Degrees of Freedom for Linear Attention: Distilling Softmax Attention with Optimal Feature Efficiency

Anonymous Author(s)
Affiliation
Address
email

Abstract

Linear attention has attracted interest as a computationally efficient approximation to softmax attention, especially for long sequences. Recent studies has explored distilling softmax attention in pre-trained Transformers into linear attention. However, a critical challenge remains: *how to choose the feature dimension that governs the approximation quality*. Existing methods fix this dimension uniformly across all attention layers, overlooking the diverse roles and complexities of them. In this paper, we propose a principled method to automatically determine the feature dimension in linear attention using the concept of statistical *degrees of freedom*, which represent the effective dimensionality of the inputs. We provide a theoretical bound on the approximation error and show that the dimension chosen by our method achieves smaller error under a fixed computational budget. Furthermore, we introduce an efficient layerwise training strategy to learn nonlinear features tailored to each layer. Experiments on multiple pre-trained transformers demonstrate that our method improves the performance of distilled models compared to baselines without increasing the inference cost. Our findings also provide insight into how the complexity of the attention mechanism evolves across layers.

1 Introduction

Transformers have become the standard for sequence modeling across diverse domains such as natural language processing (Vaswani et al., 2017), computer vision (Dosovitskiy, 2020), and speech processing (Dong et al., 2018). A key factor in their success is the attention mechanism, which effectively aggregates the information from input tokens. However, the standard softmax attention requires computing pairwise interactions between all tokens in a sequence, resulting in quadratic time and memory complexity with respect to sequence length. This scalability issue poses significant challenges for large-scale applications.

To address this limitation, numerous efforts have been made to design more efficient alternatives. One prominent approach is *linear attention*, which approximates softmax attention by replacing the kernel between queries and keys with an inner product of finite dimensional features. This reduces both time and memory complexity to linear in the sequence length, enabling scalable inference. The idea was initially proposed by Katharopoulos et al. (2020), and has since been extended in subsequent works (Peng et al., 2021; Choromanski et al., 2021; Qin et al., 2022). Recent architectures based on state space models (SSMs), such as Mamba (Gu and Dao, 2024), are also closely related to linear attention (Wang et al., 2024; Han et al., 2024).

Since linear attention is an approximation of softmax attention, one can directly distill the pre-trained softmax attention into linear attention. Despite prior studies (Chen et al., 2024; Wang et al., 2024) have indeed achieved some success in this approach, a critical challenge remains unresolved:

How should we choose the feature dimension that governs the approximation quality?

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

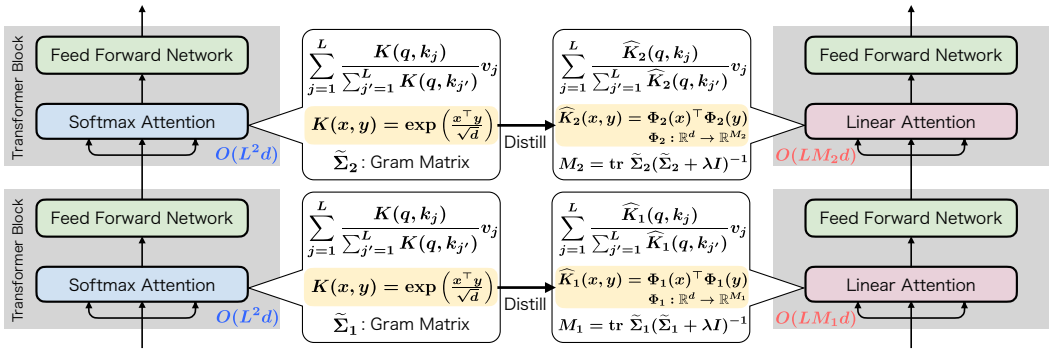

Figure 1: An overview of our method. We treat $\exp\left(x^\top y/\sqrt{d}\right)$, which appears in softmax attention, as a kernel $K(x, y)$, and perform distillation by approximating it with the inner product of nonlinear features. The required feature dimension to achieve a certain error varies depending on the input distribution, and this can be calculated using degrees of freedom derived from the Gram matrix. When the error level λ is specified, our method can automatically select the feature dimensions for linear attention, resulting in different feature dimensions for each layer.

Most existing works pre-determine the feature dimensions and fix them across all layers, which ignores the diverse functional roles and input distributions of different attention layers. As observed in prior studies (Arora et al., 2018; Ravichandran et al., 2019; Suzuki et al., 2020; Massaroli et al., 2024), the complexity of the roles played by each layer varies greatly. It is thus reasonable to expect that selecting the feature dimension adaptively, in accordance with each layer's complexity, can improve the performance of the distilled model.

In this paper, we propose a principled approach to determining the feature dimension in linear attention. We begin by theoretically deriving a bound on the approximation error and show that the number of features required to approximate the original attention kernel is governed by the statistical *degrees of freedom (DoF)*. Based on this insight, we develop a method to automatically select the feature dimension for each layer by estimating its DoF. Because the DoF captures the effective dimensionality of the input distribution, our method enables a data-adaptive and layer-specific allocation of feature dimensions.

Although our theoretical analysis identifies the optimal feature distribution, this distribution is generally intractable to compute. To obtain linear attention that achieves optimal approximation accuracy, we introduce a training strategy for learning nonlinear feature maps. Instead of relying on costly end-to-end training with the pre-training objective, we propose to train each layer independently. This *layerwise training* approach significantly reduces computational cost while allowing each attention layer to learn features tailored to its input distribution.

We validate our approach through extensive experiments on GPT-2 and Pythia-1B. Our results show that (1) the optimal feature dimension varies substantially across layers, (2) our method improves the accuracy of distilled models over fixed-dimension baselines, and (3) our layerwise feature learning yields competitive performance with significantly reduced training cost.

Our contributions can be summarized as follows:

- We propose a principled method for *automatically selecting the feature dimension* of linear attention based on the statistical degrees of freedom, which reflect the complexity of each attention layer. We provide theoretical guarantees on the approximation error under this selection scheme.
- We introduce an efficient *layerwise training* strategy to learn nonlinear features tailored to each layer, which substantially reduces the computational cost compared to end-to-end training.
- We empirically demonstrate that selecting feature dimensions by the proposed method improves performance of the distilled model and yields results comparable to the original models. The experimental results also offer insight into *how attention complexity varies across layers*.

Other related works. Some studies aim to distill the attention mechanism in pre-trained Transformers into models based on linear attention. Chen et al. (2024) propose a method to distill softmax

attention into linear attention by using a quasi Monte Carlo method, which is more accurate than standard Monte Carlo sampling. Wang et al. (2024) explore a distillation from transformers to Mamba taking into account the similarities between Mamba and linear attention. Furthermore, Bick et al. (2024) and Ralambomihanta et al. (2024) propose methods for distilling Transformers into SSM-based models, and Kasai et al. (2021) develop a method to distill Transformers into RNN-based models. In all of them, methods for selecting feature dimensions based on the complexity of the kernels have not been investigated.

We also refer to the studies of Massaroli et al. (2024) and Sakamoto and Sato (2024) that focus on distilling SSMs into smaller SSMs. These studies consider selecting dimensions of states in the distilled models. However, their methods are highly dependent on the properties of SSMs and cannot be directly applied to distillation from softmax attention to linear attention.

Wang et al. (2020) proposes a method to make softmax attention more efficient, and focus on the fact that attention matrices tend to be low-rank. They propose compressing the keys and the values from $L \times d$ to $L' \times d$ ($L' \ll L$), where L is the sequence length and d is the head size. Our method is similar in that it focuses on the effective dimension calculated from the Gram matrix of the attention kernel. However, we use it for dimension selection in linear attention instead of compressing the keys and values. Furthermore, our method significantly differs from theirs in that the feature dimension is automatically selected using data and takes different values for each layer.

