# Context-aware Biases for Length Extrapolation

Anonymous EMNLP submission

## Abstract

Transformers often struggle to generalize to longer sequences than those seen during training—a limitation known as length extrapolation. Most existing Relative Positional Encoding (RPE) methods attempt to address this by introducing either fixed linear biases or globally learned biases, which lack the capacity to adapt to different input contexts. In this work, we propose an additive RPE, **C**ontext-**A**ware **B**iases for **L**ength **E**xtrapolation (CABLE), a method that learns token-specific, context-aware biases for each attention head in transformers. By dynamically adjusting positional biases based on the input sequence, CABLE overcomes the rigidity of fixed RPEs. When evaluated on sequences longer than originally trained with, GPT-2 Medium (334M parameters) with CABLE achieves lower perplexity than counterparts using other widely adopted positional encoding methods. Additionally, by applying CABLE to the BERT base model we improved performance in long-context retrieval tasks. Our method significantly enhances the extrapolation performance of existing RPE methods tested on the FineWeb-Edu-10B and WikiText-103 datasets.

## 1 Introduction

Transformer based language models (Vaswani, 2017) have achieved state-of-the-art performance in many Natural Lnaguge Processing (NLP) tasks (Devlin et al., 2019; Liu, 2019; Chowdhery et al., 2023; Team et al., 2023; Touvron et al., 2023; Achiam et al., 2023). This is related to its attention mechanism that captures contextual information by considering inter-token interactions. However, in contrast to Convolutional Neural Networks (CNNs) (Gehring et al., 2017) and Recurrent Neural Networks (RNNs) (Sherstinsky, 2020), which implicitly consider positional information, transformers are shown to be position-agnostic and need position information (Yun et al., 2019). However,

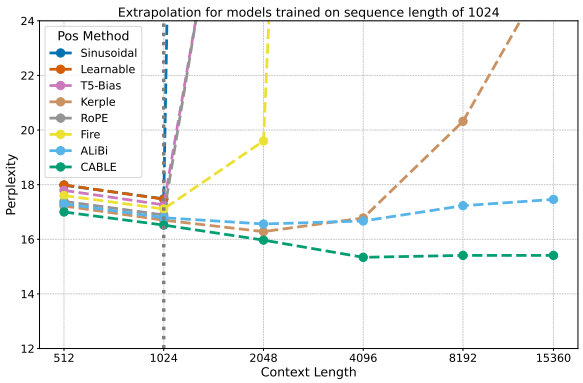


Figure 1: Next-token prediction perplexity on `FineWeb-Edu-10B` eval set with varying inference sequence lengths. The models are GPT-Medium trained on a sequence length of 1024 on `FineWeb-Edu-10B` train set.

even by incorporating positional information, transformer models often experience a sharp decline in accuracy when processing inputs longer than those seen during training (Press et al., 2021; Anil et al., 2022). This limitation arises because training is typically performed on short sequences to mitigate the quadratic cost of attention. As a result, there is increasing interest in the length extrapolation problem—namely, a model's ability to generalize to and accurately predict sequences longer than those encountered during training (Press et al., 2021).

Many commonly used positional encoding methods, such as Absolute Positional Encoding (APE) (Vaswani, 2017), fail to generalize effectively to sequence lengths beyond those seen during training (Kazemnejad et al., 2024). To address the length extrapolation challenge in transformers, various strategies have been proposed, including context window extension (Beltagy et al., 2020; Chen et al., 2023b; Peng et al., 2023; Zhu et al., 2023), memory mechanisms (Dai, 2019; Bulatov et al., 2022; Wu et al., 2022; Tworkowski et al., 2024), context compression (Mu et al., 2024; Tan et al., 2024), data formatting techniques (Shen et al., 2023; Zhou et al.,

2023), and Relative Positional Encodings (RPE) (Press et al., 2021; Raffel et al., 2020; Su et al., 2024). Among these, RPEs have emerged as one of the most prominent and widely adopted solutions for improving length extrapolation in transformer models.

Recently, a number of RPE variants have been introduced. Rotary Positional Encoding (RoPE) (Su et al., 2024) encodes token positions by rotating query and key vectors, while ALiBi (Press et al., 2021) introduces a linear bias to attention scores. Many subsequent works have built upon these foundations, either enhancing RoPE (Xu et al., 2024; Peng et al., 2023; Chen et al., 2023a) or refining ALiBi-style additive biases (Chi et al., 2022b,a; Li et al., 2023; Gao, 2024; Zhu et al., 2025).

In this work, based on ALiBi, we propose an additive RPE method which dynamically learns biases for tokens on each head of attention mechanism in transformers. In contrast to ALiBi that uses constant linear biases, our method, Context-aware biases for length extrapolation (CABLE), learns slopes for each head, enabling the model to create dynamic biases for each token. CABLE adds negligible time and memory burden to the conventional transformer (Vaswani, 2017), while achieving better performance. As shown in Figure 1, while the performance of existing positional encodings degrades with increasing sequence length, CABLE achieves even lower perplexity as sequence length increases. Our method is simple and easy to implement, and can be integrated into any existing transformer model easily.

Contributions of this paper are as follows:

- We propose CABLE, an additive relative positional encoding method that, in contrast to existing methods, uses context-aware positional information by learning token-specific biases in each attention head. CABLE is also simple, easy to implement, and have relatively fast inference time compared to the previous methods.

- We evaluate our proposed method on several benchmark datasets, using GPT-2 variants for next-token prediction and BERT models for long-context retrieval. Our approach consistently outperforms existing positional encoding methods and demonstrates superior generalization to sequences longer than those seen during training.

## 2 Related Work

The attention mechanism in Transformer models is inherently position-agnostic, necessitating the explicit injection of positional information to enable sequence order awareness. To address this, a range of positional encoding methods have been proposed. However, many of these methods face limitations when it comes to length extrapolation—the ability to generalize to sequences longer than those encountered during training (Anil et al., 2022). Several positional encoding strategies have been developed with the goal of improving extrapolation performance (Csordás et al., 2021), with relative positional encodings (RPEs) demonstrating notable effectiveness in enhancing generalization at inference time (Song and Zhong, 2023; Qin et al., 2024; Dong et al., 2024). In this section, we review key approaches to positional encoding, including absolute and relative methods, as well as recent work exploring Transformer models without any explicit positional encoding.

