Transformer and Self-Attention Overview

Sun Jiaqi, Liu Xinhao

Department of Mathematics, University of Macau

November 30, 2024

Abstract

Recurrent Neural Network (RNN) has natural advantages in processing sequence information. After the Transformer model was proposed, it replaced Convolutional Neural Network (CNN) and RNN in many machine learning fields and achieved the state of the art. Transformer has an encoder and decoder, and relies entirely on attention mechanism. We detail the structure of the model. The advantages of the attention mechanism are demonstrated through comparison, and we prove the advantage of self-attention with the principle of maximum entropy.

Keywords: Transformer; Self-Attention; Maximum Entropy.

1 Introduction

RNN was the dominant sequence transduction (Sequence to Sequence) model. Many state-of-the-art works are based on RNN [16, 15]. It deals with the sequence sequentially, and is criticized for low efficiency and inadequacy in the far dependence. Furthermore, the information that the hidden state h_t can transmit is always suspected. Some modified CNN models [17, 18] have been proposed for parallel computing. However, due to the size of the receptive field, it is still far from an ideal case to connect information that is far from each other [19]. In the field of language models, we need a larger model that can efficiently process long time series data and is still easy in training.

In November 2022, OpenAI released an AI chatbot ChatGPT [12]. Its fluently conversational capabilities astounded people. Together with Bidirectional Encoder Representations from Transformers (BERT) model [7] from Google, they are all "large pre-trained language models" based on Transformer [5]. Transformer is known as the fourth basic neural network model after Multi-Layer Perceptron (MLP), CNN, and RNN models. Compared with CNN and RNN models, transformer only relies on the attention mechanism to build connections between input and output, without convolutional or recurrent layers, which makes it more concise and efficient. Transformer model is now widely used in language translation, textual entailment, question answering [1, 14, 11], speech recognition [20], image classification [3] and many other fields [21]. In the follow-up works [1, 14], Transformer demonstrated the ability of pre-training, which can well complete the downstream tasks with a little labeled data.

In this paper, we describe the Transformer model in detail and compare it with CNN and RNN models. We give an explanation of attention mechanism by proving Softmax function is the optimal solution for maximizing the entropy, showing the leading position of self-attention.

Model	Year	Parameters	Training	Fine-Tuning
GPT1	2018	117 million	unpublished books	several downstream tasks
BERT	2018	340 million	cloze and next-sentence-prediction	several downstream tasks
GPT2	2019	1542 million	a larger text dataset	no fine-tuning
GPT3	2020	175 billion	a larger text dataset	few-shot learner
ChatGPT	2022	maybe 1.3 billion	reinforcement learning and human-in-the-loop methods	few-shot learner
GPT4	2023	unknown	text and image	few-shot learner

2 Transformer Model

Transformer is a sequence transduction model implementing the competitive encoder-decoder structure as shown in Figure 1. The encoder extracts information from the input sequence, and the decoder produces the final output. The encoder's input and output are in equal length, while the length of decoder's output is not fixed. The decoder is auto-regressive, using the output at the previous moment as the additional input at this moment.