Notations. Let μ be a probability measure on $\mathbb{R}^d$. We define $L_2(\mu)$ as the space of functions $f : \mathbb{R}^d \to \mathbb{R}$ such that $\int f(z)^2 \mu(dz) < \infty$. We denote $\int f(z)\mu(dz)$ as $\mathbb{E}_{z\sim\mu}[f(z)]$ and $\int f(z)g(z)\mu(dz)$ as $\langle f, g\rangle_{L_2(\mu)}$. For probability measures μ_1 and μ_2 on $\mathbb{R}^d$, we denote the product measure as $\mu_1 \otimes \mu_2$. For a bounded linear operator $A : X \to Y$, we denote the operator norm as $\|A\|_{\mathrm{op}}$. For $m \in \mathbb{N}$, we define $[m] \coloneqq \{1, 2, \ldots, m\}$.

2 Attention Mechanism and its Linearization

In this section, we first introduce the regular attention mechanism, and then explain linear attention, which is an approximation of the regular one. For simplicity, in this section, we explain only single-head cases, but the description can be easily extended to multi-head cases as well. Moreover, we focus on unidirectional cases, which are commonly used in language models.

Attention mechanism. Let $[x_1, x_2, \ldots, x_L]$ be a sequence of L vectors (called *tokens*), where $x_i \in \mathbb{R}^d$ is the i-th vector. The *attention mechanism* computes a new sequence $[y_1, y_2, \ldots, y_L]$ by taking a weighted sum of the projected tokens as follows:

$$y_i = \sum_{j=1}^{i} \frac{K(q_i, k_j)}{\sum_{j'=1}^{i} K(q_i, k_{j'})} v_j, \quad K(x, y) \coloneqq \exp\Big(x^\top y / \sqrt{d}\Big), \tag{1}$$

where $k_i \coloneqq W^K x_i, q_i \coloneqq W^Q x_i, v_i \coloneqq W^V x_i$, which are called the *key*, *query*, and *value*, respectively, and $W^K, W^Q, W^V \in \mathbb{R}^{d\times d}$ are the learnable parameters. The kernel $K(x, y)$ is referred to as the *attention kernel*.

The time and memory complexity of the attention mechanism is $\mathcal{O}(L^2 d)$ and $\mathcal{O}(L^2 + Ld)$, respectively. Since we need to compute the inner products between every pair of keys and queries, the cost increases quadratically with the sequence length L. Although the attention mechanism is powerful, this becomes a drawback when dealing with the long sequences typical in language and speech.

Linear attention. To deal with large computational cost, in *linear attention*, the computation of (1) is simplified by approximating $K(x, y)$ with a inner product of finite dimensional feature maps. Specifically, we suppose that K has a form $K(x, y) = \mathbb{E}_{z\sim\tau}[\phi(x; z)\phi(y; z)]$, where τ is a probability measure and $\phi(\cdot; z) : \mathbb{R}^d \to \mathbb{R}$ is a feature map. Then, we can approximate K using finite i.i.d. samples $z_1, \ldots, z_M \sim \tau$, i.e., $K(x, y) \approx \frac{1}{M}\sum_{m=1}^{M} \phi(x; z_m)\phi(y; z_m) = \Phi(x)^\top \Phi(y)$, where $\Phi(x) = \frac{1}{\sqrt{M}}[\phi_1(x), \ldots, \phi_M(x)]^\top$. Linear attention approximates the output of (1) by replacing $K(q_i, k_j)$ with $\Phi(q_i)^\top \Phi(k_j)$.

This simplification drastically reduces the computational cost. Specifically, we can compute the outputs in the following form:

$$\hat{y}_i = \sum_{j=1}^{i} \frac{\Phi(k_j)^\top \Phi(q_i)}{\sum_{j'=1}^{i} \Phi(k_{j'})^\top \Phi(q_i)} v_j = \frac{B_i^\top \Phi(q_i)}{A_i^\top \Phi(q_i)}, \quad A_i := \sum_{j=1}^{i} \Phi(k_j), \quad B_i := \sum_{j=1}^{i} \Phi(k_j) v_j^\top.$$

Since A_i and B_i ($i = 1, \ldots, L$) can be computed recursively over i, the computational complexity with respect to input length L is significantly reduced from quadratic to linear: the time and memory complexities become $\mathcal{O}(LMd)$ and $\mathcal{O}(LM + Ld + Md)$, respectively.

There are multiple options for the feature map Φ. The simplest approach involves using the decomposition $\exp\left(\frac{q^\top k}{\sqrt{d}}\right) = \exp\left(\frac{\|q\|^2}{2\sqrt{d}}\right) \exp\left(\frac{\|k\|^2}{2\sqrt{d}}\right) \exp\left(-\frac{\|q-k\|^2}{2\sqrt{d}}\right)$, and subsequently applying Bochner's theorem to break down the final component, $\exp\left(-\frac{\|q-k\|^2}{2\sqrt{d}}\right)$, into random Fourier features. While this method appears convenient, it suffers from learning instability caused by the appearance of negative values in $\sin$ and $\cos$ functions. To address this issue, Choromanski et al. (2021) propose using the decomposition of $\exp\left(q^\top k/\sqrt{d}\right) = \mathbb{E}_{z \sim \mathcal{N}(0, \mathbb{I}_d)}[\phi(q; z)\, \phi(k; z)]$, where $\phi(x; z) = \exp\left(\frac{z^\top x}{d^{1/4}} - \frac{\|x\|^2}{2\sqrt{d}}\right)$. This feature map is referred to as the *Positive Random Feature (PRF)*. Because its values are consistently positive, it facilitates more stable learning.

3 Optimal Distillation from Softmax Attention to Linear Attention

In this section, we develop a theoretical framework and a practical algorithm to select feature dimensions for linear attention, followed by a method to train these features efficiently.

3.1 Theoretical Background: Degrees of Freedom for Kernel Approximation

In this subsection, we lay the theoretical groundwork for selecting the optimal feature dimension in linear attention. Our analysis builds on kernel approximation theory and introduces the concept of statistical degrees of freedom (DoF), which quantifies the effective dimensionality required to approximate the attention kernel.

Finite dimensional approximation of kernel. To discuss generally, we consider a positive definite kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, and suppose that K has the expression $K(x, y) = \mathbb{E}_{z \sim \tau}[\phi(x; z)\phi(y; z)]$, where τ is a probability measure on a measurable set $\mathcal{Z}$, and $\phi : \mathbb{R}^d \times \mathcal{Z} \to \mathbb{R}$ is a feature map.

The simplest way to approximate K is to use the empirical kernel K' defined as $K'(x, y) = \frac{1}{M} \sum_{m=1}^{M} \phi(x; z'_m)\phi(y; z'_m)$, where $z'_1, \ldots, z'_M$ are i.i.d. samples from τ. According to the strong law of large numbers, K' converges to K almost surely as $M \to \infty$ for any fixed x and y. Rahimi and Recht (2007) show that, for any translation-invariant kernel K and compact set $\mathcal{Z} \subset \mathbb{R}^d$, K' uniformly converges to K at a rate of $\mathcal{O}(\log M/\sqrt{M})$.

