Expanding Neuro-Symbolic Artificial Intelligence for Strategic Learning

Dorian Clay
dorian@dorianclay.com
Santa Clara University
Santa Clara, CA, USA

David Anastasiu
danastasiu@scu.edu
Santa Clara University
Santa Clara, CA, USA

ABSTRACT

Today, artificial intelligence (AI) solves problems as varied as driving cars to diagnosing diseases. However, it still has significant pitfalls. For example, the most common form of AI today, machine learning, can produce near-certain predictions for some tasks yet still cannot do reasoning for complex problems and are not well explainable.

One solution to these problems is combining today's cutting-edge neural networks with an older idea in AI: symbolic AI. This *neuro-symbolic artificial intelligence* has already been shown to excel at visual question answering, a task that involves answering a question about something happening in an image. In the future, such algorithms may play a role in making more flexible and intelligent AI, such as for autonomous driving systems. Right now, neuro-symbolic AI has many areas for expansion across a wide range of applications.

This paper will seek to apply neuro-symbolic AI to strategic game play where AI agents have the same information as a human player—just an image. This paper develops the Blokboi game as an AI training and testing tool to enable testing abstraction and high-level reasoning. Blokboi pushes an AI agent to learn scene interpretation and strategic planning and provides an environment that tests an AI's ability to learn compound interpretation and reasoning. This research expands the applications of NSAI, bringing hybrid artificial intelligence increasingly close to real-world scenarios.

CCS CONCEPTS

• **Computing methodologies** → **Planning for deterministic actions**; **Scene understanding**; *Reinforcement learning*; *Planning with abstraction and generalization.*

KEYWORDS

neuro-symbolic AI, hybrid AI, neural networks, reinforcement learning, game AI

ACM Reference Format:
Dorian Clay and David Anastasiu. 2022. Expanding Neuro-Symbolic Artificial Intelligence for Strategic Learning. In *ACM SIGKDD Undergraduate Consortium (KDD-UC '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 7 pages.

KDD-UC '22, August 14–18, 2022, Washington, DC, USA.

1 INTRODUCTION

Neuro-symbolic artificial intelligence (NSAI) is a relatively recent hybrid approach to artificial intelligence (AI) systems that combines a neural network with a symbolic AI system. Together, these two systems allow one to handle raw data interpretation, like a human sensory cortex, and the other to act on symbolic information, like a prefrontal cortex. These NSAI systems excel in complex reasoning areas that machine learning (ML) have struggled at, and are particularly promising for problems that include both interpretation and decision making [16] [1]. This paper will lay the groundwork to explore whether hybrid AI approaches are superior to ML-based approaches for strategic learning. It develops the Blokboi [1] training game as a tool to train and test reinforcement learning and neuro-symbolic AI agents.

1.1 Motivation

Neuro-symbolic AI have been proposed as a kind of hybrid approach which addresses some of the shortcomings of ML-based approaches. Stronger reasoning, flexibility, and explicability are some of the goals of NSAI. This is primarily achieved by disentangling *interpretation* from *reasoning*. Interpretation is the process of parsing some raw input, such as identifying objects in an image or words in an audio recording. Reasoning is the higher-level process of deriving an understanding of the things in raw input, like knowing something about the objects in the image or the meaning of the audio recording [16]. In the future, NSAI may be used to improve solutions where AI are already used, such as robotics, autonomous vehicles, or digital assistants. In these applications, the AI algorithm is already only one component in a larger system. However, having a hybrid model may allow different types of AI to interact more closely together as one cohesive unit.

As mentioned, NSAI also seem promising for robotics and autonomous vehicles, where a device must be able to understand and navigate in a physical space using sensors. This is a kind of strategic reasoning problem, which this paper will seek to address. Essentially, rather than the model being asked to answer a question for each given scene, as in visual reasoning problems, the strategic reasoning problem will task the model with performing some action for each given scene. In doing so, this paper will seek to answer the question: can a model develop a strategy to reach some goal, based on information derived from raw sensory input, such as an image?

[1] Source code repository: https://github.com/glxiia/blokboi

1.2 Proposed Solution

This paper will develop the Blokboi training game as a tool to test this question and will provide preliminary performance of a hard-coded heuristic approach as well as a reinforcement learning model. The Blokboi game mimics the kind of high-level tasks that would need to be performed by an autonomous system. For a model to perform well in the Blokboi game, it must interpret an image of the scene—without knowledge of the internal state of the scene, such as object locations—as well as reason about how to accomplish the goal specified with each scene. This challenges the AI to learn both interpretation and reasoning. Yet unlike visual question-answering (VQA) problems, Blokboi tasks the model with reasoning about how to interact with a scene, rather than just answering a question.

