UNDERSTANDING THE ROLE OF POSITIONAL ENCODINGS IN SENTENCE REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Positional encodings are used to inject word-order information into transformer-based language models. While they can significantly enhance the quality of sentence representations, their specific contribution to language models are not fully understood, especially given recent findings that building natural-language understanding from language models with positional encodings is insensitive to word order. In this work, we investigate the role of positional encodings systematically. (1) We uncover that the core function of existing positional encodings is to symmetrically combine local units by identifying two common properties, Locality, and Symmetry. (2) We reveal that positional and contextual encodings play a distinct role in understanding sentences. (3) Based on these findings, we propose a simplified method to inject positional information into such models. Empirical studies demonstrate that this method can improve the performance of the BERT-based model on 10 downstream datasets. We hope these new probing results and findings can shed light on how to design and inject positional encodings into language models.

1 INTRODUCTION

Transformer-based language models have boosted the development of sentence representations. The cornerstone of their architectures is the self-attention mechanism (Parikh et al., 2016; Seo et al., 2016; Shen et al., 2018). Though word order is important to understand sentences, in particular in natural language (Wang et al., 2019c), self-attention does not intrinsically capture order in a sequence. Therefore, specific methods are used to infuse positional information into transformer-based models. For this, many approaches use Positional Encodings (PEs), either as Absolute Positional Encodings (APEs) (Vaswani et al., 2017; Devlin et al., 2019) or as Relative Positional Encodings (RPEs) (Shaw et al., 2018). Recent research has been remarkably vibrant, with the emergence of diverse novel and promising variants (Dai et al., 2019; Wang et al., 2019a; He et al., 2020; Ke et al., 2021; Press et al., 2021; Su et al., 2021; Dufter et al., 2022). Other approaches propose additional word order-related tasks for pre-trained language models. StructBERT (Wang et al., 2019b) proposes a structural objective, forcing the model to reconstruct the right order of words. T5 (Raffel et al., 2019) considers a basic deshuffling objective to restore the original order of a shuffled sequence.

Language models that have been prepared with PEs or word-ordering tasks can improve the performance considerably across several natural language tasks. Hence, one might expect that these models capture word order information in sentences well. However, some recent studies have revealed counter-intuitive findings: *language models with positional encodings are surprisingly insensitive to word orders* (Pham et al., 2021; Sinha et al., 2021; Gupta et al., 2021; Abdou et al., 2022) for natural language understanding tasks like GLUE (Wang et al., 2018). For example, a fine-tuned BERT can correctly know that the following sentence pair from SNLI set satisfies the entailment relation:

a. *A man playing an electric guitar on stage.* b. *A man playing guitar on stage.*

However, if we change the word order of the premise sentence so that it becomes "*an electric guitar playing a man on stage.*", BERT still finds that the premise entails the hypothesis. These somewhat counter-intuitive observations raise a question: "***How do positional encodings in language models capture word order?***"

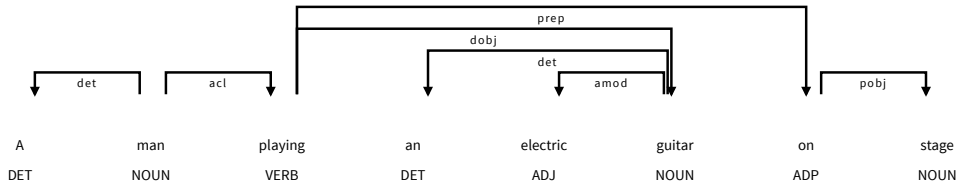

Figure 1: The dependency parsing result for the sentence *A man playing an electric guitar on stage*. We use the spaCy tool to parse and visualize.

To answer this question, we conduct a series of probing (or diagnostic) analyses to uncover the core role of positional encodings in sentence representations. We visualize the learned positional weights of various pre-trained language models by using Identical Word Probing (Wang et al., 2020). Although these models use different ways to inject positional information, we observe two common properties for their positional weights: Locality and Symmetry (as shown in Figure 2). Locality means the weights are mostly on the diagonal of positional matrices while Symmetry means these weight matrices are highly symmetrical. To confirm these two properties, we propose two probing tasks, *Word Swap Probing* and *Handcrafted Positional Encoings Probing*, which reveal that the core function of existing positional encodings is to symmetrically combine local units.

Although some research demonstrates that pre-trained language models store quite some linguistic knowledge, e.g., subject-verb agreement and dependency tree (Clark et al., 2019; Jawahar et al., 2019; Lin et al., 2019; Rogers et al., 2020), the linguistic roles of positional encodings have not yet been studied. Therefore, a natural question is: "***What syntactic knowledge is stored in positional encodings?***" To address this question, we obtain solely PEs-dependent sentence representations by removing the contextual encodings in pre-trained language models, and evaluate their linguistic capabilities, e.g., the ablity to predict the number of subjects and the sensitivity to word order, on 10 probing tasks proposed by Conneau et al. (2018). Besides, we assess the knowledge of syntax dependency captured by the positional correlation[1] in attentional heads. Our results show that there is a clear distinct role between the positional and contextual encodings in sentence comprehension: positional encodings play more of a role at syntactic level while contextual encodings serve more at semantic level (if the semantic task does not require word order information), and the combination of the two can consistently yield better performances on these probing tasks. As for the dependency relations, positional weights capture more short-distance dependencies while contextual weights capture more long-distance ones.

Finally, we find that the established methods of injecting positional encodings into the self-attention, either absolute or relative, all add positional weights to the contextual weights simultaneously (Equation 9). However, considering the core function of positional encodings and the different roles of the positional and contextual weights, another key question arises: "***How to combine the positional and contextual encodings?***" For example, Figure 1 shows the dependency parsing tree of the sentence *A man playing an electric guitar on stage*. Considering the word "*guitar*", we generally prioritize the use of the neighboring adjective "*electric*" (`amod`) to compose a phrase semantically rather than the distant verb "*playing*" (`dobj`), though both words have a direct dependency on "*guitar*". This example shows that there is a certain order of relation in the use of positional and contextual encodings. Inspired by this, we propose a sequential way to combine positional and contextual features, which is a simple yet effective way to inject positional encodings to language models. Experimental results show that our proposed method can bring improvements across 10 sentence-level downstream tasks. We hope these new probing analysis can shed light on the design and usage of positional encodings.

2 Positional Encodings Enforce Locality and Symmetry

Recent work has shown that natural language understanding built on language models with positional encodings is insensitive to word order (Pham et al., 2021; Sinha et al., 2021; Abdou et al., 2022). To better understand why this is the case, we will now visualize the positional weights of eight pre-trained language models (eight weight matrices). This reveals two properties that are

[1]The contextual correlation represents the weight in attentional heads obtained by the only input content encodings while the positional correlation represents weight obtained by positional encodings.

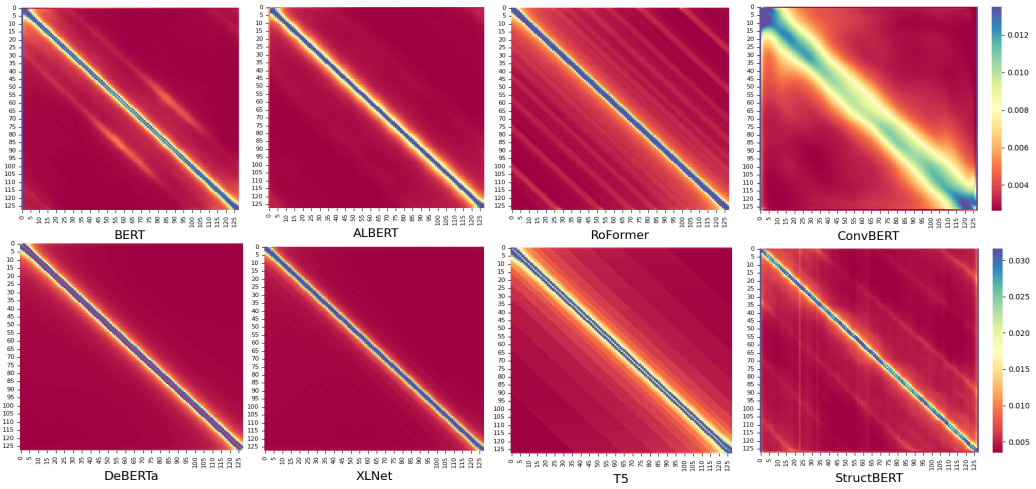

Figure 2: Visualizations of different pre-trained language models by using Identical Word Probing (Wang et al., 2020). The attention weights are averaged across different layers. Roformer is a Chinese version, and the others are English versions.

common to different positional encoding: *Locality* and *Symmetry*. Locality means that positional weights attend to nearby words (Non-zero values gather on the diagonal of matrices) while Symmetry means that weight matrices are highly symmetrical. Therefore, we hypothesize that the core function implemented by positional encodings is to symmetrically combine local units. To investigate this hypothesis, we design two probing tasks, *Word Swap Probing* and *Handcrafted Positional Encoings Probing*. The tasks confirm the hypothesis, which then explains why BERT fails on the sentence "*an electric guitar playing a man on stage*", as described in the introduction.

2.1 Visualizing positional weights in attentional heads

To understand what order information language models learn, we investigate the positional weights in attentional heads. We adapt the visualizations of Wang & Chen (2020); Wang et al. (2020), which reveal the position weights assigned to each position in a sequence. Going beyond existing work, we study multiple variants of positional encodings, including absolute and relative ones as well as positional information in models pre-trained by an additional word ordering task. Specifically, we use the following eight language models: BERT (Devlin et al., 2019), ALBERT (Lan et al., 2019), Roformer (Su et al., 2021), ConvBERT (Jiang et al., 2020), DeBERTa (He et al., 2020), XLNet (Yang et al., 2019), T5 (Raffel et al., 2019), StructBERT (Wang et al., 2019b). More details in Section B.1.

Figure 2 shows these weight matrices, where the element at (i, j) indicates the attention weight between the i-th position and the j-th position. Our results generalize observations of Wang & Chen (2020); Wang et al. (2020) across language models and positional-embedding approaches: these attention matrices are all diagonal heavy, which means various positional encodings highly attend nearby positions. Second, all matrices are nearly symmetrical and this phenomenon is inconsistent with our intuition, e.g., the meaning of "*a man playing an electric guitar on stage*" is totally different from "*an electric guitar playing a man on stage*". Third, different language models also have some particular behaviors. For example, the diagonal "band" of ConvBert is wider because it has a span-based dynamic convolution with a kernel size of 9, which expands the range of local constraints. Besides, BERT, DeBERTa and T5 assign less weights to words themselves, as previously observed by Clark et al. (2019); Wang et al. (2020). Based on the above observations, we suggest that the answer to "*why are language models insensitive to word orders*?" is because of two properties common across positional encodings: **Locality** and **Symmetry**.

