Flexformer: Flexible Linear Transformer with Learnable Attention Kernel

Haoran Zhang[1] **Zisu Dong**[2] **Feng Zhou**[1*]
[1]Center for Applied Statistics and School of Statistics, Renmin University of China
[2]School of Information Science and Technology, Beijing University of Technology
{zhrrhz2002, dongzisu}@163.com, feng.zhou@ruc.edu.cn

Abstract

Transformer models rely on attention mechanism to capture long-range dependencies but suffer from quadratic complexity, limiting their scalability to long sequences. Kernel-based linear attention reduces this complexity but typically relies on fixed or weakly learnable kernels, restricting expressiveness and performance. In this work, we propose Flexformer, a flexible linear Transformer that learns attention kernels in a fully data-driven manner. Flexformer builds on random Fourier feature-based linear attention and treats spectral frequencies as trainable parameters, enabling the model to learn a broad family of attention kernels. We develop both stationary and nonstationary variants, with the latter offering strictly greater expressiveness. Extensive experiments on language modeling and sequence classification demonstrate that Flexformer consistently outperforms baselines. Moreover, Flexformer can be effectively distilled from pretrained Transformers to recover softmax attention and exhibits strong kernel transferability across domains, achieving both high efficiency and competitive performance on long-sequence tasks. Our code is available at https://github.com/Zhanghaoran2002/Flexformer.

1 Introduction

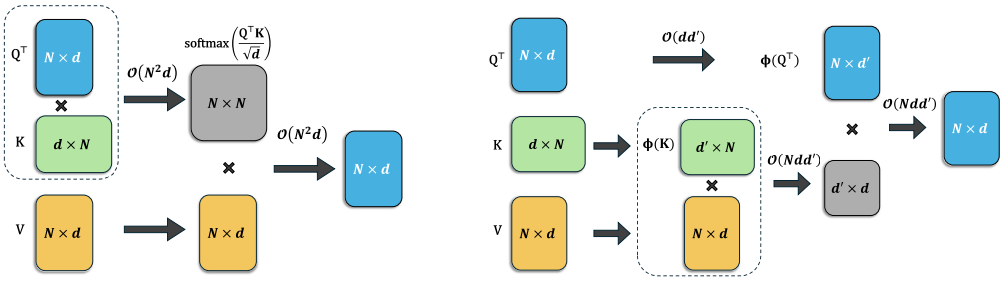
The Transformer [38] architecture has achieved remarkable success in machine translation tasks since its original proposal. Subsequently, numerous variants of Transformer have been extensively applied across prominent domains within artificial intelligence, including natural language processing [9, 18, 5], computer vision [10, 20], time series analysis [44, 25, 19], and audio processing [33, 11]. The most critical component in these Transformer-based models is the dot-product attention mechanism. It operates by computing pairwise token affinities across the entire input sequence, thereby enabling each token to dynamically aggregate the most relevant information pertaining to itself, while effectively modeling long-range dependencies within the sequence. Unlike sequential RNNs [34] processing tokens step-by-step, Transformer attention computes representations for all positions in parallel, maximizing computational hardware utilization.

This effective attention mechanism, however, suffers from efficiency limitations: it computes attention weights for all pairs of positions in the sequence, leading to both time and space complexity that grow quadratically with the sequence length. To address this issue, numerous studies have attempted to optimize the computation of the attention mechanism. These redesigned or optimized attention computation methods should effectively reduce time and space complexity while achieving performance that comparable or even surpasses that of softmax attention.

Some works are based on sparse attention [15, 32, 42], which modifies the attention patterns by restricting the attention range of each token to focus only on a subset of critical tokens. However, these attention patterns are not rigorously validated and are often based on empirically derived

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Softmax attention.

(b) Kernel-based linear attention.

Figure 1: Comparison between softmax attention and kernel-based linear attention. Softmax attention requires $\mathcal{O}(N^2d)$ time complexity and $\mathcal{O}(N^2 + Nd)$ space complexity, whereas kernel-based linear attention achieves $\mathcal{O}(Ndd')$ time complexity and $\mathcal{O}(Nd + Nd' + dd')$ space complexity, where N, d, d' denote input sequence length, hidden dimension, and feature map dimension respectively. When $N \gg d$ and $N \gg d'$, linear attention achieves significant savings in both time and space.

patterns, making it difficult to guarantee their consistently superior performance across diverse data domains. The other category is kernel-based linear attention [13, 7, 29], which represents the process of computing dot products and applying softmax using kernels. The kernel can be written as an inner product of feature maps. By rearranging the order of matrix multiplications, the quadratic complexity of attention can be reduced to linear complexity. Figure 1 illustrates the difference of the computational processes between kernel-based linear attention and softmax attention.

Here, we focus on the kernel-based linear attention whose core lies in constructing an effective feature map. While numerous methods aim to recover softmax attention, there is still no proof that softmax is invariably optimal under all circumstances. Therefore, a more effective approach is to apply learnable kernels that are directly learned from data. However, how to construct sufficiently flexible learnable kernels that guarantee strong performance remains an open problem. Hedgehog [43] and Polaformer [22] empirically exploit the low-entropy property of softmax attention by combining learnable mappings with spiky functions such as exponentials and power functions. However, these methods do not guarantee that the learned kernel is sufficiently expressive to encompass a wide variety of kernels, which can compromise their performance.

In this work, we propose a **Flex**ible Linear Trans**former** with learnable kernel (**Flexformer**). Flexformer extends kernel-based linear attention with random Fourier features by making the feature map fully learnable. In contrast to existing learnable kernel methods, Flexformer is able to model a substantially broader and more expressive family of kernels. Specifically, spectral representation theory states that a kernel admits an inverse Fourier transform representation in terms of its spectral measure, which can be approximated through Monte Carlo sampling. In Flexformer, instead of sampling frequencies from a fixed spectral density, we treat the frequencies as learnable parameters and optimize them directly from data, thereby enabling the attention kernel to be learned in a data-driven manner. We propose both stationary and nonstationary versions of Flexformer, with the nonstationary formulation being strictly more expressive. Theoretically, we can show that the kernel family of Flexformer contains the softmax kernel. This guarantees that Flexformer can, at a minimum, learn a softmax attention, and potentially learn attention patterns that outperform softmax. Flexformer can be trained either from scratch or by distilling [43] from a pretrained Transformer. We conduct extensive experiments on multiple benchmarks, including language modeling and classification tasks, demonstrating Flexformer's superior performance over existing baselines, as well as its strong capability to recover the softmax kernel and effectively transfer the learned kernel. As a variant of linear attention, it significantly improves efficiency and reduces memory consumption when processing extremely long sequences. Specifically, our contributions are summarized as follows:

(1) We propose Flexformer, which leverages learnable attention kernels constructed via random Fourier features, achieving high expressiveness while preserving linear time and space complexity w.r.t. sequence length.

(2) We evaluate Flexformer through from-scratch training on language modeling and sequence classification tasks. The results show that Flexformer consistently outperforms other linear-attention baselines, highlighting its strong effectiveness and efficiency.

(3) Flexformer can faithfully approximate softmax attention through attention weight distillation, enabling efficient linear-attention replacements with preserved performance, and demonstrates strong kernel transferability in cross-domain settings.

2 Preliminaries

In this section, we introduce the background knowledge of self-attention mechanism and kernel-based linear attention.

2.1 Self-attention Mechanism

Given an input sequence $\mathbf{X} = [\mathrm{X}_1, \cdots, \mathrm{X}_N]^\top \in \mathbb{R}^{N \times d}$, where N and d denote the input length and embedding dimension, respectively. Each token X_i is projected through linear maps to generate the query, key, and value tensors, formally expressed as $\mathrm{Q}_i = \mathrm{W}_Q \mathrm{X}_i, \mathrm{K}_i = \mathrm{W}_K \mathrm{X}_i, \mathrm{V}_i = \mathrm{W}_V \mathrm{X}_i, i = 1, \cdots, N$. For each query tensor, a similarity measure is computed with all key tensors. These similarity scores are then normalized to form attention weights, which are used to perform a weighted summation over the corresponding value tensors, thereby generating context-aware representations. Formally, the output of the self-attention module for the i-th token can be expressed as:

$$\mathrm{O}_i = \frac{\sum_{j=1}^{N} \mathrm{sim}(\mathrm{Q}_i, \mathrm{K}_j)\mathrm{V}_j}{\sum_{j=1}^{N} \mathrm{sim}(\mathrm{Q}_i, \mathrm{K}_j)}, \tag{1}$$

where $\mathrm{sim}(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ denotes the similarity measure function. In vanilla Transformer [38] with softmax attention, $\mathrm{sim}(\mathrm{Q}_i, \mathrm{K}_j)$ is defined as $\mathrm{sim}(\mathrm{Q}_i, \mathrm{K}_j) = \exp\left(\frac{\mathrm{Q}_i^\top \mathrm{K}_j}{\sqrt{d}}\right)$.