Rather than focusing on the global error bound over a compact set, we strive to achieve a more accurate approximation by *leveraging the intrinsic structure of the input vectors*. In other words, we assume that the input vectors x and y are generated from a probability measure ρ on $\mathbb{R}^d$, and adjust the approximation scheme utilizing the information of ρ. To this end, we consider approximating K using i.i.d. samples $z_1, \ldots, z_M$ drawn from a distribution with density q w.r.t. the measure τ:

$$\widehat{K}(x, y) = \frac{1}{M} \sum_{m=1}^{M} \frac{1}{q(z_m)} \phi(x; z_m)\phi(y; z_m) = \Phi(x; z)^\top \Phi(y; z),$$

where $\Phi(x; z) = (M \cdot q(z))^{-1/2} \phi(x; z), z = [z_1, \ldots, z_M]^\top$. The kernel $\widehat{K}$ defines a finite-dimensional Reproducing Kernel Hilbert Space (RKHS) $\widehat{\mathcal{H}}$, which is expected to be an approximation of the RKHS $\mathcal{H}$ defined by K. Bach (2017) analyze the approximation error between $\mathcal{H}$ and $\widehat{\mathcal{H}}$, and show that it is characterized by the value $N_{q,\lambda}$ ($\lambda > 0$) defined as

$$N_{q,\lambda} = \sup_{z \in \mathcal{Z}} \frac{1}{q(z)} \left\langle \phi(\cdot; z), (\Sigma + \lambda I)^{-1} \phi(\cdot; z) \right\rangle_{L_2(\mathrm{d}\rho)},$$

where $\Sigma : L_2(\rho) \to L_2(\rho)$ is an integral operator defined by $(\Sigma f)(x) \coloneqq \langle K(x,\cdot), f\rangle_{L_2(\rho)}$. Specifically, they provide the following result.

Proposition 1 (Proposition 1 in Bach (2017))**.** *Suppose that $z_1, \dots, z_m$ are i.i.d. samples from the distribution with density q. Then, for any $\delta \in (0,1)$, if $M \geq 5N_{q,\lambda} \log \frac{16 N_{q,\lambda}}{\delta}$, it holds $\frac{1}{M}\sum_{m=1}^{M} q(z_m)^{-1} \|\phi(\cdot; z_m)\|^2_{L_2(\mathrm{d}\rho)} \leq \frac{2\operatorname{tr}\Sigma}{\delta}$ and*

$$\sup_{f:\|f\|_{\mathcal{H}}\leq 1} \inf_{\|\beta\|_2^2 \leq \frac{4}{M}} \left\| f - \sum_{m=1}^{M} \frac{\beta_m}{q(z_m)^{1/2}} \psi(z_m, \cdot) \right\|^2_{L^2(\rho)} \leq 4\lambda,$$

with probability at least $1-\delta$.

This proposition demonstrates that the necessary number of features M to achieve the approximation error λ is determined by $N_{q,\lambda}$, which depends on the density q. In other words, by effectively selecting the density q, it is possible to achieve an approximation error of λ with a smaller M.

One of the essential part of the proof of Proposition 1 is to bound the error between the operators Σ (corresponding to K) and its empirical approximation $\widehat{\Sigma} : L_2(\rho) \to L_2(\rho)$ (corresponding to $\widehat{K}$) defined as $(\widehat{\Sigma} f)(x) \coloneqq \langle \widehat{K}(x,\cdot), f\rangle_{L_2(\rho)}$. In the proof of Proposition 1, Bach (2017) analyze the error between Σ and $\widehat{\Sigma}$ as follows.

Lemma 2 (Bach (2017))**.** *Let $\Delta_\lambda \coloneqq (\Sigma+\lambda I)^{-1/2}(\Sigma - \hat{\Sigma})(\Sigma+\lambda I)^{-1/2}$. For any $\lambda, t > 0$, it holds*

$$\mathbb{P}[\|\Delta_\lambda\|_{\mathrm{op}} > t] \leq 2N_{q,\lambda}\left(1 + \frac{6}{t^2 \log^2\left(1 + \frac{Mt}{N_{q,\lambda}}\right)}\right) \exp\left(-\frac{Mt^2/2}{N_{q,\lambda}\cdot(1+\frac{t}{3})}\right).$$

This lemma provides the upper bound of the largest difference of eigenvalues between $(\Sigma + \lambda I)^{-1/2}\Sigma(\Sigma+\lambda I)^{-1/2}$ and $(\Sigma+\lambda I)^{-1/2}\widehat{\Sigma}(\Sigma+\lambda I)^{-1/2}$, which is controlled by $N_{q,\lambda}$.

Error bound between K and $\widehat{K}$. Proposition 1 shows that the elements of $\mathcal{H}$ can be approximated by those of $\widehat{\mathcal{H}}$ in terms of the $L^2(\rho)$ norm. However, when we consider distilling softmax attention into linear attention, we need *a guarantee that the error between K and $\widehat{K}$ is small*. To this end, we establish the following result using Lemma 2.

Theorem 3. *Let $\delta \in (0,1), \lambda > 0$, and $t \in (0, 9]$. Then, if $M \geq \frac{4N_{q,\lambda}}{t}\log\frac{32N_{q,\lambda}}{\delta t}$, it holds*

$$\left\|K - \widehat{K}\right\|^2_{L^2(\rho\otimes\rho)} \leq t\Big(\lambda^2 + 2C_K\lambda + \|K\|^2_{L^2(\rho\otimes\rho)}\Big),$$

with probability at least $1-\delta$, where $C_K \coloneqq \mathbb{E}_{x\sim\rho}[K(x,x)]$.

The proof can be found in Appendix A. This theorem suggests that the error between K and $\widehat{K}$ is also governed by $N_{q,\lambda}$. Therefore, by appropriately choosing the density q, we can efficiently reduce the approximation error.

Degrees of freedom and their optimality. Next, we aim to optimally reducing the required value of $N_{q,\lambda}$ for a fixed λ. Specifically, we consider the density q_λ defined as follows:

$$q_\lambda(v) = \frac{1}{\operatorname{tr}\Sigma(\Sigma+\lambda I)^{-1}} \left\langle \psi(v,\cdot), (\Sigma+\lambda I)^{-1}\psi(v,\cdot)\right\rangle_{L_2(\rho)}.$$

By setting $q = q_\lambda$, we can see that $N_{q_\lambda,\lambda} = N^*_\lambda \coloneqq \operatorname{tr}\Sigma(\Sigma+\lambda I)^{-1}$. The value N^*_λ is called the *degrees of freedom* (DoF).

When we use the i.i.d. samples $z_1, \dots, z_M$ drawn from a distribution with the density q_λ, the required number of samples M in Proposition 1 and Theorem 3 is proportional to $N^*_\lambda \log N^*_\lambda$, which is known to be the optimal (Bach (2017), Proposition 3). In particular, the degrees of freedom are always smaller than $N_{1,\lambda}$. This suggests that *the approximation error can be reduced by choosing the distribution q depending on the input distribution ρ* compared to the case we obtain the random samples from the distribution τ.

Algorithm 1 Selecting the Feature Dimension

Input: Set $\mathcal{X}$ of T sequences with length L, sample size $J \in \mathbb{N}$, tolerance $\lambda > 0$, cost $C \in \mathbb{N}$, original Transformer with S layers and H heads.

Fed the sequences in $\mathcal{X}$ into the pre-trained Transformer and collect the queries $Q_{s,h}$ and keys $K_{s,h}$ of all layers $s \in [S]$ and heads $h \in [H]$.

for $s = 1$ **to** S **do**

  for $h = 1$ **to** H **do**

    Randomly sample $x_1, \ldots, x_J$ from $Q_{s,h} \cup K_{s,h}$ and compute the Gram matrix $\widetilde{\Sigma}_{s,h} \in \mathbb{R}^{J \times J}$.

    Compute $\widetilde{N}_\lambda^{(s,h)} = \operatorname{tr} \widetilde{\Sigma}_{s,h} (\widetilde{\Sigma}_{s,h} + \lambda I)^{-1}$.

  end for

  Obtain $\widetilde{N}_\lambda^{(s)} = \max_{h \in [H]} \widetilde{N}_\lambda^{(s,h)}$.

end for

Set $t^{-1} = C \cdot \left(S^{-1} \sum_{s=1}^{S} \widetilde{N}_\lambda^{(s)} \right)^{-1}$.

Return: Feature dimension $M_s = \operatorname{round}(t^{-1} \widetilde{N}_\lambda^{(s)})$ for layers $s = 1, \ldots, S$.

