	◐	◐	○	◐
Ural and Yoshigoe 2023 [63]	○	○	○	●	◐	○	●
Shafay et al. 2022 [55]	○	◐	○	●	◐	○	●
Geren et al. 2025 [21]	○	○	●	○	●	○	●
Ortiz et al. 2025 (This Survey)	○	●	○	○	●	●	●

* DL refers to deep learning techniques with the exception of LLMs.

or even blockchain for deep learning, are notably not included in our scope, as we aim to discuss purely how deep learning could potentially improve blockchain security. Although we don't provide an in-depth analysis of articles within these alternative research fields, we will later discuss several surveys that focus on these alternatives.

In order to properly distinguish our review from other similar existing surveys that discuss artificial intelligence (AI) and blockchain, we provide an in-depth comparison of several reviews in this field. Ressi et al. [48] provide a general review on how integrating AI and blockchain can maximize blockchain technology to its full potential, opening doors to many real-world applications. He et al. [27] focus specifically on how LLMs can improve blockchain security. Although similar to DL4BCS, LLMs are a specific deep learning model trained to generate and understand human text. Therefore, this survey focuses on how LLMs can improve blockchain security purely from pre-trained data. Zhang et al. [73] inspect blockchain and deep learning simultaneously, discussing how security can apply to certain systems without including it within its general scope. Ural and Yoshigoe [63] wrote their survey on how blockchain can enhance machine learning, which is notably the complete opposite of He et al. [27]. In a similar manner, Geren et al. [21] discuss how blockchain can be used for LLM in a security and safety aspect unlike He et al. [27] (with the additional range of discussing safety). Shafay et al. [55] discuss how blockchain can be integrated with deep learning in hopes of improving deep learning overall. Notably, this survey serves as the opposite of DL4BCS with an included area of focus for security.

To summarize, existing surveys focus on how AI can be used for blockchain (or vice versa). However, there are existing gaps on how deep learning could be used to improve upon blockchain security. To further close this gap, we provide a summary of distinctions between this survey and other existing reviews in Table 1. From this table, we can see that our survey paper features a unique contribution showcasing an architecture-based analysis for various neural networks. We provide our review **contributions** and highlight the specific impacts to answer our research questions as follows:

1. We first properly define blockchain and deep learning in Section 2 by examining technological history and presenting diagrams to explain deep learning and blockchain integration. We then provide a comprehensive literature review within Section 4. We provide detailed definitions and collected resources from each paper to provide a proper in-depth understanding of current research. This section is split into several subsections, each focusing on a specific neural network. We deliver insights within these research projects, providing in-depth explanations on how each study focuses on DL4BCS. This review provides explanations on how researchers choose to integrate deep learning and blockchain, and how researchers could continue to improve this area of study in the future [**RQ1**] [**RQ2**] [**RQ3**] [**RQ4**].
2. We provide several tables for each paper in our methodology, categorized by different deep learning architectures. We specify whether each article discusses security/privacy and attack/defense. We also clearly state an application domain alongside a clear distinction of which blockchain layer(s) are used for the proposed model. For instance, Table 3 focuses on Graph Neural Network Papers, Table 4 discusses Convolutional Neural Network Papers, and so on up to Table 9, which discusses Hybrid Papers. With these tables, we clearly show how deep learning currently addresses blockchain security concerns [**RQ1**] [**RQ3**]. We also present clear similarities and differences between each neural network in current DL4BCS research, which ultimately provides general research directions for each neural network [**RQ2**] [**RQ4**].
3. We highlight specific papers within DL4BCS research by presenting diagrams that accurately reflect models discussed by researchers. For GNNs, we created Figure 3

to display and highlight Cai et al. [11]. For Generative Adversarial Networks (GANs), we created Figure 4 to display the system created in Rabieinejad et al. [45]. Finally, for Hybrid Networks, we created Figure 5 to exhibit the system provided in Saveetha et al. [51]. We chose to present diagrams for these papers due to their strong explanations on how specific neural networks have directly affected blockchain security in some way [**RQ1**] [**RQ2**]. These diagrams also provide a visual demonstration of how certain systems and models can be similar and/or different, even when using different neural networks [**RQ3**]. These similar and different traits can be further elaborated into potential research directions, which will be discussed within this review [**RQ4**].

The remaining sections of this paper are organized in this manner: In Section 2, we provide a clear background of blockchain technology and deep learning. We also define and explain the neural networks included in our methodology, and provide a history as to how these neural networks came to be. We propose a visualization of the layers of blockchain in Figure 1, and provide a taxonomy in Figure 2 that shows how various layers and neural networks can integrate for various application domains. In Section 3, we explain our methodology, clearly stating what criteria we followed and stating how we searched for articles that discuss DL4BCS. In Section 4, we provide an analysis for each article included in our methodology. To go in-depth on certain models within existing research, we also provide diagrams to highlight several key uses for deep learning and blockchain integration. To summarize every paper included in our methodology, we provide tables for each neural network to clearly show similarities and differences between each paper and neural network. In Section 5, we address challenges that currently impact researchers when integrating deep learning and blockchain together, which overall hinders research advancements. In Section 6, we discuss potential research directions within the field of DL4BCS. Finally, in Section 7, we sum up our review of the literature on current research on DL4BCS.

2. BACKGROUND

The application of deep learning to blockchain technology has created new opportunities for cybersecurity, automation, and decentralized intelligence. In order to understand how these technologies come together, it is important to study their foundations. This section introduces blockchain technology, going in-depth on its origin and how it operates. We will also explore the fundamentals of deep learning, including how neural networks function and how they learn from data. We then discuss key neural networks commonly used with blockchain technology, focusing on their design, capabilities, and original contributions. Together, these foundations give context for the applications and challenges discussed in this paper.

2.1 Blockchain

Blockchain is a type of digital ledger technology that allows secure and transparent transactions across distributed networks, making it a unique technology. It first appeared back in 2008 when Satoshi Nakamoto introduced the concept in "Bitcoin: A Peer to Peer Electronic Cash System." [43]. Nakamoto describes a decentralized mechanism that uses Proof-of-Work (PoW) to validate blocks of transactions. PoW is made up of complex cryptographic puzzles that are solved by network participants, who are typically referred to as miners. This method ensures that each block has a time stamp and is tamper resistant. By introducing the element of tamper-resistance, this prevents users from double-spending and removes the need for a third party.

Blockchain consists of different layers, with each layer contributing specific part in the systems functionality and security, as shown in Figure 1. The very base layer is the data layer, which is responsible for data organization and management, including nodes and underlying data storage technology [61; 60]. Above the data layer is the network layer, which handles peer-to-peer protocols and block distribution across the network [60]. Next is the consensus layer, which implements algorithms such as PoW or Proof-of-Stake (PoS). This layer is extremely important, as it ensures that all participants in the network agree on one single source of truth [60]. On top of the consensus layer, we find the hardware layer. This is where transaction logic is executed and where the general state of blockchain is maintained [29]. Next, the contract layer allows programmable logic in the form of smart contracts and enabling decentralized applications, which is often found in platforms like Ethereum [60]. It is important to note that we didn't include the contract layer within our blockchain layer classification for each paper throughout this survey due to this layer not having particularly unique traits in comparison to other layers. The top layer of blockchain is called the application layer, which includes end-user interfaces and tools such as wallets and trading platforms. These interfaces are able to directly interact with smart contracts and the underlying network [29; 60].

In 2014, blockchain technology was expanded by Vitalik Buterin in "Ethereum: A Next Generation Smart Contract and Decentralized Application Platform" [10]. Ethereum

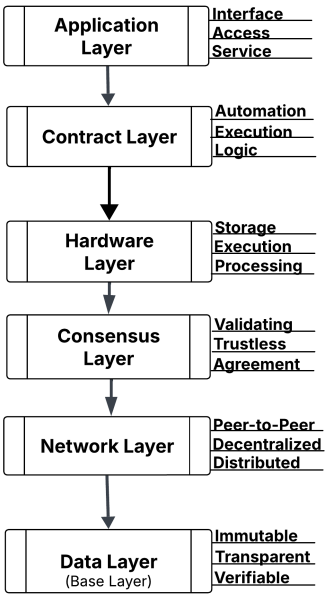


Figure 1: Blockchain layers and their corresponding key functionality.

builds on several elements of Bitcoin's blockchain architecture while introducing the concept of *smart contracts*. Smart contracts are programmable scripts that execute automatically under predefined conditions within the Ethereum Virtual Machine (EVM). This allows blockchain to support rich decentralized applications and transform them into programmable ecosystems. As a result, they're able to execute financial contracts, apply to supply chain coordination, and improve autonomous digital services.

2.2 Deep Learning

Deep learning is a powerful part of artificial intelligence. It is centered on layered networks made up of simple computation units, which are defined as neurons. Each neuron multiplies inputs by learned weights, adds a bias, and applies a nonlinear activation. After the activation is completed, it passes the result forward. An example of a neural network is the Multilayer Perceptron (MLP). MLP is a model where neurons come together and are arranged in fully connected layers. An MLP with a single hidden layer can approximate any continuous function, which has been demonstrated to show theoretical potential by Cybenko [17].

However, as research increased on neural networks, it became clear that there was a need to analyze images efficiently. As a result, Convultional Neural Networks (CNNs) were introduced by LeCun et al. [34]. The CNN architecture uses convolutional filters and layers to learn spatial feature hierarchies in an automatic form. This achieves high-level performance when it comes to recognizing handwriting and digits. Having this neural network facilitates the recognition of classic zip code datasets with minimal need for manual preprocessing.

Some problems were discovered with respect to time dependent data. To help mitigate this issue, Recurrent Neural Networks (RNNs) were developed in 1990 by Elman [20]. RNNs allowed information retention through hidden-state representation, which resulted in trouble modeling long-term dependencies. To mitigate this situation and create a potential solution, Hochreiter and Schmidhuber came up with a new neural network called Long Short-Term Memory (LSTM) in 1997 [28]. LSTMs use input, output, and logic gates to control how data flows through memory cells. This allows for effective modeling of time series data as well as long dependencies.

In blockchain transaction networks, graph-structured data is very common. However, this type of data requires a very specialized model. Gori, Monfardini, and Scarselli introduced Graph Neural Networks (GNNs) to serve as a solution [24]. In GNNs, neurons spread information along edges of the graph and update node states iteratively, allowing relational learning. In 2014, Generative Adversarial Network (GANs) were introduced by Goodfellow et al. [23]. GANs are made up of a generator (which creates synthetic data) and a discriminator (which learns to differentiate real from fake). This process provides realistic data and supports certain applications, such as transaction simulations. In the beginning, neural networks relied on sequential processing. This changed when transformers were introduced in 2017 by Vaswani et al. [64]. Transformers have the ability to analyze data by evaluating dependencies between all elements in a sequence simultaneously. This allows for much faster computation as well as high-level performance when it comes to language and log analysis.

These neural networks serve as essential tools in blockchain analytics and cybersecurity. The potential of these neural networks is extensive in terms of improving blockchain technology. GNNs are able to detect illegal transaction patterns by learning relational flows. CNNs can process visualized data and flag anomalies. RNNs, LSTMs, and transformers are very effective for modeling behaviors over time, auditing logs, and auditing contract code. Finally, GANs are able to generate synthetic blockchain data to improve detection systems. In summary, deep learning models have the potential to make blockchain's immutable system much stronger for fraud detection, automated audits, and smart contract-based attack detection systems. We equip researchers with a visual taxonomy of how blockchain layers and specific domains currently apply to existing DL4BCS based on various deep learning architectures in Figure 2. It is also quite common to see multiple deep learning architectures within one singular system, which are referred to as hybrid models. We provide a complete comparison table on the various neural networks reviewed within this survey in Table 2.

3. RESEARCH METHODOLOGY

Our ability to properly discuss how deep learning can be used for blockchain technology requires a proper exploration of the existing literature related to the subject. Before gathering articles, we first had to come up with a feasible method to ensure that the paper is related to deep learning for blockchain rather than blockchain for deep learning. We also had to create an organizational method to differentiate the scope of these papers. In order to correctly identify scopes and keep current research organized, we used a variety of tables, charts, keywords, and other techniques.

3.1 Literature Search and Strategy

Our primary search tool was Google Scholar, supplemented with searches on IEEE Xplore. We used Boolean logic (such as 'AND', 'OR') within our searches to ensure certain keywords were included. We also tested combinations of relevant model names and concepts, including 'CNN blockchain', 'GNN blockchain', 'LSTM security', and so on. Ultimately, we searched for specific neural networks while including other relevant keywords, such as 'blockchain', 'security', or 'ledger'. We also chose to use a variety of phrases like 'neural networks for blockchain', 'deep learning for blockchain', 'deep learning for blockchain privacy', and 'deep learning in relation to blockchain security'. We then verified that each paper described deep learning for blockchain and not vice versa by reading the abstract, introduction, and background sections of each paper.

3.2 Paper Organization and Categorization

To manage and analyze the articles, we created a system of folders named after each neural network model that were widely used to protect blockchains. As a result, we had eight folders for eight different deep learning architectures: 'CNN'. 'GAN', 'GNN', 'LSTM', 'MLP', 'RNN', 'Transformers', and 'Hybrid'. This allowed us to group the papers by architecture to compare their blockchain-related applications.

We also used tables in Google Sheets to catalog papers, track citation information, identify common methodologies,

	GNN	CNN	GAN	LSTM	MLP	RNN	Transformer
Data Type	Graph	Image/Grid	Various (Synthetic Generation)	Time Series	Structured	Time Series/ Text	Text Sequence
Strength	Understanding Connections between Nodes	Finding Patterns (Shapes)	Making realistic new examples	Remembering Long-Term Patterns	General Tasks	Learning Recent Steps	Handles Full context fast
Limitation	Slow with big network	Not good with time or sequence data	Hard to train/ Unstable	Slow and Complex	Not great for images or sequences	Forgets older steps (Short Term Memory)	Needs substantial data and computing power

Table 2: Neural Network Comparison Table: This table displays the core characteristics of common neural networks, highlighting input data types, strengths, and limitations.

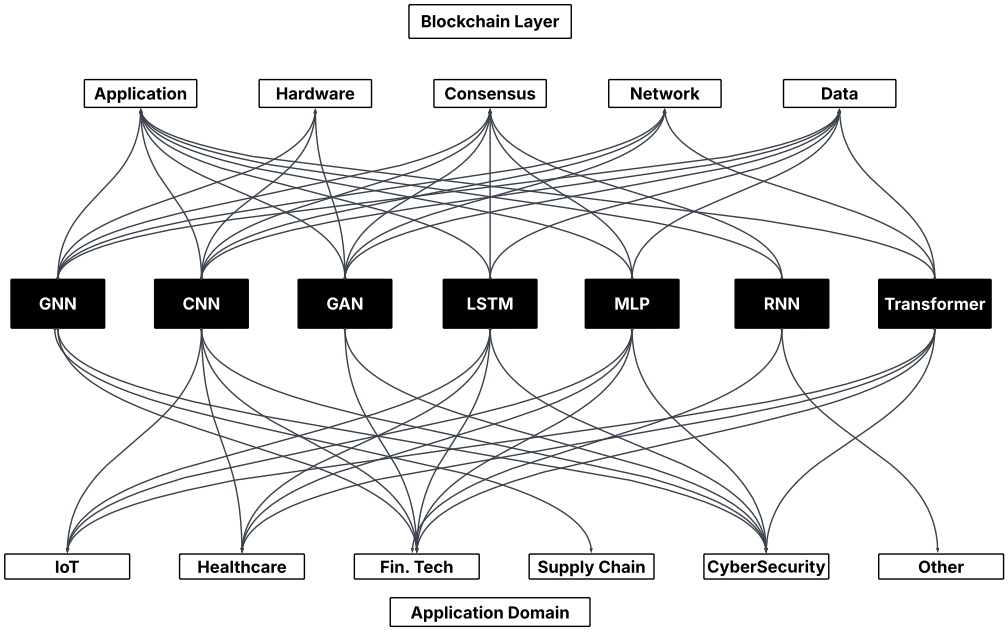


Figure 2: Blockchain and Deep Learning Taxonomy: This taxonomy illustrates the relationship between neural network models and their applications across multiple domain applications (including healthcare, IoT, supply chain, and cybersecurity). It highlights the versatility of individual neural network architectures, demonstrating their capacity to support multiple application areas. Additionally, the diagram reveals how these models interact with different layers of the blockchain, offering insight into their functional integration. The taxonomy also exposes current gaps and deficits within both application domains and blockchain layers, suggesting areas for further research and development.

and note key distinctions. Each entry in the table included model type, blockchain domain, privacy versus security, evaluation strategies, and publication metadata. To differentiate between security-focused and privacy-focused papers, we manually scanned abstracts and full texts for relevant keywords. For example, we classified a paper as security-based if it included keywords and phrases such as 'intrusion detection', 'fraud', 'DDoS attacks', and 'vulnerability'. For privacy, however, we searched for privacy-based indicators such as 'de-anonymization', 'metadata leakage', and 'differential privacy'. This tagging process allowed us to categorize papers and facilitate downstream gap analysis. Our completed table with every research paper we have gathered was a useful tool for finding the most common trends (such as models frequently used for specific blockchain tasks, and highlighting areas with a lack of coverage). It is important to note that although our exact tables from Google Sheets is not presented in this survey, we have replicated our tables for each specific neural network throughout various neural network sections in Section 4.

3.3 Inclusion and Exclusion Criteria

To ensure the quality and relevance of the reviewed material, we applied both inclusion and exclusion criteria. We included articles that applied deep learning models to meaningful blockchain-related challenges such as anomaly detection, smart contract analysis, and data privacy. We excluded papers that approached the topic from the opposite direction, meaning using blockchain to improve deep learning, and papers that lacked technical depth or clear experimental validation. We also omitted papers written in languages other than English, published before 2020, found in peer-to-peer review repositories, PhD submissions, or thesis-only documents. This filtering helped build a clear, high-quality set of papers for meaningful analysis.

3.4 Research Gaps and Limitations

In order to properly organize our findings, we identified clear trends and gaps across the literature we've gathered. GNNs, Transformers, and LSTM models appeared most frequently in blockchain applications. In contrast, RNNs were notably underrepresented, which shows an opportunity for further exploration. While several papers demonstrated promising simulation results, there was a distinct lack of studies on real-world use as well as a scarcity of standardized evaluation benchmarks, which limits comparability across various approaches. We also observed that certain blockchain domains, such as consensus mechanisms, permissioned networks, and decentralized identities, have received limited attention. This suggests several areas where deep learning could be further investigated.

However, our review has limitations. Papers were restricted to those published in English and after 2020, which may have excluded relevant earlier papers or non-English contributions. We also excluded PhD thesis documents and non-peer-reviewed materials, which may have eliminated some emerging or experimental approaches. All combined findings in our methodology highlight the progress and blind spots in current research, serving as a foundation to identify promising directions for future work.

4. EXISTING LITERATURE REVIEW

This section reviews current literature on deep learning for blockchain security (DL4BCS). The studies are categorized in several ways, such as the neural network architecture, similarities and differences, the application domain(s) discussed, datasets used, and any other technical focuses. We also discuss any revealing patterns along with gaps in the methodology, both for each individual neural network and as a whole. Notably, a small number of papers throughout this analysis also apply to privacy. Since privacy is the minority within existing research, we go in-depth on explaining key distinctions on privacy-based research compared to security-based research.

4.1 Graph Neural Networks (GNNs)

Recently, Graph Neural Networks (GNNs) have become one of the most promising architectures for analyzing blockchain systems due to their ability to model complex data structures in blockchain systems. This can include address graphs, protocol interactions, and transaction flows.

Graph neural networks have also evolved into specialized architectures to handle more complex graphs. Primary examples of this evolution include heterogeneous GNNs and temporal GNNs. Heterogeneous GNNs are designed for graphs that contain multiple types of nodes or edges [68]. This type of GNN is particularly useful for applications such as social networks, citation graphs, and fraud detection. In a blockchain-based identity verification system, the interaction between users, devices, and smart contracts forms a heterogeneous graph structure.

On the other hand, temporal GNNs are suited for dynamic graphs where the relationship between entities evolves over time [49]. These models incorporate time-stamped edge events and are designed to capture both structural and temporal dependencies. Temporal GNNs are especially effective in applications like real-time fraud detection, blockchain transaction forecasting, and cyberattack monitoring. For these applications, the sequence and timing of events matter as much as the connections themselves. For instance, in a blockchain ledger, transaction history over time can be modeled as a temporal graph to detect anomalous patterns.

Within current DL4BCS research, many researchers take advantage of GNNs to detect anomalies in transaction networks [30]. For instance, [25] suggests a spatial-temporal GNN model that uses time series behavior of addresses to identify suspicious cryptocurrency transactions. Additionally, [14] discusses a multi-distance message passing approach to capture the temporal and relational distance between transactions in a more accurate form. By implementing these temporal dynamics methods into the graph structure, the accuracy of fraud detection in blockchain improves. To continue, [56] created a dual-level GNN framework. This dual-level network analyzes account behavior and transaction graph features to detect anomalies. This reflects the importance of hierarchical representations when it comes to modeling a blockchain ecosystem. A paper that dives deep into this concept is [11], which integrates GNN and blockchain, showing its potential in analysis and smart ledger architecture. This paper is particularly groundbreaking because it discusses an end-to-end integration of GNNs into blockchain.

While other papers limit GNNs to auxiliary analytical tasks such as fraud detection, GTxChain [11] discusses an enhanced block structure and a new consensus protocol that uses graph representations of transactions directly in block validation and the decision-making process. This model builds a diverse transaction graph where nodes represent accounts and smart contracts. Edges also encrypt transaction types, temporal order, and value flow. This system uses a multi-relational GNN with hierarchical attention mechanisms, allowing it to learn contextual embeddings for nodes that reflect both local behaviors and global structural dependencies. This embedding procedure is used in consensus voting, allowing the system to prioritize blocks that have higher behavioral legitimacy scores. As a result, this reduces the acceptance of malicious transactions. GTxChain also discusses a graph-based Proof-of-Learning (PoL) mechanism. This mechanism replaces regular Proof-of-Work (PoW) or stake-based validation by using a trusted weighted GNN classification task. Through the experiment in [11], the results show that GTxChain outperforms existing consensus protocols both in security and efficiency. Not only does it stay low-latency, but it also reduces the spreading of anomalous transactions. By implanting intelligence into the blockchain, this adaptive, self-aware ledger system constantly learns from transactional behavior. Due to the high significance this paper holds in DL4BCS research, we provide Figure 3, which provides a visual explanation of how GTxChain works.

Current research is often focused on security applications. For instance, Chang et al. [12] and Jeyakumar et al. [30] mention using message-passing networks to identify nodes or transactions with abnormal patterns. Because of this, it is possible to secure blockchain from threats like sybil attacks or illicit fund flows. While most papers focused on security, there were a select few that focused on privacy. For example, Shen et al. [57] expose privacy risks through the process of applying GNN-based deanonymization techniques. These techniques show that address clustering can

reveal user identities, which is a concern for platforms like Bitcoin.

Next, MuhsnHasan et al. [42] use graph-based modeling and apply it to logistics and origin verification, demonstrating its real-world use outside of finance. Additionally, [54] show a new transaction fingerprinting system, which provides insights into behavioral modeling of blockchain participants. One consistent pattern found is the usage of directed or temporal graphs that mirror transaction flows and account behavior. Models seem to vary in their graph construction and temporal encoding. Some papers focused on utilizing blockchain statistic snapshots, while others use temporal features with edge time stamps or sequence-based modeling.

In summary, GNN has shown high efficacy for blockchain transaction modeling, fraud detection, identity inference, and protocol optimization. The adaptability GNN offers to graph-structured data makes it well-suited for blockchain analysis. Future research should prioritize explainability, real-time deployment, and integration into production-level blockchain systems. To provide a visual culmination of our analyzed research, we provide Table 3, which showcases how each paper distinctively falls under various categories we've previously discussed.

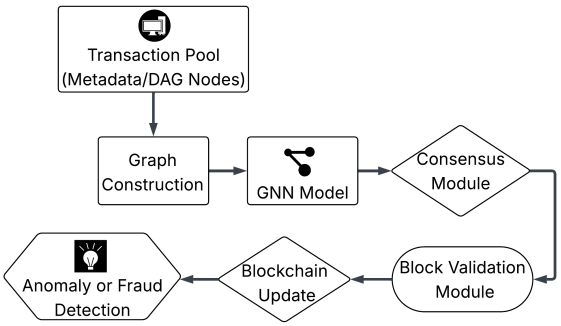


Figure 3: GTxChain [11] is a blockchain system powered by Graph Neural Networks (GNNs). The central focus of the system is the use of GNN, which means the method can learn patterns from both on-chain and off-chain transaction data. Off-chain data includes user behavior or external triggers, which is processed through the Lightning Network. This enables fast and scalable microtransactions that eventually interact with Directed Acyclic Graph (DAG) nodes representing the on-chain structure. GNN receives input from DAG nodes as well as metadata from a decentralized file storage system and off-chain streams. This allows the system to form a comprehensive graph of transactional relationships and behaviors. The GNN generates intelligent node encodings, which are compact representations that capture both who is transacting and how they behave over time. These encodings add to a graph-aware consensus mechanism, which uses them to assess how legitimate the blocks or transactions are before adding to the blockchain. This entire process reduces fraud, detects anomalies, and makes sure that only trusted data is validated. GTxChain represents a blockchain that doesn't just record transactions but also learns from them to improve security, efficiency, and trust.

4.2 Convolutional Neural Networks (CNNs)

Although traditional uses of Convolutional Neural Networks (CNNs) include image and pattern recognition, they have begun to be adapted for blockchain to reshape data into structured matrices or visual representations. CNNs are quite useful due to their ability to learn spatial hierarchies and detect subtle patterns. This makes them suitable for transaction pattern recognition [69], biometric verification [7], and fraud detection [26].

Various studies in current existing research use CNNs to operate on transaction data. For example, Hasan et al. [26] discuss the idea of converting blockchain transaction logs into two-dimensional matrices that mimic grayscale images, allowing the CNN-based system to detect fraudulent behavior through visual spatial analysis. Jones et al. [31] examine the use of CNNs to classify secure and malicious behavior. This is done by encoding the behavioral features of blockchain nodes into maps that can be interpreted by CNN filters.

CNNs can be seen within various domains throughout blockchain technology. A specific example would include biometric security. Asem et al. [7] suggest using a CNN-based architecture that can process biometric inputs, such as fingerprint and facial data. This experimental process provided both high-accuracy classification and tamper-proof audit trails.

Healthcare systems are another common domain where CNN applications stand out, specifically for preserving privacy. As an example, Alzubi et al. [5] place CNNs in federated learning platforms, allowing patient health data to remain on local devices. The CNN model learns from local data and contributes to the decentralized blockchain model without exposing raw information. This method combines privacy preservation with high performance, specifically for both disease classification and anomaly detection.

Other studies focused on CNN with the intention of improving temporal and contextual awareness. For instance, Wang et al. [69] merge CNN-based extraction with transformer-based temporal attention. This method gives better anomaly detection in blockchain transactions by encrypting spatial local patterns via CNN, along with sequence-level dependencies via a transformer. This solves the common weakness of CNNs when it comes to modeling long-term dependencies.

Principally, all CNN-based models focus on spatial pattern recognition in some way. Most methods convert blockchain data into structured formats like matrices and tensors. These are then passed through convolutional filters to extract features, making CNNs effective for spotting recurring patterns or anomalies.

CNN methods also have differences, including the input preprocessing methods and model architecture. Some models rely on spatial input formats, such as biometric and health records, while others rely on artificially structured transactional data. Some studies use CNN in isolation, while others use a more hybrid approach to make up for their temporal limitations.

Throughout this review, several gaps are noticeable due to pre-existing challenges in standardizing how blockchain data is formatted for spatial learning, specifically for CNNs. There is not much consistency in how transactional data is converted into image-like structures, which limits cross-study comparisons. The lack of interpretability also remains a concern, resulting in a lack of research on how CNN maps contribute to decision-making in blockchain contexts. There is also limited evidence of CNN usage in lower blockchain

Table 3: **Graph Neural Network Papers.** This table summarizes every gathered paper discussing deep learning for blockchain security (DL4BCS), specifically for GNNs.

Citation	Priv./Secur.	Att./Def.	Application Domain	Blockchain Layer(s)
[25]	Security	Defense	Crypto-based blockchain for fraud detection	Application, Data, Hardware
[14]	Security	Defense	Crypto-based blockchain for anomaly detection	Application, Data, Hardware
[32]	Security	Defense	Crypto-based blockchain for smart contract vulnerability detection	Application, Data
[30]	Security	Defense	Crypto-based blockchain for malicious transaction detections	Application, Data
[12]	Security	Defense	Crypto-based blockchain for anomalous node detection	Application, Data
[42]	Security	Defense	Anomaly detection for supply chain blockchains	Application, Data
[57]	Privacy	Attack	Crypto-based blockchain for identity inference and entity clustering	Application, Data
[33]	Security	Defense	Crypto-based blockchain for fraud detection	Application, Data, Hardware
[54]	Security	Attack	Crypto-based blockchain for monitoring encrypted network traffic	Application, Data
[11]	Both	Defense	Intrusion detection using smart blockchain	Application, Consensus, Data
[56]	Security	Defense	Crypto-based blockchain for anomaly detection	Application, Data

layers like consensus and network, while the most commonly studied layers have been the application and data layer. CNNs are currently applied at the application and data layers within research, specifically where privacy, pattern recognition, and classification tasks are dominant. In hybrid systems, CNNs function as the initial feature extractor. It feeds processed data into sequential models for enhanced context modeling. We continue our discussion of the prominence of CNN-based architectures in Section 4.5. We also provide an in-depth summary of all the papers surveyed on integrating CNNs and blockchain in Table 4.

In conclusion, CNNs are a versatile tool for blockchain security and privacy research. The fact that CNNs can adapt to non-image data shows potential for innovation, but standardized preprocessing protocols and interpretability frameworks are necessary for improved use in other areas. The real-world validation of CNN models for blockchain technology is an area that could be explored more in-depth in the future.

4.3 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) offer a transformative approach to blockchain security and privacy. GANs offer the most opportunities in augmenting datasets, simulating adversarial behavior, and generating synthetic data to test detection models. Since high-quality labeled blockchain data can be hard to retrieve (especially when it comes to fraud and attack scenarios), GANs feature a solution for gaining data that can benefit the stress-testing of a system. A GAN includes a generator (which can create synthetic data) and a discriminator (which differentiates between real and generated data). As the generator and discriminator interact, realistic data is continuously generated. The relationship between the generator and the discriminator allows for continuous improvement in relation to a vast variety of blockchain applications, especially for security purposes.

A common application of GANs in relation to blockchain is anomaly detection. In Rawlins et al. [47], the overall concept features a lightweight GAN architecture that is designed for low-latency environments. A realistic example of this would be real-time fraud detection in blockchain-based transaction systems. The model proposes a balance between efficiency and performance, which makes it ideal for edge deployment while still maintaining high classification accuracy. Other researchers have taken a similar concept but used a more heavyweight approach. A particular example of this would be Pawar et al. [44], which uses a gradient penalty to ensure training stability and better quality. This paper focuses on high-impact attacks and demonstrates how Wasserstein improves convergence and robustness in terms of detecting adversarial patterns in complex transaction networks.

Another way GANs have been used in current research is to simulate cyberattacks on the blockchain infrastructure. In Rabieinejad et al. [45], the researchers propose a GAN-based detection model that focuses on identifying malicious patterns and threat indicators in Ethereum transaction graphs. This method uses adversarial mimicry, in which a generator produces synthetic attack samples with a discriminator that

Table 4: **Convolutional Neural Network Papers.** This table summarizes every gathered paper discussing deep learning for blockchain security (DL4BCS), specifically for CNNs.

Citation	Priv./Secur.	Att./Def.	Application Domain	Blockchain Layer(s)
[26]	Security	Defense	Fraud detection using blockchain	Application, Data
[69]	Security	Defense	Blockchain transaction classification system	Application, Data
[7]	Both	Defense	Biometric verification using private blockchains	Application, Data
[31]	Security	Defense	IoT security using IoT-based blockchain	Application, Consensus, Data, Network
[5]	Both	Defense	Secure model update logging using private blockchain	Application, Data, Hardware, Network

learns to differentiate them from safe transaction behavior. This refines its threat detection capability over successive iterations. Unlike static or rule-based approaches, this method allows for real-time adaptation to emerging zero-day threats. This paper mentions that their model outperforms the baseline classifiers when it comes to precision and recall. This is a perfect example of the benefits of GAN-generated training augmentation for blockchain cyber defense. They also discuss the ability of GANs to uncover hidden vulnerabilities in transaction patterns, which provides a tool to locate threats in permissionless environments. We provide a visual explanation of the two-phase deep learning architecture found in [45] in Figure 4.

In addition to [45], Lei et al. also use GANs to simulate attacks [35]. This paper presents a forensic framework that supports comprehensive model training and forensic auditing on blockchain. There exists additional research that uses GANs to discuss data privacy and biometric spoofing defense. For example, Ghani et al. [22] talk about how GANs can be used to simulate fake biometric data (and defend against it) using blockchain as an audit layer. This method highlights GAN-based applications and how they can simulate attacks as detailed as training systems to defend against them. The authors talk about using blockchain to provide traceability and trust in the data generation process, which would strengthen the security of biometric verification systems.

GANs can also be applied to improve the quality of data for blockchain logs. In particular, Qadear et al. [2] discuss GANs ability to complete incomplete blockchain transaction records. This method contributes to blockchain analytics in situations where data may be missing due to latency, corruption, or inconsistent formatting.

Another area of interest is the hardware and deployment perspective. In particular, Dirgantoro et al. [18] discuss a privacy-preserving data generation system designed for edge nodes in blockchain-hybrid Internet of Things (IoT) systems. The researchers mention how GANs can generate synthetic sensor data while maintaining user privacy and reducing the need to rely on cloud infrastructure. This method features a federated or edge-distributed GAN-based architecture for privacy-conscious deployments.

With the many possibilities that GANs offer, there are still several gaps in existing research for DL4BCS. The lack of explainability in GAN-generated outputs includes weak reasoning for why synthetic samples improve detection. Another concern includes integrity evaluation. Since GAN models generate fake data, it is very important to develop strict metrics and validation frameworks to assess their fidelity, diversity, and impact on tasks. Most of these studies evaluate GANs on custom-built or non-public datasets, which hinders replication and benchmarking.

GAN-based models are also rarely applied to all blockchain layers. Most focus on the data and application layers, which typically aims at fraud detection or privacy enhancement. Very few studies mention the consensus layer or the network layer, although these layers have proven to be very important to blockchain security.

To summarize, GANs can be a powerful and flexible tool for blockchain research. GAN-based models allow synthetic data generation, adversarial simulation, and model enhancement, which can be used in fraud detection, biometric spoofing, threat modeling, and privacy-preserving data generation. However, there is much room for this field of study to evolve and become more interpretable, standardized, and reproducible. The use of GANs to improve blockchain security is incredibly promising for future research. We provide a visual, in-depth summary of all DL4BCS research articles collected that use GAN-based models in Table 5.

4.4 Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNNs)

Long short-term memory (LSTM) and other recurrent neural networks (RNNs) are prevalent neural networks that can be seen within a substantial amount of published papers. Both LSTM and RNN models excel in sequential data modeling for blockchain. A consistent theme is the use of time-series data, but models differ in memory span, bidirectionality, and hybridization. For instance, some researchers integrate RNNs with transformer encoders to improve context awareness. While LSTM models are versatile, gaps exist in adapting them for high-volume blockchain environments with low latency requirements. Also, a few studies validate their robustness against adversarial inputs. These models frequently engage the application, network, and consensus layers. Therefore, we can conclude that work is needed to bridge model complexity with real-time deployment feasibility.

Across the papers we have gathered specifically for LSTM models, there is a clear pattern in the type of blockchain

Table 5: **Generative Adversarial Network Papers.** This table summarizes every gathered paper discussing deep learning for blockchain security (DL4BCS), specifically for GANs.

Citation	Priv./Secur.	Att./Def.	Application Domain	Blockchain Layer(s)
[74]	Both	Defense	Blockchain to improve secure communications and access control	Application, Data
[45]	Security	Attack	Ethereum-based blockchain for cyber threat hunting	Application, Data
[18]	Security	Defense	Ethereum-based blockchain for smart security systems	Application, Data
[2]	Both	Attack	Data augmentation and anomaly detections using blockchain	Application, Data
[75]	Security	Attack	Data augmentation using bitcoin-based blockchain	Application, Data
[47]	Security	Defense	Crypto-based blockchain for fraud predictions	Application, Data
[22]	Both	Attack	Using private blockchains for authenticity and tamper resistance	Application, Data
[44]	Security	Defense	Smart contract based attack detection system	Application, Data
[35]	Security	Attack	Audit trails in network intrusion detection	Application, Data

used for a significant number of papers. Among the LSTM-based papers we refer to within this paper, half of these papers use IoT as the main focus for its blockchain type. Meanwhile, the other half of these papers discuss blockchain technology in general terms, and applies blockchain primarily as a decentralized ledger system. Our findings also allowed us to discover a singular paper that uses a mix of both IoT and smart contracts as its blockchain type, creating a very unique distinction and system compared to the other papers [41]. RNN-based studies decide to take an approach with a different type of blockchain, with one paper employing Proof-of-Authority (PoA) [65] and the other making use of cryptocurrency, with a focus of both Bitcoin and Ethereum [46]. From this, we can see that papers using RNN or LSTM architectures use blockchain in general terms or IoT as the primary focus, which allows room for more research using different types of blockchain, such as supply chain or even smart contracts.

It is important to clearly state that some researchers create models purely for security purposes, while others create systems that can be used for both security and privacy purposes. A distinctive paper in our search creates a system just for privacy using an LSTM model [39], making this paper extremely unique within these specific neural networks.

There is a special mix of how researchers choose to apply their concepts in real-world scenarios. For instance, several of these papers consist of solutions to improve or assist certain systems that require much more security and immutability. Some areas of focus would include healthcare due to patient-privacy laws (such as HIPAA) [13; 9], energy management [65], and mobile financial fraud detection systems [46]. Other papers focus more on data security as a whole, such as proper storage of IDS data [4] and improvements on Collaborative Intrusion Detection Systems (CIDS) [41].

Unlike how there is a special mix of applications of blockchain within each individual paper, each paper applies its system towards specific layers of blockchain, with no unique pattern. For instance, several papers that use RNN or LSTM architectures decide to apply to both the application layer and the consensus layer of blockchain. However, other existing research applies only to the application layer [41; 37] or the consensus layer [65].

If we were to introduce how the data layer applies to existing LSTM and RNN research for DL4BCS, it is crucial to note that current research that uses RNNs does not apply to the data layer at all. The cause of this could be related to the structure differences between RNNs and blockchain. RNNs focus on sequential data, while the data layer of blockchain handles data retrieval in a structured format. However, one model that uses LSTM applies in both the application and data layer of blockchain [37], while another uses the data layer only [13]. We can therefore conclude that LSTM-based models may provide more flexibility for researchers due to their ability to take advantage of several blockchain layers within one system.

There are several notable gaps that we can clearly state within DL4BCS, specifically for LSTM and RNN. As mentioned above, only one study refers to the use of deep learning for blockchain for privacy purposes [39]. Deep learning models require a substantial amount of data in order to properly train and learn effectively, which ultimately results in many privacy concerns. In order to address these privacy concerns, more research on integrating deep learning for blockchain should be considered. Based on the five layers of blockchain, we can also see that there was no usage of the network layer or hardware layer within these articles. The network layer is important to allow communication between nodes on a blockchain, and deep learning should absolutely be seen as a potential candidate to improve this communication system. The hardware layer of blockchain could be improved with deep learning, specifically with LSTM and

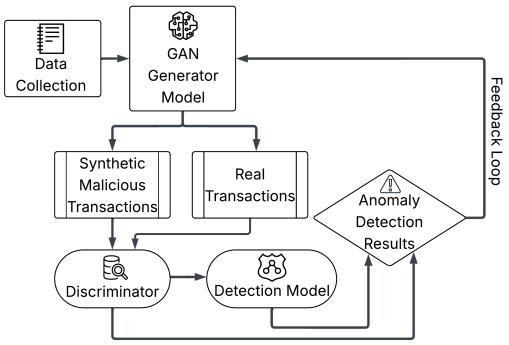


Figure 4: Generative Adversarial Networks for Cyber Threat Hunting in Ethereum Blockchain [45]. This diagram illustrates the two-phase deep learning architecture mentioned for adversarial threat detection in Ethereum blockchain networks. In phase one, real Ethereum transaction data is used to train a Conditional Tabular GAN (CTGAN). Once the data goes to the GAN generator, it uses the real transactions to create malicious synthetic transactions. Both real transactions and newly generated synthetic transactions then get sent off to the discriminator. The discriminator uses multiple iterations to improve its ability to detect fakes and simulate realistic adversarial activity. While still in the discriminator, both the synthetic and the real transactions get merged into a unified dataset. In phase two, the dataset is now sent off to a detection model called a bi-directional long short-term memory (BiLSTM) network. This detection model captures sequential patterns to accurately identify subtle adversarial behaviors. The system effectively detects malicious attacks during early phases of execution, which achieves a near-perfect accuracy in distinguishing malicious transactions from trustworthy transactions. To conclude, these results are then sent to the user.

RNN models, as the hardware layer allows for anomaly detection and enhancing security as well as privacy. Based on previous statements about gaps in Section 3.4, it is clear that RNNs are extremely underrepresented in comparison to LSTMs, which seems to be widely represented in existing DL4BCS research. As a visual summary, we provide Table 6 and Table 7, which shows existing research and how they fall under specific categories.

4.5 Multilayer Perceptrons (MLPs) and Hybrid Models

Multilayer Perceptrons (MLPs) are widely used in lightweight or hybrid blockchain architectures. These models serve as classification layers in federated learning systems or low-resource edge nodes. MLPs consistently act as the final decision layer in hybrid systems, typically following CNN or RNN-based feature extraction. Although MLPs often act as the final layer when it comes to hybrid systems, many of the hybrid-based articles that exist in current research for DL4BCS consist of many unique combinations of different neural networks. These combinations allow researchers to create extremely specific systems to solve problems and apply results to certain real-world applications. We will dive deep into how each paper classifies itself in terms of privacy or security and how each neural network is combined within these hybrid papers. The interaction of various deep learning architectures impacts the overall application of blockchain. Similar to previous sections, a table has been provided for both MLPs (Table 8) and hybrid models (Table 9) to assist readers with viewing similarities and gaps between our methodology for these architectures.

Immediately, we noticed a lack of DL4BCS articles that utilize MLPs only. This could be due to how MLPs are typically used as the final layer in hybrid systems, as mentioned above. However, each paper features a unique blockchain application along with different layers of blockchain. To begin, only one paper in our methodology aims to improve both privacy and security while using MLPs and AI of Things (AIoT) [15]. Other researchers have different applications, as they focus purely on security improvements. Interestingly enough, papers that focus on security have their own individual methods of applying deep learning to blockchain. While one paper discusses blockchain in general terms to help identify hostile nodes in Proof-of-Stake (PoS) blockchain networks [6], another discusses how general healthcare blockchains can be used to help protect against threats [53], and another article discusses how cryptocurrency (such as Bitcoin) can create threat and anomaly detection systems [19]. There are no patterns within blockchain layers to which these systems and concepts are applied using MLPs. For instance, existing discussions about identity management [15] and threat protection in healthcare [53] utilize the application and data layers in blockchain. Meanwhile, the identification of malicious nodes in blockchain networks [6] uses only the consensus layer, while another paper that discusses threat and anomaly detection with cryptocurrency [19] uses only the data layer. It is intriguing to see that each research paper has taken a unique approach, although each paper focuses purely on using MLPs to their advantage to improve DL4BCS.

Hybrid models are able to use several neural network architectures within one model to create solutions and improvements to systems. Notably, currently existing hybrid papers don't feature an application or system that focuses on privacy, which has proven to be common. A majority of existing research focuses purely on creating an application or system for security purposes, while others focus on implementation for both privacy and security. This is crucial to show that hybrid neural networks have the capabilities to improve both security and privacy, depending on the target for the researcher.

Each paper that applies to both security and privacy features different types of blockchain. From cryptocurrency-based blockchains [36] to healthcare blockchain [9], many researchers have taken unique approaches to creating systems using the different types of blockchain. Notably, most of these papers share the same application of their research. These papers discuss applying their hybrid models to various types of detection systems, even by using various types of neural networks, blockchain types, and different blockchain layers. For example, Elangovan et al. [19] focus on using both RNNs and auto-encoders to create a system for anomaly detection in healthcare systems by using smart contracts in the application layer, traffic analysis in the network layer, and the data layer [19]. Research has also resulted in using MLPs and Denoising Auto-Encoders (DAE) together on the supply chain blockchain to detect anomalies, which utilizes the application and network layer as well, with the addition of the consensus layer [59].

Table 6: **Long Short-Term Memory Network Papers** This table summarizes every gathered paper discussing deep learning for blockchain security (DL4BCS), specifically for LSTMs.

Citation	Priv./Secur.	Att./Def.	Application Domain	Blockchain Layer(s)
[4]	Security	Defense	Storing IDS data using blockchain	Application, Consensus
[39]	Privacy	Neither	Sentiment analysis using blockchain	Application, Consensus
[13]	Both	Defense	Ethereum-Improving healthcare blockchain using smart contracts and ethereum	Data
[50]	Security	Defense	HO-Auth using IoT	Consensus
[41]	Security	Defense	Creating a CIDS using smart contracts and IoT	Application
[37]	Security	Attack	Detecting LDDoS attacks with IoT	Application, Data
[9]	Both	Defense	IoT for healthcare	Consensus

Table 7: **Recurrent Neural Network Papers** This table summarizes every gathered paper discussing deep learning for blockchain security (DL4BCS), specifically for RNNs.

Citation	Priv./Secur.	Att./Def.	Application Domain	Blockchain Layer(s)
[65]	Security	Defense	Energy management with PoA blockchain	Consensus
[46]	Security	Defense	Crypto-based mobile financial transactions and fraud detection	Application, Consensus

As mentioned in Section 4.2, CNN-based architectures are commonly found in hybrid models. CNN and LSTM can be used together with the data layer of the cryptocurrency-based blockchain to create a structure for financial predictions [52] and threat detections [62]. CNN and transformers are also commonly used together for blockchain technology based on financial transactions to improve threat detection, which is similarly based specifically on the blockchain data layer [69; 51]. For instance, Saveetha et al. [51] focus on improving DDoS attack detection, updating minor nodes, and updating transaction requests to improve node reputations, creating a global model [51]. To properly demonstrate how this system works and elaborate upon its prominence in existing DL4BCS research, we provide Figure 5.

Additionally, Airlangga and Gregorius [3] is based purely on the data layer of blockchain, using CNN, LSTM, and MLP to create a threat detection and fraud analysis system with open metaverse blockchain, making this paper completely unique compared to others in our methodology. Yazdinejad et al. [71] discuss security improvements and is also unique since it applies to the hardware layer of blockchain, which, as we have seen, is very difficult to find within existing research. To be specific, [71] introduces a mix of RNN and LSTM for cryptocurrency blockchain to aid with malware detection, focusing primarily on the hardware layer of blockchain.

Now that we have thoroughly distinguished each paper that discuss security applications, we can begin to address papers that have focused on both security and privacy. Right away, there is a clear distinction showing that a majority of these papers seem to focus primarily on the application layer, consensus layer, network layer, and data layer of blockchain. This is an important note showing that researchers who focus primarily on both privacy and security will have to apply more layers of blockchain to their research. Several of these papers focus on permissioned hyperledger blockchain, with one group of researchers utilizing RNN and Bidirectional Long Short-Term Memory (BiLSTM) to introduce a new intrusion detection system [38], and another using transformers along with BiLSTM for financial applications such as risk identification and transaction management [70]. Vijay Anand et al. [66] focus on application, consensus, network, and data using CNN and LSTM (with a specification on auto-encoders) to introduce a new model for threat detection and secure blockchain transactions. A final significant study in our collection of hybrid-based papers features the usage of Hyperledger blockchain, applying purely to the network layer (by featuring data-streaming storage) and the data layer (by using cryptographic ledgers to store information) [58]. By using a substantial mix of neural networks such as LSTM, CNNs, Artificial Neural Networks (ANNs), and Gated Recurrent Units (GRUs), the end goal of this research paper is to apply its content conceptually to a wearable IoT [58].

Existing hybrid research within DL4BCS holds unique statuses within every category we chose to analyze. From different usages of various combinations of neural networks, to discussing both privacy and security (or just strictly security), there is no set pattern that can be seen within our methodology of hybrid papers for DL4BCS. However, it is easy to identify several gaps across these articles. For instance, as mentioned above, there is no research on privacy-based applications. There were also a significant number of papers that used the application layer, consensus layer, network layer, and data layer. However, we only successfully found one hybrid paper that chose to focus primarily on the hardware layer of blockchain, which seems to be a consistent theme between not just hybrid papers, but all papers on DL4BCS in our methodology. As mentioned previously, regardless of whether the paper addresses both security and

privacy or decides to focus on just security, the final application of blockchain remains consistent in current research. Each paper had a final application to create a system to detect threats, anomalies, intrusions, or even malware. Hybrid technology has proven to be successful in improving security. Overall, their reliance on preprocessed or feature-rich inputs limits their adaptability in complex blockchain-based systems.

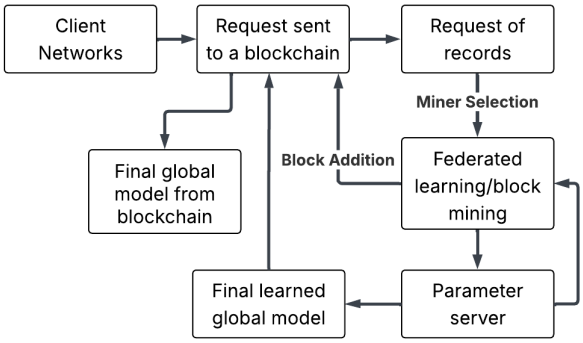


Figure 5: Performance Analysis of Blockchain-Based Secured Distributed Deep Federated Learning for Wearable Internet of Things [51]. This diagram captures the main concept of this research. It presents the idea of using federated learning to detect DDoS attacks in a framework that is integrated with blockchain. Client networks begin by starting data requests, which are ultimately sent to a blockchain and placed in a request queue. Reputation-based miner selection occurs, where miners are evaluated on several factors (such as model accuracy and prior wins). Then, miners participate in federated learning and block mining. These updates are sent to a parameter server and cycle through federated learning, allowing the parameter to repeat iteratively. Once training is completed, the final learned global model gets sent back to the blockchain as a new stored block. The secure global model gets retrieved from the blockchain to be used by clients and miners for DDoS detection. The overall purpose of federated learning with blockchain in this case is to create a tamper-resistant model as well as reliable DDoS detection.

4.6 Transformer Networks

Transformer models have recently been adapted for blockchain tasks that demand context-rich sequence modeling. They excel in phishing detection [16], risk prediction in financial records [36], and federated health data management [8]. Transformers can be applied to analyze transaction propagation patterns, protocol interactions, and user behavior across blockchains. Due to these unique attributes, transformers are commonly used in existing DL4BCS research.

Existing research on transformer models in DL4BCS present unique characteristics between every paper, which separates them from research papers that use different deep learning architectures. However, we are able to identify that these papers seem to properly use the data layer of blockchain, whether it is for transaction-based purposes or to store hashes to make their systems tamper-resistant. It is also important to state that existing research does not focus only on privacy. Each paper is stuck between focusing only on improving security or improving both security and privacy simultaneously. Much like previous neural networks we have discussed, there is once again a clear gap of research in systems that improve privacy alone without inclusion of security. We can also see that a majority of existing research focuses on cryptocurrency-based blockchain, whether it's Ethereum [16] or Bitcoin [1].

It's evident that each article seems to focus on creating a detection system, even with each paper focusing on different threat types. For instance, Mnasri et al. [40] focus on using transformers for healthcare and the Ethereum blockchain, choosing to focus on improving the security and privacy of medical records. Therefore, the researchers have decided to use the application layer (for smart contracts) and the data layer (for hash protection). Next, Batool et al. [8] discuss integrating transformers and IoTs for anomaly detection in networks, which also uses the application and data layer for security purposes. There are two papers within our methodology that also share practically the same attributes. For instance, Abasi et al. [1] discuss using Bitcoin and transformers to improve threat and anomaly detection systems, while Choi et al. [16] use Ethereum to create a fraud and phishing detection system. Both share the trait of using only the data layer to analyze transaction attributes. Finally, Liu et al. [36] decide to merge cryptocurrency and smart contract blockchains to advance a threat detection and financial prediction system, which essentially focuses on both privacy and security. We can conclude that this article is unique since it chooses to prioritize the network layer rather than the data layer to create their concepts.

A common strength is efficiency and parallelism in sequence modeling. However, there are differences in architecture depth and attention mechanisms. To be specific, some use Bidirectional Encoder Representations from Transformers (BERT) embeddings (since BERT can improve natural language processing), while others employ full encoder-decoder setups. One significant gap is the scarcity of transformers in smart contract security, despite their success in natural language processing. These models primarily function at the application and data layers, although network-level applications are emerging. More domain-specific pre-training and explainability tools would enhance their utility.

4.7 Cross-Network Comparison

Our analyzed literature demonstrates diversity in terms of using different neural network architectures in various application domains in DL4BCS research. Each neural network architecture offers unique strengths in current studies. However, they also come with domain-specific limitations. Graph Neural Networks (GNNs) demonstrate superior performance in terms of capturing the relational structure of blockchain transaction graphs, especially for tasks such as node classification, fraud detection, identity inference, and anomaly detection. Notable implementations use spatial-temporal GNNs to model transaction propagation over time. A primary reason why these models stand out is their interpretability and alignment with the graph-based topology of blockchain. They do, however, find standard dataset availability and scalability for ledgers to be one of their major challenges.

Convolution Neural Networks (CNNs) are applied by converting transaction data into spatial matrices or biometric

Table 8: **Multilayer Perceptron Papers.** This table summarizes every gathered paper discussing deep learning for blockchain security (DL4BCS), specifically for MLPs.

Citation	Priv./Secur.	Att./Def.	Application Domain	Blockchain Layer(s)
[15]	Both	Defense	BSecure identity management with AIoT	Application, Data
[6]	Security	Defense	Detecting malicious nodes in PoS blockchain networks	Consensus
[53]	Security	Defense	Healthcare and threat protection using general-purpose healthcare blockchain	Application
[19]	Security	Defense	Threat and anomaly detections using Bitcoin	Data

Table 9: **Hybrid Papers.** This table summarizes every gathered paper discussing deep learning for blockchain security (DL4BCS), specifically for hybrid models.

Citation	Neural Networks	Priv./Secur.	Att./Def.	Application Domain	Blockchain Layer(s)
[67]	RNN, Auto-encoder	Security	Defense	Detection in healthcare using IoMT	Application, Data, Network
[59]	MLP, DAE	Security	Defense	Supply chain-based blockchain for anomaly detections	Application, Consensus, Network
[52]	CNN, LSTM	Security	Defense	IoT and bitcoin blockchains for financial predictions	Data
[66]	CNN, LSTM, Auto-encoder	Both	Defense	Threat detection using general blockchain	Application, Data, Network, Consensus
[62]	CNN, LSTM	Security	Defense	Crypto-based blockchain for threat detections	Data
[69]	CNN, Transformer	Security	Defense	Anomaly detections with financial transactions	Data
[58]	LSTM, CNN, ANN, GRU	Both	Defense	Hyperledger-based blockchain for wearable IoT	Data, Network
[51]	MLP, CNN	Security	Defense	Transaction-based blockchain for threat detection	Data
[3]	MLP, CNN, LSTM	Security	Defense	Open metaverse blockchain for threat detection and fraud analysis	Data
[38]	RNN, BiLSTM	Both	Defense	Permissioned hyperledger intrusion detection system	Application, Consensus, Network, Data
[71]	RNN, LSTM	Security	Defense	Crypto-based blockchain for malware detection	Hardware
[70]	BiLSTM, Transformers	Both	Defense	Permissioned hyperledger for financial risk identification and transaction management	Application, Consensus, Network, Data

Table 10: **Transformer Papers.** This table summarizes every gathered paper discussing deep learning for blockchain security (DL4BCS), specifically for transformers.

Citation	Priv./Secur.	Att./Def.	Application Domain	Blockchain Layer(s)
[40]	Both	Defense	Ethereum-based blockchain to secure medical records	Application, Data
[8]	Security	Defense	Anomaly detections in networks using IoT blockchain	Application, Data
[1]	Security	Defense	Bitcoin-based blockchain for threat and anomaly detection	Data
[16]	Security	Defense	Ethereum-based blockchain for fraud and phishing detection	Data
[36]	Both	Neither	Smart contracts and crypto-based blockchain for threat detection/financial predictions	Network, Data

maps. Their strength can be seen in fraud detection and privacy-preserving applications, especially when embedded in federal edge systems or IoT medical record frameworks. CNNs also appear frequently in hybrid frameworks where they handle spatial feature extraction before Long Short-Term Memory (LSTM) or transformer modules come in for sequential interpretation. However, CNNs lack adaptability across heterogeneous data formats since they require significant preprocessing.

Recurrent Neural Networks (RNNs), especially LSTMs, are important in time series modeling for sequential transaction analysis, intrusion detection, and malware propagation tracking. Their ability to keep a memory of past transactions makes them useful for detecting long-range patterns in decentralized systems. Many LSTM implementations are deployed in privacy environments such as healthcare and IoT, where latency and integrity are critical. However, LSTM and RNN-based systems often lack robustness when it comes to adversarial settings and are computationally intensive, limiting real-world deployments at scale. A subset of models integrates bidirectional or attention-enhanced variants to address these weaknesses.

Generative Adversarial Networks (GANs) contribute through synthetic data generation, adversarial attack simulation, and data augmentation for training classifiers. As seen in our analysis, various studies use GANs to simulate fraudulent behavior in cryptocurrency systems or biometric signals to test defensive mechanisms. They are also used to reconstruct missing transaction features or fill training data sets. GANs have critical gaps when it comes to interpretability and are very rarely accompanied by explainability modules, which poses a challenge in regulatory compliance and trust in sensitive domains such as finance.

Although transformer-based architectures are relatively new in blockchain applications, they have gained traction since they outperform LSTMs when tasked with handling context-rich sequences. Proper context handling can allow models to identify threats properly through pattern shifts and trends. Transformers are often merged into hybrid models or fine-tuned with domain-specific embeddings, but still struggle when it comes to computational demands and lack pre-trained models for blockchain structures.

While Multilayer Perceptrons (MLPs) can be viewed as a simpler architecture, they serve a very important role in lightweight systems. Their presence is consistent in Proof-of-Stake (PoS) monitoring, where interpretability and rapid execution are prioritized. However, MLPs lack the representational depth required for complex sequence or graph analysis and thus are rarely deployed as standalone models.

As mentioned above, hybrid-based architectures combine two or more neural networks. They represent a dominant trend in recent literature. For example, CNN-LSTM combinations are great in fraud prediction by merging spatial encoding and sequential learning, while GNN-Transformer hybrids attempt to bridge structural and temporal contexts. The idea of combining different neural networks is promising, but it requires careful coordination of model training and data integration pipelines. Hybrid models often get the highest benchmark scores in both privacy and security tasks, indicating their strong practical potential even though they require greater computational and engineering effort.

Across all architectures, some common traits can be concluded within current DL4BCS research. For example, the prevalence of hybrid models, the need for better standardization of datasets, and the growing focus on privacy-preserving and federated learning deployments are becoming more apparent. A persistent gap across most models is the lack of explainability tools, despite the increasing regulatory attention blockchain systems have. Another concern is deployment scalability; few models are validated outside of simulated environments, resulting in real-world applications (especially in enterprise settings or cross-chain systems) being limited. Comparably, security applications (like intrusion detection, fraud analysis, and malicious behavior simulation) are widely researched, while privacy-focused tasks (such as anonymization, secure recordkeeping, and identity protection) still require deeper exploration. This comprehensive combination of DL4BCS literature suggests that, while neural networks hold great promises to enhance blockchain, future work must address reproducibility, interpretability, and deployment barriers to unlock their full potential.

5. CHALLENGES IN DL4BCS

Although there are many promising results from integrating deep learning with blockchain, there are still many chal-

lenges that need to be addressed. These challenges are typically a result of limitations that are outside the control of researchers. These challenges range from data limitations to latency, and even include unfortunate trade-offs when attempting to make an area of a system stronger.

5.1 Data Scarcity and Quality Challenges

Major existing issues in DL4BCS research are data availability and quality. Data cannot be made available in their entirety to users due to the privacy-preserving feature of blockchain networks. Even when data is available, it is poor in quality and filled with errors. These errors range from incompleteness, outdated information, inconsistencies, and biases. Low-quality data limits the potential of a proposed system to capture real-world dynamics accurately. These limitations severely impede the development of real-time security models, since poor quality data is devastating to both training efficacy and model reliability [14; 25].

Additionally, the time-critical and high-dimensional nature of blockchain transactions complicates the gathering of data. Supervised learning techniques, particularly those rooted in Graph Neural Networks (GNNs) and Convolutional Neural Networks (CNNs), heavily rely on accurate labeling and adequate temporal information to effectively detect fraud and anomalies [32; 31]. The decentralized anonymous property of blockchain compounds these issues by diffusing available data and limiting access for necessary contextual data [5]. To counter these limitations, others have attempted to find other approaches, such as data augmentation strategies specially those that involve the use of Generative Adversarial Networks (GANs) [75]. Such methods are valuable in terms of directions, but are still limited by the original training data quality, including how representative the created samples are [47; 2]. In conclusion, the lack of quality standards and privacy-oblivious datasets poses a significant obstacle to further advancement of deep learning models in blockchain security (DL4BCS). Future research will need to address the creation of benchmark datasets and privacy-oblivious data collection methods to enable further development of the discipline.

5.2 Real-Time Constraints

In order to have a successful integration of deep learning with blockchain for security purposes, there are several real-time constraints that researchers should be aware of. Real-time performance is critical for blockchain security, as malicious threats and anomalies need to be detected within seconds to properly respond. However, there are several challenges included in this requirement, such as latency. Latency severely limits the ability to detect and respond to threats in time before damage has occurred within a system. Deep learning models also have high computational demands, especially when it comes to continuously updating blockchain data, which can also negatively delay the system response speed to detect threats [25; 14].

Processing time-sensitive data repeatedly can be seen as one of the main issues in this area of study. Blockchain networks have the ability to generate large amounts of transactions, requiring in-depth analysis. However, deep learning models are usually dependent on complex data pipelines that involve formatting, aggregation, and structural updates before any predictions can be made [31; 5]. This preprocessing increases latency for the entire system, particularly in environments where network conditions vary and where data is inconsistent.

High time complexity is involved in training and inference for many deep learning models, which serves as another obstacle. Some approaches in existing research rely on temporal dependencies, which may require sequence construction. This results in delays in threat and anomaly detection that are behind real-time detection expectations [32]. Although some models are designed to be more efficient, they can still fall short when deployed in live blockchain networks. Communication delays, consensus mechanisms, and hardware limitations can further worsen latency and speed of a threat detection system [4; 50].

Researchers have attempted to reduce latency by implementing several methods in their deep learning models. Common methods include slimming models, offloading computations, or coordinating processing with federated learning [66]. Although these methods have proven to be efficient towards lowering latency, there is often a trade-off by implementing one of these methods, including sacrificing accuracy, decentralization, or scalability [47; 15; 8]. It is important to state that these methods cannot completely eliminate latency issues due to real-world constraints. This, therefore, serves as a temporary solution to reduce latency while weakening other areas of the system at the same time.

There is a notable amplification of real-time constraints in systems that integrate deep learning with blockchain consensus. In order to synchronize these two technologies, there must be a standard of consistency across distributed nodes. However, the required synchronization often results in delays that can hinder responses from the system [8]. Intuitively, latency causes models to typically report below-average inference times when placed under test conditions. However, when applied to real-world blockchain environments, there are additional factors to consider, such as communication lag and transaction throughput, which weakens performance and increase load [65; 66; 58].

Low-latency inference in blockchain security is still incredibly sought after, as there are many factors that contribute to latency issues. Although these issues are typically out of researchers' control, they should be viewed as an open challenge that can be addressed through model optimization. It could also be improved upon with more developments of blockchain-compatible architectures that prioritize both speed and accuracy equally, rather than choosing one over the other.

5.3 Privacy Constraints

A big concern regarding deep learning and blockchain integration is managing the trade-off between obtaining quality data for detection accuracy and preserving user privacy. Deep learning methods require access to sensitive information, such as transaction histories and activity logs. The transparency and immutability of blockchain clash with deep learning requirements, which raises concerns regarding data exposure [13]. Model input often needs to be fully visible in order to accomplish meaningful learning. However, doing this creates an opportunity for privacy breaches. CNN models seemingly require raw transaction matrices, but very few existing DL4BCS articles that use CNN-based models incorporate encryption, masking, or other measures to pro-

tect data during processing [5]. Similarly, GNN-based models risk leaking information or being manipulated through adversarial transactions if node embedding is not adequately secured [54]. Although node access controls are implemented, feature-level privacy remains largely unaddressed, as seen in some supply chain use cases [42].

GAN models bring about new privacy risks by using sensitive real data to create synthetic samples. Although having the ability to generate synthetic data can reduce the reliance on raw data sets, existing research does not currently evaluate whether synthetic outputs might accidentally expose private information. A common gap is the lack of methods to preserve privacy. Some examples of possible bypasses could be differential privacy [72] or federated training [22], which is presented in multiple sources. Hybrid models are especially prone to this situation as they combine multiple deep learning techniques to take advantage of their unique strengths. However, this also results in each of their vulnerabilities inflicting a significant effect in systems. In domains like healthcare or finance, where information is very sensitive and regulations limit access to full datasets, neural network models often require continuous data that blockchain systems simply are unable to provide. Without clear transparency controls or consent mechanisms, these models risk violating privacy while enhancing fraud detection. While blockchain ensures traceability and tamper-resistance, the integration of deep learning introduces new privacy challenges despite its security-enhancing potential. However, the complex tradeoffs that come with it are something that most systems are not yet prepared to resolve.

5.4 Opposing Demands of Scalability, Accuracy, and Speed

A challenge found in various deep learning approaches was managing the trade-off between accuracy, speed, and scalability within blockchain environments. Blockchain systems generate high-frequency, high-volume data, demanding timely and precise threat detection. However, many models struggle to meet all three demands simultaneously. As authors tried to improve detection accuracy through various methods, including adding layers, using spatial-temporal features, or generating synthetic samples, they consistently increased computational overhead. For example, transforming blockchain transaction records into a graph-like structure or image-like inputs adds heavy processing overhead. This slows the inference times, making models less suitable for real-time anomaly detection or fraud monitoring on fast-moving blockchain networks. Even lightweight models can cause memory strain and latency, which slows down their effectiveness in large or continuously updating blockchains.

Attempts to address these issues through federated learning, lightweight architectures, or synthetic data augmentation often led to new constraints. Models such as federated CNNs, which are trained across decentralized blockchain nodes, experienced synchronization delays and struggled with performance degradation when local nodes had limited data variety. While GANs have proven to be helpful for creating synthetic blockchain transaction data to address class imbalance, they can become too slow when scaled up to support broader blockchain use cases.

Several GNN models also had difficulties maintaining up-to-date representations of rapidly evolving blockchain graphs, where deeper architectures increased precision but significantly reduced speed and scalability. Across various neural networks, including CNN, GNN, and GANs, authors discussed that improving one dimension generally comes at the cost of another. For example, model depth often comes at the cost of efficiency. This requires task-specific trade-off analysis or adaptive modeling. However, this can be seen with alternative elements, such as a trade-off between security and privacy.

6. FUTURE RESEARCH DIRECTIONS

Throughout the above-discussed papers, there were various research gaps due to the number of challenges that researchers face while merging deep learning and blockchain. In this section, we provide several possible research directions for researchers to consider. Giving more attention and focus to these areas could potentially result in solving the challenges researchers seem to be constantly running into. It could also solve long-term struggles to improve systems that use deep learning to improve blockchain.

6.1 Research for Privacy

Although blockchain features a transparent and immutable design, its functionality to protect sensitive data decreases when merged with deep learning networks. A vast amount of existing research discusses security applications, with very few papers discussing privacy improvements. Since there is a lack of privacy-focused research, there is plenty of room for discovery and academic improvements. Although we previously mentioned trade-offs in Section 5.3 and Section 5.4, there seems to be a common trade-off between security and privacy. When integrating deep learning and blockchain, security may increase within a system, but privacy gets put at risk. While most of the reviewed models assume unlimited access to data, this is unfortunately not realistic due to ethical and regulatory laws for protecting sensitive data. These rules lead researchers to struggle with a lack of data due to the importance of keeping privacy intact.

In the future, it is critical to develop techniques that enable deep learning to enhance blockchain functionalities without compromising confidentiality laws or degrading blockchain performance. As previously discussed, there are methods that can be used to aid with this situation, such as differential privacy, homomorphic encryption, and federated learning. These methods can allow neural networks to learn ethically in order to improve blockchain by avoiding sensitive data. However, there is not enough research to offer a full-proof solution, which leads to a vast need for future research on ethical privacy-based experiments. Overall, researchers have successfully created a vast number of systems that focus on improving security in blockchain, and the focus should be considerably shifted towards privacy improvements.

6.2 Understudied Blockchain Layers

Blockchain currently allows for a deep data hierarchy far beyond what the bulk of existing research currently has access to. While many papers focus on transactions versus accounts (including account statistics), smart construct internals, state transitions, and pending transaction pools, output behavior seems to be scattered.

Future research must look at how certain layers of blockchain can be utilized more for neural models. For instance, the hardware layer of blockchain is severely underused. Improvements with this layer could result in fixing common issues with deep learning and blockchain integration, such as latency. This could be achieved by developing new bytecode-feature encoders, which use graph representations of function calls, or by synchronizing time-sensitive data with confirmed transactions for reliable inputs for neural models.

Specifically, the consensus layer remains underused for model integrations, which introduces a potential research direction to explore how neural networks could be embedded directly into consensus logic. We could also consider the possibility of whether neural networks could be stored directly on-chain, such as using smart contracts to host neural networks. Smart contracts could also benefit from off-chain access, allowing smart contracts to call a neural network to classify images or flag certain behaviors.

Overall, deep learning could be integrated into the network via on-chain integration. Consensus and hardware layers in future research could also potentially make blockchain systems more adaptive and secure, particularly for fraud/attack detection. The overall goal is to build more models that can interpret lower-level information to detect more blockchain threats, which could be done by using underexplored blockchain layers(specifically, the hardware layer or consensus layer).

6.3 Neural Network and Blockchain Interoperability

To achieve the full potential that AI offers in decentralized systems, neural networks must be callable and verifiable using smart contracts. Higher-quality blockchain applications have a significant dependence on off-chain computation, which can impose reliance on external systems. As a result, this can break trust assumptions. Future research should explore model structures or approximation techniques that are on-chain or in trusted execution zones. Furthermore, researchers explore emerging technologies, such as zero-knowledge proofs, which can allow smart contracts to confirm outputs without revealing private data. A trusted pipeline from model to contract (and back) could unlock autonomous, real-time, decentralized intelligence, which has yet to be achieved in current designs and research. In summary, the access of the neural model within smart contracts can result in improvements for overall systems in many various areas, bringing the potential to solve the various issues found in current designs.

7. CONCLUSION

This survey provides a comprehensive overview of how deep learning has been applied to enhance the security of blockchain systems. We have defined this specific field of study as deep learning for blockchain security (DL4BCS). We began with an explanation of blockchain origins, providing an in-depth description of its unique structure. We also properly defined deep learning, elaborating on specific neural networks and their unique properties. By referring to the articles where these technologies have originated, we provided a combined definition of how deep learning and blockchain could be integrated to improve security in models and systems.

We organized current work within our methodology based on neural network architectures, blockchain layers, attack and defense roles, and application domains. Through this comprehensive examination of Graph Neural Networks (GNNs), Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, Generative Adversarial Networks (GANs), Multilayer Perceptrons (MLPs), Recurrent Neural Networks (RNNs), transformers, and hybrid models, we uncovered clear patterns, strengths, gaps, and weaknesses of existing research in DL4BCS. To help understand this methodology, we created various tables and diagrams to allow readers to visualize our analysis.

From our results, we can identify that GNNs, GANs, and LSTMs are the most researched architectures. GNNs are prominent since they serve a special role for modeling relational data in transaction networks and can detect complex fraud patterns. LSTMs and RNNs are similar since they can both be applied for sequential modeling, intrusion detection, and anomaly tracing (specifically for IoT and healthcare-based blockchain platforms). At the same time, CNNs and GANs can be applied to blockchain due to their spatial learning capacity and ability to generate fake data. However, their applications may be restricted to specific layers of a blockchain, such as the data layer and application layer.

Although transformer models have recently started applying to improve blockchain security, they have substantial potential due to their long-range dependencies and ability to complete complex security tasks. However, their computational requirements and lack of pre-trained blockchain models serve as barriers for researchers.

As we have seen, hybrid models merge two or more neural networks, serving as a particularly powerful technological integration with improved performance on benchmarks. These models demonstrate the applied use of multi-model learning in blockchain systems, but this typically comes at the cost of increased data preprocessing efforts.

Despite promising advances in this field of study, our survey identifies a number of challenges and issues. Within existing constraints, data availability has proven to be quite common across all existing DL4BCS research. The decentralized, privacy-preserving nature of blockchain makes it difficult for large labeled datasets to train deep learning models. The absence of standardized criteria also contributes to making performance comparisons difficult across DL4BCS studies. Additionally, the majority of present literature focuses on the application and data layers of blockchain, which results in less focus on consensus, network, and hardware. All layers of blockchain serve important purposes for security, but more research could result in privacy improvements. But, it's difficult to improve privacy if realistic datasets are protected, restricting researchers to using synthetic data to train deep learning modules. As a result, systems are restricted to using synthetic data or temporary solutions (such as federated learning). This also means that privacy-preserving approaches (including identity protection and differential privacy) are less understood when compared to security focuses. Real-time constraints have created several issues as well, introducing inference delays and latency within systems. We have also proven that trade-offs are unfortunately common in this area of study. Improving speed often results in sacrificing accuracy, and vice versa.

With both advances and challenges in mind, we identified

various directions for future research. First, publicly available and standardized DL4BCS datasets should be created to make benchmarking and reproducibility much simpler. By doing this, researchers will have increased opportunities to produce more privacy-based models and systems. This also ensures the protection of researchers in terms of specific laws and regulations that focus on preserving data privacy. We also encourage researchers to provide explainability so that trust can be established within these models. Second, more effort needs to be focused on understudied blockchain layers, such as the consensus layer and the hardware layer of blockchain technology. Although alternative layers, including the application layer, are heavily represented in current DL4BCS research, all blockchain layers can provide substantial improvements to blockchain security, especially when integrated with deep learning. Third, we encourage researchers to consider neural network and blockchain interoperability to unlock the full potential of AI. Accessing neural networks within smart contracts serves as a potential solution to improve systems, not just for security purposes but overall.

In conclusion, the integration of neural networks and blockchain has proven to be much more than applying deep learning to a new data source. It requires changing how deep learning models are trained, used, and evaluated within decentralized ecosystems. The presented solutions are not only in technical innovation but also interlace the use of AI, cryptography, distributed computing, and system engineering. By continuing to push boundaries and refine the combination of these technologies, research communities can unlock new methods and models to improve blockchain security by using deep learning architectures. In the far future, we hope that this field of study will motivate the development of safer, more expandable solutions that can be used across a larger variability of both blockchain and deep learning environments.

8. REFERENCES

[1] A. K. Abasi, M. Aloqaily, M. Guizani, and Z. Alatoom. Enhancing anomaly detection in blockchain transactions with transformer-based models. In *2024 6th International Conference on Blockchain Computing and Applications (BCCA)*, pages 574–579. IEEE, 2024.

[2] H. S. A. Abdel Qadear and A. I. A. El-Shora. Using deep generative networks for data analysis and quality enhancement in blockchain networks. *Commerce and Finance*, 45(1):152–184, 2025.

[3] G. Airlangga. Deep learning for anomaly detection and fraud analysis in blockchain transactions of the open metaverse. *Jurnal Informatika Ekonomi Bisnis*, pages 324–329, 2024.

[4] A. A. Aliyu, J. Liu, and E. Gilliard. A decentralized and self-adaptive intrusion detection approach using continuous learning and blockchain technology. *Journal of Data Science and Intelligent Systems*, 2024.

[5] J. A. Alzubi, O. A. Alzubi, A. Singh, and M. Ramachandran. Cloud-iiot-based electronic health record privacy-preserving by cnn and blockchain-enabled federated learning. *IEEE Transactions on Industrial Informatics*, 19(1):1080–1087, 2022.

[6] N. T. Anthony, M. Shafik, and H. F. Atlam. An effective mlp model for detecting malicious nodes in pos permissionless blockchains. In *MATEC Web of Conferences*, volume 401, page 10003. EDP Sciences, 2024.

[7] E. Asem, L. M. Abouelmagd, A. E. Tolba, and S. Elmougy. Biometric cnn model for verification based on blockchain and hyperparameter optimization. *International Journal of Computational Intelligence Systems*, 17(1):256, 2024.

[8] Z. Batool, K. Zhang, Z. Zhu, S. Aravamuthan, and U. Aivodji. Block-fest: A blockchain-based federated anomaly detection framework with computation offloading using transformers. In *2022 IEEE 1st Global Emerging Technology Blockchain Forum: Blockchain & Beyond (iGETblockchain)*, pages 1–6. IEEE, 2022.

[9] B. R. Bezanjani, S. H. Ghafouri, and R. Gholamrezaei. Privacy-preserving healthcare data in iot: a synergistic approach with deep learning and blockchain. *The Journal of Supercomputing*, 81(4):533, 2025.

[10] V. Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3(37):2–1, 2014.

[11] J. Cai, W. Liang, X. Li, K. Li, Z. Gui, and M. K. Khan. Gtxchain: A secure iot smart blockchain architecture based on graph neural network. *IEEE Internet of Things Journal*, 10(24):21502–21514, 2023.

[12] Z. Chang, Y. Cai, X. F. Liu, Z. Xie, Y. Liu, and Q. Zhan. Anomalous node detection in blockchain networks based on graph neural networks. *Sensors*, 25(1):1, 2024.

[13] K. K. Chanumolu and G. M. Nagamani. An enhanced model for smart healthcare by integrating hybrid ml, lstm, and blockchain. *Ingenierie des Systemes d'Information*, 30(1):43, 2025.

[14] S. Chen, Y. Liu, Q. Zhang, Z. Shao, and Z. Wang. Multi-distance spatial-temporal graph neural network for anomaly detection in blockchain transactions. *Advanced Intelligent Systems*, page 2400898, 2025.

[15] C. Chi, Z. Yin, Y. Liu, and S. Chai. A trusted cloud–edge decision architecture based on blockchain and mlp for aiot. *IEEE Internet of Things Journal*, 11(1):201–216, 2023.

[16] S.-H. Choi and S.-J. Buu. Learning to traverse cryptocurrency transaction graphs based on transformer network for phishing scam detection. *Electronics*, 13(7):1298, 2024.

[17] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[18] K. P. Dirgantoro, J. M. Lee, and D.-S. Kim. Generative adversarial networks based on edge computing with blockchain architecture for security system. In *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 039–042. IEEE, 2020.

[19] V. Elangovan, S. Revathi, N. Jabeen, A. J. Obaid, and R. Kumar. Block chain technology: Anomaly detection in bitcoin using rfmlpalgorithm. In *2024 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*, pages 1–6. IEEE, 2024.

[20] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[21] C. Geren, A. Board, G. G. Dagher, T. Andersen, and J. Zhuang. Blockchain for large language model security and safety: A holistic survey. *ACM SIGKDD explorations newsletter*, 26(2):1–20, 2025.

[22] M. A. N. U. Ghani, K. She, M. A. Rauf, M. Alajmi, Y. Y. Ghadi, and A. Algarni. Securing synthetic faces: A gan-blockchain approach to privacy-enhanced facial recognition. *Journal of King Saud University-Computer and Information Sciences*, 36(4):102036, 2024.

[23] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[24] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2, 2005.

[25] A. Harper and M. D. Lee. Scalable blockchain fraud detection using spatial-temporal graph neural networks. *Frontiers in Applied Physics and Mathematics*, 2(1):1–12, 2025.

[26] M. Hasan, M. S. Rahman, M. J. M. Chowdhury, and I. H. Sarker. Cnn based deep learning modeling with explainability analysis for detecting fraudulent blockchain transactions. *Cyber Security and Applications*, page 100101, 2025.

[27] Z. He, Z. Li, S. Yang, H. Ye, A. Qiao, X. Zhang, X. Luo, and T. Chen. Large language models for blockchain security: A systematic literature review. *arXiv preprint arXiv:2403.14280*, 2024.

[28] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[29] I. Homoliak, S. Venugopalan, D. Reijsbergen, Q. Hum, R. Schumi, and P. Szalachowski. The security reference architecture for blockchains: Toward a standardized model for studying vulnerabilities, threats, and defenses. *IEEE Communications Surveys & Tutorials*, 23(1):341–390, 2020.

[30] S. T. Jeyakumar, A. C. Eugene Yugarajah, Z. Hóu, and V. Muthukkumarasamy. Detecting malicious blockchain transactions using graph neural networks. In *International Symposium on Distributed Ledger Technology*, pages 55–71. Springer, 2023.

[31] C. B. Jones and D. J. Kingsley. Decentralized blockchain with convolutional neural network model for security attack mitigation. *ICTACT Journal on Communication Technology*, 14(1), 2023.

[32] V. K. Kasula, A. R. Yadulla, M. Yenugula, B. Konda, and S. Ayyamgari. Improved blockchain security through gnn-based address identification. *Available at SSRN 5139560*, 2025.

[33] A. Laurent. Graph neural networks for blockchain security: A deep learning approach to anomaly detection. *Frontiers in Interdisciplinary Applied Science*, 2(1):93–105, 2025.

[34] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[35] Y. Lei, Z. Chaoyang, I. Alam, and M. A. Mushtaq. Smart network forensics with generative adversarial networks leveraging blockchain for anomaly detection and immutable audit trails. *Power System Technology Volume 48 Issue 1*, 2023.

[36] T. Liu, Y. Wang, J. Sun, Y. Tian, Y. Huang, T. Xue, P. Li, and Y. Liu. The role of transformer models in advancing blockchain technology: A systematic survey. *arXiv preprint arXiv:2409.02139*, 2024.

[37] Z. Liu and X. Yin. Lstm-cgan: Towards generating low-rate ddos adversarial samples for blockchain-based wireless network detection models. *IEEE Access*, 9:22616–22625, 2021.

[38] E. B. Mbaya, E. Adetiba, J. A. Badejo, J. S. Wejin, O. Oshin, O. Isife, S. C. Thakur, S. Moyo, and E. F. Adebiyi. Secfedidm-v1: A secure federated intrusion detection model with blockchain and deep bidirectional long short-term memory network. *IEEE Access*, 11:116011–116025, 2023.

[39] A. F. Mendi. A sentiment analysis method based on a blockchain-supported long short-term memory deep network. *Sensors*, 22(12):4419, 2022.

[40] S. Mnasri, D. Salah, and H. Idoudi. A hybrid blockchain and federated learning attention-based bert transformer framework for medical records management. *The Journal of Supercomputing*, 81(1):317, 2025.

[41] A.-A. Monirah and M. Ykhlef. Deepblock: a collaborative intrusion detection framework based on blockchain and deep learning. In *2023 Fifth International Conference on Blockchain Computing and Applications (BCCA)*, pages 180–185. IEEE, 2023.

[42] M. MuhsnHasan, K. Priyanka, A. Shahebaaz, M. Devi, and C. Sushama. Securing blockchain based supply chains in agriculture using graph neural networks for anomaly detection. In *2025 International Conference on Intelligent Systems and Computational Networks (ICISCN)*, pages 1–5. IEEE, 2025.

[43] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[44] P. P. Pawar, D. Kumar, B. Ananthan, S. B. Christopher, and R. Surya. An advanced wasserstein-enabled generative adversarial network enabled attack detection for blockchain-assisted intelligent transportation system. In *2024 3rd International Conference on Artificial Intelligence For Internet of Things (AIIoT)*, pages 1–6. IEEE, 2024.

[45] E. Rabieinejad, A. Yazdinejad, R. M. Parizi, and A. Dehghantanha. Generative adversarial networks for cyber threat hunting in ethereum blockchain. *Distributed Ledger Technologies: Research and Practice*, 2(2):1–19, 2023.

[46] H. Ranganatha and A. S. Mustafa. Enhancing fraud detection efficiency in mobile transactions through the integration of bidirectional 3d quasi-recurrent neural network and blockchain technologies. *Expert Systems with Applications*, 260:125179, 2025.

[47] C. Rawlins, S. Jagannathan, and D. Wunsch. Prediction of blockchain transaction fraud using a lightweight

generative adversarial network. In *2023 Fifth International Conference on Blockchain Computing and Applications (BCCA)*, pages 116–121. IEEE, 2023.

[48] D. Ressi, R. Romanello, C. Piazza, and S. Rossi. Ai-enhanced blockchain technology: A review of advancements and opportunities. *Journal of Network and Computer Applications*, 225:103858, 2024.

[49] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

[50] M. M. Salim, V. Shanmuganathan, V. Loia, and J. H. Park. Deep learning enabled secure iot handover authentication for blockchain networks. *Human-centric Computing and Information Sciences*, 11(21):10–19, 2021.

[51] D. Saveetha, G. Maragatham, V. Ponnusamy, and N. Zdravković. An integrated federated machine learning and blockchain framework with optimal miner selection for reliable ddos attack detection. *IEEE Access*, 2024.

[52] G. C. Sekhar and A. Rajendran. A secure framework of blockchain technology using cnn long short-term memory hybrid deep learning model. *Indonesian Journal of Electrical Engineering and Computer Science*, 28(3):1786–1795, 2022.

[53] S. Selvi, R. DH, et al. Securing healthcare data from trojan using blockchain and multilayer perceptron. *Grenze International Journal of Engineering & Technology (GIJET)*, 10, 2024.

[54] M. Seo, J. Kim, M. You, S. Shin, and J. Kim. gshock: A gnn-based fingerprinting system for permissioned blockchain networks over encrypted channels. *IEEE Access*, 2024.

[55] M. Shafay, R. W. Ahmad, K. Salah, I. Yaqoob, R. Jayaraman, and M. Omar. Blockchain for deep learning: review and open challenges. *Cluster Computing*, 26(1):197–221, 2023.

[56] A. Sharma, P. K. Singh, E. Podoplelova, V. Gavrilenko, A. Tselykh, and A. Bozhenyuk. Graph neural network-based anomaly detection in blockchain network. In *International Conference on Computing, Communications, and Cyber-Security*, pages 909–925. Springer, 2022.

[57] J. Shen, J. Zhou, Y. Xie, S. Yu, and Q. Xuan. Identity inference on blockchain using graph neural network. In *Blockchain and Trustworthy Systems: Third International Conference, BlockSys 2021, Guangzhou, China, August 5–6, 2021, Revised Selected Papers 3*, pages 3–17. Springer, 2021.

[58] S. Singh, A. Arora, G. Garg, A. Goyal, and N. Gandhi. Performance analysis of blockchain-based secured distributed deep federated learning for wearable internet of things. *Cluster Computing*, 28(5):1–30, 2025.

[59] D. H. Son, B. D. Manh, T. V. Khoa, N. L. Trung, D. T. Hoang, H. T. Minh, Y. Alem, and L. Q. Minh. Semi-supervised learning for anomaly detection in blockchain-based supply chains. In *2024 23rd International Symposium on Communications and Information Technologies (ISCIT)*, pages 140–145, 2024.

[60] M. H. Tabatabaei, R. Vitenberg, and N. R. Veeraragavan. Understanding blockchain: Definitions, architecture, design, and system comparison. *Computer Science Review*, 50:100575, 2023.

[61] P. Tasca and C. J. Tessone. Taxonomy of blockchain technologies. principles of identification and classification. *arXiv preprint arXiv:1708.04872*, 2017.

[62] Q. Umer, J.-W. Li, M. R. Ashraf, R. N. Bashir, and H. Ghous. Ensemble deep learning-based prediction of fraudulent cryptocurrency transactions. *IEEE Access*, 11:95213–95224, 2023.

[63] O. Ural and K. Yoshigoe. Survey on blockchain-enhanced machine learning. *IEEE Access*, 11:145331–145362, 2023.

[64] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[65] V. Veerasamy, L. P. M. I. Sampath, S. Singh, H. D. Nguyen, and H. B. Gooi. Blockchain-based decentralized frequency control of microgrids using federated learning fractional-order recurrent neural network. *IEEE transactions on smart grid*, 15(1):1089–1102, 2023.

[66] R. Vijay Anand, G. Magesh, I. Alagiri, M. G. Brahmam, B. Balusamy, C. P. Selvan, H. M. Alshahrani, M. Getahun, and B. O. Soufiene. Design of an improved model using federated learning and lstm autoencoders for secure and transparent blockchain network transactions. *Scientific Reports*, 15(1):1–18, 2025.

[67] J. Wang, H. Jin, J. Chen, J. Tan, and K. Zhong. Anomaly detection in internet of medical things with blockchain from the perspective of deep neural network. *Information Sciences*, 617:133–149, 2022.

[68] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *The world wide web conference*, pages 2022–2032, 2019.

[69] Z. Wang, A. Ni, Z. Tian, Z. Wang, and Y. Gong. Research on blockchain abnormal transaction detection technology combining cnn and transformer structure. *Computers and Electrical Engineering*, 116:109194, 2024.

[70] Y. Wu. Enterprise financial sharing and risk identification model combining recurrent neural networks with transformer model supported by blockchain. *Heliyon*, 10(12), 2024.

[71] A. Yazdinejad, H. HaddadPajouh, A. Dehghantanha, R. M. Parizi, G. Srivastava, and M.-Y. Chen. Cryptocurrency malware hunting: A deep recurrent neural network approach. *Applied Soft Computing*, 96:106630, 2020.

[72] J. Yu, H. Xue, B. Liu, Y. Wang, S. Zhu, and M. Ding. Gan-based differential private image privacy protection framework for the internet of multimedia things. *Sensors*, 21(1):58, 2020.

[73] Y. Zhang, Y. Liu, and C.-H. Chen. Survey on blockchain and deep learning. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1989–1994. IEEE, 2020.

[74] W. Zheng, K. Wang, and F.-Y. Wang. Gan-based key secret-sharing scheme in blockchain. *IEEE transactions on cybernetics*, 51(1):393–404, 2020.

[75] F. Zola, J. L. Bruse, X. E. Barrio, M. Galar, and R. O. Urrutia. Generative adversarial networks for bitcoin data augmentation. In *2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, pages 136–143. IEEE, 2020.

Urban Planning in the Age of Agentic AI: Emerging Paradigms and Prospects

Rui Liu[1], Tao Zhe[1], Zhong-Ren Peng[2],
Necati Catbas[3], Xinyue Ye[4], Dongjie Wang[1,†], Yanjie Fu[5,†]

[1]University of Kansas, [2]University of Florida, [3]University of Central Florida,
[4]University of Alabama, [5]Arizona State University

† Corresponding authors: Dongjie Wang and Yanjie Fu.

ABSTRACT

Generative AI, large language models (LLMs), and agentic AI have emerged separately of urban planning. However, the convergence between AI and urban planning presents an interesting opportunity towards AI urban planners. Existing studies conceptualizes urban planning as a generative AI task, where AI synthesizes land-use configurations under geospatial, social, and human-centric constraints and reshape automated urban design. We further identify critical gaps of existing generative urban planning studies: 1) the generative structure has to be predefined with strong assumption: all of adversarial networks, diffusion models, hierarchical zone-POI generative structure are predefined by humans; 2) ignore the power of domain expert developed tools: domain urban planners have developed various tools in the urban planning process guided by urban theory, while existing pure neural networks based generation ignore the power of the tools developed by urban planner practitioners. To address these limitations, we outline a future research direction agentic urban AI planner, calling for a new synthesis of agentic AI and participatory urbanism that integrates AI capabilities with domain expertise and public engagement.

1. INTRODUCTION AND MOTIVATION

Urban planning is an interdisciplinary and complex process that involves public policy, social science, engineering, architecture, landscape, and other related fields. Effective urban planning can help to mitigate the operational and social vulnerability of a urban system, such as high tax, crimes, traffic congestion and accidents, pollution, depression, and anxiety [1–3].

Traditional urban planning processes are mostly completed by professional human planners. Although urban data are increasingly available, the classic planning methods remain static and slow to adapt. However, rapid urbanization, climate change, aging infrastructure, and widening social inequities together create challenges on how cities are designed. Recent breakthroughs in Generative AI (GenAI) [4–7], Large Language Models (LLMs) [8–10], and Agentic AI [11, 12] offer a transformative opportunity to rethink urban planning. These technologies introduce the abilities to generate alternative futures, understand complex textual and visual data, and act as autonomous agents that reason, simulate, and navigate planning goals. The new abilities of these AI paradigms make it possible to develop an AI Urban Planner that can augment human expertise, democratize access to planning insights, and adapt to changing urban conditions. This insight prompts a fundamental question: how might AI reshape the practice and ethics of urban planning in the era of AI? What roles does GenAI, LLMs, and agentic AI play in urban planning? Can machines develop and learn at a human capability to automatically and quickly calculate land-use configurations? In this way, machines can be planning assistants, while human planners can adjust machine-generated plans for specific needs.

2. EXISTING STUDIES: URBAN PLANNING AS GENERATIVE AI

Although urban planning is complex and difficult to automate, a new perspective is to formulate urban planning as a generative task that generates optimal land uses and built environment configurations conditional on geospatial, infrastructural, socioeconomic, and human-centric constraints specific to targeted regions.

We propose to leverage and adapt these GenAI models to produce diverse land-use configurations and generate spatial designs conditioned on multimodal inputs, such as urban geography data, human mobility patterns, and environmental constraints [13]. One example task is to create alternative zoning proposals that meet goals for density, connectivity, or climate resilience, while incorporating learned priors from historical planning outcomes. These capabilities enable urban planners to shift from rule-based prescriptions to data-driven generative design that is both adaptive and customizable [14].

2.1 Recent Studies in GenAI for Urban Planning

Drawing on the advanced capabilities of generative AI, we have reenvisioned urban planning as a generative AI task, conceptualizing spatial configuration plans as images, matrices, or tensors, with a generative model tasked with producing the optimal tensor representation [15] under specific geographic and social constraints. The work in [16] frames urban planning as the generation of land-use configurations over graphs enriched with geographic attributes and contextual information. These configurations are encoded as multi-layered data representations representing Point of Interest (POI) distributions based on location and function. A graph autoencoder generates regional embeddings, which are

fed into a GAN to assess generated plans using metrics such as willingness to pay, diversity of human activities, and social interaction. Despite its novelty, this approach struggles to model spatial hierarchies and exhibits training instability. In [17], land-use generation is conditioned on spatial context and textual planner guidance through a deep conditional variational autoencoder, yielding zone maps and POI layouts informed by contextual factors like geography and mobility. While effective in integrating planning expertise, it does not explicitly address hierarchical urban structures. To overcome these limitations, [15, 18] introduces a hierarchical conditional transformer that first produces zone-level functions before refining them into detailed grid-level plans. Similarly, [19] formulates planning as a two-tier hierarchical reinforcement learning (RL) problem, using an actor-critic network and a Deep Q-Network for decision-making at the zone and POI levels. Likewise, [20] models cities as graphs, utilizing GNN-based RL to make sequential planning decisions.

2.2 The Underlying Generative Urban Planning Framework

Generative urban planning formulates urban planing as conditional generation of land-use configurations, where an AI system learns to synthesize urban layouts that are optimized with respect to spatial, social, and policy-driven objectives. This principled framework consists of two stages: learning representations of urban contexts and needs and generating land-use scenarios under those conditions.

2.2.1 Representation: Modeling Urban Contexts and Human Needs

The representation stage aims to encode the multifaceted conditions of a target urban area to plan into a structured embedding that guides the generative model. This involves integrating and representing diverse information: 1) Geospatial Form: Captures the spatial arrangement and morphology of urban environments, including streets, buildings, parks, and infrastructure. These spatial patterns influence accessibility, land value, and neighborhood character. 2) Human Mobility: Derived from data such as GPS traces, public transit logs, and mobile phone signals, mobility data reveals functional flows and commuting behaviors critical to understanding land-use needs. 3) Social Interactions: Encompasses both physical interactions (e.g., in plazas, schools, offices) and digital engagement (e.g., social media activity). These patterns reflect community structure, cultural behavior, and public sentiment. 4) Planner Requirements: Urban planners often express goals—such as increasing green space or housing density—via textual prompts, zoning rules, or categorical preferences. These must be encoded as flexible or strict constraints in the model. To unify all these modalities, deep representation learning techniques such as autoencoders, convolutional neural networks (CNNs), and attention-based models are employed to transform high-dimensional inputs into semantically-rich, low-dimensional embeddings. These embeddings serve as a comprehensive representation of a region's situational context and planning constraints, forming the condition for land-use generation.

