Seminar Report

On

ATTENTION IS ALL YOU NEED - LLMs and TRANSFORMERS

Submitted by

STEFIN SHIBY GEORGE

(KTE21EC062)

In partial fulfilment for the award of Degree of

BACHELOR OF TECHNOLOGY IN

ELECTRONICS AND COMMUNICATION ENGINEERING

For the Course

ECQ 413 SEMINAR

Under the guidance of

DR. ANIL KUMAR C.D

Department of Electronics and Communication Engineering

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

KOTTAYAM – 686 501

NOVEMBER 2024

Department of Electronics and Communication Engineering

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

KOTTAYAM – 686 501

CERTIFICATE

This is to certify that the Seminar entitled **ATTENTION IS ALL YOU NEED – LLMs and TRANSFORMERS** is a bonafide work carried out by **STEFIN SHIBY GEORGE**(KTE21EC062) during 2024-2025, in partial fulfillment for the award of the B.Tech degree in **ELECTRONICS AND COMMUNICATION ENGINEERING** under APJ Abdul Kalam Technological University, Kerala.

Seminar Guide **Coordinator**

Head of the Department

ACKNOWLEDGEMENT

First of all, I am grateful to The Almighty God for enabling me to complete this seminar. I am extremely grateful to my Internal Guide, **Dr. Anil Kumar C.D** , Professor of the Electronics and Communication Department, Rajiv Gandhi Institute of Technology, Kottayam, forinspiring and providing sincere guidance throughout the seminar.

I sincerely extend my heartfelt gratitude towards our Seminar Coordinators, **Dr. Anil Kumar C.D**, and **Prof. NelsaAbraham** for their valuable suggestions and continuous support.

I am deeply indebted to **Dr. David Solomon George** , Head of the Department of Electronics and Communication, and wish to express my sincere thanks to **Dr. Prince A**, Principal, for providing all the necessary facilities. Gratitude is extended to all the teaching and non-teaching staff of the Department of Electronics and Communication for their cooperation and guidance.

I am also thankful to my parents for their unwavering support throughout the seminar. Finally, I express my appreciation to all well-wishers and friends who encouraged me during this journey.

Rajiv Gandhi Institute of Technology, Kottayam STEFIN SHIBY GEORGE

ABSTRACT

In recent years, attention mechanisms have revolutionized the field of natural language processing and machine learning. The "Attention Is All You Need" paper by some Google researchers, introduces the Transformer model, which has become the cornerstone of modern NLP architectures.

This seminar delves into the groundbreaking contributions of the paper in shaping the LLMs that exist today. The Transformer architecture leverages self-attention and positional encodings to process sequences in parallel, enabling it to achieve superior performance on tasks such as translation, summarization, and text generation. This presentation will provide an accessible overview of key concepts such as self-attention, multi-head attention, and the encoder-decoder structure of the Transformer. Attendees will gain insights into how these innovations address limitations of prior models and why the Transformer has become the foundation for advanced models like BERT, GPT, and T5.

Through this seminar, I aim to foster a deeper understanding of the impact and future potential of attention mechanisms in machine learning, inspiring researchers and students to explore new applications and advancements in this rapidly evolving field.

CONTENTS

1. Introduction

Attention is All You Need is a seminal paper by Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, published in December 2017 at the 31st International Conference on Neural Information Processing Systems. The authors, then researchers at Google, introduced the Transformer, a novel neural network architecture that relies on attention mechanisms instead of traditional recurrence and convolutional layers to process sequential data like languages and images. This architecture has since been the pivot in AI, giving rise to advanced large language models (LLMs) like ChatGPT, BERT, and Gemini.

The paper addresses the core challenges associated with languages and sequential data, where apart from vectorising a language, capturing relationships between distant words is also truly essential for context. Traditional models like RNNs process data sequentially, often losing track of long-term dependencies and slowing down training. Furthermore, both RNNs and CNNs are limited in their ability to dynamically focus on relevant parts of a sequence based on context. Attention mechanisms resolve these issues by enabling models to assign importance to each word relative to others, supporting efficient parallel processing and a better comprehension of relationships across the sequence.

