Interpreting Positional Information in Perspective of Word Order

Xilong Zhang[1], Ruochen Liu[1]*, Jin Liu[1], Xuefeng Liang[1,2]
[1]School of Artificial Intelligence, Xidian University, Xi'an, China
[2]Guangzhou Institute of Technology, Xidian University, Guangzhou, China
{xilongzhang,liujin_1}@stu.xidian.edu.cn
{ruochenliu,xliang}@xidian.edu.cn

Abstract

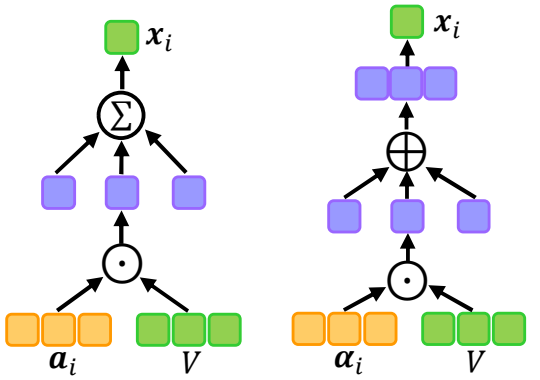
The attention mechanism is a powerful and effective method utilized in natural language processing. However, it has been observed that this method is insensitive to positional information. Although several studies have attempted to improve positional encoding and investigate the influence of word order perturbation, it remains unclear how positional encoding impacts NLP models from the perspective of word order. In this paper, we aim to shed light on this problem by analyzing the working mechanism of the attention module and investigating the root cause of its inability to encode positional information. Our hypothesis is that the insensitivity can be attributed to the weight sum operation utilized in the attention module. To verify this hypothesis, we propose a novel weight concatenation operation and evaluate its efficacy in neural machine translation tasks. Our enhanced experimental results not only reveal that the proposed operation can effectively encode positional information but also confirm our hypothesis.

1 Introduction

In recent years, attention mechanism (Bahdanau et al., 2015; Luong et al., 2015; Lin et al., 2017) has made remarkable progress on a wide range of natural language processing (NLP) tasks, such as machine translation (Vaswani et al., 2017; Radford et al., 2018), question answering and language inference (Devlin et al., 2019). It updates the contextual representation of a word by aggregating information from other words in the context, enabling it capture the content-based relevance of any two words, regardless of the distance between them.

In contrast to widely used recurrent neural networks (RNNs) or convolutional neural networks (CNNs), attention mechanism suffers from a notable disadvantage, i.e. its permutation invariance (Yun et al., 2020). This limitation results in its failure to distinguish sequences with different word orders. As a result, numerous studies have focused on encoding positional information of a sequence for the attention mechanism, such as learnable fixed-length positional encoding (Gehring et al., 2017), sinusoidal positional encoding (Vaswani et al., 2017) and relative positional encoding (Shaw et al., 2018). Typically, these methods assign positional embeddings to words at different positions through vector addition in various indexing manners, such as absolutely indexing and relatively indexing. However, it remains unclear how the attention mechanism incorporates positional information into a sequence through the addition of positional embeddings to word embeddings.

Besides, previous studies have probed the influence of word order perturbation on natural lan-



(a) Weight Sum (b) Weight Concatenation

Figure 1: Illustration of calculating the contextual representation of $\mathbf{x}_i$ with (a) weight sum (b) weight concatenation in Transformer's attention layers. $\mathbf{a}_i$ is the attention score vector of $\mathbf{x}_i$ over other words and $\mathbf{V}$ is the value matrix. $\odot$, $\sum$, $\oplus$ are scalar multiplication, vector sum and vector concatenation respectively. The weight concatenation combines weighted elements in accordance with the word order and linearly transforms the output to its original dimensions. This process serves to establish positional dependencies throughout a given sequence.

*Corresponding author.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 9600–9613

guage understanding (NLU) tasks (Abdou et al., 2022; Pham et al., 2021; Clouâtre et al., 2021) to determine whether these tasks are sensitive to word order. The findings indicate that some NLU tasks do require word order information although others do not. It should also be noted that word order is particularly important for natural language generating (NLG) tasks, such as machine translation. This is because metrics used to evaluate generated results, such as BLEU (Papineni et al., 2002), are sensitive to word order. Therefore, word order is a critical aspect of natural language processing.

In linguistics, grammars can be viewed as explanations for rules or principles of word order (Hawkins, 1990) to some extent. Given a sentence of arbitrary length, a set of positional embeddings can be seen as a group of distributed vectors that represent *temporal information* (Elman, 1990), namely *word order* in the context of language processing. This insight motivates us to correlate positional information with word order, interpret positional information in perspective of word order, and comprehend how word order impacts attention models.

In this paper, we first examine how the attention module of Transformer works and discover that its weight sum operation (Figure 1(a)) is the reason it struggles to encode positional information. Furthermore, this sheds light on how the word order impacts on NLP models, which means that the impact is achieved by its connection with positional information.

Based on this finding, we make further modifications to the attention module's working mechanism. Our goal is to implicitly encode positional information in attention module, which we posit is superior to explicit representation of positional information, as suggested by Elman (1990). To be more concrete, we devise a novel weight concatenation operation (Figure 1(b)) to calculate the final contextual representation, as an alternative to the weight sum operation (Figure 1(a)) in the attention module. To test the effectiveness of this approach, we evaluate the novel operation on the widely used, big neural machine translation datasets, including WMT 2014 English⇒German and WMT 2014 English⇒French. Our experimental results demonstrate that the proposed operation is capable of effectively encoding positional information for a sequence, leading to consistently improved performance and verifying our hypothesis.

