# Paper Review
# Attention Is All You Need

**Muhammad Umar Salman**
umar.salman@mbzuai.ac.ae

## 1 Summary

Modern sequence transduction models make use of complex recurrent neural networks which include an encoder and decoder architecture. RNNs or variants of RNNs such as LSTM and GRU tackle word-by-word inputs sequentially by generating a sequence of hidden states to factor the sequence of inputs to predict the output. However, the decoder has access to only the last hidden state of encoder stack making it lose or forget early information. After that the attention mechanism was introduced where the decoder now looks at all the states of the encoder such that it can access information about all the elements of the input sequence. In this paper the concept of Transformers is introduced which is the first sequence transduction model which eliminates the use of recurrent layers used in encoder-decoder architecture. The transformers architecture relies entirely on an attention mechanism to draw global dependencies between the input and output. The architecture comprises of stacked self-attention and point-wise, fully connected layers which help the model achieve parallelizability as well as significantly reduces the training time and cost of the model. The experiments were conducted on both WMT 2014 (En-Du) and (En-Fr) translation tasks where the model manages to achieve new state-of-the-art results. This approach has had a significant impact on introducing new state-of-the-art language models and vision models such as BERT, GPT-2 and ViT.

## 2 Strengths

The major strength and core contributions of this paper are that it proposes and entirely novel architecture without any recurrent or convolutional units to deal with sequential data, while being solely based on the attention mechanism. The paper clearly states its motivation by highlighting the sequential flaws that the RNN architecture has and argues on how the proposed model achieves parallelizability through this new architecture. The complexity analysis discussed in the paper validates that the model reduces training time due to reduced complexity in the transformer architecture making it a more efficient method than RNNs. It further manages to established a new state-of-the-art BLEU score on machine translation tasks on standard WMT datasets. The paper also reads very well and is well-structured making it easy to follow. Moreover, the experiments are clearly setup and provide enough details for replication while giving readers access to the code as well.

## 3 Weaknesses

While the architecture of the transformer is described and illustrated very well, architectural details and methodologies lack mathematical definition for example the equation of Multi-Head Attention isn't clearly defined thus making it difficult for readers to grasp it after the first few tries. The transformer model also has a time complexity of $O(n^2)$, so the larger the input sequence the more time it will take the for training and inference. Furthermore, it is argued that the self-attention models have a maximum path length of 1 while still maintaining information of long-range dependencies for longer input sequences. After making these claims of doing a better job than RNNs the paper specifically doesn't conduct experiments on longer sequences inputs with RNN approaches to strengthen their arguement. Lastly, we see many headers which the authors of the paper do explain throughout the paper but leave out in the image caption forcing the reader to go back and read them to make sense of the tables.

## 4 Methodology

Looking at the architecture we can see that we don't have the traditional encoder-decoder that is used in RNNs but there is a new defined concept of encoder-decoder used in Transformers. In Transformers architecture we can see 6 encoders and 6 decoders are stacked together where the previous' output is the next one's output. The last encoder's output is also input into each of the decoder layers. The encoder layer can be broken down into two sub layers, multi-head attention (MHA) and feed forward neural network (FFN). Whereas, the decoder can be broken down into 3 sublayers, the masked multi-head attention (MMHA), the feed forward neural network and the encoder-decoder attention. Each of these sub layers are followed by a dropout layer which acts as a regularizer and an add-norm layer which sums up the output of that layer with its input and then normalizes it using residual connections. As inputs word embeddings of the input sequence are added to a positional encoding before being fed into the MHA layer. This positional encoding is a sinusoidal function. These input embeddings are then multiplied by Wq, Wk, Wv (learnable parameters) matrices to give Query (Q), Key(K) and Value(V). These then go through the MHA layer where multiple self-attentions are performed. The purpose of MHA is to represent different subspaces of words as well as find multiple relations of different words with each other. These outputs of MHA are then fed into FFN where each embedding is independently trained using a ReLu activation to give the output of the encoder layer. This is fed on to each encoder till the 6th one whose output is fed into each of the decoder's encoder-decoder attention sub-layer. The components of the decoder work the same except here the target sequence is input in the beginning. The positional encoding along with the MMHA ensures that the predictions for position i can depend only on the known outputs at positions less than i. After the embedding passes through the MMHA the Q is extracted and is input along with the K and V from the output of the 6th encoder to the encoder-decoder attention layer. This then performs MHA and FFN and finally passes the output out to the linear layer. The linear layer is fully connected neural network that projects the vector produced by the stack of decoders, into a much, much larger vector called a logits vector. Each logit cell represents the score of a unique word. The softmax layer then turns those scores into probabilities where the cell with the highest probability is chosen, and the word associated with it is produced as the output for this time step. The time steps are then repeated till the softmax outputs an end-of-sentence tag.