The Transformer eliminates the need for recurrence (RNNs) and convolutions (CNNs), using self-attention as its central mechanism. Self-attention, also known as intra-attention, allows the model to relate various positions within a sequence, efficiently capturing the global context. The Transformer's Multi-Head Attention mechanism enhances this by enabling the model to focus on multiple sequence segments simultaneously. This innovation makes the Transformer the first architecture to rely solely on self-attention, without RNNs or CNNs, setting a new benchmark for tasks such as language translation, summarization, and question answering.

2. Embeddings

Embedding is the process of converting a non-numerical sequence, like words in a language into an N dimensional vector space such that they capture semantic similarities within themselves.

Transformers use two kind of embedding concatenated together to get a better result. They include,

- Word Embedding
- Positional Embedding

2.1 Word Embedding

Word embedding express the similarities between words in a vocabulary in a vector space as shown in Fig 2.1 below,

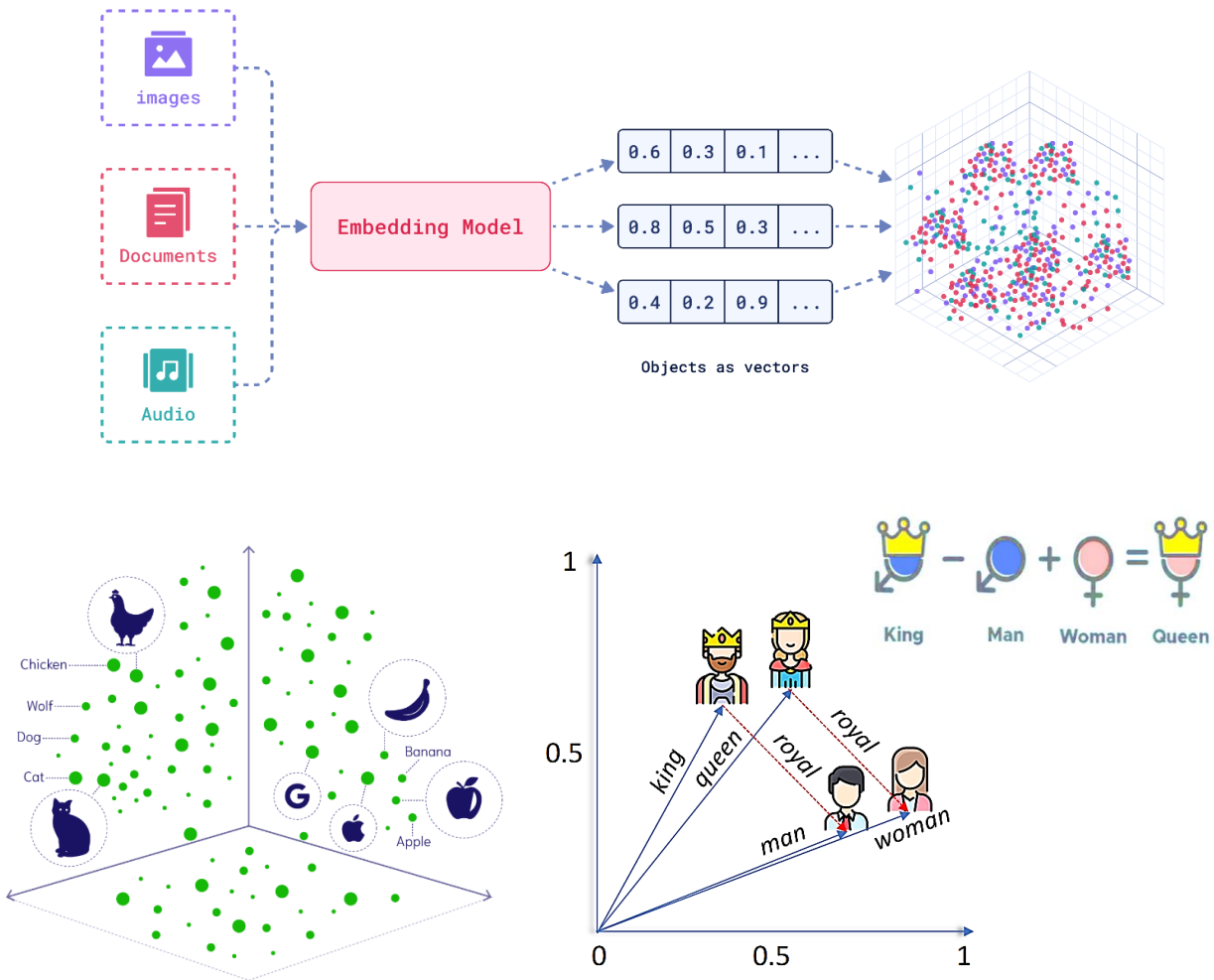

Fig 2.1 Word embeddings in a vector space

2.2 Positional Embedding