2.2 Kernel-based Linear Attention

If we directly follow the computation order in Equation (1) by first calculating the similarity between queries and keys at all pairwise positions to obtain a $N \times N$ attention weight matrix, and then multiplying with values, it would result in a time complexity of $\mathcal{O}(N^2 d)$ and a space complexity of $\mathcal{O}(N^2 + Nd)$, leading to substantial computational and memory overhead as N increases.

To address the aforementioned issues, we assume an attention kernel $k(\mathbf{x}, \mathbf{y})$ with a range of $\mathbb{R}_+$ as the similarity function, and let $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$, where d' is the dimension of the kernel-induced feature space, denote the corresponding feature map, such that:

$$k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^\top \phi(\mathbf{y}). \tag{2}$$

Then the computation of the self-attention in Equation (1) can be expressed as:

$$\mathrm{O}_i^\top = \frac{\sum_{j=1}^{N} \phi(\mathrm{Q}_i)^\top \phi(\mathrm{K}_j)\mathrm{V}_j^\top}{\sum_{j=1}^{N} \phi(\mathrm{Q}_i)^\top \phi(\mathrm{K}_j)} = \frac{\phi(\mathrm{Q}_i)^\top \sum_{j=1}^{N} \phi(\mathrm{K}_j)\mathrm{V}_j^\top}{\phi(\mathrm{Q}_i)^\top \sum_{j=1}^{N} \phi(\mathrm{K}_j)}. \tag{3}$$

This approach avoids explicitly computing the pairwise similarity between any query and key. Instead, we first precompute $\sum_{j=1}^{N} \phi(\mathrm{K}_i)\mathrm{V}_j^\top$ and $\sum_{j=1}^{N} \phi(\mathrm{K}_j)$, which are shared across all positional queries, and then multiply them with each positional query Q_i. Thus, the time complexity of this approach is $\mathcal{O}(Ndd')$ and the space complexity is $\mathcal{O}(Nd + Nd' + dd')$, achieving linear scaling w.r.t. the sequence length. When $N \gg d, d'$, it demonstrates significant efficiency improvements and memory savings compared to standard softmax attention.

3 Methodology

In this section, we first review how softmax kernel can be approximated using random Fourier features. We then extend this formulation to a fully learnable kernel, resulting in the proposed Flexformer.

3.1 Softmax Kernel with Random Fourier Features

In softmax attention, the similarity measure can be expressed as the following kernel. We hereafter refer to this as the softmax kernel:

$$k_{\text{SM}}(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{\mathbf{x}^\top \mathbf{y}}{\sqrt{d}}\right).$$

Through Taylor expansion, we can derive feature map ϕ satisfying Equation (2). Unfortunately, the dimensionality of ϕ is infinite [35]. Therefore, we need to seek a finite-dimensional ϕ such that the softmax kernel can be effectively approximated. The softmax kernel can be decomposed in the following manner:

$$k_{\text{SM}}(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{||\mathbf{x}||^2 + ||\mathbf{y}||^2 - ||\mathbf{x}-\mathbf{y}||^2}{2\sqrt{d}}\right) = \exp\left(\frac{||\mathbf{x}||^2}{2\sqrt{d}}\right)\exp\left(-\frac{||\mathbf{x}-\mathbf{y}||^2}{2\sqrt{d}}\right)\exp\left(\frac{||\mathbf{y}||^2}{2\sqrt{d}}\right). \tag{4}$$

The blue terms depend exclusively on vector norms, enabling all positions to potentially attend strongly to certain keys with large norm magnitudes at critical locations, and they are crucial for preserving the spikiness of the softmax. The red term, which focuses exclusively on a specific similarity between $\mathbf{x}$ and $\mathbf{y}$, is essentially a *Gaussian kernel* with bandwidth $\sigma^2 = \sqrt{d}$. As a bounded, continuous, and positive definite stationary (a.k.a. translation-invariant) kernel, it can be approximated by the random Fourier features [31].

Theorem 3.1. *[4] A stationary kernel $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x}-\mathbf{y}) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a bounded, continuous, and positive definite if and only if it can be represented as:*

$$k(\mathbf{x}-\mathbf{y}) = \int_{\mathbb{R}^d} \exp(i(\omega^\top(\mathbf{x}-\mathbf{y}))d\mu(\omega),$$

where $\mu(\omega)$ is a bounded non-negative measure associated to the spectral density $p(\omega) = \frac{\mu(\omega)}{\mu(\mathbb{R}^d)}$.

Let $\sigma^2 = \mu(\mathbb{R}^d)$ denote the total measure of the space. Then Theorem 3.1 can be expressed in the following expectation form, which can be approximated by Monte Carlo:

$$\begin{aligned} k(\mathbf{x}-\mathbf{y}) =& \sigma^2 \mathbb{E}_{p(\omega)}(\cos(\omega^\top(\mathbf{x}-\mathbf{y})) + i\sin(\omega^\top(\mathbf{x}-\mathbf{y}))) \\ =& \sigma^2 \mathbb{E}_{p(\omega)}(\cos\omega^\top\mathbf{x}\cos\omega^\top\mathbf{y} + \sin\omega^\top\mathbf{x}\sin\omega^\top\mathbf{y}) \\ \approx& \frac{\sigma^2}{n}\sum_{i=1}^{n}(\cos\omega_i^\top\mathbf{x}\cos\omega_i^\top\mathbf{y} + \sin\omega_i^\top\mathbf{x}\sin\omega_i^\top\mathbf{y}) \\ =& \phi_n(\mathbf{x})^\top\phi_n(\mathbf{y}), \end{aligned} \tag{5}$$

where $\phi_n(\mathbf{x}) = \frac{\sigma}{\sqrt{n}}[\cos\omega_1^\top\mathbf{x}, \cdots, \cos\omega_n^\top\mathbf{x}, \sin\omega_1^\top\mathbf{x}, \cdots, \sin\omega_n^\top\mathbf{x}]^\top$. $\phi_n(\mathbf{x})$ is called Fourier feature, $\{\omega_i\}_{i=1}^n$ are i.i.d. frequencies from $p(\omega)$ and n denotes the number of frequency samples. The second equality in the derivation holds because we constrain the range of $k(\mathbf{x}-\mathbf{y})$ to $\mathbb{R}$, which enforces symmetry in the measure p such that $p(\omega) = p(-\omega)$, consequently giving $\mathbb{E}_{p(\omega)}(i\sin(\omega^\top(\mathbf{x}-\mathbf{y}))) = 0$.

Substituting Equation (5) into the red term of Equation (4) yields a random Fourier feature representation of the softmax kernel:

$$k_{\text{SM}}(\mathbf{x}, \mathbf{y}) \approx \tilde{\phi}_n(\mathbf{x})^\top\tilde{\phi}_n(\mathbf{y}),$$

where $\tilde{\phi}_n(\mathbf{x}) = \exp\left(\frac{||\mathbf{x}||^2}{2\sqrt{d}}\right)\phi_n(\mathbf{x})$, $\{\omega_i\}_{i=1}^n$ are i.i.d. frequencies from a *Gaussian spectral density*. Based on the above representation, the kernel-based linear attention framework can be employed to reduce the complexity of softmax attention to linear [28].

The above approach relies on a crucial assumption: it implicitly treats softmax attention as optimal, and therefore fixes the similarity function to the softmax kernel, which is non-learnable. Once the hidden dimension d is specified, the softmax kernel—and consequently its corresponding Gaussian spectral density $p(\omega)$—is fully determined, rendering the resulting attention patterns non-learnable. However, a growing body of work [43, 22] has shown that softmax attention is not optimal in many scenarios, and that the attention kernel should instead be learned in a data-driven manner. This motivates extending the above framework to the learnable attention kernel.

3.2 Flexformer

In this section, we first extend the fixed, non-learnable attention kernel to a learnable one, allowing attention patterns to adapt to data and yielding Flexformer. Then we further extend this formulation from stationary kernels to nonstationary kernels, resulting in a nonstationary variant of Flexformer.

Flexformer with learnable stationary kernel. As stated in Theorem 3.1, a stationary kernel $k(\mathbf{x}-\mathbf{y})$ is in one-to-one correspondence with its spectral density $p(\omega)$. Consequently, learning a kernel from data is equivalent to learning its spectral density from data. Therefore, in this work, we directly treat the frequencies $\{\omega_i\}_{i=1}^n$ as learnable parameters and optimize them end-to-end. This is equivalent to learning an optimal spectral density $p(\omega)$, and hence learning an optimal attention kernel. Similar techniques have been extensively explored in the Gaussian process literature [17, 36]. As Theorem 3.1 holds for all stationary kernels satisfying certain conditions, the kernels learned in this manner are sufficiently flexible. We refer to this method as **Flexformer$_\mathbf{s}$**, where the subscript s denotes the stationary kernel.

