Two Stones Hit One Bird: Bilevel Positional Encoding for Better Length Extrapolation

Zhenyu He [*1] **Guhao Feng** [*2] **Shengjie Luo** [*1] **Kai Yang** [2]
Liwei Wang [1 3] **Jingjing Xu** [4] **Zhi Zhang** [4] **Hongxia Yang** [4] **Di He** [1]

Code: https://github.com/zhenyuhe00/BiPE

Abstract

In this work, we leverage the intrinsic segmentation of language sequences and design a new positional encoding method called Bilevel Positional Encoding (BiPE). For each position, our BiPE blends an intra-segment encoding and an inter-segment encoding. The intra-segment encoding identifies the locations within a segment and helps the model capture the semantic information therein via absolute positional encoding. The inter-segment encoding specifies the segment index, models the relationships between segments, and aims to improve extrapolation capabilities via relative positional encoding. Theoretical analysis shows this disentanglement of positional information makes learning more effective. The empirical results also show that our BiPE has superior length extrapolation capabilities across a wide range of tasks in diverse text modalities.

1. Introduction

In many scenarios, text can be effectively decomposed into modular segments, each expressing a self-contained unit of thought (Halliday & Matthiessen, 2013). In natural languages, documents are typically composed of sentences. Each sentence describes a distinct idea or argument. In programming languages, code is organized into lines or function classes that define coherent operation or functionality. In mathematics, proofs unfold through a series of deductive steps, each representing a logical progression from its predecessors to the final answer.

The lengths of different text sequences may vary significantly (Press et al., 2022). What is intriguing is that empirically, we observed that for sequences with different lengths, the distribution of the token number in each modular segment usually has bounded support and tends to be approximately similar. In Figure 1, we utilized the widely used PG-19 text corpus (Rae et al., 2020a) for visualization (Please refer to Appendix C for more results). It is evident that the token number distribution in each segment (i.e., sentence) remains remarkably consistent, regardless of the total sequence length. In contrast, the number of sentences linearly increases as the sequence length grows.

Given the above observations, we argue that a popular research direction in language modeling, known as the *length extrapolation* problem (Press et al., 2022; Anil et al., 2022; Chi et al., 2022; 2023; Chowdhury & Caragea, 2023; Chen et al., 2023b), should be better positioned as a *number-of-segment extrapolation* problem. In the literature, previous studies in this direction have either developed better positional encodings that can handle longer sequence (Raffel et al., 2020; Su et al., 2021; Ruoss et al., 2023a; Kazemnejad et al., 2023; Chen et al., 2023b; Li et al., 2023; Peng et al., 2023; Zhu et al., 2023; Chen et al., 2023a; Liu et al., 2023) or proposed specific inductive biases associated with the attention patterns in language models (Ratner et al., 2023; Han et al., 2023; Xiao et al., 2023), or both (Press et al., 2022; Chi et al., 2022; 2023; Sun et al., 2023). However, none of these methods adequately consider or utilize the intrinsic segmentation of language data, nor do they specifically address the extrapolation issue in terms of the number of segments.

In this paper, we introduce **BiPE** (**Bi**level **P**ositional **E**ncoding), a simple yet effective positional encoding scheme for improving length extrapolation. Different from all existing length extrapolation approaches, BiPE employs two distinct encodings for each position: an intra-segment encoding and an inter-segment encoding. The intra-segment encoding identifies the location of the token within its segment. As a complement, the inter-segment encoding specifies the segment to which it belongs. Using natural language as an illustration, different words within the same sentence share the same inter-segment positional encoding but possess different intra-segment encodings. Conversely, words in different sentences but occupying the same inter-segment

[*]Equal contribution [1]National Key Laboratory of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University [2]School of EECS, Peking University [3]Center for Machine Learning Research, Peking University [4]ByteDance Inc. Correspondence to: Di He <dihe@pku.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

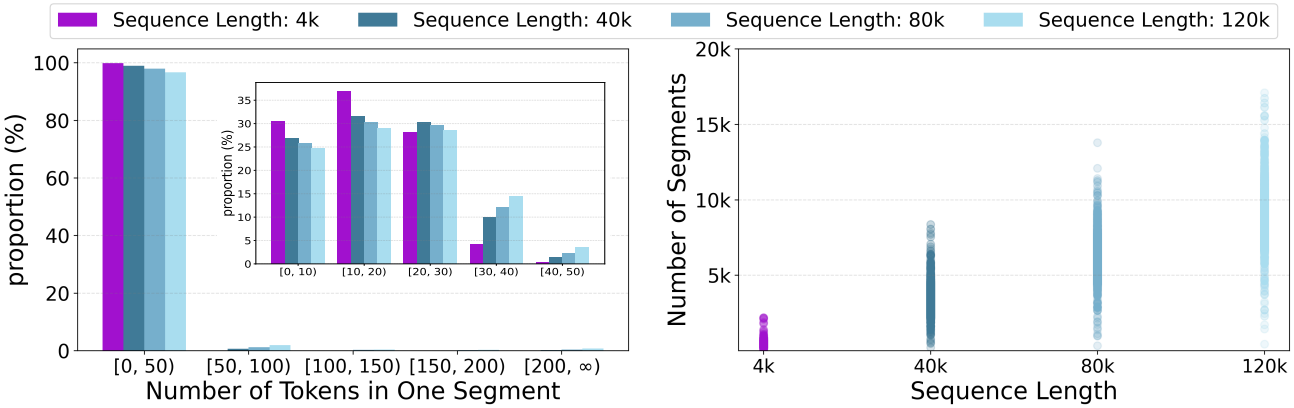

Figure 1. **Left:** The distribution of the token number in one segment with different sequence lengths. **Right:** The distribution of the number of segments with different sequence lengths. We use the tokenizer of Llama 2 (Touvron et al., 2023) for tokenization on PG-19 (Rae et al., 2020a). Full stop"`.`" and newline "`\n`" are used for segmentation. It can be seen that even when the sequence length is around 120k, the token number in most sentences is less than 50, while the number of sentences grows up to 10k.

position (e.g., the first token in different sentences) share the same intra-segment encoding while having distinct inter-segment encodings. See Figure 2 for an illustration.

BiPE disentangles the position modeling, offering greater flexibility in addressing the length extrapolation problem. At the intra-segment level, the intra-segment encoding specifies positions within the segment, helping the model capture the semantic information contained therein. Given that the number of tokens within a segment is usually bounded, we discovered that utilizing the original absolute positional encodings (APE, Vaswani et al., 2017) is already sufficient at this level. The inter-segment encoding targets to capture the relationships between segments and exhibits certain extrapolation capabilities. Therefore, we employ relative positional encodings (RPE, Su et al., 2021; Press et al., 2022). In this way, inductive biases in the two levels focus on different aspects of positional information and can be appropriately incorporated into model architectures, leading to a better learning process. We further give a theoretical justification of BiPE, which suggests that the proposed positional encoding scheme can make the Transformer model more parameter-efficient under some conditions.

Extensive experiments are conducted to demonstrate the empirical effectiveness of BiPE. First, we empirically verify the expressiveness of BiPE in mathematical reasoning tasks (Wei et al., 2022; Feng et al., 2023), which well aligns with our theoretical results. Second, for the length extrapolation problem, we evaluate BiPE and strong baselines across diverse tasks covering language modeling [PG-19 (Rae et al., 2020a), ArXiv and Github (Gao et al., 2020)] and long context benchmark [SCROLLS (Shaham et al., 2022)]. Finally, we conduct experiments on datasets of normal-length sentences. We also conduct ablation studies to verify the effectiveness of each module in BiPE. Our empirical results show the superior performance of BiPE on most problems.

2. Related Work

This work focuses on the *length extrapolation* problem in language modeling, i.e., can a language model that is trained on sequences with maximum length L_{train} still perform well when being tested on sequences with length $L_{\text{test}} > L_{\text{train}}$? (Press et al., 2022). Here we provide a literature review of existing approaches related to this problem.

2.1. Improved Positional Encodings for Length Extrapolation

The original Transformer model (Vaswani et al., 2017) encodes position information via Absolute Positional Encoding (APE), where each position is equipped with a (learnable or fixed sinusoidal) real-valued embedding. But neither the learnable nor the fixed sinusoidal embedding can generalize well to longer sequences.

Different from APE that assigns an embedding for each position i, Shaw et al. (2018) introduced Relative Positional Encoding (RPE) which encodes the relative distance $i-j$ for each position pair (i, j). Most methods incorporate RPE as an additive term in the attention module (Raffel et al., 2020; Press et al., 2022; Chi et al., 2022; 2023). These methods can mitigate the length extrapolation problem to some extent but still have several limitations. For example, Raffel et al. (2020) uses the same attention bias for all query-key pairs with a relative distance larger than K, which limits its ability to distinguish different positions in long sequences.

One of the most popularly used relative positional encoding in recent large language models is Rotary Position Encoding (RoPE) (Su et al., 2021; Chowdhery et al., 2022; Touvron et al., 2023). RoPE rotates the query and key vectors with an angle proportional to their absolute positions before the attention, which results in the attention being a function of the relative distance between tokens.

While encoder-only Transformers (e.g., BERT (Devlin et al., 2019b)) are permutation equivariant without positional encoding, Haviv et al. (2022) show that decoder-only Transformers with causal attention masks can learn positional information even without any explicit positional encoding. Recently, Kazemnejad et al. (2023) discovered that the no positional encoding (NoPE) model also can handle longer sequences to some extent on small-scale synthetic tasks, but there is no strongly positive evidence on large-scale settings.

2.2. Improved Algorithms for Length Extrapolation

To help positional encodings handle longer sequences, Ruoss et al. (2023b) recently proposed a way to randomly select a subset of positions from a much larger range than those observed during training. The positional information of longer sequences can thus be simulated. Zhu et al. (2023) proposed a similar idea called positional skip-wise fine-tuning (PoSE), which requires additional efforts for fine-tuning large-scale models.

Relative positional encoding, especially RoPE, can capture the relative positional information well, but its length extrapolation capability is not satisfactory yet. Due to this, one line of works introduces priors biased toward local window attention via additive RPEs (Press et al., 2022; Chi et al., 2022; 2023; Sun et al., 2023) or hard constraints (Ratner et al., 2023; Xiao et al., 2023; Han et al., 2023) to boost length extrapolation capabilities. Another line of works tailored to RoPE called positional embedding scaling (Chen et al., 2023b; Peng et al., 2023; Roziere et al., 2023; Chen et al., 2023a; Liu et al., 2023) adjusts the range of either the position index or the frequency basis in RoPE, achieving promising extrapolation performance. Recently, a concurrent work (Jin et al., 2024) proposed a bilevel attention mechanism for better length extrapolation of RoPE-based language models. It keeps the exact attention computation within a pre-defined neighbor range and uses the floor operation to group and map unseen large relative positions.

