**Summary**

**Introduction**

Recurrent neural networks, long short-term memory and gated recurrent neural networks were the established models and the ones of prime for sequence modelling and translation problems like machine translation. But the major drawback of all of these models was the inherent sequential nature that prevents parallelization of computation within training examples. This becomes critical while dealing with longer sequence lengths.

Attention mechanisms were used for modelling dependencies. But all these models used attention in conjunction with recurrent networks.

Transformer is the first transduction model that relied entirely on attention mechanisms to draw dependencies between input and output. It even addresses the issue of sequential computation by allowing significantly more parallelization.

So let’s dwell into the transformer for ‘attention’ is all you need :P

**But why self-attention?**

Before we look further into the nuances involving the model let’s compare how the model performed compared to the state-of-the-art models of the time. To understand this better let us look at some performances for various operations to sketch a constructive comparison.

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer Type** | **Complexity/ Layer** | **Sequential Ops** | **Max Path Lenght** |
| Self-Attention | *O(n2 \* d)* | *O(1)* | *O(1)* |
| Recurrent | *O(n\*d)* | *O(n)* | *O(n)* |
| Convolutional | *O(k\*n\*d)* | *O(1)* | *O(logk(n))* |
| Self-Attention(rest) | *O(r\*n\*d)* | *O(1)* | *O(n/r)* |

here d is the dimension of the embeddings, n is the sequence length, k is the kernel size of convolutions and r is the size of the neighbourhood in restricted self-attention

1. **Complexity per layer**

The complexity per layer as seen is greater in the transformer than the recurrent layer when d is greater than n. As one could probably guess this is rarely the case as the dimension of the representations of words embeddings can be as big as 256, 512 or even more and the sequence lengths are generally not that long.

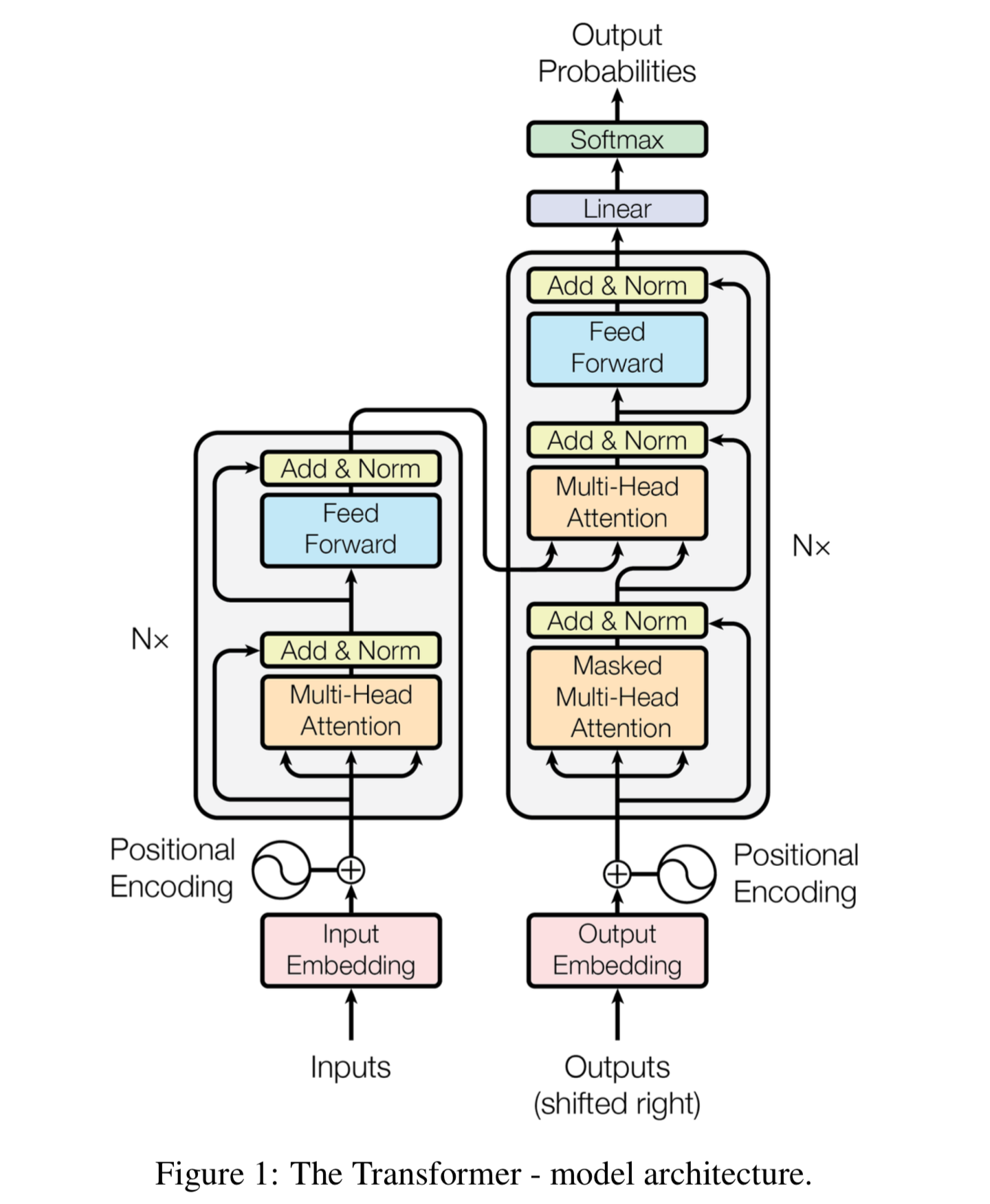
1. **The minimum number of sequential computations required**

