S^3Attention: Improving Long Sequence Attention with Smoothed Skeleton Sketching

Xue Wang, Tian Zhou, Jianqing Zhu, Jialin Liu, Kun Yuan, Tao Yao, Wotao Yin, Rong Jin, HanQin Cai, *Senior Member, IEEE*

***Abstract*—Attention-based models have achieved many remarkable breakthroughs in numerous applications. However, the quadratic complexity of Attention makes the vanilla Attention-based models hard to apply to long sequence tasks. Various improved Attention structures are proposed to reduce the computation cost by inducing low rankness and approximating the whole sequence by sub-sequences. The most challenging part of those approaches is maintaining the proper balance between information preservation and computation reduction: the longer sub-sequences used, the better information is preserved, but at the price of introducing more noise and computational costs. In this paper, we propose a smoothed skeleton sketching based Attention structure, coined S^3Attention, which significantly improves upon the previous attempts to negotiate this trade-off. S^3Attention has two mechanisms to effectively minimize the impact of noise while keeping the linear complexity to the sequence length: a smoothing block to mix information over long sequences and a matrix sketching method that simultaneously selects columns and rows from the input matrix. We verify the effectiveness of S^3Attention both theoretically and empirically. Extensive studies over Long Range Arena (LRA) datasets and six time-series forecasting show that S^3Attention significantly outperforms both vanilla Attention and other state-of-the-art variants of Attention structures.**

***Index Terms*—Deep learning, Attention, Skeleton Approximation, Dimensionality Reduction.**

I. Introduction

MODERN Attention-based models, first introduced in [1], have made significant contributions in several areas of artificial intelligence, including natural language processing (NLP) [2–5], computer vision (CV) [6–10], and time series forecasting [11, 12]. These models have also been applied in a variety of long sequences mining tasks, such as traffic [13, 14], health care [15, 16], retail [17, 18], security industry [19], and web applications [20–22]. The Attention scheme efficiently captures long-term global and short-term local correlations when the length of the token sequences is relatively small. However, due to the quadratic complexity of standard Attention, many approaches have been developed to reduce the computational complexity for long sequences (e.g., [23]). Most of them try to exploit the special patterns of the Attention matrix, such as low rankness, locality, sparsity, or graph structures. One group of approaches is to build a linear approximation for the $\mathrm{softmax}$ operator (e.g., [24–27]). Despite the efficiency of the linear approximation, these approximation methods often perform worse than the original $\mathrm{softmax}$ based Attention. More discussion of efficient Attention-based models for long sequences can be found in the section of related work.

In this work, we focus on approaches that assume a low-rank structure of the input matrix. They approximate the global information in long sequences by sub-sequences (i.e., short sequences) of landmarks, and only compute Attention between queries and selected landmarks (e.g., [23, 28–30]). Although those models enjoy linear computational cost and often better performance than vanilla Attention, they face one major challenge, i.e., how to balance between information preservation and noise reduction. By choosing a larger number of landmarks, we are able to preserve more global information but at the price of introducing more noise into the sequential model and more computational cost.

In this work, we propose an efficient Attention architecture, termed S^3Attention, that introduces two mechanisms to address the balance explicitly as illustrated in Figure 1. First, we introduce a smoothing block into the Attention architecture. It effectively mixes global information over the long sequences by the Fourier transformation and local information over the sequences by a convolution kernel. Through the information mixture, we are able to reduce the noise for individual tokens over the sequences, and at the same time, improve their representativeness for the entire sequences. Second, we introduce a matrix sketching technique to approximate the input matrix by a smaller number of rows and columns. Standard Attention can be seen as reweighing the columns of the value matrix. Important columns are assigned high attention weights and remain in the output matrix, while small attention weights eliminate insignificant columns. The Attention mechanism is equivalent to column selection if we replace the $\mathrm{softmax}$ operator with the corresponding argmax operator. However, sampling only columns may not generate a good summary of the matrix, and could be subjected to the noises in individual columns. We address this problem by exploiting Skeleton Sketching technique [31–39] in the matrix approximation community, which is also known as CUR approximation.

Manuscript received November 30, 2023; revised March 31, 2024; accepted XXX XX, 2024. This work was partially supported by NSF DMS 2304489. (*Corresponding author: HanQin Cai.*)

X. Wang, T. Zhou, J. Liu, and W. Yin are with Alibaba Group, Bellevue, WA 98004, USA (e-mail: {xue.w, tian.zt, jialin.liu, wotao.yin}@alibaba-inc.com).

J. Zhu is with Computer, Electrical and Mathematical Science and Engineering Division, King Abdullah University of Science and Technology, Thuwal 23955, Saudi Arabia (e-mail: zjq941026@gmail.com).

K. Yuan is with Center for Machine Learning Research, Peking University, Beijing 100871, China (e-mail: kunyuan@pku.edu.cn).

T. Yao is with Antai College of Economics and Management, Shanghai Jiao Tong University, Shanghai 200030, China (e-mail: taoyao@sjtu.edu.cn).

R. Jin is with Meta, Menlo Park, CA 94025, USA (e-mail: rongjinemail@gmail.com).

H.Q. Cai is with Department of Statistics and Data Science and Department of Computer Science, University of Central Florida, Orlando, FL 32816, USA (e-mail: hqcai@ucf.edu).

Theoretically, for a rank-r matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, we can take $\mathcal{O}(r\log d)$ column samples and $\mathcal{O}(r\log n)$ row samples to construct a so-called Skeleton approximation $\boldsymbol{X} \approx \boldsymbol{CUR}$, where $\boldsymbol{C}$ and $\boldsymbol{R}$ are matrices consisting of sampled columns and rows of $\boldsymbol{X}$ respectively, and $\boldsymbol{U}$ is the pseudo-inverse of $\boldsymbol{C}$ and $\boldsymbol{R}$'s intersection. By combining these mechanisms, we found, both theoretically and empirically, that S^{3}Attention is able to preserve global information over long sequences and reduce the impact of noise simultaneously, thus leading to better performance than state-of-the-art variants of Attention based models for long sequences, without having to sacrifice the linear complexity w.r.t. sequence length.

In short, we summarize our main contributions as follows:

1. We propose the S^{3}Attention, an efficient model that integrates smoother column Attention and row Attention components to unfold a randomized linear matrix sketching algorithm.
2. By randomly selecting a fixed number of rows and columns, the proposed model achieves near-linear computational complexity and memory cost. The effectiveness of this selection method is verified both theoretically and empirically.
3. We conduct extensive experiments over Long-term sequences, long-term time series forecasting and GLUE tasks. In particular, the Long Range Arena benchmark [40], achieves an average accuracy of 64% and 66% with fixed parameters (suggested setting in [40, 41]) and fine-tuned parameters respectively. It improves from 62% of the best Attention-type model. Moreover, it also has a comparable performance with the recent state-of-the-art long-term time series forecasting models for long-term time series forecasting and GLUE tasks.

Organization. We structure the rest of this paper as follows: In Section II, we briefly review the relevant literature related to efficient Attention based models. Section III introduces the model structure and performs a theoretical analysis to justify the proposed model. We empirically verify the efficiency and accuracy of S^{3}Attention in Section IV we discuss limitations and future directions in Section V. The experimental details are provided in the supplementary material.

II. Related Work

This section provides an overview of the literature mainly focusing on efficient Attention based models. The techniques include sparse or local Attention, low rankness, and kernel approximation. We refer the reader interested in their details to the survey [42].

Sparse Attention. The general idea of these methods is restricting the query token to perform Attention only within a specific small region, such as its local region or some global tokens. In this setting, the Attention matrix becomes sparse compared to the original one. [43] proposes BlockBert, which introduces sparse block structures into the Attention matrix by multiplying a masking matrix. [44] applies Attention within blocks for the image generation task. [45] divides the whole sequences into blocks and uses a stride convolution to reduce the model complexity. However, these block-type Transformers ignore the connections among blocks. To address this issue, Transformer-XL [46] and Compressive Transformer [47] propose a recurrence mechanism to connect multiple blocks. Transformer-LS [23] combines local Attention with a dynamic projection to capture long-term dependence. [48] uses a meta-sorting network to permute over sequences and quasi-global Attention with local windows to improve memory efficiency.

Another approach in this category is based on stride Attention. Longformer [49] uses dilated sliding windows to obtain a sparse Attention matrix. Sparse Transformers [50] consider approximating a dense Attention matrix by several sparse factorization methods. In addition, some methods reduce the complexity by clustering tokens. For example, Reformer [51] uses a hash similarity measure to cluster tokens, and Routing Transformer [52] uses k-means to cluster tokens. BigBird [53] proposes a generalized Attention mechanism described by a directed graph to reduce Attention complexity. [54] considers using 2D Fourier Transformation to mix the token matrix directly. [55] uses max pooling scheme to reduce the computation costs.

Low-rank and Kernel Methods. Inducing low rankness into the Attention matrix can quickly reduce the complexity and the kernel approximation is widely applied in efficient low-rank approximation. Linformer [56] and Luna [30] approximate $\mathrm{softmax}$ with linear functions, which yield a linear time and space complexity. [24, 57] use random features tricks and reach promising numerical performance. [58] proposes Low-Rank Transformer based on matrix factorization. FMMformer [29] combines the fast multipole method with the kernel method. Synthesizer [59] uses a random low-rank matrix to replace the Attention matrix. Nyströmformer [60] adopts the Nyström method to approximate standard Attention. Linear Transformer [61] expresses Attention as a linear dot-product of kernel feature maps. [28] applies the Multigrid method to efficiently compute the Attention matrix recursively. Cosformer [27] develops a cosine-based re-weighting mechanism to linearize the $\mathrm{softmax}$ function. [25] proposes the Scatterbrain, which unifies locality-sensitive hashing and the kernel method into Attention for accurate and efficient approximation.

Sequence Length Reduction. Reducing sequence length is an efficient means to reduce computational costs. Perceiver IO [62] encodes inputs to a latent space whose size is typically smaller than the inputs and outputs, making the process computationally scalable to even very large inputs and outputs. Funnel-Transformer [63] uses pooling tricks to reduce sequence length, significantly saving both FLOPs and memory. Swin Transformer [7] proposes a shifted windows method in vision Transformer, and the resulting complexity is linear in the image size. XCiT [64] considers an Attention over the hidden dimension for vision tasks. Charformer [65] downsamples the sequences of words to construct latent subwords.

Improved Recurrent Neural Network. Another research track to solve the long sequence tasks is to improve the Recurrent Neural Network (RNN). [66] propose to view RNN as the state space model and use optimal polynomial projections to improve its memorization ability. The follow-up

works [67, 68, 68–72] consider more sophisticated designs and obtain better performance and higher efficiency. The concurrent work [73] proposes coupling the moving average type of RNN structure with single-headed Attention and reaches quite promising results. Different from those works, in this paper, we consider a different research angle and focus on making improvements inside the Attention structure.