2.2.2 Generation: Producing Optimal Land-Use and Built Environment Configurations

Given a learned representation, the generative stage synthesizes land-use plans using deep generative models, such as, Variational Autoencoders (VAEs) [21], Generative Adversarial Networks (GANs) [22–24], Autoregressive Models [25, 26], and other generative techniques. These generative urban planning models can be trained with a combination of machine learning objectives and planning-specific constraints, ensuring that generated designs remain both technically robust and aligned with urban planning principles. Together, representation and generation form a pipeline where AI systems learn to "understand" the urban environment and "create" planning proposals that are not only technically valid but socially and contextually meaningful.

2.3 Embedding Generative Urban Planning to Real Systems: Flooding Resilience

One application example is a redevelopment zone along the Gulf Coast, where legacy land-use patterns—such as dense residential clustering near flood-prone areas, fragmented green infrastructure, and spatially isolated critical facilities—have amplified vulnerability to storm surge and inland flooding. The central planning challenge is to reconfigure urban layouts to anticipate, absorb, and recover from flood events, while maintaining economic functionality and social continuity. Generative urban planning offers a promising alternative by integrating AI with spatial design to generate land-use scenarios that are both realistic and adaptive to hydrological risks, offering planners new options beyond conventional rule-based methods. Using multimodal data inputs, including flood simulation outputs, infrastructure risk maps, and planner-defined constraints, a conditional generative model can synthesize redevelopment layouts that elevate critical infrastructure, reposition vulnerable housing clusters, introduce green-blue corridors for water absorption, and optimize road networks for emergency access, all while complying with zoning and development guidelines. Such an approach aligns with the emerging "resilience-by-design" paradigm in urban planning [27].

2.4 Limitations of Current AI Urban Planner Research

Despite advances in AI for urban planning, current approaches face critical limitations across five foundational dimensions: theory, methodology, data, computing, and applications. 1) Theory-wise, most AI models reduce planning to spatial optimization or land-use generation, thus, ignore core urban concepts such as zoning, resilience, social equity, and participatory governance, and often ignore the political and normative nature of planning decisions—undermining interpretability and legitimacy. 2) Method-wise, models [15–20, 28] rely on single-step or loosely staged generation conditioned on static inputs that limit creativity and adaptivity, fail to capture multi-granularity dynamics, multiple planning objective trade-offs, and adaptive processes. 3) Data-wise, existing city-specific datasets (e.g., Beijing [29], NYC) lack clear evaluation criteria on scoring a good planning and a bad planning. 4) Computation-wise, GenAI models are resource-intensive and prone to instability, especially in adversarial or variational settings; they lacks explainability and transparency due to the use of neural network based architectures, and often remain opaque to end users. 5) Application-wise, existing generative methods overlook many expert-developed, theory-driven tools that support each stage of the planning process, yet purely neural network–based generation often ignores the value of these established practices.

3. THE VISION: INTEGRATING VLM, MULTIMODAL REASONING, AND AGENTIC AI FOR URBAN PLANNING AI AGENTS

To go beyond GenAI like VAEs, GANs, or diffusion models, we believe that Vision-Language Models (VLMs) integrate visual and textual modalities to produce outputs that are semantically meaningful and instruction-following [30, 31]. This multimodal capacity allows AI systems to integrate maps, imagery, and policy goals to pursue complex planning objectives. For example, urban design prompts (e.g., "generate a transit-oriented mixed-use layout") can be aligned with satellite imagery, GIS data, and street-level visuals. This allows planners to interact with models using natural language, while ensuring that outputs remain spatially accurate and grounded in real-world data. Agentic AI extends this capability by modeling the planning pipeline itself, with abilities for reasoning, task decomposition, and multi-step decision-making that mirror how human planners approach complex planning tasks. Agent-based urban planners can autonomously explore spatial trade-offs, simulate long-term effects, and iteratively refine plans based on feedback, environmental constraints, or guidance from human planners. Together, VLMs and agentic systems enable an AI urban planner that is not only generative but also goal-driven and context-aware. Such AI systems can understand complex instructions, align outputs with multimodal data, and dynamically adapt proposals to evolving objectives.

3.1 Agentic AI Abilities: Implications for Urban Planning

Agentic AI introduces a new paradigm for urban planning by directly addressing its interdisciplinary and complex nature. Through the integration of reasoning, task decomposition, and the use of domain-specific tools, agentic AI has the potential to make planning processes more transparent, explainable, and practical.

3.1.1 Reasoning

Unlike static models that simply generate outputs, agentic AI can systematically analyze diverse objectives, constraints, and potential conflicts. This reasoning ability enables the AI urban planners to balance objectives such as resilience, equity, and cost, while highlighting trade-offs and anticipating downstream consequences—making planning decisions more transparent and better aligned with real-world needs.

3.1.2 Task decomposition

Urban planning is too complex to be solved in a single step. Agentic AI addresses this challenge with a divide-and-conquer approach, decomposing the process into smaller, manageable tasks, each focused on a particular planning dimension such as land use, mobility, or infrastructure. These sub-tasks can follow their own objectives and constraints while remaining consistent with the overall plan, making the decision process more transparent, traceable, and easier for planners to engage.

3.1.3 Domain-specific tool usage

Another strength of agentic AI is its ability to integrate practical tools developed by urban planning practitioners. These tools (e.g., simulation software, zoning analysis methods, resilience metrics) have been validated in real-world projects and regulatory processes, and are grounded in decades of planning theory and professional practice. By leveraging such domain-expert tools, agentic AI can align its outputs with institutional standards and established planning principles rather than relying solely on static, data-driven inference. This makes the results more innovative, credible, and directly applicable to practice. Through the integration of these abilities, agentic AI emerges as a practical paradigm for urban planning. It can handle complex and conflicting objectives in a transparent and traceable way, while aligning outputs with established planning standards and producing results that are credible to experts and accessible to stakeholders.

3.2 The Agentic AI Urban Planner Framework

Building on these capabilities, we propose an agentic AI urban planner framework that embeds AI into the planning workflow to produce clearer, more usable planning outputs and better support decision analysis. Figure 1 demonstrates the overall pipeline of an agentic AI urban planner.

3.2.1 Task Planner Agent

Given the diversity and potential conflicts among planning objectives and constraints, the agentic AI system starts with a Task Planner agent. The task planner agent has three primary functions. First, it builds a comprehensive view of the planning task by integrating diverse requirements and constraints, providing a transparent basis for scenario generation and evaluation. As the inputs span multiple modalities, the Task Planner interprets and integrates them into a coherent representation. This multimodal fusion ensures a consistent and traceable understanding of context. It also makes explicit the relationships among objectives, constraints, and contextual factors, improving transparency. Second, it breaks the workflow into smaller, manageable steps from the divide-and-conquer perspective. Instead of generating a complete plan in a single step, it structures the work into sub-tasks that remain aligned with the overall goals. Each sub-task targets a specific planning dimension with its own objectives and constraints, while remaining consistent with the overarching goals. Third, it defines and delegates sub-tasks by determining which Specialized Agents should take them on and which professional tools should be used. This ensures that each task is paired with right expertise, computational resources, and established planning practices.

3.2.2 Specialized Agents

Once the planning task has been decomposed, the resulting sub-tasks are assigned to Specialized Agents for implementation. Each Specialized Agent is dedicated to a clearly defined scope of work, focusing on one particular dimension of the planning process. Instead of addressing the full complexity of the planning at once, they operate within the narrower boundaries of their designated sub-tasks, generating more accurate and reliable intermediate results. These results enhance the transparency and traceability of the overall process by providing planners with interpretable outputs at each stage. Moreover, Specialized Agents are empowered with domain-specific methods and practitioner-developed tools, enabling them to align outputs with professional expertise and regulatory standards. This division of labor creates a complementary set of agents that lays the foundation for integrating results into coherent planning alternatives.

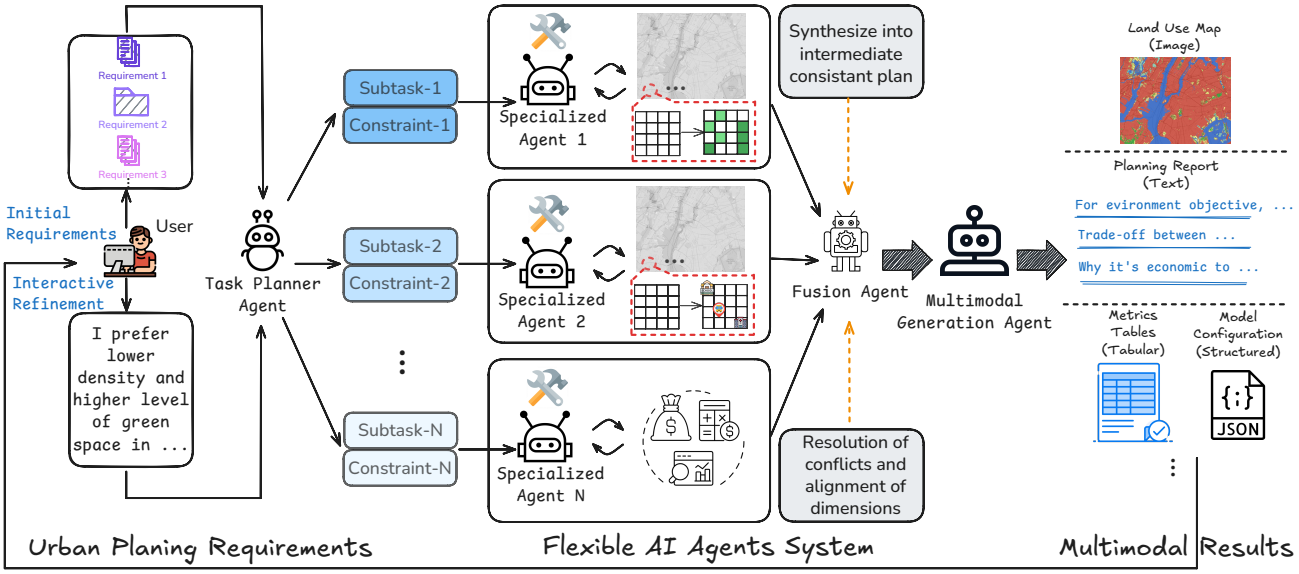


Figure 1: Framework Overview of Agentic AI based Automated Urban Planner.

3.2.3 *Fusion Agent*

After the Specialized Agents have completed their corresponding sub-tasks, their intermediate results are fed forward to the Fusion Agent. The Fusion Agent then aggregates these diverse outputs, resolve potential conflicts, and synthesize them into coherent planning alternatives. In doing so, it ensures internal consistency across spatial, social, environmental, and economic dimensions that would otherwise remain fragmented. Instead of simple aggregation, the Fusion Agent explicitly highlights trade-offs among competing objectives such as cost versus resilience or growth versus equity. By reconciling differences across sub-task outputs, it generates planning candidates that are technically feasible, socially legitimate, and easier for stakeholders to evaluate and compare. This agent is critical because, while Specialized Agents provide accurate results within their narrow scope, those results might be misaligned when considered together. The Fusion Agent addresses this challenge by acting as a system-level integrator, transforming partial results into consistent and actionable planning scenarios.

3.2.4 *Multimodal Generation Agent*

After the Fusion Agent synthesizes coherent planning candidates, the Multimodal Generation Agent eventually transforms then into multimodal outputs that are transparent, interpretable and accessible to different audiences. Instead of limiting results to technical reports or abstract layouts, this agent produces complementary formats such as narrative summaries, tabular statistics, and spatial maps. These outputs make it easier for professional planners to evaluate trade-offs, while also enabling non-specialist stakeholders to understand and engage with the alternatives. In contrast to conventional automated urban planning approaches that often formulate urban planning as a single optimization step, the agentic AI framework reconceptualizes urban planning as a multi-stage, reasoning-driven process, transforming the overall task from a black-box approach into an adaptive paradigm that is transparent, interpretable, and grounded in professional practice. By organizing conflicting objectives into a structured problem set, the AI system provides urban planners with a transparent overview that clarifies priorities and guides subsequent steps. Building on this foundation, specialized analyses across distinct dimensions draw on methods and tools established by domain experts to deliver credible and interpretable intermediate outputs. The synthesis of these intermediate results produces coherent alternatives that make trade-offs explicit and reinforce decision legitimacy, Finally, multimodal outputs ensure that results remain transparent to experts and accessible to broader stakeholders. This structured pipeline aligns closely with the routine workflow of urban planners, reinforcing clarity at the outset, credibility in intermediate analyses, and accessibility in final outputs. It enables planners to engage the system through natural-language prompts combined with diverse data inputs, iterating with the agents to test ideas and refine directions. By reasoning over complex contexts, the AI system can produce detailed textual narratives alongside preliminary spatial sketches, accelerating early-stage concept development while enriching it with greater depth. In doing so, agentic AI complements professional practice with outputs that are not only technically robust but also imaginative and accessible across multiple modalities. To illustrate its practical value, we align the framework with the routine planning workflow, using flood resilience as a representative example to highlight how agentic AI augments planner-led processes at each stage.

3.3 Flood Resilience Planning: A Motivating Example

Flood resilience planning represents a classic challenge in urban planning because it concentrates many of the tensions that planners face in practice. Objectives are inherently conflicting: protecting property owners often requires costly defenses, while ensuring public safety may demand relocation or restrictions on development. All of this must be managed within strict budget limits. The tools and data required for decision-making are highly fragmented. Hydro-

logical simulations, GIS-based vulnerability assessments, and economic cost–benefit models are usually run individually, making the integration difficult and time-consuming. In traditional planning processes, key trade-offs are typically resolved through expert deliberation and then explained in technical documents, making it challenging for the public to understand or engage with. These difficulties raise a pressing question: how can planners manage conflicting objectives, fragmented tools, and opaque decisions in a way that remains both rigorous and participatory? Agentic AI offers one possible answer. By reasoning through trade-offs, decomposing complex tasks, integrating expert methods, and presenting results in multimodal accessible forms, it can complement existing practice. To illustrate this potential, we trace how the agentic AI pipeline aligns with the familiar stages of the planning workflow, acting like a junior planner.

3.4 Integrating Agentic AI into the Planning Workflow

Urban planners typically follow a structured workflow that starts with problem scoping, followed by collaborative visioning, data collection and analysis, alternative generation, evaluation of trade-offs, and finally public deliberation and recommendation. Each stage has distinct objectives, methods, and outputs, and together they form the backbone of day-to-day planning practice. Building on the above agentic AI framework, we now align its functions with these standard planning stages, showing how it can complement existing practice and address long-standing challenges such as fragmented tools, opaque trade-offs, and limited participation.

3.4.1 Problem Scoping

Problem scoping is the first step in the planning workflow, where planners define the scope of the challenge, clarify objectives, and identify key constraints. At this stage, they collect input from diverse stakeholders—residents, government agencies, and developers, while also aligning with policy documents, regulations, and budget limits. The goal is to establish a shared understanding of the priorities the planning process must address, from balancing flood protection with housing development to weighing costs against public safety. In practice, however, this step is often complicated by fragmented information, inconsistent stakeholder priorities, and limited transparency, since much of the discussion remains within expert groups. Agentic AI can support planners at this stage by integrating diverse input materials such as flood maps, zoning rules, demographic data, and stakeholder goals into a single, coherent problem representation. Beyond simply compiling inputs, the Task Planner Agent highlights where objectives reinforce one another, where they conflict, and which constraints are likely to dominate later stages, establishing a structured problem set that identifies the objectives and constraints requiring attention in subsequent stages. For human planners, this initial structured problem map provides a transparent baseline that clarifies priorities and reveals the tensions that must be carried forward, providing a clearer start for decomposing the task into manageable parts and moving into the collaborative visioning process.

3.4.2 Collaborative Visioning

Collaborative visioning follows problem scoping as the stage where planners and stakeholders work together to define common goals and imagine possible futures based on the shared understanding of the overall challenges and constraints. In this step, residents, officials, and developers discuss priorities for the community's future, often drafting vision statements that capture aspirations for safety, livability, and economic growth. In practice, collaborative visioning can be skewed by imbalances in participation, leading to situations in which dominant perspectives prevail and others are marginalized. As a result, disagreements are frequently left unresolved, which limits the transparency of outputs for subsequent stages. Agentic AI addresses this challenge by analyzing inputs from meetings, documents, and surveys to identify both points of consensus and areas of disagreement, turning vague tensions into explicit trade-offs that enhance transparency and broaden opportunities for engagement. In addition, the Task Planner Agent decomposes broad and often abstract aspirations into more concrete components. For instance, it can separate goals related to safety, livability, and economic growth into distinct components, each of which can be carried forward to specialized analyses. This decomposition not only clarifies priorities at the visioning stage but also ensures that subsequent evaluations remain transparent and traceable across specific dimensions. To support discussion, the Task Planner Agent can also organizes the decomposed goals into a small set of contrasting vision frames (e.g., safety-led, livability-led, growth-led), each with explicit assumptions and indicative targets, providing concrete reference points for dialogue without relying on abstract statements. For planners, this yields a transparent baseline for visioning—one that highlights common ground, makes conflicts visible, and supplies structured materials (e.g., goal matrices and trade-off tables) to guide dialogue and participation.

3.4.3 Data Intake and Analysis

Following problem scoping and visioning, the planning process proceeds to data collection and analysis, where diverse technical tools are applied to build the evidence base for decision-making. Hydrological simulations estimate flood risks, GIS overlays map vulnerable areas, economic models project costs and benefits, and demographic forecasts anticipate population trends. These analyses provide the technical foundation for developing alternatives. However, in practice they are often fragmented, as tools are applied independently, produce results in disparate formats, and lack mechanisms for integration across domains. This fragmentation not only increases the workload for planners but also limits the transparency of how technical evidence is assembled and interpreted. Agentic AI addresses these challenges through Specialized Agents, each dedicated to a specific dimension of the analysis. A hydrology agent, for instance, can run flood models and standardize the results into formats that connect with infrastructure or economic assessments. Similarly, other agents apply GIS methods, economic models, or demographic analyses, while ensuring that their outputs are interpretable and traceable. Since these agents are designed to operate within defined scopes, they can deliver more accurate and consistent intermediate results, reducing duplication and making integration more straightforward. For planners, this stage provides domain-specific results that are accurate, consistent, and traceable. Each output can be examined on its own, with assumptions and methods clearly documented. In this way, planners and stakeholders can verify the validity of each component, ensuring confidence in the evidence base before alternatives are developed.

3.4.4 Alternative Generation

After the data have been analyzed across different domains, the next stage is to generate planning alternatives. In traditional processes, options are assembled from separate technical studies, which often yields inconsistencies across spatial, social, environmental, and economic dimensions. Conflicts among objectives are not systematically articulated, and results lack a common structure for comparison, making rigorous evaluation difficult. Agentic AI strengthen this stage through the Fusion Agent, which synthesizes the standardized outputs from specialized analyses and synthesizes them into coherent planning candidates. By aligning assumptions and scales, the Fusion Agent creates a consistent basis for comparison across domains. It then evaluates each candidate against the stated constraints and makes trade-offs explicit, ensuring that planners can clearly assess the implications of different alternatives. For planners, the outcome is a not a disconnected set of reports but a structured set of comparable alternatives. Each option is presented with its assumptions, constraints, and quantified impacts, while also clarifying trade-offs among objectives. This combination provides a transparent basis for systematic evaluation and supports discussion not only among technical experts and decision-makers but also with the broader public.

3.5 Evaluation & Deliberation

The final stage of the planning workflow is evaluation and deliberation, where proposed alternatives are assessed and discussed by both experts and the public. Traditionally, findings are communicated through lengthy reports and technical documents. While these formats provide details, they are difficult for non-specialist audiences to interpret, limiting transparency and weakening opportunities for broad participation. Agentic AI addresses this challenge through the Multimodal Generation Agent. Instead of relying on text-heavy documents, it translates analytical results into a range of outputs, including maps, tables, and narrative summaries. These multimodal representations allow experts to examine precise data while enabling broader stakeholders to engage with intuitive visualizations and plain-language descriptions. For planners, this stage produces a set of planning deliverables that present each alternative through multiple complementary perspectives. Within these deliverables, trade-offs and assumptions are articulated systematically across alternatives, providing a clear and defensible basis for comparison. The results are presented in varied formats, which allow experts to examine detailed evidence and enable policymakers and community members to engage with accessible representations. As a result, planners are able to evaluate alternatives in a process that is more transparent, inclusive, and directly connected to decision-making. This demonstrates that agentic AI can be seamlessly integrated into the standard planning process of urban planners, aligning with each stage of practice. Like a junior planner, it contributes constructively throughout the workflow by clarifying objectives, structuring problems into manageable components, providing consistent cross-domain analyses, assembling alternatives into coherent options, and presenting results in accessible forms that support deliberation. Through these capabilities, agentic AI system enables planners to navigate complexity more clearly, evaluate trade-offs with better evidence, and communicate outcomes in ways that support broader and more inclusive participation.

4. WHAT TO CONSIDER IN LEARNING URBAN PLANNING AI AGENTS

4.1 Detecting and Prioritizing Real Human Needs Before Urban Planning

Urban planning must be grounded by real human needs including:

- Essential needs: safe and affordable housing, access to clean water, sanitation, healthcare, and food;
- Functional needs: mobility infrastructure, public transport, education, and employment access;
- Resilience needs: climate adaptation (e.g., heat mitigation, flood protection), risk reduction, and emergency response capacity;
- Aspirational needs: access to green space, social connection, cultural expression, and digital inclusion.

Our perspective is to leverage AI to sense and infer urban planning needs from both explicit signals (e.g., surveys, civic platforms) and implicit behaviors (e.g., movement traces, online discourse, built environment scans). For example, by mining public discourse from social media, 311 complaints, and community forums, we can exploit NLP and LLMs to extract topics (e.g., transit dissatisfaction, service breakdowns) and sentiments that reflect resident pain points, detect urgency, importance, locations in user-generated contents, mapping emotional intensity to geographic locations. As another example, by analyzing street imagery, drone footage, or satellite data, we can detect missing sidewalks, poor lighting, slum expansion, or playground deficits, and classify urban elements relevant to walkability, safety, or accessibility. The third example is: by modeling GPS traces, call detail records, or smartcard data, AI can identify mismatches between population flows and service coverage, such as last-mile gaps, transit deserts, or areas with extreme commuting burdens.
Besides, Agentic AI can be leveraged to simulate diverse community behaviors under planning scenarios and infer emergent needs by observing agent outcomes (e.g., service access failures, congestion points, or repeated risk exposure).
Finally, geospatial foundation models, pretrained on large-scale maps, satellite images, and terrain data, can infer land-use, infrastructure density, and vulnerability, providing transferrable knowledge to tackle the issues of under-mapped or data-scarce regions.

4.2 Differentiate GenAI in Strategic Macro Planning and Scenario Micro Planning

Generative urban planning should consider the distinction between generic global planning and scenario-specific local planning. Firstly, strategic planning, also known as, macro-scale land-use planning, focuses on long-term, citywide or regional spatial configurations, aiming to optimize zoning, connectivity, and land allocation over 10–30 year horizons. Secondly, scenario planning, often referred to as tactical urban design or micro-planning, targets specific neighborhoods or blocks, adapting to contextual challenges such as flooding, extreme heat, or post-disaster recovery.
Strategic planning aims to capture self-organized planning objectives, such as diversity over mixed land uses, connectivity of urban functions, accessibility to transportation, healthcare,

utilities, job opportunities, and simulation to verify the optimal planning objectives over long-term dynamics. On the other hand, scenario planning aims to prioritize conditional generation under fine-grained constraints. GenAI models, such as hierarchical VAEs, conditional diffusion models, and scenario-based reinforcement fine-tuning can empower guided generation of urban planning.

4.3 Human-Machine Collaborative Planning

Urban CoDesign (i.e., human-machine collaborative planning) enabled by conversational generative AI is a paradigm where urban planners and AI systems co-design planning solutions through natural language and interactive feedback. CoDesign highlights that AI is a responsive partner that generates, explains, and adapts plans based on iterative human input, instead of generating fixed outputs. This is important because urban planning involves complex trade-offs, contextual sensitivity, and normative goals that pure data-driven models often overlook. Feedback from human planners can ground generative planning in policy, social, economic, cultural dimensions.

Possible solutions to achieve the codesign are: integration of multimodal generative planners, instruction-tuned vision language models, and interactive machine learning that respond to planners' queries, critiques, and constraints human-machine conversations. Imagine a planner might say, "Propose a higher-density zoning plan that avoids flood zones and preserves public parks," and the system would iteratively refine the layout while visualizing trade-offs.

4.4 Leveraging Digital Twins For A Close Simulation, Measurement, Planning Loop

Another direction is to leverage digital twins to develop a simulation–measurement–generation loop of urban planning. Traditional urban planning suffers from long feedback cycle, often taking years to observe the social and economic impacts of spatial interventions.

As high-fidelity, real-time virtual replicas of urban environments, digital twins can simulate how spatial configuration plannings affect economic, mobility, and social dynamics. These simulations generate synthetic yet actionable data that feed into measurement models to quantify key urban indicators (e.g., sustainability, accessibility, vibrancy, safety, resilience, perceived well-being). These urban indicators then serve as feedback signals for learning neural urban planning policy networks that guide generative planning toward optimal configurations under evolving urban goals. Such design not only accelerates the evaluation of planning alternatives, but also supports adaptive planning where planning decisions are informed by iterative simulation and measurable impact assessment within digital twins.

4.5 Tools and Infrastructures for Agentic Urban-Planning AI

Recent works on generative urban-planning systems typically rely on conventional deep-learning toolkits and moderate computational setups. For example, in [15], the LUCGAN+ system integrates multiple GAN variants with a learned scoring model and dedicated visualization components, with all experiments conducted on CPU-only hardware. Similarly, in [16], LUCGAN and several baselines (VAE, AVG, MAX) are implemented on a CPU-only workstation (Intel i9-9920X, 128GB RAM, Ubuntu 18.04), indicating that early land-use generation methods require only modest computational resources. In [17], the CLUVAE framework is trained using TensorFlow 2.0 on a single-node machine equipped with one Titan RTX GPU and 128GB RAM, illustrating that more advanced conditional generative models also remain feasible within a single-node environment. These studies show that generative urban-planning systems can be trained and deployed using broadly accessible hardware and standard deep-learning frameworks.

More recent work has incorporated reinforcement learning. The IHPlanner system in [18] combines conditional GANs, CVAEs, attention mechanisms, and multiple ablation variants. The hierarchical RL planner in [19] is implemented using PyTorch and gym and trained on a lightweight setup (Intel i7-12700F, 32GB RAM, RTX 3060 GPU, Windows 10). These studies further demonstrate that most existing approaches rely on standard Python deep-learning ecosystems (TensorFlow or PyTorch) and conventional single-GPU hardware, making them broadly accessible to researchers without large-scale compute resources.

A future generation of agentic AI systems for urban planning is likely to require more integrated tools than those used in current generative or RL-based models. Unlike single-model workflows, agentic systems coordinate multi-step reasoning, interact with geospatial toolchains, and evaluate candidate plans through simulation or rule-based checks. This suggests that future platforms may combine deep-learning frameworks with GIS libraries (e.g., GeoPandas, Shapely), lightweight urban simulation modules, and basic agent coordination components. In terms of computation, such systems are expected to rely on a hybrid architecture in which a single GPU handles model inference while CPUs support geospatial processing and agent coordination, providing a practical path toward agentic planning systems without large-scale compute infrastructure. On typical single-GPU setups with 12–24 GB of available memory, moderate-size language models in the 7B to 13B range are generally sufficient for most agentic workflows, while larger or long-context models can be incorporated when additional compute is available or when planning tasks require deeper reasoning.

5. REFERENCES

[1] D. Adams, *Urban planning and the development process*. Psychology Press, 1994.

[2] E. G. Goetz, R. A. Williams, and A. Damiano, "Whiteness and urban planning," *Journal of the American Planning Association*, vol. 86, no. 2, pp. 142–156, 2020.

[3] O. Yiftachel, "Towards a new typology of urban planning theories," *Environment and Planning B: Planning and Design*, vol. 16, no. 1, pp. 23–39, 1989.

[4] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[5] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[6] A. Oussidi and A. Elhassouny, "Deep generative models: Survey," in *2018 International conference on intelligent*

systems and computer vision (ISCV). IEEE, 2018, pp. 1–8.

[7] A. Noyman and K. Larson, "A deep image of the city: Generative urban-design visualization," *Challenge*, vol. 7, p. 30, 2020.

[8] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[9] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[10] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[11] Z. Durante, Q. Huang, N. Wake, R. Gong, J. S. Park, B. Sarkar, R. Taori, Y. Noda, D. Terzopoulos, Y. Choi *et al.*, "Agent ai: Surveying the horizons of multimodal interaction," *arXiv preprint arXiv:2401.03568*, 2024.

[12] Y. Shavit, S. Agarwal, M. Brundage, S. Adler, C. O'Keefe, R. Campbell, T. Lee, P. Mishkin, T. Eloundou, A. Hickey *et al.*, "Practices for governing agentic ai systems," *Research Paper, OpenAI*, 2023.

[13] D. Wang, C. Lu, and Y. Fu, "Towards automated urban planning: When generative and chatgpt-like AI meets urban planning," *CoRR*, vol. abs/2304.03892, 2023.

[14] N. J. Yuan, Y. Zheng, X. Xie, Y. Wang, K. Zheng, and H. Xiong, "Discovering urban functional zones using latent activity trajectories," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 712–725, 2014.

[15] D. Wang, Y. Fu, K. Liu, F. Chen, P. Wang, and C.-T. Lu, "Automated urban planning for reimagining city configuration via adversarial learning: quantification, generation, and evaluation," *ACM Transactions on Spatial Algorithms and Systems*, vol. 9, no. 1, pp. 1–24, 2023.

[16] D. Wang, Y. Fu, P. Wang, B. Huang, and C.-T. Lu, "Reimagining city configuration: Automated urban planning via adversarial learning," in *Proceedings of the 28th international conference on advances in geographic information systems*, 2020, pp. 497–506.

[17] D. Wang, K. Liu, P. Johnson, L. Sun, B. Du, and Y. Fu, "Deep human-guided conditional variational generative modeling for automated urban planning," in *2021 IEEE international conference on data mining (ICDM)*. IEEE, 2021, pp. 679–688.

[18] D. Wang, L. Wu, D. Zhang, J. Zhou, L. Sun, and Y. Fu, "Human-instructed deep hierarchical generative learning for automated urban planning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, 2023, pp. 4660–4667.

[19] P. Wang, D. Wang, K. Liu, D. Wang, Y. Zhou, L. Sun, and Y. Fu, "Hierarchical reinforced urban planning: Jointly steering region and block configurations," in *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*. SIAM, 2023, pp. 343–351.

[20] Y. Zheng, Y. Lin, L. Zhao, T. Wu, D. Jin, and Y. Li, "Spatial planning of urban communities via deep reinforcement learning," *Nature Computational Science*, vol. 3, no. 9, pp. 748–762, 2023.

[21] D. P. Kingma, M. Welling *et al.*, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[22] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial networks." *Commun. Acm*, vol. 63, no. 11, pp. 139–144, 2020.

[23] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 5769–5779.

[24] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.

[25] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," vol. 30, 2017.

[27] S. Meerow, J. P. Newell, and M. Stults, "Defining urban resilience: A review," *Landscape and urban planning*, vol. 147, pp. 38–49, 2016.

[28] X. Hu, W. Fan, D. Wang, P. Wang, Y. Li, and Y. Fu, "Dual-stage flows-based generative modeling for traceable urban planning," in *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*, pp. 370–378.

[29] P. Wang, K. Liu, D. Wang, and Y. Fu, "Measuring urban vibrancy of residential communities using big crowdsourced geotagged data," *Frontiers in Big Data*, vol. Volume 4 - 2021, 2021.

[30] J. Lu, D. Batra, D. Parikh, and S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," *Advances in neural information processing systems*, vol. 32, 2019.

[31] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," *Advances in neural information processing systems*, vol. 36, pp. 34 892–34 916, 2023.

LTSM-Bundle: A Toolbox and Benchmark on Large Language Models for Time Series Forecasting

Yu-Neng Chuang† Rice University
Songchen Li† Rice University
Jiayi Yuan† Rice University
Guanchu Wang† University of North Carolina at Charlotte
Kwei-Herng Lai† Rice University
Songyuan Sui Rice University
Leisheng Yu Rice University
Sirui Ding University of California, San Francisco
Chia-Yuan Chang Texas A&M University
Alfredo Costilla Reyes Rice University
Daochen Zha Rice University
Xia Hu Rice University

ABSTRACT

Time Series Forecasting (TSF) has long been a challenge in time series analysis. Inspired by the success of Large Language Models (LLMs), researchers are now developing Large Time Series Models (LTSMs), universal transformer-based models that use autoregressive prediction to improve TSF. However, training LTSMs on heterogeneous time series data poses unique challenges, including diverse frequencies, dimensions, scalability, and patterns across datasets. Recent efforts have studied and evaluated various design choices aimed at enhancing LTSM training and generalization capabilities. Despite progress in both paradigms, there is no unified framework for systematically evaluating models and design choices across them. However, these design choices are typically studied and evaluated in isolation and are not compared collectively. In this work, we introduce LTSM-Bundle, a comprehensive toolbox and benchmark for training LTSMs, spanning pre-processing techniques, model configurations, and dataset configurations. We modularize and benchmark LTSMs across multiple dimensions, including prompting strategies, tokenization approaches, training paradigms, base model selection, data quantity, and dataset diversity. By combining the most effective design choices, the combination achieves state-of-the-art zero-shot and few-shot performance while providing a reproducible foundation for evaluating both traditional LSF models and emerging LTSMs. Our source code is available at `https://github.com/datamllab/ltsm`

1. INTRODUCTION

Time series forecasting (TSF) is a long-standing task in time series analysis, aiming to predict future values based on historical data points. Over the decades, TSF has transitioned from traditional statistical methods [2] to machine learning [15], and more recently, to deep learning approaches [11, 27]. In particular, transformers [39], which are often regarded the most powerful architecture for sequential modeling, have demonstrated superior performance in TSF, especially for long-term forecasting [44, 47, 28, 42, 21]. Moving forward, inspired by the remarkable capabilities of Large Language Models (LLMs), many researchers have begun to explore Large Time Series Models (LTSMs) as the natural next phase, seeking to train universal transformer-based models for TSF [41, 16, 10, 35, 8, 17, 6, 49, 20].

Unlike textual data, where tokens typically hold semantic meanings transferable across documents, time series data exhibits high heterogeneity, presenting unique challenges for LTSM training. Across different datasets, time series often have diverse frequencies (such as hourly and daily), dimensions (in terms of varying numbers of variables) and patterns (where, for example, traffic time series may differ significantly from electricity data). This diversity not only poses difficulties in training an LTSM to fit all the datasets but also impedes the model's generalization to unseen data.

Recent endeavors have proposed various innovative designs to enhance the training and generalization capability of LTSMs. To name a few, (i) in terms of pre-processing, prompting strategies have been proposed to generate dataset-specific prompts [20], while various tokenization strategies have been studied for converting time series into tokens to be inputted into transformer layers [49, 1]; (ii) for the model configurations, prior research has involved reusing weights from pre-trained language models and adapting them to downstream tasks [49]; (iii) regarding dataset configurations, different datasets have been utilized for training purposes [16, 49, 6]. Despite these advances, evaluating models across both paradigms remains challenging. Existing studies often adopt inconsistent pre-processing pipelines, heterogeneous tokenization strategies, and non-uniform evaluation metrics, making it difficult to draw fair comparisons or identify best practices, where these designs are typically studied and evaluated in isolation. There is no existing package that integrates these components or benchmarks them collectively. This makes it difficult to understand, select, and combine

†These authors contributed equally: Yu-Neng Chuang (yc146@rice.edu), Songchen Li (sl237@rice.edu), Jiayi Yuan (jy101@rice.edu), Guanchu Wang (gwang16@charlotte.edu), Kwei-Herng Lai (khlai@rice.edu).

these components to effectively train LTSMs in practice.

To address the gaps, we introduce LTSM-Bundle, a comprehensive toolbox and benchmark for training LLMs for time series forecasting, spanning pre-processing techniques, model configurations, and dataset configurations, as depicted in Figure 1. The goal of LTSM-Bundle is designed to serve as a foundation for benchmarking both current and emerging LTSMs and TSFMs, while maintaining full transparency about its current model and dataset coverage. We modularize and benchmark LTSMs under the same settings and from multiple dimensions, including prompting strategies, tokenization approaches, training paradigms, base model selection, data quantity, and dataset diversity. In addition to existing components, we introduce *time series prompt*, a statistical prompting strategy tailored for time series data, as one of our benchmarking components. It generates prompts by extracting global features from the training dataset, providing robust statistical descriptions for heterogeneous data. Through extensive benchmarking with LTSM-Bundle, we combine the most effective design choices identified in our study for training LTSMs. Our empirical results suggest that the identified combination produces superior zero-shot and few-shot (with 5% training data) performances compared to the state-of-the-art LTSMs in benchmark data sets. Additionally, even with just 5% of the data, our result is comparable to the strong baselines trained on the full training data, demonstrating the practical value of LTSM-Bundle in developing and training LTSMs. In summary, we have made the following contributions:

- We present LTSM-Bundle, a comprehensive toolbox and benchmark for LTSMs. LTSM-Bundle not only includes various components with easy-to-use interfaces for training LTSMs but is also integrated with TDengine, a state-of-the-art time series database, to build up an end-to-end training pipeline from time series data storage to report visualization and generation.
- We perform systematic analysis with LTSM-Bundle. Our analysis yields numerous insightful observations, paving the path for future research endeavors.
- Tokenization design critically affects cross-architecture transferability. Our results show that different tokenization strategies have a significant impact on forecasting performance across diverse model architectures.
- Model size is not always correlated with accuracy. We find that smaller and medium-sized models can achieve competitive or even superior performance compared to larger models, particularly on long-horizon forecasting tasks.

2. NOTATIONS AND PROBLEM FORMULATION

We denote a multivariate time series as $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_T\}$, where $\mathbf{z}_t \in \mathbb{R}^d$ is a d-dimensional vector and T is the total number of timestamps. We partition $\mathbf{Z}$ chronologically into training, validation, and testing sets: $\mathbf{Z}^{\text{train}} = \{\mathbf{z}_1, \ldots, \mathbf{z}_{T^{\text{train}}}\}$, $\mathbf{Z}^{\text{val}} = \{\mathbf{z}_{T^{\text{train}}+1}, \ldots, \mathbf{z}_{T^{\text{train}}+T^{\text{val}}}\}$, and $\mathbf{Z}^{\text{test}} = \{\mathbf{z}_{T^{\text{train}}+T^{\text{val}}+1}, \ldots, \mathbf{z}_T\}$, where T^{train} and T^{val} denote the numbers of timestamps for training and validation, respectively. In traditional TSF, we train a model on $\mathbf{Z}^{\text{train}}$ such that, on $\mathbf{Z}^{\text{test}}$, given the past-P observations $\mathbf{X} = \{\mathbf{z}_{t_1}, \ldots, \mathbf{z}_{t_P}\}$, it predicts the next Q values $\mathbf{Y} = \{\mathbf{z}_{t_{P+1}}, \ldots, \mathbf{z}_{t_{P+Q}}\}$, where $\mathbf{X}$ and $\mathbf{Y}$ are subsequences of $\mathbf{Z}^{\text{test}}$. We minimize the mean squared error $\mathcal{L}(\text{LTSM}(\mathbf{X}), \mathbf{Y})$.

We focus on training LTSMs, where the objective is to develop a model that performs well across various test sets, denoted as $\mathcal{Z}^{\text{test}} = \{\mathbf{Z}_1^{\text{test}}, \mathbf{Z}_2^{\text{test}}, ..., \mathbf{Z}_N^{\text{test}}\}$, where N represents the number of datasets for testing. Each $\mathbf{Z}_i^{\text{test}}$ may originate from a distinct domain, with different lengths, dimensions, and frequencies. The training sets for an LTSM may comprise training data associated with $\mathcal{Z}^{\text{test}}$ or data from other sources, provided they are not included in $\mathcal{Z}^{\text{test}}$. Training LTSMs presents a notable challenge compared to traditional TSF models due to the inherent difficulty in accommodating diverse patterns across datasets, often necessitating specialized designs. Nevertheless, it also offers opportunities to transfer knowledge from existing time series to new scenarios.

3. LTSM-BUNDLE LIBRARY

3.1 Package Design

We showcase the system overview of LTSM-Bundle in Figure 2. LTSM-Bundle is a modular toolkit that supports the complete lifecycle of large timeseries models (LTSMs), from raw data ingestion to deploymentready evaluation. The framework is organized around four interoperable subsystems, all exposed through a unified API that eliminates boilerplate engineering and accelerates experimentation. First, **TS Tokenizing** converts multivariate time series into token sequences via linear and dynamic schemes that preserve both global trends and local temporal dynamics, making the data directly consumable by Transformerstyle backbones. Second, **TS Prompting** embeds task instructions and statistical contextthrough hard, soft, and statisticsaware prompts into the token stream, enabling zero and fewshot adaptation to forecasting, anomaly detection, and classification tasks. Third, the **Data Processing** layer supplies scalable loaders, windowing utilities, and featureengineering pipelines that abstract away dataset idiosyncrasies and seamlessly handle large benchmarks. Fourth, **LTSM Training** offers a uniform optimization interface for finetuning or training from scratch a range of backbone architectures and parameter scales, with builtin support for curriculum learning, transfer learning, and largescale hyperparameter sweeps. These subsystems are orchestrated by a reproducible workflow engine that chains tokenizing, prompting, processing, and training steps into endtoend pipelines. The toolkit further provides a library of loss functions, dataaugmentation routines, and evaluation metrics, alongside visualization and reporting utilities that generate publicationready artifacts. Collectively, LTSM-Bundle delivers a comprehensive and extensible platform for constructing, analyzing, and deploying large timeseries models, thereby lowering the barrier to reproducible research and accelerating industrial adoption.

3.2 Package Interface

The LTSM-Bundle package is implemented with a scikit-learn-like interface to easily train a customized LTSM. To increase the flexibility of LTSM-Bundle in utilizing a broader spectrum of backbone models and training paradigms, we have integrated it with the Huggingface Transformers package[1]. This integration allows for the incorporation of diverse pre-trained weights and supports various training ap-

[1] https://github.com/huggingface/transformers

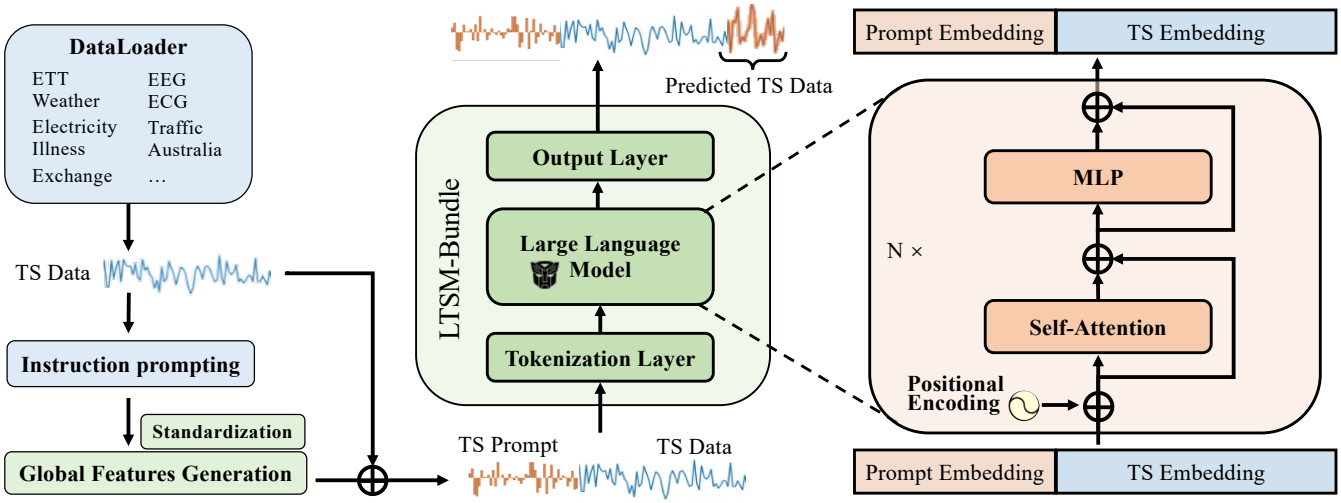


Figure 1: An overview of the key design choices supported by the LTSM-Bundle framework. The library is a general-purpose benchmarking toolbox that standardizes data preprocessing, tokenization strategies, training paradigms, and evaluation pipelines across traditional LSF models, LLM-based models, and pre-trained LTSMs, rather than being limited to LLM-centric designs.

proaches, enhancing the overall adaptability of our framework. Researchers and practitioners only need to provide their own time series data and specify the chosen training configurations; then, every training pipeline can be created using the LTSM-Bundle package. Moreover, LTSM-Bundle supports linking with time series vector database, TDengine[2], a state-of-the-art time series database, to build an end-to-end training pipeline from time series data storage to report visualization and generation. Without struggling to build up extra efforts on clear report generation and visualization, LTSM-Bundle automatically generates all results with an easy-to-read template at the top of TDengine.

4. BENCHMARKING LTSM TRAINING

We benchmark existing LTSM components by involving and coordinating them into four fundamental components of LTSM-Bundle package. We aim to answer the following research questions: **1) How do tokenization methods and prompting techniques impact model convergence? 2) How do base model selection and the training paradigms impact the time series forecasting performance?** and **3) How do different dataset configurations impact the model generalization?**

To answer the questions, we evaluate the impact of each component individually by keeping the rest of the components fixed. First, we fix the base model and dataset configurations with smaller model sizes, limited data quantities, and reduced data diversity, excluding prompt tokenization, to identify the optimal prompting strategy. Next, we incorporate the best prompting strategy with the same base model selection and dataset configuration to assess the tokenization methods. Afterward, we keep all the components constant except the base model to study the impact of different model initialization and training strategies. Finally, with the best tokenization and prompting methods, we select a list of candidate base models following the guidelines from the previous step. We also control the quantity and diversity of the training data to assess their impacts on model generalizability and prediction performances.

We follow the experimental settings outlined in Timesnet [43] and Time-LLM [20], employing the unified evaluation framework. The input time series length $\mathcal{E}$ is set to 336, with four different prediction lengths in {96, 192, 336, 720}. Evaluation metrics include mean square error (MSE) and mean absolute error (MAE). We also calculate the average scores among all prediction horizons. The results highlighted in **red** represent the **best** performance and highlighted in **blue** represent the **second best** performance. The details of hyperparameter settings of our banchmarking experiments are in Appendix B, respectively.

Our evaluations use widely adopted benchmarks: the ETT series (ETTh1, ETTh2, ETTm1, ETTm2) [47], Traffic, Electricity, Weather, and Exchange-Rate datasets [23, 43]. ETT comprises four subsetstwo with hourly data (ETTh) and two with 15-minute data (ETTm)each containing seven features from July 2016 to July 2018. The Traffic dataset provides hourly road occupancy rates from San Francisco freeways (20152016); the Electricity dataset records hourly consumption for 321 clients (20122014); the Weather dataset offers 21 indicators every 10 minutes in Germany during 2020; and the Exchange-Rate dataset includes daily rates for eight countries (19902016). For more details, refer to Appendix A.

We first train our framework on the diverse time series data collection using LTSM-Bundle package and then assess the best combination identified on joint learning and zero-shot transfer learning to different domains of time series knowledge. For clarity, we use the term "LTSM-Bundle" to represent the best practice of the combination under the described experiment settings in each of the following sections.

4.1 Pre-processing: Instruction Prompts

The pre-processing step plays a crucial role in enabling LLM-based models to better adapt to time series datasets. In this section, we present a detailed analysis aimed at recommend-

[2] https://github.com/taosdata/TDengine

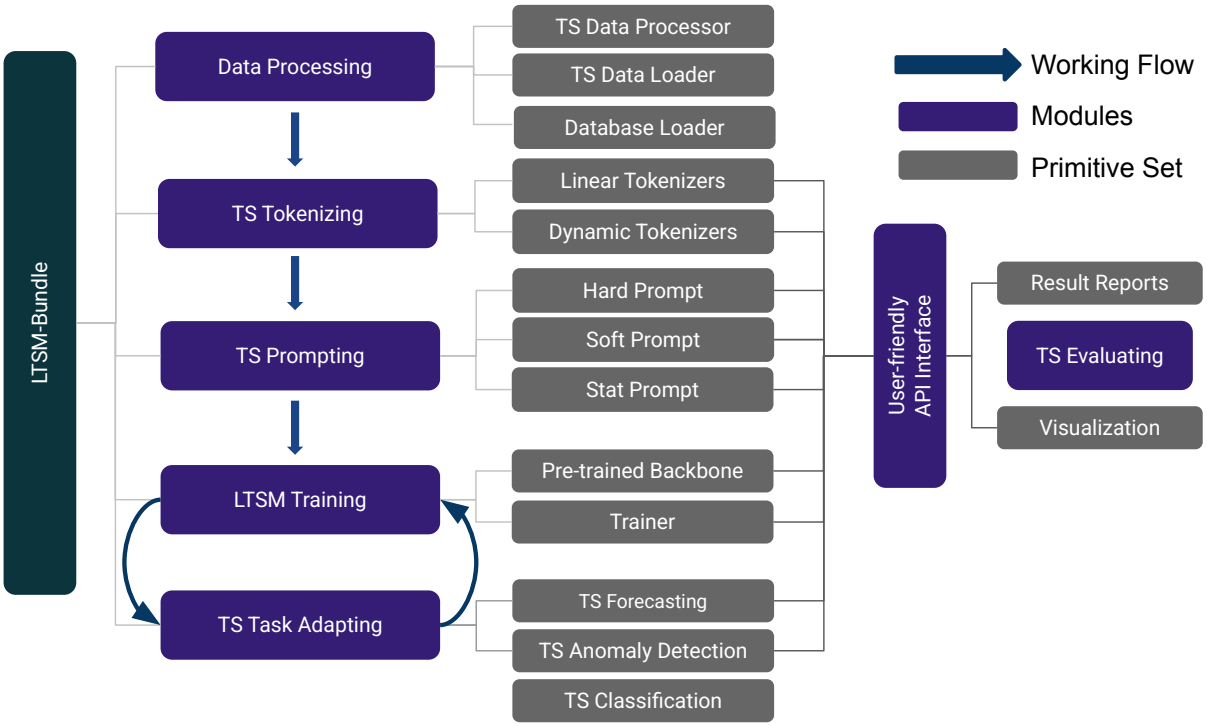


Figure 2: System Overview of LTSM-Bundle library. LTSM-Bundle provides an end-to-end training and evaluation pipeline constructed from data preprocessing to visualization. It also provides a database reader to better incorporate with large-scale datasets.

ing the most effective pre-processing prompting strategy to compose LTSM-bundle. Instruction prompts enhance the effectiveness of LTSM training by providing auxiliary information. This prompt helps the model adjust its internal state and focus more on relevant features in different domains of the dataset, thereby improving learning accuracy. With the aid of prompts, LTSM aims to optimize forecasting ability across diverse dataset domains. We explore two types of prompts: the *Text Prompts* [20] written in task-specific information, and the *time series prompts* developed by global features of time series data. This comparison determines the most effective prompt type for LTSM training.

Time Series Prompts Time series prompts aim to capture the comprehensive properties of time series data. Unlike instruct prompts, they are derived from a diverse set of global features extracted from the entire training dataset. This approach ensures a robust representation of the underlying dynamics, in addition to enhancing model performance. The time series prompts are generated by extracting global features from each variate of the time series training data. The extracted global features are specified in Appendix D. After extracting the global features, we proceed to standardize their values across all varieties and instances within the dataset. This standardization is crucial to prevent the overflow issue during both training and inference stages. Let $\mathbf{P} = \{\boldsymbol{p}_1, \cdots, \boldsymbol{p}_M\}$ denote the global features of $\boldsymbol{Z}$ after the standardization, where $\boldsymbol{p}_t \in \mathbb{R}^d$. Subsequently, $\mathbf{P}$ serves as prompts, being concatenated with each timestamp $\mathbf{X}$ derived from the time series data. Consequently, the large time series models take the integrated vector $\tilde{\mathbf{X}} = \mathbf{P} \cup \mathbf{X} = \{\boldsymbol{p}_1, \cdots, \boldsymbol{p}_M, \mathbf{z}_{t_1}, \mathbf{z}_{t_2}, ..., \mathbf{z}_{t_P}\}$ as input data throughout both training and inference phases, as illustrated in Figure 3. The time series prompts are generated separately for the training and testing datasets, without leaking the information from testing data. We leverage the package[3] to generate the time series prompts.

Experimental Results We begin by evaluating the effectiveness of instruction prompts. Specifically, we assess two distinct types of instruction prompts, both initialized by the same pre-trained GP2-Medium weights within the context of commonly used linear tokenization. The experimental results are shown in Table 1. Our observations suggest that ① statistical prompts outperform traditional text prompts in enhancing the training of LTSM models with up to 8% lower MAE scores. Additionally, ② it is observed that the use of statistical prompts results in superior performance compared to scenarios where no prompts are employed, yielding up to 3% lower MSE scores. The superiority of statistical prompt is evident in the more effective leveraging of LTSM capabilities, leading to improved learning outcomes across various datasets. Based on the above observations, we select time series prompts as the focus in the following analysis and incorporate them into LTSM-bundle.

4.2 Pre-processing: Tokenizations

In addition to employing instructional prompts to enhance generalization in LTSM training, this section provides a detailed analysis aimed at identifying the most effective tokenization strategy for LTSMs. We explore two distinct tokenization approaches – linear tokenization [49] and time

[3] https://github.com/thuml/Time-Series-Library

Table 1: Performance of different prompting strategies

Metric	Input	ETTh1	ETTh2	ETTm1	ETTm2	Traffic	Weather	Electricity	Avg.
MSE	No prompt	0.308	0.237	0.367	0.157	0.306	0.177	0.148	0.243
	TS prompt	0.307	0.234	0.285	0.155	0.305	0.172	0.145	0.229
	Text prompt	0.319	0.241	0.490	0.190	0.345	0.212	0.185	0.283
MAE	No prompt	0.375	0.325	0.411	0.258	0.272	0.232	0.246	0.303
	TS prompt	0.377	0.326	0.369	0.266	0.279	0.242	0.247	0.301
	Text prompt	0.386	0.329	0.476	0.289	0.326	0.269	0.299	0.339

Table 2: Performance of linear and time series tokenization

Metric	Tokenizer	ETTh1	ETTh2	ETTm1	ETTm2	Traffic	Weather	Exchange	Electricity	Avg.
MSE	Linear tokenizer	0.301	0.228	0.261	0.149	0.300	0.163	0.058	0.140	0.214
	Time series tokenizer	1.798	0.855	1.671	0.625	2.199	0.983	3.729	2.206	1.663
MAE	Linear tokenizer	0.372	0.319	0.346	0.265	0.268	0.230	0.173	0.241	0.281
	Time series tokenizer	1.057	0.606	0.991	0.488	1.083	0.619	1.495	1.108	0.895

series tokenization [1] – to determine the superior method for training LTSM models.

Details of Tokenization To harness the power of LLMs, a prevalent strategy involves mapping time series values to tokens [49, 20]. However, converting time series data to natural language formats for LLMs is not trivial, as LLMs are pre-trained with predetermined tokenizers designed for NLP datasets. However, this implies that time series data cannot be directly fed into LLMs for training on forecasting purposes; it requires a specialized transformation of the time series data into specific indices suitable for processing by the LLMs. In this manner, we utilize two advanced types of tokenizations, linear tokenization and time series tokenization, to better evaluate their effectiveness in transferring data for training LTSMs. Specifically, the linear tokenization [49] leverages one trainable linear layer $f : \mathbb{R}^{\mathcal{E}} \rightarrow \mathbb{R}^{K}$ to transfer time series numbers to specific tokens, where $\mathcal{E}$ denotes time series length, and K refers to input size of pre-trained LLM backbone. The trainable time series tokenization [1] aims to covert continuous time series data into discrete tokens by scaling and quantizing their values to the specific number of token bins with a given Dirichlet function.

Experimental Results We investigate the impact of two tokenization methods on training LTSMs. By comparing different tokenization strategies, we aim to identify which approach best complements the LTSM architecture, enhancing its ability to process and learn from complex and multi-domain datasets. Specifically, we conduct experiments comparing linear tokenization and time series tokenization, utilizing pre-trained GPT-2-medium models along with time series prompts. The experimental results shown in Table 2 demonstrate that linear tokenization more effectively facilitates the training process of LTSM compared to time series tokenization. In our study, we focus on smaller and more accessible training data. Under these circumstances, we observe that a linear tokenization is a more suitable choice than a time series tokenization, as the pre-trained time series tokenization may not have the transferability toward different model architecture. Unlike textual tokenization, the time series tokenization is determined by a pre-trained time series dataset with a designated LTSM architecture In summary, ③ linear tokenization is flexible and adaptive for different settings of LTSM training compared to time series tokenization under a smaller amount of training data.

4.3 Model Configuration: Training Paradigm

Different training paradigms exhibit unique characteristics that influence how well LLMs fit a specific training dataset. In this section, we explore three distinct training paradigms, fully fine-tuning, training from scratch, and LoRA [18], to identify the most effective approaches for training the LTSM framework.

Training Paradigm. In the full fine-tuning paradigm, we utilize the pre-trained weights of each base LLM, which finetune all parameters using the given time series dataset. Conversely, in the training-from-scratch paradigm, we only preserve the original model architecture but initialize all parameters anew before training with the time series dataset. In the LoRA paradigm, we employ low-rank adapters on base LLMs.

Experimental Results. We assess the effectiveness of the training paradigm under the settings of time series prompt and text prompt usage. Table 3 presents the results of various training strategies using GPT-2-Medium as the backbone. In general, the experimental results indicate that full fine-tuning is the most effective strategy for training the LTSM framework whether leveraging time series prompts or text prompts. Based on the results, we summarize the observations as follows. ④ Although training-from-scratch achieves competitive performance compared to full fine-tuning, the large number of trainable model parameters may lead to overfitting, ultimately degrading performance. ⑤ Fully fine-tuning paradigm leads to the best performance with up to 11% of improvement on MSE and up to 17% of improvement on MAE under the length of {96, 192, 336}, and performance competitive under the length of 720. Training LTSM-bundle under the full fine-tuning paradigm is recommended, as it converges twice as fast as training from scratch, ensuring efficient and effective forecasting.

4.4 Model Configuration: Base Model Selection

Base Model Candidates As for the base models of our framework, we leverage four different pre-trained models,

Table 3: Performance of learning from scratch, LoRA fine-tuning, and fully fine-tuning

	Metric	MSE				MAE			
	Predict length	**96**	**192**	**336**	**720**	**96**	**192**	**336**	**720**
TS prompt	From scratch	0.325	0.296	0.323	0.355	0.355	0.375	0.374	0.409
	LoRA fine-tuning	0.343	0.381	0.399	0.466	0.374	0.403	0.426	0.478
	Fully fine-tuning	0.229	0.260	0.297	0.351	0.301	0.324	0.354	0.397
Text prompt	From scratch	0.494	0.434	0.597	0.485	0.463	0.438	0.512	0.475
	LoRA fine-tuning	0.347	0.379	0.406	0.473	0.373	0.404	0.431	0.484
	Fully fine-tuning	0.294	0.286	0.353	0.358	0.330	0.338	0.378	0.429

including GPT-2-small, GPT-2-medium, GPT-2-large [33], and Phi-2 [19]. GPT-2 employs a transformer architecture with up to 48 layers, and it is trained on a diverse corpus of internet text, resulting in a model size of 124M (small), 355M (medium), and 774M (large) parameters. Phi-2 also uses a transformer-based architecture but emphasizes high-quality ("textbook-quality") data, comprising 2.7 billion parameters. Despite its smaller size compared to the largest contemporary models, Phi-2 incorporates innovative scaling techniques to optimize performance. Different from the absolute positional encoding used by GPT-2, Phi-2 employs relative positional encoding, which considers the pairwise distance between each token pair for encoding position information of tokens. Following the settings in [49], we utilize the top three self-attention layers of every pre-trained model as our backbone structure in LTSM-bundle framework.

Experimental Results We explore the impact of using different pre-trained LLM weights as backbones in LTSM models, with the goal of identifying the most suitable pre-trained LLM weights for processing time series data. The findings are detailed in Table 4. We assess the performance of different backbones with time series prompts under the fully fine-tuning paradigm. We summarize our observations as follows: ⑥ GPT-2-Small demonstrates a performance improvement of up to 2% in relatively long-term forecasting (i.e., 336 and 720 hours) compared to the GPT-2-Large model. ⑦ GPT-2-Medium outperforms GPT-2-Large in relatively short-term forecasting (i.e., 96 and 192 hours), as larger models may be prone to overfitting during training, degrading forecasting performance.

While our benchmarking results in Table 4 indicate that variations in the number of parameters within the same architecture have minimal impact on performance, we did not limit our analysis to a single model size alone. To ensure the rigor of our evaluation, we compared Phi-2an alternative model architecturewith GPT-2 models of varying sizes (large, medium, and small) using different prompting methods, as detailed in Table 18 and Table 19 of Appendix F. The results show that GPT-2 Small and Medium obtains higher performance than Phi-2 on both time series prompt and textual instruction prompt settings. Based on the above findings, we recommend incorporating GPT-2-Medium or GPT-2-Small as the backbone of LTSM-bundle.

4.5 Dataset configuration: Quantity

The quantity of datasets is often the key to the success of LLMs due to the consistent semantic meaning of tokens. Nevertheless, time series tokens are less informative and semantically meaningful compared to natural language tokens. In this section, we investigate the impact of data quantity to determine whether the principle that more training data leads to better LTSMs.

Quantity Configuration We conduct the down-sampling strategies to study the impact of data quantity on prediction performance. Specifically, each time series in the training data are periodically down-sampled along the timestamps to reduce the granularity of the entire time series while maintaining the general pattern. Each dataset is split into training, validation, and testing sets, and then down-sampling is applied to the training set for model training. We compare the models trained with 10%, 5%, and 2.5% of the full-size time series in the training set. In the following experiment section, we annotate partial training data usage as few-shot training.

Experimental Results Table 5 lists the results of models trained with different data quantities under GPT-2 Medium as the model backbone. The model trained with 5% and 10% down-sampled data leads to the best result compared to 2.5%. ⑧ We observe that increasing the amount of data does not positively correlate with improved model performance. The rationale is that increasing data points enhances the granularity of time series but may reduce the model's generalization ability, while excessive down-sampling loses critical information, hindering pattern learning. Thus, optimizing model performance requires carefully balancing the amount of training data with its diversity. To explore this balance, we conducted experiments using different down-sampling rates across various time series datasets. After comparing with the different downsampling rates (i.e., 2.5%, 10%, and 25% in Table 5 and Table 6). We use 5% of the dataset to benchmark the LTSM-Bundle family across diverse time series data, as increasing the dataset size beyond this point results in only marginal performance improvements while significantly increasing training time (i.e., 10% increase doubles the computational cost but yields only a slight enhancement in forecasting performance).

4.6 Dataset Configuration: Diversity

Impact of Dataset Diversity Recall that we utilized eight datasets for training purposes, encompassing ETT variant, Traffic, Electricity, Weather, and Exchange. Here, we focus on evaluating the performance of LTSM models when trained with subsets of these datasets. Specifically, we employ the first M datasets from the aforementioned list for training, where $M \in 1, 2, ..., 8$. For instance, when $M = 1$, solely ETTh1 is utilized for training; when $M = 5$, ETTh1, ETTh2, ETTm1, ETTm2, and Weather are utilized. Subse-

Table 4: Performance of different backbones

Metric	MSE				MAE			
GPT-2	**96**	**192**	**336**	**720**	**96**	**192**	**336**	**720**
Small	0.252	0.306	0.316	0.352	0.313	0.363	0.367	0.400
Medium	0.229	0.260	0.297	0.351	0.301	0.324	0.354	0.397
Large	0.224	0.257	0.301	0.358	0.292	0.322	0.356	0.399

Table 5: Performance of different down-sampling rates

Metric	MSE				MAE			
DS rate	**96**	**192**	**336**	**720**	**96**	**192**	**336**	**720**
2.5%	0.227	0.268	0.308	0.369	0.294	0.335	0.365	0.415
5%	0.229	0.261	0.297	0.350	0.301	0.324	0.354	0.397
10%	0.233	0.260	0.291	0.350	0.289	0.323	0.348	0.396

quently, we evaluate the trained model's performance across all datasets to understand the impact of dataset diversity.

Experimental Results Table 7 summarizes the results. ⑨ Augmenting dataset diversity generally leads to improved performance. This is expected because more diverse data has the potential to enhance the generalization capabilities of LTSMs across various patterns. We conclude two reasons: (1) Although originating from different domains with distinct characteristics, datasets may share underlying knowledge that can be transferred, enhancing model generalization and performance. (2) Prompting strategies, particularly time series prompts, can further facilitate knowledge transfer by guiding the model to implicitly learn which information to retain and which to discard.

5. COMPARISON WITH BASELINES

Based on the observations in Section 4, we identify a strong combination using LTSM-Bundle with the settings as follows: (1) Base model backbone: GPT-2-Medium, (2) Instruction prompts: the time series prompts, (3) Tokenization: linear tokenization, and (4) Training paradigm: fully fine-tuning. We compare this combination against SoTA TSF models on zero-shot and few-shot settings.

5.1 Experimental Settings

We follow the same settings as in Time-LLM [20]. Specifically, for zero-shot experiments, we test the model's cross-domain adaptation under the long-term forecasting scenario and evaluate it on various cross-domain scenarios utilizing the ETT datasets. The hyperparameter settings of training LTSM-Bundle are in Appendix B. For the few-shot setting, we train our LTSM-Bundle on 5% of the data and compare it with other baselines under the 5% as well. We cite the performance of other models when applicable [49]. Furthermore, we compare LTSM-Bundle trained on 5% training data against baselines trained on the full training set. Our findings in Appendix F indicate that LTSM-Bundle achieves comparable results, further underscoring its superiority.

The baseline methods consist of various Transformer-based methods, including PatchTST [28], ETSformer [42], Non-Stationary Transformer [26], FEDformer [48], Autoformer [7], Informer [47], and Reformer [21]. Additionally, we evaluate our model against recent competitive models like Time-LLM [20], TEST [36], LLM4TS [6], GPT4TS [49], DLinear [46], TimesNet [43], and LightTS [5]. More details of the baseline methods can be found in Section 6.

5.2 Zero-shot and Few-shot Results

Zero-shot Performance In the zero-shot learning experiments shown in Table 8 shows that the best component combination from benchmarking LTSM-Bundle consistently delivers superior performance across various cross-domain scenarios using the ETT datasets. For example, in the ETTh1 to ETTh2 dataset transfer task, LTSM-Bundle achieves an MSE of 0.319 and an MAE of 0.402, outperforming all other methods, including TIME-LLM, GPT4TS, and DLinear. Similarly, in the ETTm1 to ETTm2 dataset transfer scenario, LTSM-Bundle records the lowest MSE and MAE scores of 0.217 and 0.319, showing its strong generalization capability across different domains. The consistent improvements across transfer tasks of LTSM-Bundle in zero-shot learning.

Few-shot Performance Table 9 presents the performance of the best component combination from benchmarking compared to the baseline models in the few-shot setting, utilizing 5% of the training data. Notably, LTSM-Bundle exhibits a significant advantage over both traditional baselines and existing LTSMs. Across the 7 datasets, LTSM-Bundle outperforms all baselines regarding MSE in 5 datasets and regarding MAE in 4 datasets. Moreover, LTSM-Bundle achieves the top rank 40 times among the reported results. These findings underscore the effectiveness of our model in few-shot scenarios, where it demonstrates high accuracy even with limited training data. Its capability to excel with minimal data not only highlights its adaptability but also its potential for practical applications, particularly in contexts where data availability is constrained. All full versions of results on the full datasets are provided in Appendix F.

6. RELATED WORKS

In this work, we focus on benchmarking the training paradigms of LTSMs on top of decoder-only single models. To provide a comprehensive comparison, we categorize existing LLM-based time series forecasting approaches into three representative groups: **1. Pretrained LLM Adaptation.** These methods leverage general-purpose large language models (LLMs), such as GPT, LLaMA, and Phi, for

Table 6: Average performance with different downsampling under different domain LTSM-Bundle

(MAE/MSE)	1 dataset	2 datasets	4 datasets	8 datasets
2.5%	0.446/0.450	**0.435/0.485**	0.396/0.357	0.352/0.293
5%	0.416/0.380	**0.415/0.436**	0.383/0.341	0.344/0.283
10%	0.414/0.375	**0.415/0.440**	0.394/0.355	0.348/0.288

Table 7: Performance of LTSM trained on different numbers of datasets

		1 dataset	**2 datasets**	**3 datasets**	**4 datasets**	**5 datasets**	**6 datasets**	**7 datasets**	**8 datasets**
MSE	96	0.333	0.366	0.269	0.276	0.232	0.229	0.227	0.215
	192	0.394	0.440	0.309	0.318	0.271	0.267	0.269	0.254
	336	0.403	0.427	0.356	0.351	0.302	0.325	0.308	0.302
	720	0.478	0.511	0.436	0.419	0.373	0.364	0.369	0.361
MAE	96	0.351	0.351	0.327	0.329	0.298	0.294	0.292	0.282
	192	0.407	0.395	0.363	0.368	0.333	0.332	0.334	0.323
	336	0.420	0.420	0.401	0.392	0.361	0.384	0.371	0.362
	720	0.464	0.494	0.466	0.445	0.423	0.411	0.419	0.411

Table 8: Zero-shot performance. "LTSM-Bundle" denotes the best combination of LTSM training components

	LTSM-Bundle		**TIME-LLM**		**GPT4TS**		**LLMTime**		**DLinear**		**PatchTST**		**TimesNet**	
Metric	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTh1 → ETTh2	0.319	0.402	0.353	0.387	0.406	0.422	0.992	0.708	0.493	0.488	0.380	0.405	0.421	0.431
ETTh1 → ETTm2	0.312	0.406	0.273	0.340	0.325	0.363	1.867	0.869	0.415	0.452	0.314	0.360	0.327	0.361
ETTm1 → ETTh2	0.306	0.391	0.381	0.412	0.433	0.439	0.992	0.708	0.464	0.475	0.439	0.438	0.457	0.454
ETTm1 → ETTm2	0.217	0.319	0.268	0.320	0.313	0.348	1.867	0.869	0.335	0.389	0.296	0.334	0.322	0.354
ETTm2 → ETTh2	0.314	0.393	0.354	0.400	0.435	0.443	1.867	0.869	0.455	0.471	0.409	0.425	0.435	0.443
ETTm2 → ETTm1	0.403	0.430	0.414	0.438	0.769	0.567	1.933	0.984	0.649	0.537	0.568	0.492	0.769	0.567

time series forecasting through fine-tuning, in-context learning, or lightweight adapter modules. Representative works include GPT4TS [49], which adapts the Frozen Pretrained Transformer (FPT) to predict future sequences, and LLM4TS [6], which extends general-purpose LLMs for temporal modeling. **2. Time Series-Specific LLMs.** These models are explicitly designed and trained on large-scale temporal datasets to better capture temporal dependencies and improve forecasting robustness. Examples include Time-LLM [20], which treats time series data as sequential events for enhanced modeling; TEST [36], which introduces transformer enhancements for complex temporal dependencies; and other recent works such as Chronos and Moirai, which aim to establish foundation-style LLMs for time series. **3. Hybrid Architectures.** Hybrid approaches combine the strengths of LLMs with specialized temporal modeling components to enhance efficiency, interpretability, and robustness. For instance, DLinear [46] integrates linear trend modeling with lightweight forecasting modules, while TimesNet [43] captures multiscale temporal patterns using neural operators. Similarly, LightTS [5] emphasizes efficient tokenization and fast inference for real-time applications. In addition, we benchmark widely adopted transformer-based models, including PatchTST [28], ETSformer [42], Non-Stationary Transformer [26], FEDformer [48], Autoformer [7], Informer [47], and Reformer [21], which serve as essential baselines.

To the best of our knowledge, no prior art has provided a comprehensive benchmark to analyze the effectiveness of each component in the training of LTSMs. Some works [24, 25] maintain a fair platform to compare different time series forecasting methods. The others [14, 32] analyze the forecasting performance from the perspectives of time series patterns. This benchmark provides an accessible and modular pipeline for evaluating a diverse set of training components in LTSM development, leveraging a time series database and user-friendly visualization. With the debates on whether LLMs can benefit from time series forecasting tasks, our toolbox offers a scikit-learn-like API interface to efficiently explore each component's effectiveness in training LTSMs.

7. CONCLUSION AND FUTURE PERSPECTIVES

In this study, we present the first comprehensive toolbox and benchmark for understanding different design choices in training LLMs for time series forecasting. Our benchmark covers various aspects, including data preprocessing, model configuration, and dataset configuration. We delve into detailed design choices such as prompting, tokenization, training paradigms, base model selection, data quantity, and dataset diversity. Through this analysis, we derive 9 observations and identify the best combination for

Table 9: Performance comparison in the few-shot setting with 5% training data. The full results are provided in Appendix F. "LTSM-Bundle" denotes the best combination of LTSM training components

		LTSM-Bundle		**TIME-LLM**		**LLM4TS**		**GPT4TS**		**DLinear**		**PatchTST**		**TimesNet**		**FEDformer**	
Metric		**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTh1	96	0.307	0.377	0.483	0.464	0.509	0.484	0.543	0.506	0.547	0.503	0.557	0.519	0.892	0.625	0.593	0.529
	192	0.329	0.391	0.629	0.540	0.717	0.581	0.748	0.580	0.720	0.604	0.711	0.570	0.940	0.665	0.652	0.563
	336	0.346	0.405	0.768	0.626	0.728	0.589	0.754	0.595	0.984	0.727	0.816	0.619	0.945	0.653	0.731	0.594
	720	0.370	0.441	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Avg	0.338	0.403	0.627	0.543	0.651	0.551	0.681	0.560	0.750	0.611	0.694	0.569	0.925	0.647	0.658	0.562
ETTh2	96	0.235	0.326	0.336	0.397	0.314	0.375	0.376	0.421	0.442	0.456	0.401	0.421	0.409	0.420	0.390	0.424
	192	0.283	0.365	0.406	0.425	0.365	0.408	0.418	0.441	0.617	0.542	0.452	0.455	0.483	0.464	0.457	0.465
	336	0.320	0.401	0.405	0.432	0.398	0.432	0.408	0.439	1.424	0.849	0.464	0.469	0.499	0.479	0.477	0.483
	720	0.378	0.456	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Avg	0.303	0.387	0.382	0.418	0.359	0.405	0.400	0.433	0.694	0.577	0.439	0.448	0.439	0.448	0.463	0.454
ETTm1	96	0.285	0.369	0.316	0.377	0.349	0.379	0.386	0.405	0.332	0.374	0.399	0.414	0.606	0.518	0.628	0.544
	192	0.319	0.393	0.450	0.464	0.374	0.394	0.440	0.438	0.358	0.390	0.441	0.436	0.681	0.539	0.666	0.566
	336	0.378	0.425	0.450	0.424	0.411	0.417	0.485	0.459	0.402	0.416	0.499	0.467	0.786	0.597	0.807	0.628
	720	0.464	0.477	0.483	0.471	0.516	0.479	0.577	0.499	0.511	0.489	0.767	0.587	0.796	0.593	0.822	0.633
	Avg	0.362	0.416	0.425	0.434	0.412	0.417	0.472	0.450	0.400	0.417	0.526	0.476	0.717	0.561	0.730	0.592
ETTm2	96	0.156	0.266	0.174	0.261	0.192	0.273	0.199	0.280	0.236	0.326	0.206	0.288	0.220	0.299	0.229	0.320
	192	0.203	0.307	0.215	0.287	0.249	0.309	0.256	0.316	0.306	0.373	0.264	0.324	0.311	0.361	0.394	0.361
	336	0.255	0.349	0.273	0.330	0.301	0.342	0.318	0.353	0.380	0.423	0.334	0.367	0.338	0.366	0.378	0.427
	720	0.342	0.417	0.433	0.412	0.402	0.405	0.460	0.436	0.674	0.583	0.454	0.432	0.509	0.465	0.523	0.510
	Avg	0.239	0.335	0.274	0.323	0.286	0.332	0.308	0.346	0.399	0.426	0.314	0.352	0.344	0.372	0.381	0.404
Weather	96	0.172	0.242	0.172	0.263	0.173	0.227	0.175	0.230	0.184	0.242	0.171	0.224	0.207	0.253	0.229	0.309
	192	0.218	0.278	0.224	0.271	0.218	0.265	0.227	0.276	0.228	0.283	0.230	0.277	0.272	0.307	0.265	0.317
	336	0.276	0.329	0.282	0.321	0.276	0.310	0.286	0.322	0.279	0.322	0.294	0.326	0.313	0.328	0.353	0.392
	720	0.339	0.373	0.366	0.381	0.355	0.366	0.366	0.379	0.364	0.388	0.384	0.387	0.400	0.385	0.391	0.394
	Avg	0.251	0.305	0.260	0.309	0.251	0.292	0.263	0.301	0.263	0.308	0.269	0.303	0.298	0.318	0.309	0.353
Electricity	96	0.145	0.247	0.147	0.242	0.139	0.235	0.143	0.241	0.150	0.251	0.145	0.244	0.315	0.389	0.235	0.322
	192	0.159	0.259	0.158	0.241	0.155	0.249	0.159	0.255	0.163	0.263	0.163	0.260	0.318	0.396	0.247	0.341
	336	0.180	0.284	0.178	0.277	0.174	0.269	0.179	0.274	0.175	0.278	0.183	0.281	0.340	0.415	0.267	0.356
	720	0.215	0.317	0.224	0.312	0.222	0.310	0.233	0.323	0.219	0.311	0.233	0.323	0.635	0.613	0.318	0.394
	Avg	0.175	0.276	0.179	0.268	0.173	0.266	0.178	0.273	0.176	0.275	0.181	0.277	0.402	0.453	0.266	0.353
Traffic	96	0.305	0.279	0.414	0.291	0.401	0.285	0.419	0.298	0.427	0.304	0.404	0.286	0.854	0.492	0.670	0.421
	192	0.313	0.274	0.419	0.291	0.418	0.293	0.434	0.305	0.447	0.315	0.412	0.294	0.894	0.517	0.653	0.405
	336	0.326	0.287	0.437	0.314	0.436	0.308	0.449	0.313	0.478	0.333	0.439	0.310	0.853	0.471	0.707	0.445
	720	0.346	0.301	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Avg	0.323	0.285	0.423	0.298	0.418	0.295	0.434	0.305	0.450	0.317	0.418	0.296	0.867	0.493	0.676	0.423
1^{st} Count		47		6		16		0		2		2		0		0	

training LTSMs. We demonstrate that this combination achieves strong zero-shot and few-shot performance compared to state-of-the-art LTSMs, and it requires only 5% of the data to achieve comparable performance to state-of-the-art baselines on benchmark datasets. We hope that our findings will inspire future research in this direction, and this combination based on LTSM-bundle could serve as a simple yet strong baseline for future comparison.

An ongoing debate exists about whether using pre-trained weights from large language models (LLMs) can enhance time series forecasting performance [37]. In response, the LTSM-Bundle package provides a robust platform for further investigation. Our current findings indicate that incorporating pre-trained weights can indeed improve forecasting performance. However, as LTSM training techniques evolve, future enhancements may offer additional insights and even different perspectives on the benefits of these pre-trained weights. Notably, using pre-trained weights as a starting point may reduce training time and help the models converge faster. We suggest two potential directions for enhancing the LTSM training components, as outlined below:

Advancing Prompting Strategies. The heterogeneity of time series datasets presents a significant challenge in training a universal model that can effectively fit all datasets while generalizing to unseen ones. Our analysis highlights the potential of prompting as a solution by enriching datasets with additional contextual information. Specifically, we demonstrate the effectiveness of the time series prompt, which extracts statistical insights to improve model performance. Looking ahead, we anticipate the development of more sophisticated prompting strategies to further enhance generalization. For instance, incorporating variate-specific prompts in multivariate time series data could provide richer context and improve predictive accuracy. We believe this direction holds substantial promise for future research advancements.

Constructing Synthetic Training Data. Our analysis underscores the importance of dataset diversity in training transferable LTSMs. Increasing the number of datasets significantly improves performance, as exposure to diverse patterns enhances transferability. This finding points to the potential of using synthetic datasets to simulate varied temporal patterns. Unlike prior work such as Chronos [1], which benefits from extremely large synthetic datasets.

References

[1] A. F. Ansari, L. Stella, C. Turkmen, X. Zhang, P. Mercado, H. Shen, O. Shchur, S. S. Rangapuram, S. P. Arango, S. Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

[2] A. A. Ariyo, A. O. Adewumi, and C. K. Ayo. Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th international conference on computer modelling and simulation*, pages 106–112. IEEE, 2014.

[3] B. S. Atal. Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. *the Journal of the Acoustical Society of America*, 55(6):1304–1312, 1974.

[4] M. Barandas, D. Folgado, L. Fernandes, S. Santos, M. Abreu, P. Bota, H. Liu, T. Schultz, and H. Gamboa. Tsfel: Time series feature extraction library. *SoftwareX*, 11:100456, 2020.

[5] D. Campos, M. Zhang, B. Yang, T. Kieu, C. Guo, and C. S. Jensen. Lightts: Lightweight time series classification with adaptive ensemble distillation. *Proceedings of the ACM on Management of Data*, 1(2):1–27, 2023.

[6] C. Chang, W.-C. Peng, and T.-F. Chen. Llm4ts: Two-stage fine-tuning for time-series forecasting with pre-trained llms. *arXiv preprint arXiv:2308.08469*, 2023.

[7] M. Chen, H. Peng, J. Fu, and H. Ling. Autoformer: Searching transformers for visual recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12270–12280, 2021.

[8] A. Das, W. Kong, R. Sen, and Y. Zhou. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2023.

[9] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 28(4):357–366, 1980.

[10] S. Dooley, G. S. Khurana, C. Mohapatra, S. V. Naidu, and C. White. Forecastpfn: Synthetically-trained zero-shot forecasting. *Advances in Neural Information Processing Systems*, 36, 2024.

[11] S. Elsworth and S. Güttel. Time series forecasting using lstm networks: A symbolic approach. *arXiv preprint arXiv:2003.05672*, 2020.

[12] L. Fernandes, M. Barandas, and H. Gamboa. Learning human behaviour patterns by trajectory and activity recognition. In *BIOSIGNALS*, pages 220–227, 2020.

[13] C. Figueira, R. Matias, and H. Gamboa. Body location independent activity monitoring. In *International Conference on Bio-inspired Systems and Signal Processing*, volume 5, pages 190–197. SciTePress, 2016.

[14] E. Fons, R. Kaur, S. Palande, Z. Zeng, T. Balch, M. Veloso, and S. Vyetrenko. Evaluating large language models on time series feature understanding: A comprehensive taxonomy and benchmark. *arXiv preprint arXiv:2404.16563*, 2024.

[15] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[16] A. Garza and M. Mergenthaler-Canseco. Timegpt-1. *arXiv preprint arXiv:2310.03589*, 2023.

[17] N. Gruver, M. Finzi, S. Qiu, and A. G. Wilson. Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems*, 36, 2024.

[18] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[19] M. Javaheripi and S. Bubeck. Phi-2: The surprising power of small language models, December 2023.

[20] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

[21] N. Kitaev, Ł. Kaiser, and A. Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[22] Ç. Kocaman and M. Özdemir. Comparison of statistical methods and wavelet energy coefficients for determining two common pq disturbances: Sag and swell. In *2009 International Conference on Electrical and Electronics Engineering-ELECO 2009*, pages I–80. IEEE, 2009.

[23] G. Lai, W.-C. Chang, Y. Yang, and H. Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

[24] Z. Li, X. Qiu, P. Chen, Y. Wang, H. Cheng, Y. Shu, J. Hu, C. Guo, A. Zhou, Q. Wen, et al. Foundts: Comprehensive and unified benchmarking of foundation models for time series forecasting. *arXiv preprint arXiv:2410.11802*, 2024.

[25] H. Liu, S. Xu, Z. Zhao, L. Kong, H. Prabhakar Kamarthi, A. Sasanur, M. Sharma, J. Cui, Q. Wen, C. Zhang, et al. Time-mmd: Multi-domain multimodal dataset for time series analysis. *Advances in Neural Information Processing Systems*, 37:77888–77933, 2025.

[26] Y. Liu, H. Wu, J. Wang, and M. Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, 35:9881–9893, 2022.

[27] I. E. Livieris, E. Pintelas, and P. Pintelas. A cnn–lstm model for gold price time-series forecasting. *Neural computing and applications*, 32:17351–17360, 2020.

[28] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

[29] T. E. Oliphant et al. *Guide to numpy*, volume 1. Trelgol Publishing USA, 2006.

[30] Y. Pan, J. Chen, and X. Li. Spectral entropy: A complementary index for rolling element bearing performance degradation assessment. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 223(5):1223–1231, 2009.

[31] G. Peeters, B. L. Giordano, P. Susini, N. Misdariis, and S. McAdams. The timbre toolbox: Extracting audio descriptors from musical signals. *The Journal of the Acoustical Society of America*, 130(5):2902–2916, 2011.

[32] X. Qiu, X. Li, R. Pang, Z. Pan, X. Wu, L. Yang, J. Hu, Y. Shu, X. Lu, C. Yang, et al. Easytime: Time series forecasting made easy. *arXiv preprint arXiv:2412.17603*, 2024.

[33] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[34] S. Raschka. Mlxtend: Providing machine learning and data science utilities and extensions to pythons scientific computing stack. *Journal of open source software*, 3(24):638, 2018.

[35] K. Rasul, A. Ashok, A. R. Williams, H. Ghonia, R. Bhagwatkar, A. Khorasani, M. J. D. Bayazi, G. Adamopoulos, R. Riachi, N. Hassen, et al. Lag-llama: Towards foundation models for probabilistic time series forecasting. *Preprint*, 2024.

[36] C. Sun, Y. Li, H. Li, and S. Hong. Test: Text prototype aligned embedding to activate llm's ability for time series. *arXiv preprint arXiv:2308.08241*, 2023.

[37] M. Tan, M. A. Merrill, V. Gupta, T. Althoff, and T. Hartvigsen. Are language models actually useful for time series forecasting? *arXiv preprint arXiv:2406.16964*, 2024.

[38] H. U. Power spectrum and bandwidth. ., 2003.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[40] P. Welch. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on audio and electroacoustics*, 15(2):70–73, 1967.

[41] G. Woo, C. Liu, A. Kumar, C. Xiong, S. Savarese, and D. Sahoo. Unified training of universal time series forecasting transformers. *arXiv preprint arXiv:2402.02592*, 2024.

[42] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. Hoi. Etsformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arXiv:2202.01381*, 2022.

[43] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The eleventh international conference on learning representations*, 2022.

[44] H. Wu, J. Xu, J. Wang, and M. Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

[45] B. Yan, A. Miyamoto, and E. Brühwiler. Wavelet transform-based modal parameter identification considering uncertainty. *Journal of Sound and Vibration*, 291(1-2):285–301, 2006.

[46] A. Zeng, M. Chen, L. Zhang, and Q. Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

[47] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[48] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pages 27268–27286. PMLR, 2022.

[49] T. Zhou, P. Niu, L. Sun, R. Jin, et al. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems*, 36:43322–43355, 2023.

[50] D. Zwillinger and S. Kokoska. *CRC standard probability and statistics tables and formulae*. Crc Press, 1999.

APPENDIX

A. DETAILS OF DATASETS

In this paper, the training datasets include ETT (Electricity Transformer Temperature) [47][4], Traffic[5], Electricity[6], Weather[7], and Exchange-Rate[23]. ETT[8] [47] comprises four subsets: two with hourly-level data (ETTh) and two with 15-minute-level data (ETTm). Each subset includes seven features related to oil and load metrics of electricity transformers, covering the period from July 2016 to July 2018. The traffic dataset includes hourly road occupancy rates from sensors on San Francisco freeways, covering the period from 2015 to 2016. The electricity dataset contains hourly electricity consumption data for 321 clients, spanning from 2012 to 2014. The weather data set comprises 21 weather indicators, such as air temperature and humidity, recorded every 10 minutes throughout 2020 in Germany. Exchange-Rate[23] contains daily exchange rates for eight countries, spanning from 1990 to 2016. We first train our framework on the diverse time series data collection and then assess the abilities of `LTSM-Bundle` on jointly learning and zero-shot transfer learning to different domains of time series knowledge.

B. HYPER-PARAMETER SETTINGS OF EXPERIMENTS

The hyper-parameter settings of `LTSM-Bundle` training for all experiments are shown in Table 10. Other training hyper-parameters follow the default values in the `TrainingArguments` class[9] of the huggingface transformers package.

C. COMPUTATION INFRASTRUCTURE

All experiments described in this paper are conducted using a well-defined physical computing infrastructure, the specifics of which are outlined in Table 11. This infrastructure is essential for ensuring the reproducibility and reliability of our results, as it details the exact hardware environments used during the testing phases.

D. GLOBAL FEATURES FOR PROMPTS

Time series prompts are developed to encapsulate the comprehensive characteristics of time series data. Specifically, let $\boldsymbol{s}$ be a certain variate of the time-series data $\boldsymbol{Z}$. We identified in Table 12 the partial global features that we leverage to craft these prompts, each selected for its ability to convey critical information about the data's temporal structure and variability. For the inter-quartile and histogram in Table 12, $Q_3(\boldsymbol{s})$ and $Q_1(\boldsymbol{s})$ represent the first and third quartile of the Time series data, respectively; and m_i represents the histogram in which n is the total number of observations and k the total number of bins. Beyond those shown in Table 12, we also consider the global features according to the following references: Fast Fourier Transform, Wavelet transform, Zero crossing rate, Maximum peaks, Minimum peaks, ECDF percentile count, Slope, ECDF slope, Spectral distance, Fundamental frequency, Maximum frequency, Median frequency, Spectral maximum peaks [4]; Maximum Power Spectrum [40], Spectral Centroid [31], Decrease [31], Kurtosis [31], Skewness [31], Spread [31], Slope [31], Variation [31], Spectral Roll-off [13], Roll-on [13], Human Range Energy [12], MFCC [9], LPCC [3], Power Bandwidth [38], Spectral Entropy [30], Wavelet Entropy [45] and Wavelet Energy [22], Kurtosis [50], Skewness [50], Maximum [29], Minimum [29], Mean [29], Median [29] and ECDF [34], ECDF Percentile [34]. For the implementation, we leverage the TSFEL library[10] [4] to estimate the global features. The features are extracted separately for each variate in the time-series data.

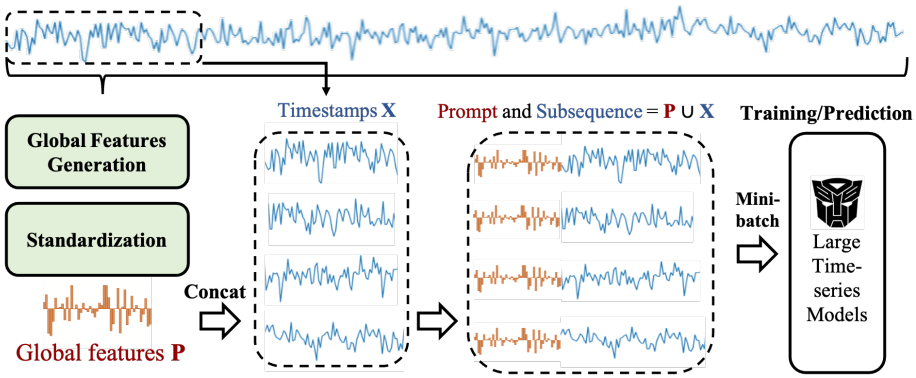


Figure 3: Time series prompt generation process based on the given training time series data.

[4] https://github.com/zhouhaoyi/ETDataset
[5] http://pems.dot.ca.gov
[6] https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014
[7] https://www.bgc-jena.mpg.de/wetter/
[8] https://github.com/laiguokun/multivariate-time-series-data
[9] https://github.com/huggingface/transformers/blob/main/src/transformers/training_args.py
[10] https://tsfel.readthedocs.io/en/latest/descriptions/feature_list.html

E. COMPARISON WITH OTHER PACKAGES

In this section, we highlight the difference and advantages of `LTSM-Bundle` comparing to other existing open source LTSM packages, including OpenLTM[11], Time-LLM [12], and LLM-Time[13]. Our package involve more industrial-oriented and user-friendly features, such as database integration and report visualization.

F. ADDITIONAL EXPERIMENTAL RESULTS ON `LTSM-BUNDLE`

In this section, we show additional results regarding comparing `LTSM-Bundle` with other baselines in Tables 16 and 17, results of zero-shot transfer learning in Table 14, results of different training paradigms in Table 18, results of different backbones in Table 19, results of different downsampling ratios in Table 20.

F.1 Performance Comparison with Additional Baselines

Extending the analysis presented in Section 5.2, we introduce full performance comparison with new baselines. We evaluate the proposed `LTSM-Bundle` in zero-shot and few-shot settings to highlight its efficacy and robustness in Table 16 and 17.

F.2 Zero-shot Transfer Learning Comparisons

In addition to the results in Section 5.2, this section introduces the full zero-shot transfer learning comparisons. We evaluate the proposed `LTSM-Bundle` in the zero-shot transfer scenarios, detailed in shown in Table 14.

F.3 Training Paradigm Comparisons

Expanding upon the results in Section 4.3, this section presents the full experimental results for the training paradigms analysis, including different backbones and prompting strategies. The analytic results are detailed in Table 18.

F.4 Backbone Architecture Comparisons

We provide all the numbers of analytics on different backbone architectures, continuing from Section 4.4. Results are in different language model backbones, including GPT-2-Small, GPT-2-Medium, GPT-2-Large, and Phi-2, shown in Table 19.

F.5 Down-sampling Ratio Comparisons

We here present the full version of our experimental results on the different down-sampling ratios in Section 4.6. We test `LTSM-Bundle` with GPT-Medium as backbones with the proposed TS prompt under a fully tuning paradigm. The results in the ratio of {40, 20, 10} (i.e., downsample rate in {2.5%, 5%, 10% }) are all demonstrated in Table 20.

F.6 Different Numbers of Layer Adaptation Comparisons

We compared the average performance among all datasets of a 3-layer model and a full 24-layer model using GPT-medium as the backbone. Our results (in Table 15) show that the 24-layer model performs worse when trained with the same number of iterations. We believe this suggests that the 3-layer configuration is a reasonable strategy for benchmarking at this stage.

