Research on Machine Translation Algorithm Based on Transformer

Abstract

The dominant deep learning architectures in machine translation are mainly Sequence-to-sequence architectures, as is the popular Transformer architecture, but its adoption of Multi-head Attention and the elimination of the previous structure of CNN and RNN have led to a better generalization of the model and corpus understanding. and corpus comprehension. In this paper, while implementing the Transformer model and analyzing the principles of each part of the algorithm, selecting English as the source language and Deutsch, French and Chinese as the target languages of the bilingual sentences selected by Tatoeba Project for experimentation on the dataset, and finally obtained the BLEU score of 49.67 for Deutsch and 49.67 for French, and the BLEU score of 49.67 for French and 49.67 for Chinese as the target languages. score of 49.67 for the target language Deutsch, 52.07 for French, and 31.25 for Chinese.

I. Introduction

With the rapid development of the Internet, people's chatting is no longer limited to the domestic, the birth of various social software makes more and more opportunities to chat with foreign friends, which to a certain extent promotes the interest in foreign customs and cultures and learning. For people who are proficient in a certain language, communication and learning is not a difficult thing; however, for ordinary people, different languages may lead to ambiguities in the transmission of information, so translation software such as Google Translate and Microsoft Translate have gradually been introduced to solve the problem of communicating in different languages of most people. In the early days, there were some common problems with these translation softwares, such as simply translating characters or words without considering the context, order, tense and other issues, which led to a certain degree of deviation between the translation results and the ideal. As a result, scientists hoped to solve these problems with the help of computers, and the concept of machine translation gradually emerged in people's eyes.

Natural Language Processing (NLP) is an important branch of the deep learning branch of machine learning, which aims to allow computers to understand, generate, and process human-generated natural language. machine translation belongs to an important application of NLP, which aims to automate the technology of translating a source language text into another target language text. The history of its research can be traced back as far as the 1930s. in 1933, French scientist G.B. Artsouni put forward the

idea of using a machine to accomplish the task of translating natural language, but the idea of machine translation was not formalized at this time. in 1946, the world's first modern electronic computer, ENIAC, was born. Shortly thereafter, the pioneer of information theory, the American scientist Warren Weaver in 1947 put forward the idea of automatic language translation by computer, followed by the publication of the "Translation Memorandum" in 1949, which formally put forward the concepts and ideas of machine translation, followed by scholars have joined the discussion and research. 1954, Georgetown University in the United States in collaboration with the IBM company, the use of the IBM-701 computer for the first time completed the English to Russian machine translation test. In this experiment, the IBM-701 computer automatically translated 60 Russian sentences into English for the first time ever, but no one mentioned that the samples obtained from these translations were manually selected and tested in multiple ways so as to exclude ambiguity. Just as the topic of machine translation was heating up, in 1964 the American Academy of Sciences established the Auto Language Processing Advisory Committee, which, after two years of research, published an ALPAC report in November 1966, which explicitly and comprehensively rejected the feasibility of machine translation, and as a result, major companies stopped The financial support for machine translation research was halted by major companies, thus beginning a frustrating period of about 11 years in which the number of machine translation projects by companies plummeted.

In the mid-to-late 1970s, the rapid development of computer technology and linguistics, the increased demand for social information services, and the breaking down of previous constraints one by one promoted the revival of machine translation research. With the development of computer-related fields, at the end of the 20th century, Brown et al. of IBM, 1993 proposed a word alignment-based translation model for the first time based on statistics, which was an important milestone and marked the birth of modern machine translation methods. Och, 2003 proposed a log-linear model and its weight training method based on the idea of machine learning, and put forward a phrase-based translation model and minimum error rate training method based on machine learning ideas, and the paper published by Koehn et al. in 2003 marked the real rise of machine translation. 2006, Google Translate was formally released as a free service, and people with Internet-connected devices have enjoyed the convenience of fast translation in different languages, which has also provided Google with more translation data to facilitate the development of more machine translation research work. and also provided Google with more translation data to facilitate more research work on machine translation. With the increase in the

amount of data, there are many rare words, long sentences, special words and other traditional machine learning algorithms can't solve the problem, if you increase the steps and complexity of the algorithm, the computer can't complete the training of the weights in a short period of time. With the development of computer hardware, graphics card GPU computing technology is gradually optimized and improved, which is a good solution to the problem of long weight training time, scholars in the algorithm, model architecture can be more bold to increase the computational steps to optimize the defects of traditional machine learning algorithms.

Kalchbrenner and Blunsom, 2013 proposed Recurrent Continuous Translation Models, a novel End-to-end Encoder-Decoder Architecture for machine translation, which uses a convolutional neural network (CNN) to encode a given source text into continuous vectors, immediately followed by the use of Recurrent Neural Network (RNN) as a decoder to convert the state vectors into the target language, which is an important research, and various architectures have subsequently emerged based on the modification of Encoder-Decoder Architecture, such as the one proposed by Sutskever, 2014 Sequence to sequence learning with neural networks, which is different from its predecessor in that both its Encoder and Decoder use RNN-structured LSTMs, and experiments have indicated that the quality of translations has surpassed that of some humans after this architecture went live on Google Translate.Bahdanau , 2015 and others proposed to add Attention weights to the Encoder-Decoder structure, and Vaswani et al., 2017 published a paper based on the idea of Attention weights that shook the field of machine translation, and even the entire field of intelligent technology, and the Encoder-Decoder structure proposed in the paper The Sequence-to-sequence model, Transformer, is different from the previous Sequence-to-sequence model, which abandons all CNNs and RNNs, and the whole network structure is almost composed of Attention mechanism. The main innovations are Multi-Head and Self-Attention, which greatly reduce the convergence time of the model training, and the experimental results show that its performance and robustness are far better than all the model architectures at that time.

In this paper, analyzing the Encoder-Decoder architecture of the Transformer model one by one, introduce and reproduce the details and training strategies of the original Transformer model, and on the basis of its English-to-German and English-to-French models, adding new datasets of English-to-Simplified Chinese to validate the performance of the reproduction.