The two initial solutions to playing Blokboi, the heuristic and reinforcement learner, show that there is significant room for NSAI to outperform the existing approaches. However, proving the NSAI approach is superior is left to future work at this time. Instead, this paper develops the tools necessary to test the hypothesis.

Section 2 will survey the existing approaches to related issues of VQA, video VQA, and physical dynamics, where NSAI have already been applied. This informs how the Blokboi game will be designed to test neuro-symbolic and other AI approaches. Then, Section 3 will expand on how the game and two proposed solutions were developed. Section 4 describes the experimental design and results of testing the solutions to the Blokboi game. Section 5 discusses how this project can be expanded in the future. Section 6 explores hybrid AI as a solution to the explicability problem for AI, and Section 7 provides closing thoughts.

2 RELATED WORKS

Approaches that disentangle the visual and language 'sensory' understanding from reasoning are referred to as 'neuro-symbolic,' or sometimes 'neural-symbolic,' models, and here referred to as neuro-symbolic AI (NSAI) [16]. In an effort to better address complex reasoning problems that deep learning fails at, NSAI seeks to separate the task of interpreting an input from reasoning about the items represented by that input.

Neuro-symbolic AI has already been proven to outperform ML-based approaches in some areas. Research around NSAI has largely centered around two related but distinct problems thus far: visual question-answering (VQA) and learning physical dynamics. In VQA problems, a system is given an image-question pair and tasked with answering the question. This task is hard for even the cutting edge computer-vision systems, because inputs cannot easily be mapped to outputs. On the other hand, NSAI are particularly suited for visual reasoning, because of how they decouple the interpretation of a visual scene from the reasoning process [16]. Similarly, NSAI have also been applied to learning physical dynamics from videos [3], where they separate detect objects in a video and learn the phycial relationships of those objects. For Blokboi, it is expected that a high-performing model will detect objects in the scene separately from reasoning about how to solve the scene's puzzle.

Yi et. al 2018 [16] defined a *neural-symbolic visual question answering* (NS-VQA) model that creates a scene representation from an image and a symbolic program trace from the associated question. They dealt only with static images in their VQA problem, and their NS-VQA model is split into three components: 1) a scene parser that identifies objects in an image and predicts categorical labels such as color, size, material, and shape, 2) a question parser that interprets the text paired with each image as a symbolic program, and 3) a program executor that runs the symbolic program on the objects extracted from the scene. Yi et al. compared the results of their model on the CLEVR dataset [9], getting an accuracy score of 99.8%, showing that their model outperformed the *inferring and executing programs* model proposed by Johnson et al. [10], which got 96.9% accuracy.

Amizadeh et al. [1] proposed a framework to disentangle the evaluation of improvements in perception from those in reasoning. They established how most advances in visual reasoning in fact come from improvements in perception—object detection and scene representation—rather than improvements in reasoning of a VQA model. They defined a *differentiable first-order logic framework* (∇-FOL), which fully disentangles representation learning from the reasoning mechanism, and provided a formal basis for this separation. They compared their results to two existing state-of-the-art models, which seem to indicate that the ∇-FOL framework often but not always performed better than their baseline. This may indicate some performance is sacrificed for fully separating perception from reasoning.

Yi et al. 2020 [15] propose a dataset, CLEVRER, for video VQA. They emphasized the importance of temporal and causal elements lacking in existing benchmarks, a gap which CLEVRER fills in. They showed how a number of existing language-only, video question answering, and compositional visual reasoning models performed generally mediocre on the dataset. Yi et al. also proposed a *neuro-symbolic dynamic reasoning* (NS-DR) model, which performed markedly better, but still not near perfectly, getting at highest an accuracy of 88.1%. The model proposed by Yi et al. still requires a small amount of supervised learning, and did not perform well on counterfactual questions–like those that ask what would happen in a scene if some object was not there.

