The Transformer Report

Yuqing Qiao
Khoury College of Computer Sciences
Northeastern University
qiao.yuqi@northeastern.edu
June 6th, 2024

1 Introduction

This report explores the application of the Transformer model, a powerful architecture that has significantly influenced the field of natural language processing, specifically in machine translation tasks. The objective of this project is to demonstrate how the Transformer model can improve translation accuracy and efficiency by leveraging self-attention mechanisms. The code is based on the PyTorch implementation from https://nlp.seas.harvard.edu/annotated-transformer/

2 Method

The Transformer model operates on the principle of self-attention, eschewing traditional recurrent layers to process data in parallel, thereby speeding up training without compromising the understanding of sequence order in languages. The model is trained using a standard cross-entropy loss and optimized via the Adam optimizer with a custom learning rate scheduler, which increases the learning rate linearly for the first few thousand training steps before decaying.

3 Dataset and Preprocessing

I utilize the Multi30k dataset, a collection designed for machine translation tasks that contains pairs of English-German sentences, which is ideal for evaluating the performance across similar language structures. The dataset is loaded and preprocessed using PyTorch's `DataLoader` and the `torchtext` library.

4 Model Architecture

The Transformer model used in this project consists of an encoder and decoder architecture with the following configurations:

Model Dimension (d_model): 512	**Batch Size**: 128
Layers: 6 encoder and 6 decoder layers	**grad_clip**: 1
Heads: 8 attention heads per multi-head attention layer	**Adam Optimizer learning rate**: 1e4
Feed Forward Dimension (d_ff): 2048	**Dropout rate**: 0.1

At its core, the model is composed of layers that include multi-head attention, feed-forward networks, and add & norm operations.Multi-head attention allows the model to focus on different parts of the input simultaneously by splitting the input into multiple heads, performing scaled dot-product attention for each head, and then combining the results. The MultiHeadAttention projects the queries, keys, and values into multiple heads, applying attention, and recombining the results. The add & norm operations, which include residual connections followed by layer normalization, help stabilize and accelerate the training process.

Multi-Head Attention

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,\dots,\text{head}_h)W^O$$
$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$
$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

5 Translation

5.1 Greedy Decoding

Greedy decoding was initially implemented, where the model selects the most probable word at each step in the sequence. This method is computationally efficient but often results in suboptimal translation sequences.

5.2 Beam Search

To enhance translation quality, beam search decoding keeps track of multiple translation hypotheses at each step, providing a more globally optimized solution by selecting the hypothesis with the highest overall probability.

6 Challenges

I encountered significant challenges related to matrix transformations, particularly with operations like squeeze and transpose, which are essential for performing matrix reshapes. Debugging the model to ensure each matrix multiplication had the correct shape was difficult and time-consuming. Despite these challenges, this process allowed me to gain a deeper understanding of the attention layer's implementation, the intricacies of cross-attention between the encoder and decoder, and the functionality of the PyTorch framework.

7 Conclusion

The Transformer model demonstrated its translation capabilities on the Multi30k dataset by leveraging the multi-head attention mechanism. Additionally, two types of inference mechanisms, greedy decoding and beam search, were explored, enhancing my understanding.