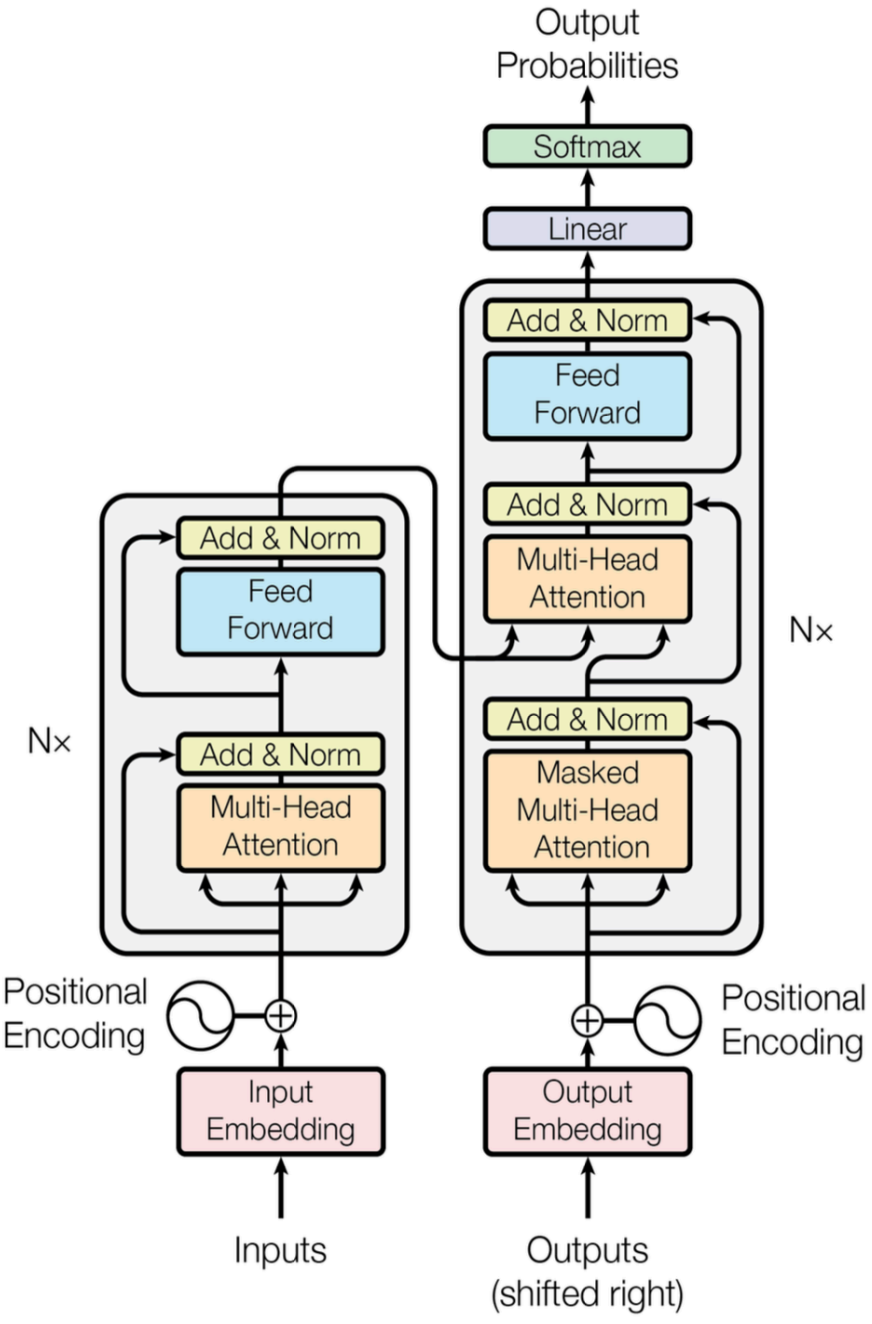
Basic Overview

In this paper the authors propose a purely "attention based" architecture consisting of encoders and decoders without the use of convolutions or recurrent connections. They do this for machine translation and since its only of attention, the resulting architecture is more parallelizable.

Model Architecture

The Model consists of an encoder - decoder architecture. The below self-explanatory image describes how the multiple different blocks in the model is connected with each other. Lets go through them one by one.



Attention

There are three kinds of attention done in this paper. The encoder attention, the decoder attention and the cross attention. Before that, lets see what self attention is. Lets say you have an input X of the shape (n, d). n being the number of tokens and d being the embed dimensions.

Now this X is multiplied with three different weight matrices {shape (d, d)} and these three matmul products are called Queries, Keys and Values, (Q, K, V). all three of them have shape (n, d). Attention is now computed with the following formula.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Now, the unique thing in this paper is multi head attention. Remember how we multiplied x with a (d, d) weight matrices first, now we just multiply them with (d, d/h) for h times, where h is the number of heads. This is similar to just splitting the K, Q and V matrix AFTER the matmul is done with X. Now we have h different 3 tuples of K, Q and V, and we just have to do the attention mechanism h times for these, and then we concat everything in the end.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

This has the same number of operations as the original self attention, but intuitively speaking, this could technically see the same sentence in h different ways. This attention computation is the same for all three types of attention that is used, but we just make a few changes here and there.

And here they divide the QKT matrix by root(dk) (dk = d/h) because they suspect that for large values of dk, the dot products grow large in magnitude. assume that the components of q and k are independent random variables with mean 0 and variance 1. Then their dot product has mean 0 and variance dk.

The encoder attention is literally just the attention mechanism described above.

In decoder attention, a mechanism called causal masking is applied. Lets say I have a sentence "I am a cat". During training, my input sentence would be "[SOS] I am a cat" and my target sentence would be "I am a cat [EOS]". So my model is trying to predict the (i+1) word in the i th embedding. But during self attention, the model would be able to see all the words after it, hence it could just cheat its way through the training. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to -inf) all values in the input of the softmax which correspond to illegal connections. This makes the final softmax outputs of those illegal connections 0, hence not being able to extract any information from the value matrix.

In the cross attention, its just that the Q matrix comes from the decoder and the K, V matrices come from the encoder. The good part here is that the sequence lengths of decoder and encoder can be different.

Feed Forward

The idea is simple enough. Attention is just a linear transformation, hence just stacking attention is useless, thus we implement feed forward for introducing non linearity. This is done position wise and in the paper, they first project it from n x dmodel to n x d ff, then apply relu, then project it to n x dmodel again. dmodel = 512 and d ff = 2048.

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Positional Embeddings

Since attention mechanism is order independent, we have to employ something which will help the model capture the spatial information, else "I am a cat" and "A I cat am" would look the same for the model. Hence, we just add some positional embeddings to the initial input embeddings and hope that the model will make sense out of it. The authors choose these functions for computing the positional embeddings.

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

The reason for selecting these embeddings is that the wavelengths form a geometric progression from 2π to 10000·2π, and they hypothesized that it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PEpos+k can be represented as a linear function of PE pos.

They also experimented using learned positional embeddings, but both versions produced the same results, hence they stuck with the functions because technically it can attend to longer sequence lengths not seen before in training.

Key Advantages of Self Attention

Computational Complexity per layer was less

The parallelizable parts in attention was much higher than compared to convolution and recurrent layers.

In recurrent layers, it was not able to capture long range dependancies because of vanishing gradients. In convolution layers, the inherent nature of the convolution operation prevents it from making any long range dependancies. But attention can easily make any connections

with sequences of any lengths, hence it can pretty easily capture the long range dependancies.