ANALYZING THE IMPLICIT POSITION ENCODING ABILITY OF TRANSFORMER DECODER

Anonymous authors
Paper under double-blind review

ABSTRACT

A common limitation of Transformer Encoder's self-attention mechanism is that it cannot automatically capture the information of word order, so one needs to feed the explicit position encodings into the target model. On the other hand, Transformer Decoder with the auto-regressive attention masks is naturally sensitive to the word order information. In this work, based on the analysis of implicit position encoding power of Transformer Decoder, we obtain the conditions that at least two or more layers are required for the Decoder to encode word positions. To examine the correlations between the implicit and explicit position encodings respectively from the Transformer Encoder and Decoder, extensive experiments conducted on two large Wikipedia datasets demonstrate that all kinds of explicit position encoding mechanisms improve the performance of Decoder, but the gap of learnable position embeddings is smaller than the others. To make use of the power of implicit position encoding, we propose a new model, called *DecBERT*, and fine-tune it on GLUE benchmarks. Experimental results show that (1) the implicit position encoding ability is strong enough to enhance language modeling and perform well on downstream tasks; and (2) our model accelerates the pre-training process and achieves superior performances than the baseline systems when pre-training with the same amount of computational resource.

1 INTRODUCTION

In recent years, Transformer model proposed by Vaswani et al. (2017) has supplanted the widely-used LSTM (Hochreiter & Schmidhuber, 1997) as an indispensable component of many NLP systems. There are two branches of model variant: Transformer Encoder and Transformer Decoder. The Encoder-based Masked Language Models, e.g., BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and DeBERTa (He et al., 2020), achieve great success on many natural language understanding benchmarks (e.g. GLUE (Wang et al., 2019b) and SuperGLUE (Wang et al., 2019a)). The Decoder-based Auto-regressive Language Models such as GPT-family (Radford & Narasimhan, 2018; Radford et al., 2019; Brown et al., 2020) have shown superior performances on natural language generation. All of them utilize Multi-Head Self-Attention (MHA) mechanism (Vaswani et al., 2017). Since MHA is designed as an order-invariant mechanism (Lee et al., 2019), Transformer Encoder without the help of position encodings should share the same intuitions with the bag-of-word model. On the other hand, in Transformer Decoder, the auto-regressive attention masks make the MHA different from that of the Transformer Encoder. Specifically, Tsai et al. (2019) have proved that MHA with such attention masks is not permutation equivalent, which indicates that Transformer Decoder is not a bag-of-word model. Irie et al. (2019) find that the Transformer-based Language Models have lower perplexity scores without position embeddings. Both of them suggest that compared with Transformer Encoder, Transformer Decoder has implicit position encoding ability.

In this work, we present an analysis of Transformer Decoder's implicit position encoding ability. For brevity, we denote Transformer Encoder as Encoder and Transformer Decoder as Decoder. We first conduct a theoretical analysis to provide the conditions that allow Decoder capable of encoding the order of a word sequence. Then we draw a conclusion that two or more Decoder layers are required to capture the order information. Compared with conventional Transformer model that is generally associated with explicit position embeddings to trace word order, such as sinusoidal position embeddings (Vaswani et al., 2017) and learnable position embeddings (Devlin et al., 2019), we thus design extensive experiments to confirm that whether Decoder can perform well without explicit position encodings. In consistent with Irie et al. (2019), our experimental results on language modeling tasks confirm the implicit position encoding power of Decoder due to the comparable performance achieved without explicit position encoding, and all kinds of explicit position encodings can further improve the performance. We also notice that the improvement in terms of PPL is small (about 0.2) when using the learnable position embeddings.

On the basis of the implicit position encoding and the associated conditions, we assume that the position encoding of multi-layer Encoders could be further enhanced accordingly. To this end, we propose a new model *DecBERT* by adding the auto-regressive attention masks into the multi-layer Encoders, and we pre-train *DecBERT* as a masked language model. Experimental results show that *DecBERT* without explicit position encodings have 77 times (353.97 vs. 4.59) lower valid PPL than the baselines without position embeddings, indicating that our model retains the power of position encoding. These experiments provide a thorough understanding of the strength of the implicit position encoding ability. To further evaluate the influence of our proposed model in the large-scale pre-training scenario, we pre-train our models with the same amount of data (160 GiB) as RoBERTa. The experimental results demonstrate that our proposed *DecBERT* can accelerate the pre-training process. Moreover, when pre-training with the same amount of computational footprint, our model achieves superior performance on GLUE benchmarks.

The contributions of this paper are summarised as follows in four folds:

- We firstly make an analysis for the conditions of Decoder capable of encoding word positions. Our analysis justifies that at least two or more embedding layers are required.
- We examine the close correlations between the implicit and explicit position encodings. Experimental results confirm the implicit position encoding of Decoder which can be further enhanced by all kinds of explicit position encodings.
- We provide a thorough understanding of the strength of the implicit position encoding. Experimental results indicate that such position encoding could enhance Encoder on language modeling and also show comparable performance on downstream tasks.
- We propose a new model *DecBERT* utilizing the implicit position encoding. Our proposed model can accelerate the pre-training process. When pre-training with the same amount of resources, *DecBERT* achieves lower validation PPL and superior performances on most GLUE tasks than the baseline systems.

2 Background

Transformer is a neural network model proposed by Vaswani et al. (2017). It relies on the multi-head self-attention (MHA) mechanism.

