Attention Is All You Need

1 Introduction

Recurrent neural networks (RNNs) and long short-term memory (LSTM) are used in sequence modeling, but have limitations in terms of parallelization. Attention mechanisms have been introduced to capture dependencies without distance constraints. The Transformer model eliminates recurrence and relies on attention mechanisms for global dependencies. It offers significant parallelization capabilities and achieves state-of-the-art translation quality with shorter training time.

2 Background

The goal of reducing sequential computation is the foundation for models like Extended Neural GPU, ByteNet, and ConvS2S. These models use convolutional neural networks to compute hidden representations in parallel. The Transformer reduces the number of operations required to relate signals from distant positions but faces a decrease in effective resolution, which is countered by Multi-Head Attention. Self-attention is an attention mechanism that relates positions in a sequence, successfully used in tasks like reading comprehension and summarization. The Transformer is the first model to rely entirely on self-attention without using RNNs or convolution, outperforming end-to-end memory networks. The advantages of self-attention over other models will be discussed further.

3 Model Architecture

Competitive neural sequence transduction models have an encoder-decoder structure, where the encoder maps input symbols to continuous representations and the decoder generates output symbols based on these representations. The model is auto-regressive, using previously generated symbols as input. The Transformer architecture follows this structure and utilizes stacked self-attention and fully connected layers for both the encoder and decoder. See Figure 1 for a visual representation of the encoder and decoder components.

3.1 Encoder and Decoder Stacks

The encoder stack consists of 6 layers with multi-head self-attention and fully connected networks. The decoder stack also has 6 layers with an additional sub-layer for multi-head attention over the encoder output. Both encoder and decoder use residual connections and layer normalization. In the decoder, self-attention is modified to prevent attending to subsequent positions. Output embeddings are masked and offset to ensure predictions depend on known outputs.

3.2 Attention

Attention is introduced as a concept in the text. An attention function is defined as a mapping between a query and a set of key-value pairs, with all vectors. The output is computed as a weighted sum of the values, where the weight is determined by a compatibility function of the query with the corresponding key.

3.2.1 Scaled Dot-Product Attention

In the field of attention mechanisms, the Scaled Dot-Product Attention is introduced, where inputs are queries, keys, and values of given dimensions. The attention computation involves the dot products of queries and keys, followed by scaling and softmax to obtain weights. This computation can be done simultaneously on a matrix of queries, keys, and values. Additive attention is an alternative method, but dot-product attention with scaling is generally more effective, especially for larger dimensions. Scaling is necessary to prevent issues with extremely small gradients caused by large dot products.

3.2.2 Multi-Head Attention

In multi-head attention, instead of using a single attention function with dmodel-dimensional keys, values, and queries, it is more beneficial to linearly project them h times with different learned linear projections. The projected versions are then used to perform the attention function in parallel, allowing the model to attend to different information subspaces at different positions. The dot product of the projected keys and queries has a mean of 0 and a variance of dk. The MultiHead function is represented as a concatenation of individual attention heads, each computed using different parameter matrices. In this case, there are 8 parallel attention layers with dk = dv = dmodel/h = 64. Despite the reduction in dimension for each head, the computational cost is similar to single-head attention with full dimensionality.

3.2.3 Applications of Attention in our Model

The text discusses the applications of attention in the model, specifically the encoder-decoder attention layers, encoder self-attention layers, decoder self-attention layers, and preventing leftward information flow in the decoder. These attention mechanisms allow for positions in the model to attend to specific positions in the input or previous layers to improve sequence-to-sequence models.

3.3 Position-wise Feed-Forward Networks

In transformer models, each layer contains a position-wise feed-forward network in addition to attention sub-layers. The feed-forward network applies the same operations to each position separately and identically. The feed-forward network consists of two linear transformations with a ReLU activation function in between. The output of the first linear

transformation is passed through the ReLU activation function, and then the second linear transformation is applied to it.

