ON POSITION EMBEDDINGS IN BERT

Anonymous authors
Paper under double-blind review

ABSTRACT

Various Position Embeddings (PEs) have been proposed in Transformer based architectures (e.g. BERT) to model word order. These are empirically-driven and perform well, but no formal framework exists to systematically study them. To address this, we present three expected properties of PEs that capture word distance in vector space: *translation invariance*, *monotonicity*, and *symmetry*. These properties formally capture the behaviour of PEs and allow us to reinterpret sinusoidal PEs in a principled way. An empirical evaluation of seven PEs (and their combinations) for classification and span prediction shows that fully-learnable absolute PEs perform better in classification, while relative PEs perform better in span prediction. We contribute the first formal analysis of desired properties for PEs and a principled discussion to its connection to typical downstream tasks.

1 INTRODUCTION

Position embeddings (PEs) are crucial in Transformer-based architectures for capturing word order; without them, the representation is bag-of-words. Fully learnable absolute position embeddings (APEs) were first proposed by Gehring et al. (2017) to capture word position in Convolutional Seq2seq architectures. Sinusoidal functions were also used with Transformers to parameterize PEs in a fixed ad hoc way (Vaswani et al., 2017). Recently, Shaw et al. (2018) used relative position embedding (RPEs) with Transformers for machine translation. More recently, in Transformer pre-trained language models, BERT (Devlin et al., 2018; Liu et al., 2019) and GPT (Radford et al., 2018) used fully learnable PEs. Yang et al. (2019) modified RPEs and used them in the XLNet pre-trained language model. Ke et al. (2020) combined APEs and RPEs in pre-trained language models. To our knowledge, the fundamental differences between the various PEs have not been studied in a principled way.

We posit that the aim of PEs is to capture the sequential nature of positions in vector space, or technically, to bridge the distances in $\mathbb{N}$ and $\mathbb{R}^D$. We therefore propose three expected properties for PEs: *monotonicity*, *translation invariance*, and *symmetry* [1]. Using these properties, we formally reinterpret existing PEs. We show in a principled way the limitations of sinusoidal PEs (Vaswani et al., 2017): they cannot adaptively meet the *monotonicity* property – thus we propose learnable sinusoidal PEs. We conclude that (a) RPEs do not meet the *symmetry* property; and that (b) fully-learnable APEs nearly meet all properties even under no constrains (see Tab. 1).

We empirically show that fully-learnable APEs perform better in classification, while RPEs perform better in span prediction tasks. This is explained by our proposed properties as follows: In classification, which heavily relies on the unshiftable `[CLS]` token[2] for inference, fully-learnable APEs which do not strictly have the *translation invariance* property still perform well because they can flexibly deal with `[CLS]`. However, span prediction inference is based on each token and can therefore benefit from the *asymmetry* of RPEs to distinguish the backward and forward words. Our experiments also show that BERT with sinusoidal APEs slightly outperforms fully-learnable APEs in span prediction but underperforms it in classification tasks. Both for APEs and RPEs, learning frequencies in sinusoidal PEs is beneficial. Lastly, sinusoidal PEs can be generalized to longer doc-

[1]Informally, derived from positions originally in integer domain $\mathbb{N}$, one may expect position vectors in vector space to have the following properties: 1) neighboring positions are embedded closer than far-way ones; 2) distances of two arbitrary m-offset position vectors are identical; 3) metric (distance) itself is symmetric.

[2] `[CLS]` is a special token of BERT for classification, which is usually fixed in the first position and its corresponding output in the last layer is used for predictions. Thus, it is crucial for predictions in classification.

Table 1: Comparison of PEs. P_x or $P(x)$ is the x-th absolute/relative position vector (the latter is parameterized by sinusoidal functions). PEs in bold are first proposed in this paper. † means "empirically observed" as fully learnable PEs cannot be assumed with any general form. All other cases can be directly inferred from the general form of PEs.

PEs	formulation	parameter scale	translation invariance	monotonicity	symmetry
fully learnable APE (Gehring et al., 2017)	$P_x \in \mathbb{R}^D$	$L \times D$	nearly†	locally†	nearly †
fixed sinusoidal APE (Vaswani et al., 2017)	$P(x) = [\cdots, \sin(\omega_i x), \cos(\omega_i x), \cdots]^T$; $\omega_i = (1/10000)^{2i/D}$	0	✓	locally	✓†
learnable sinusoidal APE	$P(x) = [\cdots, \sin(\omega_i x), \cos(\omega_i x) \cdots]^T$; $\omega_i \in \mathbb{R}$	$\frac{D}{2}$	✓	locally†	✓†
fully learnable RPE (Shaw et al., 2018)	$P_x \in \mathbb{R}^D$	$L \times D$	✓	locally†	✗
fixed sinusoidal RPE (Wei et al., 2019)	$P(x) = [\cdots, \sin(\omega_i x), \cos(\omega_i x), \cdots]^T$; $\omega_i = (1/10000)^{2i/D}$	0	✓	locally	✗
learnable sinusoidal RPE	$P(x) = [\cdots, \sin(\omega_i x), \cos(\omega_i x), \cdots]^T$; $\omega_i \in \mathbb{R}$	L	✓	locally†	✗

uments because they strictly meet the translation invariance property, while fully-learnable APEs cannot.

2 Position Embeddings

Gehring et al. (2017); Vaswani et al. (2017) use absolute word positions as additional features in neural networks. Positions $x \in \mathbb{N}$ are distributively represented as an *embedding* of x as an element $\vec{x} \in \mathbb{R}^D$ in some Euclidean space. By standard methods in representation learning, similarity between embedded objects $\vec{x}$ and $\vec{y}$ is typically expressed by an inner product $\langle \vec{x}, \vec{y} \rangle$, for instance the dot product gives rise to the usual cosine similarity between $\vec{x}$ and $\vec{y}$.