2.2 Locality and Symmetry

Locality means that the positional weights favor the combination of units in a sentence to their adjacent units when creating higher-level representations. For example, sub-tokens can be composed

into lexical meanings (e.g., {*"context","##ual"*} → "*contextual*") or words can be composed into phrase-level meaning (e.g., {*"take","off"*} → "*take off*"), and clause-level and sentence-level meaning can be obtained through an iterative combination of low-level meanings, which is consistent with the multi-layer structure in pre-trained language models. From a linguistic perspective, words linked in a syntactic dependency should be close in linear order, which forms what can be called a dependency locality (Futrell et al., 2020). Dependency locality provides a potential explanation for the formal features of natural language word order. Consider the two sentences *"John throws out the trash"* and *"John throws the trash out"*. Both are grammatically correct. There is a dependency relationship between *"throws"* and *"out"* and the verb is modified by the adverb. However, language users prefer the expression with the first sentence because it has a shorter total dependency length (Dyer, 2017; Liu et al., 2017; Temperley & Gildea, 2018). Based on the visualizations and dependency locality, we, therefore, speculate that one main function that positional encodings have learned during pre-training is local composition, which exists naturally in our understanding of sentences. Empirical studies also demonstrate that performances of shuffled language models are correlated with the violation of local structure (Clouatre et al., 2022).

The *symmetry* (also observed by Wang & Chen (2020); Wang et al. (2020)) of the positional matrices implies that the contributions of forward and backward sequences are equal when combining adjacent units under the locality constraint. This is contrary to our intuition, as the forward and backward tokens play different roles in the grammar, as we have seen in the examples of "*a man playing an electric guitar on stage*" and "*an electric guitar playing a man on stage*". However, this symmetry is less disruptive at the local level inside sentences. Recent work in psycholinguistics has shown that sentence processing mechanisms are well designed for coping with word swaps (Ferreira et al., 2002; Levy, 2008; Gibson et al., 2013; Traxler, 2014). Further, Mollica et al. (2020) hypothesizes that the composition process is robust to local word violations. Consider the following example:

a. *on their last day they were overwhelmed by farewell messages and gifts*

b. *on their last day they were overwhelmed by farewell and messages gifts*

c. *on their last they day were overwhelmed farewell messages by and gifts*

The local word swaps (colored underlined words) are introduced in the latter two sentences, leading to a less syntactically well-formed structure. However, experimental results show that the neural response (fMRI blood oxygen level-dependent) in the language region does not decrease when dealing with word order degradation (Mollica et al., 2020), suggesting that human sentence understanding is robust to local word swaps. Likewise, symmetry can be understood in this way: when a reader processes a word in a sentence, the forward and backward nearby words are the most combinable, and the comprehension of this composition is robust to its inside order. On the other hand, symmetry is not an ideal property for sentence representations (consider the case of "*an electricity guitar*"), and we will show the flaws of symmetry in the word swap probing task in the next section.

2.3 Handcrafted Positional Encodings

Given that locality and symmetry stand out as important features of existing positional encodings, it begs the question whether these properties are sufficient for current language understanding with language model. We use manually-designed positional encodings to approximate these two properties. For this purpose, we replace the positional correlations δ_{ij} in Equation 9 with two handcrafted Positional Encodings to probe: *(1) Identical Encoding:* As in convolutional operations, one option is to pay equal attention to the words within a window size of the current token (Equation 12). *(2) Attenuated Encoding:* Unlike with identical encodings, a gradual decrease of positional weight with the relative distance seems preferable. For this, Gaussian Transformers (Guo et al., 2019) use a Gaussian kernel (Equation 13).

Our two handcrafted positional encodings perfectly satisfy the properties of locality and symmetry, as shown in Figure 6 (the left two). Besides the positional weights are frozen during training and shared in all blocks. If these simple manually-designed positional encodings can achieve similar performance as the learned positional encodings, it shows that the function of positional encodings actually implement for language understanding is the symmetric combination of local units. An code example of how to inject these handcrafted positional encodings is shown in Appendix (Listing 1)

Model	Original	Shuffle-3 (Δ)	Shuffle-4 (Δ)	Shuffle-5 (Δ)	Shuffle-6 (Δ)	Random (Δ)	Original	Shuffle-SR (Δ)
BERT	89.8	-0.4	-0.6	-0.3	-0.5	-2.7	89.8	-63.9
ALBERT	91.8	-0.5	-1.1	-1.3	-1.7	-6.0	92.0	-66.8
ConvBERT	91.6	-0.8	-0.9	-1.6	-1.3	-3.9	91.4	-52.9
DeBERTa	91.6	-0.5	-0.7	-1.3	-1.1	-5.1	91.6	-58.9
XLNet	91.5	-0.2	-0.3	-0.7	-1.2	-5.4	91.3	-57.8
StrucBERT	90.9	-0.5	-0.9	-1.3	-1.1	-4.4	90.8	-44.6

Table 1: Results of Word Swap Probing, measured by accuracy. Shuffle-x means phrases with length x are shuffled. Shuffle-SR means the semantic roles of agent and patient are swapped.

2.4 Probing Tasks of Locality and Symmetry

To confirm our hypothesis that positional embeddings enforce Locality and Symmetry, we run two experiments: *Word Swap Probing* and *Handcrafted Positional Encodings Probing*.

2.4.1 Word Swap Probing

If the core properties of positional encodings are locality and symmetry, then the performances will not be influenced when we only corrupt the word order inside the local structure. For example, we can disrupt the inside order of a constituent, which is formed by the combination of adjacent units. A constituent parsing case is shown in Figure 3, and we can shuffle the word order inside some phrases, e.g., the noun phrase "*an electric guitar*" while the semantic will not be changed (the grammar structure may be destroyed). After, we probe if the prediction of a language model has changed. Specifically, the Stanford Natural Language Inference (SNLI) (Bowman et al., 2015) dataset is used in this experiment and it provided constituent structure for each sentence. In SNLI, every sample is pair of sentences annotated by three relations: Neutral, Entailment and Contradiction. We select five types of target phrases for shuffling: *Noun Phrase, Verb Phrase, Prepositional Phrase, Adverb Phrase, and Adjective Phrase*. We construct different shuffled datasets by phrase length, e.g., "*an electric guitar*" is a phrase of length 3 and we can obtain a 3-gram shuffled SNLI by only shuffling phrases whose length is 3. For a fair comparison, a subset of the SNLI test set (around 5000 samples) is used to ensure every sentence contains from 2-gram to 6-gram phrases, and some examples are shown in Table 5. After, various language models are fine-tuned on the train set and then evaluated on the original test set and shuffled sets, respectively. We refer to this dataset as shuffle-x and x means the length of phrase.

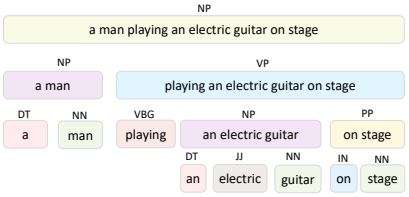

Figure 3: Illustration of constituent parsing for one sentence in SNLI "*a man playing an electric guitar on stage*". The result is generated by Berkeley Neural Parser.

On the other hand, if the global structure of a sentence is changed (e.g., by swapping the subject and object), then the symmetry property does not imply that the language model will understand the semantics of the sentence, because it is only local. Starting from this point, we use an existing semantic role tool[2] to identify the *agent* and *patient* in a sentence. In figure 3, "*a man*" as the entity that performs the action, technically known as the agent, and "*an electric guitar*" as the entity that is involved in or affected by the action, which is called the patient. In this experiment, the agent and patient are swapped and therefore result in a new sentence with contradictory semantics, e.g., "*an electric guitar playing a man on stage*". Then, we probe the capability of language models to correctly recognize the new sentence as contradicting the original sentence. For this goal, we adapt entailment sentences of the SNLI test set and manually select 300 samples for probing (some examples in Table 5) . We refer to this dataset as Shuffle-SR because it swap the semantic roles in a sentence (More details in Appendix B.2).

To probe the sensitivity of language models with positional encoding to local and global word order swaps, we fine-tune 6 pre-trained language models on SNLI training set and evaluate them on the

[2] https://demo.allennlp.org/semantic-role-labeling/semantic-role-labeling

Model	Size	Sentiment Analysis			Textual Entailment			Paraphrase Identification		Textual Similarity		Avg
		MR (22K)	SUBJ (20K)	SST-2 (68.8K)	QNLI (110K)	RTE (5.5K)	MNLI (413K)	MRPC (5.4K)	QQP (755k)	STS (8.4K)	SICK-R (9.4K)	
BERT	110M	$72.5_{\pm 5.3}$	$91.0_{\pm 2.7}$	$86.4_{\pm 2.7}$	$85.8_{\pm 1.0}$	$59.2_{\pm 1.2}$	$78.2_{\pm 0.8}$	$73.5_{\pm 1.8}$	$88.7_{\pm 0.6}$	$77.8_{\pm 4.1}$	$64.9_{\pm 6.0}$	77.8
BEET-*I*	110M	$71.4_{\pm 4.6}$	$90.0_{\pm 2.3}$	$81.6_{\pm 0.9}$	$83.2_{\pm 0.3}$	$63.4_{\pm 4.6}$	$72.6_{\pm 0.7}$	$74.7_{\pm 2.0}$	$87.2_{\pm 0.5}$	$80.1_{\pm 4.6}$	$66.5_{\pm 3.9}$	77.0
BEET-*A*	110M	$72.0_{\pm 3.6}$	$88.8_{\pm 2.6}$	$82.1_{\pm 1.2}$	$83.0_{\pm 0.2}$	$61.9_{\pm 4.4}$	$71.6_{\pm 1.2}$	$74.1_{\pm 1.7}$	$86.4_{\pm 0.3}$	$80.6_{\pm 3.4}$	$67.6_{\pm 4.2}$	76.8
BEET-A^*-*s*	113M	$79.4_{\pm 2.9}$	$93.7_{\pm 0.6}$	$88.0_{\pm 0.7}$	$86.3_{\pm 1.1}$	$59.4_{\pm 2.7}$	$78.8_{\pm 0.4}$	$81.5_{\pm 2.2}$	$88.7_{\pm 0.4}$	$83.6_{\pm 2.0}$	$76.3_{\pm 1.1}$	81.6
BEET-I^*	138M	$78.3_{\pm 2.6}$	$93.1_{\pm 0.8}$	$87.1_{\pm 0.3}$	$86.7_{\pm 0.5}$	$58.0_{\pm 2.1}$	$78.7_{\pm 3.3}$	$78.7_{\pm 2.0}$	$88.6_{\pm 0.3}$	$86.0_{\pm 1.8}$	$76.3_{\pm 4.1}$	81.5
BEET-A^*	138M	$78.2_{\pm 3.5}$	$93.0_{\pm 0.8}$	$88.1_{\pm 1.0}$	$87.0_{\pm 0.5}$	$61.0_{\pm 1.4}$	$78.9_{\pm 0.9}$	$80.9_{\pm 3.9}$	$89.2_{\pm 0.3}$	$84.3_{\pm 2.5}$	$76.0_{\pm 4.7}$	81.7

Table 2: Evaluations of handcrafted encodings across 10 downstream tasks. We report the average score of five runs using different learning rates. *I* and *A* means Identical and Attenuated handcrafted positional encodings, respectively. $*$ means the positional encodings are learnable. *s* means that positional encodings are shared within the attention headers of layers.

newly constructed Shuffle-x and Shuffle-SR datasets (see details in section B.2). The overall results of word swap probing are shown in Table 1. We first observe performances of all language models across shuffle-x sets basically do not degenerate, which confirms our assumption that the core function of positional encodings is to symmetrically combine adjacent units. Second, the accuracy of all models decreases to some extent on the randomly shuffled data set but not much, as previously observed (Pham et al., 2021; Sinha et al., 2021). However, it is worth mentioning that the random shuffling may shift the semantics of the original sentence and thus cause the change of labels. In this paper, we also assume that the labels of the randomly shuffled sentences are unchanged and use only the scores of this dataset as a reference. Third, most models fail on the Shuffle-SR dataset, which demonstrates local symmetry does not capture global position changes well, which explain the reason that BERT fails on the example: "*an electric guitar playing a man on stage*". This, on the other hand, indicates that the benchmark of some existing sentence representations does not require complex global position information. Pham et al. (2021) also pointed out that word order and syntax are not necessarily required to solve GLUE tasks. Although the local symmetry learned by positional encodings can performs well on a series of language understanding tasks, the symmetry itself has obvious flaws. The better performance of StrucBERT on the Shuffle-SR suggests that introducing additional order-sensitive training tasks may improve this problem.