3.2 Feature Dimension Selection via Degrees of Freedom

Given that the degrees of freedom N_λ^* determines the minimal required number of features, we now return to the attention kernel $K(x, y) = \exp\left(\frac{x^\top y}{\sqrt{d}}\right)$ and present a practical procedure to estimate it from data and use it to allocate layerwise dimensions. The overview of our method is presented in Algorithm 1.

Since N_λ^* is defined as the expectation over the data distribution, it cannot be computed in practice. Therefore, we approximate it using finite samples. Specifically, we define $\widetilde{N}_\lambda$ as an approximation of N_λ^*, given by $\widetilde{N}_\lambda = \operatorname{tr} \widetilde{\Sigma} (\widetilde{\Sigma} + \lambda I)^{-1}$, where $\widetilde{\Sigma} : \mathbb{R}^J \to \mathbb{R}^J$ ($J \in \mathbb{N}$) is a gram matrix of K, i.e., $\widetilde{\Sigma} = [K(x_i, x_j)]_{i \in [J], j \in [J]}$, which is the approximation of Σ based on finite samples $x_1, \ldots, x_J \sim \rho$.

For the attention kernel, the inputs are the queries and keys of the attention layers. Therefore, we compute the approximated degrees of freedom $\widetilde{N}_\lambda$ for each layer as follows:

1. Prepare T sequences of length L, and compute queries and keys for all sequences and tokens, resulting in LT queries and LT keys, respectively.
2. From the collected $2LT$ queries and keys, randomly sample J elements $x_1, \ldots, x_J$.
3. Compute approximated degrees of freedom $\widetilde{N}_\lambda$ defined above.

The degrees of freedom obtained through this procedure vary across heads and layers, as shown in Section 4. In typical Transformer implementations, the heads within the same layer are processed using a shared tensor, so it is desirable that the feature dimensions be identical as well. Hence, we define the degrees of freedom $\widetilde{N}_\lambda^{(s)}$ of the s-th layer as the maximum degrees of freedom among the heads.

According to the results of Theorem 3 (excluding the logarithm), we set the feature dimension of the s-th layer to $M^{(s)} = t^{-1} \widetilde{N}_\lambda^{(s)}$ for some $t > 0$. Then, the computational cost of inference matches that of linear attention with a fixed feature dimension given by $\frac{1}{S} \sum_{s=1}^{S} M^{(s)} = t^{-1} \cdot \frac{1}{S} \sum_{s=1}^{S} \widetilde{N}_\lambda^{(s)}$. In our experiments, we fixed λ in advance. Given a computational cost C, we determine the constant t such that the inference cost of the model matches the computational cost when we the feature dimension is uniformly set to C across all layers, which leads $t^{-1} = C \cdot \left(\frac{1}{S} \sum_{s=1}^{S} \widetilde{N}_\lambda^{(s)} \right)^{-1}$.

3.3 Layerwise Training of Features for Kernel Approximation

Theorem 3 guarantees that we can attain the optimal error between K and $\widehat{K}$ by choosing the density q appropriately, However, it is difficult to obtain such a density in practice. To obtain features z_i and $q(z_i)^{-1} (=: \alpha_i)$ ($i \in [M]$) that approximate the attention kernel as accurately as our theoretical bound, we propose to learn them from the training data.

Table 1: The dimension of features selected by our method. The feature dimensions of each layer are determined to match the average cost. For each row, the top three largest dimensions are highlighted in bold, and dimensions smaller than the top three are emphasized with underlines.

Model	Cost	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
GPT-2	64	**130**	**182**	35	44	42	65	**92**	33	28	34	46	39	-	-	-	-
Pythia-1B	128	**277**	138	**275**	132	204	167	115	142	**231**	64	96	73	40	52	34	9

One possible strategy is to train the features using the same loss function as the one used in the standard pre-training of Transformers. That said, this approach is computationally expensive, as it requires computing gradients across all layers simultaneously. Instead, we propose to train the features in a layerwise manner. Specifically, we consider two types of loss functions: **(i)** L^2 **loss** and **(ii) softmax loss**. To describe the details, let us assume that the set of queries $\mathcal{Q}_t = (q_{t,1}, \ldots, q_{t,L})$ and keys $\mathcal{K}_t = (k_{t,1}, \ldots, k_{t,L})$ of T sequences, each of length L, are given.

(i) L^2 **loss.** This loss function tries to minimize the error between the attention kernel K and its approximation $\widehat{K}$ in terms of the L^2 norm. That is, the features $z = (z_1, \ldots, z_M)$ are trained using the loss function defined as $\ell(z) = \frac{1}{LT} \sum_{t\in[T], l\in[L], l'\in[L]} \left(K(q_{t,l}, k_{t,l'}) - \widehat{K}(q_{t,l}, k_{t,l'}; z, \alpha) \right)^2$.

(ii) Softmax loss. We train the features to make the attention matrix and its approximation closer in terms of the cross entropy loss. In other words, we train the features $z = (z_1, \ldots, z_M)$ defined as $\ell(z) = -\frac{1}{LT} \sum_{t\in[T], l\in[L], l'\in[L]} p_{l'}(q_{t,l}, \mathcal{K}_t) \log \widehat{p}_{l'}(q_{t,l}, \mathcal{K}_t; z, \alpha)$, where $p_{l'}(q_{t,l}, \mathcal{K}_t) = \frac{\exp\left(q_{t,l}^\top k_{t,l'}\right)}{\sum_{j=1}^{L} \exp\left(q_{t,l}^\top k_{t,j}\right)}$, $\widehat{p}_{l'}(q_{t,l}, \mathcal{K}_t; z, \alpha) = \frac{\widehat{K}(q_{t,l}, k_{t,l'}; z, \alpha)}{\sum_{j=1}^{L} \widehat{K}(q_{t,l}, k_{t,j}; z, \alpha)}$.

In general, L^2 loss is more efficient than Softmax loss because it does not require additional nonlinear operations such as exp or normalization. In Section 4, we compare the performance of the two losses (i) and (ii) with the case where the features are directly learned using the pre-training task. Then, we demonstrate that learning with our proposed loss is as effective as training using pre-training loss.

4 Experiments

To evaluate the effectiveness of our method, we conduct experiments on two types of pre-trained Transformers: GPT-2 (Radford et al., 2019) and Pythia-1B (Biderman et al., 2023). As non-linear features for the linear attention, we use PRF (Choromanski et al., 2021), which we describe in Section 2. For dimension selection and training of features, we utilize the Wikipedia dataset[1].

4.1 Optimal Feature Dimensions *Vary across Layers and Heads*

First, we apply Algorithm 1 to the two pre-trained models and determine the feature dimensions for each layer. We set the cost C to be the same as the head size (64 for GPT-2 and 128 for Pythia-1B), following prior work that sets the feature dimension of each layer to match the head size. Furthermore, we set $\lambda = 2^{-4}$ for GPT-2 and $\lambda = 2^{-8}$ for Pythia-1B. We present the feature dimensions selected by our method in Table 1. We also show the estimated DoF for each head in GPT-2 in Table 2. We summarize the key findings from these results below.

Table 2: DoF with $\lambda = 2^{-8}$ of each head. The row and column represent the layer and head, respectively. The maximum value of each layer is highlighted in bold.