2.1 Desiderata for position embeddings

Generally, if words appear close to each other in a text (i.e., their positions are nearby), they are more likely to determine the (local) semantics together, than if they occurred far apart. Hence, *positional proximity* of words x and y should result in proximity of their embedded representations $\vec{x}$ and $\vec{y}$. One common way of formalizing this is that an embedding should preserve the *order* of distances among positions. We denote $\phi(\cdot, \cdot)$ as a function to calculate closeness/proximity between embedded positions, and any inner product can be a special case of $\phi(\cdot, \cdot)$ with good properties. We can express preservation of the order of distances as: For every $x, y, z \in \mathbb{N}$,

$$|x - y| > |x - z| \implies \phi(\vec{x}, \vec{y}) < \phi(\vec{x}, \vec{z}) \tag{1}$$

Note that on the underlying space, the property in Eq. (1) has been studied for almost 60 years (Shepard, 1962), in both algorithmics (Bilu & Linial, 2005; Badoiu et al., 2008; Maehara, 2013), and machine learning (Terada & Luxburg, 2014; Jain et al., 2016) under the name *ordinal embedding*. As we are interested in the simple case of positions from $\mathbb{N}$, Eq. (1) reduces to the following property:

Property 1. **Monotonicity**: The proximity of embedded positions decreases when positions are further apart:

$$\forall x, m, n \in \mathbb{N} : m > n \iff \phi(\vec{x}, \overrightarrow{x+m}) < \phi(\vec{x}, \overrightarrow{x+n}) \tag{2}$$

A priori, a position embedding might treat every element $\mathbb{N}$ individually. However, considering pairs of positions based on their *relative* proximity (rather than the absolute value of the positions), can lead to simplified and efficient position embeddings (Wang et al., 2020). Such embeddings satisfy *translation invariance*:

Property 2. **Translation invariance**: The proximity of embedded positions are translation invariant:

$$\forall x_1, \ldots, x_n, m \in \mathbb{N} : \phi(\vec{x}_1, \overrightarrow{x_1+m}) = \phi(\vec{x}_2, \overrightarrow{x_2+m}) = \cdots = \phi(\vec{x}_n, \overrightarrow{x_n+m}) \tag{3}$$

Finally, since the inner product is symmetric, we also consider whether $\phi(\cdot, \cdot)$ is symmetric:

Property 3. **Symmetry**: The proximity of embedded positions is symmetric,

$$\forall x, y \in \mathbb{N} : \phi(\vec{x}, \vec{y}) = \phi(\vec{y}, \vec{x})$$

Next we examine several existing PEs in relation to these properties, either formally or empirically.

3 Understanding Position Embeddings (PEs)

PEs come in two variants: *absolute* PEs (APEs) where single positions are mapped to elements of the representation space, and *relative* PEs (RPEs) where the *difference* between positions (i.e., $x - y$ for $x, y \in \mathbb{N}$) is mapped to elements of the embedding space. For Transformer-based architectures, the difference between APEs and RPEs manifests itself in the attention mechanism, in particular how the matrices of query, key, and value weights W^Q, W^K, and W^V are used to calculate attention in each attention head. Consider two positions $x, y \in \mathbb{N}$, let WE_x be the word embedding of the word at position x, and let P_x and P_{x-y} be the embeddings of the position x and relative position $x - y$, respectively. The query-key-value vector for the word at position x is typically calculated as below for APEs and RPEs[3] respectively:

$$\text{APE: } \begin{bmatrix} Q_x \\ K_x \\ V_x \end{bmatrix} = (\mathrm{WE}_x + P_x) \odot \begin{bmatrix} W^Q \\ W^K \\ W^V \end{bmatrix} \quad ; \quad \text{RPE: } \begin{bmatrix} Q_x \\ K_x \\ V_x \end{bmatrix} = \mathrm{WE}_x \odot \begin{bmatrix} W^Q \\ W^K \\ W^V \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ P_{x-y} \\ P_{x-y} \end{bmatrix} \tag{4}$$

Observe that while the APEs calculation is linear in (W^Q, W^K, W^V) with the word and position embeddings merged into the coefficient, the RPEs calculation is affine, with the relative position embedding P_{x-y} acting as an offset independent of the word embedding WE_x.

In Transformers, the resulting representation is a sum of value vectors with weights depending on $A = QK^T$, that is, $\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(QK^T/\sqrt{d_k})V$. In the rest of the paper, we examine PEs in the above architecture with respect to the properties introduced in Section 2. In particular, we study four well-known variants of PEs: (1) **fully learnable APEs** (Gehring et al., 2017), (2) **fixed sinusoidal APEs** (Vaswani et al., 2017), (3) **fully learnable RPEs** (Shaw et al., 2018), and (4) **fixed sinusoidal RPEs** (Wei et al., 2019).

3.1 Examining PEs via Identical word probing

In APEs, each element of the attention weight matrix ($A = QK^T$) in the first layer is given by:

$$\begin{aligned} a_{ij} &= (w_i + p_i)W^{Q,1}((w_j + p_j)W^{K,1})^T \\ &= \underbrace{w_i W^{Q,1}(W^{K,1})^T w_j^T}_{\text{word-word correspondence}} + \underbrace{w_i W^{Q,1}(W^{K,1})^T p_j^T}_{\text{word-word correspondence}} + \underbrace{p_i W^{Q,1}(W^{K,1})^T w_j^T}_{\text{word-word correspondence}} + \underbrace{p_i W^{Q,1}(W^{K,1})^T p_j^T}_{\text{word-word correspondence}} \end{aligned}$$

Identical word probing for PEs To study the effect of only PEs in A without considering individual words, we use *identical word probing*: feed many repeated identical words (can be arbitrary) as a sentence $\bar{w}$ to BERT to check the attention values $\bar{A}^{(1)}$, with each element

$$\bar{a}^1_{ij} = (\bar{w} + p_i)W^{Q,1}((\bar{w} + p_j)W^{K,1})^T \tag{5}$$

This shows that the selection of words does not affect the general tendency of $\bar{A}^{(1)}$, as we take an average of $\bar{A}^{(1)}$ over some randomly-selected words. Namely, $\bar{A}^{(1)}$ is context-free and only related to learned PEs. Thus, $\bar{A}^{(1)}$ can be treated as a general attention bias and can also implicitly convey position-wise proximity in Transformers. We investigate existing positions with the probe test in Sec. 5.2. Note that the probing test could also be applied to RPEs.

[3]There are many variants of RPEs (e.g., (Dai et al., 2019)). As selecting RPEs is not the main concern in this paper, we give the original (and typical) RPEs only. One can easily extend this work to other RPE variants.