The paper introduces a new kind of encoding system apart from word embeddings which is crucial to understand the context in which each word is being used, by incorporating the position information into the embedding.

Each word in the sequence as shown below has their index getting converted to a *d* dimensional vector, forming a positional encoding matrix of the sequence.

The expression for computing the entries in the vector is through the formula,

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

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

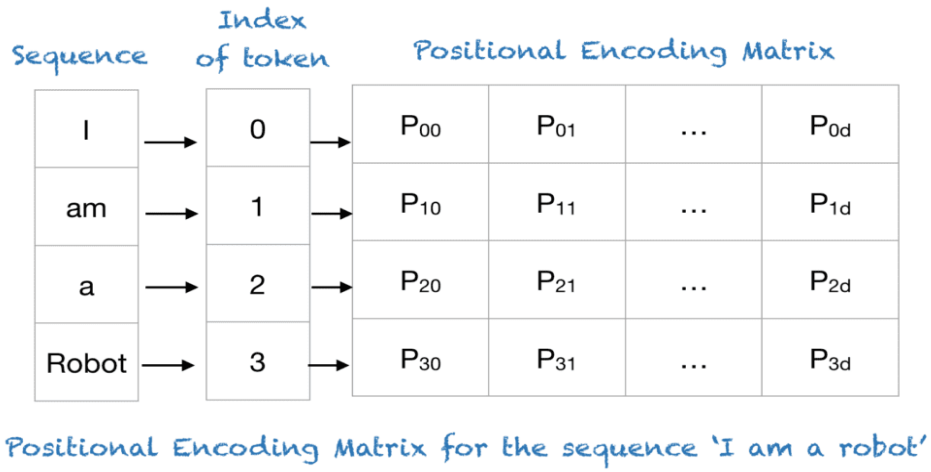

Fig 2.2 Positional Encoding of a Sequence

Both the word and positional embedding are concatenated to get a new embedding, which become the inputs to the Transformer architecture.

3.Transformer Architecture

The transformer architecture is essentially built around two components (Figure 3)

- An Encoder
- A decoder

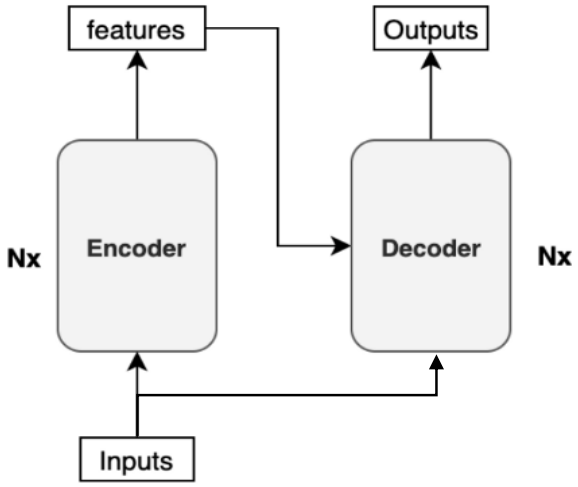

Fig 3 – The Transformer Encoder Decoder Mechanism

The encoder extracts the features and the decoder predicts the outputs from the features from the input vocabulary.

3.1 The Encoder

The encoder's primary function is to process the input sequence and generate a set of continuous representations, or embeddings, that capture the context and relationships between the input tokens.