BiPE aims to develop a new positional encoding scheme, which is orthogonal to all the methods above. All these advancements can be seamlessly combined with BiPE for better length extrapolation. Our experiments (Section 4) on several representative algorithms provide strong evidence supporting the compatibility of BiPE.

3. Method

In this section, we introduce **BiPE** (**Bi**level **P**ositional **E**ncoding), a new positional encoding scheme for better length extrapolation capabilities. First, we formally describe the modular segments of text sequence in Section 3.1. Based on this segment representation, we thoroughly illustrate the derivation of our BiPE method in Section 3.2. In Section 3.3, we conduct a theoretical study on the expressiveness of our BiPE to further demonstrate its soundness.

3.1. Modular Segments of Text Sequence

Formally, we use $\mathbf{S} = [w_1, \ldots, w_l, \ldots, w_L]$ to denote the input text sequence, i.e., an ordered collection of text tokens where w_l denotes the l-th token and L denotes the total sequence length. As previously introduced, text sequences can be decomposed into a series of non-overlapping modular segments, i.e., $\mathbf{S} = S_1 \oplus S_2 \oplus \ldots S_n \oplus \cdots \oplus S_N$ where S_n is the n-th segment, N is the total number of segments, and $\oplus$ is the concatenation operation. Each segment S_n is defined by $S_n = [w_{a_n}, w_{a_n+1}, \ldots, w_{b_n}]$. Here, a_n and b_n denote the starting and ending indices. The segmentation strategy can simply use symbol detection (e.g., newline and full stop).

3.2. Bilevel Positional Encoding

As stated in the introduction, in many practical scenarios, the number of tokens in each segment S_n follows a similar distribution regardless of the value L, and the sequence length L has a major impact on the segment number N. Consequently, we believe modeling the length extrapolation for N is a more effective approach than that for actual length L. To this end, we propose BiPE, a novel bilevel positional encoding that blends two distinct encoding schemes at each position for better length extrapolation: an intra-segment encoding and an inter-segment encoding. (See Figure 2).

Intra-Segment Encoding. In a text sequence, each modular segment describes an independent statement, and the intra-segment positional encoding serves as an anchor to identify the location of each token in the segment for capturing semantic information therein. Formally, within each segment $S_n = [w_{a_n}, w_{a_n+1}, \ldots, w_{b_n}]$, we encode the (local) position i for token w_{a_n+i}, where $1 \leq i \leq b_n - a_n + 1$. Note that the number of tokens within a segment is usually bounded, i.e., there are few sentences that are extremely long. We find using the original absolute positional encoding (Vaswani et al., 2017) is enough. For each token w_{a_n+i} in S_n, we assign a real-valued embedding $\boldsymbol{e}_i$ to it, which will be added to the input token embedding. $\boldsymbol{e}_i$ is shared among tokens at the same local position i in different S_n .

Inter-Segment Encoding. Though the intra-segment encoding can provide the location of tokens within each segment, the locations across segments are mixed, and the contextual relationships between segments are not captured. As a complement, we use the inter-segment positional encoding to specify the segment to which each token belongs. Keeping in mind that this encoding will play another role in handling longer sequences that are unseen during training, we employ relative positional encodings (Shaw et al., 2018; Raffel et al., 2020; Su et al., 2021; Press et al., 2022). Different from previous RPEs that are defined using the distance between token indexes, the inter-segment encoding is defined using

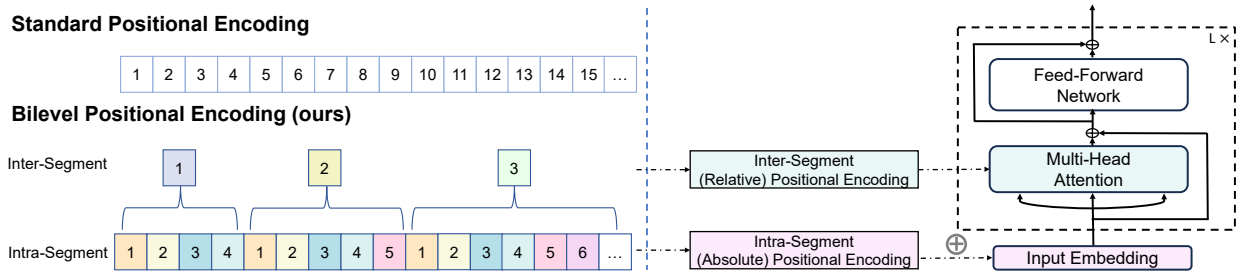

Figure 2. **Left:** The schematic comparison of Standard Positional Encoding (top) and our proposed Bilevel Positional Encoding (BiPE, bottom). BiPE differentiates positions using both intra-segment and inter-segment encodings. **Right:** Absolute positional encoding is used as Intra-Segment Encoding added to the input embedding and relative positional encoding (e.g., RoPE and ALiBi) is used as Inter-Segment Encoding in the Transformer attention module.

the distance between segment indexes.

Instantiation. BiPE uses absolute positional information for the intra-segment encoding and can leverage any RPE approaches for the inter-segment encoding. In this work, we instantiate two BiPE variants, BiPE-RoPE and BiPE-ALiBi.

BiPE-RoPE leverages RoPE (Su et al., 2021) as the inter-segment encoding. For a pair of tokens (w_{l_1}, w_{l_2}) which are in the n-th segment and the m-th segment respectively, two rotation matrices $\boldsymbol{R}_{\Theta,n}$ and $\boldsymbol{R}_{\Theta,m}$ are assigned, where Θ denotes the pre-defined parameters of the rotation matrix (Su et al., 2021). Given query-key pair $q_{l_1}, k_{l_2} \in \mathbb{R}^d$, the attention score is computed by $\frac{q_{l_1}\boldsymbol{R}_{\Theta,n}(k_{l_2}\boldsymbol{R}_{\Theta,m})^T}{\sqrt{d}} = \frac{q_{l_1}\boldsymbol{R}_{\Theta,n-m}k_{l_2}^T}{\sqrt{d}}$. BiPE-ALiBi uses ALiBi (Press et al., 2022) as the inter-segment encoding. Similarly, the relative segment distance $n-m$ is calculated for token pair (w_{l_1}, w_{l_2}). The attention score between the two tokens is computed by $\frac{q_{l_1}k_{l_2}^T}{\sqrt{d}} + r|n-m|$, where r is a pre-defined hyper-parameter.

Discussion. The original BERT (Devlin et al., 2019a) also includes two encodings for representing positions, but its approach differs significantly from BiPE. Primarily, BERT only needs to specify two segments using absolute encoding, tailored for the next sentence prediction task not for length extrapolation. Furthermore, BERT treats a sequence as a flat array of tokens and defines the segments in an arbitrary way, ignoring intrinsic segmentation of language data. See Appendix B.1 for more discussions. An empirical comparison can be found in Section 4.5.

3.3. Theoretical Analysis of BiPE

Many previous works are built upon an assumption that tokens are generated in a hierarchical manner in natural language and develop the hierarchical hidden Markov model (Fine et al., 1998), hierarchical recurrent model (Chung et al., 2016), and hierarchical topic model (Blei et al., 2003; Griffiths et al., 2003). We follow to use this assumption to investigate the parameter efficiency of BiPE. In particular, we leverage the (non-deterministic) finite automata (NFA), which is widely used in the field of theoretical computer science. Alur et al. (1999) proposed hierarchical finite automata as a practical way to represent such linguistic structures. Inspired by the framework, we introduce a simplified model, Bi-NFA, which restricts the hierarchy level of hierarchical finite automata to two. We compare the parameter efficiency of Transformers to represent NFA and Bi-NFA and show that BiPE has a theoretical advantage over existing positional encoding schemes.

NFA. A nondeterministic finite automaton (NFA) is a fundamental and essential computational model in computer science (Eilenberg, 1974). An NFA $\mathcal{N}$ can be defined as a tuple $\mathcal{N} = (Q, \Sigma, \delta, q_0, F)$, where Q is a set of states, Σ is the alphabet of input symbols, $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is a transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is a set of final states. $\mathcal{P}(Q)$ denotes the power set of Q. A string $\mathbf{S} = [w_1, w_2, \cdots, w_n] \in \Sigma^*$ is accepted by $\mathcal{N}$ if there exists a sequence of states $r_0, r_1, \cdots, r_n \in Q$ such that $r_0 = q_0$, $r_{i+1} \in \delta(r_i, w_{i+1})$ for $i = 0, 1, \ldots, n-1$, and $r_n \in F$.

Bi-NFA. We utilize the hierarchical automata to capture the structure of modular segments and introduce the Bi-NFA by restricting the hierarchy level to two. A Bi-NFA is a tuple $\mathcal{N} = (\mathcal{Q}, \Sigma, \delta, q_0, F)$, where $\mathcal{Q}$ is the collection of state sets $Q_1, Q_2, \cdots, Q_n$, Σ is the symbol set that includes a segment separator w^*, δ is the transition kernel, q_0 is the initial state, and F is the set of accept states. The main difference between Bi-NFA and NFA is that the transitions are constrained by the state sets and the segment separator. Specifically, for any state $q \in Q_i$ and any symbol w, we have $\delta(q, w) \subset Q_i$ if $w \neq w^*$ and $\delta(q, w^*) \subset \{q_1^*, \cdots, q_n^*\}$, where q_i^* is the start state in Q_i and $q_0 \in \{q_1^*, \cdots, q_k^*\}$. Thus, Bi-NFA stays within the same state set until it reads the segment separator, and then it can move to any other state set in $\mathcal{Q}$. The Bi-NFA can be viewed as a variant of NFA that processes the input sequence segment by segment.

Theorem 3.1 (Lower bound for Transformer with absolute positional encoding to represent NFA)**.** *For any size of state set, there exists an NFA $\mathcal{N} = (Q, \Sigma, \delta, q_0, F)$ such that a Transformer with APE needs at least $O(|Q|^2)$ embedding*

size to represent the NFA.