2.4.2 Handcrafted Positional Encodings Probing

As described in Section 2.3, we can manually design positional encodings that perfectly satisfy the properties of locality and symmetry. In this probing experiment, we pre-train BERT$_{base}$ with handcrafted positional encodings from scratch and compare them to learned positional encodings on downstream natural language understanding tasks. Specifically, we use 10 sentence-level tasks from SentEval (Conneau & Kiela, 2018) and GLUE (Wang et al., 2018) for evaluation, covering four types of tasks: MR (Pang & Lee, 2005), SUBJ (Pang & Lee, 2004) and SST (Socher et al., 2013) for Sentiment Analysis; QNLI (Rajpurkar et al., 2016), RTE (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009) and MNLI (Williams et al., 2018) for Textual Entailment; MRPC (Dolan & Brockett, 2005) and QQP [3] for Paraphrase Identification; STS (Cer et al., 2017) and SICK-R Marelli et al. (2014) for Textual Similarity. More details of the datasets and pre-training are shown in the Appendix Section B.5 and Section B.4, respectively.

Table 2 shows the experimental results of our manually designed PEs. We observe that the performances of both BERT-*I* and BERT-*A* can achieve competitive results with the learned absolute positional encodings in BERT, which once again verifies that the central role of existing positional encodings is to symmetrically combine neighboring units. The window size s of Identical Encoding is 3 and the attenuation degree w of Attenuated Encoding is 0.5. The main goal here is to check whether handcrafted encodings are able to approximate learned ones and therefore we do not search these hyper-parameters for better performances. On the other hand, there is a 0.8-1.0 percentage point gap between the original BERT and handcrafted positional encodings on average. We suppose that this because the behavior of these positional weights is identical across all attentional heads. In fact, there is a great diversity of behaviors of learned encodings within different attentional heads, as shown in Figure 8. Inspired by this, we make these handcrafted encodings learnable during pre-training, and we find the learned encodings (BEET-A^*-*s*) can significantly outperform the original

[3] `data.quora.com/First-Quora-Dataset-Release-Question-Pairs`

BERT and frozen handcrafted PEs on downstream tasks. If we make positional encodings in each attentional head learnable, the the benefits are not as significant (BERT-I^* and BERT-A^*). To investigate the reason, we visualize the positional weights δ_{ij} (Equation 12 and 13) in all attentional heads and Figure 9 shows these positional matrices are diverse but still satisfying Locality and Symmetry. The phenomenon suggests that introducing this prior knowledge can lead to a better induction bias for these sentence understanding tasks. See Appendix C.5 for more details.

3 Linguistic Roles of Positional Encodings

The syntactic structure is crucial to our understanding of sentences. Many studies have shown that a substantial amount of linguistic knowledge can be found in contextual word representations, e.g., subject-verb agreement (Goldberg, 2019) and dependency tree (Hewitt & Manning, 2019; Wu et al., 2020). Follow-up work further systematically analyzed the syntactic and semantic capabilities of the BERT model (Clark et al., 2019; Jawahar et al., 2019; Lin et al., 2019; Rogers et al., 2020). Nonetheless, the linguistic roles of Positional Encodings are still under-explored. In this section, we discuss what linguistic knowledge PEs have learned.

3.1 Linguistic Probing Tasks

We first conduct an ablation study to check the importance of positional and contextual encodings. The experimental details are shown in Appendix (Section C.2), and we find that the removal of any encodings degrade the model performances (Table 7). We therefore hypothesize that both encodings play a role in sentence comprehension and have different responsibilities.

Positional and contextual weights are usually entangled in every attentional head, therefore the behavior of positional encodings cannot be observed independently (as shown in Equation 9). To address this, we remove the contextual encodings $\mathbf{x}_i$ in Equation 4 of a pre-trained BERT and thus the output is only correlated with positional encodings. We refer to this adapted model as BERT-p. For comparison, we remove the positional encodings $\mathbf{p}_i$ to obtain BERT-c. We do the same for a pre-trained TUPE model, to obtain TUPE-p and TUPE-c.

In this linguistic probing, we adopt widely used 10 probing tasks (Conneau et al., 2018) with a standard evaluation toolkit (Conneau & Kiela, 2018). These tasks are a series of classification tasks that covers three categories: SentLen (Surface), WC (Surface), BShift (Syntactic), TreeDepth (Syntactic), TopConst (Syntactic), Tense (Semantic), SubjNum (Semantic), ObjNum (Semantic), SOMO (Semantic), CoordInv (Semantic). The details of this experiment are described in Section C.3.

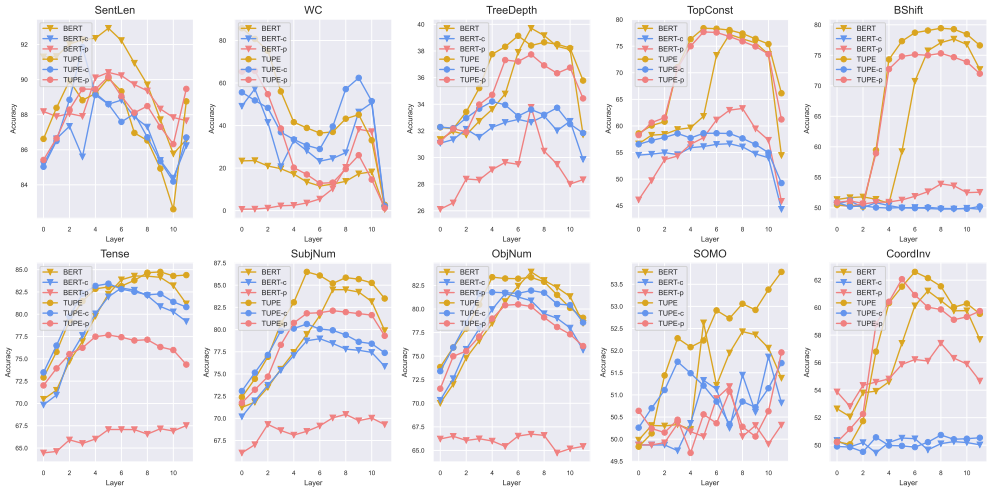

Figure 4: Results of linguistic probing tasks across different layers. BERT-based models are shown in triangle while TUPE-based models are shown in circle. The red, blue and yellow lines represent the use of positional encodings, contextual encodings and a combination of both, respectively.

Figure 4 gives the results. We first observe that the combination of contextual and positional encodings can have better performances across all probing tasks (yellow lines). Secondly, compared to contextual encodings, positional encodings perform better on syntactic tasks (TreeDepth, TopConst, BShift), which require more information of word orders. On semantic tasks, contextual encodings outperform contextual encodings on Tense and SubjNum while performs poorly when the semantic probing tasks require order information (CoordInv). Thirdly, a hierarchical structure exists here when we check the peak of probing tasks for each model, as observed by Jawahar et al. (2019). For surface tasks, the surface knowledge is stored more in bottom layer, syntactic knowledge is in middle layer and semantic knowledge is in middle and top layer. Therefore, we conclude that positional encodings play more of a role at the syntactic level tasks. On semantic tasks, especially position-independent ones, contextual encodings are more important.

3.2 Dependency Syntax of Positional Encodings

Clark et al. (2019) showed that particular heads of BERT indeed correspond remarkably well to particular long-distance dependencies, like `ccomp` and `mark`. To test the dependency knowledge stored in positional weights, we follow the syntactic probing test by Clark et al. (2019). Specifically, each head in PLMs is regarded as a simple predictor of dependency relations. Given the attention weight vector of an input word, we output the word with the highest values and think the pair of words hold some type of dependency relation. While no single head can perform well on all relations, the best-performed head is selected as the final ability of a model for each particular relation. In this experiment, we adopt the original TUPE that uses absolute positional encodings as our base model. The ability of contextual and positional weights is evaluated by removing the unrelated encodings in Eq 9, e.g., the first term is removed when checking the syntactic ability of PEs. The two variants are referred as to Contextual Attention (CA) and Positional Attention (PA).

We extract attention maps from BERT on the MRPC (Dolan & Brockett, 2005) annotated by dependency parser of spaCy [4]. We report the results on top-20 dependency relations. Figure 5 shows performance on different relations function of word distance in these relations (Table 8 gives relation-specific results). First, we observe that positional attention significantly more important than contextual attention in short-distance dependency relations (distance from 1 to 4). Second, contextual attention performs takes the lead on long-distance relations (after 6). Again, the combination of the two features can yield the best performance. The "outlier" in the lower left corner is the `Root` dependency. Because this relation is a self-reflexive edge, contextual (or self) attentions can performs well on it while learned PEs do not attend to the current word itself, e.g., visualizations of BERT, DeBERTa and T5 in Figure 2.

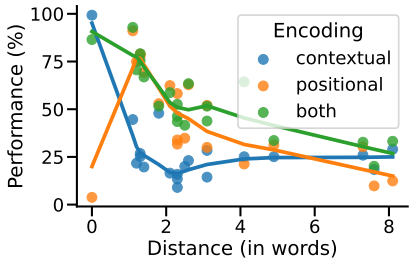

Figure 5: Accuracy of top-20 dependency relations. Detailed results in Table 8, Appendix.

