Graph-Informed Transformers for Neural Network Inference Latency Prediction

Asad Khan
University of Toronto
asadk.khan@mail.utoronto.ca

***Abstract*—Deep learning applications such as real-time object detection in autonomous vehicles, interactive voice assistants, and high-frequency trading systems often require strict adherence to inference latency constraints defined by service-level objectives. Ensuring that neural network inference times meet these constraints before deployment presents a significant challenge to developers. In this paper, we introduce a transformer-based approach to predict neural network inference latency in pre-deployment stages. Our method utilizes a diverse synthetic dataset of feedforward neural networks, characterized at the operation level. These networks are represented as graphs, where the node attributes encode the type of operation and weight count, and the edges define the topology of the network. By treating each node as a token, the transformer leverages multi-head attention to capture structural and attribute relationships that strongly correlate with inference latency. Experimental results demonstrate that the proposed transformer model achieves much better performance when compared to baseline linear regression in predicting standard neural network inference latency across a wide variety of architectures and configurations. Ultimately, our transformer-based solution facilitates the development of latency-sensitive deep learning systems by enabling more reliable and efficient architectural optimization prior to deployment.**

I. Introduction

Inference latency refers to the time delay between providing an input to a model and receiving the corresponding output. In deep learning, this metric is crucial for applications that require real-time or near-real-time responses [1]. Thus, deploying deep learning models in production environments requires careful consideration of inference latency constraints. Modern deep learning models often comprise millions or even billions of parameters [2], leading to challenges in inference latency and resource utilization. The complexity of these models poses significant interpretive challenges and can impact the efficiency of deployment [3].

A. Motivation

Previous work by Mendoza & Wang demonstrated the effectiveness of using graph embeddings and deep neural networks for latency prediction [4]. However, there remains room for improvement, particularly with the application of Transformer models.

The hierarchical and sequential nature of neural network architectures means that data is processed through multiple layers, each extracting increasingly abstract features from the input. This layered structure allows neural networks to learn complex representations, with each layer building upon the outputs of the preceding ones [5]. Given this structure, Transformer models, known for their attention mechanisms, are particularly well suited for tasks involving hierarchical and sequential data [6]. Transformers can capture long-range dependencies and intricate relationships within data sequences [6], making them effective in modeling complex interactions between different layers and operations in neural networks. This capability suggests that Transformers could be advantageous in predicting neural network inference latency by effectively understanding and representing the hierarchical architecture of neural networks.

B. Related Works

1) Inference Latency Prediction: The prediction of neural network inference latency is an important area of exploration as deep learning models become increasingly complex, with various methodologies proposed to enhance accuracy and efficiency. Mendoza and Wang's work introduced the use of graph representations, utilizing attribute node lists and adjacency matrices at the operator level to predict individual inference latency [4].

Building upon this foundation, Liu et al. developed NNLQP, a multi-platform neural network latency query and prediction system [7]. NNLQP integrates an evolving database to store latency data across diverse hardware platforms, facilitating efficient retrieval and prediction. By leveraging a unified graph embedding, NNLQP addresses challenges associated with hardware graph fusion and kernel launch costs, thereby improving prediction accuracy.

In the realm of hardware-aware neural architecture search, Beglaryan and Ringhofer proposed a deep learning estimator model designed to predict the inference latency of fully connected deep neural networks on laptop CPUs [1]. Their work underscores the importance of tailored latency prediction models for specific hardware configurations.

Collectively, these studies underscore the importance of accurate latency prediction in optimizing neural network performance across various deployment environments. However, there remains an opportunity for further work by incorporating advanced architectures, such as transformers, to capture the intricate relationships within neural network structures more effectively.

2) Transformers for Performance Prediction: Transformer models have been used for structured prediction tasks across various domains [8], largely due to their ability to model complex dependencies within sequences via self-attention mechanisms [6]. These mechanisms allow Transformers to capture intricate, multi-layered relationships in data, making them suitable for modeling latency in neural network architectures.

Recent work by Käppel et al. highlights the interpretability of attention scores, which shed light on how elements within a sequence relate to one another through self-attention [9]. This interpretability is particularly valuable when applied to latency prediction, as it provides insights into dependencies within neural network layers that traditional deep learning models might miss.