2 Background

2.1 Attention Mechanism

In this section, we provide a brief introduction about the attention mechanism in Transformer (Vaswani et al., 2017). To simplify, we consider attention layers only with a single head rather than multiple heads. Also, let $s(\cdot)$ denote the softmax operator that performs softmax operation to each row of a matrix. By dropping the residual connection (He et al., 2016) and the layer normalization (Ba et al., 2016), the attention layer can be generally formed as:

$$\mathrm{Attn}(\mathbf{Q},\mathbf{K},\mathbf{V}) = s\left(\mathbf{Q}\mathbf{K}^\top/\sqrt{d_k}\right)\mathbf{V}\,, \quad (1)$$

$$\mathbf{Q},\mathbf{K},\mathbf{V} = \mathbf{X}\mathbf{W}_Q, \mathbf{X}\mathbf{W}_K, \mathbf{X}\mathbf{W}_V\,, \quad (2)$$

where $\mathbf{Q}\in\mathbb{R}^{L\times d_q}, \mathbf{K}\in\mathbb{R}^{L\times d_k}, \mathbf{V}\in\mathbb{R}^{L\times d_v}$ are packed queries, keys and values respectively, which are results of affine transformation on the input. $\mathbf{X}\in\mathbb{R}^{L\times d}$ is the embedding result or the hidden representation of previous layer with L being the sequence length and d being the model dimension. d_q, d_k, d_v are the dimension of queries, keys and values respectively, typically with $d_q = d_k = d_v$. $\mathbf{W}_Q\in\mathbb{R}^{d\times d_q}, \mathbf{W}_K\in\mathbb{R}^{d\times d_k}, \mathbf{W}_V\in\mathbb{R}^{d\times d_v}$ are trainable linear projections.

To clearly show the integration of positional encodings in Section 2.2, only consider the contextual result of a single word. Given the embedding result or hidden representation $X\in\mathbb{R}^{L\times d}$ as input, the i-th output of the attention mechanism, denoted as $\mathbf{o}_i$, can be calculated as:

$$\mathbf{o}_i = s\left(\mathbf{q}_i\mathbf{K}^\top/\sqrt{d_k}\right)\mathbf{V} = \sum_{j=1}^{L} a_{ij}\mathbf{v}_j\,, \quad (3)$$

where a_{ij} is the attention score of $\mathbf{q}_i$ over $\mathbf{k}_j$ and calculated as:

$$a_{ij} = \frac{\exp\left(\mathbf{q}_i\mathbf{k}_j^\top/\sqrt{d_k}\right)}{\sum_{j=1}^{L}\exp\left(\mathbf{q}_i\mathbf{k}_j^\top/\sqrt{d_k}\right)}\,. \quad (4)$$

For convenience, define the function $a(\mathbf{q}_i,\mathbf{K}) : \mathbb{R}^{d_q}\times\mathbb{R}^{L\times d_k}\to\mathbb{R}^L$ as:

$$a(\mathbf{q}_i,\mathbf{K}) := s\left(\mathbf{q}_i\mathbf{K}^\top/\sqrt{d_k}\right)\,. \quad (5)$$

2.2 Positional Encoding Mechanism

To remove the permutation invariance constraint (Yun et al., 2020), various positional encodings have been proposed and they are usually integrated into embedding layers (*embedding-level*) or attention layers (*attention-level*).

Embedding-Level Positional encodings at embedding level are added to the embedding results of a sequence, immediately following the embedding layers. They are usually a set of predefined or trainable vectors, indexed with absolute position numbers. After that, the contextual representation $\mathbf{o}_i$ can be represented as:

$$\mathbf{o}_i = a\left(\mathbf{x}_i + \mathbf{b}_i, \mathbf{X} + \mathbf{B}\right)\left(\mathbf{X} + \mathbf{B}\right)\mathbf{W}_V\,, \quad (6)$$

where $\mathbf{b}_i \in \mathbb{R}^d$ is the positional embedding for the i-th word, $\mathbf{B} \in \mathbb{R}^{L\times d}$ is the packed positional embeddings for the whole sequence.

Attention-Level Positional encodings at attention level are based on an insight into the attention mechanism. Since they are employed in attention layers, they can access to both the query and the key. This enables them to encode positional information in a more complex manner, such as indexing with the relative distance between the query and the key. Taking relative positional encoding (Shaw et al., 2018) as an example, the contextual representation $\mathbf{o}_i$ can be calculated as:

$$\mathbf{o}_i = \sum_{j=1}^{L} a_{ij}\left(\mathbf{v}_j + \mathbf{r}^v_{ij}\right)\,, \quad (7)$$

$$a_{ij} = a(\mathbf{q}_i, \mathbf{k}_j + \mathbf{r}^k_{ij})\,, \quad (8)$$

where $\mathbf{r}^k_{ij}, \mathbf{r}^v_{ij} \in \mathbb{R}^{d_k}$ are trainable positional encodings for the j-th key and j-th value respectively. These encodings are indexed based on the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$.

3 Analysis and Method

3.1 Correlation Between Positional Encoding and Word Order

In this section, we first demonstrate that the attention mechanism of Transformer is permutation invariant and analyse the reason. After that, we show how to correlate positional information with the word order of a sequence and perform positional encoding through preserving word order.

It is straightforward to obtain the permutation invariance of attention mechanism in Transformer from Equation (3). Given the input $\mathbf{X} \in \mathbb{R}^{L\times d}$ and the permutation matrix $\mathbf{P} \in \mathbb{R}^{L\times L}$, the permutation invariance constraint can be demonstrated as follows:

$$\begin{aligned}\mathbf{o}_i &= a\left(\mathbf{q}_i, \mathbf{P}\mathbf{X}\mathbf{W}_K\right)\left(\mathbf{P}\mathbf{X}\mathbf{W}_V\right)\\ &= a\left(\mathbf{q}_i, \mathbf{X}\mathbf{W}_K\right)\mathbf{P}^\top\mathbf{P}\mathbf{X}\mathbf{W}_V\\ &= a\left(\mathbf{q}_i, \mathbf{X}\mathbf{W}_K\right)\mathbf{X}\mathbf{W}_V\,,\end{aligned} \quad (9)$$

where $s\left(\mathbf{X}\mathbf{P}^\top\right) = s\left(\mathbf{X}\right)\mathbf{P}^\top$ and $\mathbf{P}^\top\mathbf{P} = \mathbf{I}$ are used. From Equation (9), it is obvious that the contextual representation $\mathbf{o}_i$ is invariant to whatever permutation transformations applied to the input.

However, the permutation invariance constraint does not generalize to RNNs and CNNs, which compute the contextual representation in a different manner. Therefore, we may focus on the way of computing the contextual representation in the attention mechanism. We speculate that the constraint is a result of the weight sum operation in Equation (3). In other words, the weight sum in the attention mechanism eliminates the positional distinctions among words of a sequence.