**No Positional Encoding (NoPE).** Surprisingly, Haviv et al. (2022) showed that decoder-only Transformers with causal attention can implicitly learn positional information without explicit encodings. Kazemnejad et al. (2024) further supported this NoPE approach, especially in out-of-distribution (OOD) settings, suggesting that the causal mechanism alone can suffice (Wang et al., 2024). However, Li et al. (2023) found that NoPE generally underperforms compared to models with explicit positional encodings. In a concurrent effort with our work, FoX (Lin et al., 2025) suggested a similar idea by not using positional encoding and instead proposed a forgetting gate. While NoPE is compatible with arbitrary sequence lengths, its performance often degrades when extrapolating far beyond training lengths.

**Absolute Positional Encoding (APE).** APE was one of the earliest approaches introduced to incorporate positional information into Transformers. Vaswani (2017) proposed both fixed (sinusoidal) and learned encodings, while Gehring et al. (2017) applied learnable absolute embeddings in convolutional architectures. Later, Devlin et al. (2019) adopted learned absolute embeddings in BERT, adding them to token embeddings. Chen et al. (2021) further refined APE with a decoupled attention mechanism to better separate content and positional signals. In general, APE assigns a fixed or learned vector $e_i \in \mathbb{R}^d$ to each position $i$, forming

a matrix $E = [e_1, e_2, ..., e_t]^T$ that is added element-wise to token embeddings (Vaswani, 2017; Devlin et al., 2019; Kiyono et al., 2021; Likhomanenko et al., 2021). A key limitation of APE methods is their poor generalization to sequence lengths beyond those seen during training, making them unsuitable for length extrapolation.

**Relative Positional Encoding (RPE).** RPE is an increasingly popular way to encode positional information for Transformers. (Shaw et al., 2018) was the first to propose learning relative positional information within a clipping distance. Among the most popular methods in RPEs, is rotary positional embedding (RoPE) (Su et al., 2024). RoPE rotates a query and key pair vectors with an angle proportional to their relative positions before the dot product attention, which results in attention being a function of the relative distance between the tokens, capturing the relative positional information. One of the primary arguments for the effectiveness of RoPE—and a key reason it is widely adopted in modern LLMs—was put forth by Su et al. (2024), who claimed that RoPE enables attention scores to decay as the relative distance between tokens increases. However, Barbero et al. (2024) later provided a mathematical analysis showing that this claim is flawed: attention weights under RoPE do not necessarily decay proportionally with relative query-key distances. This insight offers a possible explanation for RoPE's limitations in length extrapolation. In RoPE-based methods, Yarn (Peng et al., 2023) modifies RoPE by integrating attention scaling and Neural Tangent Kernel (NTK) interpolation (Jacot et al., 2018), and (Chen et al., 2023a) extends the context window size of RoPE by interpolating positions in the range seen during training. However, recent studies have shown that RoPE-based language models perform poorly on sequences longer than those seen during training (Press et al., 2021; Kazemnejad et al., 2024). To address this limitation, several positional encoding methods with better length extrapolation capabilities have been proposed (Chen et al., 2023a). Among these, additive approaches have gained popularity—where a bias matrix is directly added to the pre-softmax attention logits. This design is typically intended to enforce a decay in attention weights proportional to the relative distance between query-key pairs, as shown in the following formula:

$$A_{RPE}(X) = XW_Q(XW_K)^T + B \qquad (1)$$

The bias matrix for an input sequence with $t$ tokens is $B \in \mathbb{R}^{t\times t}$, generated by a positional encoding function $b : \mathbb{N}^2 \to \mathbb{R}$, where the $(i, j)$-th entry of $B$ is given by $b(i, j)$. Naturally, different formulations of the function $b$ lead to different variants of Relative Positional Encodings (RPEs). Below are a few examples of additive RPEs that are capable of extrapolating:

**ALiBi** (Press et al., 2021). The kernel function is defined as $b(i, j) = -r|i - j|$, where $r > 0$ is a hyperparameter. ALiBi incorporates bias based on the pairwise distances into the pre-softmax attention scores. However, the function rapidly approaches the zero point (Chi et al., 2022a), hence may not be a realistic assumption.

**T5-bias** (Raffel et al., 2020). The kernel function is defined as $b(i, j) = r_{\min\{i-j,K\}}$, where $K$ is a hyperparameter and $\{r_i\}_{i=0}^{K}$ are learnable scalars. For positions beyond the training sequence length, the model reuses the maximum learned relative bias. While this approach allows some extrapolation, it suffers from latency issues on modern accelerators due to inefficient vectorized operations with long sequences.

**Kerple** (Chi et al., 2022a). The kernel function is defined as $b(i, j) = -r_1 \log(1 + r_2|i - j|)$ in its logarithmic form and $-r_1|i - j|^{r_2}$ in its power form, where $r_1, r_2 > 0$ are learnable scalars. This approach employs a shift-invariant kernel for the bias terms.

**Fire** (Li et al., 2023). The kernel function is defined as $f_\theta\left(\frac{\psi(i-j)}{\psi(\max\{L,i\})}\right)$, where $\psi\colon x \mapsto \log(cx + 1)$ is a monotonically increasing function and $L > 0$ is a learnable scaler. This formulation allows Fire to assign more attention to distant query-key pairs—contrary to methods like ALiBi, and Kerple, which tend to focus on nearby tokens. However, as our experiments demonstrate, this behavior was not beneficial in our settings, leading to inferior performance compared to ALiBi and Kerple.