Multi-head Self-attention (MHA). MHA takes a sequence of vectors $h = [h_1, h_2, ..., h_n]$ as input. Then they are transformed into three different vectors, query (Q), key (K) and value (V), by three linear transformations and passed to the multi-head self-attention (MHA). The computation process of a single head is:

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

where d_k is the dimension of a single head. MHA repeats the same process for h heads. The outputs of all heads are concatenated together and passed through a linear projection.

Position Embedding. Due to the order-invariance of MHA, a token embedding is added with a position embedding as the input of Transformer Encoder or Decoder. In the paper of Vaswani et al. (2017), they use a fixed sinusoidal PE. In the later work, Devlin et al. (2019) choose to use a learnable PE matrix.

Transformer Encoder and Decoder. An Encoder layer consists of multi-head attention following with a feed-forward network (FFN). The outputs of MHA and FFN are passed through a LayerNorm (Ba et al., 2016) with residual connections (He et al., 2016). Then we stack multi-layer to form a Transformer Encoder. The difference[1] between Decoder and Encoder is that Decoder uses the

[1]We do not consider the Encoder-Decoder Seq2seq structure with cross attention here. Encoder and Decoder are used independently.

auto-regressive attention masks to mask the attention values of the subsequent tokens so that Decoder can only decode tokens relying on the tokens in the past.

3 Conditions of Implicit Position Encoding

Tsai et al. (2019) have proved that Mha with auto-regressive attention masks is not permutation equivalent, but they do not provide the conditions that allow Transformer Decoder to encode word order. Is a Decoder layer like an RNN? In this section, we fill such gap. We first present the definition of permutation equivalence function by Lee et al. (2019):

Definition 1 *Denote Π as the set of all permutations over $[n] = \{1, ..., n\}$. A function $func : \mathcal{X}^n \to \mathcal{Y}^n$ is permutation equivalent iff for any permutation $\pi \in \Pi$, $func(\pi x) = \pi func(x)$.*

Tsai et al. (2019) have proved that Decoder's self-attention with the masks is not permutation equivalent. However, this is not enough to prove that Decoder has position encoding ability. Here, we need to give a definition of such ability of Decoder:

Definition 2 *Suppose that we have a sequence of tokens: $S = x_1 x_2 \cdots x_n$,[2] $\forall n \in \mathbb{N}^*$, and a function $func(\cdot)$ that takes a sequence as input and outputs the hidden state of each token. The output of x_i is denoted as $func(S)_i, \forall i \in \{1, ..., n\}$. We hypothesis that:*

(i) $func(\cdot)$ has position encoding ability iff for any permutation $\pi \in \Pi$, $func(x_1, ..., x_n)_n \neq func(\pi(x_1, ..., x_{n-1}), x_n)_n$.[3]

(ii) $func(\cdot)$ has partial position encoding ability iff for any permutation $\pi \in \Pi$, $func(x_1, ..., x_n)_n = func(\pi(x_1, ..., x_{n-1}), x_n)_n$ and $func(x_1, ..., x_n) \neq func(\pi(x_1, ..., x_{n-1}), x_n)$.

To verify the implicit position encoding ability, we then present the following proposition on the Decoder:

Proposition 1 *Transformer Decoder with two or more layers has position encoding ability.*

Since the linear transformations in the self-attention mechanism, the feed-forward network (FFN) layer, the residual connection and the layer normalization do not make any changes to our conclusions, we omit all of them in our proof process.

Proof for Proposition 1. Suppose that the model input is a sequence of tokens: $S = x_1, ..., x_n, \forall n \in \mathbb{N}^*$. The L^{th} layer's self-attention of Decoder is denoted as a function $func^{(L)}$, which is similar to the above discussions. We first input these tokens into the static token embeddings layer.[4] Then we can get a sequence of vectors: $W = w_1, ..., w_n$ which is the input of $func^{(1)}$. Since the static token embeddings layer is order-invariant, for any permutation $\pi \in \Pi$ over $w_{1:n-1} = (w_1, ..., w_{n-1})$, the attention score $w_i^T w_n$ will not change. Then we have: $func^{(1)}(W)_n = func^{(1)}(\pi(w_{1:n-1}), w_n)_n \propto \sum_{i=1}^{n}(w_i^T w_n) w_i$. However, for any permutation $\pi \in \Pi$, at least two vectors w_i and w_j of $w_{1:n-1}$ are swapped. Since the Decoder uses the auto-regressive attention masks, the output hidden states only rely on the vectors in the past. The past vectors lists of w_i and w_j are changed, which change the output hidden states of vectors within $w_{i:j}$. Then we have: $func^{(1)}(W) \neq func^{(1)}(\pi(w_{1:n-1}), w_n)$. Following this, the first layer of Decoder has partial position encoding ability.

The input of the second Decoder layer is the output of the first layer. For any permutation $\pi \in \Pi$ over $w_{1:n-1}$, the outputs of the first layer, $func^{(1)}(\pi(w_{1:n-1}), w_n)_{1:n-1}$, are not the same, which leads to different attention score distributions between $func^{(1)}(\pi(w_{1:n-1}), w_n)_i$ and $func^{(1)}(\pi(w_{1:n-1}), w_n)_n$. Second layer's output hidden states of w_n will be different, too. Then we have: $func^{(2)}(func^{(1)}(W))_n \neq func^{(2)}(func^{(1)}(\pi(w_{1:n-1}), w_n))_n$. Thus we can conclude that Decoder with two layers has position encoding ability.