Hence, to encode positional information for the attention mechanism, we place emphasis on how to keep the word order of a sequence when calculating the contextual representations of these words. Following this way, we propose the weight concatenation operation as an alternative to the weight sum operation. We denote the weight concatenation of attention score $\mathbf{a}_i \in \mathbb{R}^L$ and value matrix $\mathbf{V} \in \mathbb{R}^{L\times d_v}$ as $\mathbf{a}_i \oplus \mathbf{V} : \mathbb{R}^L \times \mathbb{R}^{L\times d_v} \to \mathbb{R}^{Ld_v}$, which can be formally represented as:

$$\mathbf{a}_i \oplus \mathbf{V} = \left[a_{i1}\mathbf{v}_1 : a_{i2}\mathbf{v}_2 : \cdots : a_{iL}\mathbf{v}_L\right]\,, \quad (10)$$

where : represents the vector concatenation.

Then, the contextual representation $\mathbf{o}_i$ can be represented as:

$$\mathbf{o}_i = \left(\mathbf{a}_i \oplus \mathbf{V}\right)\mathbf{\Phi}\,, \quad (11)$$

where $\mathbf{\Phi} \in \mathbb{R}^{Ld_v\times d_v}$ is a linear projection matrix.

In Equation (11), the value vectors are first weighted by the corresponding attention scores and then concatenated, followed by the linear projection $\mathbf{\Phi}$ to reduce the dimension of the concatenated result to the original dimension. Since Equation (11) concatenates the weighted value vectors exactly according to the word order of the sequence, it is obvious that the attention mechanism is permutation variant now, which can be shown as:

$$\begin{aligned}\mathbf{o}_i &= \left(a\left(\mathbf{q}_i, \mathbf{P}\mathbf{X}\mathbf{W}_K\right) \oplus \left(\mathbf{P}\mathbf{X}\mathbf{W}_V\right)\right)\mathbf{\Phi}\\ &= \left(a\left(\mathbf{q}_i, \mathbf{X}\mathbf{W}_K\right)\mathbf{P}^\top \oplus \mathbf{P}\left(\mathbf{X}\mathbf{W}_V\right)\right)\mathbf{\Phi}\\ &= \left(a\left(\mathbf{q}_i, \mathbf{X}\mathbf{W}_K\right)\mathbf{P}^\top \oplus \left(\mathbf{X}\mathbf{W}_V\right)\mathbf{P}^\top\right)\mathbf{\Phi}\end{aligned} \quad (12)$$

$$\neq \left(a\left(\mathbf{q}_i, \mathbf{X}\mathbf{W}_K\right) \oplus \left(\mathbf{X}\mathbf{W}_V\right)\right)\mathbf{\Phi}\,, \quad (13)$$

where $\mathbf{P}\mathbf{X} = \mathbf{X}\mathbf{P}^\top$ is used. Equation (12) means that the permutation transformation applied to the

input gives rise to a change in the order of both the attention scores and value vectors. Then the attention mechanism can not be permutation invariant any longer (Equation (13)).

Therefore, the weight concatenation implicitly encodes positional information for attention mechanism. To sum up, the retention of the word order information prevents the positional information from being lost during computing the contextual representation, which correlates positional encoding with the word order from a linguistic perspective.

3.2 Positional Kernels

As shown in Section 3.1, the weight concatenation is proposed to eliminate the permutation invariance constraint. However, practical space complexity concerns arise with the weight concatenation since it requires extending the dimension of each word vector from d_v to Ld_v, given a sequence of length L. Besides, we want to circumvent increasing time complexity and utilize highly efficient parallel matrix calculation on GPU. Fortunately, both of these issues can be resolved through factoring the matrix $\mathbf{\Phi}$ as:

$$\mathbf{o}_i = (\mathbf{a}_i \oplus \mathbf{V})\,\mathbf{\Phi} = \sum_{j=1}^{L} a_{ij}\mathbf{v}_j\boldsymbol{\phi}_j \tag{14}$$

$$= \sum_{j=1}^{L} a_{ij}\left(\mathbf{v}_j\boldsymbol{\phi}_j\right) = \mathbf{a}_i\left(\mathbf{V}\otimes\mathbf{\Phi}\right), \tag{15}$$

where $\boldsymbol{\phi}_j \in \mathbb{R}^{d_v\times d_v}, j\in\{1,\ldots,L\}$ is a subblock of $\mathbf{\Phi}$. $\left(\mathbf{v}_j\boldsymbol{\phi}_j\right)$ means $\mathbf{v}_j$ is multiplied by $\boldsymbol{\phi}_j$ first. $\otimes$ is the batched matrix multiplication that broadcasts along the sequence length dimension of $\mathbf{V}$ and $\mathbf{\Phi}$. $\mathbf{V}\otimes\mathbf{\Phi}$ can be defined as:

$$\left[\mathbf{V}\otimes\mathbf{\Phi}\right]_i := \mathbf{v}_i\boldsymbol{\phi}_i\,, \tag{16}$$

where $\mathbf{v}_i \in \mathbb{R}^{d_v}$ and $\boldsymbol{\phi}_i \in \mathbb{R}^{d_v\times d_v}$.

In Equation (14), we reshape the matrix $\mathbf{\Phi}\in\mathbb{R}^{Ld_v\times d_v}$ as a tensor $\mathbf{\Phi}\in\mathbb{R}^{L\times d_v\times d_v}$, namely splitting $\mathbf{\Phi}$ into a set of subblock $\boldsymbol{\phi}_j\in\mathbb{R}^{d_v\times d_v}$. Multiplying $\mathbf{v}_j$ by $\boldsymbol{\phi}_j$ first in Equation (15) makes it possible to apply efficient parallel matrix multiplication through tensor reshaping and broadcasting. The highly efficient matrix form can be represented as:

$$\begin{aligned}&\mathrm{Attn}\left(\mathbf{Q},\mathbf{K},\mathbf{V}\right)\\ &= s\left(\mathbf{Q}\mathbf{K}^\top/\sqrt{d_k}\right)\left(\mathbf{V}\otimes\mathbf{\Phi}\right).\end{aligned} \tag{17}$$

It is amazing that the attention mechanisms with matrix blocks $\boldsymbol{\phi}_j$ in Equation (17) are highly similar to the existing positional encodings in Equations (6) and (7). Both are position-specific and capable of encoding positional information, however, the way that they compute the contextual representation is totally different. Therefore, $\boldsymbol{\phi}_j$ in Equation (15) can be regarded as a novel positional encoding strategy and we dub it as "*positional kernel*".