Baradel et al. [2] propose a benchmark for the problem of predicting alternative outcomes of a physical situation, given the original past and outcome and an alternative past. This is similar to counterfactual video VQA, the area where the model proposed by Yi et al. with the CLEVRER dataset [15] did not perform well. Baradel et al.'s proposed approach differs from that of most of the other papers discussed here, as they used a single deep neural network which is biased to favor counterfactual reasoning. They compared the results of their model against human performance, showing it was worse. However, their model still performed better than simpler neural network models.

3 METHODS

A new tool was needed to test hybrid AI's capacity of hybrid AI to learn strategic reasoning and abstraction. It needed to be able to 1) train AI agents, 2) test strategic learning, 3) require AI to perform both interpretation and reasoning, and 4) have flexible goals for AI to achieve. After exploring existing games such as chess and Kaggle Inc.'s Halite [4], these goals seemed to be achieved best by creating a training game from scratch.

A good game to test the AI's ability to learn strategy and abstraction would be one with simple rules that result in complex systems. A new game, Blokboi, was created. The Block Dude game on a Texas Instruments TI-84 served as initial inspiration, but Blokboi has different rules, modified mechanics, and flexible goals. In Blokboi, the player must move the blocks around a randomly generated map to create a pattern specified by an accompanying instruction string (see the example in Figure 1). Table 1 specifies the actions available in Blokboi. Section 3.1 will expand on the development process of the training game. After completing the Blokboi training game, an initial heuristic solution was created and tested, then a basic reinforcement learning (RL) agent. These will be discussed in sections 3.2 and 3.3.

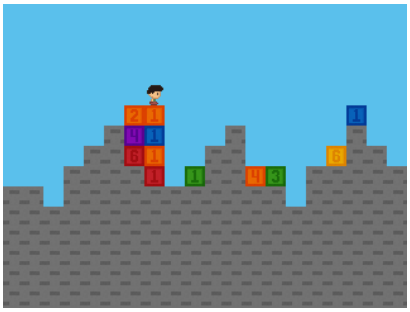

Figure 1: A randomly generated map with the instruction string "Put the yellow block next to the purple block."

Table 1: Actions Available in Blokboi

Action	Result
Left	Move blokboi one column left or turn around if facing right.
Right	Move blokboi one column right or turn around if facing left.
Jump	If facing a block at the same height, move one column in that direction and up one row, such that blokboi is now on top of the block.
Toggle hold	If holding a block, drop in in the direction facing one row above blokboi, falling with gravity. Otherwise, pick one up if next to it and at the same level.

3.1 Developing the Blokboi Training Game

The Blokboi game was developed from scratch to enable training and testing various AI agents efficiently. It has three primary components: a core C++ game library which manages the logic of the game, a wrapper which exposes the game to Python, and an OpenAI Gym environment for running the game with various algorithms (see Figure 2).

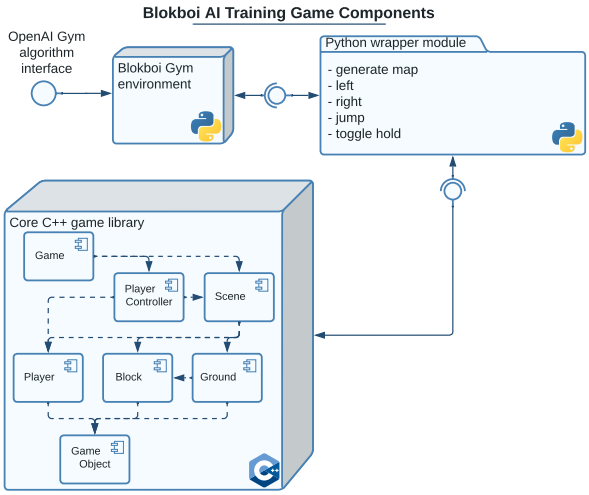

Figure 2: Blokboi component diagram

3.1.1 Core C++ Game Library. The C++ library exposed functionality to create a game, reset the current game, move left or right, jump, toggle holding a block, get an array representation of the map, get a string representing the game actions performed, verify whether the goal was achieved, and score a given solution string for the map.

The library can be used directly in C++ by including the relevant header files. However, the game library would not be accessible from Python, where the NSAI agent and other tested agents would be developed. As such, a wrapper is needed for the library. Some of these library functions are also included as part of the RL agent, which will be expanded on in Section 3.3.