II. Related studies

A. *Model architecture*

Transformer is the same as most of the competitive neural sequence transduction models, which are Encoder-Decoder architecture, and its overall architecture is shown in Fig. 1.

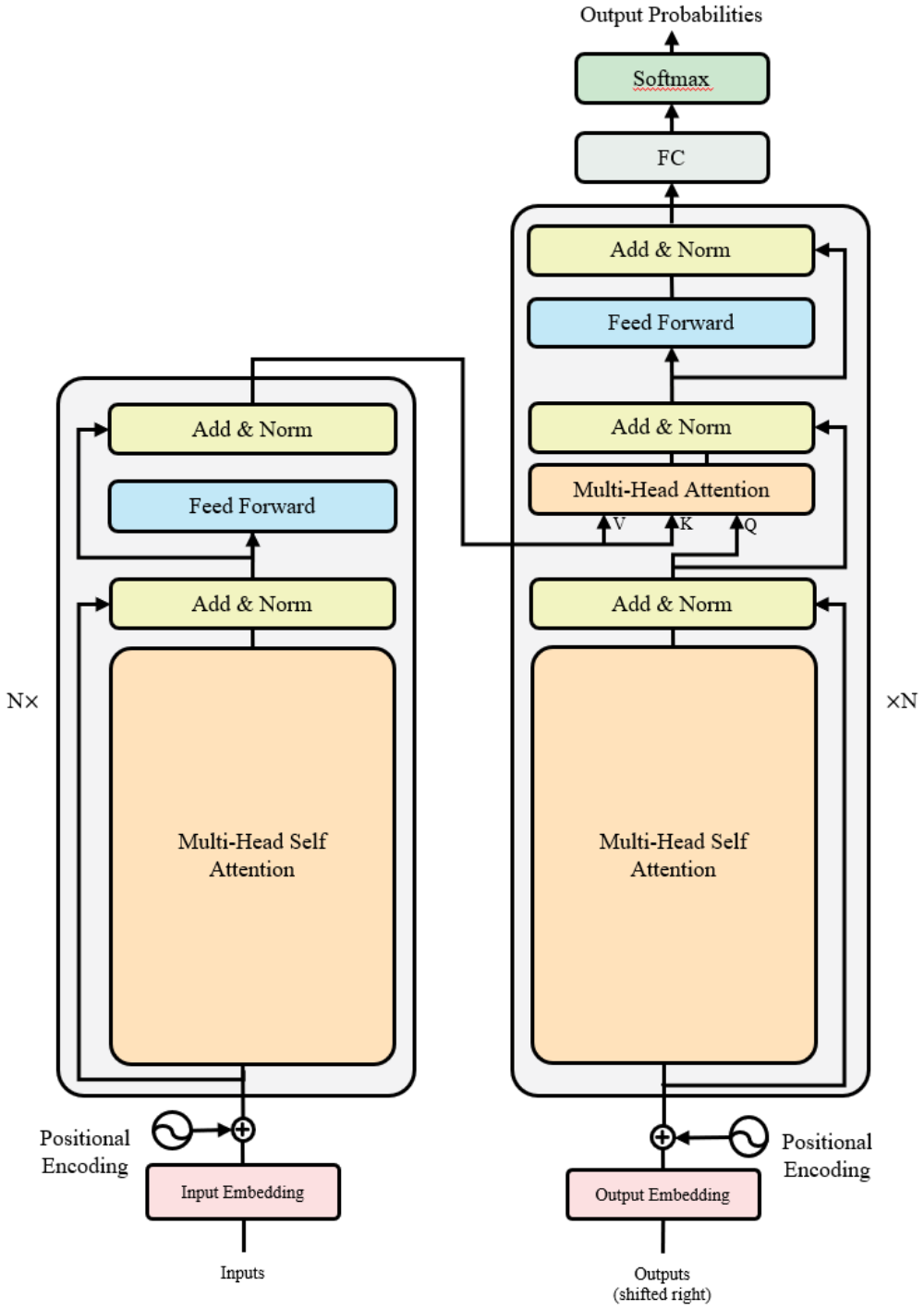

Figure 1. The Transformer model architecture.

Each Encoder/Decoder consists of N Encoder/Decoder blocks in a "sandwich style". The cyan

square FC denotes Fully Connected Layer, also known as Linear Layer; the purple square MatMul denotes Matrix Multiplication; the yellow square Concat denotes Matrix Connection; the yellow square Add & Norm denotes Residual Connection and Layer Normalization (LN).

(1) Encoder/Decoder block

In the overall architecture, both encoder and decoder are composed of N blocks in a "sandwich" design. Assuming that the encoder consists of $N = 6$ encoder blocks, the word vectors Inputs in the shape of $L_{src} \times d_m$ (L_{src} denotes the sentence length of the source language, d_m denotes the dimensionality) are computed by the first encoder block in the forward direction, and then the outputs and inputs are of the same shapes, i.e., they are all of $L \times d_m$ shape. are of $L \times d_m$ shape, and the detailed structure is shown in Fig. 2.

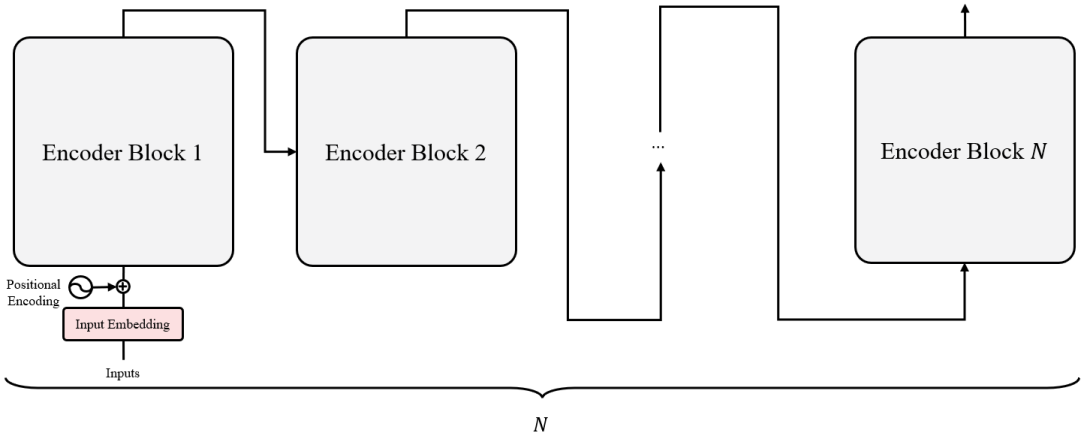

Figure 2. The Encoder forward calculation detailed architecture.