3.3 Positional Encoding Network

In this section, we integrate the proposed positional kernels into Transformer and demonstrate the existing positional encodings as special cases of positional kernels, then develop two types of positional encoding networks.

Recall that positional encodings (Vaswani et al., 2017; Shaw et al., 2018) are typically integrated into the model through adding positional embeddings to hidden representations, either in the embedding layers or attention layers. Omitting trivial distinctions, such as indexing manners, the result of incorporating positional encodings can be uniformly represented as:

$$\mathbf{H}_{\mathrm{pe}} = \mathbf{X} + \mathbf{M}\,, \tag{18}$$

where $\mathbf{X}\in\mathbb{R}^{L\times d}$ is the input, $\mathbf{M}\in\mathbb{R}^{L\times d}$ is the positional encoding matrix and $\mathbf{H}_{\mathrm{pe}}$ is the hidden representation after applying the positional embedding $\mathbf{M}$.

Likewise, we apply the proposed positional kernel tensor $\mathbf{\Phi}\in\mathbb{R}^{L\times d\times d}$ to the embedding layers or attention layers in the following manner:

$$\mathbf{H}_{\mathrm{pk}} = \mathbf{X}\otimes\mathbf{\Phi}\,. \tag{19}$$

It can be proved that existing positional encodings in Equation (18) can be regarded as special cases of the method with positional kernels in Equation (19), which is established as the following theorem:

Theorem 3.1. *Let $\epsilon > 0$, then for any given $\mathbf{X}\in\mathbb{R}^{L\times d}$ and $\mathbf{M}\in\mathbb{R}^{L\times d}$, there exists a $\mathbf{\Phi}^*\in\mathbb{R}^{L\times d\times d}$ such that, for $f\left(\mathbf{X}\right) = \mathbf{X}\otimes\mathbf{\Phi}^*$ and $g\left(\mathbf{X}\right) = \mathbf{X}+\mathbf{M}$, $\left\|f\left(\mathbf{X}\right) - g\left(\mathbf{X}\right)\right\|_2 \le \epsilon$.*

Remark 3.2. We provide the detailed proof of Theorem 3.1 in Appendix A.1. To sum up, given the input $\mathbf{X}\in\mathbb{R}^{L\times d}$ and positional encodings $\mathbf{M}\in\mathbb{R}^{L\times d}$, there exists a set of corresponding positional kernels $\boldsymbol{\phi}_j\in\mathbb{R}^{d\times d}, j\in\{1,\ldots,L\}$,

such that $f(\mathbf{X}) = \mathbf{X} \otimes \mathbf{\Phi}$ can closely approximate $g(\mathbf{X}) = \mathbf{X} + \mathbf{M}$. Therefore, the existing positional encodings (Equation (18)) can be regarded as special cases of the proposed positional kernels (Equation (19)).

To integrate the positional kernel into Transformer and make it work well, we further develop the corresponding **Pos**itional encoding **Net**work (**PosNet**) at attention level and embedding level.

Attention-Level In a manner similar to Shaw et al. (2018), we apply the positional kernel to attention layers to preserve positional information. Given the input $\mathbf{X} \in \mathbb{R}^{L \times d}$ and positional kernel tensor $\mathbf{\Phi} \in \mathbb{R}^{L \times d_k \times d_k}$, the attention mechanism with PosNet can be represented as follows:

$$\begin{aligned} &\mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \\ &= s\left(\mathbf{Q}\mathbf{K}^\top / \sqrt{d_k}\right) \mathrm{PosNet}(\mathbf{V}; \mathbf{\Phi}) , \end{aligned} \tag{20}$$

where $\mathrm{PosNet}(\cdot)$ is the function of the proposed PosNet, which can be formed as:

$$\mathrm{PosNet}(\mathbf{X}; \mathbf{\Phi}) = \sigma(\mathbf{X} \otimes \mathbf{\Phi}) , \tag{21}$$

where $\sigma(\cdot)$ is the non-linear activation function.

Embedding-Level The equivalent implementation of the weight concatenation in Equation (15) makes it possible to match $\mathbf{v}_j$ with ϕ_j, which enlightens us to apply the proposed positional kernel to the embedding layer. Besides, inspired by the feed forward network (FFN) in Transformer, we also use two linear projections to transform the input to a desired low-dimensional representation space (Yun et al., 2020). Given the embedding result $\mathbf{X} \in \mathbb{R}^{L \times d}$ and the positional kernel tensor $\mathbf{\Phi}$, the output of PosNet can be formally represented as:

$$\mathrm{PosNet}(\mathbf{X}; \mathbf{\Phi}) = \sigma(\mathbf{X}\mathbf{W}_1 \otimes \mathbf{\Phi})\mathbf{W}_2 , \tag{22}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d \times d_1}$ and $\mathbf{W}_2 \in \mathbb{R}^{d_2 \times d}$ are trainable linear projections. $\mathbf{\Phi} \in \mathbb{R}^{L \times d_1 \times d_2}$ and we use $d_1 = d_2 < d$ to alleviate the increase of parameter counts.

Residual Connection and Dropout The identity term in the proof of Theorem 3.1 (Appendix A.1) indicates that it is also helpful to retain the original information, i.e. $\mathbf{X}$ in Equations (21) and (22). This form exactly corresponds to the residual connection (He et al., 2016), which can facilitate the training of the network. Hence, the residual connection is employed to wrap PosNet modules, both at embedding level and attention level. Then, Equations (21) and (22) can be reformed as:

$$\mathrm{PosNet}(\mathbf{X}; \mathbf{\Phi}) = \sigma(\mathbf{X} \otimes \mathbf{\Phi}) + \mathbf{X} , \tag{23}$$

$$\mathrm{PosNet}(\mathbf{X}; \mathbf{\Phi}) = \sigma(\mathbf{X}\mathbf{W}_1 \otimes \mathbf{\Phi})\mathbf{W}_2 + \mathbf{X} . \tag{24}$$

Apart from the residual connection, the dropout (Srivastava et al., 2014) is also applied to the output of PosNet modules as a regularizer and immediately followed by the residual connection, which is also inspired by the delicate design of Transformer. The dropout rate of the PosNet is independent from that of Transformer. The illustration of applying the PosNet to Transformer is available in Appendix A.3.

4 Experiments

4.1 Experimental Setup

Datasets The proposed methods are evaluated on the widely used machine translation benchmarks, including WMT 2014 English⇒German (En-De) and WMT 2014 English⇒French (En-Fr), with about 4.43 million and 35.76 million parallel sentence pairs for training respectively. For both language pairs, the newstest2013 dataset is used as the validation set and the newstest2014 dataset as the test set.