3.1.2 Python Wrapper Module. The Blokboi C++ library is wrapped and provided as a Python package so AI agents written in Python can interact with the game. Python binding was done with pybind11 because a library in C++ was easier to use with C++ source code. Some functionality available in the C++ Blokboi library was not exposed to Python in order to create a safe and stable interface. For instance, the C++ interface allows a client to directly access player metadata, but the bound Python interface does not expose this functionality.

3.1.3 Gym Environment. An OpenAI Gym environment [7] was created for Blokboi in order to provide a consistent interface to the game for use by various algorithms. The Gym interface provides a game as an *environment*. At each step in the game, the an algorithm provides an action and gets back an observation, which is the game state. With the simple and robust Python wrapper, providing Blokboi as a Gym environment was very straightforward.

Even though the Gym environment was the easiest component of the Blokboi system to develop, it was critical to creating and testing AI models on the game. The simple API made it easy to implement and test existing RL agents.

3.2 Heuristic Solution

Creating a heuristic-based solution to solve Blokboi scenes played two roles in making Blokboi useful for testing AI strategic and abstraction learning. First, the heuristic approach is used to filter out maps with no solution. Second, the heuristic solution provides the first baseline for performance on each map generated. As mentioned in Section 3.1.1, a randomly-generated map is only accepted and provided as the game map after it has been solved by the heuristic approach. If it is solved at this point, then the actions taken by the heuristic are saved with the map as the baseline performance of a naïve, explicitly-programmed solution to playing Blokboi.

This approach works by formalizing several assumptions about how a human might play Blokboi as procedures, and by using information about the game state at each turn. At the highest level, the heuristic approach attempts to bring the first block specified by the instruction to the second block, then arranges the blocks in the given pattern. Each of these more 'abstract' procedures is implemented by simply trying to get the player to a specific column, either with or without a block. In turn, getting to a column is performed by moving the player towards that column or recursively searching for blocks required to allow the player to continue left, right, or jump, shown in Algorithm 1. These algorithms were implemented with values *INSURANCESTEPS* = 10000, and *WALKATTEMPTS* = 100.

It is important to note the heuristic approach does not satisfy the goal of simultaneous interpretation and reasoning because the algorithm is given game state information such as the location of all game objects and blocks that fit the instruction string, instead of being provided only an image of the scene. However, it would be reasonable for an explicitly coded solution to manually extract block locations from pixels and parse the instruction string for relevant words. As such, allowing the heuristic to directly access the game information seems to be an acceptable concession in this circumstance. Still, an approach that uses the exact same information as the proposed hybrid approach is needed as a baseline to compare NSAI against.

3.3 Reinforcement Learning Agent

The reinforcement learning (RL) agent provides a best-effort baseline representing the current state-of-the-art. RL works by rewarding the AI agent when it decides to perform an action that results in a positive outcome, and punishing the agent when it decides on an action that yields a negative outcome. With the OpenAI Gym interface implemented for Blokboi, it was relatively straight-forward to create an RL agent using existing open-source algorithms. This project used the Gym-compatible Stable Baselines [8] algorithms due to their extensive documentation and ease of use. Specifically, Proximal Policy Optimization (PPO2) [12] was selected for its ability to work in a Gym with discrete actions and GPU multi-processing. Though more algorithms and configurations should be tested in the future, this selection was necessary due to limited available training time.

The RL agent required new maps to be generated on the fly for training, which proved to be a challenging problem. The randomly-generated maps needed to be created upon request and known to be solvable. Initial attempts to solve the generation problem were based on manually designing rules required for a playable

Algorithm 1 Path Searching Algorithm

```
procedure GET_TO_COLUMN(column, steps)
    attempts ← 0
    direction
    if player_column < column then
        direction ← 1
    else
        direction ← −1
    while attempts < WALKATTEMPTS do
        attempts ← attempts + 1
        success ← walk_to(column)
        if success then return
        else
            see if blocked by a wall or a cliff
            if holding a block then
                if against a wall then
                    maneuver to place block on current column
                    steps ← steps + 1
                else
                    holding a block on a cliff, so just drop it
                    steps ← steps + 1
            else
                if against a wall then
                    nextcolumn ← player_column − direction
                else if cliff and standing on available block then
                    move to pick up block player is currently on
                    steps ← steps + 1
                    continue
                buildingblock ← farthest available block
                if player is standing on the needed block then
                    walk_to(player_column − direction)
                    walk_to(player_column)
                    pick up block
                    steps ← steps + 1
                    walk_to(nextcolumn)
                    place block
                    steps ← steps + 1
                else if player can walk straight to the block
    then
                    walk_to(buildingblock)
                    pick up block
                    steps ← steps + 1
                    walk_to(nextcolumn)
                    place block
                    steps ← steps + 1
                else if player must cut back to get block then
                    walk_to(buildingblock - direction)
                    walk_to(player_column
                    pick up block
                    steps ← steps + 1
                    walk_to(nextcolumn)
                    place block
                    steps ← steps + 1
```

