Your Transformer May Not be as Powerful as You Expect

Shengjie Luo[1,5]*, **Shanda Li**[2]*, **Shuxin Zheng**[3], **Tie-Yan Liu**[3], **Liwei Wang**[1,4]†, **Di He**[1]†
[1]National Key Laboratory of General Artificial Intelligence,
School of Intelligence Science and Technology, Peking University
[2]Machine Learning Department, School of Computer Science, Carnegie Mellon University
[3]Microsoft Research [4]Center for Data Science, Peking University [5]Zhejiang Lab
luosj@stu.pku.edu.cn, shandal@cs.cmu.edu,
{shuz, tyliu}@microsoft.com, {wanglw,dihe}@pku.edu.cn

Abstract

Relative Positional Encoding (RPE), which encodes the relative distance between any pair of tokens, is one of the most successful modifications to the original Transformer. As far as we know, theoretical understanding of the RPE-based Transformers is largely unexplored. In this work, we mathematically analyze the power of RPE-based Transformers regarding whether the model is capable of approximating any continuous sequence-to-sequence functions. One may naturally assume the answer is in the affirmative—RPE-based Transformers are universal function approximators. However, we present a negative result by showing there exist continuous sequence-to-sequence functions that RPE-based Transformers cannot approximate no matter how deep and wide the neural network is. One key reason lies in that most RPEs are placed in the softmax attention that always generates a right stochastic matrix. This restricts the network from capturing positional information in the RPEs and limits its capacity. To overcome the problem and make the model more powerful, we first present sufficient conditions for RPE-based Transformers to achieve universal function approximation. With the theoretical guidance, we develop a novel attention module, called Universal RPE-based (URPE) Attention, which satisfies the conditions. Therefore, the corresponding URPE-based Transformers become universal function approximators. Extensive experiments covering typical architectures and tasks demonstrate that our model is parameter-efficient and can achieve superior performance to strong baselines in a wide range of applications. The code will be made publicly available at https://github.com/lsj2408/URPE.

1 Introduction

Transformer [61] is well acknowledged as a powerful neural network in modeling sequential data [12, 42]. Relative Positional Encoding (RPE) is one of the most successful modifications to the Transformer model [49]. Unlike the originally designed Absolute Positional Encoding (APE) that encodes each position as an embedding vector, RPE encodes the relative distance between any pair of tokens and is usually placed in the $\mathrm{softmax}$ exponentiation in the self-attention module. Empirically, many studies show that RPE-based Transformers can achieve impressive performance on various language tasks [54, 10] and have better extrapolation ability on longer sequences [52]. Another point worth noting is that RPE makes Transformer easily be extended to other data modalities,

*Equal contribution. The order is decided by rolling the dice.
†Correspondence to: Di He<dihe@pku.edu.cn>, Liwei Wang <wanglw@pku.edu.cn>.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

such as image [14, 43] and graph [67], as the relative distance naturally preserves invariant properties for several important transformations like rotation and translation.

In this paper, we first investigate the theoretical aspect of the RPE-based Transformers. In particular, we study their expressive power which describes models' ability to approximate any continuous functions. Recently, Yun et al. [69] proved that the APE-based Transformers are universal approximators of continuous sequence-to-sequence functions on a compact domain, and one may expect that the RPE-based Transformers enjoy the same property. However, we provide a surprising theoretical finding which shows that widely-used Transformers with RPE are *not* universal function approximators, i.e., there exist continuous sequence-to-sequence functions that the models cannot approximate no matter how deep and wide the model is. One key observation is that the RPEs are placed inside the $\mathrm{softmax}$ in the attention module. The $\mathrm{softmax}$ operator always generates a right stochastic matrix, which fails to reflect enough positional information encoded in RPE to the output. Synthetic tasks are conducted to support this mathematical claim.

To design a more powerful RPE-based Transformer, we delve into the limitation of the model and theoretically derive two sufficient conditions to achieve universal function approximation: the *attentive condition* and *position-aware condition*. Both conditions together state that the RPE-based attention function class should cover some special cases of the originally designed attention and break the right-stochastic-matrix limitation. With such theoretical guidance, we develop a new attention module called Universal RPE-based (URPE) Attention that satisfies the above conditions. Therefore, the Transformers with URPE-based Attention, called *URPE-based Transformers*, are universal function approximators. We show our proposed architecture is easy to implement and parameter-efficient via extensive experiments covering typical model architectures [54, 10, 67] and tasks (synthetic tasks, language modeling, and graph learning). Our model brings consistent performance gains compared with existing RPE-based Transformers on a wide range of tasks.

The paper is organized as follows. In Section 2, we introduce background on the Transformer architecture and positional encoding approaches. In Section 3, we prove that the widely-used RPE-based Transformers are *not* universal function approximators. In Section 4, we further present sufficient conditions for RPE-based Transformers to achieve universal approximation, and develop a new attention module, URPE-based Attention, to build a universal RPE-based Transformer. Experiments are presented in Section 5 to demonstrate the effectiveness of Transformers with our proposed URPE-based Attention. Related works and the conclusion are discussed in the last two sections.

2 Preliminary

The Transformer architecture is composed of stacked Transformer blocks [61, 12]. A Transformer block is a sequence-to-sequence mapping from $\mathbb{R}^{n\times d}$ to $\mathbb{R}^{n\times d}$, where n is the sequence length and d is the dimension of token embedding. A Transformer block consists of two layers: a self-attention layer followed by a feed-forward layer, with both layers having normalization (e.g., LayerNorm [1], RMSNorm [71]) and skip connections. For an input $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, the self-attention layer and feed-forward layer are defined as follows:

$$\boldsymbol{A}^h(\boldsymbol{X}) = \mathrm{softmax}\left(\boldsymbol{X}\boldsymbol{W}_Q^h(\boldsymbol{X}\boldsymbol{W}_K^h)^\top\right); \tag{1}$$

$$\mathrm{Attn}(\boldsymbol{X}) = \boldsymbol{X} + \sum_{h=1}^{H} \boldsymbol{A}^h(\boldsymbol{X})\boldsymbol{X}\boldsymbol{W}_V^h\boldsymbol{W}_O^h; \tag{2}$$

$$\mathrm{FFN}(\boldsymbol{X}) = \boldsymbol{X} + \mathrm{ReLU}(\boldsymbol{X}\boldsymbol{W}_1)\boldsymbol{W}_2, \tag{3}$$

where $\boldsymbol{W}_O^h \in \mathbb{R}^{d_H\times d}$, $\boldsymbol{W}_Q^h, \boldsymbol{W}_K^h, \boldsymbol{W}_V^h \in \mathbb{R}^{d\times d_H}$, $\boldsymbol{W}_1 \in \mathbb{R}^{d\times r}, \boldsymbol{W}_2 \in \mathbb{R}^{r\times d}$. H is the number of attention heads, d_H is the dimension of each head, and r is the dimension of the hidden layer. $\boldsymbol{A}^h(\boldsymbol{X})$ is usually referred to as the *attention matrix*. Given pre-defined H, d_H and r, we refer to the function class of the Transformer blocks as $\mathrm{T\_blocks}(H, d_H, r)$.

Transformer with Absolute Positional Encoding. Self-attention layers and feed-forward layers defined in Eq.(2) and (3) are entirely invariant to sequence order. Therefore, purely stacked Transformer blocks cannot distinguish information at different positions. The original Transformer [61] proposes Absolute Positional Encoding (APE) to endow Transformer networks with the ability to

capture positional information. In particular, a (learnable) real-valued embedding $e_i \in \mathbb{R}^d$ is assigned to each position i, leading to an Absolute Positional Encoding matrix $\boldsymbol{E} = [e_1, \cdots, e_n]^\top$, which will be added to the input sequence. Formally speaking, the function class represented by APE-based Transformers is

$$\Omega_{\mathrm{APE}}^{H,d_H,r} = \{f(\boldsymbol{X}) = g(\boldsymbol{X} + \boldsymbol{E}) | \boldsymbol{E} \in \mathbb{R}^{n\times d}; g = g_L \circ \cdots \circ g_1; g_i \in \mathrm{T\_blocks}(H, d_H, r); L \in \mathbb{N}^*\}.$$

APE essentially enhances the expressive power of Transformers. Yun et al. [69] proved the following theoretical result, which shows that APE-based Transformers can approximate any continuous sequence-to-sequence function in a compact domain.

Theorem 1 (informal [69]). *Given $n, d \in \mathbb{N}^*$, the function class of Transformers with APE, $\Omega_{\mathrm{APE}}^{2,1,4}$, is a universal approximator for continuous functions that map a compact domain in $\mathbb{R}^{n\times d}$ to $\mathbb{R}^{n\times d}$.*

Though Transformers with APE are conceptionally simple and enjoy good theoretical properties, they have a few known shortcomings. For example, Press et al. [52] showed that APE-based Transformers usually generalize poorly to longer sequences, as those positional embeddings for large indexes are hardly trained. Many works [58, 10, 54, 33, 26] employ Relative Positional Encoding (RPE), which becomes increasingly popular as a powerful way to encode positional information for Transformers and largely overcomes the disadvantages of APE.