Model In all experiments, English, German and French datasets are tokenized via the scripts in Moses (Koehn et al., 2007). For all language pairs, the byte pair encoding (BPE) compression algorithm (Sennrich et al., 2016) is employed to reduce the size of the vocabulary and enable the model to translate rare and unknown tokens. The vocabulary size is set to 40,000 for both En-De and En-Fr translation tasks. The Adam algorithm (Kingma and Ba, 2015) is used as the optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$. The label smoothing (Szegedy et al., 2016) is adopted as well, with $\epsilon_{ls} = 0.1$.

We implement all methods on top of FAIRSEQ (Ott et al., 2019) and follow its preprocessing and training instructions on neural machine translation. We take advantage of efficient half-precision floating point (FP16) arithmetic for both the training and evaluation stage. To maintain approximately 25,000 source tokens and 25,000 target tokens in each training batch as Vaswani et al. (2017), we limit the number of

System	Approach	En-De		En-Fr		#Params
		BLEU	ChrF++	BLEU	ChrF++	(En-De)/M
reported	SAPE	27.3	-	38.1	-	-
	RPE	26.8	-	38.7	-	-
this work	w/o PE	13.49	43.99	22.59	51.45	63.08
	SAPE	27.48	53.97	40.46	62.09	63.08
	LAPE	24.34	52.15	39.75	61.50	64.13
	RPE	27.07	53.65	40.61	62.26	63.37
	Adaptive-T5	27.02	53.53	40.36	62.08	63.08
	PosNet-Attn	23.12	49.91	36.14	59.17	64.11
	PosNet-Embed	**27.68**	**54.05**	**40.81**++	**62.33**++	66.49

Table 1: Machine translation results of Transformer base model, in terms of BLEU and ChrF++ for WMT 2014 En-De and En-Fr on newstest2014 test set. In Tables 1 and 2, '++'/'+' after scores indicates that the proposed method is significantly better than the corresponding baseline method (SAPE) at significance level $p < 0.05/0.10$.

Approach	En-De		En-Fr	
	BLEU	ChrF++	BLEU	ChrF++
SAPE	28.4	-	41.0	-
RPE	29.2	-	41.5	-
SAPE	28.92	54.86	42.45	63.59
RPE	28.77	54.69	42.25	62.41
PosNet	**29.23**++	**55.12**++	**42.66**+	**63.70**

Table 2: Machine translation results of Transformer big models, in terms of BLEU and ChrF++ for WMT 2014 En-De and En-Fr on newstest2014 test set.

tokens in each batch to 4096 and accumulate the gradients of 8 batches for each update (Ott et al., 2018). We re-implement all compared methods and keep experimental sets same for these methods to ensure a consistent and fair comparison. We run all experiments with NVIDIA RTX3090 24 GB GPU cards.

Evaluation We employ beam search with a beam size of 4 and a length penalty $a = 0.6$. We evaluate the performance of models with both the common used BLEU (Papineni et al., 2002) and recently proposed ChrF++ (Popović, 2017; Marie et al., 2021), which shows better correlation with human judgments. Besides, we perform statistical significance test with bootstrap resampling (Koehn et al., 2003).

Compared Methods Compared positional encoding methods mainly include

- *w/o PE*: without any positional encoding, to provide another reference of positional encoding ability besides the baseline,
- *SAPE*: sinusoidal absolute positional encoding (Vaswani et al., 2017),
- *LAPE*: learnable absolute positional encoding, to evaluate the effect of trainable positional encoding,
- *RPE*: relative positional encoding (Shaw et al., 2018),
- *Adaptive-T5*: adaptive version of T5's relative positional encoding (Wu et al., 2021),
- *PosNet-Attn*: attention-level PosNet (Section 3.3),
- *PosNet-Embed*: embedding-level PosNet (Section 3.3).

We provide the relevant *codes and scripts*[1] of our experiments, which is helpful to reproducibility. Besides, the *datasets*[2] used in our experiments are all publicly available.

4.2 Results

Experimental results are shown in Tables 1 and 2 and Figure 2. We make the following observations:

Strong Baseline To isolate the impact of different positional encoding strategies from any other unrelated factors, such as the underlying implementation detail and experimental configuration, we re-implement the baseline model (SAPE) and other compared methods. In comparison to the reported result (Vaswani et al., 2017), the reproduced baseline result achieves better performance, especially

[1] https://github.com/vesterchang/interpret_positional_encoding

[2] https://www.statmt.org/wmt14/index.html

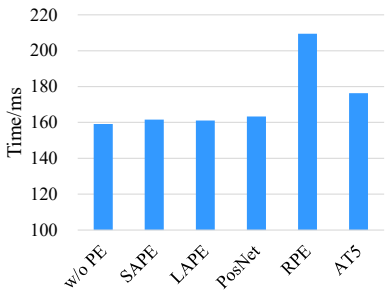


Figure 2: Time consumed by Transformer base model with different positional encoding methods to forward the same manual batch 10 times. *AT5* is short for *Adaptive-T5*.

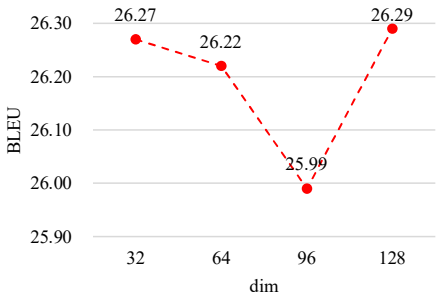


Figure 3: Effect of the dimension d_p of embedding-level PosNet, evaluated on the validation dataset.

on the WMT 2014 En-Fr translation task. This indicates that the system in this work is a strong baseline.

Positional Information is Critical To assess the significance of positional information, we conduct experiments on Transformer without any positional encodings (w/o PE). As a result, the performance decreases drastically. This suggests that positional information is critical for Transformer, which relies solely on the position insensitive attention mechanism.

Performance of PosNet The proposed approaches achieve competitive performance or statistically significant improvement over the baseline. Specifically, PosNet-Attn outperforms the w/o PE, indicating that it effectively encodes positional information, although it is worse than the baseline. However, the PosNet-Embed achieves better performance in terms of both BLEU and ChrF++ than the baseline on both of WMT 2014 En-De and En-Fr translation tasks. This demonstrates the superiority of the PosNet-Embed in encoding positional information.