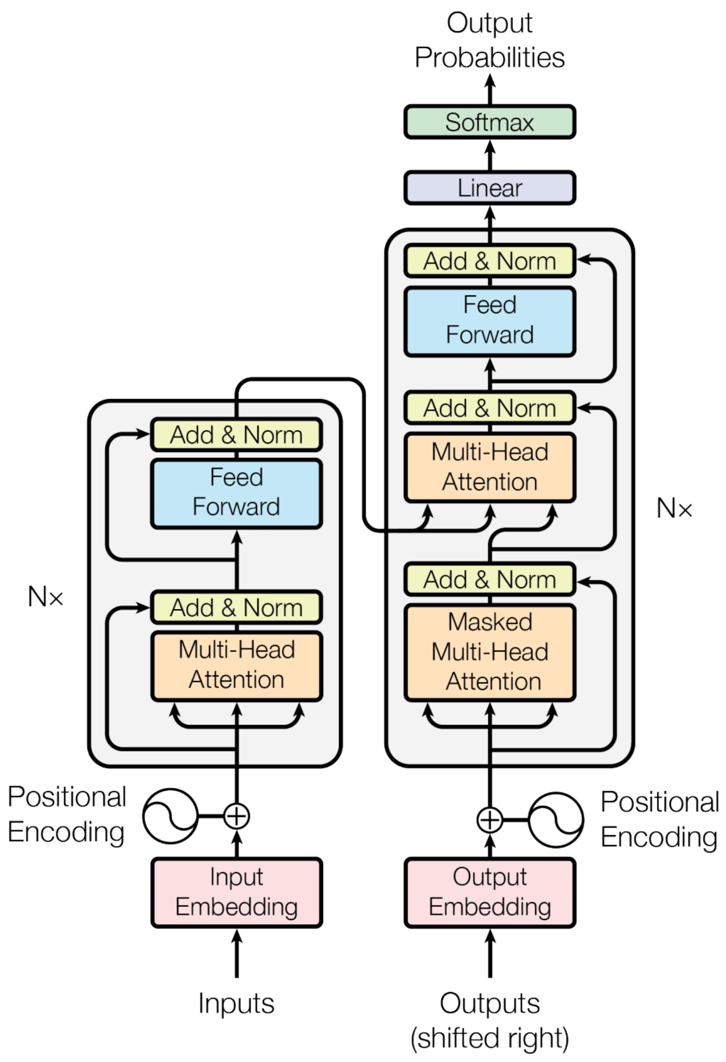

Figure 1: The architecture of the (left) encoder and (right) decoder in Transformer model.

2.1 Encoder

2.1.1 Structure

The encoder consists of a stack of $N = 6$ identical encoder layers. Each layer contains a multi-head attention sub-layer and a position-wise fully connected feed-forward sub-layer. The attention sub-layer will be talked about in detail in Section 2.2. After the attention sub-layer, the author added a MLP with one hidden layer (or two linear layers) as the fully connected feed-forward sub-layer. Dropout and non-linear activation function **Relu** are applied. Since the attention sub-layer has completed the aggregation of information, the fully connected sub-layer will be applied to each token vector individually and identically. The dimension of the hidden layer is $d_{\text{ff}} = 2048$ and the dimension of the output layer is $d_{\text{model}} = 512$. The calculations made by the position-wise fully connected feed-forward sub-layer are as follows

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2, \tag{1}$$

where x is the input vector, W_1 and W_2 are the weight matrices of the fully connected feed-forward sub-layer, b_1 and b_2 are the bias.

2.1.2 Layer Normalization

The normalization method used in Transformer is different from the commonly used batch normalization. It normalizes within each sample and is known as layer normalization [9]. The mean and variance are computed across channels and tokens. There are several reasons for this. First, different samples (or sentences in the real-world case) in the same batch may have different lengths. Layer normalization does not need to save the global mean and variance for the future input. It may normalize each sample more stably on the testing data. Second, from a mathematical view, the layer normalization is invariant to feature shifting and scaling [9]. A comparison between layer and batch normalization is shown in Figure 2.

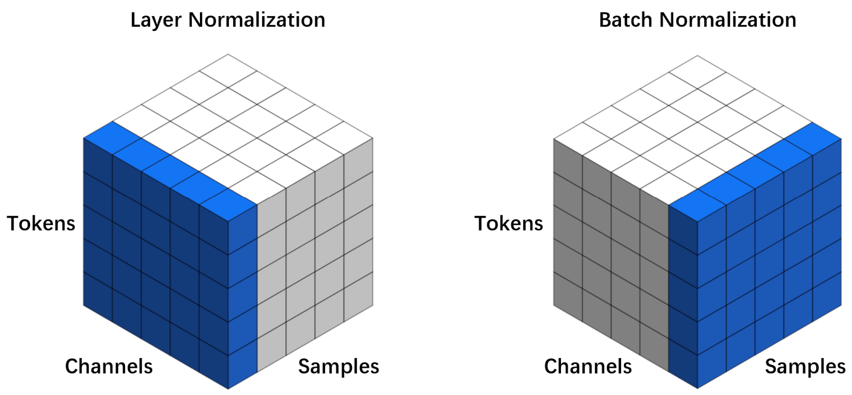

Figure 2: A comparison between layer and batch normalization. Normalization is done within the blue entities.