Transformer with Relative Positional Encoding. Different from APE that assigns an embedding e_i for each position i, Relative Positional Encoding (RPE) encodes relative distance $i - j$ for each position pair (i, j). With the relative positional encoding, most previous works modified the attention computation defined in Eq.(1) as follows:

$$\boldsymbol{A}_{\mathrm{RPE}}^h(\boldsymbol{X}) = \mathrm{softmax}\left(\boldsymbol{X}\boldsymbol{W}_Q^h(\boldsymbol{X}\boldsymbol{W}_K^h)^\top + \boldsymbol{B}\right), \tag{4}$$

where $\boldsymbol{B}$ is an $n \times n$ matrix. The (i, j)-th entry of $\boldsymbol{B}$, denoted by b_{ij}, models the interaction between the i-th and j-th position. Different parameterizations of $\boldsymbol{B}$ lead to different model architectures. A few well-known examples include:

- Shaw's RPE [58]: $b_{ij} = \boldsymbol{X}_i\boldsymbol{W}_Q^h\boldsymbol{r}_{i-j}^\top$, where $\boldsymbol{r}_{i-j}$ are learnable vectors.
- T5 [54]: $b_{ij} = m_{i-j}$, where m_{i-j} are learnable scalars, i.e., $\boldsymbol{B}$ is parameterized as a Toeplitz matrix [22, 45].
- DeBERTa [25]: $b_{ij} = \boldsymbol{X}_i\boldsymbol{W}_Q^h\boldsymbol{r}_{i-j}^\top + \boldsymbol{s}_{i-j}(\boldsymbol{X}_j\boldsymbol{W}_K^h)^\top$, where $\boldsymbol{r}_{i-j}$ and $\boldsymbol{s}_{i-j}$ are learnable vectors.
- Transformer-XL [10]: $b_{ij} = \boldsymbol{X}_i\boldsymbol{W}_Q^h(\boldsymbol{r}_{i-j}\tilde{\boldsymbol{W}}_K^h)^\top + \boldsymbol{u}(\boldsymbol{X}_j\boldsymbol{W}_K^h)^\top + \boldsymbol{v}(\boldsymbol{r}_{i-j}\tilde{\boldsymbol{W}}_K^h)^\top$, where $\boldsymbol{u}, \boldsymbol{v}$ and $\tilde{\boldsymbol{W}}_K^h$ are all learnable vectors/matrix, and $\boldsymbol{r}_{i-j}$ are sinusoidal positional encoding vectors fixed during training.

Several interesting phenomena suggest that RPE-based Transformers have many advantages compared to their APE-based counterparts. Press et al. [52] demonstrated that RPE-based Transformers generalize better on longer sequences. T5 [54] and Transformer-XL [10] show that Transformers with RPE can achieve strong performance in language understanding and language generation tasks. Recently, RPEs are also popularly used in other domains to encode translation/rotation-invariant structural signals. Typical examples include Swin Transformer [43] and Graphormer [67], both of which use RPE and achieve state-of-the-art performance in image and graph representation learning.

3 Transformers with RPE are not Universal Approximators

We are interested in the expressive power of Transformers with RPE and investigate whether this architecture is as powerful as the original APE-based Transformers. To make comparison, we similarly define the function class of the Transformer blocks with RPE-based attention (Eq.(4)) as $\mathrm{T\_blocks}_{\mathrm{RPE}}(H, d_H, r)$, in which the relative positional encoding matrix $\boldsymbol{B}$ is assumed to be an *arbitrary parameterized mapping* from the input $\boldsymbol{X}$ to an $n \times n$ matrix. The function class represented by Transformers with RPE is defined as:

$$\Omega_{\mathrm{RPE}}^{H,d_H,r} = \{g_L \circ \cdots \circ g_1 : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d} | g_1, \cdots, g_L \in \mathrm{T\_blocks}_{\mathrm{RPE}}(H, d_H, r), L \in \mathbb{N}^*\}.$$

Surprisingly, we present a negative theoretical result: we prove that the function class of Transformers with RPE, $\Omega_{\mathrm{RPE}}^{H,d_H,r}$, is *not* a universal approximator for sequence-to-sequence functions.

Theorem 2. *Given $n > 2$, d and $\mathcal{D} \subseteq \mathbb{R}^{n\times d}$, assume that the all-zero matrix $\mathbf{0} \in \mathcal{D}$. For any $M > 0$, there exists a continuous function $\tilde{g}_M : \mathcal{D} \to \mathbb{R}^{n\times d}$, such that*

$$\sup_{\boldsymbol{X}\in\mathcal{D}} \|\tilde{g}_M(\boldsymbol{X}) - g(\boldsymbol{X})\|_F > M \tag{5}$$

holds for any $g \in \Omega_{\mathrm{RPE}}^{H,d_H,r}$, where $H, d_H, r \in \mathbb{N}^*$.

Proof. Without loss of generality, we prove the theorem for $d = 1$. The proof can be easily extended to $d > 1$ settings. Given $M > 0$, we consider a specific sequence-to-sequence function as target: $\tilde{g}_M : \boldsymbol{X} \mapsto (2M, 0, \cdots, 0)^\top$. To show $\sup_{\boldsymbol{X}\in\mathcal{D}} \|\tilde{g}_M(\boldsymbol{X}) - g(\boldsymbol{X})\|_F > M$ holds for any $g \in \Omega_{\mathrm{RPE}}^{H,d_H,r}$, we pick up an input $\boldsymbol{X}^*$, which is composed of n *identical* row vectors in $\mathbb{R}^d$, i.e., the sequence consists of n identical tokens. Since the function $\boldsymbol{A}_{\mathrm{RPE}}^h(\boldsymbol{X})$ outputs a right stochastic matrix, it is easy to check that $\mathrm{Attn}(\boldsymbol{X}) = \boldsymbol{X} + \sum_{h=1}^H \boldsymbol{A}^h(\boldsymbol{X})\boldsymbol{X}\boldsymbol{W}_V^h\boldsymbol{W}_O^h$ is also composed of n *identical* row vectors in $\mathbb{R}^d$. Note that $\mathrm{FFN}(\boldsymbol{X})$ and normalizations operate identically on each row vector, we can obtain that the final output of a Transformer block with RPE-based attention is still composed of n *identical* row vectors in $\mathbb{R}^d$.

Since g is a composition of multiple Transformer blocks in $\mathrm{T\_blocks}_{\mathrm{RPE}}(H, d_H, r)$ and $\mathbf{0}$ is composed of n *identical* row vectors in $\mathbb{R}^d$, we conclude from the analysis above that $g(\mathbf{0})$ is also composed of n *identical* row vectors in $\mathbb{R}^d$, i.e., there exists $c \in \mathbb{R}^d$ such that $g(\mathbf{0}) = c\mathbf{1}_n^\top$. Therefore, by applying Cauchy-Schwartz Inequality we obtain

$$\|\tilde{g}_M(\mathbf{0}) - g(\mathbf{0})\|_F^2 = (2M - c)^2 + (n-1)c^2 \geq \frac{4M^2}{1 + \frac{1}{n-1}} > M^2 \Rightarrow \sup_{\boldsymbol{X}\in\mathcal{D}} \|\tilde{g}_M(\boldsymbol{X}) - g(\boldsymbol{X})\|_F > M,$$

which completes the proof. □

Discussions. The key observation in Theorem 2 is that $\boldsymbol{A}_{\mathrm{RPE}}(\boldsymbol{X})$ always outputs a right stochastic matrix. Even if RPE carries rich positional information, such signal will be suppressed to satisfy $\boldsymbol{A}_{\mathrm{RPE}}(\boldsymbol{X})\mathbf{1} = \mathbf{1}$ for arbitrary $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, where $\mathbf{1}$ is an all-one n-dimensional vector. As a result, the attention module fails to reflect enough positional information encoded in RPE to the output, which restricts the model capacity. The problem will be significant when the target function $\tilde{g}$ is very position-sensitive (in the extreme case, $\tilde{g}$ only depends on the position indexes). We also conduct experiments on simple sequence-to-sequence tasks using synthetic data to support this mathematical claim in Section 5.1. One may expect that simply removing the denominator in the $\mathrm{softmax}$ can break the limitation. However, this modification brings significant optimization instability in practice. Given that RPE has many advantages[3] compared to APE, it's appealing to design an RPE-based Transformer variant that is a universal approximator of sequence-to-sequence functions and easy to optimize. This is what we precisely work on in the next section.

4 Making RPE-based Transformers Universal Approximators

This section contains two sub-sections. In the first sub-section, we provide a sufficient condition for the RPE-based Transformers to achieve universal approximation. In the second sub-section, we offer a practical instantiation of the Transformer with RPE that satisfies the requirement and is parameter-efficient.

[3]On one hand, we claim that RPE-based Transformers cannot achieve universal function approximation (while APE-based models can achieve). On the other hand, we claim RPE is empirically advantageous compared to APE, according to previous works. One may feel there exists a contradiction and get confused. We would like to clarify that the two claims are made in different settings. For example, RPE empirically performs well in generalization to longer sequences, while the theoretical analysis of approximation capability focuses on functions with bounded input lengths (See Theorem 1 where n is given). Our goal is to design an RPE variant with the same expressiveness as APE in the theoretical setting and enjoys its usefulness in practical scenarios.

4.1 A Sufficient Condition to Achieve Universal Approximation

Motivated by formulation (1) and (4), we consider a general form $\boldsymbol{A}_U^h : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times n}$ and define the corresponding attention layer as

$$\mathrm{Attn}_{\mathrm{U}}(\boldsymbol{X}) = \boldsymbol{X} + \sum_{h=1}^{H} \boldsymbol{A}_{\mathrm{U}}^h(\boldsymbol{X})\boldsymbol{X}\boldsymbol{W}_V^h\boldsymbol{W}_O^h, \tag{6}$$

We further define the function class of the corresponding Transformer block as $\mathrm{T\_blocks}_{\mathrm{U}}(H, d_H, r)$, and define the function class of Transformers composed of stacked $\mathrm{T\_blocks}_{\mathrm{U}}(H, d_H, r)$ as:

$$\Omega_{\mathrm{U}}^{H,d_H,r} = \{g_L \circ \cdots \circ g_1 : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d} \mid g_1, \cdots, g_L \in \mathrm{T\_blocks}_{\mathrm{U}}(H, d_H, r), L \in \mathbb{N}^*\}. \tag{7}$$

Our goal is to investigate the requirements on $\boldsymbol{A}_{\mathrm{U}}^h$ under which the induced function class $\Omega_{\mathrm{U}}^{H,r}$ can become universal approximators of continuous sequence-to-sequence functions. We provide one sufficient condition in the following theorem. Following Yun et al. [69] and many other previous theoretical works [44, 24, 70], we study the expressiveness of a simplified version of Transformer in which normalization layers are omitted, as it is widely believed that normalization mainly helps optimization but does not hurt the expressive power of the network [32, 1].