Each encoder layer consists of two main parts:

1. **Multi-Head Self-Attention**: This component allows the encoder to focus on different parts of the input sequence simultaneously. By computing attention scores for each token relative to others, it learns to weigh their importance in the context of the entire sequence.
2. **Feed-Forward Neural Network**: After the self-attention mechanism, a position-wise feed-forward network processes the output from the attention

layer, applying non-linear transformations to enhance the representation of each token.

$$FFN(x) = ReLU\left(\sum_i x_i W_i\right)$$

The encoder consists of multiple layers stacked on top of each other, with residual connections and layer normalization applied at each step to facilitate training and improve convergence. The output from the final encoder layer is a set of context-aware representations of the input sequence, which are passed on to the decoder.

3.2 Decoder

The decoder is responsible for generating the output sequence based on the representations provided by the encoder. Like the encoder, it consists of multiple layers, but it includes an additional attention mechanism:

1. **Masked Multi-Head Self-Attention**: This prevents the decoder from attending to future tokens in the output sequence during training, ensuring that predictions for a given token are only influenced by previously generated tokens.
2. **Multi-Head Attention**: The decoder has a second attention layer that attends to the encoder's output. This allows it to incorporate information from the input sequence, enabling more contextually relevant generation.
3. **Feed-Forward Neural Network**: Similar to the encoder, each decoder layer includes a feed-forward network to refine the token representations.

The decoder produces the output sequence token by token, utilizing the representations from the encoder to ensure that the generated text is coherent and contextually accurate.

The final output is generated through a SoftMax layer, which predicts the probability distribution over the vocabulary for the next token in the sequence.

Together, the encoder and decoder in the Transformer architecture allow for efficient and effective processing of sequential data, making it particularly powerful for tasks such as language translation, text summarization, and more.

4. Attention

Attention is a mechanism in neural networks that enables the model to focus on specific parts of the input data when generating outputs. It works by calculating attention scores that determine the relevance of each input token in relation to a particular output token.

4.1 Attention Components

When an input matrix enters the transformer three component matrices are developed from it, namely

- Query
- Key
- Value

4.1.1 Query

A query is a vector representation of a specific token or input element that is used to search for relevant information in other tokens within the sequence. It serves as a request for attention, determining how much focus should be placed on other tokens when processing a given input. Each token generates its own query vector, which interacts with the key vectors of other tokens to compute attention scores. The higher the score, the more influence other tokens will have on the final representation of the query token, allowing the model to capture meaningful relationships and dependencies.

4.1.2 Key

A key is a vector representation associated with each token in the input sequence, serving as a point of reference for the attention mechanism. It acts as a descriptor that helps determine how relevant a particular token is in relation to a given query. When the attention mechanism computes scores between queries and keys, it measures the similarity between them, enabling the model to identify which tokens should contribute more significantly to the representation of the query. Essentially, keys help the model organize information by indicating the importance of each token when processing the sequence, facilitating effective attention allocation.

4.1.3 Value

A value is a vector associated with each token in the sequence that holds the actual information to be aggregated during the attention process. Once the attention scores are calculated by comparing queries and keys, these scores are used to weight the

corresponding value vectors. The weighted sum of the values produces the final output for each token, effectively combining information from multiple tokens based on their relevance. Values carry the contextual information that the model uses to construct representations, allowing it to generate more informed and contextually aware outputs.

4.2 Attention Mechanism

Given the query (Q), key (K), and value (V) vectors, the attention mechanism computes a weighted representation of the input tokens based on their relevance to the query.

First, the attention scores are calculated by taking the dot product of the query with all the key vectors, which measures the similarity between the query and each token in the input sequence. These scores are then normalized using a SoftMax function to create a probability distribution, indicating the relative importance of each token in the context of the query.

Next, this distribution is used to weight the corresponding value vectors, allowing the model to aggregate information from the input tokens according to their relevance. The final output is a weighted sum of the value vectors, resulting in a representation that reflects the most pertinent information needed to make predictions or generate outputs based on the given query. Figure 4.2(a) depicts this idea.