3.2 UNDERSTANDING SINUSOIDAL PES

With a sinusoidal parameterization in PEs, we may use a specific proximity, i.e., an efficient inner product like a dot product, to check if the sinusoidal form of PEs meets the above properties. The dot product between any two position vectors is

$$A_{x,y} = \langle \vec{x}, \vec{y} \rangle = \text{sum}\left(\begin{bmatrix} \sin(\omega_1 x) \\ \cos(\omega_1 x) \\ \cdots \\ \sin(\omega_{\frac{D}{2}} x) \\ \cos(\omega_{\frac{D}{2}} x) \end{bmatrix} \odot \begin{bmatrix} \sin(\omega_1 y) \\ \cos(\omega_1 y) \\ \cdots \\ \sin(\omega_{\frac{D}{2}} y) \\ \cos(\omega_{\frac{D}{2}} y) \end{bmatrix} \right) = \text{sum}\left(\begin{bmatrix} \sin(\omega_1 x)\sin(\omega_1 y) \\ \cos(\omega_1 x)\cos(\omega_1 y) \\ \cdots \\ \sin(\omega_{\frac{D}{2}} x)\sin(\omega_{\frac{K}{2}} y) \\ \cos(\omega_{\frac{K}{2}} x)\cos(\omega_{\frac{D}{2}} y) \end{bmatrix} \right) = \sum_{i=0}^{\frac{D}{2}} \cos(\omega_i (x-y))$$

Note that sinusoidal PEs satisfy both Property 2 (*translation invariance*) because the inner product is only associated with its position difference $x - y$, and Property 3 (*symmetry*), because the dot product itself is symmetric: $\langle \vec{x}, \vec{y} \rangle = \langle \vec{y}, \vec{x} \rangle$. Note also that checking Property 1 is equivalent to checking monotonicity of the map $\psi(m) = \sum_{i=0}^{D/2} \cos(\omega_i m)$. $\psi(m)$ is monotone on intervals where its first order derivative $\psi'(m) = \sum_{i=0}^{D/2} -\omega_i \sin(\omega_i m)$ does not change sign, and these intervals depend on the choice of ω_i. With fixed frequencies $\omega_i = (1/10000)^{2i/D}$, it is monotonous when m is roughly between 0 and 50, indicating that it can only strictly perceive a maximum distance of 50 and it is insensitive to far-way distances (e.g. longer than 50).

Although sinusoidal PEs with fixed frequencies (i.e., $\omega_i = (1/10000)^{2i/D}$) are common in APEs and RPEs, we argue that learning these frequencies is useful because it can adaptively adjust intervals of monotonicity (they do not have to be 0-50 as in fixed sinusoidal APEs) [4]. With trainable frequencies, we can adaptively allocate a number of frequencies in a data-driven way. App. A.2 explains the expressive power of sinusoidal PEs with trainable frequencies from the perspective of Fourier series. Extending existing fixed sinusoidal PEs to a learnable version with learnable frequencies gives two variants: **learnable sinusoidal APEs** and **learnable sinusoidal RPEs**.

3.3 UNDERSTANDING RPES

RPEs ignore the absolute position of words and directly encode their relative distance. The RPEs expression adheres to the *translation invariance* property during parameterization, since relative distance with the same offset will be embedded as a same embedding, namely, $p_{x_1-y1} = p_{x_2-y2}$ if $x_1 - y_1 = x_2 - y_2$. Plus, RPEs that separately embed forward and backward relative embeddings, i.e., $P_{i-j} \neq P_{j-i}$, may not meet symmetry during parameterization.

Sinusoidal RPEs can also embed neighboring relative position in close vectors with a local *monotonicity*, similarly to sinusoidal APEs. In sinusoidal RPEs, the dimension is the same as the dimension of each head (typically 64). Intervals of monotonicity are slightly different from APEs which have a dimension of 768. Note that the dot products m-offset between sinusoidal RPEs should be identical without distinguishing positive RPEs and negative RPEs [5]. This may negatively affect the perception of forward and backward words in Transformers.

4 EXPERIMENTS

We empirically compare 13 types of PEs in classification and span prediction tasks.

Datasets For classification, we use the GLUE (Wang et al., 2018) benchmark, which includes datasets for both single document classification and sentence pair classification. For span prediction, we use the SQuAD V1.1 and V2.0 datasets (100k crowdsourced question/answer pairs (Rajpurkar et al., 2016)). Given a question and a passage from Wikipedia containing the answer, the task is to predict the answer text span in the passage. In V2.0, it is possible that no short answer exists in the passage (unlike in V1.0 where this is not possible).

[4]seeing App. A.1 to intuitively understand specific functions of each frequency ω_i

[5]Namely, $\langle P_{x_1-y_1}, P_{x_2-y_2} \rangle = \langle P_{x_3-y_3}, P_{x_4-y_4} \rangle$ if $(x_1-y_1)-(x_2-y_2) = (x_3-y_3)-(x_4-y_4) = m$, in both $x - y > 0$ and $x - y < 0$.

Pre-training The pre-trained "BERT-base-uncased" checkpoint (Devlin et al., 2018) [6] is used to train by replacing the original absolute PE module with a new PE variant (including APEs and RPEs). We train the new models with sequence length of 128 for 5 epochs and then 512 for another 2 epochs. The training is the same as in the original BERT, i.e., wiki and google books (16G raw documents) with whole word masking, which is much less than RoBERTa (Liu et al., 2019). To be fair, the BERT-based-uncased is also trained in the same way. All models have about 110M parameters corresponding to a typical *base* setting, with minor differences solely depending on the parameterization of PEs (see Tab. 1.)

Finetuning The finetuning on GLUE and SQuAD is the same as in the huggingface website [7] as per Wolf et al. (2019), see App. A.3 for details. We report the average values of five runs per dataset.

Table 2: Experiments on GLUE. The evaluation metrics are following the official GLUE benchmark (Wang et al., 2018). The best performance of each task is bold.

model	single sentence CoLA acc	SST-2 acc	sentence pair MNLI acc	MRPC F1	QNLI acc	QQP F1	RTE acc	STS-B spear. cor.	WNLI acc	mean ± std
BERT without PE	39.0	86.5	80.1	86.2	83.7	86.5	63.0	87.4	33.8	76.6 ± 0.41
fully learnable (BERT-style) APE	60.2	**93.0**	84.8	89.4	88.7	87.8	65.1	**88.6**	37.5	82.2 ± 0.30
fixed sin. APE	57.1	92.6	84.3	89.0	88.1	87.5	58.4	86.9	45.1	80.5 ± 0.71
learnable sin. APE	56.0	92.8	84.8	88.7	88.5	87.7	59.1	87.0	40.8	80.6 ± 0.29
fully-learnable RPE	58.9	92.6	84.9	90.5	88.9	88.1	60.8	88.6	50.4	81.7 ± 0.31
fixed sin. RPE	60.4	92.2	84.8	89.5	88.8	88.0	62.9	88.1	45.1	81.8 ± 0.53
learnable sin. RPE	60.3	92.6	**85.2**	90.3	**89.1**	**88.1**	63.5	88.3	49.9	82.2 ± 0.40
fully learnable APE + fully-learnable RPE	59.8	92.8	85.1	89.6	88.6	87.8	62.5	88.3	51.5	81.8 ± 0.17
fully learnable APE + fixed sin. RPE	59.2	92.4	84.8	89.9	88.8	87.9	61.0	88.3	48.2	81.5 ± 0.20
fully learnable APE+ learnable sin. RPE	**61.1**	92.8	**85.2**	**90.5**	89.5	87.9	**65.1**	88.2	49.6	**82.5** ± 0.44
learnable sin. APE + fully-learnable RPE	57.2	92.7	84.8	88.9	88.5	87.8	58.6	88.0	51.3	80.8 ± 0.44
learnable sin. APE + fixed sin. RPE	57.6	92.6	84.5	88.8	88.6	87.6	63.1	87.4	48.7	81.3 ± 0.43
learnable sin. APE + learnable sin. RPE	57.7	92.7	85.0	89.6	88.7	87.8	62.3	87.5	**50.1**	81.4 ± 0.33