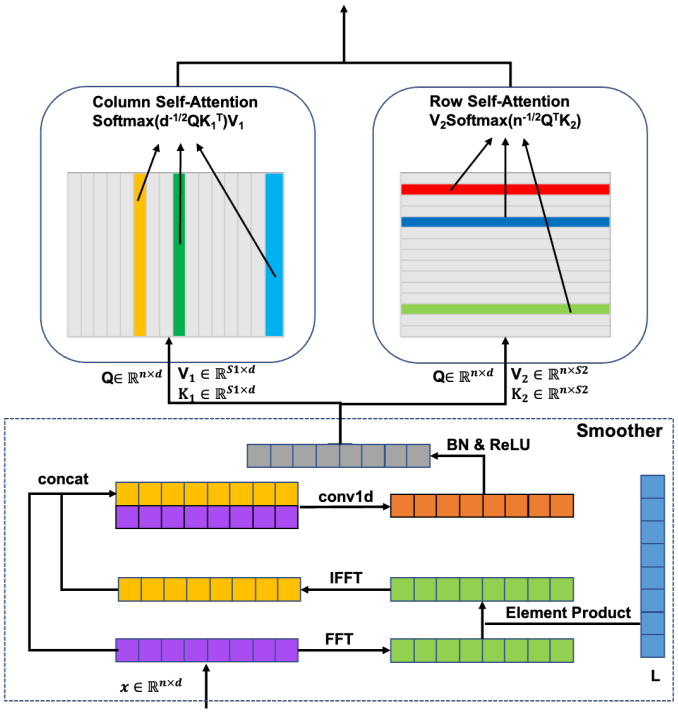


Figure 1: Illustration of the architecture of S^{3}Attention.

III. S^{3}Attention

We start by going over the vanilla Attention. For a sequence of length n, the vanilla Attention is dot-product type [1]. Following standard notation, the Attention matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is defined as:

$$\boldsymbol{A} = \mathrm{softmax}\left(\frac{1}{\sqrt{d}}\boldsymbol{Q}\boldsymbol{K}^\top\right),$$

where $\boldsymbol{Q} \in \mathbb{R}^{n\times d}$ denotes the queries while $\boldsymbol{K} \in \mathbb{R}^{n\times d}$ denotes the keys, and d represents the hidden dimension. By multiplying the attention weights $\boldsymbol{A}$ with the values $\boldsymbol{V} \in \mathbb{R}^{n\times d}$, we can calculate the new values as $\hat{\boldsymbol{V}} = \boldsymbol{A}\boldsymbol{V}$.

Intuitively, the Attention is the weighted average over the old ones, where the weights are defined by the Attention matrix $\boldsymbol{A}$. In this paper, we consider generating $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ via the linear projection of the input token matrix $\boldsymbol{X}$:

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q,\ \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K,\ \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V,$$

where $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ and $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{d\times d}$.

The vanilla procedure has two drawbacks in concentrating the information from $\boldsymbol{V}$. First, when computing the $\boldsymbol{Q}\boldsymbol{K}^\top$ part, full dense matrix multiplication is involved at a cost of $\mathcal{O}(n^2)$ vector multiplications. It can be prohibitive for long sequence problems. On the other hand, if we view the softmax operator as an approximation of the argmax counterpart, $\hat{\boldsymbol{V}}$ becomes a row selection from $\boldsymbol{V}$. From the view of matrix sketching, solely using either row or column selection could reduce the cost-effectiveness, compared to using both column and row selections.

A. Skeleton Sketching Based Attention

We propose a Skeleton Sketching induced Attention structure to address those issues. First, we modify the original Attention to build the column Attention as follows:

$$\hat{\boldsymbol{V}}_1 = \mathrm{softmax}\left(\frac{1}{\sqrt{d}}\boldsymbol{Q}\boldsymbol{K}^\top\boldsymbol{P}_1^\top\right)\boldsymbol{P}_1\boldsymbol{V},$$

where $\boldsymbol{P}_1 \in \mathbb{R}^{s_1\times n}$ denotes the sampling matrix and s_1 is the number of columns sampled. Let $i_1 < i_2 < \ldots < i_{s_1}$ be the indices of the randomly sampled columns. Let $\boldsymbol{P}_{1,ab}$ denote the element located at the a-th column and b-th row and we have $\boldsymbol{P}_{1,ab} = 1$ if $i_a = b$ and 0 otherwise. By these constructions, we can reduce the computational cost to $\mathcal{O}(ns_1d^2 + ns_1^2d)$.

Similarly, we build the row sampling matrix $\boldsymbol{P}_2 \in \mathbb{R}^{d\times s_2}$ indicating the locations of the s_2 sample rows. Compute the row Attention as:

$$\hat{\boldsymbol{V}}_2 = \boldsymbol{V}\boldsymbol{P}_2\ \mathrm{softmax}\left(\frac{1}{\sqrt{n}}\boldsymbol{P}_2^\top\boldsymbol{K}^\top\boldsymbol{Q}\right).$$

Finally, we apply the layer-norm on $\hat{\boldsymbol{V}}_1$ and $\hat{\boldsymbol{V}}_2$ and then add them together to generate the final output:

$$\hat{\boldsymbol{V}} = \mathrm{layernorm}_1(\hat{\boldsymbol{V}}_1) + \mathrm{layernorm}_2(\hat{\boldsymbol{V}}_2). \tag{1}$$

The usage of layer norm is to balance the output scales of column and row Attentions. A similar trick has been used in [23], where the layer norm is applied to resolve scale mismatches between the different Attention mechanisms.

Before going into the detailed analysis, we first introduce the incoherence parameter of a matrix, which is commonly used in many low-rank matrix applications.

Definition 1 (μ-incoherence)**.** *Given a rank-r matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$. Let $\boldsymbol{X} = \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ be its compact singular value decomposition. $\boldsymbol{X}$ is μ-incoherent if there exists a constant μ such that*

$$\max_i \|\boldsymbol{e}_i^\top\boldsymbol{W}\| \le \sqrt{\frac{\mu r}{n}} \quad \text{and} \quad \max_i \|\boldsymbol{e}_i^\top\boldsymbol{V}\| \le \sqrt{\frac{\mu r}{d}},$$

where $\boldsymbol{e}_i$ denotes the i-th canonical basis vector.

The μ-incoherence describes the correlation between the column/row spaces and the canonical basis vectors. The larger μ value implies a higher *overlapping*, which leads to a better chance of successful reconstruction from sparse row/column samples. We next use the following proposition to characterize the efficiency of sampling in both columns and rows.

Proposition 1. *Let $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ be a rank-r, μ-incoherent matrix. Without loss of generality, we assume $n \ge d$. Let $\boldsymbol{E} \in \mathbb{R}^{n\times d}$ be a noise matrix. By uniformly sampling $\mathcal{O}(\mu r \log n)$ columns and rows from the noisy $\boldsymbol{X}+\boldsymbol{E}$, Skeleton*

approximation can construct a matrix $\hat{\boldsymbol{X}}$ *such that, with probability at least* $1 - \mathcal{O}(n^{-2})$*,*

$$\|\boldsymbol{X} - \hat{\boldsymbol{X}}\| \leq \mathcal{O}\left(\frac{\|\boldsymbol{E}\|\sqrt{nd}}{\mu r \log n}\right).$$

A similar result, under a slightly different setting, can be found in [74]. For the completeness of the paper, we provide the proof here.

Proof. We resolve the sampling strategy. We consider a clear rank-r matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, i.e., no additive noise and the rank is exact. Without loss of generality, we assume $n \geq d$. Provided $\boldsymbol{X}$ is μ-incoherent, by [32, Theorem 1.1], Skeleton approximation recovers $\boldsymbol{X}$ exactly, i.e.,

$$\boldsymbol{X} = \boldsymbol{C}\boldsymbol{U}\boldsymbol{R},$$

with probability at least $1 - \mathcal{O}(n^{-2})$ if we sample $\mathcal{O}(\mu r \log n)$ rows and columns uniformly to form the submatrices $\boldsymbol{C}$ and $\boldsymbol{R}$.

Thirdly, we resolve the error bound estimation. For the noisy matrix $\boldsymbol{X} + \boldsymbol{E}$, we directly apply [75, Corollary 4.3]. Thus, we have

$$\|\boldsymbol{X} - \hat{\boldsymbol{C}}\hat{\boldsymbol{U}}\hat{\boldsymbol{R}}\| \leq \mathcal{O}\left(\sqrt{\frac{nd}{l_C l_R}}\right)\|\boldsymbol{E}\|,$$

where $\hat{\boldsymbol{C}}$ and $\hat{\boldsymbol{R}}$ are sampled from the noisy matrix, $\hat{\boldsymbol{U}}$ is the pseudo-inverse of $\hat{\boldsymbol{C}}$ and $\hat{\boldsymbol{R}}$'s intersection, and l_C (resp. l_R) is the number of columns (resp. rows) being sampled in $\hat{\boldsymbol{C}}$ (resp. $\hat{\boldsymbol{R}}$).

Note that this error bound assumes good column/row sampling, i.e., the clear submatrices corresponding to $\hat{\boldsymbol{C}}$ and $\hat{\boldsymbol{R}}$ can recover $\boldsymbol{X}$ exactly. Therefore, by combining the above two results, we show the claim in Proposition 1. □

Several works (e.g., [31, 32]) have proposed explicit methods to construct $\hat{\boldsymbol{X}}$. Those methods require computing the pseudo-inverse, generally inefficient in deep learning settings. [60] uses an approximation of the pseudo-inverse in the symmetric matrix setting. It is still an open question whether the approximated pseudo-inverse also works for the general matrix in deep learning settings. On the other hand, in the transformer model, a good matrix approximation is not our primary goal, and we thus pursue a different way that only maintains sufficient information to pass through the network via (1).

B. Smoother Component

Based on the analysis of Skeleton Sketching, the matrix incoherence parameter μ plays a crucial role in determining the number of rows and columns to sample. Decreasing in μ leads to a smaller sampling size. Furthermore, the μ-incoherence condition implies that the "energy" of the matrix is evenly distributed over its entries, i.e., the matrix is "smooth" [76]. An illustration of the Smoother in Skeleton Attention part is shown in Figure 2. We smooth the input token matrix to ensure the sampling in rows and columns containing more local and/or global information. Thus, sampling several rows and columns from the smoothed token matrix can be more effective than the samples from the original token matrix. In this subsection, we propose a novel smoother component to reduce the incoherence parameter without introducing excessive information loss.

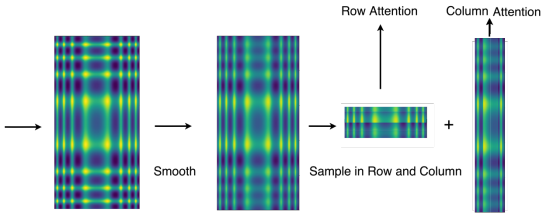


Figure 2: Illustration on the effect of the Smoother in Skeleton Attention on Token Matrix.

Fourier Convolution. The incoherence parameter μ can be viewed as a measure of the smoothness of a matrix. A "smoother" matrix tends to have a smaller incoherence parameter. Intuitively, the adjacent columns or rows have similar values for a smooth matrix. Thus a few landmark columns or rows can represent the matrix with little error. On the other hand, if the matrix is harsh (e.g., containing spiky columns or rows), more landmarks are required. A common way to smooth a matrix is to convolute it with a smoothing kernel, such as a Gaussian kernel. However, directly using a fixed smoothing kernel can potentially remove too many details and harm the final performance. In the recent literature (e.g., [77]), large convolution kernel-based Attentions show a supreme performance in vision Transformers. In this paper, we propose to use a data-driven convolution layer along the sequence dimension with a kernel size equal to the sequence length. In this setting, the information of a given row could be decentralized among the rows. As the input token matrix is computed through a FeedForward layer, the information among different rows could be already processed. Hence, we do not perform the convolution along the hidden dimension.

We use the Fast Fourier Transformation (FFT) to implement the convolution. Let $\boldsymbol{L}_0 \in \mathbb{R}^{n\times d}$ be the convolution kernel matrix. Via the convolution theorem, the circular convolutions in the spatial domain are equivalent to pointwise products in the Fourier domain, and we then have:

$$\boldsymbol{X}^{\text{smooth}} = \boldsymbol{X} * \boldsymbol{L}_0 = \mathcal{F}^{-1}\left[\mathcal{F}(\boldsymbol{X}) \cdot \mathcal{F}(\boldsymbol{L}_0)\right], \tag{2}$$

where $\mathcal{F}$, $*$, and $\cdot$ denote FFT operator, convolution operator, and point-wise product, respectively.