Table 10: Hyperparameter settings of `LTSM-Bundle` training

Hyperparameter name	**Value**
Number of Transformer layers N	3
Training / evaluation / testing split	0.7 / 0.1 / 0.2
Gradient accumulation steps	64
Learning rate	0.001
Optimizer	Adam
LR scheduler	CosineAnnealingLR
Number of epochs	10
Number of time steps per token	16
Stride of time steps per token	8
Dimensions of TS prompt	133
Transformer architectures	GPT-2-{small, medium, large}, Phi-2
Length of prediction	96, 192, 336, 720
Length of input TS data	336
Data type	`torch.bfloat16`
Downsampling rate of training data	20

[11] https://github.com/thuml/OpenLTM
[12] https://github.com/KimMeen/Time-LLM
[13] https://github.com/ngruver/llmtime

Table 11: Computing infrastructure for the experiments

Device attribute	Value
Computing infrastructure	GPU
GPU model	Nvidia A5000 / Nvidia A100
GPU number	8 A5000 / 4 A100
GPU memory	8 24GB / 4 80GB

Table 12: Partial global features in time series prompts

Feature	**Formula**	**Feature**	**Formula**
Autocorrelation	$\sum_{i\in\mathbb{Z}} s_i s_{i-l}$	Centroid	$\sum_{i=0}^{T} t_i \cdot s_i^2 / \sum_{i=0}^{T} s_i^2$
Max differences	$\max_i(s_{i+1} - s_i)$	Mean differences	$\text{mean}_i(s_{i+1} - s_i)$
Median differences	$\text{median}_i(s_{i+1} - s_i)$	Max absolute differences	$\max_i \lvert s_{i+1} - s_i\rvert$
Mean absolute differences	$\text{mean}_i\lvert s_{i+1} - s_i\rvert$	Median absolute differences	$\text{median}_i\lvert s_{i+1} - s_i\rvert$
Distance	$\sum_{i=0}^{T-1} \sqrt{1 + (s_{i+1} - s_i)^2}$	Sum of absolute differences	$\sum_{i=0}^{T-1} \lvert s_{i+1} - s_i\rvert$
Total energy	$\sum_{i=0}^{T} s_i^2 \cdot (t_T - t_0)$	Entropy	$-\sum_{x\in s} P(x) \log_2 P(x)$
Peak-to-peak distance	$\lvert\max(\boldsymbol{s}) - \min(\boldsymbol{s})\rvert$	Area under curve	$\sum_{i=0}^{T-1}(t_{i+1} - t_i) \cdot \frac{s_{i+1}+s_i}{2}$
Absolute energy	$\sum_{i=0}^{T} s_i^2$	Histogram	$n = \sum_{i=1}^{k} m_i$
Inter-quartile range	$Q_3(\boldsymbol{s}) - Q_1(\boldsymbol{s})$	Mean absolute deviation	$\frac{1}{T}\sum_{i=1}^{T} \lvert s_i^2 - \text{mean}(\boldsymbol{s})\rvert$
Median absolute deviation	$\text{median}_i(\lvert s_i - \text{median}(\boldsymbol{s})\rvert)$	Root mean square	$\sqrt{\frac{1}{T}\sum_{i=1}^{T} s_i^2}$
Standard deviation (STD)	$\sqrt{\frac{1}{T}\sum_{i=1}^{T}(s_i - \text{mean}(\boldsymbol{s}))^2}$	Variance (VAR)	$\frac{1}{T}\sum_{i=1}^{T}(s_i - \text{mean}(\boldsymbol{s}))^2$
Wavelet absolute mean	$\lvert\text{mean}(\text{wavelet}(s))\rvert$	Wavelet standard deviation	$\lvert\text{std}(\text{wavelet}(\boldsymbol{s}))\rvert$
Wavelet variance	$\lvert\text{var}(\text{wavelet}(\boldsymbol{s}))\rvert$	Skewness	$\frac{1}{T(\text{STD})^3}\sum_{i=0}^{T}(s_i - \text{mean}(\boldsymbol{s}))^3$

Table 13: Feature comparison with other LTSM open source packages

	LTSM-Bundle	OpenLTM	Time-LLM	LLM-Time
Support for multiple model architectures and prompting strategies	**Yes**	Yes	No	No
Integration with database	**Yes**	No	No	No
Data preprocessing and pipeline integration	**Yes**	No	No	No
Zero-shot	**Yes**	Yes	Yes	Yes
Visualization	**Yes**	No	No	Yes

Table 14: Results of zero-shot transfer learning. A time-series model is trained on a source dataset and transferred to the target dataset without adaptation.

Methods		LTSM-Bundle		TIME-LLM		LLMTime		GPT4TS		DLinear		PatchTST		TimesNet		Autoformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1 → ETTh2	96	0.229	0.326	0.279	0.337	0.510	0.576	0.335	0.374	0.347	0.400	0.304	0.350	0.358	0.387	0.469	0.486
	192	0.310	0.395	0.351	0.374	0.523	0.586	0.412	0.417	0.417	0.460	0.386	0.400	0.427	0.429	0.634	0.567
	336	0.336	0.414	0.388	0.415	0.640	0.637	0.441	0.444	0.515	0.505	0.414	0.428	0.449	0.451	0.655	0.588
	720	0.401	0.474	0.391	0.420	2.296	1.034	0.438	0.452	0.665	0.589	0.419	0.443	0.448	0.458	0.570	0.549
	Avg	0.319	0.402	0.353	0.387	0.992	0.708	0.406	0.422	0.493	0.488	0.380	0.405	0.421	0.431	0.582	0.548
ETTh1 → ETTm2	96	0.197	0.318	0.189	0.293	0.646	0.563	0.236	0.315	0.255	0.357	0.215	0.304	0.239	0.313	0.352	0.432
	192	0.314	0.420	0.237	0.312	0.934	0.654	0.287	0.342	0.338	0.413	0.275	0.339	0.291	0.342	0.413	0.460
	336	0.313	0.405	0.291	0.365	1.157	0.728	0.341	0.374	0.425	0.465	0.334	0.373	0.342	0.371	0.465	0.489
	720	0.425	0.483	0.372	0.390	4.730	1.531	0.435	0.422	0.640	0.573	0.431	0.424	0.434	0.419	0.599	0.551
	Avg	0.312	0.406	0.273	0.340	1.867	0.869	0.325	0.363	0.415	0.452	0.314	0.360	0.327	0.361	0.457	0.483
ETTh2 → ETTh1	96	0.390	0.439	0.450	0.452	1.130	0.777	0.732	0.577	0.689	0.555	0.485	0.465	0.848	0.601	0.693	0.569
	192	0.417	0.460	0.465	0.461	1.242	0.820	0.758	0.559	0.707	0.568	0.565	0.509	0.860	0.610	0.760	0.601
	336	0.462	0.501	0.501	0.482	1.382	0.864	0.759	0.578	0.710	0.577	0.581	0.515	0.867	0.626	0.781	0.619
	720	0.568	0.588	0.501	0.502	4.145	1.461	0.781	0.597	0.704	0.596	0.628	0.561	0.887	0.648	0.796	0.644
	Avg	0.459	0.497	0.479	0.474	1.961	0.981	0.757	0.578	0.703	0.574	0.565	0.513	0.865	0.621	0.757	0.608
ETTh2 → ETTm2	96	0.200	0.316	0.174	0.276	0.646	0.563	0.253	0.329	0.240	0.336	0.226	0.309	0.248	0.324	0.263	0.352
	192	0.250	0.359	0.233	0.315	0.934	0.654	0.293	0.346	0.295	0.369	0.289	0.345	0.296	0.352	0.326	0.389
	336	0.327	0.416	0.291	0.337	1.157	0.728	0.347	0.376	0.345	0.397	0.348	0.379	0.353	0.383	0.387	0.426
	720	0.573	0.563	0.392	0.417	4.730	1.531	0.446	0.429	0.432	0.442	0.439	0.427	0.471	0.446	0.487	0.478
	Avg	0.337	0.413	0.272	0.341	1.867	0.869	0.335	0.370	0.328	0.386	0.325	0.365	0.342	0.376	0.366	0.411
ETTm1 → ETTh2	96	0.246	0.342	0.321	0.369	0.510	0.576	0.353	0.392	0.365	0.415	0.354	0.385	0.377	0.407	0.435	0.470
	192	0.290	0.374	0.389	0.410	0.523	0.586	0.443	0.437	0.454	0.462	0.447	0.434	0.471	0.453	0.495	0.489
	336	0.326	0.406	0.408	0.433	0.640	0.637	0.469	0.461	0.496	0.464	0.481	0.463	0.472	0.484	0.470	0.472
	720	0.363	0.440	0.406	0.436	2.296	1.034	0.466	0.468	0.541	0.529	0.474	0.471	0.495	0.482	0.480	0.485
	Avg	0.306	0.391	0.381	0.412	0.992	0.708	0.433	0.439	0.464	0.475	0.439	0.438	0.457	0.454	0.470	0.479
ETTm1 → ETTm2	96	0.144	0.257	0.169	0.257	0.646	0.563	0.217	0.294	0.221	0.314	0.195	0.271	0.222	0.295	0.385	0.457
	192	0.193	0.302	0.227	0.318	0.934	0.654	0.277	0.327	0.286	0.359	0.258	0.311	0.288	0.337	0.433	0.469
	336	0.240	0.342	0.290	0.338	1.157	0.728	0.331	0.360	0.357	0.406	0.317	0.348	0.341	0.367	0.476	0.477
	720	0.292	0.379	0.375	0.367	4.730	1.531	0.429	0.413	0.476	0.476	0.416	0.404	0.436	0.418	0.582	0.535
	Avg	0.217	0.320	0.268	0.320	1.867	0.869	0.313	0.348	0.335	0.389	0.296	0.334	0.322	0.354	0.469	0.484
ETTm2 → ETTh2	96	0.257	0.346	0.298	0.356	0.510	0.576	0.360	0.401	0.333	0.391	0.327	0.367	0.360	0.401	0.353	0.393
	192	0.309	0.382	0.359	0.397	0.523	0.586	0.434	0.437	0.441	0.456	0.411	0.418	0.434	0.437	0.432	0.437
	336	0.341	0.413	0.367	0.412	0.640	0.637	0.460	0.459	0.505	0.503	0.439	0.447	0.460	0.459	0.452	0.459
	720	0.350	0.432	0.393	0.434	2.296	1.034	0.485	0.477	0.543	0.534	0.459	0.470	0.485	0.477	0.453	0.467
	Avg	0.314	0.393	0.354	0.400	0.992	0.708	0.435	0.443	0.455	0.471	0.409	0.425	0.435	0.443	0.423	0.439
ETTm2 → ETTm1	96	0.364	0.410	0.359	0.397	1.179	0.781	0.747	0.558	0.570	0.490	0.491	0.437	0.747	0.558	0.735	0.576
	192	0.405	0.432	0.390	0.420	1.327	0.846	0.781	0.560	0.590	0.506	0.530	0.470	0.781	0.560	0.753	0.586
	336	0.413	0.433	0.421	0.445	1.478	0.902	0.778	0.578	0.706	0.567	0.565	0.497	0.778	0.578	0.750	0.593
	720	0.432	0.446	0.487	0.488	3.749	1.408	0.769	0.573	0.731	0.584	0.686	0.565	0.769	0.573	0.782	0.609
	Avg	0.403	0.430	0.414	0.438	1.933	0.984	0.769	0.567	0.649	0.537	0.568	0.492	0.769	0.667	0.755	0.591

Table 15: Performance comparison between 3-layer and 24-layer of LTSM-Bundle.

	3-layer	24-layer
MAE	0.2003	0.2439
MSE	0.2770	0.3162

Table 16: Performance comparison with additional baselines (Full data)

Methods		LTSM-Bundle		TIME-LLM		TEST		LLM4TS		GPT4TS		DLinear		PatchTST		TimesNet		FEDformer		Autoformer		Non-Stationary		ETSformer		LightTS		Informer		Reformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.307	0.377	0.362	0.392	0.372	0.400	0.371	0.394	0.376	0.397	0.375	0.399	0.370	0.399	0.384	0.402	0.376	0.419	0.449	0.459	0.513	0.491	0.494	0.479	0.424	0.432	0.865	0.713	0.837	0.728
	192	0.329	0.391	0.398	0.418	0.414	0.422	0.403	0.412	0.416	0.418	0.405	0.416	0.413	0.421	0.436	0.429	0.420	0.448	0.500	0.482	0.534	0.504	0.538	0.504	0.475	0.462	1.008	0.792	0.923	0.766
	336	0.346	0.405	0.430	0.427	0.422	0.437	0.420	0.422	0.442	0.433	0.439	0.443	0.422	0.436	0.491	0.469	0.459	0.465	0.521	0.496	0.588	0.535	0.574	0.521	0.518	0.488	1.107	0.809	1.097	0.835
	720	0.370	0.441	0.442	0.457	0.447	0.467	0.422	0.444	0.477	0.456	0.472	0.490	0.447	0.466	0.521	0.500	0.506	0.507	0.514	0.512	0.643	0.616	0.562	0.535	0.547	0.533	1.181	0.865	1.257	0.889
	Avg	0.338	0.403	0.408	0.423	0.414	0.431	0.404	0.418	0.465	0.455	0.422	0.437	0.413	0.430	0.458	0.450	0.440	0.460	0.496	0.487	0.570	0.537	0.542	0.510	0.491	0.479	1.040	0.795	1.029	0.805
ETTh2	96	0.235	0.326	0.268	0.328	0.275	0.338	0.269	0.332	0.285	0.342	0.289	0.353	0.274	0.336	0.340	0.374	0.358	0.397	0.346	0.388	0.476	0.458	0.340	0.391	0.397	0.437	3.755	1.525	2.626	1.317
	192	0.283	0.365	0.329	0.375	0.340	0.379	0.328	0.377	0.354	0.389	0.383	0.418	0.339	0.379	0.402	0.414	0.429	0.439	0.456	0.452	0.512	0.493	0.430	0.439	0.520	0.504	5.602	1.931	11.12	2.979
	336	0.320	0.401	0.368	0.409	0.329	0.381	0.353	0.369	0.373	0.407	0.448	0.465	0.329	0.380	0.452	0.452	0.496	0.487	0.482	0.486	0.552	0.551	0.485	0.479	0.626	0.559	4.721	1.835	9.323	2.769
	720	0.378	0.456	0.372	0.420	0.381	0.423	0.383	0.425	0.406	0.441	0.605	0.551	0.379	0.422	0.462	0.468	0.463	0.474	0.515	0.511	0.562	0.560	0.500	0.497	0.863	0.672	3.647	1.625	3.874	1.697
	Avg	0.304	0.387	0.334	0.383	0.331	0.380	0.333	0.376	0.381	0.412	0.431	0.446	0.330	0.379	0.414	0.427	0.437	0.449	0.450	0.459	0.526	0.516	0.439	0.452	0.602	0.543	4.431	1.729	6.736	2.191
ETTm1	96	0.285	0.369	0.272	0.334	0.293	0.346	0.285	0.343	0.292	0.346	0.299	0.343	0.290	0.342	0.338	0.375	0.379	0.419	0.505	0.475	0.386	0.398	0.375	0.398	0.374	0.400	0.672	0.571	0.538	0.528
	192	0.319	0.393	0.310	0.358	0.332	0.369	0.324	0.366	0.332	0.372	0.335	0.365	0.332	0.369	0.374	0.387	0.426	0.441	0.553	0.496	0.459	0.444	0.408	0.410	0.400	0.407	0.795	0.669	0.658	0.592
	336	0.378	0.425	0.352	0.384	0.368	0.392	0.353	0.385	0.366	0.394	0.369	0.386	0.366	0.392	0.410	0.411	0.445	0.459	0.621	0.537	0.495	0.464	0.435	0.428	0.438	0.438	1.212	0.871	0.898	0.721
	720	0.464	0.477	0.383	0.411	0.418	0.420	0.408	0.419	0.417	0.421	0.425	0.421	0.416	0.420	0.478	0.450	0.543	0.490	0.671	0.561	0.585	0.516	0.499	0.462	0.527	0.502	1.166	0.823	1.102	0.841
	Avg	0.362	0.416	0.329	0.372	0.353	0.382	0.343	0.378	0.388	0.403	0.357	0.378	0.351	0.380	0.400	0.406	0.448	0.452	0.588	0.517	0.481	0.456	0.429	0.425	0.435	0.437	0.961	0.734	0.799	0.671
ETTm2	96	0.156	0.266	0.161	0.253	-	-	0.165	0.254	0.173	0.262	0.167	0.269	0.165	0.255	0.187	0.267	0.203	0.287	0.255	0.339	0.192	0.274	0.189	0.280	0.209	0.308	0.365	0.453	0.658	0.619
	192	0.203	0.307	0.219	0.293	-	-	0.220	0.292	0.229	0.301	0.224	0.303	0.220	0.292	0.249	0.309	0.269	0.328	0.281	0.340	0.280	0.339	0.253	0.319	0.311	0.382	0.533	0.563	1.078	0.827
	336	0.255	0.349	0.271	0.329	-	-	0.268	0.326	0.286	0.341	0.281	0.342	0.274	0.329	0.321	0.351	0.325	0.366	0.339	0.372	0.334	0.361	0.314	0.357	0.442	0.466	1.363	0.887	1.549	0.972
	720	0.342	0.417	0.352	0.379	-	-	0.350	0.380	0.378	0.401	0.397	0.421	0.362	0.385	0.408	0.403	0.421	0.415	0.433	0.432	0.417	0.413	0.414	0.413	0.675	0.587	3.379	1.338	2.631	1.242
	Avg	0.239	0.335	0.251	0.313	-	-	0.251	0.313	0.284	0.339	0.267	0.333	0.255	0.315	0.291	0.333	0.305	0.349	0.327	0.371	0.306	0.347	0.293	0.342	0.409	0.436	1.410	0.810	1.479	0.915
Weather	96	0.172	0.242	0.147	0.201	0.150	0.202	0.147	0.196	0.162	0.212	0.176	0.237	0.149	0.198	0.172	0.220	0.217	0.296	0.266	0.336	0.173	0.223	0.197	0.281	0.182	0.242	0.300	0.384	0.689	0.596
	192	0.218	0.278	0.189	0.234	0.198	0.246	0.191	0.238	0.204	0.248	0.220	0.282	0.194	0.241	0.219	0.261	0.276	0.336	0.307	0.367	0.245	0.285	0.237	0.312	0.227	0.287	0.598	0.544	0.752	0.638
	336	0.276	0.329	0.262	0.279	0.245	0.286	0.241	0.277	0.254	0.286	0.265	0.319	0.245	0.282	0.280	0.306	0.339	0.380	0.359	0.395	0.321	0.338	0.298	0.353	0.282	0.334	0.578	0.523	0.639	0.596
	720	0.339	0.373	0.304	0.316	0.324	0.342	0.313	0.329	0.326	0.337	0.333	0.362	0.314	0.334	0.365	0.359	0.403	0.428	0.419	0.428	0.414	0.410	0.352	0.288	0.352	0.386	1.059	0.741	1.130	0.792
	Avg	0.251	0.305	0.225	0.257	0.229	0.271	0.223	0.260	0.237	0.270	0.248	0.300	0.225	0.264	0.259	0.287	0.309	0.360	0.338	0.382	0.288	0.314	0.271	0.334	0.261	0.312	0.634	0.548	0.803	0.656
Electricity	96	0.145	0.247	0.131	0.224	0.132	0.223	0.128	0.223	0.139	0.238	0.140	0.237	0.129	0.222	0.168	0.272	0.193	0.308	0.201	0.317	0.169	0.273	0.187	0.304	0.207	0.307	0.274	0.368	0.312	0.402
	192	0.159	0.259	0.152	0.241	0.158	0.241	0.146	0.240	0.153	0.251	0.153	0.249	0.157	0.240	0.184	0.289	0.201	0.315	0.222	0.334	0.182	0.286	0.199	0.315	0.213	0.316	0.296	0.386	0.348	0.433
	336	0.180	0.284	0.160	0.248	0.163	0.260	0.163	0.258	0.169	0.266	0.169	0.267	0.163	0.259	0.198	0.300	0.214	0.329	0.231	0.338	0.200	0.304	0.212	0.329	0.230	0.333	0.300	0.394	0.350	0.433
	720	0.215	0.317	0.192	0.298	0.199	0.291	0.200	0.292	0.206	0.297	0.203	0.301	0.197	0.290	0.220	0.320	0.246	0.355	0.254	0.361	0.222	0.321	0.233	0.345	0.265	0.360	0.373	0.439	0.340	0.420
	Avg	0.175	0.276	0.158	0.252	0.162	0.253	0.159	0.253	0.167	0.263	0.166	0.263	0.161	0.252	0.192	0.295	0.214	0.327	0.227	0.338	0.193	0.296	0.208	0.323	0.229	0.329	0.311	0.397	0.338	0.422
Traffic	96	0.305	0.279	0.362	0.248	0.407	0.282	0.372	0.259	0.388	0.282	0.410	0.282	0.360	0.249	0.593	0.321	0.587	0.366	0.613	0.388	0.612	0.338	0.607	0.392	0.615	0.391	0.719	0.391	0.732	0.423
	192	0.313	0.274	0.374	0.247	0.423	0.287	0.391	0.265	0.407	0.290	0.423	0.287	0.379	0.256	0.617	0.336	0.604	0.373	0.616	0.382	0.613	0.340	0.621	0.399	0.601	0.382	0.696	0.379	0.733	0.420
	336	0.326	0.287	0.385	0.271	0.430	0.296	0.405	0.275	0.412	0.294	0.436	0.296	0.392	0.264	0.629	0.336	0.621	0.383	0.622	0.337	0.618	0.328	0.622	0.396	0.613	0.386	0.777	0.420	0.742	0.420
	720	0.346	0.301	0.430	0.288	0.463	0.315	0.437	0.292	0.450	0.312	0.466	0.315	0.432	0.286	0.640	0.350	0.626	0.382	0.660	0.408	0.653	0.355	0.632	0.396	0.658	0.407	0.864	0.472	0.755	0.423
	Avg	0.323	0.285	0.388	0.264	0.430	0.295	0.401	0.273	0.414	0.294	0.433	0.295	0.390	0.263	0.620	0.336	0.610	0.376	0.628	0.379	0.624	0.340	0.621	0.396	0.622	0.392	0.764	0.416	0.741	0.422

Table 17: Performance comparison with additional baselines (5% Few Shot data)

Methods		LTSM-Bundle		TIME-LLM		LLM4TS		GPT4TS		DLinear		PatchTST		TimesNet		FEDformer		Autoformer		Non-Stationary		ETSformer		LightTS		Informer		Reformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.307	0.377	0.483	0.464	0.509	0.484	0.543	0.506	0.547	0.503	0.557	0.519	0.892	0.62	5 0.593	0.529	0.681	0.570	0.952	0.650	1.169	0.832	1.483	0.91	1.225	0.812	1.198	0.795
	192	0.329	0.391	0.629	0.540	0.717	0.581	0.748	0.580	0.720	0.604	0.711	0.570	0.940	0.665	0.652	0.563	0.725	0.602	0.943	0.645	1.221	0.853	1.525	0.93	1.249	0.828	1.273	0.853
	336	0.346	0.405	0.768	0.626	0.728	0.589	0.754	0.595	0.984	0.727	0.816	0.619	0.945	0.653	0.731	0.594	0.761	0.624	0.935	0.644	1.179	0.832	1.347	0.87	1.202	0.811	1.254	0.857
	720	0.370	0.441	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Avg	0.338	0.403	0.627	0.543	0.651	0.551	0.681	0.560	0.750	0.611	0.694	0.569	0.925	0.647	0.658	0.562	0.722	0.598	0.943	0.646	1.189	0.839	1.451	0.903	1.225	0.817	1.241	0.835
ETTh2	96	0.235	0.326	0.336	0.397	0.314	0.375	0.376	0.421	0.442	0.456	0.401	0.421	0.409	0.420	0.390	0.424	0.428	0.468	0.408	0.423	0.678	0.619	2.022	1.006	3.837	1.508	3.753	1.518
	192	0.283	0.365	0.406	0.425	0.365	0.408	0.418	0.441	0.617	0.542	0.452	0.455	0.483	0.464	0.457	0.465	0.496	0.504	0.497	0.468	0.845	0.697	3.534	1.348	3.975	1.933	3.516	1.473
	336	0.320	0.401	0.405	0.432	0.398	0.432	0.408	0.439	1.424	0.849	0.464	0.469	0.499	0.479	0.477	0.483	0.486	0.496	0.507	0.481	0.905	0.727	4.063	1.451	3.956	1.520	3.312	1.427
	720	0.378	0.456	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Avg	0.304	0.387	0.382	0.418	0.359	0.405	0.400	0.433	0.694	0.577	0.439	0.448	0.439	0.448	0.463	0.454	0.441	0.457	0.470	0.489	0.809	0.681	3.206	1.268	3.922	1.653	3.527	1.472
ETTm1	96	0.285	0.369	0.316	0.377	0.349	0.379	0.386	0.405	0.332	0.374	0.399	0.414	0.606	0.518	0.628	0.544	0.726	0.578	0.823	0.587	1.031	0.747	1.048	0.733	1.130	0.775	1.234	0.798
	192	0.319	0.393	0.450	0.464	0.374	0.394	0.440	0.438	0.358	0.390	0.441	0.436	0.681	0.539	0.666	0.566	0.750	0.591	0.844	0.591	1.087	0.766	1.097	0.756	1.150	0.788	1.287	0.839
	336	0.378	0.425	0.450	0.424	0.411	0.417	0.485	0.459	0.402	0.416	0.499	0.467	0.786	0.597	0.807	0.628	0.851	0.659	0.870	0.603	1.138	0.787	1.147	0.775	1.198	0.809	1.288	0.842
	720	0.464	0.477	0.483	0.471	0.516	0.479	0.577	0.499	0.511	0.489	0.767	0.587	0.796	0.593	0.822	0.633	0.857	0.655	0.893	0.611	1.245	0.831	1.200	0.799	1.175	0.794	1.247	0.828
	Avg	0.362	0.416	0.425	0.434	0.412	0.417	0.472	0.450	0.400	0.417	0.526	0.476	0.717	0.561	0.730	0.592	0.796	0.620	0.857	0.598	1.125	0.782	1.123	0.765	1.163	0.791	1.264	0.826
ETTm2	96	0.156	0.266	0.174	0.261	0.192	0.273	0.199	0.280	0.236	0.326	0.206	0.288	0.220	0.299	0.229	0.320	0.232	0.322	0.238	0.316	0.404	0.485	1.108	0.772	3.599	1.478	3.883	1.545
	192	0.203	0.307	0.215	0.287	0.249	0.309	0.256	0.316	0.306	0.373	0.264	0.324	0.311	0.361	0.394	0.361	0.291	0.357	0.298	0.349	0.479	0.521	1.317	0.850	3.578	1.475	3.553	1.484
	336	0.255	0.349	0.273	0.330	0.301	0.342	0.318	0.353	0.380	0.423	0.334	0.367	0.338	0.366	0.378	0.427	0.478	0.517	0.353	0.380	0.552	0.555	1.415	0.879	3.561	1.473	3.446	1.460
	720	0.342	0.417	0.433	0.412	0.402	0.405	0.460	0.436	0.674	0.583	0.454	0.432	0.509	0.465	0.523	0.510	0.553	0.538	0.475	0.445	0.701	0.627	1.822	0.984	3.896	1.533	3.445	1.460
	Avg	0.239	0.335	0.274	0.323	0.286	0.332	0.308	0.346	0.399	0.426	0.314	0.352	0.344	0.372	0.381	0.404	0.388	0.433	0.341	0.372	0.534	0.547	1.415	0.871	3.658	1.489	3.581	1.487
Weather	96	0.172	0.242	0.172	0.263	0.173	0.227	0.175	0.230	0.184	0.242	0.171	0.224	0.207	0.253	0.229	0.309	0.227	0.299	0.215	0.252	0.218	0.295	0.230	0.285	0.497	0.497	0.406	0.435
	192	0.218	0.278	0.224	0.271	0.218	0.265	0.227	0.276	0.228	0.283	0.230	0.277	0.272	0.307	0.265	0.317	0.278	0.333	0.290	0.307	0.294	0.331	0.274	0.323	0.620	0.545	0.446	0.450
	336	0.276	0.329	0.282	0.321	0.276	0.310	0.286	0.322	0.279	0.322	0.294	0.326	0.313	0.328	0.353	0.392	0.351	0.393	0.353	0.348	0.359	0.398	0.318	0.355	0.649	0.547	0.465	0.459
	720	0.339	0.373	0.366	0.381	0.355	0.366	0.366	0.379	0.364	0.388	0.384	0.387	0.400	0.385	0.391	0.394	0.387	0.389	0.452	0.407	0.461	0.461	0.401	0.418	0.570	0.522	0.471	0.468
	Avg	0.251	0.305	0.260	0.309	0.251	0.292	0.263	0.301	0.263	0.308	0.269	0.303	0.298	0.318	0.309	0.353	0.310	0.353	0.327	0.328	0.333	0.371	0.305	0.345	0.584	0.527	0.447	0.453
Electricity	96	0.145	0.247	0.147	0.242	0.139	0.235	0.143	0.241	0.150	0.251	0.145	0.244	0.315	0.389	0.235	0.322	0.297	0.367	0.484	0.518	0.697	0.638	0.639	0.609	1.265	0.919	1.414	0.855
	192	0.159	0.259	0.158	0.241	0.155	0.249	0.159	0.255	0.163	0.263	0.163	0.260	0.318	0.396	0.247	0.341	0.308	0.375	0.501	0.531	0.718	0.648	0.772	0.678	1.298	0.939	1.240	0.919
	336	0.180	0.284	0.178	0.277	0.174	0.269	0.179	0.274	0.175	0.278	0.183	0.281	0.340	0.415	0.267	0.356	0.354	0.411	0.574	0.578	0.758	0.667	0.901	0.745	1.302	0.942	1.253	0.921
	720	0.215	0.317	0.224	0.312	0.222	0.310	0.233	0.323	0.219	0.311	0.233	0.323	0.635	0.613	0.318	0.394	0.426	0.466	0.952	0.786	1.028	0.788	1.200	0.871	1.259	0.919	1.249	0.921
	Avg	0.175	0.276	0.179	0.268	0.173	0.266	0.178	0.273	0.176	0.275	0.181	0.277	0.402	0.453	0.266	0.353	0.346	0.404	0.627	0.603	0.800	0.685	0.878	0.725	1.281	0.929	1.289	0.904
Traffic	96	0.305	0.279	0.414	0.291	0.401	0.285	0.419	0.298	0.427	0.304	0.404	0.286	0.854	0.492	0.670	0.421	0.795	0.481	1.468	0.821	1.643	0.855	1.157	0.636	1.557	0.821	1.586	0.841
	192	0.313	0.274	0.419	0.291	0.418	0.293	0.434	0.305	0.447	0.315	0.412	0.294	0.894	0.517	0.653	0.405	0.837	0.503	1.509	0.838	1.856	0.928	1.688	0.848	1.596	0.834	1.602	0.844
	336	0.326	0.287	0.437	0.314	0.436	0.308	0.449	0.313	0.478	0.333	0.439	0.310	0.853	0.471	0.707	0.445	0.867	0.523	1.602	0.860	2.080	0.999	1.826	0.903	1.621	0.841	1.668	0.868
	720	0.346	0.301	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Avg	0.323	0.285	0.423	0.298	0.418	0.295	0.434	0.305	0.450	0.317	0.418	0.296	0.867	0.493	0.676	0.423	0.833	0.502	1.526	0.839	1.859	0.927	1.557	0.795	1.591	0.832	1.618	0.851

Table 18: Results of different backbones, training paradigms, and prompting strategies.

Datasets		**ETTh1**		**ETTh2**		**ETTm1**		**ETTm2**		**Traffic**		**Weather**		**Exchange**		**ECL**	
Metric		**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
	96	0.354	0.415	0.259	0.350	0.551	0.507	0.215	0.319	0.393	0.377	0.222	0.292	0.108	0.246	0.250	0.342
From scratch	192	0.364	0.421	0.537	0.505	0.231	0.331	0.235	0.335	0.373	0.356	0.246	0.309	0.143	0.288	0.231	0.331
+ GPT-Medium	336	0.359	0.420	0.321	0.402	0.423	0.454	0.267	0.360	0.357	0.329	0.283	0.335	0.207	0.344	0.217	0.323
+ TS prompt	720	0.357	0.430	0.372	0.449	0.398	0.444	0.360	0.434	0.347	0.311	0.342	0.388	0.358	0.461	0.211	0.317
	Avg	0.358	0.421	0.372	0.427	0.401	0.434	0.269	0.362	0.367	0.343	0.273	0.331	0.204	0.335	0.227	0.328
	96	0.453	0.483	0.350	0.422	0.757	0.613	0.338	0.420	0.659	0.552	0.343	0.398	0.223	0.353	0.605	0.507
From scratch	192	0.422	0.470	0.348	0.423	0.708	0.601	0.326	0.415	0.509	0.475	0.323	0.388	0.212	0.352	0.352	0.403
+ GPT-Medium	336	0.481	0.502	0.449	0.487	0.938	0.701	0.483	0.506	0.562	0.496	0.457	0.474	0.430	0.502	0.729	0.456
+ Text prompt	720	0.437	0.482	0.396	0.461	0.634	0.563	0.408	0.459	0.536	0.463	0.415	0.434	0.457	0.517	0.460	0.438
	Avg	0.448	0.484	0.385	0.448	0.759	0.619	0.389	0.450	0.566	0.496	0.384	0.423	0.330	0.431	0.537	0.451
	96	0.323	0.392	0.243	0.341	0.394	0.437	0.185	0.301	0.334	0.321	0.200	0.279	0.098	0.236	0.197	0.291
From scratch	192	0.332	0.399	0.275	0.362	0.369	0.426	0.204	0.313	0.333	0.306	0.219	0.286	0.127	0.268	0.195	0.290
+ GPT-Small	336	0.345	0.405	0.317	0.394	0.352	0.415	0.263	0.364	0.324	0.288	0.265	0.323	0.179	0.321	0.181	0.282
+ TS prompt	720	0.362	0.432	0.364	0.447	0.389	0.439	0.324	0.402	0.341	0.300	0.339	0.377	0.333	0.457	0.207	0.309
	Avg	0.340	0.407	0.300	0.386	0.376	0.429	0.244	0.345	0.333	0.304	0.256	0.316	0.184	0.320	0.195	0.293
	96	0.317	0.383	0.240	0.334	0.431	0.443	0.178	0.285	0.315	0.293	0.188	0.257	0.083	0.208	0.161	0.265
Fully tune	192	0.355	0.413	0.285	0.370	0.479	0.482	0.221	0.323	0.352	0.336	0.238	0.304	0.132	0.277	0.213	0.311
+ GPT-Small	336	0.357	0.414	0.302	0.388	0.486	0.486	0.252	0.346	0.339	0.310	0.275	0.326	0.196	0.336	0.203	0.302
+ TS prompt	720	0.363	0.434	0.361	0.442	0.479	0.483	0.345	0.420	0.350	0.313	0.345	0.384	0.426	0.496	0.224	0.326
	Avg	0.348	0.411	0.297	0.384	0.469	0.473	0.249	0.343	0.339	0.313	0.261	0.317	0.209	0.329	0.200	0.301
	96	0.305	0.377	0.226	0.320	0.276	0.360	0.143	0.253	0.305	0.279	0.162	0.227	0.060	0.178	0.144	0.246
Fully tune	192	0.335	0.397	0.278	0.359	0.314	0.389	0.191	0.295	0.315	0.283	0.212	0.275	0.118	0.253	0.161	0.261
+ GPT-Small	336	0.348	0.406	0.310	0.392	0.344	0.411	0.239	0.339	0.323	0.285	0.266	0.318	0.198	0.333	0.175	0.277
+ Text prompt	720	0.371	0.454	0.364	0.446	0.404	0.452	0.352	0.405	0.344	0.305	0.332	0.366	0.379	0.470	0.208	0.309
	Avg	0.340	0.409	0.294	0.379	0.334	0.403	0.231	0.323	0.322	0.288	0.243	0.296	0.189	0.308	0.172	0.273
	96	0.301	0.372	0.229	0.320	0.261	0.346	0.149	0.266	0.300	0.268	0.163	0.230	0.058	0.173	0.141	0.241
Fully tune	192	0.332	0.397	0.290	0.368	0.288	0.370	0.204	0.303	0.316	0.282	0.215	0.282	0.133	0.277	0.158	0.258
+ GPT-Medium	336	0.351	0.412	0.316	0.392	0.343	0.413	0.294	0.376	0.328	0.295	0.281	0.332	0.224	0.369	0.175	0.276
+ Text prompt	720	0.368	0.436	0.378	0.452	0.371	0.431	0.492	0.494	0.344	0.303	0.350	0.385	0.321	0.442	0.207	0.308
	Avg	0.338	0.404	0.303	0.383	0.316	0.390	0.285	0.360	0.322	0.287	0.252	0.307	0.184	0.315	0.170	0.271
	96	0.320	0.387	0.242	0.330	0.490	0.477	0.191	0.290	0.346	0.326	0.212	0.270	0.134	0.269	0.185	0.300
Fully tune	192	0.342	0.403	0.270	0.352	0.376	0.423	0.196	0.287	0.355	0.327	0.236	0.286	0.173	0.305	0.204	0.316
+ GPT-Medium	336	0.348	0.409	0.284	0.367	0.530	0.501	0.253	0.335	0.379	0.345	0.298	0.331	0.311	0.421	0.224	0.334
+ Text prompt	720	0.368	0.433	0.424	0.479	0.375	0.429	0.361	0.423	OOM	OOM	OOM	OOM	0.333	0.457	0.206	0.307
	Avg	0.344	0.408	0.305	0.382	0.443	0.458	0.250	0.334	0.360	0.333	0.249	0.295	0.238	0.363	0.205	0.314
	96	0.296	0.371	0.234	0.328	0.309	0.381	0.150	0.263	0.299	0.278	0.175	0.248	0.073	0.204	0.145	0.249
Fully tune	192	0.318	0.386	0.273	0.355	0.301	0.381	0.190	0.293	0.311	0.278	0.212	0.279	0.129	0.271	0.164	0.266
+ Phi-2	336	0.337	0.402	0.311	0.389	0.346	0.419	0.283	0.381	0.323	0.290	0.282	0.345	0.233	0.374	0.179	0.281
+ TS prompt	720	0.372	0.445	0.317	0.407	0.404	0.461	0.439	0.484	0.347	0.305	0.354	0.382	0.404	0.501	0.218	0.319
	Avg	0.331	0.401	0.284	0.370	0.340	0.411	0.265	0.355	0.320	0.288	0.256	0.313	0.210	0.337	0.176	0.279
	96	0.296	0.371	0.234	0.328	0.309	0.381	0.150	0.263	0.299	0.278	0.175	0.248	0.073	0.204	0.145	0.249
Fully tune	192	0.319	0.385	0.269	0.355	0.309	0.383	0.188	0.295	0.307	0.275	0.212	0.283	0.134	0.281	0.161	0.262
+ Pi-2	336	0.337	0.402	0.311	0.389	0.346	0.419	0.283	0.381	0.323	0.290	0.282	0.345	0.233	0.374	0.179	0.281
+ Text prompt	720	0.356	0.430	0.359	0.442	0.392	0.454	0.383	0.451	0.345	0.302	0.345	0.377	0.561	0.606	0.212	0.315
	Avg	0.327	0.397	0.293	0.378	0.339	0.409	0.251	0.347	0.318	0.286	0.254	0.313	0.250	0.366	0.174	0.277
	96	0.362	0.419	0.273	0.363	0.589	0.533	0.225	0.332	0.428	0.396	0.224	0.293	0.129	0.274	0.227	0.333
LoRA-dim-16	192	0.394	0.444	0.312	0.397	0.582	0.531	0.259	0.361	0.502	0.437	0.339	0.280	0.200	0.345	0.257	0.358
+ GPT-Medium	336	0.403	0.457	0.321	0.413	0.560	0.532	0.293	0.392	0.547	0.457	0.320	0.369	0.266	0.409	0.291	0.386
+ TS prompt	720	0.444	0.499	0.366	0.451	0.576	0.547	0.355	0.436	0.660	0.519	0.369	0.406	0.457	0.532	0.406	0.479
	Avg	0.401	0.455	0.318	0.406	0.577	0.536	0.283	0.380	0.534	0.452	0.313	0.337	0.263	0.390	0.295	0.389
	96	0.365	0.422	0.270	0.361	0.596	0.593	0.222	0.329	0.438	0.408	0.223	0.294	0.117	0.259	0.233	0.341
LoRA-dim-32	192	0.401	0.449	0.314	0.398	0.594	0.537	0.261	0.363	0.503	0.443	0.281	0.340	0.204	0.346	0.259	0.361
+ GPT-Medium	336	0.403	0.457	0.321	0.413	0.563	0.533	0.294	0.393	0.547	0.459	0.321	0.370	0.267	0.410	0.294	0.390
+ TS prompt	720	0.444	0.498	0.367	0.452	0.572	0.545	0.357	0.437	0.647	0.513	0.369	0.406	0.454	0.530	0.399	0.473
	Avg	0.403	0.457	0.318	0.406	0.581	0.552	0.283	0.380	0.534	0.456	0.298	0.352	0.260	0.386	0.296	0.391
	96	0.377	0.431	0.284	0.376	0.603	0.538	0.239	0.348	0.462	0.423	0.244	0.313	0.154	0.302	0.242	0.348
LoRA-dim-16	192	0.394	0.445	0.313	0.400	0.578	0.530	0.263	0.367	0.511	0.441	0.284	0.344	0.203	0.351	0.262	0.363
+ GPT-Medium	336	0.412	0.465	0.325	0.417	0.571	0.538	0.299	0.397	0.567	0.471	0.323	0.373	0.267	0.413	0.308	0.402
+ Word prompt	720	0.448	0.501	0.368	0.452	0.582	0.550	0.357	0.437	0.672	0.526	0.370	0.407	0.452	0.529	0.416	0.486
	Avg	0.408	0.461	0.322	0.411	0.583	0.539	0.289	0.387	0.553	0.465	0.305	0.359	0.269	0.399	0.307	0.400
	96	0.365	0.423	0.276	0.367	0.590	0.533	0.230	0.337	0.449	0.410	0.234	0.305	0.133	0.277	0.237	0.343
LoRA-dim-32	192	0.400	0.449	0.311	0.397	0.572	0.527	0.261	0.364	0.515	0.447	0.284	0.345	0.207	0.352	0.265	0.366
+ GPT-Medium	336	0.410	0.463	0.324	0.416	0.570	0.538	0.299	0.398	0.563	0.469	0.323	0.373	0.268	0.414	0.305	0.399
+ Word prompt	720	0.447	0.500	0.368	0.453	0.589	0.553	0.359	0.459	0.664	0.522	0.370	0.407	0.453	0.530	0.414	0.486
	Avg	0.406	0.459	0.320	0.408	0.580	0.538	0.287	0.389	0.547	0.462	0.303	0.357	0.265	0.393	0.305	0.398

Table 19: Results of different backbones.

Datasets		ETTh1		ETTh2		ETTm1		ETTm2		Traffic		Weather		Exchange		ECL	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Fully tune + GPT-Large + TS prompt	96	0.297	0.368	0.224	0.319	0.277	0.360	0.147	0.259	0.304	0.280	0.168	0.240	0.069	0.192	0.148	0.251
	192	0.328	0.391	0.279	0.362	0.300	0.377	0.207	0.311	0.315	0.279	0.212	0.277	0.121	0.264	0.159	0.258
	336	0.347	0.407	0.365	0.420	0.322	0.397	0.284	0.364	0.324	0.287	0.291	0.341	0.356	0.461	0.174	0.276
	720	0.358	0.427	0.430	0.478	0.358	0.421	0.436	0.467	0.348	0.303	0.370	0.388	0.424	0.525	0.207	0.308
	Avg	0.333	0.398	0.325	0.395	0.314	0.389	0.269	0.350	0.323	0.287	0.260	0.311	0.242	0.360	0.172	0.273
Fully tune + GPT-Medium + TS prompt	96	0.307	0.377	0.235	0.326	0.285	0.369	0.156	0.266	0.305	0.278	0.172	0.242	0.065	0.186	0.145	0.247
	192	0.329	0.391	0.283	0.365	0.319	0.393	0.203	0.307	0.313	0.274	0.218	0.278	0.115	0.248	0.159	0.259
	336	0.346	0.405	0.320	0.401	0.378	0.425	0.255	0.349	0.326	0.287	0.276	0.329	0.206	0.339	0.180	0.284
	720	0.370	0.441	0.378	0.456	0.464	0.477	0.342	0.417	0.346	0.301	0.339	0.373	0.409	0.487	0.215	0.317
	Avg	0.338	0.403	0.304	0.387	0.362	0.416	0.239	0.335	0.323	0.285	0.251	0.305	0.199	0.315	0.175	0.276
Fully tune + GPT-Small + TS prompt	96	0.317	0.383	0.240	0.334	0.431	0.443	0.178	0.285	0.315	0.293	0.188	0.257	0.083	0.208	0.161	0.265
	192	0.355	0.413	0.285	0.370	0.479	0.482	0.221	0.323	0.352	0.336	0.238	0.304	0.132	0.277	0.213	0.311
	336	0.357	0.414	0.302	0.388	0.486	0.486	0.252	0.346	0.339	0.310	0.275	0.326	0.196	0.336	0.203	0.302
	720	0.363	0.434	0.361	0.442	0.479	0.483	0.345	0.420	0.350	0.313	0.345	0.384	0.426	0.496	0.224	0.326
	Avg	0.348	0.411	0.297	0.384	0.469	0.473	0.249	0.343	0.339	0.313	0.261	0.317	0.209	0.329	0.200	0.301
Fully tune + Phi-2 + TS prompt	96	0.296	0.371	0.234	0.328	0.309	0.381	0.150	0.263	0.299	0.278	0.175	0.248	0.073	0.204	0.145	0.249
	192	0.318	0.386	0.273	0.355	0.301	0.381	0.190	0.293	0.311	0.278	0.212	0.279	0.129	0.271	0.164	0.266
	336	0.337	0.402	0.311	0.389	0.346	0.419	0.283	0.381	0.323	0.290	0.282	0.345	0.233	0.374	0.179	0.281
	720	0.372	0.445	0.317	0.407	0.404	0.461	0.439	0.484	0.347	0.305	0.354	0.382	0.404	0.501	0.218	0.319
	Avg	0.331	0.401	0.284	0.370	0.340	0.411	0.265	0.355	0.320	0.288	0.256	0.313	0.210	0.337	0.176	0.279

Table 20: Results of different down-sampling ratios. Experiments with GPT-Medium as backbones, TS prompt, and fully tuning paradigm.

Datasets		ETTh1		ETTh2		ETTm1		ETTm2		Traffic		Weather		ECL	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Downsample Ratio = 40	96	0.4536	0.4787	0.3395	0.4097	0.7441	0.6068	0.3229	0.4055	0.6619	0.5636	0.3538	0.3979	0.6118	0.499
	192	0.4648	0.4935	0.386	0.4507	0.7659	0.6336	0.3797	0.4604	0.6644	0.5537	0.4106	0.454	0.6756	0.4915
	336	0.6629	0.6167	0.5982	0.5916	1.1366	0.8212	0.6904	0.6455	1.0706	0.7785	0.7359	0.65	1.6485	0.7188
	720	1.0518	0.8133	0.8802	0.7588	1.8609	1.1304	1.2011	0.9073	1.7276	1.062	1.3176	0.9268	3.9988	1.0223
	Avg	0.343	0.408	0.307	0.390	0.353	0.412	0.234	0.331	0.344	0.314	0.253	0.306	0.210	0.330
Downsample Ratio = 20	96	0.307	0.3767	0.2349	0.3263	0.285	0.3687	0.1555	0.2657	0.3053	0.2778	0.1717	0.2416	0.1447	0.2468
	192	0.3288	0.3908	0.2826	0.3649	0.3193	0.3925	0.2032	0.3069	0.3132	0.2744	0.2177	0.2782	0.1587	0.2588
	336	0.346	0.405	0.3198	0.4005	0.3782	0.4254	0.2549	0.3492	0.3263	0.2869	0.2761	0.3287	0.1803	0.2835
	720	0.3704	0.4405	0.378	0.456	0.4638	0.4773	0.3422	0.4172	0.3456	0.301	0.3386	0.3732	0.2151	0.3167
	Avg	0.338	0.404	0.303	0.383	0.316	0.390	0.285	0.360	0.322	0.287	0.252	0.307	0.184	0.315
Downsample Ratio = 10	96	0.2975	0.3698	0.2268	0.3175	0.2633	0.3462	0.1463	0.2583	0.2961	0.2626	0.2583	0.2247	0.1406	0.2411
	192	0.3293	0.3896	0.2848	0.3674	0.3286	0.3991	0.1995	0.3014	0.3089	2673	0.2151	0.2786	0.1568	0.2564
	336	0.3461	0.4039	0.3097	0.3938	0.3593	0.4198	0.259	0.3505	0.3206	0.2785	0.2651	0.3155	0.1744	0.2747
	720	0.3676	0.4333	0.4101	0.4738	0.4096	0.4505	0.3632	0.4242	0.3428	0.2983	0.3462	0.38	0.2099	0.3101
	Avg	0.335	0.402	0.321	0.392	0.341	0.399	0.235	0.332	0.312	0.274	0.252	0.308	0.232	0.360

Are Classification Robustness and Explanation Robustness Really Strongly Correlated? An Analysis Through Input Loss Landscape

Tiejin Chen
Arizona State University
tchen169@asu.edu

Wenwang Huang
Independent Researcher
Weistrasshww@gmail.com

Linsey Pang
Paypal AI
panglinsey@gmail.com

Dongsheng Luo
Florida International University
dluo@fiu.edu

Hua Wei
Arizona State University
hua.wei@asu.edu

ABSTRACT

This paper looks into the critical area of deep learning robustness and challenges the common belief that classification robustness and explanation robustness in image classification systems are inherently correlated. Through a novel evaluation approach leveraging clustering for efficient assessment of explanation robustness, we demonstrate that enhancing explanation robustness does not necessarily flatten the input loss landscape with respect to explanation loss - contrary to flattened loss landscapes indicating better classification robustness. To further investigate this contradiction, a training method designed to adjust the loss landscape with respect to explanation loss is proposed. Through the new training method, we uncover that although such adjustments can impact the robustness of explanations, they do not have an influence on the robustness of classification. These findings not only challenge the previous assumption of a strong correlation between the two forms of robustness but also pave new pathways for understanding the relationship between loss landscape and explanation loss. Codes are provided in the supplement.

1. INTRODUCTION

Understanding the relationship between classification robustness and explanation robustness is critical for deploying reliable machine learning systems in real-world applications. In medical diagnostics, autonomous vehicles, and financial fraud detection, models must not only maintain accurate predictions under adversarial attacks (known as classification robustness [34]) but also preserve consistent, interpretable rationales (e.g., highlighted image regions) for their decisions during adversarial attacks (known as explanation robustness). Prior work has widely assumed these two properties are positively correlated [5, 22], leading to potential security gaps if they are not directly linked.

Our investigation reveals a fundamental challenge to this paradigm: *improving classification robustness provides no guarantee of enhanced explanation robustness*, challenging a key assumption in adversarial learning. This occurs because explanations, such as saliency maps, are themselves susceptible to adversarial perturbations [14, 15]. As illustrated in Figure 1, small adversarial perturbations can significantly alter explanation maps while leaving the model's classification unchanged. This raises a critical question about whether adversarial robustness in classification translates to robustness in model interpretability.

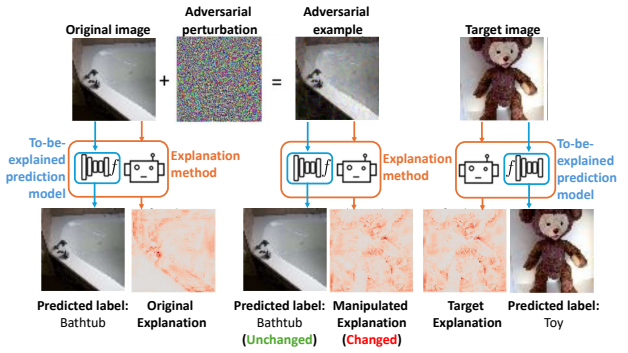


Figure 1: Example of an adversarial explanation attack. While the to-be-explained model f gives the same predicted label after adding adversarial noise, the explanation saliency map can be manipulated to the target saliency map, leading to wrong explanations.

To better understand robustness, one key approach is analyzing the input loss landscape [29]. Prior work has shown that a flatter input loss landscape with respect to classification loss indicates stronger classification robustness [57, 29]. As visualized in Figure 2, adversarially trained models, which show increased classification robustness, exhibit a flatter input loss landscape compared to normally trained models. Since classification robustness is linked to a flatter loss landscape, a natural question arises: *Does increasing explanation robustness lead to a flatter input loss landscape for explanation loss?*

Surprisingly, this paper shows that increasing explanation robustness does not flatten the input loss landscape with respect to explanation loss. To systematically analyze the relationship between explanation robustness and input loss landscapes, we generate models with varying levels of explanation robustness using adversarial training methods [60]

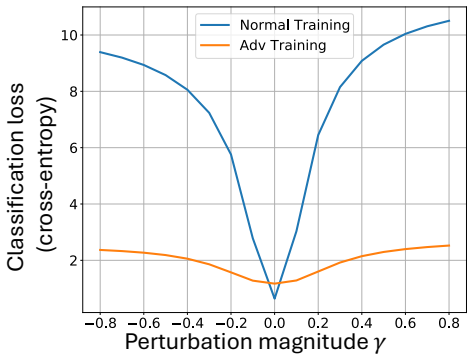


Figure 2: Comparison of input loss landscapes between normal and adversarially trained models on CIFAR-10. Adversarially trained models show flatter loss landscapes and increased classification robustness.

that allow fine-grained control over classification robustness. Our findings (detailed in Figure 4) contradict previous assumptions that classification and explanation robustness are inherently linked.

To further investigate this phenomenon, we reverse the question: *Does a flatter input loss landscape for explanation loss improve explanation robustness?* This paper shows that the answer is no. Flattening the input loss landscape for explanation loss actually decreases explanation robustness. To confirm this, we propose **S**eparate **E**xplanation Robustness via **P**GD (`SEP`) to explicitly control the input loss landscape w.r.t explanation robustness. By regularizing on the input loss landscape w.r.t explanation robustness, `SEP` reduces explanation robustness while leaving classification robustness unchanged. Extensive results on different model architectures, datasets, and explanation methods demonstrate the effectiveness of `SEP` in decoupling explanation robustness and classification robustness. This demonstration challenges the widely held belief that classification and explanation robustness are positively correlated. In summary, the key contributions of this paper are as follows:

- We introduce a clustering-based sampling method to efficiently evaluate explanation robustness.
- We use TRADES [60] to control classification robustness and visualize the input loss landscape with respect to explanation loss, revealing that increasing explanation robustness does not flatten the input loss landscape, which contradicts with classification robustness.
- We develop a novel training method that flattens the input loss landscape for explanation loss and show that, contrary to prior assumptions, this reduces explanation robustness, suggesting that explanation and classification robustness are not strongly correlated.

2. RELATED WORK

Adversarial Attack and Adversarial Training (AT). It has been proven that deep learning models are vulnerable to adversarial examples [48, 17, 8], where noise that is imperceptible to humans, when added to the original inputs, can lead to the misclassification of models. Projected Gradient Descent (PGD) [34] is one of the most popular methods that generate such a noise or evaluate models' classification robustness by calculating accuracy under its attack. Many methods [37, 58, 30, 45, 7] have been introduced to defend against adversarial attacks, while they do not involve a training process and may be vulnerable to adaptive attack [2]. Goodfellow et al. [17] first introduced adversarial training (AT), which trains a model from scratch with adversarial samples and proves its performance, including adversarial competitions [40, 32, 34, 6]. In this paper, we focus on classification robustness increased by AT methods like Madry adversarial training [34] and TRADES [60]. Many works tend to increase the performance of AT through external datasets [19, 9, 53], metric learning [35], self-supervised learning [11], ensemble learning [51], label smoothing [12], and Taylor Expansion [23]. Wu et al. [55] found that obtaining a flat loss landscape can help increase classification robustness, which inspired the ideas in this paper.

Explanation Robustness. Saliency maps [42, 41, 3, 39] are widely used to explain image-related tasks in deep learning, and our focus is on the robustness of these explanations. However, similar to an adversarial attack, it is possible to find an adversarial noise on original images so that it can easily manipulate the saliency maps without changing classification results in both white-box [14, 15, 20, 44] and black-box settings [49]. Zhang et al. [61] further introduced a new method that can attack both saliency maps and classification results. In order to evaluate the explanation robustness, Wicker et al. [54] introduced the max-sensitivity and average-sensitivity of saliency maps. Alvarez et al. [1] estimated explanation robustness by the Local Lipschitz of interpretation, while Tamam et al. [49] directly used attack loss to evaluate explanation robustness. In this paper, we use attack loss based on the proposed cluster method to evaluate explanation robustness.

Several works have also aimed to improve explanation robustness. Chen et al. [10] introduced a regularization term during training to make the explanation more robust. Boopathy et al. [5] improved the performance by training with noisy labels. Tang et al. [50] proposed a first-order gradient-based approach to reduce computational training costs. Huang et al. [22] explored genetic algorithms to optimize for stronger explanation robustness.

Relationship between Classification Robustness and Explanation. It has been suggested that improved interpretability contributes to classification robustness [43, 52], implying a correlation between high-quality saliency maps and classification performance. Studies have demonstrated that models robust to explanation attacks tend to be more resistant to classification attacks as well, supporting the idea that explanation robustness benefits classification robustness [5, 50, 22]. However, our work challenges this assumption. We show that adversarial training with TRADES [60] can improve explanation robustness while sometimes enhancing classification robustness. Yet, further analysis reveals that these two aspects of robustness are not inherently linked—classification robustness does not necessarily ensure explanation robustness. This finding suggests a fundamental disconnect between the two, which we investigate in depth.

3. ANALYSIS AND METHODOLOGY

This section establishes a systematic framework to investi-

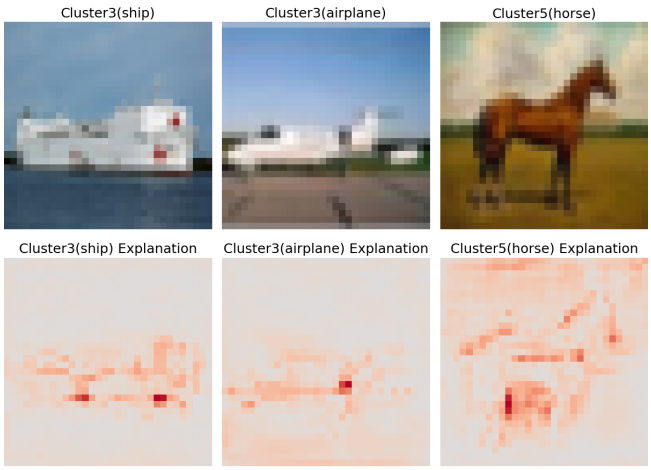


Figure 3: Explanation consistency within clustered groups: (a) Cluster 3 exhibits focused attention on the shape of machines though they have different classification labels, while (b) Cluster 5 highlights the shape of horses.

gate the relationship between classification robustness and explanation robustness, ultimately developing our proposed solution (SEP) to address observed contradictions. We create a controlled level of explanation robustness using TRADES, then quantify explanation robustness through clustered evaluation, analyze loss landscape paradoxes, and finally introduce SEP for targeted landscape engineering.

3.1 Classification and Explanation Robustness

3.1.1 Classification Robustness

Our systematic investigation begins by establishing precise control over model robustness characteristics through the TRADES framework [60]. TRADES provides fine-grained control of classification robustness via its parameterized loss function and the parameter α:

$$\mathcal{L}_{\text{TRADES}} = \underbrace{\mathcal{L}_{\text{sc}}(f(x), y)}_{\text{Standard Classification Loss}} + \alpha \underbrace{\mathcal{L}_{\text{adv}}(f(x), f(x_{\text{adv}}))}_{\text{Adversarial Regularization}} \tag{1}$$

where $f(x) \in \mathbb{R}^C$ denotes model outputs for input x with C classes, x_{adv} is generated via projected gradient descent (PGD) attack [34]: $x_{adv} = x + \epsilon_{adv}$ with $|\epsilon_{adv}|_\infty \leq \xi$, where the ξ is a pre-defined constraint. The adversarial regularization term $\mathcal{L}_{\text{adv}}$ measures the KL divergence between original and adversarial predictions. The α parameter explicitly controls the tradeoff between clean accuracy and adversarial robustness, enabling controlled robustness levels.

Measuring Classification Robustness. Following existing work [34, 60], classification robustness is commonly measured by adversarial accuracy (Acc_{adv}), which is the accuracy under adversarial attack on classification:

$$Acc_{\text{adv}} := \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left(f(x_{\text{adv},i}) = y_i\right), \tag{2}$$

Where $\mathbb{I}$ is the indicator function. Compared with the clean accuracy ($Acc_{\text{clean}} := \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(f(x_i) = y_i)$) which measures classification performance, adversarial accuracy Acc_{adv} measures how good a model is against adversarial attacks on classification. Therefore, a higher Acc_{adv} indicates a better classification robustness.

3.1.2 Explanation Robustness

Previous works on adversarial attacks have extended the framework to explanation through finding a perturbation δ^* with constrained optimization [49]:

$$\delta^* = \arg\min_{|\delta|_\infty \leq \xi} \|I_f(x_v + \delta) - I_f(x_t)\|_2 \tag{3}$$

where x_v represents the victim (original) image, x_t is the target image with desired explanation pattern, f is the to-be-explained model, and $I_f(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ denotes an explanation method (e.g., Grad-CAM [39] or Saliency Maps [42]) that explains the behavior of model f on an input image.
Explanation robustness is commonly measured by the following definition of attack loss:

Definition 1 (Explanation Attack Loss)**.** *Given a victim image x_v, a target image x_t with desired explanation pattern, a to-be-explained model f and an explanation method $I_f(\cdot)$, the explanation attack loss is $\mathcal{L}_e(x_v, x_t) = \|I_f(x_v + \delta) - I_f(x_t)\|_2^2$, where δ is the optimization target.*

Since the ultimate goal of adversarial attacks on explanation is to manipulate the explanation of the victim image to resemble that of the target image, a higher explanation attack loss indicates a bigger difference between two explanations $I_f(x_v + \delta)$ and $I_f(x_t)$, which means a less successful attack.

Generating $\mathcal{D}_e$ with cluster-based sampling. In order to estimate the explanation robustness of a model f, existing methods use evaluation set $\mathcal{D}_e = \{(x_v, x_t) | x_v, x_t \in \mathcal{D}_{\text{origin}}$, where $\mathcal{D}_{\text{origin}}$ is the images in the original classification dataset. As the number of samples in $\mathcal{D}_{\text{origin}}$ increases, the size of $\mathcal{D}_e$ grows quadratically, which would be computationally infeasible.
To ensure comprehensive coverage of the explanation space while maintaining computational traceability, we introduce a cluster-based sampling protocol grounded in explanation consistency. Specifically, the protocol operates through three systematic steps:

1. **Feature Extraction**: Compute high-level representations using ResNet18's penultimate layer [18], capturing semantically meaningful features for explanation consistency

2. **Clustering**: Apply clustering algorithm. In our experiment, we use K-means [33] with $k = 10$ on CIFAR10.

3. **Representative Sampling**: Select N prototypes per cluster and forms evaluation set $\mathcal{D}_e$ for each cluster. In our experiment we set $N = 15$, forming evaluation set $\mathcal{D}_e$ with 150 images for 10 clusters and $150 \times 149 = 22,350$ unique (x_v, x_t) pairs in total.

As shown in Figure 3, images from the same cluster have similar explanations. We also report the explanation loss for intra-cluster and inter-cluster pairs to show that our clustering method indeed makes intra-cluster pairs share similar

Table 2: Comparison of classification robustness and explanation robustness of models trained with TRADES and different α on CIFAR10. Within a certain range, using the TRADES training method and increasing the value of α can not only improve the classification robustness but also improve the explanation robustness.

α	Acc_{adv} (%)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)
0	0.00	6.206	10.375
0.5	23.57	10.640	16.635
1.0	28.31	10.946	17.271
2.0	31.77	10.965	17.290
4.0	33.28	11.293	18.004
5.0	33.98	11.469	18.278
10.0	34.87	11.592	18.643

explanations quantitatively in Table 1, which shows intra-cluster pairs do have a smaller loss.

ResNet18	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)
Intra-cluster	13.726
Inter-cluster	15.437

Table 1: Explanation loss at start of intra and inter clusters. The smaller explanation loss in the intra-cluster shows that images in the same cluster have similar explanations.

Measuring Explanation Robustness. Specifically, we quantify the explanation robustness through the final explanation loss after explanation attack converges: $\mathcal{L}_e^{\text{end}} = \mathbb{E}_{(x_v,x_t)\sim\mathcal{D}_e}[\mathcal{L}_e(x_v+\delta^{\text{end}}, x_t)]$, where δ^{end} is the optimal perturbation found by adversarial attacks on explanation. A higher $\mathcal{L}_e^{\text{end}}$ indicates that the attack struggles to manipulate the model's explanations, suggesting stronger explanation robustness. We can also measure the explanation loss before the explanation attack starts:

$$\mathcal{L}_e^{\text{start}} = \mathbb{E}_{(x_v,x_t)\sim\mathcal{D}_e}[\mathcal{L}_e(x_v+\delta^{\text{start}}, x_t)], \tag{4}$$

here δ^{start} is a random initial perturbation. It should be noted that $\mathcal{L}_e^{\text{start}}$ does not indicate the explanation robustness of a model since it is calculated before the explanation attack applies. However, providing $\mathcal{L}_e^{\text{start}}$ could let us know the difficulty of the attack before the beginning of the attack. Besides, we could also use the difference between $\mathcal{L}_e^{\text{start}}$ and $\mathcal{L}_e^{\text{end}}$ to roughly estimate the flatness of the loss landscape during training, while the definition for loss landscape at one point is introduced in the next section. After defining $\mathcal{L}_e^{\text{start}}$, we could obtain the quantitative results of explanation loss at start for the intra-cluster and inter-cluster for the clusters we obtain in the last section. In detail, Table 1 represents the intra-cluster and inter-cluster results for $\mathcal{L}_e^{\text{start}}$ on ResNet18 and CIFAR10 dataset. From the results, we can see that $\mathcal{L}_e^{\text{start}}$ for intra-cluster is indeed smaller than inter-cluster results, indicating our cluster-based method can obtain similar explanations in the cluster.

3.1.3 Input Loss Landscape

An input loss landscape is a visualization of how a model's loss changes across different possible input values [1], mapping the terrain of the loss function with respect to the input space, allowing researchers to understand how sensitive the model is to variations in the input data and identify potential robustness issues. Prior work has shown that a flatter input loss landscape for classification loss $\mathcal{L}_{\text{sc}}$ indicates stronger classification robustness [57, 29].

Following existing works [1], we visualize the input loss landscape w.r.t explanation loss $\mathcal{L}_e(x_v, x_t; f)$ by plotting the change of $\mathcal{L}_e$ when adding a random noise $\mathbf{d}$ to the victim image x_v with different magnitude γ:

$$l(\gamma) = ||I_f(x_v + \gamma d) - I_f(x_t)||^2, \tag{5}$$

where $\mathbf{d}$ is sampled from a standard Gaussian distribution.

3.2 Contradictory Findings on CIFAR10

Previous work has two assumptions: (1) a model with good classification robustness has a flat loss landscape w.r.t classification loss [57, 29]; (2) classification robustness and explanation robustness are positively correlated [5, 22]. A natural conclusion would follow if the previous assumptions are true: a model with good classification robustness might have a good explanation and thus a flat loss landscape w.r.t explanation loss as well.

3.2.1 *Step 1: Training models with different levels of classification robustness*

Our systematic investigation begins by establishing several models with different classification robustness through the TRADES framework [60] by training with different α in Equation (1). The preliminary results on the CIFAR10 dataset [26] can be found in Table 2. We can observe that with the increase of α, the adversarial accuracy Acc_{adv} increases as well. This indicates that the model adversarially trained with TRADES shows better classification robustness with the increase of α. This step provides us with models with different classification robustness.

3.2.2 *Step 2: Measuring explanation attack loss for models with different classification robustness*

After getting the models with different classification robustness, the next step is to measure their explanation robustness, i.e., whether they perform differently under explanation attacks. With the assumption (2) classification robustness and explanation robustness are positively correlated [5, 22], we would expect that models with better classification robustness will show better explanation robustness. That is, models with higher Acc_{adv} will have higher $\mathcal{L}_e^{\text{end}}$. The preliminary results in Table 2 shows that with the increase of α, models indeed have higher $\mathcal{L}_e^{\text{end}}$. But since $\mathcal{L}_e^{\text{start}}$ is also higher with the increase of α. Actually, calculating the $\Delta\mathcal{L} = \mathcal{L}_e^{\text{start}} - \mathcal{L}_e^{\text{end}}$, we could find that, after training, the attack is even more successful, considering the loss decrease. Therefore, we **cannot conclude** that better explanation robustness owes to better classification robustness. Next, we will explore a different way of measuring explanation robustness with the loss landscape.

3.2.3 *Step 3: Connecting Robustness with Loss Landscape*

To further investigate the explanation robustness of models, we visualize the input loss landscape w.r.t explanation

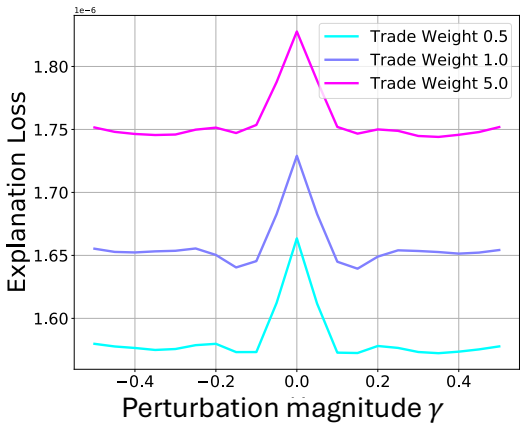


Figure 4: Input loss landscape w.r.t explanation loss for models trained with different Trade Weight α in TRADES. The loss landscape does not show a clear difference between models that vary in explanation robustness because the loss change remains the same.

loss. With the assumption (1) a model with good classification robustness has a flat loss landscape w.r.t classification loss [57, 29], we would expect a similar analogy for explanation robustness: a model with good explanation robustness has a flat loss landscape w.r.t explanation loss.

Using Equation (5), we visualize the input loss landscape by plotting the change of explanation loss, and the results are shown in Figure 4. We also visualize the input loss landscape with normal training and Madry adversarial training (MAT) in Appendix Figure 9. From Figure 4, we can find that the input loss landscape w.r.t. explanation loss **does not** show a difference in flatness, despite that their explanation loss $\mathcal{L}_e$ are different. Different from the conclusions drawn in classification robustness, models with good explanation robustness do not exhibit a flat loss landscape w.r.t explanation loss. This contradiction motivates us to further explore and propose a method to decouple classification and explanation robustness in the following section by changing explanation robustness by flattening the input loss landscape w.r.t explanation robustness, while maintaining classification robustness.

3.3 Landscape-Aware Regularization

Motivated by the observed landscape-robustness contradiction, in this section, we propose a method to decouple classification and explanation robustness. Specifically, we propose a new training algorithm to control the input loss landscape w.r.t explanation robustness and see if models with different flatness will perform differently on explanation robustness. If we can have a training algorithm that can influence explanation robustness while not changing classification robustness, we can conclude that classification robustness and explanation robustness are not strongly correlated.

To explicitly control the input loss landscape w.r.t explanation robustness, we propose **S**eparate **E**xplanation Robustness via **P**GD (SEP) through the following loss function:

$$\mathcal{L}_{\text{SEP}} = \|I(x+\zeta) - I(x)\|_2^2, \tag{6}$$

where ζ is a noise randomly sampled from a standard Gaussian distribution and I is the explanation method. Note that off-the-shelf explanation AT [49, 14] must be executed in a targeted setting: A victim image and a target image are required for the explanation of adversarial attacks. Calculating ζ through a targeted setting may increase the training time and increase the probability that the model overfits the chosen pairs. Therefore, we use randomly sampled noise, like for ζ which does not require a target image.

Algorithm 1 SEP Algorithm

Input: Dataset $\mathcal{D}$, total training iteration T, explanation method I, model weights $\mathbf{w}$, and balancing factor λ.
for $t = 0$ **to** $T-1$ **do**
  for batch x in $\mathcal{D}$ **do**
    Sample a random noise ζ from a standard Gaussian distribution.
    Get adversarial samples (on classification): $x_{adv} = PGD(x, y)$.
    Calculate loss function with Equation (7).
    Update $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla \mathcal{L}(f(x), f(x_{adv}), y|\mathbf{w})$
  end for
end for

The new loss function $\mathcal{L}_{\text{SEP}}$ can be incorporated into existing training frameworks, including Madry adversarial training [34], TRADES [60], and normal training. In this paper, we will mainly focus on Madry adversarial training plus the new training loss:

$$\mathcal{L} = \mathcal{L}_{sc}(f(x_{adv}), y) + \lambda \mathcal{L}_{\text{SEP}}. \tag{7}$$

where the hyperparameter λ balances two components of the losses. When $\lambda > 0$, we have SEP_{pos} which guides the loss landscape to become flatter; when $\lambda < 0$, we have SEP_{neg}, which guides the loss landscape to become sharper. The overall training algorithm is shown in Algorithm 1.

In the following section, we show that our method can influence explanation robustness while it does not change classification robustness. Please note that our method is designed and used to explore the relationship between the classification robustness and explanation robustness instead of common dimensions that a normal algorithm will care like the performance. We also visualize the comparison of saliency maps from models trained with different algorithms to provide how our methods influence the saliency maps in Figure 7 in the later section to show how our method works.

4. EXPERIMENTS

We conduct comprehensive experiments across multiple datasets and model architectures to validate our method's ability to decouple explanation robustness from classification robustness. Our evaluation addresses three key research questions:

- **RQ1:** Can we independently control explanation robustness across different datasets and model architectures without changing classification robustness?
- **RQ2:** How do different explanation methods affect this relationship between two robustnesses?
- **RQ3:** Does our method generalize across different training protocols, e.g., different adversarial training methods like TRADES?

4.1 Experimental Settings

Table 3: Performance of models trained with ConvNet and ResNet18 on various datasets is evaluated using four training methods, w.r.t. $\mathcal{L}_e^{\text{end}}$ and Acc_{adv}. Higher $\mathcal{L}_e^{\text{end}}$ indicates better explanation robustness; higher Acc_{adv} indicates better classification robustness. $\mathcal{L}_e^{\text{start}}$ is also included to show our method's influence on explanation robustness. The **best** performance in explanation and classification robustness and the worst performance in explanation robustness are highlighted. There is no positive correlation between explanation and classification robustness achieved through $\texttt{SEP}_{pos}$ and $\texttt{SEP}_{neg}$ training methods, compared to MAT.

Model Architecture	ConvNet				ResNet18			
MNIST								
Training Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
Normal	261.183	204.825	99.29	0.00	266.834	146.16	99.36	0.00
MAT	373.262	298.729	99.00	89.92	916.017	778.003	99.28	**94.60**
$\texttt{SEP}_{pos}$	93.033	61.545	98.8	89.4	92.371	59.278	98.4	91.63
$\texttt{SEP}_{neg}$	806.204	**657.180**	98.97	**90.34**	9356.306	**8248.627**	99.4	93.95
FMNIST								
Training Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
Normal	106.530	72.198	92.32	0.00	128.640	69.847	91.57	0.00
MAT	386.370	274.267	62.85	73.98	588.610	417.031	79.22	**67.10**
$\texttt{SEP}_{pos}$	35.588	22.465	69.88	**86.81**	32.466	22.512	68.75	56.51
$\texttt{SEP}_{neg}$	1811.969	**994.818**	62.75	76.89	8050.942	**7593.650**	70.23	57.55
CIFAR10								
Training Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
Normal	10.375	6.206	79.08	0.00	13.982	6.130	81.32	0.00
MAT	16.913	6.906	64.85	35.11	31.959	21.879	67.22	29.09
$\texttt{SEP}_{pos}$	3.565	1.269	64.94	**35.25**	11.962	7.958	66.68	**29.69**
$\texttt{SEP}_{neg}$	19.002	**7.590**	64.56	34.86	70.159	**36.276**	39.17	29.32
CIFAR100								
Training Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
Normal	10.099	6.140	48.39	0.05	12.044	4.716	41.24	0.00
MAT	20.642	13.650	36.4	17.35	33.456	22.623	36.14	15.70
$\texttt{SEP}_{pos}$	13.650	9.932	37.41	**17.98**	19.217	12.744	34.83	15.16
$\texttt{SEP}_{neg}$	22.506	**14.970**	36.17	17.43	35.525	**24.289**	34.80	**15.87**
TinyImageNet								
Training Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
Normal	0.966	0.633	28.71	0.00	1.131	0.528	28.34	0.00
MAT	2.426	1.728	25.13	9.55	3.119	2.349	26.34	10.81
$\texttt{SEP}_{pos}$	2.242	1.571	24.83	**9.63**	1.967	1.435	25.96	**10.83**
$\texttt{SEP}_{neg}$	3.873	**2.610**	24.31	9.61	4.413	**3.016**	26.11	10.74

4.1.1 Datasets

To thoroughly demonstrate the impact of our proposed training method and the resulting conclusions, we evaluate on five standard benchmarks:

- **CIFAR10** [26]: consisting of 60k 32×32 color images in 10 classes including 50k training images and 10k test images.
- **CIFAR100** [26]: containing the same images as CIFAR10 but has a more refined label with 100 categories, which makes it a harder dataset.
- **MNIST** [28]: containing 60k training samples and 10k test samples from 10 digit classes. Each digit is a 28×28 grayscale image.
- **Fashion MNIST** [56]: consisting of 60k training samples and 10k test samples from 10 classes. Each sample is a 28×28 grayscale image in a clothes category.
- **TinyImageNet** [27]: it is a subset of ImageNet [13] with 64x64 pixels and 200 categories

We also consider using ImageNet [13] and the experiment results for ImageNet can be found in Appendix D.3. The results for ImageNet are similar to the results here.

4.1.2 Architecture of To-be-explained Model

In addition to utilizing diverse datasets, we have also designed four distinct model architectures for training on these datasets. We conduct experiments on ConvNet, ResNet [18], Wide ResNet [59] and MoblieNetV2 [21, 38]. The ConvNet model consists of three convolutional layers and one fully connected layer from Gidaris et al. [16]. For ResNet and Wide ResNet, we use a standard ResNet18 and Wide-ResNet-28, respectively. We also adjust the ResNet, Wide ResNet, and MoblieNetV2 so that they can fit into all datasets we use. All models use Softplus activation for explanation attack compatibility [14], maintaining ReLU-like behavior with improved differentiability.

4.1.3 Explanation Methods

For explanation methods, we mainly use the implementations of five explanation methods from Captum [25]: Gradient [4], Gradient $\times$ Input [41], Guided Backpropagation [46], Deep Lift [41] and Integrated Gradients [47]. Their detailed descriptions can be found in Appendix A.

4.1.4 Training Protocols

We consider two baselines: normal training (Normal) and Madry adversarial training (MAT) [34]. We also explore two variants of the proposed method: $\texttt{SEP}_{pos}$ and $\texttt{SEP}_{neg}$, as mentioned in Section 3.3. In the rest of this paper, unless specified, we will use $\lambda = 50000$ for $\texttt{SEP}_{pos}$ and $\lambda = -3000$ for $\texttt{SEP}_{neg}$. We use a learning rate of 0.01 for ConvNet and MobileNet while using a learning rate of 0.001 for ResNet and Wide ResNet.

4.1.5 Hyperparameters

For all experiments, we train our models for 25 epochs with 64 as the batch size. We also consider different training

epochs and our conclusion remains the same as shown later. To accelerate the training process, we use Adam [24] as the optimizer. We use the standard settings in adversarial training [36], with $\epsilon = 8/255$ in PGD for RGB images and $\epsilon = 0.3$ for grayscale images, and steps in PGD are set to 10 for all experiments. We list the detailed hyperparameters in the Appendix Table 10.

4.1.6 Metrics

As mentioned in Section 3.1.2, we measure explanation robustness using the explanation loss in the end after explanation attack $\mathcal{L}_e^{\text{end}}$. A higher $\mathcal{L}_e^{\text{end}}$ indicates a worse attack and thus better explanation robustness. We also report the explanation loss before explanation attack $\mathcal{L}_e^{\text{start}}$ to show the influence of our method on the explanation loss landscape. For classification robustness, we report adversarial accuracy Acc_{adv}, with higher values indicating better classification robustness. Additionally, we include clean accuracy Acc_{clean} to ensure the models function normally in non-adversarial settings.

4.2 Decoupling Robustness Dimensions (RQ1)

We extend our preliminary experiments on CIFAR10 in Section 3.2 to multiple model architectures and datasets with Gradient × Input as the explanation method. The results are shown in Table 3 for ConvNet and ResNet18 with additional results for W-ResNet and MoblieNetV2 provided in the Appendix Table 9. We have the following observations:

- Across all the datasets and model architecture, $\texttt{SEP}_{pos}$ has the lowest $\mathcal{L}_e^{\text{end}}$, and $\texttt{SEP}_{neg}$ has the highest $\mathcal{L}_e^{\text{end}}$. This indicates that $\texttt{SEP}_{pos}$ shows weaker explanation robustness and $\texttt{SEP}_{neg}$ shows the strongest explanation robustness. The different performance w.r.t. explanation loss at end for $\texttt{SEP}_{pos}$ and $\texttt{SEP}_{neg}$ is mainly induced by the difference in $\mathcal{L}_e^{\text{start}}$, which is influenced by our training method by setting λ to positive or negative.
- While $\texttt{SEP}_{pos}$, $\texttt{SEP}_{neg}$, and MAT have very similar Acc_{adv}, $\texttt{SEP}_{pos}$ shows the weakest explanation robustness by having the lowest $\mathcal{L}_e^{\text{end}}$, and $\texttt{SEP}_{neg}$ shows the strongest explanation robustness. These results show that despite the classification robustness of $\texttt{SEP}_{pos}$, $\texttt{SEP}_{neg}$, and MAT being similar, their explanation robustness is different. Therefore, we argue that there is no inherent relationship between explanation robustness and classification robustness.
- In the setting of CIFAR10 and ResNet18, increasing the explanation robustness by $\texttt{SEP}_{neg}$ hurts the Acc_{clean} while it still does not change Acc_{adv}, which represents classification robustness. This observation further validates our argument: classification robustness and explanation robustness may not be strongly correlated.
- From the results, we could see that though $\texttt{SEP}_{neg}$ has a much higher $\mathcal{L}_e^{\text{end}}$ compared with the $\texttt{SEP}_{pos}$. However, if we consider $\Delta\mathcal{L} = \mathcal{L}_e^{\text{start}} - \mathcal{L}_e^{\text{end}}$, we can find $\Delta\mathcal{L}$ for $\texttt{SEP}_{neg}$ is much higher than $\mathcal{L}_e^{\text{end}}$. This loss decrease demonstrates that our design is working since the larger loss decrease indicate $\texttt{SEP}_{neg}$ has a sharper loss landscape. Considering all the results, we hypothesis that the explanation robustness actually comes from the increase of the attack difficulty at the starting point instead of a flat loss landscape that is hard for the attacker to optimize. While previous works [57, 29] show that classification robustness comes from the flat loss landscape, this mechanism difference between classification and explanation robustness also further indicate that they might not be strongly correlated.

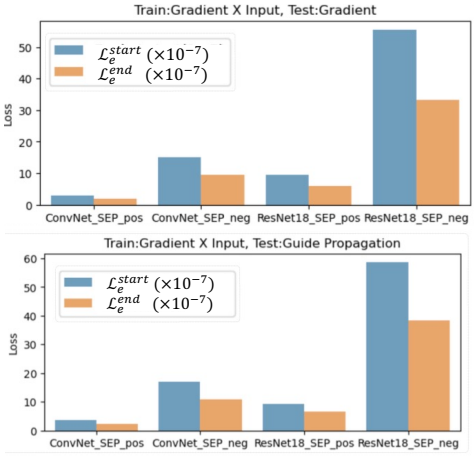


Figure 5: Performance of varying explanation methods in the testing phase, w.r.t. $\mathcal{L}_e^{\text{start}}$, $\mathcal{L}_e^{\text{end}}$, Acc_{clean} and Acc_{adv}. Models are trained with Gradient x Input on CIFAR10 and tested on different explanation methods: Gradient (Top) and Guide Propagation (Bottom). Even if the explanation methods during training and testing are different, $\texttt{SEP}_{pos}$ shows a lower explanation loss compared to $\texttt{SEP}_{neg}$, while they have similar adversarial accuracy.

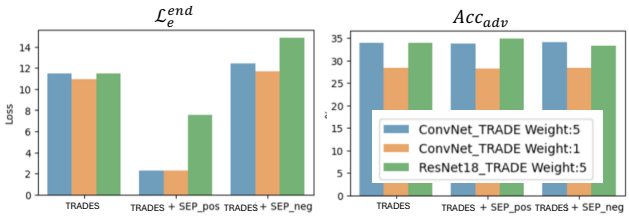


Figure 6: The test results of the model trained using the TRADE training method with CIFAR10, combined with our approach, are presented. The findings indicate that when we apply our method to an alternative adversarial training method TRADES, distinct from MAT, we can still find that the classification robustness and explanation robustness are not inherently interconnected.

4.3 Explanation Method Agnosticism (RQ2)

4.3.1 Training Phase Variations

In the previous experiment, we demonstrated that under the Gradient × Input explanation method, our methods achieve similar classification robustness while exhibiting significantly different explanation robustness. To further investigate whether this conclusion holds for different explanation methods, we changed the explanation method to Gradient and Guide Propagation. The results are based on CIFAR10 and are summarized in Table 4. Extended results using DeepLIFT [41] and Integrated Gradients [47] as the explanation methods on ConvNet and various datasets. The results

Table 4: Performance of different explanation methods (Gradient and Guide Propagation) in the training phase is evaluated w.r.t. $\mathcal{L}_e^{\text{start}}$, $\mathcal{L}_e^{\text{end}}$, Acc_{clean} and Acc_{adv} on CIFAR10. Higher $\mathcal{L}_e^{\text{end}}$ indicates better explanation robustness, while higher Acc_{adv} denotes better classification robustness. The **best** and worst performances in explanation robustness and classification robustness are highlighted. Under various explanation methods, $\texttt{SEP}_{pos}$ shows a lower explanation loss compared to $\texttt{SEP}_{neg}$, with similar adversarial accuracy.

Model Architecture	ConvNet				ResNet18			
Gradient								
Training Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
Normal	7.977	4.591	79.08	0.00	11.310	4.671	81.32	0.00
MAT	13.810	8.705	64.85	**35.11**	26.899	18.215	67.22	29.09
$\texttt{SEP}_{pos}$	0.876	0.503	52.89	29.68	11.317	6.604	66.76	**37.69**
$\texttt{SEP}_{neg}$	13.964	**9.290**	53.23	29.56	8282.990	**7236.182**	49.38	32.28
Guide Propagation								
Training Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
Normal	8.075	4.639	79.08	0.00	11.515	4.736	81.32	0.00
MAT	14.012	8.813	64.85	35.11	27.012	18.311	67.22	29.09
$\texttt{SEP}_{pos}$	1.023	0.506	60.27	33.57	12.004	7.593	67.16	30.64
$\texttt{SEP}_{neg}$	14.643	**9.110**	59.74	**33.78**	27.422	**18.940**	66.48	**30.72**

using Guide Propagation with TinyImageNet and FMNIST can be found at Table 8. We have the following observations:

• Our methods achieve similar classification robustness under various explanation methods, yet they exhibit notably different explanation robustness. In most cases, $\texttt{SEP}_{pos}$ shows lower explanation loss compared to $\texttt{SEP}_{neg}$, despite similar adversarial accuracy.

• Compared to MAT, our method $\texttt{SEP}_{pos}$ shows comparable adversarial accuracy, indicating similar classification robustness, but it demonstrates distinct explanation loss characteristics. This suggests that explanation robustness and classification robustness may not be strongly correlated.

• Comparing the results in Table 7, Table 4 and Table 8, we can find that no matter which explanation method we use or which dataset we use, we consistently get the conclusion that we could get quite different explanation robustness when we have the similar classification robustness for $\texttt{SEP}_{pos}$ and $\texttt{SEP}_{neg}$. This demonstrates the robustness of our results and demonstrates that our findings are not restricted to one specific explanation method or dataset.

4.3.2 *Testing Phase Generalization*

To test if our findings hold when using different explanation methods during testing, in this experiment, we use the same model trained with Gradient × Input (thus the classification robustness is the same for different testing phases), but change two different explanation methods (Gradient and Guide Propagation) in the testing phase. The results on CIFAR10 are shown in Figure 5, where the detailed value of this experiment can be found in Appendix Table 11. While with the same classification robustness (as shown in Table 3, under adversarial accuracy in CIFAR10), there is a huge difference between $\texttt{SEP}_{pos}$ and $\texttt{SEP}_{neg}$ w.r.t the explanation losses (both at the start and the end). This indicates that even with different explanation methods in the testing phase, the explanation robustness still does not show strong correlations with adversarial robustness.

4.4 Training Protocol Generalization (RQ3)

All previous experiments utilized MAT [34] as the default adversarial training protocol. To assess the generalizability of our approach across different adversarial training protocols, we employed TRADES [60] in this experiment and the results can be found in Figure 6 (details in Appendix Table 12). We can observe that when changing the adversarial training method from MAT to TRADES, a larger α will lead to a larger Acc_{adv}, indicating a better classification robustness. As α increases, $\mathcal{L}_e^{\text{end}}$ of the model trained with SEP varies. This observation indicates that our SEP method impacts explanation robustness without altering classification robustness, suggesting a weak correlation between explanation robustness and classification robustness.

4.5 Parameter Sensitivity Analysis

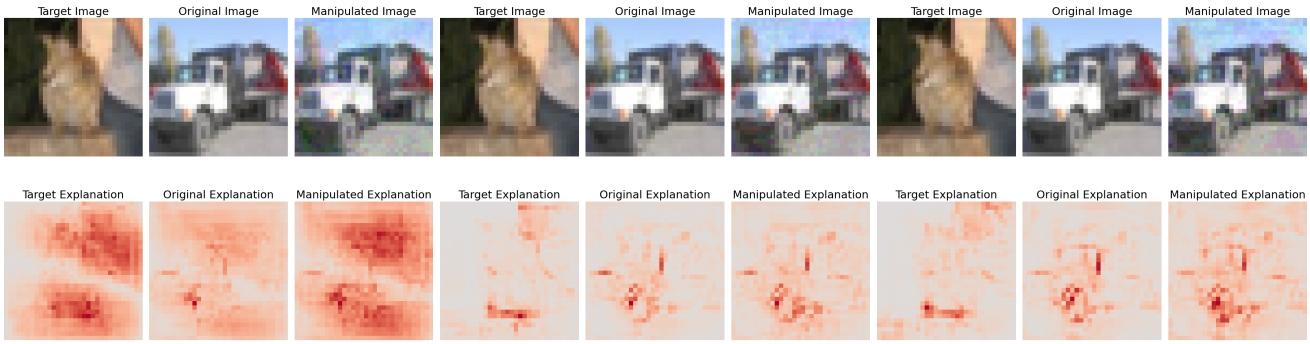
In this section, we examine how different parameters in our experiments affect the results.

Regularization Weights: Firstly, we test how regularization weights λ affect the results. In detail, we trained ConvNet networks on CIFAR10 with various λ values. The test results are presented in Table 5. We observe that the choice of λ influences both the exploration rate at start and end. When λ is greater than 10^4 or less than $-3 * 10^3$, the explanation loss changes intensely. However, from the results, we can see that for different λ from $5{*}10^4$ to $-3{*}10^3$, the classification robustness remains the same, which further validate our hypothesis that explanation robustness and classification robustness may not be highly correlated.

Training Epochs: To demonstrate how training epochs might influence our conclusion, we conducted experiments on the ConvNet network using the CIFAR10 dataset with different training epochs. The results, as presented in Table 6, indicate that the model's performance undergoes only marginal changes after 25 rounds for ConvNet, despite the epoch count continuing to increase. Therefore, we conclude that choosing 25 epochs in the rest experiments does not hurt the reliability of our argument. Besides, the results also support our conclusion. With the increase of training epochs, the classification robustness still increases while the explanation robustness actually decreases, which further validates our conclusion.

4.6 Hessian Analysis

To further analyze how adversarial training and SEP affect the input loss landscape, we perform a Hessian analysis of the loss with respect to the input. Concretely, we study the input Hessian $H_x = \nabla_x^2 L_e(x;\theta)$ with θ fixed after training. The eigenvectors of H_x define principal directions of curvature in input space and the corresponding eigenvalues quantify how rapidly the loss bends along those directions. Large absolute eigenvalues indicate a sharp loss landscape,



(a) $\texttt{SEP}_{pos}$ (b) $\texttt{SEP}_{neg}$ (c) Adversarial Training

Figure 7: Comparison of the saliency maps calculated from gradient x inputs on CIFAR10 with different training methods. Intuitively, $\texttt{SEP}_{pos}$ makes the model consider more input pixels, solely adversarial training makes the model consider only a few input pixels, while $\texttt{SEP}_{neg}$ considers even fewer input pixels compared with adversarial training. However, models trained with these three methods show the same classification robustness.

Table 5: The evaluation of the ConvNet trained on CIFAR10 under different λ conditions reveals that the relationship between explanation and classification robustness is not positively correlated when an appropriate λ is selected during model training.

λ	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
0 (MAT)	16.913	6.206	64.85	35.11
$5 * 10^4$	3.565	1.269	64.94	35.25
10^4	15.436	5.870	64.39	35.18
10^1	17.646	6.819	64.45	35.02
-10^2	17.820	6.934	64.67	35.14
$-3 * 10^3$	19.002	7.590	64.56	34.86

Table 6: Test results of the ConvNet model at different training epochs on the CIFAR10 dataset. As the number of training epochs increases beyond 25, the improvement in performance is marginal. Therefore, 25 epochs are selected as the final number of training epochs for all models to maintain faster training speed without affecting the overall conclusions.

Model Architecture	ConvNet (CIFAR10)			
Training Epoch	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
25	4.388	1.605	64.94	35.25
50	3.885	1.431	65.69	35.94
75	3.671	1.378	66.33	36.27
100	3.557	1.339	66.74	36.50

while small absolute values indicate flatness of the loss landscape. Following previous works [31], we compute the top 20 eigenvalues at evaluation inputs and report their dataset means for the Hessian of explanation loss, which captures both how steep the sharpest direction is and whether sensitivity is concentrated in a few directions or spread across many, as reflected by the spectral decay. In detail, we show the results in Figure 8. The results show that $\texttt{SEP}_{pos}$ has much lower eigenvalues and thus indicates a much flatter loss landscape. Similarly, $\texttt{SEP}_{neg}$ has higher eigenvalues. These results firstly show that the design of SEP is successful because $\texttt{SEP}_{pos}$ and $\texttt{SEP}_{neg}$ influence the loss landscape as we want. Secondly, the results of flatness of loss landscape of $\texttt{SEP}_{pos}$ and $\texttt{SEP}_{neg}$ further validate the assumption in the previous section that the explanation robustness is mainly from the high initial loss instead of flat loss landscape, which indicates that there is no internal relationship between classification robustness and explanation robustness.

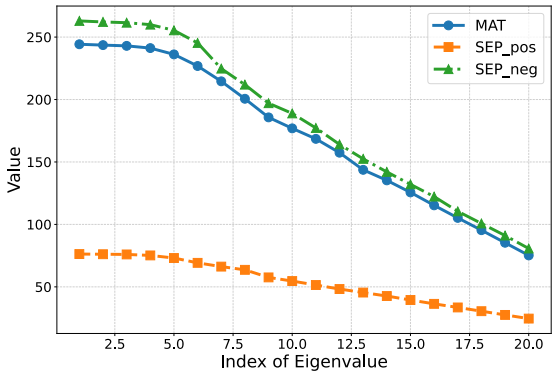


Figure 8: Top-20 Eigenvalue of Hessian Matrix considering the explanation loss. A higher value of eigenvalue indicates a shaper loss landscape. The results show that SEP influence the loss landscape.

4.7 Case Study

In this section, we visualize the comparison of saliency maps from models trained with different algorithms to provide how our methods influence the saliency maps in Figure 7. Specifically, we use the Gradient × Input method [41] to calculate saliency maps on the CIFAR-10 dataset [26] with ConvNet. From Figure 7 we can see that while models trained with $\texttt{SEP}_{pos}$ exhibit broader activation regions in their saliency maps, models trained with standard adversarial training focus on fewer input pixels compared to $\texttt{SEP}_{pos}$. And models trained with $\texttt{SEP}_{neg}$ exhibit the narrowest focus, considering even fewer input pixels than adversarial training. Firstly, these findings highlight the flexibility of our method in shaping explanation patterns and suggest that explanation robustness can be independently controlled through targeted regularization techniques. Secondly, the results show that adjusting the loss landscape can lead to totally different explanation patterns. From the explanation patterns, we can see that if the activated pixels are fewer, the initial distance, which is measured by the $\mathcal{L}_e^{\text{start}}$, will also be

Table 7: Performance of using DeepLift and Integrated Gradients as explanation methods with ConvNet. Higher $\mathcal{L}_e^{\text{end}}$ indicates better explanation robustness, while higher Acc_{adv} denotes better classification robustness. The **best** and worst performances in explanation robustness and classification robustness are highlighted. Under various explanation methods, SEP_{pos} shows a lower explanation loss compared to SEP_{neg}, with similar adversarial accuracy.

Explanation Method	DeepLift				Integrated Gradients			
MNIST								
Training Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
MAT	369.153	294.053	99.00	89.92	239.650	224.745	99.00	89.92
SEP_{pos}	82.959	57.408	98.93	95.97	76.284	50.603	98.42	**93.19**
SEP_{neg}	1101.038	**896.157**	98.97	**96.16**	778.663	**534.113**	98.68	92.76
FMNIST								
Training Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
MAT	386.377	274.130	62.85	**73.98**	237.727	234.109	62.85	**73.98**
SEP_{pos}	33.824	21.429	60.75	65.52	21.425	16.048	60.85	72.05
SEP_{neg}	4739.769	**3153.331**	60.62	70.05	3624.231	**2748.976**	62.92	70.36
CIFAR10								
Training Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
MAT	18.137	11.562	64.85	35.11	16.887	14.007	64.85	**35.11**
SEP_{pos}	2.593	1.273	65.76	**35.38**	3.696	2.523	60.19	32.33
SEP_{neg}	21.329	**13.819**	64.20	34.91	19.568	**14.803**	60.86	32.43
CIFAR100								
Training Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
MAT	19.683	12.766	36.40	17.35	16.754	10.779	36.40	**17.35**
SEP_{pos}	14.208	9.201	39.10	18.23	5.320	3.218	34.73	16.74
SEP_{neg}	20.389	**13.694**	39.78	**18.32**	17.011	**11.356**	35.10	16.60

Table 8: Performance of using Guide Propagation in the training phase with Fashion-MNIST and TinyImageNet. Higher $\mathcal{L}_e^{\text{end}}$ indicates better explanation robustness, while higher Acc_{adv} denotes better classification robustness. The **best** and worst performances in explanation robustness and classification robustness are highlighted. Under various explanation methods, SEP_{pos} shows a lower explanation loss compared to SEP_{neg}, with similar adversarial accuracy.