Table 3: Performance (average and standard deviation in 5 runs) on *dev* of SQuAD V1.1 and V2.0. † indicates stat. significance over *fully learnable APEs* using a two-sided test with p-value 0.05.

model	SQuAD V1.1 F1	SQuAD V1.1 EM	SQuAD V2.0 F1	SQuAD V2.0 EM
BERT without PE	36.47 ± 0.19	24.24 ± 0.33	50.48 ± 0.12	49.30 ± 0.14
fully learnable (BERT-style) APE	89.44 ± 0.08	81.92 ± 0.11	76.43 ± 0.63	73.07 ± 0.63
fixed sin. APE	89.45 ± 0.07	81.93 ± 0.11	76.12 ± 0.48	72.75 ± 0.55
learnable sin. APE	89.65† ± 0.11	82.24† ± 0.17	77.24 ± 0.43	73.93 ± 0.44
fully-learnable RPE	90.50† ± 0.08	83.38 † ± 0.11	79.85† ± 0.27	76.68† ± 0.49
fixed sin. RPE	90.30† ± 0.07	83.24 † ± 0.08	78.76† ± 0.29	75.38† ± 0.28
learnable sin. RPE	90.45† ± 0.11	83.49 † ± 0.14	79.40† ± 0.37	76.14† ± 0.33
fully learnable APE + fully-learnable RPE	90.57† ± 0.04	83.45 † ± 0.10	**80.31**† ± 0.10	76.94† ± 0.20
fully learnable APE + fixed sin. RPE	90.24† ± 0.17	83.06 † ± 0.21	78.74† ± 0.50	75.40† ± 0.52
fully learnable APE+ learnable sin. RPE	89.56 ± 0.28	82.26 † ± 0.30	77.82† ± 0.42	74.51† ± 0.39
learnable sin. APE + fully-learnable RPE	**90.72**† ± 0.13	**83.68** † ± 0.27	80.24 † ± 0.35	**76.98** † ± 0.34
learnable sin. APE + fixed sin. RPE	90.36† ± 0.08	83.25 † ± 0.10	78.81† ± 0.33	75.71† ± 0.28
learnable sin. APE + learnable sin. RPE	90.49† ± 0.14	83.59 † ± 0.14	79.93† ± 0.34	76.69† ± 0.39

4.1 EXPERIMENTAL RESULTS

GLUE Tab. 2 shows that fully-learnable APEs (a.k.a, BERT-style APEs) perform well. No PE variants outperform notably BERT-style APEs. Adding learnable sinusoidal does not always improve RPEs, but adding BERT-style APEs could consistently boost the performance of RPEs.

SQuAD Tab. 3 shows that nearly all BERT models with RPEs significantly outperform *fully learnable APEs*. Learnable sinusoidal APEs are slightly better than *fully learnable APEs* in most cases. Span prediction tasks do not solely rely on the first token representation `[CLS]` in the last layer, but inference on each token; this may makes span prediction more sensitive to word positions [8].

[6] Downloadable from `https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-12_H-768_A-12.zip`

[7] `https://huggingface.co/`

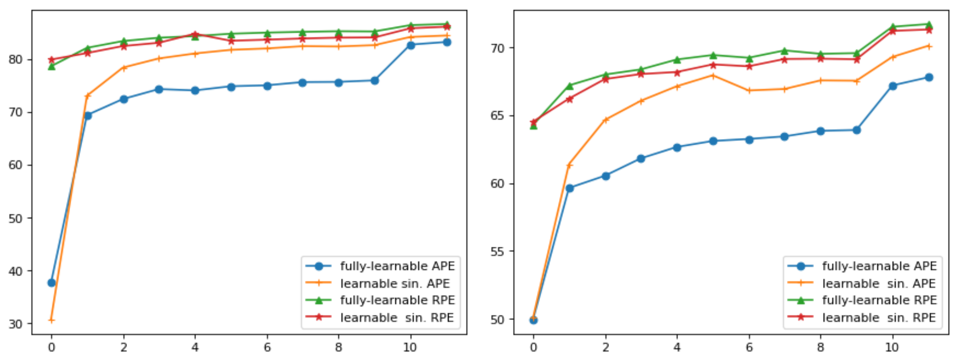
[8] Our empirical findings suggest that classification might be less sensitive to word position than span prediction. The extent to which this is true is subject to more investigation. We see that removing PEs (*BERT*

Learnable sinusoidal PEs Learning frequencies in sinusoidal PEs (learnable sinusoidal APEs/RPEs) outperforms fixed sinusoidal APEs/RPEs in GLUE and SQuADs, showing the expressive power of flexible frequencies.

Complementarity of APEs and RPEs In SQuAD, jointly adopting APEs and RPEs can slightly boost performance in some cases. For instance, *learnable sinusoidal APEs + learnable sinusoidal RPEs* achieve the best score in both SQuADs. However, this complementary effect is relatively weaker in GLUE, where *fully-learnable APEs* perform strongly.

4.2 GENERALIZATION TO LONGER SENTENCES IN DOWNSTREAM TASKS

To fairly compare all models, we train a *medium* setting (8-layer transformer) on 128-length input in the first 10 epochs and 512-length input in the last 2 epochs from scratch. Fig. 1 shows that before 512-length pre-trained (like the 10-th epoch 128-length pre-trained) *learnable sinusoidal APEs* and RPEs perform better than BERT-style (without sinusoidal parameterization) in both SQuADs. This happens because PEs with translation invariance (*learnable sinusoidal APEs* and RPEs) generalize into longer positions, while position vectors between 128-512 positions are not trained in fully-learnable PEs [9].



(a) SQuAD V1.1 (b) SQuAD V2.0

Figure 1: Experimental results on SQuADs with *BERT-medium*. X-axis: epoch number (first trained on 128-length seq. with 10 epochs and then 512-length with 2 epochs). Y-axis: F1 score.

5 ANALYSIS AND DISCUSSION

We present a post-hoc study in Sec. 5.1 and 5.2 and a discussion in Sec. 5.3.

