LLMs Enhance DL Performance Prediction

Karthick Panner Selvam[1], Phitchaya Mangpo Phothilimthana[2], Sami Abu-El-Haija[2], Bryan Perozzi[2], Mats Brorsson[1]
[1]University of Luxembourg, [2]Google

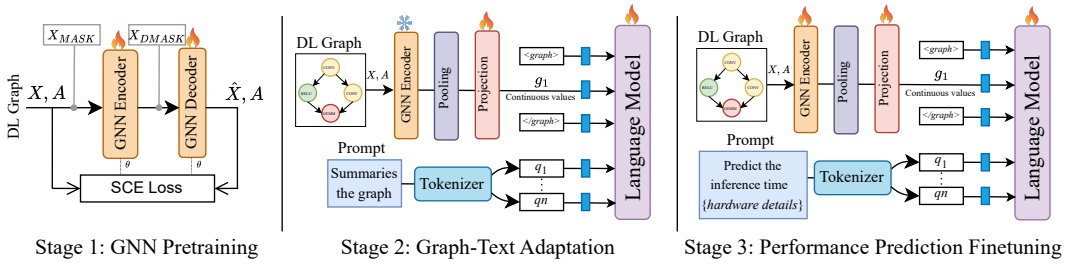

Figure 1: The three stages of our approach: (1) GNN Pre-training: Using Scaled Cosine Error (SCE) loss with masked node features (X_{MASK}) approach to pre-train the GNN. (2) Graph-Text Adaptation: Fine-tuning the pre-trained GNN encoder (frozen) and updating LLM weights and projection weights using soft prompting and LoRA techniques. (3) Performance Prediction Fine-tuning: Updating all GNN projection and LLM parameters through soft prompting and LoRA techniques to predict performance metrics for deep learning graphs on various hardware.

ABSTRACT

Accurate performance prediction of Deep Learning (DL) models is essential for efficient resource allocation and optimizations in various stages of the DL system stack. While existing approaches can achieve high prediction accuracy, they lack ability to quickly adapt to new hardware environments or emerging workloads. This paper leverages both Graph Neural Networks (GNNs) and Large Language Models (LLMs) to enhance the accuracy and adaptability of DL performance prediction. Our intuition is that GNNs are adept at capturing the structural information of DL models, naturally represented as graphs, while LLMs provide generalization and ability to quickly adapt to various tasks thanks to extensive pre-training data. We propose a structured pre-training strategy to enable model adaptation to new hardware environments, significantly reducing the need for extensive retraining. Our experiments validate the effectiveness of this approach, showing a 5.4 percentage-point improvement in accuracy over a state-of-the-art GNN baseline. Notably, when adapted to new hardware with few samples, our method achieves a remarkable 30–70 percentage-point increase in accuracy compared to the GNN baseline.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

KEYWORDS

Performance Model, Graph Neural Network, Large Language Model

ACM Reference Format:
Karthick Panner Selvam[1], Phitchaya Mangpo Phothilimthana[2], Sami Abu-El-Haija[2], Bryan Perozzi[2], Mats Brorsson[1], [1]University of Luxembourg, [2]Google . 2024. LLMs Enhance DL Performance Prediction. In . ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 BACKGROUND

Performance prediction for DL models is essential for all sorts of optimization methods in the DL system stack: from Neural Architecture Search, to model partitioning and sharding, to low-level compiler optimizations[2, 5, 6, 9, 17, 18]. Since DL models are computation graphs, researchers have employed GNNs to extract information from the DL model for various optimization decisions given hardware components [1, 4, 10, 11, 14, 15, 17, 19, 20]. Unfortunately, aforementioned GNN-based approaches require comprehensive retraining to accommodate new hardware environments or DL architectures, often requiring large labeled datasets. Recent research has explored the use of GNNs as encoders to convert graph data into embeddings as inputs to LLMs, thereby effectively bridging the gap between graph data and the textual input preferred by LLMs. However, these studies primarily focus on graph-based question answering, rather than directly on performance prediction [12, 16]. Despite these advancements, prior approaches lack online adaptability. Current methods require retraining for new DL architectures or hardware configurations. On the other hand, our proposed approach aims to overcome this challenge by integrating GNNs with LLMs to create a predictive system that is more adaptable and flexible in real-world scenarios.