For the case of decoder with more than two layers, the input hidden states now are order-variant. These layers naturally have position encoding ability.

[2] In this and the following discussions, we assume that all the tokens are different.

[3] Since Decoder uses auto-regressive attention masks, we omit the tokens on the right of x_n.

[4] We do not use the position embeddings.

4 Implicit and Explicit Position Encodings

Based on the obtained conditions of Decoder's implicit position encodings ability, we further analyze the correlations between implicit and explicit position encodings. Previous studies Irie et al. (2019) find that Transformer Decoder on Language Modeling achieve lower perplexity scores without position embeddings. In this section, we design extensive experiments to confirm that whether Decoder can perform well without explicit position encodings. If not, is the performance gap large? To this end, we specifically focus on the task of language modeling for English and Chinese, respectively.

4.1 Explicit Position Encodings Mehotds

To compare the effects of different explicit position encodings methods, we follow the same settings proposed in Dufter et al. (2021) that divide all methods into three approaches:

Adding Position Embeddings. Add position embeddings to the input before it is fed into the model. We use two commonplace methods in our experiments: the learnable position embeddings (Devlin et al., 2019), denoted by *Learnable-PE* and the sinusoidal position embeddings (Vaswani et al., 2017), denoted by *Sinusoidal-PE.*

Modifying Attention Matrix. Directly modify the attention matrix with position information. We use a simple method proposed by Dufter et al. (2020). They use a bias term to model the interaction between every two positions and use this term to adjust the raw attention score a_{ij}^{raw}:

$$a_{ij} = a_{ij}^{raw} + \beta_{ij}^{Add}. \tag{2}$$

We call their methods as additive position interaction encodings (*Add-PIE*). The reason why we choose this method is that it only contains partial position information. For two positions i and j, the bias term β_{ij} only models the relation between these two positions. If we shuffle the word order between these two positions, β_{ij} will not change. If Decoder with this method still performs well, it will further reveal the strength of the implicit position encoding. In addition, we also modify their method to replace the addition with multiplication:

$$a_{ij} = a_{ij}^{raw} \beta_{ij}^{Mul}. \tag{3}$$

We call this as multiplication position interaction encodings (*Mul-PIE*). Though one can find that they are interchangeable:

$$a_{ij}^{raw} + \beta_{ij} = a_{ij}^{raw} \left(1 + \frac{\beta_{ij}}{a_{ij}^{raw}}\right), \tag{4}$$

their optimization processes are different. The gradient of *Mul-PIE* is a_{ij}^{raw}. It is controlled by the raw attention weight, which means that the optimization process of *Mul-PIE* is adaptive. However, the gradient of *Add-PIE* is a fixed value 1 which is independent with a_{ij}^{raw}. We believe that this will make it harder to find a suitable value to adjust the raw attention weight. In different layers, we use different *PIE*. For *Mul-PIE*, all the bias terms are initialized as 1. For *Add-PIE*, all the terms are initialized as 0, so these two methods have the same effects in the early training stage.

Integration. Combine the former two approaches together. (1) *Learn+Mul*: Decoder uses learnable position embeddings and *Mul-PIE* together. (2) *Sin+Mul*: Decoder uses sinusoidal position embeddings and *Mul-PIE* together. (3) *Learn+Add*: Decoder uses learnable position embeddings and *Add-PIE* together. (4) *Sin+Add*: Decoder uses sinusoidal position embeddings and *Add-PIE* together.

4.2 Experiments Setup

Data. We use two datasets in our experiments. The first one is the WikiText-103 (Merity et al., 2017). We train and evaluate our language models on the standard splits of the WikiText-103, which contains 1.8M sentences for training and 3.76k sentences for evaluation. The second one is the Chinese Wikipedia which contains about 9.28M sentences. We randomly select 34k sentences for evaluation and 9.25M for training. We use Fairseq (Ott et al., 2019) to pre-process all the data into the binary files. All the English data is tokenized by SentencePiece tokenizer (Kudo & Richardson, 2018), which is the same as RoBERTa. All Chinese data is tokenized by character.

Basic Model. Our basic model is an 8-layer Transformer Decoder with 768 embedding size, 3072 feedforward layer hidden size, 12 attention heads and GELU activation function (Hendrycks & Gimpel, 2020), which is a smaller version of GPT and has 95M trainable parameters for English model and 77.5M for Chinese model.[5] We find that if we use a standard 12-layer GPT, the number of trainable parameters will be higher than the number of tokens in the WikiText-103 dataset. This has a risk to cause over-fitting, so we choose to use an 8-layer model. We do not use any position encodings in this model and denote it as *No-PE*. All variants in our experiments are based on it.

Training. All models are trained with Fairseq. The training objective is the Auto-regressive Language Modeling objective. We use a batch size of 128 and train for 100k steps, optimized by Adam (Kingma & Ba, 2015). We also use the polynomial learning rate decay with 10k warmup steps. All models use the same hyper-parameters. We list the details in the Appendix A. We use two NVIDIA A100 40GB GPUs to train each model. For the WikiText-103, it costs about 10 hours per model. For the Chinese Wikipedia, it costs about 8.5 hours per model.

4.3 Results and Analysis

Table 1 shows the perplexity (PPL) scores of Transformer Decoders with different position encodings on WikiText-103 and Chinese Wikipedia validation sets.