4 How to Combine Contextual and Positional Features

As discussed before, the core function of positional encodings is to symmetrically combine local units. As for the linguistic role, the contextual weights are good at long-distance syntactic relations while positional weights are suitable for short dependencies. The combination of the two weights can lead to better sentence representations. The add operation of contextual and positional correlation is widely used in many PLMs (Devlin et al., 2019; Liu et al., 2019; Yang et al., 2019; Ke et al., 2021), as shown in Eq 9. However, to comprehend a sentence, the semantic meaning is composed bottom-up: morpheme $\rightarrow$ word $\rightarrow$ phrase $\rightarrow$ clause $\rightarrow$ sentence. For example, the two sub-words are composed to a particular meaning $\{$*"context","##ual"*$\} \rightarrow$ *"contextual"* and the two words are composed to a phrase $\{$*"take","off"*$\} \rightarrow$ *"take off"*. When we reach word *take* and *off*, the two words should be composed of a phrase first and then the long-distance contextual information is used to understand the specific meaning of the phrase. At least, it is not necessary to consider contextual in-

[4] `https://spacy.io/api/dependencyparser`

Model	Size	Sentiment Analysis			Textual Entailment			Paraphrase Identification		Textual Similarity		Avg
		MR (22K)	SUBJ (20K)	SST-2 (68.8K)	QNLI (110K)	RTE (5.5K)	MNLI (413K)	MRPC (5.4K)	QQP (755k)	STS (8.4K)	SICK-R (9.4K)	
BEET-A^*	138M	$78.2_{\pm 3.5}$	$93.0_{\pm 0.8}$	$88.1_{\pm 1.0}$	$\mathbf{87.0}_{\pm 0.5}$	$61.0_{\pm 1.4}$	$\mathbf{78.9}_{\pm 0.9}$	$80.9_{\pm 3.9}$	$89.2_{\pm 0.3}$	$84.3_{\pm 2.5}$	$77.0_{\pm 4.7}$	81.8
BERT-A^*-*Cat*	152M	$\mathbf{80.1}_{\pm 1.5}$	$\mathbf{94.4}_{\pm 0.7}$	$87.7_{\pm 1.2}$	$85.4_{\pm 0.7}$	$62.8_{\pm 1.9}$	$78.2_{\pm 0.4}$	$76.9_{\pm 1.4}$	$\mathbf{89.8}_{\pm 0.3}$	$84.1_{\pm 1.4}$	$78.9_{\pm 2.2}$	81.8
BERT-A^*-*Seq*	138M	$78.7_{\pm 2.4}$	$93.3_{\pm 0.7}$	$\mathbf{88.7}_{\pm 0.3}$	$86.5_{\pm 0.4}$	$\mathbf{64.1}_{\pm 2.6}$	$78.1_{\pm 0.4}$	$\mathbf{81.1}_{\pm 2.1}$	$89.5_{\pm 0.3}$	$\mathbf{85.1}_{\pm 1.9}$	$\mathbf{79.3}_{\pm 1.1}$	**82.5**

Table 3: Evaluations of different combinations of positional and contextual encodings across 10 downstream tasks. We report the average score of five runs using different learning rates. *Cat* and *Seq* means Concatenation and Sequence combination, respectively. $*$ means the positional encodings are learnable.

formation until we compose basic semantic units like words (*"contextual"*) and short phrase (*"take off"*). Therefore, a key question arises: how to combine the contextual and positional weights?

To combine the Contextual Attention (CA) and Positional Attention (PA) better and according to our analysis, we propose a new strategy: **Sequence Combination**. This method uses two encodings in a specific order. First, positional attentions are used to combine nearby tokens for composing basic semantic units, e.g., $\{$*take, off*$\}$ $\rightarrow$ *take off*. Afterward, contextual attention are applied to these composing units and thus they can be better understood under the contexts. For example, the compound *take off* have multiple meanings and we should consider the context to understand its specific meaning. This process can be written as:

$$\bar{\mathbf{X}} = \mathrm{CA}(\mathrm{PA}(\mathbf{X})) \tag{1}$$

Here, CA and PA are Contextual Attention and Positional Attention, respectively. The computation of CA and PA are obtained by removing another item from Eq 9. A code example of this method is shown in Listing 3.

We pre-train and fine-tune different variants and compare them on 10 downstream datasets (details in Section B.4). The experimental results are shown in Table 3. The baselines are described in Section B.6. We observe our proposed Sequence combination outperforms the original add operation (improvements on 8 tasks) without introducing additional parameters while the Concatenation performs similarly to the add operation, which shows that prioritizing local semantic unit composition is beneficial for sentence representation. Besides, the Sequence combination can achieve lower training and validation loss than the BERT-A* using the same steps (Figure 7). A conclusion is that we can subtly adjust the sequence of using positional encoding without introducing additional parameters to obtain a certain degree of improvement in the language understanding tasks.

5 Conclusion

We have proposed a series of probing analyses for understanding the role of positional encodings in sentence representations. While most existing analyses found the insensitivity of positional encodings to word orders for natural language understanding tasks, we conducted systematic investigations to explain the phenomenon and uncover the core function of positional encodings. We find that the main effect of current positional encoding schemes is to impose locality and symmetry. Since human language understanding is to some degree robust to local word swaps, this is sufficient to understand a typical sentence. This is not to say that blindness to word swaps would be desirable in general. Rather, it is to say that existing architecture can be simplified to locality and symmetry without loss of expressive power or inductive bias. We further complement existing research by distinguishing the roles of positional and contextual encodings in sentence representations, and by proposing to combine them sequentially rather than additively.

The limitations of this work are two-fold. First, our analysis is limited to the natural language understanding of the English language. Different languages display different word order properties. For instance, English is subject-verb-object order (SVO) while Japanese is subject-object-verb order (SOV), and natural language generation models are not included in this work. Second, although our handcrafted positional encodings satisfy the Symmetry property, it merely replicates the limitations of current positional encoding, albeit in a simplified form. Further architecture development should address the problem of the "*electric guitar playing a man on stage.*" mentioned in the introduction.

6 REPRODUCIBILITY STATEMENT

Apart from the descriptions in the main content, we have detailed the complete experimental settings in Section B. Besides, we have submitted our source code and will make it publicly available.

REFERENCES

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

Luisa Bentivogli, Ido Kalman Dagan, Dang Hoa, Danilo Giampiccolo, and Bernardo Magnini. The fifth pascal recognizing textual entailment challenge. In *TAC 2009 Workshop*. no publisher, 2009.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. In *EMNLP*, 2015.

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, 2017.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 276–286, 2019.

Louis Clouatre, Prasanna Parthasarathi, Amal Zouaq, and Sarath Chandar. Local structure matters most: Perturbation study in nlu. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 3712–3731, 2022.

Alexis Conneau and Douwe Kiela. Senteval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *ACL 2018-56th Annual Meeting of the Association for Computational Linguistics*, volume 1, pp. 2126–2136. Association for Computational Linguistics, 2018.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pp. 177–190. Springer, 2005.

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.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

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, 2019.

Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Philipp Dufter, Martin Schmitt, and Hinrich Schütze. Position information in transformers: An overview. *Computational Linguistics*, 48(3):733–763, 2022.

William Edward Dyer. *Minimizing integration cost: A general theory of constituent order*. PhD thesis, University of California, Davis, 2017.

Fernanda Ferreira, Karl GD Bailey, and Vittoria Ferraro. Good-enough representations in language comprehension. *Current directions in psychological science*, 11(1):11–15, 2002.

Richard Futrell, Roger P Levy, and Edward Gibson. Dependency locality as an explanatory principle for word order. *Language*, 96(2):371–412, 2020.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pp. 1243–1252. PMLR, 2017.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pp. 1–9, 2007.

Edward Gibson, Leon Bergen, and Steven T Piantadosi. Rational integration of noisy evidence and prior semantic expectations in sentence interpretation. *Proceedings of the National Academy of Sciences*, 110(20):8051–8056, 2013.

Yoav Goldberg. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*, 2019.

Maosheng Guo, Yu Zhang, and Ting Liu. Gaussian transformer: a lightweight approach for natural language inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 6489–6496, 2019.

Ashim Gupta, Giorgi Kvernadze, and Vivek Srikumar. Bert & family eat word salad: Experiments with text understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 12946–12954, 2021.

R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, volume 7, 2006.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2020.

John Hewitt and Christopher D Manning. A structural probe for finding syntax in word representations. 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. 4129–4138, 2019.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does bert learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Zi-Hang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. Convbert: Improving bert with span-based dynamic convolution. *Advances in Neural Information Processing Systems*, 33:12837–12848, 2020.

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. *arXiv preprint arXiv:1412.6980*, 2014.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Roger Levy. A noisy-channel model of human sentence comprehension under uncertain input. In *Proceedings of the 2008 conference on empirical methods in natural language processing*, pp. 234–243, 2008.

Yongjie Lin, Yi Chern Tan, and Robert Frank. Open sesame: Getting inside bert's linguistic knowledge. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 241–253, 2019.

Haitao Liu, Chunshan Xu, and Junying Liang. Dependency distance: A new perspective on syntactic patterns in natural languages. *Physics of life reviews*, 21:171–193, 2017.

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.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pp. 216–223, 2014.

Francis Mollica, Matthew Siegelman, Evgeniia Diachek, Steven T Piantadosi, Zachary Mineroff, Richard Futrell, Hope Kean, Peng Qian, and Evelina Fedorenko. Composition is the core driver of the language-selective network. *Neurobiology of Language*, 1(1):104–134, 2020.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pp. 271–278, 2004.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 115–124, 2005.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2249–2255, 2016.

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

Ofir Press, Noah Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2021.

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. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. 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. 3982–3992, 2019.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

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

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 2888–2913, 2021.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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, 2013.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

Sho Takase and Naoaki Okazaki. Positional encoding to control output sequence length. *arXiv preprint arXiv:1904.07418*, 2019.

David Temperley and Daniel Gildea. Minimizing syntactic dependency lengths: Typological/cognitive universal? *Annual Review of Linguistics*, 4:67–80, 2018.

Matthew J Traxler. Trends in syntactic parsing: Anticipation, bayesian estimation, and good-enough parsing. *Trends in cognitive sciences*, 18(11):605–611, 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.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, 2018.

Benyou Wang, Donghao Zhao, Christina Lioma, Qiuchi Li, Peng Zhang, and Jakob Grue Simonsen. Encoding word order in complex embeddings. *arXiv preprint arXiv:1912.12333*, 2019a.

Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. On position embeddings in bert. In *International Conference on Learning Representations*, 2020.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. Structbert: Incorporating language structures into pre-training for deep language understanding. In *International Conference on Learning Representations*, 2019b.

Xing Wang, Zhaopeng Tu, Longyue Wang, and Shuming Shi. Self-attention with structural position representations. *arXiv preprint arXiv:1909.00383*, 2019c.

Yu-An Wang and Yun-Nung Chen. What do position embeddings learn? an empirical study of pre-trained language model positional encoding. *arXiv preprint arXiv:2010.04903*, 2020.

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, 2018.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. Perturbed masking: Parameter-free probing for analyzing and interpreting bert. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4166–4176, 2020.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

A ATTENTION AND POSITIONAL ENCODINGS

A.1 ATTENTION

The central building block of transformer architectures is the attention mechanism. It maps a query and a set of key-value pairs to a new output, where the input query, keys, values, and output are all vectors. The inputs are given as a set of n query vectors, grouped horizontally in a matrix $\mathbf{Q}$. The vectors of keys and the vectors of values, likewise, are stacked as matrices $\mathbf{K}$ and $\mathbf{V}$. Then, the new output matrix $\bar{\mathbf{V}}$ is computed by the Scaled Dot-Product Attention (Vaswani et al., 2017):

$$\bar{\mathbf{V}} = \mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Softmax}\left(\frac{\mathbf{Q}\,\mathbf{W}^Q(\mathbf{K}\,\mathbf{W}^K)^\mathsf{T}}{\sqrt{d}}\right)(\mathbf{V}\,\mathbf{W}^V) \tag{2}$$

Here, $\mathbf{W}^Q \in \mathbb{R}^{d\times d_K}, \mathbf{W}^K \in \mathbb{R}^{d\times d_K}, \mathbf{W}^V \in \mathbb{R}^{d\times d_V}$ are trainable parameter matrices, d is the input dimension of $\mathbf{Q}, \mathbf{K}, \mathbf{V}$, and d_K, d_V are the dimensions of the Key and Value after projection.

Self-Attention applies attention to a single set of tokens: the key, query, and values are all functions of token embeddings. The input (e.g., a sentence or a sequence of words) is a sequence matrix $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \in \mathbb{R}^{n\times d}$, and the output vector of the ith token is given by:

$$\bar{\mathbf{x}}_i = \sum_{j=1}^{n} \frac{\exp(\alpha_{ij})}{Z}\mathbf{x}_j\mathbf{W}^V \qquad \text{where } \alpha_{ij} = \frac{(\mathbf{x}_i\mathbf{W}^Q)(\mathbf{x}_j\mathbf{W}^K)^\mathsf{T}}{\sqrt{d}}, Z = \sum_{j=1}^{n}\exp(\alpha_{ij}) \tag{3}$$

As tokens appear in the keys and the queries, self-attention can learn the importance of the interactions between words. It has drastically improved performance in many NLP tasks.