Equation (2) requires $3d$ times faster Fourier operations which could be prohibited when facing large d. In order to save the computational cost, we use the learnable matrix $\boldsymbol{L} \in \mathbb{C}^{n\times d}$ in the frequency domain instead and apply segment-average (averaging segments of hidden dimension) to $\boldsymbol{X}$. In practice, we use the rFFT/irFFT, the fast (inverse) Fourier Transformation of real input instead of the general FFT/IFFT, and the size of the matrix $\boldsymbol{L}$ is reduced to $\boldsymbol{L} \in \mathbb{C}^{(\lfloor n/2 \rfloor + 1)\times d}$. To simplify the notation, we assume there are integers s and r with $d = sr$. Instead of using (2), we apply the following (3) to smooth the token matrix.

$$\boldsymbol{X}^{\text{smooth}} = \mathcal{F}^{-1}\left[\mathcal{F}(\boldsymbol{X}\boldsymbol{S}) \cdot \boldsymbol{L}\right], \tag{3}$$

where

$$\boldsymbol{S} = \begin{bmatrix} \frac{1}{s}\mathbf{1} & \mathbf{0} & ... & \mathbf{0} \\ \mathbf{0} & \frac{1}{s}\mathbf{1} & ... & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \frac{1}{s}\mathbf{1} \end{bmatrix} \in \mathbb{R}^{d\times d} \tag{4}$$

and $\mathbf{1}$ denotes the $s \times s$ matrix with all elements equal 1. As $\boldsymbol{XS}$ contains repeated rows, in (3), we can reduce the usage of faster Fourier operations to $r+d$ times. Moreover, to further control the computation cost, we only smooth the token matrix $\boldsymbol{X}$ instead of the query, key and value matrices $\boldsymbol{Q}, \boldsymbol{K}$ and $\boldsymbol{V}$ separately.

In the following proposition, we show the smooth ability of the Fourier convolution.

Proposition 2. *Let $\{x_1, ..., x_n\}$ be sequences with $\max_t |x_t| \le a_{\max}$ and $\max_t |x_t - x_{t-1}| \le b_{\max}$. Let $\{l_1, ..., l_n\}$ be sequences of i.i.d. $\frac{1}{n^2}\sigma^2$-subgaussian variables. Let $f(t)$ be the convolution of $\{x_t\}$ and $\{l_t\}$, i.e., $f(t) = \sum_{i=1}^{t} l_{t+1-i}x_i$. With probability at least $1-\delta$, we have:*

$$\begin{aligned} &|f(t) - f(t-1)| \\ &\le b_{\max}\sigma\sqrt{\frac{1}{2n}\log\left(\frac{2n}{\delta}\right)} + a_{\max}\sigma\sqrt{\frac{1}{2n^2}\log\left(\frac{2}{\delta}\right)}. \end{aligned}$$

Proof. As $f(t)$ is the convolution function of $\{x_t\}$ and $\{l_t\}$, from the definition of convolution for $t = 1, 2, ..., n$ we have

$$f(t) = \sum_{i=1}^{t} l_{t+1-i}x_i$$

and

$$f(t) - f(t-1) = \underbrace{\sum_{i=1}^{t-1}(l_{i+1} - l_i)x_i}_{:=(a_t)} + l_1 x_t. \tag{5}$$

By Hoffelding inequality, term (a_t) satisfies the following inequality with $\varepsilon > 0$.

$$\begin{aligned} \mathbb{P}(|(a_t)| \ge \varepsilon) &= \mathbb{P}\left(\left|\sum_{i=1}^{t-1}(l_{i+1} - l_i)x_i\right| \ge \varepsilon\right) \\ &\le \exp\left(-\frac{2\varepsilon^2}{(t-1)b_{\max}^2 \cdot \frac{1}{n^2}\sigma^2}\right) \end{aligned} \tag{6}$$

Combine (6) with the union bound over $t = 1, 2, ..., n$ and the following (6) holds with probability at least $1 - \delta/2$:

$$\max_t |(a_t)| \le b_{\max}\sigma\sqrt{\frac{1}{2n}\log\left(\frac{2n}{\delta}\right)} \tag{7}$$

Similarly, with probability $1 - \delta/2$, we have

$$\max_t |l_1 x_t| \le a_{\max}\sigma\sqrt{\frac{1}{2n^2}\log\left(\frac{2}{\delta}\right)}. \tag{8}$$

Therefore, with probability at least $1-\delta$, (7) and (8) give

$$\begin{aligned} &\max_t |f(t) - f(t-1)| \\ &\quad\le b_{\max}\sigma\sqrt{\frac{1}{2n}\log\left(\frac{2n}{\delta}\right)} + a_{\max}\sigma\sqrt{\frac{1}{2n^2}\log\left(\frac{2}{\delta}\right)}. \end{aligned}$$

This finishes the proof. □

The Proposition 2 can be used to describe the Fourier convolution layer's behavior in the early training stage. Via some standard initialization methods (e.g., Kaiming initialization or Xavier initialization), the variance of elements in learnable matrix $\boldsymbol{L}$ is $\mathcal{O}(n^{-1})$ and the scale of elements is $\mathcal{O}(n^{-1/2})$.[1] To simplify our discussion, let us assume we use Kaiming normal initialization and $\boldsymbol{L}$ becomes a random complex Gaussian matrix with zero mean and variance $n^{-1}\sigma^2$. Using the fact that the FFT of Gaussian s remains Gaussian with $2n$ times larger variance, the $n^{-1}\sigma^2$ variance Gaussian sequences through inverse FFT (IFFT) would result in Gaussian sequences with $\frac{1}{2n^2}\sigma^2$ variance. By Proposition 2, the maximum difference between adjacent elements after the convolution is scaled on $b_{\max}\sigma n^{-1/2} + a_{\max}\sigma n^{-1} \approx b_{\max}\sigma n^{-1/2}$ when sequence length n is large enough. Thus as long as $\sigma < \mathcal{O}(\sqrt{n})$, the sequences are smoothed by the Fourier convolution. This smoothness ability eventually helps stabilize the early training stage. The related experiments are summarized in Section IV-G.

When the training procedure progresses, the learnable matrix $\boldsymbol{L}$ will be apart from the Sub-Gaussian initialization and converge to some data-driven matrix. In this regime, it is possible that the convoluted sequences may not necessarily preserve a very low variance as it is in the early stage. However, a good memorization ability can happen. In particular, in Proposition 3, we show it is possible to have a learned matrix $\boldsymbol{L}$ such that the historical information in $\boldsymbol{XS}$ up to time t can be compressed to some function parameterized with t-th token in the convoluted sequences.

Proposition 3. *Let $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ be a bounded matrix and $\boldsymbol{S} \in \mathbb{R}^{d\times d}$ constructed by* (4)*. There exist matrices $\boldsymbol{G}, \boldsymbol{L} \in \mathbb{R}^{n\times d}$ such that*

$$\left\|(\boldsymbol{XS})_{1:t} - \boldsymbol{X}_t^{\text{smooth}}\boldsymbol{G}_{1:t}\right\| \le \mathcal{O}\left(r^{3/2}t\log(n)d^{-1/2}\right),$$

where $(\cdot)_{1:t}$ is the submatrix of the first t rows of a given matrix, $\boldsymbol{X}_t^{\text{smooth}}$ is the t-th row of $\boldsymbol{X}^{\text{smooth}} = \mathcal{F}^{-1}\left[\mathcal{F}(\boldsymbol{XS}) \cdot \boldsymbol{L}\right]$, and $\boldsymbol{G}$ satisfies $\boldsymbol{G}_{i,j} = \boldsymbol{G}_{i+s,j} = ... = \boldsymbol{G}_{i+r(s-1),j} = g_i(j)$. Here $\{g_1(\cdot), ..., g_s(\cdot)\}$ is an orthogonal polynomial basis.

Proof. The proof contains two parts. In the first part, we view the data sequence as a function of index t and construct the coefficients and orthogonal polynomials for function approximation. In the second part, we show such coefficients can be computed with Fourier convolution, i.e., (3)).

Function Approximation. We reformulate the matrix $\boldsymbol{XS}$ as follow:

$$\boldsymbol{XS} = \begin{bmatrix} \bar{\boldsymbol{x}}_1\boldsymbol{e} & \bar{\boldsymbol{x}}_2\boldsymbol{e} & \cdots & \bar{\boldsymbol{x}}_r\boldsymbol{e} \end{bmatrix},$$

where $\boldsymbol{e} \in \mathbb{R}^{1\times s}$ is the one vector and $\bar{\boldsymbol{x}}_i \in \mathbb{R}^{n\times 1}$ is the average of $(s(i-1)+1)$-th to (si)-th columns of $\boldsymbol{X}$.

[1]Here we omit the dependence in d for brevity.

Next, we focus on vector $\bar{\boldsymbol{x}}_j$ and view its t-th element as the output of a function $h^j(t) = \bar{\boldsymbol{x}}_{jt}$. Via analysis in [66, Appendixes C and D], we can form an approximation on $h^j(t)$ as follow:

$$h^j_{[x\leq t]}(x) \approx \sum_{i=1} c^j_i(t) g_i(x), \tag{9}$$

where $\{g_i\}$ is a sequence of orthogonal polynomial and $[c^j_1(t), c^j_2(t), ..., c^j_s(t)] := \boldsymbol{c}^j_t \in \mathbb{R}^{1\times s}$ satisfy

$$\frac{d}{dt}\boldsymbol{c}(t)^j = \frac{1}{t}\boldsymbol{c}(t)^j\boldsymbol{A}_0 + \frac{1}{ts\log n}h(t)\boldsymbol{b}_0 \tag{10}$$

where $\boldsymbol{A}_0 \in \mathbb{R}^{s\times s}$ and $\boldsymbol{b}_0 \in \mathbb{R}^{1\times s}$ are predefined matrix and vector respectively. Equation (10) is corresponding to the case with $\lambda_n = s\log n$ in [66].

We then use Forward Euler approach to discretize it:

$$\hat{\boldsymbol{c}}(t)^j = \hat{\boldsymbol{c}}(t-1)^j(\frac{1}{t}\boldsymbol{I} + \frac{1}{t}\boldsymbol{A}_0) + \frac{1}{ts\log n}h(t)\boldsymbol{b}_0, . \tag{11}$$

The standard error analysis of Forward Euler approach gives

$$\begin{aligned}
&\boldsymbol{c}(t+1)^j \\
=&\boldsymbol{c}(t)^j + \frac{1}{t}\boldsymbol{c}(t)^j\boldsymbol{A}_0 + \frac{1}{ts\log n}h(t)\boldsymbol{b}_0 + \frac{d^2}{dt^2}\boldsymbol{c}(t)^j|_{t=\xi} \\
=&\boldsymbol{c}(t)^j + \frac{1}{t}\boldsymbol{c}(t)^j\boldsymbol{A}_0 + \frac{1}{ts\log n}h(t)\boldsymbol{b}_0 + \frac{1}{\xi s\log n}h(\xi)'\boldsymbol{b}_0 \\
=&\boldsymbol{c}(t)^j + \frac{1}{t}\boldsymbol{c}(t)^j\boldsymbol{A}_0 + \frac{1}{ts\log n}h(t)\boldsymbol{b}_0 + \mathcal{O}\left(\frac{1}{ts\log n}\right),
\end{aligned}$$

where $\xi \in [t, t+1]$. It implies that for $t = 1, 2, ..., n$,

$$\|\hat{\boldsymbol{c}}(t)^j - \boldsymbol{c}(t)^j\| \leq \mathcal{O}\left(\frac{\log t}{s\log n}\right). \tag{12}$$

Combine (12) with Proposition 6 in [66], if $h^j(x)$ is quadratic spline interpolation on $\{\bar{\boldsymbol{x}}_{jt}\}$, we obtain[2],

$$\|\bar{\boldsymbol{x}}_{jt} - \sum_{i=1}^{s}\hat{\boldsymbol{c}}_i(t)g_i(x)\| \leq \mathcal{O}\left(t\log n/\sqrt{s}\right) = \mathcal{O}\left(t\log n\sqrt{\frac{r}{d}}\right). \tag{13}$$

The desirable result in Proposition 3 is obtained by repeatedly using (13) with $j = 1, 2, ..., r$.