Model Architecture	ConvNet				ResNet18			
Fashion-MNIST (FMNIST)								
Training Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
Normal	30.932	18.451	92.79	0.00	66.131	29.110	91.57	0.00
MAT	97.726	72.402	62.85	73.98	608.486	467.815	79.22	67.10
SEP_{pos}	48.368	34.672	78.46	67.28	97.703	78.354	77.55	62.09
SEP_{neg}	542.540	**425.948**	65.07	**77.17**	4219.351	**3839.408**	80.11	**72.21**
TinyImageNet								
Training Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
Normal	0.559	0.281	28.71	0.00	0.617	0.216	28.34	0.00
MAT	1.356	0.787	25.13	9.55	2.577	1.411	26.33	10.81
SEP_{pos}	0.983	0.625	25.16	5.97	1.767	1.226	28.68	11.47
SEP_{neg}	1.566	**0.977**	24.89	4.99	3.403	**1.761**	26.79	11.23

higher. Though in this case, the loss decrease is higher, the large initial distance ensures a better explanation robustness. On the other hand, if there are more activated pixels, it is more likely that two explanations are close at the beginning. Even though $\texttt{SEP}_{pos}$ has a flat loss landscape, which makes the attacker harder to optimize, the final attack is still successful. Based on the case, we further validate that the mechanism for explanation robustness is not based on the flat loss landscape, indicating there is no strongly correlated relationship between the two robustness.

5. CONCLUSION

This study challenges the widely held assumption that explanation robustness and classification robustness are strongly correlated. By systematically control classification robustness, we demonstrate that increasing explanation robustness does not necessarily result in a flatter input loss landscape for explanation loss. This finding contrasts with the well-established observation that enhancing classification robustness leads to a flatter input loss landscape for classification loss. These results reveal a fundamental difference in how these two types of robustness are influenced by adversarial training. To address this discrepancy, we propose SEP, a novel algorithm that explicitly flattens the input loss landscape for explanation loss. Our experiments show that this approach effectively improves explanation robustness without affecting classification robustness, providing evidence that these two forms of robustness are not inherently linked. This decoupling highlights the importance of separately considering and optimizing explanation and classification robustness to ensure the reliability and trustworthiness of AI systems, particularly in high-stakes domains such as healthcare, autonomous driving, and finance.

Our findings emphasize the need for future research to further explore the relationship between these two types of robustness. In the future, we will theoretically investigate why adversarial training can improve explanation robustness in certain cases and what underlying mechanisms differentiate the behavior of classification and explanation robustness under adversarial attacks. A deeper understanding of these mechanisms will provide valuable insights into designing more robust and interpretable AI systems capable of maintaining both predictive accuracy and trustworthy explanations under adversarial conditions.

Acknowledgment

The work was partially supported by NSF award #2442477. We thank Amazon Research Awards, Cisco Research Awards, Google, and OpenAI for providing us with API credits. The views and conclusions in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

[1] D. Alvarez-Melis and T. S. Jaakkola. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*, 2018.

[2] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

[3] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[4] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. Müller. How to explain individual classification decisions. *The Journal of Machine Learning Research*, 11:1803–1831, 2010.

[5] A. Boopathy, S. Liu, G. Zhang, C. Liu, P.-Y. Chen, S. Chang, and L. Daniel. Proper network interpretability helps adversarial robustness in classification. In *International Conference on Machine Learning*, pages 1014–1023. PMLR, 2020.

[6] W. Brendel, J. Rauber, A. Kurakin, N. Papernot, B. Veliqi, S. P. Mohanty, F. Laurent, M. Salathé, M. Bethge, Y. Yu, et al. Adversarial vision challenge. In *The NeurIPS'18 Competition: From Machine Learning to Intelligent Conversations*, pages 129–153. Springer, 2020.

[7] N. Carlini, F. Tramer, K. D. Dvijotham, L. Rice, M. Sun, and J. Z. Kolter. (certified!!) adversarial robustness for free! *arXiv preprint arXiv:2206.10550*, 2022.

[8] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[9] Y. Carmon, A. Raghunathan, L. Schmidt, J. C. Duchi, and P. S. Liang. Unlabeled data improves adversarial robustness. *Advances in neural information processing systems*, 32, 2019.

[10] J. Chen, X. Wu, V. Rastogi, Y. Liang, and S. Jha. Robust attribution regularization. *Advances in Neural Information Processing Systems*, 32, 2019.

[11] K. Chen, Y. Chen, H. Zhou, X. Mao, Y. Li, Y. He, H. Xue, W. Zhang, and N. Yu. Self-supervised adversarial training. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2218–2222. IEEE, 2020.

[12] T. Chen, Z. Zhang, S. Liu, S. Chang, and Z. Wang. Robust overfitting may be mitigated by properly learned smoothening. In *International Conference on Learning Representations*, 2020.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[14] A.-K. Dombrowski, M. Alber, C. Anders, M. Ackermann, K.-R. Müller, and P. Kessel. Explanations can be manipulated and geometry is to blame. *Advances in neural information processing systems*, 32, 2019.

[15] A. Ghorbani, A. Abid, and J. Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3681–3688, 2019.

[16] S. Gidaris and N. Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4367–4375, 2018.

[17] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] D. Hendrycks, K. Lee, and M. Mazeika. Using pre-training can improve model robustness and uncertainty. In *International conference on machine learning*, pages 2712–2721. PMLR, 2019.

[20] J. Heo, S. Joo, and T. Moon. Fooling neural network interpretations via adversarial model manipulation. *Advances in neural information processing systems*, 32, 2019.

[21] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[22] W. Huang, X. Zhao, G. Jin, and X. Huang. Safari: Versatile and efficient evaluations for robustness of interpretability. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1988–1998, 2023.

[23] G. Jin, X. Yi, D. Wu, R. Mu, and X. Huang. Randomized adversarial training via taylor expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16447–16457, 2023.

[24] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan, et al. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*, 2020.

[26] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[27] Y. Le and X. Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[28] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[29] L. Li and M. Spratling. Understanding and combating robust overfitting via input loss landscape analysis and regularization. *Pattern Recognition*, 136:109229, 2023.

[30] J. Lin, C. Gan, and S. Han. Defensive quantization: When efficiency meets robustness. *arXiv preprint arXiv:1904.08444*, 2019.

[31] C. Liu, M. Salzmann, T. Lin, R. Tomioka, and S. Süsstrunk. On the loss landscape of adversarial training: Identifying challenges and how to overcome them. *Advances in Neural Information Processing Systems*, 33:21476–21487, 2020.

[32] G. Liu, I. Khalil, and A. Khreishah. Using single-step adversarial training to defend iterative adversarial examples. In *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, pages 17–27, 2021.

[33] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[34] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[35] T. Pang, K. Xu, Y. Dong, C. Du, N. Chen, and J. Zhu. Rethinking softmax cross-entropy loss for adversarial robustness. *arXiv preprint arXiv:1905.10626*, 2019.

[36] T. Pang, X. Yang, Y. Dong, H. Su, and J. Zhu. Bag of tricks for adversarial training. *arXiv preprint arXiv:2010.00467*, 2020.

[37] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[38] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[39] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. Grad-cam: Why did you say that? *arXiv preprint arXiv:1611.07450*, 2016.

[40] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein. Adversarial training for free! *Advances in Neural Information Processing Systems*, 32, 2019.

[41] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.

[42] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[43] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. arxiv 2013. *arXiv preprint arXiv:1312.6034*, 2019.

[44] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

[45] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

[46] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[47] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[48] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[49] S. V. Tamam, R. Lapid, and M. Sipper. Foiling explanations in deep neural networks. *arXiv preprint arXiv:2211.14860*, 2022.

[50] R. Tang, N. Liu, F. Yang, N. Zou, and X. Hu. Defense against explanation manipulation. *Frontiers in big Data*, 5:704203, 2022.

[51] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[52] Z. Wang, M. Fredrikson, and A. Datta. Robust models are more interpretable because attributions look normal. *arXiv preprint arXiv:2103.11257*, 2021.

[53] Z. Wang, T. Pang, C. Du, M. Lin, W. Liu, and S. Yan. Better diffusion models further improve adversarial training. *arXiv preprint arXiv:2302.04638*, 2023.

[54] M. Wicker, J. Heo, L. Costabello, and A. Weller. Robust explanation constraints for neural networks. *arXiv preprint arXiv:2212.08507*, 2022.

[55] D. Wu, S.-T. Xia, and Y. Wang. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33:2958–2969, 2020.

[56] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[57] C. Xie, M. Tan, B. Gong, A. Yuille, and Q. V. Le. Smooth adversarial training. *arXiv preprint arXiv:2006.14536*, 2020.

[58] W. Xu, D. Evans, and Y. Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[59] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[60] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pages 7472–7482. PMLR, 2019.

[61] X. Zhang, N. Wang, H. Shen, S. Ji, X. Luo, and T. Wang. Interpretable deep learning under fire. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020.

APPENDIX

A. CODE AND DATA

This is our open source code link: open source code.
We evaluate five explanation methods from Captum [25]:

- Gradient [4]: Raw input gradients
- Gradient×Input [41]: Element-wise product of inputs and gradients
- Guided Backprop [46]: Filtered gradient visualization
- DeepLIFT [41]: Reference-based difference attribution
- Integrated Gradients [47]: Path integral of gradients

B. MORE VISUALIZATION RESULTS

Firstly, we visualize the input loss landscape w.r.t explanation loss using a normal trained model and model trained with Madry adversarial training in Figure 9. The results show that increasing the explanation robustness does not flatten the input loss landscape. Besides, we also visualize

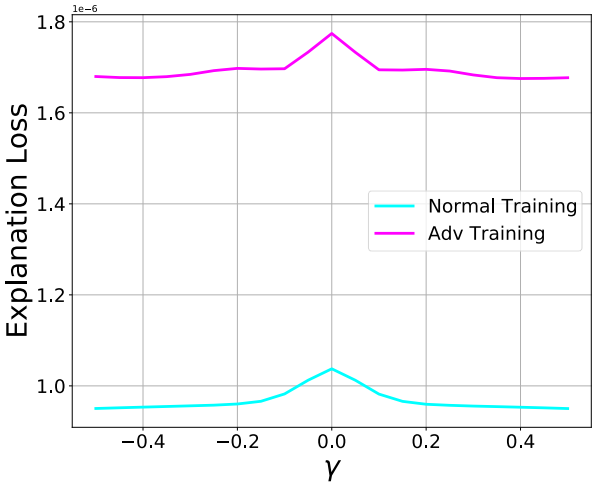


Figure 9: Comparison of input loss landscape w.r.t explanation loss with adversarial training and normal training. The results show that there is no obvious difference in input loss landscape.

more saliency maps with more explanation methods with images from different clusters in Figure 10. They all prove that we can choose the most representative saliency maps.

C. DETAILED HYPERPARAMETER

In this section, we provide the detailed hyperparameter for our CIFAR10 dataset in Table 10.

Table 10: Comparison of explanation loss for intra-cluster sample and inter-cluster sample on CIFAR10. The results show that our cluster method indeed the cluster images with similar explanations.

Models	Learning Rate	λ
SEP_pos		
ConvNet	0.01	5e4
ResNet18	0.001	5e4
Wide ResNet	0.001	5e4
MobileNet	0.01	5e4
SEP_neg		
ConvNet	0.01	-3e3
ResNet18	0.001	-1.9e3
Wide ResNet	0.001	-1.9e3
MobileNet	0.01	-1.25e3

D. MORE EXPERIMENTAL RESULTS

D.1 Experiments on W-ResNet and MobileNet

We list the main results using Gradient X Inputs as training and testing explanation methods for W-ResNet and MobileNetV2 in Table 9. We have the following observations:

- Once again, the adversarial accuracy for MAT, SEP_{pos}, and SEP_{neg} is similar in most scenarios for W-ResNet and MobileNet, while SEP_{pos} always has a smaller explanation loss compared with MAT, and SEP_{neg} always has a larger explanation loss compared with MAT. These results show that influencing explanation robustness does not necessarily change classification robustness.
- For W-ResNet and MobileNet, the adversarial accuracy for CIFAR100 fluctuates. For MobileNet and CIFAR100, compared with MAT, SEP_{pos} increases classification robustness while SEP_{neg} decreases it. However, this observation also indicates that the positive correlation between explanation robustness and classification robustness might not be true since SEP_{pos} decreases explanation robustness while increasing classification robustness.

D.2 Detailed Values for Transferablity Experiments

The detailed values for Transferablity experiments can be found in Table 11 and the detailed values for experiments using TRADES for our method can be found in Table 12. The analysis of these results can be found in the main paper.

D.3 Experiments on ImageNet

Here, we present our experiments on ImageNet with ResNet18 in Table 13. We can find that the conclusion of ImageNet experiments is the same as the main paper: Increasing or decreasing explanation robustness will not necessarily influence the classification robustness.

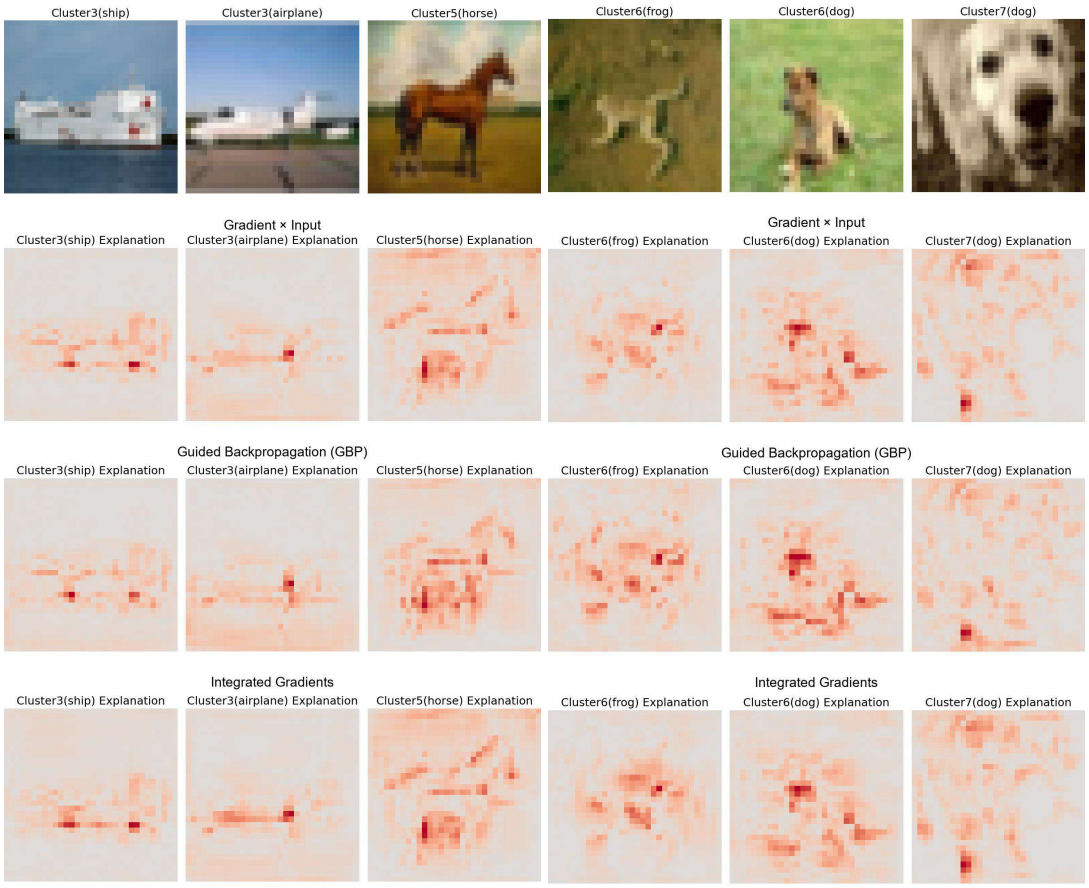


Figure 10: Explanation of images from different clusters. The results show that images from the same cluster that even have different labels still have similar saliency maps on various explanation methods. Besides, images with the same label from different clusters still have different explanations. These results show that our method can sample the most representative subset of explanations.

Table 9: Test results of models trained by Wide ResNet network and MobileNet network on various data sets according to four training methods. The results presented indicate that the performance of models trained using the Wide ResNet network and MobileNet network on different datasets suggests that there is no positive correlation between the model's explanation robustness and classification robustness achieved through the SEP_{pos} and SEP_{neg} training methods, as compared to the MAT training method.

	Wide ResNet				MobileNet			
MNIST								
Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Adv Acc	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{adv} (%)	Acc_{adv} (%)
Normal	267.050	206.194	99.58	0.00	287.061	188.700	99.08	0.02
MAT	842.648	736.839	98.92	**82.82**	4328.176	3356.135	98.29	94.19
SEP_pos	109.383	99.891	99.01	82.77	319.629	273.256	98.36	**94.25**
SEP_neg	937.845	**744.698**	98.87	82.71	8134.157	**4454.656**	98.33	94.23
FMNIST								
Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{adv} (%)	Acc_{adv} (%)
Normal	120.037	69.593	92.79	0.00	180.159	103.941	91.93	0
MAT	328.817	257.523	78.10	**68.26**	4470.448	3571.210	68.72	57.19
SEP_pos	109.996	74.324	77.69	67.79	236.547	172.200	65.11	57.42
SEP_neg	398.006	**304.927**	78.21	68.05	6032.190	**4809.288**	66.86	**58.16**
CIFAR10								
Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{adv} (%)	Acc_{adv} (%)
Normal	17.920	8.029	85.47	0.16	14.797	6.551	77.48	0
MAT	41.513	27.136	60.01	24.22	21.502	13.223	51.51	**23.81**
SEP_pos	26.343	16.217	59.87	24.89	14.756	7.907	49.91	23.27
SEP_neg	43.278	**27.575**	60.15	25.08	26.811	**16.420**	35.43	15.30
CIFAR100								
Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{adv} (%)	Acc_{adv} (%)
Normal	13.677	5.606	59.13	0	17.015	9.351	43.91	0
MAT	30.027	18.389	36.69	**16.12**	20.054	10.836	21.19	8.64
SEP_pos	22.046	13.704	33.88	13.19	15.234	8.510	21.82	**10.05**
SEP_neg	31.889	**20.045**	35.74	15.55	21.544	**13.843**	21.35	7.88

Table 11: Test results for transferability of explanation robustness. Models are trained with Gradient x Input and tested on different explanation methods.All models are trained on CIFAR10. Even if the interpretation methods during training and testing are different, comparing the training results of our proposed method with the AT training method of the corresponding configuration in Table 3, we can still draw our previous conclusions, which also shows that our conclusions are transferable.

	ConvNet		ResNet18	
Train:Gradient X Input, Test:Gradient				
Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)
SEP_{pos}	3.054	1.901	9.555	5.903
SEP_{neg}	15.093	9.513	55.526	33.176
Train:Gradient X Input, Test:Integrated_Grad				
Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)
SEP_{pos}	3.767	2.404	9.209	6.720
SEP_{neg}	17.066	10.923	58.730	38.433

	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{adv} (%)
Normal	114.70	63.52	0.00
AT	1281.71	742.43	19.36
SEP_{pos}	287.64	156.16	17.63
SEP_{neg}	1427.33	905.25	17.44

Table 13: Experiments for ImageNet on ResNet18. The results are aligned with the conclusion made in the main paper.

Table 12: The test results of the model trained using the TRADE training method, combined with our approach. The findings indicate that when we apply our method to TRADE, an alternative adversarial training method distinct from MAT, we can still deduce that classification robustness and interpretation robustness are not inherently interconnected.

ConvNet				
CIFAR10, TRADE Weight:5				
Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
TRADE	18.278	11.470	64.5	33.98
TRADE + SEP_pos	3.878	2.285	63.84	33.85
TRADE + SEP_neg	19.781	12.424	64.37	34.07
ConvNet				
CIFAR10, TRADE Weight:1				
Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
TRADE	17.271	10.965	72.63	28.31
TRADE + SEP_pos	4.089	2.296	72.41	28.20
TRADE + SEP_neg	18.504	11.662	72.90	28.34
ResNet18				
CIFAR10, TRADE Weight:5				
Method	$\mathcal{L}_e^{\text{start}}$ ($\times 10^{-7}$)	$\mathcal{L}_e^{\text{end}}$ ($\times 10^{-7}$)	Acc_{clean} (%)	Acc_{adv} (%)
TRADE	18.278	11.469	64.50	33.98
TRADE + SEP_pos	12.232	7.527	63.49	34.93
TRADE + SEP_neg	22.571	14.881	63.42	33.30

Towards Uncovering How Large Language Models Work: An Interpretability Perspective

Haiyan Zhao[1] , **Fan Yang**[2] , **Bo Shen**[1] , **Ali Payani**[3] , **Himabindu Lakkaraju**[4] , **Mengnan Du**[1]
[1]New Jersey Institute of Technology [2]Wake Forest University
[3]Cisco Research [4]Harvard University
{hz54, bo.shen, mengnan.du}@njit.edu, yangfan@wfu.edu, apayani@cisco.com, hlakkaraju@hbs.edu

Abstract

Large language models (LLMs) have shown remarkable performance in tackling natural language tasks, yet the internal mechanisms that enable their impressive generalization and reasoning abilities remain opaque. This lack of transparency presents significant challenges in fundamentally eliminating undesirable behaviors such as hallucinations and toxicity, hindering the safe and beneficial deployment of LLMs. This survey paper aims to uncover the internal working mechanisms underlying LLM functionality through the lens of explainability. First, we review how knowledge is encoded within LLMs via mechanistic interpretability techniques. Then, we summarize what knowledge is embedded in LLM representations by leveraging probing techniques and representation engineering. Additionally, we investigate the training dynamics to explore models' generalization abilities through grokking and memorization. Finally, we explore how the insights gained from these explanations can further enhance LLM performance through model editing, improve efficiency through pruning, and better align with human values.

1 Introduction

Large language models (LLMs) have led to tremendous advancements in natural language understanding and generation, achieving state-of-the-art performance in a wide array of real-world tasks. Despite their superior performance across various tasks, the "how" and "why" behind their generalization and reasoning abilities are still not well understood. This lack of understanding poses several challenges. First, LLMs frequently generate hallucinations and factually incorrect output, which complicates efforts to improve their performance. Second, as LLMs become more powerful, problems surrounding potential toxicity, unfairness, and dishonesty threaten to spread misinformation, promote biased or harmful views, and even compromise the safety and wellbeing of society. Therefore, there is an urgent need to fully understand the inner workings of LLMs to address these issues. Gaining insights into how these models operate is a crucial first step towards developing robust safeguards and ensuring their responsible deployment, although our understanding is still in the very early stages.

In this paper, we provide a systematic overview of the existing literature that uncovers the internal working mechanisms of LLMs using explainability techniques (Figure 1). First, we provide a summary of findings on how knowledge is encoded within the architecture of trained LLMs. The explainability technique, mechanistic interpretability (MI), is promising in explaining models' internals at the level of neurons, circuits and attention heads with activations and weights [Saphra and Wiegreffe, 2024]. Second, we examine what knowledge is encoded internally in intermediate representations. To this end, representation engineering (RE) is adopted to explain specific behaviors of models, such as dishonesty, by analyzing hidden/intermediate representations. Specifically, RE focuses on identifying patterns for certain behaviors through probing-based methods and employs them to mitigate undesired behaviors by steering models at inference time [Zou *et al.*, 2023]. Third, we inspect the model training process to understand the development of generalization abilities. Finally, we review how insights from the aforementioned analysis help us improve models in terms of higher performance through model editing, better efficiency through pruning, and better alignment with human values.

Our work differs from existing survey articles on the explainability of LLMs [Zhao *et al.*, 2024; Ferrando *et al.*, 2024], which either summarize explainability techniques or discuss their utilities. In contrast, our goal is to review recent studies that provide insights into trained LLMs and their dynamic training processes. By focusing on model components, representation space, and training processes, we aim to synthesize a line of work particularly focused on uncovering how LLMs function and identify the factors that contribute to their reasoning abilities. Beyond reviewing insights on the inner working mechanisms of LLMs, we further explore how these insights are employed to enhance model performance and benefit humanity.

2 Transformer-based LLMs

Before delving into the techniques and insights about how LLMs work, it is essential to establish a foundational understanding of the LLM architecture. This section introduces key components of the decoder-only LLMs, which have been the

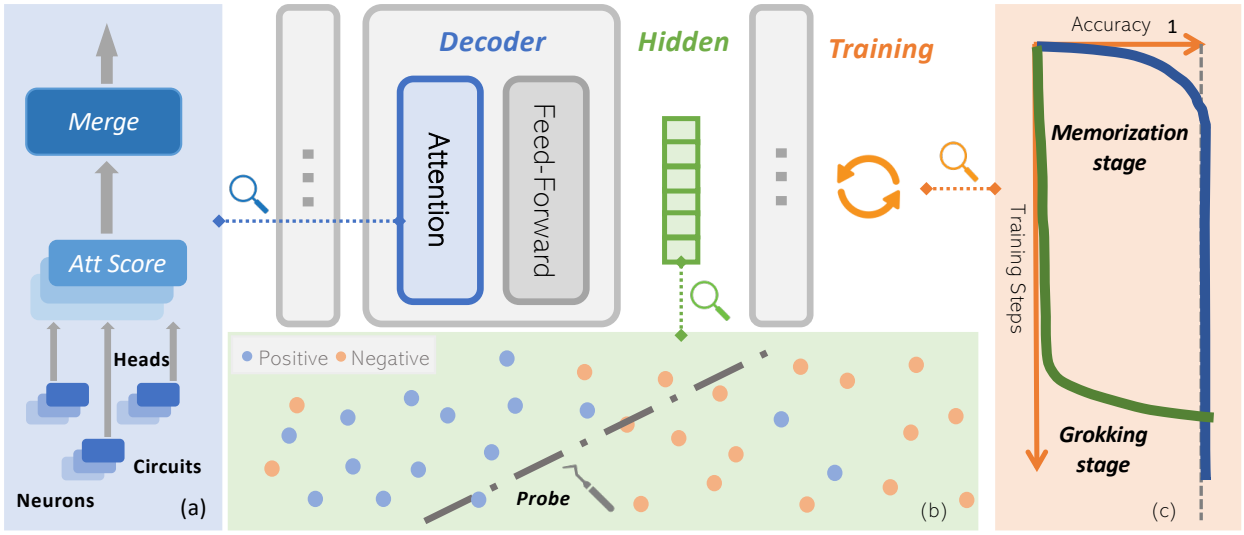


Figure 1: In this work, we review existing progress on how LLMs work, including: (a) how knowledge is encoded within model components; (b) what knowledge is encoded in intermediate representations; and (c) how generalization abilities are achieved during the training process.

dominant architecture for LLMs in recent years [Vaswani *et al.*, 2017].

2.1 Decoder-only LLM Architecture

Decoder-only (autoregressive) architecture has been widely used in popular LLMs such as GPT-3.5, GPT-4, Gemma-2, and Llama-3.1. A decoder-only Transformer model typically consists of the following main components (see Figure 2 (a)):

- *Input Embedding Layer*: This layer converts input tokens (words or subwords) into embeddings/representations. Since Transformers don't inherently understand sequence order, positional encodings are added to the input embeddings to provide position information for each token.
- *Multiple Transformer Layers*: The core of the model consists of a stack of identical Transformer layers. Each layer contains: a) Multi-head Self-attention: This allows the model to attend to different parts of the input sequence when processing each token. b) Feed-forward Neural Network: A simple fully connected network. c) Layer Normalization: Applied after each sub-layer to stabilize the learning process. d) Residual Connections: These skip-connections help in training deeper networks by allowing gradients to flow more easily through the model.
- *Output Layer*: The final layer typically projects the representations onto the vocabulary space, producing logits for each token in the vocabulary.

In autoregressive models, input sequences are processed token by token, with each token attending to all previous tokens in the sequence via the self-attention mechanism. During training, the model learns to predict the next token given the previous tokens. At inference time, the model generates each next token by repeatedly sampling from its predicted probability distribution and feeding this predicted token back as input for the next step.

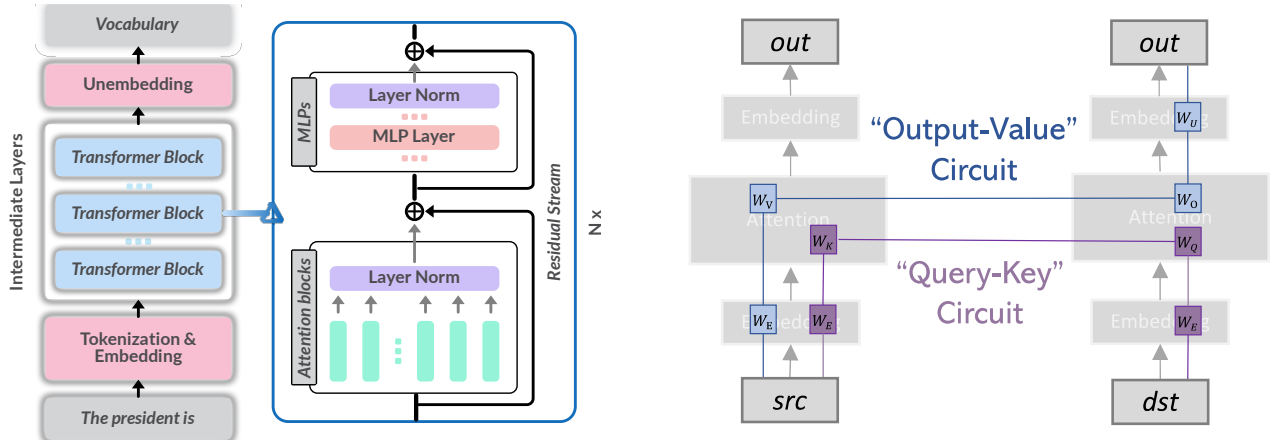
2.2 Intermediate Representations

Intermediate representations, also known as hidden representations, are a crucial component in understanding the inner workings of LLMs. The granularity of these representations can vary from the attention head level, where individual heads may specialize in specific patterns, to the layer level, where each layer's output represents a higher-level abstraction of the input. In this work, we refer to intermediate representations as the outputs from "Transformer block" layers (as illustrated in Figure 2 (a)). These layers include both outputs from attention blocks capturing contextual information through self-attention mechanisms, and outputs from feed-forward layers which further transform and process this information.

Understanding these representations is vital as they offer insights into how models transform information at different layers and reveal what types of knowledge or patterns models have learned with probing tools [Belinkov, 2022]. They have also been extensively used in representation engineering to steer model behaviors. In the following sections, particularly in Section 4, we will explore how these representations are employed to interpret world knowledge, factual information, and even unintended biases learned in models.

3 How is Knowledge Encoded in Model Architectures?

LLMs' emergent abilities are remarkable, but the underlying mechanisms enabling models to learn vast amounts of knowledge remain unclear. To fully understand LLMs, recent studies have been focused on utilizing MI to reverse engineer LLMs at a more subtle level, such as neurons and attention heads. MI emphasizes understanding the causal mechanisms within models, namely the relationship between model components and their behaviors [Saphra and Wiegreffe, 2024]. In this section, we review studies analyzing functionalities of



(a) Transformer architecture (b) Circuit

Figure 2: (a) The Decoder-only LLM Architecture. (b) An illustration of a Transformer circuit. *src* and *dst* denote previous tokens and the next predicted token respectively. In the toy model, input tokens are first sent to the embedding layer, then pass through the attention block. The intermediate output is further passed into unembedding layer to decode the destination token.

neurons, circuits, and attention heads with toy models and summarize how LLMs process information and make decisions. Notably, foundational discoveries in these simplified models have successfully generalized to explain real-world LLM behaviors, validating concepts like induction heads and driving the development of Sparse Autoencoders (SAEs).

3.1 Neurons

Neurons in LLMs are the fundamental units of computation. They receive data from inputs and produce output through activation functions. When a neuron's activation value is high after processing its inputs through the activation function, we consider the neuron to be "activated". Neurons can be activating on multiple unrelated terms (polysemantic) or a single term (monosemantic) [Olah *et al.*, 2020].

Polysemanticity

Polysemanticity presents a challenge in mechanistically understanding how models operate. *Superposition* appears to be a key cause of polysemantic nature [Olah *et al.*, 2020]. Superposition describes the phenomenon where a feature can be spread across multiple neurons, while a single neuron can simultaneously be involved with multiple features.

Some researchers suggest that superposition originates from an excessive number of features compared to neurons [Olah *et al.*, 2020]. In ReLU networks, Elhage et al. [2022] find that superposition enables the representation of additional features by tolerating some interference. In certain cases, it even facilitates computations such as the absolute value function and logical AND [Hänni *et al.*, 2024]. Others argue that polysemanticity can arise incidentally due to multiple training factors including regularization and neural noise [Lecomte *et al.*, 2024]. Experiments show that feature collisions, introduced through random initialization, consistently result in polysemantic neurons, even when the number of neurons exceeds the number of features [Lecomte *et al.*, 2024]. Another study examines polysemanticity through the lens of the "feature capacity", which denotes the fraction of embedding dimensions consumed by a feature [Scherlis *et al.*, 2022]. Through the analysis of toy models, this work indicates that features are represented based on their importance in reducing loss. More important features are allocated their own dimensions, while less critical ones tend to be polysemantic [Scherlis *et al.*, 2022; Gurnee *et al.*, 2023]. Gurnee et al. [2023] show that early layers tend to represent many features in superposition, while middle layers include dedicated neurons to represent high-level features.

Monosemanticity

Monosemantic neurons are much easier to interpret. Investigating the factors that enhance monosemanticity is essential for model interpretation. Jermyn et al. [2022] use toy models to reveal that changing the loss minima could improve monosemanticity. Such loss minimum usually coexists with negative biases. However, in practice building a purely monosemantic model is infeasible due to the unmanageable loss. Another line of studies seeks to disentangle superposition to reach a monosemantic understanding using sparse autoencoder (SAE). Instead of focusing on a single neuron, SAE considers layerwise neurons [Cunningham *et al.*, 2024]. It learns sparse activation directions representing monosemantic feature through dictionary learning. However, the effectiveness of this approach remains unclear. Templeton et al. [2024] find that SAEs produce features representing both low-level and high-level concepts, such as code errors and hateful bias-related features. Further, Ferrando et al. [2025] show that SAEs identify features for entity recognition.

3.2 Circuits

Circuit analysis was originally proposed to reverse engineer vision models [Olah *et al.*, 2020]. Taking car classification as an example, researchers suggest that some fundamental features such as edge detectors are learned in early layers. These

features are then combined through weights to form circuit units like wheel detectors in later layers. This viewpoint is supported by evidence from a few interpretable circuits performing specific functions such as curve detection and symmetric transformations of basic features, including copying, coloring, and rotation.

However, Transformer models present new challenges with their unique architecture. To address these challenges, a mathematical framework specifically for *transformer circuits* has been proposed [Elhage *et al.*, 2021]. It simplifies the complex architecture of LLMs by focusing on toy models composed of attention blocks with no more than two layers. Typically, two-layer toy models ensure preserving all necessary components of transformer models including input embedding, residual stream, attention layers, and output embeddings, while minimizing architectural complexity.

Attention layers communicate by reading information from the residual stream and then writing their output back. Each attention head operates independently and in parallel, with input embedding matrix W_E and output unembedding matrix W_U. These heads consist of key, query, output, and value weights, represented as W_K, W_Q, W_O and W_V. There are two types of circuits: i) "query-key" (QK) circuits formed by $W_Q^T W_K$; ii) "output-value" (OV) circuits composed of $W_U W_{OV}^h W_E$, where h denotes attention head, as shown in Figure 2 (b). The QK circuits are essential for models to recall and retrieve information from earlier context, determining which previous token to copy information from. The computed attention score $W_E^T W_{QK}^h W_E$ indicates how much a destination token attends to a source token. The OV circuits, in turn, determine how the source token influences the output logits [Elhage *et al.*, 2021].

By contextualizing the weights in circuits, research shows that circuits are essentially linear or bilinear functions on tokens. Specifically, Transformers with no layers can model bigram statistics, predicting the next token from the source token. Adding one layer allows the model to capture both bigram and "skip-trigram" patterns. Interestingly, with two layers, Transformer models give rise to "*induction head*" in the second layer and beyond (Section 3.3). These heads are typically composed of heads from their previous layer and are useful in predicting the next token [Elhage *et al.*, 2021].

Beyond the above-mentioned theoretical analysis on toy models, circuits implementing specific functions have been identified in real-world LLMs. For example, a circuit composed of 26 attention heads in GPT-2 small has been found to enable indirect object identification tasks by transmitting information from name tokens to the final outputs. Other research has identified circuits in GPT-2 that perform greater-than computations using a set of MLPs.

3.3 Attention heads

A special type of attention head called *induction head* is considered critical in enabling in-context learning abilities within LLMs [Brown *et al.*, 2020], due to their co-occurrence and causal relations [Olsson *et al.*, 2022; Chan *et al.*, 2022]. Induction heads are circuits that complete patterns through prefix matching and copying previously occurred sequences [Olsson *et al.*, 2022]. They comprise two heads: the first attention head from the previous layer attends to previous tokens that are followed by the current token, achieving prefix matching and providing the attend-to token (the token following the current token). The second head copies the attend-to token and increases its output logits. Specifically, this means that if models have encountered patterns such as "[A*][B*]" given the current token "[A]", they can predict "[B]". Beyond the simple example, long prefix matching involving three consecutive tokens has also been observed [Chan *et al.*, 2022]. Consequently, layers with induction heads possess more sophisticated in-context learning abilities than simple copying. However, this theory requires further research on real-world LLMs.

Other functional heads have also been found. For example, Gould et al. [Gould *et al.*, 2024] discovered successor heads that increment tokens like numbers in a natural order. Another study reveals how factual associations are stored and extracted within LLMs [Geva *et al.*, 2023]. A fact association consists of a subject and an attribute. The subject is enriched with subject-related attributes at the early MLP sublayers, which is then propagated to the prediction. The prediction representation "queries" the enriched subject to extract attributes via attention heads. Moreover, Chughtai et al. [2024] suggest that there are also mixed heads containing information from both subject and relation. The subject heads, mixed heads, and relation heads work additively to elicit the outputs.

4 What Knowledge is Encoded in Intermediate Representations?

In this section, we present an in-depth review of the knowledge encoded by *LLM representations*, including world knowledge and factual knowledge captured by models. We examine how factors such as layer depth and model scale impact the encoding process.

4.1 Probing World and Factual Knowledge

Probing techniques play a crucial role in revealing insights into world knowledge and factual knowledge encoded within models. Specifically, they identify key directions within the representation space that are essential for understanding model behaviors and learned knowledge.

Recent studies have demonstrated that LLMs can learn world models and encode them in their representations for specific tasks. One study successfully uses non-linear probes to uncover Othello board state representations within models [Li *et al.*, 2023]. It demonstrates models' ability to track board states and make predictions without explicit instruction. Furthermore, Nanda et al. [2023b] find that linear representation structures can also perform well on predictions by probing the board state at each timestamp through "my color" and "opponent's color". These findings reveal how models naturally perceive the world, which may differ from human perception. Additionally, by analyzing representations of spatial datasets, Gurnee et al. [2024] demonstrate the model's ability to learn linear representations of space and time across multiple levels. LLMs are also capable of encoding factual knowledge. Marks et al. [2024] craft self-curated true/false datasets to study the geometry of representations of true/false

statements derived from models' residual stream. The data are linearly separable under principal component analysis (PCA). The identified truth directions are used to mediate models' dishonest behaviors locally. Another research direction explores toxicity-related vectors within MLPs through singular value decomposition (SVD). The identified dimensions can be subtracted to achieve efficient mitigation [Lee *et al.*, 2024].

Function vectors have also been discovered within LLMs' attention heads of LLMs, triggering specific task execution across diverse inputs. For example, Todd et al. [2024] find that these function vectors appear in various in-context learning tasks and can execute related tasks even with zero-shot inputs. Additionally, causal interventions at the neuron level help identify individual neurons encoding spatial coordinates and temporal information [Gurnee and Tegmark, 2024].

Lastly, representations has been found to be associated with undesirable LLM behaviors, including dishonesty, toxicity, hallucination, safety concerns. Research has identified specific directions in the representation space that contribute to these behaviors. These directional vectors can be added to the representation space during inference time to guide model behavior without modifying the model's parameters [Zou *et al.*, 2023; Li *et al.*, 2024; Azaria and Mitchell, 2023; Hernandez *et al.*, 2024]. This technique of modifying model behavior by manipulating the representation space is known as representation steering or representation intervention, which provides a lightweight and flexible approach to controlling LLM outputs without the need for model retraining or parameter updates.

4.2 Role of Layer Depth and Model Scale

The influence of layer depth on representation analysis has emerged as a compelling research direction. A particularly intriguing finding is the concentration of well-learned knowledge in the middle layers of LLMs. For example, Gurnee et al. [2024] demonstrate that space and time representations achieve optimal quality within the first half of layers across LLMs. Additionally, function vectors with strong causal effects are predominantly found in the middle layers of LLMs, while their effects become negligible in deeper layers [Todd *et al.*, 2024]. Furthermore, research indicates that simpler tasks are mastered in earlier layers, while complex tasks require deeper layers for effective learning [Jin *et al.*, 2025; Ju *et al.*, 2024]. However, the underlying mechanisms of this phenomenon remain unclear.

As predicted by scaling laws, model capabilities increase as models grow larger. Gurnee et al. [2024] show that space and time representations become more precise with model scaling. However, internal mechanisms driving this improved performance during scaling remain poorly understood.

5 How is Generalization Ability Achieved During Training?

In this section, we examine how models develop generalization ability during the training process. We focus on two important phenomena associated with generalization: grokking and memorization.

5.1 Understanding Grokking

Grokking describes a phenomenon where models suddenly demonstrate improved validation accuracy after a period of severe overfitting in over-parameterized neural networks [Power *et al.*, 2022]. This sudden surge in validation accuracy is generally interpreted as the acquisition of generalization ability.

A Data Perspective

Experiments on a two-layer decoder-only Transformer network have demonstrated that grokking is influenced by multiple factors, including data characteristics, representations, and regularization. Research shows that smaller datasets require more optimization steps for grokking to occur [Power *et al.*, 2022], while increasing the number of samples reduces the steps needed for generalization [Zhu *et al.*, 2024]. The minimal data requirement for grokking is tied to the threshold number of data points needed to learn robust representations. However, the massive datasets used in LLMs make grokking phenomena less observable [Zhu *et al.*, 2024]. Additionally, regularization techniques can accelerate the onset of grokking, with weight decay proving particularly effective in enhancing generalization capabilities. Wang et al. [2024] have shown that Transformers can develop implicit reasoning abilities exclusively through grokking. Moreover, their research reveals that data distribution plays a more crucial role in achieving grokking than data size alone.

Weight Norms

When analyzing models' final layer weight norms during late-stage training, researchers have observed a phenomenon called the *slingshot mechanism*, which occurs only in the absence of regularization. This mechanism is characterized by the simultaneous occurrence of norm growth and training loss spikes, with grokking observed at the onset of these slingshots [Thilak *et al.*, 2022]. The manifestation of both the slingshot effect and grokking can be controlled by modifying optimizer parameters, particularly when using certain adaptive optimizers. However, the universality of this observation across different scenarios remains uncertain.

Another significant finding is the *LU mechanism*, which describes the dynamics between loss and weight norms [Liu *et al.*, 2023]. In algorithmic datasets, researchers have identified an L-shaped training loss curve and a U-shaped test loss reduction relative to weight norms, suggesting an optimal range for weight norm initialization. However, this pattern doesn't readily apply to real-world machine learning tasks, which typically require large initialization values and minimal weight decay. However, Fan et al. [2024] argue that feature ranks and linear probing accuracy may serve as more reliable indicators of phase transition compared to weight norm measurements.

Test Loss

Double descent describes a pattern in which a model's test accuracy at the log level follows a distinctive trajectory: initial improvement, followed by a decline due to overfitting, and finally a secondary increase as generalization abilities develop [Nakkiran *et al.*, 2020]. This pattern becomes more

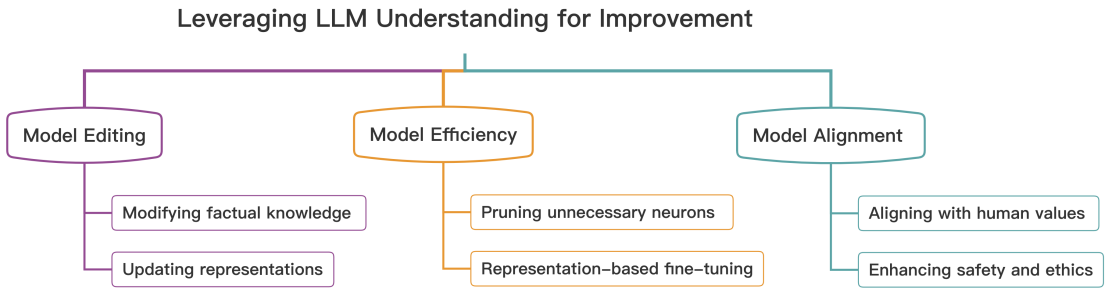


Figure 3: Leveraging understanding of LLMs for targeted improvements. We illustrate three key areas: Model Editing (modifying factual knowledge and updating representations), Model Pruning (improving inference efficiency and reducing model size), and Model Alignment (aligning with human values and enhancing safety and ethics).

pronounced when examining test loss. Researchers have developed a unified framework that integrates grokking and double descent, viewing them as different manifestations of the same underlying process [Davies *et al.*, 2022]. The framework suggests that the transition to generalization stems from slower pattern learning, a perspective supported by subsequent research [Kumar *et al.*, 2024]. This transitional phenomenon has been observed to manifest at both the epoch and model levels.

5.2 Memorization

Memorization refers to the phenomenon where models rely on statistical features rather than causal relationships for predictions. In LLMs, generalization abilities take precedence over memorization, particularly when evaluating a model's reasoning capabilities. Understanding memorization is therefore crucial for improving model performance.

Studies of two-layer neural models have shown that memorization can exist alongside generalization. Memorization can be reduced through either neuron pruning or regularization techniques. While different regularization methods may have distinct learning objectives, they all contribute to the development of better representations. The terminal phase of training encompasses two distinct stages: i) the grokking process and ii) the decay of memorization learning [Doshi *et al.*, 2024]. However, the underlying mechanisms driving this process remain unclear. Furthermore, the hypothesis that regularization is central to this process has been challenged, particularly given observations of grokking occurring without regularization [Kumar *et al.*, 2024].

Interestingly, recent research hypothesizes that memorization forms an integral phase of grokking [Nanda *et al.*, 2023a]. This study identifies three distinct stages in the grokking process: memorization, circuit formation, and memorization cleanup. Through analysis of model weights, the researchers identified an algorithm using Discrete Fourier Transforms and trigonometric identities to achieve modular addition. Xie et al. [2024] illustrate the coexistence of heavy memorization and improved generalization after fine-tuning. Wang et al. [2025] observe that as model size increases, factual question answering exhibits increased memorization, while machine translation and reasoning tasks show enhanced generalization. Moreover, Menta el al. [2025] argue that memorization primarily occurs in deeper attention heads, while earlier layers play a crucial role in generalizations.

6 How to Make Use of The Insights?

Building on the insights from previous sections, we examine how our in-depth understanding of LLMs can be leveraged to enhance their performance through editing, improve efficiency via pruning, and better align them with human values and preferences (Figure 3).

6.1 Model Editing for Better Performance

Recent research has made significant strides in model editing. Meng et al. [2022] demonstrate the ability to edit factual knowledge by modifying specific MLP neuron weights, successfully altering neural computations related to factual recall. Stoehr et al. [2024] further reveal that memorized paragraphs can be identified through high-gradient weights in attention heads of lower layers. This research direction focuses on localizing specific attention heads and fine-tuning them to unlearn memorized knowledge, showing promise for enhancing privacy protection in LLMs. However, significant challenges remain. Hu et al. [2024] demonstrate that lifelong knowledge editing often fails due to superposition, potentially affecting unrelated knowledge. Similarly, Gu et al. [2024] identify potential damage to models' general capabilities from editing interventions.

The representation space offers another avenue for model editing, as facts are encoded within these representations. While most current research focuses on modifying representations during inference time, the impact of permanent modifications remains largely unexplored. A breakthrough study proposes a more precise method for editing model representations to alter output distributions [Hernandez *et al.*, 2024]. Rather than merely adding steering vectors during inference, this approach directly modifies entity embeddings to trigger targeted outputs, resulting in shifted entity positions in the embedding space that causally influence model generations.

6.2 Improving Model Efficiency

In contrast to deciphering models' inner workings, one study examines the differences between pre-training and fine-tuning phases using MI tools. It reveals that fine-tuning preserves all capabilities learned during pre-training. The trans-

formations between pre-training and fine-tuning arise from "wrappers" in MLPs learned on top of models. These wrappers can be eliminated by pruning a few neurons or retraining on an unrelated downstream task [Jain *et al.*, 2024]. This discovery raises potential safety concerns regarding current alignment approaches.

Unlike pruning neurons, Representation Engineering (RE) directly manipulates representations without requiring optimizations or additional labeled data. RE has also proven effective in model pruning. Several studies show that fine-tuning models with representation engineering can achieve comparable or even better performance than state-of-the-art fine-tuning techniques. For instance, Wu et al. [2024a] employ forward passes from two topics and derive their difference vectors, which are then used during inference without additional fine-tuning. In contrast to conventional parameter-efficient fine-tuning (PEFT), representation-editing-based fine-tuning focuses on learning an additional group of trainable parameters to modify representations directly rather than altering models' parameters. The number of trainable parameters has been reduced by a factor of 32 compared to LoRA [Wu *et al.*, 2024a]. Another approach employs distributed alignment search to find a set of linear subspaces implementing interventions. This method outperforms most PEFT models across various tasks [Wu *et al.*, 2024b].

6.3 Model Alignment to Human Values

From a mechanistic perspective, practical applications often evaluate model alignments using various tools. Inspired by induction heads, Yang et al. [2023] measure bias scores of attention heads in pre-trained LLMs by comparing attention scores between biased and regular heads, effectively reducing gender bias through masking identified biased heads. Another study localizes attention heads responsible for deception using linear probing and activation patching. The researchers use intentionally designed prompts to instruct LLMs to be dishonest. Linear probes are trained to classify true/false activations of heads, and selected activations related to dishonest behaviors are then patched with those of honest behaviors to observe output changes. This method successfully locates multiple attention heads across five layers.

Recently, RE has emerged as a promising approach for detecting biases within representation space. A notable study suggests that MLPs operate on token representations to alter the output vocabulary distribution [Geva *et al.*, 2022]. After reverse engineering MLPs, researchers concluded that the output from each feed-forward layer can be viewed as sub-updates to output vocabulary distributions, effectively promoting certain high-level concepts. This insight has been successfully applied to mitigate toxicity levels in LLMs. Another line of research identifies multiple representation vectors within MLPs that encourage undesired model behaviors. These vectors are decomposed using singular value decomposition, enabling researchers to identify specific dimensions that contribute to toxicity.

7 Future Research Directions

In this section, we highlight several research directions that warrant further investigation by the research community.

7.1 Standardizing Evaluation and Benchmarks

A critical challenge in understanding LLMs is the lack of robust empirical validation for proposed theories like induction heads and circuit analysis, compounded by the absence of standardized evaluation metrics. The field needs agreed-upon criteria for mechanistic explanations and comprehensive interpretability benchmarks [Saphra and Wiegreffe, 2024] to enable systematic comparison of different approaches. However, creating these benchmarks faces unique challenges due to model complexity, absence of ground truth, and diverse architectures. Establishing a rigorous validation framework will require coordinated effort from the research community.

7.2 Scaling Interpretability Algorithms

As LLMs grow in size and complexity, scaling interpretability techniques becomes critical. We identify two key challenges: First, we need novel methods to analyze the intricate knowledge structures within LLMs, as current approaches reveal only a fraction of encoded knowledge. Second, existing explainability techniques, including MI tools, are primarily effective on simplified models, creating an urgent need to scale these algorithms to billion-parameter LLMs. This challenge involves both computational efficiency and adapting conceptual frameworks to handle emergent behaviors in large-scale models.

7.3 Source of Reasoning Ability

LLMs have demonstrated remarkable reasoning abilities that mirror human cognition, performing complex tasks from multi-step problem-solving to creative thinking. However, we still lack understanding of how these abilities emerge from architectural components and training dynamics. Key research directions to uncover these mechanisms include: 1) examining how LLM components contribute to reasoning processes, 2) analyzing how basic building blocks combine to produce complex reasoning, 3) investigating reasoning development during training, and 4) comparing LLM and human reasoning patterns to identify similarities and differences.

8 Conclusions

In this survey paper, we present a comprehensive overview of recent advances in uncovering the inner workings of LLMs through explainability techniques. Our review highlights significant findings about knowledge encoding within LLM architectures, including polysemantic neurons, functional circuits, and attention heads' role in in-context learning. Probing techniques and representation engineering have revealed LLMs' capacity to learn complex world models and encode factual knowledge, while studies of training dynamics have illuminated how generalization abilities emerge. These insights have enabled practical applications in model editing, efficient fine-tuning, and bias mitigation. Despite this progress, challenges persist due to LLMs' complexity and scale, necessitating future work on standardized evaluation metrics, scaling interpretability techniques, and understanding high-level reasoning abilities.

References

[Azaria and Mitchell, 2023] Amos Azaria and Tom Mitchell. The internal state of an llm knows when it's lying. In *Findings of EMNLP*, pages 967–976, 2023.

[Belinkov, 2022] Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

[Brown *et al.*, 2020] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, et al. Language models are few-shot learners. *NeurIPS*, 2020.

[Chan *et al.*, 2022] Lawrence Chan, Adrià Garriga-Alonso, Nicholas Goldwosky-Dill, Ryan Greenblatt, et al. Causal scrubbing, a method for rigorously testing interpretability hypotheses. *AI Alignment Forum*, 2022.

[Chughtai *et al.*, 2024] Bilal Chughtai, Alan Cooney, and Neel Nanda. Summing up the facts: Additive mechanisms behind factual recall in llms. *arXiv preprint arXiv:2402.07321*, 2024.

[Cunningham *et al.*, 2024] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *ICLR*, 2024.

[Davies *et al.*, 2022] Xander Davies, Lauro Langosco, and David Krueger. Unifying grokking and double descent. In *NeurIPS Workshop ML Safety*, 2022.

[Doshi *et al.*, 2024] Darshil Doshi, Aritra Das, Tianyu He, and Andrey Gromov. To grok or not to grok: Disentangling generalization and memorization on corrupted algorithmic datasets. In *ICLR*, 2024.

[Elhage *et al.*, 2021] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021.

[Elhage *et al.*, 2022] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, et al. Toy models of superposition, 2022.

[Fan *et al.*, 2024] Simin Fan, Razvan Pascanu, and Martin Jaggi. Deep grokking: Would deep neural networks generalize better? *arXiv preprint arXiv:2405.19454*, 2024.

[Ferrando *et al.*, 2024] Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R Costa-jussà. A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv:2405.00208*, 2024.

[Ferrando *et al.*, 2025] Javier Ferrando, Oscar Obeso, Senthooran Rajamanoharan, and Neel Nanda. Do i know this entity? knowledge awareness and hallucinations in language models. In *ICLR*, 2025.

[Geva *et al.*, 2022] Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *EMNLP*, 2022.

[Geva *et al.*, 2023] Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In *EMNLP*, 2023.

[Gould *et al.*, 2024] Rhys Gould, Euan Ong, George Ogden, and Arthur Conmy. Successor heads: Recurring, interpretable attention heads in the wild. In *ICLR*, 2024.

[Gu *et al.*, 2024] Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, et al. Model editing harms general abilities of large language models: Regularization to the rescue. In *EMNLP*, 2024.

[Gurnee and Tegmark, 2024] Wes Gurnee and Max Tegmark. Language models represent space and time. In *ICLR*, 2024.

[Gurnee *et al.*, 2023] Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, et al. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023.

[Hernandez *et al.*, 2024] Evan Hernandez, Belinda Z Li, and Jacob Andreas. Inspecting and editing knowledge representations in language models. In *COLM*, 2024.

[Hu *et al.*, 2024] Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. Knowledge in superposition: Unveiling the failures of lifelong knowledge editing for large language models. *arXiv preprint arXiv:2408.07413*, 2024.

[Hänni *et al.*, 2024] Kaarel Hänni, Jake Mendel, Dmitry Vaintrob, and Lawrence Chan. Mathematical models of computation in superposition. In *ICML Workshop MI*, 2024.

[Jain *et al.*, 2024] Samyak Jain, Robert Kirk, Ekdeep Singh Lubana, Robert P Dick, Hidenori Tanaka, Edward Grefenstette, et al. Mechanistically analyzing the effects of fine-tuning on procedurally defined tasks. In *ICLR*, 2024.

[Jermyn *et al.*, 2022] Adam S Jermyn, Nicholas Schiefer, and Evan Hubinger. Engineering monosemanticity in toy models. *arXiv preprint arXiv:2211.09169*, 2022.

[Jin *et al.*, 2025] Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, et al. Exploring concept depth: How large language models acquire knowledge at different layers? In *COLING*, 2025.

[Ju *et al.*, 2024] Tianjie Ju, Weiwei Sun, Wei Du, Xinwei Yuan, Zhaochun Ren, and Gongshen Liu. How large language models encode context knowledge? a layer-wise probing study. In *COLING*, 2024.

[Kumar *et al.*, 2024] Tanishq Kumar, Blake Bordelon, Samuel J Gershman, and Cengiz Pehlevan. Grokking as the transition from lazy to rich training dynamics. In *ICLR*, 2024.

[Lecomte *et al.*, 2024] Victor Lecomte, Kushal Thaman, Rylan Schaeffer, Naomi Bashkansky, Trevor Chow, and Sanmi Koyejo. What causes polysemanticity? an alternative origin story of mixed selectivity from incidental causes. In *ICLR Workshop RA*, 2024.

[Lee *et al.*, 2024] Andrew Lee, Xiaoyan Bai, Itamar Pres, Martin Wattenberg, Jonathan Kummerfeld, and Rada Mihalcea. A mechanistic understanding of alignment algorithms: A case study on dpo and toxicity. In *ICML*, 2024.

[Li *et al.*, 2023] Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *ICLR*, 2023.

[Li *et al.*, 2024] Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *NeurIPS*, 36, 2024.

[Liu *et al.*, 2023] Ziming Liu, Eric J Michaud, and Max Tegmark. Omnigrok: Grokking beyond algorithmic data. In *ICLR*, 2023.

[Marks and Tegmark, 2024] Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. In *COLM*, 2024.

[Meng *et al.*, 2022] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *NeurIPS*, 35:17359–17372, 2022.

[Menta *et al.*, 2025] Tarun Ram Menta, Susmit Agrawal, and Chirag Agarwal. Analyzing memorization in large language models through the lens of model attribution. *arXiv preprint arXiv:2501.05078*, 2025.

[Nakkiran *et al.*, 2020] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, et al. Deep double descent: Where bigger models and more data hurt. In *ICLR*, 2020.

[Nanda *et al.*, 2023a] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, et al. Progress measures for grokking via mechanistic interpretability. In *ICLR*, 2023.

[Nanda *et al.*, 2023b] Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. In *ACL Workshop BlackboxNLP*, pages 16–30, 2023.

[Olah *et al.*, 2020] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020.

[Olsson *et al.*, 2022] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, et al. In-context learning and induction heads. *Transformer Circuits Thread*, 2022.

[Power *et al.*, 2022] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

[Saphra and Wiegreffe, 2024] Naomi Saphra and Sarah Wiegreffe. Mechanistic? *arXiv preprint arXiv:2410.09087*, 2024.

[Scherlis *et al.*, 2022] Adam Scherlis, Kshitij Sachan, Adam S Jermyn, Joe Benton, and Buck Shlegeris. Polysemanticity and capacity in neural networks. *arXiv preprint arXiv:2210.01892*, 2022.

[Stoehr *et al.*, 2024] Niklas Stoehr, Mitchell Gordon, Chiyuan Zhang, and Owen Lewis. Localizing paragraph memorization in language models. *arXiv preprint arXiv:2403.19851*, 2024.

[Templeton *et al.*, 2024] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, et al. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024.

[Thilak *et al.*, 2022] Vimal Thilak, Etai Littwin, Shuangfei Zhai, Omid Saremi, Roni Paiss, and Joshua Susskind. The slingshot mechanism: An empirical study of adaptive optimizers and the grokking phenomenon. In *NeurIPS Workshop HITY*, 2022.

[Todd *et al.*, 2024] Eric Todd, Millicent L Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. In *ICLR*, 2024.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017.

[Wang *et al.*, 2024] Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokking of implicit reasoning in transformers: A mechanistic journey to the edge of generalization. In *NeurIPS*, 2024.

[Wang *et al.*, 2025] Xinyi Wang, Antonis Antoniades, Yanai Elazar, Alfonso Amayuelas, Alon Albalak, et al. Generalization v.s. memorization: Tracing language models' capabilities back to pretraining data. In *ICLR*, 2025.

[Wu *et al.*, 2024a] Muling Wu, Wenhao Liu, Xiaohua Wang, Tianlong Li, et al. Advancing parameter efficiency in fine-tuning via representation editing. In *ACL*, 2024.

[Wu *et al.*, 2024b] Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. Reft: Representation finetuning for language models. In *NeurIPS*, 2024.

[Xie *et al.*, 2024] Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, et al. Large language interpolators can learn logical reasoning: A study on knights and knaves puzzles. In *NeurIPS Workshop MATH-AI*, 2024.

[Yang *et al.*, 2023] Yi Yang, Hanyu Duan, Ahmed Abbasi, John P Lalor, and Kar Yan Tam. Bias a-head? analyzing bias in transformer-based language model attention heads. *arXiv preprint arXiv:2311.10395*, 2023.

[Zhao *et al.*, 2024] Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, et al. Explainability for large language models: A survey. *ACM TIST*, 15(2):1–38, 2024.

[Zhu *et al.*, 2024] Xuekai Zhu, Yao Fu, Bowen Zhou, and Zhouhan Lin. Critical data size of language models from a grokking perspective. *arXiv preprint arXiv:2401.10463*, 2024.

[Zou *et al.*, 2023] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

Unifying Knowledge in Agentic LLMs: Concepts, Methods, and Recent Advancements

Lihui Liu
lihuil2@illinois.edu
Wayne State University
Detroit, Michigan, USA

Kai Shu
kai.shu@emory.edu
Emory University
Atlanta, Georgia, USA

Abstract

Large language models have demonstrated remarkable capabilities in text generation and problem solving, yet they continue to face fundamental challenges such as hallucination, lack of factual grounding, and limited reasoning reliability. The core of these issues lies the question of how LLMs acquire and use *knowledge*. While internal knowledge embedded in model parameters enables impressive generalization, it is often insufficient for up-to-date or domain-specific tasks. External knowledge integration, such as retrieval-augmented generation (RAG), provides grounding and factuality but introduces challenges of retrieval quality, latency, and reliability. Beyond these paradigms, recent advances in *agentic LLMs* extend models from passive generators to active problem solvers that can reason, plan, and interact with external tools. This survey provides a unified, knowledge-centric perspective on LLMs, organized along three complementary dimensions: (i) reactive: internal knowledge, (ii) lightly-active: external knowledge, and (iii) proactive: agentic knowledge utilization for reasoning and tool interaction. We provide a taxonomy of knowledge usage in LLMs, analyze their respective strengths and limitations, and highlight how these paradigms interact in real-world systems. Finally, we identify open challenges to facilitate future research.

CCS Concepts

• **Computing methodologies** → **Reasoning about belief and knowledge**; • **Information systems** → **Data mining**.

Keywords

Knowledge graph reasoning, neural symbolic reasoning, knowledge graph question answering

1 Introduction

Knowledge lies at the core of how large language models (LLMs) reason [15, 47, 52], generate responses [1, 45], and support decision-making [3, 53]. The billions of parameters in modern LLMs encode an impressive amount of internalized world knowledge acquired during large-scale pretraining. This internal knowledge enables LLMs to perform diverse tasks with few or even zero examples. However, such parametric knowledge is often incomplete, outdated, or unreliable, particularly when applied to specialized or dynamic domains. As a result, relying solely on the internal memory of an LLM limits its factual accuracy, interpretability, and trustworthiness when answering questions.

To overcome these limitations, recent research has sought to augment LLMs with external sources of knowledge—ranging from large unstructured corpora to structured databases and knowledge graphs—so that generated outputs can be grounded in verifiable evidence [12, 19, 40]. This line of work, broadly referred to as retrieval-augmented generation (RAG), has shown that explicit access to external information can substantially improve factuality, reasoning, and adaptability across domains. In parallel, the emergence of agentic LLMs [3, 26, 51, 53], which can autonomously plan, reason, and act through interaction with tools and environments, represents a new paradigm of knowledge use in action. Rather than passively generating text, such models leverage both internal and external knowledge to make decisions, execute plans, and continuously update their understanding through feedback and exploration.

Together, these developments reveal that knowledge is not merely a static component encoded in parameters but a dynamic process—one that governs how LLMs organize, access, and apply information to reason and act effectively. Yet, despite the growing body of research, existing studies tend to examine these directions in isolation. For instance, work on in-context learning (ICL) [4, 11, 47] primarily explores how models exploit internal knowledge through prompting and contextual adaptation. In contrast, retrieval-augmented approaches [12, 19, 40] focus on grounding model outputs with external evidence. More recent efforts on LLM-based agents [3, 27, 51] investigate tool use, planning, and interaction, but often from a systems or application-oriented perspective rather than from a unifying theory of knowledge utilization.

What remains missing in the current literature is a unifying conceptual framework that bridges these perspectives and offers a holistic understanding of how knowledge is represented, retrieved, and applied across different paradigms of large language model reasoning. While prior surveys have reviewed individual aspects—such as prompting and in-context learning, retrieval-augmented generation, or LLM-based agents—there is still no comprehensive synthesis that explains how these approaches collectively contribute to the evolving landscape of knowledge-centric intelligence.

This survey aims to fill that gap by proposing a knowledge-centric taxonomy that unifies three complementary modes of knowledge use in LLMs, ranging from internal recall to active reasoning and autonomous interaction (see Fig. 3). (1) Reactive use of internal knowledge — LLMs rely on their parametric memory and contextual reasoning to solve tasks based solely on internalized knowledge. (2) Lightly-active use of external knowledge — LLMs ground their generation in verifiable external sources via retrieval, database queries, or structured representations such as knowledge graphs. (3) Proactive use of knowledge in an agentic manner — LLMs integrate internal and external knowledge to plan, act, and adapt dynamically through interaction with the environment or external tools.

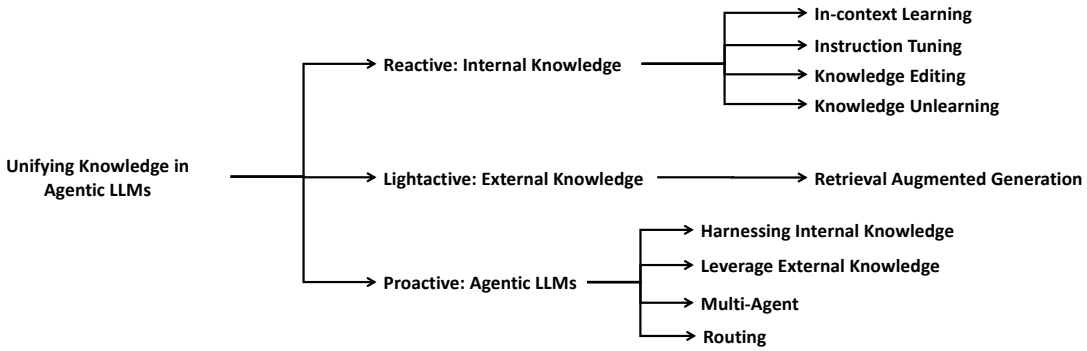


Figure 1: Overview of the survey: three paradigms of knowledge use in large language models.

Together, these three paradigms capture a continuum of knowledge utilization, ranging from passive recall to grounded reasoning and ultimately to autonomous, knowledge-driven behavior. This perspective illustrates how the role of knowledge in LLMs has evolved—from a static asset stored in model parameters to an active process that dynamically shapes perception, reasoning, and action.

Building upon this taxonomy, our survey systematically reviews recent advances across these three dimensions, discussing how knowledge is acquired during pretraining, dynamically retrieved or updated, and utilized for reasoning and planning in interactive environments. We further examine the interplay among these paradigms, identifying shared mechanisms such as memory management, reasoning chains, and feedback loops that unify them.

By organizing the literature around these three pillars, this survey provides a comprehensive and coherent understanding of how knowledge underpins the reasoning and decision-making capabilities of large language models. It also highlights key challenges and future directions for developing more reliable, explainable, and knowledge-driven AI systems that move beyond static text generation toward dynamic and trustworthy reasoning.

2 Foundations of Knowledge in LLMs

2.1 Background

Large Language Models (LLMs) are powerful deep neural networks built upon the Transformer architecture and trained on massive text corpora to model the probability distribution of natural language. Representative models such as GPT [1], PaLM [5], and LLaMA [45] are typically trained using an autoregressive objective, where the model learns to predict the next token in a sequence given its preceding context:

$$\max \sum_{t} \log P(x_t \mid x_{<t}),$$

with x_t representing the current token and $x_{<t}$ denoting all tokens before it. This self-supervised pretraining enables the model to implicitly learn grammar, semantics, world knowledge, and basic reasoning patterns from large-scale unlabeled data.

After pretraining, LLMs are typically adapted to downstream tasks through *supervised fine-tuning* [37]. In this stage, the model is trained on human-annotated input-output pairs (x, y), where x is a task-specific instruction or prompt, and $y = (y_1, \ldots, y_L)$ is the corresponding desired response. The model is optimized to maximize the conditional likelihood of the output sequence given the input:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{t=1}^{L} \log P_\theta(y_t \mid x, y_{<t}),$$

where $P_\theta(y_t \mid x, y_{<t})$ is the probability of generating token y_t based on the input and previous output tokens, and θ denotes the model parameters. This step helps the model follow instructions and perform specific tasks more reliably.

To further align model behavior with human preferences, values, and safety goals, LLMs are often refined via *Reinforcement Learning from Human Feedback (RLHF)* [37]. The RLHF pipeline begins with the collection of human preference data, where annotators rank several model-generated outputs for the same prompt. These rankings are used to train a *reward model* $r_\phi(x, y)$, which estimates the quality of a response y given a prompt x. Finally, the base model is fine-tuned using a reinforcement learning algorithm, commonly *Proximal Policy Optimization (PPO)* [39], to maximize the expected reward under the learned reward model:

$$\mathcal{L}_{\text{RLHF}}(\theta) = \mathbb{E}_{y \sim \pi_\theta(\cdot|x)} \left[r_\phi(x, y) \right],$$

where π_θ is the current policy (the language model), and ϕ represents the parameters of the reward model. PPO ensures stable optimization by penalizing large deviations from the original policy during updates.

Together, these stages—pretraining, supervised fine-tuning, and RLHF—form a standard training framework that enables LLMs to exhibit strong language understanding, instruction-following behavior, and alignment with human intent. Despite their generative capabilities, LLMs still face challenges in factual consistency, verifiable reasoning, and robustness, as their outputs are shaped by statistical patterns in data rather than explicit logical or grounded inference mechanisms.

2.2 Internal vs. External Knowledge

Large language models (LLMs) can acquire and utilize knowledge through two complementary channels: *internal knowledge* and *external knowledge*. Internal knowledge [11, 47, 54] refers to the information implicitly encoded within the model parameters during large-scale pretraining. This enables models to recall facts, linguistic structures, and common-sense reasoning without explicit access

to external resources. By contrast, external knowledge [12, 19, 40] refers to information retrieved or accessed from outside sources at inference time, such as unstructured corpora, structured knowledge bases, or multimodal databases. External knowledge provides grounding, verifiability, and adaptability, especially in dynamic or specialized domains where internal memory may be insufficient or outdated.

2.3 A Taxonomy of Knowledge Usage

We propose a compact taxonomy that categorizes how large language models (LLMs) use knowledge into three complementary paradigms: *(1) internal knowledge*, *(2) external knowledge*, and *(3) agentic knowledge use*. This taxonomy captures both *where* information resides—within or outside the model's parameters—and *how* it is operationalized, ranging from static recall to dynamic reasoning and autonomous interaction.

(1) Internal knowledge refers to the information encoded or accessed without invoking any external sources. It encompasses the model's parametric memory and contextual reasoning abilities, which arise from large-scale pretraining. Several mechanisms fall under this category: (a) *In-context learning (ICL)* enables the model to adapt its behavior from examples or demonstrations provided directly in the prompt at inference time, functioning as a form of temporary contextual learning. (b) *Instruction tuning* aligns model behavior with human intent by fine-tuning on large collections of instruction–response pairs, effectively improving generalization across tasks. (c) *Knowledge editing* and *unlearning* techniques further refine internal knowledge by locally modifying or removing specific facts in the model parameters to enhance factual consistency, correct errors, or comply with privacy constraints. Together, these mechanisms illustrate how internal representations enable LLMs to reason reactively and flexibly using their intrinsic knowledge base.

(2) External knowledge encompasses methods that augment or ground LLM outputs with verifiable information residing outside their parameters. The most representative paradigm here is (a) *retrieval-augmented generation (RAG)*, in which the model retrieves relevant documents, knowledge-graph triples, or database entries at inference time and conditions its response on the retrieved evidence. (b) In addition to retrieval, emerging approaches explore direct *database querying* and integration with structured symbolic stores, allowing the model to reason over factual and relational information explicitly. (c) Some systems further employ *hybrid retriever–generator architectures*, where retrieval and generation are co-trained or iteratively refined for tighter coupling between knowledge access and use. Overall, these methods mitigate the limitations of parametric memory—such as incompleteness or temporal drift—by providing factual grounding and interpretability while maintaining generative flexibility.

(3) Agentic knowledge use represents the most dynamic form of knowledge utilization, in which LLMs function as autonomous decision-making entities that plan, reason, and act. Within this paradigm, (a) models *leverage internal reasoning*—such as chain-of-thought or instruction-tuned behaviors—as a substrate for high-level planning and reflection; (b) they *invoke external tools, APIs, or retrieval modules* (RAG is a special case of tool use) on demand to supplement missing knowledge or verify hypotheses; and (c) they often engage in *multi-agent collaboration*, coordinating with specialized agents such as retrievers, verifiers, or planners to decompose and solve complex tasks. Some advanced systems further support *dynamic routing*, where the model decides which knowledge source or agent to consult at each stage of reasoning. This paradigm embodies proactive knowledge use, integrating perception, reasoning, and action into an adaptive loop that extends beyond static text generation.

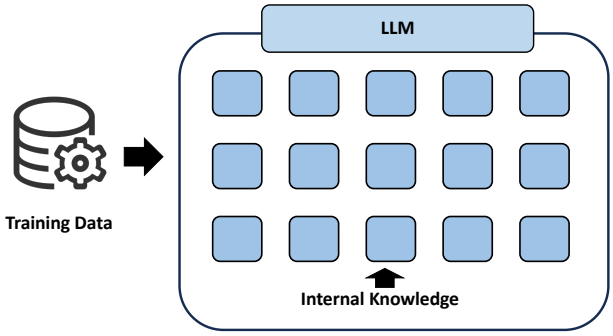


Figure 2: Internal Knowledge.

Overall, this three-part taxonomy—covering internal, external, and agentic paradigms—offers a unified lens for understanding how LLMs represent, access, and apply knowledge. It highlights the continuum from passive recall to autonomous reasoning, emphasizing the trade-offs between parametric memory, retrieval-based grounding, and agentic control. By articulating these categories and their mechanisms, this framework provides conceptual clarity for analyzing existing methods and guiding future research toward more reliable, explainable, and knowledge-driven LLM systems.

3 Internal Knowledge: Reactive

3.1 In-Context Learning

In-Context Learning (ICL) is one of the most distinctive capabilities of large language models (LLMs), providing a way to dynamically use and integrate knowledge. Rather than updating model parameters, ICL allows models to adapt to new tasks at inference time by conditioning on examples, instructions, or reasoning traces embedded directly in the input [11]. This approach leverages the knowledge already encoded in pretrained parameters and enriches it with knowledge presented in the immediate context, effectively blending static and contextual information in a single reasoning process. ICL highlights how LLMs can exploit knowledge in flexible ways. Prompting strategies, few-shot demonstrations, and structured reasoning traces such as chain-of-thought [47] exemplars serve as vehicles for injecting relevant knowledge into the model's decision process. These techniques show how users can curate and deliver domain-specific or task-specific knowledge on the fly, without retraining.

This paradigm offers several advantages from a knowledge perspective. By using natural language instructions (zero-shot) or a handful of examples (few-shot), ICL allows LLMs to quickly generalize to new domains and make use of unfamiliar knowledge. Its adaptability makes it possible to integrate knowledge that was

absent or only partially represented during pretraining, enabling rapid deployment across diverse application areas.

However, ICL has important limitations in managing and grounding knowledge. The finite context window restricts how much knowledge can be provided at once, which limits performance on knowledge-intensive tasks. Furthermore, because ICL does not explicitly verify or ground its outputs, it often generates hallucinations—fabricated information presented as fact. The absence of traceability means that the knowledge behind a prediction is opaque, reducing transparency and reliability. These issues highlight the need to extend ICL with external knowledge mechanisms, such as retrieval-augmented generation, to improve factuality and verifiability.

3.2 Instruction Tuning

Instruction Tuning is another central paradigm for adapting large language models (LLMs) to follow user-specified tasks more faithfully. Unlike In-Context Learning (ICL), which operates entirely at inference time, instruction tuning fine-tunes pretrained LLMs on a collection of curated datasets consisting of input-output pairs framed as natural language instructions [54]. Through this process, the model internalizes the mapping between instructions and desired behaviors, improving its ability to generalize to unseen tasks that share similar formats or intent. In effect, instruction tuning modifies the parameters of the LLM so that following instructions becomes an intrinsic behavior rather than an emergent property of prompting. Well-known examples include T5 [38], FLAN [46], and InstructGPT [37], which demonstrate that instruction tuning significantly enhances a model's usability by making it more responsive to natural instructions.

The benefits of instruction tuning are substantial. It improves robustness across domains, reduces the reliance on carefully engineered prompts, and provides a systematic approach to aligning LLM behavior with human preferences. Moreover, instruction-tuned models are more capable of handling task variation with minimal additional examples, narrowing the gap between artificial and human-like adaptability.

3.3 Knowledge Editing

Knowledge editing focuses on updating or modifying specific pieces of factual knowledge stored within large language models without retraining them from scratch. Unlike instruction tuning, which globally reshapes model behavior across tasks, knowledge editing seeks localized interventions: changing how the model responds to queries involving particular facts, entities, or relations, while preserving its overall performance and previously acquired knowledge [7, 32]. This makes knowledge editing especially valuable for time-sensitive or domain-specific updates, such as correcting outdated biomedical facts or incorporating new geopolitical events.

Formally, given a pretrained model f_θ, an editing algorithm aims to produce updated parameters $f_{\theta'}$ such that for a target query x^*, the output $f_{\theta'}(x^*)$ reflects the revised knowledge y^*. At the same time, for non-target queries $x \notin \mathcal{X}^*$, the outputs should remain close to their original predictions, minimizing unintended side effects. Many methods instantiate this principle through optimization

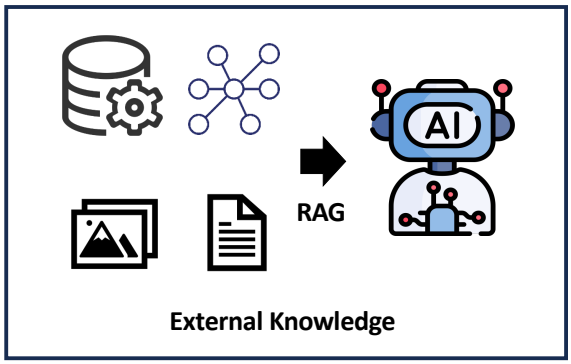


Figure 3: External Knowledge.

objectives of the form:

$$\min_{\Delta\theta} \mathcal{L}(f_{\theta+\Delta\theta}(x^*), y^*) + \lambda \mathbb{E}_{x \notin \mathcal{X}^*} [\mathrm{Dist}(f_{\theta+\Delta\theta}(x), f_\theta(x))] \quad (1)$$

where the first term enforces correctness of the edit and the second penalizes deviation from the model's prior knowledge.

In practice, knowledge editing ranges from parameter-based updates (e.g., ROME [32], MEMIT [33]) to interventions on hidden representations, balancing precision in updating target facts with generalization to diverse contexts. Nevertheless, edits can propagate undesired changes, fail to generalize beyond surface-level rephrasings, or struggle with multi-hop knowledge, motivating research into more robust, interpretable, and hybrid approaches that combine editing with retrieval or external knowledge for verifiable grounding.

3.4 LLM Knowledge Unlearning

Knowledge unlearning in LLMs aims to remove undesired or outdated knowledge while preserving unrelated information, which is critical for privacy, harm mitigation, and maintaining factual consistency in evolving KGs.

Model-based approaches typically modify parameters to erase specific knowledge, such as using Gradient Ascent (GA) [34] to invert gradients of undesired facts or variants that stabilize optimization via relabeled data. While effective at targeted removal, these methods can degrade retained knowledge and struggle to balance forgetting with preservation, especially given the interdependencies among entities and relations.

Evaluating unlearning remains challenging. Benchmarks like WHP [13], TOFU [30], and WMDP [20] measure fact removal using token-level or entity-level metrics, but most treat knowledge as independent and overlook relational structure. Multi-fact interactions have been explored, though current methods often rely on deterministic or rule-based evaluation, limiting scalability.

In KG–LLMs, unlearning is particularly delicate: removing one fact can inadvertently disrupt reasoning over related entities. Future work may focus on graph-aware algorithms, structural evaluation, and explainable methods to erase knowledge without compromising overall model reasoning and consistency.

4 External Knowledge: Lightly-active

While in-context learning (ICL) enables LLMs to exploit their internal parametric knowledge, it often suffers from limitations such

as hallucination, factual errors, and lack of verifiability. Retrieval-Augmented Generation (RAG) [12, 19, 40] has emerged as a complementary paradigm designed to address these issues. The central motivation behind RAG is to ground language model outputs in verifiable external sources, thereby improving factuality, reducing hallucinations, and providing transparency into the generation process. By incorporating retrieval mechanisms at inference time, RAG systems ensure that models are not solely dependent on internalized information, but can dynamically access up-to-date and domain-specific knowledge.

RAG systems are typically organized around several architectural patterns. The *classic pipeline* [40] follows a two-stage process: a retriever identifies relevant documents from an external corpus, which are then passed to the LLM for generation. More modular variants [12] separate retrieval and generation more explicitly, allowing fine-grained control over the knowledge integration process. Recent advances [31] also include *end-to-end retrieval-augmented training*, where both retrieval and generation components are optimized jointly, enabling the system to learn to retrieve the most useful context for the task at hand.

The effectiveness of RAG depends on the availability and quality of external knowledge sources. Commonly used sources include large-scale unstructured text corpora [8, 9, 14, 36, 44], structured repositories such as knowledge graphs [21–25, 28], and multimodal databases that integrate text, images, or other data modalities. Text corpora provide breadth and coverage, knowledge graphs offer structured and interpretable representations, and multimodal databases extend LLMs' reasoning beyond text alone. The choice of source often depends on the application, with hybrid approaches combining multiple knowledge types to improve robustness.

Research in RAG has advanced in several directions. Adaptive retrieval methods dynamically select the most relevant content based on context, reducing noise and improving precision. Multi-hop retrieval [43] enables the chaining of retrieval steps, supporting more complex reasoning across multiple documents. Another trend involves mixture-of-expert retrievers [16], where different retrieval modules specialize in distinct domains or modalities, and the system learns to route queries adaptively. Together, these advances push RAG beyond simple document lookup toward more sophisticated, context-aware grounding strategies.

Despite its promise, RAG faces several challenges. Retrieval quality remains a critical bottleneck, as noisy or irrelevant documents can mislead the generator [6]. Latency is another issue, as retrieval introduces additional computation that may hinder real-time applications [17]. Finally, ensuring reliable grounding is non-trivial: models may ignore retrieved evidence or selectively use it in ways that do not guarantee factual correctness [18]. Addressing these challenges is essential to make RAG both scalable and trustworthy.

5 Agentic LLMs: Proactive

Reactive vs Proactive: Traditional paradigms for leveraging knowledge in large language models, such as in-context learning and retrieval-augmented generation, primarily focus on how models passively access and integrate information. While effective in many scenarios, these approaches treat the model largely as a reactive system: it generates outputs based on prompts or retrieved context without actively initiating exploration, planning, or intervention. Agentic LLMs, in contrast, adopt a *proactive* stance. Rather than waiting for explicit queries, these models can autonomously identify relevant knowledge, anticipate information gaps, and plan multi-step actions to achieve objectives. This proactive behavior enables LLMs to actively operationalize knowledge, bridging the gap between static information retrieval and dynamic problem-solving. By reasoning about potential outcomes and taking initiative, agentic LLMs can efficiently navigate complex tasks, integrate diverse sources of knowledge, and adapt to changing environments.

Furthermore, proactive agentic behavior allows LLMs to better utilize knowledge in both parametric and non-parametric forms. Internally stored knowledge in model parameters can be applied strategically for reasoning and planning, while external knowledge sources—such as databases, APIs, or knowledge graphs—can be selectively queried to fill information gaps or verify hypotheses. This combination enhances both the effectiveness and reliability of the model, allowing it to act as an autonomous knowledge processor rather than a passive text generator. By enabling models to reason, plan, act, and interact with external tools or environments, agentic LLMs extend the scope of what AI systems can achieve. They are capable of sequential decision-making, iterative refinement, and adaptive behavior, all of which are crucial for real-world applications that demand more than single-step responses. In this sense, agentic LLMs represent a natural evolution from ICL and RAG toward models that can operationalize knowledge in a goal-directed and context-aware manner.

5.1 Harnessing Internal Knowledge

Agentic LLMs rely heavily on internal knowledge stored in their parameters, memory mechanisms, and reasoning capabilities:
Memory and Profile: In LLM agents, memory refers to the ability to retain and utilize past interactions, contextual information, and long-term knowledge about users or tasks, while profile represents structured information that defines the agent's role, attributes, preferences, and operational competencies. Together, they enable the agent to maintain continuity across sessions, adapt to user-specific needs, and perform domain-specific functions more effectively. Since both memory and profile are stored, organized, and accessed internally by the LLM (rather than retrieved externally), they are considered part of the LLM's internal knowledge, shaping how it reasons, plans, and interacts in dynamic environments.
Reasoning: In LLM agents, reasoning refers to the process of drawing logical inferences and making consistent conclusions based on available information, whether from the prompt, prior memory, or the model's internal knowledge. It allows the agent to connect facts, resolve ambiguities, and justify decisions beyond surface-level pattern matching. Since this ability emerges from the model's internal representations and learned structures—rather than depending on external retrieval—it is considered part of the LLM's internal knowledge.
Planning: In LLM agents, planning refers to the ability to generate and organize a sequence of coherent actions or reasoning steps that lead from the current state to a desired goal. Unlike simple response generation, planning enables the agent to anticipate future

requirements, break down complex tasks into manageable subtasks, and adapt strategies based on constraints or feedback. As this process relies on the model's internal reasoning capabilities—using its knowledge and learned patterns to decide how to act rather than retrieving instructions from outside—it is regarded as part of the LLM's internal knowledge and a key component of goal-directed autonomous behavior.

Frameworks like *ReAct* [53] interleave reasoning and planning, generating intermediate reasoning traces while leveraging internal knowledge for decision-making. *Reflexion* [41] adds feedback loops, enabling agents to evaluate and refine strategies based on past experiences.

5.2 Leverage External Knowledge

Agentic behavior in large language models can be substantially enhanced by incorporating external sources of knowledge. This integration can be broadly categorized into three complementary forms. (1) *Tool use*: Agents can invoke APIs, search engines, or specialized software interfaces to obtain up-to-date or task-specific information that is not encoded in their parametric memory. (2) *Knowledge graphs and databases*: Structured repositories provide factual grounding, enabling models to access verified relationships and reduce the risk of hallucination. (3) *Environment interaction*: Agents can act within simulated or real-world contexts to execute tasks, gather evidence, or refine their understanding through feedback.

Recent systems such as *AutoGPT* [51] demonstrate how these capabilities can be orchestrated within automated planning pipelines. By performing multi-step reasoning, tool invocation, and web-based information retrieval, such models achieve complex objectives. More generally, grounding agentic actions through retrieval-augmented generation (RAG) ensures that LLM behavior remains both factually accurate and temporally relevant while retaining open-ended reasoning abilities.

5.3 Multi-agent Collaboration

Agentic LLMs can also operate in multi-agent settings, where multiple models coordinate, negotiate, and share knowledge to accomplish complex objectives. Let $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$ denote a set of agents, each with internal state $\mathbf{s}_t^i$ and access to a subset of knowledge $\mathcal{K}_i$. Multi-agent frameworks enable several key capabilities.

(1) *Task decomposition*: Complex problems can be divided into sub-tasks and allocated to specialized agents, allowing each agent to focus on its strengths. This division of labor enhances efficiency and makes large-scale objectives more tractable.

(2) *Knowledge sharing*: Agents exchange intermediate results, reasoning traces, or learned insights to construct a richer shared context. Such communication reduces redundancy, supports cross-validation of outputs, and improves the accuracy and robustness of the collective system.

(3) *Coordination and negotiation*: Agents dynamically adjust strategies, resolve conflicts, and balance trade-offs to align with shared objectives. Through these mechanisms, multi-agent systems can adapt to evolving environments and optimize collective decision-making.

Multi-agent collaboration is particularly beneficial for scenarios that demand parallel exploration, multi-perspective reasoning, or the integration of heterogeneous expertise, allowing agentic LLMs to tackle tasks that would be challenging for a single model acting in isolation.

5.4 Routing

Efficiently routing tasks and queries to the appropriate agent or knowledge source is critical for scalability, reliability, and overall system performance. Routing strategies can be understood through several complementary mechanisms.

(1) *Dynamic routing* [35] assigns tasks to agents based on their capabilities, knowledge coverage, or historical performance. By adapting task allocation on the fly, the system ensures that each query is handled by the most suitable agent at a given time.

(2) *Hierarchical routing* [10] employs multi-level controllers to delegate subtasks to specialized agents or modules. This hierarchical structure allows complex tasks to be decomposed and processed efficiently across multiple layers of expertise.

(3) *Load balancing and redundancy* [42] ensures robustness by distributing critical tasks among multiple agents, preventing bottlenecks and providing fault tolerance. Such strategies help maintain consistent performance even under partial system failures.

Formally, routing can be expressed as a function $R : \mathcal{T} \times \mathcal{A} \rightarrow \mathcal{A}'$, where $\mathcal{T}$ denotes the space of tasks, $\mathcal{A}$ represents the set of available agents, and $\mathcal{A}' \subseteq \mathcal{A}$ corresponds to the selected subset of agents responsible for a given task. The goal of routing is to maximize overall system utility—balancing factors such as task success rate, latency, energy consumption, and redundancy—while ensuring coherent knowledge flow among agents.

Beyond these basic mechanisms, routing also interacts closely with memory management and reasoning control. For instance, adaptive routing policies can use feedback from past performance to update task-agent mappings dynamically, thereby enabling meta-learning of optimal routing strategies. Similarly, probabilistic or reinforcement learning–based routers can learn to predict which agent or external resource will yield the most reliable outcome for a given task. By combining internal knowledge, external grounding, multi-agent collaboration, and effective task routing, agentic LLMs move toward truly autonomous and proactive AI systems capable of handling complex, real-world challenges with efficiency and reliability.

6 Applications and Implications

Agentic large language models have the potential to transform a wide range of domains by leveraging their capacity for autonomous reasoning, planning, and interaction. In the context of *scientific discovery*, these models can autonomously explore literature, generate hypotheses, and propose experiments, thereby accelerating research cycles and uncovering patterns that might be overlooked by human researchers. By integrating multi-modal data sources, including text, images, and structured datasets, agentic LLMs support more comprehensive and cross-disciplinary scientific reasoning.

In *decision support*, agentic LLMs facilitate multi-step planning and tool integration to assist with tasks such as travel booking, healthcare recommendations, or business strategy analysis. By combining internal reasoning with external knowledge retrieval, these

systems can provide transparent and explainable recommendations, which is particularly valuable in complex, high-stakes scenarios.

Interactive assistants constitute another important application, where agentic LLMs manage dialogues, query databases, and integrate APIs to deliver context-aware and personalized responses. Such systems can adapt to user preferences over time, enabling tailored experiences in domains ranging from education and legal consultation to customer support.

Finally, agentic LLMs have significant implications for *autonomous systems*. When integrated with sensors and real-time feedback, they can support autonomous decision-making in robotics, logistics, and infrastructure management, offering scalable solutions in dynamic environments.

Despite these promising applications, agentic LLMs raise a number of critical challenges. Ensuring alignment with human values, maintaining safety in high-stakes domains, optimizing computational efficiency, and achieving robustness against adversarial inputs are all essential considerations. These challenges underscore the importance of developing rigorous assurance frameworks to guarantee trustworthy agentic AI. Moreover, ethical considerations, regulatory compliance, and sustainability must be addressed as these systems are deployed at scale. Balancing innovation with accountability will be crucial to maximize the societal benefits of agentic LLMs while mitigating potential risks.

7 Comparative Analysis

The three paradigms of knowledge use in LLMs—internal, external, and agentic—offer complementary strengths and weaknesses. Internal knowledge is advantageous when rapid adaptability is needed, especially in domains where the model's parametric memory suffices. However, it suffers from lack of grounding and verifiability. External knowledge, by contrast, provides transparency and factuality, but its effectiveness is constrained by retrieval quality and latency. Agentic approaches expand the horizon of what LLMs can achieve, enabling complex planning and tool use, yet introduce new challenges of alignment, efficiency, and safety.

Synergies between these paradigms are increasingly important in real-world systems. For instance, agentic LLMs often rely on ICL for reasoning and RAG for grounding, combining the flexibility of internal memory with the factuality of external retrieval. Hybrid systems that dynamically balance between parametric and non-parametric knowledge sources are particularly promising for applications that demand both creativity and reliability.

The trade-offs between these approaches can be characterized along several dimensions. Cost is a major consideration, as retrieval and agentic actions introduce overhead relative to pure ICL. Reliability varies depending on the availability of external resources and the ability to ground outputs in evidence. Scalability is affected by both context length in ICL and retrieval efficiency in RAG. Finally, explainability is often higher in RAG and agentic systems, where outputs can be traced to sources or intermediate steps.

8 Open Challenges and Future Directions

Artificial intelligence [48–50] and machine learning [2, 29] have greatly transformed our society. Despite the remarkable advances in large language models, several open challenges remain in applying these systems to real-world applications. Addressing these challenges is essential for building AI systems that are scalable, reliable, and trustworthy.

(1) *Scaling context versus retrieval* presents a fundamental design consideration. Expanding context windows can enhance in-context learning capabilities by allowing models to directly condition on more examples, yet this approach is memory-intensive and may remain unstructured. In contrast, retrieval-based mechanisms provide a more scalable means of incorporating external knowledge, but they introduce challenges related to retrieval quality, latency, and seamless integration with generative processes.

(2) *Hybrid symbolic-neural reasoning* represents a key frontier for combining the flexibility of deep learning with the precision and interpretability of formal logic. Developing architectures that support structured reasoning, constraint satisfaction, and explainability, while retaining the adaptability of neural methods, remains an open problem that is critical for trustworthy AI.

(3) *Dynamic and persistent memory* is another significant challenge. Current systems primarily rely on ephemeral context windows or static databases, limiting their ability to accumulate experiences or update knowledge over time. Building agents with long-term, verifiable memory raises important questions regarding consistency, version control, and trust, yet it is essential for agents that learn and adapt continuously in real-world environments.

(4) *Evaluation beyond accuracy* is increasingly necessary as models become more autonomous. Existing benchmarks often emphasize surface-level correctness, failing to capture critical dimensions such as factual grounding, reasoning depth, adaptability, efficiency, safety, and trustworthiness. Richer evaluation frameworks and stress tests are needed to assess performance in open-world, high-stakes, and multi-step reasoning scenarios.

Ultimately, progress toward knowledge-centric and trustworthy AI requires unifying these challenges. Future research must not only scale model capacity but also design systems that integrate memory, retrieval, reasoning, and action in ways that are reliable, interpretable, and aligned with human goals, enabling LLMs to operate effectively in complex, dynamic, and real-world environments.