Theorem 3.2 (Upper bound for Transformer with BiPE to represent Bi-NFA)**.** *For any Bi-NFA $\mathcal{N} = (\mathcal{Q}, \Sigma, \delta, q_0, F)$, $\mathcal{Q} = \{Q_1, Q_2, \cdots, Q_k\}$ there exists a Transformer with BiPE and $O(k^2 + \sum_{i\in[k]} |Q_i|^2)$ embedding size can represent the Bi-NFA.*

The proof of Theorems 3.1 and 3.2 can be found in Appendix A and Theorem 3.1 can be extended to relative positional encodings. For a Bi-NFA $\mathcal{N} = (\mathcal{Q}, \Sigma, \delta, q_0, F)$, denote $T = \sum_{i\in[k]} |Q_i|$ as the number of states, and assume $|Q_i| = O(\sqrt{T})$. If naively treating $\mathcal{N}$ as an NFA with T states, we can directly obtain from Theorem 3.1 that an absolute positional encoding-based Transformer requires at least $O(T^2)$ dimensions to represent it. However, from Theorem 3.2, by well exploiting the hierarchical structure, a Transformer with BiPE only requires $O(N^{\frac{3}{2}})$ dimensions. This suggests the superiority of the BiPE over previous method from a theoretical perspective.

4. Experiments

In this section, we empirically study the effectiveness of our BiPE method. In particular, we aim to answer the following questions through experiments:

- **Q1**: Do the theoretical results regarding parameter efficiency of BiPE hold in practice? (Sec 4.1)
- **Q2**: Does BiPE bring superior length extrapolation capabilities in real-world tasks? (Sec 4.2)
- **Q3**: Does BiPE help Transformer-based language models better understand long text? (Sec 4.3)
- **Q4**: Does BiPE hurt performance on normal-length text? (Sec 4.4)
- **Q5**: Is each design choice in BiPE helpful? (Sec 4.5)

We will thoroughly answer each question with carefully designed experiments on widely used benchmarks as below. We also cover different modalities in the experiments, including math (arithmetical reasoning task in Section 4.1), natural language (PG19&ArXiv task in Section 4.2) and code (Github task Section 4.2). We run each experiment multiple times with different random seeds and report the averaged results. Due to space limits, we present more details and additional results in Appendix D.

4.1. Capacity Experiments

Tasks. To empirically verify the parameter efficiency brought by our BiPE method, we conduct experiments on the Arithmetic task (Feng et al., 2023), which is recently used as a proxy to examine the mathematical reasoning capability of language models. Given an arithmetical expression consisting of numbers, basic operations $(+, -, \times, \div, =)$ and brackets, e.g., $(1+2)\times(3+5) =$, this task requires language models to calculate and generate the correct result, e.g., 24. Following Feng et al. (2023), we train all models using Chain-of-Thought demonstrations (See Appendix D.1). The evaluation metric is the accuracy of the final answer.

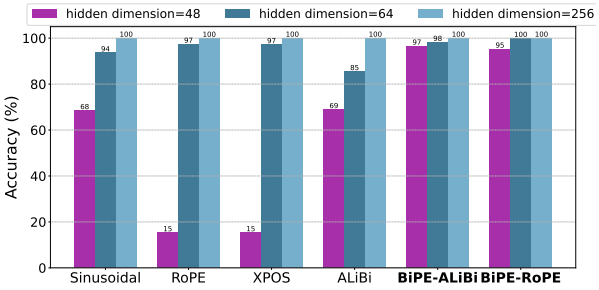

Figure 3. Accuracy of Transformer models with different positional encoding methods on the Arithmetic task. Our BiPE method consistently performs best on different scales of parameters.

Settings. The Arithmetic dataset from Feng et al. (2023) consists of 1 million training samples and 100k test samples in total. We choose the standard decoder-only Transformer language model as the base model and compare our BiPE method with the following competitive positional encodings: 1) Sinusoidal PE (Vaswani et al., 2017); 2) RoPE (Su et al., 2021); 3) XPOS (Sun et al., 2023); 4) ALiBi (Press et al., 2022). In particular, we implement two versions of our BiPE, BiPE-RoPE and BiPE-ALiBi, which instantiates the inter-segment encoding via RoPE and ALiBi respectively. The segment boundary is simply determined by the equal sign "=". Following Feng et al. (2023), we set the number of layers to 3 and the number of attention heads to 4. To evaluate the parameter efficiency, we vary the hidden dimension in $[48, 64, 256]$. Additional experimental details are presented in Appendix D.1.

Results. In Figure 3, it can be easily seen that given a similar amount of parameters, BiPE-based language models consistently outperform other baselines on this task. For example, when the hidden dimension is 48, other positional encoding methods achieve inferior accuracy (below 70%), while BiPE-ALiBi and BiPE-RoPE achieve high accuracy of 97% and 95% respectively. This result indeed well aligns with our theoretical results in Section 3.3, which further serves as a strong support for the bilevel design of our BiPE.

4.2. Length Extrapolation Experiments

Tasks. We test the length extrapolation capability of Transformer-based language models with different positional encoding methods. Following Chi et al. (2022), we use the Pile (Gao et al., 2020) dataset as the pre-training corpus and evaluate the log perplexity of pre-trained language models on the test set of PG19 (Rae et al., 2020b), arXiv and Github (Gao et al., 2020). We conduct the non-overlapping evaluation when computing the perplexity score.

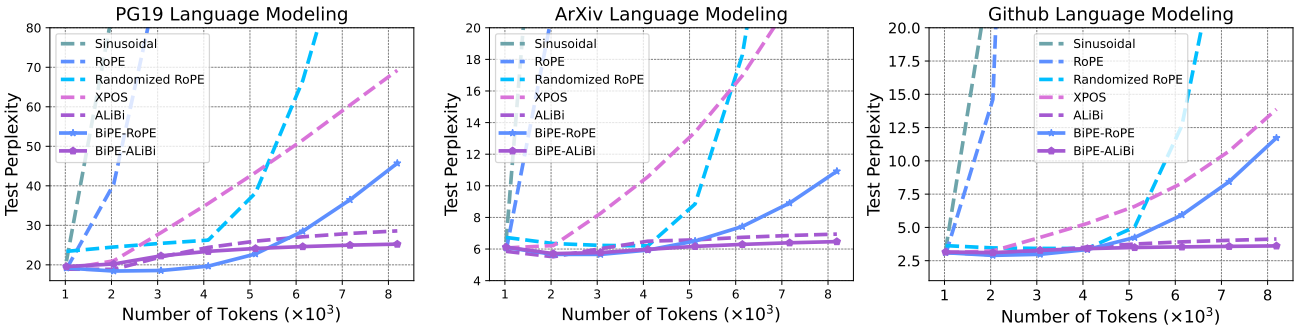

Figure 4. Language modeling perplexity with varying evaluation sequence lengths for models trained on sequence length 1024.

Settings. We set the pre-training sequence length to 1024, and evaluate the zero-shot perplexity on sequence lengths $[1024, 2048, 3072, 4096, 5120, 6144, 7168, 8192]$ on downstream datasets. We choose the standard decoder-only Transformer as the base model and compare our BiPE methods (BiPE-RoPE and BiPE-ALiBi) with the following positional encodings: 1) Sinusoidal PE (Vaswani et al., 2017); 2) RoPE (Su et al., 2021); 3) Randomized RoPE (Ruoss et al., 2023a); 4) XPOS (Sun et al., 2023); 5) ALiBi (Press et al., 2022). The segment boundary is determined by full stop "`.`" and newline "`\n`" for general purposes. For the Transformer-based language model, we set the number of layers to 12, the hidden dimension to 768, and the number of attention heads to 12. The total number of model parameters is approximately 155M. Additional experimental details are presented in Appendix D.2.

Results. The results are presented in Figure 4. Our BiPE methods achieve consistently superior performance on sequences with lengths larger than the training length. For example, our BiPE-ALiBi outperforms its counterpart ALiBi, which is also the best baseline method, by 3.35 points (25.24 v.s. 28.59 perplexity) on PG19 with 8192 sequence length. Compared to RoPE which performs well on sequences with the in-distribution length but yields a significant performance drop on longer sequences, our BiPE method substantially improves its length extrapolation capabilities, e.g., 19.67 v.s. 158 perplexity on PG19 with the 4096 sequence length. Notably, the benefit brought by our BiPE is also consistent across all three evaluation datasets covering text data in different modalities, underscoring the better length extrapolation capability of our BiPE in real-world tasks.

Integrating BiPE with fine-tuning strategies. One line of recent improvements on length extrapolation comes from continued fine-tuning RoPE-based language models with Position Interpolation techniques (Chen et al., 2023b; Peng et al., 2023). To further investigate the compatibility of our BiPE method with Position Interpolation, we use YaRN (Peng et al., 2023) to finetune the language model pre-trained on the Pile dataset with RoPE (Su et al., 2021) and our BiPE-RoPE, and check the improvements on downstream datasets. The results are presented in Figure 5. Similar to the zero-shot evaluation setting, our BiPE-RoPE achieves consistently better performance on longer sequences compared to RoPE after finetuning. Furthermore, although YaRN improves the length extrapolation capability of RoPE to some extent, it still suffers from performance drop when being evaluated on very long sequences, e.g., 11k/16k/16k for PG19/ArXiv/Github. In contrast, our BiPE-RoPE combined with YaRN yields much better length extrapolation capability, i.e. maintaining a consistently low perplexity across sequences with lengths up to 20k. Please refer to Appendix D.3 for more experimental details.

4.3. Long Context Benchmark

Tasks and settings. To evaluate the model's performance of long context understanding, we further fine-tune the pre-trained checkpoints on SCROLLS (Shaham et al., 2022), a long text benchmark that consists of seven distinct datasets covering different tasks. Following Shaham et al. (2022); Ainslie et al. (2023), we use three evaluation metrics for different tasks: Rgm score (the geometric mean of ROUGE-1,2,L), unigram overlap (F1) and exact match (EM). The average score across different datasets is also reported. We finetune pre-trained models using a sequence length of 8192 and select the model checkpoint that achieves the best performance on the validation set for the final evaluation. The test results are obtained from the official SCROLLS website. Additional experimental details are presented in Appendix D.4.