It is worth noting that when the learned frequencies $\{\omega_i\}_{i=1}^n$ follow a Gaussian distribution, this yields an approximation of the softmax kernel in Equation (4), analogous to Random Feature Attention [28]. This observation implies that the kernel family induced by Flexformer strictly contains the unbiased estimate of the softmax kernel. As a result, Flexformer is guaranteed to recover softmax attention closely, while also being capable of learning potentially better attention patterns directly from data.

Flexformer with learnable nonstationary kernel. The red term in the softmax kernel in Equation (4) is stationary, which motivates the use of Theorem 3.1 to derive Flexformer$_\mathrm{s}$. However, a stationary kernel implies that its value depends only on the difference $\mathbf{x}-\mathbf{y}$, which limits flexibility. An immediate and natural extension is to replace this stationary kernel with a nonstationary one, thereby enabling the model to learn more diverse attention patterns and adapt more flexibly to the data. Since Theorem 3.1 only characterizes stationary kernels, it cannot be directly applied here. Instead, we resort to the Yaglom theorem, which generalizes Bochner's result to nonstationary kernels.

Theorem 3.2. *[40] A nonstationary kernel $k(\mathbf{x},\mathbf{y}):\mathbb{R}^d\times\mathbb{R}^d\to\mathbb{R}$ is a bounded, continuous, and positive definite if and only if it can be represented as:*

$$k(\mathbf{x},\mathbf{y})=\int_{\mathbb{R}^d\times\mathbb{R}^d}\exp(i(\omega_1^\top\mathbf{x}-\omega_2^\top\mathbf{y}))d\mu(\omega_1,\omega_2),$$

where $\mu(\omega_1,\omega_2)$ is a Lebesgue-Stieltjes measure associated to some bounded positive semi-definite spectral density $p(\omega_1,\omega_2)=\frac{\mu(\omega_1,\omega_2)}{\mu(\mathbb{R}^d,\mathbb{R}^d)}$.

In fact, the red term in the softmax kernel in Equation (4) is indeed a positive definite kernel. Therefore, after extending it to the nonstationary setting, we continue to assume that it remains positive definite. When approximating the kernel in Theorem 3.2 via Monte Carlo sampling methods, to ensure the positive semi-definiteness of the spectral density $p(\boldsymbol{\omega}_1,\boldsymbol{\omega}_2)$, we first assume that it is symmetrized by an arbitrary joint probability density:

$$p(\omega_1,\omega_2)=\frac{1}{4}(g(\omega_1,\omega_1)+g(\omega_2,\omega_2)+g(\omega_1,\omega_2)+g(\omega_2,\omega_1)).$$

Let $\sigma^2=\mu(\mathbb{R}^d,\mathbb{R}^d)$ denote the total measure of the space. Then Theorem 3.2 can be expressed in the following expectation form, which can be approximated by Monte Carlo:

$$\begin{aligned}
k(\mathbf{x},\mathbf{y})=&\frac{\sigma^2}{4}\mathbb{E}_{g(\omega_1,\omega_2)}((\cos\omega_1^\top\mathbf{x}+\cos\omega_2^\top\mathbf{x})(\cos\omega_1^\top\mathbf{y}+\cos\omega_2^\top\mathbf{y})\\
&+(\sin\omega_1^\top\mathbf{x}+\sin\omega_2^\top\mathbf{x})(\sin\omega_1^\top\mathbf{y}+\sin\omega_2^\top\mathbf{y}))\\
\approx&\frac{\sigma^2}{4n}\sum_{i=1}^n((\cos\omega_{1i}^\top\mathbf{x}+\cos\omega_{2i}^\top\mathbf{x})(\cos\omega_{1i}^\top\mathbf{y}+\cos\omega_{2i}^\top\mathbf{y})\\
&+(\sin\omega_{1i}^\top\mathbf{x}+\sin\omega_{2i}^\top\mathbf{x})(\sin\omega_{1i}^\top\mathbf{y}+\sin\omega_{2i}^\top\mathbf{y}))\\
=&\frac{\sigma^2}{4n}\sum_{i=1}^n(\cos\omega_{1i}'^\top\mathbf{x}\cos\omega_{2i}'^\top\mathbf{x}\cos\omega_{1i}'^\top\mathbf{y}\cos\omega_{2i}'^\top\mathbf{y}+\sin\omega_{1i}'^\top\mathbf{x}\cos\omega_{2i}'^\top\mathbf{x}\sin\omega_{1i}'^\top\mathbf{y}\cos\omega_{2i}'^\top\mathbf{y})\\
=&\phi_n(\mathbf{x})^\top\phi_n(\mathbf{y}),
\end{aligned}\tag{6}$$

Table 1: Performance comparison on LRA benchmarks. All reported metrics represent test accuracy (%), where higher values are better. The best results are in **bold**, and the second best in underlined.

Model	ListOps	Text	Retrieval	Image	Pathfinder	Average
Transformer [38]	37.23	65.66	64.49	41.89	73.46	56.55
Reformer [15]	36.62	63.52	57.60	40.06	73.61	54.28
Longformer [3]	36.46	63.09	59.24	41.23	72.44	54.49
Linear Trans. [13]	17.42	62.64	58.92	43.40	75.11	51.49
BigBird [42]	36.88	65.34	64.23	41.93	73.58	56.39
Performer [7]	16.97	63.67	55.17	41.23	75.38	50.48
RFA [28]	17.20	64.89	58.76	41.00	74.51	51.27
Skyformer [6]	36.71	65.51	68.05	41.36	74.18	57.16
Cosformer [29]	**37.44**	65.16	64.04	43.56	72.32	56.50
Hedgehog [43]	37.02	64.10	68.67	41.64	75.89	57.47
Polaformer [22]	36.93	**69.65**	65.39	40.95	73.26	57.23
Flexformer$_{\text{s}}$ (Ours)	37.09	67.38	71.55	43.39	76.15	59.11
Flexformer$_{\text{n}}$ (Ours)	37.18	65.72	**73.94**	**45.42**	**77.66**	**59.99**

where $\phi_n(\mathbf{x}) = \frac{\sigma}{2\sqrt{n}}[\cos\omega_{11}'^{\top}\mathbf{x}\cos\omega_{21}'^{\top}\mathbf{x}, \cdots, \cos\omega_{1n}'^{\top}\mathbf{x}\cos\omega_{2n}'^{\top}\mathbf{x}, \sin\omega_{11}'^{\top}\mathbf{x}\cos\omega_{21}'^{\top}\mathbf{x}, \cdots, \sin\omega_{1n}'^{\top}\mathbf{x}\cos\omega_{2n}'^{\top}\mathbf{x}]^{\top}$, $\{(\omega_{1i}, \omega_{2i})\}_{i=1}^{n}$ are i.i.d. frequencies from $g(\omega_1, \omega_2)$, n denotes the number of frequencies, and $\omega_{1i}' = \frac{\omega_{1i}+\omega_{2i}}{2}, \omega_{2i}' = \frac{\omega_{1i}-\omega_{2i}}{2}$. Substituting Equation (6) into the red term of Equation (4) yields a random Fourier feature representation of the attention kernel. For the blue term $\exp\left(\frac{\|\mathbf{x}\|^2}{2\sqrt{d}}\right)$, we preserve its functional form while replacing the fixed scaling factor $2\sqrt{d}$ with a learnable parameter $\exp(\tau)$. This leads to the following attention kernel approximation:

$$k(\mathbf{x}, \mathbf{y}) \approx \tilde{\phi}_n(\mathbf{x})^{\top}\tilde{\phi}_n(\mathbf{y}),$$

where $\tilde{\phi}_n(\mathbf{x}) = \exp\left(\frac{\|\mathbf{x}\|^2}{\exp(\tau)}\right)\phi_n(\mathbf{x})$. Both scaling parameter τ and spectral pairs $\{(\omega_{1i}, \omega_{2i})\}_{i=1}^{n}$ are treated as learnable parameters. We refer to this nonstationary variant as **Flexformer$_{\text{n}}$**, where the subscript n denotes the nonstationary kernel. We provide an analysis of the time and space complexity, as well as the additional parameter overhead of Flexformer, in Appendix B.

