Sketching as a Tool for Understanding and Accelerating Self-attention for Long Sequences

Yifan Chen[1]*, **Qi Zeng**[1]*, **Dilek Hakkani-Tur**[2], **Di Jin**[2], **Heng Ji**[1], **Yun Yang**[1]
[1]University of Illinois Urbana-Champaign [2]Amazon Alexa AI
[1]{yifanc10, qizeng2, hengji, yy84}@illinois.edu
[2]{hakkanit, djinamzn}@amazon.com

Abstract

Transformer-based models are not efficient in processing long sequences due to the quadratic space and time complexity of the self-attention modules. To address this limitation, Linformer and Informer reduce the quadratic complexity to linear (modulo logarithmic factors) via low-dimensional projection and row selection, respectively. These two models are intrinsically connected, and to understand their connection we introduce a theoretical framework of matrix sketching. Based on the theoretical analysis, we propose Skeinformer to accelerate self-attention and further improve the accuracy of matrix approximation to self-attention with column sampling, adaptive row normalization and pilot sampling reutilization. Experiments on the Long Range Arena benchmark demonstrate that our methods outperform alternatives with a consistently smaller time/space footprint[1].

1 Introduction

Transformer (Vaswani et al., 2017) utilizes softmax self-attention modules to capture the dependency between tokens in a sequence and has been widely used in various Natural Language Processing tasks. The time and space complexity of the dot-product self-attention is quadratic in the input sequence length, which restricts the largest sequence length and batch size. To adapt transformers to long sequences, documents have to be truncated, chunked using a sliding window, or processed in parallel on multiple GPUs. These additional operations usually cause the loss of long-range dependency and introduce additional computational costs.

In this paper, we focus on efficient self-attention methods (Xiong et al., 2021; Qiu et al., 2020; Zaheer et al., 2020; Beltagy et al., 2020; Kitaev et al., 2020a; Roy et al., 2021), among which Linformer (Wang et al., 2020b) and Informer (Zhou et al., 2020) are two representative approaches to reducing the $\mathcal{O}(n^2)$ self-attention to an $\widetilde{\mathcal{O}}(n)$ operation ($\widetilde{\mathcal{O}}(\cdot)$ means $\mathcal{O}(\cdot)$ modulo poly-log terms and n is the sequence length) in both space and time complexity. Linformer forms a low-rank factorization of the original attention by decomposing it into smaller attentions, while Informer allows each key to only attend to a certain number of queries.

To better understand self-attention, we introduce a theoretical framework, sketching (Woodruff, 2014), to help explain the key ideas in Informer and Linformer from the perspective of matrix approximation. Specifically, sketching methods replace the original matrix $\boldsymbol{B}$ with its random sketch $\boldsymbol{BS}$ to reduce computations. In Section 3.3 we introduce some concrete instances of commonly used distributions for constructing the random sketching matrix $\boldsymbol{S}$. Furthermore, taking matrix approximation as a general guideline, we recognize the deficiency in Informer and Linformer, that they either do not fully utilize the information in the value matrix $\boldsymbol{V}$, or deviate from the original self-attention output. This guideline also motivates us to propose **Skeinformer** through the theoretical analysis under the sketching framework.

To improve the approximation accuracy in terms of the original attention output, Skeinformer applies sub-sampling sketching to reduce time complexity and exploits the information from the value matrix $\boldsymbol{V}$ with **column sampling**. Skeinformer also incorporates an **adaptive row normalization** step, which approximates the un-selected rows by a vector with all elements $\frac{1}{n}$ and has significantly boosted the performance of Informer. In addition, we introduce a simple yet effective step, **pilot sampling reutilization**, which reuses the computation from pilot sampling to improve both approximation accuracy and training efficiency. Our experiments on the LRA benchmark show that Skeinformer con-

*Equal contribution. The majority of this work was done prior to the first author's internship at Amazon Alexa AI.

[1]Our code is released at https://github.com/pkuzengqi/Skeinformer

Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5187 - 5199

sistently uses less space and time while achieving better accuracy than most baseline methods.

In summary, our contributions are twofold:

- We introduce sketching as a theoretical framework for analyzing and developing efficient transformers. Specifically, the randomized sketching theory covers these two methods from the perspective of approximate matrix multiplication. This framework connects the studies on efficient transformers and randomized sketching theory, so that future development in efficient transformers and sketching can benefit each other.

- We propose Skeinformer as a straightforward product of the sketching framework to accelerate the training and inference of transformers. Skeinformer consists of three components: the initial column sampling that incorporates the information from the value matrix $\boldsymbol{V}$ into the sampling probabilities, the adaptive row normalization that fills un-selected columns with the averaged selected columns, and the pilot sampling re-utilization.

2 Related Work

The ability to process long sequences is critical for many Natural Language Processing tasks, including Document Summarization (Xiao and Carenini, 2019; Huang et al., 2021), Question Answering (Wang et al., 2020a), Information Extraction (Li et al., 2021; Du and Cardie, 2020; Ebner et al., 2020; Du et al., 2022), and Machine Translation (Bao et al., 2021). However, the quadratic computational cost of self-attention in transformer-based models limits their application in long-sequence tasks. Recent methods have been proposed to accelerate attention computation by selectively attending to a subset of the tokens or with low-rank matrix approximation (Tay et al., 2020b).

Selective attention methods limit the scope of matrix operation with sparse attention patterns or column/row sampling methods. BlockBERT (Qiu et al., 2020) introduces sparse block structures into the attention matrix. Sparse Transformer (Child et al., 2019) introduces dilated patterns. Big Bird (Zaheer et al., 2020) proposes a combination of random, window, and global attention. Longformer (Beltagy et al., 2020) combines local windowed attention with task-motivated global attention. The most related work to ours is Informer (Zhou et al., 2020), which allows each key to only attend to the top queries under the Kullback-Leibler divergence based sparsity measurement.