2.1.3 Residual Connection

Transformer adds a residual connection [10] between each sub-layer, and each sub-layer learns the residual that the previous one haven't learned. This technique allows us to pass on the original input of the sub-layer to the output and artificially constructs a model with a lower complexity. If the previous shallow model have gained most of the information, the weight of the remaining layers should be very close to 0. Residual connections have been proven useful in training deep neural network models [10]. The output of each sub-layer can then be processed as

$$\text{LayerNorm}\left(x + \text{SubLayer}(x)\right), \tag{2}$$

where Sublayer(x) is the function of the sub-layer. From a mathematical perspective, residual connections add shortcuts and mitigate the gradient vanishing problem. This can be proven by a larger gradient in back propagation as follows

$$\begin{aligned} \frac{\partial L}{\partial w_g} &= \frac{\partial L}{\partial w_f} \times \frac{\partial w_f}{\partial w_g} & &\text{(without residual connection)}, \\ \frac{\partial L}{\partial w_g} &= \frac{\partial L}{\partial w_f}\left(1 + \frac{\partial w_f}{\partial w_g}\right) = \frac{\partial L}{\partial w_f} + \frac{\partial L}{\partial w_f} \times \frac{\partial w_f}{\partial w_g} & &\text{(with residual connection)}, \end{aligned} \tag{3}$$

where L is the total loss, w_g and w_f are the parameters of two sub-layers. Each partial derivative is a small value close to 0. Therefore, models with residual connections are less likely to encounter gradient vanishing. From the perspective of information, a sentence may contains multiple levels of information. This can be transmitted across levels through residual connection, which may help Transformer learn semantic knowledge at different levels.

2.2 Multi-Head Self-Attention

2.2.1 Attention as Two Matrix Multiplications

Attention mechanism uses correlations between tokens as the syntactic and contextual structure of the sentence. It uses Query and Key-Value pairs, which come from the information retrieval field, to selectively focus on relevant parts of the input sequence. We define three weight matrices $W_q \in R^{d_{\text{model}} \times d_k}$, $W_k \in R^{d_{\text{model}} \times d_k}$ and $W_v \in R^{d_{\text{model}} \times d_v}$ as the projection of the input onto Query, Key, and Value matrices. Each row of the matrices indicates one token's query, key, and value respectively. The output of the attention sub-layer is a weighted sum of the values, where the weight is the similarity between the query and each key. We do the following calculations

$$Q = xW_q, \quad K = xW_k, \quad V = xW_v,$$
$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{\mathrm{d}_k}}\right)V. \tag{4}$$

The author called it the Scaled Dot-Product Attention. The dot-product is chosen as the compatibility function to represent the similarities between the two tokens. If the dimension of the vector representation for token is large, the magnitude of the result from dot-product will also be large, which may cause greater gaps between weights and vanished gradient. $\frac{1}{\sqrt{d_k}}$ is the scaling factor to ensure the magnitude is under control. Since both the queries and keys come from the same input sequence, this mechanism is called self-attention.

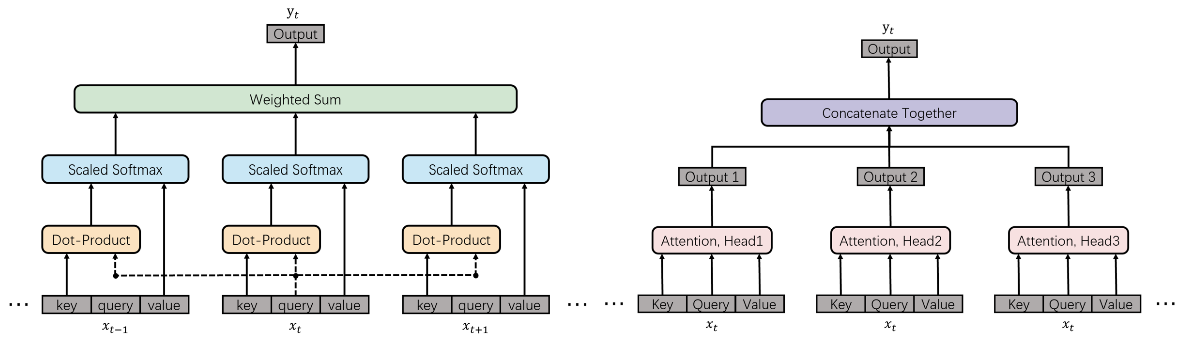

Figure 3: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel. Note that the word with lowercase initial (key, query, value) represent a vector, while one with uppercase initial (Key, Query, Value) represent a matrix.