A.2 POSITIONAL ENCODINGS

Self-attention heads do not intrinsically capture the word orders in a sequence. Therefore, specific methods are used to infuse positional information into self-attention.

Absolute Positional Encoding (APE). The original Transformer (Vaswani et al., 2017) and BERT (Devlin et al., 2019) compute a positional encoding for each token and add it to the input content embedding to inject position information in the original sequence. The α_{ij} in Equation 3 are then written

$$\alpha_{ij} = \frac{(\mathbf{x}_i + \mathbf{p}_i)\mathbf{W}^Q\big((\mathbf{x}_j + \mathbf{p}_j)\mathbf{W}^K\big)^\mathsf{T}}{\sqrt{d}} \tag{4}$$

Here, $\mathbf{p}_i \in \mathbb{R}^d$ is a position embedding of the ith token, obtained by predefined functions (Vaswani et al., 2017; Dehghani et al., 2018; Takase & Okazaki, 2019) or learned embeddings (Gehring et al., 2017; Devlin et al., 2019).

Relative Positional Encoding (RPE) produces a value $\mathbf{r}_{ij}$ that depends on the relative distance of tokens. Specifically, these methods apply such positional encoding to the attention head so that the corresponding attention weights can be updated based on the relative distance of two tokens (Shaw et al., 2018):

$$\alpha_{ij} = \frac{\mathbf{x}_i\mathbf{W}^Q\big(\mathbf{x}_j\mathbf{W}^K + \mathbf{r}^K_{ij}\big)^\mathsf{T}}{\sqrt{d}} \tag{5}$$

The T5 approach (Raffel et al., 2019) simplifies Equation 5 by replacing the second term with a scalar β_{ij} encoding the relative distance, which is then added to the corresponding attention weights:

$$\alpha_{ij} = \frac{(\mathbf{x}_i\mathbf{W}^Q)(\mathbf{x}_j\mathbf{W}^K)^\mathsf{T} + \beta_{ij}}{\sqrt{d}}, \tag{6}$$

The Transformer with Untied Positional Encoding (TUPE) (Ke et al., 2021) separates the context correlation and positional correlation and then adds them up:

$$\alpha_{ij} = \frac{(\mathbf{x}_i\mathbf{W}^Q)(\mathbf{x}_j\mathbf{W}^K)^\mathsf{T} + (\mathbf{p}_i\mathbf{U}^Q)(\mathbf{p}_j\mathbf{U}^K)^\mathsf{T}}{\sqrt{d}}, \tag{7}$$

Here, $\mathbf{U}^Q$ and $\mathbf{U}^K$ are weights that need to be learned, capturing positional queries and keys, respectively. The reason for this strategy can be understood by contrasting it to the standard positional encoding. Expanding Equation 4 gives:

$$\alpha_{ij} = \frac{(\mathbf{x}_i\mathbf{W}^Q)(\mathbf{x}_j\mathbf{W}^K)^\mathsf{T}}{\sqrt{d}} + \frac{(\mathbf{x}_i\mathbf{W}^Q)(\mathbf{p}_j\mathbf{W}^K)^\mathsf{T}}{\sqrt{d}} + \frac{(\mathbf{p}_i\mathbf{W}^Q)(\mathbf{x}_j\mathbf{W}^K)^\mathsf{T}}{\sqrt{d}} + \frac{(\mathbf{p}_i\mathbf{W}^Q)(\mathbf{p}_j\mathbf{W}^K)^\mathsf{T}}{\sqrt{d}} \tag{8}$$

There are four terms in this expression: context-to-context, context-to-position, position-to-context, and position-to-position. While the first and the fourth term are intuitive, the token embeddings and positional embeddings do not have strong correlations with each other, and the context-to-position term may even induce unnecessary noise. This intuition is confirmed by empirical studies that demonstrate that the second and third term can be removed (Ke et al., 2021).

Unified Positional Encoding. Inspired by TUPE (Ke et al., 2021), we rewrite all above absolute and relative positional encodings in a unified way:

$$\alpha_{ij} = \frac{\overbrace{\gamma_{ij}}^{contextual} + \overbrace{\delta_{ij}}^{positional}}{\sqrt{d}} \tag{9}$$

Here, γ, the left half of the numerator, captures contextual correlations (or weights), i.e., the semantic and syntactic relations between token x_i and x_j. In BERT and T5, it is $\gamma_{ij} = (\mathbf{x}_i\mathbf{W}^Q)(\mathbf{x}_j\mathbf{W}^K)^\mathsf{T}$. δ, the right half, captures positional information, i.e., the positional relations between tokens x_i and x_j. For example, BERT's absolute encoding can be obtained by setting:

$$\delta_{ij} = \frac{(\mathbf{x}_i\mathbf{W}^Q)(\mathbf{p}_j\mathbf{W}^K)^\mathsf{T}}{\sqrt{d}} + \frac{(\mathbf{p}_i\mathbf{W}^Q)(\mathbf{x}_j\mathbf{W}^K)^\mathsf{T}}{\sqrt{d}} + \frac{(\mathbf{p}_i\mathbf{W}^Q)(\mathbf{p}_j\mathbf{W}^K)^\mathsf{T}}{\sqrt{d}} \tag{10}$$

The relative encoding in Shaw et al. (2018) can be obtained by setting

$$\delta_{ij} = \frac{\mathbf{x}_i\mathbf{W}^Q(\mathbf{r}^K_{ij})^\mathsf{T}}{\sqrt{d}} \tag{11}$$

Thus, existing positional encodings all add contextual and positional correlations together in every attention head.

B Details of Experiments

B.1 Visualizations of Positional Encodings

Model	Size	Version	Language
BERT	110M	*bert-base-uncased*	English
ALBERT	235M	*albert-xxlarge-v2*	English
Roformer	110M	*junnyu/roformer_chinese_base*	Chinese
ConvBERT	106M	*YituTech/conv-bert-base*	English
DeBERTa	100M	*microsoft/deberta-base*	English
XLNet	110M	*xlnet-base-cased*	English
T5	340M	*t5-base*	English
StrucBERT	220M	*bayartsogt/structbert-large*	English

Table 4: Details of pre-trained language models used in visualizations.

To understand what positional encodings learn after pre-training, we visualize the positional weights in attentional heads. The Identical Word Probing is adopted in this experiment (Wang et al., 2020). The used pre-trained language models are shown in Table 4, and the repeated words are randomly

	Original	Shuffled
Shuffled-3	*An old man* *with a package* *poses in front* *of an advertisement* .	*An man old* *with package a* *poses in front* *of advertisement an* .
Shuffled-4	A land rover is being driven across a river .	A land rover is being a driven river across .
Shuffled-5	A man reads the paper in a bar with green lighting .	A man reads the paper in with green a lighting bar .
Shuffled-6	A little boy in a gray and white striped sweater and tan pants is playing on a piece of playground equipment .	A little boy in striped a sweater and white gray and tan pants is playing piece playground of equipment on a .
Shuffled-SR	several women are playing volleyball .	volleyball are playing several women .
Shuffled-SR	a man and woman are sharing a hotdog .	a hotdog are sharing a man and woman .

Table 5: Some cases of the shuffled SNLI datasets in our word swap probing. Texts in the same color mean the corresponding phrases.

selected from the corresponding vocabulary. Note that sub-tokens like single characters and non-physical words are removed. For visualization, we adopt the Identical Word Probing proposed by Wang et al. (2020), which feeds many repeated identical words to pre-trained language models and thus the attention values (α_{ij} in Equation 9) are disentangled with contextual weights. More specifically, we randomly select 100 words from the corresponding vocabulary (filtering out single characters and sub-words such as "`##nd`"). Note that the vocabulary of Roformer is Chinese. We repeat each word to compose a sentence of length 128. These 100 sentences are fed into a language model and the attention weights across different layers are averaged as the positional weight matrix of a particular language model.

B.2 Word Swap Probing

To valid if language models with positional encodings are sensitive to the local and global word swaps, we construct shuffle-x and shuffle-SR SNLI datasets. Shuffle-x means the word orders of phrases with length x are disrupted, e.g. "*an electric guitar*" is a 3-gram phrase, and it might be "*guitar an electric*" in shuffle-3 SNLI. Through this way, a new sentence with the same meaning can be obtained and therefore the initial label of the sample will not be changed. To construct such shuffled datasets, the premise sentences in SNLI test set are shuffled and we keep the hypothesis sentences intact. Here, we let $x \in [3, 6]$ and select a subset from SNLI to make sure that every premise sentence has at least one phrase with lenth from 2 to 6. We select five types of target phrases for shuffling: *Noun Phrase, Verb Phrase, Prepositional Phrase, Adverb Phrase, and Adjective Phrase*. Finally, a shuffle-x SNLI is obtained by disrupt the order inside a phrase with length x and the size for each shuffle-x is around 5000. The first fourth rows in Table 5 shows some samples.

As for the shuffle-SR SNLI dataset, the semantic roles of agent and patient are swapped in a sentence. We use the Algorithm 1 to collect a subset from SNLI test set. This algorithm is applied successively to the premise and hypothesis sentence for a sample whose label is entailment, and if the result of either of them is not null, we consider it a valid shuffled sample, which means we only shuffle the premise or hypothesis. After, we can obtain a new sample and the pair of sentences are contradicted with each other. In total, there are 1329 samples. To ensure that all sentences are semantically correct, we manually selected 300 pairs from them. The last two rows in Table 5 shows two examples in shuffle-SR dataset.

To probe the capabilities of language models on our newly constructed datasets, we adopt six different pre-trained language models (as shown in Table 6) and we use Hugging Face for implementation (Wolf et al., 2020). These models are fine-tuned on the training set of SNLI, and the model with the best score on validation set is stored for the follow experiments. Note that there are off-the-shell ALBERT and XLNet for natural language inference, we therefore use them directly without fine-tuning. During fine-tuning stage, the maximum length of the tokenized input sentence pair is 128, and the optimizer is Adam (Kingma & Ba, 2014) with learning rate of 2e-5. The batch size is 32 and the epoch is 3. After fine-tuning, the best model is evaluated on our shuffle SNLI test set, an we record their performances when faced with local and global word swaps.

Algorithm 1: Construction of Shuffle-SR Sentences

Input:
s: a premise or hypothesis sentence in SNLI,
$\mathcal{A}$: Auxiliary verb list
$\mathcal{M}$: Semantic Role Labeling Model,
$\mathcal{D}$: Subject and Object Case Mapping `// e.g., I ↔ me`
Output: A sentence s* with shuffled agent and patient or *None*

$\mathcal{R} \leftarrow$ Predict the semantic roles of words in sentence s by using the model $\mathcal{M}$
$\mathcal{V} \leftarrow$ Take the verb list from $\mathcal{R}$
foreach *verb v in $\mathcal{V}$* **do**
  if *v appears in $\mathcal{A}$* **then**
    continue
  if *$\mathcal{R}$ does not contain an agent or patient* **then**
    continue
  $a, p \leftarrow$ Take the agent and patient from $\mathcal{R}$
  s* $\leftarrow$ Swap the a, p in sentence s
  s* $\leftarrow$ Transform the subject and object case in s* if a or p in $\mathcal{D}$
  return s*
return *None*

B.3 The Computation of Handcrafted Positional Encodings

As in convolutional operations, one option is to pay equal attention to the words within a window size of the current token:

$$\delta_{ij} = \Phi(l_{ij}, s) = \begin{cases} \frac{1}{s} & l_{ij} * 2 + 1 \leq s \\ 0 & l_{ij} > s \end{cases} \tag{12}$$

Here, l_{ij} is the relative distance and s is the length of window.

