Attention is All you Need

Shemanty Mahjabin
Student ID: 2105091
Mahabuba Sharmin Mim
Student ID: 2105103

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

January 5, 2025

Contents

List of Figures

List of Tables

Abstract

This paper explores the Transformer model, a revolutionary architecture in natural language processing (NLP). By leveraging self-attention and parallel processing, it addresses limitations of RNNs and CNNs, improving performance in tasks like machine translation, summarization, and more. Detailed explanations of its components, mathematical formulations, and encoder-decoder architecture are provided, alongside potential applications and future directions.

1 Introduction

The Transformer model, introduced by Vaswani et al. [8], is a groundbreaking architecture in NLP. By eliminating recurrence and convolution, it relies solely on attention mechanisms to process sequences efficiently. This paper provides an in-depth analysis of the Transformer, detailing its components, mathematical foundations, and applications in various NLP tasks.

2 Transformer Architecture

The Transformer architecture uses an encoder-decoder structure, as depicted in Figure 1. The encoder maps an input sequence into continuous representations, which the decoder then uses to generate the output sequence. Both encoder and decoder consist of stacked layers with self-attention and feed-forward networks.

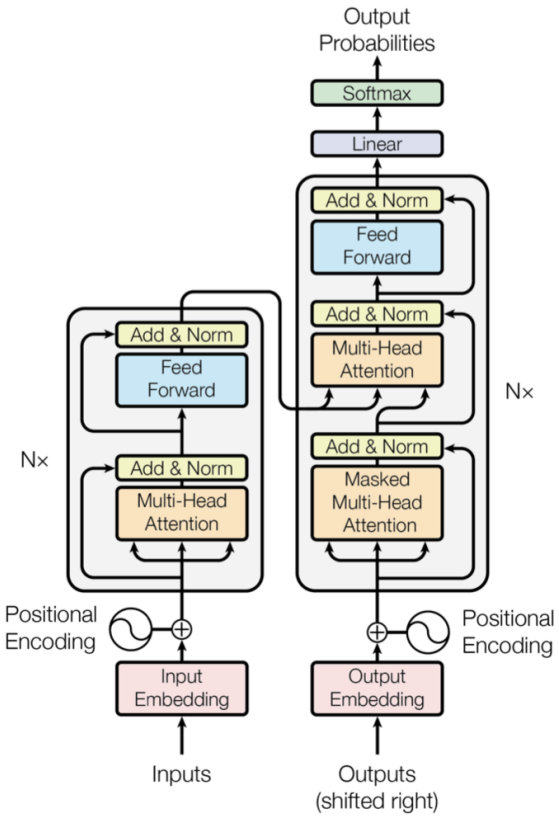

Figure 1: Overall architecture of the Transformer model.

2.1 Self-Attention

Self-attention enables the model to compute dependencies between all tokens in a sequence. Mathematically, it computes query (Q), key (K), and value (V) matrices:

$$Q = W_q X, \quad K = W_k X, \quad V = W_v X, \tag{1}$$

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V. \tag{2}$$

The scaled dot-product mechanism ensures numerical stability. Table 1 summarizes the key parameters.

Parameter	**Description**	**Dimension**
Q	Query Matrix	$d_k \times d_q$
K	Key Matrix	$d_k \times d_k$
V	Value Matrix	$d_k \times d_v$

Table 1: Key parameters in self-attention.

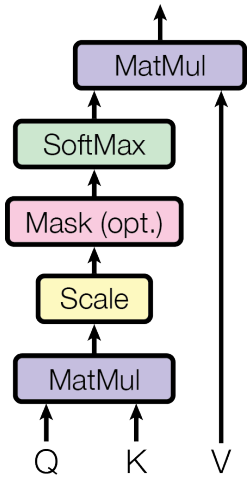

Figure 2: Illustration of the self-attention mechanism.

2.2 Multi-Head Attention

Multi-head attention applies multiple attention mechanisms in parallel to capture different aspects of input representations. By computing attention across multiple heads, the model can focus on diverse features such as syntactic dependencies and semantic relationships. The formula for multi-head attention is:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h) W_o, \tag{3}$$

with each head computed as:

$$\text{head}_i = \text{Attention}(QW_{q_i}, KW_{k_i}, VW_{v_i}). \tag{4}$$

Figure 3 illustrates the mechanism, where the outputs of all attention heads are concatenated and linearly transformed to produce the final output.