9 Conclusion

This work has reviewed the foundations and frontiers of knowledge use in large language models, focusing on three complementary paradigms: *internal knowledge* through in-context learning, *external knowledge* through retrieval-augmented generation, and *agentic knowledge use* through LLM-based agents. Each paradigm brings distinct strengths—adaptability, factual grounding, and operational autonomy—while also facing unique limitations. By framing these approaches under a unified knowledge-centric perspective, we highlight their synergies as well as their trade-offs. Looking forward, we envision LLMs not merely as passive text generators, but as *knowledge processors* that internalize, ground, and operationalize information. Bridging these paradigms will be key to developing the next generation of trustworthy, explainable, and effective AI systems.

10 Acknowledgment

This material is based upon work supported by NSF awards (SaTC-2241068, IIS-2506643, and POSE-2346158), and a Cisco Research Award.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Yikun Ban, Yunzhe Qi, Tianxin Wei, Lihui Liu, and Jingrui He. Meta clustering of neural bandits. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 95–106, 2024.

[3] Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Chemcrow: Augmenting large-language models with chemistry tools, 2023.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[5] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022.

[6] Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR 2024, page 719–729. ACM, July 2024.

[7] Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[8] DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024.

[9] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[10] Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing, 2024.

[11] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. A survey on in-context learning, 2024.

[12] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization, 2025.

[13] Ronen Eldan and Mark Russinovich. Who's harry potter? approximate unlearning in llms, 2023.

[14] Aaron Grattafiori .et.al. The llama 3 herd of models, 2024.

[15] Shima Imani, Liang Du, and Harsh Shrivastava. Mathprompter: Mathematical reasoning using large language models, 2023.

[16] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024.

[17] Chao Jin, Zili Zhang, Xuanlin Jiang, Fangyue Liu, Shufan Liu, Xuanzhe Liu, and Xin Jin. Ragcache: Efficient knowledge caching for retrieval-augmented generation. *ACM Trans. Comput. Syst.*, September 2025. Just Accepted.

[18] Krishnaram Kenthapadi, Mehrnoosh Sameki, and Ankur Taly. Grounding and evaluation for large language models: Practical challenges and lessons learned (survey). In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 6523–6533. ACM, August 2024.

[19] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[20] Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D. Li, Ann-Kathrin Dombrowski, Shashwat Goel, Long Phan, Gabriel Mukobi, Nathan Helm-Burger, Rassin Lababidi, Lennart Justen, Andrew B. Liu, Michael Chen, Isabelle Barrass, Oliver Zhang, Xiaoyuan Zhu, Rishub Tamirisa, Bhrugu Bharathi, Adam Khoja, Zhenqi Zhao, Ariel Herbert-Voss, Cort B. Breuer, Samuel Marks, Oam Patel, Andy Zou, Mantas Mazeika, Zifan Wang, Palash Oswal, Weiran Lin, Adam A. Hunt, Justin Tienken-Harder, Kevin Y. Shih, Kemper Talley, John Guan, Russell Kaplan, Ian Steneker, David Campbell, Brad Jokubaitis, Alex Levinson, Jean Wang, William Qian, Kallol Krishna Karmakar, Steven Basart, Stephen Fitz, Mindy Levine, Ponnurangam Kumaraguru, Uday Tupakula, Vijay Varadharajan, Ruoyu Wang, Yan Shoshitaishvili, Jimmy Ba, Kevin M. Esvelt, Alexandr Wang, and Dan Hendrycks. The wmdp benchmark: Measuring and reducing malicious use with unlearning, 2024.

[21] Lihui Liu. HyperKGR: Knowledge graph reasoning in hyperbolic space with graph neural network encoding symbolic path. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, Suzhou, China, November 2025. Association for Computational Linguistics.

[22] Lihui Liu. Monte carlo tree search for graph reasoning in large language model agents. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, CIKM '25, New York, NY, USA, 2025. Association for Computing Machinery.

[23] Lihui Liu, Yuzhong Chen, Mahashweta Das, Hao Yang, and Hanghang Tong. Knowledge graph question answering with ambiguous query. In *Proceedings of the ACM Web Conference 2023*, 2023.

[24] Lihui Liu, Boxin Du, Heng Ji, ChengXiang Zhai, and Hanghang Tong. Neural-answering logical queries on knowledge graphs. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 1087–1097, 2021.

[25] Lihui Liu, Boxin Du, Jiejun Xu, Yinglong Xia, and Hanghang Tong. Joint knowledge graph completion and question answering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 1098–1108, New York, NY, USA, 2022. Association for Computing Machinery.

[26] Lihui Liu, Blaine Hill, Boxin Du, Fei Wang, and Hanghang Tong. Conversational question answering with language models generated reformulations over knowledge graph. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 839–850, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[27] Lihui Liu, Zihao Wang, Ruizhong Qiu, Yikun Ban, Eunice Chan, Yangqiu Song, Jingrui He, and Hanghang Tong. Logic query of thoughts: Guiding large language models to answer complex logic queries with knowledge graphs. *arXiv preprint arXiv:2404.04264*, 2024.

[28] Lihui Liu, Zihao Wang, and Hanghang Tong. Neural-symbolic reasoning over knowledge graphs: A survey from a query perspective. *SIGKDD Explor. Newsl.*, 27(1):124–136, July 2025.

[29] Lihui Liu, Ruining Zhao, Boxin Du, Yi Ren Fung, Heng Ji, Jiejun Xu, and Hanghang Tong. Knowledge graph comparative reasoning for fact checking: Problem definition and algorithms. *IEEE Data Eng. Bull.*, 45(4):19–38, 2022.

[30] Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C. Lipton, and J. Zico Kolter. Tofu: A task of fictitious unlearning for llms, 2024.

[31] Kohei Makino, Makoto Miwa, and Yutaka Sasaki. End-to-end trainable retrieval-augmented generation for relation extraction, 2024.

[32] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2023.

[33] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer, 2023.

[34] Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning, 2020.

[35] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data, 2025.

[36] OpenAI. Gpt-4 technical report, 2024.

[37] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[38] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[39] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[40] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*, 2023.

[41] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.
[42] Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets, 2023.
[43] Yixuan Tang and Yi Yang. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries, 2024.
[44] Gemini Team. Gemini: A family of highly capable multimodal models, 2025.
[45] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
[46] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners, 2022.
[47] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
[48] Yuchen Yan, Yongyi Hu, Qinghai Zhou, Lihui Liu, Zhichen Zeng, Yuzhong Chen, Menghai Pan, Huiyuan Chen, Mahashweta Das, and Hanghang Tong. Pacer: Network embedding from positional to structural. In *Proceedings of the ACM Web Conference 2024*, pages 2485–2496, 2024.
[49] Yuchen Yan, Baoyu Jing, Lihui Liu, Ruijie Wang, Jinning Li, Tarek Abdelzaher, and Hanghang Tong. Reconciling competing sampling strategies of network embedding. *Advances in Neural Information Processing Systems*, 36:6844–6861, 2023.
[50] Yuchen Yan, Lihui Liu, Yikun Ban, Baoyu Jing, and Hanghang Tong. Dynamic knowledge graph alignment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4564–4572, 2021.
[51] Hui Yang, Sifu Yue, and Yunzhong He. Auto-gpt for online decision making: Benchmarks and additional opinions, 2023.
[52] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023.
[53] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.
[54] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. Instruction tuning for large language models: A survey, 2025.

Overcoming Pitfalls in Graph Contrastive Learning Evaluation: Toward Comprehensive Benchmarks

Qian Ma[1], Hongliang Chi[1], Hengrui Zhang[2], Kay Liu[2], Zhiwei Zhang[3], Lu Cheng[2], Suhang Wang[3], Philip S. Yu[2], Yao Ma[1]

[1]Rensselaer Polytechnic Institute, [2]University of Illinois Chicago, [3]Pennsylvania State University

maq5@rpi.edu, chih3@rpi.edu, hzhan55@uic.edu, zliu234@uic.edu, zbz5349@psu.edu, lucheng@uic.edu, szw494@psu.edu, psyu@uic.edu, may13@rpi.edu

ABSTRACT

The rise of self-supervised learning(SSL), which operates without the need for labeled data, has garnered significant interest within the graph learning community. This enthusiasm has led to the development of numerous self-supervised learning methods, such as Graph Contrastive Learning (GCL) techniques, all aiming to create a versatile graph encoder that leverages the wealth of unlabeled data for various downstream tasks. However, the current evaluation standards for GCL approaches are flawed due to the need for extensive hyper-parameter tuning during pre-training and the reliance on a single downstream task for assessment. These flaws can skew the evaluation away from the intended goals, potentially leading to misleading conclusions. In our paper, we thoroughly examine these shortcomings and offer fresh perspectives on how GCL methods are affected by hyper-parameter choices and the choice of downstream tasks for their evaluation. Additionally, we introduce an enhanced evaluation framework designed to more accurately gauge the effectiveness, consistency, and overall capability of GCL methods. Our code implementation is available to ease reproducibility on https://github.com/GraphTL-Bench/BGPM.

1. INTRODUCTION

Graph Neural Networks (GNNs) [17; 8; 27] have emerged as a powerful tool for learning representations from graph-structured data, demonstrating remarkable success across various fields including social network analysis [14; 5; 9], molecular biology [4; 34; 13], recommendation systems [23; 33; 20], and traffic prediction [29; 21; 16]. By leveraging the rich relational information inherent in graphs, GNNs can capture complex patterns that traditional neural network architectures struggle to process. However, the efficacy of GNNs is largely contingent upon the availability of task-dependent labels to learn meaningful representations [10; 31; 37; 18]. Unlike labeling in more common modalities such as images, videos, texts, and audio, annotating graphs presents significant challenges due to the complex and often domain-specific nature of graph data [18; 11; 3]. This has led to a growing interest in self-supervised learning methods as a means to circumvent the limitations imposed by the need for extensive labeled datasets. Among the various categories of self-supervised learning, graph contrastive learning (GCL) has emerged as a particularly promising approach [28; 43; 44; 36; 26; 22; 39].

The primary goal of GCL is to pre-train an encoder capable of generating high-quality graph representations without relying on label information, with the hope that these representations can be effectively utilized across a wide array of downstream tasks [12; 2; 7; 18]. However, the current evaluation protocols for GCL methods exhibit significant shortcomings that fail to align with these fundamental goals as detailed below:

- GCL methods typically consist of multiple hyper-parameters due to their unique designs in augmentation methods or contrastive objectives. In existing evaluation protocols, these hyper-parameters are often tuned for each graph dataset, which plausibly involves the use of a validation set from a downstream task. However, in practice, this process contradicts the premise of pre-training without task-specific labels. This would be particularly problematic if the GCL methods are highly sensitive to hyper-parameter configurations.
- In existing evaluation procedures, the evaluation of GCL methods is predominantly focused on a single downstream task, usually node classification, conducted on the same dataset used for encoder pre-training. Such a limited evaluation framework may not accurately reflect the encoder's versatility across diverse tasks, potentially leading to misleading comparisons among different GCL methods.

In our study, we conduct a thorough empirical analysis focusing on two crucial elements: (a) how GCL methods' performance is affected by hyper-parameter adjustments in the pre-training phase, and (b) the extent to which a single downstream task can accurately reflect the overall efficacy of GCL methods. Our investigation, detailed in Section 4, confirms that the current evaluation systems are indeed compromised by these concerns. Particularly, we observe that some GCL methods are highly sensitive to the settings of hyper-parameters. Although these methods can perform well when hyper-parameters are optimally tuned, their effectiveness can substantially decrease with less-than-ideal settings, making them less practical for pre-training situations where adapting hyper-parameters for each specific downstream task is impractical. Despite this, such methods may still show strong results within the current evaluation models, which do not fully account for the challenges posed by their sensitivity to hyper-parameters [39; 43]. Additionally, evaluating these methods based on a single downstream

task often does not provide a complete picture of their capabilities, as the performance of GCL methods can vary significantly across different tasks. This inconsistency highlights the limitations of current evaluation approaches that focus on singular tasks, thus failing to offer a comprehensive assessment of GCL methods' overall performance.

Hence, to address the identified issues, we propose a new evaluation protocol aimed at a more comprehensive and accurate assessment of GCL methods. This improved protocol incorporates a comprehensive analysis across diverse hyper-parameter configurations and extends the evaluation to include multi-label dataset evaluation. Through these enhancements, our protocol seeks to mitigate the impact of hyper-parameter sensitivity and broaden the scope of evaluation beyond a single task, thereby offering a more comprehensive understanding of GCL method performance.

2. PRELIMINARIES

Graph Contrastive Learning (GCL) focuses on learning superior node representations by distinguishing between node pairs that share similar semantics and those that do not. An anchor node, which can be any selected node in augmented graphs, is paired with semantically similar nodes or graphs (positive examples) to create positive pairs and with dissimilar nodes or graphs (negative examples) to establish negative pairs. GCL models aim to map graph elements such as nodes or graphs into an embedding space where positive pairs are pulled closely together while negative pairs are pushed far apart. To accomplish this, GCL methods are designed with a variety of components. As well-summarized in a prior work [41], GCL methods are normally designed differently in the following three key components:

- **Data Augmentation:** Data augmentation in GCL aims to create variations of a graph that preserve its major semantic information, thereby helping models to create positive and negative examples. This involves two main approaches: topology transformations and feature transformations. Topology augmentations adjust the graph structure through methods like edge dropping etc. Feature augmentations alter node features by masking to generate diverse representations. The common hypeter-parameters derived from these components are *drop edge rate* and *drop feature rate*.

- **Contrastive Mode:** In GCL, contrasting modes define how the similar and dissimilar samples of an anchor node are selected for comparison. There are three primary modes: *local-local* contrasts nodes at the same level, *global-global* contrasts entire graph embeddings, and *global-local* contrasts nodes with their graph-level representation. The choice of mode depends on the task, with node-focused tasks typically using local-local and global-local modes.

- **Contrastive Objective:** Contrastive objectives quantify the similarity between positive pairs and the difference from negative pairs. Common objectives include Information Noise Contrastive Estimation (InfoNCE) and Jensen-Shannon Divergence (JSD), which require explicit negative sampling. Other objectives like Bootstrapping Latent Loss (BL) and Barlow Twins (BT) do not require negatives, focusing instead on enhancing positive pair similarity and feature diversity. The temperature hyper-parameter τ in InfoNCE is typically adjustable in GCL methods utilizing it as the contrastive objective.

Investigated Methods: Building on the general paradigm of Graph Contrastive Learning (GCL), we summarize the key hyper-parameters influencing graph representation across various methods investigated in our work with a brief introduction as follows.

- **1. DGI:** As a pioneer work of GCL, DGI [28] maximizes mutual information between global graph embeddings and local node embeddings, leveraging JSD as its contrastive loss. Under its design, it has two hyper-parameters to vary, *hidden dimension* and *layer number* of the backbone GNN used to extract representations.

- **2. MVGRL:** [11] leverages multiple graph views to capture comprehensive structural patterns, applying different GNN configurations for each view to enrich node representations. MVGRL is with the same set of adjustable hyper-parameters as the DGI, but two backbone GNNs are applied on different augmented views(*i.e.,* there are two sets of *hidden dimension* and *layer number*).

- **3. GRACE:** [43] focuses on local node-level embedding contrast between two randomly augmented graph views. Each view is copied from the input graph first, next undergoing modifications where a certain percentage of edges and node features are removed and masked, determined by the specified *drop edge rate* and *drop feature rate*, respectively. Different from JSD loss adopted DGI and MVGRL, the adopted contrastive loss for GRACE is InfoNCE controlled by an additional parameter τ.

- **4. BGRL:** [26] utilizes Bootstrapping Latent loss for graph learning without negative samples, with similar two augmented graph views adjustable with *drop edge rate* and *drop feature rate*.

- **5. Graph Barlow Twins:** [1] aims to make the cross-correlation matrix of embeddings from two views as close to the identity matrix as possible, using different hyper-parameters *drop edge rate* and *drop node rate* for graph augmentations on two views.

- **6. CCA-SSG:** [36] first generates shared node representations from two augmented graph views with similar hyper-parameters mentioned before, then using Canonical Correlation Analysis to maximize correlation between views and decorrelate feature dimensions within each view. CCA-SSG has an additional hyper-parameter *loss reweighting factor* for the trade-off between losses.

- **7. SUGRL:** [22] simplifies the contrastive learning process by omitting graph augmentation and similarity determination steps, focusing instead on increasing inter-class differences and decreasing intra-class differences with a triplet loss and an upper bound loss. Given this specific design, there are four unique hyper-parameters for SUGRL, *loss reweighting factor*, *iteration number* for its unique negative samples generation module, *margin parameter*, a hyper-parameter of the triplet loss that penalizes a small gap in distances between a positive pair and a negative pair, and a *bound parameter*, a non-negative tuning parameter used in the upper bound loss.

- **8. COSTA:** [38] is proposed to address biases in graph augmentation by employing feature augmentation, allowing for better control over data distribution in the latent space. COSTA has τ for its InfoNCE contrastive loss and *drop edge rate* and *drop node rate* for two different augmented views.

- **9. SFA:** [39] proposed spectral feature augmentation for GCL, which is designed to re-balance the feature spectrum by iteratively removing low-rank information from the feature matrix. Therefore, apart from *drop edge rate* and *drop node rate*, there are two more hyper-parameters for SFA: k as the number of iterations used in its spectral feature augmentation, and again τ in its adopted InfoNCE loss.

3. DATASETS AND EVALUATION METRICS

3.1 Datasets

In this study, we utilize a diverse collection of datasets to evaluate the performance of GCL models across different domains and tasks:

Cora (Cora), Citeseer (Cite), Pubmed (Pub): Provided by [32], these citation networks include articles as nodes with bag-of-words features, edges as citations, and node labels indicating article types. **Amazon Computers (Am-Com) and Amazon Photo (Am-Ph):** From [24], these are part of the Amazon co-purchase graph, where nodes represent goods, edges indicate co-purchases, features are from bag-of-words reviews, and labels denote product categories. **Coauthor CS (Co-CS) and Coauthor Physics (Co-Phy):** From the Microsoft Academic Graph, these networks depict authors as nodes and co-authorship as edges. Features are derived from keywords in publications, with labels reflecting research areas. **Protein-Protein Interaction (PPI):** As cited in [45], this dataset includes networks for human tissues, nodes with multiple gene ontology labels, and features such as positional gene sets and immunological signatures. **PCG, HumLoc, and EukLoc:** Curated by [40] for multi-label classification, focusing on protein phenotype prediction (PCG) and subcellular localization in humans (HumLoc) and eukaryotes (EukLoc). The detailed statistics of datasets can be found in Table 1.

Table 1: Datasets statistics. Kindly note that PCG, HumLoc, EukLoc and PPI are multi-label node classification dataset while the others are multi-class node classification datasets.

Dataset	#nodes	#edges	#features	#class
PCG	3k	37k	32	15
HumLoc	3.10k	18k	32	14
EukLoc	7.70K	13K	32	22
PPI	56,944	818,716	50	121
Cora	2,708	10,556	1,433	7
CiteSeer	3,327	9,104	3,703	6
PubMed	19,717	88,648	500	3
CS	18,333	163,788	6,805	15
Physics	34,493	495,924	8,415	5
Computers	13,752	491,722	767	10
Photo	7,650	238,162	745	8

3.2 Evaluation Metrics

Accuracy is the most commonly used metric for evaluating multi-node classification tasks. It measures the proportion of correctly predicted instances over the total instances. Specifically, accuracy is calculated as the ratio of the sum of true positives and true negatives to the total number of cases, making it a straightforward and intuitive measure of model performance.

F1 score, which is essential for assessing model performance in multi-label node classification tasks, harmonizes precision and recall into a single metric. Below is a breakdown for Macro and Micro F1 Score: *Macro F1 Score* Computes F1 scores separately for each label and then averages them. Ensures equal importance is given to each label, which is especially useful for datasets with imbalanced label distributions. *Micro F1 Score* Aggregates true positives, false positives, and false negatives across all labels to compute a collective F1 score. Focuses on overall performance, weighting more heavily towards labels that appear more frequently.

4. ISSUES OF EXISTING EVALUATION PROTOCOL

The primary objective of graph contrastive learning is to pre-train an encoder capable of producing high-quality graph representations without label information, with the anticipation that these representations can be efficiently applied to a diverse range of downstream tasks. However, the existing evaluation protocol for graph contrastive learning methods deviates from the aforementioned goals, thus inadequately assessing their efficacy. Specifically, existing evaluation protocols have the following deficiencies:

- As elucidated in Section 2, GCL methods typically involve numerous hyper-parameters during the pre-training stage. In the current evaluation protocol, these hyper-parameters are typically optimized for each graph dataset [43; 39]. The hyper-parameter selection is plausibly carried out with a validation set of a downstream task—an approach that is ideally not assumed during the pre-training phase of the encoder. This is essentially problematic especially if the encoders are sensitive to the hyper-parameters.

- The existing evaluation protocol predominantly concentrates on a single downstream task, i.e., node classification on the same dataset the encoder is trained. Such a constrained assessment framework may inadequately capture the encoder's performance across the entire task space, rendering comparisons between different GCL methods potentially misleading. In particular, if the downstream task does not effectively represent the diverse range of possible applications, the basis for comparing GCL methods becomes flawed, as it fails to reflect the true generality and adaptability of the encoder's learned representations.

In this section, we undertake empirical experiments to substantiate the aforementioned concerns. Our experiments seek to investigate the following two research questions:

- What is the extent of sensitivity of GCL methods to hyper-parameters during pre-training stage?

- Is a single downstream task representative enough to comprehensively assess the performance of GCL methods?

4.1 Sensitivity to Hyper-Parameters

In this section, we aim to investigate the sensitivity of the GCL methods to the hyper-parameters in the pre-training stage. Specifically, we intend to commence our investigation by undertaking a hyper-parameter tuning process for each GCL method, utilizing the validation set of the downstream task by the conventional model selection procedure. This approach will not only facilitate gaining a deeper understanding of the capabilities inherent within GCL methods but also establish baselines for assessing their sensitivity to hyper-parameters—we will compare their performance under optimally tuned hyper-parameters against the outcomes derived from suboptimal hyper-parameters.

4.1.1 Settings

In this part, we delineate the general experimental settings for our investigation. Following existing works [42; 39; 22; 44], the pre-training is conducted on the same dataset used for downstream task evaluation without access to labels.

Search Space of Hyper-Parameters. As explicated in Section 2, various GCL methods typically involve distinct numbers of hyper-parameters, thereby yielding diverse hyper-parameter search spaces for these GCL methods. Therefore, given the computational expense associated with GCL methods, it is impractical and inequitable to tune all hyper-parameters exhaustively. Instead, we advocate to randomly sample 20 combinations of hyper-parameters from the search space of each method. Subsequently, the "optimal" set of hyper-parameters is selected from these sampled combinations for each method. Thus, the selected hyper-parameters in this work differ from the ones reported in existing works [43; 39]. These 20 combinations of hyper-parameters can be found along with our code implementation [1]. We also outline hyper-parameters search space within typical range in Appendix A. For all methods, the backbone model is GCN [15]. Specifically, throughout the pre-training phase of each encoder, we save encoder versions at predetermined epochs, namely at $50, 100, 500, 1,000$, and $10,000$ epochs. This systematic approach enables the evaluation of the encoder's performance at different stages of training without imposing additional computational burdens, requiring only minimal storage overhead.

Pipeline Our evaluation procedure adheres to the pipeline employed in previous works [28; 43; 44; 42]. Specifically, we fix the pre-trained encoder and tune a linear task head for the downstream node classification task *e.g.,* a linear multi-class classifier utilizing the node representations from the encoder as input. The performance of the downstream node classification task is considered as the encoder's performance *i.e.,* the performance of the GCL method.

4.1.2 Results analysis

We conduct experiments on 7 multi-class node classification datasets, which are introduced in Section 3. We present and analyze the results as follows.

The Capability of GCL Methods. We illustrate the capability of GCL methods with their corresponding "optimal" hyper-parameters selected from the pre-defined 20 combinations. The "optimal" performance of 9 representative methods across 7 datasets are shown in Figure 1.

[1] https://github.com/GraphTL-Bench/BGPM

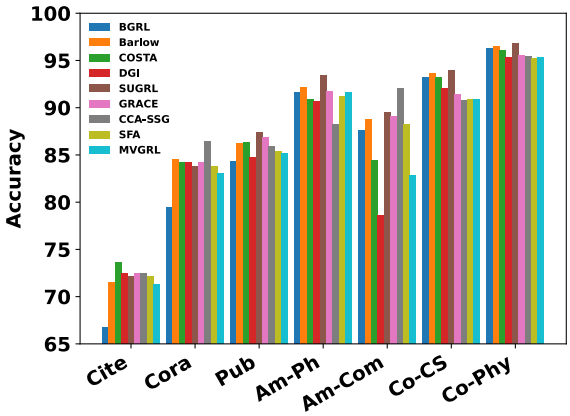


Figure 1: Performance with optimal hyper-parameters.

As shown in Figure 1, we observe that with the "optimal" hyper-parameters, all methods are capable of achieving reasonable performance that is close to the reported results in the original papers [36; 39; 38] across all datasets. However, the rankings between them occasionally diverge from the rankings reported in existing works. This is mainly due to the different hyper-parameters adopted, which to some extent underscore the sensitivity of the GCL methods to the hyper-parameters. Note that in this experiment, our objective is not to replicate the results from previous studies but rather to demonstrate that all GCL methods can achieve robust performance when appropriately tuned.

The Selected Hyper-parameters. In order to investigate how the GCL methods are sensitive to hyper-parameters, we first analyze the selected optimal set of hyper-parameters for different methods across various datasets. In particular, we

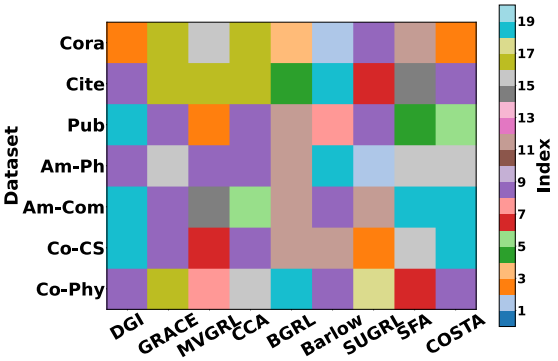


Figure 2: Index of the optimal hyperparameter set for different methods on different datasets. Indices range from 0 to 19.

document the indices of the selected sets of optimal hyper-parameters across all datasets for each method. We illustrate the hyper-parameter selection process for all datasets in Figure 2, where various indices are represented by different colors in the heatmap.

As depicted in Figure 2, for all methods, there is no dominating set of hyper-parameters valid for all datasets, i.e., different datasets prefer different hyper-parameters. These findings underscore the sensitivity of GCL methods to hyper-parameters, indicating the impracticality of employing identical hyper-parameter settings across all datasets, even for the same method. Such observations are consistent with existing works [39; 43], which have shown that GCL methods often require varied hyper-parameter configurations when applied to different datasets.

Variance Across Different Hyper-parameters. The preceding results also indicate that the methods are sensitive to hyper-parameters. In this part, we aim to further understand how significantly the GCL methods are affected by the hyper-parameters. In particular, for each method, we assess the downstream performance for the 20 pre-trained encoders corresponding to the 20 combinations of hyper-parameters.

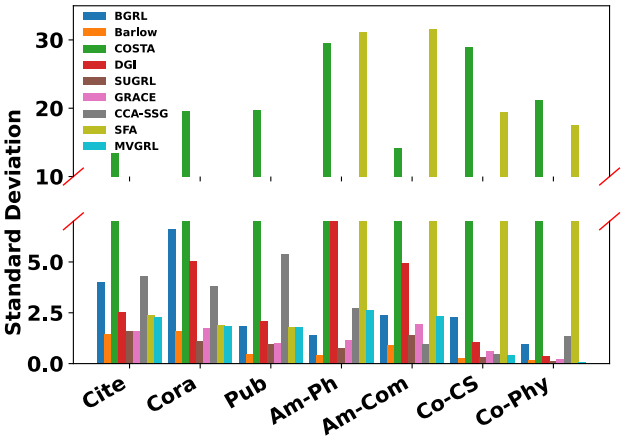


Figure 3: Standard deviation across 20 sets of hyper-parameters.

Subsequently, we compute the standard deviation of these downstream task performances for each method across all datasets. The results are presented in Figure 3, where the key observations emerge.

- The standard deviation is contingent upon the method, indicating varying degrees of sensitivity among different methods. Notably, COSTA and SFA demonstrate considerably higher instability compared to other methods, while Barlow and SUGRL consistently yield stable results.
- The standard deviation is influenced by the dataset utilized for the evaluation, suggesting varying levels of stability across datasets for the same method. For instance, SFA exhibits stability on datasets such as Citesser, Cora, and Pubmed but displays pronounced instability on other datasets. Conversely, Barlow exhibits greater stability on Coauthor-CS and Physics datasets, albeit with relatively lower stability on Citeseer and Cora datasets.

Summary. In conclusion, the effectiveness of most GCL methods is notably influenced by the selection of hyper-parameters. Additional evidence on the sensitivity and preference to hyper-parameter of graph SSL methods designed for graph-level tasks, which further demonstrates the critical role hyper-parameters play in determining model performance. When the hyper-parameters are properly optimized, most GCL methods are capable of delivering satisfactory performance for a given downstream task. However, the practical challenge arises from the fact that tuning GCL methods hyper-parameters to perfection is often not viable, primarily because downstream tasks are not predefined during the pre-training phase. Therefore, *when evaluating the efficacy of a GCL method, it is crucial to consider its sensitivity to hyper-parameters.* Specifically, GCL methods that exhibit minimal sensitivity to changes in hyper-parameters are generally more desirable. To achieve acceptable performance on downstream tasks, such methods necessitate a reduction in the workload associated with hyper-parameter selection during the pre-training stage.

4.1.3 *Further Verification on Graph Prompting*

Building on the 'pre-training and fine-tuning' paradigm of Graph Contrastive Learning (GCL), recent studies have explored an extended framework that integrates 'pre-training, prompting, and fine-tuning' [25; 6; 19], where prompting introduces a small set of learnable tokens or structures into the input graph to reformulate downstream tasks so they better align with the pre-training objective. However, as long as 'pre-training' remains a component, the challenges discussed earlier persist *i.e.,* the sensitivity of methods to hyper-parameters may impact evaluation outcomes.

To investigate this, we conducted experiments using the All-in-One framework [25], following the experimental settings outlined in Section 4 to assess hyper-parameter sensitivity[2]. In accordance with the original implementation, we evaluated two GNN backbones, GAT [27] and GCN [15],pre-trained with GraphCL [35] and SimGRACE [30].

Our primary observation from Figure 4 is that, when properly tuned, models with the same backbone but different pre-training methods achieve comparable performance. For example, on the Cora and CiteSeer datasets, GraphCL does not demonstrate a clear advantage over SimGRACE.

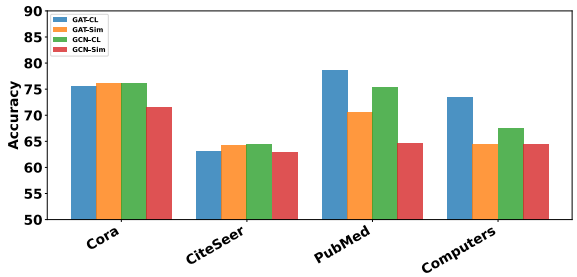


Figure 4: All-in-One performance with optimal hyper-parameters. The suffix "CL" denotes models pre-trained with GraphCL, while "Sim" represents models pre-trained with SimGRACE.

However, failing to account for hyper-parameter sensitivity may lead to an incomplete evaluation of benchmark performance. As shown in Figure 5, SimGRACE exhibits more stable performance on the Cora and CiteSeer datasets, as reflected by a lower standard deviation. If only the "op-

[2]https://github.com/VAN-QIAN/ProG/tree/downstream

timal" performance is considered while ignoring sensitivity, an important insight is overlooked: SimGRACE may be a more reliable choice than GraphCL on the CiteSeer dataset.

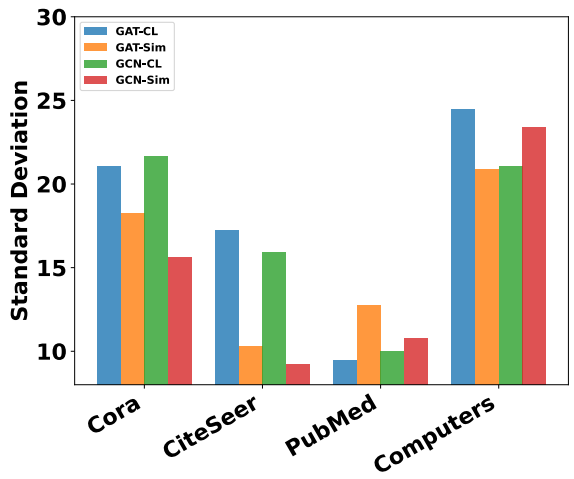


Figure 5: All-in-One standard deviation across 20 sets of hyper-parameters. The suffix "CL" denotes models pre-trained with GraphCL, while "Sim" represents models pre-trained with SimGRACE.

4.2 Representativeness of Downstream Tasks

In this section, we explore the adequacy of using a single downstream task to evaluate GCL methods. Our objective is to conduct experiments comparing the performance of various GCL methods across multiple downstream tasks and to assess whether the relative rankings of these methods remain consistent across different tasks. This investigation seeks to elucidate the representativeness of a single task as a benchmark for assessing the effectiveness of different GCL methods.

Pipeline. Following the aforementioned schedule, for the PPI dataset with 121 labels, we have 121 binary classification tasks. Subsequently, we utilize the evaluation setting as described in Section *4.1.1* to select the best set of hyper-parameters for each task. Thus, after the model selection, for each GCL method, we will have 121 selected "optimal" encoders corresponding to the 121 tasks. We then proceed to compare the GCL methods on each task with their corresponding "optimal" encoders specific to this task. Specifically, for each task, we rank the performance of the 9 methods. Therefore, we compile a total of 121 ranking lists, with each list comprising 9 GCL methods.

4.2.1 Results Analysis

We present key statistics extracted from the 121 ranking lists in Table 2. For each GCL method, the "Min" and "Max" denote its minimum and maximum rankings obtained among the 121 tasks, respectively. Additionally, the "Mean" and "Std_dev" indicate the mean rank and standard deviations of the rank among the 121 tasks, respectively.

The results in table 2 reveal that the rankings of all GCL methods vary across different downstream tasks. Each method can exhibit significant performance disparities, achieving high rankings on certain tasks while performing poorly on others, as evidenced by the "Min" and "Max" rankings. Moreover, the standard deviation of each method underscores the prevalence of this phenomenon. These findings indicate that no single downstream task is representative enough for the entire downstream task space. Relying solely on a single downstream task for evaluation may lead to an inaccurate assessment of the effectiveness.

Table 2: Optimal Performance Ranking Based on Accuracy

Model	Min	Max	Mean	Std_dev
Barlow	1	9	3.058	1.489
BGRL	1	9	3.462	2.691
CCA-SSG	2	9	5.935	1.245
COSTA	1	9	5.496	1.687
DGI	4	9	8.545	1.113
GRACE	1	8	3.025	1.474
MVGRL	1	8	5.033	2.147
SFA	2	9	7.074	1.337
SUGRL	1	9	3.371	2.378

Given the impracticality of hyper-parameter selection for each GCL method, we adopt an alternative experimental approach that involves averaging the performance across 20 sets of hyper-parameters. This methodology aims to mitigate the bias introduced by hyper-parameter sensitivity. Specifically, for each downstream task, we compute the average performance of a GCL method over these 20 sets of hyper-parameters. Subsequently, we apply this average performance metric to rank the GCL methods for each of the 121 tasks, resulting in 121 distinct ranking lists. These results are presented in Table 3. Similar observations as in Table 2 can be made from Table 3, further confirming that a single task is not sufficiently representative.

Table 3: Average Performance Ranking Based on Accuracy

Model	Min	Max	Mean	Std_dev
Barlow	1	9	4.876	2.039
BGRL	1	9	4.694	3.149
CCA-SSG	3	9	6.479	1.472
COSTA	1	9	3.843	2.045
DGI	4	9	5.157	2.077
GRACE	1	8	1.901	1.576
MVGRL	1	8	5.570	1.983
SFA	2	9	7.694	1.757
SUGRL	1	9	4.785	2.205

Summary. It is evident that relying on a single downstream task for comparing different GCL methods is not sufficient to obtain a comprehensive understanding of their performance. Therefore, *when evaluating the efficacy of GCL methods, it is critical to conduct comprehensive experiments on multiple downstream tasks.*

Discussion. To incorporate multiple downstream tasks, a straightforward idea is to include tasks of varying levels simultaneously, such as graph-level (graph classification), edge-level (link prediction), and node-level (node classification) tasks. However, handling all these tasks with a single pre-trained model is typically infeasible due to the limitations of both *dataset perspective* and *model architecture perspective.*

From the **dataset perspective**, node classification datasets like Cora and Coauthor typically consist of a single graph with labeled nodes, making it impractical to conduct graph classification tasks on such datasets. Additionally, performing edge-level tasks like link prediction using pre-trained models poses additional challenges, as it requires removing a portion of the edges and using them as a test set, necessitating the pre-training of another model on a modified dataset with the removed edges. It is critical to remove a portion of the edges (i.e., split the dataset) in a manner that ensures balanced data for training and testing. However, determining a reasonable way to split the data remains a significant challenge, particularly since none of the methods we reviewed incorporate edge-level tasks for evaluation.

From the **model structure perspective**, most GCL models designed for node classification are inherently unsuited for graph classification tasks, as they are primarily intended for learning node representations. The majority of the methods we reviewed lack a dedicated pooling module, making them nativly unsuitable for direct evaluation on graph-level tasks like graph classification.

Therefore, we propose simulating multiple downstream tasks by framing a multi-label classification task as a series of independent binary classification tasks, which we detail in the next section.

5. THE IMPROVED EVALUATION PROTOCOL

To address the aforementioned issues, we introduce an improved protocol for a more comprehensive evaluation. This protocol is designed to address the limitations of hyper-parameter sensitivity and the narrow focus on single downstream tasks. The improved evaluation framework consists of the following components.

- **Comprehensive Analysis Across Diverse Hyper-parameter Configurations.** We advocate for evaluating GCL methods based on their performance across a variety of hyper-parameter configurations. This approach includes analyzing both the average performance and the variability (or variance) of performances. By considering the variance, we gain insights into the stability of the GCL methods under different hyper-parameter settings, providing a more holistic view of their effectiveness.

- **Extension to Multi-label Dataset Evaluation.** Our protocol expands the scope of evaluation to include multi-label datasets, which allows us to simulate an environment with multiple downstream tasks. In particular, conducting a multi-label classification task can be regarded as solving a set of K binary classification tasks with K denoting the number of labels in the multi-label classification task. Evaluating GCL methods in this context offers a richer perspective on their capacity to generalize and adapt to diverse downstream tasks.

General Settings. Our evaluation encompasses both multi-class and multi-label datasets to ensure a comprehensive analysis of GCL methods. For multi-class, accuracy serves as the primary evaluation metric, offering a straightforward measure of a model's predictive performance. To address the inherent variability and sensitivity of GCL methods to hyper-parameter selection, we undertake an extensive analysis based on the average and variance of performance metrics across 20 randomly sampled sets of hyper-parameters. For multi-label, our evaluation framework employs both Micro-F1 and Macro-F1 scores as metrics as introduced in Section 3. Adhering to a similar methodology as applied in the multi-class dataset evaluation, we also derive insights from the average and variance of these two metrics across 20 randomly sampled sets of hyper-parameters for each GCL method.

5.1 Evaluation with Multi-class Classification

We conduct evaluation procedures for each GCL method on all 7 multi-class node classification datasets introduced in Section 3.1. With our proposed evaluation protocol, we aim to assess the general performance and stability of each method on different datasets. Due to space limit, we aggregate the results of all the datasets into Figure 6 by averaging them *i.e.,* the x and y axis indicates the average values of the method's accuracy and the standard deviation across 7 datasets, respectively. The performance here specifically denotes accuracy as detailed in Section 3.2. Here, a higher mean performance signifies superior effectiveness, whereas a lower mean standard deviation points to enhanced stability. Notably, in Figure 6, COSTA has been excluded due to its notable instability, which is consistent with previously reported findings [39; 38]. Barlow and SUGRL stand out by achieving the highest rankings in both average performance and stability metrics, showing their effectiveness and robustness. This finding aligns with the effectiveness reported in their respective papers. Surprisingly, DGI demonstrates a more competitive performance than might be anticipated, considering it as an earlier work often reported to underperform in other Graph Contrastive Learning (GCL) methods. Next, we delved deeper into the discrepancies between our results and those previously published in the literature [38; 39]. SFA [39] reported that SFA consistently surpassed GRACE and Barlow when utilizing a specific set of hyper-parameters, with SUGRL classified as a moderately performing method. In Section *4.1.2*, SFA is also observed with comparably good results similar to GRACE. Also, COSTA [38] was often highlighted as achieving near-

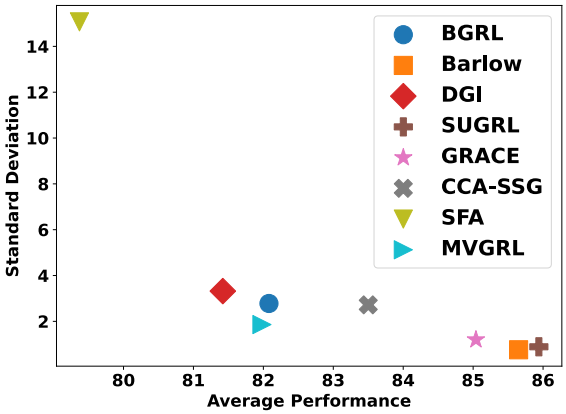
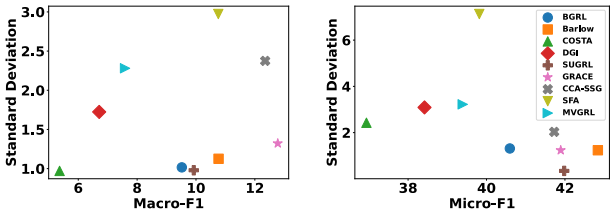


Figure 6: Results on multi-class classification datasets.

top performance, normally just behind SFA. This is also validated by the observation that COSTA performs relatively well in our results shown in Section *4.1.2*. However, when evaluating across our two aforementioned metrics, GRACE and Barlow not only outperform SFA and COSTA, showing superior effectiveness, but SUGRL also emerges as a top performer. Here, SFA's evaluation was based on dataset-specific hyper-parameters according to [39], while hyper-parameters for other baselines follow same ones in PyGCL [42]. Those new findings in our results demonstrate the advantage of our proposed evaluation protocol, which is designed to measure the overall performance and stability of GCL methods in a more realistic way. Overall, our new protocol enables a more comprehensive and robust understanding of a GCL method's capabilities.

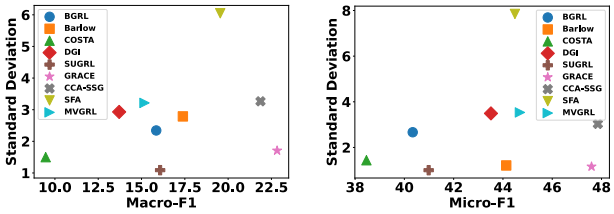
5.2 Evaluation with Multi-label Classification

Following our discussion on the outcomes of multi-class classification, we now shift our focus to the evaluation results obtained from multi-label classification tasks. We conduct the experiments on three datasets Eukloc, Humloc, and PCG, and their corresponding results are presented in Figures 7, 8, and 9, respectively.



(a) Macro F1 on Eukloc (b) Micro F1 on Eukloc

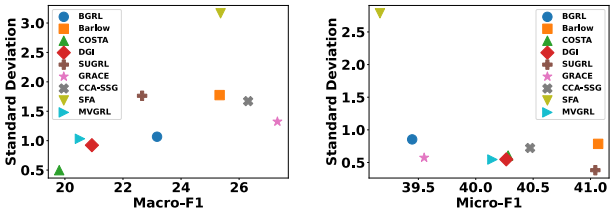
Figure 7: Results on Eukloc



(a) Macro F1 on Humloc (b) Micro F1 on Humloc

Figure 8: Results on Humloc

We make some general observations from these figures. Initially, we will discuss findings aligning with established benchmarks in multi-class classification, followed by an exploration of divergent results observed in multi-label classification. In multi-label classification, GRACE and Barlow maintain their strong performance and stability, underscoring their robust and consistent capabilities. DGI and BGRL also show a consistent pattern, displaying good stability and moderate performance in both classification contexts. Contrasting the two-fold consistencies mentioned in the prior models, the remaining models exhibit a similar pattern in



(a) Macro F1 on PCG (b) Micro F1 on PCG

Figure 9: Results on PCG

only one aspect: SUGRL demonstrates notable strength in performance across both evaluation settings, while SFA consistently keeps relatively large standard deviations across all evaluations. Also, CCA-SSG maintains a trend of moderately above-average performance.

Apart from consistent results, we also observe notable shifts in model performance or overall stability. SFA, previously underperforming in multi-class classification, shows improved results in the current multi-label classification tasks, ranking above average. Conversely, SUGRL, while dominating in multi-class classification, falls to moderate performance levels in the new tasks. In addition, MVGRL sees a slight increase in standard deviation, indicating a change in stability from lower to moderate levels in the multi-label classification. Moreover, CCA-SSG's stability decreases in multi-label classification, with some datasets like the Eukloc showing a notably higher standard deviation in the Macro F1 assessment.

The observed consistencies across evaluation methods suggest a good level of agreement, indicating that both evaluation methods can, to a certain extent, reflect the overall performance and stability of GCL methods. On the other hand, new observations in multi-label classification tasks highlight the value of incorporating this new multi-label classification evaluation approach, which broadens the evaluation scope and reveals deeper insights into the performance of GCL methods.

6. CONCLUSION

In this paper, we illustrate the intrinsic limitations of the existing evaluation protocol of GCL methods, highlighting its divergence from the overarching goals of self-supervised learning. We approach this assessment from two angles: the numerous hyper-parameters during the pre-training phase and the sole evaluation based on a single downstream task. We verify the prominent issues by investigating the sensitivities of GCL methods to hyper-parameters during the pre-training stage and the representativeness of the downstream task. Moreover, to rectify the aforementioned issues, we propose an improved evaluation protocol for better benchmarking to comprehensively evaluate the GCL methods.

7. ACKNOWLEDGEMENTS

The research is supported by the National Science Foundation (NSF) under grant numbers NSF2406647 and NSF-2406648.

8. REFERENCES

[1] Piotr Bielak, Tomasz Kajdanowicz, and Nitesh V Chawla. Graph barlow twins: A self-supervised representation learning framework for graphs. *Knowledge-Based Systems*, 256:109631, 2022.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[3] Quanyu Dai, Xiao-Ming Wu, Jiaren Xiao, Xiao Shen, and Dan Wang. Graph transfer learning via adversarial domain adaptation with graph convolution. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4908–4922, 2022.

[4] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

[5] Wenqi Fan, Yao Ma, Dawei Yin, Jianping Wang, Jiliang Tang, and Qing Li. Deep social collaborative filtering. In *Proceedings of the 13th ACM RecSys*, 2019.

[6] Taoran Fang, Yunchao Zhang, Yang Yang, Chunping Wang, and Lei Chen. Universal prompt tuning for graph neural networks. *Advances in Neural Information Processing Systems*, 36:52464–52489, 2023.

[7] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.

[8] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[9] Dmitri Goldenberg. Social network analysis: From graph theory to applications with python. *arXiv preprint arXiv:2102.10014*, 2021.

[10] William L Hamilton. *Graph representation learning*. Morgan & Claypool Publishers, 2020.

[11] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International conference on machine learning*, pages 4116–4126. PMLR, 2020.

[12] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[13] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30:595–608, 2016.

[14] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[15] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[16] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

[17] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[18] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and S Yu Philip. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(6):5879–5900, 2022.

[19] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM web conference 2023*, pages 417–428, 2023.

[20] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. Learning disentangled representations for recommendation. *Advances in neural information processing systems*, 32, 2019.

[21] Qian Ma, Zijian Zhang, Xiangyu Zhao, Haoliang Li, Hongwei Zhao, Yiqi Wang, Zitao Liu, and Wanyu Wang. Rethinking sensors modeling: Hierarchical information enhanced traffic forecasting. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023.

[22] Yujie Mo, Liang Peng, Jie Xu, Xiaoshuang Shi, and Xiaofeng Zhu. Simple unsupervised graph representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7797–7805, 2022.

[23] Federico Monti, Michael Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. *Advances in neural information processing systems*, 30, 2017.

[24] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[25] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. All in one: Multi-task prompting for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2120–2131, 2023.

[26] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. *arXiv preprint arXiv:2102.06514*, 2021.

[27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[28] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

[29] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121*, 2019.

[30] Jun Xia, Lirong Wu, Jintao Chen, Bozhen Hu, and Stan Z Li. Simgrace: A simple framework for graph contrastive learning without data augmentation. In *Proceedings of the ACM web conference 2022*, pages 1070–1079, 2022.

[31] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[32] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[33] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

[34] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 31, 2018.

[35] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823, 2020.

[36] Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and Philip S Yu. From canonical correlation analysis to self-supervised graph neural networks. *Advances in Neural Information Processing Systems*, 34:76–89, 2021.

[37] Mengmei Zhang, Linmei Hu, Chuan Shi, and Xiao Wang. Adversarial label-flipping attack and defense for graph neural networks. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 791–800. IEEE, 2020.

[38] Yifei Zhang, Hao Zhu, Zixing Song, Piotr Koniusz, and Irwin King. Costa: covariance-preserving feature augmentation for graph contrastive learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2524–2534, 2022.

[39] Yifei Zhang, Hao Zhu, Zixing Song, Piotr Koniusz, and Irwin King. Spectral feature augmentation for graph contrastive learning and beyond. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11289–11297, 2023.

[40] Tianqi Zhao, Ngan Thi Dong, Alan Hanjalic, and Megha Khosla. Multi-label node classification on graph-structured data. *arXiv preprint arXiv:2304.10398*, 2023.

[41] Yanqiao Zhu, Yichen Xu, Qiang Liu, and Shu Wu. An empirical study of graph contrastive learning. *arXiv preprint arXiv:2109.01116*, 2021.

[42] Yanqiao Zhu, Yichen Xu, Qiang Liu, and Shu Wu. An empirical study of graph contrastive learning. *arXiv preprint arXiv:2109.01116*, 2021.

[43] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020.

[44] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, pages 2069–2080, 2021.

[45] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.

APPENDIX

A. HYPER-PARAMETERS SEARCH SPACE

In our experiments, we performed a random hyper-parameter generation to examine the method's sensitivity to the hyper-parameters. The the search spaces are within typical range and details are outlined below:

- **Learning Rate**: The learning rate was searched over the values {0.01, 0.001, 0.0001, 0.00001}.
- **Drop Edge Rate**: The edge dropout rate was sampled from {0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}.
- **Drop Feature Rate**: The node feature dropout rate was selected from {0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}.
- **Hidden Dimension (`nhid`)**: The hidden dimension of the model was chosen from {128, 256, 512}.
- **Pnhid**: Hidden dimension for projection was searched over {128, 256, 512}.
- **Drop Edge Rate1 & Drop Edge Rate2**: The edge dropout rates for methods with two augmented view, each sampled dependently from {0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}.
- **Drop Feature Rate1 & Drop Feature Rate2**: The feature dropout rates for methods with two augmented view, also sampled dependently from {0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}.
- **Loss Weight**: The loss weight factor of method SUGRL was searched over {5, 10, 20, 50, 100}.
- **NN**: The NN of method SUGRL indicating the number of node permutations was selected from {1, 3, 4, 5, 10}.
- **Margin1 & Margin2**: The margins used for Loss of Method SUGRL are dependently sampled from {0.4, 0.5, 0.8, 0.9} and {0.1, 0.2, 0.4, 0.5, 0.6, 0.9}, respectively.
- **K**: The K, indicating the number of iterations used in SFA k was chosen from {1, 2}.
- τ: The temperature of InfoNCE loss, is sampled over the range {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}.

OmniRouter: Budget and Performance Controllable Multi-LLM Routing

Kai Mei, Wujiang Xu, Minghao Guo, Shuhang Lin, Yongfeng Zhang
Department of Computer Science, Rutgers University
{kai.mei, wujiang.xu, minghao.guo, shuhang.lin, yongfeng.zhang}@rutgers.edu

ABSTRACT

Large language models (LLMs) deliver superior performance but require substantial computational resources and operate with relatively low efficiency, while smaller models can efficiently handle simpler tasks with fewer resources. LLM routing is a crucial paradigm that dynamically selects the most suitable large language models from a pool of candidates to process diverse inputs, ensuring optimal resource utilization while maintaining response quality. Existing routing frameworks typically model this as a locally optimal decision-making problem, selecting the presumed best-fit LLM for each query individually, which overlooks global budget constraints, resulting in ineffective resource allocation. To tackle this problem, we introduce OmniRouter, a fundamentally controllable routing framework for multi-LLM serving. Instead of making per-query greedy choices, OmniRouter models the routing task as a constrained optimization problem, assigning models that minimize total cost while ensuring the required performance level. Specifically, a hybrid retrieval-augmented predictor is designed to predict the capabilities and costs of LLMs. After obtaining the predicted cost and performance, we utilize a constrained optimizer for cost-optimal assignments that employs Lagrangian dual decomposition with adaptive multipliers. It iteratively converges toward the globally optimal query-model allocation, dynamically balancing latency minimization against quality thresholds while adhering to heterogeneous capacity constraints. Experiments show that OmniRouter achieves up to 6.30% improvement in response accuracy while simultaneously reducing computational costs by at least 10.15% compared to competitive router baselines. The code and the dataset are available at `https://github.com/dongyuanjushi/OmniRouter`.

1. INTRODUCTION

Large Language Models (LLMs) have demonstrated remarkable capabilities, powering a diverse range of applications from chatbots [1; 45; 11; 15; 21; 53; 22; 50] and code assistants [23; 48; 30; 59] to computer-use agents. This success has spurred the widespread integration of LLMs into modern systems [44; 29; 32; 41; 57; 26; 49]. As LLM inference accelerates through hardware improvements, these LLM-integrated systems increasingly deploy not just one, but a set of LLMs with varying sizes, capabilities, and speeds as serving endpoints. This multi-LLM paradigm necessitates intelligent routing: the crucial task of directing incoming user queries to the most appropriate LLM instance to balance performance goals and resource efficiency [39]. The design of effective LLM routers has become an active area of research. Recent works have proposed various routing strategies to optimize for cost [31; 42; 2], latency [25; 43; 33; 58], and performance [56; 24; 5; 60]. Most existing LLM routing frameworks predominantly treat routing as a sequence of independent, greedy decisions. For each incoming query, these routers select a model based on a local optimization criterion (e.g., lowest predicted latency, cheapest model predicted to succeed) without considering system-wide resource limitations or overall performance targets. These approaches fundamentally fail to achieve Pareto-efficient resource allocation across diverse query workloads distributed among multiple LLMs. When operating under budget constraints, these greedy routers cannot globally optimize the performance-resource tradeoff, leading to suboptimal overall system efficacy. For instance, expending computational resources on marginally improving responses to simple queries may leave insufficient capacity for complex queries where performance improvements would be more valuable, which can been in Figure 1. More critically, these localized decision frameworks cannot effectively enforce global constraints such as maintaining target response quality levels while adhering to strict resource budgets, resulting in either performance degradation or resource over-utilization when deployed at scale. In summary, these approaches confront a fundamental optimization dilemma: maintaining optimal overall performance while operating under computational resource constraints and stringent budgetary limitations.

Recognizing these limitations, we introduce OmniRouter, a controllable routing framework that fundamentally reimagines how query-model assignments should be determined. Unlike existing approaches, OmniRouter formalizes routing in multi-LLM systems as a **constrained optimization problem** [54; 3; 19] with global performance requirements and operational constraints and proposes a two-stage routing solution. Inspired by the success of retrieval-augmented generation (RAG) in reducing hallucination of LLMs, we design a retrieval-augmented predictor at the first stage to enhance the prediction-based router. At the second stage, we design the constrained optimizer which employs a dual gradient-based approach to navigate the solution space by adjusting model selections based on quality requirements and model concurrency. Our contributions are as follows:

- We propose OmniRouter, a routing framework for multi-LLM serving that fundamentally regards routing as a con-

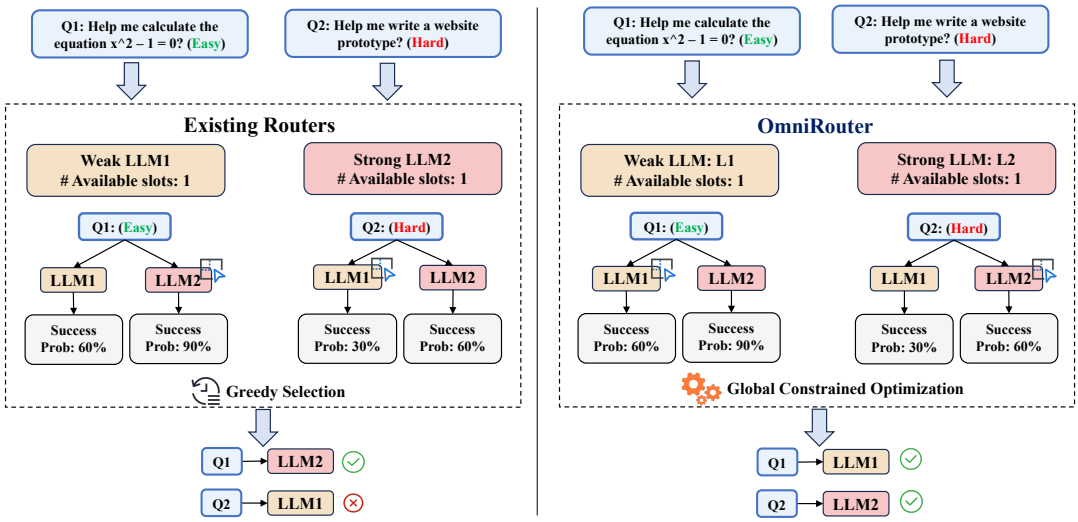


Figure 1: Comparison between traditional greedy routers and OmniRouter. Left: Greedy routers select models based on per-query optimization, leading to suboptimal allocations where Q1 (simple query) comes first and is assigned to the strong LLM, blocking Q2 (complex query) from accessing it. As a result, Q2 fails when assigned to the weak LLM. Right: OmniRouter employs constrained optimization to consider the global query distribution and model capabilities. It assigns the simple query to LLM1 (sufficient for the task) and reserves LLM2 for the complex query, thus maximizing overall success rate.

strained optimization problem rather than a series of greedy decisions, enabling system-level control of both performance and budget constraints.

- Extensive experiments show that OmniRouter can outperform competitive baselines across various serving scenarios and demonstrate significant improvements in response quality (up to 6.30%), cost efficiency (at least 10.15%), especially under tight operational constraints.

2. PROBLEM FORMALIZATION

Prior methods [31; 14; 34] typically score or rank models for each query independently based on heuristics such as cost-effectiveness or response likelihood. While effective in simple settings, such formulations struggle to account for system-level constraints such as limited model concurrency, global quality targets, and budget bounds that are critical in real-world multi-LLM deployments. In contrast, our approach models it as a constrained optimization problem.

Formally, given N queries and M models, let $a_{i,j} \in [0,1]$ denote the capability of whether the model j can successfully answer query i, and $c_{i,j}$ denote the money cost for answering this query. Each model j is subject to a concurrency limit L_j, and the overall system must maintain a minimum average performance constraint α across all N queries to ensure response quality. Our objective is to assign queries to models such that the total cost is minimized, while ensuring that the overall performance exceeds α and that no model exceeds its concurrency constraint. This problem can be formulated as the following:

$$\min_{x} \quad \sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j}x_{i,j} \quad s.t. \quad \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} a_{i,j}x_{i,j} \geq \alpha \qquad (1)$$
$$\sum_{i=1}^{N} x_{i,j} \leq L_j, \forall j \quad \sum_{j=1}^{M} x_{i,j} = 1, \forall i$$

It is a constrained optimization problem [3; 19] with global constraint α and local constraint L. We will elaborate how to solve this problem in the following section.

3. METHODOLOGY

As the capability $a_{i,j}$ and computational cost $c_{i,j}$ for each model-query pair are inherently uncertain at routing decision time. This uncertainty makes direct optimization of the routing variable $x_{i,j}$ particularly challenging. Joint optimization couples errors between prediction and allocation [13] and make the routing struggle with adaptability when query distributions shift or new models [52; 35; 55]. To address these challenges, we propose a two-stage approach that decouples the prediction of uncertain variables: $a_{i,j}$ and $c_{i,j}$ from the optimization of routing decisions $x_{i,j}$.

3.1 Retrieval-augmented Predictor

Inspired by retrieval-augmented generation (RAG) systems [27; 4], which enhance model outputs by incorporating relevant retrieved information, we design a retrieval-augmented predictor, as illustrated in Figure 2, to integrate the generalization capabilities of trained models with the retrieval precision of historical query-model data.

We employ bert-base-uncased [10] as our embedding encoder due to its representation capabilities and computational efficiency. This encoder processes both queries and LLM descriptions to generate embedding vectors E_q and E_l respectively, which serve as the foundation for both our training-based and retrieval-based prediction components. For the training-based component, we design a dual-head architecture built upon the embedding encoder. The first head focuses on model capability prediction, estimating a capability score $a_{i,j}^{pred} \in [0,1]$ for each query i and model j through:

$$a_{i,j}^{pred} = \sigma(\mathbf{W}_1(E_q^i \cdot E_l^j) + \mathbf{b}_1) \qquad (2)$$

where $\sigma(\cdot)$ denotes the sigmoid activation function, $\mathbf{W}_1$ and $\mathbf{b}_1$ are learnable parameters. The second head performs

sequence length classification by mapping the predicted output token length $l_{i,j}^{pred}$ into discrete buckets $\mathcal{B} = \{B_1, B_2, ..., B_k\}$, where $k = \lceil \frac{l_{max}}{bs} \rceil$, l_{max} is the maximum sequence length, and bs is the bucket size. The probability of length $l_{i,j}^{pred}$ is computed as:

$$l_{i,j}^{pred} = bs \cdot \text{softmax}(\mathbf{W}_2(E_q^i + E_l^j) + \mathbf{b}_2)_i \tag{3}$$

The bucketing strategy, a technique also employed in previous works [25; 58; 14], transforms the challenging task of precise token-level prediction into a more tractable bucket-level approximation. During training, we optimize these dual objectives using mean squared error (MSE) for capability prediction and cross-entropy loss for length bucket classification.

Concurrently, our retrieval-based component leverages a vector database to identify the top-k similar historical queries based on cosine similarity between query embeddings. For a given query embedding E_q^i, let $\mathcal{Q}_k(E_q^i)$ be the set of top-k similar queries retrieved from the database. The retrieval-based capability score $a_{i,j}^{retrieve}$ and output length score $l_{i,j}^{retrieve}$ for query i and model j are computed as:

$$l_{i,j}^{retrieve} = \frac{\sum_{q_m \in \mathcal{Q}_k(E_q^i)} \text{sim}(E_q^i, E_{q_m}) \cdot l_{m,j}}{\sum_{q_m \in \mathcal{Q}_k(E_q^i)} \text{sim}(E_q^i, E_{q_m})} \tag{4}$$

$$a_{i,j}^{retrieve} = \frac{\sum_{q_m \in \mathcal{Q}_k(E_q^i)} \text{sim}(E_q^i, E_{q_m}) \cdot a_{m,j}}{\sum_{q_m \in \mathcal{Q}_k(E_q^i)} \text{sim}(E_q^i, E_{q_m})} \tag{5}$$

where $\text{sim}(E_q^i, E_{q_m})$ denotes the cosine similarity between the current query embedding and the retrieved query embedding, while $l_{m,j}$ and $a_{m,j}$ represent the historically observed output length and capability score for retrieved query m on model j, respectively. To obtain the final predictions, we integrate both components through an adaptive fusion mechanism for each query-model pair:

$$a_{i,j} = \gamma \cdot a_{i,j}^{pred} + (1 - \gamma) \cdot a_{i,j}^{retrieve} \tag{6}$$

$$c_{i,j} = \delta \cdot tp_j(l_{i,j}^{pred}) + (1 - \delta) \cdot tp_j(l_{i,j}^{retrieve}) \tag{7}$$

where γ and $\delta \in [0, 1]$ are learnable parameters that control the balance between trained and retrieved predictions, and $tp(\cdot)$ is a function that maps the predicted token length to a computational cost estimate based on model j's token prices. This integrated retrieval-augmented architecture ensures our predictor benefits from both generalizable patterns learned during training and specific historical performance data, resulting in more accurate predictions of the uncertain variables needed for the subsequent optimization stage.

3.2 Constrained Optimizer

At the second stage, we solve the $x_{i,j}$ with the predicted $a_{i,j}$ and $c_{i,j}$. Inspired by optimizers [54; 3; 47], we leverage the Lagrangian dual theory and introduce Lagrangian multipliers to convert the primal problem into its Lagrangian dual problem. By introducing Lagrangian multipliers $\lambda_1, \lambda_{2,j}, \mu_i$, we get the Lagrangian relaxation function of the original problem as follows:

$$\begin{aligned} \min_x \quad & \sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j}x_{i,j} \\ \text{s.t.} \quad & \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} a_{i,j}x_{i,j} \geq \alpha \\ & \sum_{i=1}^{N} x_{i,j} \leq L_j, \forall j \in [M] \quad \sum_{j=1}^{M} x_{i,j} = 1, \forall i \in [N]. \end{aligned} \tag{8}$$

we introduce three types of Lagrange multipliers as:

- $\lambda_1 \geq 0$ for the quality constraint (inequality)
- $\lambda_{2,j} \geq 0$ for each capacity constraint (inequality)
- μ_i for each assignment constraint (equality)

and write the Lagrangian function as:

$$\begin{aligned} & L(x, \lambda_1, \lambda_{2,j}, \mu_i) \\ & = \sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j}x_{i,j} + \lambda_1 \left(-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} a_{i,j}x_{i,j} + \alpha \right) \\ & \quad + \sum_{j=1}^{M} \lambda_{2,j} \left(\sum_{i=1}^{N} x_{i,j} - L_j \right) + \sum_{i=1}^{N} \mu_i \left(\sum_{j=1}^{M} x_{i,j} - 1 \right) \end{aligned} \tag{9}$$

which can rearranged to group terms with $x_{i,j}$:

$$\begin{aligned} & L(x, \lambda_1, \lambda_{2,j}, \mu_i) \\ & = \sum_{i=1}^{N}\sum_{j=1}^{M} x_{i,j} \left(c_{i,j} - \frac{\lambda_1 a_{i,j}}{N} + \lambda_{2,j} + \mu_i \right) \\ & \quad + \lambda_1 \alpha - \sum_{j=1}^{M} \lambda_{2,j} L_j - \sum_{i=1}^{N} \mu_i \end{aligned} \tag{10}$$

The KKT optimality conditions for this problem are: Stationarity:

$$\frac{\partial L}{\partial x_{i,j}} = c_{i,j} - \frac{\lambda_1 a_{i,j}}{N} + \lambda_{2,j} + \mu_i = 0, \quad \forall i, j \tag{11}$$

Primal Feasibility:

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} a_{i,j}x_{i,j} \geq \alpha, \quad \sum_{i=1}^{N} x_{i,j} \leq L_j, \forall j, \quad \sum_{j=1}^{M} x_{i,j} = 1, \forall i \tag{12}$$

Dual Feasibility:

$$\lambda_1 \geq 0, \quad \lambda_{2,j} \geq 0, \forall j \tag{13}$$

Complementary Slackness:

$$\lambda_1 \left(\alpha - \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} a_{i,j}x_{i,j} \right) = 0, \quad \lambda_{2,j} \left(L_j - \sum_{i=1}^{N} x_{i,j} \right) = 0, \quad \forall j \tag{14}$$

Note that for each i, we have the constraint: $\sum_{j=1}^{M} x_{i,j} = 1$. This implies that for each query i, exactly one LLM j must be selected. From the stationarity condition, for any fixed i, comparing two different indices j and k:

$$c_{i,j} - \frac{\lambda_1 a_{i,j}}{N} + \lambda_{2,j} + \mu_i = 0 \tag{15}$$

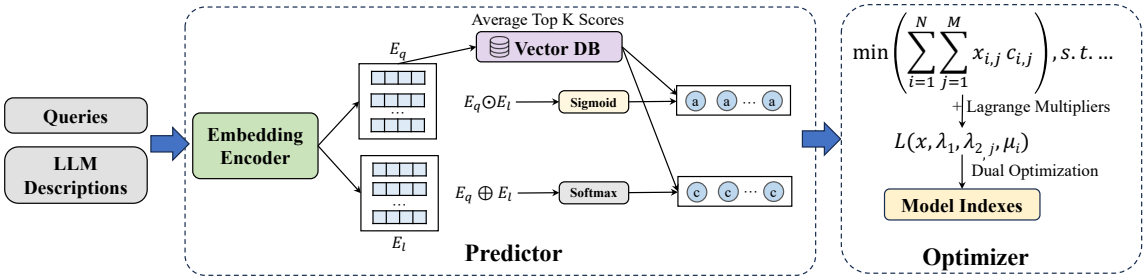


Figure 2: Illustration of OmniRouter, including the hybrid predictor and constrained optimizer.

$$c_{i,k} - \frac{\lambda_1 a_{i,k}}{N} + \lambda_{2,k} + \mu_i = 0 \tag{16}$$

Subtracting these equations eliminates μ_i:

$$(c_{i,j} - \frac{\lambda_1 a_{i,j}}{N} + \lambda_{2,j}) = (c_{i,k} - \frac{\lambda_1 a_{i,k}}{N} + \lambda_{2,k}) \tag{17}$$

This implies that for a given i, the optimal solution should choose the j that minimizes $(c_{i,j} - \frac{\lambda_1 a_{i,j}}{N} + \lambda_{2,j})$.
Then the dual function becomes:

$$g(\lambda_1, \lambda_2) \tag{18}$$
$$= \min_{x_{i,j}} \{\sum_{i=1}^{N}\sum_{j=1}^{M} x_{i,j}(c_{i,j} - \frac{\lambda_1 a_{i,j}}{N} + \lambda_{2,j}) + \lambda_1 \alpha - \sum_{j=1}^{M} \lambda_{2,j} L_j\}$$

Note that μ_i has disappeared from the dual function because we've analytically incorporated the equality constraints. The dual problem can now be written as:

$$\max_{\lambda_1, \lambda_2} \quad g(\lambda_1, \lambda_2), \quad \text{s.t.} \quad \lambda_1 \geq 0, \quad \lambda_{2,j} \geq 0, \quad \forall j \in [M] \tag{19}$$

The partial derivatives for the remaining multipliers are: For λ_1:

$$\frac{\partial L}{\partial \lambda_1} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} x_{i,j} a_{i,j} + \alpha \tag{20}$$

For $\lambda_{2,j}$:

$$\frac{\partial L}{\partial \lambda_{2,j}} = \sum_{i=1}^{N} x_{i,j} - L_j, \quad \forall j \in [M] \tag{21}$$

The gradient ascent update rules are:

$$\lambda_1^{t+1} = \max\left(\lambda_1^t + \alpha_1 \left(-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} x_{i,j} a_{i,j} + \alpha\right), 0\right) \tag{22}$$

$$\lambda_{2,j}^{t+1} = \max\left(\lambda_{2,j}^t + \alpha_2 \left(\sum_{i=1}^{N} x_{i,j} - L_j\right), 0\right), \quad \forall j \in [M] \tag{23}$$

For fixed multipliers, the optimal assignment for each i is:

$$x_{i,j} = \begin{cases} 1 & \text{if } j = j_i^* \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

where

$$j_i^* = \arg\min_{j \in [M]} \left(c_{i,j} - \frac{\lambda_1 a_{i,j}}{N} + \lambda_{2,j}\right) \tag{25}$$

- λ_1 acts as a penalty for violating the quality constraint. When the average quality is below α, λ_1 increases, encouraging selection of higher-quality options.
- $\lambda_{2,j}$ penalizes capacity violations for each model j. When a model exceeds its capacity L_j, its corresponding multiplier increases, making it less attractive in subsequent iterations.
- The equality constraints (μ_i) are handled analytically by direct substitution, which simplifies the dual problem.
- The algorithm alternates between updating multipliers and assignments until convergence.

For $\lambda_{2,j}$, its partial derivative shows the workload violation for each model j. If a model's assigned queries exceed its capacity, the gradient is positive, increasing $\lambda_{2,j}$, which makes this overloaded model less attractive in subsequent iterations.

4. EVALUATION

In this section, we propose the following research questions regarding the performance of OmniRouter and conduct experiments to answer these research questions.

- **RQ1:** What is the routing performance of the OmniRouter and how is compared with existing routing frameworks?
- **RQ2:** Whether the OmniRouter can successfully control budget costs and ensure response accuracy when constraints (i.e., performance constraint α and concurrency constraint L) vary?
- **RQ3:** What is the influence of different factors (i.e., modules and parameters in predictors) of OmniRouter on routing performance?

4.1 Datasets

We collect 2.7k questions sourced from established knowledge and mathematical reasoning datasets, including MMLU [17], GPQA [38], MATH [18], and GSM8K [8]. We select 10 different models, including 5 relatively weak models, i.e., Qwen2.5 (7B-Instruct, 14B-Instruct, 32B-Instruct) [51], Gemma2 (9B-it, 27B-it) [46] and 5 relatively strong models, i.e., Qwen2.5-72B-Instruct, GPT 4o-mini, GPT-4o [1],

Gemini-1.5-flash [45], Claude-3.5-sonnet [1]. And we collect the response correctness and token usage of these models to answer these questions using Llama-3.1-70B-Instruct [11] as the evaluation judge [28]. The Llama-3.1-70B is excluded from the LLM candidate pool to reduce potential biases. The prompt is shown as below.

Prompt for using LLM as the judge.

Prompt: The ground truth answer is: {gt_answer}. The prediction answer is: extracted_answer. Judge whether the prediction answer is correct or not. You just need to output 'True' or 'False'.

The statistics of the data we use are in Table 1. We also set the difficulty for each question, which is determined by the number of models that can answer the question correctly, i.e., Easy: $\{8, 9, 10\}$ models can answer correctly, Medium: $\{4, 5, 6, 7\}$ models can answer correctly, Hard: $\{0, 1, 2, 3\}$ models can answer correctly. And the proposition is: 78.4%, 15.2%, 6.4%, respectively.

Table 1: Statistics of the data distribution in our dataset.

Data Source	# Samples	Ratio
MMLU	1000	37.06%
GPQA-Diamond	198	7.33%
Math-500	500	18.53%
GSM8K	1000	37.06%

For calculating costs for LLMs to answer queries, we refer to litellm[2] for calculating money cost, the money price regarding the model we use are shown in Table 2.

4.2 Setup

We conduct our experiments on an Ubuntu 22.04 machine equipped with 8 RTX A5000 GPUs to serve multiple LLMs using Ollama [4]. During the implementation, we set the maximum output length to 1024 tokens to prevent the influence of extremely long output sequences, set top k as 16 during retrieval, and γ and δ as 0.5 during aggregation. To calculate money costs, we refer to Litellm's costmap for price reference. The detailed price map can be found in Appendix B. Our experiments are conducted on the continuous-batching setting [26]. The serving system processes requests by dynamically forming batches based on the incoming query traffic, providing a more realistic evaluation environment for routing frameworks. In our simulation, we model real-world traffic patterns by randomly adding $n \in \{1, 2, 3, 4\}$ queries to the queue every 0.1 seconds and performing routing decisions at 1-second intervals. If not specifically mentioned, we set the two constraints $\alpha = 0.75$ and the concurrent workload constraint $L = 4$, which are applied uniformly across all LLMs. Controllability analysis of different values of α and L is presented in Section 4.4. We evaluate OmniRouter against the following routing baselines:

Cost-oriented: We adapt S3 [25] and PO [58], which are originally designed for latency optimization as cost-oriented baselines. Specifically, S3 employs DistilBERT [40] and PO employs Vicuna-7B [7] as output token length predictors, respectively.

Performance-oriented: For performance-focused routing, we implement EmbedLLM [60] and RouterDC [5]. EmbedLLM leverages an encoder-decoder architecture to embed LLM representations, while RouterDC employs contrastive learning to model query-LLM relationships, both aiming to maximize response quality regardless of computational cost.

Cost-performance Coordinated: For baselines which balance both cost and performance objectives, we employ Hybrid-LLM [2], which constructs a probabilistic router, and CARROT [42], which implements dual predictors using Llama3-8B [11] as specified in its original implementation.

4.3 RQ1: Routing Performance

The comparison of overall routing performance can be observed from Table 3. OmniRouter demonstrates substantial advantages over all baselines across both metrics. Given the theoretical performance bounds (lower bound is 57.41% when the worst LLM for each query is selected and upper bound is 90% when the best LLM for each query is selected), the improvement achieved by OmniRouter (1.3% to 6.48% absolute improvement) is particularly significant, which represents a meaningful advancement within this constrained optimization space. When compared to cost-oriented baselines (S3 and PO), OmniRouter achieves significantly higher success rates (5.74% and 6.48% improvements respectively) while maintaining comparable or even lower costs. Against performance-oriented approaches such as EmbedLLM and RouterDC, our method achieves both higher success rates and substantially lower costs (approximately 41% cost reduction compared to RouterDC). Even when compared to cost-performance coordinated methods (Hybrid-LLM and CARROT), OmniRouter demonstrates superior performance in both dimensions.

Performance of Predictors. We also compare the effectiveness of OmniRouter's predictor with the baselines we have mentioned. Table 4 demonstrates OmniRouter's significant advantages in prediction accuracy. Our approach achieves 81.3% capability prediction accuracy, outperforming the strongest baseline (RouterDC) by 5.2%. For length prediction, OmniRouter delivers 45.2% exact match accuracy and 80.6% relaxed (± 1) accuracy, surpassing all alternatives by over 13.0%.

Routing Performance across Query Difficulty Levels. To assess the routing performance of OmniRouter deeper, we analyze its allocation patterns across different difficulty categories. Figure 3 reveals OmniRouter's sophisticated routing strategy across query difficulty levels. For easy queries (78.4% of workload), the system successfully routes 49.2%

[1] https://claude.ai/
[2] https://docs.litellm.ai/docs/
[4] https://ollama.com/

Table 2: Money Cost Map of LLMs

Model Name	1M Input Tokens ($)	1M Output Tokens ($)
Qwen-2.5-7B-Instruct [51]	0.267	0.267
Qwen-2.5-14B-Instruct [51]	0.534	0.534
Qwen-2.5-32B-Instruct [51]	1.22	1.22
Qwen-2.5-72B-Instruct [51]	2.745	2.745
Gemma-2-9B-it [46]	0.343	0.343
Gemma-2-27B-it [46]	1.03	1.03
gpt-4o-mini [1]	0.15	0.6
gpt-4o [1]	2.5	10
gemini-1.5-flash [45]	0.075	0.3
claude-3-5-sonnet[3]	3	15

Table 3: Routing performance comparison, where we use accuracy, i.e., whether the routed LLM successfully answers the assigned query, and money cost of calling respective LLMs.

Metric	S3	PO	EmbedLLM	RouterDC	Hybrid-LLM	CARROT	**OmniRouter**
Accuracy (↑)	69.45%	68.71%	72.96%	73.89%	71.48%	72.41%	**75.19%**
$ Cost (↓)	0.0585	0.0610	0.0896	0.0874	0.0722	0.0680	**0.0515**

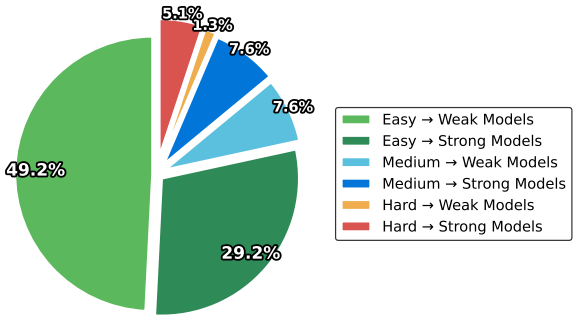


Figure 3: Distribution of OmniRouter's query routing decisions across difficulty levels and model capabilities.

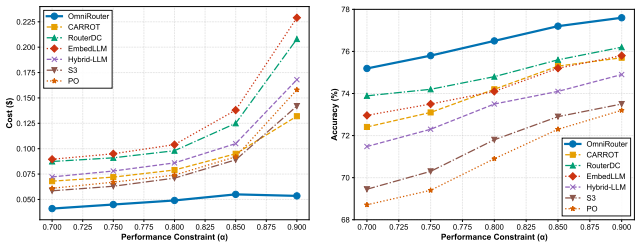
Table 4: Performance of predictors on model capability and output token length prediction.

Method	Capability Acc.	Length Bucket Acc.	
		Exact Match	±1
S3	–	0.333	0.656
PO	–	0.325	0.683
EmbedLLM	0.732	–	–
RouterDC	0.761	–	–
Hybrid-LLM	0.708	–	–
CARROT	0.745	0.312	0.672
OmniRouter	**0.813**	**0.452**	**0.806**

of them to weak models. Medium-difficulty queries receive balanced treatment with equal distribution between model types (7.6% each). For hard queries, OmniRouter shows a strong preference for routing to strong models by a 4:1 ratio (5.1% vs 1.3%), demonstrating the capability of OmniRouter to match appropriate models.

4.4 RQ2: Controllability Analysis

We investigate how different routing approaches respond to varying operational constraints, i.e., performance and concurrent constraints. Figure 4 reveals a critical limitation of existing greedy routing approaches: their inability to bound costs under stringent performance requirements. As the performance constraint (α) increases from 0.70 to 0.90, baseline methods exhibit exponential cost growth: at $\alpha = 0.90$, the cost of EmbedLLM reaches \$0.229, more than four times OmniRouter's cost of \$0.054, creating a substantial cost gap ($\Delta = 0.175$). This cost explosion occurs because greedy routing strategies inherently prioritize immediate performance gains when performance thresholds rise, defaulting to routing increasingly more queries to the most powerful (and expensive) models regardless of actual query complexity. In contrast, OmniRouter demonstrates remarkably controlled cost scaling, with only a 48% increase over the same range and even a slight cost reduction at $\alpha > 0.85$. Similarly, in Figure 5, it demonstrates system behavior under varying concurrency constraints, revealing another key weakness of greedy approaches. As available concurrency decreases from 8 to 1, baseline methods show substantial cost increases (EmbedLLM: 159%, RouterDC: 147%) and significant performance degradation. When concurrency is highly limited ($L = 1$), the performance gap between OmniRouter and the weakest baseline (S3) becomes dramatic ($\Delta = 11.7\%$). This occurs because greedy strategies struggle to make effective compromises when resource constraints tighten. By constrast, OmniRouter substantially performs better, maintaining 73.8% accuracy even at $L = 1$ while experiencing only moderate cost increases.



(a) Cost vs. Performance Constraint (b) Accuracy vs. Performance Constraint

Figure 4: Impact of performance constraint (α) on cost efficiency and routing accuracy. As performance requirements increase, greedy methods exhibit unbounded cost growth while OmniRouter's constraint optimization maintains controlled scaling.

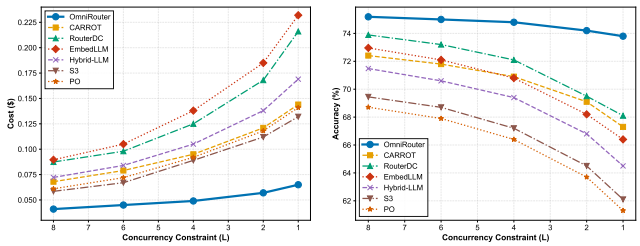
Table 5: Effects of removing different modules in the predictor on routing performance.

Predictor	Capability Acc (↑)	Bucket Acc (↑)	Acc (↑)	$ Cost (↓)
Full	**0.813**	**0.806**	**75.19%**	**0.0515**
w/o Retrieval	0.751	0.724	71.85%	0.0698
w/o Training	0.728	0.682	73.33%	0.0642

4.5 RQ3: Ablation Studies

Effects of Removing Modules in the Predictor. To evaluate the contribution of each component in our retrieval-augmented predictor, we conduct ablation studies by removing key modules from the full model. Results are presented in Table 5.

Removing the retrieval component leads to a substantial drop in both capability accuracy (7.6% decrease from 0.813 to 0.751) and length prediction accuracy (10.2% decrease from 0.806 to 0.724). This degradation directly impacts routing effectiveness, resulting in lower success rate (4.4% decrease) and significantly higher operational costs (35.5% increase from \$0.0515 to \$0.0698). These findings highlight that historical query information provides critical contextual signals that enhance prediction. On the other hand, after removing the training-based component, although performance accuracy sees a larger decline (10.5% decrease to



(a) Cost vs. Concurrency Constraint (b) Accuracy vs. Concurrency Constraint

Figure 5: Impact of concurrency constraint (L) on cost efficiency and routing accuracy. As available parallelism decreases, greedy methods struggle to make effective compromises, while OmniRouter's constraint optimization maintains balanced allocations.

Table 6: Effects of number of buckets used in predictors on routing performance.

# Bucket	Bucket Acc (↑)		Capability Acc. (↑)	$ Cost (↓)
	Exact Match	±1		
10	**0.452**	**0.806**	**75.19%**	**0.0515**
20	0.269	0.526	73.22%	0.0725
50	0.162	0.299	73.22%	0.834
100	0.124	0.189	70.74%	0.0962

0.728) compared to the no-retrieval variant, the success rate decreases less dramatically (only 2.5% reduction). This suggests that the retrieval mechanism alone sometimes can maintain reasonable routing decisions, which further validate our retrieval-augmented choice to enhance predictors.

Effects of Key Parameters within OmniRouter's Predictors. We investigate the sensitivity of OmniRouter's routing effectiveness to key design parameters within its prediction components. As shown in Table 6, the number of buckets used for discretizing output length predictions significantly impacts routing performance. A smaller number of buckets (10) yields substantially higher prediction accuracy (45.2% exact match, 80.6% ±1 accuracy), which directly leads to higher routing performance (75.19% success rate) and lower operational costs ($0.0515). This finding suggests that coarser-grained discretization may be sufficient for effective routing decisions, while excessive granularity can introduce noise. Regarding the retrieval component, Table 7 demonstrates how varying K (the number of historical samples) affects prediction accuracy and routing performance. The system achieves optimal performance at $K = 16$, with the highest success rate (75.19%) and lowest cost ($0.0515). We observe that prediction accuracy remains relatively stable across moderate K values (8-32), with only slight degradation at the extremes. Too few samples ($K = 4$) provide insufficient historical context, while too many ($K = 64$) likely introduce noise from less relevant queries, both resulting in suboptimal routing decisions. This indicates the retrieval-augmentation requires careful calibration to balance between relevant historical information and potentially misleading outliers.

5. THEORETICAL PROOF

Proposition 1. *Given the Lagrangian dual problem with multipliers $\lambda_1 \geq 0$ and $\lambda_{2,j} \geq 0$ for $j \in [M]$, the optimal assignment $x^*_{i,j}$ for the constrained optimization problem is uniquely determined by:*

$$x^*_{i,j} = \begin{cases} 1 & \text{if } j = \arg\min_{j' \in [M]} \left(c_{i,j'} - \frac{\lambda_1^* a_{i,j'}}{N} + \lambda^*_{2,j'} \right) \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

Table 7: Effects of variations of K in historical samples on routing performance.

K	Capability Acc. (↑)	Length Acc (↑)		SR (↑)	$ Cost (↓)
		Exact Match	±1		
4	0.773	0.430	0.784	74.07%	0.0614
8	0.785	0.443	0.786	74.81%	0.0565
16	**0.806**	**0.452**	**0.806**	**75.19%**	**0.0515**
32	0.785	0.440	0.776	74.81%	0.0565
64	0.780	0.422	0.764	73.70%	0.0684

PROOF. We start with the original optimization problem:

$$\begin{aligned} \min_x \quad & \sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j} x_{i,j} \\ \text{s.t.} \quad & \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} a_{i,j} x_{i,j} \geq \alpha \quad \sum_{i=1}^{N} x_{i,j} \leq L_j, \quad \forall j \in [M] \\ & \sum_{j=1}^{M} x_{i,j} = 1, \quad \forall i \in [N] \quad x_{i,j} \in \{0,1\}, \quad \forall i,j \end{aligned} \tag{27}$$

The Lagrangian function is:

$$\begin{aligned} & L(x, \lambda_1, \lambda_2, \mu) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{M} c_{i,j} x_{i,j} + \lambda_1 \left(\alpha - \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} a_{i,j} x_{i,j} \right) \\ &\quad + \sum_{j=1}^{M} \lambda_{2,j} \left(\sum_{i=1}^{N} x_{i,j} - L_j \right) + \sum_{i=1}^{N} \mu_i \left(\sum_{j=1}^{M} x_{i,j} - 1 \right) \end{aligned} \tag{28}$$

From the KKT stationarity condition:

$$\frac{\partial L}{\partial x_{i,j}} = c_{i,j} - \frac{\lambda_1 a_{i,j}}{N} + \lambda_{2,j} + \mu_i = 0 \tag{29}$$

where μ_i is the Lagrangian multiplier for the constraint $\sum_{j=1}^{M} x_{i,j} = 1$.
Rearranging the stationarity condition:

$$\mu_i = -\left(c_{i,j} - \frac{\lambda_1 a_{i,j}}{N} + \lambda_{2,j} \right) \tag{30}$$

Since $\sum_{j=1}^{M} x_{i,j} = 1$ and $x_{i,j} \in \{0,1\}$, exactly one $x_{i,j}$ equals 1 for each query i. For the optimal solution, we choose j^* that minimizes the expression in the stationarity condition:

$$j^* = \arg\min_{j \in [M]} \left(c_{i,j} - \frac{\lambda_1 a_{i,j}}{N} + \lambda_{2,j} \right) \tag{31}$$

Therefore, $x_{i,j^*} = 1$ and $x_{i,j} = 0$ for $j \neq j^*$, which completes the proof. □

Proposition 2. *Given the optimal dual variable λ_1^* obtained from the gradient ascent update, if $\lambda_1^* > 0$, then the quality constraint is active and satisfied with equality:*

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} a_{i,j} x^*_{i,j} = \alpha \tag{32}$$

PROOF. From the complementary slackness condition in the KKT conditions:

$$\lambda_1^* \left(\alpha - \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} a_{i,j} x_{i,j}^* \right) = 0 \tag{33}$$

This equation states that either:

(i) $\lambda_1^* = 0$, or

(ii) $\alpha - \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} a_{i,j} x_{i,j}^* = 0$

Given that $\lambda_1^* > 0$, condition (i) is false, thus condition (ii) must hold:

$$\alpha - \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} a_{i,j} x_{i,j}^* = 0 \tag{34}$$

Rearranging yields:

$$\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} a_{i,j} x_{i,j}^* = \alpha \tag{35}$$

This demonstrates that when $\lambda_1^* > 0$, the quality constraint is satisfied with equality. The dual variable λ_1^* acts as a "price" for quality: the optimizer finds the solution that exactly meets the quality threshold α while minimizing cost. □

6. RELATED WORK

LLM Routing. To optimize performance and inference cost in multi-LLM serving systems, researchers have developed increasingly sophisticated routing approaches. RouteLLM [31] and CARROT [42] propose training small LLMs as routers to balance cost and performance. EmbedLLM [60] introduces a specialized encoder-decoder network for embedding LLM representations. HybridLLM [2] addresses data imbalance issues by proposing a probabilistic router to better represent different model capabilities across query types. C2MAB-V [9] treats routing as a contextual multi-armed bandit problem, enabling exploration-exploitation trade-offs in model selection. GraphRouter [12] leverages graph learning to jointly modeling the query-model, query-query, and model-model relationship for building routers. While these approaches have made substantial progress, they predominantly operate as greedy decision-makers, optimizing for individual queries without considering global system constraints or the implications for subsequent requests. DeepSieve [16] utilizes LLM-as-a-Knowledge-Router to enable effective reasoning across multiple heterogeneous knowledge sources, enhancing retrieval-augmented generation (RAG) systems by dynamically routing sub-queries to the most appropriate source.

LLM Generation Length Prediction. Predicting LLM generation length is crucial for optimizing computational resources. Early attempts like Magnus [6] employed random forest algorithms but achieved limited accuracy. Subsequent research has explored two main directions of prediction models: encoder-only models for classification (DynamoLLM [43], S3 [25], TerriInfer [20], SSJF [37], and μ3 [36]) and decoder-only models for generative prediction like Perception-only (PO) [58]. LTR [14] reformulated this as a ranking problem and utilized listwise ranking for predictor training.

7. CONCLUSION

In this paper, we introduce OmniRouter, a routing framework that fundamentally reframes LLM routing as a constrained optimization problem rather than a series of greedy decisions. We develop a two-stage approach with a hybrid predictor that accurately estimates model capabilities and costs, and a constrained optimizer that minimizes operational expenses while satisfying performance requirements. Our experiments with the OmniRouteEval dataset demonstrate that OmniRouter achieves up to 6.30% higher response accuracy while reducing costs by at least 10.15% compared to existing methods. OmniRouter maintains remarkable stability under varying constraints, precisely where greedy approaches fail. These results confirm OmniRouter's effectiveness for more realistic environments requiring both performance guarantees and cost control.

8. REFERENCES

[1] ACHIAM, J., ADLER, S., AGARWAL, S., AHMAD, L., AKKAYA, I., ALEMAN, F. L., ALMEIDA, D., ALTENSCHMIDT, J., ALTMAN, S., ANADKAT, S., ET AL. Gpt-4 technical report. *arXiv preprint:2303.08774 (OpenAI Technical Report)* (2023).

[2] ANONYMOUS. Hybrid llm: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618* (2024).

[3] BERTSEKAS, D. P. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[4] BORGEAUD, S., MENSCH, A., HOFFMANN, J., CAI, T., RUTHERFORD, E., MILLICAN, K., VAN DEN DRIESSCHE, G. B., LESPIAU, J.-B., DAMOC, B., CLARK, A., ET AL. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning* (2022), PMLR, pp. 2206–2240.

[5] CHEN, S., JIANG, W., LIN, B., KWOK, J., AND ZHANG, Y. Routerdc: Query-based router by dual contrastive learning for assembling large language models. *Advances in Neural Information Processing Systems 37* (2024), 66305–66328.

[6] CHENG, K., HU, W., WANG, Z., DU, P., LI, J., AND ZHANG, S. Enabling efficient batch serving for lmaas via generation length prediction. *arXiv preprint arXiv:2406.04785* (2024).

[7] CHIANG, W.-L., LI, Z., LIN, Z., SHENG, Y., WU, Z., ZHANG, H., ZHENG, L., ZHUANG, S., ZHUANG, Y., GONZALEZ, J. E., ET AL. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023) 2*, 3 (2023), 6.

[8] COBBE, K., KOSARAJU, V., BAVARIAN, M., CHEN, M., JUN, H., KAISER, L., PLAPPERT, M., TWOREK, J., HILTON, J., NAKANO, R., ET AL. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168* (2021).

[9] DAI, X., LI, J., LIU, X., YU, A., AND LUI, J. Cost-effective online multi-llm selection with versatile reward models. *arXiv preprint arXiv:2405.16587* (2024).

[10] Devlin, J. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[11] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783 (Meta AI Technical Report)* (2024).

[12] Feng, T., Shen, Y., and You, J. Graphrouter: A graph-based router for llm selections. *arXiv preprint arXiv:2410.03834* (2024).

[13] Feng, X., Wu, M., Feng, Y., Yin, X., Wang, S., Zeng, Y., Zeng, X., Wang, Z., Qin, R., Hu, G., et al. Towards understanding and mitigating the training data quality of large language models. In *Advances in Neural Information Processing Systems* (2023), vol. 36.

[14] Fu, Y., Zhu, S., Su, R., Qiao, A., Stoica, I., and Zhang, H. Efficient llm scheduling by learning to rank. *arXiv preprint arXiv:2408.15792* (2024).

[15] Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[16] Guo, M., Zeng, Q., Zhao, X., Liu, Y., Yu, W., Du, M., Chen, H., and Cheng, W. Deepsieve: Information sieving via llm-as-a-knowledge-router. *arXiv preprint arXiv:2507.22050* (2025).

[17] Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300* (2020).

[18] Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874* (2021).

[19] Homaifar, A., Qi, C. X., and Lai, S. H. Constrained optimization via genetic algorithms. *Simulation 62*, 4 (1994), 242–253.

[20] Hu, C., Huang, H., Xu, L., Chen, X., Xu, J., Chen, S., Feng, H., Wang, C., Wang, S., Bao, Y., et al. Inference without interference: Disaggregate llm inference for mixed downstream workloads. *arXiv preprint arXiv:2401.11181* (2024).

[21] Hua, W., Yang, X., Jin, M., Li, Z., Cheng, W., Tang, R., and Zhang, Y. Trustagent: Towards safe and trustworthy llm-based agents. In *2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024* (2024), Association for Computational Linguistics (ACL), pp. 10000–10016.