Layer	1	2	3	4	5	6	7	8	9	10	11	12
1	4.7	8.1	4.5	2.0	3.6	16.2	4.5	2.6	4.1	6.4	5.0	**150.0**
2	10.5	28.5	155.1	44.6	**173.8**	51.1	9.9	16.5	17.5	8.1	88.9	19.5
3	9.9	8.1	1.8	2.7	3.4	2.5	**24.5**	5.8	2.8	2.6	12.1	6.4
4	15.1	3.0	1.4	1.7	**39.8**	12.1	1.6	1.5	1.5	2.5	12.6	1.9
5	1.8	1.8	11.3	1.8	42.1	2.3	4.0	7.1	22.6	3.3	**31.6**	1.1
6	8.9	3.1	3.7	3.5	3.7	3.0	6.9	7.5	4.7	52.4	21.4	**66.6**
7	3.3	6.9	68.9	14.4	43.5	17.2	**107.4**	6.0	4.3	1.8	32.2	3.0
8	3.6	11.9	3.1	12.3	7.6	17.5	**33.0**	9.1	4.9	5.1	2.3	12.6
9	**24.9**	6.5	19.6	9.3	5.5	3.2	10.2	3.1	5.7	16.7	22.4	3.9
10	18.5	13.1	15.9	3.3	22.7	**29.9**	14.1	6.4	20.2	5.1	3.9	6.2
11	7.9	6.6	26.1	19.8	18.2	3.3	8.7	6.7	**43.2**	2.3	17.1	13.5
12	5.1	21.8	22.5	10.6	30.0	**39.8**	27.2	31.3	5.0	20.0	2.8	2.6

[1]The dataset can be accessed through the Hugging Face library: `https://huggingface.co/datasets/legacy-datasets/wikipedia`

Table 3: The results of distillation for GPT-2. *Fix*, *DoF* and *DoF + Clip* in the first column imply the strategies of dimension selection. For all downstream tasks, the evaluation metric is accuracy. The top three highest-performing are highlighted in bold.

Strategy	Method	PiQA	logiQA	ARC-E	ARC-C	Winogrande	MMLU	WSC	Average
Original GPT-2		**0.5985**	0.3103	**0.3325**	0.3003	0.5122	0.2789	**0.6538**	**0.4266**
—	DiJiang	0.5065	0.2550	0.2113	0.2244	0.4846	0.2639	0.4615	0.3409
	Performer	0.5468	0.2934	0.3039	0.2747	0.4996	0.2517	0.5962	0.3952
Fix	direct	0.5832	**0.3195**	0.2921	0.2995	**0.5335**	0.2552	0.6154	0.4141
	softmax	0.5718	0.2673	0.2479	**0.3029**	0.5020	0.2634	0.6154	0.3958
	L^2	0.5822	**0.3195**	0.2483	0.2773	0.5107	0.2520	0.5962	0.3980
DoF	direct	0.5669	0.3011	**0.3241**	**0.3012**	**0.5280**	0.2712	**0.6346**	0.4182
	softmax	0.5751	0.3026	0.3224	0.2995	**0.5328**	0.2564	**0.6442**	**0.4190**
	L^2	0.5664	0.3088	0.2736	0.2824	0.4972	0.2608	0.5865	0.3965
DoF + Clip	direct	**0.5892**	**0.3164**	0.3136	0.2952	0.5075	**0.2993**	**0.6346**	**0.4223**
	softmax	**0.5860**	0.3026	**0.3401**	0.2816	0.4996	**0.2832**	**0.6346**	0.4182
	L^2	0.5822	**0.3164**	0.2942	**0.3063**	0.5091	**0.2799**	0.5673	0.4079

The evolution of effective dimensionality across layers. The result indicates that the effective dimensionality varies significantly across layers. This emphasizes the difference of the complexity of the roles played by each layer. Taking a closer look at the results, we observe that the feature dimensions selected by our method are generally *larger in the early and middle layers*, and *small in the latter layers*. This shows that Transformers engage in complex token interactions from the shallow layers to the middle layers, and relatively simple processing in the later layers. This observation aligns with the insights obtained in previous studies on fully-connected/convolutional neural networks (Arora et al., 2018; Ravichandran et al., 2019; Suzuki et al., 2020). Furthermore, it is worth noting that, between the early layers and the middle layers, there are several layers with relatively small dimensions.

Comparison of DoF across heads in the same layer. We additionally observe that the degrees of freedom are largely different across heads as well. Interestingly, *most of the heads have much smaller degrees of freedom than the maximum of each layer*. This indicates that the heads performing complex processing are limited to a few within a single layer.

4.2 Dimension Selection with Layerwise Training *Enhances the Performance*

We now evaluate the effectiveness of our method in distillation from softmax attention for two pre-trained models. In these experiments, we examine the following two claims: (i) selecting feature dimensions using our DoF-based method improve performance compared to using fixed dimensions, and (ii) our efficient layerwise training match the performance of full end-to-end training.

We consider three types of dimension selection strategies: *Fix* is the method that sets the feature dimensions to the same value across all layers, which is conventionally used in existing works. *DoF* is the method that selects the feature dimensions using Algorithm 1. We also consider *DoF + Clip*, in which we clamp the feature dimensions to be at most the head size. This strategy is expected to maintain the performance of Fix, while reducing the computational cost

For training the feature maps during distillation, we consider three types of loss functions: the first is *direct* loss, which we refer to the cross-entropy loss for next-token prediction, which provides the best performance in the pre-training task but requires full end-to-end training. the remaining two are *softmax* loss and L^2 loss, which are the layerwise losses introduced in Section 3.3, and more efficient and allow independent training per layer. We will show that the layerwise training achieves performance comparable to the direct loss, while significantly reducing training cost.

As comparison baselines, we evaluate against Performer(Choromanski et al., 2021) and DiJiang(Chen et al., 2024). Both methods employ PRF for linear attention but do not incorporate any data-driven dimension selection. DiJiang additionally utilizes quasi Monte Carlo sampling of features. The learnable parameters of DiJiang are trained on the same dataset used in distillation.

To assess downstream performance, we fine-tune all distilled models (as well as the original Transformers) on a suite of seven tasks: PiQA (Bisk et al., 2020), logiQA (Liu et al., 2020), ARC-Easy/Challenge (Clark et al., 2018), Winogrande (Sakaguchi et al., 2019), WSC (Levesque et al., 2011), and MMLU (Hendrycks et al., 2021). Further details on the training settings and datasets are provided in Appendix C.

Table 4: Comparison of the inference time per sequence with 1,000 tokens for different feature dimensions.

	Fix	DoF	DoF + Clip
Speed (sec/sequence)	15.589	15.579	14.662

Table 5: Comparison of the consuming time per each sample during training for different loss types.

Type of loss	direct	softmax	L^2
Speed (ms/sample)	109.4	90.4	60.6

Table 6: The downstream accuracy for Pythia-1B. We use softmax loss for distillation.

Method	PiQA	logiQA	ARC-E	ARC-C	Winogrande	MMLU	WSC	Average
Original Pythia-1B	0.6213	0.2995	0.3295	0.3072	0.5146	0.2883	0.6346	0.4279
Fix	0.5691	0.2965	0.2849	0.2679	0.5051	0.2903	0.5962	0.4014
DoF	0.5860	0.3026	0.3110	0.3106	0.4949	0.2811	0.5962	0.4118

Results for GPT-2. We first show the experimental results for GPT-2 in Table 3. We outline the noteworthy observations below.

- **Enhanced performance through dimension selection:** Distilled models using feature dimensions selected by Algorithm 1 (DoF) achieve higher or near-equal performance compared to those using fixed dimensions (Fix). In particular, in the average performance across all tasks (see rightmost column of Table 3), when using direct or softmax loss, the distilled models with DoF significantly outperforms the ones with Fix. This indicates that our method effectively captures the varying complexity of each layer, leading to improved accuracy across downstream tasks.
- **Effectiveness of DoF + Clip for efficiency:** The DoF + Clip strategy, which caps feature dimensions at the head size, achieves comparable or better performance than the fixed-dimension approach. Notably, it also yields faster inference (Table 4), showing that selectively reducing feature dimensions based on DoF makes the model efficient without compromising performance.
- **Stronger results compared to prior baselines:** Our distilled models surpass both Performer and DiJiang, which use fixed-dimension linear attention and (quasi) Monte Carlo feature sampling. Additionally, the average performace our distilled model is comparable to (and sometimes win) the original GPT-2 model. This shows that our data-adaptive, layer-specific dimension selection combined with learned features yields a more accurate approximation of softmax attention.
- **Efficient and effective layerwise feature training:** Using the softmax loss, distilled models perform comparably to those trained end-to-end with the standard pre-training loss (direct). As shown in Table 5, layerwise training is more efficient than direct training. Although the L2 loss yields slightly lower performance than both direct and softmax losses, it still outperforms prior baselines while offering notable training efficiency gains.