Theorem 3. *Given $n, d \in \mathbb{N}^*$, $p \in [1, +\infty)$, $\varepsilon > 0$, a compact set $\mathcal{D} \subseteq \mathbb{R}^{n\times d}$, and a continuous sequence-to-sequence function $f : \mathcal{D} \to \mathbb{R}^{n\times d}$. Assume that $\boldsymbol{A}_U^h$ satisfies the following conditions:*

- ***Attentive condition.*** *For any $\boldsymbol{u} \in \mathbb{R}^{d\times 1}$ and $c \in \mathbb{R}$, there exists a parametrization of $\boldsymbol{A}_U^h$, such that $\boldsymbol{A}_U^h(\boldsymbol{X}) = \mathrm{softmax}\left(\boldsymbol{X}\boldsymbol{u}(\boldsymbol{X}\boldsymbol{u} - c\boldsymbol{1})^\top\right)$.*
- ***Position-aware condition.*** *There exists a parametrization of $\boldsymbol{A}_U^h$ and a vector $\boldsymbol{v} \in \mathbb{R}^n$ whose entries are all distinct, such that $\boldsymbol{A}_U^h(\boldsymbol{X})\boldsymbol{1} = \boldsymbol{v}$ for any $\boldsymbol{X} \in \mathbb{R}^{n\times d}$.*

Then there exists a Transformer network $g \in \Omega_{\mathrm{U}}^{2,1,4}$ such that $\left(\int_{\mathcal{D}} \|f(\boldsymbol{X}) - g(\boldsymbol{X})\|_p^p \mathrm{d}\boldsymbol{X}\right)^{\frac{1}{p}} < \varepsilon$, where $\|\cdot\|_p$ denotes the entry-wise ℓ^p norm for matrices.

The detailed proof of Theorem 3 can be found in Appendix A. Theorem 3 presents two conditions to make RPE-based Transformers become universal approximators. Intuitively, the *attentive condition* states that $\boldsymbol{A}_{\mathrm{U}}^h$ should contain a special case of the original attention matrix in Eq.(1), where $\boldsymbol{W}_Q = \boldsymbol{W}_K \in \mathbb{R}^{d_H\times d}$ and $d_H = 1$. The *position-aware condition* states that $\boldsymbol{A}_U^h$ needs to break the limitation of $\boldsymbol{A}(\boldsymbol{X})$ being a right stochastic matrix (i.e., $\boldsymbol{A}(\boldsymbol{X})\boldsymbol{1} = \boldsymbol{1}$ for all $\boldsymbol{X} \in \mathbb{R}^{n\times d}$). We will present a concrete example satisfying these conditions in the next subsection.

4.2 A Universal RPE-based Transformer

We develop a new Transformer variant which satisfies the conditions above. In particular, we multiply the softmax attention matrix with another matrix, and obtain

$$\boldsymbol{A}_{\mathrm{U}}(\boldsymbol{X}) = \mathrm{softmax}\left(\boldsymbol{X}\boldsymbol{W}_Q(\boldsymbol{X}\boldsymbol{W}_K)^\top + \boldsymbol{B}\right) \odot \boldsymbol{C}, \tag{8}$$

where $\odot$ denotes entry-wise product. $\boldsymbol{B}$ can take any form described in Section 2. We refer to this attention variant as **Universal RPE-based Attention** (URPE-based attention).

To make URPE-based attention rely relative positional information, we set $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ to be a learnable Toeplitz matrix in which each element on the descending diagonal from left to right has the same value. Note that a Toeplitz matrix of shape $n \times n$ has $2n - 1$ degrees of freedom. Therefore, we only need $2n - 1$ new parameters for each $\boldsymbol{C}$. It can be proved that URPE-based Attention satisfies the two conditions in Theorem 3 and the proof can be found in Appendix B.

Proposition 4. *URPE-based Attention defined in Eq.(8) satisfies the conditions in Theorem 3. Consequently, given $n, d \in \mathbb{N}^*$, Transformers using this form of attention are universal approximators of continuous sequence-to-sequence functions that map a compact domain in $\mathbb{R}^{n\times d}$ to $\mathbb{R}^{n\times d}$.*

To further improve parameter efficiency, we set $\boldsymbol{C}$ to be shared across different layers but to be unique for each attention head. As an example, in Transformer-XL (see Section 5.2), we introduce only about

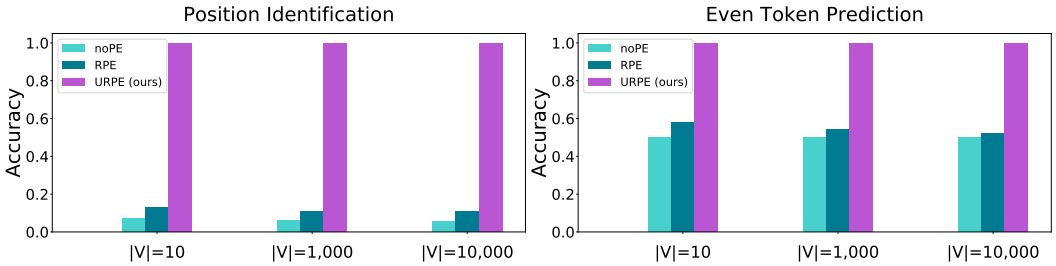

Figure 1: Results on synthetic sequence-to-sequence tasks: (1) Position Identification (Left Panel); (2) Even Token Prediction (Right Panel). $|V|$ is the vocabulary size. The URPE-based Transformer model consistently solves both tasks across different settings while other methods fail.

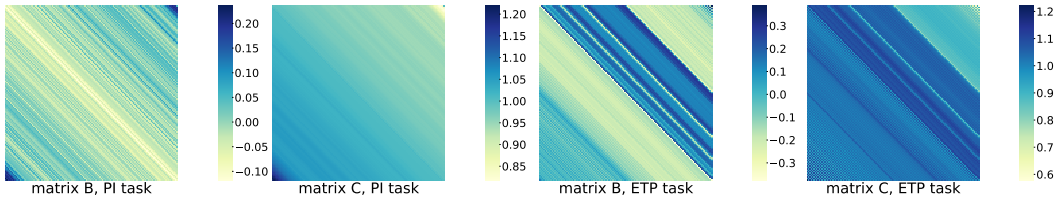

Figure 2: Visualizations of the learned Universal RPE (matrix B and C in Eq.(8)). It can be easily seen that the matrix $\boldsymbol{B}$ and $\boldsymbol{C}$ capture different aspects of positional information.

4K new parameters, which is negligible compared to the 151M parameters of the Transformer-XL model but still leads to non-trivial improvements.

We make two more discussions for our designed URPE-based attention. The first one is about applying URPE-based attention in the causal setting. Causal attention is important in language generation tasks. URPE-based Attention is compatible with it as one can set the (i, j)-th entry of $\boldsymbol{C}$ to be 0 for $i > j$. The second one is on the initialization of C. In practice, the matrix $\boldsymbol{C}$ can be initialized as an all-one matrix. Therefore, the model behaves identically to the original RPE-based model at initialization, and the additional parameters in $\boldsymbol{C}$ are learned gradually. We can also fine-tune any well-trained RPE-based Transformer to its URPE-based counterpart by setting $\boldsymbol{C}$ as an all-one matrix and further fine-tuning the model to learn $\boldsymbol{C}$.

5 Experiments

In this section, we empirically study the effectiveness of the proposed model. In particular, we aim at answering the following questions through experiments:

- **Question 1**: Can the theoretical results on the approximation capability of RPE-based Transformer and URPE-based Transformer be reflected in certain experiments?
- **Question 2**: With different RPE methods (the matrix B in Eq.(8)), can URPE-based Transformer outperform its RPE-based counterpart in real-world applications?
- **Question 3**: Can URPE-based Attention serve as a versatile module to improve the general Transformers beyond language tasks?

We will answer each question with carefully designed experiments in the following sub-sections. Due to space limitation, we only present results on representative model architectures, task types, and data modalities in the main body of the paper. More results are presented in the appendix. All codes are implemented based on the official codebases of Fairseq [50] and Graphormer [67] in PyTorch [51].

5.1 Synthetic Tasks

To empirically verify our theoretical results on the approximation capability of RPE-based Transformer and URPE-based Transformer, we design two synthetic tasks: 1) Position Identification; 2) Even Token Prediction.

Table 1: Language model perplexity scores on WikiText-103 validation and test set. We use * to indicate the best performance. All the results of the baseline methods are reported in [10]

Model	#Params	Valid Perplexity	Test Perplexity
LSTM [21]	-	/	48.7
TCN [2]	-	/	45.2
GCNN-8 [11]	-	/	44.9
LSTM+Neural cache [21]	-	/	40.8
GCNN-14 [11]	-	/	37.2
QRNN [46]	151M	/	33.0
Hebbian+Cache [53]	-	/	29.9
Transformer-XL Base [10]	151M	23.1	24.0
Transformer-XL Base + URPE-based Attention (ours)	151M	22.4*	23.2*

Both Position Identification (PI) task and Even Token Prediction (ETP) task are sequence-to-sequence prediction tasks. Given a sequence of tokens $s = (w_1, w_2, \cdots, w_n)$, the PI task is to predict the position index of each token in the sequence, i.e., the target sequence-to-sequence function f to approximate can be defined as

$$f_{\mathrm{PI}}(w_1, w_2, \cdots, w_n) = (1, 2, \cdots, n) \tag{9}$$

The ETP task is defined as follows: for the first half of positions in a sequence, the task requires the model to output the input tokens at positions with even number index; for the remaining half of positions, the task requires the model to output the special token End-Of-Sentence (EOS), i.e.,

$$f_{\mathrm{ETP}}(w_1, w_2, \cdots, w_n) = (w_2, w_4, \cdots, w_n, \mathrm{EOS}, \cdots, \mathrm{EOS}) \tag{10}$$

Both tasks require the model to accurately encode the positional information, which would be difficult for RPE-based Transformers to capture. For both tasks, we use synthetic datasets with randomly generated sequences. In detail, we vary the token vocabulary size from [10, 1000, 10000] and set the sequence length to 128. We choose the vanilla Transformer as the base model and compare the following ways to encode positional information: 1) no positional encodings (noPE); 2) T5-style relative positional encoding (RPE) [54]; 3) URPE with T5-style RPE backbone Transformer. The number of layers and the number of attention heads are set to 3 and 12, respectively. The hidden dimension is set to 768.