Results. The empirical results are provided in Table 1. First, BiPE-RoPE and BiPE-ALiBi exhibit better performance than RoPE and ALiBi, respectively. For example, our BiPE-RoPE outperforms its counterpart RoPE, which is also the best baseline method, by 3.98 points (22.36 v.s. 18.38 average score). Besides, BiPE-RoPE achieves the highest average score, surpassing other methods by a margin of over 3 points. On a task-by-task basis, BiPE-RoPE achieves the top score in 4 out of the 7 tasks. We also compare the two YaRN-finetuned models, i.e., BiPE-RoPE$_{yarn}$ and RoPE$_{yarn}$. We can see that BiPE-RoPE$_{yarn}$ still consistently outperforms RoPE$_{yarn}$ across 6 of 7 tasks and achieves a better average score. The results strengthen the

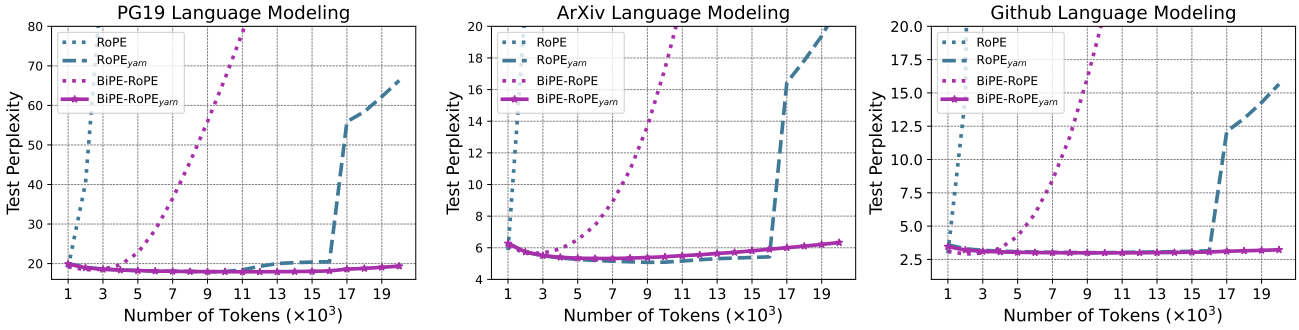

Figure 5. Language modeling perplexity with varying evaluation sequence lengths for RoPE and BiPE-RoPE finetuned with YaRN.

Table 1. Performance comparison on SCROLLS benchmark. Abbreviations for dataset names: Qasper (Qas), ContractNLI (CNLI), QMSum (QMS), NarrativeQA (NQA), SummScreenFD (SumS), GovReport (GovR), and QuALITY (QuAL). Rgm denotes the geometric mean of ROUGE-1,2, L. The statistics of median sequence lengths are from Ainslie et al. (2023); Li et al. (2023). Best performing results are highlighted in **bold**.

	QAS	CNLI	QMS	NQA	SumS	GovR	QuAL	Average
Metric	F1	EM	Rgm	F1	Rgm	Rgm	EM	
Median length	5472	2148	14197	57829	9046	8841	7171	
Sinusoidal	9.3	57.7	**12.42**	10.1	7.46	12.49	1.9	15.89
Randomized RoPE	12.3	52.4	11.80	10.8	7.19	18.95	**10.4**	17.71
ALiBi	12.0	68.8	10.27	3.2	6.00	23.14	0.0	17.62
BiPE-ALiBi	12.7	67.8	10.44	2.6	7.89	27.52	0.0	18.34
RoPE	16.4	67.8	10.13	9.7	**9.88**	14.33	0.4	18.38
BiPE-RoPE	**21.2**	**68.9**	10.64	**12.3**	8.13	**27.92**	7.4	**22.36**
RoPE$_{yarn}$	14.7	66.9	9.04	12.2	8.48	27.56	**22.2**	23.01
BiPE-RoPE$_{yarn}$	**20.9**	**69.0**	**10.57**	**13.3**	**9.40**	**28.31**	20.3	**24.53**

effectiveness of BiPE in long-context modeling.

Discussions. We can also observe that the performance gap between RoPE and BiPE-RoPE is more significant than that between ALiBi and BiPE-ALiBi. We hypothesize that this phenomenon is due to the design differences between ALiBi and RoPE. ALiBi incorporates relative positional information as an additive term in the attention module with an exponential decay rate as the relative distance increases. Using ALiBi on segment indexes (BiPE-ALiBi) will still bias the attention module towards local attention (Chi et al., 2023). Thus, the performance gap is not substantial. Different from ALiBi, RoPE rotates the query and key vectors and allows the context to determine the positional correlations. Therefore, the change between BiPE-RoPE and RoPE is more significant, leading to a larger performance gap.

4.4. Normal-length Benchmark

Tasks and settings. In this experiment, we evaluate the zero-shot and few-shot performance (Gao et al., 2023) of pre-trained models on a range of "in-distribution" benchmark tasks where the sequence length is normal. In particular, we use RACE (Lai et al., 2017), WinoGrade (Sakaguchi et al., 2020), TruthfulQA mc2 (Lin et al., 2022), and PIQA (Gao et al., 2020) benchmarks for the zero-shot evaluation, and employ 10-shot HellaSwag (Zellers et al., 2019) and 5-shot MMLU (Hendrycks et al., 2021) for the few-shot evaluation. The evaluation metrics are task-specific: for RACE, WinoGrande, TruthfulQA, PIQA, and MMLU, we report accuracy; and for HellaSwag, we report normalized accuracy.

Results. The empirical results are provided in Table 2. It can be easily seen that BiPE-RoPE and BiPE-ALiBi achieve comparable performance with other positional encoding methods on sequences with in-distribution lengths, which demonstrates that our BiPE methods achieve better length extrapolation performance without sacrificing in-distribution performance.

4.5. Ablation Study

Effectiveness of each positional encoding. BiPE leverages two positional encodings, with one corresponding to the token index within each segment (intra-segment encoding) and another for the segment index (inter-segment encoding).

Table 2. Zero-shot and few-shot performance on standard benchmarks. BiPE-based models perform on par with other methods.

Model	Zero-Shot				Few-Shot	
	RACE	WinoGrande	TruthfulQA	PIQA	HellaSwag	MMLU
Sinusoidal	27.85	52.09	45.92	60.88	29.70	26.49
Randomized RoPE	26.41	50.99	45.53	60.66	29.30	25.20
XPOS	27.56	52.96	45.22	60.88	30.86	25.96
ALiBi	27.08	52.72	46.24	60.50	31.47	26.49
BiPE-ALiBi	28.42	49.25	45.79	60.72	30.60	25.74
RoPE	29.00	51.54	44.67	60.66	30.86	26.43
BiPE-RoPE	28.04	52.01	45.64	59.74	30.93	26.91
RoPE$_{yarn}$	27.08	53.35	45.69	60.12	30.52	26.16
BiPE-RoPE$_{yarn}$	27.56	51.38	45.80	60.72	30.61	26.22

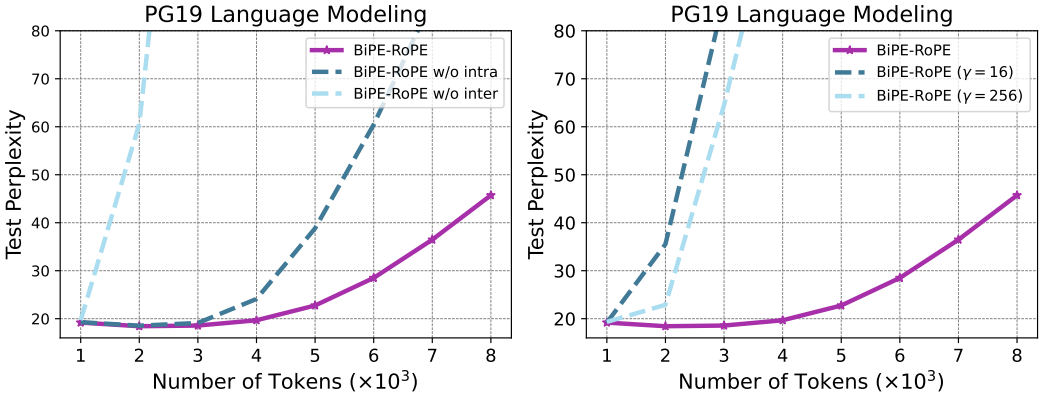

Figure 6. **Left:** Language modeling perplexity with varying evaluation sequence lengths for BiPE-RoPE without intra-segment encoding or inter-segment encoding on PG19 dataset. **Right:** Language modeling perplexity with varying evaluation sequence lengths for BiPE-RoPE with fixed segment lengths on PG19 dataset.

To check their effectiveness, we conducted an experiment where we removed one encoding at a time to assess the impact on model performance. We follow the same pre-training setting in Section 4.2 and evaluate perplexity on PG19. In Figure 6 left, the degradation in performance is observed upon the removal of either encoding, which clearly demonstrates that both of them are important.

Segmentation choices. In Section 4.2, we use full stop "`.`" and newline "`\n`" for text segmentation. One may wonder whether using a fixed segment length instead of pre-defined symbols works in practice. To check this, we assume each length is constant γ, pre-train BiPE language models and evaluate the performance on the PG19 dataset. In Figure 6 right, we can see that this naive approach does not achieve the same level of performance as using natural segmentation.

5. Conclusion and Future Directions

In this paper, we introduce BiPE, a novel bilevel positional encoding scheme designed to improve length extrapolation. For each position, our BiPE combines 1) an intra-segment encoding that identifies the location within its segment via APE, and 2) an inter-segment encoding that specifies the segment to which it belongs via RPE. The intra-segment encoding assists the model in capturing the semantic information within each segment and and the inter-segment encoding models the relationships between segments. This bilevel design well aligns with the intrinsic segmentation of text data and enhances length extrapolation. Our BiPE is further supported by theoretical analysis of its expressiveness. All experiments verify the length extrapolation capability of our BiPE across tasks of different text modalities.

There are also several future directions worth investigating. First, the intrinsic segmentation of text data yields a hierarchical structure, e.g., sentences→paragraphs→documents. It would be beneficial to confirm whether expanding our bilevel design to a hierarchical version results in improved length extrapolation. Second, there exist sequence data that do not have clear boundary of segmentations, e.g., time series, amino acid and gene sequence. Future research could explore better and more comprehensive segmentation methods for general purposes with our BiPE method.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgement

Di He is supported by National Key R&D Program of China (2022ZD0160300) and National Science Foundation of China (NSFC62376007). Liwei Wang is supported by National Science Foundation of China (NSFC62276005).

References

Ainslie, J., Lei, T., de Jong, M., Ontanon, S., Brahma, S., Zemlyanskiy, Y., Uthus, D., Guo, M., Lee-Thorp, J., Tay, Y., Sung, Y.-H., and Sanghai, S. CoLT5: Faster long-range transformers with conditional computation. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Alur, R., Kannan, S., and Yannakakis, M. Communicating hierarchical state machines. In *Automata, Languages and Programming: 26th International Colloquium, ICALP'99 Prague, Czech Republic, July 11–15, 1999 Proceedings 26*, pp. 169–178. Springer, 1999.

Anil, C., Wu, Y., Andreassen, A., Lewkowycz, A., Misra, V., Ramasesh, V., Slone, A., Gur-Ari, G., Dyer, E., and Neyshabur, B. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556, 2022.

