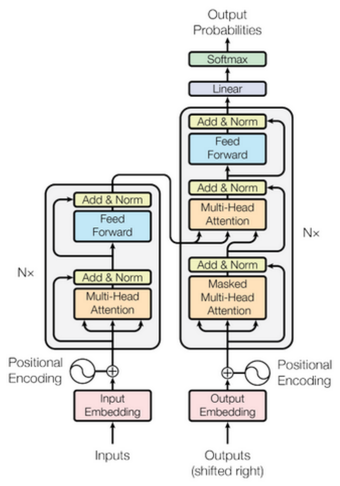
The classical attention model

Recurrent Neural Networks, LSTMs and gated recurrent neural networks were the previous state of the art architectures used for transduction problems. The sequential nature of these models prevented parallelization. Along with that, long sequence context losses were another reason attention mechanisms were developed.

Attention mechanisms help map dependencies irrespective of the distance between tokens. Transformers are a set of model architectures which use only attention mechanisms to map dependencies between input and output.

ByteNet and ConvS2S use convolutional neural networks for modeling representations. Although parallelization is possible in these models, the range of dependencies is an issue. Here the steps needed to compute representations increase with distance. Self-attention, sometimes called intra-attention, is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence [1]. Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence aligned RNNs or convolution [1].



The transformer consists of input and output embeddings, encoder and decoder stacks, and linear and softmax activation layers. The encoder and decoder stacks are essentially multiple encoders and decoders stacked upon each other. Each stack in the architecture proposed by the paper is composed of 6 identical layers. The encoder stack is composed of a fully connected feedforward network and a multi - head attention mechanism.

The decoder stack is similar to the encoder stack. The decoder stack has a third sub-layer that performs multi-head attention over the output of the encoder stack. This sub layer is the masked multi - head attention mechanism. Residual connection and normalization are employed after every sub layer.

The attention mechanism

Attention mechanisms take query, key and value vectors as inputs to generate an output. The output is a weighted sum of values, where weight is assigned on a priority basis.

Scaled Dot-Product Attention

This mechanism computes the dot product of the query with all keys, divides each value with the square root of the dimension of input and applies a softmax function to obtain weights of each value. The function of this mechanism is shown below.

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

Where Q, K, V are the Query, Key and Value matrices respectively and d_k is the dimension of the input. The scaling factor of $\frac{1}{\sqrt{d_k}}$ helps avoid pushing the softmax output to regions where the gradient vanishes.

Multi-Head Attention

This mechanism linearly projects the queries, keys and values h times with different, learned linear projections to $d_k,\ d_k\ and\ d_v$ dimensions respectively.

This mechanism makes use of multiple attention heads, instead of using single attention heads. The attention function is performed in parallel on each of the above projections and the outputs are concatenated and projected once again thus giving us the final outputs.

$$MultiHead(Q,\ K,\ V)\ =\ Concat(head_1,\ ...,\ head_h)W^O$$
$$Where\ head_i = Attention(QW_i^Q,\ KW_i^Q, VW_i^Q)$$

The reduced dimensions of the attention heads means reduced computation, thus making the computation similar to models which use single attention heads.

Feed Forward networks

They are present in both the encode and decoder stacks. The feed forward networks are fully connected. They are made up of two linear transformations and a ReLU activation function. This sublayer is applied to every position. The parameters of linear transformations are varied across layers.

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

Embeddings and softmax

Learned embeddings are used to convert input tokens and output tokens to vectors with dimensions according to the model requirements. The decoder output is converted using linear transformation and softmax function to predict next-token probabilities. The same weight matrix is shared across the two embedding layers and pre-softmax linear transformation.

Positional Encoding

Positional encodings are added to the input embeddings at the bottoms of the encoder and decoder stacks to provide information about the positions of the tokens to the model. To allow the model to easily learn to attend to relative positions, sinusoidal functions (sine and cosine) have been used.

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

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

Here “pos” is the position and “i” is the dimension which implies that each dimension of the positional encoding corresponds to a sinusoid.

Training

For English - German, the WMT 2014 English-German dataset was used. It contains approximately 4.5 million sentence pairs. Byte pair encoding was used to encode sentences. The WMT 2014 English-French dataset was used for English - French. It consists of 36 million sentences. . Each training batch contained a set of sentence pairs containing approximately 25000 source tokens and 25000 target tokens [1].

Hardware and schedule :
Hardware used : NVIDIA P100 GPUs
Number of GPUs : 8

For base models -
Time taken per step : 0.4 sec

Number of steps : 100,000
Total time taken : 12 hours

For big models-
Time taken per step : 1.0 sec
Number of steps : 300,000
Total time taken : 3.5 days

The Adam optimizer was used with following configuration
$\beta_1 = 0.9,\ \beta_2 = 0.98,\ \epsilon = 10^{-9}$
warmup_steps = 4000

Learning rate was varied according to the formula below:

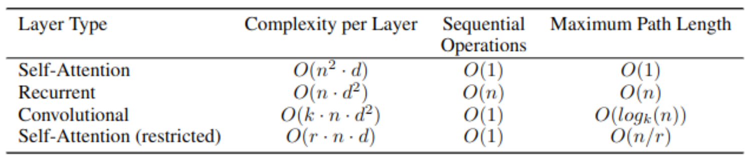
$$lrate = d_{model}^{-0.5} \cdot min(step\_num^{-0.5},\ step\_num \cdot warmup\_steps^{-1.5})$$

The following types of regularization were employed during training :

1. Residual Dropout : Here, dropout is applied to the sum of embeddings, positional encodings, and to the output of each layer before normalization and addition. A dropout rate of 0.1 is used.
2. Label smoothing : During training, label smoothing of value [] was used. This hurts perplexity but improves accuracy and BLEU score.[1].
   $\epsilon = 0.1$

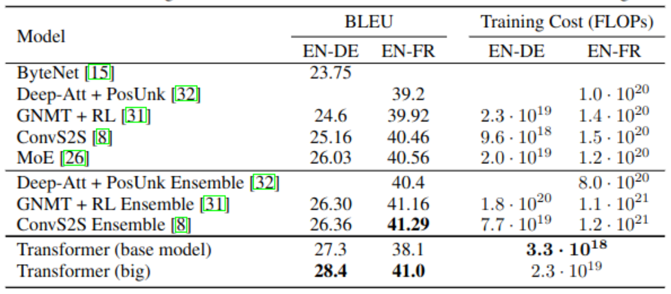
Results

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.