C. Problem Definition

Our work implements a novel Transformer-based architecture for predicting neural network inference latency that takes graph representations of a neural network as input and outputs a continuous value representing the predicted inference latency. By treating each layer or operation as a token, our model leverages multi-head attention to capture the structural and attribute relationships that correlate with inference latency. This approach provides a practical tool for developers to optimize neural network deployment without extensive testing, facilitating the development of latency-sensitive deep learning systems.

II. Methodology

In this section, we layout the data and model architecture used for training the transformer model.

A. Data

The dataset used to train the transformer model consists of 270,000 synthetically generated feedforward neural networks (FNNs), represented as graphs. In this representation, nodes correspond to operations within the network, such as input layers, hidden layers, and output layers, while edges signify the connections between these operations. Each node is annotated with attributes that detail the type of operation and the number of associated weights, encapsulating both the structural and computational characteristics of the network. This graph-based representation captures the intricacies of the architecture, providing a rich foundation for predictive modeling.

The dataset is generated through a systematic process to ensure diversity and complexity. Neural network architectures are sampled with depths ranging from three to ten layers. The number of units in each hidden layer is randomly assigned from a range of powers of two, spanning from 1 to 16,384 units. This approach allows for the creation of networks with varying computational demands, capturing both shallow and deep architectures. Once the architecture is defined, it is represented as a graph with an adjacency matrix to map the topological structure and a node attribute matrix to encode the operation type and weight count for each layer.

To capture the computational characteristics of the networks, inference latency is measured for each architecture. Each network is instantiated and profiled on a standardized hardware platform (M2 Macbook Pro w/ 16GB RAM), where its latency is determined through a series of forward passes with randomly generated input data. The median latency across 5 runs is recorded to ensure consistency and robustness against measurement variability. This process results in a dataset containing graph representations of architectures paired with their corresponding latency values.

B. Model Architecture

The transformer model for inference latency prediction processes neural network architectures represented as graphs. In these graphs, nodes correspond to layers in the network, annotated with attributes that include the operation type and the number of weights. Edges capture the connections between these layers. The model predicts a single scalar value representing the network's inference latency based on this graph representation.

The input to the model consists of two tensors. The first tensor encodes the node attributes and has a shape of $(\text{batch size}, \text{maximum depth of the network}, \text{attribute dimension})$. The maximum depth of the network is set to 10, representing the largest number of layers expected in the dataset. This value ensures the model can handle most architectures without exceeding memory constraints. The attribute dimension is set to 3, which corresponds to the three features encoded for each node: the type of operation (e.g., input, dense), the number of weights, and any additional characteristics relevant to inference. The second tensor is a binary mask with a shape of $(\text{batch size}, \text{maximum depth of the network})$, which indicates valid nodes with 'True' and padded nodes with 'False'. The mask ensures that padded nodes do not contribute to the predictions.

The model begins by projecting the node attributes into a higher-dimensional embedding space using a fully connected dense layer. This transforms the input tensor to a shape of $(\text{batch size}, \text{maximum depth of the network}, 64)$, where 64 is the embedding dimension. This value balances the need for rich feature representation and computational efficiency. Learnable positional embeddings are then added to the node embeddings to encode the sequential nature of the layers and distinguish nodes based on their positions within the network.

The core of the model is composed of two stacked Transformer encoder blocks. Each block uses a multi-head attention mechanism with four attention heads, which capture intricate dependencies between layers. The feedforward network within each block has a hidden size of 128, chosen to provide sufficient capacity for processing complex relationships while avoiding overfitting. Dropout is applied with a rate of 0.1 after the attention and feedforward layers to prevent overfitting, and layer normalization is included before and after each component to stabilize training.