Attenuated Encoding Unlike with identical encodings, a gradual decrease of positional weight with the relative distance seems preferable. For this, Gaussian Transformers (Guo et al., 2019) use a Gaussian kernel:

$$\delta_{ij} = \Phi(l_{ij}) = \frac{\exp(-w\, l_{ij}^2)}{\sum_{j=1}^{n} \exp(-w\, l_{ij}^2)} \tag{13}$$

Here, l_{ij} is the relative distance and $w > 0$ is a scalar parameter, which controls the degree of attenuation.

B.4 Details of Pre-training

We use the configuration of the original BERT$_{base}$ (Devlin et al., 2019) with 110M parameters for pre-training. Our model is implemented with PyTorch using the pytorchic-bert tool[5]. The number of layers, attention heads, and the projection dimension are 12, 12, and 768 respectively. We use the original vocabulary with a size of 30522. The training corpus is the English Wikipedia (20200101 dumps), which totals 13G after preprocessing by WikiExtractor. We pre-train with sequences of at most $T = 512$ tokens, and set the batch size as 64 to pre-training 600K steps. The optimizer is Adam with a learning rate of 5e-4, $\beta 1$ = 0.9, $\beta 2$ = 0.999, L2 weight decay of 0.01, and a warmup rate of 0.1. The dropout probability is always set as 0.1.

We use the original BERT$_{base}$ as our backbone and vary the positional encodings to pre-training different variants for comparison.

Listing 1 shows a code example about how to inject handcrafted positional encodings to the BERT backbone. Each variant is fine-tuned on the training dataset with different learning rates (among

[5] `https://github.com/dhlee347/pytorchic-bert`

Model	Size	Version	Fine-tuned by us
BERT	110M	*bert-base-uncased*	✓
ALBERT	223M	*ynie/albert-xxlarge-v2-snli_mnli_fever_anli_R1_R2_R3nli*	×
ConvBERT	106M	*YituTech/conv-bert-base*	✓
DeBERTa	100M	*microsoft/deberta-base*	✓
XLNet	340M	*ynie/xlnet-large-cased-snli_mnli_fever_anli_R1_R2_R3-nli*	×
StrucBERT	340M	*bayartsogt/structbert-large*	✓

Table 6: Details of pre-trained language models used in word swap probing.

9e-5, 7e-5, 5e-5, 3e-5, 1e-5). After, we evaluate the fine-tuned model on the Dev set and report the average score of five learning rates. Apart from BERT, we introduce the TUPE model as another baseline. Specifically, we pre-train the following variants:

- BERT is the original one and we use it as a baseline.
- BERT-*I* uses Identical handcrafted positional encodings and these encodings are fixed and shared across all layers.
- BERT-*A* uses Attenuated handcrafted positional encodings and these encodings are fixed and shared across all layers.
- BERT-A^* and BERT-I^* are varaints of the former two, but the encodings are learnable during pre-training.
- BERT-A^*-*s* shares learnable positional encodings within a layer.
- BERT-*only-c* is for ablation study, and the positional encodings $\mathbf{p}_i$ in Equation 4 are removed.
- BERT-*only-p* is for ablation study, and the contextual encodings $\mathbf{x}_i$ in Equation 4 are removed.
- BERT-A^*-*Cat* combines the contextual and positional features by concatenating them in a learnable way.
- BERT-A^*-*Seq* combines the two features in a sequential way, and the positional attentions are first used and then contextual attentions.
- TUPE computes contextual and positional features with different parameterizations and then adds them together. We use it for comparing different ways of combining contextual and positional features.
- TUPE-*Cat* and TUPE-*Seq* are another two variants for validating the effect of our proposed combination methods.

Suppose that the hidden dimension is 768, the layer number is 12, the head number is 12, and the maximum length is 512 for BERT$_{base}$ model, we can calculate the size for each variant. The number of parameters of handcrafted positional encoding for each head is 262K (512×512). If positional heads are different across all layers, the total cost is 37.7M ($512 \times 512 \times 12 \times 12$). If the positional encodings are shared across heads, the total cost is 3.1M ($512 \times 512 \times 12$).

The the number of parameters of the combination layer is 14.2M ($2 \times 768 \times 768 \times 12$).

```
class MultiHeadedSelfAttention(nn.Module):
   """ Multi-Headed Scaled Dot Product Attention """
   def __init__(self, config):
      super().__init__()
      self.n_heads = config.n_heads
      self.drop = nn.Dropout(config.p_drop_attn)
      self.proj_q = nn.Linear(config.dim, config.dim)
      self.proj_k = nn.Linear(config.dim, config.dim)
      self.proj_v = nn.Linear(config.dim, config.dim)


   def forward(self, x, mask, pe):
      """
      x, q(query), k(key), v(value) : (B(batch_size), S(seq_len), D(dim))
      mask : (B(batch_size) x S(seq_len))
      pe: positional weights (B(batch_size), H(Head_number)), S(seq_len), S(seq_len))
      * split D(dim) into (H(n_heads), W(width of head)) ; D = H * W
      """
      # (B, S, D) -proj-> (B, S, D) -split-> (B, S, H, W) -trans-> (B, H, S, W)
      q, k, v = self.proj_q(x), self.proj_k(x), self.proj_v(x)
      q, k, v = (split_last(x, (self.n_heads, -1)).transpose(1, 2)
              for x in [q, k, v])
      # (B, H, S, W) @ (B, H, W, S) -> (B, H, S, S) -softmax-> (B, H, S, S)
      scores = q @ k.transpose(-2, -1) / np.sqrt(k.size(-1))

      # inject positional weights into contextual weights
      # (B, H, S, S) + (B, H, S, S) -> (B, H, S, S)
      scores = scores + pe

      if mask is not None:
         mask = mask[:, None, None, :].float()
         scores -= 10000.0 * (1.0 - mask)

      scores = self.drop(F.softmax(scores, dim=-1))
      # (B, H, S, S) @ (B, H, S, W) -> (B, H, S, W) -trans-> (B, S, H, W)
      h = (scores @ v).transpose(1, 2).contiguous()
      # -merge-> (B, S, D)
      h = merge_last(h, 2)
      return h
```

Listing 1: A code example of how to inject handcrafted positional encodings into self-attentions.

To inject handcrafted positional encodings, we pre-compute the positional weights and add them to the contextual weights directly, as shown in Listing 1. These weights can be either frozen or learnable during pre-training. The code of the two proposed combinations are shown in Listing 2 and Listing 3, respectively.

B.5 Details of Downstream Datasets

SentEval is based on a set of existing text classification tasks Involving one or two sentences as input. However, most tasks in SentEval are closely related to sentiment analysis and thus not diverse enough. GLUE benchmark introduces a series of difficult natural language understanding tasks while some particular tasks only contain one dataset, e.g., sentiment analysis and textual similarity. Moreover, the size of WNLI in GLUE is rather small and the GLUE webpage notes that there are issues with the construction of this dataset [6]. To better evaluate the capability of models for sentence representation, we, therefore, select 10 datasets from SentEval and GLUE, covering four types of sentence-level tasks:

- **Sentiment Analysis** is also known as opinion mining, which aims to classify the polarity of a given text, whether the expressed opinion is positive, negative, or neutral. We use MR (Pang & Lee, 2005), SUBJ (Pang & Lee, 2004), and SST (Socher et al., 2013) for this task.
- **Textual Entailment** describes the inference relation between a pair of sentences, whether the premise sentence entails the hypothesis sentence. Actually, this is a classification task with three labels: entailment, contradiction, and neutral. Here, we use QNLI (Rajpurkar et al., 2016), RTE (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009) and MNLI (Williams et al., 2018) for evaluation. Note that we report the average score for the two test sets of MNLI.

[6] https://gluebenchmark.com/faq

- **Paraphrase Identification** is to determine whether a pair of sentences have the same meaning. We use MRPC (Dolan & Brockett, 2005) and QQP [7] for evaluation.

- **Textual Similarity** deals with determining how similar two pieces of texts are. We use STS (Cer et al., 2017) and SICK-R (Marelli et al., 2014) for evaluation.

B.6 Experiments of The Combination

To valid the effect of the sequential combination, we pre-train it from scratch and compare the performances on 10 downstream datasets. The baselines here are: BERT that uses an additive combination and BERT with a Concatenation combination. The Concatenation combine contextual and positional information by concatenating the two types of attentions and combining them in a learnable way:

$$\bar{\mathbf{X}} = \mathrm{Concat}\big(\mathrm{CA}(\mathbf{X});\ \mathrm{PA}(\mathbf{X})\big)\,\mathbf{W}^O \tag{14}$$

where $\mathbf{W}^O \in \mathbb{R}^{(d_V+d_V)\times d_O}$ is a trainable matrix. d_V is the dimension of values in CA and PA, and d_O is that of $\bar{\mathbf{X}}$. A code example of this method is shown in Listing 2.

C Additional Experiments

C.1 Loss Curves of Pre-training

Apart from the performances on downstream tasks, the loss curves are also checked for different variants. For this goal, the training loss and validation loss are stored after certain steps. We use a hold-set as the validation set. As shown in Figure 7, our proposed BERT-A*-*Cat* and BERT-A*-*Seq* have smaller loss than other baselines. This can be observed again on the validation set.

C.2 Ablation Study of Positional and Contextual Encodings

To check the importance of positional and contextual Encodings, we conduct an ablation study. For this goal, the contextual encodings $\mathbf{x}_i$ or positional encodings $\mathbf{p}_i$ in Equation 4 are removed, respectively, during pre-training and the two new models are evaluated on 10 sentence-level datasets. As shown in Table 7, the BERT-*only-c* and BERT-*only-p* both lag behind the original BERT models, which means the combination of the two features is beneficial for sentence representations. On the other hand, positional encodings are more important for sentiment analysis, and the cross-attentions from contextual embeddings matter in sentence-pair tasks.

Model	Sentiment Analysis			Textual Entailment			Paraphrase Identification		Textual Similarity		Avg
	MR (22K)	SUBJ (20K)	SST-2 (68.8K)	QNLI (110K)	RTE (5.5K)	MNLI (413K)	MRPC (5.4K)	QQP (755k)	STS (8.4K)	SICK-R (9.4K)	
BERT	$72.5_{\pm 5.3}$	$91.0_{\pm 2.7}$	$86.4_{\pm 2.7}$	$85.8_{\pm 1.0}$	$59.2_{\pm 1.2}$	$78.2_{\pm 0.8}$	$73.5_{\pm 1.8}$	$88.7_{\pm 0.6}$	$77.8_{\pm 4.1}$	$64.9_{\pm 6.0}$	77.8
BERT-*only-c*	$73.0_{\pm 4.6}$	$88.9_{\pm 2.5}$	$82.9_{\pm 0.4}$	$82.0_{\pm 0.1}$	$62.7_{\pm 4.3}$	$70.8_{\pm 0.8}$	$74.1_{\pm 0.3}$	$86.9_{\pm 0.5}$	$78.5_{\pm 0.3}$	$64.5_{\pm 5.7}$	76.2
BERT-*only-p*	$73.8_{\pm 4.6}$	$90.8_{\pm 1.4}$	$84.0_{\pm 0.7}$	$79.8_{\pm 1.2}$	$50.9_{\pm 1.4}$	$68.3_{\pm 1.0}$	$73.9_{\pm 1.6}$	$85.8_{\pm 0.6}$	$47.1_{\pm 18.0}$	$51.7_{\pm 9.2}$	70.6

Table 7: Ablation study across 10 sentence-level tasks. We report the average score of five runs using different learning rates.