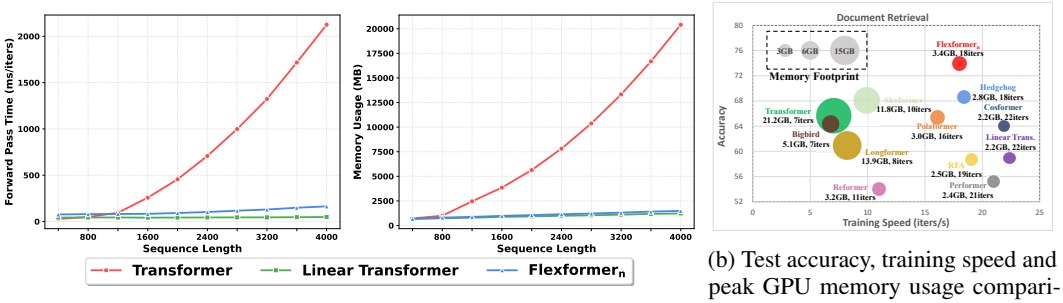
4 Experiments

To evaluate the effectiveness and efficiency of Flexformer as a linear-attention variant, we conduct extensive experiments on long-context classification, autoregressive language modeling, softmax attention recovery, and cross-domain kernel transfer. All results report the mean over three runs, with full results including standard deviations provided in Appendix C. Additional hyperparameter sensitivity analyses, and experimental results are deferred to Appendix D.

4.1 Long Sequence Classification

To evaluate Flexformer's ability to model long sequences and reduce cost, we conduct long-sequence classification experiments on the Long Range Arena (LRA) benchmark [37]. LRA is designed to assess both the effectiveness and efficiency of sequence models under long-context settings. It includes tasks such as Long ListOps [24], byte-level text classification [21], byte-level document retrieval [30], image classification on sequences of pixels [16], and Pathfinder [12]. The input sequence lengths in these tasks range from 1K to 4K. We adopt the same hyperparameter configurations as those used in the official open-source implementation.

We present in Table 1 a comprehensive performance comparison between Flexformer, the vanilla Transformer, several classical Transformer variants, and a range of existing linear attention methods. Early approaches that rely on random Fourier features to obtain unbiased estimators of the softmax kernel, such as RFA and Performer, substantially underperform the vanilla Transformer, particularly on challenging tasks such as ListOps and document retrieval. Methods based on learnable kernels, including Hedgehog and Polaformer, exhibit strong performance on certain specific tasks, but their



(a) Forward Pass Time & GPU Memory Usage vs. Sequence Length.

(b) Test accuracy, training speed and peak GPU memory usage comparison on document retrieval.

Figure 2: Comparison of runtime and memory usage across models. As shown in (a), Flexformer scales linearly with sequence length in both time and memory. Together with (b), the results show that Flexformer achieves higher efficiency and lower memory usage than other baselines.

effectiveness does not consistently generalize across all benchmarks. In contrast, the proposed Flexformer, benefiting from its flexible and expressive feature map representation, achieves consistently strong performance across all tasks. In particular, Flexformer$_{\text{n}}$ attains the best results on document retrieval, image classification, and Pathfinder, while remaining highly competitive on the remaining tasks. Overall, it achieves the highest average accuracy, corresponding to a 4.4% relative improvement over the best existing linear attention method. The stationary variant, Flexformer$_{\text{s}}$, ranks second overall and also outperforms all prior linear attention approaches.

We further compare the computational time and memory usage of different Transformer variants on the LRA benchmark in Figure 2. In Figure 2a, we evaluate the forward-pass computation time and memory consumption of the vanilla Transformer, Linear Transformer, and our proposed Flexformer$_{\text{n}}$ under same hyperparameter configurations on the document retrieval task using input sequences of varying lengths. As expected, both the computation time and memory usage of the vanilla Transformer scale quadratically with the sequence length, resulting in a rapid increase as the input length grows. In contrast, Flexformer$_{\text{n}}$ exhibits linear complexity with respect to sequence length, closely matching the scaling behavior of the Linear Transformer.

In Figure 2b, we compare different Transformer variants under identical training hyperparameter settings in terms of test accuracy, training throughput, and peak memory usage on the document retrieval task. This configuration uses a maximum sequence length of $N = 4000$ and a per-head dimension of $d = 64$ in the encoder, satisfying $N \gg d$. Under this setting, all linear attention models demonstrate substantially faster training speed and significantly lower memory consumption compared to the vanilla Transformer. Compared with other linear attention methods, Flexformer$_{\text{n}}$ introduces additional parameters and employs a slightly more complex feature map, leading to marginally reduced efficiency and memory savings. However, this modest overhead is offset by significantly improved model performance while preserving linear time and space complexity. On the document retrieval task, Flexformer achieves a $2.6\times$ training speedup and an 84% reduction in memory usage relative to the vanilla Transformer.

4.2 Autoregressive Language Modeling

To evaluate Flexformer's effectiveness on text generation tasks, we conduct autoregressive language modeling experiments on WikiText-103 [23]. For autoregressive modeling, the summations in Equation (3) are restricted to the prefix $j \leq i$. Using the feature map $\tilde{\phi}_n$, we maintain prefix states $\mathbf{S}_i \in \mathbb{R}^{d' \times d}$ and $\mathbf{z}_i \in \mathbb{R}^{d'}$, initialized as $\mathbf{S}_0 = \mathbf{0}$ and $\mathbf{z}_0 = \mathbf{0}$:

$$\mathbf{S}_i = \mathbf{S}_{i-1} + \tilde{\phi}_n(\mathrm{K}_i)\mathrm{V}_i^\top, \qquad \mathbf{z}_i = \mathbf{z}_{i-1} + \tilde{\phi}_n(\mathrm{K}_i), \qquad \mathrm{O}_i^\top = \frac{\tilde{\phi}_n(\mathrm{Q}_i)^\top \mathbf{S}_i}{\tilde{\phi}_n(\mathrm{Q}_i)^\top \mathbf{z}_i + \epsilon},$$

where ϵ is a small constant for numerical stability. During training, token-wise projections and feature mappings are computed in parallel, while causal aggregation is evaluated through a recurrent scan without constructing the $N \times N$ attention matrix. During autoregressive decoding, only $\mathbf{S}_i$ and $\mathbf{z}_i$

are cached and updated for each new token. This preserves $\mathcal{O}(Ndd')$ time complexity for a sequence of length N, with $\mathcal{O}(dd')$ per-token attention cost and state-cache size during decoding. Following the hardware-efficient implementation strategies in FLA [41], we can further accelerate the recurrent scan using fused Triton kernels that combine state updates, normalization, and output computation, avoiding Python-level iteration and reducing kernel-launch overhead. Intermediate states can be recomputed in a custom backward pass to reduce the activations retained across layers.

We use the vanilla Transformer with adaptive inputs [2] as the base architecture and replace its softmax attention with various linear attentions. Following the protocol of [28], we evaluate small and large models with roughly 41M and 247M parameters, respectively. The detailed hyperparameter settings are provided in Appendix A.2.1. For all experiments, we set the sentence block size to 512 and train the models for 100K steps using a batch size of 128.

The final evaluation results of all baselines on both the development and test sets are reported in Table 2. Under both model configurations, the vanilla Transformer with softmax attention consistently achieves strong performance. Hedgehog and Polaformer enable linear attention to preserve the low-entropy property of softmax attention, resulting in substantial gains over earlier linear-attention methods. Nevertheless, they still fall short of matching the vanilla Transformer, although the performance gap narrows under the large model configuration.

Notably, both Flexformer variants outperform all other linear attention methods under the small model configuration and achieve performance very close to that of the vanilla Transformer. Under the big model configuration, Flexformer further surpasses the vanilla Transformer, with Flexformer$_{\text{n}}$ demonstrating a particularly pronounced improvement. This observation suggests that learnable kernels benefit more from increased model capacity, as the additional parameters introduced by kernel learning allow better utilization of larger model scales. Importantly, the relative increase in parameter count remains nearly constant across configurations (see Appendix A.2.1). Moreover, Flexformer$_{\text{s}}$ introduces a comparable number of additional parameters to Hedgehog and fewer than Polaformer, yet still achieves superior performance.

Table 2: Language model perplexity (PPL) on the WikiText-103 development and test sets. Lower PPL is better. The best results for each benchmark are in **bold**, and the second best in underlined.