Decoder is the same, but its difference with Encoder block is that an MHA is added after Multi-head Self Attention (MHSA), which is used to accept the output of Encoder block and the output of the MHSA inside the decoder, so as to complete the attention mechanism computation, and the detailed structure is shown in Fig. 3.

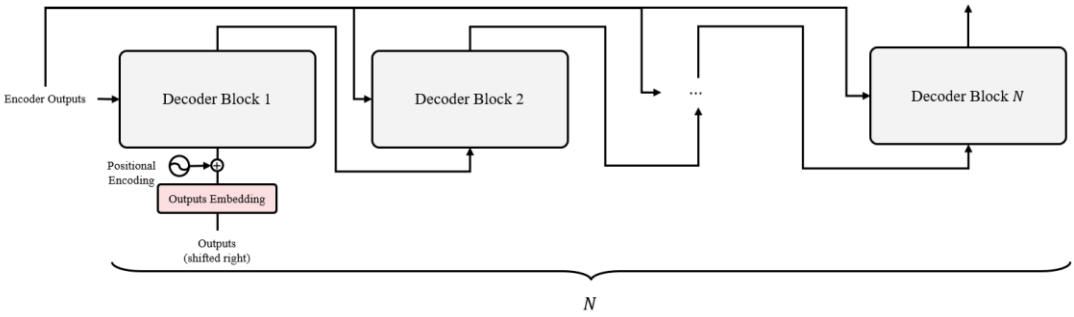

Figure 3. The Decoder forward calculation detailed architecture.

In the training process, the target language word vectors in the shape of $L_{tgt} \times d_m$ will be requested for training, where shifted right denotes the rightward shift of the target language word vectors. Since the data preprocessing part will add "<BOS>", "<EOS>" special symbols to mark the beginning and end of the sentence, the prediction part will be based on the first word "<BOS>" and the input source language word vector. " and the input source language sentence to predict the next word which is the real first word of the sentence. The purpose of this operation is to ensure that for the decoding process, the output of each position is based on the information input from the previous position, so the training process requires the target language sentence to be shifted to the right once in the length dimension.

(2) Multi-Head Attention

Both MHSA and MHA mentioned above are the same computational process, with different inputs leading to different interpretations of their computed attention matrices. The original paper describes that MHA consists of h Single Head Attention, which splits the inputs into h parts, i.e., the shape of the word vectors of each head input is $L \times \frac{d_m}{h}$, which is based on the FC linear mapping and thus realizes the parallel accelerated computation, and finally Concat obtains the same shape as the inputs, in which the attention of the single head is computed using Scaled Dot-product Attention, the detailed structure of MHA is shown in Fig. 4.

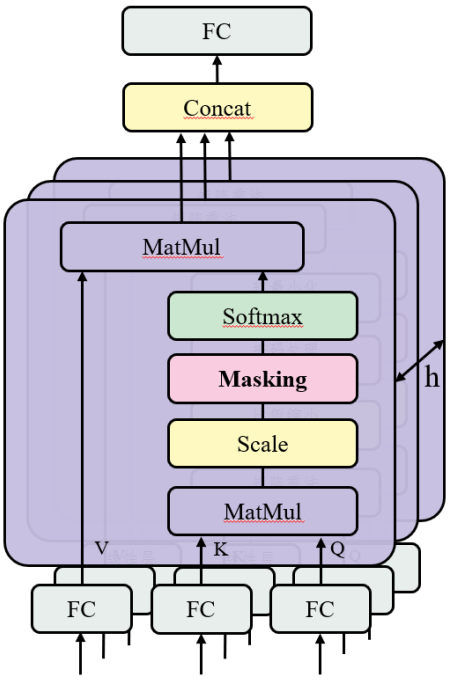

Figure 4. Multi-head Attention architecture.

In the Scaled Dot-product Attention of a single header, the inputs are mapped by three FC layers, Q, K, V, which represent "Query", "Key" and "Value", respectively, "Value", where the matrix dimensions of Q and K are d_k, and the matrix dimension of V is d_v. Obviously, the Q, K, V of MHSA come from the replication of inputs, while the MHA will be different, which is specifically reflected in the next layer of MHA of MHSA of Decoder block, whose K and V come from Encoder outputs. The formula is as in Eq. (1).

$$\text{Attention}(Q,K,V) = \text{Softmax}\left(\frac{QK^T + M}{\sqrt{d_k}}\right)V \tag{1}$$

According to Fig 4. and Eq. (1), it can be known that QK^T is is $L \times L$ square matrix. In encoder Q and K are both linear mappings originating from inputs, so this square matrix can represent the attention weight matrix of inputs to themselves after Scaled, Masking, Softmax, etc.; in decoder Q originates from outputs, which belongs to a kind of soft querying way of outputs to inputs; In the decoder Q comes from the outputs, which is a soft query for the inputs, so that the prediction of the current position is based on the decoding of the previous position. Eventually, the attention matrix is combined with V Matmul to obtain an attention "processed" word vector, which to a certain extent can make the model understand the contextual relationship, thus obtaining better translation results.

In Attention computation, a crucial operation is Masking, in the huge data, it is impossible to guarantee that every sentence is of the same length, so before training, the data will be padding operation to fill the blank characters so that the sentence length is the same in the current batch. In Scaled Dot-product Attention, when computing the attention weight matrix, don't want the blank characters to generate more "attention" to itself, so it need a 0-1 binary Mask to "mask" the padding characters. "Masking is essentially adding an infinitesimal number to the word vector at the position of the filled character, so the corresponding mask value at the filled character is True (i.e., 1), and the rest of the characters are False (i.e., 0), and an infinitesimal number will be accumulated at the True position. Fig. 5.

