KERPLE: Kernelized Relative Positional Embedding for Length Extrapolation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Relative positional embeddings (RPE) have received considerable attention since RPEs effectively model the relative distance among tokens and enable length extrapolation. We propose KERPLE, a framework that generalizes relative position embedding for extrapolation by kernelizing positional differences. We achieve this goal using conditionally positive definite (CPD) kernels, a class of functions known for generalizing distance metrics. To maintain the inner product interpretation of self-attention, we show that a CPD kernel can be transformed into a PD kernel by adding a constant offset. This offset is implicitly absorbed in the Softmax normalization during self-attention. The diversity of CPD kernels allows us to derive various RPEs that enable length extrapolation in a principled way. Experiments demonstrate that the logarithmic variant achieves excellent extrapolation performance on three large language modeling datasets.

1 Introduction

Transformer-based models have excelled in various natural language processing tasks such as chatbot [Roller et al., 2021], code completion [Chen et al., 2021a], and paper abstract summarization [Zhang et al., 2020]. These sequence modeling tasks often require the model to operate well on significantly longer text sequences than the fixed maximum length L used at training time. Training (or retraining) the model using a substantially larger value of L is often infeasible since the transformer training cost is $O(L^2)$. Hence, one desires a transformer that continues to perform well on longer sequences than those used during training; i.e., perform length extrapolation at inference time. Most transformer designs do not have this property [Press et al., 2022]. While recent work on absolute positional embeddings demonstrated the extrapolation ability [Kiyono et al., 2021, Likhomanenko et al., 2021], it is believed that relative positional embeddings are more robust to input length change [Likhomanenko et al., 2021], for example, ALiBi [Press et al., 2022] and T5 [Raffel et al., 2020]. Hence, we are motivated to study the inner workings of relative positional embeddings.

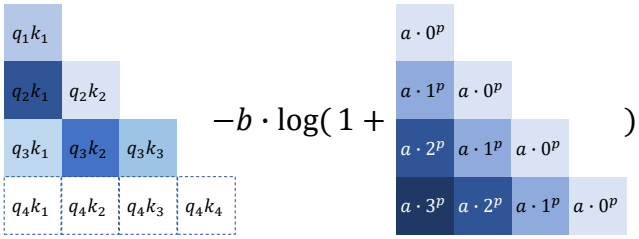
Relative positional embeddings (RPE) encode the idea of shift-invariance: for any shift p, $(m+p)-(n+p)=m-n$. It is often added directly to the self-attention matrix before Softmax normalization [Chen et al., 2021b]. Inspired by shift-invariance and the ability of a kernel to define a similarity function, there have been studies on shift-invariant kernels for RPE [Wennberg and Henter, 2021] with a focus on Gaussian kernel. However, in our preliminary experiments, the Gaussian kernel demonstrates limited length extrapolation ability (see Appendix A.3). Hence, a distinct class of shift-invariant kernels is needed to achieve adequate length extrapolation.

To this end, we note a set of well-established conditionally positive definite (CPD) kernels suitable for modeling distance metrics [Schölkopf, 2000]. However, CPD kernels do not conform to an inner

[†] These authors contributed equally to this work.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Figure 1: **The 3-Para-Log Variant of Our KERPLE Framework.** a, b, and p are learnable parameters in each attention head shared across layers. Since # of heads is H, there are $3 \cdot H$ learnable parameters. The learnable parameters are trained with length-3 sequences. At the inference time, the last row (in dashed squares) becomes active, and the model extrapolates to length-4 sequences. Note we focus on causal language modeling following ALiBi, so the matrices are triangular.



product. We can remedy this issue by transforming a CPD kernel into a PD kernel by adding a sufficiently large constant. This constant offset is subsequently absorbed implicitly in the Softmax normalization (see the discussion below Eq. (2)). For example, ALiBi implicitly admits a PD kernel of the form $c - |m - n|$ (see the end of section 4), which is reduced to a CPD kernel $-|m - n|$. The CPD kernel and Softmax normalization combination opens the door to a sea of possible CPD kernels. We investigate structures from this class that exhibit a strong length extrapolation ability, like ALiBi.

Our main result is a framework for **KE**rnelize **R**elative **P**ositional Embedding for **L**ength **E**xtrapolation (**KERPLE**). The framework elucidates key principles that encourage the length extrapolation property. We show that ALiBi is a particular instance within our framework. Our subsequent experiments suggest that the proposed method yields better length extrapolation on large datasets such as OpenWebText2, GitHub, and ArXiv.

2 Background and Related Work

2.1 Preliminary

Let $\{w_m\}_{m=1}^L$ be the input tokens to a transformer model, where L is the total number of tokens. Each w_m is a scalar and is used to index the embedding vector $\boldsymbol{e}_m \in \mathbb{R}^d$ as the input to the transformer. A transformer converts each $\boldsymbol{e}_m$ into query, key, and value vectors in $\mathbb{R}^d$: $\boldsymbol{q}_m = \boldsymbol{W}_q \boldsymbol{e}_m$, $\boldsymbol{k}_m = \boldsymbol{W}_k \boldsymbol{e}_m$, $\boldsymbol{v}_m = \boldsymbol{W}_v \boldsymbol{e}_m$, where $\boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v \in \mathbb{R}^{d \times d}$ are learnable matrices. Then, the self-attention module computes the scaled attention scores and generates the output vector $\boldsymbol{o}_m$ at position m as:

$$a_{m,n} = \frac{\exp(\boldsymbol{q}_m^\top \boldsymbol{k}_n / \sqrt{d})}{\sum_{i=1}^L \exp(\boldsymbol{q}_m^\top \boldsymbol{k}_i / \sqrt{d})}, \quad \boldsymbol{o}_m = \sum_{n=1}^L a_{m,n} \boldsymbol{v}_n.$$

Since the operation is position-agnostic, it is believed that positional information helps the token interactions modeling [Vaswani et al., 2017], which we survey in the next subsection.

2.2 Positional Embedding

Absolute. Absolute positional embeddings assign a positional vector $\boldsymbol{p}_m$ to each position m and adds $\boldsymbol{p}_m$ to the embedding vector $\boldsymbol{e}_m$. The very first version of which is the predefined sinusoidal function [Vaswani et al., 2017]. Followed by the success of BERT [Devlin et al., 2019], learnable absolute positional embeddings have been applied on the task of masked language modeling [Devlin et al., 2019, Liu et al., 2019, Clark et al., 2020, Lan et al., 2020], Autoregressive-decoding [Radford et al., 2018, 2019], and sequence-to-sequence [Gehring et al., 2017, Lewis et al., 2019] settings. Recent work studied ways to extrapolate sinusoidal positional embeddings to longer sequences by randomly shifting absolute positions during training [Kiyono et al., 2021] or augmenting with continuous signals [Likhomanenko et al., 2021].

Relative. As opposed to the modeling of absolute position m, relative positional embeddings (RPE) that model the positional difference $m - n$ has become popular in the literature [Shaw et al., 2018, Huang et al., 2019, Dai et al., 2019, Yang et al., 2019, Huang et al., 2020, He et al., 2021, Ke et al.,

2021, Chen et al., 2021b]. In particular, the T5 model that considers bucketed relative distances and log-binning has been shown to perform well on various transformer architectures [Raffel et al., 2020]. Rotary positional embedding [Su et al., 2021] encodes the position with rotations: $f(\boldsymbol{q}_m, m) = R_m \boldsymbol{q}_m$ where R_m is a rotation matrix with angles proportional to m. With the rotation's property, the query-key product exhibits a positional difference: $f(\boldsymbol{q}_m, m)^\top f(\boldsymbol{k}_n, n) = \boldsymbol{q}_m^\top R_{n-m} \boldsymbol{k}_n$.

We note that the overview above focuses on the NLP domain. Recent work has applied positional embeddings to other domains such as vision [Wu et al., 2021a] and speech [Likhomanenko et al., 2021]. A survey can be found in [Dufter et al., 2022].

2.3 Kernel and its Application in Transformer

The kernel trick is a classic approach to generalize the inner product to high dimensional spaces [Mika et al., 1998, Schölkopf, 2000, Leslie et al., 2001, Dhillon et al., 2004, Takeda et al., 2007]. In the context of transformers, there have been interests in applying kernels to the self-attention structure to enhance the performance. Examples of such work include kernel for positional embeddings [Tsai et al., 2019, Wu et al., 2021b, Wennberg and Henter, 2021, Luo et al., 2021]. Another line of research leverages the kernel's feature map [Rahimi and Recht, 2007] to linearize the self-attention module and reduce the computational cost [Katharopoulos et al., 2020, Chen et al., 2021c, Xiong et al., 2021, Peng et al., 2021, Choromanski et al., 2021, Qin et al., 2022].

3 Theoretical Foundations of CPD Kernels

3.1 PD and CPD Kernels

In this work, we use shift-invariant conditionally positive definite (CPD) kernels to model the effect of relative positional differences. We propose this formulation because the notion of *relative* is modeled by a shift-invariant function: a bivariate function k over two positions (m, n) such that $k(m, n) = f(m - n)$ for some univariate f. The notion of *positional difference* $m - n$ is generalized by the CPD kernel. To elaborate, we review the definitions of PD and CPD kernels as below.

Definition 1 (PD Kernel)**.** *A (real) symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite kernel if for any integer N and any $\{x_i \in \mathcal{X}\}_{i=1}^N$, $\{c_i \in \mathbb{R}\}_{i=1}^N$, the quadratic form is nonnegative: $\sum_{i=1}^N \sum_{j=1}^N c_i c_j k(x_i, x_j) \geq 0$.*

Definition 2 (CPD Kernel)**.** *A (real) symmetric function $\tilde{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a conditionally positive definite kernel if for any integer N and any $\{x_i \in \mathcal{X}\}_{i=1}^N$, the quadratic form is conditionally nonnegative: $\sum_{i=1}^N \sum_{j=1}^N c_i c_j \tilde{k}(x_i, x_j) \geq 0$ for $\{c_i \in \mathbb{R}\}_{i=1}^N$ with $\sum_{i=1}^N c_i = 0$.*

Fact 1 (Berg et al. [1984] and Prop. 5 of Schölkopf [2000])**.** *Let $\tilde{k} : \mathcal{X} \times \mathcal{X} \to (-\infty, 0]$ be a CPD kernel with $\tilde{k}(x, x) = 0\ \forall x \in \mathcal{X}$. Then, there exists a Hilbert space $\mathcal{H}$ and a mapping $\phi : \mathcal{X} \to \mathcal{H}$ such that $\|\phi(x) - \phi(x')\|^2 = -\tilde{k}(x, x')$.*

Fact 1 suggests that CPD kernels generalize the distance metric to high dimensional spaces. Since we are interested in the positional difference, it is of our interest to model the distance between positions using CPD kernels.