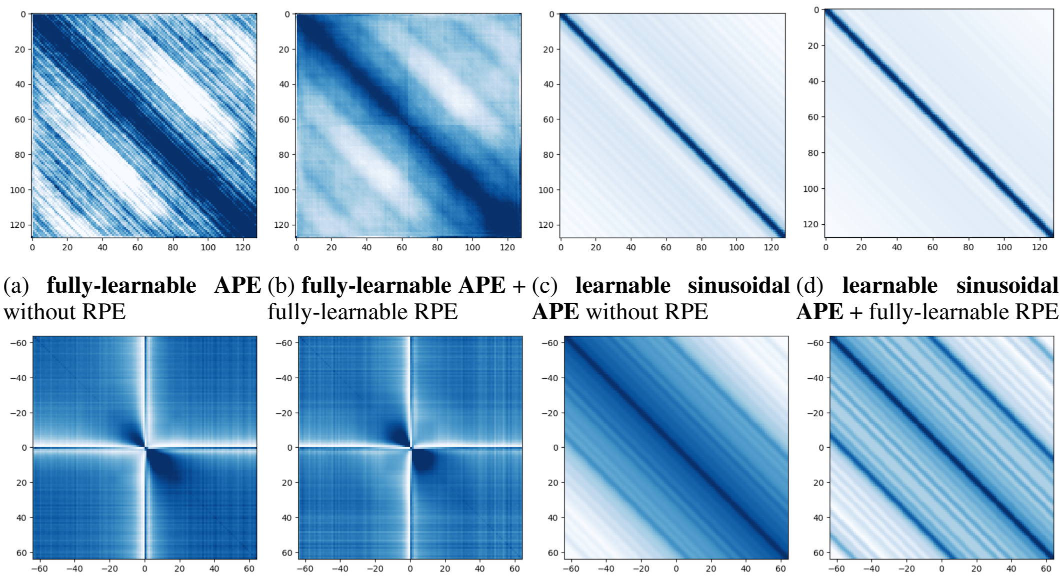
5.1 DOT PRODUCT BETWEEN POSITION VECTORS

We calculate dot products between two arbitrary position vectors for APEs and RPEs (Fig. 2). For APEs, neighboring position vectors are generally closer compared to far-way ones. This trend is clearer in the *learnable sinusoidal APEs*, which impose a strict sinusoidal regularization for PEs. Note that additionally adopting RPEs does not affect too much PE patterns, as can be seen by comparing Fig. 2(a) and 2(b), or Fig. 2(c) and 2(d).

For RPEs, in a **fully-learnable RPE** setting, forward RPEs (from 1 to 64) and backward RPEs (from -64 to -1) are not close to each other (see the slightly bright parts in the top-right and bottom-left). This means that *fully-learnable RPEs* can significantly distinguish forward and backward

without PEs) dramatically harms SQuAD V1.1 and V2.0. The drop in *BERT without PEs* in SQuAD V2.0 is slightly smaller than in SQuAD V1.0 since directly assigning no answer could provide a better lower bound for performance in SQuAD V2.0.

[9]In practice, the document length of some tasks, like summarization, document-level translation, etc. may be much longer than the maximum length typical BERT models can deal with, i.e., 512. Then, learnable sinusoidal PEs or RPEs would be beneficial. Note that they also save parameters compared to typical BERT models, especially when document length is very long like (Beltagy et al., 2020).



(a) **fully-learnable APE** without RPE (b) **fully-learnable APE** + fully-learnable RPE (c) **learnable sinusoidal APE** without RPE (d) **learnable sinusoidal APE** + fully-learnable RPE

(e) **fully-learnable RPE** without APE (f) fully-learnable APE + **fully-learnable RPE** (g) **learnable sinusoidal RPE** without APE (h) fully-learnable APE + **learnable sinusoidal RPE**

Figure 2: Dot products between absolute position vectors [10](top row) and relative position vectors (bottom row). Darker means the two position vectors are closer.

position shifts, which cannot be handled by **sinusoidal RPEs**. This happens because sinusoidal RPEs indistinguishably parameterize forward and backward relative positions with a translation invariance in RPEs. We conclude that sinusoidal parameterization of RPEs in (Wei et al., 2019) may be defective, although sinusoidal RPEs can also express some asymmetry when query/value transformation is considered in the probing test (in Sec. 5.2). Lastly, note that **fully-learnable RPEs** also do not significantly distinguish far-distant RPEs (from -64 to -20 and from 20 to 64), suggesting that truncating RPEs into a distance of 64, like (Shaw et al., 2018), is reasonable.

5.2 Identical word probing

In Fig. 3, *BERT without PEs* nearly treats all words uniformly (bag-of-words). Almost all APEs (including BERT-style APEs and learnable sinusoidal APEs) and RPEs have a clear pattern of translation invariance, local monotonicity in a neighboring window, and symmetry. Note that this is nontrivial since no specific constraints or priors were imposed on fully-learnable APEs/RPEs, seeing App. A.4 for an example of the evolution in fully learnable APEs. Note that 64-truncated RPEs will have some unexpected patterns with far-distancing positions, which is assumed to be insensitive to the modeling. BERT with *learnable sinusoidal RPEs* generally attends more on forwarding tokens than backward tokens.

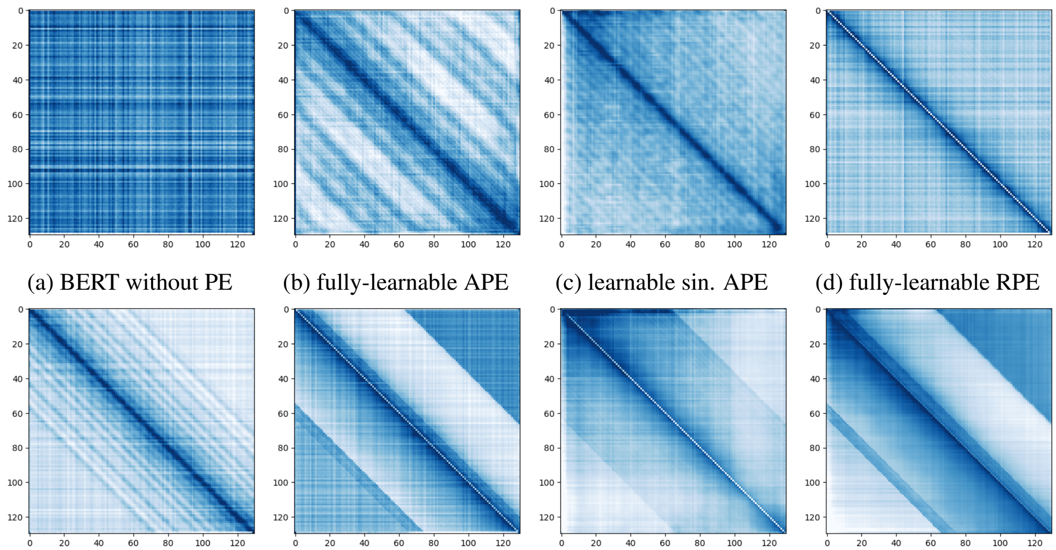
5.3 Discussion of the proposed properties in downstream tasks

Monotonicity Monotonicity holds locally in a small neighboring window (usually in 5-20 offsets) for all PE variants, see Fig.3. In Transformers, where attention calculation does not consider word order, modeling neighboring words is crucial. To check monotonicity guided by learned frequencies of learnable sinusoidal APEs in individual tasks, see App. A.5.