Results for Pythia-1B. We also carried out experiments on distilling Pythia-1B. In this experiment, the model was distilled using the softmax loss. The results are presented in Table 6. The distilled model with feature dimensions selected by our method performs better than the model with fixed dimension on average, and become comparable to the original Pythia-1B. This result indicates the efficacy of our dimension selection and layerwise training when distilling large models.

5 Conclusion

We proposed a principled method for selecting feature dimensions in linear attention, grounded in statistical theory on finite-dimensional kernel approximation, and demonstrated its effectiveness in distilling softmax attention. Our approach adaptively assigns dimensions to each layer based on the statistical degrees of freedom of attention kernel, and trains nonlinear features in a layerwise manner to reduce computation. Experiments on GPT-2 and Pythia-1B show that our method improves performance without increasing inference cost, highlighting the importance of layer-specific design for efficient linear attention models.

Limitation and Future Work. While our method assigns feature dimensions adaptively per layer, we use a shared dimension across all heads within a layer to maintain implementation efficiency. Interestingly, our analysis revealed that the DoFs vary significantly even among heads in the same layer. Exploiting this head-level variability—for instance, through head pruning or sparse feature allocation—could further enhance performance and efficiency. In addition, although our experiments focus on moderate-scale language models, applying our approach to larger models and models for other modalities remain an important direction to validate its scalability.

References

S. Arora, R. Ge, B. Neyshabur, and Y. Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International conference on machine learning*, pages 254–263. PMLR, 2018.

F. Bach. On the equivalence between kernel quadrature rules and random feature expansions. *Journal of machine learning research*, 18(21):1–38, 2017.

A. Bick, K. Li, E. P. Xing, J. Z. Kolter, and A. Gu. Transformers to ssms: Distilling quadratic knowledge to subquadratic models. *Advances in neural information processing systems*, 38, 2024.

S. Biderman, H. Schoelkopf, Q. G. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.

Y. Bisk, R. Zellers, R. L. Bras, J. Gao, and Y. Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

H. Chen, Z. Liu, X. Wang, Y. Tian, and Y. Wang. Dijiang: Efficient large language models through compact kernelization. *arXiv preprint arXiv:2403.19928*, 2024.

K. M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

L. Dong, S. Xu, and B. Xu. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5884–5888. IEEE, 2018.

A. Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

D. Han, Z. Wang, Z. Xia, Y. Han, Y. Pu, C. Ge, J. Song, S. Song, B. Zheng, and G. Huang. Demystify mamba in vision: A linear attention perspective. *Advances in neural information processing systems*, 38, 2024.

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

J. Kasai, H. Peng, Y. Zhang, D. Yogatama, G. Ilharco, N. Pappas, Y. Mao, W. Chen, and N. A. Smith. Finetuning pretrained transformers into rnns. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10630–10643, 2021.

A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

H. J. Levesque, E. Davis, and L. Morgenstern. The Winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, volume 46, page 47, 2011.

J. Liu, L. Cui, H. Liu, D. Huang, Y. Wang, and Y. Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*, 2020.

S. Massaroli, M. Poli, D. Fu, H. Kumbong, R. Parnichkun, D. Romero, A. Timalsina, Q. McIntyre, B. Chen, A. Rudra, et al. Laughing hyena distillery: Extracting compact recurrences from convolutions. *Advances in Neural Information Processing Systems*, 36, 2024.

H. Peng, N. Pappas, D. Yogatama, R. Schwartz, N. A. Smith, and L. Kong. Random feature attention. 2021.

Z. Qin, W. Sun, H. Deng, D. Li, Y. Wei, B. Lv, J. Yan, L. Kong, and Y. Zhong. cosformer: Rethinking softmax in attention. In *International Conference on Learning Representations*, 2022.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

T. R. Ralambomihanta, S. Mohammadzadeh, M. S. N. Islam, W. Jabbour, and L. Liang. Scavenging hyena: Distilling transformers into long convolution models. *arXiv preprint arXiv:2401.17574*, 2024.

K. Ravichandran, A. Jain, and A. Rakhlin. Using effective dimension to analyze feature transformations in deep neural networks. In *ICML 2019 Workshop on Identifying and Understanding Deep Learning Phenomena*, 2019.

K. Sakaguchi, L. B. Ronan, B. Chandra, and C. Yejin. Winogrande: An adversarial winograd schema challenge at scale. 2019.

H. Sakamoto and K. Sato. Data-driven h2 model reduction for linear discrete-time systems. *arXiv preprint arXiv:2401.05774*, 2024.

T. Suzuki, H. Abe, T. Murata, S. Horiuchi, K. Ito, T. Wachi, S. Hirai, M. Yukishima, and T. Nishimura. Spectral pruning: Compressing deep neural networks via spectral analysis and its generalization error. *International Joint Conferences on Artificial Intelligence*, 2020.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 2017.

J. Wang, D. Paliotta, A. May, A. M. Rush, and T. Dao. The mamba in the llama: Distilling and accelerating hybrid models. *arXiv preprint arXiv:2408.15237*, 2024.

S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

—— Appendix ——

A Proof of Theorem 3

Here, we restate Lemma 2, which can be found in the proof of Proposition 1 in Bach (2017).

Lemma 4. *Let* $\Delta_\lambda \coloneqq (\Sigma+\lambda I)^{-1/2}(\Sigma-\hat{\Sigma})(\Sigma+\lambda I)^{-1/2}$. *For any* $\lambda, t>0$, *it holds*

$$\mathbb{P}[\|\Delta_\lambda\|_{\mathrm{op}} > t] \le 2\hat{d}(\lambda)\left(1+\frac{6}{t^2\log^2(1+nt/\hat{d}(\lambda))}\right)\exp\left(-\frac{nt^2/2}{\hat{d}(\lambda)(1+t/3)}\right).$$

This lemma produces the following high probability bound.

Lemma 5. *For any* $\lambda, \delta \in (0,1)$, *if*

$$n \ge \frac{4\hat{d}(\lambda)}{t^2}\log\frac{32\hat{d}(\lambda)}{\delta t^2},$$

it holds

$$-t(\Sigma+\lambda I) \preceq \Sigma-\hat{\Sigma} \preceq t(\Sigma+\lambda I)$$

with probability at least $1-\delta$.

Proof. We first note that

$$\begin{aligned}
&2\hat{d}(\lambda)\left(1+\frac{6}{t^2\log^2(1+nt/\hat{d}(\lambda))}\right)\exp\left(-\frac{nt^2/2}{\hat{d}(\lambda)(1+t/3)}\right)\\
&\qquad\le \frac{2\hat{d}(\lambda)}{t^2}\underbrace{\left(t^2+\frac{6}{\log^2(1+nt/\hat{d}(\lambda))}\right)}_{\text{(a)}}\underbrace{\exp\left(-\frac{nt^2/2}{\hat{d}(\lambda)(1+t/3)}\right)}_{\text{(b)}}
\end{aligned}$$

Let us consider

$$n \ge B\frac{\hat{d}(\lambda)}{t^2}\log\frac{C\hat{d}(\lambda)}{\delta t^2},$$

for some constants $B, C>0$ which will be determined later. First, we consider the factor (b). Then, for any $t\in(0,3]$, we have

$$\text{(b)} \le \exp\left(-\frac{B/2}{1+t/3}\log\frac{C\hat{d}(\lambda)}{\delta t^2}\right) = \left(\frac{\delta t^2}{C\hat{d}(\lambda)}\right)^{B/4}.$$