Coefficients via Fourier Convolution. The remaining task is to show that $\{\hat{\boldsymbol{c}}(t)^j\}$ can be generated via Fourier convolution. To simplify the notation, we denote $\boldsymbol{A} = \frac{1}{t}\boldsymbol{I} + \frac{1}{t}\boldsymbol{A}_0$ and $\boldsymbol{b} = \frac{1}{t\log n}\boldsymbol{b}_0$ and (11) becomes

$$\hat{\boldsymbol{c}}(t)^j = \hat{\boldsymbol{c}}(t-1)^j\boldsymbol{A} + h(t)\boldsymbol{b}. \tag{14}$$

We then repeatedly use (14) with $t = 1, 2, ...,$ and one can see

$$\begin{aligned}
&\hat{\boldsymbol{c}}^j_t = \sum_{i=1}^{t-1}\boldsymbol{b}\boldsymbol{A}^{t-i}h(i) = \sum_{i=1}^{t-1}\boldsymbol{b}\boldsymbol{A}^{t-i}\bar{\boldsymbol{x}}_{ji} \\
\implies\quad &\boldsymbol{C}^j = \bar{\boldsymbol{A}}_j * (\bar{\boldsymbol{x}}_j\boldsymbol{e}),
\end{aligned} \tag{15}$$

[2]If h^j is $(k+1)$-th order spline interpolation on $\{\bar{\boldsymbol{x}}_{jt}\}$, then$\|\bar{\boldsymbol{x}}_{jt} - \sum_{i=1}^{s}\hat{\boldsymbol{c}}_i(t)g_i(x)\| \leq \mathcal{O}\left(t^k\log^k n\left(\frac{r}{d}\right)^{-k+1/2}\right)$.

where

$$\boldsymbol{C}^j = \begin{bmatrix}\hat{\boldsymbol{c}}^j_1\\ \hat{\boldsymbol{c}}^j_2\\ \vdots\\ \hat{\boldsymbol{c}}^j_n\end{bmatrix} \in \mathbb{R}^{n\times s}, \quad \text{and} \quad \bar{\boldsymbol{A}}_j = \begin{bmatrix}\boldsymbol{b}\\ \boldsymbol{b}\boldsymbol{A}\\ \vdots\\ \boldsymbol{b}\boldsymbol{A}^{n-1}\end{bmatrix} \in \mathbb{R}^{n\times s}. \tag{16}$$

Next we repeatedly use (15) from $j = 1, 2, .., r$, and have

$$\begin{aligned}
&\underbrace{\begin{bmatrix}\boldsymbol{C}^1 & \boldsymbol{C}^2 & \cdots & \boldsymbol{C}^r\end{bmatrix}}_{:=\boldsymbol{X}^{\text{smooth}}} \\
=&\underbrace{\begin{bmatrix}\bar{\boldsymbol{A}}_1 & \bar{\boldsymbol{A}}_2 & \cdots & \bar{\boldsymbol{A}}_r\end{bmatrix}}_{:=\boldsymbol{L}_0} * \underbrace{\begin{bmatrix}\bar{\boldsymbol{x}}_1\boldsymbol{e} & \bar{\boldsymbol{x}}_2\boldsymbol{e} & \cdots & \bar{\boldsymbol{x}}_r\boldsymbol{e}\end{bmatrix}}_{=\boldsymbol{X}\boldsymbol{S}} \\
\implies\quad &\boldsymbol{X}^{\text{smooth}} = \boldsymbol{L}_0 * \boldsymbol{X}\boldsymbol{S} \\
\implies\quad &\boldsymbol{X}^{\text{smooth}} = \mathcal{F}^{-1}\left(\mathcal{F}(\boldsymbol{L}_0)\cdot\mathcal{F}(\boldsymbol{X}\boldsymbol{S})\right) \\
\implies\quad &\boldsymbol{X}^{\text{smooth}} = \mathcal{F}^{-1}\left(\boldsymbol{L}\cdot\mathcal{F}(\boldsymbol{X}\boldsymbol{S})\right),
\end{aligned}$$

where we use the fact that $\boldsymbol{L}$ is constructed in frequency domain in Fourier convolution in Eq. (3). □

The Proposition 3 states that if we properly train the matrix $\boldsymbol{L}$, the information in $\boldsymbol{X}\boldsymbol{S}$ up to row t can be compressed into t-th row of $\boldsymbol{X}^{\text{smooth}}$ with a moderate tolerance. Therefore, when we sample in rows $\boldsymbol{X}^{\text{smooth}}$, they will contain more information than the same number of rows in the original $\boldsymbol{X}\boldsymbol{S}$. Similar results are also discussed in FNet [54] and several RNN literature, such as [66] and [78]. In [66], several specific types of polynomials (e.g., Legendre or Chebyshev) are explored, and the corresponding matrix $\boldsymbol{L}$ is predefined instead of data-driven. Recently, [67] propose a sophisticated method that can be used to compute $\boldsymbol{X}^{\text{smooth}}$. We leave it for future work.

Convolution Stem. $\boldsymbol{X}^{\text{smooth}}$ may encounter an over-smoothing situation that local details can be wiped out. We use a convolution stem (CNNs + BN + ReLU) to tackle this problem. We first concatenate $\boldsymbol{X}^{\text{smooth}}$ with the original token matrix $\boldsymbol{X}$ into a $n\times 2d$ matrix and then apply a 1D convolution with kernel size 3 to transform it back to $n\times d$ dimensions. At last, the output is normalized with the Batchnorm layer and truncated by the ReLU activation function to stabilize the training procedure. [79] report the ReLU activation coupled with the normalization layer plays an important role in various vision transformers and analyzes this phenomenon theoretically.

IV. Experiments

In this section, we test our S^{3}Attention on Long Range Arena (LRA) datasets [40] and six real-world time series benchmark datasets for long-term forecasting. We also evaluate the transfer learning ability of S^{3}Attention on GLUE tasks. In recent literature (e.g., [66–68]), the RNN type model is also widely discussed for long sequence tasks. We don't include them as benchmark models since in this paper we focus on improving the Attention structure. The testing environment contains 12 Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz CPUs, 1 TESLA V100 SXM2 32G, and 90 GB memory. We implement the S^{3}Attention based on the official codes of [23] and [12] for LRA and time-series forecasting tasks respectively. The implementation detail (source code) for S^{3}Attention is provided in the supplementary material.

Table 1: Experimental results on Long-Range Arena benchmark. The best model is in boldface and the second best is underlined. The standard deviations of the S^{3}Attention are reported in parentheses.

Model	ListOps	Text	Retrieval	Image	Pathfinder	Average
Transformer	36.37	64.27	57.46	42.44	71.40	54.39
Local Attention	15.82	52.98	53.39	41.46	66.63	46.06
Sparse Transformer	17.07	63.58	59.59	44.24	71.71	51.24
Longformer	35.63	62.85	56.89	42.22	69.71	53.46
Linformer	35.70	53.94	52.27	38.56	76.34	51.36
Reformer	37.27	56.10	53.40	38.07	68.50	50.67
Sinkhorn Transformer	33.67	61.20	53.83	41.23	67.45	51.39
Synthesizer	36.99	61.68	54.67	41.61	69.45	52.88
BigBird	36.05	64.02	59.29	40.83	74.87	55.01
Linear Transformer	16.13	65.90	53.09	42.34	75.30	50.55
Performer	18.01	65.40	53.82	42.77	77.05	51.41
Nystromformer	37.34	65.75	81.29	41.58	70.94	59.38
H-Transformer-1D	**49.53**	**78.69**	63.99	46.05	68.78	61.41
Transformer-LS	38.36	68.40	81.85	45.05	76.48	62.03
FNet	35.33	65.11	59.61	38.67	77.08	54.42
Luna	38.01	65.78	79.56	47.86	**78.89**	62.02
FMMformer	36.74	67.84	81.88	45.10	72.12	60.74
PoNet	38.80	69.82	80.35	46.88	70.39	61.05
Cosformer	37.9	63.41	61.36	43.17	70.33	55.23
Scatterbrain	38.6	64.55	80.22	43.65	69.91	59.38
S^{3}Attention ($r, s_1, s_2 = 8$)	38.30(0.40)	69.27(0.83)	83.26(0.45)	53.90(1.54)	75.82(0.97)	64.11(2.07)
S^{3}Attention (best)	39.15(0.48)	71.58(0.95)	**83.73(0.61)**	**57.73(1.83)**	78.20(1.32)	**66.08(2.56)**

A. Long-Range Arena

The open-source Long-Range Arena (LRA) benchmark [40] is originally proposed as a standard way to test the capabilities of Attention variants architectures on long sequence tasks.

We benchmark our model with several recent state-of-art efficient Attention architectures, including Sparse Transformer [50], Longformer [49], Linformer [56], Reformer [51], Sinkhorn Transformer [48], Synthesizer [59], BigBird [53], Linear Transformers [61], Performer [24], H-Transformer-1D [28], Nyströmformer [60], Transformer-LS [23], FNet [54], Luna [30], FMMformer [29], Cosformer [27] and Scatterbrain [25]. S^{3}Attention achieves the highest 66.08% average accuracy with tuned parameters and the second best 64.11% result with fixed parameters as shown in Table 1.

In particular, S^{3}Attention significantly outperforms the benchmarks on Image tasks by relatively large margins (12.6% and 20.6%, respectively), which support S^{3}Attention's smoothness effect on the low-level features and will benefit the high-level image classification tasks.

Moreover, we highlight the sampling efficiency of S^{3}Attention. The sequence length of LRA tasks is over one thousand. The efficient Transformers in literature usually can not project the token matrix to a very small size while maintaining comparable numerical performance, by only sampling 8 rows and columns from the token matrix, S^{3}Attention has already obtained 64.11% average score improving the previous best 62.03% score of Transformer-LS.

B. Long-Term Forecasting Tasks for Time Series

To further evaluate the proposed S^{3}Attention, we also conduct extensive experiments on six popular real-world benchmark datasets for long-term time series forecasting, including traffic, energy, economics, weather, and disease as shown in Table 2

To highlight the relevant comparison, we include five state-of-the-art (SOTA) Attention-based models, i.e., FEDformer [12], Autoformer [80], Informer [81], LogTrans [82], and Reformer [51] for comparison. FEDformer is selected as the main baseline as it achieves SOTA results in most settings. More details about baseline models, datasets, and implementations are described in the supplementary material.

Compared with SOTA work (i.e., FEDformer), the proposed S^{3}Attention yields a comparable performance in those tasks, with 3/6 datasets having larger winning counts in MSE/MAE. It is worth noting that the improvement is even more significant on certain datasets, e.g., Exchange ($> 30\%$ reduction in MSE and MAE). Although Exchange does not exhibit an apparent periodicity pattern, S^{3}Attention still achieves superior performance.

