Analyzing the Effectiveness of 1D convolutional layer as a Replacement for FFN in Transformer

Youngjin Na

yj.na08@gmail.com

Abstract

This experiment investigates the performance comparison between the standard Dense layer and a 1D convolutional layer with a kernel size of 1 (Conv1D) in the feed-forward network (FFN) of the Transformer. In the original Transformer study [6], the authors mentioned that a pointwise network could be replaced by using a convolutional layer with kernel size=1 instead of a Dense layer. While both architectural components perform similar transformations, I experimentally analyze their differences in computation time in neural machine translation tasks. The result represents that, under certain conditions, Conv1D achieves around 14% faster computation compared to the Dense layer. This efficiency gain may be attributed to GPU optimization differences due to distinct computational methods. These finding could offer basic insight for understanding the roll of two component layers in Transformer-based models.

Introduction

The Transformer architecture [6] has revolutionized deep learning, serving as the foundation for major advancements in natural language processing (NLP), computer vision, speech recognition, and multimodal models. The Transformer leverages self-attention mechanisms and FFNs to model complex sequential dependencies, offering significant advantages over traditional recurrent architectures. This design has led to remarkable progress across a wide range of tasks and has inspired numerous derivative models, including BERT [2], GPT [3], and Transformer-XL [1], further pushing the boundaries of artificial intelligence.

One of fundamental component of the Transformer is the FFN, which enhances the model's capacity to capture intricate patterns through non-linear transformations. While dense layers are conventionally employed in FFNs, 1D convolutions with a kernel size of 1 (Conv1D) have been proposed as a potential alternative to dense layers.

This experiment examines that the difference of replacing the dense layers in the Transformer's FFN with Conv1D to enhance computational efficiency while maintaining model performance. Specifically, I searched the difference between convolutional and dense layer operations, analyzing their impact on computational speed.

Background

Natural language processing (NLP) has witnessed significant advancements over the past few years. Early NLP models, such as recurrent neural networks (RNNs) and long short-term memory networks (LSTMs), demonstrated strong capabilities in processing sequential data. However, these models inherently suffer from limitations in capturing long-range dependencies due to their sequential nature and the vanishing gradient problem.

The introduction of the Transformer architecture marked a paradigm shift in sequence modeling. Unlike traditional recurrent architectures, Transformers eliminate the need for sequential processing by leveraging self-attention mechanisms, enabling all input tokens to be processed in parallel. This fundamental change not only enhances the model's ability to capture long-range dependencies but also significantly reduces training time. As a result, Transformer-based models have become the cornerstone of modern NLP and have been widely adopted in various domains, including computer vision, speech recognition, and multi-modal learning.

Transformer Architecture

Transformer enables parallel computation, significantly improving processing efficiency for large-scale datasets. Furthermore, by computing relationships between all tokens simultaneously through self-attention mechanisms, the Transformer captures where the model have to focus more effectively than previous architectures.

The Transformer follows an encoder-decoder structure, composed of the following key components:

- **Self-Attention:** Computes contextual relationships between all tokens in the input sequence.
- **Multi-Head Attention:** Executes multiple attention mechanisms in parallel to extract diverse linguistic features.
- **Feed-Forward Network (FFN):** Independently transforms the representation of each position.
- **Residual Connections and Layer Normalization:** Improve stability and mitigate gradient vanishing issues.

Pointwise Feedforward Network (FFN)

Within each encoder and decoder block of the Transformer, the FFN plays a crucial role in applying non-linear transformations to each token independently. The standard FFN of Transformer in the paper [6] consists of two fully connected (Dense) layers with an intermediate activation function. This structure expands the feature space before reducing it back to its original dimension, thereby enhancing the model's representational capacity.

Method

This experiment examined the impact of replacing the standard Dense layer in the Transformer's Conv1D with a kernel size of 1. As illustrated in Figure1, both architectures are implemented within the encoder FFN, the decoder follows an identical structure as well.

Previous research on efficient deep learning architectures has analyzed the computational properties of different layer types in neural networks, particularly in terms of efficiency. [5]. It is mentioned that the convolutional layer is related to the Dense layer, and the fundamental component of both is the multiply-and-accumulate operation.

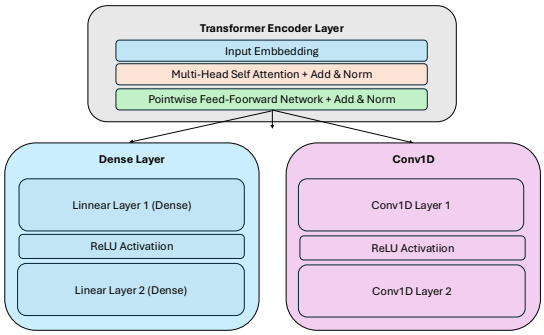

Figure 1: Comparison of the FFN in the Transformer model using a Dense layer and Conv1D. The figure illustrates how each layer processes input data and performs inter-channel transformations in Encoder Block.