However, Fact 1 also implies that CPD kernels do not encode inner products as required by self-attention for pairwise relations computation. It is PD kernels that represent inner products. To better understand the effect of CPD kernels on self-attention, we need to establish relations between CPD and PD kernels. As noted in Schölkopf [2000], if one takes any PD kernel and offsets it by a constant, the result is at least a CPD kernel. In the next subsection, we show that the converse is *nearly* true: if $\tilde{k}$ is CPD, so is $c + \tilde{k}$ for large enough $c \in \mathbb{R}$ (Lemma 1). Therefore, we may generate the CPD kernels of our interest and transform them into PD kernels if needed.

3.2 Constructing PD Kernels From CPD Kernels via Constant Shifts

In this subsection, we review a few properties of CPD kernels and use these to generate a variety of CPD kernels. Then, we present a lemma that transforms CPD kernels into PD kernels via constant shifts. This enables the production of a family of PD kernels from CPD kernels. Finally, we present

our critical observation that the exact value of the constant shift is not needed, thanks to a nice property of Softmax normalization.

Below are some important facts about CPD kernels.

Fact 2 (Scaling and Summation)**.** *If $\tilde{k}_1$ and $\tilde{k}_2$ are CPD, then so are $a \cdot \tilde{k}_1$ (for $a > 0$) and $\tilde{k}_1 + \tilde{k}_2$.*

Fact 3 (Berg et al. [1984] and Prop. 4 of Schölkopf [2000])**.** *If $\tilde{k} : \mathcal{X} \times \mathcal{X} \to (-\infty, 0]$ is CPD, then so are $-(-\tilde{k})^\alpha$ for $0 < \alpha < 1$ and $-\log(1 - \tilde{k})$.*

Fact 4 (Page 3 of Schölkopf [2000])**.** *The negative squared distance $-\|x - x'\|^2$ is CPD.*

The three Facts above jointly yield a rich family of CPD kernels as shown below.

Corollary 1. *The following are CPD kernels.*

(a) $\tilde{k}(x, x') = -a\|x - x'\|^p$ *with* $0 < p \leq 2$ *and* $a > 0$.

(b) $\tilde{k}(x, x') = -b \cdot \log(1 + a\|x - x'\|^p)$ *with* $0 < p \leq 2$ *and* $a, b > 0$.

We note that it is possible to keep iterating between Fact 2 and 3 and generate more complicated ones, for example, $-a\|x - x'\|^p - b \cdot \log(1 + a\|x - x'\|^p)$ or $-b \cdot \log(1 + a\|x - x'\|^p)^c$ for $0 < c < 1$. Considering kernel is a means to model relative positional embeddings, we only present CPD kernels with reasonable complication. Those with complicated forms can be deferred to our future work.

Now that Corollary 1 has presented a few class of CPD kernels, we prove a lemma (in Appendix A.1) that constructs PD kernels from CPD kernels through shifting. Later in Eq. (2), we will see that the shifting construction is combined neatly with the Softmax normalization of self-attention.

Lemma 1 (CPD Shift Lemma)**.** *Let $\tilde{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a CPD kernel. There exists $c \geq 0$ such that $c + \tilde{k}$ is a PD kernel.*

Lemma 1 implies the CPD kernels in Corollary 1 can be made PD if they are added with large enough constants; for example, $c - \|x - x'\|^p$ for large enough c. Although Lemma 1 does not have an explicit construction of c, thanks to the shift-invariant property of the Softmax normalization, we can leave it as an under-determined constant in our positional embedding design, which is Eq. (1) in section 4. Still, given a set of test points $\{x_i\}_{i=1}^N$, one can do a geometric sequence search[1] to search for a c such that the $N \times N$ matrix $[c + \tilde{k}(x_i, x_j)]_{i,j=1}^N \succeq 0$. Hence, in this work, we do not need the value of c , but we can compute it if we do need its value, e.g., deriving the feature map of $c + \tilde{k}$.

Alternative Proof of $c - \|x - x'\|^p$. While the CPD shift lemma is convenient, one can prove $c - \|x - x'\|^p$ is PD for large enough c using a kernel representation theorem in Schoenberg [1938]. See Appendix A.2 for details.

4 Kernelized Relative Positional Embedding

Let $\{\boldsymbol{q}_m\}_{m=1}^L$ and $\{\boldsymbol{k}_n\}_{n=1}^L$ be the input queries and keys. Let $(r_1, ..., r_\ell)$ be learnable parameters. We propose a kernelized relative positional embedding as follows.

$$a_{m,n} = \frac{\exp\left((\boldsymbol{q}_m^\top \boldsymbol{k}_n + \tilde{k}_{r_1,...,r_\ell}(m, n))/\sqrt{d}\right)}{\sum_{i=1}^L \exp((\boldsymbol{q}_m^\top \boldsymbol{k}_i + \tilde{k}_{r_1,...,r_\ell}(m, i))/\sqrt{d})}, \tag{1}$$

where $\tilde{k}_{r_1,...,r_\ell}(m, n)$ is any shift-invariant CPD kernel with ℓ number of parameters. Due to Lemma 1, the above can be reformulated into its kernel form as follows.

$$\begin{aligned}\text{Eq. (1)} &\overset{(*)}{=} \frac{\exp\left((\boldsymbol{q}_m^\top \boldsymbol{k}_n + c + \tilde{k}_{r_1,...,r_\ell}(m, n))/\sqrt{d}\right)}{\sum_{i=1}^L \exp((\boldsymbol{q}_m^\top \boldsymbol{k}_i + c + \tilde{k}_{r_1,...,r_\ell}(m, i))/\sqrt{d})} \\ &\overset{\text{Lemma 1}}{=} \frac{\exp\left(\boldsymbol{q}_m^\top \boldsymbol{k}_n + k_{r_1,...,r_\ell}(m, n))/\sqrt{d}\right)}{\sum_{i=1}^L \exp(\boldsymbol{q}_m^\top \boldsymbol{k}_i + k_{r_1,...,r_\ell}(m, i))/\sqrt{d})} = \frac{\exp\left(k^{\text{comp}}([\boldsymbol{q}_m, m], [\boldsymbol{k}_n, n])/\sqrt{d}\right)}{\sum_{i=1}^L \exp\left(k^{\text{comp}}([\boldsymbol{q}_m, m], [\boldsymbol{k}_i, i])/\sqrt{d}\right)}.\end{aligned} \tag{2}$$

[1] By geometric sequence search, we can enlarge c by 2, 4, 8, 16, and so on until we find the required large enough constant.

(*) is due to the shift-invariant property of the Softmax normalization: $\frac{\exp(x_i)}{\sum_j \exp(x_j)} = \frac{\exp(x_i+c)}{\sum_j \exp(x_j+c)}$ for any $c \in \mathbb{R}$. The second equality defines a *bias kernel* which is positive definite using Lemma 1:

$$k_{r_1,\ldots,r_\ell} = c + \tilde{k}_{r_1,\ldots,r_\ell}. \tag{3}$$

The last equality introduces a *composite kernel* $k^{\text{comp}} : \mathbb{R}^{d+1} \times \mathbb{R}^{d+1} \to \mathbb{R}$ as

$$k^{\text{comp}}([\boldsymbol{q}_m, m], [\boldsymbol{k}_n, n]) = \boldsymbol{q}_m^\top \boldsymbol{k}_n + k_{r_1,\ldots,r_\ell}(m, n). \tag{4}$$

Interpretation. The proposed method can be interpreted as applying a composite kernel to self-attention. The composite kernel combines the information from query $\boldsymbol{q}_m$, key $\boldsymbol{k}_n$, and positions (m, n) in a way that augments the original self-attention structure by multiplicative and additive position embeddings. The augmentation allows k^{comp} to not only retain the original $\boldsymbol{q}_m^\top \boldsymbol{k}_n$ but also include positional information from the bias kernel $k_{r_1,\ldots,r_\ell}$.

Practical Choice. In section 5.2, we fix $\ell = 2$ and experiment on two variants of the composite kernel, Eq. (4), where we call these the *power* variant and the *logarithmic* variant of our proposed KERPLE framework, Eq. (2). These are from a combination of Corollary 1 and Eq. (3).

$$\text{(power)} \quad k^{\text{comp}}([\boldsymbol{q}_m, m], [\boldsymbol{k}_n, n]) = \boldsymbol{q}_m^\top \boldsymbol{k}_n + c - r_1|m-n|^{r_2} \text{ with } r_1 > 0 \text{ and } 0 < r_2 \leq 2.$$

$$\text{(logarithmic)} \quad k^{\text{comp}}([\boldsymbol{q}_m, m], [\boldsymbol{k}_n, n]) = \boldsymbol{q}_m^\top \boldsymbol{k}_n + c - r_1 \cdot \log(1 + r_2|m-n|) \text{ with } r_1, r_2 > 0.$$

We note that these are not the only variants of the composite kernel. In section 5.3, we experiment with two more complicated variants, but only find lower training speeds and marginal improvement in perplexities (e.g., logarithmic variant vs. 3-para-log). Thus, based on our study, the choices above hold advantages in both performance and speed.

Connection to Prior Work. When the bias kernel, Eq. (3), is a triangle kernel: $c - |m-n|$, our model reduces to ALiBi [Press et al., 2022]. Wennberg and Henter [2021] discuss the situation where the bias kernel is a Gaussian kernel. Tsai et al. [2019] is the case where there is no bias kernel and the attention product $\boldsymbol{q}_m^\top \boldsymbol{k}_n$ is multiplied by an exponentiated inner product kernel, $\exp(\boldsymbol{x}^\top \boldsymbol{y})$. Since ALiBi is the state-of-the-art and has great input length extrapolation, we will focus on the comparison with ALiBi in our experiments.

The logarithmic variant has an implicit connection to T5 positional bias [Raffel et al., 2020]. According to the official GitHub repository `https://github.com/google-research/text-to-text-transfer-transformer` and the HuggingFace Transformer [Wolf et al., 2020], T5 bias is implemented with a log-binning strategy. For each head of the transformer, they maintain a bucket of 32 learnable parameters and assign the relative positional bias b_{m-n} to these parameters as

$$b_{m-n} = \begin{cases} \text{bucket}[0] & \text{if } m-n < 0 \\ \text{bucket}[m-n] & \text{if } 0 \leq m-n < 16 \\ \text{bucket}[\min(31, \lfloor \frac{\log((m-n)/16)}{\log(128/16)}) \cdot 16 \rfloor] & \text{if } m-n \geq 16, \end{cases}$$

where $\lfloor \cdot \rfloor$ is the floor function. Note that the log factor is approximately $7.7 \log \frac{m-n}{16}$. Therefore, T5 is using a logarithmic bucket assignment, which turns out to extrapolate to different input lengths. Compared with T5, our logarithmic variant uses less parameters (2x12 vs. 32x12) but cannot learn non-monotonic relations (the log function is monotonic). We will conduct more comparisons with T5 bias in our experiments.

5 Experiments

5.1 Dataset and Implementation Description

Dataset. We conduct experiments on OpenWebText2, GitHub, and ArXiv datasets gathered in Gao et al. [2020]. OpenWebText2 includes recent content from Reddit submissions until 2020, content from multiple languages, document metadata, multiple dataset versions, and open-source replication code. GitHub includes open-source repositories written in primary coding languages such as Java, C/C++, Python, and Go. ArXiv includes papers written in LaTex in Math, Computer Science, Physics, and some related fields. These tasks are motivated by the downstream applications such as online chatting [Roller et al., 2021], code completion [Chen et al., 2021a], and academic paper summarization [Zhang et al., 2020].