Once the node embeddings have been processed by the Transformer blocks, the mask is applied to zero out contributions from padded nodes. The embeddings of valid nodes are pooled using a weighted mean, where the embeddings are first

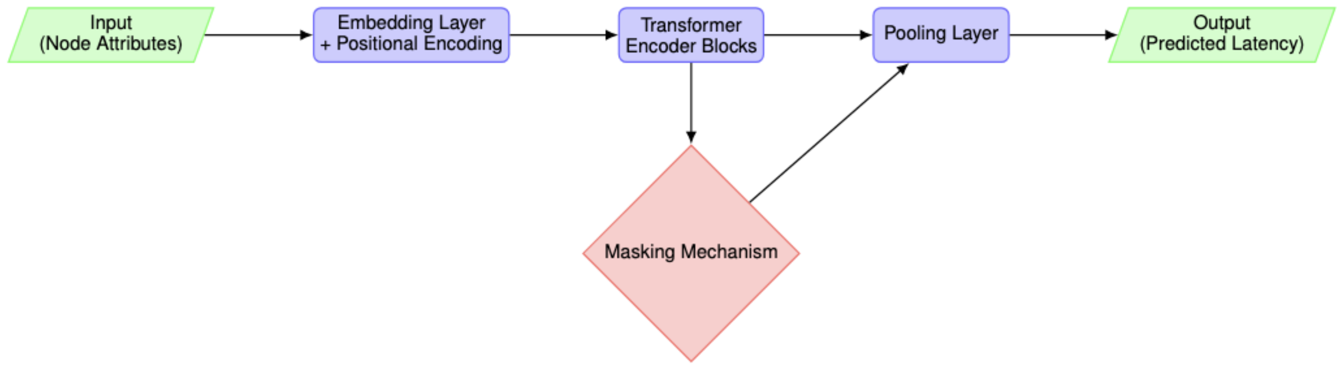

Fig. 1. Transformer Model Architecture for Inference Latency Prediction

multiplied by the binary mask, summed across the sequence dimension, and then divided by the number of valid nodes. This pooling mechanism aggregates information from all valid nodes while ensuring that padded regions do not influence the output.

The final dense layer produces a single scalar value for each graph, representing the predicted latency. The model is trained using mean squared error (MSE) as the loss function, which is suitable for regression tasks, and mean absolute error (MAE) is used as an additional evaluation metric to provide an intuitive measure of prediction accuracy.

The training procedure uses a batch size of 64 to strike a balance between computational efficiency and gradient stability. Training is conducted over 100 epochs, allowing the model sufficient time to converge without overfitting. The dataset is split into 80% for training, 10% for validation, and 10% for testing. The Adam optimizer is employed with a learning rate of 0.001 to ensure efficient convergence.

The architecture of the transformer model is well-aligned with the task of latency prediction. The use of multi-head attention allows the model to capture both local and global dependencies within the network architecture. Positional embeddings enable the model to encode the sequential nature of the layers, while the pooling mechanism effectively handles variable-length graphs by focusing on valid nodes. These design choices, combined with the chosen hyperparameters, make the model highly effective for predicting inference latency across a diverse range of neural network architectures. The values for the hyperparameters were chosen to balance representational capacity, computational efficiency, and the scale of the dataset. This alignment ensures the model is not only effective but also practical for real-world use cases.

III. Results

The transformer model achieved a **test mean absolute error (MAE) of 0.0038 seconds**. This represents a substantial improvement over the baseline linear regression model introduced in the work of Mendoza and Wang, which achieved a test set MAE of **0.01619 seconds** [4]. By comparison, the transformer model reduces the average prediction error by nearly **76.5%**.

The mean absolute error (MAE) was selected as the primary evaluation metric for this task due to its interpretability and direct relevance to the problem of latency prediction. MAE measures the average absolute difference between predicted and actual latency values, providing an intuitive sense of prediction accuracy. The MAE of **0.0038** implies that, on average, the model's latency predictions deviate by just **3.8 milliseconds** from the true values. This precision is particularly significant given that the mean latency in the test set is **0.0400 seconds**, meaning the transformer model achieves an error of less than **10%** of the average latency.

The baseline linear regression model used in Mendoza and Wang's work is a strong baseline due to its simplicity, efficiency, and ability to capture basic patterns in the data. Its interpretability and low computational cost make it ideal for benchmarking more complex models. The transformer model's significant improvement over this baseline highlights its ability to capture the nuanced dependencies within neural network architectures, validating its advanced design.

TABLE I
Latency Prediction Mean Absolute Error (MAE)

Model	MAE	Mean Latency (Test Set)
Transformer	0.0038	0.0400
Linear Regression	0.0162	0.0214

The transformer model's performance in predicting inference latency represents a significant improvement in neural network inference latency prediction. By achieving a test set mean absolute error (MAE) of **0.0038 seconds**, this improvement demonstrates the transformative potential of transformer-based architectures for capturing the intricate relationships inherent in neural network graphs.