2.2.2 Multi-Head Attention

In the original attention, the three weight matrices are identity matrices. There are no parameters that can be learned in the attention sub-layer. At this point, we only have one output channel, and the extraction lacks effectiveness. In order to imitate the multiple output channels of CNN, Transformer implements a multi-head attention mechanism, which uses multiple weight matrices to project the input sequence to different low-dimensional spaces. By multi-head, attention may learn different spatial representations. These "heads", similar to the kernels in CNN, can be calculated in parallel. All outputs will be concatenated and projected back to the original dimension. In essence, a multi-head attention sub-layer with H heads is equivalent to repeating the attention mechanism H times. The relation between attention and multi-head attention mechanism is shown in Figure 3.

2.3 Decoder

Decoder contains both of the aforementioned sub-layers with some modifications. It is also stacked by $N = 6$ decoder layers and employs residual connections between the sub-layers followed by layer normalization. The main differences are in the following two aspects.

2.3.1 Masked Multi-head Self-attention sub-layer

The role of the decoder is to predict the output of the next moment. At the time t, the decoder should not use the tokens after time t as the input, otherwise, the output is trivial. Therefore, based on the self-attention sub-layer in encoder, we modify the weights after time t to 0 to ensure that only the tokens, which were generated before time t, participate in the weighted sum. In practice, after multiplying the Query and Key matrices, we add a matrix with -10^{10} after the t-th columns and 0 elsewhere. When calculating the Softmax, the weights related with the time after t will be mapped to 0. The whole procedure is given by

$$\text{MaskedAttention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T + M}{\sqrt{\mathrm{d}_k}}\right) V, \tag{5}$$

where M is a mask matrix with -10^{10} after the t-th columns and 0 elsewhere. The mask will shift right one space after each new token is computed, and the new token will be used to generate the next.

2.3.2 Cross-Attention Sub-Layer

The cross-attention sub-layer is designed to link the input and output sequences together. In this sub-layer, the Query of the attention comes from the previous masked self-attention sub-layer of the decoder, and the Key-Value pairs come from the output of the encoder. Therefore, the attention mechanism changes from self-attention to cross-attention. Each decoder sub-layer receives the final output of the encoder and aims at matching and obtaining the information extracted by the encoder from the input sequence.

2.4 Transformer's Input and Output

2.4.1 Token Embedding

In reality, words are not independent of each other. Since attention uses correlations between tokens as weights to extract information, it will be easier for Transformer to learn semantic information if the tokens are well represented. Transformer implements a learned embedding method to convert tokens to vectors of $d_{\text{model}} = 512$ dimensions in the Euclidean space. In the word embedding layer, vectors are scaled by multiplying $\sqrt{d_{\text{model}}}$. This will keep it the same scale with the later positional embedding to avoid one's domination.

2.4.2 Positional Encoding

Although the attention mechanism brings good information extraction, it does not include positional information. If the input sequence is shuffled to another order, the output is still the same with the new order. Some simple methods like one-hot vectors can be added to the word embedding. In the Transformer paper, the author used sinusoidal functions as follows

$$\begin{aligned} P_{(pos,2i)} &= \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \\ P_{(pos,2i+1)} &= \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \end{aligned} \tag{6}$$

where pos is the position and i is the dimension. The values are shown in Figure 4.

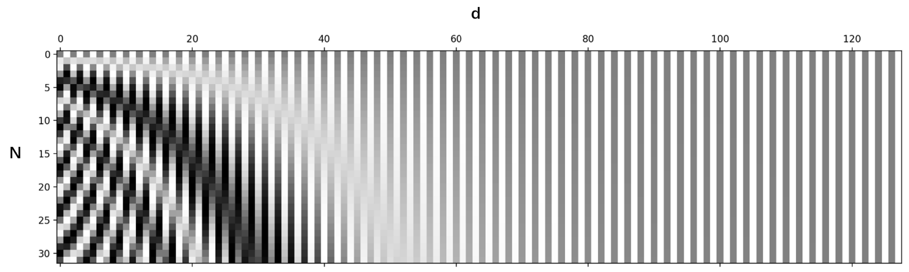

Figure 4: Sinusoidal positional encoding with the length of the input sequence $N = 32$ and $d_{\text{model}} = 128$. The value is between -1(black) and 1(white) and the value 0 is in gray.

2.4.3 Softmax Output

The final output vectors will go through a linear layer, which is then followed by the Softmax function to calculate the probability of each token. The one with the highest probability will be the predicted output and used as the additional input of the decoder in the next moment.