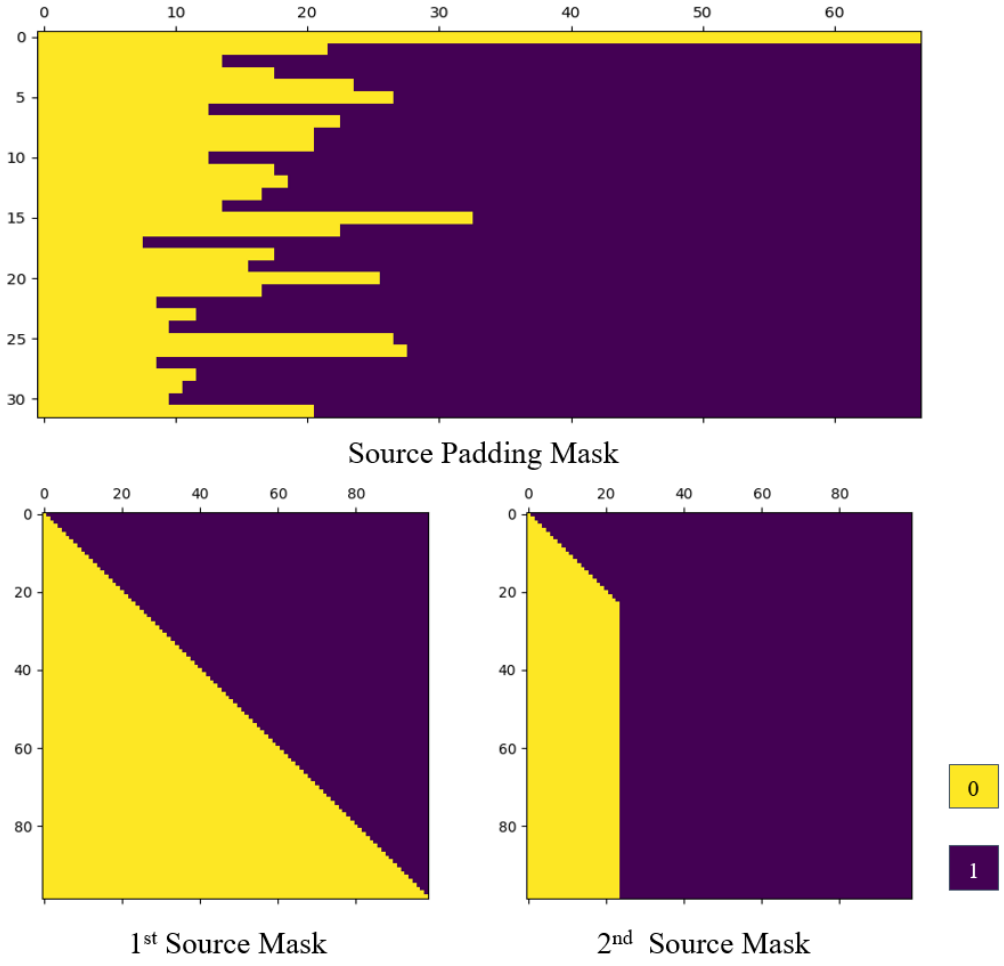

Figure 5. Visualzation of the generated source mask.

For each sentence, a square mask is generated. in batch data training mode, if another batch size number of masks is generated, it will undoubtedly increase the computational consumption of the GPU, so the final source/target Mask is adjusted according to the source/target Padding Mask.

(3) Feed Forward Network

In each Encoder/Decoder block a Feed Forward Network is included, consisting of FC-ReLU-Dropout-FC, which is computed as in Eq. (2).

$$\mathrm{FFN(x)} = \max(0, xw_1 + b_1)w_2 + b_2 \tag{2}$$

where $w_1 \in R^{d_m \times d_{ff}}, w_2 \in R^{d_{ff} \times d_m}$, and d_{ff} is much larger than d_m. The original paper mentions that encoder or decoder blocks may not have the same d_{ff}, and the subsequent experiments in this paper will unify that the d_{ff} is the same. dropout is a means of preventing overfitting, i.e., the forward calculation Dropout is a means of preventing overfitting, i.e., forward computation randomly removes a certain percentage of connections during the process, which can reduce model parameter

overfitting to a certain extent.

(4) Add & Norm

After each MHA or FFN, a Residual Add and Layer Normalization (LN) are performed.Residual Add is a method in the article published by K. He et al., 2015, which proposes that the model structure in which the inputs and outputs of the same shape are accumulated, to a certain extent, avoids forward loss of features in the computation.

Introducing Normalization in neural networks is an effective way to reduce the training time of the model, and the most commonly used method is Batch Normalization (BN), which can accelerate the convergence speed of the model by constraining the data under the current batch through the mean and variance of the batch data in the batch training. However, this approach is not applicable to machine translation models with different sentence lengths, the distribution probability of the same word position in different sentences is different, and the Batch statistics have to be saved and computed in each step, so it is meaningless and time-consuming to constrain the model with BN. Since there may be longer sentences than the maximum length of Training in Testing in machine translation data, there is no training parameter available for the excess word vectors, so it is meaningless to use Batch Normalization as in CNN. In the related research of RNN, Ba, 2016 found that the effect of using LN is better than that of BN, so LN becomes the default config for the subsequent related work. The comparison between LN and BN is visualized as Fig. 6.