Table 1: **Dataset Overview.** Raw Size is the size before any up- or down-sampling.

	OpenWebText2	GitHub	ArXiv
Raw Size	66.77 GB	95.16 GB	56.21 GB
Type	Internet	Coding	Academic

Implementation. We adapt our model from GPT-NeoX [Black et al., 2021], a transformer implementation by the EleutherAI team. The codebase is based on NVIDIA Megatron Language Model [Shoeybi et al., 2019] and further accelerated using Microsoft DeepSpeed library [Rasley et al., 2020].

Our model is trained on a machine with one NVIDIA A100 GPU with 40 GB of memory. We adopt almost all configurations of small GPT-NeoX[2], except that we change the train-micro-batch-size to 32, seq-length to 512, and max-position-embeddings to 512. Table 2 summarizes the important configurations fixed throughout our experiments. In particular, the floating-point encoding is set as

Table 2: **162M Model Configurations.**

# Layers	Hidden Size	# Attention Heads	Train Seq. Len.	# Trainable Params.
12	64	12	512	162M
Optimizer	Batch Size	Train Steps	Precision	# Trainable Params. for RPEs
Adam (lr 6e-4)	32	50,000	bfloat16	at most 36

bfloat16 (Brain Floating Point, developed by Google Brain) so that the training can be accelerated by half-precision computation with reliable stability [Kalamkar et al., 2019]. Hidden size 64 means that $d = 64$ in Eq. (1).

5.2 Experimental Results (Also c.f. Appendix A.7)

We conduct experiments to cover aspects such as input length extrapolation, application on different domains, and comparison with the prior work. These are elaborated as below. (i) Motivated by the input length extrapolation demonstrated in [Press et al., 2022], we train our model with length 512 and test on lengths ranging from 512 to 16384. We hope that the emphasis on extrapolation enables the application of transformers to longer sequences. (ii) To evaluate the applicability of the model in different domains, we conduct experiments on OpenWebText2, GitHub, and ArXiv datasets. (iii) To validate the effectiveness of our method, we compare KERPLE with Sinusoidal [Vaswani et al., 2017], Rotary [Su et al., 2021], T5 [Raffel et al., 2020], and ALiBi [Press et al., 2022].

Table 3 reports the perplexity at different extrapolation lengths. We perform non-overlapping evaluation: Suppose text is segmented in different manner for 512 and 1024 tokens, we have N sentences and N/2 correspondingly to evaluate. We also perform a paired two-sided t-test to validate the statistical significance (significance level=0.05). We compare each candidate RPE with our proposed logarithmic variant and mark the candidate with a † *if the log variant is statistically significantly better*. Table 4 reports the training speeds. These tables yield three conclusions. First, within the KERPLE framework, the logarithmic variant is better than the power variant. Secondly, the logarithmic variant is 9.7% faster than T5. In terms of extrapolation, the logarithmic variant generally does better than T5 but could be slightly worse than T5 at shorter lengths. Third, the logarithmic variant is slightly slower than two of the prior work (ALiBi, Rotary, and Sinusoidal) but consistently outperform them at all extrapolation lengths. These are elaborated as follows.

Logarithmic Variant vs. Power Variant. In our proposed KERPLE framework, the logarithmic variant is better than the power variant. Precisely, the logarithmic variant is 4.4% faster and has lower perplexities across all extrapolation lengths and all tasks.

Logarithmic Variant vs. T5. In terms of speed, the logarithmic variant is 9.7% faster than T5. In terms of extrapolation perplexity, the logarithmic variant is close to or slightly worse than T5 when the extrapolation length is shorter than 2048, and consistently excels T5 at longer extrapolation lengths. The tendency of extrapolation holds for all datasets evaluated in this work.

[2] `https://github.com/EleutherAI/gpt-neox/blob/main/configs/small_bf16.yml`

Logarithmic Variant vs. ALiBi, Rotary, and Sinusoidal. The logarithmic variant is 1.6% slower, 7.5% faster, and 3.0% slower than ALiBi, Rotary, and Sinusoidal. The speed comparison makes sense because we require only a limited amount of learnable parameters for RPEs (at most $3 \cdot H$). Also, the logarithmic variant consistently outperforms prior work at all extrapolation lengths and tasks.

Table 3: **Perplexity Comparison on OpenWebText2, GitHub, and ArXiv.** All models are trained for 50k steps with training length 512 and five random seeds. $x^\dagger$ means our log variant is statistically significantly *better* than x. The test used is paired two-sided t-test with $\alpha = 0.05$.

OpenWebText2						
Extrp.	KERPLE (log)	KERPLE (power)	ALiBi	T5	Rotary	Sinusoidal
512	23.9 ± 0.6	23.9 ± 0.6	23.9 ± 0.6	$\mathbf{23.7 \pm 0.6}$	$24.2 \pm 0.6^\dagger$	$33 \pm 1^\dagger$
1024	22.0 ± 0.6	22.1 ± 0.7	$22.4 \pm 0.5^\dagger$	$\mathbf{21.9 \pm 0.6}$	$32.8 \pm 1.7^\dagger$	$750 \pm 346^\dagger$
2048	$\mathbf{21.6 \pm 0.3}$	$21.9 \pm 0.2^\dagger$	$22.5 \pm 0.2^\dagger$	21.7 ± 0.2	$62.4 \pm 6.1^\dagger$	$5507 \pm 2607^\dagger$
4096	$\mathbf{21.2 \pm 0.4}$	$21.5 \pm 0.5^\dagger$	$22.2 \pm 0.4^\dagger$	$22.5 \pm 0.6^\dagger$	$111 \pm 13.8^\dagger$	$14039 \pm 2325^\dagger$
8192	$\mathbf{21.3 \pm 0.4}$	$21.6 \pm 0.4^\dagger$	$22.3 \pm 0.3^\dagger$	$25.5 \pm 1.3^\dagger$	$185 \pm 18.9^\dagger$	$22621 \pm 1927^\dagger$
16384	$\mathbf{21.4 \pm 0.6}$	21.6 ± 0.6	$22.5 \pm 0.5^\dagger$	$31.4 \pm 3.1^\dagger$	$269 \pm 33.0^\dagger$	$30046 \pm 4824^\dagger$
GitHub						
Extrp.	KERPLE (log)	KERPLE (power)	ALiBi	T5	Rotary	Sinusoidal
512	3.40 ± 0.20	3.42 ± 0.20	3.42 ± 0.21	$\mathbf{3.38 \pm 0.21}$	$3.44 \pm 0.20^\dagger$	$4 \pm 0.2^\dagger$
1024	3.04 ± 0.14	3.07 ± 0.16	$3.15 \pm 0.17^\dagger$	$\mathbf{3.02 \pm 0.14}$	$3.86 \pm 0.25^\dagger$	$105 \pm 39^\dagger$
2048	2.86 ± 0.10	$2.90 \pm 0.08^\dagger$	$3.13 \pm 0.10^\dagger$	$\mathbf{2.84 \pm 0.09}$	$5.94 \pm 0.64^\dagger$	$1380 \pm 404^\dagger$
4096	$\mathbf{2.74 \pm 0.05}$	2.79 ± 0.06	$3.04 \pm 0.08^\dagger$	$2.78 \pm 0.04^\dagger$	$11.1 \pm 1.55^\dagger$	$5217 \pm 1118^\dagger$
8192	$\mathbf{2.71 \pm 0.05}$	2.76 ± 0.05	$3.04 \pm 0.03^\dagger$	$2.95 \pm 0.13^\dagger$	$20.2 \pm 2.75^\dagger$	$10081 \pm 3583^\dagger$
16384	$\mathbf{2.75 \pm 0.16}$	2.76 ± 0.13	$3.02 \pm 0.13^\dagger$	$3.35 \pm 0.27^\dagger$	$31.3 \pm 5.20^\dagger$	$16443 \pm 8503^\dagger$
ArXiv						
Extrp.	KERPLE (log)	KERPLE (power)	ALiBi	T5	Rotary	Sinusoidal
512	6.07 ± 0.26	6.10 ± 0.26	$6.12 \pm 0.26^\dagger$	$\mathbf{6.03 \pm 0.26}$	6.07 ± 0.27	43 ± 44
1024	5.61 ± 0.10	$5.65 \pm 0.10^\dagger$	$5.82 \pm 0.09^\dagger$	$\mathbf{5.58 \pm 0.09}$	$7.49 \pm 0.34^\dagger$	$221 \pm 136^\dagger$
2048	5.22 ± 0.12	$5.26 \pm 0.13^\dagger$	$5.71 \pm 0.14^\dagger$	$\mathbf{5.21 \pm 0.14}$	$14.2 \pm 1.81^\dagger$	$730 \pm 343^\dagger$
4096	$\mathbf{5.20 \pm 0.10}$	5.25 ± 0.09	$5.87 \pm 0.08^\dagger$	$5.32 \pm 0.16^\dagger$	$30.1 \pm 4.32^\dagger$	$1998 \pm 497^\dagger$
8192	$\mathbf{5.01 \pm 0.10}$	5.06 ± 0.15	$5.74 \pm 0.13^\dagger$	$5.54 \pm 0.39^\dagger$	$54.3 \pm 6.22^\dagger$	$4228 \pm 2645^\dagger$
16384	$\mathbf{5.07 \pm 0.16}$	5.07 ± 0.19	$5.78 \pm 0.15^\dagger$	$6.25 \pm 0.61^\dagger$	$85.4 \pm 7.40^\dagger$	6674 ± 5696

Table 4: **Training Time Comparison on GitHub**

	KERPLE (log)	KERPLE (power)	ALiBi	T5	Rotary	Sinusoidal
sec/step	0.307	0.321	0.302	0.340	0.332	0.298

5.3 Experiments on Complicated Kernels

In addition to the practical variants (power & logarithmic) in section 4, we consider two complicated versions of the composite kernel, Eq. (4), as follows.

(bias+wht) bias + weight:
$k^{\text{comp}}([\boldsymbol{q}_m, m], [\boldsymbol{k}_n, n]) = \boldsymbol{q}_m^\top \boldsymbol{k}_n \cdot \exp(-r_3|m-n|^{r_4}) + c - r_1|m-n|^{r_2}$
with $r_1, r_3 > 0$ and $0 < r_2, r_4 \leq 2$.

(3-para-log) 3-parameter-logarithmic:
$k^{\text{comp}}([\boldsymbol{q}_m, m], [\boldsymbol{k}_n, n]) = \boldsymbol{q}_m^\top \boldsymbol{k}_n + c - r_1 \cdot \log(1 + r_2|m-n|^{r_3})$
with $r_1, r_2 > 0$ and $0 < r_3 \leq 2$.