The parameters of the feed-forward network vary from layer to layer, even though the operations performed are the same for all positions. One way to interpret the feed-forward network is as two convolutions with a kernel size of 1. The input and output of the feed-forward network have a dimensionality of 512, while the inner-layer of the network has a dimensionality of 2048.

3.4 Embeddings and Softmax

In sequence transduction models, learned embeddings are used to convert input and output tokens to vectors. A shared weight matrix is used in the embedding layers and pre-softmax linear transformation. This approach is similar to a study mentioned as "(<>)24".

3.5 Positional Encoding

In order to encode the order of a sequence in a model without recurrence or convolution, positional encodings are added to the input embeddings. A table provides information on the complexity and maximum path lengths for different layer types. Positional encodings have the same dimension as the embeddings and can be either learned or fixed. The chosen approach involves using sine and cosine functions to create positional encodings. This allows the model to easily learn to attend by relative positions. Learned positional embeddings were also tested, but the sinusoidal version was chosen for its potential to handle longer sequence lengths.

4 Why Self-Attention

The introduction discusses the comparison between self-attention layers, recurrent layers, and convolutional layers used for mapping sequences. Three desiderata motivate the use of self-attention: computational complexity, parallelizability, and path length between long-range dependencies. Self-attention layers have faster computational complexity than recurrent layers for shorter sequences. Self-attention can be restricted to a neighborhood for very long sequences. Convolutional layers require a stack of layers to connect all input and output positions. Self-attention can lead to more interpretable models by providing insights into sentence structure.

5 Training

The text briefly mentions that there is a training regime for the models but does not provide any specific details about it.

5.1 Training Data and Batching

The training data used for English-German translation consists of about 4.5 million sentence pairs encoded using byte-pair encoding. The vocabulary size is approximately 37,000 tokens. For English-French translation, a larger dataset of 36 million sentences was used, with a vocabulary size of 32,000 tokens. Sentence pairs were batched together based on sequence length, with each training batch containing around 25,000 source tokens and 25,000 target tokens.

5.2 Hardware and Schedule

The models were trained on a machine with 8 NVIDIA P100 GPUs. The training step for the base models took 0.4 seconds each and lasted for 100,000 steps or 12 hours. The big models had a longer step time of 1.0 seconds and were trained for 300,000 steps, equivalent to 3.5 days.

5.3 Optimizer

We used the Adam optimizer with specific parameters for our training. The learning rate was determined by a formula that increases linearly for a certain number of steps and then decreases proportionally to the inverse square root of the step number. We used a warm-up step value of 4000.

5.4 Regularization

Regularization techniques, including Residual Dropout and Dropout, are applied during training to improve performance. The Transformer model achieves better BLEU scores compared to previous models while requiring less training cost. Label smoothing is used during training to improve accuracy and BLEU score, despite potentially increasing perplexity.

6 Results

The big transformer model achieves state-of-the-art results for machine translation tasks in English-to-German and English-to-French. It outperforms previous models and ensembles at a lower training cost. The process of obtaining the final model and details on hyperparameters and inference are described. Table 2 summarizes the results and compares them to other model architectures. The estimation of floating point operations for training is also explained.

6.2 Model Variations

The study evaluates the importance of different components of the Transformer model for English-to-German translation. They varied the model in different ways and measured the

change in performance using beam search. Variation in attention heads and dimensions showed that having too few or too many heads decreases quality. Performance metrics are listed in Table 3, and further variations in model parameters showed that reducing key size hurts model quality, bigger models perform better, and dropout helps avoid overfitting. The study also replaced positional encoding with learned embeddings and observed similar results to the base model.

7 Conclusion

The Transformer is a sequence transduction model based on attention, replacing recurrent layers with multi-headed self-attention. It can be trained faster and achieves state-of-the-art performance in translation tasks. Future plans include applying attention-based models to other tasks, handling input and output modalities beyond text, investigating efficient attention mechanisms, and improving generation. The code is available on GitHub, and the authors acknowledge Nal Kalchbrenner and Stephan Gouws for their contributions.