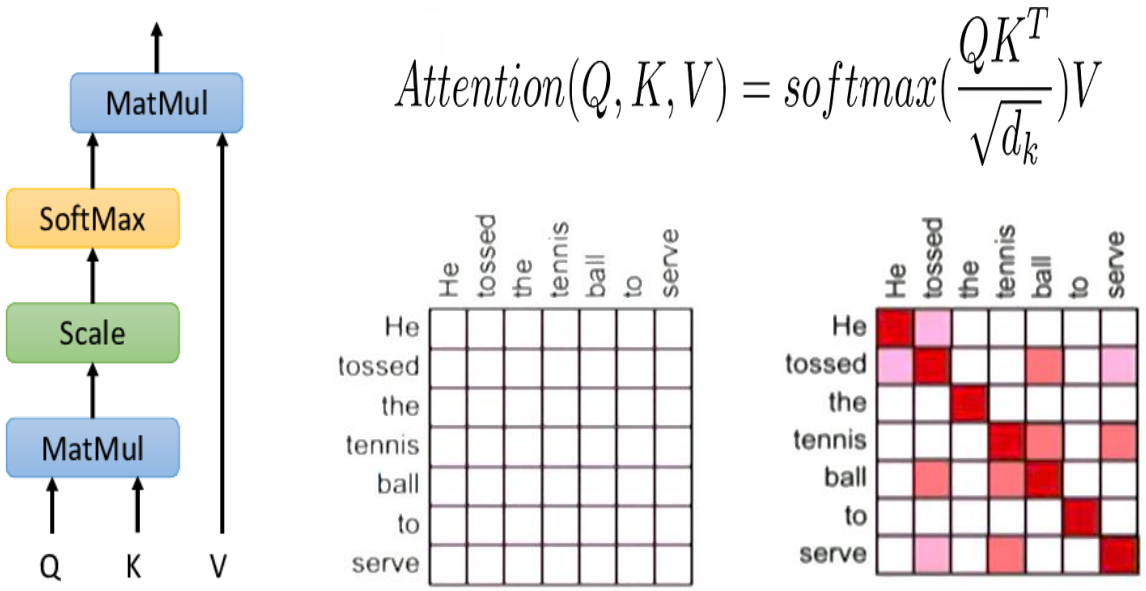

Fig 4.2 (a) Attention from Q,K,V with an example

Attention mechanisms are done several times parallelly and thus called a multi head attention mechanism. Each head focuses on different contexts in the data. A detailed diagram has been shown in figure 4.2(b).