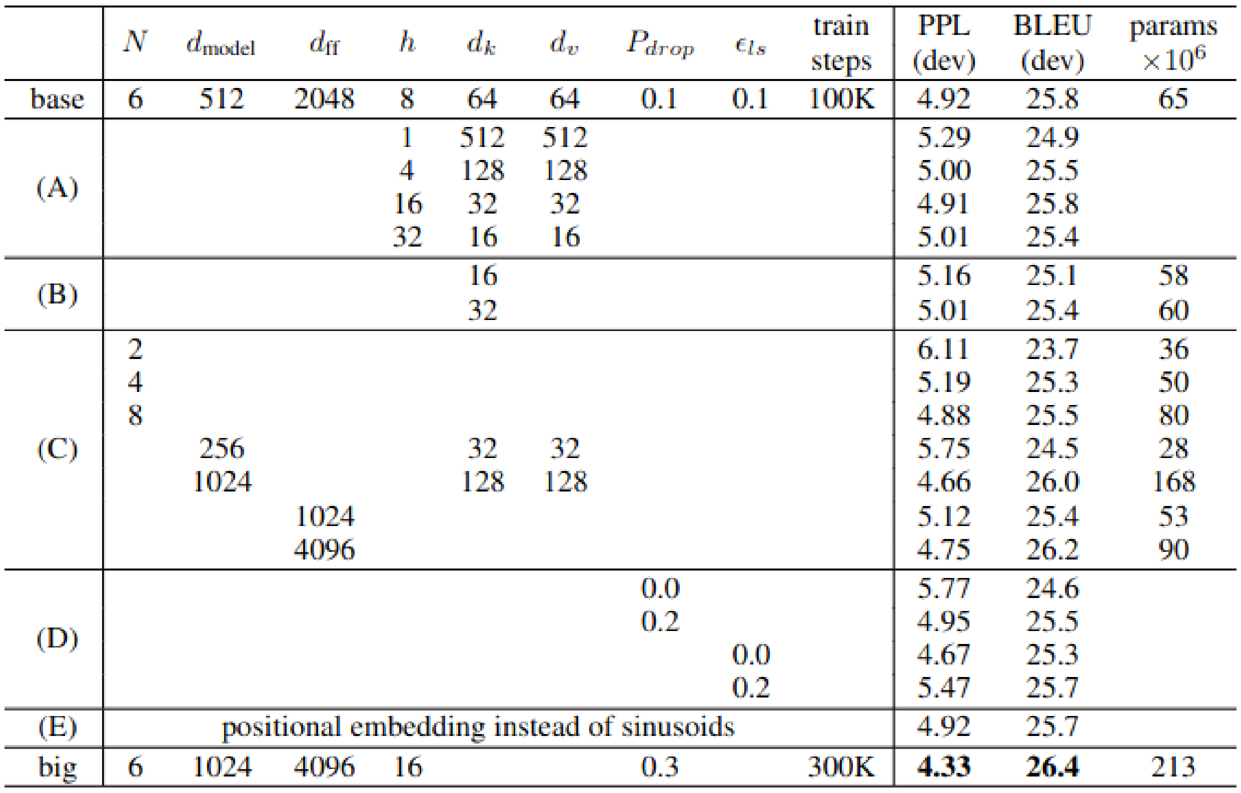
Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.



Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

Table 3: Variations on the Transformer architecture. Unlisted values are identical to those of the base model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.



	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

English constituency parsing

Experiments on English constituency parsing were conducted to test model performance on other tasks. A 4 layer transformer with d-model = 24 was trained on two settings : a WSJ only setting (40K training sentences) and a semi - supervised setting (17M sentences). A vocabulary of 16K tokens was used for WSJ and a vocabulary of 32K was used for semi-supervised setting. During interference, the maximum output length was set to input + 100. For the semi supervised setting, the following parameters were set :

1. Beam size = 21
2. $\alpha = 0.3$

Limitations

- The attention mechanism proposed can only deal with fixed-length text strings. To feed text as input, it has to be split into chunks or fragments.
- Chunking of text leads to context fragmentation, i.e., context is broken into parts and it can lead to the model misinterpreting the provided context

Conclusion

- Transformers have taken over the NLP domain and many variations have become current state-of-art models in many tasks.
- Currently, transformers such as Generative Pretrained Transformers (GPT) models have made big waves in the industry for providing human-like interaction capabilities.
- The biggest drawbacks transformers faced was that they only worked with fixed length sequences and context fragmentation. These were resolved with the introduction of Transformer XL [6].
- Transformers use a huge amount of computation during training. Quantum Hybrid transformers have been proposed to solve some

intractable classical problems and also provide computational benefits by replacing some layers where it is possible to utilize quantum computing to speed up and enhance transformer models' training and inference.

References

[1] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).
[2] Joshi, Prateek. "How Do Transformers Work in NLP? A Guide to the Latest State-of-the-Art Models." Analytics Vidhya, 19 June 2019, How do Transformers Work in NLP? A Guide to the Latest State-of-the-Art Models.