Models	WikiText-103	Chinese Wiki
W/o any position encodings		
No-PE	23.52	12.96
Adding position embeddings		
Learnable-PE	23.37	12.75
Sinusoidal-PE	22.95	12.46
Modifying Attention Matrix		
Add-PIE	23.36	12.71
Mul-PIE	22.92	12.33
Integration		
Learn+Add	23.23	12.58
Sin+Add	22.83	12.35
Learn+Mul	22.99	12.29
Sin+Mul	**22.58**	**12.12**

Table 1: Transformer Decoders perplexity (PPL) on WikiText-103 and Chinese Wikipedia validation sets. **Bold** indicates the lowest PPL of each task.

With or Without Position Encodings. From Table 1, we find that *No-PE* still can perform well, which is only about 0.2 higher than *Learnable-PE*. This from the side reveals the strength of the implicit position encoding. Besides, we still can find that all kinds of explicit position encodings can improve the performance of Decoder.

PIE vs. Position Embeddings. Though *PIE* only contains partial position information, we can still find that the *Mul-PIE* outperforms the *Sinusoidal-PE* and the *Add-PIE* outperforms the *Learnable-PE*. This partial position information is more helpful for Decoder than the position embeddings, which also from the side proves the strength of the implicit position encoding.

Additive vs. Multiplicative PIE. Table 1 indicates that the *Mul-PIE* outperforms the *Add-PIE*, which corroborates our hypothesis that the adaptive optimization process can help *Mul-PIE* find more suitable bias terms to adjust the raw attention weights. When the training step is less than 20k, the valid PPL scores of *Mul-PIE* are much lower than *Add-PIE*. This reveals that the adaptive optimization process makes *Mul-PIE* faster and easier to find suitable values to adjust a_{ij}^{raw} in the early training stage. Though the differences become smaller after 20k steps, the PPL scores of *Mul-PIE* are still lower than *Add-PIE* in the whole training process.

Combining PIE and Position Embeddings. We use *PIE* and position embeddings together to see whether this can further improve Decoder's performance. Table 1 shows that combining these two techniques, Decoder performs better than only using position embeddings. *Sin+Mul* achieves the lowest PPL among all models.

5 Strength of Implicit Position Encoding

In Section 3–4, we have learned about the conditions of implicit position encoding and the relations with explicit position encodings. However. whether such results can help us create better pre-trained

[5]The Chinese vocabulary size is smaller than English, so the Chinese model has fewer parameters.

language models remains unexplored. In this section, we introduce a new model modification, called *DecBERT* to make use of the implicit position encoding ability of the Decoder layers.

5.1 Our Proposed Approach

We introduce two new models to make use of the power of the implicit position encoding. Since we have proved that Decoder with two or more layers has position encoding ability in section 3, we choose to use two Decoder layers to design our models:

DecBERT-Same: This model has a similar structure as RoBERTa-base, but we use the auto-regressive attention masks to change the first two Encoder layers to two Decoder layers with the same direction (from left to right). Then model has 10 Encoder layers and 2 Decoder layers. As a result, its first two layers are sensitive to word order by design;

DecBERT-Diff: This model has a similar structure as *DecBERT-Same*, but the second Decoder layer has the opposite direction (from right to left) as the first one. As a result, this design can help the model gain more information from both directions.

In Section 4, we show that the explicit position encodings are helpful for Decoder, so both of our models have two versions, with or without learnable position embeddings.[6] One would think that *DecBERT* is similar to Transformer with RNN layer (Neishi & Yoshinaga, 2019). We make it clear that *DecBERT* has the same structure as RoBERTa and both of them require the same amount of computational time, which is much faster than Transformer+RNN.

The baseline model has the same structure as RoBERTa-base which is a 12-layer Encoder with 768 embedding size and 12 attention heads. We denote it as *RoBERTa-reImp*. To analyze the importance of position information, we also add another baseline model, which is the same as the former one, but without any position encodings.

5.2 Experiments Setup

We pre-train all models from scratch similar to RoBERTa-base with Fairseq. The pre-training data is the English Wikipedia Corpus. We randomly select 158.4M sentences for training and 50k sentences for validation. We tokenize it with the same method as RoBERTa-base. The pre-training objective is the Masked Language Modeling objective. We use a batch size of 256 and train for 200k steps, optimized by Adam. All models use the same hyper-parameters. We list the details in the Appendix A. We use four NVIDIA A100 40GB GPUs to pre-train each model, costing about 34.5 hours per model.

Models	W/ PE	Valid PPL
Baseline		
RoBERTa-reImp	False	353.97
RoBERTa-reImp	True	4.28
Ours (w/o position embeddings)		
DecBERT-Same	False	4.59
DecBERT-Diff	False	4.59
Ours (w/ position embeddings)		
DecBERT-Same	True	4.12
DecBERT-Diff	True	**4.07**

Table 2: The validation set perplexity of all models. (W/ PE = with position embeddings)

We fine-tune our models with seven tasks of GLUE benchmark (Wang et al., 2019b), including SST-2 (Socher et al., 2013), QNLI (Rajpurkar et al., 2016), MNLI (Williams et al., 2018), QQP,[7] MRPC (Dolan & Brockett, 2005), RTE[8] and STS-B (Cer et al., 2017). All fine-tuning hyper-parameters details are listed in the Appendix A.