Recall the tensor product property of a kernel: if k_1 is a kernel on $\mathcal{X}$ and k_2 is a kernel on $\mathcal{Y}$, then $k((x,y),(x',y')) = k_1(x,x')k_2(y,y')$ is a kernel on $\mathcal{X} \times \mathcal{Y}$. Therefore, (bias+wht) is the setting

where we train a weight $\exp(-r_3|m-n|^{r_4})$ and a bias kernel $c - r_1|m-n|^{r_2}$. $\boldsymbol{q}_m^\top \boldsymbol{k}_n$ is multiplied by the weight kernel and then added with the bias kernel. (3-para-log) is the setting where we consider $|m-n|^{r_3}$ in the log. When $r_3 = 1$, it is reduced to the logarithmic variant proposed in section 4.

We plug in these composite kernel k^{comp} into our KERPLE framework, Eq. (2), and test the performance of these RPE. Compared with section 5.2, Table 5 suggests that these variants do not have clear advantage in extrapolation performance, e.g., 3-para-log is slightly better in perplexity than the (two-parameter) logarithmic variant. Thus, enlarging the complexity of kernels does not necessarily give better performance in the context of RPE.

Table 5: **Perplexity Comparison for KERPLE with Complicated Kernels on OpenWebText2, GitHub, and ArXiv.** All models are trained for 50k steps with training length 512 and five seeds random. OOM means out of memory.

Extrp.	OpenWebText2		GitHub		ArXiv	
	(bias+wht)	(3-para-log)	(bias+wht)	(3-para-log)	(bias+wht)	(3-para-log)
512	24.1 ± 0.6	23.8 ± 0.6	3.44 ± 0.21	3.40 ± 0.20	6.11 ± 0.27	6.06 ± 0.27
1024	22.2 ± 0.6	22.0 ± 0.7	3.08 ± 0.15	3.04 ± 0.13	5.66 ± 0.09	5.61 ± 0.10
2048	21.9 ± 0.4	21.6 ± 0.2	2.90 ± 0.12	2.85 ± 0.10	5.28 ± 0.12	5.21 ± 0.12
4096	21.5 ± 0.5	21.2 ± 0.4	2.79 ± 0.06	2.73 ± 0.05	5.31 ± 0.08	5.18 ± 0.09
8192	21.4 ± 0.5	21.3 ± 0.4	2.76 ± 0.03	2.68 ± 0.04	5.16 ± 0.18	5.00 ± 0.11
16384	OOM	OOM	OOM	OOM	OOM	OOM

6 Additional Analyses

Since the power and logarithmic variants derived from KERPLE achieve superior performance on length extrapolation across various datasets, we investigate the underlying reason by visualizing the *effective length* as shown in Figure 2. The visualization works in the following procedure.

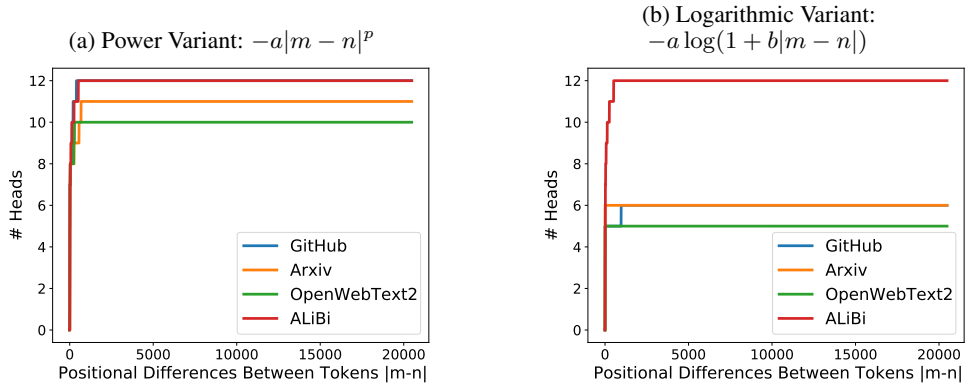
1. For each training dataset, the learnable parameters $(r_1^{(h)}, ..., r_\ell^{(h)})$ associated with each head h (12 in total) are extracted from the model checkpoint. The CPD kernel at head h is $\tilde{k}^{(h)} = \tilde{k}_{r_1^{(h)},...,r_\ell^{(h)}}$. Both the power and the logarithmic variants in corollary 1 undergo a similar procedure. The only difference is that their $\tilde{k}$'s are different.
2. For each head h, we compute the *effective length* of $\tilde{k}^{(h)}$ as $\text{eff}^{(h)} = \min_{\tilde{k}^{(h)}(0,|m-n|)<-2} |m-n|$. That is, the relative positional difference $|m-n|$ such that $\tilde{k}(m,n) \overset{\text{shift-inv.}}{=} \tilde{k}(0,|m-n|)$ just becomes smaller than -2. Note $\tilde{k}(0,|m-n|)$ strictly decreases in $|m-n|$, so there is only one possible value. We pick -2 here because $\tilde{k}$ is a bias and is followed by the Softmax normalization. A bias of -2 or smaller can make a great impact on the output of Softmax[3]. $\text{eff}^{(h)}$ is interpreted as the *effective length* because, when $|m-n| < \text{eff}^{(h)}$, the attenuation due to $\tilde{k}^{(h)}$ is not strong. When $|m-n| > \text{eff}^{(h)}$, the attenuation is strong and has a large impact on $\boldsymbol{q}_m^\top \boldsymbol{k}_n + \tilde{k}^{(h)}(m,n)$.
3. Then, for each $|m-n| \in [0, ..., 20480]$, we count the number of heads that satisfies $\text{eff}^{(h)} \leq |m-n|$. This gives a cumulative plot as shown in Figure 2, where the x-axis is $|m-n|$ and the y-axis is $\text{Count}(\{h : \ h \in [1, ..., 12], \ \text{eff}^{(h)} \leq |m-n|\})$.
4. Repeat the above steps for other datasets and kernels.

Interpretation of Curves. For a point (x, y) on a curve, it means that there are y heads with at least -2 bias when the token distance $|m-n|$ is greater than x. In other words, the slower the y converges to 12, the longer the inter-token range that the model focuses on.

The Advantage of Learnable Parameters. We observe that ALiBi [Press et al., 2022] produces the same curve no matter which dataset is used. The reason is that ALiBi selects a fixed parameter

[3]Since Softmax is an exponentiated function, a -2 bias in the Softmax's argument roughly gives an attenuation of $\exp(-2) \approx 0.135$.

Figure 2: **Number of Heads with Effective Lengths $\leq |m-n|$ for different choices of CPD kernels and datasets.** See section 6 for details.



$r = 2^{\frac{-8h}{H}}$ at head h for its linear bias $-r|m-n|$ (H heads in total) regardless of the dataset. While this strategy is useful for extrapolation, we hypothesize that different datasets might have different characteristics, e.g., the average distance of highly related tokens should differ among the datasets as shown in Figure 2. These characteristics are easier adapted by learnable parameters. Thus, we believe that learnable parameters have more advantages in capturing the dataset-dependent characteristics.

Trends Across Datasets. We notice that both kernels trained on OpenWebText2 tend to focus more on distant relations. This makes sense because OpenWebText2 has the highest perplexity scores among all datasets, implying that more context is needed to disambiguate the next predicted token. The opposite trend holds for Arxiv and GitHub datasets, which is reasonable considering their lower perplexity scores.

Characteristics Learned by Kernels. Under any dataset, the logarithmic variant tends to focus more on distant relations than the power variant does. We can explain it through their functional forms. Because logarithm ($-a\log(1+b|m-n|)$) decays much slower than power ($-a|m-n|^p$) does, the log variant might encourage the focus on distant relations.

7 Conclusion and Future Work

A general framework, KERPLE, is proposed to kernelize relative positional embeddings for length extrapolation. At the core of this framework is the application of CPD kernels and the derivation of practical variants. We show that these CPD kernels can be implicitly converted to PD kernels, which keep the inner product interpretation of self-attention. We also demonstrate that the logarithmic variant achieves exceptional extrapolation performance on three large language modeling datasets. We believe our work paves the way for some interesting future directions that resolve our limitations. For instance, we can consider general kernel families and model non-monotonic effects due to positional differences. In addition, the use of learnable parameters in KERPLE might enable better generalization to inputs higher than 1-dimensional. Last but not least, there is always room for improving memory efficiency by adjusting the model architecture and training procedure.

8 Broader Impact

Our work develops a better understanding of relative positional embedding for transformers based on expressive kernel classes that adapt well to various datasets. The results apply to domains where the positional information is helpful in the modeling, e.g., natural language, programming language, and DNA/protein sequences for biology/medicine. The studies of transformers may have positive economic effects by enabling new tasks which cannot be done by humans or enhancing accuracy and efficiency. But inappropriate use can have negative societal impacts. These include job loss due to automation, the ethical challenges from improper text generation, and the privacy issues in the data collection process. These implications apply to any research on natural language processing and are not associated with any specific work.

References

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. Recipes for building an open-domain chatbot. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 300–325, Online, April 2021. Association for Computational Linguistics.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021a.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR, 2020.

Ofir Press, Noah Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022.

Shun Kiyono, Sosuke Kobayashi, Jun Suzuki, and Kentaro Inui. SHAPE: Shifted absolute position embedding for transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3309–3321, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

Tatiana Likhomanenko, Qiantong Xu, Gabriel Synnaeve, Ronan Collobert, and Alex Rogozhnikov. Cape: Encoding relative positions with continuous augmented positional embeddings. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 16079–16092. Curran Associates, Inc., 2021.

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

Pu-Chin Chen, Henry Tsai, Srinadh Bhojanapalli, Hyung Won Chung, Yin-Wen Chang, and Chun-Sung Ferng. A simple and effective positional encoding for transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2974–2988, 2021b.

Ulme Wennberg and Gustav Eje Henter. The case for translation-invariant self-attention in transformer-based language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 130–140, Online, August 2021. Association for Computational Linguistics.

Bernhard Schölkopf. The kernel trick for distances. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.

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

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

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1243–1252. JMLR.org, 2017.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

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

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. In *International Conference on Learning Representations*, 2019.

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

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Zhiheng Huang, Davis Liang, Peng Xu, and Bing Xiang. Improve transformer models with better relative position embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3327–3335, Online, November 2020. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. {DEBERTA}: {DECODING}-{enhanced} {bert} {with} {disentangled} {attention}. In *International Conference on Learning Representations*, 2021.

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

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

Kan Wu, Houwen Peng, Minghao Chen, Jianlong Fu, and Hongyang Chao. Rethinking and improving relative position encoding for vision transformer. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10013–10021, 2021a.

Philipp Dufter, Martin Schmitt, and Hinrich Schütze. Position Information in Transformers: An Overview. *Computational Linguistics*, pages 1–31, 07 2022.

Sebastian Mika, Bernhard Schölkopf, Alex Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel pca and de-noising in feature spaces. *Advances in neural information processing systems*, 11, 1998.

Christina Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for svm protein classification. In *Biocomputing 2002*, pages 564–575. World Scientific, 2001.

Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556, 2004.

Hiroyuki Takeda, Sina Farsiu, and Peyman Milanfar. Kernel regression for image processing and reconstruction. *IEEE Transactions on image processing*, 16(2):349–366, 2007.

Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4344–4353, Hong Kong, China, November 2019. Association for Computational Linguistics.