Examining existing additive RPEs, we observe that their bias functions are static—dependent only on token positions and entirely independent of the input context. Once trained, these biases remain fixed across all sequences, limiting their adaptability. This rigidity can hurt performance, especially in long-context generation. To address this, we introduce CABLE, a context-aware additive RPE designed for better length extrapolation in transformer-based language models. Unlike

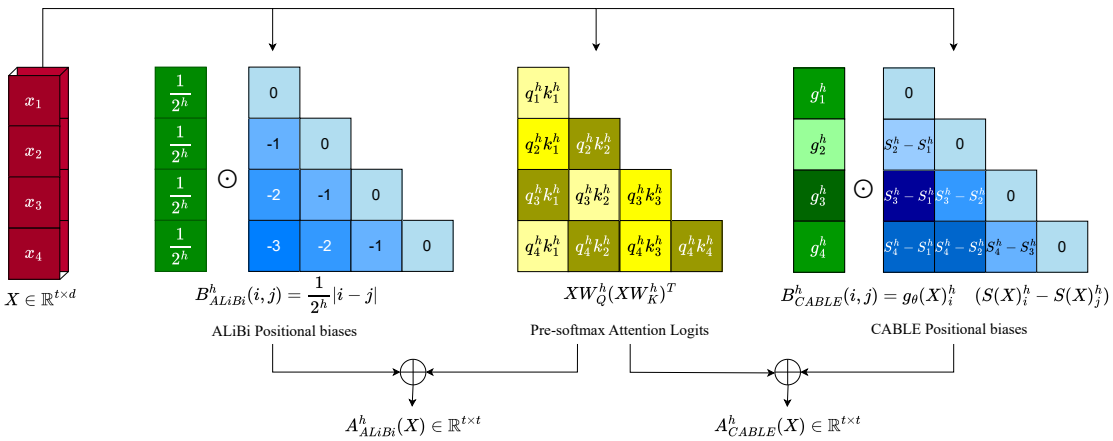


Figure 2: Comparison of how ALiBi and CABLE compute final attention scores per head. Left: ALiBi adds constant linear biases with head-specific slopes, fixed across tokens. Right: CABLE adds learned, token-specific context-aware biases and weights to the scores.

methods such as ALiBi that use fixed linear biases, CABLE learns dynamic, token-specific biases conditioned on the input, enabling more flexible and effective positional encoding.

## 3 Proposed Method

In this section, we formally introduce CABLE (**C**ontext-**A**ware **B**iases for **L**ength **E**xtrapolation), a novel additive relative positional encoding (RPE) approach designed to enhance the length generalization capabilities of Transformer models.

CABLE computes context-aware positional bias scores for each attention head and adds them to the pre-softmax attention logits. Unlike existing RPE methods, which are typically static and independent of the input sequence, our proposed biases are dynamically conditioned on the input context. Similar to the ALiBi method, we incorporate relative positional biases at the attention score level. However, CABLE introduces two key modifications: (1) the biases are learned and explicitly dependent on the input context, and (2) we learn distinct scalar weights for these biases. To implement this, we employ two separate linear layers—one to generate the context-aware biases and another to compute their associated weights.

Let $t$ and $d$ be the sequence length and the dimension of embeddings on each head, respectively. The learned bias for each token in the input sequence $X \in \mathbb{R}^{t \times d}$ is as follows:

$$f_\theta(X) = ReLU(Xw_c) \tag{2}$$

Where $f_\theta : \mathbb{R}^{t \times d} \rightarrow \mathbb{R}^t_+$, and $\theta = \{w_c \in \mathbb{R}^d\}$. Hence, we obtain context dependent biases for each token. Here ReLU ensures the biases are positive. Then, by taking a cumlative sum of these biases we inherently make these biases aware of how much positional bias should be incorporated until their position and obtain $S(X) \in \mathbb{R}^t_+$.

$$S(X) = Lf_\theta(X), \tag{3}$$

$$L_{i,j} = \begin{cases} 1 & \text{if } j \leq i \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Where $L \in R^{t \times t}$ is a lower triangular matrix of ones. Therefore, the relative biases $B(X) \in \mathbb{R}^{t \times t}$ for each pair of tokens in the input sequence is calculated by:

$$B(X)_{i,j} = S(X)_i - S(X)_j \tag{5}$$

At this stage, we have obtained context-aware biases that can be directly added to the pre-softmax attention logits. In our experiments, we refer to this version—without any additional learnable parameters for the biases—as $\text{CABLE}_{\text{NW}}$. To further enhance flexibility, we introduce a linear layer that learns a bias weight vector $g_\theta(X)$ for each token, allowing the model to modulate (dampen or amplify) the positional biases based on the input context.

$$g_\theta(X) = Softplus(Xw_s) \tag{6}$$

Where $g_\theta : \mathbb{R}^{t \times d} \rightarrow \mathbb{R}^t_+$, and $\theta = \{w_s \in \mathbb{R}^d\}$. Finally, we multiply each relative bias by its corresponding weight to produce the final CABLE bias for each attention head, as follows:

$$B(X)_{i,j} = g_\theta(X)_i(S(X)_i - S(X)_j) \tag{7}$$

| FineWeb-Edu-10B | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Sequence Length | CABLE | $\text{CABLE}_{\text{NW}}$ | ALiBi | Fire | T5-bias | Kerple | RoPE | Learnable | Sinusoidal |
| 512 | **17.00** | 17.22 | 17.30 | 17.60 | 17.79 | 17.22 | 17.39 | 17.99 | 17.98 |
| 1024 | **16.52** | 16.73 | 16.79 | 17.11 | 17.26 | 16.70 | 16.89 | 17.47 | 17.48 |
| 2048 | **15.97** | 16.24 | 16.56 | 19.60 | 38.32 | 16.28 | 38.95 | — | 219.80 |
| 4096 | **15.34** | 15.79 | 16.67 | 101.98 | 243.69 | 16.78 | 146.72 | — | 1058.84 |
| 8192 | **15.41** | 15.97 | 17.23 | 383.08 | 799.53 | 20.32 | 361.26 | — | 2485.84 |
| 15360 | **15.41** | 16.03 | 17.46 | 835.92 | 1450.83 | 26.13 | 691.90 | — | 3355.86 |

| WikiText-103 | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Sequence Length | CABLE | $\text{CABLE}_{\text{NW}}$ | ALiBi | Fire | T5-bias | Kerple | RoPE | Learnable | Sinusoidal |
| 512 | 23.70 | 24.32 | 24.09 | 24.34 | 25.06 | 23.95 | **23.66** | 24.94 | 25.18 |
| 1024 | 22.32 | 23.01 | 22.74 | 22.90 | 23.60 | 22.56 | **22.26** | 23.53 | 23.73 |
| 2048 | **21.48** | 22.19 | 22.05 | 22.68 | 27.64 | 21.72 | 41.40 | — | 172.33 |
| 4096 | **20.94** | 21.70 | 21.73 | 29.57 | 73.99 | 21.32 | 114.77 | — | 607.48 |
| 8192 | **20.65** | 21.46 | 21.58 | 54.89 | 198.64 | 21.33 | 220.56 | — | 1348.23 |
| 15360 | **20.33** | 21.13 | 21.30 | 104.79 | 411.09 | 21.58 | 375.62 | — | 2017.42 |