5.3 Results and Analysis

Table 2 shows the perplexity of all the systems on the validation set. We observe that our proposed models not only reveal the strength of implicit position encoding, but achieves superior performances.

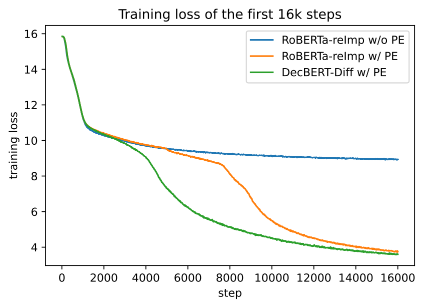
[6]We also try to use position interaction encodings (PIE), but we find that PIE's parameters keep unchanged. We give the reasons in the Appendix C.

[7]https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

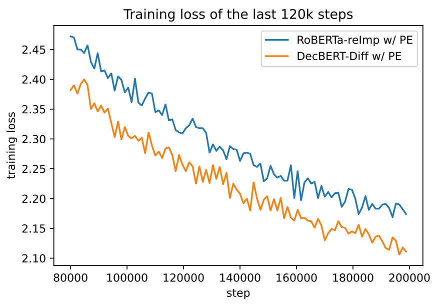
[8]https://aclweb.org/aclwiki/Recognizing_Textual_Entailment

Models	SST-2	QNLI	QQP	RTE	MNLI-m/mm	MRPC	STS-B	Avg.
Baseline (w/ position embeddings)								
RoBERTa-reImp	89.56	89.24	90.14	64.40	80.14/80.62	**86.60**	86.22	83.37
Ours (w/o position embeddings)								
DecBERT-Same	89.58	89.50	90.16	62.68	79.56/80.42	85.88	**86.58**	83.05
DecBERT-Diff	90.30	88.86	90.28	59.28	79.78/80.78	86.08	86.06	82.68
Ours (w/ position embeddings)								
DecBERT-Same	90.12	89.18	**90.32**	64.78	80.48/80.64	86.24	86.34	83.51
DecBERT-Diff	**90.78**	**89.56**	90.08	**65.98**	**80.92/81.26**	85.86	86.24	**83.84**

Table 3: Different models' performance on the dev sets of GLUE benchmark. All results are averaged over five different random seeds (1, 2, 3, 4 and 5). MNLI-m is the matched version and MNLI-mm is the mismatched version. All tasks except STS-B use accuracy as their evaluation metrics. STS-B uses the Spearman rank correlation. The results are reported as $r \times 100$. **Bold** indicates the best score for each task.



(a) The pre-training loss of the first 16k steps.



(b) The pre-training loss of the last 120k steps.

Figure 1: The pre-training loss of our models.

Comparing All Models w/o PE. Since the self-attention of Encoder is order-invariant, the extra position information is inevitable for it to model language. Otherwise, it just becomes a bag-of-word model. From Table 2, we can find that the valid PPL score of *RoBERTa-reImp w/o PE* is up to 353.97, which is about 82 times higher than its counterpart with position embeddings (4.28), revealing that this bag-of-word model cannot model language well. Comparing *DecBERT-Same/Diff w/o PE* and *RoBERTa-reImp w/o PE*, Table 2 shows that changing the first two Encoder layers with two Decoder layers can decrease the valid PPL scores by a large margin (from 353.97 to 4.59), which is only 0.31 higher than *RoBERTa-reImp w/ PE*. This reveals that Decoder layers can help Encoder model language and corroborates our theoretical analysis in section 3 that Decoder with two layers has implicit position encoding ability. Comparing our models on downstream tasks, Table 3 shows that our models w/o PE retain the same level performance as the baseline w/ PE.

Comparing All Models w/ PE. Table 2 shows that the explicit position embeddings further boosts the performances of our models. The valid PPL scores of *DecBERT-Same/Diff w/ PE* are lower than *RoBERTa-reImp w/ PE*. With the help of different directional Decoder layers, *DecBERT-Diff w/ PE* achieves the lowest PPL score. After fine-tuning on the downstream tasks, Table 3 reveals that they have better performance on most tasks. Comparing *DecBERT-same* and *DecBERT-diff*, the information from both directions is more useful for model to achieve better performance.

Why Models can Benefit from Decoder Layers? Jawahar et al. (2019) perform a series of experiments to analyze the language structure information learned by BERT. Their results reveal that the lower layers tend to capture the surface-level structure information. Since the multi-head attention of BERT is a "balance" structure without any inductive bias, the model needs to learn to be aware of word order during pre-training. Our models' pre-training loss curves in Figure 1 corroborate our analysis. We can find that the pre-training process of *RoBERTa-reImp w/ PE* can be divided into three stages: 1. plateau stage (0-8000 steps), 2. "diving" stage (8000-10000 steps), 3. convergence stage (10000-final steps). In the plateau stage, *RoBERTa-reImp w/ PE* has almost the same training loss as its counterpart without PE, which indicates that it is still a bag-of-word model and does not learn

Models	SST-2	QNLI	QQP	RTE	MNLI-m/mm	CoLA	MRPC	STS-B	Avg.
Original PLMs									
BERT-base	93.5	90.5	89.2	66.4	84.6/83.4	52.1	84.8	85.8	81.1
BERT-large	94.9	92.7	89.3	70.1	86.7/85.9	60.5	85.4	86.5	83.6
RoBERTa-base	95.3	93.2	89.3	71.5	87.4/86.5	55.2	86.5	89.3	83.8
Our Models									
RoBERTa-300k	**94.7**	91.5	**89.4**	66.5	85.9/85.1	56.3	85.4	**86.8**	82.4
DecBERT-300k	94.5	**92.0**	89.3	**72.0**	**86.8/85.5**	**59.6**	**86.0**	**86.8**	**83.6**