Low-rank attention matrix approximation methods are based on the assumption of low-rank structure in the full self-attention matrix. Linformer (Wang et al., 2020b) compresses the size of the key and value matrices by the Johnson–Lindenstrauss transform (Johnson and Lindenstrauss, 1984). Performer (Choromanski et al., 2020) recognizes the attention score matrix as an empirical Gaussian kernel matrix and constructs a low-rank projection for both the query and key matrices through random Fourier features (Rahimi and Recht, 2007). Nyströmformer (Xiong et al., 2021) instead utilizes Nyström method (Williams and Seeger, 2000; Drineas and Mahoney, 2005) to approximate the attention score matrix. Skyformer (Chen et al., 2021) replaces the softmax structure with a Gaussian kernel and adapts the Nyström method to accelerate the computation.

Some other methods follow a similar principle to decompose the attention score matrix, although they are not necessarily aiming to approximate the original self-attention itself. The representative methods include Linear Transformer (Katharopoulos et al., 2020), which claims that the exponential transform of the dot-product in the softmax operation can be replaced by the direct matrix multiplication of the projected query and key matrices, and Reformer (Kitaev et al., 2020b), which forces the query and key matrices to be identical and applies locality-sensitive hashing (LSH) (Har-Peled et al., 2012) to simplify the computation of the attention score matrix. Those methods are effective alternatives of the original self-attention, while they do not fall into the scope of matrix approximation. We spare the discussion of those methods in this paper.

3 Sketching Framework

3.1 Problem Formulation

Given an input sequence $\boldsymbol{X} \in \mathbb{R}^{n \times d_{input}}$, where n is the sequence length and d_{input} is the embedding dimension, the dot-product attention for a single attention head in transformer (Vaswani et al., 2017) is defined as

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{p}}\right)\boldsymbol{V}$$

where $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q$, $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K$, and $\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V$. $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{d_{input} \times p}$ are the query,

Algorithm 1: Skeinformer.

Input: query matrix $\boldsymbol{Q}$, key matrix $\boldsymbol{K}$, value matrix $\boldsymbol{V}$ (all are n-by-p), and sub-sample size d
Output: Attention output matrix $\boldsymbol{R}$ with the same shape as $\boldsymbol{V}$

Uniformly sample d indices $j_1, \cdots, j_d$ with replacement;
Construct the $d \times p$ matrix $\boldsymbol{Q}_J$ as to the index set $J := \{j_k\}_{k=1}^d$, whose k-th row is $\boldsymbol{Q}_{(j_k)}$;
Compute the matrix $\boldsymbol{B}_J = \text{softmax}\left(\boldsymbol{Q}_J \boldsymbol{K}^T / \sqrt{p}\right)$; `// pilot sampling`
Based on $\boldsymbol{B}_J$, give the estimated sub-sampling probabilities $\{\hat{p}_i\}_{i=1}^n$ as in Equation (5);
With $\{\hat{p}_i\}_{i=1}^n$ sample d indices $j'_1, \cdots, j'_d$ without replacement;
Construct the d-by-p matrix $\boldsymbol{K}_{J'}$ (resp., $\boldsymbol{V}_{J'}$) according to the indices list $J' := \{j'_k\}_{k=1}^d$, whose k-th row is $\boldsymbol{K}_{(j'_k)}$ (resp., $\boldsymbol{V}_{(j'_k)}$);
Compute the two matrices $\boldsymbol{A}^{J'} = \exp\left(\boldsymbol{Q}\boldsymbol{K}_{J'}^T / \sqrt{p}\right)$, and $\boldsymbol{R}_{J'} = \boldsymbol{A}^{J'}\boldsymbol{V}_{J'}$; `// column sampling`
Construct a length n column vector $\boldsymbol{g}$ whose i-th element is $(\prod_{k=1}^d a_{ij'_k})^{\frac{1}{d}}, \forall i \in [n]$;
Compute the row sum vector $\boldsymbol{d} := \boldsymbol{A}^{J'}\mathbf{1}_d + (n-d)\boldsymbol{g}$; `// adaptive row normalization`
Denote the un-selected part of $\boldsymbol{V}$ as $\boldsymbol{V}_{(J')^C}$, and compute the vector $\boldsymbol{v} = \boldsymbol{V}_{(J')^C}^T \mathbf{1}_{n-d}$;
Obtain the intermediate output $\boldsymbol{R} = \text{diag}(\boldsymbol{d}^{-1})(\boldsymbol{R}_{J'} + \boldsymbol{g}\boldsymbol{v}^T)$, where $\boldsymbol{d}^{-1}$ is the element-wise inverse of $\boldsymbol{d}$;
Compute $\boldsymbol{B}_J\boldsymbol{V}$ and assign it to the corresponding rows of $\boldsymbol{R}$; `// pilot sampling reutilization`
Return the matrix $\boldsymbol{R}$ as the ultimate output of this algorithm;

key, and value weight metrics that linearly project the input $\boldsymbol{X}$ of dimension d_{input} to an output tensor of dimension p.

To ease the future analysis, the softmax term can be rewritten into $\boldsymbol{D}^{-1}\boldsymbol{A}$, where $\boldsymbol{A} := \exp(\mathbf{Q}\mathbf{K}^T/\sqrt{p})$, and $\boldsymbol{D}$ is a diagonal matrix whose diagonal is $\exp(\boldsymbol{Q}\boldsymbol{K}^T/\sqrt{p}) \cdot \mathbf{1}$ ($\mathbf{1}$ is a size-n vector with all elements being 1).

3.2 Sketching Methods

Beyond current attempts to accelerate self-attention, research in the random matrix approximation community can be potentially applied to fast attention. Among the theoretical frameworks, the sketching method (Woodruff, 2014) is the most comprehensive one as its general concept can incorporate many different approaches.

The core idea of the sketching method is to replace an original matrix $\boldsymbol{B} \in \mathbb{R}^{n_B \times n}$ with its random sketch $\boldsymbol{B}\boldsymbol{S}$, where $\boldsymbol{S} \in \mathbb{R}^{n \times d}$ is a random sketching matrix. In practice, to apply the sketching method we plug an identity matrix into the original expression, and then formally replace the identity matrix with the product $\boldsymbol{S}\boldsymbol{S}^T$, as the distribution of $\boldsymbol{S}$ is usually designed to satisfy the constraint that

$$\mathbb{E}(\boldsymbol{S}\boldsymbol{S}^T) = \boldsymbol{I}. \tag{1}$$

Common methods to construct a sketching matrix include sub-Gaussian maps (Vershynin, 2010; Halko et al., 2011), subsampled randomized Hadamard transform (SRHT) (Ailon and Chazelle, 2006; Lu et al., 2013; Yang et al., 2017), sparse oblivious subspace embeddings (Cohen et al., 2016), very sparse random projection (Li et al., 2006), accumulative sketching (Chen and Yang, 2021), and sub-sampling sketching (Monte Carlo algorithms) (Drineas et al., 2006). Specifically, Informer and Linformer, two efficient transformer-based methods mentioned above, can be understood as applications of sub-sampling sketching and sub-Gaussian maps, respectively. We further elaborate the connections in the next subsection.