map. However, this proved insufficient, so a multi-layer system was used instead. This approach first generates a map with parameters that give it a reasonable chance of being playable, such as terrain contour and a minimum and maximum block height. It then fills the map with the number of blocks needed to create linear steps out of all the map contour, such that there is no step up greater than one block high. Then, it checks several heuristics for patterns that make a map unsolvable. Lastly, the naïve heuristic-based algorithm tries to play the map. If the map fails any of these tests, the system throws it out and generates a new one recursively.

Additionally, the RL agent needs scores for each action in order to reward or punish the agent. These verification scores are the inverse of the final solution scoring, because the RL agent seeks to maximize scores. So, each step taken yields a score of -1, and each invalid action gets an additional -100. This punishes the agent for inaccurate moves, such as attempting to walk off a cliff. If the solution is reached, the agent gets a score of 1. For each game the RL agent plays, the beginning game state and final solution are saved for later evaluation.

4 EVALUATION

4.1 Experimental Design

To illustrate how solving Blokboi requires more than just simple solutions, two approaches based on common algorithms were tested. The first was a heuristic approach (hereon called the 'heuristic'), whose algorithm was static and based on human learning, rather than computer learning. The second was a reinforcement learning (RL) agent, intended as the baseline for future work.

Algorithms were tested based on their ability to correctly solve for the challenge pattern in Blokboi scenes. Each step in a solution was scored, such that the final score on a scene is the total of all the step scores. The best score on a Blokboi scene is the lowest score. The lowest possible solution score for each scene is the number of steps required to create the goal pattern with no mistakes or extraneous actions. If any invalid action is performed, such as attempting to pick up a block that does not exist, a penalty is added to the score. Furthermore, if the end of the solution is reached without achieving the goal pattern, an even higher penalty is given. If an undefined action occurs, the highest penalty is given. These scores are summarized in Table 2.

Table 2: Blokboi Solution Scoring

Case	Score
Any action	1
Invalid action	100
Goal pattern not satisfied	1000000
Undefined action	1000000000

For this experiment, the baseline is the heuristic, which is used to validate all scenes that are provided to the models. Given the design of this heuristic, it was predicted that the RL model should easily be able to outperform it, yet would still have disappointing performance on its own.

Because the heuristic is designed from human learning and does no automated learning, no training was done for it. The RL agent, on the other hand, was trained on 5,000 scenes. All Blokboi scenes are randomly generated on the fly, so no train-test split was performed.

Tests were performed on a PC with an Intel i9-9900KF, two Nvidia Titan XPs, 62GB of memory, running Linux 18.04. The project requires C++17 and Python 3.8, and has dependencies for Pybind11, Numpy, Pandas, Pillow, Tensorflow 1.15, Gym 0.21, and Stable Baselines.

4.2 Experimental Results

The heuristic and RL approaches to playing Blokboi were tested, with the hypothesis that the RL would outperform the heuristic, but still have poor performance. To test this, both models played 500 scenes of the game, with a maximum of 10,000 steps allowed. Scenes and the models' solutions to those scenes were saved, then scored after all solutions were generated. As mentioned, scores were assigned using the scheme in Table 2. Results from this experiment are shown in Figure 3 and Figure 4, for the heuristic and RL agents, respectively.

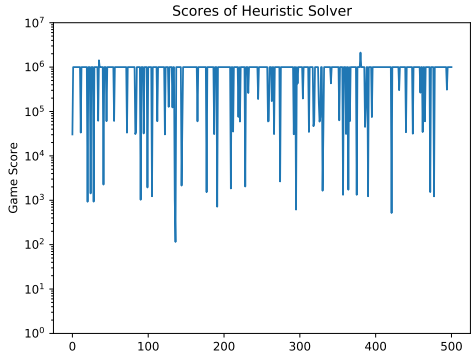

Figure 3: Scores for solutions generated by the heuristic.