	Small (41M Params)		Big (247M Params)	
Model	Dev.	Test	Dev.	Test
Vanilla Trans.	**30.0**	**31.3**	22.6	23.8
Linear Trans.	34.1	35.6	27.9	29.2
RFA	33.8	35.3	26.7	28.0
Cosformer	32.8	34.2	24.9	26.2
Hedgehog	31.7	33.2	23.0	24.2
Polaformer	31.5	32.9	23.1	24.3
Flexformer$_{\text{s}}$	31.1	32.7	22.2	23.4
Flexformer$_{\text{n}}$	30.3	31.6	**21.2**	**22.3**

4.3 Recovering Softmax Attention

Besides training from scratch, another scenario that demands efficient linear attention is when we already have a traditional Transformer pretrained or finetuned, and we need to replace its softmax attention with linear attention while recovering the original performance. We adopt the attention weight distillation loss [43] to supervise the attention weights computed by Flexformer with the attention weights derived from softmax. We first download a pretrained 125M RoBERTa [18] model using the fairseq [26] library and then finetune it on each dataset of GLUE [39] benchmark. We compare the following categories of methods: (1) Directly finetuning the pretrained RoBERTa on the corresponding dataset. (2) Directly replace the attention computation in RoBERTa with Performer and RFA (they are not learnable), and then finetune the models. (3) Starting from a finetuned RoBERTa model, we perform attention weights distillation on the corresponding dataset to convert it into Flexformer$_{\text{n}}$ and Hedgehog, respectively, followed by finetuning of the resulting models. More detailed experimental settings are provided in Appendix A.3. For these three methods, we use the same finetuning hyperparameter configuration and present the experimental results in Table 3.

Compared to directly finetuning pretrained RoBERTa, replacing its softmax attention with linear attention methods that provide unbiased estimates of the softmax kernel severely degrades performance, especially on several datasets. In contrast, distilled Hedgehog and Flexformer both nearly fully recover the softmax attention behavior. Furthermore, the learnable kernel demonstrates performance that even surpasses standard softmax attention on CoLA.

Table 3: Performance comparison on GLUE. On CoLA, the Matthews correlation coefficient is reported; on STS-B, the average of Pearson and Spearman correlation coefficients is reported; for all other tasks, accuracy is reported. Higher values indicate better performance for all metrics. The best results of each task (excluding RoBERTa) are in **bold**, and the second best in underlined.

Model	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	Average
RoBERTa (reference)	61.9	94.8	88.5	90.2	91.7	87.6	92.7	76.9	85.5
Performer (unlearnable)	22.8	88.9	66.3	38.8	84.4	77.5	76.3	51.6	63.3
RFA (unlearnable)	27.6	90.1	64.2	41.2	86.3	80.8	75.8	53.0	64.9
Hedgehog (distilled)	62.5	93.3	**86.0**	87.7	**91.2**	83.7	**91.2**	71.0	83.3
Flexformer$_n$ (distilled)	**63.3**	**93.6**	85.8	**88.3**	91.1	**85.1**	91.0	**71.5**	**83.7**

Table 4: Comparison of the transfer performance of kernels learned by Flexformer$_n$ and Hedgehog. For each target task, we report the average performance over kernels learned from different source domains. The best results of each task are in **bold**.

Model	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE
Hedgehog	55.6	92.6	76.3	85.3	89.6	82.7	87.5	53.1
Flexformer$_n$	**60.3**	**93.3**	**79.5**	**86.0**	**90.7**	**84.8**	**89.5**	**56.1**

4.4 Transferability of Learned Kernel

In this section, we examine whether the attention kernels learned by Flexformer on one dataset can be transferred to datasets from other domains, thus maintaining strong generalization performance. To this end, we first perform attention weight distillation on a pretrained RoBERTa model using a single source dataset, converting it into a linear-attention model. Specifically, we use four GLUE datasets—STS-B, MNLI, QQP, and QNLI—to obtain four distilled models, each learned from a different source domain. We then fine-tune these four distilled models on the remaining GLUE datasets, enabling us to evaluate their cross-domain transferability. The complete results are in Table 11. In Table 4, we report the averaged performance, where each column shows the average performance on the target task over kernels learned from different source datasets.

We observe that, when transferring the learned attention kernel from a source domain to a target domain, Flexformer$_n$ consistently achieves superior performance across all settings. Moreover, compared to the in-domain results reported in Table 3, Hedgehog suffers a particularly severe performance degradation on the CoLA dataset under kernel transfer, whereas Flexformer$_n$ exhibits a substantially smaller performance drop. These results demonstrate that the attention kernels learned by Flexformer possess stronger cross-domain transferability than those of existing baselines.

5 Related Works

Kernel-based linear attention methods reduce the quadratic complexity of softmax attention by expressing the attention kernel as an inner product of feature maps. Existing approaches differ in how these feature maps are constructed. Early methods approximate the exponential function via Taylor expansion [1], but even low-order approximations lead to prohibitively high feature dimensions [8]. Other works adopt simple element-wise mappings, such as Linear Transformer [13] and Cosformer [29], which are efficient but limited in expressiveness. Random-feature-based methods, including Performer [7] and RFA [28], use random Fourier features to obtain unbiased estimators of the softmax kernel, assuming softmax attention is optimal and fixing the kernel a priori. Recent approaches introduce learnable feature maps: Hedgehog [43] exploits the low-entropy property of softmax using exponential-based mappings, while PolaFormer [22] incorporates polarity-aware mixing and power functions. These methods empirically design kernels with desirable properties, but the expressiveness of their kernel families is not explicitly characterized. In contrast, Flexformer is grounded in spectral representation theory, which establishes a one-to-one correspondence between a kernel and its spectral measure. By treating spectral frequencies as learnable parameters and

optimizing them directly from data, Flexformer enables a principled, data-driven construction of attention kernels and can model a broader and more expressive kernel family.

6 Conclusions

We proposed Flexformer, a flexible kernel-based linear attention framework that learns attention kernels directly from data using Fourier features. By treating spectral frequencies as trainable parameters, Flexformer expands the expressiveness of linear attention while preserving linear time and space complexity with respect to sequence length. We further introduced stationary and nonstationary variants, depending on the desired level of flexibility in the kernel. Empirically, extensive experiments on language modeling and sequence classification tasks demonstrate that Flexformer consistently outperforms existing linear attention baselines. Moreover, when distilled from pretrained Transformers, Flexformer can approximate softmax attention while enabling efficient linear attention inference. We also observe strong cross-domain kernel transferability. Overall, Flexformer provides an effective and scalable alternative to softmax attention for long-sequence modeling.

Limitations. Flexformer assumes the learned attention kernel is positive definite, but whether positive definite kernels are truly the best choice for attention remains theoretically unexplored. Establishing a rigorous justification for this assumption is an important direction for future work.

Acknowledgments

This work was supported by the NSFC Project (No.62576346), the MOE Project of Key Research Institute of Humanities and Social Sciences (22JJD110001), the fundamental research funds for the central universities, the research funds of Renmin University of China (24XNKJ13).

References

[1] Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Re. Simple linear attention language models balance the recall-throughput tradeoff. In *International Conference on Machine Learning*, 2024.

[2] Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*, 2019.

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

[4] Salomon Bochner. *Harmonic analysis and the theory of probability*. Courier Corporation, 2005.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

[6] Yifan Chen, Qi Zeng, Heng Ji, and Yun Yang. Skyformer: Remodel self-attention with gaussian kernel and nyström method. In *Advances in Neural Information Processing Systems*, 2021.

[7] 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.

[8] Alexandre de Brébisson and Pascal Vincent. An exploration of softmax alternatives belonging to the spherical loss family. In *International Conference on Learning Representations*, 2016.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

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

[11] Florian Eilers and Xiaoyi Jiang. Building blocks for a complex-valued transformer architecture. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2023.

[12] Roos Houtkamp and Pieter Roelfsema. Parallel and serial grouping of image elements in visual perception. *Journal of Experimental Psychology: Human Perception and Performance*, 2010.

[13] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Francois Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, 2020.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[15] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

[16] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[17] Miguel Lázaro-Gredilla, Joaquin Quiñonero-Candela, Carl Edward Rasmussen, and Aníbal R. Figueiras-Vidal. Sparse spectrum gaussian process regression. *Journal of Machine Learning Research*, 2010.

[18] 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*, 2019.

[19] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *International Conference on Learning Representations*, 2024.

[20] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[21] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Annual Meeting of the Association for Computational Linguistics*, 2011.

[22] Weikang Meng, Yadan Luo, Xin Li, Dongmei Jiang, and Zheng Zhang. Polaformer: Polarity-aware linear attention for vision transformers. In *International Conference on Learning Representations*, 2025.

[23] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

[24] Nikita Nangia and Samuel R. Bowman. Listops: A diagnostic dataset for latent tree learning. *arXiv*, 2018.

[25] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

[26] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*, 2019.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.

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

[29] 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.

[30] Dragomir R. Radev, Pradeep Muthukrishnan, and Vahed Qazvinian. The acl anthology network corpus. In *Workshop on Text and Citation Analysis for Scholarly Digital Libraries*, 2009.

[31] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, 2007.

[32] Aurko Roy, Mohammad Taghi Saffar, David Grangier, and Ashish Vaswani. Efficient content-based sparse attention with routing transformers. In *arXiv*, 2020.