3.3 Sketching in Self-attention Approximation

A naïve step in applying sketching method to approximate the self-attention output $\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{V}$ is to construct a random sketch of the un-normalized attention score matrix $\boldsymbol{A}$, the bottleneck in computation. Informer and Linformer construct two types of sketches, $\boldsymbol{A}^T\boldsymbol{S}$ and $\boldsymbol{A}\boldsymbol{S}$ respectively.

3.3.1 Informer

Informer selects d important rows of $\boldsymbol{D}^{-1}\boldsymbol{A}$, though deterministically, to represent $\boldsymbol{D}^{-1}\boldsymbol{A}$. This process can be related to a sketched approximation $\boldsymbol{D}^{-1}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{A}$, where $\boldsymbol{S}$ is a sub-sampling matrix defined as follows:

Definition 3.1 (Sub-sampling matrix)**.** *Consider a discrete distribution which draws i with probability $p_i > 0, \forall i \in [n]$. For a random matrix $\boldsymbol{S} \in \mathbb{R}^{n \times d}$, if $\boldsymbol{S}$ has independent and identically distributed (i.i.d.) columns and each column $\boldsymbol{S}^{(j)}$ is $\frac{1}{\sqrt{dp_i}}\boldsymbol{e}_i$ with probability p_i, where $\boldsymbol{e}_i$ is the i-th column of the n-by-n identity matrix $\boldsymbol{I}_n$, then $\boldsymbol{S}$ is called a sub-sampling matrix with sub-sampling probabilities $\{p_i\}_{i=1}^n$.*

Some researchers in the field of approximate ma-

trix multiplication have provided a practical guideline for the choice of the sub-sampling probabilities $\{p_i\}_{i=1}^n$ in $\boldsymbol{S}$. Specifically for the matrix multiplication $\boldsymbol{BC}$ of two arbitrary matrices $\boldsymbol{B}$ and $\boldsymbol{C}$, Drineas et al. (2006) approximate it with $\boldsymbol{BSS}^T\boldsymbol{C}$ and set the sampling probability p_i in $\boldsymbol{S}$ proportional to the product $\|\boldsymbol{B}^{(i)}\|_2\|\boldsymbol{C}_{(i)}\|_2$, where $\boldsymbol{B}^{(i)}$ is the i-th column in matrix $\boldsymbol{B}$ and $\boldsymbol{C}_{(i)}$ is the i-th row in matrix $\boldsymbol{C}$. For the product $\boldsymbol{D}^{-1}\boldsymbol{A}$, the probability in sketching will be $p_i = \frac{\sqrt{\sum_{j=1}^n a_{ij}^2}}{\sum_{j=1}^n a_{ij}}$, where a_{ij} is the j-th element of the i-th row in matrix $\boldsymbol{A}$.

The above sampling probability choice $\{p_i\}_{i=1}^n$ is highly related to the sparsity measurement used in Informer, which is $M_i = \ln \frac{\sum_{j=1}^n a_{ij}}{(\prod_{j=1}^n a_{ij})^{1/n}}$. Here p_i is the ratio between the quadratic mean and the arithmetic mean of $\{a_{ij}\}_{j=1}^n$; M_i is the logarithm of the ratio between the arithmetic mean and the geometric mean. It is clear that M_i will increase with p_i as these two ratios will both be large when $\{a_{ij}\}_{j=1}^n$ are highly non-uniform. We conclude that in Informer, the main idea to select the rows with high sparsity measurement can be taken as a special variant of the sub-sampling method above with probabilities $\{p_i\}$.

3.3.2 Linformer

Another type of sketch $\boldsymbol{AS}$ is mentioned (but not finally used) in Linformer. The sketching matrix $\boldsymbol{S}$ takes a form different from sub-sampling. The construction of $\boldsymbol{S}$ in Linformer is motivated by Johnson-Lindenstrauss (JL) transform, which applies the sketching matrix $\boldsymbol{S}$ satisfying the (ε, δ)-JL guarantee:

Definition 3.2 (Oblivious Johnson-Lindenstrauss guarantee (Johnson and Lindenstrauss, 1984))**.** *A distribution $\mathcal{D}$ over $\mathbb{R}^{n\times d}$ satisfies "oblivious Johnson-Lindenstrauss guarantee" if for some $\varepsilon, \delta \in (0, 1/2)$:*

$$\forall \boldsymbol{b} \in \mathbb{R}^n, \mathop{\mathbb{P}}_{\boldsymbol{S}\sim\mathcal{D}}\left(\left|\|\boldsymbol{Sb}\|_2^2 - \|\boldsymbol{b}\|_2^2\right| > \varepsilon\|\boldsymbol{b}\|_2^2\right) < \delta. \tag{2}$$

Specifically, a matrix with i.i.d. Gaussian elements can meet the above requirement. It has been proven (Johnson and Lindenstrauss, 1984) that with $d = \mathcal{O}(\varepsilon^{-2}\log(1/\delta))$, a Gaussian sketching matrix $\boldsymbol{S}$ can satisfy the oblivious (ε, δ)-JL guarantee. To extend the conclusion from a single vector $\boldsymbol{b} \in \mathbb{R}^n$ to a matrix $\boldsymbol{B} \in \mathbb{R}^{n_B\times n}$, the size d still needs to suffer from an additional $\log n_B$ term (Vershynin, 2010), which matches the bound in sub-sampling sketching (Drineas et al., 2006, Theorem 1).