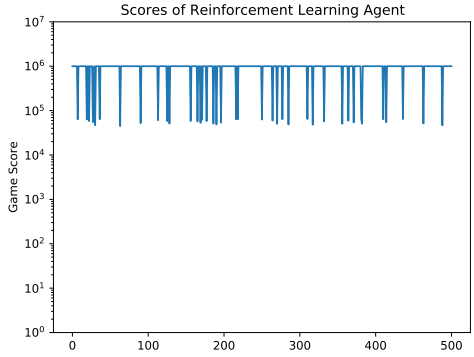

Figure 4: Scores for solutions generated by the PPO2 RL algorithm.

These results show that the RL agent did not outperform the heuristic. The heuristic does not achieve good performance, but its solutions have lower scores than the RL agent. The heuristic was able to solve 73 of the 500 scenes within the 10,000-step limit, which yields a 14.6% success rate. The RL agent was able to solve only 40 of the 500 scenes, for an 8% success rate. This was surprising, given that the heuristic is repeatedly trying the same pattern with some randomness until it achieves the result. However, it does make sense, as the RL agent is playing the scene effectively blind—it doesn't have much ability to disentangle interpretation of the raw image from reasoning about the scene. A better RL approach may involve explicitly training computer vision separately from the agent.

Furthermore, Figure 4 shows the RL model often provided solutions that did not achieve the instructed pattern of blocks, but also had a number of solutions that did correctly solve a map. When the map was solved, the scores were still poor, often indicating that the model performed on the order of 1000 invalid actions, such as attempting to walk into a block or jump when there is no block to jump onto.

The performance of the heuristic and RL solutions demonstrate that neither is capable of effectively solving the maps in the Blokboi game. Their lack of efficacy show why a neuro-symbolic hybrid model may have room to improve here, particularly because NSAI can better separate interpreting blocks from the image and the reasoning that must be performed.

5 FUTURE WORK

After developing the Blokboi game as a tool for testing AI's ability to learn strategy and abstraction, this project can be expanded upon in several ways. Most importantly, an NSAI model should be developed and tested to verify the hypothesis that it outperforms both the explicitly-coded heuristic and the RL model. It is unclear at this time whether a high performing NSAI model will beat human performance. It is possible, because human players tend to make mistakes like moving to an occupied space. Additionally, the RL baseline can be refined and given a better chance of competing against an NSAI model. The current RL model has room for further optimization and testing against other algorithms, such one based on the Aloe model by Ding et al. [5]. As a matter of code base hygiene, the existing RL model should at least be updated to use the newer Stable Baselines 3 [11] repository.

In addition to testing the hypothesis that hybrid AI will outperform current leading approaches, some improvements can be made to the underlying game itself. The maps generated by the game and provided to the models are currently limited by the heuristic approach to playing the game autonomously. Each map must be first played by the heuristic before it is given to the target model as a training sample. However, the heuristic is limited by its design and is unable to solve all maps with a solution. It only confirms that maps which pass the heuristic component of map generation have a solution. As such, any model's learning is limited by the heuristic's ability to solve the maps. This likely results in a bias in the models towards strategies that the patterns the heuristic is able to solve.

6 ETHICAL CONSIDERATIONS

One of the suggested benefits of NSAI is better explicability. Because NSAI use more easily interpretable symbolic logic for reasoning, it is proposed that the outcomes of such algorithms are more explicable [16]. However, it is unclear the reality is matches this optimistic claim. Explicability is a difficult concept to define, but a working definition for this paper is created by combining the shared elements of various definitions of explicability. For a system to have good *explicability* is for 1) its internal workings to be explainable to a wide range of stakeholders, from specialist to the general public [14][6][13], 2) for the design choices made in the process of creating the system to be documented [6][13], and 3) for information to be provided about how outcomes of a system are used in decision making [13][6].

Hybrid AI contribute to efforts of making machine learning more explicable by separating fundamental parsing tasks from reasoning about features. As a byproduct of hybrid AI's design, how the system arrived at a conclusion can be explained by the symbols deduced symbols and symbolic logic used the question. This is intuitively attractive because it provides the strong input interpretation of machine learning and the reasoning and explicability of symbolic AI. However, the fundamental reliance of a hybrid system on opaque ML subsystems highlights a concern that extends to other attempts to take a machine learning system and make it explicable. Even when a more explicable model can be produced based on the learning of a ML model, it may be subject to the errors and subsequent procedural unfairness of the underlying model.