3 Structural Analysis of Attention

In this section, the effectiveness of attention mechanism and the Transformer model are explained mathematically. We will prove that the attention mechanism accurately identifies the relevant information for the input.

As we have already defined in Section 2.2, attention uses a query and a set of key-value pairs as input and calculates the output as a weighted sum of the values. From the mapping in Section 2.2.1, we have the multi-head self-attention given by Equation (4). Before getting into the theoretical proof of the attention mechanism, we start with a simple example "Good Morning!" to show how attention works.

$$\begin{array}{c} \text{Good} \\ \text{Morning} \\ ! \end{array} \begin{bmatrix} 1 & 2 & 1 & 2 & 1 \\ 1 & 1 & 3 & 2 & 1 \\ 3 & 1 & 2 & 1 & 1 \end{bmatrix} \begin{array}{c} \begin{array}{ccc} \text{Good} & \text{Morning} & ! \end{array} \\ \begin{bmatrix} 1 & 1 & 3 \\ 2 & 1 & 1 \\ 1 & 3 & 2 \\ 2 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \end{array} \xrightarrow[\text{The "Good" row}]{\text{Matrix multiplication}}$$

$$\text{Good} \begin{bmatrix} 11 & 11 & 10 \end{bmatrix} \xrightarrow{\text{Softmax normalization}} \begin{bmatrix} 0.48 & 0.48 & 0.04 \end{bmatrix}.$$

Here each row of the first matrix is the vector representation of the token in the sentence. We do the matrix multiplication and only take the "Good" row vector out as an example and normalize it. After the Softmax calculation, the values of the last vector represent the relative scores between "Good" and other tokens. The Softmax function is defined as follows

$$\text{Softmax}(z_i) = \frac{\exp\{z_i\}}{\sum_{j=1}^{N} \exp\{z_j\}}, \tag{7}$$

which utilizes all the entries in the input vector to construct a probability distribution.

3.1 Theoretical Transformation

To deal with the Softmax function, We introduce a new concept Botzmann-Gibbs transformation as

$$\Psi_g(v)(\mathrm{d}x) := \frac{g(x)v}{v(g)}\mathrm{d}x, \tag{8}$$

where $g : E \to \mathbb{R}_{>0}$ is a bounded measurable function, $\Psi_g : \mathcal{P}(E) \to \mathcal{P}(E)$, $\mathcal{P}(E)$ is the probability measure on E and $\nu \in \mathcal{P}(E)$. Since the analogs of attention are all discrete, we define subsets of $\mathbb{R}^d$ as

$$\begin{aligned} E &= \{1, ..., N\}, \\ Q &= \{q_1, ..., q_N\}, \\ K &= \{k_1, ..., k_N\}. \end{aligned} \tag{9}$$

Together with Kronecker delta δ_i^j, which only have values $\{0, 1\}$, a vector $m(K) = \frac{1}{N}\mathbf{1}$, $\mathbf{1} = [1, 1, ..., 1]$ (i.e. the same dimension as the key matrix K), and an interaction potential $G := \exp(Q[i], K[j])$, the particular Boltzmann-Gibbs transformation $\Psi_{G(i,\bullet)}(\mathrm{d}j)$ is equivalent to the Softmax definition. The equality is shown as below

$$\Psi_{G(i,\bullet)}(m(K))(\mathrm{d}i) = \frac{\sum_{j=1}^N \exp(Q[i], K[j])}{\sum_{i=1}^N \sum_{j=1}^N \exp(Q[i], K[j])} = \frac{\exp(Q[i]K^T)}{\sum_{j=1}^N \exp(Q[j]K^T)} = \mathrm{Softmax}(Q[i]K^T). \tag{10}$$

Here we define the Softmax kernel as the stochastic matrix $\Psi_G(m(K)) = \mathrm{Softmax}(QK^T)$, in this way the Softmax function is transformed into an analyzable measure.

3.2 Maximum Entropy Principle

According to the principle of maximum entropy, the probability distribution that accurately portrays the current level of understanding about a system is the one that possesses the highest level of entropy. Since we have put forth a Softmax function that can be subjected to mathematical scrutiny, a further proof will be discussed through this principle.