However, the direct use of Gaussian sketching matrix, i.e. the approximation $\boldsymbol{D}^{-1}\boldsymbol{ASS}^T\boldsymbol{V}$ (Wang et al., 2020b, Eqn. (5)), requires the computation of the whole matrix $\boldsymbol{A}$. To avoid this computational burden, Linformer replaces the form of sketching method with softmax $\left((\boldsymbol{QK}^T/\sqrt{p})\boldsymbol{S}\right)\boldsymbol{S^TV}$, which sacrifices the accuracy for efficiency in some tasks as shown in later experimental results.

4 Methodology: Skeinformer

Based on the previous discussion, we observe that Informer omits the information from the value matrix $\boldsymbol{V}$, and Linformer deviates from the usual sketching form for efficiency. To address these issues and fully exploit the power of sketching, we strengthen the attention approximation with the following components.

In Section 4.1, we introduce column importance sampling, which allows the information incorporation from $\boldsymbol{V}$ to accelerate the matrix multiplication (compared to JL transform); in Section 4.2 and Section 4.3, we leverage the sampled columns to perform the row normalization and reuse the pilot row sampling, which further improves the approximation and makes the training more stable.

We describe the proposed method **Skeinformer** in Algorithm 1 and verify its performance on matrix approximation in Section 5. We also provide complexity analysis in Section 4.5 to show that our method enjoys the same $\mathcal{O}(n\log n)$ complexity as other methods.

4.1 Column Sampling

The row selection in Informer has been derived as a special variant of the sketching method and can be further improved by utilizing the information from $\boldsymbol{V}$, in a form similar to Linformer:

$$\boldsymbol{D}^{-1}\boldsymbol{ASS}^T\boldsymbol{V},$$

where $\boldsymbol{S}$ above is a sub-sampling matrix defined in Definition 3.1 with sampling probabilities

$$p_i \propto \|(\boldsymbol{D}^{-1}\boldsymbol{A})^{(i)}\|_2\|\boldsymbol{V}_{(i)}\|_2, \quad i = 1, 2, \ldots, n.$$

We remark that using the sub-sampling sketching in this way can both circumvent the computation burden of Gaussian sketching, and also allow the incorporation of the information from $\boldsymbol{V}$.

As $\boldsymbol{S}$ formally samples some columns from $\boldsymbol{D}^{-1}\boldsymbol{A}$, we name the procedure as column sampling in our method. The performance regarding

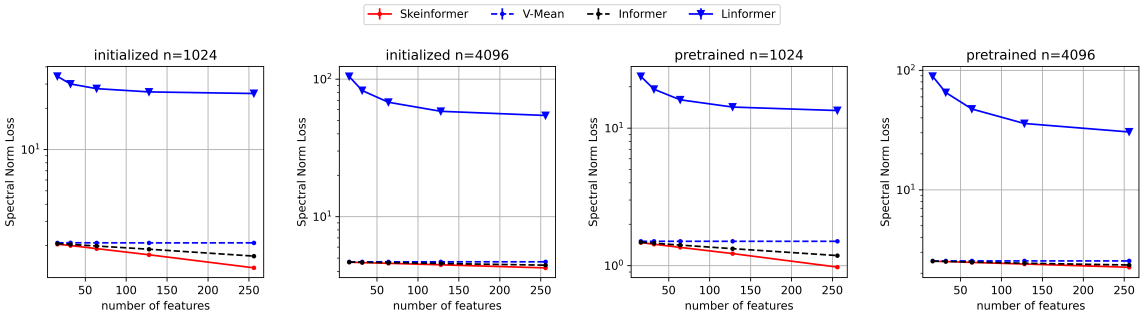

Figure 1: Spectral norm results with the sequence length of 1024 and 4096. Y axis: Lower percentage score means better approximation. X axis: Higher number of feature means larger computation cost. "V-Mean" is an artificial baseline against sampling that always uses a rank-one matrix $\frac{1}{n}\mathbf{1}\mathbf{1}^T V$ to approximate the original self-attention.

the Frobenius norm loss of the approximate matrix multiplication can be guaranteed by the following proposition:

Proposition 1 (Adapted from Theorem 1 (Drineas et al., 2006)). *Suppose the attention score matrix $\boldsymbol{B} := \boldsymbol{D}^{-1}\boldsymbol{A} \in \mathbb{R}^{n\times n}$, the value matrix $\boldsymbol{V} \in \mathbb{R}^{n\times p}$, the number of sampled columns $d \in \mathbb{Z}^+$ such that $1 \le d \le n$, and the sampling probabilities $\{p_i\}_{i=1}^n$ are such that $\sum_{i=1}^n p_i = 1$ and such that for a quality coefficient $\beta \in (0, 1]$*

$$p_i \ge \beta \frac{\|B^{(i)}\|\|V_{(i)}\|}{\sum_{i'=1}^n \|B^{(i')}\|\|V_{(i')}\|}, \forall i \in [n]. \quad (3)$$

Construct a sub-sampling matrix $\boldsymbol{S} \in \mathbb{R}^{n\times d}$ with sub-sampling probabilities $\{p_i\}_{i=1}^n$ as in Definition 3.1, and let $\boldsymbol{B}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{V}$ be an approximation to $\boldsymbol{B}\boldsymbol{V}$. Let $\delta \in (0, 1)$ and $\eta = 1+\sqrt{(8/\beta)\log(1/\delta)}$. Then with probability at least $1-\delta$,

$$\|\boldsymbol{B}\boldsymbol{V} - \boldsymbol{B}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{V}\|_F^2 \le \frac{\eta^2}{\beta d}\|\boldsymbol{B}\|_F^2\|\boldsymbol{V}\|_F^2. \quad (4)$$

Remark. Proposition 1 guides Informer and our method to pick the important rows and columns of the attention score matrix $\boldsymbol{B}$. In self-attention, it is feasible to compute the norm $\|V_{(i')}\|$ of each row in $\boldsymbol{V}$ with $\mathcal{O}(n)$ time, assuming the dimension p in each head is fixed and independent of n. However, similar to Informer, it is inefficient to exactly compute the ℓ-2 norm of each column in the n-by-n matrix $\boldsymbol{B}$, and we need pilot sampling as well to estimate the norm of the columns in $\boldsymbol{B}$. We show that $\mathcal{O}(\log n)$ samples in the pilot sampling are sufficient to guarantee the quality coefficient $\beta \ge \sqrt{\frac{1}{3}}$ with high probability by the following lemma. (See proof in Appendix.)