Chuhan Wu, Fangzhao Wu, and Yongfeng Huang. DA-transformer: Distance-aware transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2059–2068, Online, June 2021b. Association for Computational Linguistics.

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

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR, 13–18 Jul 2020.

Yifan Chen, Qi Zeng, Heng Ji, and Yun Yang. Skyformer: Remodel self-attention with gaussian kernel and nystr\" om method. *Advances in Neural Information Processing Systems*, 34, 2021c.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nystöm-based algorithm for approximating self-attention. In *Proceedings of the... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, volume 35, page 14138. NIH Public Access, 2021.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations*, 2021.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. In *International Conference on Learning Representations*, 2022.

Christian Berg, Jens Peter Reus Christensen, and Paul Ressel. *Harmonic analysis on semigroups: theory of positive definite and related functions*, volume 100. Springer, 1984.

Isaac J Schoenberg. Metric spaces and positive definite functions. *Transactions of the American Mathematical Society*, 44(3):522–536, 1938.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Sid Black, Stella Biderman, Alex Andonian, Quentin Anthony, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Jason Phang, Michael Pieler, Shivanshu Purohit, Tri Songz, Phil Wang, and Samuel Weinbach. GPT-NeoX: Large scale autoregressive language modeling in pytorch, 2021. URL `http://github.com/eleutherai/gpt-neox`.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506, 2020.

Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, et al. A study of bfloat16 for deep learning training. *arXiv preprint arXiv:1905.12322*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the end of sections 4 and 7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See section 8.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.1.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See section 5 and the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Table 2.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Table 3.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See section 5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See section 5.

(b) Did you mention the license of the assets? [Yes] We use publicly released assets from Black et al. [2021], Gao et al. [2020] and the licenses are under Apache License 2.0.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See section 5.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] The data are publicly released. See section 5.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] The data comes from Gao et al. [2020] where the provider has already anonymized all user-contributed content. See section 5.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Proof of CPD Shift Lemma

Lemma 1 (CPD Shift Lemma)**.** *Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a conditionally positive definite (CPD) kernel. Then, there exists $c \geq 0$ such that $c + k(x, y)$ is a positive definite kernel.*

Proof. Let $K = [k(x_i, x_j)]_{i,j=1}^N$ be the matrix generated by $\{x_1, ..., x_N\}$ with $N \in \mathbb{N}$. Consider

$$f_c(v) = v^\top (c\mathbb{1}\mathbb{1}^\top + K)v = c(v^\top \mathbb{1})^2 + v^\top Kv.$$

We want to show there exists a large enough c such that $f_c(v) \geq 0$ for all $v \in \{v : \|v\| = 1\}$.

(i) It is sufficient to consider $a^* = \min_{v:\|v\|=1} v^\top Kv < 0$.

Let a^* be the solution to the minimization:

$$a^* = \min_{v:\|v\|=1} v^\top Kv.$$

Since $v^\top Kv$ is continuous in v and $\{v : \|v\| = 1\}$ is compact (i.e., closed and bounded), a^* must exist. If $a^* \geq 0$, K is positive semidefinite and $f_c(v) \geq 0$ for $c \geq 0$. Thus, without loss of generality, we assume $a^* < 0$.

(ii) It is sufficient to consider K without zero eigenvalues (i.e., full rank).

If there exists v_0 such that $Kv_0 = 0$, then $c \geq 0$ is enough to satisfy $f_c(v_0) \geq 0$. For any v_1 satisfying $v_1^\top v_0 = 0$, we have $(v_1 + v_0)^\top K(v_1 + v_0) = v_1^\top Kv_1$. Therefore, whether there exists c to have $f_c(v) \geq 0$ doesn't depend on the eigenvector corresponding to zero eigenvalue (if there is such a vector). This means it is enough to consider K without zero eigenvalues.

(iii) It is sufficient to consider strict CPD.

By definition of conditional positive definiteness (CPD), we know $v^\top Kv \geq 0$ when $v^\top \mathbb{1} = 0$. Since K has no zero eigenvalue, we cannot have $v^\top Kv = 0$ when $v^\top \mathbb{1} = 0$[4]. This means the inequality is strict here: $v^\top Kv > 0$ when $v^\top \mathbb{1} = 0$, and it is enough to consider strict CPD.

(iv) It is sufficient to show there exists (small enough) $\delta > 0$ such that $v' \in T_\delta \Rightarrow v'^\top Kv' > 0$.

Note T_δ is defined as

$$T_\delta = \{v' : |v'^\top \mathbb{1}| < \delta,\ \|v'\| = 1\}.$$

[4] By spectral decomposition, $v^\top Kv = \sum_i \lambda_i (v^\top u_i)^2 \geq 0$. Since there is no $\lambda_i = 0$, the inequality is strict.

For any $v' \in T_\delta$, if v' satisfies $v'^\top K v' > 0$, then $c \geq 0$ is enough to have $f_c(v') \geq 0$. Conversely, when $|v^\top \mathbb{1}| \geq \delta$ and $\|v\| = 1$, observe that

$$f_c(v) = c(v^\top \mathbb{1})^2 + v^\top K v \geq c(v^\top \mathbb{1})^2 + a^* \geq c\delta^2 + a^*$$

Then, $f_c(v) \geq 0$ when $c \geq \frac{-a^*}{\delta^2}$. Therefore, we need to prove $v' \in T_\delta \Rightarrow v'^\top K v' > 0$ for small enough δ.

(v) Leveraging Continuity. Consider v' satisfying $\|v' - v\| < \delta_2$ with $v \in S = \{v : \|v\| = 1,\ v^\top \mathbb{1} = 0\}$. Since $v^\top K v$ is continuous in v and $\|K\| < \infty$, for any $\epsilon > 0$ and any $v \in S$, a small enough $\delta_2 > 0$ gives $|v'^\top K v' - v^\top K v| < \epsilon$.

To see this, taking $v' = v + p$ with $\|p\| < \delta_2$, we have

$$|v'^\top K v' - v^\top K v| = |p^\top K v + v^\top K p + p^\top K p| \underset{\|v\|=1}{\leq} \|K\|(2\|p\| + \|p\|^2) \leq \|K\|(2\delta_2 + \delta_2^2).$$

Therefore, $0 < \delta_2 < \sqrt{1 + \epsilon/\|K\|} - 1$ is enough to have $|v'^\top K v' - v^\top K v| < \epsilon$.

By definition of strict CPD, we know $\min_{v \in S} v^\top K v = \lambda > 0$. Thus, take $\epsilon < \lambda$, a small enough δ_2 gives $v'^\top K v' > v^\top K v - \epsilon >= \lambda - \epsilon > 0$. In other words, there exists a small enough δ_2 such that $v'^\top K v' > 0$ for $v' \in S_{\delta_2} = \{v' : \|v' - v\| < \delta_2,\ v \in S\}$.

(vi) Proving $\exists\, \delta > 0$ s.t. $v' \in T_\delta \Rightarrow v'^\top K v' > 0$.

Due to (iv), we want to show $\exists\ \delta > 0$ s.t. $v' \in T_\delta \Rightarrow v'^\top K v' > 0$. We will prove by the conclusion of (v). Let $\|v'\| = 1$, $v'^\top \mathbb{1} = r$ with $|r| < \delta$ and $v'' = v' - \frac{r}{n}\mathbb{1}$. We have

$$v''^\top \mathbb{1} = 0, \quad \|v' - v''\| = \frac{r}{\sqrt{n}}, \quad \|v''\| = \sqrt{\|v' - \frac{r}{n}\mathbb{1}\|^2} = \sqrt{1 - \frac{r^2}{n}}.$$

Take $v = \frac{v''}{\|v''\|} = \frac{v''}{q}$, where $q = \sqrt{1 - \frac{r^2}{n}}$. We have $v^\top \mathbb{1} = 0, \quad \|v\| = 1$ and

$$\begin{aligned}
\|v' - v\|^2 =& \|(1 - \frac{1}{q})v' + \frac{1}{q}\frac{r}{n}\mathbb{1}\|^2 = (1 - \frac{1}{q})^2 + \frac{r^2}{nq^2} + 2(1 - \frac{1}{q})\frac{1}{q}\frac{r^2}{n} \\
=& (1 - \frac{1}{q})^2 + \frac{r^2}{n}(\frac{2}{q} - \frac{1}{q^2}) = \frac{1}{q^2}\Big((q-1)^2 + \frac{r^2}{n}(2q-1)\Big) \\
=& \frac{1}{1 - \frac{r^2}{n}}\Big(1 - \frac{r^2}{n} - 2q + 1 + \frac{r^2}{n}2q - \frac{r^2}{n}\Big) \\
=& \frac{1}{1 - \frac{r^2}{n}}\Big(1 - \frac{r^2}{n} - 2q(1 - \frac{r^2}{n}) + 1 - \frac{r^2}{n}\Big) = 2 - 2q \\
=& 2(1 - \sqrt{1 - \frac{r^2}{n}}) \approx 2(1 - 1 + \frac{1}{2}\frac{r^2}{n}) = \frac{r^2}{n} \quad (\sqrt{1-x} \approx 1 - \frac{x}{2} \text{ when } |x| \ll 1)
\end{aligned}$$

Thus, $\|v' - v\| = O(\frac{|r|}{\sqrt{n}}) \leq O(\frac{\delta}{\sqrt{n}}) < \delta_2$ for small enough δ. This implies that, with a small enough δ, for any $v' \in T_\delta$, we can find $v \in S$ such that $\|v' - v\| < \delta_2$. Thus, $v' \in S_{\delta_2}$, and by (v), we arrive at $v'^\top K v' > 0$.

□

A.2 Shift-invariant Kernels with Bounded and Unbounded Ranges

Definition 1 implies a shift-invariant kernel is generated by a univariate function $f : \mathcal{X} \to \mathbb{R}$. To characterize the set of valid univariate functions, we introduce the positive definite functions as below.

Definition 3 (Positive definite function)**.** *A (real) positive definite function is a function $f : \mathcal{X} \to \mathbb{R}$ such that for any integer N and any set of N points $\{x_i \in \mathcal{X}\}_{i=1}^N$, the $N \times N$ matrix $\boldsymbol{A} = [f(x_i - x_j)]_{i,j=1}^N$ is positive semidefinite.*

We will interchange the ideas of shift-invariant kernels and positive definite functions because they are equivalent by definition. Any statement in positive definite functions can be translated into shift-invariant kernels, and vice versa. Because of this, we will use some facts about the positive definite functions to derive the shift-invariant kernels of our interest.

Generalizing Classical Bounded Shift-invariant Kernel. In the literature, there has been studies on applying kernels in the attention mechanism [Tsai et al., 2019, Choromanski et al., 2021, Peng et al., 2021]. One of the most common approach is to consider the Gaussian kernel:

$$k(m,n) = \exp(-\gamma(m-n)^2), \quad \gamma > 0.$$

Note the Gaussian kernel is bounded ($k(m,n) \in (0,1]$ for the case above). To generalize it to a broader class of bounded shift-invariant kernels, observe that the Gaussian kernel generated by a positive definite function of the form $f(x) = \exp(-|x|^2)$. Since there is no strong reason to stick to the power of 2, one may generalize it to a broader class of positive definite functions as below.