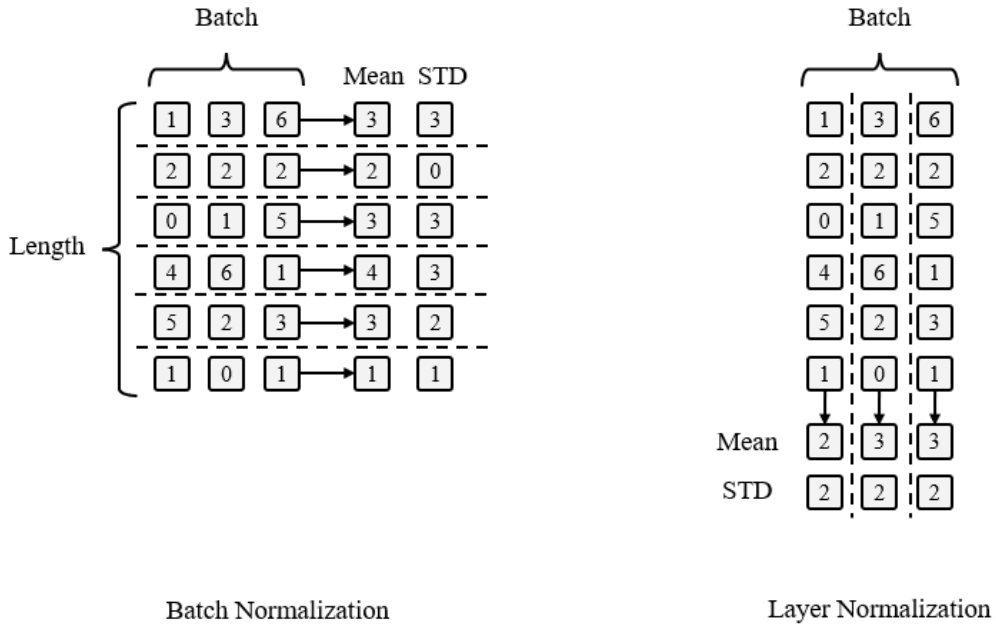

Figure 6. Batch Normalization vs Layer Normalization.

Obviously, BN is normalized to the latitude of batch, and operates on the same feature for different samples, while LN is normalized to the dimension of Hidden, and operates on different features for a single sample. As a result, LN can be unrestricted by the number of samples and ensure that the distribution of each sample is stable, so it is very suitable for machine translation in which the length of sentences varies.

Let the $\mathrm{Sublayer}(x)$ of input x denote the result of the operation of MHA or FFN, from which it can get the result of Add & Norm's calculation x' as in Eq. (3).

$$x' = \mathrm{LN}\big(x + \mathrm{Sublayer}(x)\big) \tag{3}$$

(5) Positional Encoding

A sentence is contextualized, but the elements of the word vector obtained after Embedding are independent of each other, and there is no way to be able to get specific contextual information. Therefore, before entering the Encoder/Decoder, Positional Encoding will be accumulated to make the word vector get positional features. Two methods of positional encoding are provided in the original paper:

(a) Obtained from human empirical formulas, usually taken as the sine-cosine function as follows.

$$\mathrm{PE}_{(\mathrm{pos},2i)} = \sin\left(\frac{\mathrm{pos}}{10000^{\frac{2i}{d_m}}}\right)$$

$$\mathrm{PE}_{(\mathrm{pos},2i)} = \cos\left(\frac{\mathrm{pos}}{10000^{\frac{2i}{d_{d_m}}}}\right)$$

where pos denotes the word index in the matrix and i denotes the dimension. That is, each dimension of the positional encoding corresponds to a sinusoidal curve. The wavelength forms a geometric progression of words from 2π to $10000 \cdot 2\pi$. The sine-cosine function is chosen because it is assumed that it allows the model to easily learn the relative position, based on the empirical formula to visualize its position encoding matrix as in Fig. 7.

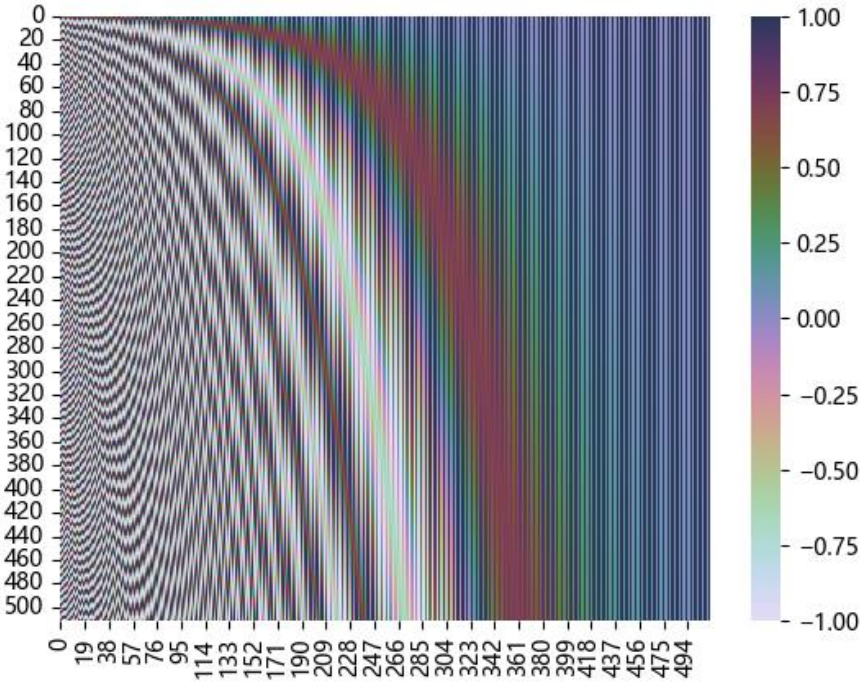

Figure 7. Positional Encoding matrix visualization

(b) Use the weights of the Embedding layer with the word vectors to do Matmul to generate the final input word vectors.

The advantage of the first method is that it can make the model converge in a short period of time, but the upper limit of the model's translation quality depends to a certain extent on the selection of empirical formulas; whereas the advantage of the second method is that once the overall model has converged, the upper limit of its translation quality will be much higher than that of the first one, and the disadvantage is that it is difficult for the model training process to converge, and it may take a long time for the model to converge.

B. Loss function

The original article does not mention the loss function used, according to the requirements of the machine translation task, the processing of the final output results is equivalent to a multi-classification task, in the multi-classification task many scholars take the Cross Entropy loss function, such as Eq. (4).

$$L(y, \tilde{y}) = -\sum_{c=1}^{C} y_c \log \tilde{y_c} \tag{4}$$

where C denotes the total number of categories, which in this case corresponds to the number of dictionaries in the target language. Since Padding operation is performed for sentences of different

lengths in machine translation data, it is not necessary to calculate the loss for padding characters.