Lemma 1. *Assume for any $i \in [n]$, $\|\boldsymbol{B}^{(i)}\|^2/n$ is uniformly lower bounded by a constant C, where $\boldsymbol{B} := \boldsymbol{D}^{-1}\boldsymbol{A}$. For another constant $\delta \in (0, 1/2)$, we uniformly sample d indices $\{j_k\}_{k=1}^d$ from $[n]$ with replacement, and let d be a constant multiple of $\log(n/\delta)$. Then with probability at least $1-\delta$, the estimated sub-sampling probabilities*

$$\hat{p}_i := \frac{(\sum_{k=1}^d b_{j_k i}^2)^{\frac{1}{2}}\|V_{(i)}\|}{\sum_{i'=1}^n (\sum_{k=1}^d b_{j_k i'}^2)^{\frac{1}{2}}\|V_{(i')}\|}, \forall i \in [n], \quad (5)$$

satisfy the constraints (3) with $\beta = \sqrt{\frac{1}{3}}$, where b_{ji} is the element of $\boldsymbol{B}$ from the j-th row and i-th column.

This lemma states the sub-sampling weights used in our proposed method. Its computation only requires accesses to $\{\boldsymbol{B}^{(j_k)}\}_{k=1}^d$ obtained from the pilot sampling, and thus has greatly reduced the time cost. Combining the preceding lemma and Proposition 1, we conclude that with the sampling probabilities $\{\hat{p}_i\}_{i=1}^n$ estimated by $\mathcal{O}(\log n)$ pilot samples, the sampled d important columns suffice to capture the essence of the original output $\boldsymbol{B}\boldsymbol{V}$. We conclude this subsection with a remark that the theoretical result that sub-sampling sketching can well approximate the original self-attention, indeed matches the rank collapse phenomenon observed by Dong et al. (2021) that self-attention can be well approximated by a low-rank matrix.

4.2 Adaptive Row Normalization

In addition to the theoretical guarantee of the sub-sampling sketching method, we identify an important component behind Informer, row normalization, which implicitly fills the un-selected rows with $\frac{1}{n}$. The experiments in Section 5 reveal that even the rank-one pure row normalization benchmark $\frac{1}{n}\mathbf{1}\mathbf{1}^T V$, as an ablation, will have acceptable spectral norm loss $\|\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{V} - \frac{1}{n}\mathbf{1}\mathbf{1}^T V\|_2$. Therefore, we incorporate adaptive row normalization

to provide an even better attention score distribution in each row. It fills un-selected columns with the averaged selected columns. Moreover, from the model training perspective, it allows the whole value matrix $\boldsymbol{V}$ in Skeinformer to participate in the computation (compared to only using the sub-sampling sketch $\boldsymbol{S}^T\boldsymbol{V}$), and thus can improve the efficiency of updating $\boldsymbol{W}_V$ during the training.

Specifically, in adaptive row normalization any row in the matrix $\boldsymbol{A}$ can be divided into two parts, the exactly computed elements in the selected columns with indices $\{j'_k\}_{k=1}^d \subset [n]$ and the other elements in the un-selected columns. For the latter, in each row, we set all the un-selected elements as the geometric mean of the selected ones, considering the exponentiation in softmax. We then perform row normalization based on the above construction, in which the i-th diagonal element in $\boldsymbol{D}$ is estimated as

$$\hat{d}_{ii} = \sum_{k=1}^{d} a_{ij_k} + (n-d)(\prod_{k=1}^{d} a_{ij_k})^{\frac{1}{d}}, \quad (6)$$

where each a_{ij} is the corresponding element in matrix $\boldsymbol{A}$. Next we normalize rows composed of exact elements in the selected columns, and the other elements estimated with the mean value above. We comment that though the component of adaptive row normalization makes the proposed method inapplicable to Proposition 1, it benefits the performance on matrix approximation and avoid the cost to compute the diagonal normalization matrix $\boldsymbol{D}$. (c.f. Section 5)

4.3 Pilot Sampling Reutilization

Since we have already computed $\boldsymbol{B}_J$ in pilot sampling step (defined in Ln. 3 of Algorithm 1), we can exactly reproduce the d rows in the original self-attention output with an additional product $\boldsymbol{B}_J\boldsymbol{V}$ in $\mathcal{O}(n\log n)$ time. This allows for more precise approximation with little cost. In addition, the computation of those rows involves the whole key matrix $\boldsymbol{K}$, which benefits the training of the parameters $\boldsymbol{W}_K$.

4.4 Implementation Details

Applying the sub-sampling-based methods requires the support for padding masks commonly used in Natural Language Processing tasks. However, a naïve implementation of Algorithm 1 will result in the unnecessary sampling of the padding tokens. Therefore, we count the number of the unpadded tokens m, and only perform the pilot sampling within the certain range $[m]$. After the matrix $\boldsymbol{B}_J$ is computed, we set its columns belonging to the padded part to be all zero, so that the probability $\hat{p}_i$ of choosing column i from the padded part will be zero and the column will not be sampled in the later importance sampling. Similar modifications can also be applied to Informer to enable its applications in NLP tasks in Section 6.

4.5 Complexity Analysis

With the mild assumption in Lemma 1, we claim that our method can have an $\mathcal{O}(n\log n)$ time and space complexity. The claim is shown by the following analysis of the complexity, which heavily relies on the notations in Algorithm 1.