Results. We use token-level accuracy as the evaluation metric. The experimental results are shown in Figure 1. From the figure, it can be easily seen that the Transformer without PE and the Transformer with T5-style RPE cannot perfectly solve the synthetic tasks (less than 60% accuracy). On the contrary, the URPE-based Transformer achieves 100% accuracy on both tasks. Firstly, this result clearly indicates that our proposed model outperforms the backbone T5-style RPE-based Transformer by a large margin. Furthermore, we can see that even for such simple tasks, the Transformer with T5-style RPE sometimes fails, while the URPE-based Transformer succeeds and approximates the target function well, which is consistent with our theoretical findings. Lastly, we provide visualizations of the learned Universal RPE (Eq.(8)) on both tasks in Figure 2, which show that the matrix $\boldsymbol{B}$ and $\boldsymbol{C}$ capture different aspects of positional information.

5.2 Language Modeling

We use language modeling to study the effectiveness of the proposed URPE-based Attention. Language modeling is an important practical application which usually requires the modelling of long-term dependency between tokens. We conduct experiments on the WikiText-103 dataset [47], which contains 103M training tokens from 28K articles, with an average length of 3.6K tokens per article. Relative positional encoding methods are popularly used in language modelling. We choose Transformer-XL model [10] as the backbone model of our URPE-based Transformer. Following [10], the number of layers and the number of attention heads are set to 16 and 10 respectively. The dimension of hidden layers and feed-forward layers are set to 410 and 2100. The detailed descriptions of the baselines and training settings are presented in the appendix.

Table 2: Mean Absolute Error (MAE) on ZINC test set. We use * to indicate the best performance.

Model	#Params	Test MAE on ZINC-Subset	Test MAE on ZINC-Full
GIN [66]	509,549	0.526±0.051	0.088±0.002
GraphSAGE [23]	505,341	0.398±0.002	0.126±0.003
GAT [62]	531,345	0.384±0.007	0.111±0.002
GCN [35]	505,079	0.367±0.011	0.113±0.002
MoNet [48]	504,013	0.292±0.006	0.090±0.002
GatedGCN-PE [5]	505,011	0.214±0.006	-
MPNN(sum) [20]	480,805	0.145±0.007	-
HIMP [17]	614,516	0.151±0.006	0.036±0.002
PNA [8]	387,155	0.142±0.010	-
GT [15]	588,929	0.226±0.014	-
SAN [37]	508,577	0.139±0.006	-
Graphormer [67]	489,321	0.122±0.006	0.052±0.005
Graphormer+URPE-based Attention (ours)	491,737	0.086±0.007*	0.028±0.002*

Table 3: Results on PCQM4M from OGB-LSC. We use * to indicate the best performance. The results of the baselines are reported in [67, 29].

Model	#Params	Valid MAE
GCN [35]	2.0M	0.1691
GIN [66]	3.8M	0.1537
GCN-VN [35, 20]	4.9M	0.1485
GIN-VN [66, 20]	6.7M	0.1395
GINE-VN [6, 20]	13.2M	0.1430
DeeperGCN-VN [38, 20]	25.5M	0.1398
GT [15]	0.6M	0.1400
GT-Wide [15]	83.2M	0.1408
Graphormer [67]	12.5M	0.1264
Graphormer + URPE-based Attention (ours)	12.5M	0.1238*

Results. We show the perplexity scores on both validation and test set of different models in Table 1. It can be easily seen that the Transformer-XL equipped with our URPE-based attention achieves 22.4 and 23.2 valid and test perplexity scores, respectively, which are 0.7 and 0.8 lower than the backbone Transformer-XL model and also significantly better than other baselines. First, the results suggest that our proposed URPE-based attention can be well applied to Transformer-XL in real-world applications. Together with the observations in Section 5.1, we believe our URPE can be used in other RPE-based architectures, such as [58, 26]. It is worth noting that our model has negligible more parameters (about 4k) compared to the backbone Transformer-XL. Thus, the improvement in the perplexity should be mostly attributed to the stronger expressiveness of the model.

5.3 Graph Learning

We further examine whether the proposed URPE-based Attention can serve as a versatile module to improve the general RPE-based Transformers beyond language tasks. The Transformer-based models have become increasingly popular in the graph learning area [67, 37, 15]. Among those models, the recently proposed Graphormer [67] achieves state-of-the-art performance in many graph learning tasks [68, 59]. In Graph Transformers, RPE is used rather than APE since RPE only calculates distances between nodes, which naturally preserves many invariant and equivariant properties.

The attention computation in Graphormer also follows the Eq.(4) in Section 2. Specifically, Graphormer calculates the shortest-path distance between any pair of nodes and encodes this information as a bias term in the softmax attention to reflect the relative position of any node in the

Table 4: Comparison of RPE-based and UPRE-based Transformer-XL models of different sizes. We report perplexity scores on WikiText-103 validation set. L denotes the number of layers.

Model	$L = 4$	$L = 8$	$L = 16$
Transformer-XL	29.6	26.0	23.1
Transformer-XL + URPE-based Attention (ours)	28.7	25.2	22.4

Table 5: Inference Runtime (ms in log base 2) and Peak Memory Usage (GB) of RPE-based Transformer and URPE-based Transformer. N denotes the input sequence length.

Model	Inference Runtime			Peak Memory Usage		
	$N = 128$	$N = 256$	$N = 512$	$N = 128$	$N = 256$	$N = 512$
RPE-based Transformer	4.55	5.60	6.79	0.96	1.12	1.86
URPE-based Transformer (ours)	4.59	5.66	6.91	0.97	1.17	2.04

graph. We refer the readers to [67] for the detailed description of Graphormer. Similar to previous experiments, we adapt the URPE-based Attention to the Graphormer and compare them on two benchmark datasets covering graph representation learning tasks from small to large scale datasets: ZINC from Benchmarking-GNNs [16] and PCQM4M from Open Graph Benchmark Large Scale Challenge (OGB-LSC) [29]. For both tasks, we choose several competitive Transformer based models and GNNs as our baselines. Details of the experimental settings are presented in the appendix.

ZINC. ZINC is a real-world dataset which consists of 250K molecular graphs. The task is to predict the constrained solubility of a molecule which is an important chemical property for drug discovery. We train our models on both the ZINC-Full and ZINC-Subset (12K selected graphs following [16]). To demonstrate the power of our method and for fair comparison, we set the parameter budget of the model to be less than 500K following [16, 67]. We build on the Graphormer [67] model which consists of 12 layers. The dimension of hidden layers and feed-forward layers are set to 80. The number of attention heads are set to 32.

PCQM4M. PCQM4M is a quantum chemistry regression task in OGB-LSC [29]. The PCQM4M dataset contains more than 3.8 million molecular graphs in total, which is currently the largest graph-level prediction dataset. The state-of-the-art architecture for this task is the Graphormer model introduced above. We still follow [67] to set the hyper-parameters in the Graphromer model and equip it with URPE. In detail, our Graphormer with URPE-based Attention consists of 6 layers and 32 attention heads. The dimension of hidden layers and feed-forward layers are set to 512.

Results. The experimental results on ZINC and PCQM4M are shown in Table 2 and 3, where the score is averaged over four experiments with different seeds. It can be easily seen that the Graphormer model equipped with our URPE-based Attention consistently outperforms the backbone Graphormer model on both ZINC and PCQM4M tasks. In particular, our URPE-based Attention enables the Graphormer model to reduce more than 40% relative mean absolute error on the test set of ZINC-Subset and ZINC-Full. On the PCQM4M task, the improvement is around 0.003 mean absolute error which is also a significant improvement under the quantum chemistry precision. It is worth noting that our Graphormer model with URPE-based Attention achieves competitive performance compared to the Graphormer Base model with 48.3M parameters reported in [67]. Therefore, we believe our proposed architecture significantly improves the expressive power of the Transformer backbones and can be well extended to practical scenarios beyond language tasks.

5.4 More Analyses

URPE-based Attention consistently improves the performance of models of different sizes. We conduct ablation experiments on the Language Modeling task and vary the number of layers in [4, 8, 16] to investigate different model sizes. Following the experiment settings in Section 5.2, the number of attention heads is set to 10. The hidden dimension is set to 41. The dimension of the feed-forward layer is set to 2100. The results are presented in Table 4. It can be easily seen that our URPE-based Attention consistently reduces the perplexity scores of Transformer-XL models of different sizes, which indeed demonstrates the versatility of our URPE-based Attention.

Runtime and Memory Usage Evaluation. We further conduct memory and time costs profiling experiments on our URPE-based Transformers. We choose the vanilla Transformer as the backbone model. The number of layers and the hidden dimension are set to 12 and 768 respectively. The number of attention heads is set to 12. The batch size is set to 32. We vary the sequence length from [128, 256, 512]. We run profiling of all the models on a 16GB NVIDIA Tesla V100 GPU. Following Combiner [55], we compare the inference speed and memory costs of the vanilla Transformer with RPE and our URPE. The results are presented in Table 5, which show that our URPE only increases minor computational costs.

Summary. In this section, we design a series of experiments to answer the questions on the effectiveness and applicability of the proposed URPE-based Attention. All the experimental results suggest that our theoretical findings are convincing, and Transformers with our modification are effective, powerful, and widely applicable across different tasks.

6 Related Work

Expressive power of Neural Networks. Quantifying the capacity of neural networks is an important research direction in the literature on deep learning. [9, 19, 27, 4] showed that a neural network with one hidden layer and unbounded width can approximate arbitrary continuous functions on compact support with arbitrarily small error. Many works also studied the width efficiency on the expressive power of neural networks [44, 24, 40] and proved that ReLU networks with a bounded width but unlimited depth can achieve universal function approximation. Recently, there has been increasing interest in the theoretical understanding of Transformer models. Yun et al. [69] theoretically showed that Transformers can approximate any continuous sequence-to-sequence functions (i.e., universal approximation) on a compact domain by proving that stacking of self-attention layers can compute contextual mappings of the input embeddings. Dong et al. [13] analyzed the limitations of attention-only Transformers without considering FFN blocks, normalizations, and skip connections. Hron et al. [28] analyzed the behavior of multi-head attention and connect it with the Gaussian process when the number of heads tends to be infinity. In [7], it is proved that a multi-head attention layer with enough heads is at least as expressive as any convolution layer. All works above consider Transformers with absolute positional encoding, which is the main difference between them and our work.