C. Transfer Learning in GLUE Tasks

We evaluate the transfer learning ability of the proposed model in the pretraining-finetuning paradigm in NLP tasks. We pre-train vanilla BERT [2], FNet [54], PoNet [55] and our S^{3}Attention with the same MLM loss in [2] on English Wikitext-103 and BooksCorpus datasets. All models are uncased and pre-trained with the same configuration with 1 million steps at most. We report the best GLUE results for each model from multiple hyper-parameters configurations in Table 3, and the detailed training configurations in Table 15 in Table 3. Our S^{3}Attention reaches 77.01 average scores (**96.0%** of the accuracy of vanilla BERT), which also outperform FNet by **4.6%** and PoNet by 0.3% relatively.

D. Training Speed and Peak Memory Usage

We compared the training speed (in terms of steps per second) and peak memory usage with several baseline models in LRA text classification task with various input lengths. The results

Table 2: Multivariate long-term series forecasting results on six datasets with input length of 96 and prediction length $O \in \{96, 192, 336, 720\}$ (For ILI dataset, we set prediction length $O \in \{24, 36, 48, 60\}$) with input length 36. A lower MSE indicates better performance. All experiments are repeated 5 times.

Methods		S^{3}Attention		FEDformer		Autoformer		Informer		LogTrans		Reformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm2	96	**0.192**	**0.283**	0.203	0.287	0.255	0.339	0.705	0.690	0.768	0.642	0.658	0.619
	192	**0.255**	**0.324**	0.269	0.328	0.281	0.340	0.924	0.692	0.989	0.757	1.078	0.827
	336	**0.324**	**0.364**	0.325	0.366	0.339	0.372	1.364	0.877	1.334	0.872	1.549	0.972
	720	0.431	0.433	**0.421**	**0.415**	0.422	0.419	0.877	1.074	3.048	1.328	2.631	1.242
Electricity	96	0.218	0.332	**0.183**	**0.297**	0.201	0.317	0.304	0.405	0.258	0.357	0.312	0.402
	192	0.259	0.361	**0.195**	**0.308**	0.222	0.334	0.313	0.413	0.266	0.368	0.348	0.433
	336	0.267	0.367	**0.212**	**0.313**	0.231	0.338	0.290	0.381	0.280	0.380	0.350	0.433
	720	0.293	0.385	**0.231**	**0.343**	0.254	0.361	0.262	0.344	0.283	0.376	0.340	0.420
Exchange	96	**0.086**	**0.204**	0.139	0.276	0.197	0.323	1.292	0.849	0.968	0.812	1.065	0.829
	192	**0.188**	**0.292**	0.256	0.369	0.300	0.369	1.631	0.968	1.040	0.851	1.188	0.906
	336	**0.356**	**0.433**	0.426	0.464	0.509	0.524	2.225	1.145	1.659	1.081	1.357	0.976
	720	**0.727**	**0.669**	1.090	0.800	1.447	0.941	2.521	1.245	1.941	1.127	1.510	1.016
Traffic	96	0.592	0.352	**0.562**	**0.349**	0.613	0.388	0.824	0.514	0.684	0.384	0.732	0.423
	192	0.583	0.343	**0.562**	**0.346**	0.616	0.382	1.106	0.672	0.685	0.390	0.733	0.420
	336	0.598	0.346	**0.570**	**0.323**	0.622	0.337	1.084	0.627	0.733	0.408	0.742	0.420
	720	0.641	0.397	**0.596**	**0.368**	0.660	0.408	1.536	0.845	0.717	0.396	0.755	0.423
Weather	96	**0.182**	**0.262**	0.217	0.296	0.266	0.336	0.406	0.444	0.458	0.490	0.689	0.596
	192	**0.228**	**0.306**	0.276	0.336	0.307	0.367	0.525	0.527	0.658	0.589	0.752	0.638
	336	**0.295**	**0.355**	0.339	0.380	0.359	0.395	0.531	0.539	0.797	0.652	0.639	0.596
	720	**0.383**	**0.418**	0.403	0.428	0.578	0.578	0.419	0.428	0.869	0.675	1.130	0.792
ILI	24	2.431	0.997	**2.203**	**0.963**	3.483	1.287	4.631	1.484	4.480	1.444	4.400	1.382
	36	2.287	**0.972**	**2.272**	0.976	3.103	1.148	4.123	1.348	4.799	1.467	4.783	1.448
	48	2.418	1.002	**2.209**	**0.981**	2.669	1.085	4.066	1.36	4.800	1.468	4.832	1.465
	60	**2.425**	**1.043**	2.545	1.061	2.770	1.125	4.278	1.41	5.278	1.560	4.882	1.483
	1st Count	12	13	12	11	0	0	0	0	0	0	0	0

Table 3: GLUE validation results. We report the mean of accuracy and F1 for QQP and MRPC, matthew correlations for CoLA, spearman correlations for STS-B, and accuracy for other tasks. For MNLI task, we consider the matched test set.

Model	MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
BERT-Base	81.98	89.25	88.22	91.07	48.08	87.98	86.43	69.98	80.37
FNet-Base	73.20	85.83	80.57	88.66	40.67	80.64	80.88	57.41	73.48
PoNet-Base	77.02	87.59	84.37	89.29	45.38	84.66	81.82	64.27	76.80
S^{3}Attention (Ours)	76.86	87.67	84.12	90.14	46.72	84.87	81.84	63.87	77.01

are reported in Table 4. S^{3}Attention achieves a 4x time speed advantage and 87% memory reduction compared to vanilla transformer models with 3k input setting and has a neck-to-neck performance compared to the most efficient baseline models.

E. Robustness Analysis

We conduct a noise-resistant experiment for S^{3}Attention and 5 other Attention based models as shown in Table 5. We use the Image experiment setting in LRA datasets. During generating sample sequences, we randomly add noise with uniform distribution $\mathcal{U}(-a, a)$ to each position in the sequences. We consider $a \in [0, 2, 4, 8]$ and train every model with 5k steps and 5 replicates. S^{3}Attention remains robust with a high level of noise injection. This supports our theoretical robustness analysis and shows S^{3}Attention indeed makes an appropriate tradeoff between information preservation and noise reduction.

F. Model Parameters Impact

S^{3}Attention introduces three extra hyperparameters, r, s_1 and s_2. We test the influence when varying them and report results in Table 6. We use S^{3}Attention $(r, s_1, s_2 = 8)$ as the baseline model and other parameters are reported in Table 10 in the supplementary material.

Influence of r in Fourier Convolution. The r parameter is used to determine the number of segment-averages to compute in (3). The smaller r leads the matrix with more duplicate columns, and more details information is lost. On the other hand, according to Proposition 3, the larger r would potentially decrease the memorization ability and yield a high approximation error. In Table 6a, the best performance is observed when $r = 8$ or $r = 16$. For the case with $r = 1$, the token matrix is smoothed to rank one matrix, and the average accuracy drops 3.55 from the best setting. When the r value goes larger than 16, the accuracy in all experiments slightly decreases. We believe it is due to the over-fitting since the smoothed token matrix contains more flexibility and more irrelevant information training dataset is learned.

Influence of Sample Number s_1 in Row Attention. In Row Attention part, we randomly sample s_1 from key and value tokens. Table 6b reports that the optimal sampling amounts are

Table 4: Benchmark results of all Xformer models with a consistent batch size of 32 across all models with various input lengths on the LRA text classification task The speed-up and memory-saving multipliers relative to Transformer shown in parentheses.

Model	Training Speed (Steps per second) 1 K	2 K	3 K	4 K	Peak Memory Usage (GB) 1 K	2 K	3 K	4 K
Transformer	23.8	7.8	3.9	OOM	3.7	11.1	22.1	OOM
Linformer	37.0(1.5x)	20.8(2.6x)	14.9(3.7x)	11.9	2.3	3.3	4.3	5.2
Reformer	28.5(1.2x)	15.1(1.9x)	11.9(3.0x)	9.1	2.2	3.2	4.2	4.9
Nystroformer	33.3(1.4x)	22.7(2.9x)	17.2(4.3x)	14.7	1.6	2.2	2.4	2.9
Performer	29.4(1.2x)	16.9(1.9x)	8.7(11.7x)	9.3	2.3	3.1	4.0	4.8
S^3Attention	32.2(1.4x)	20.4(2.6x)	15.9(4.0x)	12.3	1.8	2.3	2.8	3.2

Table 5: Average Accuracy on Image task (CIFAR-10 dataset) in Long Range Arena with noise injections. The relative performance changes are reported in parentheses.

Noise level	0	2	4	8
Transformer	41.39	40.29 (-2.82%)	28.56 (-31.12%)	28.12 (-32.18%)
Linformer	38.43	37.99 (-1.49%)	37.04 (-3.95%)	36.65 (-4.97%)
Reformer	38.04	37.64 (-1.12%)	35.26 (-7.37%)	34.88 (-8.37%)
Nystroformer	41.52	40.89 (-1.66%)	38.39 (-7.67%)	37.84 (-8.99%)
Performer	42.66	41.95 (-1.93%)	39.61 (-7.40%)	38.86 (-9.15%)
S^3Attention	57.47	57.06 (-0.82%)	55.32 (-3.84%)	54.70 (-4.92%)

different among tasks. In Pathfinder task, the optimal result is associated with $s_1 = 256$, while the best performance of other tasks the reached with $s_1 = 32$. Pathfinder task requires learning extreme long-range dependence (the connectivity between two circles far away from each other). The lack of enough tokens leads to inaccurate long-range dependence estimation and damages the final results. For tasks like Image or Retrieval, the modest range dependence may already be enough to get promising performance, and we thus could use fewer token samples.

Influence of Sample Number s_2 in Column Attention. In Column Attention, s_2 columns are selected. The experiment results are shown in Table 6c. When setting $s_2 = 1$, average performance decreases by 13.24%. Similar behavior is also observed in the first row of Table 6a with $r = 1$. The information loss due to lack of rank limits the final performance. In an average sense, $s_2 = 16$ gives the best result, and further increasing in s_2 slightly harms the accuracy in all tasks except Pathfinder.

G. Learning Curve for LRA Experiments

In this section, we present the training and testing performance for the first 5000 training steps on five LRA datasets. All hyperparameters are kept the same as the baseline model in the ablation study. The average training accuracy/loss and test accuracy/loss are reported in Figure 3. The results indicate the smoothed tokens lead to better learning behavior and faster convergence, which supports its stabilization ability.

H. Ablation Study

This subsection provides an ablation test on four components: Fourier Convolution, Convolution Stem, Column Attention, and Row Attention. We use S^3Attention with $(r, s_1, s_2 = 8)$ as the baseline, and the settings are detailed in Table 11 in the supplementary material. Table 7 presents the accuracy changes when removing each component. The performance-decreasing results indicate all four components of S^3Attention are necessary to reach promising results. The most significant component is Column Attention which leads 8.28 average accuracy difference. It states that a good summary of the whole sequence is important. Similar observations are also reported in Transformer-LS [23] and XCiT [64], where the spirit of Attention over columns is used in the dynamic project

Table 6: Experimental results on varying r, s_1 and s_2. The best result is in boldface and the second best is underlined. And Ablation experiments for each component.

(a) Experimental results on varying r parameter in smoothing component.