[33] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

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

[35] Jerome Sieber, Carmen Amo Alonso, Alexandre Didier, Melanie Zeilinger, and Antonio Orvieto. Understanding the differences in foundation models: Attention, state space models, and recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2024.

[36] Zicheng Sun, Yixuan Zhang, Zenan Ling, Xuhui Fan, and Feng Zhou. Nonstationary sparse spectral permanental process. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, 2024.

[37] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[39] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP at EMNLP*, 2018.

[40] Akiva M. Yaglom. *Correlation Theory of Stationary and Related Random Functions, Volume I: Basic Results*. Springer, 1987.

[41] Songlin Yang and Yu Zhang. Fla: A triton-based library for hardware-efficient implementations of linear attention mechanism, January 2024.

[42] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, 2020.

[43] Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Re. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. In *International Conference on Learning Representations*, 2024.

[44] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI Conference on Artificial Intelligence*, 2021.

A Experimental Details

A.1 Introduction of Datasets

Statistics on the sizes of all datasets used in our experiments are provided in Table 5. Next, we provide a brief overview of these datasets.

Table 5: Statistics of the datasets used in our experiments.

Dataset	Train	Valid	Test
ListOps	96K	2K	2K
IMDB	25K	–	25K
AAN	147K	18K	17K
CIFAR-10	45K	5K	10K
Pathfinder	160K	20K	20K
WikiText-103	103M	218K	246K
CoLA	8.5K	–	1K
SST-2	67K	–	1.8K
MRPC	3.7K	–	1.7K
STS-B	7K	–	1.4K
QQP	364K	–	391K
MNLI	393K	–	20K
QNLI	105K	–	5.4K
RTE	2.5K	–	3K
WNLI	634K	–	146

A.1.1 Long Range Arena

Next, we provide a detailed introduction to the datasets of the five tasks in the LRA.

Long ListOps A 10-class classification task based on a longer variant of the synthetic ListOps dataset [24]. The sequences with a max length of 2K consist of nested expressions using operators like MAX, MIN, and MEDIAN, and models must predict the final result (an integer from 0 to 9), testing their ability to model hierarchical structure over long contexts.

Byte-level Text Classification A binary classification task using IMDb movie reviews [21] processed as raw byte sequences with fixed length of 4K. The goal is to classify each review as positive or negative sentiment without word segmentation or pretraining.

Byte-level Document Retrieval A binary classification task based on the ACL Anthology Network (AAN) [30] dataset, where models determine whether two 4K-byte academic documents share a citation link, using a two-tower encoding architecture.

Image Classification on Sequences of Pixels A 10-class classification task using grayscale CIFAR-10 images [16], flattened into 1,024-pixel sequences. Models classify each image into one of ten object categories using only sequential pixel values.

Pathfinder A binary classification task derived from the Pathfinder dataset [12] introduced in cognitive psychology and adapted for machine learning. It uses 32×32 synthetic images (1,024 pixels) containing two marked points and dashed paths; the model must determine whether the points are connected by a continuous path, evaluating its capacity for long-range spatial reasoning.

A.1.2 Wikitext-103

WikiText-103 [23] is a large-scale language modeling benchmark introduced in this paper to address the limitations of smaller datasets like Penn Treebank. It is constructed from 28,475 high-quality English Wikipedia articles (including 23,805 "Good" and 4,790 "Featured" articles), resulting in a training set of over 103 million words. The dataset preserves original casing, punctuation, and named

entities, making it more realistic for real-world applications. After preprocessing using the Moses tokenizer and replacing words occurring fewer than 3 times with <unk>, it has a vocabulary size of 267,735 and a very low out-of-vocabulary rate of 0.4%. Unlike shuffled corpora, WikiText-103 maintains full article structure, enabling the evaluation of models on long-range dependencies and rare word prediction.

A.1.3 GLUE

We introduce below the eight datasets in GLUE corresponding to their respective tasks.

CoLA (Corpus of Linguistic Acceptability) The task is to judge whether a sentence is grammatically acceptable in English. The data is drawn from books and journal articles on linguistic theory. Labels are binary: acceptable or unacceptable.

SST-2 (Stanford Sentiment Treebank) The task is binary sentiment classification of sentences from movie reviews. The data comes from movie reviews. Labels are positive or negative sentiment.

MRPC (Microsoft Research Paraphrase Corpus) The task is to determine if two sentences are semantically equivalent (paraphrases). Sentence pairs are automatically extracted from online news sources. Labels are binary: paraphrase or not paraphrase.

STS-B (Semantic Textual Similarity Benchmark) The task is to predict the degree of semantic similarity between two sentences. The data is drawn from news headlines, video/image captions, and natural language inference datasets. Labels are continuous similarity scores ranging from 1 to 5 (a regression task).

QQP (Quora Question Pairs) The task is to determine whether two questions from the Quora community QA platform are semantically equivalent (duplicates). Labels are binary: duplicate or not duplicate.

MNLI (Multi-Genre Natural Language Inference) The task is to classify the relationship between a premise and a hypothesis sentence as entailment, contradiction, or neutral. Premise sentences are gathered from ten different sources, including transcribed speech, fiction, and government reports. Labels are three-way: entailment, contradiction, or neutral.

QNLI (Question Natural Language Inference) This task is derived from the SQuAD question-answering dataset. It is recast as an NLI problem to determine if a sentence from a Wikipedia paragraph contains the answer to a given question. Labels are binary: entailment (contains answer) or not entailment.

RTE (Recognizing Textual Entailment) The task is to recognize textual entailment in a binary setting. The data combines examples from the RTE1, RTE2, RTE3, and RTE5 challenges, based on news and Wikipedia text. Labels are binary: entailment or not entailment.

A.2 Setup

All of our experiments are implemented using PyTorch 2.9.1 [27] and conducted on a single Nvidia RTX5090 GPU with 32GB memory. We use the Adam [14] optimizer for all experiments.

A.2.1 Autoregressive Language Modeling

The detailed hyperparameter configurations for the autoregressive language modeling experiments are listed in Table 6. Total number of parameters refers to the Transformer architecture without any additional parameters introduced. Methods with learnable feature maps have a total number of parameters slightly larger than this value. Their specific amounts of additional parameters are shown in Table 7. As discussed in Section B, the additional parameters introduced by Flexformer are minimal, all less than 1%. The additional parameters of $\text{Flexformer}_\text{n}$ are almost exactly twice those of Hedgehog, whereas $\text{Flexformer}_\text{s}$ introduces nearly the same amount of extra parameters as Hedgehog. Polaformer, in contrast, incurs a larger number of additional parameters because it applies

a linear projection after concatenating all heads, rather than using separate projections for each head. In large language models, a greater number of parameters often leads to better performance. Our results show that Flexformer achieves a higher performance gain per additional parameter compared to other models.

Table 6: Detailed hyperparameter settings for autoregressive language modeling experiments.

Hyperparameter	Small	Big
Embedding Size	512	1024
Layers	6	16
Heads	8	16
Head Size	64	64
FFN Size	2048	4096
Batch Size	128	128
Learning Rate	5×10^{-4}	5×10^{-4}
Total Steps	100,000	100,000
Warmup Steps	4,000	4,000
Dropout	0.1	0.2
Gradient Clipping Norm	0.1	0.1
Total Number of Parameters	41 M	247 M

Table 7: Comparison of parameter counts between methods with learnable feature maps and the Vanilla Transformer. "Relative Parameter Increase" denotes the percentage increase in model parameters compared to a Vanilla Transformer under the same configuration.

	Small		Big	
Model	Total Params	Rel. Increase (%)	Total Params	Rel. Increase (%)
Transformer	41M	0.00	247M	0.00
Hedgehog	41M	0.49	248M	0.43
Polaformer	43M	3.83	262M	6.74
$\text{Flexformer}_\text{s}$	41M	0.49	248M	0.43
$\text{Flexformer}_\text{n}$	42M	0.97	249M	0.86

A.3 Recovering Softmax Attention and Transferability of Learned Kernel

For each dataset of GLUE, we use the preprocessing script from the Fairseq repository[2] to further split the training set into training and validation subsets. The downloaded pre-trained 125M RoBERTa model uses a GPT-style BPE tokenizer with a vocabulary size of approximately 50K, an embedding dimension of 512, 12 encoder layers, and 8 attention heads. All methods use identical settings during finetuning, and both Hedgehog and Flexformer employ the same configuration for distillation.

Finetuning Setting. The fine-tuning configurations are also taken from the Fairseq repository[3]: each task uses a batch size of 16 or 32, a learning rate of 1×10^{-5} or 2×10^{-5}, and is trained for 10 epochs, with the checkpoint achieving the best validation performance selected for testing.