Table 1: Perplexity comparison on the FineWeb-Edu-10B and WikiText-103 evaluation sets. The models in the upper table are GPT-Medium variants trained on the FineWeb-Edu-10B training set for 19k steps with a sequence length of 1024. The models in the lower table are GPT-Tiny variants trained on the WikiText-103 training set for 9k steps, also with a sequence length of 1024.

CABLE exhibits an inductive bias similar to sliding window attention by penalizing distant query-key pairs—penalties that increase with positional distance. It can be seen as a generalization of ALiBi. while ALiBi applies fixed linear biases, CABLE learns context-aware biases for each token. Notably, if we set each token's bias and weight to -1 and $1/2^h$ respectively, CABLE reduces to ALiBi, with relative biases simply reflecting token distances. However, CABLE's key advantage is its ability to adapt these biases based on token context, enabling more expressive and flexible positional encoding.

As with most RPE methods, CABLE adds positional information only to the queries and keys (not the values), a practice shown to enhance length extrapolation in methods like ALiBi, T5-bias, and RoPE.

CABLE is simple, lightweight, and easily integrates into standard attention mechanisms. It requires only two additional linear layers, minimal parameters, and can be implemented in a few lines of code. The design involves two unfolding operations, a cumulative summation, and bias addition to the attention logits. Despite its simplicity, CABLE significantly improves extrapolation performance with negligible time and memory overhead compared to the vanilla transformer. Furthermore, it offers training time and memory usage on par with existing RPE methods, while maintaining low inference overhead and demonstrating notable gains in extrapolation, as shown in Section 5.

## 4 Experiment Setup

### 4.1 Datasets

For training, we use the FineWeb dataset (Penedo et al., 2024), a large-scale dataset (15 trillion tokens) for LLM pretraining, derived from 96 CommonCrawl snapshots. FineWeb has been shown to produce better-performing LLMs than other open pretraining datasets (Penedo et al., 2024). More specifically, we use a 10B sample of the FineWeb-Edu dataset, which consists of 1.3T tokens from educational web pages filtered from the FineWeb dataset. We allocate 9.9B tokens for training and 0.1B for evaluation. Furthermore, we also train the models on WikiText-103 (Merity et al., 2016), a small dataset containing a preprocessed version of Wikipedia, widely used in many NLP tasks. For evaluation, we use the test sets of FineWeb-Edu, WikiText-103, and a 1B-token sample of the FineWeb dataset.

### 4.2 Settings

For all next-token prediction tasks, we use the GPT-2 variants (Brown et al., 2020). For the FineWeb-Edu-10B dataset, we use its small version (12 layers, 10 heads, and a hidden dimension of 768) with 124M parameters, and its medium version (24 layers, 16 heads, and a hidden dimension of 1024) with 334M parameters. We also incorporate a tiny version of GPT-2 (44M parameters) with 6

layers, 8 heads, and a hidden dimension of 512 for the WikiText-103 dataset, as it is a relatively small dataset. The evaluation metric is perplexity (PPL), and we train the models with sequence length of 1024. All the models are trained on eight H100 GPUs with 80G GPU RAM. Training settings are the same as those used for GPT-2 (Radford et al., 2019). Gradients are updated after processing 524,288 tokens and vocab size is 50304. For training on the FineWeb-Edu-10B dataset, we run 19k steps (~1 epoch) with batch sizes of 64, 32, and 16 for the tiny, small, and medium models, respectively. On WikiText-103, the tiny, small, and medium variants are trained for 9k, 5k, and 3k steps(~10, 5, 3 epochs respectively). The learning rate starts at 0.0006, with a linear warmup over 750 steps, followed by cosine decay to a minimum of 0.00006.

### 4.3 Baselines

We compare our method against the following positional encoding approaches:

**Learnable** (Vaswani, 2017): A trainable APE where each position is associated with a learned embedding. The number of positions is fixed and predefined during training.

**Sinusoidal** (Vaswani, 2017): A fixed APE used in early Transformer models (Vaswani, 2017; Baevski and Auli, 2018; Ott et al., 2018; Lewis et al., 2021).

**RoPE** (Su et al., 2024): A non-learnable RPE widely adopted in LLMs such as GPT-2 (Brown et al., 2020), LLaMA (Touvron et al., 2023), PaLM (Chowdhery et al., 2023), and Gemma (Team et al., 2024a,b).

**ALiBi** (Press et al., 2021): A non-learnable RPE used in models like BLOOM (Le Scao et al., 2023) and Falcon (Almazrouei et al., 2023).

**T5-bias** (Raffel et al., 2020): A learnable RPE used in the T5 model.

**Kerple** (Chi et al., 2022a): A learnable RPE with logarithmic and power variants; we use the logarithmic variant due to its superior performance.

**Fire** (Li et al., 2023): A learnable RPE designed to give more weight to distant query-key pairs than other methods.

## 5 Results

We evaluate the performance of our method across multiple scenarios. First, we assess its extrapolation ability in decoder-only models for next-token prediction, comparing it with several existing RPEs and widely used APEs. Then, we demonstrate its effectiveness in encoder-only models on long-context retrieval tasks. We also report runtime and memory usage, followed by ablation studies to further validate the contributions of our approach.