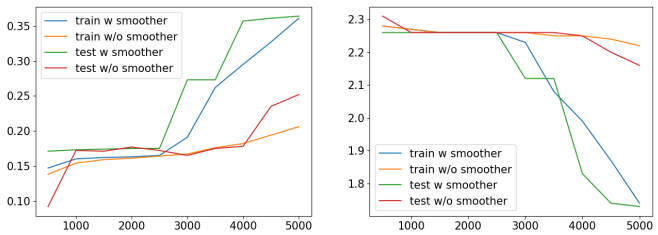
r	LisOps	Text	Retrieval	Image	Pathfinder	Average
1	37.30	65.25	78.65	51.36	71.23	60.76
8	<u>38.30</u>	<u>69.27</u>	**83.26**	<u>53.90</u>	<u>75.82</u>	<u>64.11</u>
16	**38.62**	**70.02**	<u>83.21</u>	**54.20**	**76.15**	**64.44**
32	38.19	69.27	82.05	53.73	75.58	63.76
64	37.89	69.73	81.79	51.28	75.52	63.24

(b) Experimental results on varying s_1 parameter in Row Attention.

s_1	LisOps	Text	Retrieval	Image	Pathfinder	Average
8	<u>38.30</u>	69.27	<u>83.26</u>	<u>53.90</u>	75.82	64.11
32	**38.44**	**70.85**	**83.41**	**54.92**	77.97	**65.12**
64	37.88	<u>70.53</u>	83.02	51.22	78.02	<u>64.33</u>
128	37.33	69.24	81.58	49.08	<u>78.12</u>	63.07
256	37.02	65.72	79.30	46.24	**78.14**	61.29

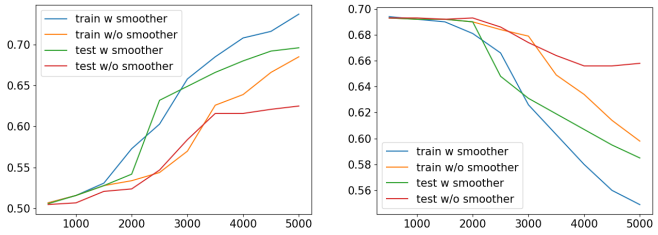
(c) Experimental results on varying s_2 parameter in Column Attention.

s_2	LisOps	Text	Retrieval	Image	Pathfinder	Average
1	37.32	55.28	57.37	40.97	66.25	51.44
4	<u>37.82</u>	52.05	72.58	46.74	73.17	57.47
8	**38.30**	<u>69.27</u>	<u>83.26</u>	<u>53.90</u>	75.82	<u>64.11</u>
16	37.77	**70.24**	**83.42**	**54.11**	<u>77.92</u>	**64.73**
32	37.62	68.32	80.11	51.66	**78.18**	62.98



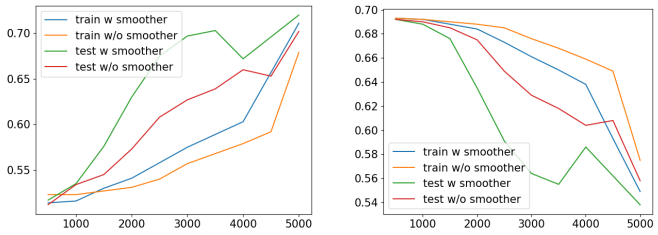
(a) Listops Accuracy

(b) Listops Loss



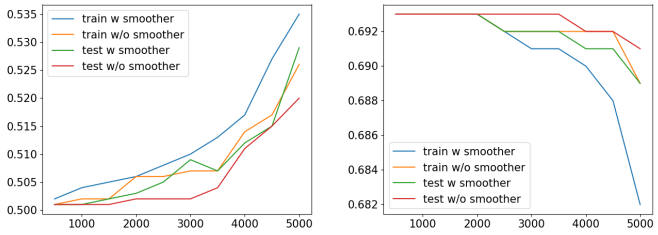
(c) Text Accuracy

(d) Text Loss



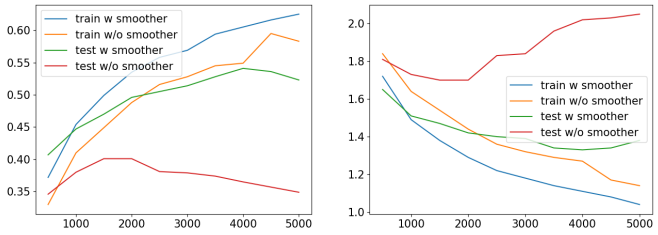
(e) Retrieval Accuracy

(f) Retrieval Loss



(g) Pathfinder Accuracy

(h) Pathfinder Loss



(i) Image Accuracy

(j) Image Loss

Figure 3: Learning Curve for LRA experiments.

and Cross-Covariance Attention respectively. The second most effective part is Fourier Convolution. It reaches a 13.89% accuracy difference in the Retrieval task involving two 4k sequences. Fourier Convolution also works well on shorter sequence tasks (e.g., Image and Pathfinder) and brings a 6.12% accuracy difference.

Table 7: Ablation experiments. The S^{3}Attention $(r, s_1, s_2 = 8)$ is used as baseline. The differences by removing each component from the baseline model are reported.

Model	LisOps	Text	Retrieval	Image	Pathfinder	Average
Baseline	38.30	69.27	83.26	53.90	75.82	64.11
Four. Conv.	-0.47	-4.04	-13.98	-5.64	-6.59	-6.14
Conv. Stem	-0.13	-0.55	-1.51	-1.76	-0.47	-0.88
Col. Attn.	-1.16	-8.00	-9.16	-10.63	-12.45	-8.28
Row Attn.	-0.38	-1.92	-1.97	-2.64	-2.56	-1.89

V. Concluding Remarks

We propose S^{3}Attention, a robust and efficient Attention architecture for modeling long sequences with a good balance between feature preserving and noise resistance. It aggregates a Fourier convolutional stem smoothing information among tokens and a Skeleton-Sketching-inspired efficient Attention. In particular, our proposed Skeleton Attention directly samples the columns and rows of the token matrix. Such a design increases the model's robustness and gives us a positive near-linear complexity side effect. We conduct a thorough theoretical and experimental analysis of the proposed model and show its effectiveness. Lastly, extensive experiments show that the proposed model achieves the best performance on Long Range Arena and a state-of-art performance in long-term time series forecasting tasks compared to various Attention-based baselines.

One limitation of the current S^{3}Attention is that we need to use both FFT and IFFT in a sequential manner, which is potentially slower than the existing Fourier-based Attention architectures (e.g., [54]) that only involve the FFT. As our primary goal using Fourier convolution is to smooth the token matrix and reduce the incoherent parameter, we can use the Random Fourier Transformation [83] to modify S^{3}Attention with only FFT. Another limitation is that the size of $\boldsymbol{L}$ matrix in the Fourier Convolution part is the same as the input sequences. On longer sequences, $\boldsymbol{L}$ will contain more learnable parameters that make the model easier to overfit. We may introduce low rankness or use a more sophisticated design, such as [67], to tackle this issue in the future.

Another future research direction is to incorporate the breakthroughs in state-space model fields (e.g., [66–68, 73]). Those works obtain very competitive performance and even beat various Attention variant architectures by a large margin in several long sequence tasks. In recent work [84], the state space model even reaches comparable performance in large-scale NLP pretraining tasks to the BERT-type models. It is possible to further boost S^{3}Attention's performance by combining with those breakthroughs.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for lan-

guage understanding," *arXiv preprint arXiv:1810.04805*, 2018.

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

[4] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[5] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[7] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.

[8] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 347–10 357.

[9] L. Yuan, Q. Hou, Z. Jiang, J. Feng, and S. Yan, "Volo: Vision outlooker for visual recognition," *arXiv preprint arXiv:2106.13112*, 2021.

[10] J. Zhou, P. Wang, F. Wang, Q. Liu, H. Li, and R. Jin, "Elsa: Enhanced local self-attention for vision transformer," *arXiv preprint arXiv:2112.12786*, 2021.

[11] J. Xu, J. Wang, M. Long *et al.*, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[12] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *39th International Conference on Machine Learning (ICML)*, 2022.

[13] P. Deshpande, K. Marathe, A. De, and S. Sarawagi, "Long horizon forecasting with temporal point processes," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM'21)*, 2021, pp. 571–579.

[14] S. Elmi and K.-L. Tan, "Deepfec: energy consumption prediction under real-world driving conditions for smart cities," in *Proceedings of the Web Conference*, 2021, pp. 1880–1890.

[15] B. Lim, A. Alaa, and M. van der Schaar, "Forecasting treatment responses over time using recurrent marginal structural networks." *NeurIPS*, vol. 18, pp. 7483–7493, 2018.

[16] J. Zhang and K. Nawata, "Multi-step prediction for influenza outbreak by an adjusted long short-term memory," *Epidemiology & Infection*, vol. 146, no. 7, pp. 809–816, 2018.

[17] J.-H. Böse, V. Flunkert, J. Gasthaus, T. Januschowski, D. Lange, D. Salinas, S. Schelter, M. Seeger, and Y. Wang, "Probabilistic demand forecasting at scale," *Proceedings of the VLDB Endowment*, vol. 10, no. 12, pp. 1694–1705, 2017.

[18] A. Chauhan, A. Prasad, P. Gupta, A. Prashanth Reddy, and S. Kumar Saini, "Time series forecasting for cold-start items by learning from related items using memory networks," in *Companion Proceedings of the Web Conference*, 2020, pp. 120–121.

[19] G. Sreenu and M. A. Durai, "Intelligent video surveillance: a review through deep learning techniques for crowd analysis," *Journal of Big Data*, vol. 6, p. 48, 06 2019.

[20] C. Faloutsos, V. Flunkert, J. Gasthaus, T. Januschowski, and Y. Wang, "Forecasting big time series: Theory and practice," in *Proceedings of the ACM Web Conference 2020*, 2020, p. 320–321.

[21] M. Hou, C. Xu, Z. Li, Y. Liu, W. Liu, E. Chen, and J. Bian, "Multi-granularity residual learning with confidence estimation for time series prediction," in *Proceedings of the ACM Web Conference*, 2022, pp. 112–121.

[22] D. Zhou, L. Zheng, Y. Zhu, J. Li, and J. He, "Domain adaptive multi-modality neural attention network for financial forecasting," in *Proceedings of The Web Conference 2020*, 2020, pp. 2230–2240.

[23] C. Zhu, W. Ping, C. Xiao, M. Shoeybi, T. Goldstein, A. Anandkumar, and B. Catanzaro, "Long-short transformer: Efficient transformers for language and vision," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[24] 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*, 2020.

[25] B. Chen, T. Dao, E. Winsor, Z. Song, A. Rudra, and C. Ré, "Scatterbrain: Unifying sparse and low-rank attention," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[26] S. P. Chowdhury, A. Solomou, A. Dubey, and M. Sachan, "On learning the transformer kernel," *arXiv preprint arXiv:2110.08323*, 2021.

[27] 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*, 2021.

[28] Z. Zhu and R. Soricut, "H-transformer-1d: Fast one-dimensional hierarchical attention for sequences," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021, pp. 3801–3815.

[29] T. Nguyen, V. Suliafu, S. Osher, L. Chen, and B. Wang, "Fmmformer: Efficient and flexible transformer via decomposed near-field and far-field attention," *Advances in*

Neural Information Processing Systems, vol. 34, 2021.

[30] X. Ma, X. Kong, S. Wang, C. Zhou, J. May, H. Ma, and L. Zettlemoyer, "Luna: Linear unified nested attention," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[31] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, "Relative-error CUR matrix decompositions," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 2, pp. 844–881, 2008.

[32] J. Chiu and L. Demanet, "Sublinear randomized algorithms for skeleton decompositions," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 3, pp. 1361–1383, 2013.

[33] H. Cai, K. Hamm, L. Huang, J. Li, and T. Wang, "Rapid robust principal component analysis: CUR accelerated inexact low rank estimation," *IEEE Signal Processing Letters*, vol. 28, pp. 116–120, 2021.

[34] H. Cai, K. Hamm, L. Huang, and D. Needell, "Robust CUR decomposition: Theory and imaging applications," *SIAM Journal on Imaging Sciences*, vol. 14, no. 4, pp. 1472–1503, 2021.