Table 4: Different models' performance on the **test sets** of GLUE benchmark. MNLI-m is the matched version and MNLI-mm is the mismatched version. All tasks except STS-B and CoLA use accuracy as their evaluation metrics. STS-B uses the Spearman rank correlation. CoLA (Warstadt et al., 2019) uses the Matthews correlation coefficient. The results are reported as $r \times 100$. The scores of BERT-base and BERT-large are from Devlin et al. (2019). The scores of RoBERTa-base are fine-tuned by ourselves. **Bold** indicates the best score of our models for each task.

to make use of the position information. In the diving stage, the training loss of *RoBERTa-reImp w/ PE* decreases rapidly, while *RoBERTa-reImp w/o PE* starts to converge. This reveals that the word order information becomes more useful for a model to understand language. Why are Decoder layers helpful? The first two Decoder layers can break the "balance" of the multi-head attention by design. The inductive bias from the auto-regressive attention masks makes the first two layers easier capture the surface-level structure information, like word order. From Figure 1a, we can find that the plateau stage is shortened. Though the gap between *RoBERTa-reImp w/ PE* and *DecBERT-Diff w/ PE* become smaller in the convergence stage, Figure 1b indicates that *DecBERT-Diff w/ PE* still has lower training loss in the whole pre-training process.

The analysis and results reveal that the implicit position encoding of Decoder layers is powerful enough to help models understand language and have better performance on downstream tasks. One should notice that the Decoder layers only consider one-side information flow, which is weaker than the Encoder layers. This also proves the strength of the implicit position encoding.

5.4 Large-scale Pre-training Scenario

The results in Table 3 indicate that Decoder layers are helpful for models. However, limited by computational resources, our models are pre-trained with less data, fewer steps and smaller batch size than the original RoBERTa-base model. It is necessary for us to figure out whether such modification can benefit the pre-trained language models in the large-scale pre-training scenario. It is impossible for us to pre-train all 6 models in Table 2 from scratch with the same amount of computational footprint as RoBERTa-base. Thus, we decide to pre-train our best model *DecBERT-Diff w/ PE* and the baseline model *RoBERTa-reImp w/ PE* with the same amount of data (160GiB[9]) as RoBERTa-base. The batch size is set to 4096 and the pre-training steps are 300k. Though these settings are still smaller than the original RoBERTa-base (8k batch size, 500k pre-training steps), we believe that the results of our experiments are enough to understand the influence of our modification in the large-scale pre-training scenario. We pre-train each model with 8 NVIDIA A100 40GB GPUs, costing about 15 days per model. The hyper-parameters details can be seen in the Appendix A. For brevity, we denote *DecBERT-Diff w/ PE* as *DecBERT-300k* and *RoBERTa-reImp w/ PE* as *RoBERTa-300k*.

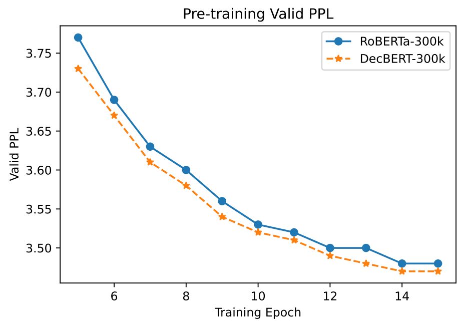


Figure 2: The PPL scores on validation set from epoch 5 to epoch 15 of our models.

Results and Analysis in the Large-scale Pre-training Scenario. The experimental results are similar to the small-scale pre-training. Figure 2 indicates that *DecBERT-300k* has lower valid PPL

[9]The details of our pre-training corpus can be seen in the Appendix B.

scores in the whole pre-training process. At the 13^{th} epoch (265k steps), the valid PPL score of *DecBERT-300k* is 3.48, which is the same as *RoBERTa-300k* at the 15^{th} epoch (300k steps). This suggests that the pre-training process of *DecBERT-300k* is about 2 epochs faster than *RoBERTa-300k*. In the large-scale pre-training scenario, our modification still can accelerate the pre-training process. Comparing on the downstream tasks, Table 4 shows that the gap between *DecBERT-300k* and *RoBERTa-300k* even becomes larger. The average score is 1.2 points higher. To better analyzing the performances of our models, we also include the results of BERT-base, BERT-large and RoBERTa-base in Table 4. We surprisingly find that the average score of *DecBERT-300k* is only 0.2 lower than RoBERTa-base, which only costs around 1/3 computational footprint of the original RoBERTa-base.

All results in this part indicate that our modification is more helpful in the large-scale pre-training scenario. It can accelerate the pre-training process. When pre-training with the same amount of computational resources, our modification can achieve superior performance on language modeling and downstream tasks.