[7] `data.quora.com/First-Quora-Dataset-Release-Question-Pairs`

```
class Concatenation(nn.Module):
""" Concatenation Block """

  def __init__(self, config):
    super().__init__()
    # mode = ["Identical", "Attenuated"]
    self.pos_mode = config.pos_mode
    self.pos_learnable = config.pos_learnable
    self.self_attention = MultiHeadedSelfAttention(config)
    self.positional_attention = PositionalAttention(config, mode=self.pos_mode,
          learnable=self.pos_learnable)
    self.combine = nn.Linear(config.dim * 2, config.dim)

  def forward(self, x, mask):
    # contextual attention
    sa = self.self_attention(x, mask)
    # positional attention
    pa = self.positional_attention(x, mask)
    # concatenate and combine
    return self.combine(torch.cat([sa, pa], dim=-1))
```

Listing 2: A code example of the Concatenation combination of positional and contextual features.

```
class Sequence(nn.Module):
""" Sequence Block """

  def __init__(self, config):
    super().__init__()
    # mode = ["Identical", "Attenuated"]
    self.pos_mode = config.pos_mode
    self.pos_learnable = config.pos_learnable
    self.self_attention = MultiHeadedSelfAttention(config)
    self.positional_attention = PositionalAttention(config, mode=self.pos_mode,
          learnable=self.pos_learnable)

  def forward(self, x, mask):
    # positional attention
    pa = self.positional_attention(x, mask)
    # contextual attention
    sa = self.self_attention(pa, mask)
```

Listing 3: A code example of the Sequence combination of positional and contextual features.

C.3 Results of Linguistic Probing

To understand the linguistic role of positional encodings, we adopt the probing tasks proposed by Conneau et al. (2018), which feeds sentence representations obtained by neural models to a series of linguistic classification tasks. We use the standard SentEval (Conneau & Kiela, 2018) to test every model. For obtaining sentence representations, we apply the *mean* pooling strategy to the output of each layer of a frozen language model, which is suggested by Reimers & Gurevych (2019). We follow the standard setup and run a grid search for selecting the layer with the best score on the dev set, and report the best score across layers for each probing task (Table 9), which represents the linguistic ability of each model. The following are the details of each probing task, including three categories:

- SentLen (Surface) aims to predict the length of sentences in terms of number of words.
- WC (Surface) means word content, which checks whether it is possible to recover information about the original word from the embedding of the sentence.
- BShift (Syntactic) means bigram shift. In this task, two random adjacent words in a sentence are swapped and the goal is to detect if a model is sensitive to legal word orders.
- TreeDepth (Syntactic) tests whether a model can infer the depth of the syntactic tree of sentences.
- TopConst (Syntactic) tests whether a model can recognize the top constituents of sentence, e.g., "*[Then] [very dark gray letters on a black screen] [appeared] [.]*" has top constituent sequence: "ADVP NP VP "
- Tense (Semantic) asks for the tense of the main-clause verb.
- SubjNum (Semantic) focuses on the number of the subject of the main clause.
- ObjNum (Semantic) tests for the number of the direct object of the main clause.

Relation	Distance	*contextual*	*positional*	*both*
Root	0.0	99.3	3.8	86.5
auxpass	1.1	44.6	91.1	92.9
compound	1.2	21.7	75.0	70.6
aux	1.3	25.2	77.9	79.1
nummod	1.3	26.8	78.9	75.5
amod	1.4	19.7	69.3	66.9
det	1.8	47.9	52.9	51.6
advmod	2.1	16.5	62.4	58.7
pobj	2.3	9.0	33.9	46.3
nsubj	2.3	13.4	58.2	52.6
poss	2.3	15.9	31.7	43.5
dobj	2.5	20.0	34.8	41.6
prep	2.6	23.1	62.8	63.4
npadvmod	3.1	14.4	30.0	43.8
cc	3.1	28.4	52.0	51.6
mark	4.1	25.1	21.3	64.4
conj	4.9	25.1	31.2	33.6
punct	7.3	25.9	30.3	32.7
advcl	7.6	18.4	9.8	20.1
ccomp	8.1	29.0	12.4	33.2
short	≤ 4	28.4	54.3	61.6
long	> 4	24.7	21.0	36.8
Macro Avg	-	27.5	46.0	55.4

Table 8: Evaluations of predictions of dependency relations on MRPC dataset. The top-20 common relations are shown. The distinction of *"short"* and *"long"* is whether the average length of the relation is greater than 4.

- SOMO (Semantic) checks the sensitivity of a model to random replacement of a noun or verb.
- CoordInv (Semantic) tests whether a model can recognize the order of clauses is inverted.

C.4 Dependency Analysis of Positional Encodings

The detailed scores of each relation are shown in Table 8. We find that positional attentional heads outperform contextual heads on short-distance relations, e.g., `auxpass` and `compound`. Contextual attention can capture better long-distance relations than positional attention while contextual attention itself has a certain degree of ability to detect some long-distance relations such as `conj` and `punct`. Note that there exists a head in contextual attention maps attending the token itself, therefore, the score on the `Root` relation is the best.

C.5 Visualizations

In Figure 2 and Figure 6, we visualize the averaged positional weights of various pre-trained language models and identify they have similar properties (Locality and Symmetry). However, We find that the behavior of positional encodings is actually very diverse across attention heads. Note that there are totally 144 attentional heads (12 layers $\times$ 12 heads) for the BERT$_{base}$ model. For example, the visualizations of BERT (Figure 8) validates this phenomenon. Besides, we observe that BERT exhibits a hierarchical structure: positional weights of lower layers are nearly uniform (Layer-4), middle layers attend more to local units (Layer-7) and higher layers demonstrate the asymmetric property (Layer-12). We also visualize all the positional heads in BERT-A* (Figure 9) and BERT-A*-*Seq* (Figure 10).

Visualization Analysis of BERT-A*. BERT-A* outperforms BERT by 3.1 percentage points on average across 10 downstream tasks. The main difference between BERT-A* and BERT is the learnable humancrafted positional encodings. For visualizations, we take the positional weight δ_{ij} in Equation 9 instead of Identical Word Probing.. The Figure 9 shows that most positional heads perfectly satisfy the properties of locality and symmetry, which can bring better inductive bias for sentence representations. Another observation is that the diagonal band widths are diverse across positional heads after learning, which means proximity units can be combined at different distances.

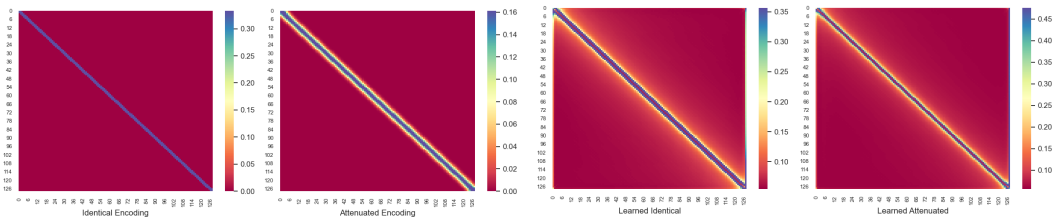

Figure 6: Visualizations of our handcrafted positional encodings. The attention weights are averaged across different layers. The left two are frozen while the right two are learnable during pre-training.

We conclude that, Compared to randomly initialized positional encodings, the encodings initialized with locality and symmetry properties lead to better sentence representation models.

Visualization Analysis of BERT-A*-*Seq*. Compared to BERT-A*, BERT-A*-*Seq* can further improve the scores on downstream tasks by combining positional and contextual features in a sequential way. Figure 10 visualize all positional heads in BERT-A*-*Seq*, we observe these matrices are still perfectly localized and symmetrical, which is similar to the behaviors of BERT-A*. We conclude that the Sequence combination can yield better better sentence representations than the traditional Add combination.

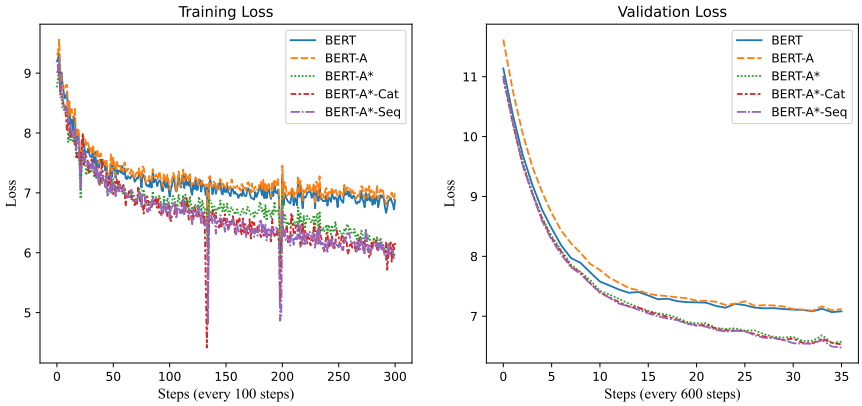

Figure 7: Loss curves of different variants adapted from BERT$_{base}$ model.

Model	Surface		Syntactic			Semantic				
	SentLen	WC	TreeDepth	TopConst	BShift	Tense	SubjNum	ObjNum	SOMO	CoordInv
BERT	**92.55**	23.54	**38.75**	**77.21**	**77.67**	**85.30**	**84.50**	**83.94**	**52.43**	**61.22**
BERT-*c*	89.22	**57.59**	32.86	56.65	51.16	82.94	78.97	81.63	51.86	50.35
BERT-*p*	89.64	38.27	33.78	63.15	53.68	67.17	71.17	66.73	51.19	57.42
TUPE	90.08	**85.76**	**39.14**	**78.38**	**79.44**	**87.13**	**86.50**	**86.36**	**53.78**	**62.60**
TUPE-*c*	**91.85**	62.42	34.21	58.63	50.86	83.43	80.63	81.96	51.75	50.74
TUPE-*p*	90.07	65.59	37.74	77.58	75.32	77.67	82.13	80.40	51.96	62.07

Table 9: Linguistic role probing of positional encodings. *c* and *p* means only using contextual and positional weights, respectively.