C. Evaluation metric

In order to assess the goodness of the model, therefore an evaluation metric is required. In this paper is taken two evaluation metrics Accuracy and BLEU. The reason for using Accuracy is because the loss function uses Cross Entropy and hence it is desired to add one more evaluation metric to measure the model performance.

(1) Accuracy

Accuracy is the simplest and most direct evaluation metric for classification tasks. Suppose that in a positive and negative binary classification task, for the category prediction classification label $\hat{c}_i = \text{positive}$, the original classification label corresponding to it is $c_i = \text{positive}$, i.e., $\hat{c}_i$ is successful in prediction based on c_i, which is denoted by True Positive (TP). Conversely if $c_i = \text{negative}$, i.e. $\hat{c}_i$ fails to predict on the basis of c_i, denoted by False Negative (FN). Similarly True Negative (TN) and False Positive (FP) can be obtained, which usually takes the form of a table or matrix to represent the four values of TP, TN, FP, and FN, i.e., confusion matrix, as in Table 1.

Table 1. Binary classification confusion matrix

	Positive	Negative
Predict Positive	TP	FP
Predict Negative	FN	TN

This leads to the formula for Accuracy as in Eq. (5).

$$\text{Accuracy} = \frac{\text{TP} + \text{FP}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{5}$$

For multi-categorization tasks, it is sufficient to apply the dichotomy to each category.

(2) BELU score

BLEU is an evaluation metric proposed by Papineni, 2002 for machine translation tasks. Assuming that the given reference translation text reference is "there is a cat on the mat", and the model predicts that the sentence candidate translation text candidate is "the the the the the the the", then the original idea of calculating the BLEU score is to evaluate each word in the reference for each word in the candidate. ", then the original idea of calculating the BLEU score is to use the number of times each word of candidate appears in reference as the numerator and the number of words in reference as the denominator, and then the BLEU scores of reference and candidate are 7/7=1, which is obviously not

reasonable. Therefore define the multivariate accuracy $n-\text{gram}$ and call the above single word accuracy score calculation as uni-gram or $1-\text{gram}$, this kind of index can only measure the adequacy of translation. If the above candidate is split into {"the the", "the the", ... , "the the"}; reference is split into {"there is", "is a", "a cat ", "on the", "the mat"} such a group of two neighboring words is called a bi-gram or $2-\text{gram}$, and so on there are ternary, quaternary, and multivariate groups, and then the BLEU score can be calculated to measure the quality of the machine translation model more comprehensively. Generally the BLEU scores of 1-4 tuples are taken, and for the n-gram of BLEU the formula BLEU_n is calculated as Eq. (6).

$$\text{BLEU}_n = P_n = \frac{\sum_{\text{n-gram}\in\text{candidate}} \text{Count}_{\text{clip}}(\text{n}-\text{gram})}{\sum_{\text{n-gram}'\in\text{reference}} \text{Count}(\text{n}-\text{gram}')} \tag{6}$$

where $\text{Count}_{\text{clip}}\ (n-\text{gram})$ denotes the number of occurrences of a particular $n-\text{gram}$ in the reference; $n-\text{gram}'$ denotes the n-grams of the translated text. in a general machine translation task, candidate and reference are one-to-one correspondences, and so the final summed average of the P_n for each control group, the BLEU_n scores of the model in the current dataset. Finally these BLEU_n scores are combined to synthesize a measure of model quality, i.e., a weighted average of the n BLEU_n.

When the lengths of candidate and reference texts are not equal, continuing to calculate BLEU scores according to the above may result in shorter modeled sentences getting higher scores, so the original text defines "best match length": if the length of a candidate translation is greater than or equal to that of the reference text, it is considered to satisfy the best match length, and then it is considered to be the best match length, and the model is considered to be the best match length. If the length of the candidate translation is greater than or equal to the length of the reference text, it is considered to satisfy the best match length, otherwise a Brevity Penalty (BP) is applied, as in Eq. (7).

$$\text{BP} = \begin{cases} 1, & \text{if c} > \text{r} \\ e^{\left(1-\frac{r}{c}\right)}, & \text{if c} \le \text{r} \end{cases} \tag{7}$$

where r denotes the number of $n-\text{gram}$ of reference and c denotes the number of $n-\text{gram}$ of candidate. This leads to the calculation of the final BLEU score as in Eq. (8).

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{8}$$

where w_n denotes the weight of the current $n-\text{gram}$, and w_n is defined as $w_n = \frac{1}{N}$, i.e., weighted average.

III. Analysis, discussion, and future work

A. Bilingual sentence pairs dataset

In this paper, selected bilingual sentence pair datasets from Tatoeba Project will be used to verify the feasibility of the Transformer model. In total, three kinds of bilingual data are taken: English-to-Deutsch, English-to-French, and English-to-Chinese, and the details are shown in Table 2.

Table 2. Dataset details

Language	**Sentences**	**Source vocabulary**	**Target vocabulary**
English-to-Deutsch	255,817	18,973	41,042
English-to-French	197,463	17,473	28,283
English-to-Chinese	29,155	7,899	11,332

Since these datasets are not divided into Training, Validation, and Testing, this paper will divide these datasets, in which 100 are randomly disrupted and selected as Testing and 1000 as validation.

B. Data preprocessing and post-processing

Before the model training, the sentences are segmented and indexed, and before indexing, start and end symbols need to be inserted into the sentences to mark the start and end positions of sentences of different lengths. In this paper, "<BOS>" and "<EOS>" are taken as the start and end symbols, in addition to "<PAD>" as a filler symbol, "<UNK>" as a filler symbol, "<UNK>" as a filler symbol, and "<UNK>" as a filler symbol. "<UNK>" is used as a substitution symbol for the low number of word frequencies. Due to the reason that some words have a low number of occurrences in the whole data, similar to the operation of eliminating data outliers, the words with very low word frequency are replaced with "<UNK>" to prevent the model training process from receiving the influence of out-of-the-box words that lead to a long convergence time.