Comparison with Other Methods In order to make a direct and fair comparison with other positional encoding methods (LAPE and RPE), we re-implement them and evaluate them with new metric, i.e. ChrF++. For RPE, we follow the configuration in Shaw et al. (2018). As shown in Table 1, both LAPE and RPE are worse than PosNet-Embed on both BLEU and ChrF++, which demonstrates that the PosNet-Embed encodes the positional information better again.

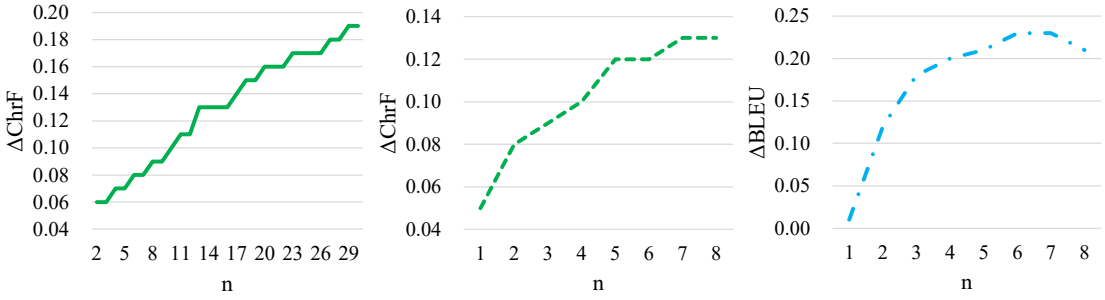
Computation Complexity of Different Methods To evaluate the computation complexity of different positional encoding methods, we manually construct a batch of samples and only perform the forward stage with Transformer base model since the model with different positional encoding methods would perform different numbers of decoding steps in practical translation scenario. We perform the forward stage with the same batch 10 times and record the time consumed by the model. As shown in Figure 2, all methods exhibit similar time complexity, with the exception of the RPE and AT5, which is obviously more time-consuming.

Furthermore, we conduct an analysis of the GPU memory consumption of the Transformer base model with different methods on the WMT 2014 En-De and En-Fr tasks. This is done to determine their relative space complexity. The results of this analysis can be found in Appendix A.2, which demonstrate that all methods exhibit comparable space complexity, except for RPE, which consumes more GPU memory on the WMT 2014 En-Fr task.

Scaling to Transformer Big We also implement these methods in the Transformer big model to observe the performance when scaling to big models. As shown in Table 2, the reproduced SAPE and RPE achieve competitive or better results in terms of BLEU compared with their reported results (Vaswani et al., 2017; Shaw et al., 2018). Besides, the PosNet-Embed achieves better performance than other methods, especially its statistically significant improvement on most results with significance level 0.05/0.10.

5 Discussion

Effect of Positional kernel's dimension As stated previously in Equation (22), the positional kernel tensor $\mathbf{\Phi}$ has the shape of $L \times d_1 \times d_2$. In practice, we use $d_1 = d_2 = d_p$. To further investigate the effect of the dimension d_p, we conduct experiments on PosNet-Embed with a series of varying d_p and evaluate the performance of dif-



(a) n-gram chr F2 + 2-gram word F2 (b) 6-gram chr F2 + n-gram word F2 (c) n-gram word precision

Figure 4: Performance gap of Transformer base model with the proposed method over the baseline method on WMT14 En-De with different character or word n-grams in (a) (b) ChrF and (c) BLEU.

ferent d_p with the BLEU score. Figure 3 shows the result of ablation experiment on the validation dataset, i.e. the newstest2013 dataset. Notably, $d_p = 96$ has a detrimental effect on the performance of PosNet-Embed and $d_p = 128$ with the best performance. Thereby, we conduct the experiments on the PosNet-Embed in Table 1 with $d_1 = d_2 = 128$. Besides, we use $d_1 = d_2 = 256$ for the Transformer big model, which is just twice the size of the base model.

Evaluation with different n-grams The default evaluation setup is 6-gram character F2 and 2-gram word F2 for ChrF++, 4-gram word precision for BLEU. However, since the PosNet preserves positional information through concatenating the weighted value vectors, it is intriguing what the performance is when evaluating with coarser-grain unit. Therefore, to observe the performance superiority of the proposed method on coarser grain, we evaluate the performance gap of PosNet-Embed over SAPE on WMT14 En-De with different n-grams. Specifically, we test the ChrF with n-gram character F2 and 2-gram word F2 ($1 \leq n \leq 30$), the ChrF with 6-gram character F2 and n-gram word F2 ($1 \leq n \leq 8$), and the BLEU with n-gram word precision ($1 \leq n \leq 8$) respectively. As shown in Figure 4, it is amazing that the performance superiority of the PosNet-Embed over the SAPE consistently enlarges as the evaluating grain becomes coarser, i.e. bigger n-gram character or word. We speculate that this is attributable to the weight concatenation of PosNet since the concatenation operation promotes the model to capture the whole context precisely according to the word order of a sequence, which preserves the coarser-grained feature.

6 Related Work

Since Transformer relies solely on the attention mechanism, it is position-insensitive to sequences with different word orders in the absence of positional encodings. Many studies attempt to utilize various positional encodings to inject the sequence's positional information into the model. Typically, embedding-level positional encodings (Vaswani et al., 2017; Devlin et al., 2019; Gehring et al., 2017) build positional dependencies via adding positional encodings to embedding results. In addition, attention-level positional encodings (Shaw et al., 2018; Raffel et al., 2020; Huang et al., 2020; Wu et al., 2021) usually capture positional information based on an analysis of attention mechanisms and leverage delicate indexing manners, such as the relative distance to the querying word. However, it is unknown why the attention mechanism of the Transformer is position agnostic and what the working principle behind these positional encodings is.

There has been increasing interest in understanding and explaining how these positional encodings construct positional dependencies for a sequence. The proof in Yun et al. (2020) suggests that the key of eliminating Transformer's permutation invariance is to quantify the embedding results to distinct intervals, which ensures a one-to-one mapping. Wu et al. (2021) analysed and improved the scalar relative positional encoding of T5 (Raffel et al., 2020) from a probabilistic perspective and took it as a prior distribution. The difference is that, in this work, we associate the positional information with the word order of a sequence, providing a novel linguistic perspective on positional encoding. Furthermore, we propose a new and effective positional encoding mechanism. We perform the

comparison of different positional encoding methods and the results are in Tables 1 and 2 except that the theoretical positional encoding scheme in the proof of Yun et al. (2020) doesn't work well in practice and thus are not reported.