First, we point out that the row/column retrieving operation after index sampling can be implemented by only forming a view and thus the cost is negligible. For Line $1 \sim 4$ in Algorithm 1, the time complexity of the uniform pilot sampling is $\mathcal{O}(d) = \mathcal{O}(\log n)$, while the computation of the matrix $\boldsymbol{B}_J$ and the corresponding probabilities $\{\hat{p}_i\}_{i=1}^n$ costs $\mathcal{O}(nd) = \mathcal{O}(n\log n)$ time and space. For Lines $5 \sim 7$, with probabilities $\{\hat{p}_i\}_{i=1}^n$, the importance sampling takes $\mathcal{O}(n + d\log n) = \mathcal{O}(n)$ time, and similar to the computation above it takes $\mathcal{O}(n\log n)$ time and space to obtain $\boldsymbol{A}_{J'}$ and $\boldsymbol{R}_{J'}$. For Lines $8 \sim 10$, it is clear that the three vectors $\boldsymbol{g}$, $\boldsymbol{d}$, and $\boldsymbol{v}$ can be computed in $\mathcal{O}(n\log n)$ time. As for the last step in Line 11, since it just requires the matrix product involving a diagonal matrix, we can finish the computation also in $\mathcal{O}(n\log n)$ time and space. In summary, the total time and space complexity for Algorithm 1 is at most $\mathcal{O}(n\log n)$, much lower than the $\mathcal{O}(n^2)$ complexity for the original softmax self-attention.

Remark. The complexity above is derived based on the high probability bound in Proposition 1, which is different than the derivation by some previous methods to claim the linear $\mathcal{O}(n)$ complexity.

5 Approximation Evaluation

As a preliminary justification of our proposed methods, we compute the spectral norm loss, a common metric for approximate matrix multiplication, to evaluate the effect of different models to approximate the original self-attention. We compare the spectral norms of the differences between the outputs from vanilla self-attention and other fast attention methods given the same input $\mathbf{Q}, \mathbf{K}, \mathbf{V}$.

Specifically we compute $\|\boldsymbol{BV} - \boldsymbol{R}\|_2$, where $\boldsymbol{B} := \boldsymbol{D}^{-1}\boldsymbol{A}$ is the attention score matrix in the original method, and $\boldsymbol{R}$ is the output of each approximation method.

To construct the inputs $\mathbf{Q}, \mathbf{K}, \mathbf{V}$, we first truncate the raw text from Wikitext-2 dataset (Merity et al., 2017) into sequences of length 512. Then we transform the input X into $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ with the query, key, and value weight metrices from a pretrained model or a randomly initiated model.

We report the spectral norm loss of different sketching-based methods in Figure 1. The results are averaged over 768 trials, and the error bars in the figure represent the standard error of the reported values. For size d (x-axis), either the number of columns/rows selected or the projection dimension, it is set in the range from 2^3 to 2^8.

V-Mean uses a rank-one matrix $\frac{1}{n}\mathbf{1}\mathbf{1}^T V$ to approximate the original self-attention, and thus its approximation error does not change with the size d. V-Mean can also be seen as an ablation for the row normalization step (equivalent to adaptive row normalization without sub-samples). We observe the row normalization step greatly contributes to the approximation of self-attention that involves a softmax structure. Among the candidates, Skeinformer tends to have the smallest spectral norm loss, especially when d is large. It attains a higher accuracy than Informer and Linformer regarding the matrix approximation performance.

6 Experiment

6.1 Benchmark

We experiment on Long Range Arena (LRA) benchmark (Tay et al., 2020a), including ListOps (Nangia and Bowman, 2018), Text Classification (Maas et al., 2011), Document Retrieval (Radev et al., 2013), Pathfinder (Linsley et al., 2018), and Image Classification (Krizhevsky et al., 2009). LRA is designed for long-context scenarios and more appropriate for evaluating efficient transformers comparing to GLUE (Mutton et al., 2007) with shorter input context . Following (Xiong et al., 2021) we use a 2-layer transformer model with 64 embedding dimensions, 128 hidden dimensions, and 2 attention heads for all experiments. More details can be found in Appendix.

6.2 Baseline Methods

We compare our method with the standard quadratic self-attention (Vaswani et al., 2017), Big Bird (Zaheer et al., 2020), Linformer (Wang et al., 2020b), Informer (Zhou et al., 2020), Performer (Choromanski et al., 2020), and Nyströmformer (Xiong et al., 2021). In addition to their vanilla implementations, we compare with standard self-attention without dropout (since most fast attention methods do not have this component), Linformer with unreduced Johnson-Lindenstrauss Transform (the original form that Linformer deviates from), and Informer with padding masks.

Ablation studies include replacing the column sampling with uniform sampling, disabling the adaptive row normalization or replacing it with the simple row normalization implemented in Informer, and disabling the pilot sampling reutilization.

For clarification, deep transformers or pretrained language models are not appropriate baselines. Training a deep transformer from scratch requires large computational resources and much more data to converge, and therefore is not adopted by previous work. A shallow transformer structure, on the other hand, has been justified by previous work to be enough for fair comparison in attention acceleration performance. Pretrained models are trained for token-level text-based tasks, and are not suitable for image pixel sequences (as in Pathfinder and Image Classification), character sequences (as in Text Classification) and math operation sequences (as in ListOps).

6.3 Results

We conclude the results in Table 1 and Table 2 with the following observations:

Most $\widetilde{\mathcal{O}}(n)$ attention acceleration methods have comparable or better performance with vanilla attention. After all models converge to their long-time limits, Linformer tends to have worse performance possibly due to the violation of the sketching form, while Skeinformer has the best overall performance.

While surprising, those approximation methods tend to outperform the original transformer in most tasks. We speculate the reason behind this phenomenon is that a good approximation can recover the main signals in the original self-attention matrix, and also restrain the noise via the sparse / low-rank structure. A similar phenomenon can be found in CNN (Sanyal et al., 2018), that a low-rank regularizer, such as SVD, applied to the representation of the intermediate layers can allow the model to have lower prediction errors. This speculation