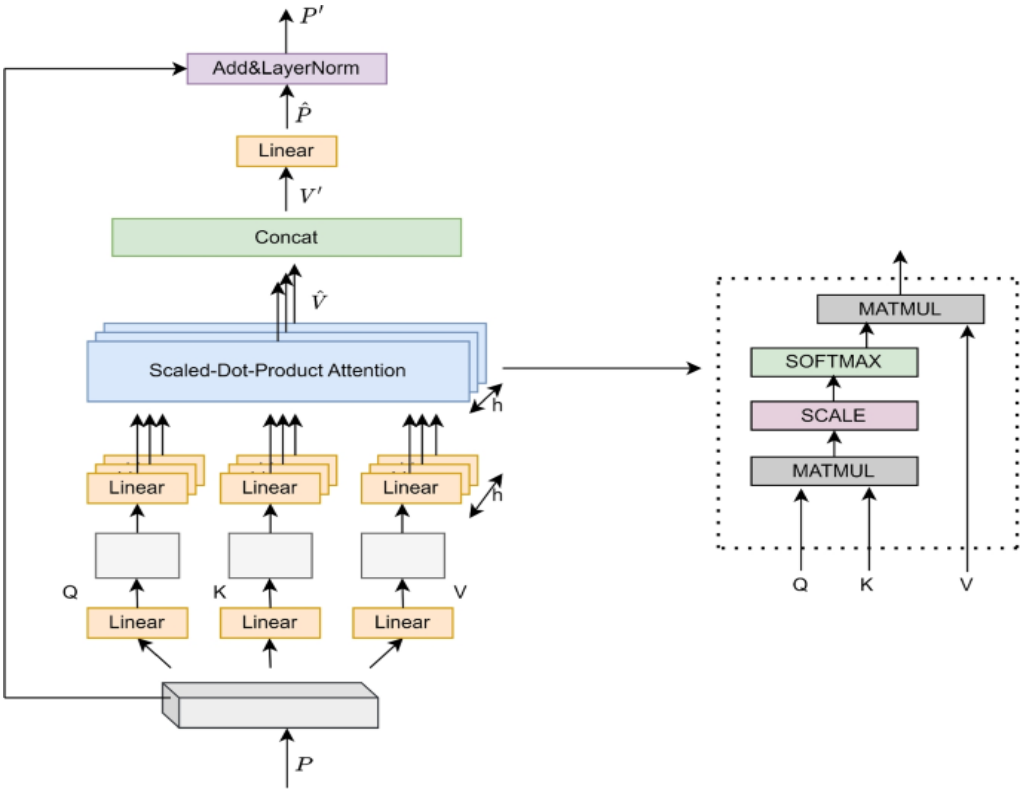

Fig 4.2 (b) Multihead attention

5.The Mechanism

The Transformer begins with input tokens, which are discrete symbols (like words or sub words). These tokens are converted into embeddings through a lookup table that maps each token to a dense vector representation, capturing semantic information. To maintain the positional context of the tokens, positional encodings are added to these embeddings. Positional encodings are sinusoidal functions that help the model understand the order of tokens, as Transformers process input data in parallel rather than sequentially.

The encoder comprises N identical layers (typically six in the original architecture), each containing two primary components: multi-head self-attention and a feed-forward neural network.

1. Multi-Head Self-Attention

    - Each input embedding is transformed into three vectors: queries (Q), keys (K), and values (V) using learned linear transformations. This is achieved through weight matrices which are actually learnt while training the transformer.
    - The attention scores are calculated using the formula and the SoftMax function normalizes the scores to create a probability distribution.
    - Multi-head attention involves splitting Q, K, and V into multiple heads (e.g., 8 heads) to allow the model to attend to different parts of the sequence independently. The outputs from each head are concatenated and linearly transformed.

2. Feed-Forward Neural Network

    - After self-attention, the output passes through a feed-forward network consisting of two linear transformations with a ReLU activation function.
    - Each layer also employs residual connections and layer normalization to stabilize training and improve gradient flow.

The final output from the encoder is a set of enriched embeddings that encapsulate contextual information for each token. These embeddings are then fed into the decoder as the queries and keys to another multi head attention in the decoder layer.

The decoder is structured similarly to the encoder but with an additional masked multi-head self-attention layer and a cross-attention layer.

1. Masked Multi-Head Self-Attention
    - This layer is identical to the self-attention layer in the encoder, but it masks future tokens to prevent information leakage during training. This is achieved by setting the attention scores for future tokens to a very low value (negative infinity) before applying the SoftMax function.
2. Cross-Attention:
    - The cross-attention layer allows the decoder to attend to the encoder's output. It computes attention scores using the decoder's queries and the encoder's keys and values, facilitating the integration of input context when generating outputs.
3. Feed-Forward Neural Network (FFN):
    - Like the encoder, each decoder layer contains a feed-forward network with residual connections and layer normalization.

The output from the final decoder layer is processed through a linear transformation that maps it to the target vocabulary size, followed by a SoftMax activation to generate probabilities for each token in the vocabulary. During inference, methods such as greedy decoding or beam search are used to select the most likely output sequence based on these probabilities.

Figure 5 shows the transformer model as said by the paper.