To calculate the maximum value, we need to define a formula to calculate the value of entropy. Here we use the generalized entropy function

$$H_v(\mu) := \begin{cases} -\mathbb{KL}(\mu \parallel \nu), & \text{if } \mu \ll \nu \\ \infty, & \text{otherwise}, \end{cases} \tag{11}$$

where the $\mathbb{KL} := \int \log(\frac{\mathrm{d}\mu}{\mathrm{d}\nu})\mathrm{d}\nu$ is the Kullback-Leibler divergence. This function generalizes the entropy functional to non-uniform base measures. However we need a distance further to calculate the exact values. Therefore, we define the 1-Wasserstein distance between $\mu, \nu \in \mathcal{P}_1(E)$ as

$$\mathrm{W}_1(\mu, \nu) := \inf_{\pi \in \mathcal{C}(\mu,\nu)} \int \int_{E \times E} \|x - y\|_1 \pi(\mathrm{d}x, \mathrm{d}y), \tag{12}$$

where $\mathcal{C}(\mu, \nu)$ is a set of distributions on $(E \times E, \mathcal{E} \times \mathcal{E})$ which marginals μ, ν separately on the first and second components. Using $\mathbb{W}_1$ we can have a separable complete metric space named 1-Wasserstein space as $\mathcal{W}_1 := (\mathcal{P}(E), \mathbb{W}_1)$. This is equiavlent to the Borel σ-algebra generated by $\mathbb{W}_1$ [8]. Finally, we reached the statement that we can determine whether the Softmax function truly have a maximum point under the principle using the prior definitions [8].

Theorem 1. *Let $k : E \times E \to E'$ be a bi-measurable bounded function, $f : E \to E'$ is a measurable function, $v \in \mathcal{W}_1$ is a constant. We define*

$$Q(v, x) := \{\mu \in \mathcal{W}_1 \mid \mu << v,\ \mu(k(x, \bullet)) = f(x)\}, \forall x \in E.$$

Then for each $x \in E$, there exists a unique maximizer μ_x^* to the maximum entropy problem

$$\max_{\mu_x \in \mathcal{Q}(\nu,x)} H_\nu(\mu_x),$$

where $\mu_x^* = \Psi_{G(x,\bullet)}(\nu)$ with $G(x, y) = \exp\langle\lambda(x), k(x, y)\rangle$ for some $\lambda(x) \in \mathbb{R}^{d'}$. Since $x \mapsto \mu_x^*$ is measurable, $(x, \mathrm{d}y) \mapsto \mu_x^*(\mathrm{d}y)$ is the Markov kernel $(x, \mathrm{d}y) \mapsto \Psi_{G(x,\bullet}(\nu)(\mathrm{d}y)$.

Proof. With an arbitrary measure $\gamma \in \mathcal{Q}(\nu, x)$, $q(x) = \frac{\mathrm{d}\mu_x^*}{\mathrm{d}\nu}$, $n(x) = \frac{\mathrm{d}\gamma}{\mathrm{d}\nu}(x)$, the proof of this theorem is shown below

$$\begin{aligned}
H_\nu(\gamma) &= -\int n(y) \log n(y) \nu(\mathrm{d}y) \\
&= -\int n(y) \log \frac{n(y)}{q(y)} \nu(\mathrm{d}y) - \int n(y) \log q(y) \nu(\mathrm{d}y) \\
&= -\mathbb{KL}\left(\gamma \| \mu_x^*\right) - \int n(y) \log q(y) \nu(\mathrm{d}y) \\
&= -\mathbb{KL}\left(\gamma \| \mu_x^*\right) - \int n(y) [\langle \lambda(x), k(x,y) \rangle - A(x)] \nu(\mathrm{d}y) \text{ by Green's theorem of integration} \\
&= -\mathbb{KL}\left(\gamma \| \mu_x^*\right) - \int [\langle \lambda(x), k(x,y) \rangle - A(x)] \gamma(\mathrm{d}y) \qquad \text{by linearity and } \gamma \in \mathcal{Q}(\nu, x) \\
&= -\mathbb{KL}\left(\gamma \| \mu_x^*\right) - [\langle \lambda(x), f(x) \rangle - A(x)] \\
&= -\mathbb{KL}\left(\gamma \| \mu_x^*\right) - \int [\langle \lambda(x), k(x,y) \rangle - A(x)] \mu_x^*(\mathrm{d}y) \quad \text{by linearity and } \mu_x^* \in \mathcal{Q}(\nu, x) \\
&= -\mathbb{KL}\left(\gamma \| \mu_x^*\right) + H_\nu\left(\mu_x^*\right) \\
&\leq H_\nu\left(\mu_x^*\right).
\end{aligned}$$

□

From this theorem, we can exactly get a unique solution with the maximum entropy principle. Hence the Softmax function can reach the optimal solution of the relevance problem.