### 5.1 Length Extrapolation

Our method demonstrates strong length extrapolation performance on the FineWeb-Edu-10B dataset, when trained with a sequence length of 1024 and evaluated on shorter and longer sequences, as shown in Figure 1. Table 1 further compares the extrapolation capabilities of CABLE against baseline methods on the test sets of FineWeb-Edu-10B and WikiText-103. [1] The sinusoidal method suffers a sharp performance drop even with slight increases in sequence length. RoPE shows a similar trend—initial improvement followed by a significant decline at longer lengths. The learnable method performs competitively at 512 and 1024 tokens but lacks extrapolation ability beyond the training length. T5-bias follows a similar trend, but its performance degrades more gradually than Sinusoidal. It initially extrapolates well to sequences slightly longer than those seen during training, thanks to its mechanism of learning relative positional information and reusing the maximum relative distance for unseen positions. ALiBi performs well on longer contexts overall but experiences slightly degradation at extreme lengths.

In contrast, our method consistently achieves lower PPL on longer sequences. Specifically, for models trained on sequence length of 1024, our method achieves lower PPL even when extrapolating to sequences 16 times longer.

Moreover, on the FineWeb-Edu-10B dataset, which contains far more tokens than WikiText-103, a model trained with ALiBi on T=1024 performs well on T=2048 but begins to degrade with longer sequences. In contrast, CABLE shows consistent improvement, even for T=15360, and achieves a better PPL than it does on T=1024, the sequence length seen during training.

Our results demonstrate that the learned biases in CABLE capture contextual information more effectively than ALiBi, leading to superior length extrapolation, especially on very long sequences. Notably, even $\text{CABLE}_{\text{NW}}$ outperforms ALiBi, high-

[1] For the longest sequences tested, we report results for 15,360 tokens instead of 16,384 due to computational constraints.

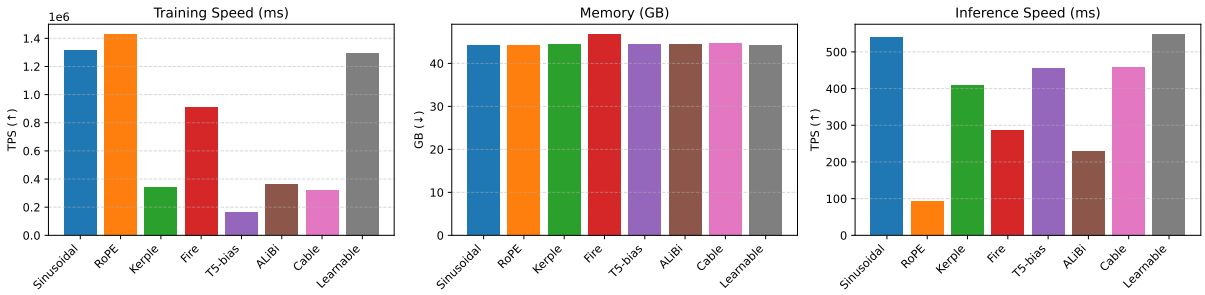


Figure 3: Comparison of batched training time, memory usage in training, and unbatched inference time among the Sinusoidal, RoPE, Kerple, Fire, T5-bias, ALiBi, CABLE, and Learnable positional encoding methods.

lighting the strength of our context-aware design. Additionally, the improved performance with the full CABLE model underscores the benefit of the learned weight function $g_\theta(X)$.

### 5.2 Bidirectional Models

In our second major experiment, we evaluate the effectiveness of our proposed additive RPE method in learning contextual representations. To do this, we replace the original fixed learnable positional encodings in BERT (Devlin et al., 2019) with CABLE during pre-training. We use the 10B-sample FineWeb-Edu dataset and train the models using only the masked language modeling (MLM) objective, following Liu (2019), who showed that removing next sentence prediction (NSP) can improve performance. Our BERT models are based on the bert-base-uncased architecture and are trained on four H100 GPUs with a batch size of 32 and a maximum sequence length of 512 for 14k steps (~1 epoch on FineWeb-Edu-10B). We use Adam (Kingma and Ba, 2014) with a learning rate of 1e-4. As shown in Figure A, CABLE achieves faster convergence in MLM loss compared to other positional encoding baselines.

Since RPE methods do not impose strict limitations on context length, BERT models trained with these encodings show improved long-context performance compared to most existing encoder-only models. However, standardized long-context benchmarks for encoder-only architectures remain limited. Following Warner et al. (2024), we evaluate long-context performance using the English subset of MLDR (Chen et al., 2024), a retrieval benchmark consisting of over 200,000 long documents.

To adapt BERT models for this task, we fine-tune them on MS-MARCO (Nguyen et al., 2016) using mined hard negatives (Xuan et al., 2020), with 1.25M samples, a batch size of 128, and a 5% learning rate warmup over one epoch, leveraging the sentence-transformers framework (Reimers and Gurevych, 2019). We then evaluate the fine-tuned models on the MLDR test set using nDCG@10 as the evaluation metric. Table 2 presents the results for several competitive positional encoding methods.[2]

At shorter sequence lengths (512 tokens), RoPE performs slightly better than other methods. However, as sequence length increases, CABLE consistently outperforms all baselines, showing notable gains beyond 1024 tokens. ALiBi maintains reasonable performance but still trails behind CABLE. In contrast, RoPE's effectiveness drops sharply after 1024 tokens, becoming nearly unusable at longer lengths. Learnable absolute encodings perform not competitively even at 512 tokens and cannot generalize to longer sequences. Sinusoidal encoding is effective at short lengths but fails completely beyond 1024 tokens. Overall, CABLE exhibits the strongest scalability to long sequences, while others either degrade significantly or become unusable.

| MLDR – nDCG@10 vs. Sequence Length | | | | | |
|---|---|---|---|---|---|
| Seq. Len. | CABLE | ALiBi | RoPE | Learnable | Sinusoidal |
| 512 | 14.96 | 14.02 | **15.16** | 10.42 | 13.57 |
| 1024 | **15.15** | 12.88 | 14.41 | — | 12.80 |
| 2048 | **16.77** | 14.30 | 10.26 | — | 1.03 |
| 4096 | **21.36** | 18.71 | 1.17 | — | 0.00 |
| 8192 | **24.59** | 22.86 | 0.12 | — | 0.00 |
| 16384 | **25.10** | 23.44 | 0.12 | — | 0.00 |

Table 2: Retrieval performance (nDCG@10) on the MLDR test set for BERT models with different positional encodings, trained at sequence length 512 and evaluated on longer inputs.