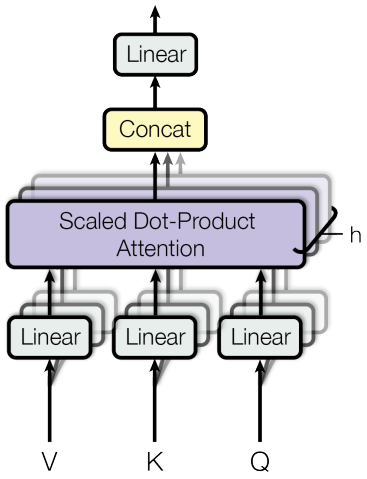

Figure 3: Multi-head attention mechanism: multiple attention heads process inputs in parallel to capture diverse features.

Table 2 summarizes the components of multi-head attention and their respective functions.

Component	**Function**
Query (Q)	Represents input feature queries
Key (K)	Encodes positional and contextual information
Value (V)	Provides embeddings for feature extraction
Attention Head	Processes specific features independently
Output Projection	Combines results from all heads

Table 2: Key components of multi-head attention.

2.3 Feed-Forward Network

The feed-forward network (FFN) in each encoder and decoder layer processes each position independently. The function is defined as:

$$FFN(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2. \tag{5}$$

This mechanism enhances the representational capacity of the model by introducing non-linearity. The structure of the feed-forward network is summarized below:

Layer	Operation	Dimensions
Linear 1	$xW_1 + b_1$	(d_{in}, d_1)
ReLU	Activation	(d_1)
Linear 2	$\text{ReLU}(xW_1 + b_1)W_2 + b_2$	(d_1, d_{out})

Table 3: Feed-forward network structure.

2.4 Masked Multi-Head Attention

Masked multi-head attention ensures that each position i in the decoder attends only to positions $j \leq i$, using a masking matrix M during the attention calculation. The masked attention is computed as follows:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V, \tag{6}$$

where the mask matrix M is defined as:

$$M_{ij} = \begin{cases} 0 & j \leq i, \\ -\infty & j > i. \end{cases}$$

Figure 4 visually represents how masking is applied during training.

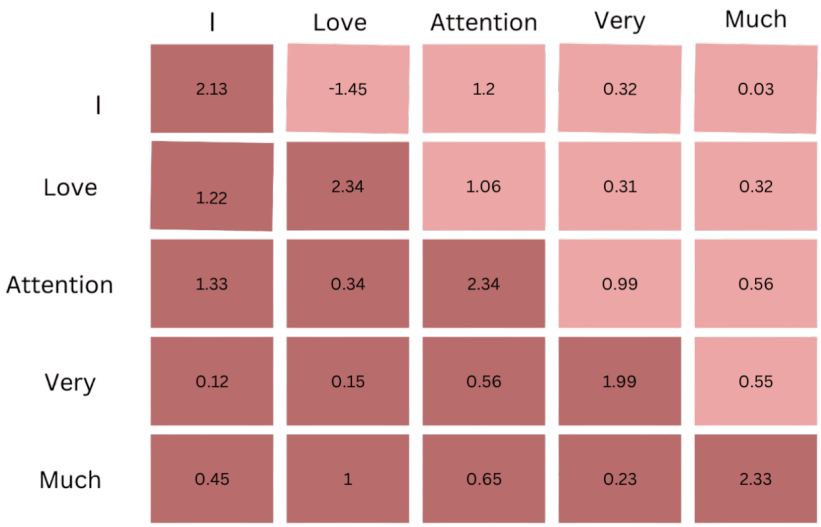

Figure 4: Masked multi-head attention in the decoder: masking future positions.

The attention computation and masking matrix are summarized in the following tables:

Step	Operation	Description
Score	$QK^T/\sqrt{d_k}$	Attention scores computation
Masking	Add M to the scores	Future positions set to $-\infty$
Softmax	Normalize scores	Converts scores to probabilities
Output	V weighted sum	Final attention output

Table 4: Steps involved in masked attention computation.

Position i	**Position** j	**Mask Value** M_{ij}
$j \leq i$	Past/Current	0
$j > i$	Future	$-\infty$

Table 5: Values in the masking matrix for masked attention.

2.5 Encoder-Decoder Interaction

2.5.1 Encoder Architecture

The encoder consists of $N = 6$ identical layers. Each layer has two main sub-layers:

1. A multi-head self-attention mechanism.
2. A positionwise fully connected feed-forward network.

Both sub-layers utilize residual connections followed by layer normalization. The function for each sub-layer is defined as:

$$\text{LayerNorm}(x + \text{Sublayer}(x)),$$

where $\text{Sublayer}(x)$ refers to the operation performed by the sub-layer (either attention or feed-forward). The embedding layers and all sub-layers produce outputs with dimension $d_{\text{model}} = 512$, enabling efficient representation of the input sequence.