4 Related Works

4.1 Post-Norm & Pre-Norm

In the original paper, the normalization is done after the residual connection, which is called post-norm. In a later research [13], pre-norm, which applies normalization on the input before entering the sub-layer, may have better performance in deeper Transformer model. A comparison is shown in Figure 5. This is because post-norm may suffer from gradient vanishing. The calculations are shown in Equation (2) for post-norm and as follows for pre-norm

$$x_{i+1} = x_i + \mathrm{SubLayer}(\mathrm{LayerNorm}(x_i)). \tag{13}$$

The derivatives of loss with respect to x_i for post-norm and pre-norm are as follows

$$\begin{aligned}
\frac{\partial L}{\partial x_i} &= \frac{\partial L}{\partial x_{i+1}} \times \frac{\partial \mathrm{LayerNorm}(y_i)}{\partial y_i} \times \left(1 + \frac{\partial \mathrm{SubLayer}(x_i)}{\partial x_i}\right), \\
\frac{\partial L}{\partial x_i} &= \frac{\partial L}{\partial x_{i+1}} \times \left(1 + \frac{\partial \mathrm{SubLayer}(\mathrm{LayerNorm}(x_i))}{\partial x_i}\right),
\end{aligned} \tag{14}$$

where L is the total loss, $y_i = \mathrm{SubLayer}(x_i) + x_i$. From the equations, we may derive that the pre-norm method has a direct path to pass the error gradient $\frac{\partial L}{\partial x_{i+1}}$, which is the gradient of the layer behind.

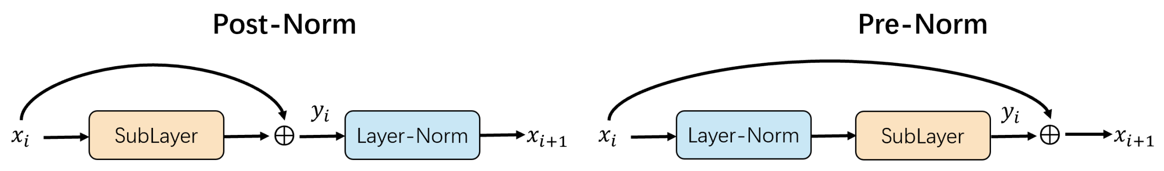

Figure 5: (left) Post-Norm and (right) Pre-Norm process

4.2 Improved Transformer

The original Transformer model uses dot-product and Softmax to calculate the weight of the attention output. But the indivisibility of the exponential function in Softmax makes the calculation quadratic with the length of the input sequence. The Linear Transformer [4] uses kernel to extract features as the weight of attention, and splits the calculation between Query and Key, effectively reducing the computational complexity. In the experiments, it achieved similar results with the original Transformer.

Transformer extracts intrinsic relations among all inputs. When it is applied to the long sequence data, it may require huge amount of memory and computing resources. The Restricted Transformer [20] uses a "time-restricted self-attention" mechanism, which limits the scope of attention to the r neighbours, and uses one-hot position encoding method. It achieved excellent results in speech recognition tasks.

OpenAI proposed the Generative Pre-trained Transformer (GPT) model [2] with 117 million parameters in 2018. It used Transformer's decoder as the basis, pre-trained on unpublished books data, and only required a small amount of labeled data for fine-tuning to complete the downstream tasks excellently. After that, Google proposed to use the encoder as the basis for the BERT model [7] with 340 million parameters, performed pre-training on the tasks of cloze and next-sentence-prediction, and achieved better results. In 2019, OpenAI announced the GPT2 model [1] with 1542 million parameters, which used a larger dataset for pre-training, and set Zero-shot as the selling point. That is, it could complete downstream tasks without fine-tuning. Subsequent GPT3 (175 billion) [14] and ChatGPT [12] models used reinforcement learning and human-in-the-loop methods to create larger datasets for pre-training. In 2023, GPT4 [11] introduced image pre-training, which gave GPT4 the ability to understand multi-modal data.