Self-attention models require a constant number of sequential computations but recurrent models take *O(n)* operations

1. **Minimum Path Length**

A key factor that affects the learning of long-range dependencies is the length of the path forward and backward signals have to transverse. The longer the length between any two input and output sequences the tougher it is to learn long-relations. Self-attention layers have a constant maximum path length.

**Model Architecture**

****

Transformer uses an encoder-decoder structure. The encoder maps a sequence of representations to a sequence of intermediate representations which are in turn used to map to an output sequence. At each step, the model is auto-regressive which means it consumes the previously generated symbols as an additional output while generating the next.

1. **Encoder and Decoder Stacks**

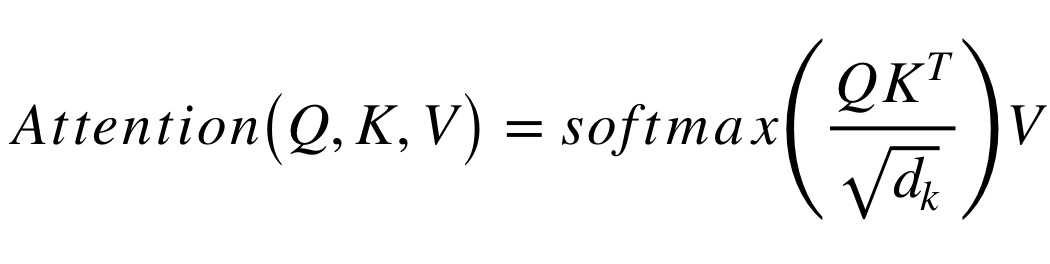
Encoder stack consists of 6 identical layers like the one shown in the picture below. Each layer consists of a multi-head attention layer and a feed-forward layer. We will look at each of these individually later in this article. Each layer is connected through a residual connection which is later normalized. So each sublayer is LayerNorm(*x + Sublayer(x)).* You can refer to [this](https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4) post to know more about ResNets.

Suppose we are trying to train a model a translate Spanish to English(as if enough Americans don’t know Spanish already). A Spanish sentence would be fed into the encoder and a corresponding English sentence would be fed into the decoder with the sequences shifted one position to the right. Okay, so why do that?

This is done to prevent the model from learning a potential copy-paste task which the model might learn since the ith word in one language is generally the ith word in the other. If the decoder sequence is shifted one position the model now starts to predict the (i+1)th given the words of the sequences less than i. Now to make the model predict we mask the subsequent positions in the decoder layer.