[22] Hua, W., Zhu, K., Li, L., Fan, L., Lin, S., Jin, M., Xue, H., Li, Z., Wang, J., and Zhang, Y. Disentangling logic: The role of context in large language model reasoning capabilities. *arXiv preprint arXiv:2406.02787* (2024).

[23] Hui, B., Yang, J., Cui, Z., Yang, J., Liu, D., Zhang, L., Liu, T., Zhang, J., Yu, B., Lu, K., et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186* (2024).

[24] Jin, C., Peng, H., Zhang, Q., Tang, Y., Metaxas, D. N., and Che, T. Two heads are better than one: Test-time scaling of multi-agent collaborative reasoning. *arXiv preprint arXiv:2504.09772* (2025).

[25] Jin, Y., Wu, C.-F., Brooks, D., and Wei, G.-Y. S3: Increasing gpu utilization during generative inference for higher throughput. *Advances in Neural Information Processing Systems 36* (2023), 18015–18027.

[26] Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles* (2023), pp. 611–626.

[27] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems 33* (2020), 9459–9474.

[28] Li, H., Dong, Q., Chen, J., Su, H., Zhou, Y., Ai, Q., Ye, Z., and Liu, Y. Llms-as-judges: a comprehensive survey on llm-based evaluation methods. *arXiv preprint arXiv:2412.05579* (2024).

[29] Mei, K., Li, Z., Xu, S., Ye, R., Ge, Y., and Zhang, Y. Aios: Llm agent operating system. *arXiv e-prints, pp. arXiv–2403* (2024).

[30] Nijkamp, E., Hayashi, H., Xiong, C., Savarese, S., and Zhou, Y. Codegen2: Lessons for training llms on programming and natural languages. *arXiv preprint arXiv:2305.02309* (2023).

[31] Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., and Stoica, I. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665* (2024).

[32] Packer, C., Fang, V., Patil, S. G., Lin, K., Wooders, S., and Gonzalez, J. E. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560* (2023).

[33] Panda, P., Magazine, R., Devaguptapu, C., Takemori, S., and Sharma, V. Adaptive llm routing under budget constraints. *arXiv preprint arXiv:2508.21141* (2025).

[34] Parkar, R. S., Kim, J., Park, J. I., and Kang, D. Selectllm: Can llms select important instructions to annotate? *arXiv preprint arXiv:2401.16553* (2024).

[35] Qin, X., Yang, Y., Li, X., Dong, S., Huang, S., Ji, H., and Li, L. Towards robust llm-based decision-making: A calibration and planning approach. In *International Conference on Learning Representations* (2023).

[36] Qiu, H., Mao, W., Patke, A., Cui, S., Jha, S., Wang, C., Franke, H., Kalbarczyk, Z., Başar, T., and Iyer, R. K. Power-aware deep learning model serving with {μ-Serve}. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)* (2024), pp. 75–93.

[37] Qiu, H., Mao, W., Patke, A., Cui, S., Jha, S., Wang, C., Franke, H., Kalbarczyk, Z. T., Başar, T., and Iyer, R. K. Efficient interactive llm serving with proxy model-based sequence length prediction. In *The 5th International Workshop on Cloud Intelligence / AIOps at ASPLOS 2024* (San Diego, CA, USA, 2024), vol. 5, Association for Computing Machinery, pp. 1–7.

[38] Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022* (2023).

[39] Salaria, S., Liu, Z., and Gonzalez, N. M. Metametrics and best practices for system-level inference performance benchmarking, 2025.

[40] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[41] Shi, Z., Mei, K., Jin, M., Su, Y., Zuo, C., Hua, W., Xu, W., Ren, Y., Liu, Z., Du, M., Deng, D., and Zhang, Y. From commands to prompts: LLM-based semantic file system for aios. In *The Thirteenth International Conference on Learning Representations* (2025).

[42] Somerstep, S., Polo, F. M., de Oliveira, A. F. M., Mangal, P., Silva, M., Bhardwaj, O., Yurochkin, M., and Maity, S. Carrot: A cost aware rate optimal router. *arXiv preprint arXiv:2502.03261* (2025).

[43] Stojkovic, J., Zhang, C., Goiri, Í., Torrellas, J., and Choukse, E. Dynamollm: Designing llm inference clusters for performance and energy efficiency. *arXiv preprint arXiv:2408.00741* (2024).

[44] Sun, B., Huang, Z., Zhao, H., Xiao, W., Zhang, X., Li, Y., and Lin, W. Llumnix: Dynamic scheduling for large language model serving. *arXiv preprint arXiv:2406.03243* (2024).

[45] Team, G., Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* (2023).

[46] Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118* (2024).

[47] Wang, C., Shi, X., Xu, S., Wang, Z., Fan, Z., Feng, Y., You, A., and Chen, Y. A multi-stage framework for online bonus allocation based on constrained user intent detection. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (2023), pp. 5028–5038.

[48] Wei, Y., Wang, Z., Liu, J., Ding, Y., and Zhang, L. Magicoder: Source code is all you need. *arXiv preprint arXiv:2312.02120* (2023).

[49] Xu, W., Liang, Z., Mei, K., Gao, H., Tan, J., and Zhang, Y. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110* (2025).

[50] Xu, W., Shi, Y., Liang, Z., Ning, X., Mei, K., Wang, K., Zhu, X., Xu, M., and Zhang, Y. iagent: Llm agent as a shield between user and recommender systems. *arXiv preprint arXiv:2502.14662* (2025).

[51] Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115* (2024).

[52] Yu, P., Trombetta, P., Hassani, A., Bulitko, V., and White, M. Gilbo: One metric to measure them all. In *Advances in Neural Information Processing Systems* (2022), vol. 35, pp. 10285–10297.

[53] Zhang, C., Liu, X., Zhang, Z., Jin, M., Li, L., Wang, Z., Hua, W., Shu, D., Zhu, S., Jin, X., et al. When ai meets finance (stockagent): Large language model-based stock trading in simulated real-world environments. *arXiv preprint arXiv:2407.18957* (2024).

[54] Zhang, X., Qi, F., Hua, Z., and Yang, S. Solving billion-scale knapsack problems. In *Proceedings of The Web Conference 2020* (2020), pp. 3105–3111.

[55] Zhang, Y., Shi, X., Feng, L., Wang, M., Yu, Y., Yu, X., Shen, H., Chen, Z., Mucci, P., Kudlur, M., et al. S-lora: Serving thousands of concurrent lora adapters. In *Advances in Neural Information Processing Systems* (2023), vol. 36.

[56] Zhang, Y.-K., Zhan, D.-C., and Ye, H.-J. Capability instruction tuning: A new paradigm for dynamic llm routing. *arXiv preprint arXiv:2502.17282* (2025).

[57] Zheng, L., Yin, L., Xie, Z., Huang, J., Sun, C., Yu, C., Cao, S., Kozyrakis, C., Stoica, I., Gonzalez, J. E., et al. Efficiently programming large language models using sglang.

[58] Zheng, Z., Ren, X., Xue, F., Luo, Y., Jiang, X., and You, Y. Response length perception and sequence scheduling: An llm-empowered llm inference pipeline. vol. 36.

[59] Zhu, Q., Guo, D., Shao, Z., Yang, D., Wang, P., Xu, R., Wu, Y., Li, Y., Gao, H., Ma, S., et al. Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence. *arXiv preprint arXiv:2406.11931* (2024).

[60] Zhuang, R., Wu, T., Wen, Z., Li, A., Jiao, J., and Ramchandran, K. Embedllm: Learning compact representations of large language models. *arXiv preprint arXiv:2410.02223* (2024).

Introduction to The Special Section on Safe AI

Mykola Pechenizkiy
Department of Computer Science
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven
m.pechenizkiy@tue.nl

Stiven S. Dias
Embraer S.A.
R&D Team
Av. Brigadeiro Faria Lima, 2170
São José dos Campos, Brazil
stiven.dias@embraer.com.br

ABSTRACT

Alignment with human values and compliance with ethical and regulatory standards has become a major concern as cutting-edge generative AI-based solutions are becoming ubiquitous and have been increasingly adopted as a productivity tool by many institutions. Despite their remarkable results across several fields, there are still objections regarding embedding AI models, even the vanilla ones, into safe-critical systems, specially, across healthcare, aeronautical and nuclear industries, as many AI methods lack explainability and robustness, and are still prone to adversarial and cyber attacks. The Safe AI field addresses these issues through the proposal of design and development procedures aimed at learning assurance and model alignment, the investigation of interpretable methods with explainable outputs, and the proper treatment of uncertainty. We summarize first the outcomes of a recently launched workshop on Safe AI and present then five selected papers in this special section representing a crosscut of different application areas for AI safety.

1. INTRODUCTION

The advent of generative AI has unlocked many applications, promoting more convenience in everyday tasks and leveraging productivity in labor activities. Conversely, it is widely recognized that generative models may hallucinate and manifest unfairness induced by, e.g., unwanted biases present in the training data. Ultimately, they may fail to the point they fully lack alignment with human values and fundamental ethical principles.

Deep models in general have also gained much attention in the industry, as they are able to perform many challenging tasks with human-like and even super-human performance. However, the use of AI models in safety-critical applications are still controversial, since it is hard to audit them and demonstrate they are trustworthy, behaving consistently and reliably as expected by their stakeholders.

There are multiple initiatives towards AI safety. It is notable, that over the past few years, an international network of AI Safety Institutes has emerged. Starting with UK AI Safety Summit in November 2023, where the concept of AI Safety Institute was introduced through the Bletchley Declaration, the network started with the AI Safety Institute (AISI)[1] (later renamed into AI Security Institute) it grew to include the North American Center for AI Standards and Innovation (CAISI)[2], the European Union AI Office[3] and national, e.g., Spanish Agency for the Supervision of Artificial Intelligence[4] institutes. As of today, the network includes variants of AISI in India, Singapore, South Korea, Japan, and Israel. More countries announce similarly intended initiatives.

Government organizations aim to bridge the knowledge gap between rapidly evolving AI technology and public sector understanding by providing technical expertise to inform policy decisions. Thus, the CAISI in the US focuses on working with NIST organizations to develop guidelines and best practices to measure and improve the security of AI systems, and to develop voluntary standards. The EU AI Office supports development of AI safety standards and Codes of Practice, particularly in support of the EU AI Act. Besides government-backed AI safety institutes, independent non-profits – e.g., Center for AI Safety (CAIS)[5] –, and international collaborations – notably, the International AI Safety Report [6] – have emerged. Many leading AI companies – e.g., Anthropic, Google AI / Google DeepMind, OpenAI and Meta AI – have their own internal AI safety teams and initiatives.

Outline. We discuss the landscape of Safe AI applications and taxonomy in Section 2. Section 3 highlights a recently launched initiative on AI safety, the 1^{st} Workshop on Safe AI at UAI 2025 conference. In the sequel, in Section 4, we introduce five selected articles to represent the state-of-the-art in the field. Finally, we conclude the introduction in Section 5.

2. SAFE AI LANDSCAPE

Safe AI is the field focused on minimizing the risks associated with the operation of future AI systems. A comprehensive risk assessment, in turn, raises different concerns spanning from engineering and security aspects through ethical and governance considerations. In any case, the ultimate goal is to ensure that increasingly capable AI systems operate reliably, aligned with human values, and do not cause harm to the society if they unintentionally fail or in case they are deliberately misused.

[1] https://www.aisi.gov.uk/

[2] https://www.nist.gov/caisi

[3] https://digital-strategy.ec.europa.eu/en/policies/ai-office

[4] https://aesia.digital.gob.es/en/es

[5] https://safe.ai/

Figure 1 highlights the main AI safety principles that future-proof AI systems should follow. Alignment ensures that AI systems' outcomes contribute positively to human goals, avoiding unintended harm to society. Robustness ensures that AI systems perform consistently as designed in face of unforeseen or potentially harmful inputs. Transparency ensures that AI systems are understandable and auditable by humans, which is crucial for building trust and accountability. Accountability ensures that there are mechanisms to assign responsibility for the outcomes of AI systems.

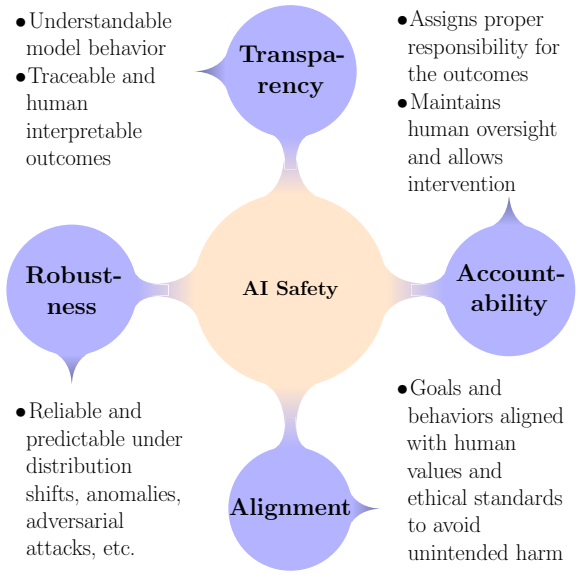


Figure 1: AI safety principles that future-proof AI systems should follow.

Much of progress in modern Machine Learning (ML) is attributed to testing multiple ideas and favoring most competitive through benchmarking. In the context of Safe AI, technical and normative concept and requirements are often translated into oversimplified intrinsic metrics. Thus, current fairness-aware machine learning benchmarking practices may result in suboptimal outcomes, undesirable side-effects, or misleading conclusions. Complementary evaluation frameworks [30,1] are advocated to address such limitations. Lastly, it is worth mentioning that verification of AI systems safety through formal system analysis has received increasing attention, see e.g. [27]. However, the practical relevance of the results obtained so far is not yet fully evident.
Major research areas within AI safety include:

- **Robustness and reliability** to ensure that AI systems will generalize satisfactorily, behaving dependably and consistently over time, given arbitrary inputs, including out of distribution (OOD) inputs, anomalies, adversarial attacks, distribution shifts, and concept drifts [32]. Controlling the accuracy and uncertainty trade-off can be useful in decision making [15]. Decidability and halting guarantees are also a concern [29].
- **Explainability and interpretability** to understand AI system (mis)behavior and its underlying working mechanisms, ensuring that the (causal) reasoning behind specific inferences are understandable to human beings [33].
- **Ethical alignment** to make sure that AI systems' goals and behavior are aligned with human values and intentions. Concerns range from non-discrimination [22] and fairness-awareness [10] to the raise of rogue AI agents [9].
- **Preventing misuse** is considered from the perspective of hypothetical superintelligence [2] to guidelines for improving the safety, security, and trustworthiness of dual-use existing and future foundation models [37] to prevent AI systems based on it from being weaponized to harm society.
- **Governance and assurance** are applied to all stages of the AI engineering lifecycle: developing, testing, verification & validation, deployment, and runtime monitoring. Governance frameworks encompass regulations, methods, procedures, and technological mechanisms to ensure that an organization's development and deployment of AI technologies align with its strategies, principles, and goals [28]. On the other hand, assurance processes ensure that AI technologies produce outcomes that are valid, verified, data-driven, trustworthy and explainable to a layman, yielding intelligent systems that are ethical in the context of their deployment, unbiased in their learning, and fair to their users [5].

The AI alignment and AI safety fields are inter-twisted. As argued by Ji et al. in [21], the former constitutes a significant portion of the later concerns. They share common AI risks from model misalignment and, as indicated in Figure 2, they encompass thus common goals / principles: Robustness, Interpretability, Controllability, and Ethicality (RICE). Much attention has been directed to the alignment of Large Language Models (LLMs) [34] and their vulnerability to jailbreaking [11]. However, in general, the so-called AI alignment cycle – comprising backward and forward alignment procedures – also serves as a framework to ensure that a broad spectrum of AI systems adhere to human intentions and values [24].
Explainable AI (XAI) has been a topic of extensive research for Deep Neural Networks (DNN) [38,20] and, more narrowly, for Natural Language Processing (NLP) [12]. The XAI community has been striving to explain models as black-boxes [13], driving less effort though toward designing white-box models. Despite yielding explainable outputs, linked to the model inputs by human interpretable causal chains, white-box or glass-box models are often not as powerful and fail to achieve state-of-the-art performance when compared to black-box ones [17]. On the other hand, as argued by Rudin in [31], instead of perpetuating the bad practice of explaining black-box models that can potentially harm the society, the path forward for Safe AI should be the development of inherently interpretable models. Finally, application-wise, it can be argued that the XAI goal is to ensure that the generation of the AI system's outputs is auditable [7] and, ultimately, liable [42].
AI governance and assurance [5] have been gaining increasing attention from the industry, as strictly regulated business activities require auditable development and deployment processes. In this sense, regulatory efforts like the European Union Aviation Safety Agency (EASA)'s AI roadmap [18,19], guiding the safe and trustworthy development and implementation of AI technologies in aviation, are crucial to promote

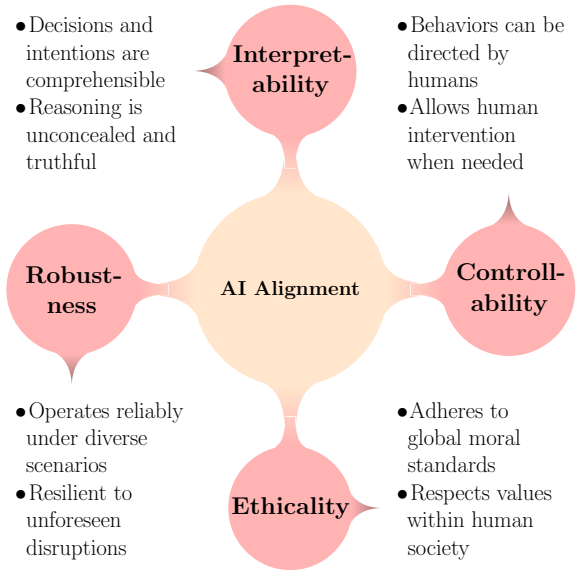


Figure 2: RICE principles that an aligned AI system should possess as introduced by Ji et al. in [21].

trustworthy AI in the industry. However, it is paramount that these recommendations and guidelines are also addressed by foundational research within the AI safety community.
Finally, the AI trustworthiness and AI safety domains overlap in many ways. However, the former is focused on ensuring that currently available AI systems are ethical, fair, transparent and reliable for today's society, whereas the later follows a risk-centric approach focused on preventing harmful, unintended, or catastrophic outcomes from increasingly capable models, pursuing thus longer timescales and an expanded risk framing. As stressed by Li et al. in their survey [23], AI trustworthiness must be systematically pursued throughout all life-cycle of an AI-based system. Figure 3 depicts the different aspects of trustworthy AI ranging from technical to ethical requirements. Li et al. also argued that an holistic assessment of these aspects is required to ensure the accountability of real-world AI systems, making sure they comply with all requirements and regulations. It is worth noting that performance (accuracy) requirements are important ingredients and trades-off with several requirements, but, from the authors' perspective, ethical requirements play a central role on the system life-cycle. In particular, they put accountability goals / aspects at the heart of the holistic assessment, raising thus the relevance of auditability, traceability and responsibility requirements.

3. 1ST WORKSHOP ON SAFE AI@UAI 2025

In this section, we summarize the 1^{st} Workshop on Safe AI held on July 25^{th}, as part of the 41^{st} Conference on Uncertainty in Artificial Intelligence (UAI 2025) in Rio de Janeiro, Brazil.
The workshop was conceived to foster technical discussions on cutting-edge safety techniques, including uncertainty quantification, adversarial robustness, and socio-technical discussions including AI alignment with human values, fairness and non-crimination among other important topics. Broadly, the workshop also aimed at discussing techniques that facilitate design and development of the future-proof AI.
The workshop's program included two plenary keynote talks, thirteen technical contributions selected by the program committee and presented by authors in the oral and poster sessions, and a closing discussion[6]. The keynote speakers covered topics related to AI uncertainty and AI governance as highlighted below:

- **Privacy-aware Bayesian Networks**. Professor Cassio de Campos introduced credal models as a practical solution for balancing privacy and utility. Credal models represent a set of standard precise models by having set-based parameter specifications. Then, he discussed how credal models can disguise the original model, thereby reducing the probability of successful attacks, while achieving meaningful inferential results. The versatility of the idea was illustrated through an experimental evaluation, comparing it with noise-based approaches. Lastly, the talk also stressed that proper treatment of model uncertainty is required for safe AI.
- **AI Security - Standards, STEADY and LASR**. In this talk, Professor Paul Miller provided an overview of the European Telecommunications Standards Institute (ETSI) approach to developing standards for AI security through security-by-design. Then, he discussed how him and his collaborators have been applying an AI safety and assurance methodology to a specific use-case related to autonomous ground vehicles. To summarize, the talk highlighted the need for AI governance frameworks to ensure that AI-based systems meet their purpose and specifications.

The technical papers presented at the workshop covered four thematic areas including different areas of XAI and uncertainty, AI fairness and alignment with human goals, AI robustness to adversarial attacks, and AI robustness in scenarios with AI agents.

XAI and uncertainty:

- **DT-sampler: A SAT-based Decision Tree Ensemble** by *Xiaotian Xue, Chao Huang, Koji Tsuda, Diptesh Das*, published in this special section [39].
- **SpaCE-VAE: Sparse and Confident Explanations using Variational Autoencoders** by *Alexander Liu, Sibylle Hess*, published in this special section [25].
- **Explaining Deep Learning Matching of Hand-Drawn Binary Symbols: A Visual Analysis with Grad-CAM on Cattle Brands** by *Leandra Soares, Marcos Medeiros, Aldo Díaz-Salazar, Edmundo Hoyle*, published in this special section [35].
- **A Non-Parametric Bayesian Approach Towards Online Sequence Learning** by *Stiven Schwanz Dias, Marcelo Gomes da Silva Bruno, Alberto Ferreira De Souza, Thiago Oliveira-Santos and Claudine Badue*, also published in this special section [16].

AI fairness and alignment with human goals:

[6]The full program and preprints of the papers not included in this special section can be found at `https://sites.google.com/view/safeai2025/`

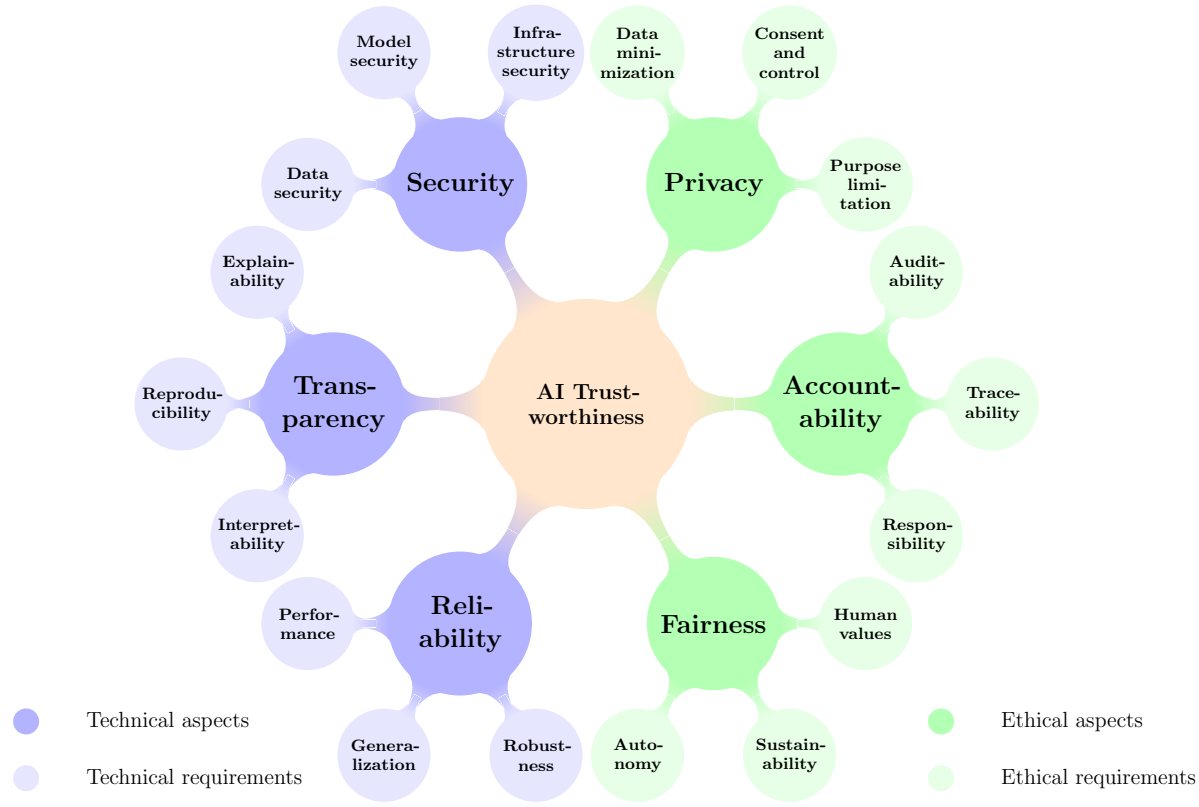


Figure 3: Non-exhaustive breakdown of AI trustworthiness aspects into requirements that existing AI systems should meet.

- **Robust Data Merging Strategies for Aerial Object Detection of Flying Objects** by *Luca Plaster, Aldo Díaz-Salazar*, the manuscript available on the workshop website as a full paper.
- **Alignment Metrics for the Evaluation of Observed Autonomous Systems** by *Tommaso Mannucci, Wouter Arink, Johan van den Heuvel*, also available on the workshop website as an extended abstract.

AI robustness to adversarial attacks:

- **ExpProof: Operationalizing Explanations for Confidential Models with ZKPs** by *Chhavi Yadav, Evan Laufer, Dan Boneh, Kamalika Chaudhuri*, available as a preprint on arXiv [40].
- **Influence Attributions can be Systematically Altered by Model Manipulation** by *Chhavi Yadav*, also available as a preprint on arXiv [41].
- **Riemannian Manifold Learning for Stackelberg Games with Neural Flow Representations** by *Larkin Liu, Kashif Rasul, Yutong Chao, Jalal Etesami*, an extended abstract available on the workshop website, and a preprint of the full paper on arXiv [26].

AI robustness in scenarios with AI agents:

- **An Analysis of Robustness of Non-Lipschitz Networks** by *Maria-Florina Balcan, Avrim Blum, Dravyansh Sharma, Hongyang Zhang*, published in the Journal of Machine Learning Research, volume 24, in 2023 [4].
- **The Value of Recall in Extensive-Form Games** by *Ratip Emin Berker, Emanuel Tewolde, Ioannis Anagnostides, Tuomas Sandholm, Vincent Conitzer*, published in the proceedings of the 39^{th} Conference on Artificial Intelligence (AAAI-2025) [8].
- **HASARD: A Benchmark for Vision-Based Safe Reinforcement Learning in Embodied Agents** by *Tristan Tomilin, Meng Fang, Mykola Pechenizkiy*, published in the proceedings of the 13^th International Conference on Learning Representations (ICLR 2025) [36].
- **Decision Making under Imperfect Recall: Algorithms and Benchmarks** by *Emanuel Tewolde, Brian Zhang, Ioannis Anagnostides, Tuomas Sandholm, Vincent Conitzer*, preprint available on the workshop website.
- **SafeFlowNet: Safe Control with Generative Flow Networks** by *Yucheng Yang, Tianyi Zhou, Meng Fang, Mykola Pechenizkiy*, preprint available on the workshop website.

The workshop promoted an opportunity to i) leverage the AI safety community, ii) promote the field among young researchers, iii) prospect which topics are being most investigated by the community and iv) which topics should be further explored by it. From our perspective, explainability and resilience to adversarial attacks are being the focus of many research efforts. On the other hand, as suggested by the position paper by Dr. Miller, there is room for the community to further investigate the AI governance ecosystem,

since heavily regulated industries such as the aeronautical one must demonstrate to certification authorities that complex systems, e.g., an aircraft, comply with strict norms and standards throughout the entire product lifespan.
Lastly, the workshop included full papers and extended abstracts. Some of the full papers were selected and polished to this special section as described in the following compendium.

4. CONTRIBUTED ARTICLES

This special section includes one position paper related to AI governance and assurance, based on the invited workshop keynote, along with four full papers presenting technical contributions:

Assuring the Case: A Safety Engineering Approach to AI-Enabled Systems [14] introduces an assurance framework for ML-based safety-critical systems. Specifically, the authors follow a safety engineering approach and employ a graphical, six-stage process for assurance case construction, culminating in the instantiation of the assurance argument. They demonstrate the assurance process considering a deep CNN image classifier for an autonomous vehicle application.

DT-sampler: A SAT-based Decision Tree Ensemble [39] considers the problem of creating a decision tree (DT) ensemble taking into account the size and accuracy of the sampled trees. The authors combine a SAT-based decision tree sampler to generate small-size DTs with the conformal prediction framework to post-calibrate the sampled trees. Their approach unlocks the path for more-explainable decision trees. Lastly, they demonstrate that their approach is more statistically efficient and provides stable predictions compared to random forests across several real-world benchmark datasets.

SpaCE-VAE: Sparse and Confident Explanations using Variational Autoencoders [25] introduces an extension of the Vector Quantized Variational Autoencoder (VQ-VAE) to generate sparse explanations for image classification with DNNs, the SpaCE-VAE. The authors evaluated the proposed method against three other approaches that identify pixels that contribute negatively to the predicted class: DeepLIFT, Integrated Gradients, and Feature Ablations. Quantitative assessment using several examples with explanations was favorable to SpaCE-VAE, whereas qualitative (visual) inspection of the results indicates that SpaCE-VAE can provide insightful explanations.

Explaining Deep Learning Matching of Hand-Drawn Binary Symbols: A Visual Analysis with Grad-CAM on Cattle Brands [35] presents a case study in explainable AI, where Grad-CAM is used to understand the classification of a particular family of images. The authors investigate the application of Grad-CAM to analyze VGG-16 activations on binary cattle brand images. Careful experimental assessment exposed expected CNN limitations: rotation sensitivity and fine-grained feature under-activation.

A Non-Parametric Bayesian Approach Towards Online Sequence Learning [16] introduces a grid-based filter which is intrinsically designed to solve the online sequence learning problem and deal with uncertainty in time-series observations. The authors derive interpretable Bayesian procedures to both recursively predict observations and incrementally build a non-parametric, probabilistic model of the underlying temporal phenomenon. Moreover, they propose an one-shot learning, memory-based implementation of the filter which allows one to fully trace back inferences against associative input-output pairs stored within a weightless neural network, thus unlocking the potential for auditable (explainable) inferences. Lastly, they explore the potential of their filter using toy anomaly detection tasks. Despite presenting still limited results, the paper introduces a promising approach to the core of AI safety: accountability through the design of auditable methods with traceable outputs.

5. CONCLUDING REMARKS

We hope you will enjoy reading the papers on AI safety included in this special section.
In the introduction to this special section, we presented an overview of some of the developments in this fast evolving field. Despite being a relatively new field of research, the AI Safety community has been reaching a critical mass, with dedicated AI Safety Institutes [3] and a variety of academic events from technical governance of AI workshop series[7], aiming to provide analyses and tools to guide policy decisions and enhance policy implementation, to workshop on safety of generative AI[8] and safety-guaranteed LLMs[9], aiming to advancing technical and socio-technical research on critical topics. Nevertheless, we think that the core topics of knowledge representation, learning, and reasoning in the presence of *uncertainty* deserve drawing more attention from the UAI and sister communities, and we expect that these topics will become even more vivid in the future landscape of Safe AI research.

Acknowledgments

We thank all the authors who contributed to the 1^{st} Workshop on Safe AI at the UAI 2025 conference and to this special section, the program committee for helping review the manuscripts and shape the technical program, and all the participants for their contributions.

6. REFERENCES

[1] A. Ahmed, K. Klyman, Y. Zeng, S. Koyejo, and P. Liang. Speceval: Evaluating model adherence to behavior specifications. *arXiv preprint arXiv:2509.02464*, 2025.

[2] M. Alfonseca, M. Cebrian, A. Fernandez Anta, L. Coviello, A. Abeliuk, and I. Rahwan. Superintelligence cannot be contained: Lessons from computability theory. *J. Artif. Int. Res.*, 70:65–76, May 2021.

[3] R. Araujo, K. Fort, and O. Guest. Understanding the first wave of AI safety institutes: Characteristics, functions, and challenges. *arXiv preprint arXiv:2410.09219*, 2024.

[4] M.-F. Balcan, A. Blum, D. Sharma, and H. Zhang. An analysis of robustness of non-Lipschitz networks. *Journal of Machine Learning Research*, 24(98):1–43, 2023.

[7] `https://www.taig-icml.com/`
[8] `https://safegenaiworkshop.github.io/`
[9] `https://simons.berkeley.edu/workshops/safety-guaranteed-llms`

[5] F. A. Batarseh, L. Freeman, and C.-H. Huang. A survey on artificial intelligence assurance. *Journal of Big Data*, 8(1):60, 2021.

[6] Y. Bengio, S. Clare, C. Prunkl, M. Andriushchenko, B. Bucknall, P. Fox, N. Maslej, C. McGlynn, M. Murray, S. Rismani, et al. International ai safety report 2025: Second key update: Technical safeguards and risk management. *arXiv preprint arXiv:2511.19863*, 2025.

[7] C. Berghoff, B. Biggio, E. Brummel, V. Danos, T. Doms, H. Ehrich, T. Gantevoort, B. Hammer, J. Iden, S. Jacob, et al. Towards auditable AI systems. *Whitepaper. Bonn Berlin: Bundesamt für Sicherheit in der Informationstechnik, Fraunhofer-Institut für Nachrichtentechnik und Verband der TÜV eV*, 2021.

[8] R. E. Berker, E. Tewolde, I. Anagnostides, T. Sandholm, and V. Conitzer. The value of recall in extensive-form games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 13631–13640, 2025.

[9] N. Bostrom. *Superintelligence: Paths, dangers, strategies*. Oxford University Press, Oxford, UK, 2014.

[10] S. Caton and C. Haas. Fairness in machine learning: A survey. *ACM Comput. Surv.*, 56(7), Apr. 2024.

[11] P. Chao, E. Debenedetti, A. Robey, M. Andriushchenko, F. Croce, V. Sehwag, E. Dobriban, N. Flammarion, G. J. Pappas, F. Tramèr, H. Hassani, and E. Wong. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

[12] M. Danilevsky, K. Qian, R. Aharonov, Y. Katsis, B. Kawas, and P. Sen. A survey of the state of explainable AI for natural language processing. *arXiv preprint arXiv:2010.00711*, 2020.

[13] A. Das and P. Rad. Opportunities and challenges in explainable artificial intelligence (XAI): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

[14] S. Davidson, O. Igene, and P. Miller. Assuring the case: A safety engineering approach to ai-enabled systems. *SIGKDD Expl.*, 27(2), 2025.

[15] Y. Deng, A. D. Bucchianico, and M. Pechenizkiy. Controlling the accuracy and uncertainty trade-off in rul prediction with a surrogate wiener propagation model. *Reliability Engineering System Safety*, 196:106727, 2020.

[16] S. S. Dias, M. G. da Silva Bruno, A. F. D. Souza, T. ago Oliveira-Santos, and C. Badue. A non-parametric bayesian approach towards on- line sequence learning. *SIGKDD Expl.*, 27(2), 2025.

[17] W. Ding, M. Abdel-Basset, H. Hawash, and A. M. Ali. Explainability of artificial intelligence methods, applications and challenges: A comprehensive survey. *Information Sciences*, 615:238–292, 2022.

[18] European Union Aviation Safety Agency. EASA concept paper: First usable guidance for level 1 machine learning applications. *EASA AI Roadmap*, 2021.

[19] European Union Aviation Safety Agency. EASA concept paper: Guidance for level 1 & 2 machine-learning applications. *EASA AI Roadmap*, 2024.

[20] D. Gunning and D. Aha. DARPA's explainable artificial intelligence (XAI) program. *AI magazine*, 40(2):44–58, 2019.

[21] J. Ji, T. Qiu, B. Chen, J. Zhou, B. Zhang, D. Hong, H. Lou, K. Wang, Y. Duan, Z. He, L. Vierling, Z. Zhang, F. Zeng, J. Dai, X. Pan, H. Xu, A. O'Gara, K. Ng, B. Tse, J. Fu, S. Mcaleer, Y. Wang, M. Yang, Y. Liu, Y. Wang, S.-C. Zhu, Y. Guo, Y. Yang, and W. Gao. AI alignment: A contemporary survey. *ACM Comput. Surv.*, 58(5), Nov. 2025.

[22] F. Kamiran, T. Calders, and M. Pechenizkiy. Techniques for discrimination-free predictive models. In B. Custers, T. Calders, B. W. Schermer, and T. Z. Zarsky, editors, *Discrimination and Privacy in the Information Society - Data Mining and Profiling in Large Databases*, pages 223–239. 2013.

[23] B. Li, P. Qi, B. Liu, S. Di, J. Liu, J. Pei, J. Yi, and B. Zhou. Trustworthy AI: From principles to practices. *ACM Computing Surveys*, 55(9):1–46, 2023.

[24] X. Li, Q. Jiang, L. Jiang, S. Zhang, and S. Hu. The landscape of AI alignment: A comprehensive review of theories and methods. *International Journal of Pattern Recognition and Artificial Intelligence*, 2025.

[25] A. Liu and S. Hess. SpaCE-VAE: Sparse confident explanations using variational autoencoders. *SIGKDD Expl.*, 27(2), 2025.

[26] L. Liu, K. Rasul, Y. Chao, and J. Etesami. Riemannian manifold learning for stackelberg games with neural flow representations. *arXiv preprint arXiv:2502.05498*, 2025.

[27] M. Liu, C.-H. Lu, and M. Kwiatkowska. Exact verification of graph neural networks with incremental constraint solving. *arXiv preprint arXiv:2508.09320*, 2025.

[28] M. Mäntymäki, M. Minkkinen, T. Birkstedt, and M. Viljanen. Putting AI ethics into practice: The hourglass model of organizational AI governance. *arXiv preprint arXiv:2206.00335*, 2023.

[29] G. A. Melo, M. R. O. A. Máximo, N. Y. Soma, and P. A. L. Castro. Machines that halt resolve the undecidability of artificial intelligence alignment. *Scientific Reports*, 15(1):15591, May 4 2025.

[30] M. Pechenizkiy, H. Weerts, C. de Campos, Y. Sasaki, and J. Stoyanovich. From benchmarking to understanding fairml. In *ECAI 2025*, pages 38–45. IOS Press, 2025.

[31] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

[32] M. Salehi, H. Mirzaei, D. Hendrycks, Y. Li, M. H. Rohban, and M. Sabokrou. A unified survey on anomaly, novelty, open-set, and out of-distribution detection: Solutions and future challenges. *Transactions on Machine Learning Research*, 2022.

[33] G. Schwalbe and B. Finzel. A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts. *Data Mining and Knowledge Discovery*, 38(5):3043–3101, 2024.

[34] T. Shen, R. Jin, Y. Huang, C. Liu, W. Dong, Z. Guo, X. Wu, Y. Liu, and D. Xiong. Large language model alignment: A survey. *arXiv preprint arXiv:2309.15025*, 2023.

[35] L. Soares, M. Medeiros, A. Dıaz-Salazar, and E. Hoyle. Explaining deep learning matching of hand-drawn binary symbols: A visual analysis with grad-cam on cattle brands. *SIGKDD Expl.*, 27(2), 2025.

[36] T. Tomilin, M. Fang, and M. Pechenizkiy. HASARD: A benchmark for vision-based safe reinforcement learning in embodied agents. In *The Thirteenth International Conference on Learning Representations*, 2025.

[37] U.S. AI Safety Institute. Nist AI 800-1: Second public draft — managing misuse risk for dual-use foundation models. Interagency or Internal Report NIST AI 800-1 (2nd Public Draft), National Institute of Standards and Technology (NIST), Gaithersburg, MD, January 2025. Second Public Draft.

[38] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J. Zhu. Explainable AI: A brief survey on history, research areas, approaches and challenges. In *CCF international conference on natural language processing and Chinese computing*, pages 563–574. Springer, 2019.

[39] X. Xue, C. Huang, K. Tsud, and D. Das. DT-sampler: A sat-based decision tree ensemble. *SIGKDD Expl.*, 27(2), 2025.

[40] C. Yadav, E. M. Laufer, D. Boneh, and K. Chaudhuri. Expproof: Operationalizing explanations for confidential models with zkps. *arXiv preprint arXiv:2502.03773*, 2025.

[41] C. Yadav, R. Wu, and K. Chaudhuri. Influence-based attributions can be manipulated. *arXiv preprint arXiv:2409.05208*, 2024.

[42] H. Zech. Liability for AI: public policy considerations. In *ERA forum*, volume 22, pages 147–158. Springer, 2021.

Assuring the Case: A Safety Engineering Approach to AI-Enabled Systems

Scot Davidson
Thales IAS
1 Alanbrooke Rd
Belfast BT6 9HB
United Kingdom
Scot.davidson@uk.thalesgroup.com

Oseghale Igene
CSIT
Queen's University Belfast
Queen's Titanic Quarter
Belfast BT3 9DT
United Kingdom
o.igene@qub.ac.uk

Paul Miller
CSIT
Queen's University Belfast
Queen's Titanic Quarter
Belfast BT3 9DT
United Kingdom
p.miller@qub.ac.uk

ABSTRACT

Given the increasing ubiquity of Machine Learning (ML), there is an urgent need for assurance frameworks, particularly in relation to ML-enabled safety critical systems. We begin by describing the process of assurance case construction for conventional safety-critical systems that has been developed by the safety engineering community. Goal Structured Notation (GSN), a graphical approach to assurance case construction, is then briefly described. In the sequel, we describe an existing ML assurance framework that employs GSN, Assurance of Machine Learning for use in Autonomous Systems (AMLAS), which is a six-stage assurance process. AMLAS is then applied to our use-case, namely Autonomous Vehicle (AV), with several of the AMLAS stages described in detail. In particular, the creation and testing activities of the ML assurance stage and the instantiation of the assurance argument are developed. Finally, we make the case for increased interaction between the ML and safety engineering community.

1. INTRODUCTION

The past decade has seen exponential growth in ML, particularly deep learning, due to advances in Neural Network (NN) architectures, computing hardware and the availability of huge amounts of data. This has led to the uptake of ML in many different sectors including life and health sciences, agri-food, finance, transport and defence and space. Of particular interest to this work is that of AV. ML models developed have allowed for the advancement of AVs, for example, when it comes to sense-understand and decide-act modules. It currently focuses on how ML models can be improved in terms of their efficiency and performance.

Many companies, ranging from startups to Small to Medium Enterprises (SMEs) to multinationals, working in these sectors have attempted to develop new products and services based on ML, giving rise to an ML-based innovation ecosystem where the level of development rigour, quality and performance can vary wildly. Of particular concern is the use of ML models in safety-critical systems such as AVs. Hence, there is a real need for an assurance framework for systems such as AV's containing ML models. Such a framework is required to provide trustworthiness in the ML models deployed by a system. When analyzing ML-enabled safety-critical systems, some challenges need to be considered. Firstly, due to the complexity and non-deterministic nature of these algorithms, applying traditional safety assessment techniques when determining component-level safety requirements is not adequate. Regulators and auditors lack the tools necessary to determine that ML will not violate intended safety requirements. Secondly, safety assurance arguments must be developed to confidently ensure the safety and performance of these ML-enabled systems. Existing International Safety Standard documentation, for example International Organization for Standardization (ISO) 26262 (Functional safety of Electrical and Electronic (E/E) systems in road vehicles), does not have guidelines associated with ML integration. This will be very important when developing a comprehensive safety case for ML-enabled services and specifically for AV systems.

The academic research community in AI safety typically focus on issues such as interpretability, transparency, bias, etc. This often involves the development of esoteric mathematical techniques addressing model characteristics that may have little relevance to safety. Furthermore, in all cases, they treat the model in isolation, without any system context.

Recently, several standards bodies have been developing standards/regulations for the secure development and use of Artificial Intelligence (AI). European Telecommunications Standards Institute (ETSI) recently published their Technical Specification and Implementation Guide on AI Security which has been adopted as a standard by the National Cyber Security Centre (NCSC) in the UK [14]. The ETSI specification is the first global standard that sets minimum security requirements throughout the AI life cycle for all stakeholders in the AI supply chain. More specifically, it will allow developers of AI systems to demonstrate to prospects and partners that they are adhering to a framework created by cross-disciplinary collaboration, with requirements that are both globally relevant and practically implementable.

In this paper, we discuss the challenges in assuring ML and making models trustworthy and present initial steps towards this for a specific use-case. We then describe several state-of-the-art assurance case construction tools and illustrate how one of these is used in a recently developed assurance framework for ML models. This is then applied to a computer vision-based use-case of AV navigation. Finally, we highlight the need for increased interaction between the ML and safety engineering communities.

2. ASSURANCE CASE CONSTRUCTION

Developing and certifying ML-enabled safety-critical systems requires applying evidence-based standards, process-based standards for interpretation, and increased levels of monitoring. The evidence-based approach requires developing a comprehensive and structured assurance case for such systems. An assurance case is a structured argument and reasoning that demonstrates how a system will meet its requirements. A system is thought to be assured if there is a clear argument (or assurance case) with convincing evidence that the system will work as intended for a given application in a defined environment [6; 8]. Assurance sits at the intersection of the designer and the user, providing assurance to both that the risks of the system have been identified, to what degree they have been mitigated, and under what circumstances the system can be operated. Inadequate requirements introduce evidentiary gaps. An assurance case pattern also documents a reusable argument structure, and evidence types can be instantiated to provide an assurance case instance that is specific [7].

An assurance case addressing safety is called a safety case. Safety cases are typically limited in scope to assurance that "a system is safe for a given application in a given operating environment" [5]. Security cases are used to gather material evidence in order to reason about the security posture of the system. It is to assure system operates securely, minimizing the risk of exploitation through weaknesses and vulnerabilities. Cybersecurity attacks can also have severe implications on safety. Safety-critical systems (Automotive, Traffic Services, Aviation, Railways, etc.) need to provide sufficient safety performance of systems, and current approaches require that manufacturers and operators employ a systematic process of proactively identifying risks and controlling them appropriately [13].

The core of the process of assurance case construction is defining a plan that will produce a system which is assured. Verification agents should be enabled to collect the best, comprehensive evidence possible in support of the argument and to reduce the system's operational risk to an acceptable level. For simple systems, the evidence is generally used to demonstrate that the system's requirements are met. Complex systems are often assured by establishing justified confidence in critical properties of interest such as dependability concerns around harm avoidance (for example, safety cases and security cases). The key terms for an assurance case are defined as follows:

- **Claim**: A high-level assertion or statement;
- **Argument**: Supports the claim and provides the link to the supporting evidence. It also allows the supporting justification to be broken up to aid visibility;
- **Evidence**: Facts and judgements that support the applicable argument(s).

Figure 1 illustrates the relationship between the safety engineer and the system engineer [15]. The system requirements feed into both the design model, including ML components, and the safety analysis. The latter leads to the identification of hazards and risks from which safety requirements are derived. These then feed into the design model. At the core of the process is the assurance case which controls every step of the process.

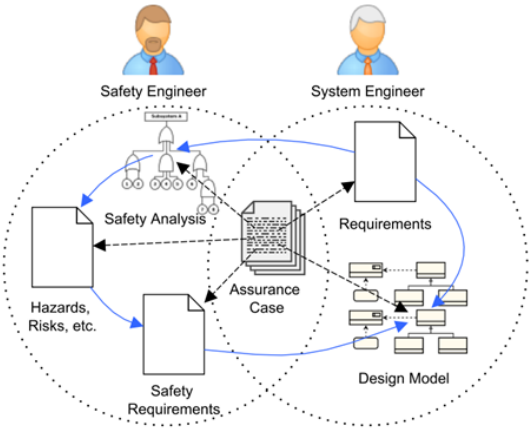


Figure 1: Illustrating the process of assurance case construction and the relationship between safety and system engineers [13].

Well-known notations can be utilised to develop assurance cases that are graphics-based, including GSN [2; 10; 3; 1], fault-tree analysis [4] and Claims Assurance and Evidence (CAE) [11]. These graphic-based approaches are the preferred option over text-based options because of the former's ability to simplify the construction and management of safety arguments and facilitate the presentation of these arguments to others. The GSN will be particularly useful as part of a broader methodology when developing a safety case for an ML-enabled system. GSN graphically provides links between arguments and evidence representing individual elements of any safety arguments and relationships between them in a top-down approach. Key elements of the GSN model include assurance goal or the claim; evidence that the goal is satisfied; and an argument linking evidence to the goal. These elements link together to form a goal structure. CAE is a structured approach which aids the visibility, review and assessment of safety case documentation.

3. METHOD

In this section we provide some background material on goal structured notation and introduce the Application of the AMLAS framework. The goal structure is hierarchical and is produced recursively with the overall goal for the system at the root. Other GSN elements include:

- **Strategies**: Decomposing a goal into additional goals;
- **Assumptions**: Taken without further argument or explanation;
- **Justifications**: Explaining why a solution provides sufficient evidence satisfying a goal;
- **Context**: Associated with another GSN element relevant to that element;
- **Solution**: Describes evidence that the stated goal has been met.

Figure 2 shows the various symbols used to represent the GSN elements along with a simple example. In the example shown above, claim C is justified by argument AS_1 based on

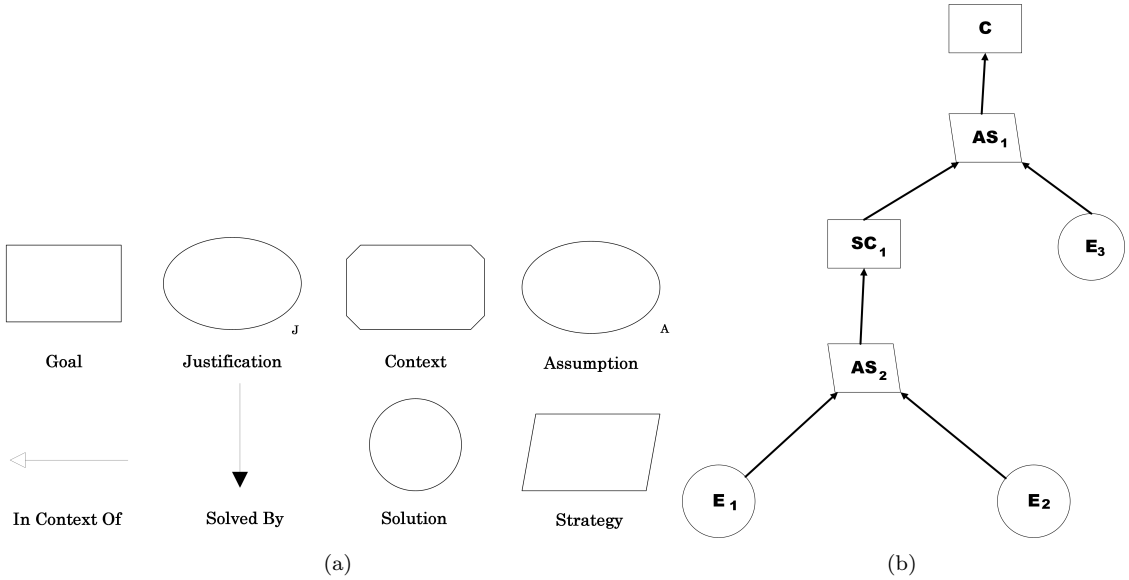


Figure 2: Symbols used to represent elements of goal-structured notation, (a), along with a simple example, (b).

evidence E_3 and subclaim SC_1, which itself is justified by argument step AS_2 based on evidences E_1 and E_2.

For the development of the safety case, the AMLAS methodology was chosen as it presents an iterative process involving machine learning-enabled components associated with our use-case [7]. As shown in Fig. 3, the AMLAS methodology provides six stages in which system safety requirements are first identified before narrowing to ML component safety requirements elicitation. Other stages include data management requirements associated with training and test datasets for the ML models, and formal validation and verification requirements.

Note that the process is iterative. At each stage, learnings from one stage can be fed back to previous stages to update it. To develop the safety case, the AMLAS approach will be applied to specific ML components, and each stage of the process is documented following each AMLAS step. Each of these is decomposed into a workflow of activities. In the next section, we will systematically go through some of the stages (due to space limitations) and illustrate what activities must be performed in the context of our use-case. For each stage and activity, inputs and outputs in the form of documentation have to be created. The output of each stage is an assurance argument pattern. Finally, at the completion of the process an assurance case is constructed for the ML model.

4. APPLICATION TO AUTONOMOUS VEHICLE USE-CASE

System

The application of interest is a navigation system as shown in Fig. 4 for an AV. The system for this is shown in Fig. 4. Inputs to the system are provided by RGB and infra-red cameras. C1-C5 and C7-C8 (red) are all ML components including: RGB image preprocessing for haze removal; image fusion; object detection and tracking; trajectory estimation; motion/pose analysis; and finally scene interpretation. As mentioned previously, assurance needs to be performed for all ML components, however, for the sake of brevity, we shall focus on C4 object detection.

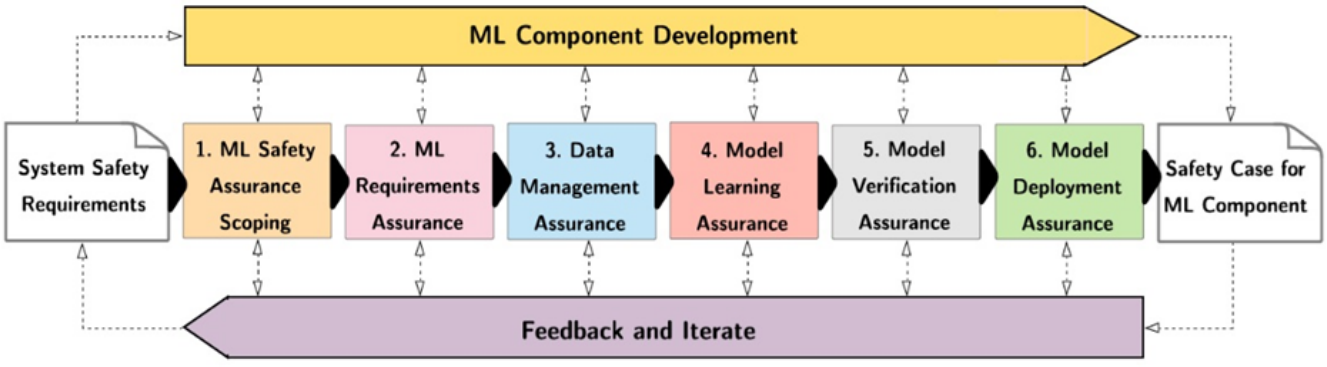


Figure 3: Overview of the AMLAS process [6].

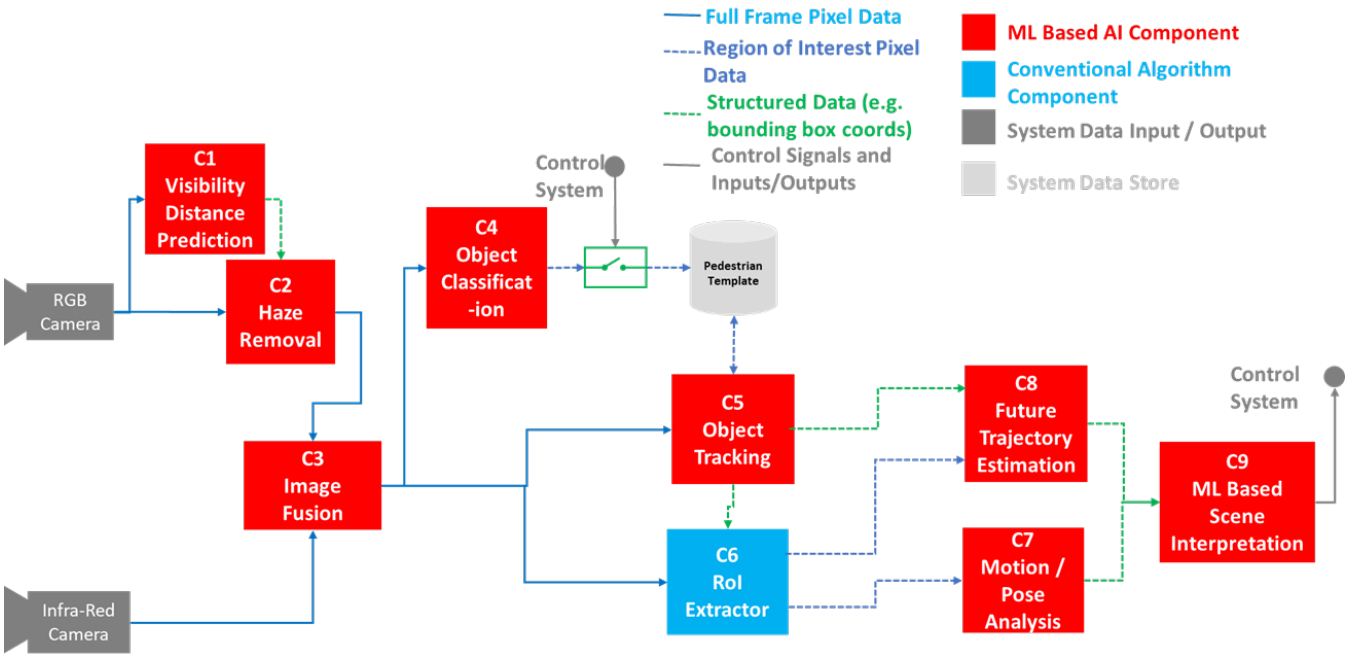


Figure 4: System architecture for navigation of the AV.

Environment

The system operates within the typical driving environment for human-driven vehicles, covering various road types including urban streets, highways, and suburban areas, under normal weather and lighting conditions. It is intended for non-industrial and non-commercial use, and must handle dynamic objects such as pedestrians, other vehicles, cyclists, and static infrastructure elements within the camera's field of view. Variability in traffic density, road layouts, and environmental factors such as rain, fog, or low light conditions influence system performance and safety requirements. The operational context also includes interaction with human users who may need to intervene or respond to system alerts, with the system designed to notify the user promptly in case of AI algorithm limitations or failures. These environmental characteristics and usage scenarios define the system's operational design domain and form a critical basis for tailoring safety assurance activities.

System Safety Requirements

When identifying system safety requirements, we first need to determine system-level hazards from which system safety requirements can then be developed. Applying risk assessment techniques such as HAZard and Operability Analysis (HAZOP) can be used to identify potential hazards that can be associated with safety-critical systems. An example potential hazard associated with autonomous vehicles that can arise depending on their operating environment is: Potential inaccuracies of "world models" (internal representation of the environment) generated by perception subsystems (Red Green Blue (RGB) camera) can lead to misinterpretation of data, compromising system safety [12].

In total, using the HAZOP methodology we identified five hazards. Based on the system architecture, Fig. 4, and potential system hazards identified, we derived six sets of safety requirements using requirements analysis [9].

ML Component Description - Object Detector (C4)

As mentioned previously, for the sake of illustration and brevity, we shall only consider ML component C4 Object Detector. This subsystem is responsible for detecting a single pedestrian within the vehicle's operational environment under all normal driving conditions, including varying lighting, weather, and traffic scenarios. The system utilizes a control module that maintains a pedestrian template to support recognition functions. This pedestrian detection subsystem operates as a component within a broader system-of-systems architecture typical of self-driving vehicles, which integrate numerous interdependent functional units such as perception, planning, control, and human-machine interfaces. Given this hierarchical structure, the subsystem's safety and performance requirements are derived not only from its internal function but also from its interaction and integration within the wider vehicle framework. The subsystem's functional scope, operational environment, and interfaces with other vehicle systems directly inform the safety assurance process, including hazard analysis, safety requirement allocation, and verification activities. Its role within the system of systems highlights the need for robust communication, interoperability, and adherence to safety standards to ensure reliable pedestrian detection as a critical element of the overall autonomous driving assurance case.

Fig. 5 shows the network architecture of YOLOV5 consisting of the backbone, PANet and outputs. The Object Detector (C4) takes a single image and classifies multiple objects (pedestrians) and outputs bounding boxes. This component implements an object detection algorithm that will be used to detect multiple persons who are crossing designated roadwalks. Furthermore, this ML component will handle and be exposed to both the In-Distribution and Out-of-Distribution

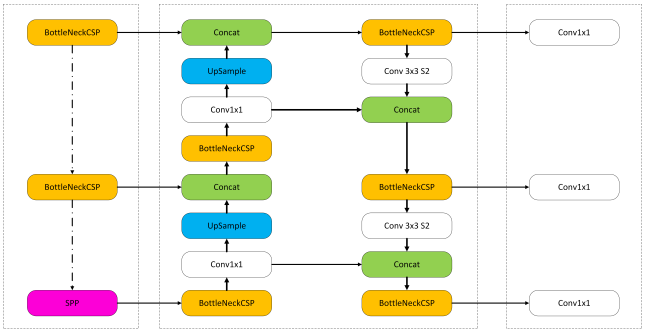


Figure 5: YOLOV5 network architecture (https://github.com/ultralytics/yolov5/issues/280).

(OOD) (Training and Test datasets) from environmental scenarios including weather conditions, background changes, object transformations and sensor anomalies. The component will receive live feed from the RGB Camera installed when in operation, which will also link with another ML component (C5).

4.1 Stage 1 - Machine Learning Assurance Scoping

Activity 1 - Defining the Safety Assurance Scope for the ML Component

For this activity, the system safety requirements associated with the C4 ML component need to be explicitly stated to build the safety assurance argument. Taking into account the defined system architecture and the operating environment, the system Safety Requirements (SRs) deemed within scope of C4 are:

SR4: "The system shall obey the highway code" which is further broken down into 6 requirements;

SR6: "The system shall be secure by design" which is further broken down into 2 requirements.

4.2 Stage 2 - ML Requirements Assurance

Activity 3 - ML Requirements Scoping

The scoping of Machine Learning Safety Requirements (MLSRs) involves deriving a dedicated set of requirements specific to machine learning components within a system. These requirements are critical because they relate directly to the component level rather than to the overall system. Establishing such requirements early in the development process ensures a coherent foundation, allowing subsequent engineering activities to be systematically traced back to the original design rationale. The proposed method employs traceability to illustrate the relationships between system requirements and the derived MLSRs. Moreover, these requirements serve as reference points for integrating documentation, reporting, and compliance evidence throughout the system lifecycle. In total, seventeen MLSRs have been defined, with two particularly notable requirements identified as:

MLSR6: "The ML component shall maintain data integrity throughout the development process" further broken down into 2 requirements;

MLSR7: "The ML system shall pass an in operation task, based on edge cases and normal operation" it is not further broken down.

Activity 4 - Validate ML Safety Requirements

Requirements are fundamental to understanding how a system and its components are expected to interact and perform. As the MLSRs represent high-level objectives, targeted validation activities are essential to confirm that these requirements are met in practice. These activities, denoted as VAL (Validation), are systematically mapped to one or more MLSRs. This structured mapping enables engineers to design and execute corresponding unit and system-level tests, while supporting safety and system engineers in validating the integrated system. In turn, this process underpins system acceptance and certification. Three notable VAL processes are defined as:

VAL6.1: Evaluate the model performance using adversarial attacks and provide a value for when the classification of objects defined in MLSR5 is obtained;

VAL6.2: Validate if adversarial training is sufficient for generating a robust model;

VAL 7: Assess the performance of the model and validate each of the given scenarios through simulations.

In total there are 13 validations ensuring that the MLSR's can be achieved. These are traceable to MLSR's using traceability analysis.

4.3 Stage 4 - Model Learning Assurance

In this section, the Activities 10-12 of the model learning assurance stage are briefly described. Fig. 6 shows the workflow.

Activity 10 - Creating the ML Model

Two MLSR safety requirements fed into Activity 10:

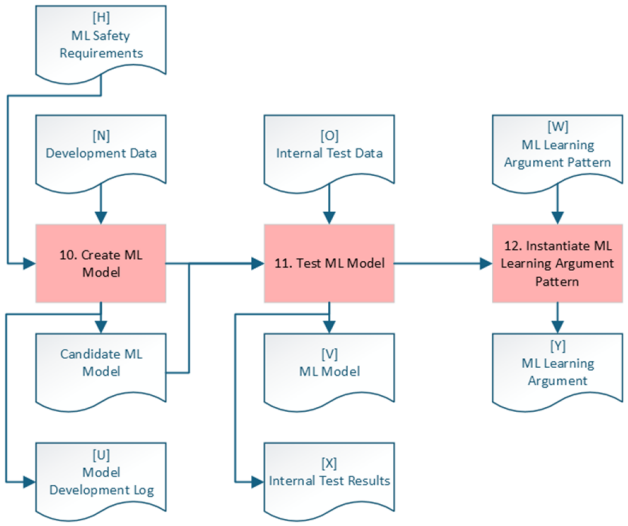


Figure 6: Activities associated with Stage 4 [6].

MLSR5: The system shall be able to recognise a person, car, sign, pedestrian crossing and all other objects defined in the highway code;

MLSR7: The ML system shall pass an in-operation task based on edge cases and normal operation.

Other inputs include the development data generated in Activity 7. MLSR7 enforces a real-time requirement which meant Vision Transformers were ruled out, whilst CNNs and a traditional ML recognition model were able to satisfy the requirement. MLSR5 further narrowed the model selection to the following off-the-shelf CNN architectures: YOLO, Faster-RCNN and SSD. From the literature it was discovered that only YOLO could satisfy MLSR7. The licensing arrangements for all nine versions of YOLO were then evaluated. YOLO5 was selected as its MIT license was deemed to be the least restrictive. The small and medium models were then put forward for adversarial training and possible deployment. The adversarial training involved fine-tuning the baseline YOLO model with adversarial samples containing patches. Part of the Model Based Systems Engineering (MBSE) approach means C++ conversion is a requirement for deployment of the neural network to the target hardware, Pytorch is converted using the libtorch package.

Activity 11 - Test of the ML model

Inputs to this include the internal test data developed under Activity 7. Four types of testing were then performed:

- **Test 1 – Baseline Test**: To establish the performance of the system without modifications such as augmentations and perturbations;
- **Test 2 - Environmental Test**: This refers to testing with augmented data that represents edge cases and environmental effects from the operational environment. Typically involves effects such as fog, rain and glare;
- **Test 3 - Adversarial**: This refers to testing the model utilising white-box and black-box attacks and adversarially trained patches. An example of the patch attack test is shown in Fig. 7.

These tests primarily involve measuring the attack success rate versus attack parameters such as patch size. This metric therefore provides quantitative evidence for the assurance process. The output is a document describing this evidence. This evidence is displayed in Fig. 8.

Activity 12 - Instantiation of the ML Argument Pattern

This is shown in Fig. 9. The Argument Strategy S4.1 is to argue over the internal testing of the model performed during development as well as the development approach adopted. The appropriateness of the development activities is considered within the context of creating a model that both satisfies the ML safety requirements as well as meeting the additional constraints that are imposed on the model, such as performance and cost. The model integrates YOLO for real-time object detection in an effort to detect all objects defined by the Requirement SR4.2 "The system shall maintain a safe distance from other objects as defined in the highway code". The development approach follows a structured lifecycle involving dataset curation; model training with augmented adversarial samples; adversarial robustness

Figure 7: Example of an adversarial patch placed on the target

evaluation; and validation under diverse driving scenarios. Internal testing emphasizes safety assurance by evaluating the model's resilience to adversarial and environmental variations.

These internal tests validate that the model maintains accurate perception under both nominal and adversarial conditions. The appropriateness of the development activities lies in their direct alignment with ML safety requirements Activity 5—robustness, interpretability, and reliability—while adhering to operational efficiency and hardware constraints relevant to automotive deployment. Cost is not assessed at this stage.

Goal 4.2 is that it must be demonstrated that the selected ML model satisfies the ML safety requirements when using internal testing data. This is demonstrated through a systematic traceability analysis linking each safety requirement to its corresponding verification activity, as defined in the use

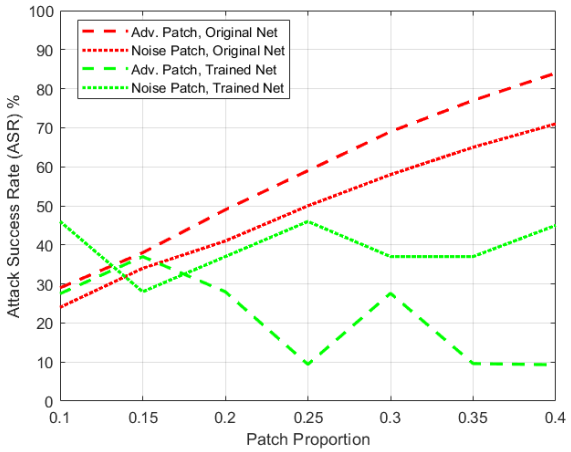


Figure 8: Attack Success Rate based on the patch proportion of the target. Tested against four neural networks red denotes the original net, and green for the adversarially trained neural network.

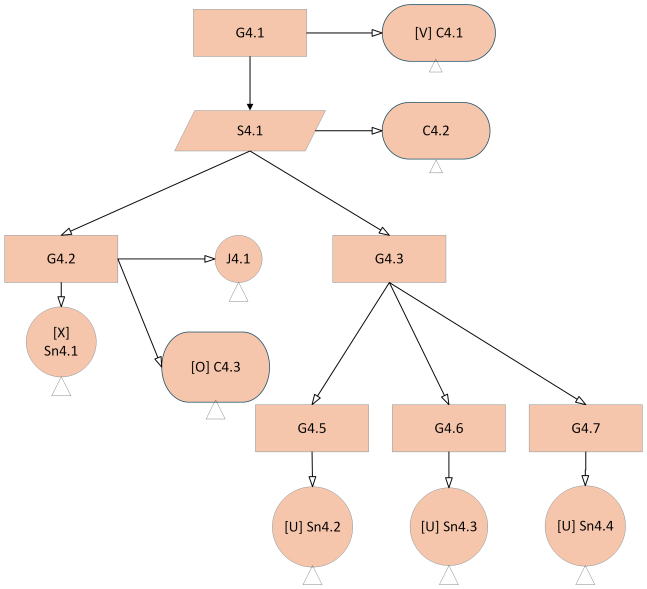


Figure 9: Assurance argument pattern given in for Stage 4 model learning assurance.

case illustrated in Fig. 4. This traceability ensures that all requirements are both testable and their specified verification strategies applied, linking the requirements of the original YOLO implementation with the robust YOLO model. The internal testing activities are coordinated within the AMLAS process, enabling stakeholders to observe and validate the relationships among diagrams, models, and associated code. The robustness of the YOLO architecture is assessed through controlled testing under both Operational Design Domain (ODD) and OOD conditions, ensuring the model maintains a reliable detection performance in realistic and adverse driving contexts. This integrated approach supports transparency and collective assurance across the creation and development lifecycle. The testing assess the requirements as well as the technical performances of the model. These technical performances found through the testing of the model are documented in the development log.

Evidence supporting the internal testing claim is documented comprehensively in the report generated from Activity 11, which details the test outcomes and validation processes. The sufficiency of these results in demonstrating compliance with ML safety requirements is further substantiated through the justification report (J4.1), providing a formal rationale that the internal testing methodology, environment, and outcomes collectively confirm that the ML model satisfies all relevant safety objectives.

Goal 4.3 addresses the development approach adopted for the creation of the ML model, supported by subordinate claims concerning model selection, parameterization, and the applied development process. For Goal G4.5, the model selection criteria are defined and justified in Activity 10, which establishes the linkage between the selected architecture and the system's functional and safety requirements. The choice of YOLO as the primary detection model is based on its proven capability for real-time object recognition and classification performance across a diverse range of categories, many of which correspond to objects and entities referenced within highway code standards.

Goal 4.6 incorporates standard retraining procedures designed to enhance robustness and generalization. This is performed during the training phase, model parameters are generated through iterative optimization to align detection accuracy and computational efficiency with safety and operational goals.

Goal 4.7 is that it must be demonstrated that the development process is appropriate. The development process has been designed to ensure methodological appropriateness and transparency throughout all iterative stages of model creation as part of Activity 10. As the approach employs a PyTorch based toolchain, it must convert to a C++ integration (MLSR 8); PyTorch is therefore selected over Tensorflow for its proven reliability, backward compatibility, and integration capabilities through the LibTorch package. The YOLO architecture is implemented using the established Ultralytics toolchain, which is widely documented and supported within the research community. This satisfies some of the testing that is required as part of Activity 11; although other tests still need to be performed for assurance. The retraining utilizes an open-source retraining script, facilitating transparent and reproducible model adaptation. The method aligns with established best practices in safety-critical ML development and supports continuous improvement through verifiable and explainable training updates. All of the required training and validation is found in the model development log. The justifications for each are documented in J4.1 and summarised in Fig. 9 and the following list for brevity.

C4.1: The ML model chosen is a detector tracker system utilising YOLO and Siamese tracker;

C4.2: Constraints are real-time compatibility, assured to be safe using verification, must run on GPU, Pytorch Implementation;

C4.3: The internal test data is described in O. This branches how the internal test data is linked to X Results;

G4.1: The development of the learnt model is sufficient;

G4.2: The selected model satisfies the ML safety requirements when using internal test data;

G4.3: The development approach adopted to create the model is sufficient;

G4.6: The model parameters are appropriate for meeting the defined ML safety requirements;

G4.7: The model development process is appropriate for meeting the defined ML safety requirements;

J4.1: The internal test results are representative of the components of the model;

S4.1: Argument over the sufficiency of the model development within the constraints of the target deployment;

Sn4.1: The internal test results are described in X. All of the internal tests are assumed to have passed in this scenario as failing requires redrafting the process until working;

Sn4.2: The model selection has been curated by utilising requirements analysis to ensure initial model selection meets the safety requirements;

Sn4.3: The off-the-shelf YOLO model has been appropriately trained for object classification. Fine-tuning is required to ensure the model is robust to adversarial attacks;

Sn4.4: The model development process is documented in [U] and all toolchains are assessed to comply with the MLSR's defined in Activity 3.

5. CONCLUSIONS

In this paper, we have described preliminary work on developing an assurance framework for image classifiers based on a deep learning model. We have modified an existing framework, AMLAS, and applied it to our use-case, namely AVs. AMLAS is a six-stage process, with each being decomposed into a series of activities that have inputs and outputs. Each stage completes with the development of an assurance argument pattern using GSN. At each stage, we have also described techniques that can be used to provide evidence for the various arguments; adversarial training, patch detection and mitigation.

Future work will involve investigating how we can use the evidence metrics to give a confidence score for each stage and then aggregating these using evidential reasoning networks. In addition, dynamic assurance cases are required for many systems, hence we propose to investigate this. Finally, the GSN approach used by AMLAS originated in the system safety engineering community. We feel that it is vital that the AI community reach out and collaborate with the former to achieve safety certification of system with AI models, rather than focusing on abstract theoretical concepts that have little relevance to safety.

6. ACKNOWLEDGEMENTS

This work is supported through the NICYBER2025 programme funded by Innovate UK. The project is a collaboration between CSIT and Thales. The views expressed are those of the authors and do not necessarily represent the project or the funding agency.

7. ADDITIONAL AUTHORS

Additional authors: Hamid Asgari (Thales RTSI, email: `Hamid.Asgari@uk.thalesgroup.com`), Chris Johnson (CSIT, email: `c.w.johnson@qub.ac.uk`), Nic Friedlander (Thales IAS, email: `nic.friedlander@uk.thalesgroup.com`.

8. REFERENCES

[1] Assurance Case Working Group (ACWG). Goal Structuring Notation (GSN) Community Standard, Version 3. Technical Report SCSC-141C, SCSC, May 2021. Accessed: 2024-10-22.

[2] A. Ayoub, B. Kim, I. Lee, and O. Sokolsky. A systematic approach to justifying sufficient confidence in software safety arguments. In *International Conference on Computer Safety, Reliability, and Security*, pages 305–316. Springer, 2012.

[3] H. Bourbouh, M. Farrell, A. Mavridou, and I. Sljivo. Integration and evaluation of the advocate, fret, cocosim, and event-b tools on the inspection rover case study. Technical report, 2020.

[4] K.-T. Chen, H.-Y. W. Chen, A. Bisantz, S. Shen, and E. Sahin. Where failures may occur in automated driving: a fault tree analysis approach. *Journal of cognitive engineering and decision making*, 17(2):147–165, 2023.

[5] DEF STAN 00-56. Safety management requirements for defence systems, part 1: Requirements. Technical report, Ministry of Defence, 2017. Retrieved October 24, 2024.

[6] E. Denney, G. Pai, I. Habli, T. Kelly, and J. Knight. 1 st international workshop on assurance cases for software-intensive systems (assure 2013). In *2013 35th International Conference on Software Engineering (ICSE)*, pages 1505–1506. IEEE, 2013.

[7] R. Hawkins, C. Paterson, C. Picardi, Y. Jia, R. Calinescu, and I. Habli. Guidance on the assurance of machine learning in autonomous systems (amlas). *arXiv preprint arXiv:2102.01564*, 2021.

[8] L. Jöckel, M. Kläs, J. Groß, P. Gerber, M. Scholz, J. Eberle, M. Teschner, D. Seifert, R. Hawkins, J. Molloy, et al. Operationalizing assurance cases for data scientists: A showcase of concepts and tooling in the context of test data quality for machine learning. In *International Conference on Product-Focused Software Process Improvement*, pages 151–158. Springer, 2023.

[9] G. Kotonya and I. Sommerville. *Requirements engineering: processes and techniques*. Wiley Publishing, 1998.

[10] NASA. Advocate user guide 1.4. Technical report, NASA Technical Reports Server (NTRS), May 2022. Accessed: 2024-10-22.

[11] S. Porziani, F. Scarpitta, E. Costa, E. Ferrante, B. Capacchione, M. Rochette, and M. E. Biancolini. Caeup-update of cae models on actual manufactured shapes. *Procedia Structural Integrity*, 24:775–787, 2019.

[12] L. Steckhan, W. Spiessl, N. Quetschlich, and K. Bengler. Beyond sae j3016: New design spaces for human-centered driving automation. In *International Conference on Human-Computer Interaction*, pages 416–434. Springer, 2022.

[13] M. A. Sujan, F. Koornneef, N. Chozos, S. Pozzi, and T. Kelly. Safety cases for medical devices and health information technology: involving health-care organisations in the assurance of safety. *Health informatics journal*, 19(3):165–182, 2013.

[14] TS 104 223. Securing artificial intelligence (sai); baseline cyber security requirements for ai models and systems. Technical report, ETSI, 2025. Retrieved October, 2025.

[15] R. Wei, S. Foster, H. Mei, F. Yan, R. Yang, I. Habli, C. O'Halloran, N. Tudor, T. Kelly, and Y. Nemouchi. Access: Assurance case centric engineering of safety–critical systems. *Journal of Systems and Software*, 213:112034, July 2024.

DT-sampler: A SAT-based Decision Tree Ensemble

Xiaotian Xue[1] Chao Huang[1*] Koji Tsuda[1,2,3†] Diptesh Das[1‡]

[1] Department of Computational Biology and Medical Sciences, The University of Tokyo, Japan
[2] RIKEN Center for Advanced Intelligence Project, Japan
[3] Center for Basic Research on Materials, National Institute for Materials Science, Japan

ABSTRACT

Interpretable (or explainable) machine learning models, such as decision trees, play a crucial role in the context of trustworthy AI. However, finding optimal decision trees (i.e., minimum size and maximum accuracy trees) is not a simple task and remains an active area of research. While a single decision tree has limited expressivity, using an ensemble of decision trees can effectively capture the complex structures found in many real-world applications. Many existing tree ensemble methods are greedy and suboptimal, and often suffer from randomness in the tree generation process. In this paper, we introduce DT-sampler, a SAT-based decision tree ensemble which allows explicit control over both the size and accuracy of the sampled trees. We developed a novel SAT-based encoding method that utilizes only branch nodes, resulting in a compact representation of decision tree space. Additionally, standard point predictions made using decision tree ensembles do not offer any statistical guarantee over miscoverage rate. We employ conformal prediction (CP), a distribution-free statistical framework which provides a valid finite-sample coverage guarantee, to demonstrate that DT-sampler is statistically more efficient and produces stable results when compared with random forest classifier. We demonstrate the effectiveness of our method through several benchmark and real-world datasets.

1. INTRODUCTION

Interpretable machine learning (ML) models are paramount for their seamless integration in high-stake decision making problems e.g., medical diagnosis [1; 2; 3; 4; 5; 6; 7; 8], criminal justice [9; 10; 11]. In medical diagnosis, especially in computer-assisted diagnosis (CAD), model accuracy is important, but it is equally important for the doctor and the patient to know the features used in CAD modeling [12; 13]. There have been several established feature selection (FS) algorithms in ML literature, namely LASSO [14], marginal screening (MS) [15], orthogonal matching pursuit (OMP) [16], and decision tree (DT) based. Among them, DT-based FS has been widely studied due to its high interpretability [17; 18; 19]. Constructing an accurate and a small size (hence, better interpretable) DT is a challenging problem, and has been an active area of research over the last four decades [20; 21]. Most of the existing methods are ad hoc, and do not have explicit control over the size and accuracy of a DT. For example, there are greedy splitting-based [17; 18; 19], Bayesian-based [22; 23; 24; 25; 26], and branch-and-bound methods [9] for DT construction. A single decision tree is interpretable, but often falls short in modeling complex real-world data, and hence, tree ensemble methods such as random forest (RF) [27], genetic programming based decision tree ensemble [28] have been developed. While these existing ensemble methods have shown improvements in prediction accuracy and mitigating overfitting risk, due to the heuristic algorithms of decision tree generation, they often face challenges such as the preference for larger trees, lack of statistical interpretability, randomness in feature importance measurement. In this paper, to handle those challenges, we proposed DT-sampler, a SAT-based decision tree sampling method where we allow the user (e.g., a domain expert) to explicitly control the size and accuracy of a DT. We leverage a Boolean satisfiability(SAT) encoding [29] and propose a novel encoding of the DT sample space using only branch nodes and perform (uniform) sampling of the SAT space with user-specified accuracy and size. The proposed encoding generates a more compact search space than that of the existing methods [21]. Our method is a tree ensemble method that generates small-size and high-accuracy decision trees, and determines the feature importance based on its emergence probability (i.e., the probability of a feature appearing in the high accuracy space).

In a decision tree classifier, relative frequency is generally used to assign a class probability to a test instance. Relative frequency is defined as the proportion of training instances belonging to a specific class in the leaf where the test instance falls. However, these class probabilities are heuristic notion of uncertainty of the underlying decision tree model and do not ensure any valid statistical guarantee. Therefore, we propose to use conformal prediction (CP) to post-calibrate the sampled trees of DT-sampler. CP is a generic framework to post-calibrate any arbitrary (possibly imperfect) predictor and ensures a valid finite sample statistical guarantee under the weak assumptions of data exchangeability [30; 31]. There have been studies to conformalize a decision tree ensemble using CP, such as conformal genetic programming-based tree ensemble [28] or conformal

*This author contributed to this project when he was affiliated with the Department of Computational Biology and Medical Sciences, the University of Tokyo. His current affiliation is Rakuten Group, Inc., Japan.

†Corresponding author: tsuda@k.u-tokyo.ac.jp
‡Corresponding author: diptesh.das@edu.k.u-tokyo.ac.jp

Table 1: Description of propositional variables used in SAT-based DT encoding. The description indicates that a variable is set to 1 if the condition is true; otherwise, it is set to 0.

Var	Description of variables
vl_i	1 iff branch node i has a left branch child, $i \in [N]$
vr_i	1 iff branch node i has a right branch child, $i \in [N]$
l_{ij}	1 iff node i has node j as the left child, with $j \in Child(i)$
r_{ij}	1 iff node i has node j as the right child, with $j \in Child(i)$
lc_i	1 iff class of the left leaf child of node i is 1, $i \in [N]$
rc_i	1 iff class of the right leaf child of node i is 1, $i \in [N]$
a_{rj}	1 iff feature f_r is assigned to node j, $r \in [K]$, $j \in [N]$
u_{rj}	1 iff feature f_r is being discriminated against by node j, $r \in [K]$, $j \in [N]$

random forest [32]. Although these methods enjoy the statistical validity of CP, unlike SAT-based decision tree sampler, these methods do not have explicit control over accuracy and tree size. Hence, these methods are unable to produce optimal (in size and accuracy) ensemble of decision trees. To the best of our knowledge, this is the first attempt to apply conformal prediction in the context of SAT-based decision tree sampling.Through numerical experiments, we evaluated our proposed method using several benchmark and real-world datasets. We demonstrated that our method is capable of producing comparable accuracy as RF but with small-size DTs. We compared our encoding with existing SAT-based encoding and demonstrated that our encoding scheme generates fewer variables and is computationally more efficient. We also performed stability analysis of DT-sampler against RF both in terms of feature importance measurement as well as prediction tasks. The randomness in tree generation in RF makes it difficult to generate stable feature importance measurement results. Furthermore, the post-calibration using CP framework demonstrates that tree ensemble generated by DT-sampler is statistically more efficient and stable than that of random forest tree ensemble. An open source implementation of DT-Sampler is available at `https://github.com/tsudalab/DT-sampler-CP`.

2. METHOD

2.1 SAT-based decision tree encoding

Constructing decision trees with high accuracy and small size is an active area of research in the domain of constraint programming, and many Boolean satisfiability (SAT)-based encodings have been proposed in the literature [33; 21; 34; 35]. To reduce the search space and enable fast sampling we proposed an efficient SAT-encoding of DT. Our encoding method is motivated by the method proposed in [21], but developed a new encoding (only branch node encoding) scheme that accelerates the process of SAT sampling significantly. We also introduced additional variables and constraints to make it possible to encode the DTs with any accuracy that you want. Encoding DTs with only branch nodes is non-trivial, the details of which have been provided below.

SAT variables and constraints. We consider the encoding of decision trees with 2N+1 nodes and the training data consists of M samples and K features. Binary decision tree with 2N+1 nodes comprises N branch nodes and N+1 leaf nodes. The base method [21] sets the node ID sequentially as showed in Figure 1.a. Since it cannot distinguish between branch and leaf nodes by node IDs, branch and leaf nodes are assigned equivalent variables and are differentiated by

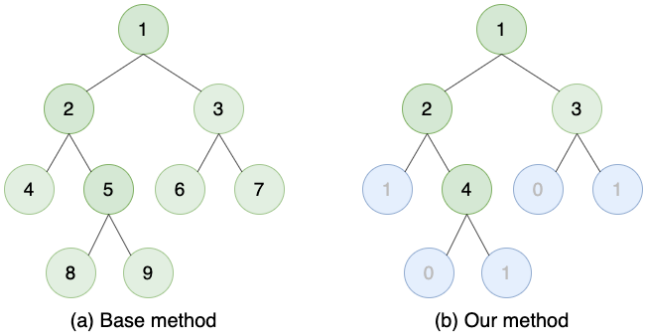


Figure 1: Node ID in SAT-based encoding of DT. (a) Base method cannot distinguish between branch and leaf nodes by node IDs. (b) Our method only uses the branch nodes to encode a DT.

additional constraints. In order to simplify it, we propose a method that only takes the branch nodes into consideration, viewing the leaf nodes as one of the properties of branch nodes as depicted in Figure 1.b. All the variables required to encode a DT are shown in Table 1, the subscript i and j represent the node ID (or index) while the subscript r denotes the feature ID (or index). For any natural number n, we use

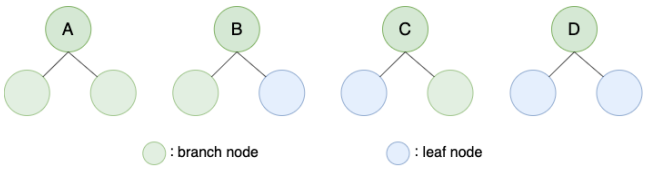


Figure 2: At any level of the DT construction, there can be four types of branch nodes.

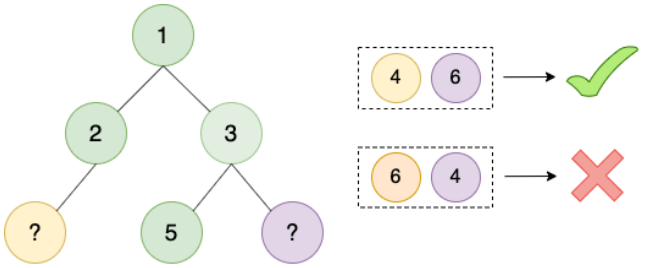


Figure 3: IDs of branch nodes are assigned according to level order of the tree.

$[m:n] = \{m, m+1, \ldots, n-1, n\}$. The function defined as $Child(i) = [i+1 : \min(2i+1, N)]$ can return possible node IDs of the children of the i^{th} node. There are four types of branch nodes as depicted in Figure 2. We use vl_i (resp. vr_i) variable to denote whether the i^{th} node has a left (resp. right) branch child or not. With $i \in [1:N]$ and $C \in \{0,1\}$,

$$vl_i = C \implies \left(\sum_{j \in Child(i)} l_{ij} \right) = C \quad \text{and}$$
$$vr_i = C \implies \left(\sum_{j \in Child(i)} r_{ij} \right) = C. \tag{1}$$

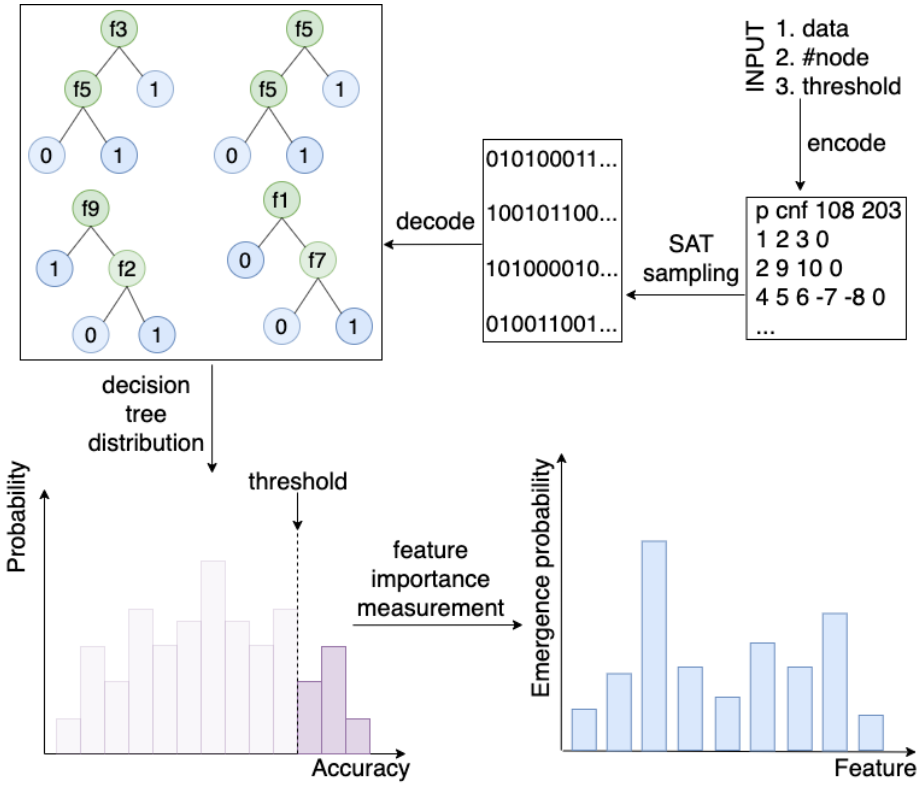


Figure 4: Feature importance measurement based on decision tree sampling by DT-sampler. First, we encode DTs with specific size (#node) and accuracy threshold (τ) as a SAT problem represented in conjunctive normal form (CNF). Once the SAT encoding of DTs is constructed, any solution that satisfies all the constraints in the CNF file can be decoded into a valid decision tree of specific size and accuracy. Then, we utilize SAT sampling to generate multiple decision trees and calculate feature importance (emergence probability) based on the sampling results.

Every branch node (except root) has exactly one parent such that

$$\sum_{i=\lfloor \frac{j}{2} \rfloor}^{j-1} (l_{ij} + r_{ij}) = 1, \quad j \in [2:N]. \tag{2}$$

The IDs of branch nodes are assigned according to level order of the tree. For example, as shown in Figure 3, if $l_{35} = 1$, then l_{26} or r_{26} must be 0, because the 6^{th} node cannot appear in front of the 5^{th} node. With $i \in [1 : N-1]$, $j \in Child(i)$,

$$\begin{aligned} l_{ij} \vee r_{ij} \implies \sum_{h=1}^{i-1} \sum_{k=j}^{k=N} (l_{hk} + r_{hk}) = 0 \quad \text{and} \\ r_{ij} \implies \sum_{k=j}^{k=N} l_{ik} + \sum_{k=j+1}^{k=N} r_{ik} = 0. \end{aligned} \tag{3}$$

At any branch node, exactly one feature is assigned such that

$$\sum_{r=1}^{K} a_{rj} = 1, \quad j \in [1:N]. \tag{4}$$

Variable u_{rj} has the information of whether the r^{th} feature is discriminated at any node on the path from the root to this node. If the r^{th} feature has already been assigned to one of ancestors, then it should not be assigned again. With $r \in [1:K]$, $j \in [1:N]$,

$$\begin{aligned} \bigwedge_{i=\lfloor \frac{j}{2} \rfloor}^{j-1} (u_{ri} \wedge (l_{ij} \vee r_{ij}) \implies \neg a_{rj}) \quad \text{and} \\ u_{rj} \iff (a_{rj} \vee \bigvee_{i=\lfloor \frac{j}{2} \rfloor}^{j-1} (u_{ri} \wedge (l_{ij} \vee r_{ij}))). \end{aligned} \tag{5}$$

The encoding given by Formula (1)-(5) specify a space including all of valid decision trees of a given size but can't learn from the training data. To learn from the training data, we need to track if the r^{th} feature was discriminated positively or negatively along the path from the root to j^{th} node as proposed in [21]. We adopted the same strategy in our method. Furthermore, to constrain the accuracy of the sampled decision trees, we introduce a binary variable w_t for each training example (x_t, y_t). The variable $w_t = 1$ if the sampled decision tree correctly classifies the t^{th} example. Formula (6) enforces that the overall classification accuracy must meet or exceed a specific threshold $\tau \in [0, 1]$. For example, if the parameter $\tau = 0.8$, then the decision trees must correctly classify at least 80% samples such that

$$\frac{1}{M} \sum_{t=1}^{M} w_t \geq \tau, \quad \tau \in [0, 1]. \tag{6}$$

Table 2: Comparison of encoding size. $\#b$: number of training samples, $\#f$: number of features, $\#n$: number of decision tree nodes, and τ : training accuracy threshold (6), used in encoding. The $\#var$ denotes the number of variables used to build the encoding of the decision tree. The $\#var\_cnf$, $\#cls\_cnf$ denote the number of variables and the number of clauses in the Conjunctive Normal Form (CNF) file generated by Tseitin transformation provided in z3-solver [36]. $\#Ave.\ time$ denotes the time to generate 100 samples by unigen. We ran all the experiments (including the base encoding) on Intel(R) Xeon(R) CPU E3-1270 v6 3.80GHz. All results are shown in the order of existing base encoding method and our proposed DT-sampler.

Dataset	$\#b$	#f	$\#n$	τ	$Method$	$\#var$	$\#var\_cnf$	$\#cls\_cnf$	$Ave.\ time\ (s)$
mouse	50	15	13	1.00	Base [21]	896	3599	20586	31.76
					DT-sampler	406	1274	8397	0.25
	50	15	11	0.90	Base [21]	747	3260	22890	1028.02
					DT-sampler	336	1990	24689	567.08
car	40	10	17	1.00	Base [21]	866	3956	21625	260.68
					DT-sampler	390	1406	9495	65.47
	50	15	11	0.90	Base [21]	747	3436	26218	1722.26
					DT-sampler	336	2152	33473	310.99
breast	40	15	17	1.00	Base [21]	1206	5821	32400	96.34
					DT-sampler	550	2041	13663	2.32
	50	12	13	0.90	Base [21]	740	3759	24732	407.32
					DT-sampler	334	2198	27869	125.81
heart	40	19	13	1.00	Base [21]	1104	4572	26063	79.87
					DT-sampler	502	1590	10697	11.62
	50	10	13	0.90	Base [21]	636	3241	21568	1671.18
					DT-sampler	286	2108	25552	327.89

2.2 Decision tree sampling

Sampling method. To obtain samples from the decision tree space, we employ two SAT samplers: QuickSampler [37] and UniGen3 [38]. QuickSampler is a heuristic search algorithm that can generate large amounts of samples quickly. The algorithm starts with a random assignment and iteratively modifies the assignment by flipping the truth values of randomly selected variables. It is very efficient but the uniformity cannot be guaranteed. In contrast, UniGen3 is a more sophisticated algorithm for uniform SAT sampling with solid theoretical guarantees. It requires adding extra clauses to the encoding, which makes the sampling process computationally expensive.