Attention Distillation Setting. We use the attention weight distillation loss proposed in [43]:

$$\mathcal{L}_{awd} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\exp(\frac{\mathrm{Q}_i^\top \mathrm{K}_j}{\sqrt{d}})}{\sum_{l=1}^{N}\exp(\frac{\mathrm{Q}_i^\top \mathrm{K}_l}{\sqrt{d}})}\log\frac{\tilde{\phi}_n(\mathrm{Q}_i)^\top\tilde{\phi}_n(\mathrm{K}_j)}{\sum_{l=1}^{N}\tilde{\phi}_n(\mathrm{Q}_i)^\top\tilde{\phi}_n(\mathrm{K}_l)}.$$

It computes the cross-entropy between the distribution obtained by normalizing each query with our kernel and the target distribution from softmax attention, then averages this loss over all query positions. During attention weight distillation, we freeze all parameters of RoBERTa, replace its attention computation with our learnable kernel, and train only the kernel parameters for 20,000 steps

[2] https://github.com/facebookresearch/fairseq/blob/main/examples/roberta/preprocess_GLUE_tasks.sh

[3] https://github.com/facebookresearch/fairseq/tree/main/examples/roberta/config/finetuning

on a dataset with a batch size of 32 and a learning rate of 1×10^{-3}. For the experiments in Section 4.3, we perform attention weight distillation followed by fine-tuning directly on the target dataset. In contrast, for Section 4.4, we first conduct attention weight distillation on a different dataset and then finetune the model on the target dataset. We select STS-B, QQP, MNLI, and QNLI as distillation datasets because they span multiple domains and are representative of diverse tasks.

B Complexity Analysis

B.1 Time and Space Complexity

As discussed in Section 2.2, the overall time and space complexity of linear attention is $\mathcal{O}(Ndd')$ and $\mathcal{O}(Nd + Nd' + dd')$ respectively. For Flexformer, $d' = 2n$, where n denotes the number of frequencies. Generally, larger d' enhances the expressive power of the learned kernel but increases both parameter count and complexity. A trade-off between expressiveness and efficiency is therefore necessary to determine the optimal hyperparameter d'. In subsequent comparative experiments, to ensure maximal fairness, we set $n = d, d' = 2d$ following prior work [28, 43].

B.2 Additional Parameters

Compared to early linear attention methods [13, 7], more recent studies [43, 22] incorporate additional learnable parameters to enhance the expressive power of kernel-based linear attention. Flexformer follows the same line. Let h denote the number of heads. Assuming $n = d$, Flexformer requires approximately d^2 extra parameters per head for stationary variant ($2d^2$ for nonstationary). To maximize expressiveness, we assume these parameters independent across all heads. In this configuration, Flexformer requires approximately $\frac{1}{12h}$ extra parameters per encoder/decoder layer for stationary variant ($\frac{1}{6h}$ for nonstationary), which accounts for only about 1% of the total parameters when $h = 8$.

C Error Bar

All our main experiments were repeated under three different random seeds, and we report the mean results in the main text. In this section, we additionally report the standard deviations in Table 8, Table 9 and Table 10.

Table 8: Mean and standard deviation of performance metrics on the LRA benchmarks. All reported metrics represent test accuracy (%), where higher values indicate better performance. The best results for each benchmark are in **bold**, and the second best in underlined.

Model	ListOps	Text	Retrieval	Image	Pathfinder	Average
Transformer [38]	37.23±0.17	65.66±0.58	64.49±0.61	41.89±0.29	73.46±0.43	56.55±0.21
Reformer [15]	36.62±0.18	63.52±0.56	57.60±0.58	40.06±0.30	73.61±0.46	54.28±0.22
Longformer [3]	36.46±0.23	63.09±0.56	59.24±0.60	41.23±0.30	72.44±0.42	54.49±0.21
Linear Trans. [13]	17.42±0.12	62.64±0.54	58.92±0.58	43.40±0.32	75.11±0.44	51.49±0.20
BigBird [42]	36.88±0.16	65.34±0.60	64.23±0.60	41.93±0.31	73.58±0.45	56.39±0.23
Performer [7]	16.97±0.14	63.67±0.57	55.17±0.56	41.23±0.29	75.38±0.44	50.48±0.20
RFA [28]	17.20±0.13	64.89±0.58	58.76±0.61	41.00±0.30	74.51±0.48	51.27±0.21
Skyformer [6]	36.71±0.16	65.51±0.59	68.05±0.63	41.36±0.31	74.18±0.45	57.16±0.23
Cosformer [29]	**37.44±0.21**	65.16±0.57	64.04±0.61	43.56±0.30	72.32±0.42	56.50±0.22
Hedgehog [43]	37.02±0.18	64.10±0.58	68.67±0.63	41.64±0.32	75.89±0.48	57.47±0.23
Polaformer [22]	36.93±0.17	**69.65±0.61**	65.39±0.62	40.95±0.29	73.26±0.44	57.23±0.22
Flexformer$_s$ (Ours)	37.09±0.18	67.38±0.60	71.55±0.62	43.39±0.29	76.15±0.48	59.11±0.24
Flexformer$_n$ (Ours)	37.18±0.18	65.72±0.59	**73.94±0.66**	**45.42±0.32**	**77.66±0.48**	**59.99±0.25**

D Additional Results

In this section, we provide additional experimental results.

Table 9: Mean and standard deviation of PPL on the WikiText-103 development and test sets. Lower PPL indicates better performance. The best results for each benchmark are in **bold**, and the second best in underlined.

	Small (41M Params)		Big (247M Params)	
Model	Dev.	Test	Dev.	Test
Vanilla Trans.	**30.0±0.06**	**31.3±0.07**	22.6±0.03	23.8±0.06
Linear Trans.	34.1±0.12	35.6±0.10	27.9±0.08	29.2±0.08
RFA	33.8±0.11	35.3±0.11	26.7±0.09	28.0±0.10
Cosformer	32.8±0.09	34.2±0.11	24.9±0.05	26.2±0.06
Hedgehog	31.7±0.08	33.2±0.05	23.0±0.06	24.2±0.07
Polaformer	31.5±0.08	32.9±0.08	23.1±0.03	24.3±0.02
$\text{Flexformer}_{\text{s}}$	31.1±0.06	32.7±0.08	22.2±0.02	23.4±0.04
$\text{Flexformer}_{\text{n}}$	30.3±0.06	31.6±0.07	**21.2±0.03**	**22.3±0.04**

Table 10: Mean and standard deviation of performance metrics on GLUE. On CoLA, the Matthews correlation coefficient is reported; on STS-B, the average of Pearson and Spearman correlation coefficients is reported; for all other tasks, accuracy is reported. Higher values indicate better performance for all metrics. The best results of each task are in **bold**, and the second best in underlined.

Model	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	Average
Performer	22.8±1.21	88.9±0.32	66.3±0.69	38.8±0.45	84.4±0.40	77.5±0.30	76.3±0.53	51.6±0.26	63.3±0.31
RFA	27.6±0.92	90.1±0.29	64.2±0.71	41.2±0.52	86.3±0.37	80.8±0.29	75.8±0.45	53.0±0.33	64.9±0.29
Hedgehog	62.5±0.32	93.3±0.11	**86.0±0.12**	87.7±0.18	**91.2±0.10**	83.7±0.16	**91.2±0.14**	71.0±0.22	83.3±0.11
$\text{Flexformer}_{\text{n}}$	**63.3±0.28**	**93.6±0.12**	85.8±0.12	**88.3±0.16**	91.1±0.13	**85.1±0.16**	91.0±0.15	**71.5±0.20**	**83.7±0.10**

D.1 Hyperparameter Sensitivity

In all main experiments of this paper, to ensure a fair comparison, we set $n = d, d' = 2d$. However, as shown by our analysis in Section 3, a larger n can fit more complex kernel functions, but it also incurs higher computational cost. To guide the practical choice of an appropriate n, we investigated how model performance and computational cost vary with n on WikiText-103 and on the Image and ListOps tasks in LRA.

The results in Figure 3, Figure 4 and Figure 5 show that as n increases, the computational cost grows accordingly. On larger-scale models and datasets, increasing n consistently improves generalization, while on smaller-scale benchmarks such as LRA, performance saturates beyond a certain point. In the main experiments, we set $n = d$ to keep the feature dimension consistent with prior work, under which Flexformer already achieves notable performance gains. Moreover, the additional results show that even with a moderately smaller n, Flexformer still outperforms the baselines while maintaining better efficiency.

D.2 Recovering Softmax Attention

In Figure 6, we selectively compare the loss curves of attention weight distillation using Hedgehog and Flexformer on the MNLI and QNLI datasets. It can be observed that Flexformer consistently achieves lower cross-entropy loss on the validation set compared to Hedgehog. Moreover, Flexformer converges slightly faster than Hedgehog: its validation loss stabilizes around 8,000 training steps, whereas Hedgehog does not stabilize until nearly 12,000 steps. This further demonstrates that the learnable kernel in Flexformer is more flexible.