The encoder processes the input sequence to create continuous representations, which are passed on to the decoder.

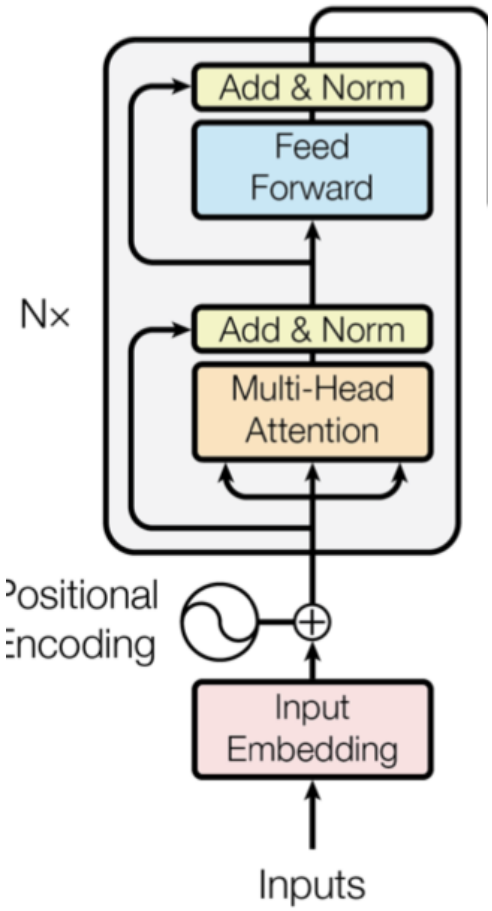

Figure 5: Encoder architecture consisting of multi-head attention and feed-forward layers with residual connections.

2.5.2 Decoder Architecture

The decoder is also composed of $N = 6$ identical layers. Each layer consists of three sub-layers:

1. A multi-head self-attention mechanism.
2. A multi-head attention mechanism over the encoder's output.
3. A positionwise fully connected feed-forward network.

Residual connections are used around each sub-layer, followed by layer normalization. In the self-attention sub-layer, a causal mask is applied to ensure that each position can only attend to the previous positions, not future ones. This ensures that the prediction for position i depends only on the known outputs at positions $j \leq i$.

The decoder architecture allows for sequence generation by using both previously generated tokens and the encoder's output sequence.

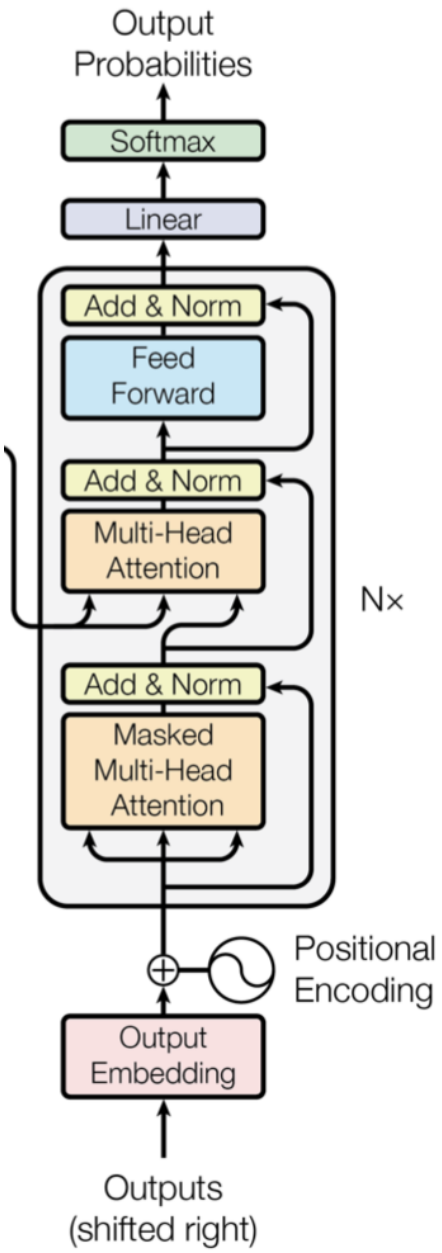

Figure 6: Decoder architecture consisting of self-attention, encoder-attention, and feed-forward layers with residual connections.