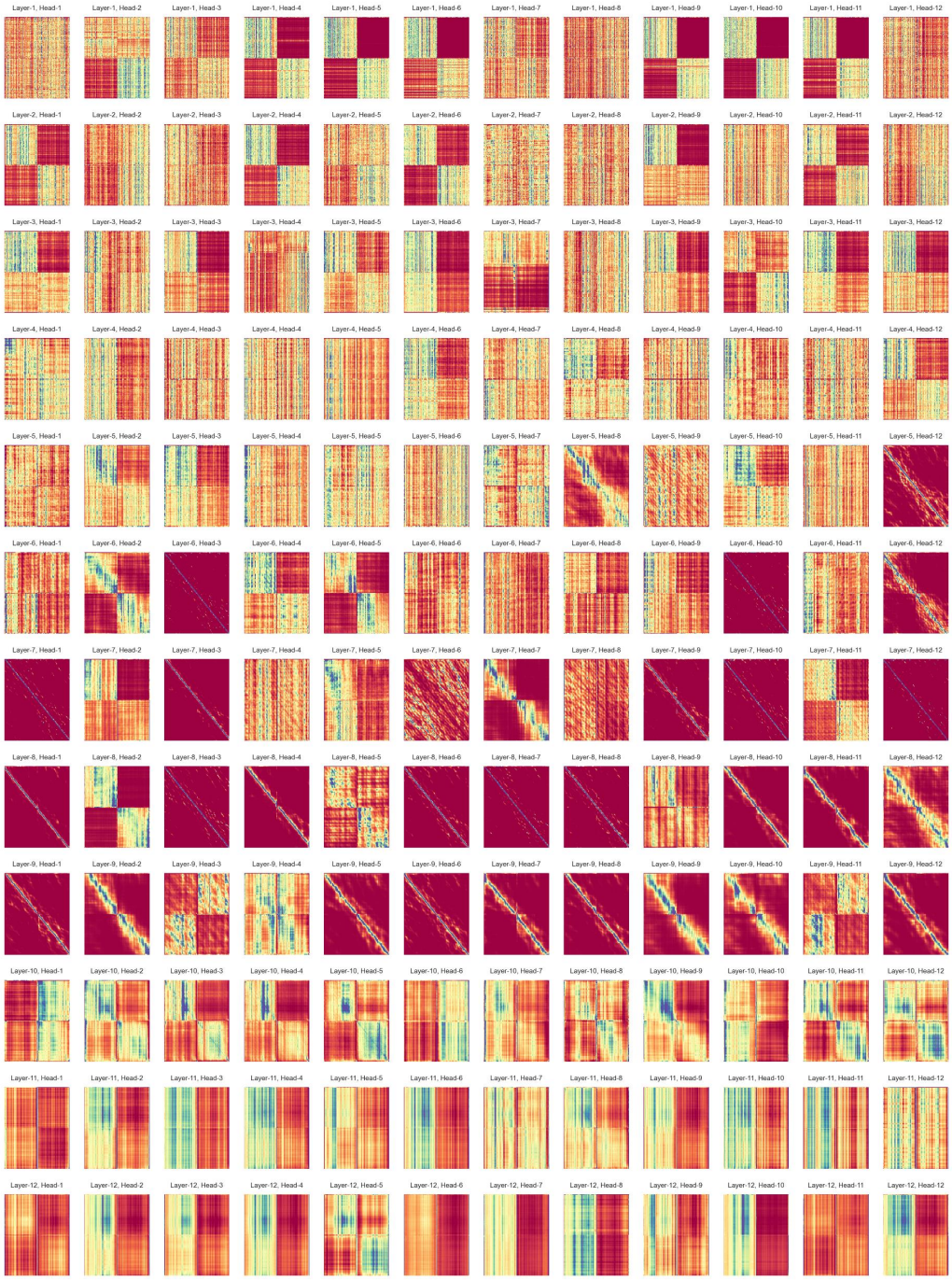

Figure 8: Visualizations of positional weights of BERT across all layers. The weights are computed by Identical Word Probing. Red color means lower values and blue color means higher values.

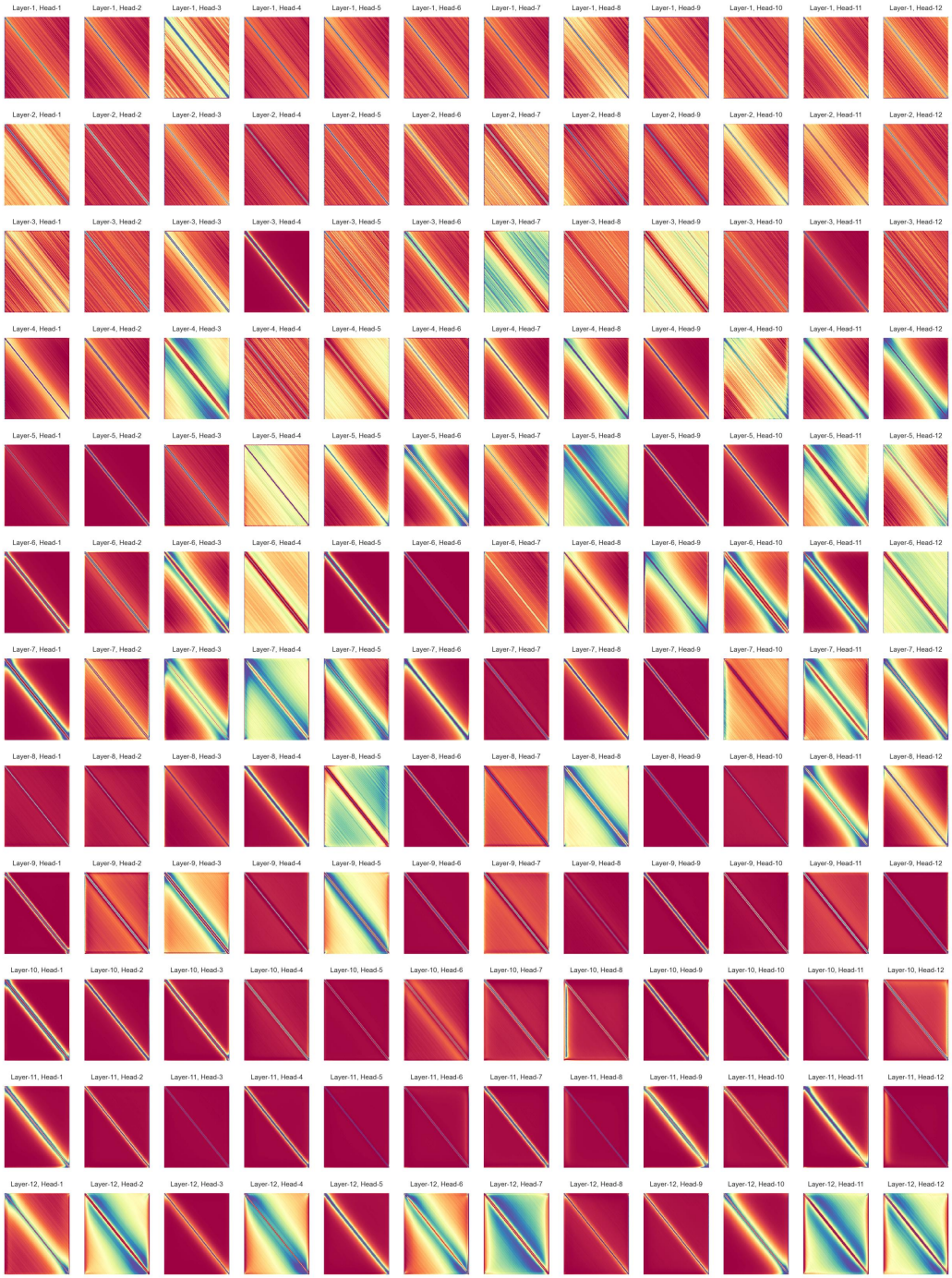

Figure 9: Visualizations of positional weights of BERT-A* across all layers. Red color means lower values and blue color means higher values.

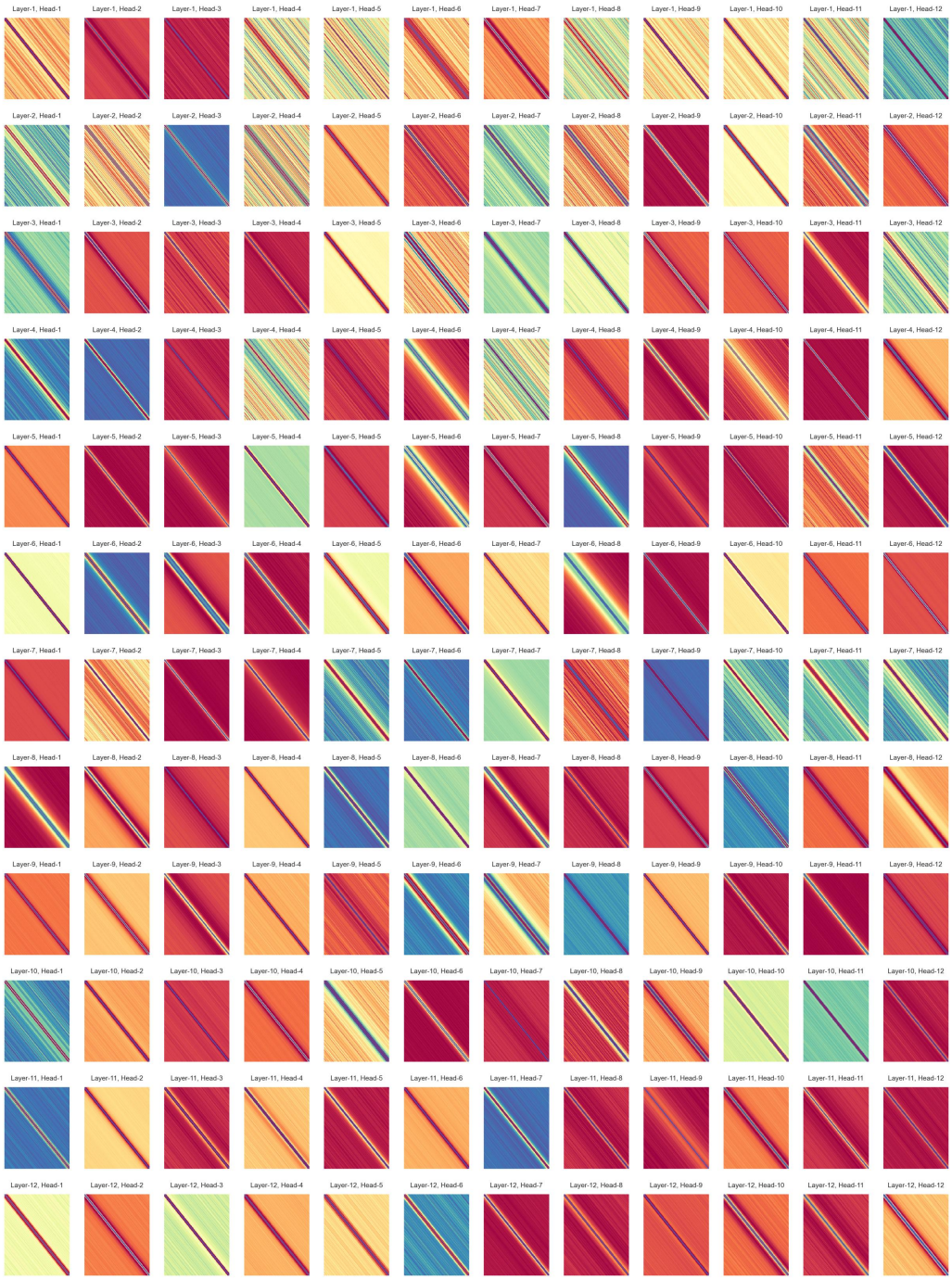

Figure 10: Visualizations of positional weights of BERT-A*-*Seq* across all layers. Red color means lower values and blue color means higher values.