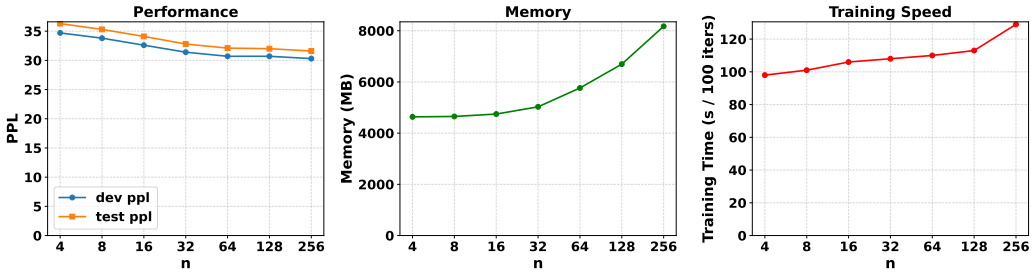


Figure 3: Performance, memory consumption, and training speed of Flexformer$_n$ in the autoregressive language modeling experiments under different values of the sampling frequency hyperparameter n. On large-scale models and datasets, increasing n improves both the fitting and generalization abilities of the model, while also incurring higher computational cost. In the main experiments, we set $n = 64$.

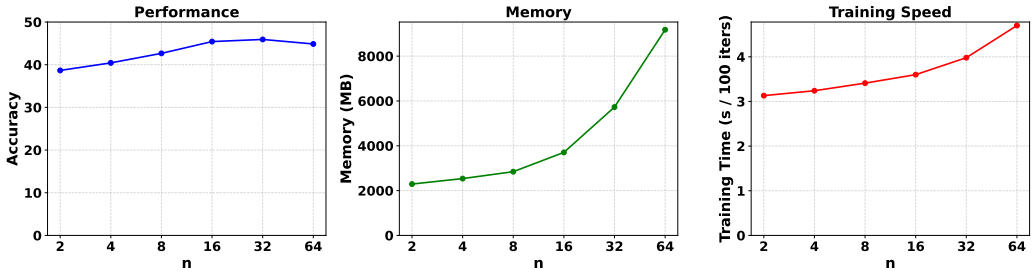


Figure 4: Performance, memory consumption, and training speed of Flexformer$_n$ on the Image dataset for Long Sequence Classification under different values of the sampling frequency hyperparameter n. Both memory usage and training time increase with larger n. On the Image dataset, increasing n improves the model's generalization when n is small; however, as n becomes larger, it increases computational cost and may even lead to overfitting. In the main experiments, we set $n = 16$.

D.3 Transfer Performance

In Table 11, we fully present the results of transferring the model—after performing attention weight distillation on a specific dataset—to other datasets. Flexformer achieves better performance than Hedgehog in almost all cases. Among them, the relative performance improvement reaches up to 13%. Moreover, it can also be observed that Hedgehog's performance is relatively unstable when distillation is performed on different datasets.

Table 11: Comparison of performance when learning the kernel on one dataset and transferring it to other datasets. The best results of each situation are in **bold**.

Source/Target	Model	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE
STS-B	Hedgehog	58.0	92.3	77.2	–	89.6	82.0	87.8	55.2
	Flexformer$_n$	**63.5**	**93.2**	**80.2**	–	**90.7**	**84.7**	**89.8**	**56.7**
QQP	Hedgehog	55.2	**93.3**	75.2	**85.1**	–	83.3	87.7	50.7
	Flexformer$_n$	**58.8**	**93.3**	**79.0**	84.8	–	**84.8**	**89.1**	**53.5**
MNLI	Hedgehog	57.7	92.8	76.5	85.5	90.1	–	86.9	52.6
	Flexformer$_n$	**60.6**	**93.1**	**80.4**	**86.7**	**90.4**	–	**89.5**	**54.9**
QNLI	Hedgehog	51.6	91.8	76.2	85.1	89.0	82.8	–	53.8
	Flexformer$_n$	**58.3**	**93.2**	**78.4**	**86.4**	**90.6**	**84.8**	–	**59.4**

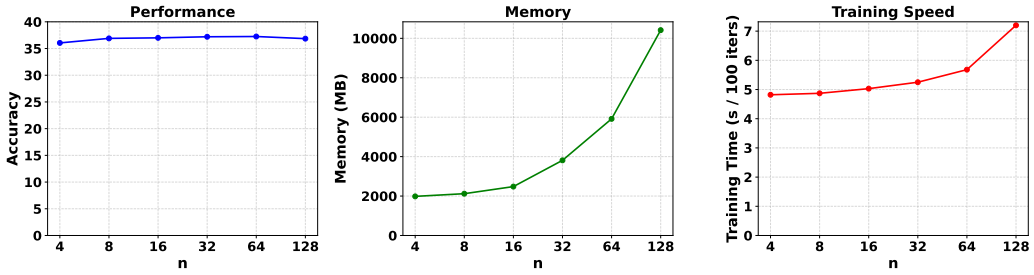
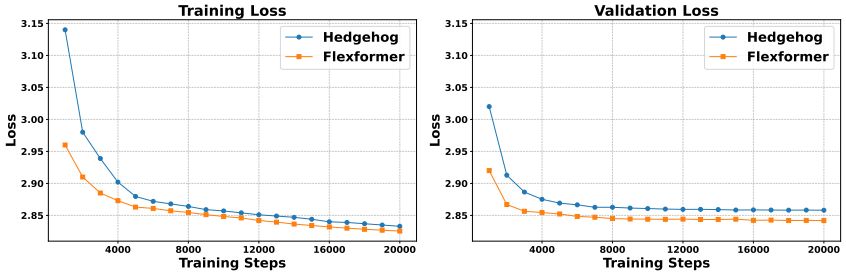
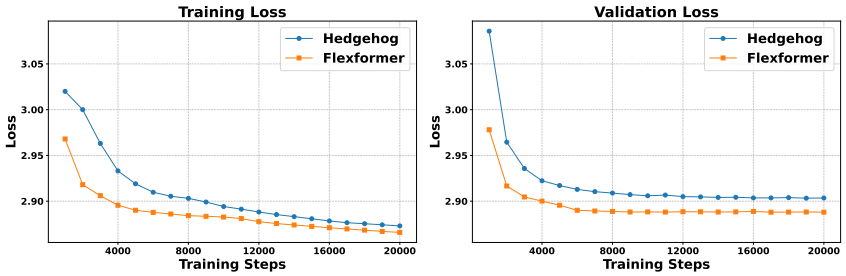


Figure 5: Performance, memory consumption, and training speed of Flexformer$_{\text{n}}$ on the ListOps dataset for Long Sequence Classification under different values of the sampling frequency hyperparameter n. As n increases, both memory usage and training time also increase. On ListOps, competitive performance can be achieved even with a very small n. In the main experiments, we use $n = 64$.



(a) Distillation on MNLI.



(b) Distillation on QNLI.

Figure 6: Comparison of loss curves during attention weight distillation training.

E Limitations and Future Work

Theoretical analysis in this paper assumes the learned kernel is positive definite, as are the softmax and Gaussian kernels. Empirically, positive definite kernels appear to induce desirable attention behaviors and avoid pathological patterns (e.g., weak self-attention but strong incoming attention). However, whether positive definite kernels are fundamentally optimal for attention mechanisms remains unexplored. We hope future work will provide a deeper understanding of this issue.

Our experiments are primarily conducted on tasks and datasets related to natural language processing. In fact, there exist numerous other scenarios involving long sequences that require attention mechanisms—for instance, processing high-resolution images or extremely long protein sequences—each of which may demand substantially different attention patterns for effective modeling. Although the LRA benchmark does provide a valuable testbed for evaluating the ability to capture long-range dependencies, thereby complementing the limitation of natural language data (where dependencies are often predominantly local), it remains insufficient to fully reflect the diverse range of attention patterns required across different modalities and domains. We plan to validate the effectiveness of our approach across a broader range of domains in future work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope. See the Abstract and Introduction sections.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitation of the work is discussed in the Conclusion section and Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: In this paper, we use established theorems and further derive our results based on them in the Methodology section.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper fully discloses all the information needed to reproduce the main experimental results in the paper. See the Experiments section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide available sources in supplement materials to reproduce the experimental results.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify all the training and test details necessary to understand the results. See the Experiments section and Appendix A.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report experimental results with standard deviation in Appendix C that support the main claims of the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the hardware we use to perform all experimetns. See Appendix A.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All code, models, and datasets mentioned in the text are appropriately cited with their original papers.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: New assets introduced in the paper, such as code, are well documented. The documentation is provided alongside the assets in the supplementary material.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.