Fact 5 (Corollary 3 of Schoenberg [1938])**.** *$\exp(-|x|^p)$ is positive definite if $0 < p \leq 2$ and not positive definite if $p > 2$.*

Fact 5 implies that, if one wants to find a class of bounded shift-invariant kernel (i.e., $k(m,n)$ is within some fixed interval for any m, n), then $k(m,n) = \exp(-a|m-n|^p)$ with $a > 0$ and $p \in (0,2]$ may be of interest.

Constructing Unbounded Shift-invariant Kernels. A limitation of Fact 5 is that it only generates kernels with a bounded range (here, the range is bounded in $(0,1]$). In situations where there are no explicit bounds, one might want to consider kernels with unbounded range. To construct such kernels, we utilize a kernel representation theorem presented in Schoenberg [1938]:

Fact 6 (Theorem 4 of Schoenberg [1938])**.** *$f(x)$ is bounded away from zero ($f(x) > 0$) and its positive powers $f(x)^\lambda$ ($\lambda > 0$) are all positive definite if and only if $f(x)$ is of the form*

$$f(x) = \exp(c + \psi(x)),$$

where $\psi(x)$ is positive definite and c is a real constant.

Since Fact 6 works for non-negative kernels, we combine it with Fact 5 and show the following class of shift-invariant kernel with an unbounded range.

Proposition 1 (Kernel from Distance Powers)**.** *For any $p \in (0,2]$, there exists $c_{\min} \in \mathbb{R}$ such that for any $c \geq c_{\min}$,*

$$k(m,n) = c - |m-n|^p,$$

is a positive definite kernel. When $p > 2$, there is no c to make $k(m,n)$ positive definite.

Proof. Due to Fact 5, we know $\exp(-|x|^p)$ is positive definite when $p \in (0,2]$. Since $\exp(-|x|^p) > 0$ and $\exp(-\lambda|x|^p)$ is positive definite for any $\lambda > 0$[5], Fact 6 implies there exists a $c' \in \mathbb{R}$ and a positive definite $\psi(x)$ such that

$$\exp(-|x|^p) = \exp(c' + \psi(x)).$$

In other words, $-c' - |x|^p$ is a positive definite function. Take $c_{\min} = -c'$. We see that $c_{\min} - |m-n|^p$ is a shift-invariant positive definite kernel. Finally, let $k(m,n) = c - |m-n|^p$ with $c \geq c_{\min}$. The $N \times N$ matrix $[k(x_i, x_j)]_{i,j=1}^N$ generated by $k(m,n)$ on points $\{x_i \in \mathbb{R}\}_{i=1}^N$ obeys

$$[k(x_i,x_j)]_{i,j=1}^N = [c_{\min} - |x_i - x_j|^p]_{i,j=1}^N + (c - c_{\min})\mathbb{1}\mathbb{1}^\top \succeq (c - c_{\min})\mathbb{1}\mathbb{1}^\top \succeq 0,$$

where $\succeq$ is the Loewner order and $\mathbb{1} = [1, ..., 1]^\top$ in the N-dimensional vector with all ones. This shows $k(m,n)$ is a shift-invariant positive definite kernel when $0 < p \leq 2$. The conclusion on $p > 2$ is proved by contradiction. When $p > 2$, if there exists a c such that $k(m,n)$ is positive definite, then $\exp(k(m,n))$ is positive definite, which contradicts to the case of $p > 2$ in Fact 5. □

[5] $\exp(-\lambda|x|^p) = \exp(-|\lambda^{1/p}x|^p)$ is a constant rescaling of $\exp(-|x|^p)$ and therefore is positive definite.

Prop. 1 introduces a kernel with unbounded range ($k(m,n) \in (\infty, c]$) and is adapted from the p-th power of the distance $|m-n|$. Since the distance is a notion of "dissimilarity", $-|m-n|^p$ becomes a notion of "similarity", which gives a sense of kernel. Thereby, we can interpret the constant c as the required value to shift $-|m-n|^p$ such that $c - |m-n|^p$ becomes a positive definite kernel.

In fact, $-|m-n|^p$ is a conditional positive definite kernel for $p \in (0, 2]$ Schölkopf [2000]. Therefore, the fact that $-|m-n|^p$ can become a positive definite kernel by shifting is not a coincidence, as it has already had an intimate relation to positive definite kernels.

A.3 Experiments on Gaussian-like Kernels

Since the prior work on shift-invariant kernels mainly focuses on Gaussian kernels, we present preliminary experiments on Gaussian-like kernels. Compared with section 5.2, the perplexities of these kernels are large at every extrapolation length. This verifies our previous assertion that the Gaussian-like kernels have limited extrapolation ability.

Because the kernel can be used as a weight or a bias, we consider four kinds of the composite kernel (see section 4) as follows.

(2-para-bias) $r_1, r_2 > 0$.
$k^{\text{comp}}([\boldsymbol{q}_m, m], [\boldsymbol{k}_n, n]) = \boldsymbol{q}_m^\top \boldsymbol{k}_n + r_1 \exp(-r_2|m-n|^2)$.

(3-para-bias) $r_1, r_2 > 0$ and $0 < r_3 \leq 2$.
$k^{\text{comp}}([\boldsymbol{q}_m, m], [\boldsymbol{k}_n, n]) = \boldsymbol{q}_m^\top \boldsymbol{k}_n + r_1 \exp(-r_2|m-n|^{r_3})$.

(1-para-wht) $r_1 > 0$.
$k^{\text{comp}}([\boldsymbol{q}_m, m], [\boldsymbol{k}_n, n]) = \boldsymbol{q}_m^\top \boldsymbol{k}_n \cdot \exp(-r_1|m-n|^2)$.

(2-para-wht) $r_1 > 0$ and $0 < r_2 \leq 2$.
$k^{\text{comp}}([\boldsymbol{q}_m, m], [\boldsymbol{k}_n, n]) = \boldsymbol{q}_m^\top \boldsymbol{k}_n \cdot \exp(-r_1|m-n|^{r_2})$.

(2-para-bias) and (1-para-wht) are the settings where we put the Gaussian kernel as a bias and a weight, respectively. (3-para-bias) and (2-para-wht) generalize these settings by considering a learnable power between 0 and 2. Note we must constrain the power in (0,2]; otherwise, the function is not positive definite. See Fact 5 for details.

These composite kernel k^{comp} are plugged into the KERPLE framework, Eq. (2), and are evaluated on OpenWebText2, GitHub, and ArXiv datasets. Table 6 shows the Gaussian-like kernel is better to be a weight instead of a bias. As discussed in Appendix A.2, the Gaussian-like kernels are bounded. To some extent, this implies that the bounded positive kernel can model a weight. However, compared with section 5.2 the Gaussian-like kernels have limited advantages in extrapolation. Although the performance might be improved if the power of $\exp(-|x|^p)$ is relaxed from $p = 2$ to $p \in (0, 2]$, still it cannot be as good as the logarithmic variant as we demonstrate in section 5.2. Therefore, while the Gaussian kernel is frequently used in the literature, we need a better class of shift-invariant kernels to tackle the length extrapolation challenge.

A.4 Position-wise Perplexity Evaluation

We follow the reviewer's suggestion and plot the position-wise perplexity with different evaluation lengths in Figure 3. The evaluation is done by measuring the loss at each position in each sequence and averaging over the sequences to get the position-wise perplexity.

First, we notice that as a general trend, tokens appear at the beginning of an evaluation sequence have much higher PPL; This is expected as these tokens have access to only a limited amount of context (early token curse). Second, we focus on Figure 3(a) and observe that all methods except rotary can extrapolate smoothly to length=4096. We can also observe that our proposed KERPLE-log variant achieves the lowest PPL with a continuously decreasing trend toward length=4096. We want to highlight the decreasing trend indicates successful extrapolation as our model is using distant token information not seen during training time (training length=512). Finally, we push the evaluation sequence length to 16384 and plot the result in Figure 3(b); This time we can see that the PPL of T5 bias starts to explode after length=5000, which also corroborates the finding (that T5 bias cannot extrapolate well) in the original ALiBi paper. In addition, we can see that the decreasing trend of our PPL is not as prominent as Figure 3(a). This is also expected as there is very likely a upper limit of how distant two relavant tokens could be. In summary, using larger context and embedding is really helpful to our model up to length=4096 on the GitHub dataset.

Table 6: **Extrapolation of Gaussian-like kernels on OpenWebText2, GitHub, and ArXiv.** All models are trained for 50k steps with training length 512 and five random seeds.

Extrp.	OpenWebText2			
	1-para-wht	2-para-wht	2-para-bias	3-para-bias
512	33.8 $\pm$ 1.1	24.8 $\pm$ 0.9	58.4 $\pm$ 71.6	26.4 $\pm$ 0.5
1024	32.5 $\pm$ 0.8	23.0 $\pm$ 0.8	88.7 $\pm$ 62.6	75.3 $\pm$ 37.8
2048	34.1 $\pm$ 0.6	22.7 $\pm$ 0.4	406 $\pm$ 101	2629 $\pm$ 4024
4096	35.6 $\pm$ 0.9	22.6 $\pm$ 0.6	2590 $\pm$ 3211	37557 $\pm$ 67936
8192	39.2 $\pm$ 1.1	23.2 $\pm$ 0.3	10829 $\pm$ 18855	189216 $\pm$ 369499
Extrp.	GitHub			
	1-para-wht	2-para-wht	2-para-bias	3-para-bias
512	7.78 $\pm$ 0.48	3.56 $\pm$ 0.23	4.08 $\pm$ 0.85	3.67 $\pm$ 0.22
1024	7.85 $\pm$ 0.40	3.19 $\pm$ 0.17	4.63 $\pm$ 0.59	4.23 $\pm$ 0.57
2048	8.08 $\pm$ 0.21	3.01 $\pm$ 0.09	18.8 $\pm$ 6.8	20.0 $\pm$ 4.8
4096	8.47 $\pm$ 0.43	2.93 $\pm$ 0.09	75.8 $\pm$ 32.2	94.0 $\pm$ 24.7
8192	9.41 $\pm$ 0.75	3.05 $\pm$ 0.20	207 $\pm$ 110	261 $\pm$ 86
Extrp.	ArXiv			
	1-para-wht	2-para-wht	2-para-bias	3-para-bias
512	10.6 $\pm$ 0.4	6.18 $\pm$ 0.25	6.73 $\pm$ 0.30	7.12 $\pm$ 1.43
1024	10.7 $\pm$ 0.2	5.73 $\pm$ 0.11	7.07 $\pm$ 0.63	7.27 $\pm$ 0.69
2048	10.8 $\pm$ 0.3	5.35 $\pm$ 0.15	20.4 $\pm$ 9.3	23.5 $\pm$ 8.4
4096	11.6 $\pm$ 0.3	5.44 $\pm$ 0.14	80.6 $\pm$ 49.4	131 $\pm$ 140
8192	12.1 $\pm$ 0.2	5.50 $\pm$ 0.27	220 $\pm$ 138	437 $\pm$ 591

Table 7: **Training Time Comparison for Gaussian-like Kernels on GitHub.**

	1-para-wht	2-para-wht	2-para-bias	3-para-bias
sec/step	0.326	0.327	0.324	0.351

Following reviewer's further suggestion, we provide the zoom-in plots in Figure 4 and 5 and simulate a model with window attention using the existing numbers. To do so, we make one small change to the original Figure 3: Instead of plotting the horizontal line signifying the minimum value of KERPLE-log until length=4096, we plot another curve, which is the KERPLE-log-windowed@512 variant. This curve is constructed by sharing the same first 512 PPLs (y-values) with KERPLE-log followed by a horizontal line equal to PPL@512 of KERPLE-log (i.e. for pos. diff.>512, y-value = KERPLE-log PPL@512). 512 is chosen because it was used as the train length. We note that PPL@512 of KERPLE-log is the lowest PPL among the first 512 positions of all model variants. We can derive several critical observations for evaluation length=4096 in Figure 4:

1. KERPLE-log lies below KERPLE-log-windowed@512, indicating its usage of more distant information than window attention: If our model is not using more information other than a fixed-window=512 tokens, the y-values after position=512 should also overlap with the new horizontal line (window attention). This is clearly not the case. In addition, the PPL of KERPLE-log continues to decrease till the end of 4096 positions (Not plateauing).
2. T5 lies below KERPLE-log-windowed@512 for most of the time and fluctuates around KERPLE-log-windowed@512 after length=3000. It is still worse than KERPLE-log.
3. ALiBi lies above KERPLE-log-windowed@512 for almost all the positions, indicating that window attention might be a better choice than ALiBi.