2 METHODOLOGY

Stage 3 of Figure 1 displays our proposed model architecture. Our approach takes a DL model graph and a textual prompt as inputs.

The DL graph is initially processed by a GNN encoder and then projected as an embedding to an LLM, along with the token embeddings of the textual prompt. We hypothesize that directly fine-tuning both LLMs and GNNs for performance prediction tasks, starting from scratch, may not yield optimal adaptability for new tasks. The challenge lies in the initial lack of domain-specific knowledge, which is crucial for model to effectively process and predict the DL performance metrics. To address this, we propose a novel structured pre-training methodology, designed to enhance the model's intrinsic understanding of DL graph structures before fine-tuning for performance prediction. The pre-training strategy comprises the three stages as shown in Figure 1.

Stage 1. GNN Pre-training. We employ the Graph Maked Auto Encoder technique (GraphMAE) for GNN pre-training[7]. We use GIN as both encoder and decoder. Given a DL graph with Node feature matrix $\mathbf{X}$ and Adjacency matrix $\mathbf{A}$, we mask a portion of $\mathbf{X}$ using a learnable mask vector to produce $\widetilde{\mathbf{X}}$. The GIN encoder processes $(\widetilde{\mathbf{X}}, A)$ to generate latent embeddings $\mathbf{Z}$, effectively capturing the obscured structural details. The GIN decoder reconstructs the node features from $\mathbf{Z}$ to $\widehat{\mathbf{X}}$, aimed at closely approximating the original $\mathbf{X}$. Reconstruction accuracy is quantified using SCE, which evaluates alignment in both direction and magnitude of the feature vectors. Using GIN for both encoding and decoding optimizes the preservation and reconstruction of local graph structures, essential for understanding DL graphs.

Stage 2. Graph-Text Adaptation. For this stage we update only projection layer and LLM weights. The projection layer $\mathbf{W}_p$ adapts the graph embeddings $\mathbf{g}_1$ for integration with the LLM. During training, we update the projection layer weights using soft prompting techniques[3]. $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_p} = \frac{\partial \mathcal{L}}{\partial \text{Output}} \cdot \frac{\partial \text{Output}}{\partial \mathbf{g}_1} \cdot \mathbf{g}_1^T$ Here, $\frac{\partial \mathcal{L}}{\partial \text{Output}}$ represents the gradient of the loss with respect to the LLM's output, and $\frac{\partial \text{Output}}{\partial \mathbf{g}_1}$ captures how changes in $\mathbf{g}_1$ affect the output. We used cross-entropy loss for the next word prediction. We utilize the LoRA [8] technique to efficiently update the LLM weights. The updates for the low-rank matrices $\mathbf{B}$ and $\mathbf{C}$ are given by: $\frac{\partial \mathcal{L}}{\partial \mathbf{B}} = \frac{\partial \mathcal{L}}{\partial \Delta \mathbf{W}} \cdot \mathbf{C}^T$, $\frac{\partial \mathcal{L}}{\partial \mathbf{C}} = \mathbf{B}^T \cdot \frac{\partial \mathcal{L}}{\partial \Delta \mathbf{W}}$ where $\Delta \mathbf{W} = \mathbf{BC}$ represents the low-rank update to the LLM weights. The GNN encoder weights remain frozen during this stage to preserve the integrity of the initial graph embeddings learned during pre-training.

Stage 3. Performance Prediction Fine-Tuning. In this final stage, we load the pre-trained GIN encoder weights from Stage 1 and the projection $\mathbf{W}_p$ and LoRA weights from Stage 2. We fine-tune the entire GNN to LLM model for performance prediction. Note that naively feeding the GNN embedding outputs as multiple concrete text tokens to the LLM does not work because the gradient does not flow from the LLM to the GNN. This is why we adopt the proposed approach.

3 EXPERIMENT AND RESULTS

All experiments were conducted on a system with 4 x NVIDIA A100 GPU with 40 GB HBM. **Training Datasets.** We utilize three distinct datasets for the different stages of our model's training process. For GNN pre-training, we use a dataset containing 20,000 unlabeled DL graphs [11]. For graph-to-text adaptation, we introduce a novel dataset based on the NNLQP dataset [11]. Each summary provides comprehensive details, including the total number of nodes, edges, model complexity, and statistics for each layer. The dataset comprises 20,000 prompts. For performance prediction fine-tuning, we utilize the NNLQP Multi-platform dataset, which encompasses DL graphs, platform IDs, and inference latency metrics across 10 different hardware platforms. The dataset consists of 7,396 graphs designated for training and 3,201 for testing, totaling 10,597 graphs.