Dense Layer

The Dense layer, or fully connected layer, applies a linear transformation to each input independently, followed by a non-linear activation function. This operation is mathematically expressed as:

$$\mathbf{y} = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$$

where $\mathbf{W} \in R^{m \times n}$ is the weight matrix, $\mathbf{x} \in R^n$ is the input vector, and $\mathbf{b} \in R^m$ is the bias term where n is the input dimension and m is the output dimension. The Dense layer requires a full matrix multiplication.

Conv1D (Kernel Size = 1)

A 1D convolution with a kernel size of 1 (Conv1D) performs a linear transformation but with shared weights across spatial positions. This operation is defined as:

$$\mathbf{y}_i = W \cdot x_i + b$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ represents the input sequence. It can perform the same operations as Dense layer, except that they apply the same weights to each location.

Experiments and Results

In this experiment, I conducted experiments on the Korean-English translation task using the Korean-English corpus [4]. The dataset was preprocessed by removing irrelevant characters and normalizing the text. Sentence pairs were tokenized with SentencePiece, resulting in a vocabulary size of 20,000 tokens for each language. Sentences exceeding 50 tokens in length were excluded, and the remaining sequences were padded to ensure uniform input length.s

Hyper Parameter Settings

I evaluated two variations of Transformer-based models, differing in their choice of FFN implementation: one utilizing Conv1D-based FFN and the other employing the standard Dense-based FFN. Both models consist of two layers with a model dimension of 512 and 8 attention heads. The FFN dimension is set to 512, and positional encoding is applied with a maximum sequence length of 50 tokens. To mitigate overfitting, a dropout rate of 0.2 is applied.
For optimization, I utilize the Adam optimizer with a custom learning rate scheduler. The optimizer parameters are set as follows: $\beta_1 = 0.9$ for the first moment estimate, $\beta_2 = 0.98$ for the second moment estimate, and $\epsilon = 10^{-9}$.

Experimental Results

Table1 presents a comparison of the average execution times per step for the Dense and Conv1D layers in the each FFN block of the Transformer model. The Dense layer demonstrates a faster execution time of 0.01006 seconds, while the Conv1D layer takes 0.01173 seconds. As shown in Table2, this difference becomes more significant when considering the total training time. Despite both models having the same number of parameters (39,155,232), the Conv1D-based model requires slightly more training time, with a total of 192.47 seconds for 1 epoch, compared to the Dense layer's 182.61 seconds.

Type of FFN	Average Execution Time
Dense layer	0.01006 (± 0.00001) sec
Conv1D layer	0.01173 (± 0.00051) sec

Table 1: Average execution time per step for Dense layer and Conv1D layer in each Transformer's FFN block . The Dense layer has an average execution time of 0.01006 seconds, while the Conv1D layer has an average execution time of 0.01173 seconds

Conclusions

In this experiment, I compared the use of Dense and Conv1D layers in the Transformer's FFN in terms of training time.

Type of FFN	Total Parameters	Train time
Dense layer	39,155,232	182.61 sec
Conv1D layer	39,155,232	192.47 sec

Table 2: Comparison of total parameters and training time between Dense layer and Conv1D layer in the Transformer with an epoch training. The model having Dense layer and Conv1D layer have identical number of parameters with (39,155,232), but Dense layer requiring 182.61 seconds and the Conv1D layer taking 192.47 seconds for training in an epoch

The results indicate that Conv1D is slightly slower than the Dense model. According to a paper [5], the Dense layer is mapped to matrix multiplication, which fully leverages the GPU's SIMD architecture to perform highly parallelized computations. Through optimized matrix multiplication, Dense layers can take full advantage of the hardware acceleration provided by GPUs. The paper further explains that temporal architectures use Arithmetic Logic Unit (ALU) under centralized control to perform continuous-memory accesses, a method that is highly beneficial for matrix multiplication. In contrast, although Conv1D performs the same MAC operations, its independent computations at each position make it difficult to achieve the same level of optimized computational efficiency with ALU as the Dense layer.

Despite the observed performance differences, this study has several limitations. These discrepancies may be influenced by factors specific to our implementation environment. First, the evaluation was limited to a specific Transformer architecture and dataset, which may restrict the generalizability of the findings. Additionally, the effect of sequence length on performance was not thoroughly examined. Since longer sequences increase memory and computational demands, varying sequence lengths could impact the relative efficiency of these layers. Moreover, this study focused on training time without assessing translation quality using quantitative metrics such as BLEU or accuracy.

Future experiments should involve a more comprehensive analysis across various sequence lengths and batch sizes to better understand the conditions under which Conv1D may outperform Dense layers, or vice versa. Additionally, evaluating both computational efficiency and translation quality will offer deeper insights into the trade-offs between Dense and Conv1D layers in the Transformer FFN.

References

[1] Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q. V.; and Salakhutdinov, R. 2019. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context.

[2] Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

[3] OpenAI; Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; et al. 2023. GPT-4 Technical Report.

[4] Park, J.; Hong, J.-P.; and Cha, J.-W. 2017. Korean-English Parallel Evaluation Dataset (JHE). https://github.com/jungyeul/korean-parallel-corpora.

[5] Sze, V.; Chen, Y.-H.; Yang, T.-J.; and Emer, J. 2017. Efficient Processing of Deep Neural Networks: A Tutorial and Survey.

[6] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is All You Need.