7 CONCLUSION

This project has developed the Blokboi game as a robust tool for testing the compound scene interpretation and strategic learning of AI models. Experimentation has demonstrated that both a bespoke heuristic solution and a reinforcement learning baseline leave significant room for improvement in solutions to playing the game. These existing solutions are bogged down either by having no learning ability, as in the case of the heuristic, or by being incapable of simultaneously interpreting and reasoning, as is the case for the reinforcement learner. This can be immediately expanded to test how much better neuro-symbolic AI can perform at the game. More broadly, this project begun work in exploring how hybrid AI can be used for systems that not only reason about, but also interact with, a physical environment. Such work may have future applications to autonomous robots and vehicles.

This paper has also discussed whether hybrid AI provide a feasible solution to the explicability problem for AI. It found that some parts of the hybrid AI may make it easier to laypersons to understand how an AI system operates for a specific decision, but that the underlying reliance on machine learning leaves outstanding concerns. In some instances, the smaller role of machine learning in hybrid approaches may even unreasonably diminish concerns about the 'black box' issue for AI.

By developing the game Blokboi to encourage complex AI learning and reasoning and by providing early experimentation about the performance of common existing approaches, this project has laid the groundwork for continued research into how hybrid artificial intelligence can perform in increasingly real-world scenarios.

REFERENCES

[1] Saeed Amizadeh, Hamid Palangi, Alex Polozov, Yichen Huang, and Kazuhito Koishida. 2020. Neuro-symbolic visual reasoning: disentangling "visual" from "reasoning". In *Proceedings of the 37th International Conference on Machine Learning* (Proceedings of Machine Learning Research). Hal Daumé III and Aarti Singh, (Eds.) Vol. 119. PMLR, (July 2020), 279–290. https://proceedings.mlr.press/v119/amizadeh20a.html.

[2] Fabien Baradel, Natalia Neverova, Julien Mille, Greg Mori, and Christian Wolf. 2020. Cophy: counterfactual learning of physical dynamics. *International Conference on Learning Representations*. http://arxiv.org/abs/1909.12000.

[3] Zhenfang Chen, Jiayuan Mao, Jiajun Wu, Kwan-Yee Kenneth Wong, Joshua B. Tenenbaum, and Chuang Gan. 2021. Grounding physical concepts of objects and events through dynamic visual reasoning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=bhCDO_cEGCz.

[4] Alex Cook. 2022. Getting started with halite. (Jan. 2022). Retrieved June 1, 2022 from https://kaggle.com/alexisbcook/getting-started-with-halite.

[5] David Ding, Felix Hill, Adam Santoro, Malcolm Reynolds, and Matt Botvinick. 2021. Attention over learned object embeddings enables complex visual reasoning. In *Advances in Neural Information Processing Systems 34*. arXiv: 2012.08508 [cs.CV].

[6] Luciano Floridi and Josh Cowls. 2019. A unified framework of five principles for ai in society. *Harvard Data Science Review*, 1, 1, (July 1, 2019). DOI: 10.1162/99608f92.8cd550d1.

[7] [SW], Gym version 0.24.0, June 3, 2022. URL: https://github.com/openai/gym.

[8] Ashley Hill et al. 2018. Stable baselines. https://github.com/hill-a/stable-baselines. (2018).

[9] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. 2017. Clevr: a diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (July 2017).

[10] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. 2017. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. (Oct. 2017).

[11] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-baselines3: reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22, 268, 1–8. http://jmlr.org/papers/v22/20-1364.html.

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[13] Andrew D. Selbst and Solon Barocas. 2018. The intuitive appeal of explainable machines. *SSRN Electronic Journal*. DOI: 10.2139/ssrn.3126971.

[14] The IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems. 2019. *Ethically Aligned Design: A Vision for Prioritizing Human Well-being with Autonomous and Intelligent Systems*. en. (1st ed.). IEEE. Retrieved Apr. 16, 2022 from https://standards.ieee.org/content/ieee-standards/en/industry-connections/ec/%20autonomous-systems.html.

[15] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. 2020. Clevrer: collision events for video representation and reasoning. In *International Conference on Learning Representations*. https://arxiv.org/abs/1910.01442.

[16] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B Tenenbaum. 2018. Neural-symbolic vqa: disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems (NIPS)*.