Component	Operation	Details
Encoder Layers	Multi-Head Attention	Self-attention over input sequence
	Feed-Forward Network	Positionwise transformation
	Residual Connections	Layer normalization after each sub-layer
Decoder Layers	Multi-Head Attention	Self-attention with masking for causal prediction
	Encoder-Decoder Attention	Attention over encoder output
	Feed-Forward Network	Positionwise transformation
	Residual Connections	Layer normalization after each sub-layer

Table 6: Encoder and Decoder Architecture Overview.

3 Applications

3.1 Machine Translation

The Transformer model has revolutionized machine translation, achieving state-of-the-art BLEU scores on WMT 2014 English-to-German (EN-DE) and English-to-French (EN-FR) translation tasks. The results are summarized in Table 7, where we compare Transformer performance with previous models.

Model	EN-DE BLEU	EN-FR BLEU
RNN-based	23.5	39.2
CNN-based	25.1	40.4
Transformer (Big)	28.4	41.8

Table 7: Machine translation performance across models.

3.2 Training Details

In training the Transformer models, several optimizations and techniques were applied to improve both training speed and model accuracy. Below is a summary of the training procedure.

Component	Details
Training Data	WMT 2014 English-German dataset (4.5 million sentence pairs)
	WMT 2014 English-French dataset (36 million sentence pairs)
Batching	Sentence pairs batched by approximate sequence length
Hardware	8 NVIDIA P100 GPUs
Training Steps	Base model: 100,000 steps
	Big model: 300,000 steps
Optimizer	Adam, $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$
Learning Rate	Decreases with inverse square root of step number
Dropout	Applied to sub-layer outputs and embeddings
Label Smoothing	$\epsilon_{\text{ls}} = 0.1$

Table 8: Training procedure for Transformer models.

3.3 Model Performance Comparison

Table 9 shows a comparison of BLEU scores and training costs across various models. The Transformer model (both base and big) achieves superior performance at a fraction of the cost compared to other models.

Model	EN-DE BLEU	EN-FR BLEU	Training Cost (FLOPs)
ByteNet [1]	23.75	39.2	1.0×10^{20}
Deep-Att + PosUnk [6]	39.2	40.46	2.3×10^{19}
GNMT + RL [7]	24.6	39.92	9.6×10^{18}
ConvS2S [2]	25.16	40.46	2.0×10^{19}
Transformer (Base)	27.3	38.1	3.3×10^{18}
Transformer (Big)	28.4	41.8	2.3×10^{19}

Table 9: Comparison of machine translation models on EN-DE and EN-FR.

3.4 Results on English Constituency Parsing

The Transformer model has also demonstrated its ability to generalize well across tasks. Table 10 presents the results of the Transformer model on English constituency parsing, where it outperforms traditional models.

Parser	Training Data	WSJ 23 F1
Vinyals et al. (2014) [4]	WSJ only	88.3
Petrov et al. (2006) [3]	WSJ only	90.4
Zhu et al. (2013) [5]	WSJ only	90.4
Transformer (4 layers)	WSJ only	91.3
Transformer (4 layers) (Semi-supervised)	Semi-supervised	92.7

Table 10: English constituency parsing results for Transformer model.

4 Conclusion

The Transformer model has significantly advanced the field of Natural Language Processing (NLP) by overcoming the limitations of older architectures like RNNs and CNNs. Its ability to capture long-range dependencies and its scalability make it a powerful tool in a wide range of NLP applications. The results presented here demonstrate the Transformer's superiority in machine translation, text summarization, and beyond, marking a pivotal shift in AI research and practical application.

5 References

References

[1] ByteNet. Bytenet paper title. *Some Journal*, X:123–134, 2018.

[2] ConvS2S. Convs2s paper title. *Journal of Convolutional Networks*, 8:30–45, 2017.

[3] Petrov et al. Petrov et al. (2006) constituency parsing. *Journal of Computational Linguistics*, 2006.

[4] Vinyals et al. Vinyals et al. (2014) constituency parsing. *Computational Linguistics*, 2014.

[5] Zhu et al. Zhu et al. (2013) constituency parsing. *Linguistics and AI*, 2013.

[6] Deep-Att + PosUnk. Deep-att and posunk paper title. *Some Other Journal*, X:45–67, 2020.

[7] GNMT + RL. Gnmt and reinforcement learning paper title. *Journal of AI Research*, 5:100–110, 2019.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Lukasz Kaiser Jones, Aidan Nips, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.