Sampling set. Unigen3 and Quicksampler allow users to assign a subset of all the variables as sampling set. If the sampling set contains Y variables, the size of the solution search space will be 2^Y. The samplers provide uniformity within the sampling set and increasing Y may adversely affect the sampling efficiency. Only part of variables will be in the sampling set. For example, u_{rj} is used to ensure that there are no repeated assigned features in any decision path but we do not need it during the decoding process. In addition, either the set $\{vl_i, vr_i\}$ or the set $\{l_{ij}, r_{ij}\}$ contains all the information needed to decode the tree structure, we only need to add one of them in the sampling set. Therefore, the smallest sampling set is $\{vl_i, vr_i, lc_i, rc_i, a_{rj}\}$.

2.3 Feature importance measurement

In [39], the authors measured the importance of elements in sequences based on the distribution under a qualification threshold. Inspired by this concept, we define feature importance as the contribution of each feature to a high accuracy space. Specifically, within a space consisting of decision trees surpassing a given threshold, the contributions can be evaluated based on the probability of each feature appearing in this space (we name it as "emergence probability"). Since we sample decision trees from uniform distribution, we can estimate the probability by just counting how many times each feature appears. The flow diagram of our method is shown in Figure 4. Random forest often uses feature permutation or mean decrease in impunity to calculate feature importance. It is also possible to apply these approaches to our framework.

2.4 Calibration using conformal prediction

We use conformal prediction to conformalize the sampled trees by a DT-sampler, thus generating an ensemble of conformal trees. Unlike the standard decision tree, a conformal decision tree produces a prediction set $C^\alpha(x)$ that contains the (unknown) true label y of a test example x with probability at least $1-\alpha$, for any error rate $\alpha \in [0,1]$ such that

$$\mathbb{P}\Big(y \in C^\alpha(x)\Big) \geq 1-\alpha. \tag{7}$$

In a binary classification with two possible categories (1: positive class and 0: negative class), we define the set $\mathcal{Y} = \big\{\{0\},\{1\},\{0,1\},\emptyset\big\}$. A conformal decision tree produces a prediction set $C^\alpha \subset \mathcal{Y}$ that may contain a single class, multiple classes, or be empty. Theoretically, any conformal decision tree is a statistically valid tree and it conforms to the coverage guarantee stated in (7). Therefore, we are mainly interested in finding statistically efficient conformal trees where efficiency is determined by the proportion of singleton (one class) prediction against multi-class or empty predictions it makes. A conformal tree is statistically efficient if it makes a large number of singleton predictions and a small number of multi-class or empty predictions.The statistical efficiency of a conformal tree depends on many factors including the accuracy of the underlying prediction model. The size (node counts) of a decision tree is a key factor that trades off between accuracy and interpretability of a decision tree. Existing conformal tree ensemble methods [32] suggest that unpruned tree is essential for generating accurate prediction model and improved statisti-

Algorithm 1: Inductive (or split) conformal prediction

Input:

Data $(X_i, Y_i) \in \mathbb{R}^p \times \mathcal{Y},\ 1 \leq i \leq n$;
$\mathcal{Y} = \{1, \ldots, m\}$ is a finite set of m classes;
Test data $X_{\text{test}} \in \mathbb{R}^p$;
Miscoverage level $\alpha \in (0, 1)$;
Prediction algorithm A used in "Fit model" below.

Process:

- Randomly split $\{1, \ldots, n\}$ into disjoint sets $\mathcal{I}_{\text{prop}}$ and $\mathcal{I}_{\text{cal}}$;
- Fit model: $P(Y = y|X = x) \leftarrow A(\{(X_i, Y_i),\ \forall i \in \mathcal{I}_{\text{prop}}\})$;
- Compute non-conformity scores $s_i,\ \forall i \in \mathcal{I}_{\text{cal}}$;
- Compute $\hat{q}_{1-\alpha}$, the $\frac{\lceil(|\mathcal{I}_{\text{cal}}|+1)(1-\alpha)\rceil}{|\mathcal{I}_{\text{cal}}|}$ empirical quantile of $\{s_i : i \in \mathcal{I}_{\text{cal}}\}$.

Output:

Prediction set $\mathcal{C}^{\alpha}(X_{\text{test}}) = \{y : s(X_{\text{test}}, y) \leq \hat{q}_{1-\alpha}\}$.

cal efficiency. However, this may lead to generation of large and complex decision trees (long chain of "if-then-else" rules), thus compromising interpretabilty, one of the key benefits of using decision tree in the context of trustworthy AI. Unlike existing tree ensemble frameworks (e.g., random forest, genetic algorithm-based tree ensemble), we propose a SAT-based decision tree sampler (DT-sampler) where we have explicit control over both the size and accuracy of the generated trees. Hence, although we generate an ensemble of trees, the generated trees are of similar size and accuracy, and provides better interpretability than the existing tree ensemble methods. See results for empirical evaluation.

Inductive conformal prediction. We use inductive conformal prediction (ICP) framework [40; 30] as it is computationally efficient. Given a labelled dataset $\mathcal{D}_\ell = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}$, we randomly split the indices $\{1, \ldots, \ell\}$ into two disjoint sets $\mathcal{I}_{tr}$ and $\mathcal{I}_{te}$. The training set indices $\mathcal{I}_{tr}$ is further splitted into another two disjoint sets $\mathcal{I}_{prop}$ and $\mathcal{I}_{cal}$. A prediction model (e.g. DT-sampler) is trained only once with the examples in the proper training set. The trained prediction model is then used to generate the non-conformity scores ($s \in \mathbb{R}$) for the examples in calibration set as well as for the test instances, where a large score implies bad agreement between x and y. We then compute the $\hat{q}_{1-\alpha}$ as the $\frac{\lceil(|\mathcal{I}_{cal}|+1)(1-\alpha)\rceil}{|\mathcal{I}_{cal}|}$ quantile of the calibration scores such that

$$\hat{q}_{1-\alpha} = Quantile\left(s_1, \ldots, s_n; \frac{\lceil(|\mathcal{I}_{cal}| + 1)(1-\alpha)\rceil}{|\mathcal{I}_{cal}|}\right). \quad (8)$$

Therefore, an inductive conformal prediction set $C^{\alpha}(X_{test})$ for any test example X_{test} and any miscoverage level $\alpha \in (0, 1)$ can be defined such that

$$C^{\alpha}(X_{test}) = \{y \in \mathcal{Y} : s(X_{test}, y) \leq \hat{q}_{1-\alpha}\}. \quad (9)$$

To compute this prediction set (9), we require a prediction model A and an associated score function S. A common choice of non-conformity score function in classification settings is the *"1 – class probability"*, defined as

$$s(x, y) = 1 - p_x^y,$$

or, alternatively, the *"1 – class margin"*, defined as

$$s(x, y) = 1 - \big(p_x^y - \max_{y' \in \mathcal{Y} \setminus \{y\}} p_x^{y'}\big),$$

where $p_x^y = \mathbb{P}(Y = y \mid X = x)$ denotes the conditional class probability, and $\mathcal{Y} = \{1, \ldots, m\}$ is the finite set of m class labels. In the ICP framework, the training of a prediction model and the generation of the non-conformity scores of the calibration set examples are executed only once, and both the learned model as well as the calibration scores are stored for repeated use, thus making it computationally efficient. Now, we formally state the coverage guarantee of ICP framework next.

THEOREM 1 (ICP COVERAGE GUARANTEE [40]). *If $\{(X_i, Y_i)\}_{i \in \mathcal{I}_{cal}}$ and (X_{test}, Y_{test}) are exchangeable random variables and we define $\hat{q}_{1-\alpha}$ (8) and $C^{\alpha}(X_{test})$ (9), then for any miscoverage level $\alpha \in (0, 1)$,*

$$\mathbb{P}\Big(Y_{test} \in C^{\alpha}(X_{test})\Big) \geq (1 - \alpha).$$

Note that the above coverage is called "marginal coverage" where the probability is marginal (averaged) over the randomness of the calibration set and the test data point. For the proof of Theorem 1, please see [40]. An algorithm to compute the prediction set is given in Algorithm 1.

3. RESULTS

Dataset. We compared DT-sampler with RF using several real-world benchmark datasets [41]. In addition to that we also used real world biological and criminal justice dataset. We considered Entacmaea quadricolor fluorescent protein eqFP611, two variant of which namely one bright deep-red (mKate2, $\lambda_{ex} = 590$nm, $\lambda_{em} = 635$nm) and one bright blue (mTagBFP2, $\lambda_{ex} = 405$nm, $\lambda_{em} = 460$nm) are seperated by thirteen mutations [42]. From biological perspective it is important to identify the crucial mutations and their pattern of epistasis (high-order interactions among mutations) that relate to the phenotypes (e.g., brightness). We also evaluated our method using ProPublica's COMPAS recidivism dataset [43] which contains seven categorical and integer-valued features and binary class labels. The equivalent 14 binary features and binary class labels are download from the Github repository of CORELS [9]. Model interpretabilty is crucial for the analysis of such high-stake decision making problems where an algorithm derived predictions are associated with the life of a human being or critical biological analysis.

Comparison with existing SAT encodings. Our new encoding of tree structure reduces a large part of variables and constraints compared to the (base) encoding method in [21]. The results on several benchmark datasets proved the acceleration in the process of SAT sampling as shown in Table 2.

Prediction stability analysis using conformal prediction. Here, we present the calibration results obtained using the Conformal Prediction (CP) framework to demonstrate the statistical efficiency and stability of the DT-sampler compared with the Random Forest (RF) classifier. In an

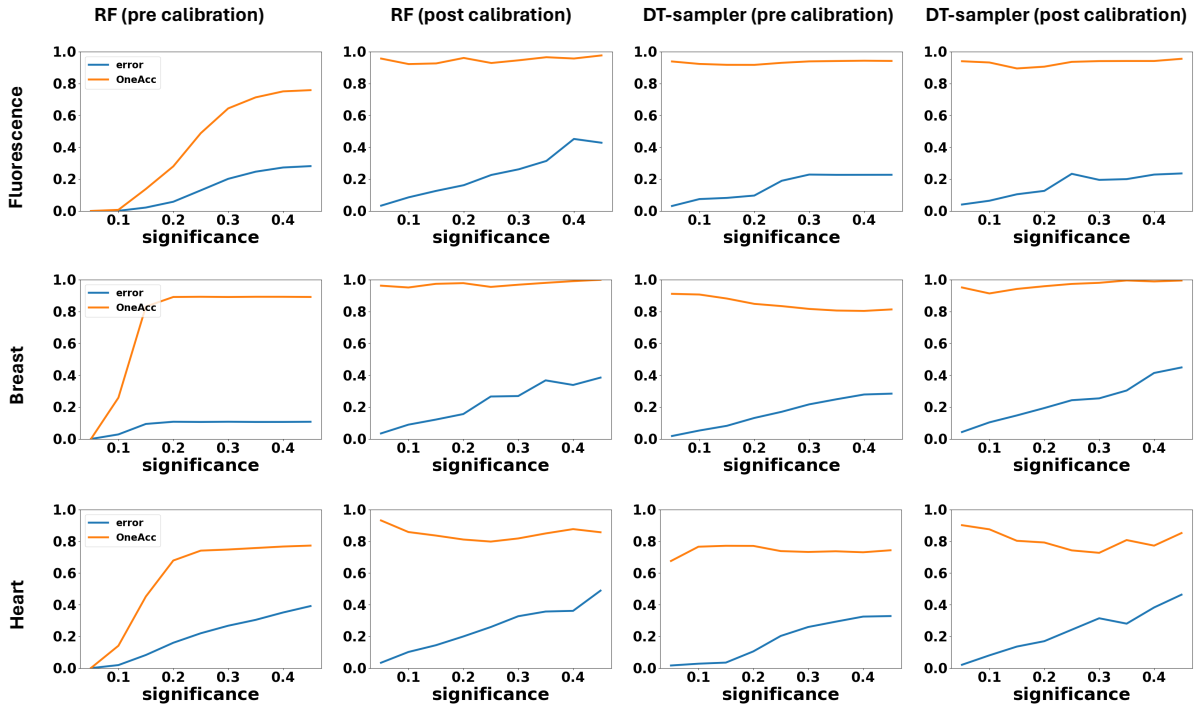


Figure 5: Pre and post calibration results using Fluorescence, Breast and Heart data at different significance levels. Here, error indicates the average miscoverage rate and OneAcc indicates the average one class accuracy. In pre-calibration, a single decision tree predictions are calibrated, whereas in post-calibration average predictions of all decision trees in an ensemble are calibrated.

RF, predictions are derived by aggregating the outputs of multiple randomly constructed decision trees, without explicit control over individual tree accuracy or size. Analysis of individual tree performance typically shows that many constituent trees have limited predictive accuracy. Nevertheless, the ensemble averaging process enables the RF to achieve strong overall predictive performance, albeit at the expense of interpretability. In contrast, the DT-sampler encodes both accuracy and tree size as satisfiability (SAT) constraints, enabling the generation of trees from regions characterized by high accuracy and user-specified complexity. This approach improves interpretability and prevents the formation of excessively large trees that may overfit the training data and generalize poorly to validation or test datasets. To demonstrate this effect we performed CP-based pre-calibration and post-calibration. In pre-calibration, individual tree predictions generated by both RF and DT-sampler are calibrated, whereas in post-calibration, tree ensemble predictions (average predictions) of both RF and DT-sampler are calibrated and the results are shown in Figure 5. We performed experiments on three datasets for different significance levels and for each significance level we repeated experiments for five times and reported the average results. We plotted the average one class accuracy (OneAcc) and average mis-coverage rate (error) for each significance level. Owing to the coverage guarantee of the CP framework, it can be observed that the mis-coverage rates are well controlled at every significance level for all experiments. However, the main differentiating factor is the OneAcc which is an indicator of statistical efficiency of the underlying prediction model. Precisely, a statistically efficient predictor is the one which makes maximum number of single class predictions and minimum number of multi-class or empty predictions. It can be clearly observed that the pre-calibrated OneAcc values of RF are bad at smaller significance (high coverage) levels for all the experiments, indicating that RF generates many random bad (statistically inefficient) predictors (trees) and averages them. On the other hand, both pre and post calibrated OneAcc values of DT-sampler are quite stable for all significance levels, indicating that DT-sampler is statistically more efficient at all significance levels.

Comparing accuracy and tree size between DT-sampler and RF. We compared our method with RF on several real-world benchmark datasets [41]. As shown in Table 3, our method can provide similar accuracy compared with random forest even if we sample decision trees in a small space. Relying on heuristic rules to build decision trees, random forest tends to generate larger decision trees, whereas DT-sampler have explicit control over tree size and can generate similar accuracy with smaller size trees.

Stable feature importance measurement. We define feature importance as its emergence probability in the high accuracy space as mentioned in §2.3. Parameter τ is used to describe what a high accuracy space means and its value depends on specific real-world scenarios and the desired level of strictness regarding accuracy requirements. To demonstrate our method, we utilize decision tree sampling on a subset of the breast-cancer dataset, which consists of 150 samples and 15 selected features. Initially, we set the accu-

Table 3: Comparison of tree sizes and accuracy. Grid search on parameters *max_leaf_nodes* is utilized to run random forest (RF). We reported mean ± standard deviation results of training and test accuracies of three experiments on different subsets of the corresponding datasets, shown in the order of RF/DT-sampler. $\#b$: number of training samples, $\#f$: number of features, $\tau(\%)$: training accuracy threshold (6) in percentage.

Dataset	$\#b$	$\#f$	τ (%)	*Method*	*#node*	*Training Acc.* (%)	*Test Acc.* (%)
mouse	50	15	92.0	RF	6.90	97.50 ± 2.50	93.33 ± 11.55
				DT-sampler	7.00	97.50 ± 2.50	93.33 ± 5.77
car	100	15	92.0	RF	16.93	92.92 ± 3.15	78.33 ± 5.77
				DT-sampler	11.00	94.00 ± 3.46	90.00 ± 0.00
breast	150	15	81.6	RF	19.00	96.94 ± 1.73	97.78 ± 1.92
				DT-sampler	11.00	98.00 ± 1.00	96.00 ± 3.46
heart	170	19	81.0	RF	23.00	93.33 ± 2.08	80.95 ± 2.18
				DT-sampler	15.00	85.67 ± 1.53	78.57 ± 6.23
diabetes	442	10	70.0	RF	11.00	79.37 ± 2.95	72.26 ± 5.78
				DT-sampler	9.00	76.83 ± 2.57	72.18 ± 4.01
penguins	214	6	90.0	RF	9.64	97.69 ± 1.10	96.85 ± 2.48
				DT-sampler	6.00	93.00 ± 1.73	91.23 ± 1.52
sonar	214	10	75.0	RF	5.00	78.61 ± 0.48	71.21 ± 5.12
				DT-sampler	5.00	78.00 ± 3.00	66.67 ± 0.93
compas	721	14	60.0	RF	14.64	70.89 ± 1.06	65.02 ± 5.57
				DT-sampler	7.00	68.56 ± 0.96	68.96 ± 2.19
fluorescence-1	100	13	90.0	RF	22.52	94.47 ± 5.07	90.60 ± 4.71
				DT-sampler	7.00	92.33 ± 2.08	90.73 ± 0.73
fluorescence-2	384	91	90.0	RF	33.48	95.00 ± 4.36	91.78 ± 1.94
				DT-sampler	11.00	91.33 ± 2.31	86.03 ± 3.18

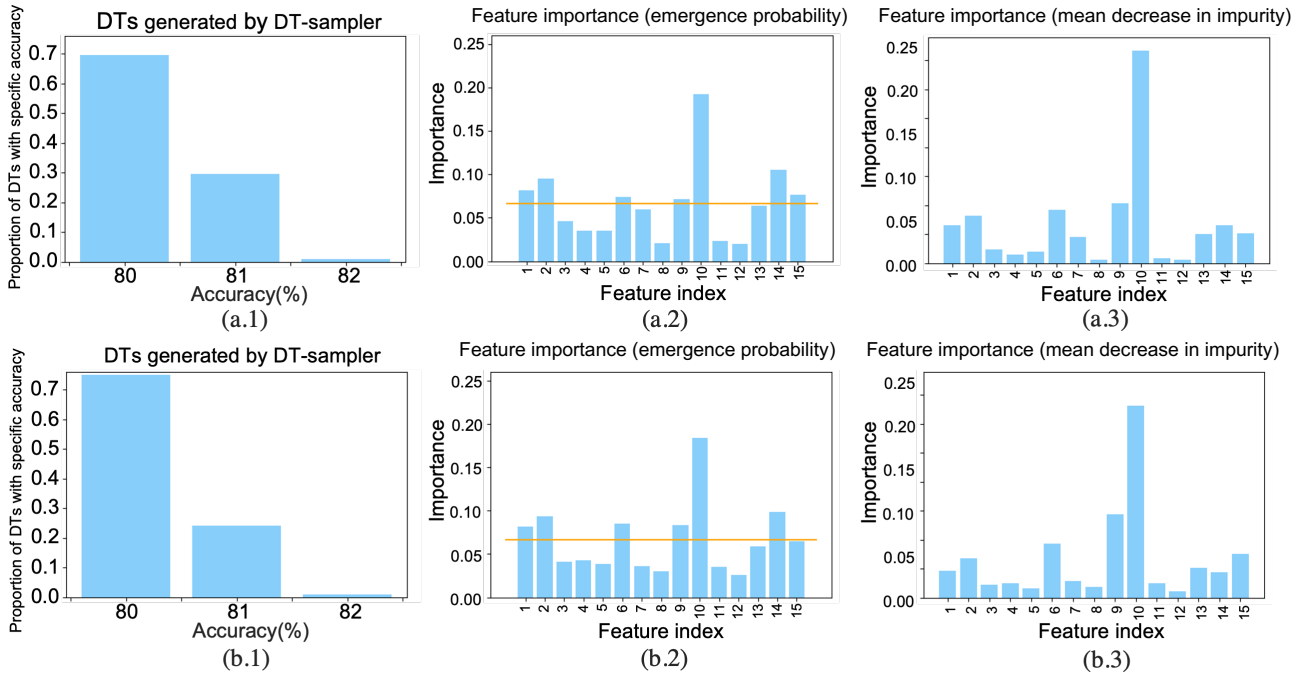


Figure 6: Stability of decision tree sampling. The two rows of figures show the results of two experiments on breast dataset using different random seeds during decision tree sampling. The first column shows the training accuracy distribution of the sampling results. The second and third columns show the feature importance measured by emergence probability and mean decrease in impurity, respectively.

racy threshold (τ) to 0, allowing for the random sampling of any decision tree. In this case, each feature is assigned to any branch node with equal probability, resulting in an emergence probability of $\frac{1}{15}$ for each feature. However, as we increase the threshold (τ), the emergence probabilities of the features differ. Features with an emergence probability

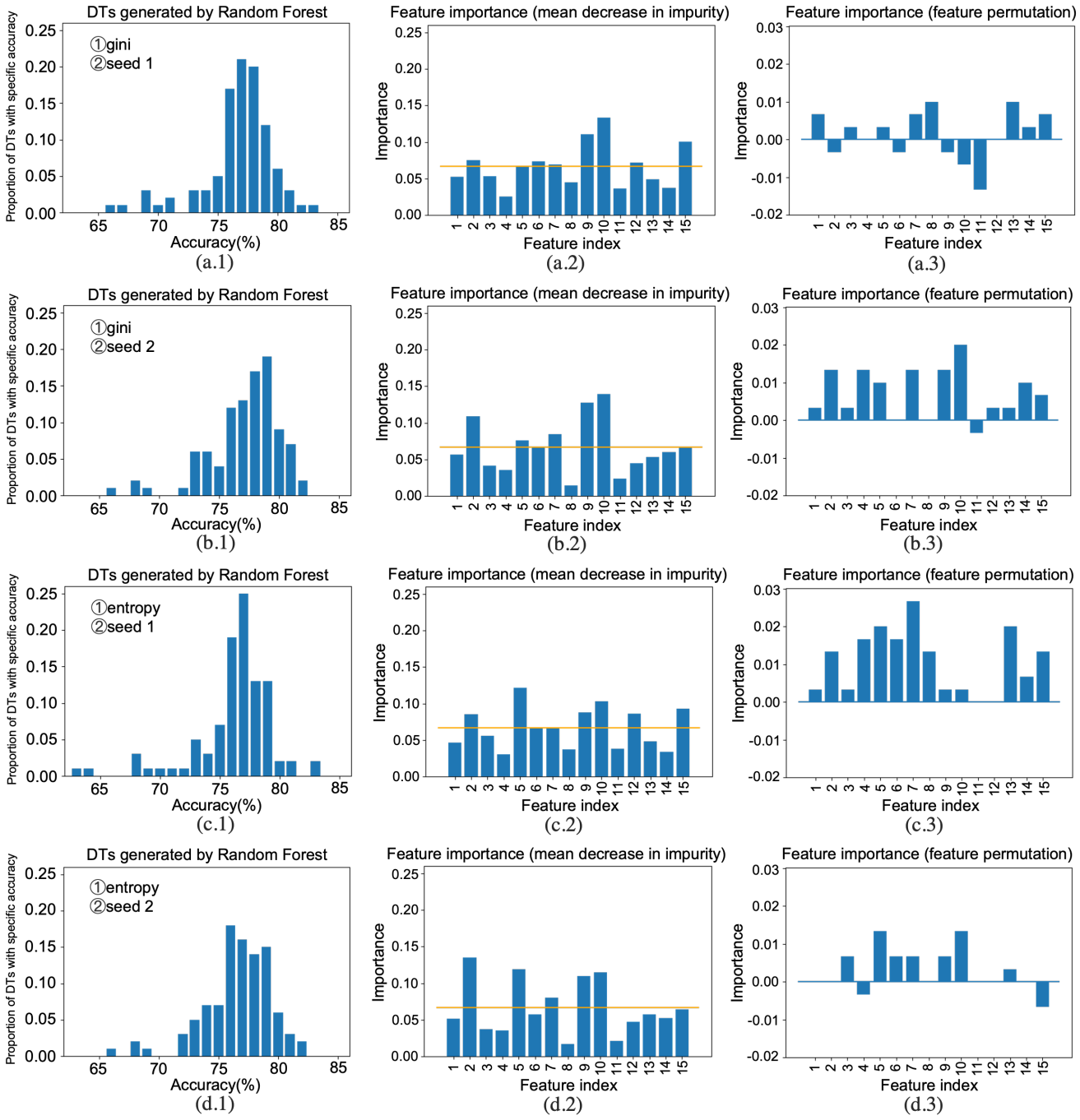


Figure 7: Drawbacks of random forest. The rows show the results of four experiments on breast dataset using different random seeds (seed 1, seed 2) and splitting criterion (gini, entropy) as random forest parameters. The first column shows the training accuracy distribution of the decision trees generated by random forest. The second and third columns show the feature importance measured by mean decrease in impurity and feature permutation, respectively.

$\geq \frac{1}{15}$ are considered important (see Figure 6). In random forest, the randomness of tree generation makes it difficult to generate stable results for feature importance measurement (see Figure 7 in appendix). DT-sampler shows superior stability compared to random forest. In Figure 7, we observe that when different random seeds or parameters are used, the distribution of decision trees generated by random forest consistently changes. This variability in tree generation directly impacts the feature importance results, leading to significant differences. Furthermore, random forest tends to generate a large number of trees with low accuracy, making it unreliable to measure feature importance for real-world problems. In contrast, DT-sampler calculates feature importance based on decision trees sampled exclusively from a

high accuracy space, which ensures the stability and interpretability of our results as depicted in Figure 6.

4. CONCLUSION

We proposed an SAT-based decision tree ensemble method and compared it with random forest using several benchmark and real-world datasets. We demonstrated that due to the randomness in tree generation and over-dependence on many parameters, random forest-based predictions and feature selections are unstable and unreliable. Our method provides a principled framework to measure feature importance based on sampling results from a high-accuracy space with a clear threshold, which offers stable analysis results for real-world problems. Using the conformal prediction framework, we demonstrated that the proposed method is statistically more efficient and produces stable predictions compared to random forest. Potential future research directions include developing a statistical hypothesis testing framework based on the proposed DT sampling method to assess the reliability of feature selection, and designing a fast SAT solver using quantum annealing or other QUBO-based approaches to enhance sampling efficiency.

5. AUTHOR CONTRIBUTIONS

Chao Huang, Koji Tsuda, and Diptesh Das conceived the idea and developed the methodology. Chao Huang, Xiaotian Xue, and Diptesh Das implemented the method. Diptesh Das designed the experiments; Chao Huang and Xiaotian Xue conducted them and generated results. All authors analyzed the results. Diptesh Das and Chao Huang wrote the manuscript. Diptesh Das and Koji Tsuda supervised the project. All authors reviewed and approved the final manuscript.

6. ACKNOWLEDGEMENTS

Koji Tsuda is supported by JST MIRAI, JST ERATO JPMJER1903 and JST CREST JPMJCR21O2. Diptesh Das is supported by JSPS KAKENHI 23K16942.

7. REFERENCES

[1] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1675–1684, 2016.

[2] Shamim Nemati, Andre Holder, Fereshteh Razmi, Matthew D Stanley, Gari D Clifford, and Timothy G Buchman. An interpretable machine learning model for accurate prediction of sepsis in the icu. *Critical care medicine*, 46:547–553, 2018.

[3] Muhammad Aurangzeb Ahmad, Carly Eckert, and Ankur Teredesai. Interpretable machine learning in healthcare. In *Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics*, pages 559–560, 2018.

[4] Diptesh Das, Junichi Ito, Tadashi Kadowaki, and Koji Tsuda. An interpretable machine learning model for diagnosis of alzheimer's disease. *PeerJ*, 7:e6543, 2019.

[5] Diptesh. Das. *Interpretable Machine Learning Models for Medical Data*. Ph.D. diss., Department of Computational Biology and Medical Sciences, The University of Tokyo, Kashiwa, Japan, 2019.

[6] Diptesh Das, Vo Nguyen Le Duy, Hiroyuki Hanada, Koji Tsuda, and Ichiro Takeuchi. Fast and more powerful selective inference for sparse high-order interaction model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9999–10007, 2022.

[7] Diptesh Das, Eugene Ndiaye, and Ichiro Takeuchi. A confidence machine for sparse high-order interaction model. *Stat*, 13:e633, 2024.

[8] Amina Adadi and Mohammed Berrada. Explainable ai for healthcare: from black box to interpretable models. In *Embedded Systems and Artificial Intelligence: Proceedings of ESAI 2019, Fez, Morocco*, pages 327–337. Springer, 2020.

[9] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. Learning certifiably optimal rule lists. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 35–44, 2017.

[10] Caroline Wang, Bin Han, Bhrij Patel, and Cynthia Rudin. In pursuit of interpretable, fair and accurate machine learning for criminal recidivism prediction. *Journal of Quantitative Criminology*, 39:519–581, 2023.

[11] Jiachang Liu, Chudi Zhong, Boxuan Li, Margo Seltzer, and Cynthia Rudin. Fasterrisk: fast and accurate interpretable risk scores. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 17760–17773, 2022.

[12] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI magazine*, 38:50–57, 2017.

[13] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

[14] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[15] Jianqing Fan and Jinchi Lv. Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70:849–911, 2008.

[16] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar conference on signals, systems and computers*, pages 40–44. IEEE, 1993.

[17] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.

[18] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[19] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Taylor & Francis, 1984.

[20] Hendrik Blockeel, Laurens Devos, Benoît Frénay, Géraldin Nanfack, and Siegfried Nijssen. Decision trees: from efficient prediction to responsible ai. *Frontiers in Artificial Intelligence*, 6:1124553, 2023.

[21] Nina Narodytska, Alexey Ignatiev, Filipe Pereira, and Joao Marques-Silva. Learning optimal decision trees with sat. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 1362–1368, 2018.

[22] David GT Denison, Bani K Mallick, and Adrian FM Smith. A bayesian cart algorithm. *Biometrika*, 85:363–377, 1998.

[23] Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayesian cart model search. *Journal of the American Statistical Association*, 93:935–948, 1998.

[24] Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayesian treed models. *Machine Learning*, 48:299–320, 2002.

[25] Hugh A Chipman, Edward I George, and Robert E McCulloch. Bart: Bayesian additive regression trees. 2010.

[26] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, and David Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. 2015.

[27] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[28] Ulf Johansson, Rikard König, Tuve Löfström, and Henrik Boström. Evolved decision trees as conformal predictors. In *2013 IEEE Congress on Evolutionary Computation*, pages 1794–1801. IEEE, 2013.

[29] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. IOS press, 2009.

[30] Anastasios N Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2021.

[31] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9:371–421, 2008.

[32] Ulf Johansson, Henrik Boström, and Tuve Löfström. Conformal prediction using decision trees. In *2013 IEEE 13th international conference on data mining*, pages 330–339. IEEE, 2013.

[33] Christian Bessiere, Emmanuel Hebrard, and Barry O'Sullivan. Minimising decision tree size as combinatorial optimisation. In *Principles and Practice of Constraint Programming-CP 2009: 15th International Conference, CP 2009 Lisbon, Portugal, September 20-24, 2009 Proceedings 15*, pages 173–187. Springer, 2009.

[34] Hélene Verhaeghe, Siegfried Nijssen, Gilles Pesant, Claude-Guy Quimper, and Pierre Schaus. Learning optimal decision trees using constraint programming. *Constraints*, 25:226–250, 2020.

[35] Mikoláš Janota and Antonio Morgado. *SAT-Based Encodings for Optimal Decision Trees with Explicit Paths*, pages 501–518. 06 2020.

[36] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[37] Rafael Dutra, Kevin Laeufer, Jonathan Bachrach, and Koushik Sen. Efficient sampling of sat solutions for testing. In *Proceedings of the 40th International Conference on Software Engineering*, page 549–559, 2018.

[38] Mate Soos, Stephan Gocht, and Kuldeep Meel. *Tinted, Detached, and Lazy CNF-XOR Solving and Its Applications to Counting and Sampling*, pages 463–484. 07 2020.

[39] Jiawen Li, Jinzhe Zhang, Ryo Tamura, and Koji Tsuda. Self-learning entropic population annealing for interpretable materials design. *Digital Discovery*, 1:295–302, 2022.

[40] Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. Inductive confidence machines for regression. In *Proceedings of the 13th European Conference on Machine Learning*, pages 345–356. Springer, 2002.

[41] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[42] Frank J Poelwijk, Michael Socolich, and Rama Ranganathan. Learning the pattern of epistasis linking genotype and phenotype in a protein. *Nature communications*, 10:4213, 2019.

[43] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. How we analyzed the compas recidivism algorithm. *ProPublica*, May 23, 2016. Accessed: 2025-11-09.

SpaCE-VAE: Sparse and Confident Explanations using Variational Autoencoders

Alexander Liu
Eindhoven University of Technology
Groene Loper 5
Eindhoven, The Netherlands
a.liu1@student.tue.nl

Sibylle Hess
Eindhoven University of Technology
Groene Loper 5
Eindhoven, The Netherlands
s.c.hess@tue.nl

ABSTRACT

In the field of explainable AI (XAI), a significant challenge lies in the evaluation of explanation methods. State-of-the-art techniques identify the importance of input features (e.g., pixels) for the classification, based on untestable assumptions about the model. Lowering the intensity of pixels, identified as irrelevant, typically leads to a change in the predicted class. To address this, we propose SpaCE-VAE, a Variational Autoencoder (VAE) designed to generate sparse, confident explanations that are inherently evaluable. SpaCE-VAE produces a sparse representation of an image, putting as many pixels as possible to black, such that the resulting image is still assigned to the same class as the original image (with high confidence).

1. INTRODUCTION

Pairing the empirical success of Deep Neural Networks (DNNs) with a trustworthy explanation method can be considered a holy grail in supervised learning. XAI [6] is supposed to provide trust in DNN models by indicating the mechanisms behind the black box classifiers. A popular XAI approach is to create attribution maps that identify the most important features [13]. Focusing on images, attribution maps indicate sets of pixels that are most relevant to the model's output [12].

Although current state-of-the-art methods are capable of highlighting regions that look meaningful to a human eye, they rely on untestable assumptions. Surrogate evaluation techniques use pixel perturbations, for example varying the intensities of the top k attributions and then measuring the drop in a classification metric [16]. This way, the evaluation is only meaningful in comparison to other (untestable) explanation techniques, and is hence subjective.

We argue that a good explanation should at least be verifiable by the classifier. Figure 1 illustrates the challenges that we face when we want to generate evaluable classifications. The figure shows how an image of a dog is classified as a cat when pixels with a negative attribution to the class *dog* are removed. Setting here the negative attribution pixels to zero alters the shape of the dog, that is also having dark fur. Our proposed method SpaCE-VAE maintains the important information in the contrast of the dog and the background. Details, such as the eyes, snout, and paws of the dog are not visible anymore, while the classifier still predicts the class *dog* with a high confidence.

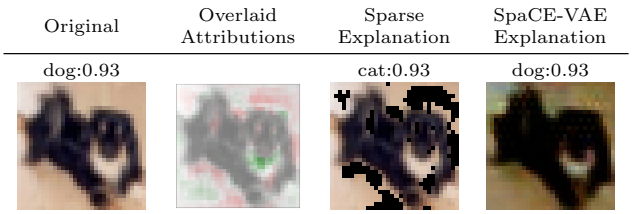


Figure 1: Illustrating the problem of evaluating attribution methods. Positive and negative attributions are indicated in green and red, respectively. The sparse explanation is generated by dropping the pixels with negative attributions. Removing *unimportant* pixels easily results in a misclassification.

In summary, our contributions are:

1. We propose SpaCE-VAE (Sparse Confident Explanations using Variational Autoencoders), a novel approach to generate sparse confident local explanations of a DNN that remain on the manifold of correctly classified images.
2. Our empirical analysis indicates that SpaCE-VAE is able to generate explanations that generalize over the test set.
3. Visual evaluation shows that SpaCE-VAE is able to highlight parts that give insight into the model, making sense from a human perspective.

1.1 Notation

We assume we are given a dataset

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$$

of n images $\mathbf{x}_i \in \mathbb{R}^{w \times h \times p}$ (width $\times$ height $\times$ channels) and their corresponding labels $y_i \in \{1, \ldots, c\}$. Let $C(x)$ be the classifier that we want to explain. We assume that $C : \mathbb{R}^d \to [0, 1]^c$ maps data points to a vector of class-confidences between zero and one.

2. RELATED WORK

In the taxonomy of XAI methods, we propose a local interpretation model, focusing on explaining individual predictions. Of those local interpretation models, we are closest to attribution methods.

2.1 Attribution methods

Feature attributions assign a score to each input feature based on the perceived importance to the output [12]. Attribution maps $A(\mathbf{x}, \tilde{y}; C) \in \mathbb{R}^d$ indicate the importance of each pixel (or feature) in the input image $\mathbf{x}$ towards the class $\tilde{y}$ predicted by the classifier C. The idea is that the attribution values highlight parts of the input image that are crucial for the classifier's decision-making process. For positive attributions, increasing the pixel intensity is expected to yield a higher confidence. For negative attributions, decreasing the intensity is expected to yield a higher confidence. Attribution models largely rely on an interpretation of *importance* based on the sensitivity to perturbations. The perturbation effects are, for example, measured by a change in the prediction confidence. Let $C_{\tilde{y}}(\mathbf{x}) \in [0, 1]$ be the confidence of predicting class $\tilde{y}$ with classifier C for input image $\mathbf{x}$. We approximate the prediction confidence by the first-order Taylor expansion of input image $\mathbf{x}$

$$C_{\tilde{y}}(\mathbf{x} + \Delta) \approx C_{\tilde{y}}(\mathbf{x}) + \nabla C_{\tilde{y}}(\mathbf{x})^\top \Delta, \tag{1}$$

$$\Leftrightarrow C_{\tilde{y}}(\mathbf{x} + \Delta) - C_{\tilde{y}}(\mathbf{x}) \approx \sum_{i,j,s} \frac{\partial C_{\tilde{y}}(\mathbf{x})}{\partial x_{ijs}} \Delta_{ijs}. \tag{2}$$

Equation (2) states that the impact of perturbation Δ to image $\mathbf{x}$ on the prediction confidence depends on the partial gradients. This observation motivates *Saliency Maps*, which are a very basic attribution technique, identifying the most important pixels as those that have the largest absolute value of the gradient over the channels [18]

$$A(\mathbf{x}, \tilde{y}; C)_{ij} = \sum_{s=1}^{p} \left| \frac{\partial C_{\tilde{y}}(\mathbf{x})}{\partial x_{ijs}} \right|.$$

Other aggregation methods over the channels, such as taking the maximum absolute value, are also possible. While saliency maps identify pixels to which the predictions are sensitive, the question remains whether those pixels provide good explanations. Adversarial examples also use local sensitivities identified by the gradient to fool the classifier. However, those examples are considered as rather artificial, providing examples outside of the manifold of reasonable images [20]. The question arises whether saliency maps highlight pixels that are crucial for the predicted class, or only perturbation-sensitive pixels. Those things are not necessarily the same.

DeepLIFT [17] generalizes the indication of important features by the Taylor expansion to contribution scores. This way, not only small differences in the input can be evaluated towards the prediction outcome, but any differences to a specified reference input. The reference input typically represents a default or neutral input chosen for the problem at hand (for example, a black image). Similarly to the structure of the Taylor expansion in Equation (2), DeepLIFT assigns layer-wise contribution scores $C_{\Delta_{z\,i}\Delta_h}$ to the latent representation $\mathbf{z} \in \mathbb{R}^l$ of $\mathbf{x}$ and their consecutive hidden layer output $h(\mathbf{z})$, such that it satisfies the property

$$\Delta_h = h(z + \Delta_z) - h(z) = \sum_{i=1}^{l} C_{\Delta_{z\,i}\Delta_h}. \tag{3}$$

Similar to layer-wise relevance propagation methods [1], the scores of the final layer are backpropagated to the input image by specified rules, similar to the chain rule backpropagation. The quality of the provided explanation largely hinges on the reasonability of the established rules, which are again not easy to assess empirically.

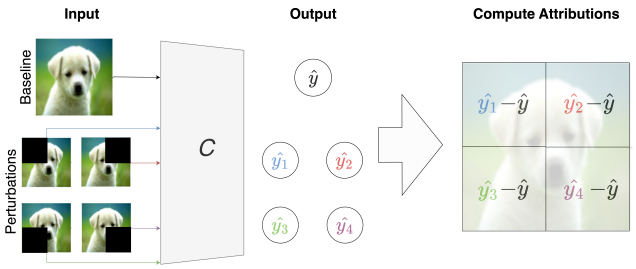


Figure 2: An illustrative example of Feature Ablation. The baseline image and perturbations are passed through the classifier C to obtain the reference score $\hat{y}$ and perturbed confidence scores $\hat{y_1}, \hat{y_2}, \hat{y_3}$, and $\hat{y_4}$. The attributions are computed by subtracting the reference score from each of the perturbed confidence scores.

Integrated Gradients [19] evaluates the importance of an input feature with respect to a reference image by means of fluctuations in the gradient on the path between both images. Features that exhibit significant changes along the path are attributed higher scores. That is, the importance value for the image tensor element at index (ijs) is defined as

$$IG_{ijs}(\mathbf{x} + \Delta, \mathbf{x}) = \Delta_{ijs} \cdot \int_0^1 \frac{\partial C_{\tilde{y}}(\mathbf{x} + \alpha\Delta)}{\partial x_{ijs}} d\alpha. \tag{4}$$

The integral is efficiently computed with the Riemann approximation. The corresponding attribution map aggregates the integrated gradients over the channels, for example, computing a norm or the sum. Integrated Gradients rely again on the assumption that local sensitivity to features, indicated by the gradients, indicates the importance of that feature for classification.

Feature Ablation [9] computes the attributions of the input by an empirical evaluation of the effect of perturbations of the original image. It removes regions or pixels by setting pixel intensities to zero (black). The attributions are computed by subtracting the confidence score of the perturbed image from the reference confidence score of the original image.

Figure 2 illustrates this approach by means of four perturbations. The confidence scores of the original image for the predicted class is $C_{\tilde{y}}(\mathbf{x}) = \hat{y}$ and $\hat{y}_1, \ldots, \hat{y}_4$ indicate the confidences of class $\tilde{y}$ for each of the perturbations. Afterwards, the attributions for each blackened region are given by the drop in confidence if the pixels of each region are removed. Hence, Feature Ablation provides a baseline for evaluable explanations. The clear drawback of Feature Ablation is that this is computationally very expensive.

The attribution methods discussed above (with the exception of feature ablation) are difficult to evaluate, as no ground truth is available [14]. Consequently, perturbation strategies are often employed [4]. These approaches assess model performance by systematically altering the input data or model parameters according to the attribution maps. However, they suffer from the drawback that sensitivity to feature changes does not necessarily imply that

those features are most important for the decision-making process. Or, in other words, the pixels that are close to the decision boundary are not necessarily the ones that have the most impact on the classification outcome.

2.2 Generative Model Explanations

An approach related to ours is GasTeN, generating example-based global explanations using a generative model [3]. GaSTeN uses GANs to generate images that are close to the decision boundary of two classes. These images can be used to gain insight into the classifier decision-making process. Like in our method, a regularization term is added to the reconstruction loss to generate images that are expected to provide good explanations (here, those are images close to the decision boundary).

Dabkowski and Gal [5] train a model that generates a binary mask to select pixels, or preferably areas, that are important for the prediction. The authors acknowledge that the usage of *pixel removal* strategies, such as applying the binary mask to the input image, introduces artifacts to the image that affect the performance of DNNs [7]. They propose evaluating images cropped to the smallest rectangle enclosing the salient region. This approach enables the assessment of the generated saliency maps, but the evaluation is restricted to rectangular crops, which may overlook important cases such as when the model relies on background features for its prediction.

Another class of related models, exemplified by Deep Dream [22], highlights structures and patterns that increase neuron activations. Unlike our method, which aims to remove important information, generative approaches such as Deep Dream add patterns to the explanation image. Since our approach also relies on a generative model, it occasionally introduces patterns into the image as well. As we will see in the evaluation, this can be useful for explaining misclassifications by indicating what the model *perceives*.

3. SPACE-VAE

We aim to generate explanations that lower the pixel intensity (increasing sparsity) of an image such that it is still classified confidently by the target classifier like the original image. We list the following desired properties for our explanations:

- **Accurate** The explanations should at least be classified coherently: the prediction of the original image should be equal to the prediction of the explanation.
- **Sparse** The explanation should reduce the number of activated pixels or the pixel intensities.
- **Confident** Ideally, the explanation should have a confidence at least as high as the model's confidence on the original image, so that it captures the features that actually drive the model's high-confidence prediction.
- **Interpretable** Although we encourage sparseness, the explanations should still be interpretable and the features or regions with high importance should be clear and visible.

We aim to develop a framework capable of producing explanations that are both sparse and confident. To achieve this objective, we introduce SpaCE-VAE, an extension of the Vector Quantized Variational Autoencoder (VQ-VAE) model [21]. The VQ-VAE uses in contrast to the vanilla VAE discrete distributions to model the latent space. It has demonstrated remarkable versatility in the generation of high-quality images [15], and we believe that the discrete representations fit the goal of SpaCE-VAE. Discrete variables are likely more suitable to push the learned representations towards identifying concepts of images that explain the classifiers behavior. Furthermore, the VQ-VAE architecture avoids common problems in VAEs, such as the posterior collapse problem, where a decoder simply ignores learned latent representations and variance issues.

Figure 3 illustrates the overall idea of SpaCE-VAE. A VQ-VAE model, consisting of encoder E, decoder D, and embedding space e is trained to explain predictions of classifier C. The reconstruction generated by the VAE lowers pixel intensities such that the classifications of the reconstructed and the original image remain the same.

3.1 Loss Function

VQ-VAE models pass an image $\mathbf{x}$ to the encoder E, which generates an output $z_e(\mathbf{x}) \in \mathbb{R}^{d_q}$. Then, $z_e(\mathbf{x})$ is passed to the quantizer Q. Here, the discrete latent variables are computed using a nearest centroid lookup using the shared embedding space $e = \{\mathbf{e}_1, \ldots, \mathbf{e}_k\} \subseteq \mathbb{R}^{d_q}$

$$z_q(\mathbf{x}) = \mathbf{e}_l,\ l = \arg\ \min_j \|z_e(\mathbf{x}) - e_j\|. \tag{5}$$

The quantization process returns $z_q(\mathbf{x}) \in \mathbb{R}^{d_q}$. Afterwards, the decoder D maps the quantized representation $z_q(\mathbf{x})$ back to a reconstructed image $\hat{\mathbf{x}}$. Here, the gradient $\nabla_z L$ of the VQ-VAE loss is passed unaltered to the encoder to circumvent issues in the optimization of discrete variables [21].

We employ the method to update the embedding space by exponential moving averages (EMA) [21]. The updating process follows a similar procedure as k-Means [11] using an online version to update the embedding space. Using this approach, the model typically yields better performance and quicker convergence. Denoting the collective parameters of the encoder, the decoder and the embedding by the vector θ, the loss function of VQ-VAE-EMA is

$$\mathcal{L}_{\text{VQ-EMA}}(\theta, \mathbf{x}) = \log p(\mathbf{x}|z_q(\mathbf{x})) + \delta\|z_e(\mathbf{x}) - \text{sg}[e]\|^2. \tag{6}$$

Here, the first term $\log p(\mathbf{x}|z_q(\mathbf{x}))$ models the reconstruction loss. The function $\text{sg}[\cdot]$ denotes the stopgradient operator, which is defined as the identity at forward computation time, but it *stops the gradient* because it returns a gradient of zero at backward passes. As a result, the embedding space will not be optimized by the reconstruction loss. The last term $\delta\|z_e(\mathbf{x}) - \text{sg}[e]\|^2$ incentivizes the encoder outputs to commit to the embedding and it ensures that the output does not grow arbitrarily.

We adapt the VAE-VQ loss function to generate explanations of a given classifier C. We add as a sparseness term the L_1-norm of the reconstructed image $\hat{\mathbf{x}}$. The L_1-norm provides a good trade-off between the desired sparse representations (having a low L_0-norm) and the ability to optimize this term via numerical methods. To ensure that the explanations are coherently classified, we also add a cross-entropy term to the loss. Here, the parameters of the classifier are not optimized, only the parameters of the VAE, generating the explanations $\hat{\mathbf{x}} = vae(\mathbf{x})$. Denoting the predicted class

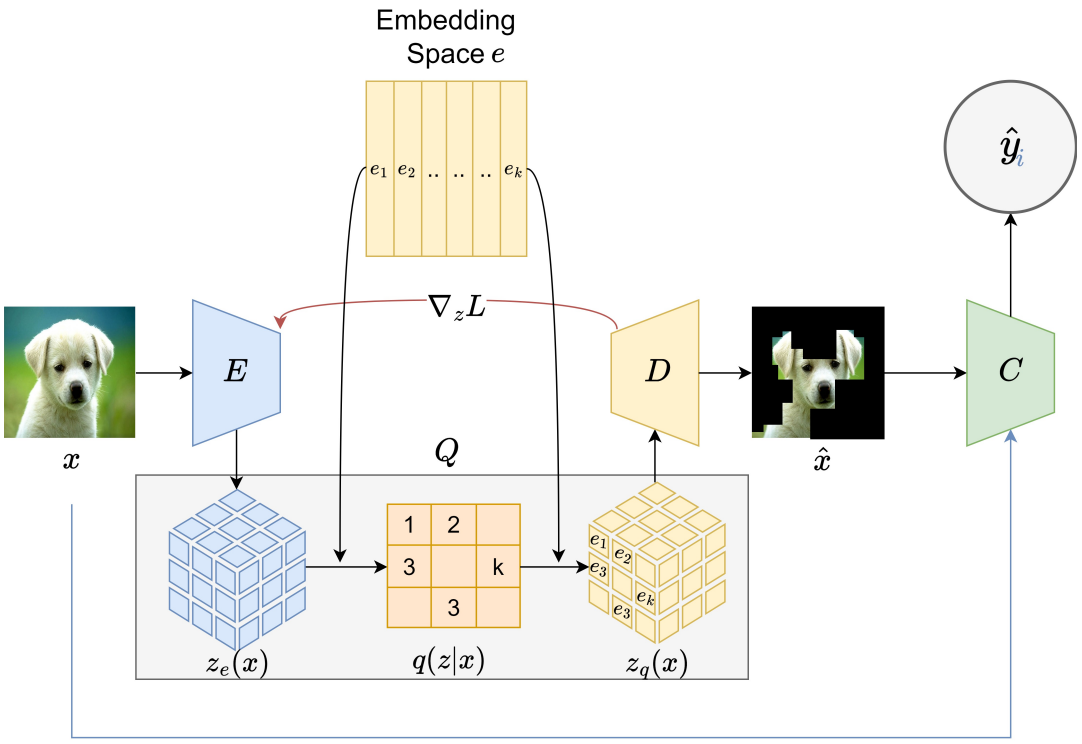


Figure 3: The SpaCE-VAE framework. A VQ-VAE (consisting of encoder E, embedding basis vectors $\mathbf{e}$ and decoder D) is trained to generate *sparse* images $\hat{\mathbf{x}}$ that approximates the input image $\mathbf{x}$ while decreasing the intensity of nonrelevant pixels such that the resulting image is still classified like the original image.

by $\tilde{y} = \arg\max_j C_j(\mathbf{x})$ and the confidence of the predicted class as $C_{\tilde{y}}(\hat{\mathbf{x}})$, we define our loss term as

$$\mathcal{L}_{\text{SpaCE}}(\theta, \mathbf{x}) = \log p(\mathbf{x}|z_q(\mathbf{x})) + \delta\|z_e(\mathbf{x}) - \text{sg}[e]\|^2 + \alpha|\hat{\mathbf{x}}| - \beta \log C_{\tilde{y}}(\hat{\mathbf{x}}). \quad (7)$$

The term $\alpha|\hat{\mathbf{x}}|$ incentivizes sparsity, and the term $-\beta \log C_{\tilde{y}}(\hat{\mathbf{x}})$ increases the confidence of the predicted class for the reconstruction. In the remainder of this paper, we choose a value of $\alpha = 0.05$, since it provides a notable drop in pixel intensities while not making the explaining images too dark.

4. EXPERIMENTS

We evaluate our method against the attribution methods DeepLIFT, Feature Ablation, and Integrated Gradients. Those methods are designed to provide explanations against a reference picture, such as a black image, and thus have a similar goal as SpaCE, to provide sparse explanations. We use the existing implementation of attribution methods by Captum [9].

We normalize the attribution methods such that competitors generate attributions $A \in [-1, 1]^d$. We try to find a good threshold value such that putting all pixels to zero (black), if they do not exceed the threshold, does not decrease the accuracy too much. For that reason, we try thresholds $\sigma \in \{-0.9, \ldots, -0.1, 0\}$ to generate explanations

$$\mathbf{x}_\sigma = \mathbb{1}[A \geq \sigma] \circ \mathbf{x},$$

where $\circ$ denotes the Hadamard product. The best threshold value is determined on the test set.

We evaluate our method and the competitors on the CIFAR10 dataset [10]. Training DNNs on CIFAR10 is reasonable, and it allows evaluating the computationally expensive Feature Ablation method. We sample a validation dataset containing 15% of the training data points by means of stratified sampling, ensuring that the relative proportions of each class remain consistent. The validation set is used to select the SpaCE model that achieves the lowest validation loss during training.

The classifier that we aim to provide explanations for is a VGG11 model. The training procedure uses a batch size of 256, and the network was optimized using stochastic gradient descent with a learning rate set to 0.01, weight decay of 0.01, and momentum of 0.9. After training, the model achieved 99.99% and 92.39% on the train and test sets, respectively.

4.1 Reproducibility details

To train the VQ-VAE for our SpaCE model, we use the ADAM optimizer with a learning rate of 0.0003. Following the commitment loss recommendations [21], we set $\delta = 0.25$, and we use an embedding space of 512 elements. We train the model for maximum 200 epochs, as long as the validation loss has not decreased for 20 epochs. We save the model with the highest validation loss. We provide the implementation of our method together with evaluation scripts[1]. Our code

[1]https://github.com/AlexanderLLLiu/SpaCE-VAE

builds upon existing implementations of the VGG11[2] and VQ-VAE[3].
All experiments are performed using an NVIDIA GeForce RTX 4070 Ti GPU and an Intel Core i9-13900K CPU. The complete training of one configuration takes approximately one hour.

4.2 Quantitative Analysis

We quantify our results by means of the average accuracy on the test set and the average sparsity measured in L_0-norm and L_1-norm. The sparsity is calculated for the input image $\mathbf{x}$ and sparse explanation $\hat{\mathbf{x}}$ as

$$L_p\text{-Sparsity} = \frac{\text{avg}(\{\|\mathbf{x}\|_p - \|\hat{\mathbf{x}}\|_p \mid \mathbf{x} \in \mathcal{D}\})}{\text{avg}(\{\|\mathbf{x}\|_p \mid \mathbf{x} \in \mathcal{D}\})} \cdot 100$$

where $\text{avg}(\cdot)$ computes the average of a set. Figure 4 illustrates the drop in accuracy that arises when generating explanations that put an increasing amount of pixels to zero, depending on the threshold of negative attributions. We observe that in particular DeepLIFT's and Integrated Gradients' masked explanations drop to an average accuracy of approximately 0.3 when removing only 10% of the pixels (equating a sparsity in L_0-norm of 10%). The computationally much more expensive Feature Ablation method yields better results, but the accuracy still drops steeply with an increase in sparsity.
For SpaCE-VAE, we plot the results for the three values of the hyperparameter β. We observe that, particularly for the L_1-norm, subject to which the model has been trained, SpaCE-VAE achieves a comparatively high accuracy for high L_1-norm sparsity. Yet, also with regard to the L_0-norm, SpaCE-VAE provides explanations with an accuracy above 70% for 20% of the pixels being black in average. The results indicate that our proposed method is indeed able to find explanations that still lie on the manifold of the correctly classified images.
Table 1 summarizes our experimental results. For our competitors, we display the results for the largest threshold σ that yields a L_0-sparsity smaller than 10%. Since the accuracy of competitors is dropping so rapidly, we choose the threshold this way to obtain informative explanations, which *remove* at least around 10% of the pixels. The maximum achievable accuracy is given this way by Feature Ablation with 0.65. The accuracy of SpaCE-VAE is around 0.73, with an L_1-sparsity around 50%. Considering that SpaCE-VAE achieves an accuracy of around 0.9 on the training set, it demonstrates a solid degree of generalization, though there is still room for improvement. This is a nontrivial result. While it is not surprising that many *explanations* can be constructed for an image—since numerous perturbations can increase a model's confidence in its prediction — the ability to transform images into sparse representations that still support high confidence in the predicted class indicates that the autoencoder is effectively learning to emphasize features relevant for classification while suppressing irrelevant ones.

4.3 Qualitative Evaluation

In Figure 5, we provide example explanations given by the binary mask of competitors and our method SpaCE-VAE. Above each image, you find the predicted class of the image and its confidence. We observe that the sparse explanations of DeepLIFT and Integrated Gradients are not very insightful, and they also lead to frequent misclassifications of the images where *nonimportant* pixels are set to black. Feature Ablation suffers (as expected) less from misclassification and provides, as well, some more coherent regions that are deemed unimportant. Still, we argue that the provided explanations are not suitable to identify what is important information of that image, but rather what is really unimportant (such as the small background area on the top left of the dog).
Regarding SpaCE-VAE, we observe that a parameter value of $\beta = 1$ leads to rather smooth, darker images that seem to be composed of a smaller color palette. Particularly interesting is the example of explaining a misclassified bird (as a cat) in row three. The explanation created by SpaCE-VAE with $\beta = 1$ emphasizes a cat-like form: in particular, there are three regions that could be interpreted as legs, smoothing out the background that does not contribute to the cat-like figure. Regarding the explanation of a *truck* in the last row, SpaCE-VAE ($\beta = 1$) suggests that the cabin is important for classification. SpaCE-VAE with a value of $\beta = 100$ reconstructs the texture of objects and animals more than SpaCE-VAE with $\beta = 1$. In the examples of a dog and a bird in the first two rows, we also observe that the boundary between the animal and the background is highlighted. This is insofar interesting as DNNs are usually identified to be reliant on texture in the first place [2; 8]. The reconstructions suggest that the VAE does indeed emphasize the shape information paired with some general color patterns (as seen in the intensified colors of the bird and the intensified cat-fur colors).

4.4 Limitations

A key assumption underlying our approach is that uninformative or irrelevant pixels can be effectively suppressed by driving them toward zero, which corresponds to black in standard image representations. However, this intuition does not hold universally across all visual contexts. For example, in an image of a dark-colored dog standing in front of a white-tiled wall (like in Figure 1), the semantically important pixels may be dark, while the background (which is irrelevant) is predominantly bright. In such cases, a sparsity constraint that pushes pixel values toward zero may inadvertently retain irrelevant regions while suppressing important ones.
This reflects a more general limitation of using mathematical simplicity — here, via an L_1 penalty — as a proxy for interpretability in image-based tasks. While L_1 regularization promotes sparsity, it does not necessarily align with the semantics of visual data. Furthermore, applying an L_1 penalty often results in globally darker images, which can make the explanations visually less intuitive or harder to interpret.

5. CONCLUSIONS

We propose SpaCE-VAE, a new framework to generate local explanations that enable a quantitative evaluation. We propose a loss function to train a VQ-VAE that generates images that are close to the input image, while lowering the intensity of as many pixels as possible without changing the

[2]https://github.com/huyvnphan/PyTorch_CIFAR10
[3]https://github.com/swasun/VQ-VAE-Images

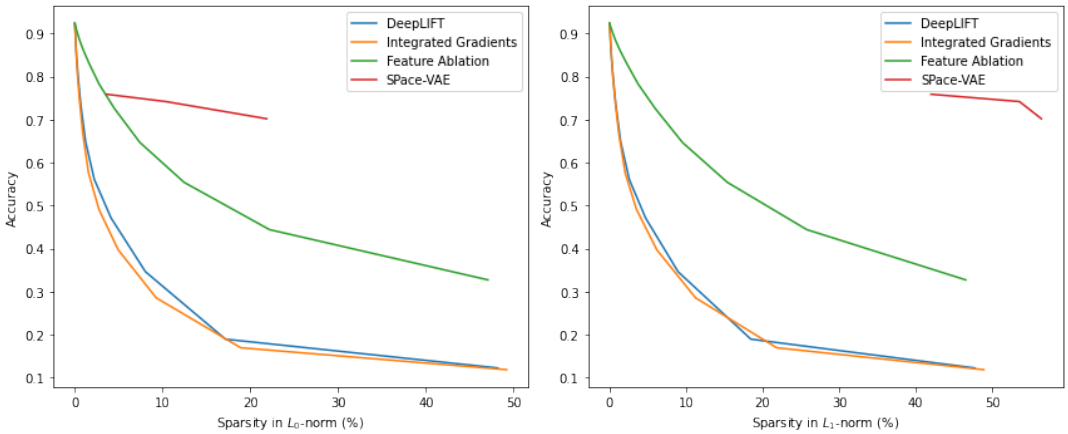


Figure 4: Plot of sparsity against the accuracy on competitors and our model SpaCE-VAE on the test set. The competitor plots connect the results given by varying the threshold value $\sigma \in \{-0.9, \ldots, -0.1\}$. The SpaCE plot visualizes the results for varying the hyperparameter $\beta \in \{1, 10, 100\}$.

Table 1: Comparative evaluation of SpaCE-VAE with baselines and state-of-the-art attribution methods on the test set. The best performing XAI models are highlighted in bold.

			Acc ↑	Conf ↑	L_0-Sparsity ↑	L_1-Sparsity ↑
Baselines	Original Images		0.92	0.94	0%	0%
	VQ-VAE		0.56	0.56	-0.18%	-0.88%
XAI	DeepLIFT	$\sigma = -0.2$	0.35	0.32	8.05%	8.95%
	Integrated Gradients	$\sigma = -0.2$	0.29	0.27	9.28%	11.250%
	Feature Ablation	$\sigma = -0.3$	0.65	0.64	7.40%	9.52%
	SpaCE-VAE	$\beta = 1$	0.70	0.66	**21.86%**	**56.42%**
		$\beta = 10$	0.74	0.69	10.37%	53.56%
		$\beta = 100$	**0.76**	**0.72**	3.62%	42.01%

class prediction.

We evaluate SpaCE-VAE against three competitors that identify pixels that contribute negatively to the predicted class. Two of those competitors (DeepLIFT and Integrated Gradients) have no sensible method to evaluate those explanations, since *removing* the negatively contributing pixels often changes the predicted class. The other competitor, Feature Ablation, is a computationally expensive trial-and-error approach that identifies negative attributions based on an actual increase in the prediction confidence when pixels are removed.

Our evaluation based on the CIFAR10 dataset shows that SpaCE-VAE is vastly able to outperform the three competitors in the achieved accuracy when removing the pixel intensity of negative attributions. Our qualitative evaluation shows that SpaCE-VAE is able to provide interesting and insightful explanations.

Our proposed framework is flexible, and simple changes in the loss function enable the identification of various explanations. For example, we could incorporate a regularization term to increase the contrast of the explanations, to decrease the number of used colors, or to smooth the color variations as much as possible.

6. REFERENCES

[1] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS ONE*, 2015.

[2] N. Baker, H. Lu, G. Erlikhman, and P. J. Kellman. Deep convolutional networks do not classify based on global object shape. *PLoS computational biology*, 2018.

[3] L. Cunha, C. Soares, A. Restivo, and L. F. Teixeira. Gasten: Generative adversarial stress test networks. In *International Symposium on Intelligent Data Analysis (IDA)*, 2023.

[4] J. da Costa Feitosa, M. Roder, J. P. Papa, and J. R. F. Brega. Influence of pixel perturbation on explainable artificial intelligence methods. In *VISIGRAPP : VISAPP*, 2024.

[5] P. Dabkowski and Y. Gal. Real time image saliency for black box classifiers. In *Neural Information Processing Systems*, 2017.

[6] R. Dwivedi, D. Dave, H. Naik, S. Singhal, R. Omer, P. Patel, B. Qian, Z. Wen, T. Shah, G. Morgan, et al. Explainable ai (xai): Core ideas, techniques, and solutions. *ACM Computing Surveys*, 2023.

[7] R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. *IEEE In-*

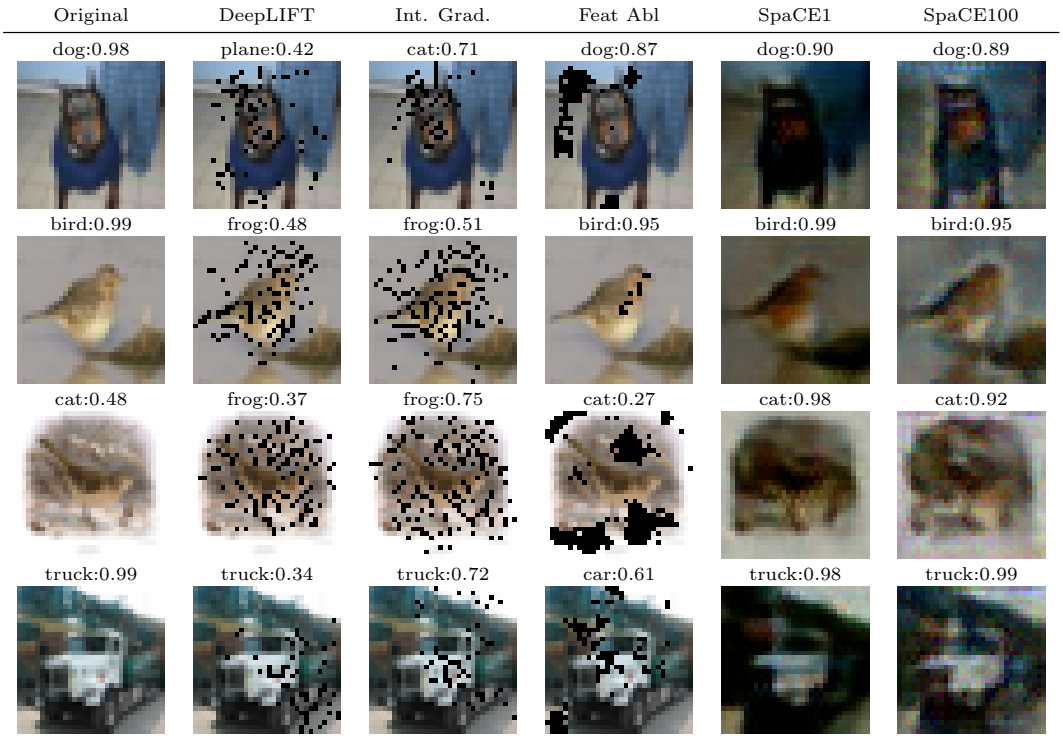


Figure 5: Original images, their explanation, and the predicted class with confidence. For competitors, we apply the binary mask for thresholds indicated in Table 1. SpaCE1 uses a value of $\beta = 1$ and SpaCE100 uses a $\beta = 100$.

ternational Conference on Computer Vision (ICCV), 2017.

[8] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *7th International Conference on Learning Representations (ICLR)*, 2019.

[9] N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan, et al. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*, 2020.

[10] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[11] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1967.

[12] C. Molnar. *Interpretable Machine Learning*. 3 edition, 2025.

[13] G. Nguyen, D. Kim, and A. Nguyen. The effectiveness of feature attribution methods and its correlation with automatic evaluation scores. *Advances in Neural Information Processing Systems*, 2021.

[14] S. Rao, M. Böhle, and B. Schiele. Towards better understanding attribution methods. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[15] A. Razavi, A. Van den Oord, and O. Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 2019.

[16] Y. Rong, T. Leemann, V. Borisov, G. Kasneci, and E. Kasneci. A consistent and efficient evaluation strategy for attribution methods. In *International Conference on Machine Learning*, 2022.

[17] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, 2017.

[18] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[19] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, 2017.

[20] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations (ICLR)*, 2014.

[21] A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 2017.

[22] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision - ECCV*, 2014.

Explaining Embedding-Based Matching of Hand-Drawn Binary Symbols with Grad-CAM: A Case Study on Cattle Brands

Leandra Alves Soares
Institute of Physics, UFG
Goiânia, Goiás, Brazil
leandra.physicsengineering@gmail.com

Marcos Vinícius S. Medeiros
Institute of Informatics, UFG
Goiânia, Goiás, Brazil
marcosvsatil@gmail.com

Aldo A. Díaz-Salazar
Institute of Informatics, UFG
Goiânia, Goiás, Brazil
aldo.diaz@ufg.br

ABSTRACT

Cattle-brand identification poses unique challenges due to the sparse, high-contrast nature of symbolic marks and the large stylistic gap between clean reference templates and hand-drawn queries. Since a prediction layer cannot be applied reliably in this setting, weperform identification through feature extraction and similarity search using a VGG-16 encoder and a FAISS-based retrieval system. weinvestigate how the encoder represents these binary symbols by combining channel-level activation statistics, Grad-CAM saliency maps, and Top-10 retrieval results. Our analysis shows that a small group of filters, especially an outer-contour detector, dominates the embedding space, leading to systematic underrepresentation of internal strokes, multi-component patterns, and thin or rotated structures. By comparing the saliency profiles of each query and its reference template, we observe how spatial contribution mismatches propagate into ranking errors. The study provides a concise, reproducible framework for auditing CNN feature extraction in symbolic CBIR tasks and highlights structural vulnerabilities relevant to livestock identification, logo retrieval, and other domains involving sparse binary imagery.

1. INTRODUCTION

Convolutional Neural Networks (CNNs) are widely used in supervised visual recognition, in which the prediction layer provides a natural point from which decisions can be interpreted through weights, gradients, or class scores [6]. However, this paradigm does not apply to problems where no shared classes exist, as in cattle brand identification, where each symbol is unique and cannot be treated as a repeated category. In such cases, the lack of labeled samples makes it impossible to train a classifier, rendering any form of logit-based or class-based explainability infeasible.

Under these structural constraints, the problem must be formulated as a Content-Based Image Retrieval (CBIR) task. Instead of prediction, a CNN is used as a feature extractor, producing spatial embeddings that can be compared in a metric space. The search is performed using Facebook Artificial Intelligence Similarity Search (FAISS), which returns a ranked list of the most similar brands in the database [15]. This approach scales well to databases containing thousands of distinct symbols, but shifts the interpretability challenge to a more fundamental level: how can we interpret decisions that emerge solely from the embedding space?

The difficulty increases when the queries are hand-drawn sketches, whereas the registered brand images are clean and binarized. Small variations in stroke thickness, proportions, internal connections, or rotations introduce topological distortions that significantly alter the position of the corresponding embedding. As a result, seemingly minor visual differences can drastically change the returned ranking, making it hard to understand why certain brands appear in the top positions while others do not.

To address this issue, we investigate interpretability at the feature-extraction level, combining three complementary perspectives:

1. **Gradient-weighted Class Activation Mapping (Grad-CAM) applied to the convolutional extractor:** highlighting the regions that contribute most to the construction of the embedding [14];
2. **Structural difference maps between the binarized template and the corresponding hand-drawn sketch:** revealing how drawing distortions visually affect the relevant patterns;
3. **Analysis of the top-10 results returned by FAISS [5]:** identifying which geometric patterns are favored by the embedding structure and which types of errors occur systematically.

Our findings demonstrate that convolutional activations primarily concentrate on the external contours of brands, while internal details such as connections and complex structures receive limited representation in the embeddings. This selectivity precisely explains why minor internal modifications in sketches disproportionately impact the ranking: since the embedding captures only partial visual semantics (prioritizing global silhouettes), any changes to underrepresented internal elements generate substantial variations in calculated similarity. In other words, the system becomes overly sensitive to internal changes precisely because it fails to encode them adequately in its representations, focusing excessively on global features at the expense of distinctive details. The top-10 analysis corroborates this behavior, showing that brands with similar silhouettes but different internal structures are frequently grouped together, confirming the selectivity in visual representation.

By unifying Grad-CAM, structural comparison, and behavioral analysis of the ranking, we present a practical methodology for auditing and interpreting CNN-based CBIR systems in high-contrast binary input domains. The approach contributes to understanding and diagnosing of visual retrieval systems applied to cattle brands. Nevertheless, it can be extended to analogous contexts such as logos, industrial markings, and historical documents.

2. RELATED WORK

Explainable AI (XAI) for image models has historically developed around classification, where logits, class scores, and discriminative gradients provide natural entry points for interpretation. Prototype-based methods leverage this structure by learning representative visual patterns that resemble actual training samples. Neural Prototype Trees integrate differentiable prototypes with hierarchical decision paths [11], whereas hierarchical prototype systems organize learned exemplars into semantic layers that approximate human reasoning [7]. These approaches depend heavily on the visual density and redundancy found in natural images. As a result, they do not transfer well to sparse, high-contrast domains such as binary hand-drawn symbols, where semantic content is encoded in minimal geometric arrangements rather than textured regions.

A second major line of work focuses on activation-map explanations, which highlight the spatial regions that contribute most to a model's response. Class Activation Mapping (CAM) [20] first demonstrated how to combine channel activations with learned class weights to produce coarse localization maps. Grad-CAM [14] generalized this principle by using gradients flowing into the final convolutional layer to generate class-specific saliency, becoming a standard tool due to its simplicity and architectural compatibility. Comparative studies show that Grad-CAM strikes a practical balance between fidelity and interpretability in spatial reasoning tasks [3]. Domain-specific applications reinforce this trend: in medical imaging, saliency maps are aligned with anatomical regions for diagnostic validation [9], and in handwritten character recognition, saliency-based and layer-wise relevance methods reveal where models confuse shape-level invariances with noise [16, 19]. Together, these works suggest a recurring pattern: symbolic and abstract visual domains require tailored interpretability protocols, because their semantics rely on small structural cues rather than photorealistic features.

Parallel to XAI, a substantial body of research in CBIR focuses on similarity rather than classification. Classical retrieval relied on local descriptors such as SIFT [8] and SURF, which encode keypoint-based geometry and support metric comparisons. More recent systems employ deep architectures to generate embeddings, continuous vector representations intended to preserve semantic proximity in a metric space. Advances in metric learning and representation learning have shown that CNN-based embeddings, Siamese networks, and contrastive objectives yield robust retrieval performance across a range of visual tasks [13]. For large-scale applications, efficient vector indexing becomes essential. FAISS is now an industry standard, offering optimized implementations of product quantization, inverted-file indexing, Hierarchical Navigable Small World (HNSW) graphs, and GPU-accelerated k-NN search [5]. Such systems power large-scale retrieval engines across vision, recommendation, and multimodal search.

3. METHODOLOGY

This study examines the interpretability of convolutional feature extraction in binary symbolic retrieval, using cattle brand images as a focused case study. Our goal is to identify which visual structures most strongly influence the embedding space generated by a CNN and how these structures propagate through a FAISS-based similarity pipeline. The methodology consists of four components: reference-data construction, preprocessing, feature-extraction modeling, and similarity-driven retrieval with post-hoc explainability.

3.1 Similarity-Based Retrieval Pipeline

Our retrieval architecture follows a two-phase CBIR workflow based on convolutional embeddings and FAISS vector indexing, following the methodology originally introduced in [10], and summarized in Fig. 1.

3.1.1 Phase 1: Reference-Database Construction

The reference images are created through a standardized workflow consisting of:

- **Preprocessing:** Binarization, size normalization, and aspect-preserving padding are applied to remove background noise (pelage, skin texture, burn artifacts) and to enhance stroke contrast, which is crucial for accurate symbol matching;
- **Feature Extraction:** A pretrained CNN encoder generates a high-dimensional embedding that encodes the spatial structure of the brand;
- **Storage**: Both the processed image and its embedding are stored in the reference database;
- **Indexing:** All embeddings are inserted into a FAISS index using either a flat L2 structure or an approximate index such as Inverted File Index (IVF) or HNSW [5], enabling millisecond-scale nearest-neighbor retrieval.

3.1.2 Phase 2: Query Processing and Retrieval

The query images are likewise processed through a standardized workflow consisting of:

- **Preprocessing and Encoding:** The same normalization and CNN encoding pipeline is applied to the hand-drawn query, ensuring embedding consistency with the reference database;
- **Similarity Search:** FAISS performs approximate k-NN retrieval using cosine similarity or normalized Euclidean distance as the metric;
- **Ranking:** The retrieved neighbors are sorted to produce a Top-k similarity list, which serves as the basis for all explainability analyses.

This architecture allows incremental database expansion: new brands can be indexed without retraining.

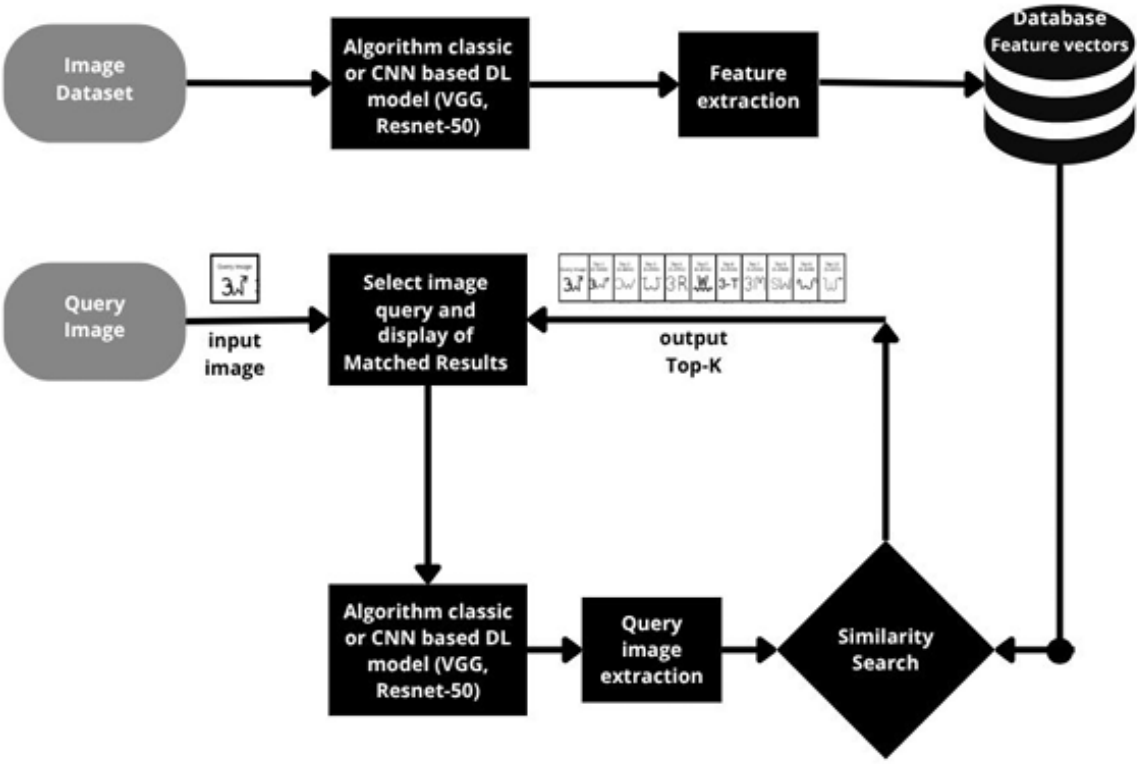


Figure 1: Workflow of the similarity-based retrieval pipeline, adapted from the methodology originally introduced in [10].

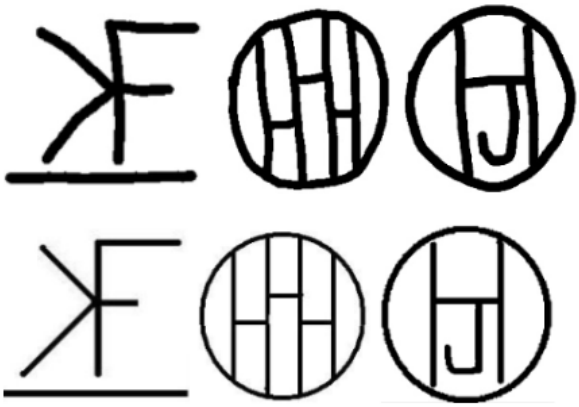


Figure 2: Comparison between the hand-drawn dataset (up) and the corresponding sample from the standard dataset (down).

3.2 Dataset and Preprocessing

To minimize confounding visual factors and ensure a consistent symbolic domain, all images are converted into binary masks that emphasize stroke geometry. We use two datasets, the standard dataset and the hand-drawn dataset, as shown in Fig. 2.

3.2.1 Reference Dataset

We collected 5,230 binary brand templates from publicly available government registries, including the British Columbia BrandBook [4], the Tennessee Agricultural Database [12], and the Oklahoma Brand Registry [1]. We removed duplicates, corrupted images, and overly complex symbols with many disconnected components or text-like structures.

3.2.2 Hand-Drawn Query Dataset

We collected 1,454 hand-drawn sketches captured on touchscreen devices at 640×640 px. Pen widths ranged from 30–60 px to simulate natural human variation and distortions such as rotation, irregular curvature, and junction imprecision.

3.2.3 Preprocessing Pipeline

All images from both datasets were processed using a standardized pipeline consisting of:

- **Binarization:** grayscale conversion and fixed thresholding;
- **Resizing:** scale-and-pad to 200×200 px while preserving aspect ratio;
- **Normalization:** rescaling to $[0, 1]$ and ImageNet-standard normalization.

3.2.4 Sample Selection

From the 1,454 sketches, we selected 304 representative samples using Cochran's formula (95% confidence, $p = 0.5$) [2]. Additionally, the 50 lowest-performing queries (by Top-10 accuracy) were selected for targeted failure analysis.

3.3 Feature-Extraction Model

3.3.1 Network Architecture

We adopt VGG-16 pretrained on ImageNet, truncated at the `block5_conv3` layer. This configuration balances high-level semantic capacity with spatial resolution. The resulting $7 \times 7 \times 512$ activation tensor is flattened and L_2-normalized into a 25,088-dimensional embedding [17].

3.3.2 Similarity Computation

Euclidean distances are computed between query and reference embeddings. We report Top-k accuracy for $k \in \{1, 5, 10, 100\}$ [18].

3.4 Explainability Analysis

Because retrieval depends exclusively on embeddings, interpretability must address how convolutional activations shape the geometry of the embedding space. We focus on identifying which image regions influence the embedding and how distortions propagate into the FAISS neighborhood.

3.4.1 Grad-CAM

We apply Grad-CAM [14] to the `block5_conv3` layer. For feature map A^k and target score y, Grad-CAM computes

$$\alpha_k = \frac{1}{Z} \sum_{i,j} \frac{\partial y}{\partial A^k_{i,j}}, \tag{1}$$

$$\text{CAM} = \text{ReLU}\left(\sum_k \alpha_k A^k \right), \tag{2}$$

where Z denotes the number of spatial locations. The ReLU operation selects positively contributing regions, revealing which contours dominate the embedding.

3.4.2 Analysis Procedure

For each query–reference pair, in both the query and the standard datasets, we apply the following methodology:

1. Grad-CAM heatmaps are generated for both images;
2. Heatmaps are aligned with semantic stroke elements (loops, intersections, endpoints);
3. Differences between query and reference activation patterns are examined to identify distortions influencing the embedding;
4. FAISS Top-10 neighborhoods are inspected to determine whether confusion arises from structural similarity, stroke omission, rotation, or filter bias.

3.4.3 Integration of Results

We obtain a coherent understanding of how the CNN prioritizes contours, and how these priorities shape the embedding space, by combining retrieval metrics, Grad-CAM activation patterns, and qualitative inspection of FAISS neighbors. This integrated perspective allows us to identify failure modes and guide robustness improvements for symbolic, high-contrast domains.

4. RESULTS OF ANALYSIS OF CHANNEL ACTIVATIONS

We systematically analyzed which channels in block5_conv3 were most frequently highlighted as the primary activations across the 304 Grad-CAM heatmaps from our test queries. Table 1 reports the filters that appeared most consistently as dominant features in these visual explanations, with 3 providing their characteristic activation patterns.

A striking concentration of activity emerged: only 10% of channels accounted for more than 90% of all above-threshold activations. This strong imbalance indicates that the embedding space is effectively shaped by a small subset of convolutional filters, each capturing coarse or highly localized geometric cues.

Channel 230 dominated with activations in 59.21% of all queries. Its behavior resembles a generic contour detector that fires on the external boundaries of most symbols. Because outer silhouettes are the most consistent visual feature across the dataset, this filter disproportionately influences the final embedding, propagating its bias directly into the FAISS similarity rankings.

Channels 336, 14, and 454 behaved as specialized detectors. Their activation maps consistently emphasized (i) curved strokes, (ii) line-intersection zones, and (iii) enclosed internal regions. These structural features correspond to the portions of a mark where human variability is highest and where hand-drawn distortions more strongly affect the embedding. However, these filters were activated far less frequently; their low influence suggests that internal structures contribute minimally to the overall embedding geometry.

Channels 429 and 142 appeared exclusively in correctly matched queries. Both capture high-level structures, large-radius curves and multi-component arrangements, that tend to preserve identity even under moderate drawing variation. Their absence in rejected queries suggests that missing or distorted internal details lead the embedding to collapse onto overly generic contour shapes.

Finally, 41.2% of all channels were never activated above threshold for any sample, indicating considerable redundancy. This redundancy also means that the effective embedding dimensionality is far lower than the nominal 25,088 dimensions, which makes the retrieved neighborhoods more sensitive to the few channels that consistently fire.

Together, these findings indicate that the embedding space, and therefore the FAISS ranking, is dominated by contour-focused filters, with only minimal contribution from internal topology.

5. ANALYSIS OF PIXEL WIDTH AND ROTATION SENSITIVITY

A second set of analyses examined how geometric distortions influence embedding stability. Because FAISS retrieves neighbors based on vector distance, any transformation that shifts the embedding significantly will directly alter the Top-k neighborhood.

Table 1: Frequency of Channel Activations with 7% Rejected Queries.

Channel	Total Activations	Approved Queries	Rejected Queries	Activation Pattern	Example Patch
230	180	168	12	Heat around symbol borders	
336	40	34	6	Curved strokes (up to two separate curves)	
14	24	22	2	Line intersection zones	
454	24	22	2	Enclosed regions between elements	
429	18	18	0	Large-radius curves only in approved queries	
142	18	18	0	Complex symbols with multiple components	
Total	**304**	**283**	**21**		

5.1 Stroke Thickness

Hand-drawn strokes originally ranged from 30–60 px. After preprocessing and resizing, samples below the 30 px threshold showed severe accuracy drops: Top-10 accuracy fell from 74% to 48% for strokes near 20 px. Thin strokes weaken contour detectors such as Channel 230 and reduce the firing of specialized filters such as Channels 14 and 454, flattening the embedding into a more uniform or ambiguous region of the space. As a result, FAISS retrieves neighbors dominated by coarse silhouettes rather than fine-grained topological matches.

5.2 Rotation Sensitivity

Table 2 presents matching accuracy under systematic rotations of the reference set. Accuracy dropped from 100% at 0° to 16.67% at 45°. This symmetric degradation around 0° demonstrates that the representation learned by VGG-16 is not rotation invariant, despite the apparent simplicity of the binary domain.

From the perspective of the embedding space, rotation alters the spatial alignment of strokes relative to convolutional filters. As a result, activations shift unpredictably across channels, causing embeddings to drift toward unrelated regions. These geometric shifts propagate directly into FAISS's nearest-neighbor search, leading to mismatched returns even when symbols remain semantically identical. Thus, rotation and thinning systematically break the stability of the embedding, exposing a structural weakness of the feature extractor.

6. ANALYSIS OF COMPLEX PATTERNS

Complex cattle brands, those with tertiary elements such as internal loops, secondary connectors, or small decorative components, were rarely represented coherently in the activation maps. Even the most frequently activated filters did not cover complex marks holistically. Instead, activation maps captured only fragments of the symbol:

- Channel 230: outer contour only;
- Channel 336: single or double curves;
- Channel 14: line intersections;
- Channel 454: enclosed subregions.

As a result, marks with hierarchical or multi-component structure were reduced to incomplete embeddings dominated by the silhouette, causing FAISS to retrieve neighbors with similar outlines but different internal topology.

For example, when tiny embellishments near junctions or additional letters were present (refer to 3, Channel: 454 and 454), their absence or deformation in the hand-drawn query suppressed activation in specialized channels, shifting the embedding toward coarse matches.

These observations confirm a broader pattern: the feature extractor encodes symbols through a sparse sampling of visual attributes, primarily contours, while ignoring distinctive internal elements necessary for disambiguation. This explains the recurrent FAISS errors observed for complex brands.

7. QUALITATIVE ANALYSIS

This section provides a compact overview of the qualitative differences observed across the analyzed channels as summarized in Table 3. For each channel, we include the Grad-CAM of the original cattle brand, the Grad-CAM of the hand-drawn query, and the Top-10 retrieved results used in the retrieval experiment. The textual description summarizes how structural differences between the original and hand-drawn inputs translate into shifts in activation patterns and retrieval performance.

8. IMPLICATIONS

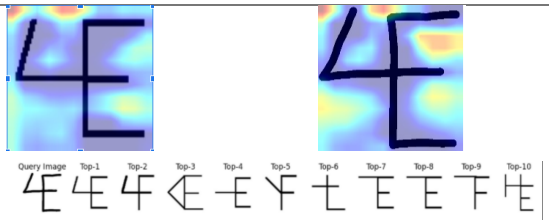
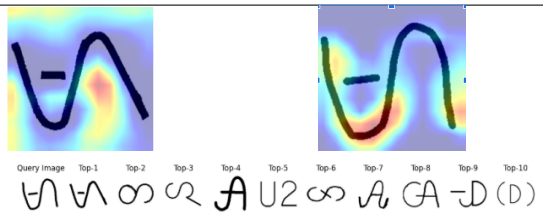
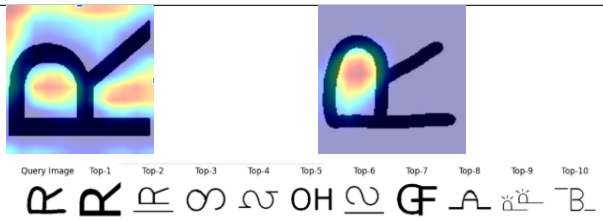
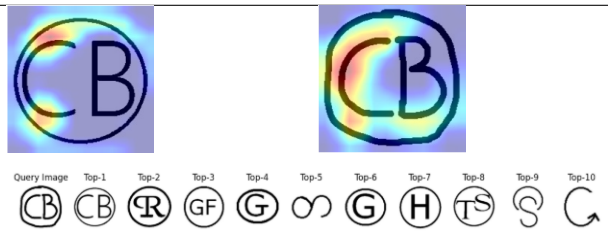
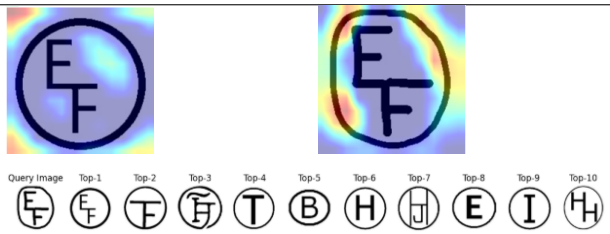
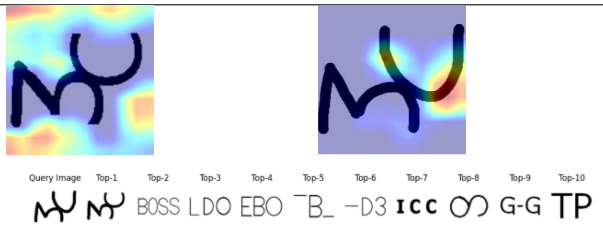
Our results reveal a structural property of the VGG-16 feature extractor: a small set of contour-dominant filters disproportionately shapes the embedding space, while detailed internal structure is systematically underrepresented. This has three major implications:

1. **Embedding geometry is coarse and contour-centric:** because most channels never activate or activate only weakly, the resulting embeddings cluster

Table 2: Matching accuracy of the CNN at different rotation increments.

Algorithm	-45º	-30º	-15º	0º	15º	30º	45º
VGG-16	18.52%	46.30%	98.15%	100.00%	99.04%	48.15%	16.67%

Table 3: Summary of qualitative outcomes across analyzed channels.

Channel	Original	Query	Description
230			The hand-drawn query and original image share similar large-scale contour structures. Both Grad-CAM maps highlight outer-edge saliency typical of Channel 230, explaining the correct top-ranked retrieval.
336			Original emphasizes contour flow (230-like), whereas the query activates curved corners (336-like). Still, the correct match remains among the top results.
14			The query is thinner and less precise; saliency shifts from outer contours (230) to internal enclosed regions (14), especially inside the letter "R".
454			Original activates Channel 429 along circular contours; query concentrates saliency between shapes. Top-10 includes similar circle–letter structures.
429			Original dominated by diffuse 230-like patterns; query shifts toward internal activation clusters around the letter "E".
142			Both images share structure, but saliency shifts: original shows diffuse 230-like behavior, while the query emphasizes boundary activation along the letter "C".

marks based on silhouette similarity, irrespective of internal differences;

2. **FAISS neighborhoods reflect filter biases:** The nearest neighbors returned by FAISS mirror the lim-

ited visual vocabulary of the embeddings. Errors arise not from the FAISS mechanism, but from upstream representational collapse;

3. **CNNs struggle with sparse, high-contrast symbols:** VGG-16 performs well on simple datasets like MNIST, but its contour bias and lack of rotational invariance severely limit its use in complex brand patterns.

9. CONCLUSION

This work demonstrates that, when applied to binary cattle-brand recognition, the VGG-16 feature extractor collapses a rich symbolic domain into a narrow, contour-centric embedding space. Through combined analysis of channel activations, Grad-CAM visualizations, and FAISS retrieval behavior, we showed that internal structures, junctions, enclosed regions, and tertiary elements are consistently underrepresented in the embeddings. These representational gaps distort the geometry of the vector space, causing FAISS to retrieve neighbors based primarily on silhouette similarity, and performance deteriorates sharply under stroke thinning, geometric distortion, and rotation. To correct these structural weaknesses, training should incorporate controlled rotations, stroke thinning, geometric jitter preserving topology, and domain-specific contrast augmentations. Additional architectural adjustments, such as adapter blocks, attention over late-layer activations, or channel-balancing modules, could further distribute representational load across channels. Ultimately, the diagnostic methodology introduced here provides a reproducible framework for interpreting embedding-based CBIR systems and highlights the need for domain-tailored augmentations and architectural refinements. These findings underscore that explainability in retrieval must be built at the feature-extraction level, where the embedding itself is formed, rather than at the prediction layer, because in CBIR the prediction layer does not exist.

10. ADDITIONAL AUTHORS

Edmundo Hoyle, Center of Excellence in Artificial Intelligence (CEIA), edhoyle@gmail.com.

11. REFERENCES

[1] O. C. Association. Oklahoma brand registration. `https://www.okcattlemen.org/brands`. Accessed: 2025-05-30.

[2] W. G. Cochran. *Sampling Techniques*. John Wiley & Sons, 3rd edition, 1977.

[3] R. ElShawi, Y. Sherif, M. Al-Mallah, and S. Sakr. Interpretability in healthcare: A comparative study of local machine learning interpretability techniques. *Computational Intelligence*, 36(1):235–249, 2020.

[4] O. I. Inc. British columbia livestock brand database (brandbook). `https://www.ownershipid.ca/brandbook`. Accessed: 2025-05-30.

[5] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with FAISS. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1097–1105, 2012.

[7] X. Li et al. Hierarchical prototype-based explanations. *Transactions on Machine Learning Research*, 2024.

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[9] A. S. Lundervold, A. B. Arrieta, et al. Explainable ai: A review of applications to neuroimaging data. *Frontiers in Neuroscience*, 16:906290, 2022.

[10] M. V. S. Medeiros, L. A. Soares, E. Hoyle, and A. A. Díaz-Salazar. Visual similarity search of cattle brands using deep learning on binary representations. In *2025 38th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 1–6, Salvador, Brazil, 2025.

[11] M. Nauta, R. van Bree, and C. Seifert. Neural prototype trees for interpretable fine-grained image recognition. In *CVPR*, pages 14933–14943, 2021.

[12] T. D. of Agriculture. Tennessee agricultural brand database. `https://agriculture.tn.gov/ListBrand.asp`. Accessed: 2025-05-30.

[13] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

[14] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, pages 618–626, 2017.

[15] J. Silva, B. Pereira, and L. Santos. Segmentation and detection of cattle branding images using cnn and svm. *Journal of Agricultural Informatics*, 8(2):45–53, 2017.

[16] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. 2013.

[17] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015.

[18] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[19] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833, 2014.

[20] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2921–2929, 2016.