7 Conclusion

Positional encoding is a critical issue for Transformer-like models. However, it has not been explored how positional encoding builds positional dependencies for a sequence. In this paper, we analyse the working manner of the attention module and find that its weight sum operation leads to the failure to encode positional information. Then, we modify the attention module with a proposed novel weight concatenation operation, following the guideline of retaining positional information according to the word order of a sentence. The modification correlates a sequence's word order with positional information, thus shedding light on how the word order impacts on NLP models. Competitive experimental results substantiate our hypothesis.

Limitations

In this paper, we present a novel approach to remove the permutation invariance of the attention module. Specifically, we propose a weight concatenation operation that exactly follows the word order of a sentence, leading to an increase in dimensionality and the introduction of affine transformations aimed at reducing it. Hence, the effect of increased parameter counts cannot be well isolated. While our preliminary experiments show that an increase in the number of parameter counts does not necessarily enhance the experimental results, we acknowledge the increased complexity resulting from direct concatenation and, thus, have utilized the equivalent form of the proposed operation in practice. In the future, we aim to explore alternative operations that implicitly encode positional information based on word order, without resorting to affine transformations, to replace the weight sum operation of the attention module.

Acknowledgments

This work was supported by the Provincial Natural Science Foundation of Shaanxi of China (No. 2019JZ-26). We acknowledge all the anonymous reviewers for their valuable comments and suggestions.

References

Mostafa Abdou, Vinit Ravishankar, Artur Kulmizev, and Anders Søgaard. 2022. Word order does matter and shuffled language models know it. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6907–6919, Dublin, Ireland. Association for Computational Linguistics.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Louis Clouâtre, Prasanna Parthasarathi, Amal Zouaq, and Sarath Chandar. 2021. Demystifying neural language models' insensitivity to word-order. *CoRR*, abs/2107.13955.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR.

John A. Hawkins. 1990. A parsing theory of word order universals. *Linguistic Inquiry*, 21(2):223–261.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhiheng Huang, Davis Liang, Peng Xu, and Bing Xiang. 2020. Improve transformer models with better relative position embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3327–3335, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Benjamin Marie, Atsushi Fujita, and Raphael Rubino. 2021. Scientific credibility of machine translation research: A meta-evaluation of 769 papers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7297–7306, Online. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation, Volume 1: Research Papers*, pages 1–9, Belgium, Brussels. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Thang Pham, Trung Bui, Long Mai, and Anh Nguyen. 2021. Out of order: How important is the sequential order of words in a sentence in natural language understanding tasks? In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1145–1160, Online. Association for Computational Linguistics.

Maja Popović. 2017. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Junshuang Wu, Richong Zhang, Yongyi Mao, and Junfan Chen. 2021. On scalar embedding of relative positions in attention models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14050–14057.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. 2020. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*.

Martin Zinkevich. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, pages 928–936. AAAI Press.

A Appendix

A.1 Proof of Theorem 3.1

Proof. To make the proof clear and concise, take into account only the individual word $\mathbf{x}_i$. The corresponding positional embedding and positional kernel are $\mathbf{m}_i$ and $\boldsymbol{\phi}_i$. Let's ignore the subscript i which indexes position for simplification if no confusion is possible and then $f(\cdot)$ and $g(\cdot)$ can be represented as:

$$f(\mathbf{x}) = \mathbf{x}\boldsymbol{\phi}\,, \tag{25}$$
$$g(\mathbf{x}) = \mathbf{x} + \mathbf{m}\,. \tag{26}$$

Then, to obtain the conclusion, we reconstruct the $\boldsymbol{\phi}$ as $\boldsymbol{\phi}^{'} + \mathbf{I}$, which exactly corresponds to the widespreadly used and effective residual connection (He et al., 2016). Then,

$$f(\mathbf{x}) = \mathbf{x}\left(\boldsymbol{\phi}^{'} + \mathbf{I}\right) = \mathbf{x}\boldsymbol{\phi}^{'} + \mathbf{x}\,. \tag{27}$$

The l_2 norm is

$$\|f(\mathbf{x}) - g(\mathbf{x})\|_2 = \left\|\mathbf{x}\boldsymbol{\phi}^{'} + \mathbf{x} - \mathbf{x} - \mathbf{m}\right\|_2 \tag{28}$$
$$= \left\|\mathbf{x}\boldsymbol{\phi}^{'} - \mathbf{m}\right\|_2 \tag{29}$$

Then, we only need to find a $\boldsymbol{\phi}^{'*}$ such that, for $\epsilon > 0$, $\left\|\mathbf{x}\boldsymbol{\phi}^{'*} - \mathbf{m}\right\|_2 \leq \epsilon$.

Let's take into account the following optimization problem:

$$\min_{\boldsymbol{\phi}^{'}} \left\|\mathbf{x}\boldsymbol{\phi}^{'} - \mathbf{m}\right\|_2 \tag{30}$$

Since we adopt stochastic gradient method to train the network, it is straightforward that if $h(\mathbf{x}) = \left\|\mathbf{x}\boldsymbol{\phi}^{'} - \mathbf{m}\right\|_2$ is convex, then we can obtain a $\boldsymbol{\phi}^{'*}$ such that $\left\|\mathbf{x}\boldsymbol{\phi}^{'*} - \mathbf{m}\right\|_2 \leq \epsilon$ (Zinkevich, 2003). Therefore, we have only to prove $h(\mathbf{x})$ is convex.