Bai, H., Shi, P., Lin, J., Xie, Y., Tan, L., Xiong, K., Gao, W., and Li, M. Segatron: Segment-aware transformer for language modeling and understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 12526–12534, 2021.

Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan): 993–1022, 2003.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chen, G., Li, X., Meng, Z., Liang, S., and Bing, L. Clex: Continuous length extrapolation for large language models. *arXiv preprint arXiv:2310.16450*, 2023a.

Chen, M., Chu, Z., Wiseman, S., and Gimpel, K. SummScreen: A dataset for abstractive screenplay summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8602–8615, May 2022.

Chen, S., Wong, S., Chen, L., and Tian, Y. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023b.

Chi, T.-C., Fan, T.-H., Ramadge, P. J., and Rudnicky, A. Kerple: Kernelized relative positional embedding for length extrapolation. *Advances in Neural Information Processing Systems*, 35:8386–8399, 2022.

Chi, T.-C., Fan, T.-H., Rudnicky, A., and Ramadge, P. Dissecting transformer length extrapolation via the lens of receptive field analysis. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13522–13537, 2023.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. Palm: Scaling language modeling with pathways, 2022.

Chowdhury, J. R. and Caragea, C. Monotonic location attention for length generalization. *arXiv preprint arXiv:2305.20019*, 2023.

Chung, J., Ahn, S., and Bengio, Y. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016.

Dasigi, P., Lo, K., Beltagy, I., Cohan, A., Smith, N. A., and Gardner, M. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2021.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. 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)*, June 2019a.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. 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, June 2019b.

Eilenberg, S. *Automata, languages, and machines*. Academic press, 1974.

Feng, G., Zhang, B., Gu, Y., Ye, H., He, D., and Wang, L. Towards revealing the mystery behind chain of thought: A theoretical perspective. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Fine, S., Singer, Y., and Tishby, N. The hierarchical hidden markov model: Analysis and applications. *Machine learning*, 32:41–62, 1998.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Griffiths, T., Jordan, M., Tenenbaum, J., and Blei, D. Hierarchical topic models and the nested chinese restaurant process. *Advances in neural information processing systems*, 16, 2003.

Halliday, M. A. K. and Matthiessen, C. M. *Halliday's introduction to functional grammar*. Routledge, 2013.

Han, C., Wang, Q., Xiong, W., Chen, Y., Ji, H., and Wang, S. Lm-infinite: Simple on-the-fly length generalization for large language models, 2023.

Haviv, A., Ram, O., Press, O., Izsak, P., and Levy, O. Transformer language models without positional encodings still learn positional information. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 1382–1390, December 2022.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Huang, L., Cao, S., Parulian, N., Ji, H., and Wang, L. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1419–1436, June 2021.

Jin, H., Han, X., Yang, J., Jiang, Z., Liu, Z., Chang, C.-Y., Chen, H., and Hu, X. Llm maybe longlm: Self-extend llm context window without tuning. *arXiv preprint arXiv:2401.01325*, 2024.

Kazemnejad, A., Padhi, I., Ramamurthy, K. N., Das, P., and Reddy, S. The impact of positional encoding on length generalization in transformers. *arXiv preprint arXiv:2305.19466*, 2023.

Kočiský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6, 2018.

Koreeda, Y. and Manning, C. ContractNLI: A dataset for document-level natural language inference for contracts. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 1907–1919, November 2021.

Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 785–794, September 2017.

Li, S., You, C., Guruganesh, G., Ainslie, J., Ontanon, S., Zaheer, M., Sanghai, S., Yang, Y., Kumar, S., and Bhojanapalli, S. Functional interpolation for relative positions improves long context transformers. *arXiv preprint arXiv:2310.04418*, 2023.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Lin, S., Hilton, J., and Evans, O. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, May 2022.

Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. Transformers learn shortcuts to automata. *arXiv preprint arXiv:2210.10749*, 2022.

Liu, X., Yan, H., Zhang, S., An, C., Qiu, X., and Lin, D. Scaling laws of rope-based extrapolation. *arXiv preprint arXiv:2310.05209*, 2023.

Liu, Y. Fine-tune bert for extractive summarization. *arXiv preprint arXiv:1903.10318*, 2019.

Pang, R. Y., Parrish, A., Joshi, N., Nangia, N., Phang, J., Chen, A., Padmakumar, V., Ma, J., Thompson, J., He, H., and Bowman, S. QuALITY: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5336–5358, July 2022.

Peng, B., Quesnelle, J., Fan, H., and Shippole, E. Yarn: Efficient context window extension of large language models, 2023.

Press, O., Smith, N., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations (ICLR)*, 2022.

Rae, J. W., Potapenko, A., Jayakumar, S. M., Hillier, C., and Lillicrap, T. P. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations (ICLR)*, 2020a.

Rae, J. W., Potapenko, A., Jayakumar, S. M., Hillier, C., and Lillicrap, T. P. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020b.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Ratner, N., Levine, Y., Belinkov, Y., Ram, O., Magar, I., Abend, O., Karpas, E., Shashua, A., Leyton-Brown, K., and Shoham, Y. Parallel context windows for large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6383–6402, 2023.

Roziere, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Remez, T., Rapin, J., et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Ruoss, A., Delétang, G., Genewein, T., Grau-Moya, J., Csordás, R., Bennani, M., Legg, S., and Veness, J. Randomized positional encodings boost length generalization of transformers. In *Association for Computational Linguistics (ACL)*, July 2023a.

Ruoss, A., Delétang, G., Genewein, T., Grau-Moya, J., Csordás, R., Bennani, M., Legg, S., and Veness, J. Randomized positional encodings boost length generalization of transformers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1889–1903, July 2023b.

Sakaguchi, K., Le Bras, R., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 8732–8740, 2020.

Shaham, U., Segal, E., Ivgi, M., Efrat, A., Yoran, O., Haviv, A., Gupta, A., Xiong, W., Geva, M., Berant, J., et al. Scrolls: Standardized comparison over long language sequences. *arXiv preprint arXiv:2201.03533*, 2022.

Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding, 2021.

Sun, Y., Dong, L., Patra, B., Ma, S., Huang, S., Benhaim, A., Chaudhary, V., Song, X., and Wei, F. A length-extrapolatable transformer. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, July 2023.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., brian ichter, Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks, 2023.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Zhong, M., Yin, D., Yu, T., Zaidi, A., Mutuma, M., Jha, R., Awadallah, A. H., Celikyilmaz, A., Liu, Y., Qiu, X., and Radev, D. QMSum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5905–5921, June 2021.

Zhu, D., Yang, N., Wang, L., Song, Y., Wu, W., Wei, F., and Li, S. Pose: Efficient context window extension of llms via positional skip-wise training. *arXiv preprint arXiv:2309.10400*, 2023.

A. Proofs

A.1. Technical Lemmas

In this subsection, we present some technical lemmas that show how the MLP and Transformer can perform basic operations. We show that the MLP with GeLU activation can perform scalar multiplication, selection, and Boolean matrix multiplication; and that the attention layer can perform the COPY operation.

Lemma A.1 (From Feng et al. (2023))**.** *For any $\epsilon > 0$ and $M > 0$, there exist a two-layer MLP $f : \mathbb{R}^2 \to \mathbb{R}$ with* GeLU *activation and parameters with ℓ_∞ norm upper bounded by $O(\mathrm{poly}(M, 1/\epsilon))$ such that $|f(a, b) - ab| \leq \epsilon$ holds for all $a, b \in [-M, M]$.*

Lemma A.2 (From Feng et al. (2023))**.** *Let $\boldsymbol{g} : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ be a two-layer MLP with* ReLU *activation, and all parameter values are upper bounded by M. For any $\epsilon > 0$, there exists a two-layer MLP $\boldsymbol{f}$ of the same size with* GeLU *activation and parameters upper bounded by $O(\mathrm{poly}(M, 1/\epsilon))$ in the ℓ_∞ norm, such that for all $\boldsymbol{x} \in \mathbb{R}^{d_1}$, we have $\|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{x})\|_\infty \leq \epsilon$.*

Lemma A.3 (From Feng et al. (2023))**.** *Define the selection function $\boldsymbol{g} : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ as follows:*

$$\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}, t) = \begin{cases} \boldsymbol{x} & \text{if } t \geq 0, \\ \boldsymbol{y} & \text{if } t < 0. \end{cases} \tag{1}$$

Let $\boldsymbol{f} : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ be a two-layer MLP with GeLU *activation. Then, for any $\epsilon > 0$, $\alpha > 0$, and $M > 0$, there exist MLP parameters with ℓ_∞ norm bounded by $O(\mathrm{poly}(M, 1/\alpha, 1/\epsilon))$, such that for all $\boldsymbol{x} \in [-M, M]^d$, $\boldsymbol{y} \in [-M, M]^d$, and $t \in [-\infty, -\alpha] \cup [\alpha, +\infty]$, we have $\|\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}, t) - \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}, t)\|_\infty \leq \epsilon$.*

Lemma A.4. *Let $\boldsymbol{f} : \mathbb{R}^{d_1 \times d_2} \times \mathbb{R}^{d_2 \times d_3} \times \to \mathbb{R}^{d_1 \times d_3}$ be a two-layer MLP with* GeLU *activation, and given a Boolean matrix $\mathbf{B}$, let $\boldsymbol{e}_{\mathbf{B}}$ be the vector by flattening the matrix $\mathbf{B}$. Then, for any $\epsilon > 0$, $\alpha > 0$, and $M > 0$, there exist MLP parameters with ℓ_∞ norm bounded by $O(\mathrm{poly}(1/\epsilon))$, such that for all $\mathbf{A} \in \{0, 1\}^{d_1 \times d_2}$ and $\mathbf{B} \in \{0, 1\}^{d_2 \times d_3}$, we have $\|\boldsymbol{f}(\boldsymbol{e}_{\mathbf{A}}, \boldsymbol{e}_{\mathbf{B}}) - \boldsymbol{e}_{\mathbf{A} \cdot \mathbf{B}}\|_\infty \leq \epsilon$.*

The proof of Lemmas A.1 to A.3 can be found in the appendix of Feng et al. (2023), and we will give the proof of Lemma A.4.