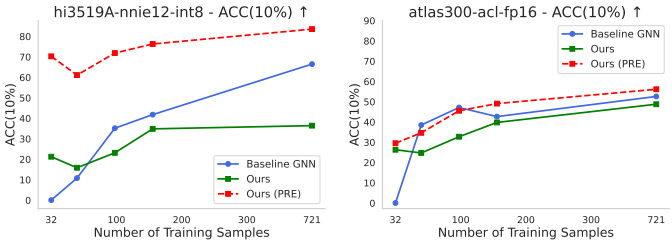

Figure 2: Adaptation Comparison of our approach.

Experiment: Comparison with State-of-the-Art GNN: To rigorously evaluate our proposed architecture, we analyzed the established GNN baseline [11] model against our model with both GNN pre-training and graph-text adaptation using multi-platform performance prediction dataset. The baseline GNN model was utilized with no architectural modifications as described in its original implementation. For our model, we used the LLama3-8B[13] as the base LLM, Adam optimizer with a learning rate of 0.0001 for the LLM and 0.001 for the GNN, across 10 training epochs conducted 3 times to ensure reliability and consistency in the results. Results show, our GNN-LLM model with the pre-training strategy significantly outperformed the GNN baseline: approximately a 3.55% reduction in MAPE and a 10.40% (5.4 percentage-point) increase in Acc(10%). These results highlight the critical impact of the effective model representation and the pre-training strategy on the performance prediction capabilities of LLMs.

Adaptation Experiment: This experiment assesses the real-world adaptability of our model to new hardware environments, particularly under conditions of limited training data. Our comparative analysis involved three models: a GNN baseline and two variants of our model, one with and one without both GNN pre-training and graph-text adaptation. Each model was trained across eight distinct hardware platforms for ten epochs, after which the learned weights were transferred to additional, unseen hardware platforms for further training for three epochs. As results shown in Figure 2, On the hi3519A-nnie12-int8 and atlas300-acl-fp16 platform, our model equipped with the structured pre-training achieves 70% and 29% Acc(10%) respectively, while GNN achieves 0%, when training on just 32 samples. The results also highlight the importance of our structured pre-training strategy, increasing the accuracy of the LLM-GNN model by up to 50 percentage-point. Notice that without the structured pre-training, the LLM-GNN model even underperformed the GNN baseline in some scenarios.

Discussion: This paper has investigated the integration of GNNs and LLMs to enhance the accuracy and adaptability of DL performance prediction. We have proposed a structured pre-training strategy that enables model adaptation to new hardware environments with minimal retraining. We believe that our research offers a promising direction for advancing the field of DL performance prediction.

REFERENCES

[1] Lu Bai, Weixing Ji, Qinyuan Li, Xilai Yao, Wei Xin, and Wanyi Zhu. 2022. DNNAbacus: Toward Accurate Computational Cost Prediction for Deep Neural Networks. arXiv:2205.12095 [cs.LG]

[2] Noureddine Bouhali, Hamza Ouarnoughi, Smail Niar, and Abdessamad Ait El Cadi. 2021. Execution Time Modeling for CNN Inference on Embedded GPUs. In *Proceedings of the 2021 Drone Systems Engineering and Rapid Simulation and Performance Evaluation: Methods and Tools Proceedings* (Budapest, Hungary) *(DroneSE and RAPIDO '21)*. Association for Computing Machinery, New York, NY, USA, 59–65.

[3] Adrian Bulat and Georgios Tzimiropoulos. 2023. LASP: Text-to-Text Optimization for Language-Aware Soft Prompting of Vision Language Models. arXiv:2210.01115 [cs.CV]

[4] Łukasz Dudziak, Thomas Chau, Mohamed S. Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas D. Lane. 2020. BRP-NAS: Prediction-Based NAS Using GCNs. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS'20)*. Curran Associates Inc., Red Hook, NY, USA, Article 879, 11 pages.