As for the long run comparison (evaluation length=16384) in Figure 5:

1. KERPLE-log lies below KERPLE-log-windowed@512 for most of the time, indicating its usage of more distant information than window attention.
2. The PPL of T5 explodes.
3. The PPL of ALiBi does not explode, but it is still worse than window attention, i.e. lies above KERPLE-log-windowed@512.

Although window attention is a strong baseline, our KERPLE-log is almost like a free upgrade compared to window attention: With only 24 additional learnable parameters (2 para. for each head), the almost same training speed, and the same train length=512 as window attention, our model is able to achieve lower PPLs across different positions.

Figure 3: **Position-wise Perplexity on GitHub at different Evaluation Lengths.**

(a) Evaluation Length=4096 (b) Evaluation Length=16384

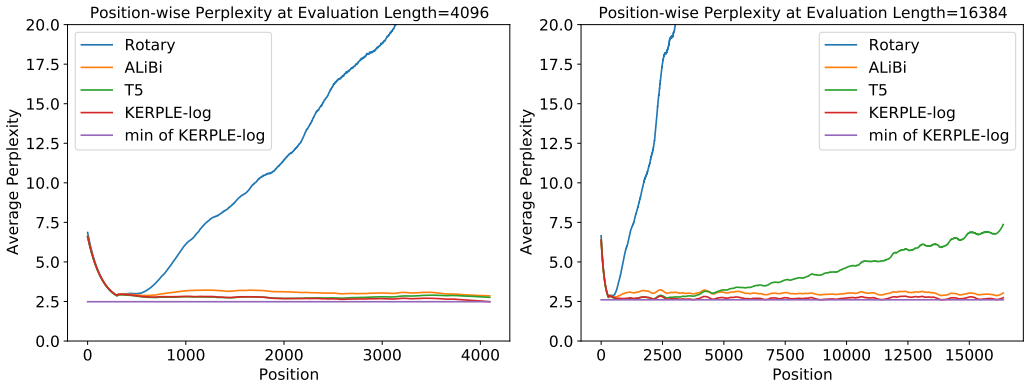
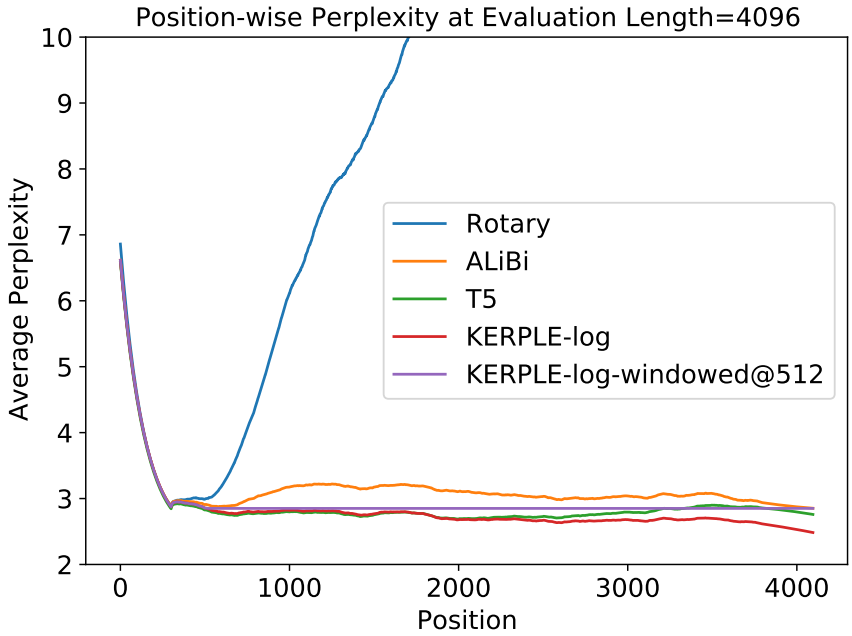


Figure 4: **Position-wise Perplexity on GitHub at Evaluation Length=4096 Compared to Window Attention@512.**



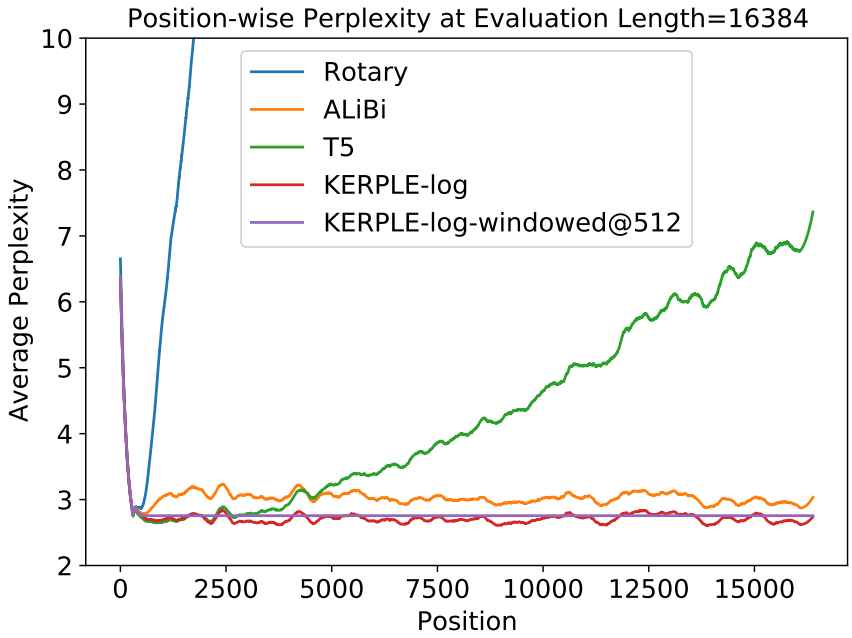
A.5 Experiments on Large Model, Longer Training Length, and Wikitext-103

In this subsection, we present additional experiments on (a) large models, (b) longer training length, and (c) Wikitext-103. Below is the summary of the experiments.

(a) The 1.3B large model is trained on a machine with two NVIDIA A100 GPU with 40 GB of memory. We adopt almost all configurations of XL GPT-NeoX[6], except that we change the train-micro-batch-size to 16, model-parallel-size to 2, seq-length to 512, and max-position-embeddings to 512. Table 8 summarizes the configurations of the 1.3B model.

[6] `https://github.com/EleutherAI/gpt-neox/blob/main/configs/XL.yml`

Figure 5: **Position-wise Perplexity on GitHub at Evaluation Length=16384 Compared to Window Attention@512.**



(b) The 162M Model with training sequence length=1024 follows the same configurations as the ones in Table 2 except that the train seq. length is changed to 1024.

(c) The Wikitext-103 model is implemented on ALiBi's GitHub[7] with exactly the same configurations (247M parameters), except that the function buffered_future_mask() at line 1011 of attention_with_linear_biases/fairseq/models/transformer.py is adapted to our KERPLE-log.

Table 8: **1.3B Model Configurations.**

# Layers	Hidden Size	# Attention Heads	Train Seq. Len.	# Trainable Params.
24	128	16	512	1.3B
Optimizer	Batch Size	Train Steps	Precision	# Trainable Params. for RPEs
Adam (lr 2e-4)	32	150,000	float16	48

Table 9 shows the results on the large model (1.3B). Compared with the small model results in Table 3, we see that T5 bias becomes weaker than KERPLE-log and ALiBi, and KERPLE-log remains stronger than ALiBi. This makes sense because T5 is more susceptible to overfitting and the large model has a higher risk of overfitting.

Again, Table 9 shows the results on long training length (1024). compared with the short training length (512) in Table 3, KERPLE-log remains better than ALiBi and T5 bias, especially on longer evaluation length. This shows the robustness of KERPLE-log over different training lengths.

Table 10 compares KERPLE-log with ALiBi using ALiBi's implementation and configurations. The results show that KERPLE-log is superior to ALiBi on Wikitext-103.

A.6 Plots of Kernel Functions

We follow reviewer's suggestion to add the plots of kernel functions including the power, log variants, and ALiBi for different heads (not doing thresholding and selecting exp(-2)) to see the differences in tail and contribution to softmax. We again use the GitHub dataset for demonstration. Please see Figure 6, 7, and 8 for details. We can see that both the ALiBi and its generalized power variant quickly reach a very negative value. In contrast, the log variant successfully discovers several kernels that

[7] `https://github.com/ofirpress/attention_with_linear_biases`

Table 9: **Perplexity Comparison for Large Models (1.3B) and Long Training Length (1024) on GitHub, ArXiv, OpenWebText2.** Due to the time constraint and limited computing resources, we are not able to obtain the numbers for the large model (1.3B) on OpenWebText2 for now. All models are trained with five random seeds. $x^\dagger$ means our log variant is statistically significantly *better* than x. The test used is paired two-sided t-test with $\alpha = 0.05$.