Proof of Lemma A.4. Given two Boolean matrices $\mathbf{A} \in \{0, 1\}^{d_1 \times d_2}$ and $\mathbf{B} \in \{0, 1\}^{d_2 \times d_3}$. We can represent the output $\mathbf{A} \cdot \mathbf{B}$ by the following formula:

$$\begin{aligned} (\mathbf{A} \cdot \mathbf{B})_{i,j} &= \bigvee_{k \in [d_2]} (\mathbf{A}_{i,k} \wedge \mathbf{B}_{k,j}) \\ &= \mathrm{ReLU}\Big(\sum_{k \in [d_2]} \mathrm{ReLU}(\mathbf{A}_{i,k} + \mathbf{B}_{k,j} - 1) \Big) - \mathrm{ReLU}\Big(\sum_{k \in [d_2]} \mathrm{ReLU}(\mathbf{A}_{i,k} + \mathbf{B}_{k,j} - 1) - 1 \Big) \end{aligned}$$

Therefore, we can implement the Boolean matrix multiplication by the MLP with ReLU activation, and according to Lemma A.2, the MLP with GeLU activation can perform the Boolean matrix multiplication. □

Then we introduce a basic operation that can be implemented by the attention layer. And following it, we give a special form of this operation used in the proof of our main theorems.

Let n be an integer and $\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_n$ be a sequence of vectors, whose ℓ_∞ norm is bounded by a large constant M. Let $\mathbf{K}, \mathbf{Q} \in \mathbb{R}^{d' \times d}$ be any matrices, and let $0 < \rho < M$ be any real numbers. Denote $\boldsymbol{q}_i = \mathbf{Q}\boldsymbol{x}_i$ and $\boldsymbol{k}_j = \mathbf{K}\boldsymbol{x}_j$. The output of the COPY operation is a sequence of vectors $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n$ with $\boldsymbol{u}_i = \boldsymbol{e}_{\mathrm{pos}(i)}$, where $\mathrm{pos}(i) = \mathrm{argmax}_{j \in [i]} \boldsymbol{q}_i \cdot \boldsymbol{k}_j$. Moreover, we assume that the matrices $\mathbf{Q}, \mathbf{K}$ and scalars δ satisfy that for all considered sequences $\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_n$, we have for any i and $j \in [n] \backslash \{\mathrm{pos}(i)\}$, either $\boldsymbol{q}_i \cdot \boldsymbol{k}_{\mathrm{pos}(i)} - \boldsymbol{q}_i \cdot \boldsymbol{k}_j \geq \delta$. This assumption guarantees that there are sufficient gaps between the attended position and other positions. Then we prove that the attention layer can implement the COPY operation.

Lemma A.5 (From Feng et al. (2023))**.** *For any $\epsilon > 0$, there exists an attention layer with embedding size $O(d)$ and one causal attention head that can approximate the COPY operation defined above. Formally, for any considered sequence of vectors $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n$, denote the corresponding attention output as $\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_n$. Then, we have $\|\boldsymbol{o}_i - \boldsymbol{u}_i\|_\infty \leq \epsilon$ for all $i \in [n]$. Moreover, the ℓ_∞ norm of attention parameters is bounded by $O(\mathrm{poly}(M, 1/\delta, \log(n), \log(1/\epsilon)))$.*

The proof of this lemma can be found in Feng et al. (2023). Based on the Lemmas A.1 and A.5, given the token index i as the absolute PE, the attention layer can copy the embedding at the specific position by the following construction. Given a specific position i, we can construct the query $\boldsymbol{q} = (1, i^2, i)$ and the key of the j-th token $\boldsymbol{k}_j = (-j^2, -1, 2j)$ by MLP, according to Lemma A.1. Then $\boldsymbol{q} \cdot \boldsymbol{k}_j = -(i-j)^2$ gets the maximum when $i = j$, and we can concentrate the attention on the i-th token and copy its embedding by Lemma A.5.

A.2. Proofs of Main Theorems

In this subsection, we will prove Theorems 3.1 and 3.2, for ease of reading we restate the theorems here and then give proof.

A.2.1. Proof of Theorem 3.1

Theorem A.6 (Lower Bound for Transformer with APE to Represent NFA)**.** *For any size of state set, there exists an NFA $\mathcal{N} = (Q, \Sigma, \delta, q_0, F)$ such that a Transformer with APE needs at least $O(|Q|^2)$ embedding size to represent the NFA.*

We prove this theorem in the log-precision setting, which is a realistic and practical setting for transformers. In this setting, each value in the transformer is encoded by $O(\log N)$ bits, where N is the input length. This corresponds to the practical scenario where the transformer handles input sequences of up to a few thousand tokens, using 16 or 32 bits floating-point numbers. Theoretically, the log-precision number can approximate any real number of magnitude $O(\text{poly}(N))$ with an error of $O(\text{poly}(1/N))$. Each neuron in the transformer can store only $O(\log(n))$-bits information and thus cannot retain the full information of the entire input sequence, which is reasonable and aligned with practical scenarios.

Proof of Theorem A.6. Given the state set Q, we can construct a NFA $\mathcal{N} = (Q, \Sigma, \delta, q_0, F)$ as follows:

- The state set Q is given, and assuming $Q = \{q_1, q_2, \cdots, q_n\}$.
- Σ is the set of all maps from the state set Q to its power set, i.e. $\Sigma = \{f : Q \to \mathcal{P}(Q)\}$.
- Given a state $q \in Q$ and a symbol $f \in \Sigma$, $\delta(q, f) = f(q)$.
- $q_0 = q_1$.
- $F = \{q_n\}$.

To prove that the embedding dimension of the transformer to represent $\mathcal{N}$ is at least $O(n^2)$, we use the following argument. The size of the alphabet is 2^{n^2}, so we need at least $O(n^2)$ bits to embed each input symbol. Suppose the embedding dimension is $o(n^2)$. Then we can find two input sequences of constant size that have the same embeddings in the transformer but different outcomes for the NFA. Since the length of the input sequence is constant, the transformer uses $o(n^2)$ bits to represent the embedding for each token. Therefore, there exist two input tokens f and f' that have the same embedding. Let $q_i \in f(q_j)$ and $q_i \notin f'(q_j)$. Then we can construct two input sequences $\mathbf{S} = [f_1, f, f_2]$ and $\mathbf{S}' = [f_1, f', f_2]$, where $f_1(q_0) = \{q_j\}$, $f_2(q_i) = \{q_n\}$, and $f_2(q) = \emptyset$ for $q \neq q_i$. The embeddings of these two sequences in the transformer are the same, but one is accepted by $\mathcal{N}$ while the other is not. Hence, the transformer with embedding size $o(n^2)$ cannot represent the NFA $\mathcal{N}$. A transformer with APE needs at least $O(n^2)$ embedding size to represent the NFA. □

A.2.2. Proof of Theorem 3.2

Theorem A.7 (Upper Bound for Transformer with BiPE to Represent Bi-NFA)**.** *For any Bi-NFA $\mathcal{N} = (\mathcal{Q}, \Sigma, \delta, q_0, F)$, $\mathcal{Q} = \{Q_1, Q_2, \cdots, Q_k\}$ there exists a Transformer with BiPE and $O(k^2 + \sum_{i\in[k]} |Q_i|^2)$ embedding size can represent the Bi-NFA.*

In this proof, we use the log-precision transformer with the GeLU activation function and $O(\log N)$ layers, where N is the input length. This choice of layers is crucial for the transformer to represent automata, as a constant-layer transformer would require a super-polynomial embedding size (in the input length) to do so (Liu et al., 2022). Without loss of generality, we focus on the Bi-PE model with T5-relative PE as the inter-segment positional encoding and APE as the intra-segment positional encoding. Moreover, our proof can be easily extended to other variants of Bi-PE, such as those with RoPE or AliBi as the inter-segment positional encoding.

T5-relative PE. We use the T5-relative PE method to compute the attention logits before applying the softmax function. The attention logits are given by the following equation:

$$\mathbf{A}_{\text{RPE}}(\mathbf{X}) = \mathbf{X}\mathbf{W}_Q(\mathbf{X}\mathbf{W}_K)^\top + \mathbf{B}$$

where $\mathbf{B}_{i,j} = r_{\min(i-j,K)}$, K is a hyper-parameter, and $\{r_i\}_{i=0}^K$ are learnable scalars that represent the relative position embeddings.

Proof Sketch. In this proof, we construct a transformer containing two modules to represent the Bi-NFA. The first module computes the state transitions in the segment, and the second module computes the state transitions between the segments. Each module contains $O(\log N)$ attention layers and implements a classic divide-and-conquer algorithm. □

Proof of Theorem A.7. Given a Bi-NFA $\mathcal{N} = (\mathcal{Q}, \Sigma, \delta, q_0, F)$, we will first introduce some notations and the basic idea behind our construction.

The State Transition in Each Segment. Given $\mathcal{Q} = \{Q_1, \cdots, Q_k\}$, we use $q_{i,j}$ for the j-th state in Q_i and define $Q = \bigcup_{i\in[k]} Q_i$, $Q^* = \{q_i^*\}$. Without loss of generality, we assume that $q_{i,1}$ is the start state q_i^* in Q_i and $q_0 = q_1^*$. Given an input symbol w, we view it as a map f_w from Q to $\mathcal{P}(Q)$ such that $f_w(q) = \delta(w, q)$, and an input string $\mathbf{S} = [w_1, w_2, \cdots, w_n]$ as the composition of $f_s = f_{w_1} \cdot f_{w_2} \cdots f_{w_n}$. Denoting the segment separator as $w^* \in \Sigma$, for any input symbol $w \neq w^*$, we use a tuple of boolean matrices $\mathbf{M}(w) = \big(\mathbf{M}_1(w), \mathbf{M}_2(w), \cdots, \mathbf{M}_n(w)\big)$ to represent it, such that $\mathbf{M}_{i,j,k}(w) = \big(\mathbf{M}_i(w)\big)_{j,k} = \mathbb{I}[q_{i,j} \in \delta(w, q_{i,k})]$. We define the multiplication of two tuples of matrices as $\mathbf{M}(w) \cdot \mathbf{M}(w') = \big(\mathbf{M}_1(w) \cdot \mathbf{M}_1(w'), \cdots, \mathbf{M}_n(w) \cdot \mathbf{M}_n(w')\big)$, which is the composition of $f_w \cdot f_{w'}$. In the transformer, we flatten $\mathbf{M}(w)$ to a vector $\boldsymbol{m}(w)$. We use the MLP to implement the multiplication of two tuples of matrices according to Lemma A.4. Moreover, a string without a segment separator can be viewed as the multiplication of these tuples of matrices. Given a string without segment separator $S = [w_1, w_2, \cdots, w_n]$, we compute $\mathbf{M}(S) = \prod_{i\in[n]} \mathbf{M}(w_i)$ and denote the vector flattened from $\mathbf{M}(S)$ as $\boldsymbol{m}(S)$. Given the start state q_i^*, we get the state set of the Bi-NFA $\{q_{i,j} | \mathbf{M}_{i,1,j}(s) = 1\}$ after taking in the string S. Moreover, we use the notation $w_{i,j} : w_{i,k}$ and $\mathbf{M}(w_{i,j} : w_{i,k})$ to represent the substring from $w_{i,j}$ to $w_{i,k}$ and its state transition matrix tuple. We can use the classic divide-and-conquer algorithm to compute $\mathbf{M}(s) = \prod_{i\in[n]} \mathbf{M}(w_i)$, and the first module of the transformer we construct implements the algorithm.