Positional encoding methods in Transformers. In [61], the vanilla Transformer encodes the positional information via the absolute positional encoding (APE). Shaw et al. [58] is the first to introduce relative positional encoding (RPE) to Transformer. From then on, many works explored different RPE strategies based on [58]. Transformer-XL [10] re-parameterizes the self-attention to integrate relative positional encoding and enables long sequence modelling. T5 [54] simplifies the vector representations of relative positions to scalars. Kitaev et al. [36] disentangles the positional and content information in the Transformer encoder, which leads to an improved constituency parser. Ke et al. [33] further shows that such disentanglement also improves Transformer in general language pre-training and achieves superior performance on various downstream tasks. There are also works that encodes the positional information via other tools like trees [60], complex numbers [63], dynamic systems [41], Fourier features [39]. Compared to most previous works inspired by practical scenarios, our URPE-based attention is theoretically motivated by investigating the expressive power of RPE-based Transformers in a principled way.

7 Conclusion

In this paper, we first investigate the theoretical aspect of the RPE-based Transformers. In particular, we study their expressive power and provide a surprising theoretical finding which shows that widely-used Transformers with RPE are *not* universal function approximators. To design a more powerful RPE-based Transformer, we present sufficient conditions on the attention module to achieve universal function approximation and develop a novel *Universal RPE-based Transformer* using a new relative positional encoding approach. We conduct carefully designed experiments on synthetic sequence data, natural language, and graphs to show that our model brings consistent performance gains compared with existing RPE-based Transformers. In the future, we will benchmark our Universal RPE on other RPE-based Transformers and typical tasks to further verify its versatility as a basic module for Transformer-based models.

Acknowledgements

We thank Tianle Cai and Jianlin Su for the helpful discussions. We also thank all the anonymous reviewers for the very careful and detailed reviews as well as the valuable suggestions. Their help has further enhanced our work. This work is supported by National Science Foundation of China (NSFC62276005), The Major Key Project of PCL (PCL2021A12), Exploratory Research Project of Zhejiang Lab (No. 2022RC0AN02), and Project 2020BD006 supported by PKUBaidu Fund.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[3] Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Songhao Piao, Jianfeng Gao, Ming Zhou, et al. Unilmv2: Pseudo-masked language models for unified language model pre-training. *arXiv preprint arXiv:2002.12804*, 2020.

[4] Andrew R Barron. Approximation and estimation bounds for artificial neural networks. *Machine learning*, 14(1):115–133, 1994.

[5] Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

[6] Rémy Brossard, Oriel Frigo, and David Dehaene. Graph convolutions that can finally model local structure. *arXiv preprint arXiv:2011.15069*, 2020.

[7] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In *International Conference on Learning Representations*, 2020.

[8] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271, 2020.

[9] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[10] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, 2019.

[11] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR, 2017.

[12] 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.

[13] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*, pages 2793–2803. PMLR, 2021.

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[15] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.

[16] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[17] Matthias Fey, Jan-Gin Yuen, and Frank Weichert. Hierarchical inter-message passing for learning on molecular graphs. *arXiv preprint arXiv:2006.12179*, 2020.

[18] Gerald B Folland. *Real analysis: modern techniques and their applications*, volume 40. John Wiley & Sons, 1999.

[19] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192, 1989.

[20] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR, 2017.

[21] Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*, 2016.

[22] Robert M Gray. *Toeplitz and circulant matrices: A review*. now publishers inc, 2006.

[23] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[24] Boris Hanin and Mark Sellke. Approximating continuous functions by relu nets of minimal width. *arXiv preprint arXiv:1710.11278*, 2017.

[25] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[26] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2021.

[27] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[28] Jiri Hron, Yasaman Bahri, Jascha Sohl-Dickstein, and Roman Novak. Infinite attention: Nngp and ntk for deep attention networks. In *International Conference on Machine Learning*, pages 4376–4386. PMLR, 2020.

[29] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*, 2021.

[30] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016.

[31] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin Benson. Combining label propagation and simple models out-performs graph neural networks. In *International Conference on Learning Representations*, 2020.

[32] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[33] Guolin Ke, Di He, and Tie-Yan Liu. Rethinking positional encoding in language pre-training. In *International Conference on Learning Representations*, 2021.

[34] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[35] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[36] Nikita Kitaev and Dan Klein. Constituency parsing with a self-attentive encoder. *arXiv preprint arXiv:1805.01052*, 2018.

[37] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34, 2021.

[38] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020.

[39] Yang Li, Si Si, Gang Li, Cho-Jui Hsieh, and Samy Bengio. Learnable fourier features for multi-dimensional spatial positional encoding. *Advances in Neural Information Processing Systems*, 34, 2021.

[40] Hongzhou Lin and Stefanie Jegelka. Resnet with one-neuron hidden layers is a universal approximator. *Advances in neural information processing systems*, 31, 2018.

[41] 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.

[42] 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.

[43] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[44] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. *Advances in neural information processing systems*, 30, 2017.

[45] Shengjie Luo, Shanda Li, Tianle Cai, Di He, Dinglan Peng, Shuxin Zheng, Guolin Ke, Liwei Wang, and Tie-Yan Liu. Stable, fast and accurate: Kernelized attention with relative positional encoding. *Advances in Neural Information Processing Systems*, 34, 2021.

[46] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. An analysis of neural language modeling at multiple scales. *arXiv preprint arXiv:1803.08240*, 2018.

[47] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[48] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017.

[49] Sharan Narang, Hyung Won Chung, Yi Tay, William Fedus, Thibault Fevry, Michael Matena, Karishma Malkan, Noah Fiedel, Noam Shazeer, Zhenzhong Lan, et al. Do transformer modifications transfer across implementations and applications? *arXiv preprint arXiv:2102.11972*, 2021.

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

[51] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[52] 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*, 2022.

[53] Jack Rae, Chris Dyer, Peter Dayan, and Timothy Lillicrap. Fast parametric learning with activation memorization. In *International Conference on Machine Learning*, pages 4228–4237. PMLR, 2018.

[54] 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.

[55] Hongyu Ren, Hanjun Dai, Zihang Dai, Mengjiao Yang, Jure Leskovec, Dale Schuurmans, and Bo Dai. Combiner: Full attention transformer with sparse computation cost. *Advances in Neural Information Processing Systems*, 34:22470–22482, 2021.

[56] Emanuele Rossi, Fabrizio Frasca, Ben Chamberlain, Davide Eynard, Michael Bronstein, and Federico Monti. Sign: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198*, 2020.

[57] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

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

[59] Yu Shi, Shuxin Zheng, Guolin Ke, Yifei Shen, Jiacheng You, Jiyan He, Shengjie Luo, Chang Liu, Di He, and Tie-Yan Liu. Benchmarking graphormer on large-scale molecular modeling datasets. *arXiv preprint arXiv:2203.04810*, 2022.

[60] Vighnesh Shiv and Chris Quirk. Novel positional encodings to enable tree-based transformers. In *Advances in Neural Information Processing Systems*, pages 12058–12068, 2019.

[61] 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*, pages 6000–6010, 2017.

[62] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[63] 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*, 2019.

[64] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413, 2020.

[65] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[66] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[67] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34, 2021.

[68] Chengxuan Ying, Mingqi Yang, Shuxin Zheng, Guolin Ke, Shengjie Luo, Tianle Cai, Chenglin Wu, Yuxin Wang, Yanming Shen, and Di He. First place solution of kdd cup 2021 & ogb large-scale challenge graph prediction track. *arXiv preprint arXiv:2106.08279*, 2021.

[69] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*, 2019.

[70] Chulhee Yun, Yin-Wen Chang, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. O (n) connections are expressive enough: Universal approximability of sparse transformers. *Advances in Neural Information Processing Systems*, 33:13783–13794, 2020.

[71] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] see Section 7
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proof of Theorem 3

In this section, we provide the proof of the Theorem 3. First, we define a modified version of Transformer blocks, which will be used in the subsequent proof. Then we present a few technical lemmas. Finally, this section is completed with the proof of Theorem 3.

A.1 Modified Transformer blocks

Following [69, 70], we use a modified version of Transformer blocks in our construction first, and show that such modified Transformer blocks can be approximated by the originally defined $\mathrm{T\_blocks_U}$ up to arbitrary precision.

Definition 5 (Modified Transformer blocks)**.** *A modified Transformer block is defined as a composition of modified self-attention layer* Attn_m *and modified feed-forward layer* FFN_m*, where*

$$\mathrm{Attn}_m(\boldsymbol{X}) = \boldsymbol{X} + \sum_{h=1}^{H} \boldsymbol{A}^h(\boldsymbol{X})\boldsymbol{X}\boldsymbol{W}_V^h\boldsymbol{W}_O^h; \tag{11}$$

$$\mathrm{FFN}_m(\boldsymbol{X}) = \boldsymbol{X} + \phi(\boldsymbol{X}\boldsymbol{W}_1)\boldsymbol{W}_2. \tag{12}$$

In Eq.(11), the attention matrix $\boldsymbol{A}_U^h(\boldsymbol{X}) = \mathrm{hardmax}\left(\boldsymbol{X}\boldsymbol{u}(\boldsymbol{X}\boldsymbol{u} - c\mathbf{1})^\top\right)$ *, where* $\boldsymbol{u} \in \mathbb{R}^{d\times 1}$ *and* $c \in \mathbb{R}$ *are learnable parameters. Thus it expresses the* ***hard attention operation****.*

In Eq.(12), the activation ϕ *is a (possibly discontinuous) piece-wise linear function with at most three pieces.*

We denote the function class of modified Transformer blocks by $\mathrm{T\_blocks_m}(H, d_H, r)$*.*

A nice property of the modified Transformer blocks is that compositions of modified Transformer blocks can be approximated by compositions of the originally defined $\mathrm{T\_blocks_U}$ up to arbitrary precision, which is proved in Lemma 9 of [69]. Another important property of the modified Transformer blocks is that they can implement the following "selective shift" operator:

Lemma 6. *For any real numbers* a, b_Q *and* b_Q' *satisfying* $b_Q < b_Q'$ *and a vector* $z \in \mathbb{R}^d$*, there exists* $g \in \mathrm{T\_blocks_m}(2, 1, 1)$ *such that* $g(\boldsymbol{X}) = \boldsymbol{X} + a\Psi(\boldsymbol{X}; b_Q, b_Q')$*, where*

$$\Psi(\boldsymbol{X}; b_Q, b_Q')_{i,1} = \begin{cases} \max_k \boldsymbol{X}_k\boldsymbol{z} - \min_k \boldsymbol{X}_k\boldsymbol{z} & b_Q < \boldsymbol{X}_i\boldsymbol{z} < b_Q', \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

$$\Psi(\boldsymbol{X}; b_Q, b_Q')_{i,j} = 0 \quad (j \geq 2), \tag{14}$$

and $\boldsymbol{X}_k$ *denotes the* k*-th row of* $\boldsymbol{X}$*.*

The proofs of Lemma 6 can be found in [69].