6 Related Works

In the previous works (Vaswani et al., 2017; Shaw et al., 2018; Huang et al., 2019; Dai et al., 2019; Child et al., 2019), they indicate that the self-attention mechanism of Transformer Encoder is permutation equivalent, so it needs to use the position embedding. Tsai et al. (2019) have proved that Decoder's self-attention is not permutation equivalent, indicating that Decoder is not a bag-of-word model as Encoder, but they do not conduct further analysis on Decoder's implicit position encoding ability. In section 3, we show that only if Decoder has two or more layers, it has such ability. Apart from the theoretical analysis, Irie et al. (2019) train the Transformer Language Models with speech dataset. They find that models without position embeddings have lower perplexity scores. Schlag et al. (2021a) introduce a new Linear Transformer Language Model with fast weight memories (Schmidhuber, 1992; Schlag et al., 2021b), which has lower perplexity without position encodings on the WikiText-103 dataset.

Furthermore, an explosion of work focuses on proposing a better method to add the position information into the pre-trained language model. Dufter et al. (2021) give a comprehensive introduction of different position encodings methods of Transformer. They divide position encodings into three approaches. One line of such work is to add position embeddings to the input before it is fed to the actual Transformer model (Vaswani et al., 2017; Shaw et al., 2018; Devlin et al., 2019; Kitaev et al., 2020; Liu et al., 2020; Press et al., 2020; Wang et al., 2020). The second line of work directly modify the attention matrix (Dai et al., 2019; Dufter et al., 2020; He et al., 2020; Wu et al., 2021a; Ke et al., 2021; Su et al., 2021). The last one combine the first two approaches together. However, all of them focus on finding a better method to use an extra set of parameters to trace the word order. Our work provide a better understanding of Decoder's implicit position encoding ability.

Most similar to our modification in Section 5, Im & Cho (2017) propose a self-attention based model which achieve better performance on SNLI task (Bowman et al., 2015) without the help of explicit position encodings. However, their models are different from the standard Transformer and use extra local attention masks to control the information flow. With the popularity of the Transformer model in the Computer Vision field, some works propose different methods to make Vision Transformer know word order implicitly (Chu et al., 2021; Yuan et al., 2021; Wu et al., 2021b), but all of them modify the models with convolution neural network (Lecun et al., 1998).

7 Conclusion

In this paper, we analyze the implicit position encoding ability of Transformer Decoder. We justify that at least two layers are needed for Decoder to encode word order. Furthermore, we provide a better understanding between the implicit and explicit position encodings. Decoder can benefit from all kinds of explicit position encodings, but the improvement is small from the learnable position embeddings. To make use of the strength of the implicit position encoding ability, we introduce a model, called *DecBERT*. Our models retain the same level of performance as the baseline without using extra parameters to trace position. In the large-scale pre-training scenario, our modification converges faster and has lower valid PPL. When pre-training with the same amount of resources, our model achieves better performance on most downstream tasks.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 632–642, Lisbon, Portugal, 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1075. URL https://www.aclweb.org/anthology/D15-1075.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 1–14, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2001. URL https://www.aclweb.org/anthology/S17-2001.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019.

Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers, 2021.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1285. URL https://www.aclweb.org/anthology/P19-1285.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pp. 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005. URL https://www.aclweb.org/anthology/I05-5002.

Philipp Dufter, Martin Schmitt, and Hinrich Schütze. Increasing learning efficiency of self-attention networks through direct position interactions, learnable temperature, and convoluted attention. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 3630–3636, Barcelona, Spain (Online), 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.324. URL https://www.aclweb.org/anthology/2020.coling-main.324.

Philipp Dufter, Martin Schmitt, and Hinrich Schütze. Position information in transformers: An overview, 2021.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL `https://doi.org/10.1109/CVPR.2016.90`.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention, 2020.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `https://doi.org/10.1162/neco.1997.9.8.1735`.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJe4ShAcF7`.

Jinbae Im and Sungzoon Cho. Distance-based self-attention network for natural language inference, 2017.

Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney. Language modeling with deep transformers. *Interspeech 2019*, 2019. doi: 10.21437/interspeech.2019-2225. URL `http://dx.doi.org/10.21437/Interspeech.2019-2225`.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3651–3657, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1356. URL `https://www.aclweb.org/anthology/P19-1356`.

Guolin Ke, Di He, and Tie-Yan Liu. Rethinking positional encoding in language pre-training. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=09-528y2Fgf`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=rkgNKkHtvB`.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL `https://aclanthology.org/D18-2012`.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753. PMLR, 09–15 Jun 2019. URL `http://proceedings.mlr.press/v97/lee19d.html`.

Xuanqing Liu, Hsiang-Fu Yu, Inderjit S. Dhillon, and Cho-Jui Hsieh. Learning to encode position for transformer with continuous dynamical model. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6327–6335. PMLR, 2020. URL `http://proceedings.mlr.press/v119/liu20n.html`.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=Byj72udxe`.

Masato Neishi and Naoki Yoshinaga. On the relation between position information and sentence length in neural machine translation. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pp. 328–338, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/K19-1031. URL `https://aclanthology.org/K19-1031`.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 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)*, pp. 48–53, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4009. URL `https://www.aclweb.org/anthology/N19-4009`.

Ofir Press, Noah A. Smith, and Mike Lewis. Shortformer: Better language modeling using shorter inputs, 2020.

A. Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL `https://www.aclweb.org/anthology/D16-1264`.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight memory systems, 2021a.

Imanol Schlag, Tsendsuren Munkhdalai, and Jürgen Schmidhuber. Learning associative inference using fast weight memory. In *International Conference on Learning Representations*, 2021b. URL `https://openreview.net/forum?id=TuK6agbdt27`.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992. doi: 10.1162/neco.1992.4.1.131.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 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)*, pp. 464–468, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2074. URL `https://www.aclweb.org/anthology/N18-2074`.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/D13-1170`.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2021.

Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4344–4353, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1443. URL `https://www.aclweb.org/anthology/D19-1443`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html`.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 3261–3275, 2019a. URL `https://proceedings.neurips.cc/paper/2019/hash/4496bf24afe7fab6f046bf4923da8de6-Abstract.html`.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019b. URL `https://openreview.net/forum?id=rJ4km2R5t7`.

Benyou Wang, Donghao Zhao, Christina Lioma, Qiuchi Li, Peng Zhang, and Jakob Grue Simonsen. Encoding word order in complex embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=Hke-WTVtwr`.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019. doi: 10.1162/tacl_a_00290. URL `https://www.aclweb.org/anthology/Q19-1040`.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL `https://www.aclweb.org/anthology/N18-1101`.

Chuhan Wu, Fangzhao Wu, and Yongfeng Huang. Da-transformer: Distance-aware transformer, 2021a.

Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers, 2021b.

Kun Yuan, Shaopeng Guo, Ziwei Liu, Aojun Zhou, Fengwei Yu, and Wei Wu. Incorporating convolution designs into visual transformers, 2021.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books, 2015.

A HYPER-PARAMETERS DETAILS

Hyper-parameter	*No-PE*
Number of Layers	8
Hidden size	768
FNN inner hidden size	3072
Attention Heads	12
Attention Head size	64
Dropout	0.1
Warmup Steps	10k
Max Steps	100k
Learning Rates	5e-5
Batch Size	128
Weight Decay	0.001
Learning Rate Decay	Polynomial
Adam ϵ	1e-6
Adam β_1	0.9
Adam β_2	0.998
Gradient Clipping	0.1
Random Seed	1

Table 5: Hyper-parameters for pre-training the multi-layer Decoder Models. Since all models share the same hyper-parameters, we only report the parameters of *No-PE*.

Hyper-parameter	*RoBERTa-reImp*
Number of Layers	12
Hidden size	768
FNN inner hidden size	3072
Attention Heads	12
Attention Head size	64
Dropout	0.1
Warmup Steps	10k
Max Steps	200k
Learning Rates	1e-4
Batch Size	256
Weight Decay	0.01
Learning Rate Decay	Polynomial
Adam ϵ	1e-6
Adam β_1	0.9
Adam β_2	0.98
Gradient Clipping	0.5
Random Seed	1

Table 6: Hyper-parameters for pre-training the multi-layer Encoder Models (small-scale pre-training). Since all models share the same hyper-parameters, we only report the parameters of *RoBERTa-reImp*.

Hyper-parameter	*RoBERTa-300k* and *DecBERT-300k*
Number of Layers	12
Hidden size	768
FNN inner hidden size	3072
Attention Heads	12
Attention Head size	64
Dropout	0.1
Warmup Steps	24k
Max Steps	500k
Learning Rates	3e-4
Batch Size	4096
Weight Decay	0.01
Learning Rate Decay	Tri_stage
Adam ϵ	1e-6
Adam β_1	0.9
Adam β_2	0.98
Gradient Clipping	2.0

Table 7: Hyper-parameters for pre-training the multi-layer Encoder Models (large-scale pre-training).

Hyper-parameter	MNLI	QNLI	QQP	RTE	SST-2	MRPC	STS-B	CoLA
Learning Rates	1e-5	1e-5	1e-5	2e-5	1e-5	{1e-5, 2e-5}	2e-5	1e-5
Weight Decay	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Batch Size	32	32	32	16	32	16	16	16
Warmup Steps	7432	1986	28318	122	1256	137	214	320
Max Steps	123873	33112	113272	2036	20935	2296	3598	5336
Adam ϵ	1e-6	1e-6	1e-6	1e-6	1e-6	1e-6	1e-6	1e-6
Adam β_1	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
Adam β_2	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999
Gradient Clipping	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 8: Hyper-parameters for fine-tuning all models on downstream tasks. All models use the polynomial learning rate decay. Most of the hyper-parameters are recommended by Fairseq `https://github.com/pytorch/fairseq/tree/main/examples/roberta/config/finetuning`.

B The details of the large-scale pre-training corpus

Due to the licensing issues, the RoEBRTa team does not share their 160 GiB pre-training corpus.[10] We build the pre-training corpus by ourselves. The first part is the same as BERT. We use the English wikipedia dump (about 17 GiB) and the bookcorpus (Zhu et al., 2015) (about 4 GiB). The second part is based on the Pile dataset (Gao et al., 2020), which is a large datasets with 800 GiB diverse text data. We randomly extract 64 GiB data from the Pile-cc block, 35 GiB data from the OpenWebText2 block and 43 GiB data from the Books3 block. The overall size of all data is about 163 GiB, which is the same as RoBERTa-base.

C Why Do We Not Use PIE?

We have conducted two extra experiments to examine whether *PIE* is useful for Masked Language Models. We pre-train a 12-layer GPT+PIE model and a RoBERTa-base+PIE model with the Masked Language Modeling objective. We surprisingly find that the values of *PIE* do not change. This indicates that *PIE* learns nothing. However, when we pre-train a GPT+PIE model with the Auto-regressive Language modeling objective, the values of *PIE* will change. This indicates that it is the objective function that causes this problem. Further analysis remains for future work.

[10]`https://github.com/pytorch/fairseq/issues/2947`