The State Transition between Segments. A segment $S = [w_1, w_2, \cdots, w_n, w^*]$ can be viewed as a map from $\{q_1^*, q_2^*, \cdots, q_k^*\}$ to $\mathcal{P}(\{q_1^*, q_2^*, \cdots, q_k^*\})$. For the segment separator w^*, we can also view it as a tuple of matrices $\mathbf{M}(w^*) = \big(\mathbf{M}_1(w^*), \mathbf{M}_2(w^*), \cdots, \mathbf{M}_k(w^*)\big)$, where $\mathbf{M}_{i,j,k}(w^*) = \mathbb{I}[q_k^* \in \delta(w^*, q_{i,j})]$. Therefore, we can compute the state transition matrix $\mathbf{A}(S)$ of the segment S such that $\mathbf{A}_{i,j}(S) = \mathbf{M}_{i,1,j}(S)$. We have $\mathbf{A}_{i,j}(S) = \mathbb{I}[q_j^* \in \delta(S, q_i^*)]$. Given a sequence of segments $\mathbf{S} = S_1 \oplus S_2 \oplus \cdots \oplus S_n$, we can compute $\mathbf{A}(\mathbf{S}) = \Pi_{i\in[n]}\mathbf{A}(S_i)$. Then given the start state q_0, we can get the final state set $Q(\mathbf{S})$ after the Bi-NFA taking in the input $\mathbf{S}$ such that $Q(\mathbf{S}) = \{q_i^* | \mathbf{A}_{1,i}(\mathbf{S}) = 1\}$. Then the Bi-NFA accepts $\mathbf{S}$ if and only if $Q(\mathbf{S}) \cap F \neq \emptyset$, and this condition judgment can be formulated as the product of two Boolean vectors and therefore, can be implemented by MLP. For convenience and clarity in presenting our proof, in the representation of the transformer, we flatten the matrices $\mathbf{A}(\mathbf{S})$ and $\mathbf{A}(S)$ to vectors and we denote them as $\boldsymbol{a}(\mathbf{S})$ and $\boldsymbol{a}(S)$, respectively. We use the notation $S_i : S_j$ and $\mathbf{A}(S_i : S_j)$ to represent the substring from S_i to S_j and its state transition matrix.

Token Embeddings. Given a input sequence $\mathbf{S} = S_1 \oplus S_2 \oplus \cdots \oplus S_n$ and $S_i = [w_{i,1}, w_{i,2}, \cdots w_{i,l_i-1}, w_i^*]$, assuming the length of the sequence is upper boumded by N. As a standard way in NLP, we add a <SOS> at the beginning of the string. For each token $w_{i,j} \neq w^*$, and $w_{i,j} \neq$ <SOS>, after combining the absolute positional embedding, the embedding at the beginning is $\boldsymbol{x}_{i,j}^0 = \big(\boldsymbol{m}(w_{i,j}), 1, 0, 0, i\big)$, and the embeddings for w^* and <SOS> is $(\mathbf{0}, 0, 1, 0, l_i)$ and $(\mathbf{0}, 0, 0, 1, 1)$, respectively. The embedding size at begining is $O(\sum_{i\in[k]} |Q_i|^2)$.

Module I. The first module contains $(\lceil \log(N) \rceil + 1)$ layers, and in this module, the token only attends to the tokens in the same segment. Therefore, the T5-relative PE for these layers is $r_i = -\infty$ for $i \neq 0$ and $r_0 = 0$. At the layer l, the input embeddings of token $w_{i,j}$ is $\boldsymbol{x}_{i,j}^{1,l} = \big(\boldsymbol{m}(w_{i,\max(1,j-2^l)} : w_{i,j}), 1, 0, 0, i\big)$, where $w_{i,j_1} : w_{i,j_2} = [w_{i,j_1}, w_{i,j_1+1}, \cdots, w_{i,j_2}]$. The first module completes the following tasks:

- Copy the embedding of token $w_{i,j-2^l}$, note that when $j - 2^l < 1$, the embedding copied is meaningless.

- Calculate the multiplication of two tuples of transition matrices defined in the previous paragraph.
- Select the embedding, when $j - 2^l < 1$, the output embedding is the same as the input, and when $j - 2^l \geq 1$, the output is the outcome of the multiplication.

Therefore, the output embeddings of the token $w_{i,j}$ at the layer l is $\boldsymbol{x}_{i,j}^{1,l+1} = \big(\boldsymbol{m}(w_{i,\max(1,j-2^{(l+1)})} : w_{i,j}), 1, 0, 0, i\big)$. According to Lemmas A.3 to A.5 we can implement the COPY operation by the attention layer, and implement the multiplication of two tuples of matrices, and the selection operation by the MLP. After $\lceil \log(N) \rceil$ layers, the output embedding of $w_{i,j}$ is $\boldsymbol{x}_{i,j}^{1,\lceil \log(N) \rceil} = \big(\boldsymbol{m}(w_{i,1} : w_{i,j}), 1, 0, 0, i\big)$. At the final layer, the token w^* copies the embedding of the previous token w_{i,l_i-1}, and uses the MLP to compute the transition matrix of this segment. The final output of this block for the token w_i^* is $\big(\boldsymbol{a}(S_i), 0, 1, 0, l_i\big)$, for the token <SOS> is $(\mathbf{0}, 0, 0, 1, 1)$, and for the token $w_{i,j}$ is $(\mathbf{0}, 1, 0, 0, j)$. The embedding size of the first module is $O(\sum_{i\in[k]} |Q_i|^2)$.

Module II. The second module contains $\lceil \log(N) \rceil$ layers, and we only need to concentrate our attention on the embedding of the last token of each segment. Similar to the previous block, at the layer l, the input embeddings of token w_i^* is $\big(\boldsymbol{a}(S_{\max(1,i-2^l)} : S_i), 0, 1, 0, l_i\big)$, where $S_i : S_j = S_i \oplus S_{i+1} \oplus \cdots \oplus S_j$. The second module completes the following tasks:

- Copy the embedding of last token of segment S_{i-2^l}, note that when $i - 2^l < 1$, the embedding copied is meaningless.
- Calculate the multiplication of two transition matrices defined in the previous paragraph.
- Select the embedding: when $i - 2^l < 1$, the output embedding is the same as the input, and when $i - 2^l \geq 1$, the output is the outcome of the multiplication.

Therefore, the output embeddings of the token w_i^* is $\big(\boldsymbol{a}(S_{\max(1,i-2^{(l+1)})} : S_i), 0, 1, 0, l_i\big)$. we design the relative positional embedding as $r_i = -\infty$ for $i \neq 2^l$ and $r_{2^l} = 0$. Note that, when $i - 2^l < 1$, the token will give uniform attention to the last token of each segment previous to it, therefore, we can use the embedding of <SOS> to detect this case. When the value indicates <SOS> is greater than $\frac{1}{N}$, we have $i - 2^l < 1$ and maintain the embeddings. According to Lemmas A.3 to A.5 we can implement the copy operation by the attention layer, and implement the multiplication of two matrices and the selection operation by the MLP. Then we can get the final state set from the embedding of the last token of the input sequence and we can use an MLP to compute the outcome to determine accepting the input or not. The embedding size of the second module is $O(k^2)$.

Therefore, we construct a transformer with BiPE, and $O(k^2 + \sum_{i\in[k]} |Q_i|^2)$ embedding size can represent the Bi-NFA. □

B. More Related Works

B.1. Previous Bi-level Positional Encoding Approaches.

The original BERT (Devlin et al., 2019a) model also includes two encodings for representing positions, but its approach differs significantly from our BiPE. Primarily, BERT only needs to specify two segments using absolute encoding, tailored for the next sentence prediction task, not for length extrapolation. Furthermore, BERT treats a sequence as a flat array of tokens and defines the segments in an arbitrary way, ignoring the intrinsic segmentation of language data. Liu (2019) further extends BERT for summarization tasks. They modify the BERT configuration by encoding whether the segment index is odd or even and encoding the absolute token position in the whole sequence. It is noteworthy that the first one cannot carry enough segment-level positional signals, and the latter one faces the same problem as BERT and cannot extrapolate to longer contexts. Another similar work is Segatron (Bai et al., 2021). It introduces paragraph and sentence segmentation to the relative position encoding, which resembles the inter-segment encoding in BiPE. However, the absolute token position is still calculated in the whole sequence, which still incurs length extrapolation issues. In contrast, in our work, the intra-segment encoding only identifies the location of each token within the segment, and the inter-segment encoding specifies the segment indexes. By properly using different kinds of positional encodings, our BiPE can be used in longer sequences.

C. Visualization of Distribution

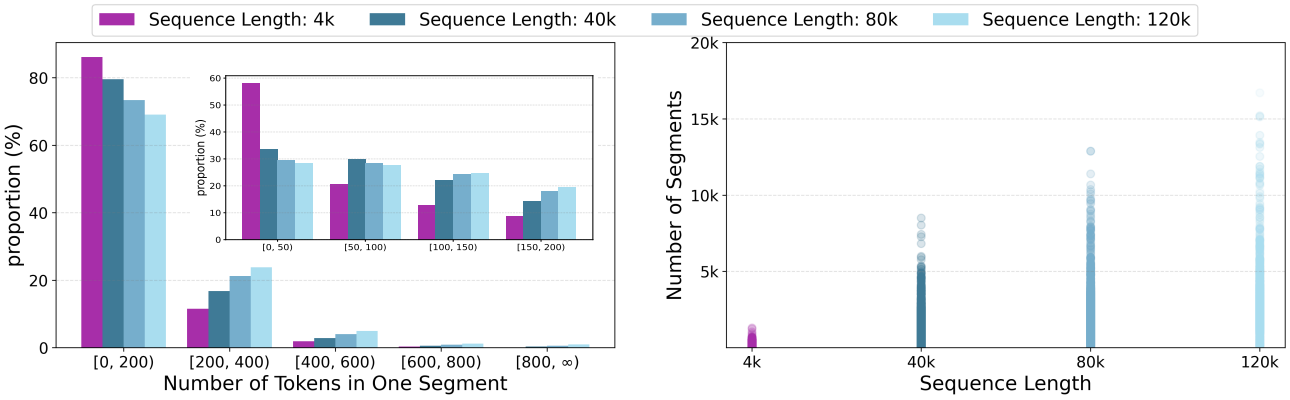

Figure 7. **Left:** The distribution of the token number in one segment with different sequence lengths. **Right:** The distribution of the number of segments with different sequence lengths. We use the tokenizer of Llama 2 (Touvron et al., 2023) for tokenization on ArXiv. Full stop"`.`" and newline "`\n`" are used for segmentation.