Models	Text	ListOps	Retrieval	Pathfinder	Image	Average
Standard (Vaswani et al., 2017)	57.69	38.15	80.10	73.59	37.97	57.50
· w/o dropout	59.44	38.17	79.35	72.35	37.58	57.38
V-Mean	65.29	28.78	80.49	61.01	34.33	53.98
BigBird (Zaheer et al., 2020)	61.91	38.86	79.73	71.75	35.00	57.45
Performer (Choromanski et al., 2020)	57.67	37.70	75.69	56.50	37.40	52.99
Nystromformer (Xiong et al., 2021)	60.91	37.76	79.87	72.53	31.93	56.60
Reformer (Kitaev et al., 2020a)	62.69	37.94	78.85	69.21	36.42	57.02
Linformer (Wang et al., 2020b)	58.52	37.97	77.40	55.57	37.48	53.39
· w/ unreduced JLT	59.12	37.48	79.39	68.45	35.96	56.08
Informer (Zhou et al., 2020)	61.55	38.43	80.88	59.34	36.55	55.35
· w/ padding mask	60.98	37.26	79.92	62.51	37.19	55.57
Skeinformer	62.47	38.73	80.42	71.51	37.27	**58.08**
· w/o column sampling	64.48	30.02	80.57	64.35	36.97	55.28
· w/o row normalization	60.67	37.69	78.67	66.35	37.06	56.09
· w/ simple row normalization	60.26	38.35	78.97	65.41	39.72	56.54
· w/o pilot sampling reutilization	62.39	38.12	79.88	71.53	37.20	57.83

Table 1: Classification accuracy (%) on the test sets of LRA benchmark. Skeinformer does not always outperform other baseline methods but has consistently comparable general performance. The approximation methods are not expected to outperform the original methods (standard self-attention) though they surprisingly do.

Models	Text		ListOps		Retrieval		Pathfinder		Image	
	time ↓	bz ↑	time ↓	bz ↑	time ↓	bz ↑	time ↓	bz ↑	time ↓	bz ↑
Standard	50.63	16	22.30	64	53.27	16	13.91	128	21.40	64
· w/o dropout	39.49	8	19.50	32	41.88	8	11.79	64	14.88	32
V-Mean	3.62	128	4.14	256	3.90	64	3.67	512	4.44	256
BigBird	20.59	64	17.28	128	21.73	32	17.83	256	18.84	256
Performer	2.63	64	9.31	128	12.50	32	10.40	256	8.94	256
Nyströmformer	12.18	64	12.28	128	13.35	32	19.58	128	10.30	256
Reformer	10.53	64	8.28	128	11.27	64	9.25	256	11.88	256
Linformer	7.91	64	6.25	128	8.08	64	6.90	256	6.65	256
· w/ unreduced JLT	36.87	8	21.49	32	35.93	4	15.17	128	22.03	128
Informer	33.13	16	21.89	32	36.52	16	26.14	64	24.92	128
· w/ mask	25.94	32	21.50	64	35.95	32	15.79	128	22.58	128
Skeinformer	9.60	64	9.66	128	10.61	64	9.25	256	11.86	256
· w/o column sampling	7.60	128	6.66	256	6.70	64	7.27	512	7.76	256
· w/o row normalization	25.02	16	16.02	64	55.72	4	11.12	256	15.52	128
· w/ simple row normalization	6.80	128	8.16	256	8.03	64	6.84	512	11.27	256
· w/o pilot sampling reutilization	7.15	128	7.31	256	8.68	64	7.09	512	10.19	256

Table 2: Training time (minute per thousand steps) and actual batch size (in batch accumulation) on LRA benchmark. Less training time per thousand steps indicates higher time efficiency. Higher batch size indicates higher space efficiency, and within a certain range means more accurate gradient estimations. we simulate the case of real-world applications of efficient transformers that models are trained with their maximum batch size conditioned on memory.

motivates us to turn to some theoretical framework for matrix approximation to better analyze the fast attention methods, which will potentially benefit transformer pruning, compression and distillation.

Skeinformer has the comparable general performance in terms of time/space complexity and classification accuracy. A long transformer is considered efficient when it (1) reduces space complexity and supports larger sequence length and larger batch size, (2) reduces time complexity with less training time per step and less total time to converge, and (3) shows comparable performance with vanilla softmax without much loss from approximation.

For convergence efficiency, Skeinformer efficiently converges to the long-time limit. Regarding the training efficiency, we focus on how soon the model can attain the stationary distribution of its long-time limit (He et al., 2019). The loss decay plot on ListOps in Appendix shows significant differences in the convergence rate of each method in addition to classification accuracy.

Though our method does not always outperform others (with the fastest convergence or the highest accuracy) , but we remark that **Skeinformer attains the best accuracy-efficiency trade-off** based on experimental results. On the opposite, some model converges fast but gets stuck in a local optimum, like Linformer in some cases.

7 Conclusion

We conclude in this paper that sketching can be applied as a theoretical framework for analyzing fast attention models, through which we are able to recognize the potential improvements upon previous work. Theoretical results are provided to guarantee the high accuracy of the approximation to the original self-attention by our proposed Skeinformer. We empirically validate the contributions of the components in Skeinformer, including column sampling, adaptive row normalization and pilot sampling reutilization, with extensive comparisons with various baseline and ablation methods.

Acknowledgement

We thank the anonymous reviewers for their helpful suggestions. This research is based upon work supported by U.S. NSF grant DMS-1810831, DARPA AIDA Program No. FA8750-18-2-0014 and U.S. DARPA KAIROS Program No. FA8750-19-2-1004. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Nir Ailon and Bernard Chazelle. 2006. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 557–563. ACM.

Guangsheng Bao, Yue Zhang, Zhiyang Teng, Boxing Chen, and Weihua Luo. 2021. G-transformer for document-level machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 3442–3455. Association for Computational Linguistics.

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

Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. 2013. *Concentration Inequalities - A Nonasymptotic Theory of Independence*. Oxford University Press.

Yifan Chen and Yun Yang. 2021. Accumulations of projections - A unified framework for random sketches in kernel ridge regression. 130:2953–2961.

Yifan Chen, Qi Zeng, Heng Ji, and Yun Yang. 2021. Skyformer: Remodel self-attention with gaussian kernel and nyström method. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. 2020. Rethinking attention with performers. *CoRR*, abs/2009.14794.

Michael B. Cohen, Jelani Nelson, and David P. Woodruff. 2016. Optimal approximate matrix product in terms of stable rank. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 11:1–11:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. 2021. Attention is not all you need: pure attention loses rank doubly exponentially with depth. 139:2793–2803.

Petros Drineas, Ravi Kannan, and Michael W. Mahoney. 2006. Fast monte carlo algorithms for matrices I: approximating matrix multiplication. *SIAM J. Comput.*, 36(1):132–157.