For any $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$,

$$h(\mathbf{x}_2) - h(\mathbf{x}_1) - \langle \Delta h(\mathbf{x}_1), \mathbf{x}_2 - \mathbf{x}_1 \rangle \tag{31}$$
$$= \left\|\mathbf{x}_2\boldsymbol{\phi}^{'} - \mathbf{m}\right\|_2 - \left\|\mathbf{x}_1\boldsymbol{\phi}^{'} - \mathbf{m}\right\|_2 - \frac{\left(\mathbf{x}_1\boldsymbol{\phi}^{'} - \mathbf{m}\right)\boldsymbol{\phi}^{'\top}}{\left\|\mathbf{x}_1\boldsymbol{\phi}^{'} - \mathbf{m}\right\|_2}\left(\mathbf{x}_2 - \mathbf{x}_1\right)^{\top} \tag{32}$$
$$= \left\|\mathbf{x}_2\boldsymbol{\phi}^{'} - \mathbf{m}\right\|_2 - \frac{\left(\mathbf{x}_1\boldsymbol{\phi}^{'} - \mathbf{m}\right)\left(\mathbf{x}_2\boldsymbol{\phi}^{'} - \mathbf{m}\right)^{\top}}{\left\|\mathbf{x}_1\boldsymbol{\phi}^{'} - \mathbf{m}\right\|_2} \tag{33}$$

Since

$$\left[\left\|\mathbf{x}_2\boldsymbol{\phi}^{'} - \mathbf{m}\right\|_2 \cdot \left\|\mathbf{x}_1\boldsymbol{\phi}^{'} - \mathbf{m}\right\|_2\right]^2 = \left[\left(\mathbf{x}_1\boldsymbol{\phi}^{'} - \mathbf{m}\right)\left(\mathbf{x}_2\boldsymbol{\phi}^{'} - \mathbf{m}\right)^{\top}\right]^2, \tag{34}$$

then

$$\left\|\mathbf{x}_2\boldsymbol{\phi}^{'} - \mathbf{m}\right\|_2 \cdot \left\|\mathbf{x}_1\boldsymbol{\phi}^{'} - \mathbf{m}\right\|_2 - \left(\mathbf{x}_1\boldsymbol{\phi}^{'} - \mathbf{m}\right)\left(\mathbf{x}_2\boldsymbol{\phi}^{'} - \mathbf{m}\right)^{\top} \geq 0\,. \tag{35}$$

Therefore

$$h(\mathbf{x}_2) - h(\mathbf{x}_1) - \langle \Delta h(\mathbf{x}_1), \mathbf{x}_2 - \mathbf{x}_1 \rangle \geq 0\,, \tag{36}$$

namely

$$h(\mathbf{x}_2) \geq h(\mathbf{x}_1) + \langle \Delta h(\mathbf{x}_1), \mathbf{x}_2 - \mathbf{x}_1 \rangle\,. \tag{37}$$

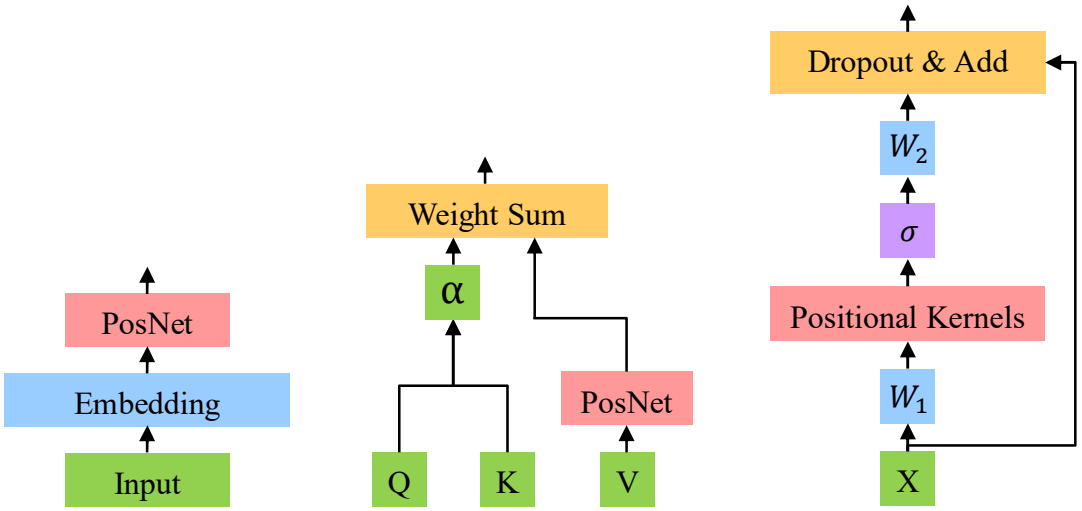
Now, we have proved that $h(\mathbf{x})$ is convex. Thus, after training for certain steps with proper learning rate setting, we can obtain a $\boldsymbol{\phi}^{'*}$ such that, for $\epsilon > 0$, $\left\|\mathbf{x}\boldsymbol{\phi}^{'*} - \mathbf{m}\right\|_2 \leq \epsilon$. It is easy to generalize the proof above when taking into account the whole sequence, i.e. $\mathbf{X} \in \mathbb{R}^{L \times d}$. □

A.2 GPU Memory Consumption

The results of GPU memory consumption is presented in Table 3. A lower GPU memory consumption translates to reduced space complexity, which is a desirable trait.

A.3 Illustration of the PosNet

The illustration of applying the PosNet to Transformer is presented in Figure 5.



(a) Embedding Layer (b) Attention Layer (c) Embdding-level PosNet

Figure 5: Illustration of applying the PosNet to (a) embedding layer and (b) attention layer. (c) illustrates embedding-level PosNet. Attention-level PosNet is identical to embedding-level PosNet, excluding $\mathbf{W}_1$ and $\mathbf{W}_2$. α is the attention score and σ is the non-linear activation function.

Approach	En-De/GB (↓)	En-Fr/GB (↓)
w/o PE	**6.3**	**9.4**
SAPE	6.4	9.4
LAPE	6.4	9.4
RPE	6.8	10.6
Adaptive-T5	6.4	9.4
PosNet	6.8	9.6

Table 3: GPU Memory Consumption of Transformer base model with different methods on WMT 2014 En-De and En-Fr tasks.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
8

☒ A2. Did you discuss any potential risks of your work?
no risks

☑ A3. Do the abstract and introduction summarize the paper's main claims?
abstract,1

☑ A4. Have you used AI writing assistants when working on this paper?
quillbot, a system providing writing assistance, check grammar mistakes and typo, 1-8

B ☑ Did you use or create scientific artifacts?

4

☑ B1. Did you cite the creators of artifacts you used?
4

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
datasets and tools used in this paper are widely used and publicly available

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
datasets and tools used in this paper are widely used and publicly available

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
datasets and tools used in this paper don't include private information or offensive content

☒ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
datasets and tools used in this paper are widely used and publicly available, detailed documentations are available on the corresponding website.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
4

C ☑ Did you run computational experiments?

4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.