Translation invariance In pre-trained language models (especially BERT), we argue that absolute positions of words are uninformative since 1) absolute positions of the second segment [11] depend on

[10] In all figures we only show the first 128 positions instead of 512 positions since they are in principle compatible. There is a minor discrepancy between the first 128 positions and the remaining positions due to the typical training strategy (first training on 128-length input and then 512-length input).

[11] *Segment* is used in BERT to recognize different sentences, especially in the next sentence prediction task.



(a) BERT without PE (b) fully-learnable APE (c) learnable sin. APE (d) fully-learnable RPE

(e) fixed sin. RPE (f) learnable sin. RPE (g) learnable sin. APE + fully-learnable RPE (h) learnable sin. APE + learnable sin. RPE

Figure 3: Dot products between relative position vectors. Darker means that vectors are closer.

the length of the first sentence; 2) words are randomly truncated in the beginning or end if a sentence exceeds the expected maximum length, which may shift absolute positions of all tokens with an unexpected offset (Devlin et al., 2018). That is, absolute positions of words in pre-trained language models are arbitrarily replaceable, thus adopting translation invariance is generally reasonable.

Sinusoidal PEs and RPEs follow a strict translation invariance during paramerization; fully-learnable APE empirically meets translation invariance but strictly. When finetuning, especially in classification tasks like GLUE, the special token `[CLS]` (always in first position) is crucial for inference, since only its corresponding output in the last layer is used for prediction in classification. Thus, the first position with `[CLS]` is more informative than the other positions and should be separately treated. Fully-learnable APEs do not strictly meet translation invariance, and could flexibly deal with fixed-position `[CLS]`. PEs with translation invariance (like sinusoidal PEs and RPES) cannot treat the irreplaceable first position (`[CLS]` token) and remaining positions separately, making it difficult to make `[CLS]` an overall representation and potentially harming classification performance. In span prediction, we empirically see that having a strict translation invariance (like in sinusoidal PEs and RPES) allows PEs o better perceive word order. Overall, models with strict translation invariance (all RPEs and sinusoidal APEs) could generalise PEs to longer documents, since this same pattern can be repeated with translation invariance. The empirical evidence can be found in Sec. 4.2.

Symmetry APEs nearly express symmetry patterns without distinguishing the direction between positions as shown in Fig 3. RPEs can to some extent have an asymmetry, as shown in Fig. 2(e) and 2(f), and Fig. 3 (f) and (h), where forward and backward relative embeddings are separately embedded. This may be an advantage of RPEs over APEs to perceive forward and backward words, especially in span prediction tasks where capturing this matters.

6 Conclusion

To formally understand position embeddings (PEs), we define three properties (translation invariance, monotonicity, and symmetry) inspired by distance mapping between the original domain of positions in $\mathbb{N}$ and their PEs in $\mathbb{R}^D$. Using these properties, we examine various existing PEs. Especially, to flexibly deal with monotonicity, we propose to learn frequencies in sinusoidal PEs. Our post-hoc study shows these PEs nearly meet most properties even when they are fully-learnable without constraints. We also evaluate various PEs in classification and span prediction and find that fully-learnable absolute PEs are better for classification and relative PEs are better for span prediction tasks, which can be explained by the mismatch between their properties and task characteristics.

REFERENCES

George B Arfken and Hans J Weber. Mathematical methods for physicists, 1999.

Mihai Badoiu, Erik D. Demaine, MohammadTaghi Hajiaghayi, Anastasios Sidiropoulos, and Morteza Zadimoghaddam. Ordinal embedding: Approximation algorithms and dimensionality reduction. In Ashish Goel, Klaus Jansen, José D. P. Rolim, and Ronitt Rubinfeld (eds.), Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, volume 5171 of Lecture Notes in Computer Science, pp. 21–34. Springer, 2008.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150, 2020.

Yonatan Bilu and Nathan Linial. Monotone maps, sphericity and bounded second eigenvalue. J. Comb. Theory, Ser. B, 95(2):283–299, 2005.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert's attention. arXiv preprint arXiv:1906.04341, 2019.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

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

Lalit Jain, Kevin G. Jamieson, and Robert D. Nowak. Finite sample prediction and recovery bounds for ordinal embedding. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (eds.), Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems, pp. 2703–2711, 2016.

Guolin Ke, Di He, and Tie-Yan Liu. Rethinking positional encoding in language pre-training. arXiv preprint arXiv:2006.15595, 2020.

Xuanqing Liu, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. Learning to encode position for transformer with continuous dynamical model. arXiv preprint arXiv:2003.09229, 2020.

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. arXiv preprint arXiv:1907.11692, 2019.

Hiroshi Maehara. Euclidean embeddings of finite metric spaces. Discrete Mathematics, 313(23): 2848 – 2856, 2013.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250, 2016.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. arXiv preprint arXiv:2002.12327, 2020.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. arXiv preprint arXiv:1803.02155, 2018.

Roger N Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function. Psychometrika, 27(2):125–140, 1962.

Yoshikazu Terada and Ulrike Luxburg. Local ordinal embedding. volume 32 of Proceedings of Machine Learning Research, pp. 847–855. PMLR, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pp. 5998–6008, 2017.

Jesse Vig. A multiscale visualization of attention in the transformer model. arXiv preprint arXiv:1906.05714, 2019.

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. arXiv preprint arXiv:1804.07461, 2018.

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.

Junqiu Wei, Xiaozhe Ren, Xiaoguang Li, Wenyong Huang, Yi Liao, Yasheng Wang, Jiashu Lin, Xin Jiang, Xiao Chen, and Qun Liu. Nezha: Neural contextualized representation for chinese language understanding. arXiv preprint arXiv:1909.00204, 2019.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing. ArXiv, abs/1910.03771, 2019.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Self-attention with functional time representation learning. In Advances in Neural Information Processing Systems, pp. 15915–15925, 2019.