In the word indexing of sentences, the frequency of all words and punctuation marks in the whole text is first counted, sorted from largest to smallest according to the frequency, and the words are converted into indexes one by one according to their positions. Due to the Encoder-Decoder structure adopted in Transformer, it is not necessary to slice or fill each sentence according to the maximum length, so it is only necessary to fill the batch according to the maximum length of the batch, for example, in the case of batch size 4, such as Fig. 8.

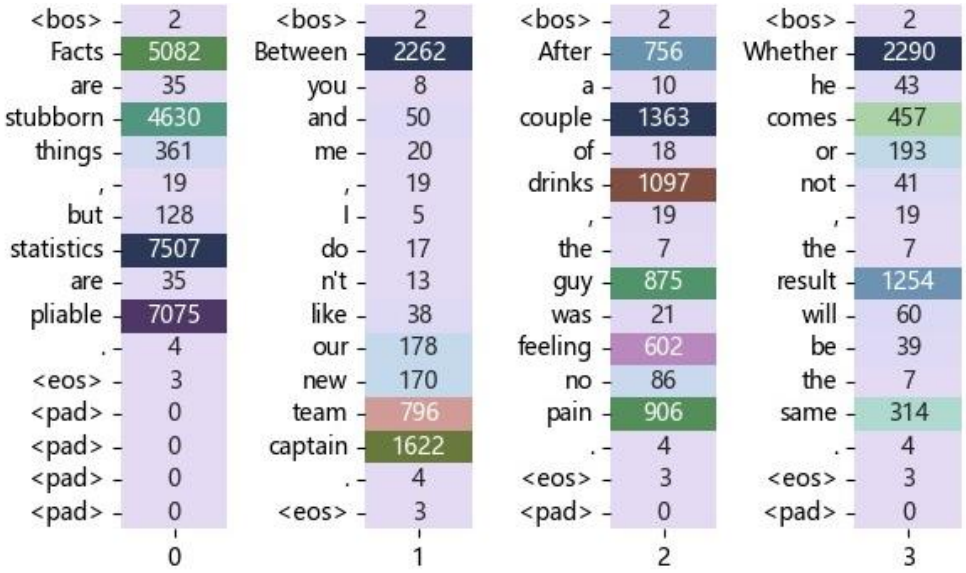

Figure 8. Batch word index vector

Among them, the length of the sentence with index 1 is 16 (including the start and end symbols), so it is enough to fill the current batch with 16 as the maximum length, or you can add the maximum length max_len set by human in the data preprocessing, when the length of the longest sentence in the batch is greater than max_len, the longest sentence will be sliced according to max_len, and at this time, you don't need to insert the termination symbols. When the longest sentence in the batch is longer than max_len, the longest sentence will be sliced according to max_len, in which case no termination symbol needs to be inserted, and other sentences can be filled.

After the end of model training, when testing the model, it is necessary to machine translate the word vectors of the source text, so there is some difference between the translation inference and the Transformer forward computation, i.e., the greedy algorithm is adopted for the final inference result to translate the sentences. In fact, it just selects the position with the highest probability in the output probability matrix as the word index, called greedy decode, and the pseudo-code is as follows.

Algorithm 1. Greedy Decode

Input： source token vector $\boldsymbol{src}$

1 Get the outputs of Encoder inference $\boldsymbol{src}$**:** $\boldsymbol{hs}$**;**

2 Initialize prediction sequence $\hat{\boldsymbol{y}} = [$"<BOS>"**];**

3 for i = 1 to target max length do

4 generate square subsequent mask $\boldsymbol{m} = [\boldsymbol{m_{ij}}], \begin{cases} \boldsymbol{m_{ij}} = \boldsymbol{1}, \boldsymbol{i} \leq \boldsymbol{j} \\ \boldsymbol{m_{ij}} = \boldsymbol{0}, \boldsymbol{i} > \boldsymbol{j} \end{cases}$**;**

5 Use $\hat{\boldsymbol{y}}, \boldsymbol{hs}, \boldsymbol{m}$ for Decoder inference and get the probability outputs: $\boldsymbol{prob} \in \boldsymbol{R}^{(\boldsymbol{i+1})\times \boldsymbol{T}_{\mathbf{vocab}}}$;

6 Get the **index** of maximum probability position of $\boldsymbol{T}_{\mathbf{vocab}}$ dimension at position $\boldsymbol{i+1}$;

7 Let the **index** obtained in the previous step be added to the end of $\hat{\boldsymbol{y}} = [<\text{BOS}>, \mathbf{word_1}]$;

8 **if** $\mathbf{word}_{\boldsymbol{i}}$ is "<EOS>" **do**

9 **break**;

10 **return** $\hat{\boldsymbol{y}} = [<\mathbf{BOS}>, \mathbf{word_1}, \cdots, \boldsymbol{word_T}]$;

C. Optimizer

To restore the original Transformer experiment, the same Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^{-9}$ was taken.

D. Results

Experiments were conducted on three bilingual sentence pair data in the selected bilingual sentence pair dataset of the Tatoeba Project, with training hardware of 13th Gen Intel® Core™ i7-13700K, NVIDIA GeForce RTX 2060 SUPER 8GB, and thus limited parameters that could be set as in Table 4.

Table 4. Hyperparameter details.

Hyperparameter	**Value**
max_len	40
N	6
h	8
d_{model}	512
d_{ff}	2048
dropout	0.1
lr	0.0001

where N denotes the number of blocks of Encoder and Decoder, which is uniformly set to 6. h denotes the number of heads in the MHA. dropout denotes its percentage of culling in Add & Norm, i.e., 10% of the portion is randomly selected not to be added in Residual Add. lr denotes the initial learning rate of Adam optimizer. Using the above hyper-parameters, the dataset is trained for 100 cycles (batch size is 64) using three bilingual sentences, and the optimal results on Validation are obtained in Table 5.

Table 5. Optimal results of the training process.