[35] ——, "Mode-wise tensor decompositions: Multi-dimensional generalizations of CUR decompositions," *Journal of Machine Learning Research*, vol. 22, no. 185, pp. 1–36, 2021.

[36] K. Hamm, M. Meskini, and H. Cai, "Riemannian CUR decompositions for robust principal component analysis," in *Topological, Algebraic and Geometric Learning Workshops 2022*. PMLR, 2022, pp. 152–160.

[37] H. Cai, L. Huang, P. Li, and D. Needell, "Matrix completion with cross-concentrated sampling: Bridging uniform sampling and CUR sampling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 8, pp. 10 100–10 113, 2023.

[38] H. Cai, Z. Chao, L. Huang, and D. Needell, "Robust tensor CUR decompositions: Rapid low-tucker-rank tensor recovery with sparse corruption," *SIAM Journal on Imaging Sciences*, vol. 17, no. 1, pp. 225–247, 2024.

[39] H. Cai, L. Huang, C. Kundu, and B. Su, "On the robustness of cross-concentrated sampling for matrix completion," in *58th Annual Conference on Information Sciences and Systems*, 2024, pp. 1–5.

[40] Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler, "Long range arena : A benchmark for efficient transformers," in *International Conference on Learning Representations*, 2021.

[41] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through ffts: International conference on learning representations (iclr2014), cbls, april 2014," in *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

[42] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *arXiv preprint arXiv:2009.06732*, 2020.

[43] J. Qiu, H. Ma, O. Levy, S. W.-t. Yih, S. Wang, and J. Tang, "Blockwise self-attention for long document understanding," *arXiv preprint arXiv:1911.02972*, 2019.

[44] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, "Image transformer," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4055–4064.

[45] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer, "Generating wikipedia by summarizing long sequences," *arXiv preprint arXiv:1801.10198*, 2018.

[46] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.

[47] J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap, "Compressive transformers for long-range sequence modelling," *arXiv preprint arXiv:1911.05507*, 2019.

[48] Y. Tay, D. Bahri, L. Yang, D. Metzler, and D.-C. Juan, "Sparse sinkhorn attention," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9438–9447.

[49] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[50] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[51] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[52] A. Roy, M. Saffar, A. Vaswani, and D. Grangier, "Efficient content-based sparse attention with routing transformers," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 53–68, 2021.

[53] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang *et al.*, "Big bird: Transformers for longer sequences," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 283–17 297, 2020.

[54] J. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontanon, "Fnet: Mixing tokens with fourier transforms," *arXiv preprint arXiv:2105.03824*, 2021.

[55] C.-H. Tan, Q. Chen, W. Wang, Q. Zhang, S. Zheng, and Z.-H. Ling, "Ponet: Pooling network for efficient token mixing in long sequences," *arXiv preprint arXiv:2110.02442*, 2021.

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

[57] H. Peng, N. Pappas, D. Yogatama, R. Schwartz, N. A. Smith, and L. Kong, "Random feature attention," *arXiv preprint arXiv:2103.02143*, 2021.

[58] G. I. Winata, S. Cahyawijaya, Z. Lin, Z. Liu, and P. Fung, "Lightweight and efficient end-to-end speech recognition using low-rank transformer," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6144–6148.

[59] Y. Tay, D. Bahri, D. Metzler, D.-C. Juan, Z. Zhao, and C. Zheng, "Synthesizer: Rethinking self-attention for transformer models," in *International conference on*

machine learning. PMLR, 2021, pp. 10 183–10 192.

[60] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh, "Nyströmformer: A nyström-based algorithm for approximating self-attention," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 16, 2021, pp. 14 138–14 148.

[61] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5156–5165.

[62] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer *et al.*, "Perceiver io: A general architecture for structured inputs & outputs," *arXiv preprint arXiv:2107.14795*, 2021.

[63] Z. Dai, G. Lai, Y. Yang, and Q. Le, "Funnel-transformer: Filtering out sequential redundancy for efficient language processing," *Advances in neural information processing systems*, vol. 33, pp. 4271–4282, 2020.

[64] A. Ali, H. Touvron, M. Caron, P. Bojanowski, M. Douze, A. Joulin, I. Laptev, N. Neverova, G. Synnaeve, J. Verbeek *et al.*, "Xcit: Cross-covariance image transformers," *Advances in neural information processing systems*, vol. 34, 2021.

[65] Y. Tay, V. Q. Tran, S. Ruder, J. Gupta, H. W. Chung, D. Bahri, Z. Qin, S. Baumgartner, C. Yu, and D. Metzler, "Charformer: Fast character transformers via gradient-based subword tokenization," in *International Conference on Learning Representations*, 2021.

[66] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, "Hippo: Recurrent memory with optimal polynomial projections," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1474–1487, 2020.

[67] A. Gu, K. Goel, and C. Re, "Efficiently modeling long sequences with structured state spaces," in *International Conference on Learning Representations*, 2022.

[68] A. Gupta, A. Gu, and J. Berant, "Diagonal state spaces are as effective as structured state spaces," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022.

[69] A. Gu, I. Johnson, K. Goel, K. Saab, T. Dao, A. Rudra, and C. Ré, "Combining recurrent, convolutional, and continuous-time models with linear state space layers," *Advances in neural information processing systems*, vol. 34, pp. 572–585, 2021.

[70] J. T. Smith, A. Warrington, and S. Linderman, "Simplified state space layers for sequence modeling," in *International Conference on Learning Representations*, 2023.

[71] R. Hasani, M. Lechner, T.-H. Wang, M. Chahine, A. Amini, and D. Rus, "Liquid structural state-space models," in *International Conference on Learning Representations*, 2023.

[72] A. Gu, I. Johnson, A. Timalsina, A. Rudra, and C. Re, "How to train your HIPPO: State space models with generalized orthogonal basis projections," in *International Conference on Learning Representations*, 2023.

[73] X. Ma, C. Zhou, X. Kong, J. He, L. Gui, G. Neubig, J. May, and L. Zettlemoyer, "Mega: Moving average equipped gated attention," in *International Conference on Learning Representations*, 2023.

[74] H. Cai, K. Hamm, L. Huang, and D. Needell, "Robust CUR decomposition: Theory and imaging applications," *SIAM Journal on Imaging Sciences*, vol. 14, no. 4, pp. 1472–1503, 2021.

[75] K. Hamm and L. Huang, "Perturbations of CUR decompositions," *SIAM Journal on Matrix Analysis and Applications*, vol. 42, no. 1, pp. 351–375, 2021.

[76] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, pp. 717–772, 2009.

[77] M.-H. Guo, C.-Z. Lu, Z.-N. Liu, M.-M. Cheng, and S.-M. Hu, "Visual attention network," *arXiv preprint arXiv:2202.09741*, 2022.

[78] A. Voelker, I. Kajić, and C. Eliasmith, "Legendre memory units: Continuous-time representation in recurrent neural networks," *Advances in neural information processing systems*, vol. 32, 2019.

[79] P. Wang, X. Wang, H. Luo, J. Zhou, Z. Zhou, F. Wang, H. Li, and R. Jin, "Scaled relu matters for training vision transformers," *arXiv preprint arXiv:2109.03810*, 2021.

[80] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 101–112.

[81] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[82] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[83] N. Ailon and B. Chazelle, "Approximate nearest neighbors and the fast johnson-lindenstrauss transform," in *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, 2006, pp. 557–563.

[84] J. Wang, J. N. Yan, A. Gu, and A. M. Rush, "Pretraining without attention," *arXiv preprint arXiv:2212.10544*, 2022.

[85] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 95–104.

Supplementary Material for S^{3}Attention: Improving Long Sequence Attention with Smoothed Skeleton Sketching

A. Algorithms

In this section, the pseudo-codes of the proposed methods are presented in Algorithms 1 and 2. They give the model architectures of the Skeleton Attention and Smoother component. Algorithms 3 and 4 give the details for forecasting experiments in Section IV-B. The implemented code is available online at https://github.com/wxie9/S3Attention.

Algorithm 1 Skeleton Attention

```
class Skeleton_Attention(nn.Module):
 def __init__(self, num_head = 2, head_dim = 32,seq_len, left_rank = 8,right_rank = 8, dropout = 0.1):
    super(Skeleton_Attention, self).__init__()
    self.num_head = num_head
    self.head_dim = head_dim
    self.seq_len = seq_len
    self.left_rank = left_rank
    self.right_rank = right_rank

    self.ln_1 = nn.LayerNorm(self.num_head * self.head_dim)
    self.ln_2 = nn.LayerNorm(self.num_head * self.head_dim)

    self.drop_attn = torch.nn.Dropout(p=dropout)

    self.index_set_right = torch.randperm(self.head_dim)
    self.index_set_right = self.index_set_right[:self.right_rank]

    self.index_set_left = torch.randperm(self.seq_len)
    self.index_set_left = self.index_set_left[:self.left_rank]

 def combine_heads(self, X):
    X = X.transpose(1, 2)
    X = X.reshape(X.size(0), X.size(1), self.num_head * self.head_dim)
    return X

 def split_heads(self, X):
    X = X.reshape(X.size(0), X.size(1), self.num_head, self.head_dim)
    X = X.transpose(1, 2)
    return X

 def forward(self,Q, K, V):
    #### Row Attention ####
    if self.left_rank <= self.seq_len:
       K1 = K[:,:,self.index_set_left,:]
       V1 = V[:,:,self.index_set_left,:]
    else:
       K1 = K
       V1 = V

    dots = Q @ K1.transpose(-1,-2)
    dots = dots / math.sqrt(self.head_dim)
    attn = nn.functional.softmax(dots,dim=-1)
    attn = self.drop_attn(attn)

    #### Column Attention ####
    Q2 = Q.transpose(-1,-2)
    if self.right_rank <= self.head_dim:

       K2 = K[:,:,:,self.index_set_right]
       V2 = V[:,:,:,self.index_set_right]
    else:
       K2 = K
       V2 = V

    dots_r = Q2 @ K2
    dots_r = dots_r / math.sqrt(self.seq_len)
    attn_r = nn.functional.softmax(dots_r,dim=-1).transpose(-1,-2)
    attn_r = self.drop_attn(attn_r)

    X = self.split_heads(self.ln_1(self.combine_heads(torch.matmul(attn,V1))))/2 + self.split_heads(self.ln_2(self.
        combine_heads(torch.matmul(V2,attn_r))))/2

    return X
```

B. Experiment Configurations

In this section, we report the configurations in Tables 8-11 for the experiments in Section IV.