Hang Yan, Bocao Deng, Xiaonan Li, and Xipeng Qiu. Tener: Adapting transformer encoder for name entity recognition. arXiv preprint arXiv:1911.04474, 2019.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In Advances in neural information processing systems, pp. 5753–5763, 2019.

A Appendix

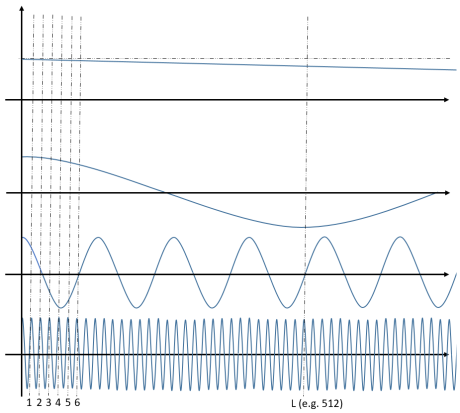
A.1 Understanding individual frequency

We argue in this paper that a learning schema for such frequencies will be useful in a sense it could adaptively adjust frequencies to meet different functions, seeing Fig. 4.

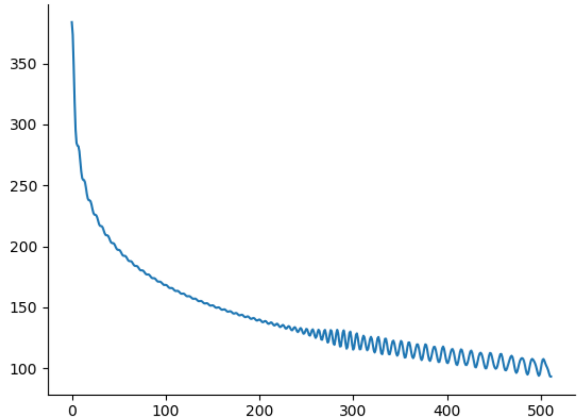
A.2 Expressive Power of LEARNABLE SINUSOIDAL PEs

In Transformers, linear transformation is commonly-used, for example *query*, *key*, and *value* transformations on word representations. Let $\boldsymbol{r}_i$ be the word representation paramertezied by the sum of word embeddings and position embeddings (like the learnable sinusoidal APEs). Then, each element in $\boldsymbol{x}_i$

$$r_{i,k}(t) = e_{i,k} + p_k(t) = \begin{cases} e_{i,k} + \sin(\omega_{\frac{k}{2}} t), & \text{if } k \text{ is even} \\ e_{i,k} + \cos(\omega_{\frac{k}{2}} t), & \text{if } k \text{ is odd} \end{cases} \tag{6}$$



(a) Examples of some cosine functions



(b) $\phi(m)$, a sum of cosine functions with frequencies $\omega_i = (1/10000)^{2i/D}$.

Figure 4: $\phi(m)$ in (b) is a sum of many cosine functions of individual frequencies with increasing m, which determine the closeness between arbitrary two m-distance position vectors. As shown in (a), each frequency could play different roles: 1) the extremely small frequencies has few effects on the overall word representation ($WE_i + p_i$ in Eq. 4 since it makes such position embedding being almost identical with increasing positions; 2) some smaller frequencies can be beneficial to guarantee Property 1 if $\omega_i < \frac{\Pi}{L}$; 3) some bigger frequencies would promote the locally attending mechanism since such $\cos$ functions in Eq. 3.2 drop dramatically in the beginning if ω_i is great enough; 4) Some big frequencies which $\omega_i > \Pi$ would be smooth factors for the overall pattern since it would be randomly impose a bias to all positions.

After a linear transformation parameterized by $\boldsymbol{w}$ (for example, the key transformation W^K in the first Transformer layer), $\boldsymbol{r}_i$ is linearly transformed as $\boldsymbol{h}_i(t) = \boldsymbol{w}\boldsymbol{r}_i$ ($\boldsymbol{h}_i(t)$ can be one of query/key/value vectors Q_x, K_x, V_x in t-th position) with each element

$$h_{i,k}(t) = \sum_{k}^{D} w_{j,k} e_{i,k} + \sum_{k}^{D/2} \left(w_{j,2k} \sin(\omega_{2k} t) + w_{j,2k+1} \cos(\omega_{2k+1} t) \right) \tag{7}$$

The RHS is a typical Fourier series with a base term $\sum w_{j,k} e_{i,k}$ and Fourier coefficients $\{w_{j,2k}, w_{j,2k+1}\}$. It is customarily assumed in Physics and Signal Processing (Arfken & Weber, 1999) that the RHS in Eq. 7 with infinite D and appropriate frequencies could approximate any continuous function on a given interval.

Since infinite D is impossible in practice, dynamic allocation of a limited number of frequencies in a data-driven way could be beneficial for general approximation. The predefined frequencies $\omega_i = (1/10000)^{2i/D}$ in the Transformer (Vaswani et al., 2017) can be considered as a special case when it enumerates various frequencies ranging from $1/10000$ to 1 under a specific distribution.

A.3 Detailed Experimental Setting

we train BERT base and BERT medium with both masked language prediction and next sentence prediction tasks. Some parameters are listed in the Tab. 4. Other parameters are following the original paper. Note that we share RPE in different heads and layers. Like (Shaw et al., 2018) RPE are truncated from -64 to 64.

We run five runs for SQuAD and GLUE benchmark. The results in GLUE are for the last checkpoint during finetuning while SQuAD takes the best one for every 1000 steps. Finally, we calculate the average over 5 runs. All these setting are the same for all PEs. We use Mismatched MNLI. In GLUE (Wang et al., 2018), the train and dev are somewhat adversarial: training samples (in train and dev) containing the same sentence usually have opposite labels. Models may get worse when it overfits in train set, resulting in unexpected results. Therefore, we exclude WNLI to calculate average in

Training	Max Length	Epoch	continue training	learning rate	batch size	weight decay
BERT-base on 128 length	✓	128	5	5e-5	64	0.01
BERT-base on 512 length	✓	512	2	5e-5	512	0.01
BERT-medium on 128 length	✗	128	10	5e-5	128	0.01
BERT-medium on 512 length	✗	512	2	5e-5	512	0.01
GLUE	-	128	3	2e-5	32	0.01
SQuAD	-	384	3	3e-5	32	0.01