[2]We report only ALiBi among additive RPEs for BERT, as it outperformed other variants in our experiments.

### 5.3 Runtime and Memory Overhead

We also evaluate our method against existing methods in terms of training/inference runtime and memory usage. As shown in Figure 3, our method achieves the same training Token Per Second (TPS) as ALiBi and Kerple. However, both our method and ALiBi have slightly higher overhead compared to RoPE, Sinusoidal, and Learnable methods, while T5-bias exhibits significant overhead. During inference, our method achieves faster performance than other RPEs. It is the third fastest overall—trailing only Sinusoidal and Learnable encodings—while outperforming ALiBi in speed. Moreover, CABLE uses almost the same GPU memory as other methods during training and adds negligible overhead compared to methods like ALiBi. It should be noted that, due to the extrapolation ability of CABLE, it can be trained on shorter sequence lengths and effectively tested on much longer sequences. This approach addresses training overhead by reducing the sequence length during training, making it feasible on commonly available GPUs. The overhead of our method is primarily related to the cumulative sum operation in our computations. Importantly, for inference, we cache the cumulative sums, so there is no need to re-calculate them for all tokens each time. This optimization helps CABLE achieving superior inference time to other methods, such as ALiBi.

Note that training time is measured with batching, while inference is measured without batching by recording the time to sequentially generate a set number of tokens.

### 5.4 Ablation Study: Kernelized CABLE

For models with a low number of layers, we tested a kernelized version of our method (K-CABLE). In this setting, we use $-\log(b^2+1)$ as a kernel, which is applied to the relative biases before adding them to the attention scores. Figure 4 shows the extrapolation comparison between CABLE and K-CABLE. As can be seen, K-CABLE achieves better PPL when trained on sequence length of 1024, compared to original CABLE, demonstrating improved extrapolation. This improvement is related to the sliding window nature of additive RPEs, where, with a low number of layers, they struggle to propagate information across longer sequences. In contrast, K-CABLE has a slower slope for biases and behaves less like a sliding window, making it more suitable for networks with fewer layers. More generally, different types of kernels can be applied to CABLE based on the network architecture, allowing it to achieve optimal performance.

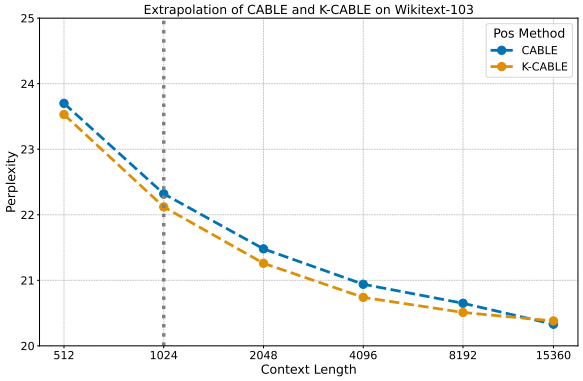


Figure 4: Extrapolation performance of CABLE vs. its kernelized variant (K-CABLE). Both models are GPT-Tiny architectures trained on the WikiText-103 training set with a sequence length of 1024.

## 6 Conclusion

We introduced CABLE, a novel relative positional encoding method that learns context-aware biases for each token. This is done by adding learned biases into the attention matrix at each layer of decoder-based transformers. Unlike existing methods such as ALiBi, which rely on constant linear biases, CABLE adapts biases based on the tokens' roles within the sequence. Our experiments demonstrated that CABLE achieves lower perplexity compared to existing methods and significantly enhances length extrapolation capabilities. Results on the edu-fineweb10B and wikitext-103 datasets revealed that CABLE effectively extrapolates on sequences longer than those seen during training, consistently outperforming other approaches. Ablation studies further validated CABLE's effectiveness, showing that a model trained with CABLE on sequences of length 1024 can outperform a sinusoidal method trained and tested on sequences of length 8192. These advancements were achieved with only a minimal increase in training time and memory usage, while the increase in inference time was only negligible.

## Limitation

While CABLE demonstrates strong performance in length extrapolation, it has several limitations. First, it incurs higher training time compared to RoPE due to its dynamic bias computation, though this overhead is negligible in inference. Second,

CABLE occasionally underperforms RoPE at base sequence lengths (e.g., 1024 tokens in our experiments), particularly in tasks where fixed positional patterns suffice, suggesting a trade-off between adaptability and consistency for shorter contexts. Additionally, the method's computational overhead, though minimal, may become more pronounced for extremely long sequences (>100K tokens), and its extrapolation capabilities remain dependent on the diversity of positional patterns in training data. While empirical results are promising, theoretical analysis of its attention dynamics at arbitrary lengths remains an open question. Future work could explore optimizations for training efficiency and head-specific bias adaptation to further enhance flexibility.

## References


Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. 2023. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. 2022. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556.

Alexei Baevski and Michael Auli. 2018. Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*.

Federico Barbero, Alex Vitvitskyi, Christos Perivolaropoulos, Razvan Pascanu, and Petar Veličković. 2024. Round and round we go! what makes rotary positional encodings useful? *arXiv preprint arXiv:2410.06205*.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. 2022. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Pu-Chin Chen, Henry Tsai, Srinadh Bhojanapalli, Hyung Won Chung, Yin-Wen Chang, and Chun-Sung Ferng. 2021. A simple and effective positional encoding for transformers. *arXiv preprint arXiv:2104.08698*.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023a. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023b. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*.

Ta-Chung Chi, Ting-Han Fan, Peter J Ramadge, and Alexander Rudnicky. 2022a. Kerple: Kernelized relative positional embedding for length extrapolation. *Advances in Neural Information Processing Systems*, 35:8386–8399.

Ta-Chung Chi, Ting-Han Fan, Alexander I Rudnicky, and Peter J Ramadge. 2022b. Dissecting transformer length extrapolation via the lens of receptive field analysis. *arXiv preprint arXiv:2212.10356*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. 2021. The devil is in the detail: Simple tricks improve systematic generalization of transformers. *arXiv preprint arXiv:2108.12284*.

Zihang Dai. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186.