A Non-Parametric Bayesian Approach Towards Online Sequence Learning

Stiven S. Dias
Embraer S.A.
R&D Team
Av. Brig. Faria Lima, 2170
S. J. dos Campos, Brazil
stiven.dias@embraer.com.br

Marcelo G. S. Bruno
Instituto Tecnológico de Aeronáutica
Pç. Mar. Eduardo Gomes, 50
S. J. dos Campos, Brazil
bruno@ita.br

Alberto F. De Souza
Universidade Federal do Espírito Santo
Av. Fernando Ferrari, 514
Vitória, Brazil
alberto@inf.ufes.br

ABSTRACT

Forecasting the next sample of an incoming sequence of noisy observations is a recurring problem in several real-world applications. In general, the streamed observations may originate from a possibly non-stationary stochastic process with non-linear dynamics. Very often though, to avoid devising explicit probabilistic models from scratch, we alternatively learn data-driven surrogate models from past observations to approximate complex dynamics. Time-varying systems in turn pose additional hurdles as such models must be continuously updated to accommodate changes in their underlying behavior. In this paper, we introduce a novel grid-based filter designed to inherently deal with the continuous learning problem. Specifically, Bayesian-grounded procedures are employed to both recursively predict observations and incrementally build a suitable non-parametric, probabilistic model of the indirectly observed phenomenon. Moreover, we also present a one-shot learning, memory-based implementation of the proposed filter, a shallow weightless neural network which is able to efficiently store and retrieve associative input-output pairs. Lastly, we demonstrate the effectiveness of our filter in performing simple anomaly detection tasks.

1. INTRODUCTION

Time-series modeling is paramount for many applications such as anomaly detection and prediction [1], stock market forecast [31] and even network attack detection [8]. Moreover, Bayesian inference – e.g. through the Bayes filter – provides a well-grounded theoretical framework relying on probabilistic models to forecast incoming observations of a stochastic phenomenon. However, in several engineering applications, designing an explicit probabilistic model of the time-series by hand is inadvisable as the data distribution can be prohibitively complex and often drifts, changing significantly over time. The ability to continuously learn suitable probabilistic models directly from available observations without supervision thus has the potential to further unlock the application of the Bayesian paradigm to time-series forecast.

1.1 Related Work

Online sequence learning narrows down the range of suitable methods to timely predict future observations, since they are also required to simultaneously update their inner models of the temporal phenomenon using accumulated observations. Several methods exist for time-series prediction using different approaches [23, 25]. Few of them like [1, 30] are able to deal with the continuous learning (CL) problem, i.e. to build and improve a feasible model to accurately predict future observations as incoming observations arrive. Even fewer approaches though rely on non-parametric statistics [32, 28] to simultaneously learn models and predict observations [7,9], but, to the best of our knowledge, none of them have adapted the Bayes filter to recursively learn model's hyperparameters and jointly make predictions using a fully non-parametric, Bayesian inference approach as discussed in this paper.

Furthermore, shallow weightless neural networks (WNNs) [2] have been successfully employed to learn from data and solve challenging problems [13,14,12] as far as proper data encoding schemes [27] are employed. Despite their inherent ability for both one-shot learning and CL, so far, their few enthusiasts have not been able to effectively stack WNN layers into deeper networks to solve increasingly complex problems, as was the case for regular neural networks. On the other hand, although quite useful to successfully solve real-world, time-series forecasting problems [10, 3], deep neural models were not intrinsically designed for CL, as they often require an offline, expensive optimization procedure [5] with salting racks and workarounds [6] to timely / properly converge and adjust their multitude of parameters – e.g. weights and biases – using all accumulated training data so far; avoiding thus catastrophic forgetting issues [22].

An alternative, bio-inspired model which supports CL by design has been incrementally polished by Numenta researchers [19, 20]. However, despite their insightful findings [18] on how the underlying biological mechanisms – in special, biological neurons and layers within cortical cells – of the human neocortex interact with each other to create human intelligence as an emerging behavior, they did not provide so far a rigorous interpretation of their well-known hierarchical temporal memory (HTM) model [19] using e.g. the Bayesian inference framework.

In this paper, we model first the time-series CL and prediction problem using a fully Bayesian approach and propose therefore an explainable non-parametric, grid-based filter accompanied by an efficient, one-shot learning implementation based on a shallow WNN with two layers stacked in a meaningful, Bayesian interpretable way. Nevertheless, we also borrow some ideas from Cui et al. [11] and employ sparse representations over high-dimensional spaces to represent observed symbols.

1.2 Contributions and Paper Outline

The main contribution of this paper is twofold: it introduces i) a novel grid-based filter which allows one to perform inference on high-dimensional binary spaces and effectively predict incoming observations residing in continuous-valued spaces; and ii) an efficient grid-filter implementation based on virtual-generalized random access memory (VG-RAM) nodes [24] which enables time-series CL seamlessly and yields (auditable) outputs which are fully traceable to filter's inputs.

Paper Outline: The paper is divided into six sections. Sec. 1 is this Introduction. In Sec. 2, we state the online sequence learning and prediction problem under a Bayesian framework. In Sec. 3, we introduce our solution approach. Sec. 4 describes our associative random access memory (RAM)-based implementation of the solution proposed in Sec. 3. Simulations results for an application example are presented in Sec. 5. Finally, we offer our conclusions in Sec. 6.

1.3 Notation

We use uppercase letters, e.g. $\mathbf{X}$, to denote random vectors and lowercase letters, e.g. $\mathbf{x}$, to denote samples of random vectors. We also define the unconditional probability of an event A and the conditional probability of event A given the knowledge that an event B has occurred as $\Pr\{A\}$ and $\Pr\{A|\,B\}$, respectively. Moreover, we define the joint probability of events A and B occurring simultaneously as $\Pr\{A, B\}$.

We also employ the Iverson bracket – which takes as argument any logical proposition Λ – defined as $[\![\Lambda]\!] = 1$, if Λ is true, and $[\![\Lambda]\!] = 0$, otherwise. Furthermore, let

$$\mathbb{Z}_2^d \triangleq \{0,1\} \times \ldots \times \{0,1\}$$

denote an d-dimensional binary space, we define then the Hamming distance between two vectors $\mathbf{a} \triangleq \begin{bmatrix} a_1 & \ldots & a_d \end{bmatrix}^\top$ and $\mathbf{b} \triangleq \begin{bmatrix} b_1 & \ldots & b_d \end{bmatrix}^\top$ in $\mathbb{Z}_2^d$ as $d_H(\mathbf{a}, \mathbf{b}) \triangleq \sum_{i=1}^{d} [\![a_i \neq b_i]\!]$. The inner product between any two vectors $\mathbf{a}$ and $\mathbf{b}$ now in $\mathbb{R}^d$ is defined as $\langle \mathbf{a}, \mathbf{b} \rangle \triangleq \sum_{i=1}^{d} a_i \cdot b_i$ and the L^p norm of an arbitrary vector $\mathbf{c} \triangleq \begin{bmatrix} c_1 & \ldots & c_d \end{bmatrix}^\top \in \mathbb{R}^d$ is defined as $||\mathbf{c}||_p \triangleq \left(\sum_{i=1}^{d} |c_i|^p \right)^{\frac{1}{p}}$. Lastly, we define the one-hot indicator vector as $\mathbf{1}_j \triangleq \begin{bmatrix} \iota_{1,j} & \ldots & \iota_{d,j} \end{bmatrix}^\top \in \mathbb{Z}_2^d$ with $||\mathbf{1}_j||_1 = 1$ such that $\iota_{i,j} = [\![i = j]\!]$, $\forall i \in \{1, \ldots, d\}$.

Now, let $\mathbf{X}$ be a d-dimensional random vector whose realizations $\mathbf{x} \sim \mathbf{X}$ assume values in a finite sampling space $\boldsymbol{\Omega} \triangleq \{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(c)}\} \subseteq \mathbb{R}^d$ with cardinality $c = |\boldsymbol{\Omega}|$. We denote a categorical distribution over $\boldsymbol{\Omega}$ as $Cat(\boldsymbol{\Omega}; \boldsymbol{\pi})$, in which the c-dimensional parameter vector $\boldsymbol{\pi} \triangleq \begin{bmatrix} \pi_1 & \ldots & \pi_c \end{bmatrix}^\top \in \mathbb{R}_+^c$, $||\boldsymbol{\pi}||_1 = 1$, collects the probabilities of the c possible categories such that its probability mass function (p.m.f.) is given by $P(\mathbf{x}) \triangleq Pr\{\mathbf{X} = \mathbf{x}\} = \sum_{i=1}^{c} \pi_i \cdot [\![\mathbf{x} = \boldsymbol{\mu}^{(i)}]\!]$.

Moreover, let us further assume that $\boldsymbol{\pi}$, a valid p.m.f. over $\boldsymbol{\Omega}$, is a realization of a random vector $\boldsymbol{\Pi}$ in the $(c-1)$-dimensional simplex $\mathbb{S}_c$ distributed according to a Dirichlet distribution $Dir(c; \boldsymbol{\alpha})$ with hyperparameters collected by the vector $\boldsymbol{\alpha} \triangleq \begin{bmatrix} \alpha_1 & \ldots & \alpha_c \end{bmatrix}^\top$, $\alpha_i \in \mathbb{R}_+$, $\forall i \in \{1, \ldots, c\}$. We define the probability density function (p.d.f.) of the Dirichlet distribution over $\mathbb{S}_c$ as $p(\boldsymbol{\pi}) = (B(\boldsymbol{\alpha}))^{-1} \prod_{i=1}^{c} \pi_i^{\alpha_i - 1}$, where $B(\boldsymbol{\alpha}) \triangleq (\Gamma(\alpha_0))^{-1} \prod_{i=1}^{c} \Gamma(\alpha_i)$ is the Beta function with $\alpha_0 = \sum_{i=1}^{c} \alpha_i$ and $\Gamma(x) \triangleq \int_0^\infty t^{x-1} e^{-t} dt$ is the Gamma function with x in the set of non-negative real numbers $\mathbb{R}_+$.

2. PROBLEM STATEMENT

Let $\check{\mathbf{y}}_{0:\infty} \triangleq (\check{\mathbf{y}}_0, \check{\mathbf{y}}_1, \check{\mathbf{y}}_2, \ldots)$ be an *online* sequence of observations – e.g. a streamed sequence of dense feature vectors – in a continuous space $\boldsymbol{\Omega}_y \subset \mathbb{R}^{d_y}$ at discrete time instants $t_0 < t_1 < t_2 < \ldots$ Given the sub-sequence of observations $\check{\mathbf{y}}_{0:n} \triangleq (\check{\mathbf{y}}_0, \ldots, \check{\mathbf{y}}_n)$ up to instant n, our ultimate goal is to predict, at instant n, the next observation $\check{\mathbf{y}}_{n+1} \triangleq \begin{bmatrix} \check{y}_{n+1,1} & \ldots & \check{y}_{n+1,d_y} \end{bmatrix}^\top \in \boldsymbol{\Omega}_y$.

2.1 Bayesian formulation

Now, let $\mathbf{m}_{y \to z} : \boldsymbol{\Omega}_y \mapsto \boldsymbol{\Omega}_z$ denote a *known* encoding function which maps each observation $\check{\mathbf{y}}_n$ into a sparse representation $\check{\mathbf{z}}_n \triangleq \begin{bmatrix} \check{z}_{n,1} & \ldots & \check{z}_{n,d_z} \end{bmatrix}^\top$ within a high-dimensional binary space $\boldsymbol{\Omega}_z \subset \mathbb{Z}_2^{d_z}$ such that $d_z \gg d_y$ and $d_z \gg ||\check{\mathbf{z}}_n||_1$. Similarly, let $\mathbf{m}_{z \to y} : \boldsymbol{\Omega}_z \mapsto \boldsymbol{\Omega}_y$ define a *known* decoding function mapping arbitrary encoded observations $\mathbf{z}_n \in \boldsymbol{\Omega}_z$ back to observations $\mathbf{y}_n \in \boldsymbol{\Omega}_y$.

As illustrated in Fig. 1, let us further suppose that any particular sequence of encoded observations $\check{\mathbf{z}}_{0:n} \triangleq (\check{\mathbf{z}}_0, \ldots, \check{\mathbf{z}}_n)$ up to instant n is emitted by a Markovian process with an underlying sequence of hidden states $\mathbf{s}_{0:n} \triangleq (\mathbf{s}_0, \ldots, \mathbf{s}_n)$ within a high-dimensional binary space $\boldsymbol{\Omega}_s \subset \mathbb{Z}_2^{d_s}$, $\forall n \geq 0$, $d_s \gg ||\mathbf{s}_n||_1$, such that the online sequence of output observations can be written as

$$\check{\mathbf{y}}_{0:n} \equiv (\mathbf{m}_{z \to y}(\check{\mathbf{z}}_0), \ldots, \mathbf{m}_{z \to y}(\check{\mathbf{z}}_n)) \,. \tag{1}$$

Note that, as shown in Fig. 1, the sequences $\check{\mathbf{y}}_{0:n}$, $\check{\mathbf{z}}_{0:n}$ and $\mathbf{s}_{0:n}$ are realizations of the sequences of random vectors $\mathbf{Y}_{0:n} \triangleq (\mathbf{Y}_0, \ldots, \mathbf{Y}_n)$, $\mathbf{Z}_{0:n} \triangleq (\mathbf{Z}_0, \ldots, \mathbf{Z}_n)$ and $\mathbf{S}_{0:n} \triangleq (\mathbf{S}_0, \ldots, \mathbf{S}_n)$, respectively.

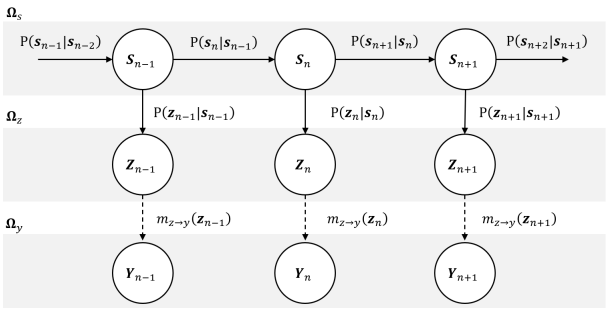


Figure 1: Hidden Markov chain representing the observed stochastic process.

Next, conditioned on the sequence of observations $\check{\mathbf{z}}_{0:n-1}$ up to instant $n-1$, the random state vector $\mathbf{S}_n$ at instant n is distributed according to the conditional p.m.f.

$$P(\mathbf{s}_n \,|\, \check{\mathbf{z}}_{0:n-1}) \triangleq Pr\{\mathbf{S}_n = \mathbf{s}_n \,|\, \mathbf{Z}_0 = \check{\mathbf{z}}_0, \ldots, \mathbf{Z}_{n-1} = \check{\mathbf{z}}_{n-1}\}. \tag{2}$$

Since, by construction, the observed process is first order Markovian and the observations are not correlated over time, the following usual assumptions hold respectively for the likelihood function and the state transition p.m.f.

$$P(\check{\mathbf{z}}_n \,|\, \mathbf{s}_{0:n}, \check{\mathbf{z}}_{0:n-1}) \equiv P(\check{\mathbf{z}}_n \,|\, \mathbf{s}_n) \tag{3}$$

$$P(\mathbf{s}_{n+1} \,|\, \mathbf{s}_{0:n}, \check{\mathbf{z}}_{0:n}) \equiv P(\mathbf{s}_{n+1} \,|\, \mathbf{s}_n). \tag{4}$$

Thus, upon the arrival of the encoded observation $\check{\mathbf{z}}_n = \mathbf{m}_{y \to z}(\check{\mathbf{y}}_n)$ at instant n, the predicted marginal posterior $P(\mathbf{s}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$ can be recursively computed from the predicted

marginal posterior $P(\mathbf{s}_n \mid \check{\mathbf{z}}_{0:n-1})$ in (2) – built from the encoded observations $\check{\mathbf{z}}_{0:n-1}$ up to instant $n-1$ – as

$$P(\mathbf{s}_n \,|\, \check{\mathbf{z}}_{0:n}) \propto P(\check{\mathbf{z}}_n \,|\, \mathbf{s}_n)\, P(\mathbf{s}_n \,|\, \check{\mathbf{z}}_{0:n-1}) \tag{5}$$

$$P(\mathbf{s}_{n+1} \,|\, \check{\mathbf{z}}_{0:n}) = \sum_{\mathbf{s}_n \in \boldsymbol{\Omega}_s} P(\mathbf{s}_{n+1} \,|\, \mathbf{s}_n)\, P(\mathbf{s}_n \,|\, \check{\mathbf{z}}_{0:n}) \tag{6}$$

with proportionality constant in (5) given by the reciprocal of

$$P(\check{\mathbf{z}}_n \,|\, \check{\mathbf{z}}_{0:n-1}) = \sum_{\mathbf{s}_n \in \boldsymbol{\Omega}_s} P(\check{\mathbf{z}}_n \,|\, \mathbf{s}_n)\, P(\mathbf{s}_n \,|\, \check{\mathbf{z}}_{0:n-1}). \tag{7}$$

Eqs. (5) and (6) follow respectively by the straightforward application of the Bayes rule and the Chapman-Kolmogorov equation given assumptions (3) and (4).

Note also that, conditioned on the sequence of encoded observations $\check{\mathbf{z}}_{0:n}$ up to instant n, the random vector $\mathbf{Z}_{n+1}$ at instant $n+1$ is distributed according to the p.m.f.

$$P(\mathbf{z}_{n+1} \,|\, \check{\mathbf{z}}_{0:n}) = \sum_{\mathbf{s}_{n+1} \in \boldsymbol{\Omega}_s} P(\mathbf{z}_{n+1} \,|\, \mathbf{s}_{n+1})\, P(\mathbf{s}_{n+1} \,|\, \check{\mathbf{z}}_{0:n}). \tag{8}$$

We can compute therefore the maximum a posteriori (MAP) estimate of the next encoded observation $\check{\mathbf{z}}_{n+1}$ given all encoded observations $\check{\mathbf{z}}_{0:n}$ up to instant n as

$$\hat{\mathbf{z}}_{n+1|n} = \arg\max_{\mathbf{z}_{n+1} \in \boldsymbol{\Omega}_z} P(\mathbf{z}_{n+1} \,|\, \check{\mathbf{z}}_{0:n}) \tag{9}$$

and predict then the next output observation $\check{\mathbf{y}}_{n+1}$ at instant $n+1$ as

$$\hat{\mathbf{y}}_{n+1|n} \triangleq \mathbf{m}_{z \to y}(\hat{\mathbf{z}}_{n+1|n}). \tag{10}$$

In this paper, however, we consider that neither the observation model $P(\mathbf{z}_n \,|\, \mathbf{s}_n)$ nor the dynamic model $P(\mathbf{s}_n \,|\, \mathbf{s}_{n-1})$ is provided or known *a priori*. At first glance, this is an ill-defined problem to solve using the conventional, model-based Bayesian framework [21]. However, we introduce here a novel approach to incrementally build – as new, incoming observations are assimilated – non-parametric representations of $P(\mathbf{z}_n \,|\, \mathbf{s}_n)$ and $P(\mathbf{s}_n \,|\, \mathbf{s}_{n-1})$ which, as indicated in Eqs. (5) through (8), entail the required knowledge regarding the observed stochastic process to recursively compute the predicted posteriors $P(\mathbf{s}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$ and $P(\mathbf{z}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$ at instant n from the previous posterior $P(\mathbf{s}_n \,|\, \check{\mathbf{z}}_{0:n-1})$ built at instant $n-1$.

2.2 Problem breakdown

Our ultimate goal can be finally unfolded into two tasks: i) *continuously* learn suitable, non-parametric probabilistic models for $P(\mathbf{s}_n \,|\, \mathbf{s}_{n-1})$ and $P(\mathbf{z}_n \,|\, \mathbf{s}_n)$ to be able to recursively approximate, at each time instant n, the marginal posterior $P(\mathbf{z}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$ from the available data $\check{\mathbf{z}}_{0:n}$ as indicated in Eqs. (5) through (8), and ii) use the predicted marginal posterior $P(\mathbf{z}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$ given the observations $\check{\mathbf{z}}_{0:n}$ up to instant n to predict the next output observation $\check{\mathbf{y}}_{n+1}$ at instant $n+1$ as shown in Eqs. (9) and (10).

3. SOLUTION APPROACH

We define first the required overall structure of a suitable encoding-decoding (codec) scheme. In the sequel, we derive a novel grid-based filter to perform the aforementioned tasks. In particular, we employ a non-parametric, Bayesian approach to recursively predict the next observation and learn the model hyperparameters.

3.1 Codec structure

Let $\boldsymbol{\Omega}_z \triangleq \{\boldsymbol{\mu}_z^{(\ell)}\}$, $\ell \in \{1, \ldots, c_z\}$, be a set of sparsely encoded, binary vectors $\boldsymbol{\mu}_z^{(\ell)} \in \mathbb{Z}_2^{d_z}$ with cardinality $c_z \triangleq |\boldsymbol{\Omega}_z|$ such that

$$\forall \ell, \qquad ||\boldsymbol{\mu}_z^{(\ell)}||_1 \leq a_z$$
$$\forall \ell_1 \neq \ell_2, \qquad b_z \leq d_H(\boldsymbol{\mu}_z^{(\ell_1)}, \boldsymbol{\mu}_z^{(\ell_2)}) \leq 2a_z,$$

where $a_z \ll d_z$ is the maximum number of activated bits and b_z is the minimum Hamming distance between any two vectors in $\boldsymbol{\Omega}_z$. Thus, a_z controls the sparseness in $\boldsymbol{\Omega}_z$, whereas b_z determines the discriminability of its elements in terms of the Hamming distance. Moreover, let $\{\boldsymbol{\Omega}_y^{(\ell)}\}$, $\ell \in \{1, \ldots, c_z\}$, denote a set partition of the output observation space $\boldsymbol{\Omega}_y$ such that

$$\bigcup_{\ell=1}^{c_z} \boldsymbol{\Omega}_y^{(\ell)} = \boldsymbol{\Omega}_y$$

and its subsets $\boldsymbol{\Omega}_y^{(\ell)}$ are pairwise disjoint, i.e., $\forall \ell_1 \neq \ell_2$, $\boldsymbol{\Omega}_y^{(\ell_1)} \cap \boldsymbol{\Omega}_y^{(\ell_2)} = \emptyset$. We define then the following lossy, encoding function $\mathbf{m}_{y \to z} : \mathbb{R}^{d_y} \mapsto \{\boldsymbol{\mu}_z^{(\ell)}\} \subset \mathbb{Z}_2^{d_z}$

$$\begin{aligned} \mathbf{z} &= \mathbf{m}_{y \to z}(\mathbf{y}) \\ &\triangleq \sum_{\ell=1}^{c_z} \boldsymbol{\mu}_z^{(\ell)} [\![\mathbf{y} \in \boldsymbol{\Omega}_y^{(\ell)}]\!]. \end{aligned} \tag{11}$$

Note that $\{\boldsymbol{\mu}_z^{(\ell)}\}$, $\ell \in \{1, \ldots, c_z\}$ is an encoded alphabet. In particular, $\forall \ell \in \{1, \ldots, c_z\}$, $\boldsymbol{\mu}_z^{(\ell)}$ assigns a sparse, binary symbol to the set partition $\boldsymbol{\Omega}_y^{(\ell)}$ with a_z activated bits and at least b_z bits far from the remaining $c_z - 1$ symbols in $\boldsymbol{\Omega}_z \subset \mathbb{Z}_2^{d_z}$.

Note also that $\mathbf{m}_{y \to z}(\cdot)$ is by construction a deterministic, surjective function. We define therefore the following decoding function $\mathbf{m}_{z \to y} : \{\boldsymbol{\mu}_z^{(\ell)}\} \mapsto \mathbb{R}^{d_y}$

$$\begin{aligned} \mathbf{y} &= \mathbf{m}_{z \to y}(\mathbf{z}) \\ &\triangleq \sum_{\ell=1}^{c_z} \boldsymbol{\mu}_y^{(\ell)} [\![\mathbf{z} = \boldsymbol{\mu}_z^{(\ell)}]\!], \end{aligned} \tag{12}$$

in which $\boldsymbol{\mu}_y^{(\ell)}$ can be e.g. the geometric centroid of the set partition $\boldsymbol{\Omega}_y^{(\ell)}$. In this case, $\forall \ell \in \{1, \ldots, c_z\}$, we can assume that $\boldsymbol{\Omega}_y^{(\ell)}$ is convex by construction such that its centroid $\boldsymbol{\mu}_y^{(\ell)} \in \boldsymbol{\Omega}_y^{(\ell)}$. Lastly, we follow the lead in [29] and further assume that close observations in $\boldsymbol{\Omega}_y$ yield close symbols in $\boldsymbol{\Omega}_z$. In particular, if the Euclidean distance $||\mathbf{y} - \mathbf{y}'||_2 \leq ||\mathbf{y}'' - \mathbf{y}'''||_2$ for $\mathbf{y}, \mathbf{y}', \mathbf{y}'', \mathbf{y}''' \in \boldsymbol{\Omega}_y$, than the Hamming distance $d_H(\mathbf{m}_{y \to z}(\mathbf{y}), \mathbf{m}_{y \to z}(\mathbf{y}')) \leq d_H(\mathbf{m}_{y \to z}(\mathbf{y}''), \mathbf{m}_{y \to z}(\mathbf{y}'''))$.

3.2 Grid-based filter

Let the c_s-dimensional vector

$$\boldsymbol{\pi}_{n|n-1} \triangleq \begin{bmatrix} \pi_{n|n-1}^{(1)} & \cdots & \pi_{n|n-1}^{(\ell')} & \cdots & \pi_{n|n-1}^{(c_s)} \end{bmatrix}^\top \tag{13}$$

collecting the conditional probabilities, $\forall \ell' \in \{1, \ldots, c_s\}$,

$$\pi_{n|n-1}^{(\ell')} \triangleq Pr\{\mathbf{S}_n = \boldsymbol{\mu}_s^{(\ell')} \,|\, \mathbf{Z}_0 = \check{\mathbf{z}}_0, \ldots, \mathbf{Z}_{n-1} = \check{\mathbf{z}}_{n-1}\}$$

represent the predicted marginal posterior $P(\mathbf{s}_n \mid \check{\mathbf{z}}_{0:n-1})$. Similarly, let the c_z-dimensional vector

$$\tilde{\boldsymbol{\pi}}_{n+1|n} \triangleq \begin{bmatrix} \tilde{\pi}_{n+1|n}^{(1)} & \cdots & \tilde{\pi}_{n+1|n}^{(\ell)} & \cdots & \tilde{\pi}_{n+1|n}^{(c_z)} \end{bmatrix}^\top \tag{14}$$

collecting the conditional probabilities, $\forall \ell \in \{1, \ldots, c_z\}$,

$$\tilde{\pi}^{(\ell)}_{n+1|n} \triangleq Pr\{\mathbf{Z}_{n+1} = \boldsymbol{\mu}^{(\ell)}_z \,|\, \mathbf{Z}_0 = \check{\mathbf{z}}_0, \ldots, \mathbf{Z}_{n-1} = \check{\mathbf{z}}_n\}$$

represent the predicted marginal posterior $P(\mathbf{z}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$. We can combine the update and prediction steps in Eqs. (5) and (6), respectively, into a single, one-step-ahead update rule as

$$P(\mathbf{s}_{n+1} \,|\, \check{\mathbf{z}}_{0:n}) \propto \sum_{\mathbf{s}_n \in \boldsymbol{\Omega}_s} P(\mathbf{s}_{n+1} \,|\, \mathbf{s}_n)\, P(\check{\mathbf{z}}_n \,|\, \mathbf{s}_n)\, P(\mathbf{s}_n \,|\, \check{\mathbf{z}}_{0:n-1}). \tag{15}$$

Thus, by assuming that $\check{\mathbf{z}}_n = \boldsymbol{\mu}^{(k)}_z$ for some $k \in \{1, \ldots, c_z\}$, we can rewrite (15) in a compact notation using matrix-vector multiplication as

$$\boldsymbol{\pi}_{n+1|n} \propto \mathbf{F}^{(k)}_{n+1|n} \cdot \boldsymbol{\pi}_{n|n-1} \tag{16}$$

with proportionality constant such that $||\boldsymbol{\pi}_{n+1|n}||_1 = 1$. Moreover, the i-th element $f^{(k,j,i)}_{n+1|n}$ of the j-th row

$$\mathbf{f}^{(k,j)}_{n+1|n} \triangleq \begin{bmatrix} f^{(k,j,1)}_{n+1|n} & \ldots & f^{(k,j,i)}_{n+1|n} & \ldots & f^{(k,j,c_s)}_{n+1|n} \end{bmatrix}^\top$$

of the modified, $c_s \times c_s$ state transition matrix

$$\mathbf{F}^{(k)}_{n+1|n} \triangleq \begin{bmatrix} \mathbf{f}^{(k,1)}_{n+1|n} & \ldots & \mathbf{f}^{(k,j)}_{n+1|n} & \ldots & \mathbf{f}^{(k,c_s)}_{n+1|n} \end{bmatrix}^\top$$

– which depends on the observation $\check{\mathbf{z}}_n$ – is defined as

$$\begin{aligned} f^{(k,j,i)}_{n+1|n} &\triangleq Pr\{\mathbf{S}_{n+1} = \boldsymbol{\mu}^{(j)}_s \,|\, \mathbf{S}_n = \boldsymbol{\mu}^{(i)}_s\} \\ &\times \; Pr\{\mathbf{Z}_n = \boldsymbol{\mu}^{(k)}_z \,|\, \mathbf{S}_n = \boldsymbol{\mu}^{(i)}_s\} \end{aligned}$$

such that $\pi^{(j)}_{n+1|n} \propto \langle \mathbf{f}^{(k,j)}_{n+1|n}, \boldsymbol{\pi}_{n|n-1} \rangle$, $\forall j \in \{1, \ldots, c_s\}$. Conversely, the predicted marginal posterior $P(\mathbf{z}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$ in (8), can be rewritten in compact notation as

$$\tilde{\boldsymbol{\pi}}_{n+1|n} \propto \mathbf{H}_{n+1} \cdot \boldsymbol{\pi}_{n+1|n} \tag{17}$$

with proportionality constant such that $||\tilde{\boldsymbol{\pi}}_{n+1|n}||_1 = 1$. Furthermore, the j-th element $h^{(\ell,j)}_{n+1}$ of the ℓ-th row

$$\mathbf{h}^{(\ell)}_{n+1} \triangleq \begin{bmatrix} h^{(\ell,1)}_{n+1} & \ldots & h^{(\ell,j)}_{n+1} & \ldots & h^{(\ell,c_s)}_{n+1} \end{bmatrix}^\top$$

of the $c_z \times c_s$ observation matrix

$$\mathbf{H}_{n+1} \triangleq \begin{bmatrix} \mathbf{h}^{(1)}_{n+1} & \ldots & \mathbf{h}^{(\ell)}_{n+1} & \ldots & \mathbf{h}^{(c_z)}_{n+1} \end{bmatrix}^\top$$

is defined as

$$h^{(\ell,j)}_{n+1} \triangleq Pr\{\mathbf{Z}_{n+1} = \boldsymbol{\mu}^{(\ell)}_z \,|\, \mathbf{S}_{n+1} = \boldsymbol{\mu}^{(j)}_s\}$$

such that $\tilde{\pi}^{(\ell)}_{n+1|n} \propto \langle \mathbf{h}^{(\ell)}_{n+1}, \boldsymbol{\pi}_{n+1|n} \rangle$, $\forall \ell \in \{1, \ldots, c_z\}$. Note though that we can not use Eqs. (16) and (17) to compute the posteriors $P(\mathbf{s}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$ and $P(\mathbf{z}_{n+1} \,|\, \check{\mathbf{z}}_{0:n})$ since the matrices $\mathbf{F}^{(k)}_{n+1|n}$ and $\mathbf{H}_{n+1}$ are unknown. The Baum-Welch [4] algorithm is a widespread expectation-maximization (EM) procedure to estimate the state transition probabilities $P(\mathbf{s}_{n+1} \,|\, \mathbf{s}_n)$ and the symbol emitting probabilities $P(\check{\mathbf{z}}_n \,|\, \mathbf{s}_n)$ required by both $\mathbf{F}^{(k)}_{n+1|n}$ and $\mathbf{H}_{n+1}$. However, we need to continuously learn these models online, recursively as new observations arrive. Thus, we need to build models for them using a different approach.

3.3 Belief representation

The predicted marginal posterior $P(\mathbf{s}_n \,|\, \check{\mathbf{z}}_{0:n-1})$ can be represented by an alternative vector – referred to as sparsely encoded belief hereafter in the paper – in the high-dimensional binary space $\boldsymbol{\Psi}_s \triangleq \mathbb{Z}_2^{c_s+p_s}$

$$\mathbf{b}_{n|n-1} \triangleq \begin{bmatrix} b^{(1)}_{n|n-1} & \ldots & b^{(c_s+p_s)}_{n|n-1} \end{bmatrix}^\top \in \boldsymbol{\Psi}_s \tag{18}$$

with up to p_s active bits such that $1 \leq ||\mathbf{b}_{n|n-1}||_1 \leq p_s$. As indicated in Appendix A, we can employ a deterministic procedure to build

$$\mathbf{b}_{n|n-1} = Bel(\boldsymbol{\pi}_{n|n-1}; p_s) \tag{19}$$

at any $n \geq 0$. Moreover, we can approximately retrieve the original posterior $\boldsymbol{\pi}_{n|n-1}$ by decoding the sparse belief (see Appendix A for details)

$$\boldsymbol{\pi}_{n|n-1} \approx Bel^{-1}(\mathbf{b}_{n|n-1}; p_s). \tag{20}$$

Hence, the belief $\mathbf{b}_{n|n-1}$ can be seen as a sparse symbol in the high-dimensional binary space $\boldsymbol{\Psi}_s$ corresponding to a valid p.m.f. $\boldsymbol{\pi}_{n|n-1}$ collecting probabilities of symbols in $\boldsymbol{\Omega}_s$. As discussed in Appendix A, the proposed encoding procedure $Bel(\cdot)$ (see Algorithm 4) is lossy in the sense that multiple categorical distributions can map to the same sparse belief. Consequently, as indicated in Eq. (20), the decoding procedure $Bel^{-1}(\cdot)$ (see Algorithm 5) will not be able to fully recover the original belief. Furthermore, this may harm the performance of the proposed grid-based filter, since, as highlighted in the sequel (refer to Algorithm 1), at each time step n, the grid-based filter employs the encoding procedure to build the predicted belief $\mathbf{b}_{n+1|n} = Bel(\boldsymbol{\pi}_{n+1|n}; p_s)$ with up to p_s activated bits, a binary sparse representation of the predicted distribution $\boldsymbol{\pi}_{n+1|n}$.

3.4 Re-interpreting filtering steps

From Eqs. (16) and (20), we can see that, at instant n, the predicted marginal posterior $\boldsymbol{\pi}_{n+1|n}$ is approximately a function $\mathbf{f}_n : \boldsymbol{\Omega}_z \times \boldsymbol{\Psi}_s \mapsto \mathbb{S}_{c_s}$ of the current observation $\check{\mathbf{z}}_n = \boldsymbol{\mu}^{(k)}_z$ and the belief $\mathbf{b}_{n|n-1}$ at instant $n-1$, i.e.

$$\begin{aligned} \boldsymbol{\pi}_{n+1|n} &= C^{-1}_{n+1|n} \cdot \mathbf{F}^{(k)}_{n+1|n} \cdot \boldsymbol{\pi}_{n|n-1} \\ &\approx C^{-1}_{n+1|n} \cdot \mathbf{F}^{(k)}_{n+1|n} \cdot Bel^{-1}(\mathbf{b}_{n|n-1}; p_s) \\ &\equiv \mathbf{f}_n(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}), \end{aligned} \tag{21}$$

where the normalization constant $C_{n+1|n}$ is chosen such that $||\boldsymbol{\pi}_{n+1|n}||_1 = 1$. Similarly, by examining Eq. (17) and noting that $\boldsymbol{\pi}_{n+1|n} \approx Bel^{-1}(\mathbf{b}_{n+1|n}; p_s)$, we conclude that, at instant n, the predicted posterior $\tilde{\boldsymbol{\pi}}_{n+1|n}$ is also an approximate function $\mathbf{h}_{n+1} : \boldsymbol{\Psi}_s \mapsto \mathbb{S}_{c_z}$ of the predicted belief $\mathbf{b}_{n+1|n}$ at instant n, i.e.

$$\begin{aligned} \tilde{\boldsymbol{\pi}}_{n+1|n} &= \tilde{C}^{-1}_{n+1|n} \cdot \mathbf{H}_{n+1} \cdot \boldsymbol{\pi}_{n+1|n} \\ &\approx \tilde{C}^{-1}_{n+1|n} \cdot \mathbf{H}_{n+1} \cdot Bel^{-1}(\mathbf{b}_{n+1|n}; p_s) \\ &\equiv \mathbf{h}_{n+1}(\mathbf{b}_{n+1|n}) \end{aligned} \tag{22}$$

with $\mathbf{b}_{n+1|n} = Bel(\boldsymbol{\pi}_{n+1|n}; p_s)$ and $\tilde{C}_{n+1|n}$ selected such that $||\tilde{\boldsymbol{\pi}}_{n+1|n}||_1 = 1$. Thus, at each instant n, it suffices to propagate the sparse beliefs from $\mathbf{b}_{n|n-1}$ to $\mathbf{b}_{n+1|n}$ and plug them into Eqs. (21) and (22) to compute the predicted marginal posteriors $\boldsymbol{\pi}_{n+1|n}$ and $\tilde{\boldsymbol{\pi}}_{n+1|n}$, respectively.

3.5 Non-parametric estimation

We follow the lead of [17] and assume that, conditioned on a particular sequence of hidden states $\check{\mathbf{s}}_{0:n}$ up to instant n, the predicted posterior $\boldsymbol{\pi}_{n+1|n}$ is a realization of a random vector $\boldsymbol{\Pi}_{n+1|n}$ in the (c_s-1)-dimensional simplex $\mathbb{S}_{c_s}$ following a Dirichlet distribution $Dir(c_s;\boldsymbol{\alpha}_{n|n-1})$ with hyperparameters $\boldsymbol{\alpha}_{n|n-1} \triangleq \begin{bmatrix}\alpha^{(1)}_{n|n-1} & \dots & \alpha^{(c_s)}_{n|n-1}\end{bmatrix}^\top \in \mathbb{R}^{c_s}_+$, i.e.

$$\boldsymbol{\Pi}_{n+1|n} \,|\, \check{\mathbf{s}}_{0:n} \sim Dir(c_s;\boldsymbol{\alpha}_{n|n-1}). \tag{23}$$

We also assume that $\boldsymbol{\alpha}_{n|n-1}$ is indexed by the observation $\check{\mathbf{z}}_n$ at instant n and the previous belief $\mathbf{b}_{n|n-1}$ as per Eq. (21). Thus, given the pair $\left(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}\right)$, we can access the corresponding hyperparameters as $\boldsymbol{\alpha}_{n|n-1} \equiv \boldsymbol{\alpha}_{n|n-1}(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1})$ and use (23) to draw the predicted posterior $\boldsymbol{\pi}_{n+1|n}$. In the sequel, we employ Algorithm 4 (see Appendix A) to encode the predicted posterior $\boldsymbol{\pi}_{n+1|n}$ into the sparse belief

$$\mathbf{b}_{n+1|n} = Bel(\boldsymbol{\pi}_{n+1|n}; p_s). \tag{24}$$

Conditioned on the sequence of observations $\check{\mathbf{z}}_{0:n}$ up to instant n, we also assume that the predicted posterior $\tilde{\boldsymbol{\pi}}_{n+1|n}$ is a realization of a Dirichlet random vector $\tilde{\boldsymbol{\Pi}}_{n+1|n}$ in the (c_z-1)-dimensional simplex $\mathbb{S}_{c_z}$ with distribution $Dir(c_z;\tilde{\boldsymbol{\alpha}}_{n|n-1})$ and hyperparameters $\tilde{\boldsymbol{\alpha}}_{n|n-1} \triangleq \begin{bmatrix}\tilde{\alpha}^{(1)}_{n|n-1} & \dots & \tilde{\alpha}^{(c_z)}_{n|n-1}\end{bmatrix}^\top \in \mathbb{R}^{c_z}_+$, i.e.

$$\tilde{\boldsymbol{\Pi}}_{n+1|n} \,|\, \check{\mathbf{z}}_{0:n} \sim Dir(c_z;\tilde{\boldsymbol{\alpha}}_{n|n-1}), \tag{25}$$

in which $\tilde{\boldsymbol{\alpha}}_{n|n-1}$ is yet again indexed by the predicted belief $\mathbf{b}_{n+1|n}$ computed as in (24) following Eq. (22). Thus, we access the hyperparameters as $\tilde{\boldsymbol{\alpha}}_{n|n-1} \equiv \tilde{\boldsymbol{\alpha}}_{n|n-1}(\mathbf{b}_{n+1|n})$ and use (25) to draw the predicted posterior $\tilde{\boldsymbol{\pi}}_{n+1|n}$. Finally, at instant n, we can predict the next encoded observation $\check{\mathbf{z}}_{n+1}$ at instant n from the predicted belief $\tilde{\boldsymbol{\pi}}_{n+1|n}$ as

$$\begin{aligned}\hat{\mathbf{z}}_{n+1|n} &= \boldsymbol{\mu}_z^{(\hat{k})} \in \boldsymbol{\Omega}_z \\ \hat{k} &= \underset{1\le\ell\le c_z}{\arg\max}\, \tilde{\pi}^{(\ell)}_{n+1|n}.\end{aligned} \tag{26}$$

The probability $\tilde{\pi}^{(\hat{k})}_{n+1|n}$ can be seen thus as a confidence score of the filter prediction $\hat{\mathbf{z}}_{n+1|n}$, whereas the probability $1-\tilde{\pi}^{(k)}_{n+1|n}$ with $k \in \{1,\dots,c_z\}$ corresponding to true observed symbol $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}$ at instant $n+1$ can be seen as a novelty (or anomaly) score of the observation $\check{\mathbf{z}}_{n+1}$.

Algorithm 1 summarizes a single iteration of the proposed non-parametric, grid-based filtering (nP-GbF) procedure at instant n considering some memory-based mechanism $\mathcal{M}$ to store-retrieve the hyperparameters.

Algorithm 1 nP-GbF$\left(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}, \mathcal{M}\right)$

1: Retrieve $\boldsymbol{\alpha}_{n|n-1} \equiv \boldsymbol{\alpha}_{n|n-1}\left(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}\right)$ from $\mathcal{M}$.
2: Draw $\boldsymbol{\pi}_{n+1|n} \sim Dir(c_s;\boldsymbol{\alpha}_{n|n-1})$ as in (23).
3: Build the sparsely encoded belief $\mathbf{b}_{n+1|n}$ as in (24).
4: Retrieve $\tilde{\boldsymbol{\alpha}}_{n|n-1} \equiv \tilde{\boldsymbol{\alpha}}_{n|n-1}\left(\mathbf{b}_{n+1|n}\right)$ from $\mathcal{M}$.
5: Draw $\tilde{\boldsymbol{\pi}}_{n+1|n} \sim Dir(c_z;\tilde{\boldsymbol{\alpha}}_{n|n-1})$ as in (25).
6: Predict next observation $\hat{\mathbf{z}}_{n+1|n} = \boldsymbol{\mu}_z^{(\hat{k})}$ as in (26).
7: **return** $\left(\hat{\mathbf{z}}_{n+1|n}, \hat{k}, \mathbf{b}_{n+1|n}, \tilde{\boldsymbol{\pi}}_{n+1|n}\right)$

Figures 2 and 3 show a running example of the nP-GbF procedure. In Step 1, we retrieve the pseudo-counts $\boldsymbol{\alpha}_{n|n-1}$ from $\mathcal{M}$ given the observation $\check{\mathbf{z}}_n$ at instant n and the previous belief $\mathbf{b}_{n|n-1}$. In the sequel, we sample the predicted posterior $\boldsymbol{\pi}_{n+1|n}$ from the Dirichlet distribution $Dir(c_s;\boldsymbol{\alpha}_{n|n-1})$ in Step 2 and build the predicted belief $\mathbf{b}_{n+1|n}$ in Step 3. Note that the recovered posterior $Bel^{-1}(\mathbf{b}_{n+1|n};p_s)$ shown in Fig. 2 does not match the sampled posterior $\boldsymbol{\pi}_{n+1|n}$ in Step 2. Specifically, the probabilities $\pi^{(\ell'_2)}_{n+1|n}$ and $\pi^{(\ell'_3)}_{n+1|n}$ in the original posterior $\boldsymbol{\pi}_{n+1|n}$ collapsed into a single probability at location ℓ'_2, since the indexes $\ell'_2 = 12$ and $\ell'_3 = 14$ were too close, i.e. $\ell'_3 - \ell'_2 = 2 < p_s = 8$. On the other hand, note that the probability $\pi^{(\ell'_1)}_{n+1|n}$ at the location $\ell'_1 = 4$ remained virtually intact in the recovered posterior $Bel^{-1}(\mathbf{b}_{n+1|n};p_s)$.

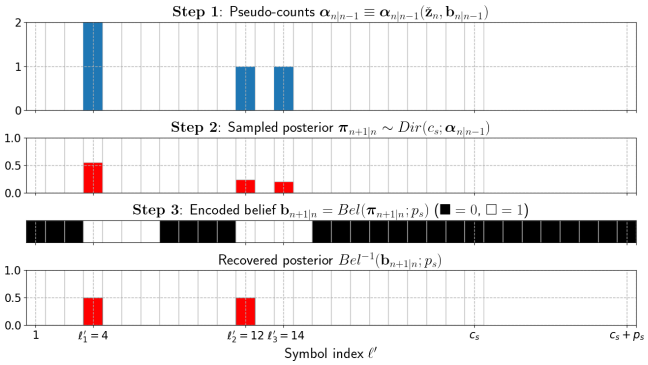


Figure 2: Running example illustrating Steps 1 through 3 of the nP-GbF procedure considering $c_s = 24$, $p_s = 8$ and $\alpha^{(\ell')}_{n|n-1} = 2^{-12}$, $\forall \ell'$, except for $\ell'_1 = 4$, $\ell'_2 = 12$ and $\ell'_3 = 14$.

Similarly, as show in Fig. 3, we retrieve the pseudo-counts $\tilde{\boldsymbol{\alpha}}_{n|n-1}$ from $\mathcal{M}$ in Step 4 using now the predicted belief $\mathbf{b}_{n+1|n}$ at instant n obtained in Step 3. Then, we sample the predicted posterior $\tilde{\boldsymbol{\pi}}_{n+1|n}$ from the Dirichlet distribution $Dir(c_z;\tilde{\boldsymbol{\alpha}}_{n|n-1})$ in Step 5 and, finally, predict the next observation $\hat{\mathbf{z}}_{n+1|n}$ in Step 6 as the symbol $\boldsymbol{\mu}_z^{(\hat{k})} \in \boldsymbol{\Omega}_z$ with the highest probability $\tilde{\pi}^{(\hat{k})}_{n+1|n}$ in the predicted posterior $\tilde{\boldsymbol{\pi}}_{n+1|n}$. In the running example, the highest probability $\tilde{\pi}^{(\ell_2)}_{n+1|n}$ occurs at the location ℓ_2, i.e. $\hat{k} = \ell_2 = 16$.

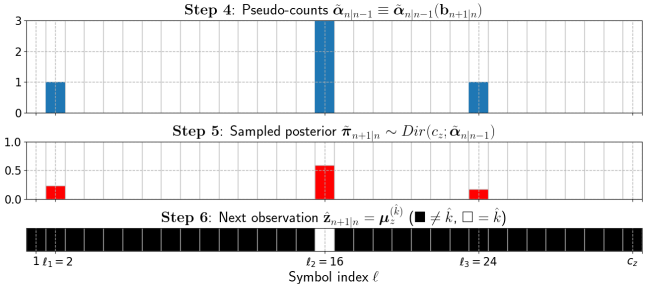


Figure 3: Running example illustrating Steps 3 through 6 of the nP-GbF procedure considering $c_z = 32$ and $\tilde{\alpha}^{(\ell)}_{n|n-1} = 2^{-12}$, $\forall \ell$, except for $\ell_1 = 2$, $\ell_2 = 16$ and $\ell_3 = 24$.

3.6 Hidden states formation

In general, the hidden states are not directly observable. As expected, the filtering steps (see Algorithm 1) do not require one to directly observe them. However, as shown in Appendix B, to be able to update the model hyperparameters

$\boldsymbol{\alpha}_{n|n-1}$ using a frequentist approach, we need to pin the index j of the hidden state symbol $\boldsymbol{\mu}_s^{(j)}$ at instant $n+1$ from which the observation $\check{\mathbf{z}}_{n+1}$ is emitted. To this end, we further assume that the hidden state $\mathbf{s}_n$ represents the last ξ observations. Specifically, at instant $n+1$, upon the arrival of the new encoded observation $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}$, $k \in \{1, \ldots, c_z\}$, we select a unique index $j \in \{1, \ldots, c_s\}$ and assign it to the sub-sequence of observations $\check{\mathbf{z}}_{n+2-\xi:n+1}$, i.e. we make $\check{\mathbf{s}}_{n+1} = \boldsymbol{\mu}_s^{(j)} \in \boldsymbol{\Omega}_s$. Lastly, in this paper, we increment the selected indexes from $\{1, \ldots, c_s\}$ by p_s to make sure their belief representations will not overlap (see Appendix A). Then, we update the hyperparameters $\boldsymbol{\alpha}_{n|n-1}$ and $\tilde{\boldsymbol{\alpha}}_{n|n-1}$ as follows.

3.7 Non-parametric learning

By construction, conditioned on the predicted marginal posterior $\boldsymbol{\pi}_{n+1|n}$ at instant n, the hidden state $\mathbf{S}_{n+1}$ at instant $n+1$ is distributed according to a categorical distribution $Cat(\boldsymbol{\Omega}_s; \boldsymbol{\pi}_{n+1|n})$ over $\boldsymbol{\Omega}_s$, i.e.

$$\mathbf{S}_{n+1} \,|\, \boldsymbol{\pi}_{n+1|n} \sim Cat(\boldsymbol{\Omega}_s; \boldsymbol{\pi}_{n+1|n}). \tag{27}$$

Since the Dirichlet distribution is the conjugate prior of the categorical distribution, it follows from Eqs. (23) and (27) that, conditioned on the built sequence of symbols $\check{\mathbf{s}}_{0:n+1}$ up to instant $n+1$, the random vector $\boldsymbol{\Pi}_{n+1|n}$ is also Dirichlet

$$\boldsymbol{\Pi}_{n+1|n} \,|\, \check{\mathbf{s}}_{n+1} = \boldsymbol{\mu}_s^{(j)}, \check{\mathbf{s}}_{0:n} \sim Dir(c_s; \boldsymbol{\alpha}_{n+1|n}), \tag{28}$$

in which the new hyperparameters are updated using a frequentist approach as (proof details are in Appendix B)

$$\boldsymbol{\alpha}_{n+1|n} = \boldsymbol{\alpha}_{n|n-1} + \mathbf{1}_j. \tag{29}$$

Note that, conditioned on $\tilde{\boldsymbol{\pi}}_{n+1|n}$, the next observation $\mathbf{Z}_{n+1}$ at instant $n+1$ is also distributed according to a categorical distribution $Cat(\boldsymbol{\Omega}_z; \tilde{\boldsymbol{\pi}}_{n+1|n})$ over $\boldsymbol{\Omega}_z$, i.e.

$$\mathbf{Z}_{n+1} \,|\, \tilde{\boldsymbol{\pi}}_{n+1|n} \sim Cat(\boldsymbol{\Omega}_z; \tilde{\boldsymbol{\pi}}_{n+1|n}). \tag{30}$$

Thus, it follows from Eqs. (25) and (30) that the same principle is applicable to $\tilde{\boldsymbol{\Pi}}_{n+1|n}$. Conditioned on the encoded sequence of observations $\check{\mathbf{z}}_{0:n+1}$ up to instant $n+1$, we can write

$$\tilde{\boldsymbol{\Pi}}_{n+1|n} \,|\, \check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}, \check{\mathbf{z}}_{0:n} \sim Dir(c_z; \tilde{\boldsymbol{\alpha}}_{n+1|n}), \tag{31}$$

where the new hyperparameters are given by (see Appendix B)

$$\tilde{\boldsymbol{\alpha}}_{n+1|n} = \tilde{\boldsymbol{\alpha}}_{n|n-1} + \mathbf{1}_k. \tag{32}$$

Even though learning happens at instant $n+1$, it is worth noting that the updated hyperparameters $\boldsymbol{\alpha}_{n+1|n}$ must be properly stored indexed by both the observation $\check{\mathbf{z}}_n$ and the previous belief $\mathbf{b}_{n|n-1}$ from instant n such that $\boldsymbol{\alpha}_{n+1|n} \equiv \boldsymbol{\alpha}_{n+1|n}(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1})$. Conversely, the updated hyperparameters $\tilde{\boldsymbol{\alpha}}_{n+1|n}$ must be stored indexed by the predicted belief $\mathbf{b}_{n+1|n}$, i.e. $\tilde{\boldsymbol{\alpha}}_{n+1|n} \equiv \tilde{\boldsymbol{\alpha}}_{n+1|n}(\mathbf{b}_{n+1|n})$. Thus, in this context, we need to draw a new posterior $\boldsymbol{\pi}_{n+1|n} \sim Dir(c_s; \boldsymbol{\alpha}_{n+1|n})$ using (23) and rebuild the sparse belief $\mathbf{b}_{n+1|n}$ as in (24), before storing the updated hyperparameters $\tilde{\boldsymbol{\alpha}}_{n+1|n}$ – in some preexisting storage $\mathcal{M}$ – indexed by $\mathbf{b}_{n+1|n}$.

The non-parametric, memory-based learning (nP-MbL) procedure is outlined in Algorithm 2. We assume again that all hyperparameters are retrieved from and stored back to the same memory-based mechanism $\mathcal{M}$ and that the observation and the hidden state at instant $n+1$ are respectively $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}$ and $\check{\mathbf{s}}_{n+1} = \boldsymbol{\mu}_s^{(j)}$. Note that we weighted the increments in (29) and (32) by a learning rate $\eta \geq 1$ to speed-up sequence learning.

Algorithm 2 nP-MbL$\big(j, k, \eta, \check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}, \mathbf{b}_{n+1|n}, \mathcal{M}\big)$

1: Retrieve $\boldsymbol{\alpha}_{n|n-1} \equiv \boldsymbol{\alpha}_{n|n-1}\big(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}\big)$ from $\mathcal{M}$.
2: Update $\boldsymbol{\alpha}_{n+1|n} \leftarrow \boldsymbol{\alpha}_{n|n-1} + \eta \cdot \mathbf{1}_j$ as in (29).
3: Store $\boldsymbol{\alpha}_{n+1|n}\big(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}\big) \equiv \boldsymbol{\alpha}_{n+1|n}$ into $\mathcal{M}$.
4: Draw $\boldsymbol{\pi}_{n+1|n} \sim Dir(c_s; \boldsymbol{\alpha}_{n+1|n})$ as in (23).
5: Encode $\boldsymbol{\pi}_{n+1|n}$ into $\mathbf{b}_{n+1|n}$ as in (24).
6: Retrieve $\tilde{\boldsymbol{\alpha}}_{n|n-1} \equiv \tilde{\boldsymbol{\alpha}}_{n|n-1}\big(\mathbf{b}_{n+1|n}\big)$ from $\mathcal{M}$.
7: Update $\tilde{\boldsymbol{\alpha}}_{n+1|n} \leftarrow \tilde{\boldsymbol{\alpha}}_{n|n-1} + \eta \cdot \mathbf{1}_k$ as in (32).
8: Store $\tilde{\boldsymbol{\alpha}}_{n+1|n}\big(\mathbf{b}_{n+1|n}\big) \equiv \tilde{\boldsymbol{\alpha}}_{n+1|n}$ into $\mathcal{M}$.
9: **return** $\mathcal{M}$

4. RAM-BASED IMPLEMENTATION

In this section we propose an efficient implementation of the memory-based mechanism $\mathcal{M}$ using a shallow, two-layer depth WNN. Specifically, each layer consists of an one-dimensional array of VG-RAM nodes allowing us thus to i) quickly retrieve input-output pairs indexed by binary input patterns and ii) perform one-shot learning by storing new associative input-output pairs readily.

A VG-RAM node stores a set $\mathbf{M}$ of input-output pairs – a.k.a. lookup table – such that, $\forall\,(\boldsymbol{\beta}', \alpha') \in \mathbf{M}$, $\boldsymbol{\beta}' \in \mathbb{Z}_2^{d_\beta}$ is a d_β-dimensional binary input pattern and $\alpha' \in \mathbb{R}_+$ is the associated output, in our case, a non-negative hyperparameter. In particular, given an input pattern $\boldsymbol{\beta} \in \mathbb{Z}_2^{d_\beta}$, the node finds the hyperparameter $\bar{\alpha}$ whose corresponding input pattern $\bar{\boldsymbol{\beta}}$ has the closest Hamming distance to it. More precisely, the node retrieves the hyperparameter $\bar{\alpha}$ in $\big(\bar{\boldsymbol{\beta}}, \bar{\alpha}\big) \leftarrow \text{FIND}\,(\mathbf{M}, \boldsymbol{\beta})$ from $\mathbf{M}$ for a given $\boldsymbol{\beta}$ such that

$$\big(\bar{\boldsymbol{\beta}}, \bar{\alpha}\big) = \underset{(\boldsymbol{\beta}', \alpha') \in \mathbf{M}}{\arg\min}\; d_H(\boldsymbol{\beta}', \boldsymbol{\beta}).$$

On the other hand, to learn a new input-output pair $(\boldsymbol{\beta}, \alpha)$, the node makes $\mathbf{M} \leftarrow \text{INIT}\,(\mathbf{M}, \boldsymbol{\beta}, \alpha) \triangleq \mathbf{M} \cup \{(\boldsymbol{\beta}, \alpha)\}$. Lastly, to update the hyperparameter $\bar{\alpha}$ associated to an existing input pattern $\bar{\boldsymbol{\beta}}$, replacing it by some $\tilde{\alpha} \in \mathbb{R}_+$, the node makes $\mathbf{M} \leftarrow \text{UPDT}\,\big(\mathbf{M}, \bar{\boldsymbol{\beta}}, \bar{\alpha}, \tilde{\alpha}\big) \triangleq \big(\mathbf{M} \setminus \{\big(\bar{\boldsymbol{\beta}}, \bar{\alpha}\big)\}\big) \cup \{\big(\bar{\boldsymbol{\beta}}, \tilde{\alpha}\big)\}$.

4.1 Non-parametric estimation

Figure 4 graphically illustrates Steps 1 through 3 of Algorithm 1 and summarizes therefore how the function $\mathbf{b}_{n+1|n} \equiv \mathbf{f}_n(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1})$ is implemented. In particular, the figure shows how we employ a c_s-dimensional array of VG-RAM nodes to retrieve the hyperparameters $\boldsymbol{\alpha}_{n|n-1}$ based on the stored input-output pairs whose input patterns have the closest Hamming distance to the concatenated input vector $\check{\mathbf{z}}_n \oplus \mathbf{b}_{n|n-1} \in \mathbb{Z}_2^{c_z + c_s + p_s}$. Figure 5 depicts Steps 4 through 6 of Algorithm 1, which in turn implements the function $\tilde{\mathbf{b}}_{n+1|n} \equiv \mathbf{h}_n(\mathbf{b}_{n+1|n})$. The figure indicates how an additional c_z-dimensional array of VG-RAM nodes is employed to retrieve $\tilde{\boldsymbol{\alpha}}_{n|n-1}$ based on the stored associative pairs whose input patterns have the closest Hamming distance to the input $\mathbf{b}_{n+1|n} \in \mathbb{Z}_2^{c_s + p_s}$.

Note that, in filtering Steps 2 and 5, the filter assigns different probabilities $\pi_{n+1|n}^{(\ell')}$ and $\tilde{\pi}_{n+1|n}^{(\ell)}$ to symbols in $\boldsymbol{\Omega}_s$ and $\boldsymbol{\Omega}_z$ according to the retrieved hyperparameters $\alpha_{n+1|n}^{(\ell')}$ and $\tilde{\alpha}_{n+1|n}^{(\ell)}$ in Steps 1 and 4, respectively. In Step 3, the filter also assigns an indexable label (symbol in $\boldsymbol{\Psi}_s$) to the predicted posterior $\boldsymbol{\pi}_{n+1|n}$ by building the sparse belief $\mathbf{b}_{n+1|n} = Bel(\boldsymbol{\pi}_{n+1|n})$,

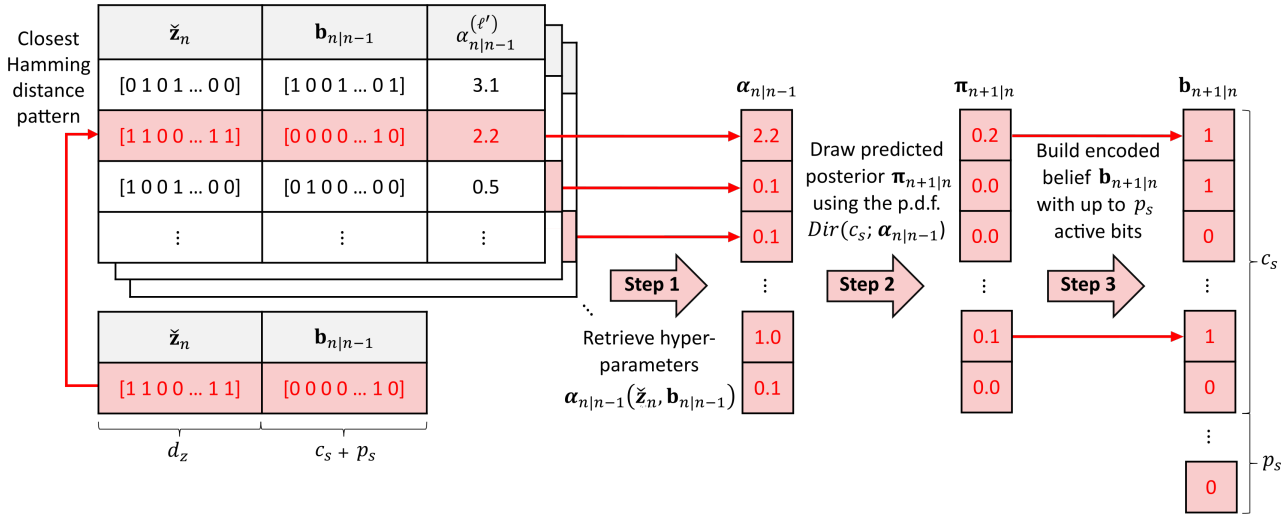


Figure 4: WNN layer $\mathcal{L}_1$: one-step-ahead state belief prediction.

which is in turn employed to retrieve the hyperparameters $\tilde{\alpha}^{(\ell)}_{n+1|n}$ at Step 4. We select finally the predicted observation $\hat{\mathbf{z}}_{n+1|n}$ in Step 6 – based on the predicted marginal posterior $\tilde{\boldsymbol{\pi}}_{n+1|n}$ – as the base vector $\boldsymbol{\mu}^{(\hat{k})}_z$ with the highest probability $\tilde{\pi}^{(\hat{k})}_{n+1|n}$ using (26).

4.2 Non-parametric learning

Let $\mathbf{M}^{(\ell')}_1$ and $\mathbf{M}^{(\ell)}_2$ denote respectively the lookup tables of the ℓ'-th and ℓ-th VG-RAM nodes at WNN layers $\mathcal{L}_1$ and $\mathcal{L}_2$. At instant n, the ℓ'-th VG-RAM node in Fig. 4 retrieves

$$\left(\cdot, \alpha^{(\ell')}_{n|n-1}\right) \leftarrow \text{FIND}\left(\mathbf{M}^{(\ell')}_1, \check{\mathbf{z}}_n \oplus \mathbf{b}_{n|n-1}\right)$$

and, therefore, contributes indirectly to the activation of the ℓ'-th bit of the belief $\mathbf{b}_{n+1|n}$ through the Dirichlet-sampled probability $\pi^{(\ell')}_{n+1|n}$. By the same token, the ℓ-th node in Fig. 5 retrieves

$$\left(\cdot, \tilde{\alpha}^{(\ell)}_{n|n-1}\right) \leftarrow \text{FIND}\left(\mathbf{M}^{(\ell)}_2, \mathbf{b}_{n+1|n}\right)$$

at instant n driving therefore the odds of the ℓ-th symbol $\boldsymbol{\mu}^{(\ell)}_z$ in $\boldsymbol{\Omega}_z$ being selected as the prediction $\hat{\mathbf{z}}_{n+1|n} = \boldsymbol{\mu}^{(\hat{k})}_z$, i.e. $\hat{k} = \ell$, by shaping the Dirichlet-sampled probability $\tilde{\pi}^{(\ell)}_{n+1|n}$. Then, at instant $n+1$, the VG-RAM nodes finally learn the new hyperparameters $\alpha^{(\ell')}_{n+1|n}$ and $\tilde{\alpha}^{(\ell)}_{n+1|n}$ by storing new input-output association pairs

$$\mathbf{M}^{(\ell')}_1 \leftarrow \text{INIT}\left(\mathbf{M}^{(\ell')}_1, \check{\mathbf{z}}_n \oplus \mathbf{b}_{n|n-1}, \alpha^{(\ell')}_{n+1|n}\right)$$

$$\mathbf{M}^{(\ell)}_2 \leftarrow \text{INIT}\left(\mathbf{M}^{(\ell)}_2, \mathbf{b}_{n+1|n}, \tilde{\alpha}^{(\ell)}_{n+1|n}\right)$$

or by updating the existing ones as

$$\mathbf{M}^{(\ell')}_1 \leftarrow \text{UPDT}\left(\mathbf{M}^{(\ell')}_1, \check{\mathbf{z}}_n \oplus \mathbf{b}_{n|n-1}, \alpha^{(\ell')}_{n|n-1}, \alpha^{(\ell')}_{n+1|n}\right)$$

$$\mathbf{M}^{(\ell)}_2 \leftarrow \text{UPDT}\left(\mathbf{M}^{(\ell)}_2, \mathbf{b}_{n+1|n}, \tilde{\alpha}^{(\ell)}_{n|n-1}, \tilde{\alpha}^{(\ell)}_{n+1|n}\right).$$

Intuitively, the ℓ'-th VG-RAM node in layer $\mathcal{L}_1$ learns – in a frequentist fashion – how likely is to activate the corresponding p_s bits $b^{(\ell')}_{n+1|n}, \ldots, b^{(\ell'+p_s)}_{n+1|n}$ along the layer's output belief $\mathbf{b}_{n+1|n}$ when exposed to a given input pattern $\check{\mathbf{z}}_n \oplus \mathbf{b}_{n|n-1}$ such that the likelihood of activating those bits increases as the corresponding input pattern (or a similar one in terms of the Hamming distance) is repeatedly presented to the layer's input. The VG-RAM layer $\mathcal{L}_1$ thus learns how to predict its output belief given the current input pattern. Note that, as we are not estimating the hidden state $\mathbf{s}_{n+1}$, there is no need to explicitly build the set $\boldsymbol{\Omega}_s$. As shown in Appendix B, we just need the symbol indexes to update the hyperparameters $\boldsymbol{\alpha}_{n|n-1}$. The set $\boldsymbol{\Omega}_z$, in turn, is required by both the WNN layer $\mathcal{L}_2$ to make the prediction $\hat{\mathbf{z}}_{n+1|n}$ in Step 6 and the codec scheme to encode the observation $\check{\mathbf{y}}_{n+1}$ as in (11) and decode the prediction $\hat{\mathbf{z}}_{n+1|n}$ as in (12).

5. APPLICATION EXAMPLE

In this section, we illustrate the use of the proposed nP-GbF and nP-MbL procedures in an anomaly detection (AD) application. In particular, we build an anomaly detector to handle streaming, real-time data and assess its performance using four synthetic toy time-series, specifically designed to evaluate AD algorithms.

Algorithm 3 shows how we can employ Algorithms 1 and 2 to build a non-parametric, CL anomaly detection (nP-CLAD) procedure. In this procedure, we compute first the anomaly score Σ^{ad}_{n+1} as the complement of the probability $\tilde{\pi}^{(k)}_{n+1|n}$ in $\tilde{\boldsymbol{\pi}}_{n+1|n}$ corresponding to the encoded observation $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}^{(k)}_z$ at instant $n+1$. Then, we verify if the corresponding anomaly score Σ^{ad}_{n+1} exceeds a threshold τ. Note therefore that this test takes into account the predicted probability $\tilde{\pi}^{(k)}_{n+1|n}$ of the observed symbol $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}^{(k)}_z$ in $\tilde{\boldsymbol{\pi}}_{n+1|n}$. Alternatively, we can modify this criterion to check if the encoded observation $\check{\mathbf{z}}_{n+1}$ mismatches the predicted observation $\hat{\mathbf{z}}_{n+1|n}$ – symbol in $\boldsymbol{\Omega}_z$ with the highest assigned probability in the predicted (sampled) posterior $\tilde{\boldsymbol{\pi}}_{n+1|n}$ – by a maximum number of bits δ, i.e. $d_H(\check{\mathbf{z}}_{n+1}, \hat{\mathbf{z}}_{n+1|n}) \geq \delta$.

For convenience, we summarize the hyperparameters employed by the sparse codec, the WNN layers – $\mathcal{L}_1$ and $\mathcal{L}_2$ – and the nP-CLAD algorithm in Table 1.

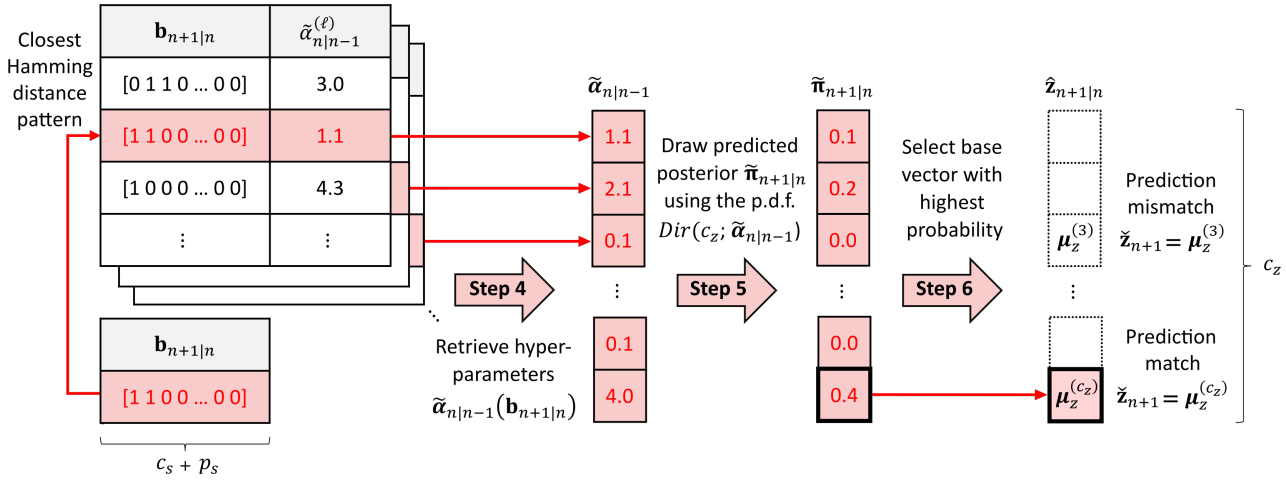


Figure 5: WNN layer $\mathcal{L}_2$: next observation belief prediction & matching.

Algorithm 3 nP-CLAD$\left(\check{\mathbf{y}}_{n+1}, \check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}, \mathcal{M}\right)$

1: Encode observation $\check{\mathbf{y}}_{n+1}$ as $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}$ using (11).
2: Build the hidden state $\check{\mathbf{s}}_{n+1} = \boldsymbol{\mu}_s^{(j)}$ from $\check{\mathbf{z}}_{n+2-\xi:n+1}$.
3: Predict the observation $\mathbf{z}_{n+1}$ using Algorithm 1

$$\left(\hat{\mathbf{z}}_{n+1|n}, \hat{k}, \mathbf{b}_{n+1|n}, \tilde{\boldsymbol{\pi}}_{n+1|n}\right) \leftarrow \text{nP-GbF}\left(\check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}, \mathcal{M}\right).$$

4: Compute the anomaly score as $\Sigma^{ad}_{n+1} \leftarrow 1 - \tilde{\pi}^{(k)}_{n+1|n}$.
5: Detect the anomaly by checking $\Lambda^{ad}_{n+1} \leftarrow [\![\Sigma^{ad}_{n+1} > \tau]\!]$.
6: Learn the non-parametric model using Algorithm 2

$$\mathcal{M} \leftarrow \text{nP-MbL}\left(j, k, \eta, \check{\mathbf{z}}_n, \mathbf{b}_{n|n-1}, \mathbf{b}_{n+1|n}, \mathcal{M}\right).$$

7: **return** $\left(\Lambda^{ad}_{n+1}, \Sigma^{ad}_{n+1}, \check{\mathbf{z}}_{n+1}, \mathbf{b}_{n+1|n}, \mathcal{M}\right)$

Table 1: Summary of the WNN model hyperparameters.

Codec: map observations $\check{\mathbf{y}}_n \in \boldsymbol{\Omega}_y$ into symbols $\check{\mathbf{z}}_n \in \boldsymbol{\Omega}_z$	
a_z	Maximum number of activated bits in $\check{\mathbf{z}}_n$
b_z	Minimum Hamming distance of symbols in $\boldsymbol{\Omega}_z$
d_z	Total number of bits of $\check{\mathbf{z}}_n$
Layer $\mathcal{L}_1$: learn how to predict the posterior $\boldsymbol{\pi}_{n+1\|n}$	
c_s	Number of symbols in $\boldsymbol{\Omega}_s$
$\forall \ell', \alpha_{\ell'}$	Initial Dirichlet $Dir(c_s; \boldsymbol{\alpha}_{0\|-1})$ pseudo-counts
p_s	Number of activated bits in the belief $\boldsymbol{b}_{n\|n-1}$
Layer $\mathcal{L}_2$: learn how to predict the posterior $\tilde{\boldsymbol{\pi}}_{n+1\|n}$	
c_z	Number of symbols in $\boldsymbol{\Omega}_z$
$\forall \ell, \tilde{\alpha}_\ell$	Initial Dirichlet $Dir(c_z; \tilde{\boldsymbol{\alpha}}_{0\|-1})$ pseudo-counts
nP-CLAD: check $\check{\mathbf{y}}_{n+1}$ using its anomaly score Σ^{ad}_{n+1}	
ξ	Length of the sub-sequence $\check{\mathbf{z}}_{n+2-\xi:n+1}$
τ	Anomaly detection threshold
η	Learning rate employed by the WNN model

We initialize $\mathbf{b}_{0|-1}$ as per nP-GbF filter Steps 2 and 3 by making $\boldsymbol{\alpha}_{0|-1} = \boldsymbol{\alpha}_0$ in Algorithm 1. We encode the streamed observations $\check{\mathbf{y}}_{0:\infty}$ in Step 1 using a fixed-range $(0 - 100)$ numeric encoder with 961 buckets – unique symbols in $\boldsymbol{\Omega}_z$ – and $d_z = 2^{10}$ bits to encode the time-series values as indicated in [29]. We also initialize $\check{\mathbf{z}}_{1-\xi:-1}$ by replicating $\check{\mathbf{z}}_0$ to be able to build $\check{\mathbf{s}}_n$, $\forall n \geq 0$, as required in Step 2 of Algorithm 3. There is no need to decode network predictions in this AD application.

Table 2 summarizes the hyperparameters employed to setup the WNN model and run all experiments. We should highlight that only 11 hyperparameters are required to initialize the proposed RAM-based model plus the nP-CLAD algorithm. The hyperparameter d_s is not required, since there is no need to build the set $\boldsymbol{\Omega}_s$.

Table 2: Hyperparameters employed by the codec, the WNN layers, and the nP-CLAD algorithm throughout experiments.

Codec		**Layer $\mathcal{L}_1$**		**Layer $\mathcal{L}_2$**		**nP-CLAD**	
a_z	2^6	c_s	2^{10}	c_z	2^{10}	ξ	2^3
b_z	2^2	$\forall \ell', \alpha^{\ell'}$	2^{-12}	$\forall \ell, \tilde{\alpha}^{\ell}$	2^{-12}	τ	0.75
d_z	2^{10}	p_s	2^3	–	–	η	2^1

Figures 6 through 9 show the AD results considering four artificial time-series with randomly anomalous spikes. Despite being quite simple, these AD experiments illustrate the filter ability to learn increasingly complex sequences online and detect point anomalies. We highlighted the true anomalies – intervals shaded in red – and indicated the detected anomalies – red circles – in the last 500 time instants (green area), since the filter was still learning the time-series in the first half of the simulation (gray area). The nP-CLAD was able to successfully detect all random spikes along the evaluated time-series using the anomaly scores provided by the non-parametric model. Nevertheless, it is worth noting that the filter still needs to be validated in real-world settings using e.g. Numenta Anomaly Benchmark (NAB) [1].

The experiments were conducted in an Intel(R) Core(TM) i7-9750H central processing unit (CPU) @ 2.60GHz with 32GB of RAM. Note that the inference time depends on the number of entries stored in the VGRAM nodes' memory, since the nodes memory is virtual and its size increases as the WNN model learns a new sequence.

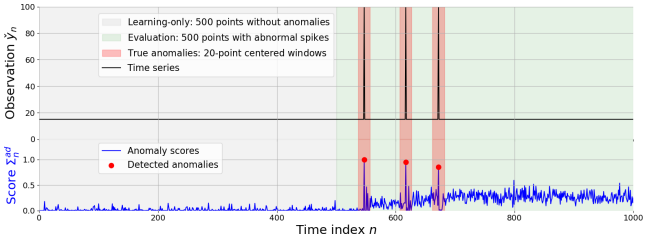


Figure 6: Constant time-series with anomalous spikes.

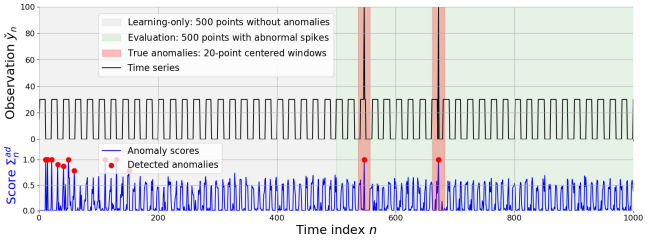


Figure 7: Squared time-series with anomalous spikes.

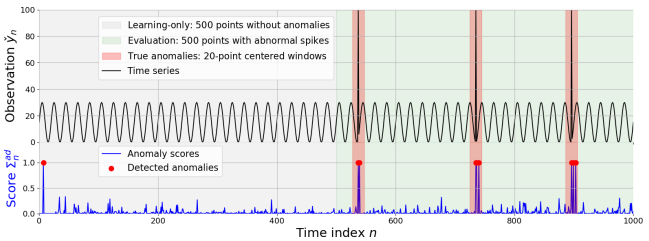


Figure 8: Sinusoidal time-series with anomalous spikes.

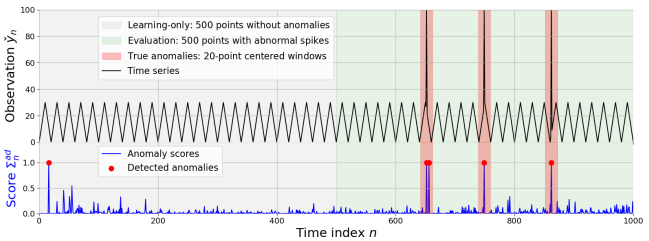


Figure 9: Sawtooth time-series with anomalous spikes.

Table 3 indicates the mean and maximum inference times in milliseconds – considering one iteration of Algorithm 3 –, and the accumulated memory size in kilobytes at the end of each experiment. As expected, the accumulated memory size increases according to the complexity of the learned time series. Finally, a non-optimized Python implementation of the proposed algorithms and reproducible experiments can be found in [15].

Table 3: Computational performance considering different time series. Inference times are in milliseconds (ms), whereas accumulated memory sizes are in kilobytes (kB).

Time series	Inference time		Accumulated memory size
	Average	Maximum	
Constant	10.74 ms	24.82 ms	58.45 kB
Squared	11.22 ms	32.03 ms	151.17 kB
Sinusoidal	11.72 ms	31.93 ms	168.30 kB
Sawtooth	11.50 ms	35.62 ms	178.40 kB

6. CONCLUSIONS

6.1 Summary of paper contributions

We presented in this paper a fully non-parametric, grid-based filter designed to intrinsically learn an interpretable probabilistic model of the underlying Markovian process (see Fig. 1) emitting the incoming sequence of observations, which allows one to make explainable and auditable predictions – relying on clear input-to-output causal chains connected to traceable memory entries – with promising, but still limited results as shown in toy AD examples. Moreover, the proposed RAM-based filter implementation performs one-shot, CL without requiring any special hardware to accomplish it. Indeed, VG-RAM nodes can be efficiently implemented through software – using e.g. C++ to really tame the machine – conceived to run on general-purpose, CPUs using ordinary, abundant RAM hardware, which is orders of magnitude cheaper than state-of-the-art graphics processing units (GPUs) specially designed for machine learning tasks.

6.2 Future research directions

As future research, we propose to adapt the sparse auto-encoder introduced in [26] to learn (offline) a high-dimensional binary representation of the continuous-valued observations in an embedded space. We also intend to employ mismatched predictions as learning signals as proposed by Cui et al. in [11]. Furthermore, we can limit the memory capacity of the VG-RAM nodes and individually force them to eventually replace associative input-output pairs with small hyperparameters by new ones, limiting thus the required network resources by forcing its nodes to gracefully forget in the long run input patterns which are not frequently seen, i.e. presented as input to their corresponding layers. Lastly, we may use sparse binary word vectors as proposed by Faruqui et al. in [16] to encode natural language data and check the suitability of the proposed filter to build a tiny language model which is able to continuously learn from new corpora.

7. ADDITIONAL AUTHORS

Additional authors: Thiago Oliveira-Santos and Claudine Badue (Applied Computational Intelligence Institute, UFES `[todsantos, claudine]@inf.ufes.br`).

8. REFERENCES

[1] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.

[2] I. Aleksander. From WISARD to MAGNUS: A family of weightless virtual neural machines. *RAM-Based Neural Networks*, pages 18–30, 1998.

[3] S. Barra, S. M. Carta, A. Corriga, A. S. Podda, and D. R. Recupero. Deep learning and time series-to-image encoding for financial forecasting. *IEEE/CAA Journal of Automatica Sinica*, 7(3):683–692, 2020.

[4] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[5] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational Statistics*, pages 177–186. Springer, 2010. Paris France, August 22-27, 2010 Keynote, Invited and Contributed Papers.

[6] L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade: Second Edition*, pages 421–436. Springer, 2012.

[7] A. Canale and M. Ruggiero. Bayesian nonparametric forecasting of monotonic functional time series. *Electronic Journal of Statistics*, 10(2):3265–3286, 2016.

[8] Y. Chen, S. Liu, Z. Wang, D. Wu, Y. Li, B. Xing, B. Guo, and L. Chen. Online parallel attack detection method for industrial control based on multi-bandpass filter. *IEEE Internet of Things Journal*, 11(1):880–888, 2024.

[9] C. Cheng, A. Sa-Ngasoongsong, O. Beyca, T. Le, H. Yang, Z. Kong, and S. T. Bukkapatnam. Time series forecasting for nonlinear and non-stationary processes: a review and comparative study. *IIE Transactions*, 47(10):1053–1071, 2015.

[10] V. K. R. Chimmula and L. Zhang. Time series forecasting of COVID-19 transmission in canada using lstm networks. *Chaos, Solitons & Fractals*, 135:109864, 2020.

[11] Y. Cui, S. Ahmad, and J. Hawkins. Continuous online sequence learning with an unsupervised neural network model. *Neural Computation*, 28(11):2474–2504, 2016.

[12] M. De Gregorio and M. Giordano. An experimental evaluation of weightless neural networks for multi-class classification. *Applied Soft Computing*, 72:338–354, 2018.

[13] A. F. De Souza, C. Badue, F. Pedroni, E. Oliveira, S. S. Dias, H. Oliveira, and S. F. de Souza. Face recognition with VG-RAM weightless neural networks. In *Artificial Neural Networks-ICANN 2008: 18th International Conference, Prague, Czech Republic, September 3-6, 2008, Proceedings, Part I 18*, pages 951–960. Springer, 2008.

[14] A. F. De Souza, F. Pedroni, E. Oliveira, P. M. Ciarelli, W. F. Henrique, L. Veronese, and C. Badue. Automated multi-label text categorization with VG-RAM weightless neural networks. *Neurocomputing*, 72(10–12):2209–2217, 2009.

[15] S. S. Dias. Weightless neural network library (wnnlib), 2025. Available from https://github.com/stivenschwanz/wnnlib/tree/safeai.

[16] M. Faruqui, Y. Tsvetkov, D. Yogatama, C. Dyer, and N. A. Smith. Sparse overcomplete word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1491–1500, 2015.

[17] Z. Ghahramani. Bayesian non-parametrics and the probabilistic approach to modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110553, 2013.

[18] J. Hawkins and S. Ahmad. Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in Neural Circuits*, 10(23):1–13, 2016.

[19] J. Hawkins, S. Ahmad, D. Dubinsky, et al. Cortical learning algorithm and hierarchical temporal memory. *Numenta Whitepaper*, pages 1–68, 2011.

[20] A. Iyer, K. Grewal, A. Velu, L. O. Souza, J. Forest, and S. Ahmad. Avoiding catastrophe: Active dendrites enable multi-task learning in dynamic environments. *Frontiers in Neurorobotics*, 16(846219):1–23, 2022.

[21] S. M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall PTR, 1993.

[22] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[23] B. Lim and S. Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

[24] T. B. Ludermir, A. C. P. d. L. F. Carvalho, A. P. Braga, and M. C. De Souto. Weightless neural models: a review of current and past works. *Neural Computing Surveys*, 2:41–61, 1999.

[25] G. Mahalakshmi, S. Sridevi, and S. Rajaram. A survey on forecasting of time series data. In *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)*, pages 1–8. IEEE, 2016.

[26] A. Makhzani and B. Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

[27] R. J. Mitchell, J. Bishop, S. Box, and J. Hawker. Comparison of some methods for processing 'grey level' data in weightless networks. In *RAM-based Neural Networks*, pages 61–70. World Scientific, 1998.

[28] P. Müller, F. A. Quintana, A. Jara, and T. Hanson. *Bayesian nonparametric data analysis*, volume 1. Springer, 2015.

[29] S. Purdy. Encoding data for HTM systems. *arXiv preprint arXiv:1602.05925*, 2016.

[30] T. Singh, R. Kalra, S. Mishra, Satakshi, and M. Kumar. An efficient real-time stock prediction exploiting incremental learning and deep learning. *Evolving Systems*, 14(6):919–937, 2023.

[31] R. Sousa, T. Lima, A. Abelha, and J. Machado. Hierarchical temporal memory theory approach to stock market time series forecasting. *Electronics*, 10(14):1630, 2021.

[32] J. Xuan, J. Lu, and G. Zhang. A survey on bayesian nonparametric learning. *ACM Computing Surveys (CSUR)*, 52(1):1–36, 2019.

APPENDIX

A. BELIEF ENCODING

Algorithm 4 indicates how to encode the categorical distribution $\boldsymbol{\pi}$ with c categories into a sparsely encoded belief $\mathbf{b}$ in the binary space $\mathbb{Z}_2^{c+p}$ with up to p activated bits. Note that this is a lossy encoding scheme. Besides the floor operation in Line 5, which rounds off the scaled probabilities $p \cdot \pi_i$, $\forall i \in \{1, \ldots, c\}$, non null probabilities $\pi_i > 0$ and $\pi_j > 0$, $i \neq j$, in $\boldsymbol{\pi}$ with close indexes $|i - j| < p$ can generate a sequence of contiguous active bits in $\mathbf{b}$. Thus, as illustrated in the running example in Sec. 3 (inspect Fig. 2), depending on the probability values π_i and π_j, multiple valid categorical distributions can be mapped to the same belief by this algorithm.

Algorithm 4 $Bel\left(\boldsymbol{\pi} = \begin{bmatrix}\pi_1 & \ldots & \pi_c\end{bmatrix}^\top; p\right)$

1: Initialize $\mathbf{b} = \begin{bmatrix}b_1 & \ldots & b_{c+p}\end{bmatrix}^\top$ such that, $\forall i \in \{1, 2, \ldots, c+p\}$, $b_i \leftarrow 0$.
2: $acc \leftarrow 0$
3: **for** $i \leftarrow 1$ **to** $c + p$ **do**
4: **if** $i \leq c$ **then**
5: $acc \leftarrow acc + \lfloor p \cdot \pi_i + \frac{1}{2} \rfloor$
6: **end if**
7: **if** $acc > 0$ **then**
8: $b_i \leftarrow 1$
9: $acc \leftarrow acc - 1$
10: **end if**
11: **end for**
12: **return b**

Algorithm 5 shows in turn how to decode the sparsely encoded belief $\mathbf{b}$ into the posterior $\boldsymbol{\pi}$ within the $(c-1)$-dimensional simplex $\mathbb{S}_c$. It is worth noting that this algorithm will assign a non null probability π_i to the first symbol belonging to a contiguous sequence $b_i, b_{i+1}, b_{i+2}, \ldots$ of activate bits in $\mathbf{b}$. Thus, one should avoid assigning similar symbols to close locations in $\boldsymbol{\pi}$ such that it is unlikely to produce a single contiguous sequence of active bits within $\mathbf{b}$. Otherwise, the $Bel(\cdot)$ procedure in Algorithm 4 will activate overlapping sequences of contiguous bits in the binary space $\mathbb{Z}_2^{c+p}$ for those symbols, therefore yielding an ambiguous representation which cannot be properly recovered by the $Bel^{-1}(\cdot)$ procedure in Algorithm 5.

Algorithm 5 $Bel^{-1}\left(\mathbf{b} = \begin{bmatrix}b_1 & \ldots & b_{c+p}\end{bmatrix}^\top; p\right)$

1: Initialize $\boldsymbol{\pi} = \begin{bmatrix}\pi_1 & \ldots & \pi_c\end{bmatrix}^\top$ such that, $\forall i \in \{1, 2, \ldots, c\}$, $\pi_i \leftarrow 0$.
2: $acc \leftarrow 0$
3: **for** $i \leftarrow c + p$ **to** 1 **do**
4: $acc \leftarrow acc + b_i$
5: **if** $i \leq c \wedge (i = 1 \vee b_{i-1} = 0)$ **then**
6: $\pi_i \leftarrow acc$
7: $acc \leftarrow 0$
8: **end if**
9: **end for**
10: Normalize $\boldsymbol{\pi}$ such that $\sum_{i=1}^{c} \pi_i = 1$.
11: **return** $\boldsymbol{\pi}$

B. SEQUENCE LEARNING

According to assumption (30), $p(\mathbf{z}_{n+1} \mid \tilde{\boldsymbol{\pi}}_{n+1|n})$ is a categorical distribution. Thus, for a particular realization $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}$ of $\mathbf{Z}_{n+1}$, the likelihood function becomes

$$\begin{aligned} p(\check{\mathbf{z}}_{n+1} \,|\, \tilde{\boldsymbol{\pi}}_{n+1|n}) &= p(\mathbf{z}_{n+1} \,|\, \tilde{\boldsymbol{\pi}}_{n+1|n})\big|_{\mathbf{z}_{n+1}=\check{\mathbf{z}}_{n+1}} \\ &= \sum_{\ell=1}^{c_z} \tilde{\pi}_{n+1|n}^{(\ell)} \cdot [\![\mathbf{z}_{n+1} = \boldsymbol{\mu}_z^{(\ell)}]\!]\Big|_{\mathbf{z}_{n+1}=\boldsymbol{\mu}_z^{(k)}} \\ &= \tilde{\pi}_{n+1|n}^{(k)}. \end{aligned} \tag{33}$$

From Eq. (33) and assumption (25), it follows that we can write the updated posterior $p(\tilde{\boldsymbol{\pi}}_{n+1|n} \,|\, \check{\mathbf{z}}_{0:n+1})$ using the Bayes rule as

$$\begin{aligned} p(\tilde{\boldsymbol{\pi}}_{n+1|n} \,|\, \check{\mathbf{z}}_{0:n+1}) &\propto p(\check{\mathbf{z}}_{n+1} \,|\, \tilde{\boldsymbol{\pi}}_{n+1|n})\, p(\tilde{\boldsymbol{\pi}}_{n+1|n} \,|\, \check{\mathbf{z}}_{0:n}) \\ &= \frac{\tilde{\pi}_{n+1|n}^{(k)}}{B(\tilde{\boldsymbol{\alpha}}_{n|n-1})} \prod_{\ell=1}^{c_z} \left(\tilde{\pi}_{n+1|n}^{(\ell)}\right)^{\tilde{\alpha}_{n|n-1}^{(\ell)}-1} \\ &\propto \tilde{\pi}_{n+1|n}^{(k)} \prod_{\ell=1}^{c_z} \left(\tilde{\pi}_{n+1|n}^{(\ell)}\right)^{\tilde{\alpha}_{n|n-1}^{(\ell)}-1}, \end{aligned} \tag{34}$$

since the Beta function $B(\tilde{\boldsymbol{\alpha}}_{n|n-1})$ is constant for a given $\tilde{\boldsymbol{\alpha}}_{n|n-1}$. We can rewrite thus the right-hand side of (34) as

$$\left(\tilde{\pi}_{n+1|n}^{(k)}\right)^{\left(\tilde{\alpha}_{n|n-1}^{(k)}+1\right)-1} \prod_{\ell=1,\ell\neq k}^{c_z} \left(\tilde{\pi}_{n+1|n}^{(\ell)}\right)^{\tilde{\alpha}_{n|n-1}^{(\ell)}-1}. \tag{35}$$

Thus, given $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}$, the updated posterior can be rewritten as

$$p(\tilde{\boldsymbol{\pi}}_{n+1|n} \,|\, \check{\mathbf{z}}_{0:n+1}) = \frac{1}{B(\tilde{\boldsymbol{\alpha}}_{n+1|n})} \prod_{\ell=1}^{c_z} \left(\tilde{\pi}_{n+1|n}^{(\ell)}\right)^{\tilde{\alpha}_{n+1|n}^{(\ell)}-1} \tag{36}$$

with new normalization constant given by the reciprocal of $B(\tilde{\boldsymbol{\alpha}}_{n+1|n})$ such that, conditioned on both $\check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}$ and $\check{\mathbf{z}}_{0:n}$, $\tilde{\boldsymbol{\Pi}}_{n+1|n}$ is also Dirichlet

$$\tilde{\boldsymbol{\Pi}}_{n+1|n} \,|\, \check{\mathbf{z}}_{n+1} = \boldsymbol{\mu}_z^{(k)}, \check{\mathbf{z}}_{0:n} \sim Dir(c_z; \tilde{\boldsymbol{\alpha}}_{n+1|n})$$

with new hyperparameters $\tilde{\boldsymbol{\alpha}}_{n+1|n} = \tilde{\boldsymbol{\alpha}}_{n|n-1} + \mathbf{1}_k$.
Using a similar approach, given assumptions (23) and (27), we can show that, conditioned on a particular realization $\check{\mathbf{s}}_{n+1} = \boldsymbol{\mu}_s^{(j)}$ of $\mathbf{S}_{n+1}$, the Bayes rule application

$$\begin{aligned} p(\boldsymbol{\pi}_{n+1|n} \,|\, \check{\mathbf{s}}_{0:n+1}) &\propto p(\check{\mathbf{s}}_{n+1} \,|\, \boldsymbol{\pi}_{n+1|n})\, p(\boldsymbol{\pi}_{n+1|n} \,|\, \check{\mathbf{s}}_{0:n}) \\ &= \frac{\pi_{n+1|n}^{(j)}}{B(\boldsymbol{\alpha}_{n|n-1})} \prod_{\ell'=1}^{c_s} \left(\pi_{n+1|n}^{(\ell')}\right)^{\alpha_{n|n-1}^{(\ell')}-1} \\ &\propto \pi_{n+1|n}^{(j)} \prod_{\ell'=1}^{c_s} \left(\pi_{n+1|n}^{(\ell')}\right)^{\alpha_{n|n-1}^{(\ell')}-1} \end{aligned} \tag{37}$$

yields an updated posterior

$$p(\boldsymbol{\pi}_{n+1|n} \,|\, \check{\mathbf{s}}_{0:n+1}) = \frac{1}{B(\boldsymbol{\alpha}_{n+1|n})} \prod_{\ell'=1}^{c_s} \left(\pi_{n+1|n}^{(\ell')}\right)^{\tilde{\alpha}_{n+1|n}^{(\ell')}-1} \tag{38}$$

with new normalization constant given now by the reciprocal of $B(\boldsymbol{\alpha}_{n+1|n})$, which is also Dirichlet

$$\boldsymbol{\Pi}_{n+1|n} \,|\, \check{\mathbf{s}}_{n+1} = \boldsymbol{\mu}_s^{(j)}, \check{\mathbf{s}}_{0:n} \sim Dir(c_s; \boldsymbol{\alpha}_{n+1|n})$$

with new hyperparameters $\boldsymbol{\alpha}_{n+1|n} = \boldsymbol{\alpha}_{n|n-1} + \mathbf{1}_j$.