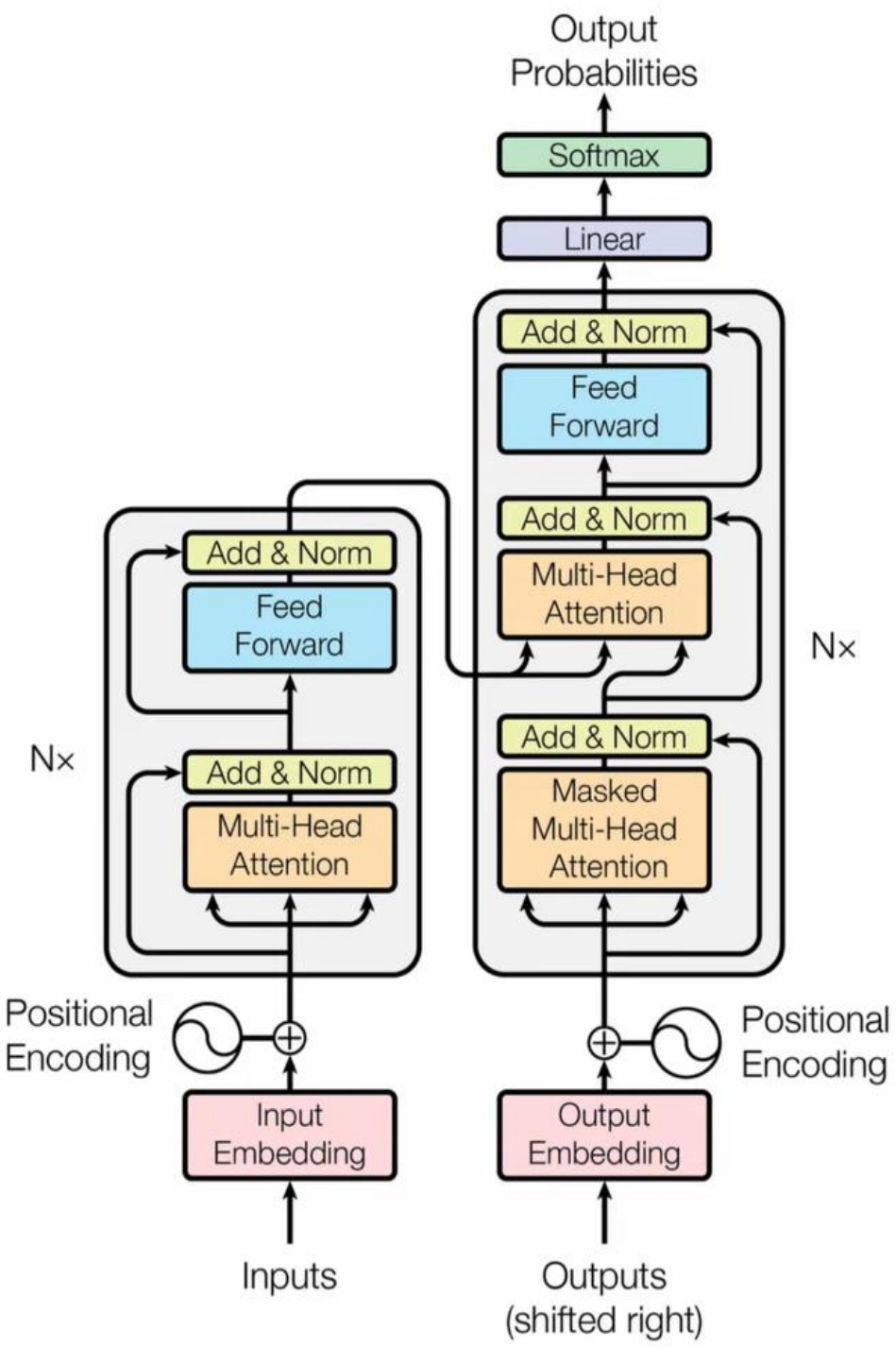

Fig 5 Transformer model as given in the paper

6. Conclusion

In short, the Transformer architecture represents a groundbreaking advancement in the field of natural language processing, enabling models to effectively capture intricate relationships within sequential data. By employing self-attention mechanisms, the Transformer can dynamically weigh the significance of different tokens in relation to each other, overcoming the limitations of traditional models that rely on sequential processing. The use of multi-head attention allows the model to focus on various aspects of the input simultaneously, enhancing its understanding and representation of context. With its parallel processing capabilities and sophisticated architectural components—such as encoder-decoder structures, positional encodings, and feed-forward networks—the Transformer has paved the way for state-of-the-art applications in language translation, summarization, and beyond. As a foundational technology for modern large language models, the Transformer not only demonstrates remarkable performance across diverse tasks but also opens new avenues for research and development in artificial intelligence, shaping the future of human-computer interaction.

.

7. References

[1] https://www.youtube.com/watch?v=ySEx_Bqxvvo (MIT OCW Video)

[2] https://www.datacamp.com/tutorial/how-transformers-work

[3] https://medium.com/codex/understanding-the-transformer-architecture-in -simple-english-8ee30770a1e0

[4] https://jalammar.github.io/illustrated-transformer/

[5] Zhongqiang Huang and Mary Harper. Self-training PCFG grammars with latent annotations across languages. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 832–841. ACL, August 2009.

[6] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In International Conference on Learning Representations, 2017.

[7] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28, pages 2440–2448. Curran Associates, Inc., 2015.