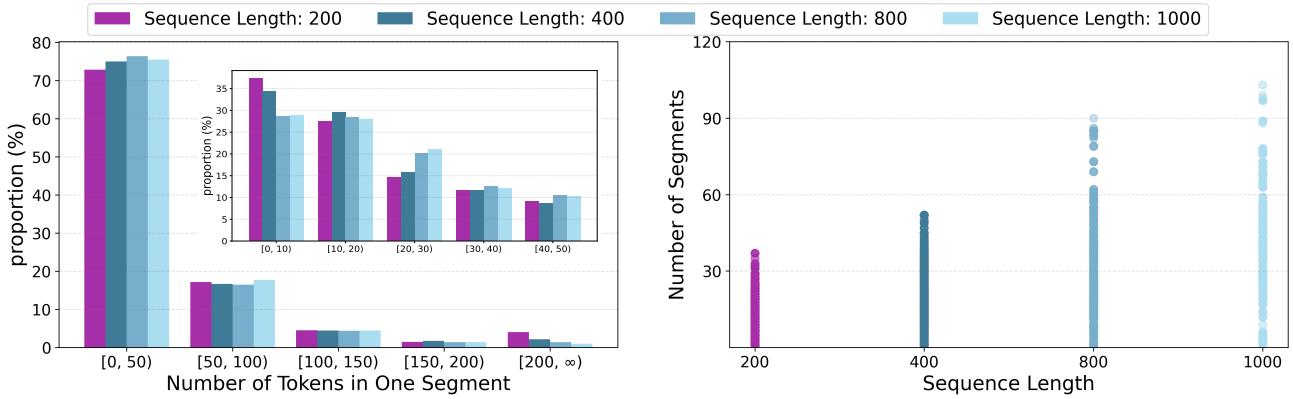

Figure 8. **Left:** The distribution of the token number in one segment with different sequence lengths. **Right:** The distribution of the number of segments with different sequence lengths. We use the tokenizer of Llama 2 (Touvron et al., 2023) for tokenization on PRM800K (Lightman et al., 2023). Full stop"`.`" and newline "`\n`" are used for segmentation.

D. Experimental Details

D.1. Capacity Experiments

Table 3. Model configurations for capacity experiments.

Layers	3
Attention heads	4
Head dimensions	$\{12, 16, 64\}$
Hidden dimensions	$\{48, 64, 256\}$
FFN dimensions	$\{192, 256, 1024\}$
Model parameters	$\{87\text{K}, 153\text{K}, 2.4\text{M}\}$

Table 4. Training recipes for capacity experiments.

Batch size	512
Epochs	100
Dropout	0.1
Weight decay	0.01
Optimizer	AdamW
Learning rate	$1\text{e}-4$

In this experiment, we use the Arithmetic task (Feng et al., 2023) to empirically verify the parameter efficiency brought by our BiPE method. Given an arithmetical expression consisting of numbers, basic operations $(+, -, \times, \div, =)$ and brackets, e.g., $(1+2)\times(3+5)=$, this task requires language models to calculate and generate the correct result, e.g., 24. Following Feng et al. (2023), we train all models using Chain-of-Thought demonstrations, e.g., for the input sequence $(7+8)\div(5+2\times7-2\times8)$, the output sequence is $15\div(5+2\times7-2\times8)=15\div(5+14-2\times8)=15\div(19-2\times8)=15\div(19-16)=15\div3=5$. The evaluation metric is the accuracy of the final answer. We refer interested readers to Appendix H in Feng et al. (2023) for additional details.

Model configurations. In this experiment, we train decoder-only Transformer-based language models with different positional encoding techniques while keeping all the other configurations the same. For Sinusoidal PE, we follow Vaswani et al. (2017) to set the hyperparameters in sine and cosine functions. For RoPE and XPOS, we follow Su et al. (2021); Sun et al. (2023) to set the hyperparameters in the rotary matrix respectively. For ALiBi, we follow Press et al. (2022) to set the slope values in each attention head. For the intra segment encoding of our BiPE, we use the learnable absolute positional encoding. For the inter segment encoding of our BiPE-RoPE, the hyperparameters are kept the same as (Su et al., 2021). For the inter segment encoding of our BiPE-ALiBi, the slope values are set to 96 times of the original ALiBi's setting. Other model configurations are provided in Table 3.

Training recipes. The next token prediction objective (Brown et al., 2020) is adopted for language model training. The number of operators in the arithmetic dataset is set to 6, which yields a total sequence length of 223 for Chain-of-Thought demonstrations. The training recipes are provided in Table 4. All models are trained on 4 NVIDIA A100 GPUs.

D.2. Length Extrapolation Experiments

Table 5. Model configurations for length extrapolation experiments.

Layers	12
Attention heads	12
Head dimensions	64
Hidden dimensions	768
FFN dimensions	3072
Model parameters	155M

Table 6. Training recipes for length extrapolation experiments.

Batch size	256
Total training steps	500k
Dropout	0.0
Weight decay	0.01
Optimizer	AdamW
Learning rate	$1e-4$

Model configurations. In this experiment, we train decoder-only Transformer language models with different positional encoding techniques while keeping all the other configurations the same. For Sinusoidal PE, we follow Vaswani et al. (2017) to set the hyperparameters in sine and cosine functions. For RoPE and XPOS, we follow Su et al. (2021); Sun et al. (2023) to set the hyperparameters in the rotary matrix respectively. For Randomized RoPE, we set the extended positions 4 times of the training length. We also conducted experiments on the extended positions 16 times of the training length in Figure 9, which shows performance degradation. For ALiBi, we follow Press et al. (2022) to set the slope values in each attention head. For the intra segment encoding of our BiPE, we use the learnable absolute positional encoding. For the inter segment encoding of our BiPE-RoPE, the hyperparameters are kept the same as (Su et al., 2021). For the inter segment encoding of our BiPE-ALiBi, the slope values are set to 96 times of the original ALiBi's setting. Other model configurations are provided in Table 5.

Training recipes. The next token prediction objective (Brown et al., 2020) is adopted for language model training. All models are trained on the Pile dataset[1] (Gao et al., 2020) with a total sequence length of 1024. The training recipes are shown in Table 6. All models are trained on 8 NVIDIA A100 GPUs.

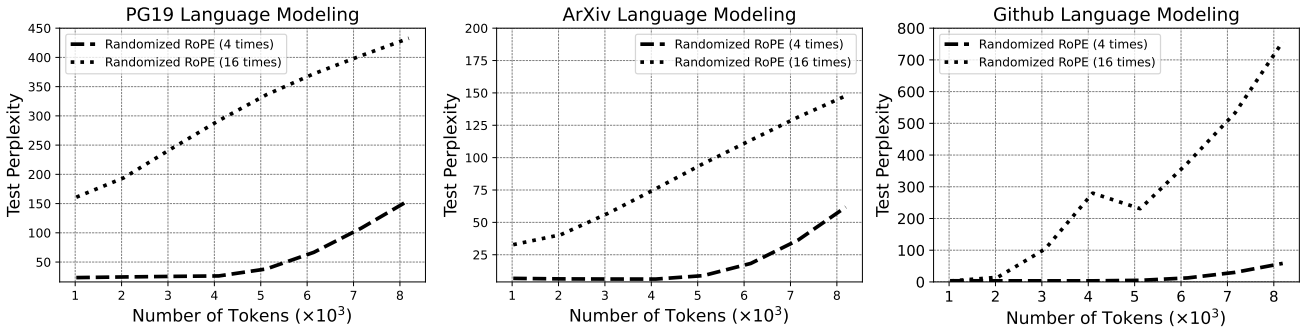

Figure 9. Language modeling perplexity with varying evaluation sequence lengths for Randomized RoPE trained on the Pile dataset with different times of the training length for extended positions.

[1]We use a copy of the Pile dataset with all copyrighted contents removed: https://huggingface.co/datasets/monology/pile-uncopyrighted.

D.3. Integrating BiPE with Fine-tuning Strategies

Table 7. Finetuning recipes for the YaRN strategy.

Batch size	64
Total training Steps	500
Dropout	0.0
Weight decay	0.01
Optimizer	AdamW
Learning rate	$2e-5$

Table 8. Finetuning recipes for long context benchmark.

Batch size	64
Total training steps	5000
Dropout	0.0
Weight decay	0.01
Optimizer	AdamW
Learning rate	$1e-5$

Model configurations. In this experiment, we use the YaRN strategy to fine-tune pre-trained language models with RoPE and our BiPE-RoPE. All the model configurations are the same as those in Table 5.

Fine-tuning recipes. We set the scale factor in YaRN to 16 and fine-tune models using the next token prediction task for 500 steps on the Pile (Gao et al., 2020) dataset with a sequence length of 4096. The finetuning recipes are shown in Table 7. All models are fine-tuned on 8 NVIDIA A100 GPUs.

D.4. Long Context Benchmark

Model configurations. In this experiment, we fine-tune pretrained language models with different positional encoding methods on SCROLLS (Shaham et al., 2022). It is a long context benchmark that consists of seven distinct datasets covering different tasks, e.g, Question-Answering (Qasper (Dasigi et al., 2021), NarrativeQA (Kočiský et al., 2018), and QuALITY (Pang et al., 2022)), Natural Language Inference (ContractNLI (Koreeda & Manning, 2021)) and Summarization (QMSum (Zhong et al., 2021), SummScreenFD (Chen et al., 2022), and GovReport (Huang et al., 2021)). All the model configurations are the same as those in Table 5.

Fine-tuning recipes. We fine-tune models using the next token prediction objective on each task with a sequence length of 8192. The finetuning recipes are provided in Table 8. The model checkpoint that achieves the best performance on the validation set is selected for the final evaluation. The test results are obtained from the official SCROLLS website[2]. All models are fine-tuned on 8 NVIDIA A100 GPUs.

[2] https://www.scrolls-benchmark.com/submission