Zican Dong, Junyi Li, Xin Men, Wayne Xin Zhao, Bingbing Wang, Zhen Tian, Weipeng Chen, and Ji-Rong Wen. 2024. Exploring context window of large language models via decomposed positional vectors. *arXiv preprint arXiv:2405.18009*.

Weiguo Gao. 2024. Mep: Multiple kernel learning enhancing relative positional encoding length extrapolation. *arXiv preprint arXiv:2403.17698*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International conference on machine learning*, pages 1243–1252. PMLR.

Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. 2022. Transformer language models without positional encodings still learn positional information. *arXiv preprint arXiv:2203.16634*.

Arthur Jacot, Franck Gabriel, and Clément Hongler. 2018. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. 2024. The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, 36.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Shun Kiyono, Sosuke Kobayashi, Jun Suzuki, and Kentaro Inui. 2021. Shape: Shifted absolute position embedding for transformers. *arXiv preprint arXiv:2109.05644*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pages 6265–6274. PMLR.

Shanda Li, Chong You, Guru Guruganesh, Joshua Ainslie, Santiago Ontanon, Manzil Zaheer, Sumit Sanghai, Yiming Yang, Sanjiv Kumar, and Srinadh Bhojanapalli. 2023. Functional interpolation for relative positions improves long context transformers. *arXiv preprint arXiv:2310.04418*.

Tatiana Likhomanenko, Qiantong Xu, Gabriel Synnaeve, Ronan Collobert, and Alex Rogozhnikov. 2021. Cape: Encoding relative positions with continuous augmented positional embeddings. *Advances in Neural Information Processing Systems*, 34:16079–16092.

Zhixuan Lin, Evgenii Nikishin, Xu He, and Aaron Courville. 2025. Forgetting transformer: Softmax attention with a forget gate. In *The Thirteenth International Conference on Learning Representations*.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364.

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

Jesse Mu, Xiang Li, and Noah Goodman. 2024. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human-generated machine reading comprehension dataset.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. *arXiv preprint arXiv:1806.00187*.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*.

Ofir Press, Noah A Smith, and Mike Lewis. 2021. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*.

Zhen Qin, Yiran Zhong, and Hui Deng. 2024. Exploring transformer extrapolation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18897–18905.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

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

Ruoqi Shen, Sébastien Bubeck, Ronen Eldan, Yin Tat Lee, Yuanzhi Li, and Yi Zhang. 2023. Positional description matters for transformers arithmetic. *arXiv preprint arXiv:2311.14737*.

Alex Sherstinsky. 2020. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306.

Jiajun Song and Yiqiao Zhong. 2023. Uncovering hidden geometry in transformers via disentangling position and context. *arXiv preprint arXiv:2310.04861*.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Sijun Tan, Xiuyu Li, Shishir Patil, Ziyang Wu, Tianjun Zhang, Kurt Keutzer, Joseph E Gonzalez, and Raluca Ada Popa. 2024. Lloco: Learning long contexts offline. *arXiv preprint arXiv:2404.07979*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024a. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024b. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. 2024. Focused transformer: Contrastive training for context scaling. *Advances in Neural Information Processing Systems*, 36.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Jie Wang, Tao Ji, Yuanbin Wu, Hang Yan, Tao Gui, Qi Zhang, Xuanjing Huang, and Xiaoling Wang. 2024. Length generalization of causal transformers without position encoding. *arXiv preprint arXiv:2404.12224*.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, et al. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *arXiv preprint arXiv:2412.13663*.

Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. 2022. Memorizing transformers. *arXiv preprint arXiv:2203.08913*.

Mingyu Xu, Xin Men, Bingning Wang, Qingyu Zhang, Hongyu Lin, Xianpei Han, et al. 2024. Base of rope bounds context length. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Hong Xuan, Abby Stylianou, Xiaotong Liu, and Robert Pless. 2020. Hard negative examples are hard, but useful. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 126–142. Springer.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. 2019. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. 2023. What algorithms can transformers learn? a study in length generalization. *arXiv preprint arXiv:2310.16028*.

Dawei Zhu, Nan Yang, Liang Wang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. 2023. Pose: Efficient context window extension of llms via positional skip-wise training. *arXiv preprint arXiv:2309.10400*.

Jiajun Zhu, Peihao Wang, Ruisi Cai, Jason D Lee, Pan Li, and Zhangyang Wang. 2025. Rethinking addressing in language models via contexualized equivariant positional encoding. *arXiv preprint arXiv:2501.00712*.

# A Appendix

## A.1 Bert Models Training

Figure 5 shows the masked language modeling loss during BERT pre-training with different positional encodings. Traditional methods like learnable and sinusoidal fail to match the loss achieved by RPEs, highlighting the effectiveness of RPEs. CABLE also converges faster than other methods.

Figure 6 shows the contrastive loss during fine-tuning BERT models with different positional encoding methods on the MS-MARCO training set. Once again, learnable and sinusoidal methods lag behind, while CABLE achieves the lowest loss among all methods.

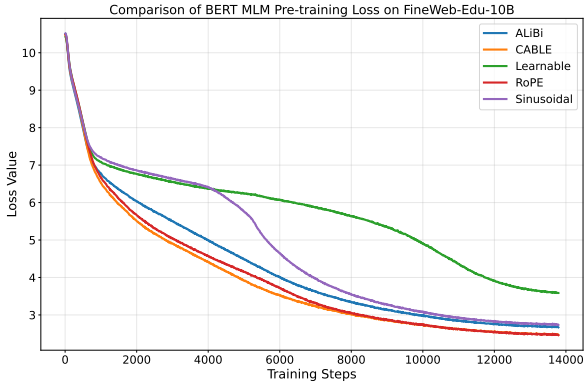


Figure 5: Masked language modeling (MLM) loss during BERT pre-training on FineWeb-Edu-10B with different positional encoding methods. CABLE achieves the fastest convergence and lowest final loss, demonstrating superior training efficiency over traditional and other RPE methods.