Table 4: Detailed Experimental Setting

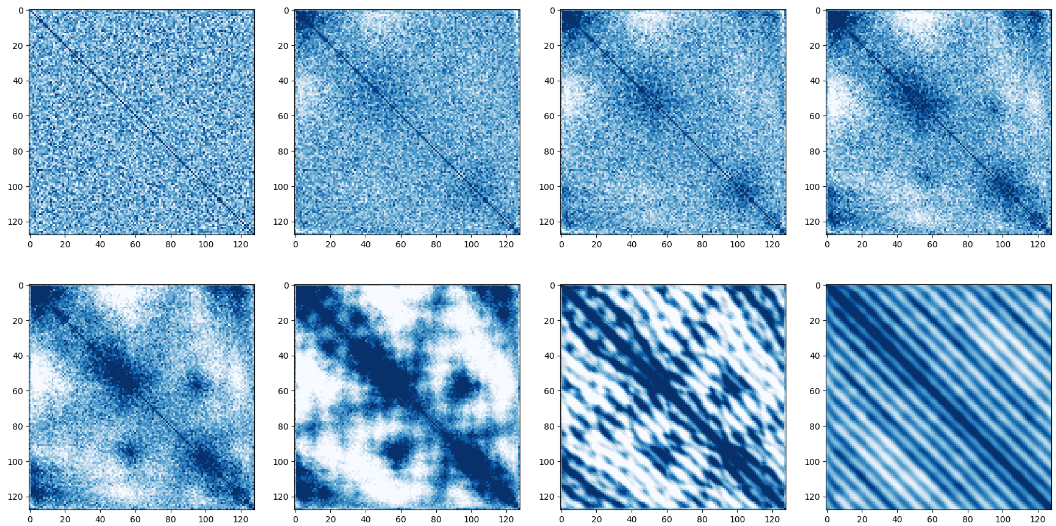


Figure 5: Dot products between absolute position vectors evoling with traing steps.

the last column in Tab. 2. The finetunig parameters are using default values in Huggingface project Wolf et al. (2019).

A.4 The evolution of proximity between Position embeddings

We show a dot product between position vectors during training a BERT-medium, as shown in Fig. 5. It is in chaos at the beginning and then turns to have a regular pattern with translation invariance and local monotonicity.

A.5 Learned frequencies of Learnable sinusoidal APE in different tasks

As shown in Fig.6 finetuned models (including SQuAD and SQuAD2) in span prediction tasks have a strict monotonicity in a longer widows like 18 words, while classification have a strict monotonicity in a longer widows like 12 words. Note that the patterns in pretraining language models are more close to classification tasks than span prediction tasks.

A.6 Discussions on related works

Complementary between APE and RPE Ke et al. (2020) propose that combining APE and RPE could be beneficial for classification tasks (GLUE), which in this paper, this complementary effect is not significant since most of PE combinations (APE and RPE) do not outperform the BERT-style fully-learnable APE on classification. Instead, we empirically conclude that most of PE combinations boost the performance in span prediction tasks. The benefit in classification tasks in (Ke et al., 2020) may come from other modifications. For example, it adopts a special relative position embedding like (Raffel et al., 2019): a simplified form of PE that each "embedding" is simply a scalar bias added to the corresponding logit when computing the attention weights. The fundamental difference between the 'position bias' and position embedding is unknown from now.

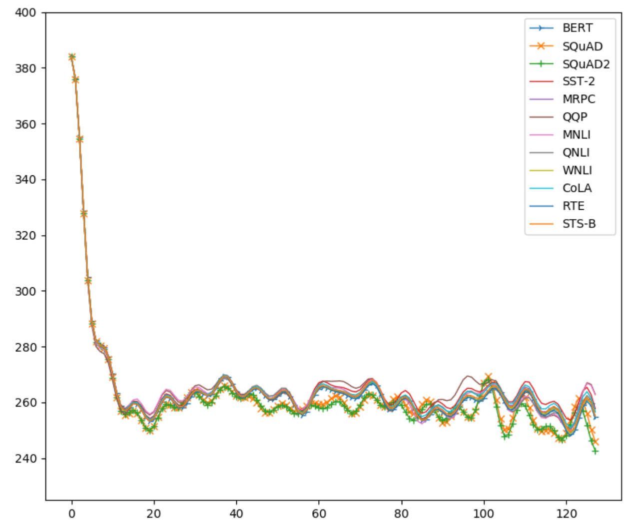
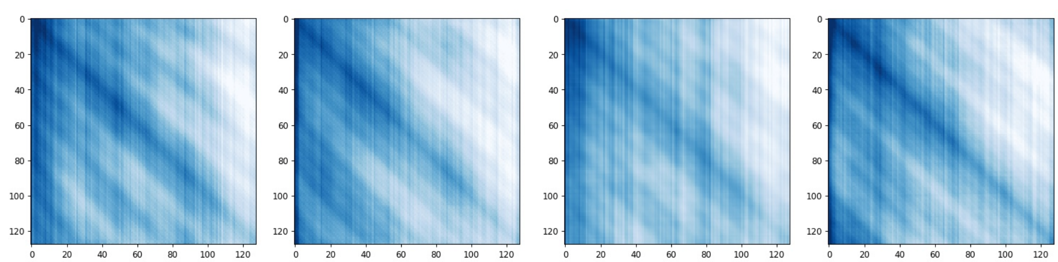


Figure 6: Dot products between two absolute positions with increasing offset, indicates neighboring APES are embedded together. x axe refers to offset between positions.

Study on attention visualization. There are many works focusing on understanding attention patterns in individual head. For example Vig (2019) introduced a tool for visualizing attention in the Transformer at multiple scales; Rogers et al. (2020) suggest attention mechanisms like Vertical, Diagonal, Vertical + diagonal, Block, and Heterogeneous. Clark et al. (2019) found some attention mechanisms like attending broadly, to next, to `[CLS]` or `[SEP]`, attend to punctuation. While our paper focuses the general attention introduced by PEs from an average point of view, without considering any specific attention head.

Asymmetry in sequential labeling Yan et al. (2019) suggested asymmetry of position embedding in Named-entity recognition task (without involving pre-trained language models) which is a kind of sequential labeling tasks like span prediction (SQuAD) in this paper. Their conclusion is generally compatible with ours, but we question its assumption that 'the property of distance-awareness disappears when query and key projection are conducted'. As shown in Fig. 7, we could slightly see some distance-awareness.



(a) bert-base-cased (b) bert-base-uncased (c) bert-large-cased (d) bert-large-uncased

Figure 7: Position-wise correlation matrix ($PW^{Q,1}(W^{K,1})^T P^T$) for first 128 positions.

Functional parameterization of PEs Xu et al. (2019) proposes various variants of sinusoidal positional encodings inspired by functional analysis. Liu et al. (2020) use ODE to parameterize position encoding as a continuous dynamical model. Wang et al. (2020) proposed a sinusoid-like complex word embedding to encode word order. All of these papers are inspiring. Since selecting suitable parameterization type is not the main concern, we instantly take the typical sinusoidal PE.