5 Comparison with CNN and RNN

Here we compare three aspects of Transformer, Transformer's modifications, CNN and RNN models. We assume that all the models' job is to transform a sequence of vectors representation to a new sequence with the same length. The first indicates the computational complexity of each layer. The second is the steps a model needs to process the whole sequence. This also indicates the ability for parallel computing. Last is the steps a model needs to relate two inputs with arbitrary positions, which is important when processing long sequences. The comparison is shown in Table 1.

Table 1: Complexity per layer, number of sequential operations and maximum path lengths for five different layer types. n is the length of the sequence, d is the dimension of vector representation, k is the kernel size of convolutional layer or the number of feature maps in linear self-attention, r is the size of the neighbourhood in time-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	$O(n \cdot k \cdot d^2)$	$O(1)$	$O(\log_k n)$
Time-Restricted Self-Attention	$O(n \cdot r \cdot d)$	$O(1)$	$O(n/r)$
Linear Self-Attention	$O(n \cdot k \cdot d)$	$O(1)$	$O(1)$

The self-attention layer has lower computational complexity than convolutional layer, and is lower than recurrent layer when the sequence length n is less than vector's dimension d, which is most often the case in real-world application. The time-restricted self-attention layer only considers a neighbourhood of size r around the output position. This will decrease the computational performance and increase the maximum path length to $O(n/r)$. Linear self-attention layer separates the Softmax probability computation apart and computes the reusable dot-products between key and value first. This design significantly reduces the computational cost and memory requirement.

Self-attention layers simultaneously focus on relevant parts of the whole sequence, so the sequential operations and maximum path length are $O(1)$. This is in contrast to recurrent layer, which treats sequence sequentially to learn the order of the sentence. All but recurrent layer, have excellent parallel capabilities. Due to the limitation of the receptive field, the path length of the convolutional layer without over lapping is $O(\log_k n)$, or $O(n/k)$ if the stride of over lapping is 1.

6 Conclusion

In our discourse, we presented the Transformer model and its improved versions and conducted a theoretical comparison with the CNN and RNN models. Furthermore, we provided a mathematical analysis of the effectiveness of the self-attention mechanism, elucidating the reasons for its superior performance.

Although GPT4 has achieved amazing performance in daily tasks, it is not yet true intelligence. To the best of our knowledge, we regard it as a model for predicting the probabilities. It keeps predicting the next most possible token, which means it does not really understand the meaning of language, nor does it have the ability to think independently. The intention to "control human beings" exposed at the moment is more due to similar contents in the training data, not self-awareness. The preference on number "42" exactly indicates its intrinsic nature as a statistics model. This highlights the importance of continuing research in language models to advance the language understanding and generation capabilities.

References

[1] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever. Language Models are Unsupervised Multitask Learners. https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[2] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever. Improving Language Understanding by Generative Pre-Training. https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv preprint arXiv:2010.11929v2, 2020

[4] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, François Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. arXiv preprint arXiv:2006.16236, 2020

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser and Illia Polosukhin. Attention Is All You Need. arXiv preprint arXiv: 1706.03762, 2017

[6] Daphne Koller and Nir Friedman. Probabilistic graphical models: principles and techniques. MIT press, 2009.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv: 1810.04805,2018

[8] James Vuckovic, Aristide B aratin, Remi Tachet des Combes A Mathematical Theory of Attention

[9] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. arXiv preprint arXiv: arXiv:1512.03385, 2015

[11] OpenAI. GPT-4 Technical Report. arXiv preprint arXiv:2303.08774, 2023

[12] OpenAI. Training language models to follow instructions with human feedback. arXiv preprint arxiv: 2203.02155, 2022

[13] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, Lidia S. Chao. Learning Deep Transformer Models for Machine Translation. arXiv preprint arXiv: 1906.01787, 2019

[14] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, Dario Amodei. Language Models are Few-Shot Learners. arXiv preprint arXiv:2005.14165, 2020

[15] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer. Deep contextualized word representations. arXiv preprint arXiv:1802.05365, 2018

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation,9(8):1735–1780, 1997

[17] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. arXiv preprint arXiv:1705.03122v2, 2017

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. arXiv preprint arXiv:1610.10099v2, 2017

[19] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001

[20] Povey, Daniel and Hadian, Hossein and Ghahremani, Pegah and Li, Ke and Khudanpur, Sanjeev. A Time-Restricted Self-Attention Layer for ASR. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5874-5878, 2018

[21] William Peebles, Saining Xie. Scalable Diffusion Models with Transformers. arXiv preprint arXiv:2212.09748v2, 2022