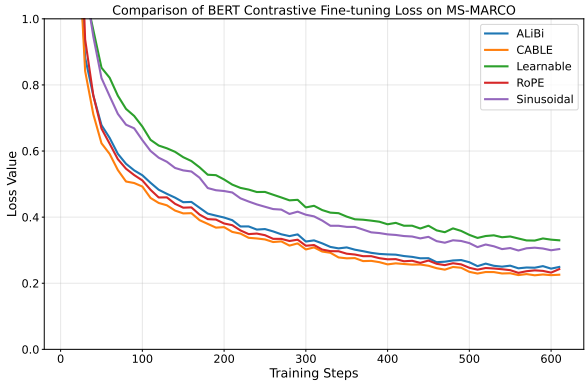


Figure 6: Contrastive fine-tuning loss on the MS-MARCO dataset for BERT models using different positional encoding methods. CABLE achieves the lowest loss, while learnable and sinusoidal encodings underperform.

## A.2 Results for GPT-Small and Tiny

Tables 3 and 4 present the extrapolation results for GPT-Small and GPT-Tiny models evaluated on the FineWeb-Edu-10B dataset. As shown in Table 3, CABLE consistently outperforms other positional encoding methods across all sequence lengths, particularly at extrapolated lengths beyond 1024 tokens. Both GPT-Small and GPT-Tiny models trained with CABLE achieve significantly lower perplexity than those using ALiBi, RoPE, T5-bias, and other baselines. Notably, standard methods such as sinusoidal or learnable encodings degrade sharply at longer lengths, whereas CABLE maintains stable and superior performance. These results further confirm CABLE's effectiveness in enhancing length extrapolation, even in smaller model regimes.

| GPT-Small | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Sequence Length | CABLE | $\text{CABLE}_{\text{NW}}$ | ALiBi | Fire | T5-bias | Kerple | RoPE | Learnable | Sinusoidal |
| 512 | **21.19** | 21.42 | 21.55 | 21.84 | 22.17 | 21.46 | 21.43 | 22.16 | 22.38 |
| 1024 | **20.63** | 20.89 | 20.99 | 21.26 | 21.57 | 20.86 | 20.87 | 21.56 | 21.83 |
| 2048 | **20.02** | 20.34 | 20.67 | 22.48 | 29.37 | 20.38 | 58.59 | - | 207.53 |
| 4096 | **19.24** | 19.67 | 21.23 | 53.25 | 131.36 | 21.11 | 225.78 | - | 956.41 |
| 8192 | **19.31** | 19.81 | 22.42 | 155.32 | 405.94 | 26.59 | 554.12 | - | 2376.51 |
| 15360 | **19.28** | 19.82 | 22.89 | 333.91 | 757.36 | 34.91 | 957.87 | - | 3589.97 |

| GPT-Tiny | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Sequence Length | CABLE | $\text{CABLE}_{\text{NW}}$ | ALiBi | Fire | T5-bias | Kerple | RoPE | Learnable | Sinusoidal |
| 512 | **29.37** | 30.12 | 29.88 | 30.23 | 30.78 | 29.60 | 29.44 | 30.73 | 30.67 |
| 1024 | **28.73** | 29.57 | 29.25 | 29.56 | 30.08 | 28.95 | 28.81 | 30.11 | 30.03 |
| 2048 | **27.96** | 28.88 | 28.82 | 29.60 | 33.81 | 28.32 | 76.29 | — | 275.28 |
| 4096 | **26.90** | 27.85 | 28.28 | 37.86 | 86.33 | 28.31 | 239.95 | — | 1166.46 |
| 8192 | **26.97** | 27.92 | 26.80 | 70.72 | 222.60 | 32.00 | 452.67 | — | 2561.54 |
| 15360 | **26.80** | 27.75 | 28.52 | 124.29 | 448.08 | 37.67 | 652.52 | — | 3679.78 |

Table 3: Perplexity comparison on the FineWeb-Edu-10B evaluation sets. The upper table shows GPT-Small variants, and the lower table shows GPT-Tiny variants—both trained on the FineWeb-Edu-10B training set for 19k steps with a sequence length of 1024.

| GPT-Medium | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Sequence Length | CABLE | $\text{CABLE}_{\text{NW}}$ | ALiBi | Fire | T5-bias | Kerple | RoPE | Learnable | Sinusoidal |
| 512 | **20.33** | 20.80 | 20.80 | 22.18 | 23.33 | 20.96 | 20.81 | 22.52 | 24.16 |
| 1024 | **19.12** | 19.63 | 19.62 | 20.89 | 22.00 | 19.73 | 19.60 | 21.23 | 22.77 |
| 2048 | **18.36** | 18.91 | 19.04 | 21.86 | 35.46 | 19.01 | 20.78 | — | 143.58 |
| 4096 | **17.87** | 18.47 | 18.91 | 46.60 | 124.19 | 18.63 | 31.22 | — | 467.51 |
| 8192 | **17.58** | 18.23 | 18.89 | 106.27 | 363.59 | 18.61 | 51.54 | — | 1006.37 |
| 15360 | **17.36** | 17.95 | 18.60 | 195.93 | 726.18 | 18.79 | 91.53 | — | 1516.97 |

| GPT-Small | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Sequence Length | CABLE | $\text{CABLE}_{\text{NW}}$ | ALiBi | Fire | T5-bias | Kerple | RoPE | Learnable | Sinusoidal |
| 512 | **20.93** | 21.34 | 21.46 | 22.03 | 22.80 | 21.50 | 21.38 | 22.51 | 23.09 |
| 1024 | **19.71** | 20.15 | 20.22 | 20.74 | 21.49 | 20.22 | 20.13 | 21.21 | 21.73 |
| 2048 | **18.95** | 19.42 | 19.62 | 21.28 | 33.85 | 19.45 | 30.14 | — | 163.49 |
| 4096 | **18.47** | 18.98 | 19.39 | 36.02 | 123.05 | 19.04 | 63.75 | — | 580.86 |
| 8192 | **18.20** | 18.75 | 19.29 | 81.77 | 347.22 | 18.91 | 117.81 | — | 1121.08 |
| 15360 | **17.92** | 18.48 | 19.06 | 163.91 | 659.72 | 19.00 | 202.23 | — | 1652.53 |

Table 4: Perplexity comparison on the WikiText-103 evaluation set. The upper table shows GPT-Medium variants trained for 3k steps, and the lower table shows GPT-Small variants trained for 5k steps. All models use a sequence length of 1024.