Next, we consider the factor (a). Let $D>0$ be a constant to be determined later. Then, if $\hat{d}(\lambda)/t \ge D$, we have

$$\frac{nt}{\hat{d}(\lambda)} \ge \frac{B}{t}\log\frac{C\hat{d}(\lambda)}{\delta t^2} \ge \frac{B}{3}\log\frac{C\hat{d}(\lambda)}{3\delta t} \ge \frac{B}{3}\log\frac{CD}{3},$$

which implies

$$\text{(a)} \le 9+\frac{6}{\log^2\left(1+\frac{B}{3}\log\frac{CD}{3}\right)}.$$

On the other hand, if $\hat{d}(\lambda)/t \le D$, using $n\ge 1$, we have

$$\text{(a)} \le 9+\frac{6}{\log^2\left(1+\frac{1}{D}\right)}.$$

Finally, we choose $B, C, D>0$. If we set $B=4$, we obtain the upper bound of the probability as

$$\frac{2\hat{d}(\lambda)}{t^2}\cdot\text{(a)}\cdot\frac{\delta t^2}{C\hat{d}(\lambda)} = \frac{2\delta}{C}\cdot\text{(a)}.$$

Additionally, if we set $C = 32, D = 1/2$, we have (a) ≤ 15. Thus, if we set

$$n \geq \frac{4\hat{d}(\lambda)}{t^2} \log \frac{32\hat{d}(\lambda)}{\delta t^2},$$

we have $\|\Delta_\lambda\|_{\mathrm{op}} \leq t$ with probability at least $1 - \delta$.

The result $\|\Delta_\lambda\|_{\mathrm{op}} \leq t$ implies that

$$-tI \preceq (\Sigma + \lambda I)^{-1/2}(\Sigma - \hat{\Sigma})(\Sigma + \lambda I)^{-1/2} \preceq tI.$$

From the right-hand side inequality, we have

$$\Sigma - \hat{\Sigma} \preceq t(\Sigma + \lambda I).$$

Indeed, for any $f \in L_2(\mathrm{d}\rho)$, we have

$$\begin{aligned}
&\left\langle f, (t(\Sigma + \lambda I) - (\Sigma - \hat{\Sigma}))f \right\rangle_{L_2(\mathrm{d}\rho)} \\
&= \left\langle f, (\Sigma + \lambda I)^{1/2}(tI - (\Sigma + \lambda I)^{-1/2}(\Sigma - \hat{\Sigma})(\Sigma + \lambda I)^{-1/2})(\Sigma + \lambda I)^{1/2} f \right\rangle_{L_2(\mathrm{d}\rho)} \\
&= \left\langle (\Sigma + \lambda I)^{1/2} f, (tI - (\Sigma + \lambda I)^{-1/2}(\Sigma - \hat{\Sigma})(\Sigma + \lambda I)^{-1/2})(\Sigma + \lambda I)^{1/2} f \right\rangle_{L_2(\mathrm{d}\rho)} \\
&\geq 0.
\end{aligned}$$

since $(\Sigma + \lambda I)^{1/2}$ is self-adjoint. Similarly, we have $\Sigma - \hat{\Sigma} \succeq -t(\Sigma + \lambda I)$. This completes the proof. □

The following lemma will be used to connect Lemma 5 with the approximation error between k and $\hat{k}$.

Lemma 6. *Let $k_1 : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}, k_2 : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be positive definite kernels. We define $k_{12} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ as*

$$k_{12}(x, y) = k_1(x, y) \cdot k_2(x, y).$$

Then, k_{12} is also a positive definite kernel.

Proof. Using Schur product theorem, we can see that the gram matrix of k_{12} is positive definite. This completes the proof. □

Theorem 7. *Let $\delta \in (0, 1), \lambda > 0$, and $t \in (0, 3]$. Then, if*

$$n \geq \frac{4\hat{d}(\lambda)}{t^2} \log \frac{32\hat{d}(\lambda)}{\delta t^2},$$

it holds

$$\begin{aligned}
&\int_{\mathbb{R}^d} \int_{\mathbb{R}^d} \left(\hat{k}(x, y) - k(x, y)\right)^2 \mathrm{d}\rho(x)\mathrm{d}\rho(y) \\
&\quad \leq t^2 \left(\lambda^2 + 2\lambda \int_{\mathbb{R}^d} k(x, x)\mathrm{d}\rho(x) + \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} k(x, y)^2 \mathrm{d}\rho(x)\mathrm{d}\rho(y)\right),
\end{aligned}$$

with probability at least $1 - \delta$.

Proof. Let $S : L_2(\mathrm{d}\rho) \to L_2(\mathrm{d}\rho)$ be the operator defined as

$$Sf = \int_{\mathbb{R}^d} \left\{ t^2 (k(x, \cdot) + \lambda k_I(x, \cdot))^2 - \left(\hat{k}(x, \cdot) - k(x, \cdot)\right)^2 \right\} f(x)\mathrm{d}\rho(x),$$

where

$$k_I(x, y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases}$$

We first prove that S is a positive definite operator. It holds

$$\begin{aligned}
&\Big(t^2(k(x,y)+\lambda k_I(x,y))^2-\hat{k}(x,y)-k(x,y)\Big)^2\\
&=\Big(t(k(x,y)+\lambda k_I(x,y))+\Big(\hat{k}(x,y)-k(x,y)\Big)\Big)\\
&\qquad\qquad\times\Big(t(k(x,y)+\lambda k_I(x,y))-\Big(\hat{k}(x,y)-k(x,y)\Big)\Big)\\
&=k_1(x,y)k_2(x,y),
\end{aligned}$$

where

$$\begin{aligned}
k_1(x,y)&=t(k(x,y)+\lambda k_I(x,y))+\Big(\hat{k}(x,y)-k(x,y)\Big),\\
k_2(x,y)&=t(k(x,y)+\lambda k_I(x,y))-\Big(\hat{k}(x,y)-k(x,y)\Big).
\end{aligned}$$

Recall that

$$\begin{aligned}
\Big(t(\Sigma+\lambda I)-(\Sigma-\hat{\Sigma})\Big)f&=\int_{\mathbb{R}^d}\left(t(k(x,\cdot)+\lambda k_I(x,\cdot))^2-\Big(\hat{k}(x,\cdot)-k(x,\cdot)\Big)^2\right)f(x)\mathrm{d}\rho(x),\\
\Big(t(\Sigma+\lambda I)+(\Sigma-\hat{\Sigma})\Big)f&=\int_{\mathbb{R}^d}\left(t(k(x,\cdot)+\lambda k_I(x,\cdot))^2+\Big(\hat{k}(x,\cdot)-k(x,\cdot)\Big)^2\right)f(x)\mathrm{d}\rho(x).
\end{aligned}$$

Therefore, Lemma 5 implies that k_1 and k_2 are positive definite kernels. Using Lemma 6, we can see that k_1k_2 is also a positive definite kernel. Thus, $S: f\mapsto\int_{\mathbb{R}^d}k_1(x,\cdot)k_2(x,\cdot)f(x)\mathrm{d}\rho(x)$ is a positive definite operator.