162M Model. Train length, steps=1024, 50k.				1.3B Model. Train length, steps=512, 150k.		
Extrp.	GitHub			GitHub		
	KERPLE-log	ALiBi	T5 bias	KERPLE-log	ALiBi	T5 bias
512	-	-	-	$\mathbf{2.88 \pm 0.11}$	$\mathbf{2.88 \pm 0.11}$	$2.93 \pm 0.11^\dagger$
1024	2.83 ± 0.16	$2.84 \pm 0.16^\dagger$	$\mathbf{2.81 \pm 0.16}$	$\mathbf{2.60 \pm 0.12}$	$2.62 \pm 0.11^\dagger$	$2.64 \pm 0.11^\dagger$
2048	2.70 ± 0.07	$2.82 \pm 0.07^\dagger$	$\mathbf{2.68 \pm 0.07}$	$\mathbf{2.44 \pm 0.05}$	$2.58 \pm 0.05^\dagger$	$2.47 \pm 0.07^\dagger$
4096	$\mathbf{2.53 \pm 0.04}$	$2.77 \pm 0.06^\dagger$	2.54 ± 0.04	$\mathbf{2.46 \pm 0.11}$	$2.65 \pm 0.12^\dagger$	2.49 ± 0.12
8192	$\mathbf{2.42 \pm 0.03}$	$2.74 \pm 0.02^\dagger$	$2.57 \pm 0.06^\dagger$	$\mathbf{2.44 \pm 0.13}$	$2.57 \pm 0.13^\dagger$	$2.57 \pm 0.13^\dagger$
16384	$\mathbf{2.48 \pm 0.11}$	$2.80 \pm 0.11^\dagger$	$3.10 \pm 0.34^\dagger$	$\mathbf{2.60 \pm 0.07}$	2.61 ± 0.07	$3.16 \pm 0.35^\dagger$
Extrp.	ArXiv			ArXiv		
	KERPLE-log	ALiBi	T5 bias	KERPLE-log	ALiBi	T5 bias
512	-	-	-	$\mathbf{5.56 \pm 0.15}$	5.58 ± 0.16	$5.62 \pm 0.15^\dagger$
1024	5.23 ± 0.09	5.26 ± 0.09	$\mathbf{5.20 \pm 0.10}$	$\mathbf{4.87 \pm 0.07}$	$4.94 \pm 0.07^\dagger$	$4.92 \pm 0.06^\dagger$
2048	4.76 ± 0.12	$4.98 \pm 0.18^\dagger$	$\mathbf{4.74 \pm 0.12}$	$\mathbf{4.50 \pm 0.16}$	$4.87 \pm 0.17^\dagger$	$4.55 \pm 0.16^\dagger$
4096	$\mathbf{4.75 \pm 0.10}$	$5.31 \pm 0.13^\dagger$	4.97 ± 0.27	$\mathbf{4.45 \pm 0.06}$	$4.97 \pm 0.13^\dagger$	$4.53 \pm 0.08^\dagger$
8192	$\mathbf{4.54 \pm 0.10}$	$5.25 \pm 0.15^\dagger$	$6.55 \pm 0.97^\dagger$	$\mathbf{4.47 \pm 0.20}$	$4.94 \pm 0.16^\dagger$	$4.65 \pm 0.15^\dagger$
16384	$\mathbf{4.62 \pm 0.15}$	$5.35 \pm 0.19^\dagger$	$16.0 \pm 4.77^\dagger$	$\mathbf{4.65 \pm 0.24}$	4.94 ± 0.07	$5.25 \pm 0.26^\dagger$
Extrp.	OpenWebText2					
	KERPLE-log	ALiBi	T5 bias			
512	-	-	-			
1024	19.2 ± 0.1	19.3 ± 0.2	$\mathbf{19.1 \pm 0.1}$			
2048	19.3 ± 0.2	19.5 ± 0.1	$\mathbf{19.2 \pm 0.2}$			
4096	$\mathbf{18.6 \pm 0.3}$	$19.0 \pm 0.3^\dagger$	$19.2 \pm 0.4^\dagger$			
8192	$\mathbf{18.7 \pm 0.5}$	$19.3 \pm 0.4^\dagger$	$24.0 \pm 1.1^\dagger$			
16384	$\mathbf{18.8 \pm 0.5}$	$19.2 \pm 0.3^\dagger$	$50.8 \pm 6.5^\dagger$			

Table 10: **Perplexity Comparison on Wikitext-103.** To ensure fair comparison, the model (247M) is trained on ALiBi's codebase with exactly the same configurations except for the positional embeddings. The results show that KERPLE-log is superior to ALiBi on Wikitext-103.

	train length 512					train length 2048	
Extrp. length	512	1024	1536	2048	3072	2048	3072
ALiBi	19.73	18.81	18.50	18.48	18.40	17.91	17.64
KERPLE-log	**19.69**	**18.76**	**18.37**	**18.29**	**18.24**	**17.84**	**17.56**

remain flat, effectively extending the window attention. This corroborates our previous observation that KERPLE-log can utilize more distant token information: Flat kernel functions permit the usage of more distant token information in that distant information would not be masked out by the implicit window attention effect in ALiBi.

A.7 The Choice of codebase and Hyperparameters

We adopt almost all the hyperparameters (except batch size to fit in our GPU) and all implementations of the T5 bias, ALiBi, Rotary, and Sinusoidal baselines from the GPT-NeoX codebase. To ensure fair comparisons, we did not fine-tune hyper-parameters for KERPLE. The datasets we used are exactly the same as the ones released with the GPT-NeoX codebase. We just ran their prepare_data.py script to automatically download and parse the datasets. All our code was uploaded with the submission on openreview, and `https://github.com/EleutherAI/gpt-neox` is the original GitHub repository. As a side note, we chose this codebase and adopt their parameter settings because it is built by

Figure 6: Kernel Functions of Different of the Log Variant.

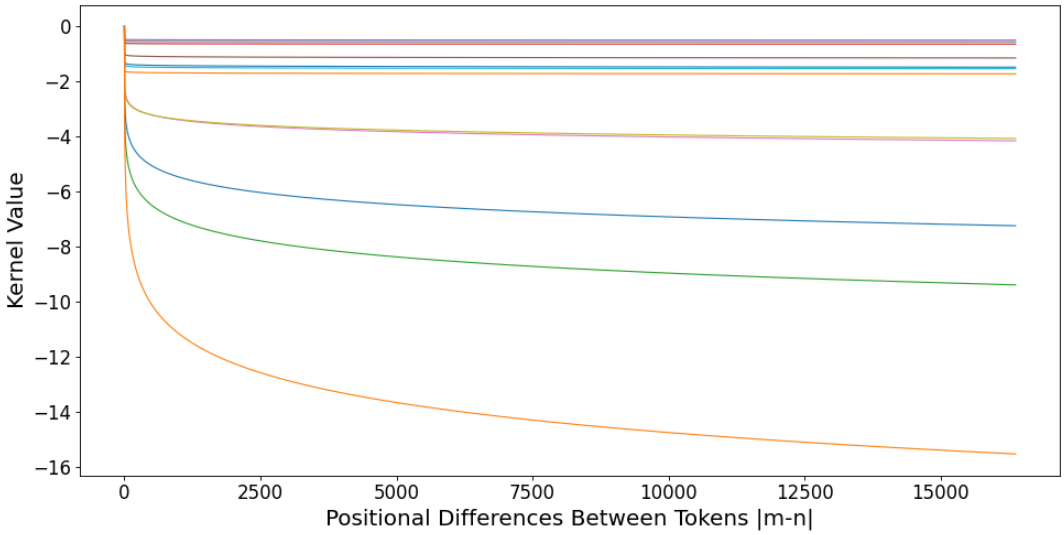
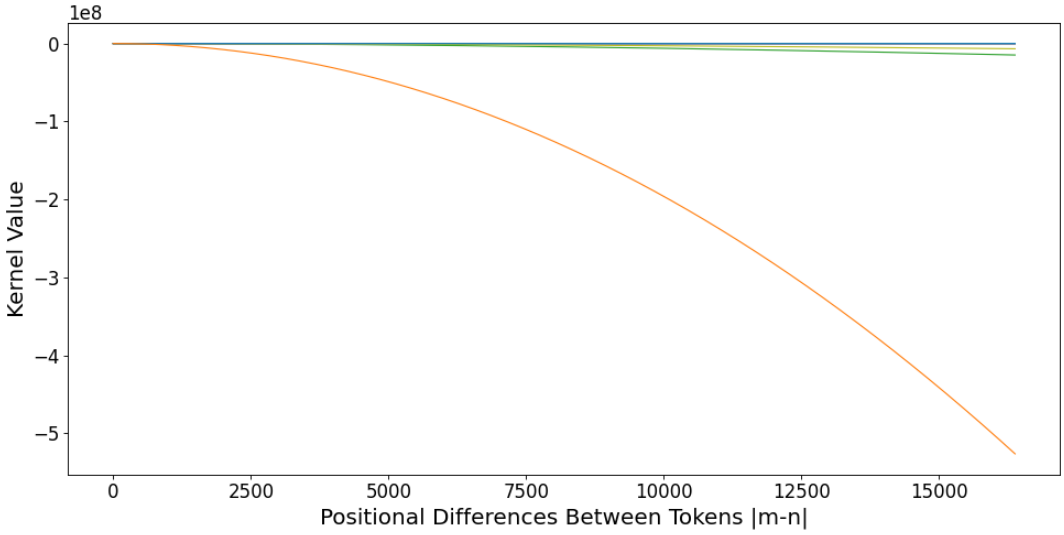


Figure 7: Kernel Functions of Different of the Power Variant. Note the y-axis should be multiplied by $1e8$, which is a very negative value.



EleutherAI, which is a well-known and truly non-profit group of researchers publishing various famous pretrained models for the academia including GPT-J-6B and GPT-NeoX-20B.

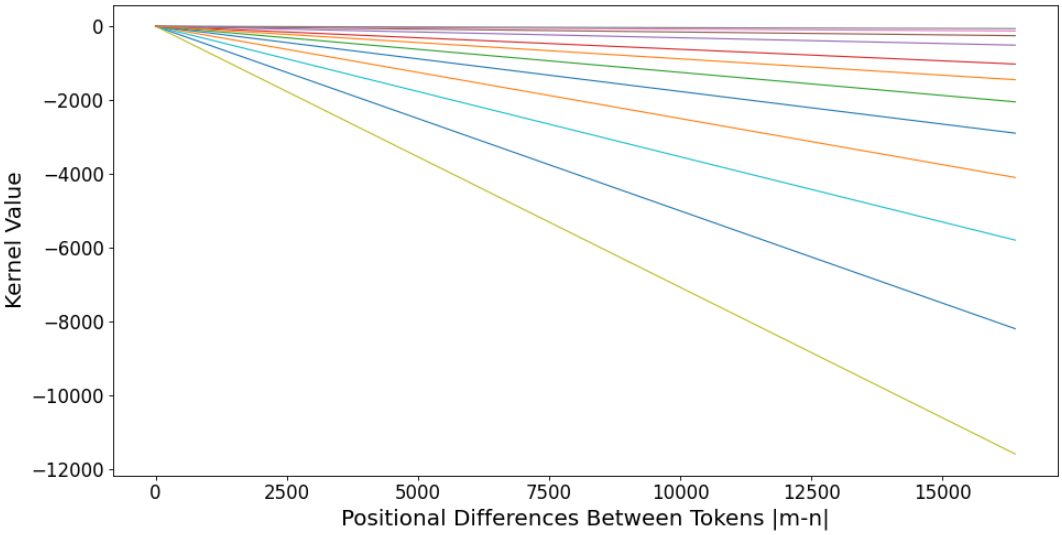



Figure 8: Kernel Functions of Different of ALiBi.