A.2 Input quantization

In this subsection, we show how to quantize any input $\boldsymbol{X} \in [0,1]^{n\times d}$ to a δ-grid point in $\{0, \delta, \ldots, n-\delta\}^{n\times d}$. In fact, the quantization procedure can be conducted with $\frac{d}{\delta}$ modified Transformer blocks.

Lemma 7. *Consider an entry-wise quantization mapping defined as* $g_1^{\mathrm{entry}}(t) = \delta\lfloor\frac{t}{\delta}\rfloor$*. There exists a function* $g_1 : [0,1]^{d\times n} \to \mathbb{R}^{d\times n}$ *composed of* $\frac{d}{\delta}$ *modified Transformer blocks in* $\mathrm{T\_blocks_m}(2, 1, 1)$*, which implements the entry-wise quantization mapping* g_1^{entry} *over each entry of its input.*

Proof. For each column in the input $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, we show how to use $\frac{1}{\delta}$ stacked modified feed-forward layers to quantize it while keeping all the other columns untouched. For the r-th row, we use $\frac{1}{\delta}$ layers of the following form, in which the k-th layer is:

$$\mathrm{FFN}_m(\boldsymbol{X}) = \boldsymbol{X} + \phi(\boldsymbol{X}\boldsymbol{e}^{(r)} - k\delta\mathbf{1}_n)\boldsymbol{e}^{(r)\top}, \tag{15}$$

where $\boldsymbol{e}^{(r)} = (0, \cdots, 0, 1, 0, \cdots)^\top$ is the r-th d-dimensional one-hot vector and $\phi(t) = -t\,\mathbb{1}_{0\leq t<\delta}$ is a piece-wise linear activation function.

Therefore, we can construct $\frac{d}{\delta}$ modified Transformer blocks, where the modified self-attention layers express the identity mapping, and the $\left(\frac{r-1}{\delta}+k\right)$-th modified feed-forward layer is defined as Eq.(15), to implements the entry-wise quantization mapping g_1^{entry}. □

A.3 Injecting positional information with the position-aware condition

In this subsection, we introduce new techniques to leverage the position-aware condition defined in Theorem 3 and inject positional information into the Transformer block. Note that the new techniques we use are non-trivial since we do not rely on any explicit absolute positional encoding in the definition of the position-aware condition.

Lemma 8. *Assume $H, d_H, r \in \mathbb{N}^*$ and that $\boldsymbol{A}_U^h$ satisfies the **position-aware condition** defined in Theorem 3. Then there exists $\boldsymbol{u} = (u_1 \ \cdots \ u_n)^\top \in \mathbb{R}^n$ and a Transformer block $g_2 \in \mathrm{T\_blocks_U}(2,1,4)$ such that*

- $|u_i - u_j| > 1$ *for any $i \neq j$;*
- $g_1(\boldsymbol{X}) = \boldsymbol{X} + \boldsymbol{U}$, *where $\boldsymbol{U} = \boldsymbol{u}\boldsymbol{1}_d^\top$ and $\boldsymbol{1}_d$ is a d-dimensional all-one vector.*

Proof. By leveraging the position-aware condition, we can construct a self-attention layer such that the attention matrix in each head satisfies $\boldsymbol{A}_U^h(\boldsymbol{X})\boldsymbol{1} = \boldsymbol{v}$, where $\boldsymbol{v} \in \mathbb{R}^n$ is a row vector with n distinct entries.

We rewrite the generalized self-attention layer (Eq.(6)) with explict bias term as

$$\mathrm{Attn_U}(\boldsymbol{X}) = \boldsymbol{X} + \sum_{h=1}^{H} \left(\boldsymbol{A}_{\mathrm{U}}^h(\boldsymbol{X})(\boldsymbol{X}\boldsymbol{W}_V^h + c_V^h\boldsymbol{1})\right)\boldsymbol{W}_O^h + \boldsymbol{1}\boldsymbol{c}_O^{h\top}, \tag{16}$$

where $\boldsymbol{W}_V^h \in \mathbb{R}^{d\times 1}$, $\boldsymbol{W}_O^h \in \mathbb{R}^{1\times d}$, $c_V^h \in \mathbb{R}$ and $\boldsymbol{c}_O^{h\top} \in \mathbb{R}^d$ (note that $d_H = 1$ and $H = 2$ in this lemma). To obtain the desired mapping g_2, we set

$$\boldsymbol{W}_V^h = 0,\ \boldsymbol{W}_O^h = \boldsymbol{1}_d^\top,\ c_V^h = 1/\min_{i\neq j}|v_i - v_j|,\ \boldsymbol{c}_O^{h\top} = \boldsymbol{0} \quad (h = 1, 2). \tag{17}$$

Assume that $\boldsymbol{u} = (u_1 \ \cdots \ u_n)^\top = Hc_V^1\boldsymbol{v}$. Then $\boldsymbol{u}$ is an n-dimensional vector in which $|u_i - u_j| > 1$ for any $i \neq j$. Besides, we have

$$\sum_{h=1}^{H} \left(\boldsymbol{A}_{\mathrm{U}}^h(\boldsymbol{X})(\boldsymbol{X}\boldsymbol{W}_V^h + c_V^h\boldsymbol{1})\right)\boldsymbol{W}_O^h + \boldsymbol{1}\boldsymbol{c}_O^{h\top} = \begin{pmatrix} u_1 & u_1 & \cdots & u_1 \\ u_2 & u_2 & \cdots & u_2 \\ \vdots & \vdots & & \vdots \\ u_n & u_n & \cdots & u_n \end{pmatrix} = \boldsymbol{U}. \tag{18}$$

Finally, we set all the learnable parameters in the subsequent feed-forward layer to be 0. In this case, the feed-forward layer is equivalent to an identity mapping due to the skip connection. Therefore, $g_2(\boldsymbol{X}) = \mathrm{FFN}(\mathrm{Attn_U}(\boldsymbol{X})) = \boldsymbol{X} + \boldsymbol{U}$. □

With the position-aware condition satisfied, the Lemma 8 indeed shows that the positional information can be injected into the (quantized) input $\boldsymbol{X}$ and further be fed into the subsequent Transformer blocks, since $\boldsymbol{U}$ is composed of n distinct rows which help to distinguish each position.

A.4 Contextual mapping with the attentive condition

In this subsection, we follow [69, 70] to show that the modified attention layer with the (hard) attentive condition can implement "contextual mapping" defined in [69] and achieves universal approximation.

The concept of "contextual mapping" is defined as below.

Definition 9 (Contextual mapping [69])**.** *Consider a finite set $\mathbb{L} \subset \mathbb{R}^{n\times d}$. A map $q : \mathbb{L} \to \mathbb{R}^n$ defines a **contextual mapping** over $\mathbb{L}$ if and only if the map satisfies the following properties:*

- *For any $\boldsymbol{L} \in \mathbb{L}$, the entries of $q(\boldsymbol{L})$ are all distinct.*

- *For any $\boldsymbol{L}, \boldsymbol{L}' \in \mathbb{L}$ satisfying $\boldsymbol{L} \neq \boldsymbol{L}'$, all entries of $q(\boldsymbol{L})$ and $q(\boldsymbol{L}')$ are distinct.*

With this concept, we present the following lemma:

Lemma 10. *Assume n and δ^{-1} are two positive integers and that $n, \delta^{-1} \geq 2$. Suppose that the matrix $\boldsymbol{U} = \boldsymbol{u}\boldsymbol{1}_d^\top$, where $\boldsymbol{u}$ is an n-dimensional vector satisfying $|u_i - u_j| > 1$ for any $i \neq j$. Then, there exist a function $g_3 : \{0, \delta, 2\delta, \cdots, 1\}^{n\times d} \to \mathbb{R}^{n\times d}$ composed of stacked modified attention blocks, and a vector $\boldsymbol{z} \in \mathbb{R}^d$, such that the mapping $q(\boldsymbol{X}) = g_3(\boldsymbol{X} + \boldsymbol{U})\boldsymbol{z}$ is a contextual mapping over $\{0, \delta, 2\delta, \cdots, 1\}^{n\times d}$.*

Proof. This proof mainly follows and extends the technique in [69].

We sort the entries in $\boldsymbol{u}$ and define $u_{(1)} < \cdots < u_{(n)}$ to be a permutation of $(u_1, \cdots, u_n)$. Without the loss of generality, we assume that $u_{(n)} = u_n$.

Define $\boldsymbol{z} = (1, \delta^{-1}, \cdots, \delta^{-d})^\top$ and $s_r = u_{(r)} \sum_{k=0}^{d-1} \delta^{-k}$.