The argument above implies that, for any $f\in L_2(\mathrm{d}\rho)$, it holds $\langle f,Sf\rangle_{L_2(\mathrm{d}\rho)}\geq 0$. Specifically, if we set $f(x)=1$, we have

$$\int_{\mathbb{R}^d}\int_{\mathbb{R}^d}\left(t^2(k(x,y)+\lambda k_I(x,y))^2-\Big(\hat{k}(x,y)-k(x,y)\Big)^2\right)\mathrm{d}\rho(x)\mathrm{d}\rho(y)\geq 0,$$

i.e.,

$$\begin{aligned}
&\int_{\mathbb{R}^d}\int_{\mathbb{R}^d}\Big(\hat{k}(x,y)-k(x,y)\Big)^2\mathrm{d}\rho(x)\mathrm{d}\rho(y)\\
&\leq t^2\int_{\mathbb{R}^d}\int_{\mathbb{R}^d}(k(x,y)+\lambda k_I(x,y))^2\mathrm{d}\rho(x)\mathrm{d}\rho(y)\\
&=t^2\left(\lambda^2+2\lambda\int_{\mathbb{R}^d}k(x,x)\mathrm{d}\rho(x)+\int_{\mathbb{R}^d}\int_{\mathbb{R}^d}k(x,y)^2\mathrm{d}\rho(x)\mathrm{d}\rho(y)\right)
\end{aligned}$$

This completes the proof. □

B The Experimental Results on Next-Token Prediction

In this section, we report the performance of next-token prediction for the distilled models in Section 4. The results are presented in Table 7. Here, we highlight the following two key observations:

- Among the three types of loss, the layerwise losses sometimes underperform compared to the "direct" loss. This is natural because the cross entropy loss for next-token prediction is used in "direct", while L^2 loss (Softmax loss) just aims to make the attention kernel (attention weights) and its approximation close. As we reported in Section 4, for the downstream tasks, the models distilled with layerwise loss have comparable performance to the models distilled with "direct" loss.
- We observe that the models distilled using (layerwise) softmax loss outperform DiJiang, in which the learnable parameters are directly trained using next-token prediction loss. This result emphasizes the validity of our approach in training the features in linear attention.

Table 7: The performance of next-token prediction for the models distilled from GPT-2.

	direct	softmax	L^2
Fix	3.9657	5.4082	6.2453
DoF	5.2651	4.0170	6.6802
DoF + Clip	5.4547	4.0355	6.2378

Baseline method	Loss
DiJiang	5.5965
Performer	8.1586
Original GPT-2	3.3558

Table 8: Summary of learning rates.

Model	PiQA	logiQA	ARC-E	ARC-C	Winogrande	MMLU	WSC
GPT-2	1e-4	5e-5	1e-4	1e-5	5e-4	1e-6	1e-6
Pythia-1B	1e-5	5e-5	5e-5	5e-5	5e-5	5e-6	5e-5

C Details of the Experiment

C.1 Computational Resources and Hyperparameters

We provide the details of the experimental settings as follows:

- Experiments for GPT-2 are conducted on four devices of A100 40GB. Experiments for Pythia-1B are conducted on four devices of A100 80GB.
- For Wikipedia dataset, we randomly sample 10% segment, and use it as one dataset.
- The context lengths for GPT-2 and Pythia-1B were set to 1024 and 2048, respectively.
- Training for distillation (learning features in proposed method / learning parameters in DiJiang) is conducted over 1 epoch. This consumes about 0.5 day for GPT-2 and 1 day for Pythia-1B.
- For the downstream tasks except for MMLU. training is conducted over three epochs for the tasks except for MMLU, and the best accuracy among three epochs is reported. For MMLU, we train the model for one epoch, and the accuracy for the last checkpoint is reported.
- As for the learning rates of distillation,
  - when using DiJiang, we set `0.02`.
  - when training feature maps, we set `0.02` for $z_1, \ldots, z_M$ and `0.2` for $\alpha_1, \ldots, \alpha_M$.
- The learning rates of downstream task are chosen to maximize the accuracy when we fine-tune the original model with softmax attention, and the same learning rates are used for the distilled model. The choices of learning rates are `1e-6, 5e-6, 1e-5, 5e-5, 1e-4, 5e-4` for GPT-2, and `1e-7, 5e-7, 1e-6, 5e-6, 1e-5, 5e-5` for Pythia-1B. The selected learning rates are summarized in Table 8.
- The batch sizes are summarized in Table 9. When out-of-memory error occurs, we utilized gradient accumulation.

C.2 Datasets

All the datasets used in this paper are publicly available from HuggingFace Datasets library. The dataset URLs and licenses are summarized in Table 10.

Table 9: Summary of batch sizes.

Model	Distillation	PiQA	logiQA	ARC-E	ARC-C	Winogrande	MMLU	WSC
GPT-2	128	128	64	64	64	128	64	128
Pythia-1B	64	64	32	32	32	64	32	64

Table 10: Summary of datasets.

Dataset	URL	License
Wikipedia	`https://huggingface.co/datasets/legacy-datasets/wikipedia`	CC BY-SA 3.0
PiQA	`https://huggingface.co/datasets/piqa`	AFL-3.0
logiQA	`https://huggingface.co/datasets/EleutherAI/logiqa`	Apache License, Version 2.0
ARC-E	`https://huggingface.co/datasets/allenai/ai2_arc`	CC BY-SA 4.0
ARC-C	`https://huggingface.co/datasets/allenai/ai2_arc`	CC BY-SA 4.0
Winogrande	`https://huggingface.co/datasets/allenai/winogrande`	Apache License, Version 2.0
MMLU	`https://huggingface.co/datasets/mmlu`	MIT
WSC	`https://huggingface.co/datasets/wsc`	CC BY 4.0

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 claims in the abstract and introduction are clearly aligned with the theoretical and empirical contributions presented in the paper (see Section 3 and 4).

   Guidelines:

   - The answer NA 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 NA 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 paper includes a "Limitations and Future Work" paragraph in Section 5, which discusses several open issues.

   Guidelines:

   - The answer NA 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: The theoretical development includes detailed assumptions and complete proofs, especially in Section Section 3 and Appendix A. The derivation builds upon known kernel approximation theory (e.g., Bach 2017), and all critical assumptions are stated clearly.

Guidelines:

- The answer NA 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 provides detailed experimental setups including datasets used, model configurations, batch sizes (Table 9), learning rates, number of epochs, and training strategies (Appendix C). This level of detail supports reproducibility.

Guidelines:

- The answer NA 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: The code is provided in the supplemental material. The all data used in the experiments are publicly available datasets (Wikipedia, PiQA, logiQA, ARC-E, ARC-C, Winogrande, MMLU and WSC).

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Appendix C describes the training settings in detail, including learning rates, number of epochs, batch sizes, and hardware specifications. The dataset and its usage are also specified.

   Guidelines:

   - The answer NA 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: [No]

   Justification: While the experimental results are averaged across tasks and compared with strong baselines, we did not report statistical significance such as error bars or confidence intervals due to computational constraints.

   Guidelines:

   - The answer NA 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: The paper provides detailed information on computational resources in Appendix C. It specifies the computational resources used for training and inference, including the type of compute workers, memory, and time of execution.

   Guidelines:

   - The answer NA 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 was conducted in accordance with the NeurIPS Code of Ethics. We used only publicly available datasets and open-source pre-trained models. No personally identifiable information, private data, or human subjects were involved. We have also ensured that all evaluations preserve fairness and transparency.

   Guidelines:

   - The answer NA 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: [NA]

Justification: Our paper contributes to the efficiency and scalability of Transformer-based models, which does not have direct societal impacts.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our work does not involve the release of a new dataset or a novel pretrained language model. Instead, we focus on a method to distill existing models using already public data and models. As such, the paper poses no significant risk of misuse that would necessitate additional safeguards.

Guidelines:

- The answer NA 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: We properly cite the papers or URLs for the datasets and models used in our experiments. The URLs and licenses for the datasets and models are provided in Table 10.

Guidelines:

- The answer NA 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: [NA]

Justification: The paper does not introduce new assets such as datasets or model code. It uses publicly available datasets and pre-trained models, and focuses on theoretical and empirical improvements to existing architectures.

Guidelines:

- The answer NA 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: [NA]

Justification: The paper does not involve any crowdsourcing or research with human subjects.

Guidelines:

- The answer NA 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: [NA]

Justification: This paper does not involve any human subjects, and therefore IRB approval is not applicable.

Guidelines:

- The answer NA 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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: LLMs are not used as tools for developing or augmenting the core methodology. The work focuses on distilling attention mechanisms and does not involve using LLMs in a novel or generative capacity within the research method itself.

Guidelines:

- The answer NA 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 (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.