[5] Yanjie Gao, Yu Liu, Hongyu Zhang, Zhengxian Li, Yonghao Zhu, Haoxiang Lin, and Mao Yang. 2020. Estimating GPU memory consumption of deep learning models. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Virtual Event, USA) *(ESEC/FSE 2020)*. Association for Computing Machinery, New York, NY, USA, 1342–1352. https://doi.org/10.1145/3368089.3417050

[6] Eugenio Gianniti, Li Zhang, and Danilo Ardagna. 2018. Performance Prediction of GPU-Based Deep Learning Applications. In *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. 167–170. https://doi.org/10.1109/CAHPC.2018.8645908

[7] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. GraphMAE: Self-Supervised Masked Graph Autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 594–604.

[8] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs.CL]

[9] Daniel Justus, John Brennan, Stephen Bonner, and Andrew Stephen McGough. 2018. Predicting the Computational Cost of Deep Learning Models. In *2018 IEEE International Conference on Big Data (Big Data)*. 3873–3882.

[10] Sam Kaufman, Phitchaya Phothilimthana, Yanqi Zhou, Charith Mendis, Sudip Roy, Amit Sabne, and Mike Burrows. 2021. A Learned Performance Model for Tensor Processing Units. In *Proceedings of Machine Learning and Systems*, A. Smola, A. Dimakis, and I. Stoica (Eds.), Vol. 3. 387–400.

[11] Liang Liu, Mingzhu Shen, Ruihao Gong, Fengwei Yu, and Hailong Yang. 2022. NNLQP: A Multi-Platform Neural Network Latency Query and Prediction System with An Evolving Database. In *51 International Conference on Parallel Processing - ICPP (ICPP '22)*. Association for Computing Machinery, Article 43, 14 pages.

[12] Zheyuan Liu, Xiaoxin He, Yijun Tian, and Nitesh V. Chawla. 2024. Can we Soft Prompt LLMs for Graph Learning Tasks?. In *Companion Proceedings of the ACM on Web Conference 2024 (WWW '24)*. ACM. https://doi.org/10.1145/3589335.3651476

[13] meta llama. [n. d.]. GitHub - meta-llama/llama3: The official Meta Llama 3 GitHub site. https://github.com/meta-llama/llama3

[14] Karthick Panner Selvam and Mats Brorsson. [n. d.]. Can Semi-Supervised Learning Improve Prediction of Deep Learning Model Resource Consumption?. In *Machine Learning for Systems Workshop at 37th NeurIPS Conference, 2023, New Orleans, LA, USA*. https://openreview.net/forum?id=C4nDgK47OJ

[15] Karthick Panner Selvam and Mats Brorsson. 2023. DIPPM: A Deep Learning Inference Performance Predictive Model Using Graph Neural Networks. In *Euro-Par 2023: Parallel Processing*. Springer Nature Switzerland, 3–16.

[16] Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. 2024. Let Your Graph Do the Talking: Encoding Structured Data for LLMs. arXiv:2402.05862 [cs.LG]

[17] Phitchaya Mangpo Phothilimthana, Sami Abu-El-Haija, Kaidi Cao, Bahare Fatemi, Mike Burrows, Charith Mendis, and Bryan Perozzi. 2023. TpuGraphs: A Performance Prediction Dataset on Large Tensor Computational Graphs. arXiv:2308.13490 [cs.LG]

[18] Hang Qi, Evan R. Sparks, and Ameet Talwalkar. 2017. Paleo: A Performance Model for Deep Neural Networks. In *International Conference on Learning Representations*.

[19] Yun Yi, Haokui Zhang, Rong Xiao, Nannan Wang, and Xiaoyu Wang. 2023. NAR-Former V2: Rethinking Transformer for Universal Neural Network Representation Learning. arXiv:2306.10792 [cs.LG]

[20] Yanqi Zhou, Sudip Roy, Amirali Abdolrashidi, Daniel Wong, Peter Ma, Qiumin Xu, Hanxiao Liu, Mangpo Phitchaya Phothilimtha, Shen Wang, Anna Goldie, Azalia Mirhoseini, and James Laudon. 2020. Transferable graph optimizers for ML compilers. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 1161, 12 pages.