By leveraging Lemma 6, we construct $n\delta^{-d}$ modified Transformer blocks, where the $(r\delta^{-d} + k)$-th block express the mapping $\boldsymbol{X} \mapsto \boldsymbol{X} + \Psi\left(\boldsymbol{X}; s_r + k\delta - \frac{\delta}{2}, s_r + k\delta + \frac{\delta}{2}\right)$ for $0 \leq r < n$ and $0 \leq k < \delta^{-d}$. Given an input $\boldsymbol{X}$, we denote the output of these stacked layers by $\tilde{\boldsymbol{X}}$. With a similar argument to that in Lemma 6 of [69], one can show that the mapping from $(\boldsymbol{X}_1\boldsymbol{z} \quad \boldsymbol{X}_2\boldsymbol{z} \quad \cdots \quad \boldsymbol{X}_n\boldsymbol{z})$ to $\tilde{\boldsymbol{X}}_n\boldsymbol{z}$ is one-to-one.

Finally, we add an extra modified Transformer block, in which the modified feed-forward layer expresses an identity mapping, and the modified self-attention layer is

$$\mathrm{Attn}_m(\boldsymbol{X}) = \boldsymbol{X} + \left(\boldsymbol{X}\boldsymbol{u}(\boldsymbol{X}\boldsymbol{u})^\top\right)\boldsymbol{X}\boldsymbol{u}(n\delta^{-(n+1)d-1}\boldsymbol{e}^{(1)\top}).$$

Note that we only use one attention head here and all the parameters in the other attention head are set to 0. This block shifts all the layers by $n\delta^{-(n+1)d-1}\tilde{\boldsymbol{X}}_n\boldsymbol{z}$, and ensures that any input $\boldsymbol{X}$ would be mapped to a unique number $q(\boldsymbol{X})$, thus implementing a contextual mapping. □

A.5 Function value mapping via FFN

Once obtaining a contextual mapping, we can use stacked feed-forward layers to implement the function value mapping and obtain the desired sequence-to-sequence function.

Lemma 11 (Lemma 8 in [70])**.** *Given $\delta > 0$ a mapping $h : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ and assume there exists a vector $\boldsymbol{u}$ such that the mapping $q(\boldsymbol{X}) = h(\boldsymbol{X})\boldsymbol{z}$ is a contextual mapping over $\{0, \delta, 2\delta, \cdots, 1\}^{n\times d}$. Then for any $f : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$, there exists a function $g_4 : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ which is compositions of modified Transformer blocks in $\mathrm{T\_blocks_m}(2, 1, 1)$, such that*

$$g_4(h(\boldsymbol{X})) = f(\boldsymbol{X}) \quad (\forall \boldsymbol{X} \in \{0, \delta, 2\delta, \cdots, 1\}^{n\times d}).$$

A.6 Finishing the Proof of Theorem 3

With the preparations in the previous subsections, we present the proof of Theorem 3.

Proof. Without the loss of generality, assume that $\mathcal{D} \subseteq [0, 1]^{n\times d}$. Applying the Tietze extension theorem [18], one obtain an extended mapping $f : [0, 1]^{n\times d} \to \mathbb{R}^{n\times d}$. Therefore, it suffices to prove the theorem for any continuous sequence-to-sequence $f : [0, 1]^{n\times d} \to \mathbb{R}^{n\times d}$.

Suppose δ is a positive real number such that $\delta^{-1} \in \mathbb{Z}$. Applying Lemma 7, Lemma 8, Lemma 10 and Lemma 11, we obtain $\tilde{g}_1$ (the quantization mapping), g_2 (the positional mapping), $\tilde{g}_3$ (the contextual mapping) and $\tilde{g}_4$ (the function value mapping), such that

$$\tilde{g}_4 \circ \tilde{g}_3 \circ g_2 \circ \tilde{g}_1(\boldsymbol{X}) = f(\boldsymbol{X}) \quad (\forall \boldsymbol{X} \in \{0, \delta, 2\delta, \cdots, 1\}^{n\times d}). \tag{19}$$

Note that g_2 is an originally defined generalized Transformer blocks in $\mathrm{T\_blocks_U}(2, 1, 4)$, while $\tilde{g}_1$, $\tilde{g}_3$ and $\tilde{g}_4$ are all compositions of modified Transformer blocks in $\mathrm{T\_blocks_m}(2, 1, 1)$. Applying

Lemma 9 in [69], there exist compositions of originally defined generalized Transformer blocks g_1, g_3 and g_4, such that

$$\left(\int_{\mathcal{D}} \|g(\boldsymbol{X}) - \tilde{g}(\boldsymbol{X})\|_p^p \mathrm{d}\boldsymbol{X}\right)^{\frac{1}{p}} < \frac{\varepsilon}{2}, \tag{20}$$

where $g = g_4 \circ g_3 \circ g_2 \circ g_1 \in \Omega_{\mathrm{U}}^{2,1,4}$ and $\tilde{g} = \tilde{g}_4 \circ \tilde{g}_3 \circ g_2 \circ \tilde{g}_1$.

Define $\tilde{g}$ as a piece-wise linear approximation of f, such that $f(\boldsymbol{X}) = \bar{f}(\boldsymbol{X})$ for any $\boldsymbol{X} \in \{0, \delta, 2\delta, \cdots, 1\}^{n\times d}$. Then, by choosing δ to be sufficiently small, we have

$$\left(\int_{\mathcal{D}} \|f(\boldsymbol{X}) - \tilde{g}(\boldsymbol{X})\|_p^p \mathrm{d}\boldsymbol{X}\right)^{\frac{1}{p}} < \frac{\varepsilon}{2}. \tag{21}$$

The proof is completed by combining Eq.(20) and (21). □

B Proof of Proposition 4

Proof. Recall that the Universal RPE-based Attention is defined as

$$\boldsymbol{A}_{\mathrm{U}}(\boldsymbol{X}) = \mathrm{softmax}\left(\boldsymbol{X}\boldsymbol{W}_Q(\boldsymbol{X}\boldsymbol{W}_K + \mathbf{1}\boldsymbol{c}_K^\top)^\top + \boldsymbol{B}\right) \odot \boldsymbol{C}, \tag{22}$$

where $\mathbf{1}$ is an n-dimensional all-one vector and $\mathbf{1}\boldsymbol{c}_K^\top$ is the omitted bias term in Eq.(8). It suffices to show that $\boldsymbol{A}_{\mathrm{U}}$ defined in Eq.(22) satisfies the two conditions in Theorem 3.

Attentive condition. Note that the all-one matrix $\mathbf{1}\mathbf{1}^\top$ is a Toeplitz matrix by definition. Therefore, we can set $\boldsymbol{W}_Q = \boldsymbol{W}_K = \boldsymbol{u}$, $\boldsymbol{c}_K = c$ and $\boldsymbol{C} = \mathbf{1}\mathbf{1}^\top$, and obtain $\boldsymbol{A}_U^h(\boldsymbol{X}) = \mathrm{softmax}\left(\boldsymbol{X}\boldsymbol{u}(\boldsymbol{X}\boldsymbol{u} - c\mathbf{1})^\top\right)$.

Position-aware condition. First we set $\boldsymbol{C}$ as a upper triangular Toeplitz matrix:

$$\boldsymbol{C} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & 1 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{23}$$

Then we set $\boldsymbol{W}_Q = \boldsymbol{W}_K = \boldsymbol{c}_K = \mathbf{0}$. It's also easy to see that for any parametrization of $\boldsymbol{B}$ descibed in Section 2, we can force $\boldsymbol{B} = \mathbf{0}$ by properly setting all the learnable parameters to 0. In this case, for any $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ we have

$$\mathrm{softmax}\left(\boldsymbol{X}\boldsymbol{W}_Q^h(\boldsymbol{X}\boldsymbol{W}_K^h + \mathbf{1}\boldsymbol{c}_K^\top)^\top + \boldsymbol{B}\right) = \frac{1}{n}\mathbf{1}\mathbf{1}^\top; \tag{24}$$

$$\Rightarrow \quad \boldsymbol{A}_U^h(\boldsymbol{X})\mathbf{1} = \frac{1}{n}\boldsymbol{C}\mathbf{1} = \begin{pmatrix} 1 & \frac{n-1}{n} & \cdots & \frac{1}{n} \end{pmatrix}^\top. \tag{25}$$

To sum up, $\boldsymbol{A}_{\mathrm{U}}$ defined in Eq.(22) satisfies the two conditions in Theorem 3, which completes the proof. □

C Experiments

C.1 Synthetic Tasks

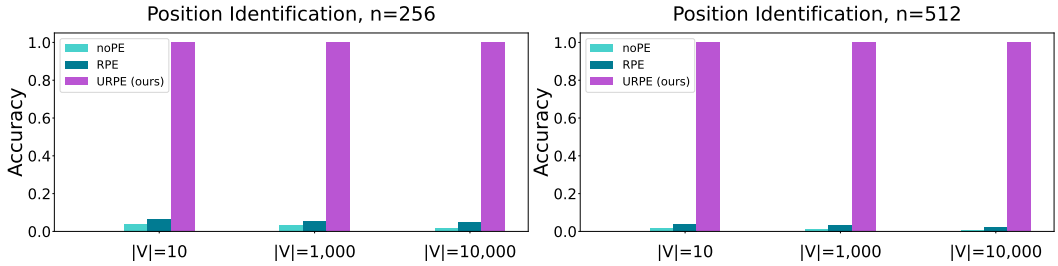

Figure 3: Results on the Position Identification task with input sequences with different lengths: (1) Sequence length $n = 256$ (Left Panel); (2) Sequence length $n = 512$ (Right Panel). With input sequences with different lengths, our URPE-based Transformer model consistently solves the task while other methods fail.

Baselines. We choose the vanilla Transformer as the base model and compare the following ways to encode positional information: (1) no positional encodings (noPE); (2) T5-style relative positional encoding (RPE) [54]; (3) URPE with T5-style RPE backbone Transformer.

Settings. For both the Position Identification (PI) and Even Token Prediction (ETP) tasks mentioned in Section 5.1, we use synthetic datasets with randomly generated sequences. In detail, we vary the token vocabulary size from [10, 1000, 10000] and set the sequence length to 128. The number of layers and the number of attention heads are set to 3 and 12, respectively. The hidden dimension is set to 768. We use Adam [34] as the optimizer, and set its hyperparameters ϵ to $1e-8$ and (β_1, β_2) to $(0.9, 0.999)$. The peak learning rate is set to $7e-5$ with a 6K-step warm-up stage. After the warm-up stage, the learning rate decays linearly to zero. The model is trained for 40K steps in total with the batch size as 512. We set the dropout probability, gradient clip norm and weight decay to 0.0. All models are trained on 4 NVIDIA Tesla V100 GPUs.