Algorithm 2 Smoother component

```
class Smoother(nn.Module):

   def __init__(self, hidden_size, seq_len, dropout = 0.5, num_head = 2,transformer_dim = 64, fold = 1):

      super(Smoother, self).__init__()

      self.hidden_size = hidden_size
      self.seq_len = seq_len
      self.dropout = dropout
      self.num_head = num_head
      self.dim = transformer_dim
      self.fold = fold

      self.weights_fft = nn.Parameter(torch.empty(self.seq_len//2+1, self.hidden_size,2))
      nn.init.kaiming_normal_(self.weights_fft, mode='fan_in', nonlinearity='relu')

      self.tiny_conv_linear = torch.nn.Conv1d(in_channels = self.hidden_size*2 , out_channels = self.hidden_size,
          kernel_size = 3, padding= 1, groups = 1)
      self.dropout = torch.nn.Dropout(p=self.dropout)
      self.bn_1 = nn.BatchNorm1d(self.seq_len)

   def forward(self, x):

      #### Compute Segment Average ####
      B,S,H = x.shape
      u = x.reshape(B,S,self.fold,H//self.fold)
      u = torch.mean(u,dim = -1)

      #### Fourier Convolution ####
      fft_u = fft.rfft(u, n = self.seq_len, axis = -2)
      fft_u = torch.view_as_real(fft_u)
      fft_u = fft_u.repeat(1,1,H//self.fold,1)
      self.weight_used = self.weights_fft.unsqueeze(0)
      temp_real = fft_u[...,0]*self.weight_used[...,0] - fft_u[...,1]*self.weight_used[...,1]
      temp_imag = fft_u[...,0]*self.weight_used[...,1] + fft_u[...,1]*self.weight_used[...,0]
      out_ft = torch.cat([temp_real.unsqueeze(-1),temp_imag.unsqueeze(-1)],dim = -1)
      out_ft = torch.view_as_complex(out_ft)
      m = fft.irfft(out_ft, n = self.seq_len, axis = -2)

      #### Convolution Stem ####
      input_h = torch.cat((m, x), dim = -1)
      h = self.tiny_conv_linear(input_h.permute(0,2,1)).permute(0,2,1)
      h = self.dropout(F.relu(self.bn_1(h)))

      return h
```

Algorithm 3 pseudo code for Time-Series Forecasting

```
   def forward(self, x_in):
      B1,H1,C1 = x_in.shape
      for i in range(len(self.encoder)):
         attn_layer = self.encoder[i]
         #standardize the input data
         if i == 0:
            tmp_mean = torch.mean(x_in[:,:,:],dim = 1,keepdim = True)
            tmp_std = torch.sqrt(torch.var(x_in[:,:,:],dim = 1,keepdim = True)+1e0)
            x_in = (x_in - tmp_mean)/(tmp_std)

            enc_out1 = self.enc_embedding(x_in)

         enc_out1= attn_layer(enc_out1) + enc_out1

      #decoder via Fourier Extrapolation
      dec_out = self.fourierExtrapolation(post(enc_out1))
      output = (dec_out.reshape(B1,-1,C1))*(tmp_std)+tmp_mean
      return output
```

Table 8: Experiment Configuration of S^{3}Attention $(r, s_1, s_2 = 8)$.

Parameters	ListOps	Text	Retrieval	Image	Pathfinder
Epoch	5	30	15	60	100
Learning Rate	1e-4	1e-4	1e-4	1e-3	1e-4
Weight Decay	0	1e-2	1e-2	1e-2	1e-2
Batch Size	32	32	32	256	256
r, s_1, s_2	8,8,8	8,8,8	8,8,8	8,8,8	8,8,8
dropout in embedding	0	0.5	0.1	0.1	0
dropout in attention	0	0.1	0.1	0.1	0
dropout in smoother	0	0.5	0.1	0.5	0.5

Algorithm 4 Fourier Extrapolation

```
class fourierExtrapolation(nn.Module):
   def __init__(self,inputSize,n_harm = 8,n_predict = 96):
      super().__init__()
      self.n = inputSize
      self.n_harm = n_harm
      self.f = torch.fft.fftfreq(self.n)
      self.indexes = list(range(self.n))


      # sort indexes by frequency, lower -> higher
      self.indexes.sort(key = lambda i: torch.absolute(self.f[i]))
      self.indexes = self.indexes[:1 + self.n_harm * 2]

      self.n_predict = n_predict


      # compute init phase
      self.t = torch.arange(0, self.n + self.n_predict)
      self.t1 = self.t.unsqueeze(0).unsqueeze(-1).float().to('cuda')
      self.f = self.f.unsqueeze(0).unsqueeze(-1).to('cuda')
      self.t = self.t.unsqueeze(0).unsqueeze(-1).unsqueeze(-1)to('cuda')
      self.g = self.f[:,self.indexes,:].permute(0,2,1).unsqueeze(1)
      self.phase_init = 2 * 3.1415 * self.g * self.t


   def fourierExtrapolation(self,x):

      # x in frequency domain
      x_freqdom = torch.fft.fft(x,dim = -2)
      x_freqdom = torch.view_as_real(x_freqdom)
      # select importance frequencies
      x_freqdom = x_freqdom[:,self.indexes ,:,:]
      x_freqdom = torch.view_as_complex(x_freqdom)
      ampli = torch.absolute(x_freqdom) / self.n # amplitude
      phase = torch.angle(x_freqdom) # phase

      ampli = ampli.permute(0,2,1).unsqueeze(1)
      phase = phase.permute(0,2,1).unsqueeze(1)

      self.restored_sig = ampli * torch.cos(self.phase_init + phase)

      return torch.sum(self.restored_sig,dim = -1)
```

Table 9: Experiment Configuration of S^{3}Attention (best).

Parameters	ListOps	Text	Retrieval	Image	Pathfinder
Epoch	10	30	15	60	100
Learning Rate	1e-4	1e-4	1e-4	1e-3	1e-4
Weight Decay	1e-2	1e-2	1e-2	1e-2	1e-2
Batch Size	32	32	32	256	256
r, s_1, s_2	8,8,8	8,8,8	8,32,32	8,16,16	8,128,32
dropout in embedding	0	0.5	0.1	0.5	0.1
dropout in attention	0	0.1	0.1	0.1	0.1
dropout in smoother	0	0.5	0.1	0.5	0.5

Table 10: Experiment Configuration for Model Parameters Impact.

Parameters	ListOps	Text	Retrieval	Image	Pathfinder
Epoch	5	30	15	60	100
Learning Rate	1e-4	1e-4	1e-4	1e-3	1e-4
Weight Decay	0	1e-2	1e-2	1e-2	1e-2
Batch Size	32	32	32	256	256
dropout in embedding	0	0.5	0.1	0.1	0
dropout in attention	0	0.1	0.1	0.1	0
dropout in smoother	0	0.5	0.1	0.5	0.5

C. Additional Results on LRA

We have already provided the average of five runs with different random seeds in Table 1. Here we also provide the standard deviations for these experiments in Table 12.

D. Dataset and Implementation Details

In this section, we summarize the details of the datasets used in this paper as follows:

Table 11: Experiment Configuration for Ablation.

Parameters	ListOps	Text	Retrieval	Image	Pathfinder
Learning Rate	1e-4	1e-4	1e-4	1e-3	1e-4
Weight Decay	0	1e-2	1e-2	1e-2	1e-2
Batch Size	32	32	32	256	256
r, s_1, s_2	8,8,8	8,8,8	8,8,8	8,8,8	8,8,8
dropout in embedding	0	0.5	0.1	0.1	0
dropout in attention	0	0.1	0.1	0.1	0
dropout in smoother	0	0.5	0.1	0.5	0.5

Table 12: Accuracy on Long Range Arena (LRA) with standard errors shown in parenthesis. All results are averages of 5 runs with different random seeds.

Model	LisOps	Text	Retrieval	Image	Pathfinder
S^3Attention ($r, s_1, s_2 = 8$)	38.30 (0.40)	69.27 (0.83)	83.26 (0.45)	53.90 (1.54)	75.82 (0.97)
S^3Attention (best)	39.15 (0.48)	71.58 (0.95)	83.73 (0.61)	57.73 (1.83)	78.20 (1.32)

LRA datasets: **ListOps**(2K length mathematical expression task which investigates the parsing ability); **Text** (up to 4K byte/character-level document classification task that tests capacity in character compositionality); **Retrieval** (byte/character-level document matching task, which examines the information compression ability with two 4K length sequences); **Image** (pixel-wise sequence image classification based on the CIFAR-10 dataset); **Pathfinder** (long-range spatial dependency identification task. The input images contain two small points/circles and dash-line paths. The model needs to identify whether two points/circles are connected); The LRA has several desirable advantages that made us focus on it as the evaluation benchmark: **generality** (only requires the encoder part); **simplicity** (data augmentation and pretraining are out of scope); **challenging long inputs** (difficulty enough and room to improve); **diversity aspects** (tasks covering math, language, image, and spatial modeling); and **lightweight** (run with low resource requirement).

Time series datasets:1) ETT [81] dataset contains two sub-datasets: ETT1 and ETT2, collected from two separated counties. Each of them has two versions of sampling resolutions (15min & 1h). ETT dataset contains multiple time series of electrical loads and one time sequence of oil temperature. 2) Electricity[3] dataset contains the electricity consumption for more than three hundred clients with each column corresponding to one client. 3) Exchange [85] dataset contains the current exchange of eight countries. 4) Traffic[4] dataset contains the occupation rate of freeway systems in California, USA. 5) Weather[5] dataset contains 21 meteorological indicators for a range of one year in Germany. 6) Illness[6] dataset contains the influenza-like illness patients in the United States. Table 13 summarizes all the features of the six benchmark datasets. They are all split into the training set, validation set and test set by the ratio of 7:1:2 during modeling.

Table 13: Details of time series benchmark datasets.

DATASET	LENGTH	DIMENSION	FREQUENCY
ETTM2	69680	8	15 MIN
EXCHANGE	7588	9	1 DAY
WEATHER	52696	22	10 MIN
ELECTRICITY	26304	322	1H
ILI	966	8	7 DAYS
TRAFFIC	17544	863	1H

GLUE datasets: The GLUE benchmark covers various natural language understanding tasks and is widely used in evaluating transferring ability. The tasks can be divided into two types, single-sentence tasks (SST-2 and CoLA), and sentence-pair tasks (MNLI, QQP, QNLI, STS-B, MRPC, and RTE). Following the same settings in [2], we exclude WNLI task.

E. Experiments on the Smoothness Effect of Fourier Convolution

In this section, we verify Fourier convolution component in the Smoother block can reduce the incoherence value in the early training stage. We use S^3Attention with ($r, s_1, s_2 = 8$) as the test model and test on an NLP dataset: Text, and a vision dataset: Pathfinder. We compute the μ-incoherence value. of the token matrix before and after the Fourier convolution (denoted as $\mu_{\boldsymbol{X}}$ and $\mu_{\boldsymbol{X}^{\text{smooth}}}$, respectively) for each sample in the validation dataset. Since we do not explicitly force the token matrix to be

[3]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014.
[4]http://pems.dot.ca.gov.
[5]https://www.bgc-jena.mpg.de/wetter.
[6]https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html.

low-rank required by Definition 1, we report the incoherence value for different rankness settings ($\mathrm{rank} = 16$ and $\mathrm{rank} = 32$) approximately, and the mean and standard deviation of incoherence value can be found in Table 14. The average incoherence value was reduced 30% after the Fourier convolution in both datasets. Moreover, We observe that the standard deviation significantly decreases, which suggests the Fourier convolution may also potentially stabilize the training procedure.

Table 14: The average incoherence parameters after 100 training steps with standard errors shown in the parenthesis.

Dataset	$\mu_{\boldsymbol{X}}$ $(\mathrm{rank} = 32)$	$\mu_{\boldsymbol{X}^{\mathrm{smooth}}}$ $(\mathrm{rank} = 32)$	$\mu_{\boldsymbol{X}}$ $(\mathrm{rank} = 16)$	$\mu_{\boldsymbol{X}^{\mathrm{smooth}}}$ $(\mathrm{rank} = 16)$
Text	2.75 (0.027)	2.05 (0.007)	3.98 (0.046)	3.23 (0.038)
Pathfinder	3.83 (0.221)	1.99 (0.001)	4.88 (0.264)	3.48 (0.001)

Table 15: The training configurations for Pretraining and GLUE tasks

	Pre-training	GLUE
Max Steps	1000K	-
Max Epochs	-	[4,20]
Learning Rate	1e-4	[5e-5,1e-4]
Batch Size	256	[16,32]
Warm-up Steps	5000	-
Sequence Length	512	128
Learning Rate Decay	-	Linear
Clip	-	1
Dropout	-	0.1