Language	BLEU/%	Accuracy/%
English-to-Deutsch	49.67	78.90
English-to-French	52.07	80.38
English-to-Chinese	31.25	65.95

According to the experimental results, English to French has the best BLEU score of 52.07, followed by English to Deutsch with 49.67, and English to Chinese with 31.25. The highest BLEU score for French may be partly due to the following reasons The highest BLEU score for French may be partly due to the fact that the number of dictionaries in the target language is smaller than that of Deutsch, i.e., there are fewer unfamiliar words in the dictionary. While Chinese has less data compared to Deutsch and French, and the lexicon of Chinese is more specific, English to Chinese is a word-to-character model, so it may need to formulate a more detailed training strategy if it hopes that English-to-Chinese will converge. The loss functions and BLEU scores of these three languages are shown in Fig. 9.

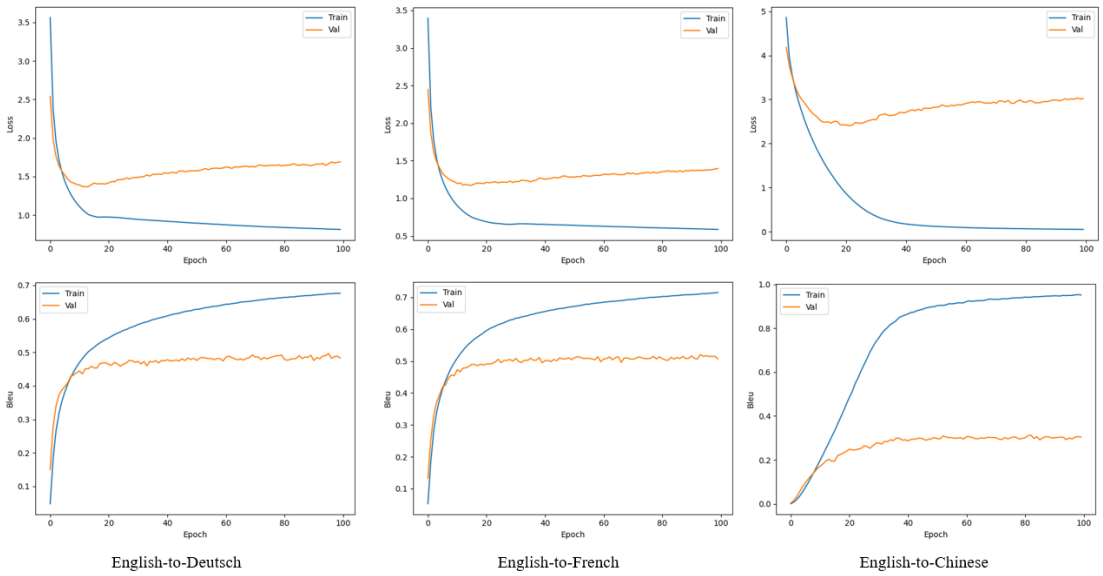

Figure 9. Loss function and BLEU score changes.

It can be found that Deutsch and French have converged around 20 epochs, while Chinese needs around 40 epochs to converge, and subsequently there has been overfitting, with a gradual increase in the loss of validation. Therefore, for a special language like Chinese, more work needs to be done.

E. Advantages and Disadvantages

The reproduction and experimentation of Transformer model in this paper can reflect the importance of this model in the field of machine translation to a certain extent, pioneering the use of neural networks

without CNN architectures, with better performance and robustness than the same type of CNN architectures, and more reliable inference results. In Transformer's core MHA, different inputs are used to form a kind of "attention" weighting for word vectors, so that the input word vectors can amplify the feature values to a certain extent, and the sentence can appear similar to the contextual relationship of positional encoding, so the model pays more attention to the global and coherence of the whole sentence, instead of focusing on some parts like CNNs. It no longer focuses on partial features as CNN does.

Since the Transformer does not use CNN and other structures, the number of parameters in the model training is huge, and the training time is inevitable. However, in the model inference process, due to its huge structure even the base architecture inference speed is still slow. Due to the limitation of the equipment, it cannot perfectly reproduce the training process of the Transformer, i.e., it only reproduce the base, so it hopes to train the model on multi-GPU servers in order to reproduce the experiment closer to the original text. Moreover, it did not get satisfactory results in the English-to-Chinese experiments, mostly due to the word-to-character model, which leads to unsatisfactory recognition results.

IV. Conclusions

The overall model reproduction of Transformer uses PyTorch deep learning framework, so the description of vector shape is based on PyTorch style. The model training and validation part is a single-computer, single-card training method, which can only train and validate the base architecture due to the limitation of the equipment, and the dataset is a bilingual dataset carefully selected from the Tatoeba Project, in which English-to-French, English-to-Deutsch, English-to-French, and English-to-Deutsch are selected with reference to the original article. Deutsch, English-to-French, English-to-Deutsch, and English-to-Chinese were added for the purpose of exploring and researching the machine translation task from English to Chinese. The experimental results prove that even Transformer is not able to solve the word-to-character problem well, and more detailed data processing or model training programs need to be customized.

Reference

[1] T. Poibeau, “The 1966 ALPAC report and its consequences,” in Machine Translation, MIT Press, pp.75-89, 2017

[2] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and et al., “The mathematics of statistical machine translation: Parameter estimation,” *Computational Linguistics,* vol. 19, no. 2, pp. 263-311, 1993.

[3] P. Koehn, F. J. Och, and D. Marcu, “Statistical Phrase-Based Translation,” *In Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 127-133, 2003.

[4] F. J. Och, "Minimum Error Rate Training for Statistical Machine Translation," *In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, ACL, 2003.

[5] N. Kalchbrenner and P. Blunsom, “Recurrent Continuous Translation Models,” *In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1700-1709, 2013.

[6] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14)*, pp. 3104-3112, 2014.

[7] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *presented at 3rd International Conference on Learning Representations*, 2015.

[8] A. Vaswani, et al., "Attention is all you need," Advances in neural information processing systems, 30, 2017.

[9] K. He, X. Zhang, S. Ren, and et al., “Deep residual learning for image recognition,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778, 2017.

[10] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.

[11] K. Papineni, S. Roukos, T. Ward, and et al., “BLEU: a method for automatic evaluation of machine translation,” *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. pp. 311-318, 2002.