Ablation Study on the length of input sequences. To comprehensively investigate the approximation capability of RPE-based Transformer and our URPE-based Transformer, we vary the sequence length in [256, 512] on the Position Identification task. The results are presented in Figure 3. Together with the results in Figure 1, we can see that our URPE-based Transformer consistently achieves 100% accuracy, while the Transformer without PE and the Transformer with T5-style RPE perform worse when the task becomes more difficult (i.e., the input sequences become longer). The above results serve as further evidences that our URPE-based Transformer is consistent with our theoretical findings in Section 4.

C.2 Language Modeling

Baselines. We choose the Transformer-XL as the base model and compare the following ways to encode positional information: (1) Transformer-XL style relative positional encoding [10]; (2) URPE with Transformer-XL style RPE backbone model. Besides, we also include results of several competitive sequence models: (1) LSTM [21]; (2) Temporal Convolutional Network (TCN) [2]; (3) Gated Convolutional Neural Network (GCNN) [11]; (4) LSTM with Neural Cache [21]; (5) Quasi-Recurrent Neural Network [46]; (6) Hebbian Learning with Neural Cache [53].

Settings. We conduct experiments on the WikiText-103 dataset [47], which contains 103M training tokens from 28K articles, with an average length of 3.6K tokens per article, which allows testing the ability of long-term dependency modeling. We build our model based on the Transformer-XL Base model which consists of 16 decoder layers. The number of attention head is set to 10. The hidden dimension is set to 41. The dimension of feed-forward layer is set to 2100. The dropout ratio and the weight decay are set to 0.1 and 0.01, respectively. We use Adam [34] as the optimizer, and set its hyperparameters ϵ to $1\mathrm{e}-8$ and (β_1, β_2) to (0.9, 0.999). The peak learning rate is set to $2.5\mathrm{e}-4$. The total training steps is set to 200K. All models are trained on 4 NVIDIA Tesla V100 GPUs.

C.3 Graph Learning

We conduct experiments on three benchmark datasets covering graph and node representation learning tasks from small to large scale datasets: ZINC from Benchmarking-GNNs [16], PCQM4M and MAG24M from Open Graph Benchmark Large-Scale Challenge (OGB-LSC) [29].

C.3.1 ZINC

ZINC is a real-world dataset which consists of 250K molecular graphs. The task is to predict the constrained solubility of a molecule which is an important chemical property for drug discovery. We train our models on both the ZINC-Full and ZINC-Subset (12K selected graphs following [16]).

Baselines. To demonstrate the power of our method and for fair comparison, we set the parameter budget of the model to be less than 500K following [16, 67]. We include results of several competitive GNNs: (1) Graph Isomorphism Network (GIN) [66]; (2) GraphSAGE [23]; (3) Graph Attention Network (GAT) [62]; (4) Graph Convolutional Network [35]; (5) Mixture Model Network (MoNet) [48]; (6) Gated Graph Convolutional Network [5] with Positional Encodings [16] (GatedGCN-PE); (7) Message Passing Neural Network (MPNN(sum)) [20]; (8) Hierarchical Inter Message Passing (HIMP) [17]; (9) Principal Neighbourhood Aggregation (PNA) [8]. Two representative Transformer-based models GraphTransformer (GT) [15] and Spectral Attention Network (SAN) [37] are also compared.

Settings. We build on the Graphormer [67] model which consists of 12 layers. The dimension of hidden layers and feed-forward layers are set to 80. The number of attention heads are set to 32. The batch size is selected from [128, 256, 512]. We use Adam [34] as the optimizer, and set its hyperparameter ϵ to $1\mathrm{e}-8$ and (β_1, β_2) to (0.9, 0.999). The peak learning rate is selected from $[4\mathrm{e}-4, 5\mathrm{e}-4]$. The model is trained for 600k and 800k steps with a 60K-step warm-up stage for ZINC-Subset and ZINC-Full respectively. After the warm-up stage, the learning rate decays linearly to zero. The dropout ratio is selected from [0.0, 0.1]. The weight decay is selected from [0.0, 0.01]. All models are trained on 4 NVIDIA Tesla V100 GPUs.

C.3.2 PCQM4M

PCQM4M is a quantum chemistry regression task in OGB-LSC [29]. The PCQM4M dataset contains more than 3.8 million molecular graphs in total, which is currently the largest graph-level prediction dataset. The state-of-the-art architecture for this task is the Graphormer [67] model introduced above.

Baselines. Following [67], we include results of several competitive baselines: (1) Graph Convolutional Network [35]; (2) Graph Isomorphism Network (GIN) [66]; (3) Graph Convolutional Network [35] with Virtual Node [20]; (4) Graph Isomorphism Network [66] with Virtual Node [20]; (5) Graph Isomorphism Network with Edge feature and Virtual Node (GINE-VN) [6, 20]; (6) Deeper Graph Convolutional Network (DeeperGCN) [38]; (7) GraphTransformer [15].

Settings. To demonstrate the power of our method and for fair comparison, We set the parameter budget of the model to be less than 12.5M. We still follow [67] to set the hyper-parameters in the Graphromer model and equip it with URPE. In detail, our Graphormer with URPE-based Attention consists of 6 layers and 32 attention heads. The dimension of hidden layers and feed-forward layers are set to 512. The batch size is 512. We use Adam [34] as the optimizer, and set its hyperparameters ϵ to $1\mathrm{e}-8$ and (β_1, β_2) to (0.9, 0.999). The peak learning rate is set to $3\mathrm{e}-4$. The model is trained for 1M steps with a 60k-step warm-up stage. After the warm-up stage, the learning rate decays linearly to zero. The dropout ratio for the input embedding, attention matrix and the hidden representation are set to 0.0, 0.1, 0.0 respectively. The weight decay is set to 0.0. All models are trained on 8 NVIDIA Tesla V100 GPUs.

C.3.3 MAG240M

The MAG240M dataset contains a large-scale heterogeneous academic graph with more than 240 million nodes and 1.7 billion edges. This giant graph is extracted from the Microsoft Academic Graph (MAG) [64]. Given arXiv papers situated in the heterogeneous graph, the task requires the model to automatically annotate the topics of those papers, i.e., predicting the primary subject area of each arXiv paper. Thus, it can be formulated as a node representation learning task.

Table 6: Results on MAG240M from OGB-LSC. We use * to indicate the best performance. The results of the baselines are reported in [29].

Model	#Params	Valid Acc
MLP [29]	0.5M	0.5267
LabelProp [29]	0	0.5844
SGC [65]	0.7M	0.6582
SIGN [56]	3.8M	0.6664
MLP+C&S [31]	0.5M	0.6698
GraphSAGE [23]	4.9M	0.6679
GAT [62]	4.9M	0.6715
R-GraphSAGE [23, 57]	12.2M	0.6986
R-GAT [62, 57]	12.3M	0.7002
Graphormer [67]	11.0M	0.7013
Graphormer + URPE-based Attention (ours)	11.0M	0.7074*

Baselines. We include results of several competitive baselines: (1) graph-agnostic MLP (MLP) [29]; (2) Label Propagation [29]; (3) Simplified Graph Convolution [65]; (4) Scalable Inception Graph Neural Network (SIGN) [56]; (5) MLP with Correct and Smooth Procedure (MLP+C&S) [31]; (6) GraphSAGE [23]; (7) Graph Attention Network [62]. Due to the heterogeneous property of the academic graph, we also choose two baselines from [29] which learn distinct weights for each individual relational type: R-GraphSAGE [23, 57] and R-GAT [62, 57].

Settings. To demonstrate the power of our method and for fair comparison, We set the parameter budget of the model to be less than 12.5M. We build on the Graphormer [67] model which consists of 6 layers. The dimension of hidden layers and feed-forward layers are set to 512. The number of attention heads are set to 32. The batch size is 1024. We use Adam [34] as the optimizer, and set its hyperparameter ϵ to $1\mathrm{e}-8$ and (β_1, β_2) to (0.9, 0.999). The peak learning rate is set to $3\mathrm{e}-4$. The model is trained for 100k steps with a 6k-step warm-up stage. After the warm-up stage, the learning rate decays linearly to zero. The dropout ratio and the weight decay is set to 0.5 and 0.01 respectively. We also employ the stochastic depth [30] and set the probability to 0.1. We follow the sub-graph sampling strategy from [29]. All models are trained on 32 NVIDIA Tesla V100 GPUs.

Results Table 6 summarizes performance comparisons on MAG240M dataset. It can also be easily seen that the URPE-based Attention consistently improve the performance of the Graphormer model on the MAG240M node classification task. The validation accuracy of the Graphormer model is improved by 0.06, which is currently the state-of-the-art single model performance on the leaderboard of MAG240M dataset.

C.4 URPE-based Transformers v.s. Transformers with both APE and RPE

We conduct experiments on the Language Pre-training task to evaluate different positional encoding strategies. It is worth noting that in some competitive pre-training methods like UniLMv2 [3], APE and RPE have already been used together. Thus, we choose this task as the benchmark. We mainly test three model variants: Transformers with both APE and RPE, Transformers with RPE only, and our URPE-based Transformers. For all the models, RPE is set to the T5 [54] version, following UniLMv2. We roughly keep the number of parameters of different models to the same and train the models in the BERT-base setting using the same hyper-parameters. We choose the validation loss (masked language modeling loss after 1M iterations on a hold-out validation set) in the pre-training stage as the metric. We observed that the validation losses of the Transformers with both APE and RPE, Transformers with RPE only, our URPE-based Transformers are 1.86, 1.94, and 1.87, respectively. The results show that our URPE Transformer is competitive with Transformers with both APE and RPE and is much better than Transformers with RPE only. Together with the above observations on the synthetic dataset in Section 5.1, we can see that URPE is competitive or even superior to previous APE and RPE or their combinations.