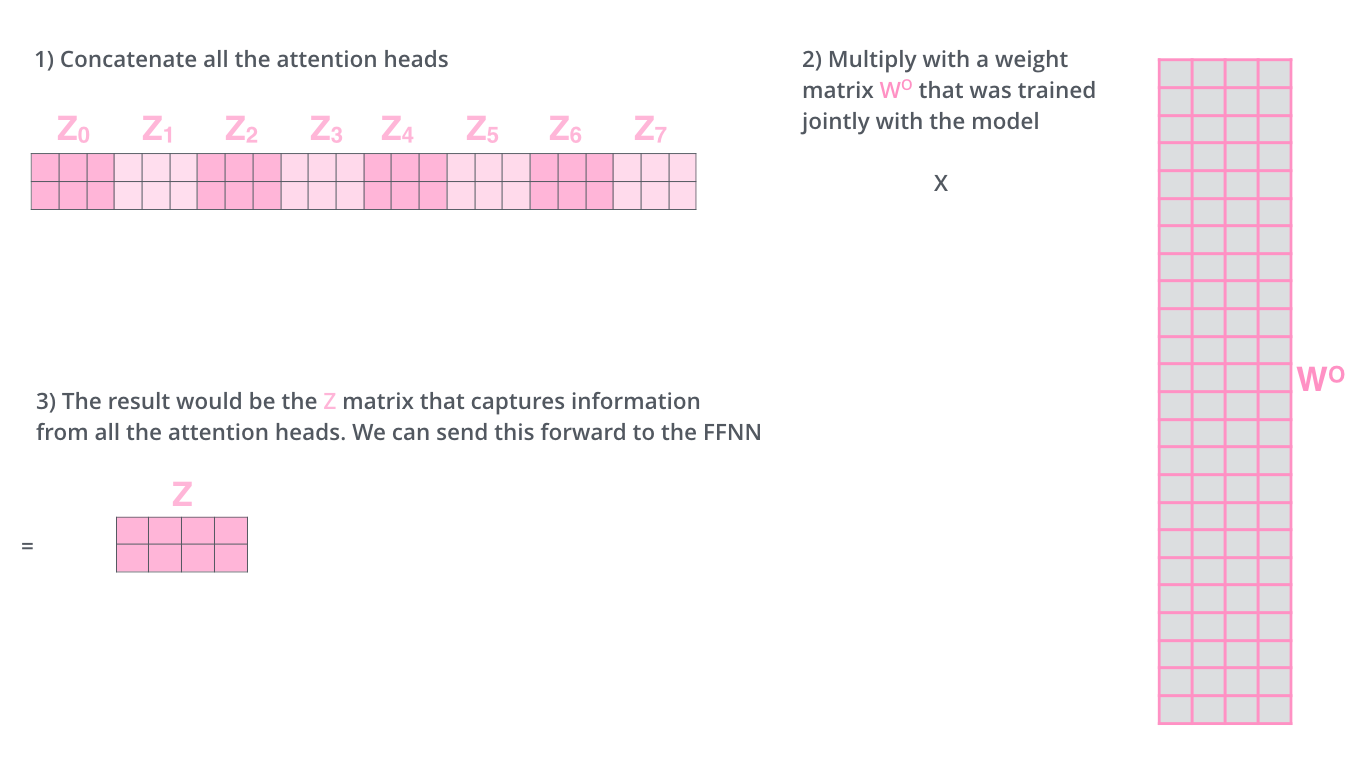
1. **Attention**

The attention block is the heart of a transformer model. It consists of query, key and value matrices mapped to an output.



The scaling factor is used to prevent the dot product to extremely large values which can push the softmax function to extremely small values.

In practice, however, multi-head attention is performed. In multi-head attention, the above process is repeated h times. The resulting matrices are concatenated, multiplies with another weight matrix to produce the attention function.



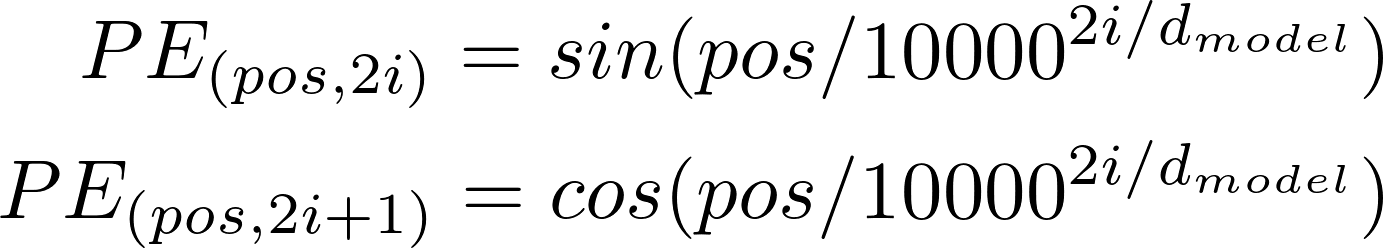
1. **Position wise Feed-Forward Networks**

Each FFN consists of two linear transformations with a ReLU activation in between These are present in each encoder sublayer.

FFN(*x)* = max(0, *xW1 + b1) + b2*

1. **Positional Encoding**

In order to inject some information about the relative or absolute positions of the token in the sequence, positional are added at the button of both encoder and decoder sequence. The dimensions of the encodings are the same as *dmodel*  and hence embeddings are positional encodings can be directly summed up. There are many kinds of positional encodings that can be used. The ones used in the original paper are given below.



**Conclusion**

Transformer outperformed all of the state-of-the-model of that time with even their base model outperforming most. Not only was the transformer model faster it even established a new best BLEU score(an algorithm used to determine the quality of machine translation). The models which have been further built upon the transformer namely BERT, RoBERTa have also better results.