The MAE of **0.0038** is particularly impressive when contextualized against the mean latency of **0.0400 seconds**. This indicates that the transformer model produces predictions that are both accurate and actionable. For latency-critical applications, where small errors in prediction can have significant real-world consequences, such precision is invaluable.

The baseline linear regression model, while effective in capturing simple relationships between graph structure and latency, is inherently limited in its ability to model non-linear and hierarchical dependencies. This limitation is evident in its significantly higher MAE of **0.01619**, which corresponds to an average error of approximately **40% of the mean latency**. By contrast, the transformer model's ability to process graph-structured data with self-attention mechanisms allows it to uncover patterns and dependencies that the linear regression model cannot.

IV. Conclusion

In this paper, we presented a novel transformer-based approach for predicting neural network inference latency during the pre-deployment stages. Using a synthetic data set of graph-represented feedforward neural networks, our method captures both structural and computational characteristics of neural architectures. The use of multi-head sequential attention in the transformer model enables it to effectively model intricate relationships between layers, significantly improving latency prediction accuracy compared to baseline methods such as linear regression. This demonstrates the capability of our approach to generalize across diverse architectures and configurations.

While the transformer model demonstrates excellent performance in predicting inference latency, certain limitations highlight areas for further investigation and potential improvement.

The dataset used for training and evaluation consists of synthetically generated feedforward neural networks. While this approach ensures diversity in the architectures, it may not fully capture the complexities and variations present in real-world deployments. Factors such as hardware-specific optimizations, memory hierarchies, and non-ideal runtime behaviors are not reflected in the dataset. As a result, the model's performance on real-world neural networks deployed in different environments may differ from the results reported here.

To address this, there are avenues for improving the transformer model's capabilities. Incorporating more diverse datasets, including real-world neural network architectures and multi-platform latency measurements, would enhance the model's robustness. It would also be beneficial to integrate explicit hardware features into the input representation which could improve cross-platform performance.

Despite these limitations, the transformer model demonstrates significant promise for inference latency prediction. Addressing the outlined challenges and exploring areas for improvement would further enhance its utility and applicability in real-world scenarios. These efforts would make the model an even more effective tool for optimizing neural network deployment in latency-sensitive applications.

By facilitating more reliable and efficient optimization of neural network architectures prior to deployment, our approach empowers developers to design systems that adhere to strict inference latency constraints. This work contributes to advancing pre-deployment tools for deep learning systems, with potential extensions to incorporate hardware-specific adaptations or to explore hybrid models for further enhancing prediction accuracy.

REFERENCES

[1] L. Beglaryan and C. Ringhofer, “Development and training of a deep learning approach to estimate latency of deep neural network inference,” *American University of Armenia*, 2023.

[2] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, “Zero: Memory optimizations toward training trillion parameter models,” *arXiv preprint arXiv:1910.02054*, 2020. [Online]. Available: https://arxiv.org/abs/1910.02054

[3] S. Sinha and Y. M. Lee, “Challenges with developing and deploying ai models and applications in industrial systems,” *Discover*, 2024.

[4] D. M. Mendoza and S. Wang, “Predicting latency of neural network inference,” Stanford University, Tech. Rep., 2020. [Online]. Available: http://cs230.stanford.edu/projects_fall_2020/reports/55793069.pdf

[5] M. M. Taye, “Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions,” *Computation*, vol. 11, no. 3, p. 52, 2023.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5998–6008.

[7] L. Liu, M. Shen, R. Gong, F. Yu, and H. Yang, “Nnlqp: A multi-platform neural network latency query and prediction system with an evolving database,” in *Proceedings of the 51st International Conference on Parallel Processing*, ser. ICPP ’22, 2022.

[8] B. Wang, L. He, L. Song, R. Niu, and M. Cheng, “Attention-linear trajectory prediction,” *Sensors*, vol. 24, no. 20, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/20/6636

[9] M. Käppel, L. Ackermann, S. Jablonski, and S. Härtl, “Attention please: What transformer models really learn for process prediction,” in *Business Process Management - 22nd International Conference, BPM 2024, Krakow, Poland, September 11-15, 2024, Proceedings*, 2024, pp. 203–220. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-70396-6_12