Petros Drineas and Michael W. Mahoney. 2005. On the nyström method for approximating a gram matrix for improved kernel-based learning. *J. Mach. Learn. Res.*, 6:2153–2175.

Xinya Du and Claire Cardie. 2020. Document-level event role filler extraction using multi-granularity contextualized encoding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8010–8020. Association for Computational Linguistics.

Xinya Du, Sha Li, and Heng Ji. 2022. Dynamic global memory for document-level argument extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*.

Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. 2020. Multi-sentence argument linking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8057–8077. Association for Computational Linguistics.

Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288.

Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. 2012. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory Comput.*, 8(1):321–350.

Fengxiang He, Tongliang Liu, and Dacheng Tao. 2019. Control batch size and learning rate to generalize well: Theoretical and empirical evidence. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 1141–1150.

Luyang Huang, Shuyang Cao, Nikolaus Nova Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 1419–1436. Association for Computational Linguistics.

William B Johnson and Joram Lindenstrauss. 1984. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020a. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020b. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images.

Ping Li, Trevor Hastie, and Kenneth Ward Church. 2006. Very sparse random projections. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pages 287–296. ACM.

Sha Li, Heng Ji, and Jiawei Han. 2021. Document-level event argument extraction by conditional generation. pages 894–908.

Drew Linsley, Junkyung Kim, Vijay Veerabadran, Charles Windolf, and Thomas Serre. 2018. Learning long-range spatial dependencies with horizontal gated recurrent units. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 152–164.

Yichao Lu, Paramveer S. Dhillon, Dean P. Foster, and Lyle H. Ungar. 2013. Faster ridge regression via the subsampled randomized hadamard transform. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 369–377.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 142–150. The Association for Computer Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Andrew Mutton, Mark Dras, Stephen Wan, and Robert Dale. 2007. GLEU: automatic evaluation of sentence-level fluency. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*. The Association for Computational Linguistics.

Nikita Nangia and Samuel R. Bowman. 2018. Listops: A diagnostic dataset for latent tree learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 2-4, 2018, Student Research Workshop*, pages 92–99. Association for Computational Linguistics.

Jiezhong Qiu, Hao Ma, Omer Levy, Wen-tau Yih, Sinong Wang, and Jie Tang. 2020. Blockwise self-attention for long document understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 2555–2565. Association for Computational Linguistics.

Dragomir R. Radev, Pradeep Muthukrishnan, Vahed Qazvinian, and Amjad Abu-Jbara. 2013. The ACL

anthology network corpus. *Lang. Resour. Evaluation*, 47:919–944.

Ali Rahimi and Benjamin Recht. 2007. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 1177–1184. Curran Associates, Inc.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Trans. Assoc. Comput. Linguistics*, 9:53–68.

Amartya Sanyal, Varun Kanade, Philip HS Torr, and Puneet K Dokania. 2018. Robustness via deep low-rank representations. *arXiv preprint arXiv:1804.07090*.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2020a. Long range arena: A benchmark for efficient transformers. *CoRR*, abs/2011.04006.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020b. Efficient transformers: A survey. *CoRR*, abs/2009.06732.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Roman Vershynin. 2010. Introduction to the non-asymptotic analysis of random matrices. *CoRR*, abs/1011.3027.

Martin J Wainwright. 2019. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press.

Shuohang Wang, Luowei Zhou, Zhe Gan, Yen-Chun Chen, Yuwei Fang, Siqi Sun, Yu Cheng, and Jingjing Liu. 2020a. Cluster-former: Clustering-based sparse transformer for long-range dependency encoding. *CoRR*, abs/2009.06097.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020b. Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768.

Christopher K. I. Williams and Matthias W. Seeger. 2000. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 682–688. MIT Press.

David P. Woodruff. 2014. Sketching as a tool for numerical linear algebra. *Found. Trends Theor. Comput. Sci.*, 10(1-2):1–157.

Wen Xiao and Giuseppe Carenini. 2019. Extractive summarization of long documents by combining global and local context. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3009–3019. Association for Computational Linguistics.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. 2021. Nyströmformer: A nyström-based algorithm for approximating self-attention. *CoRR*, abs/2102.03902.

Yun Yang, Mert Pilanci, Martin J Wainwright, et al. 2017. Randomized sketches for kernels: Fast and optimal nonparametric regression. *The Annals of Statistics*, 45(3):991–1023.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2020. Informer: Beyond efficient transformer for long sequence time-series forecasting. *CoRR*, abs/2012.07436.

A Further experiment Details

A.1 Implementation Details

As it is not realistic to exhaustively fine-tune all models and search for the best performance under limited computation resources, we instead replace the self-attention module in transformer with the various drop-in attention methods and keep other experimental settings the same. Following (Xiong et al., 2021) we use a 2-layer transformer model with 64 embedding dimensions, 128 hidden dimensions, and 2 attention heads for all experiments. Mean pooling is used in all classifiers.

For comparable computation complexity, we control the number of features used in all methods, which leads to 256 as the number of features in Skeinformer, 256 as k in Linformer, 256 as the number of landmarks in Nyströmformer, $(256/\log n)$ as the factor in Informer, and 256 as the number of features in Performer. Additionally, the number of random blocks and block size in Big Bird are by default 3 and 64, under which setting Big Bird will visit $640 \cdot n$ elements in the attention matrix while other models visit $256 \cdot n$ elements. A clearer complexity evaluation on the FLOPs of each method is provided in Appendix.

We use Adam optimizer (Kingma and Ba, 2015) with a learning rate of $1e-4$. Batch size is selected conditioned on the memory requirements of Skeinformer, which leads to 128 for Text Classification, 256 for ListOps, 64 for Document Retrieval, 512 for Pathfinder and 256 for Image. For methods reporting out-of-memory errors, we apply gradient accumulation and report the accumulated steps. Instead of setting a fixed epoch number, we train all models until convergence with a stopping strategy (if better performance is not observed for 10 checking steps on the validation set we will stop the training process).

We conduct all experiments on one Tesla V100 SXM2 16GB. For numerical consistency, all experiment results are averaged across three random runs.

A.2 LRA Dataset

We evaluate on five classification tasks in LRA benchmark (Tay et al., 2020a), excluding Pathfinder-X, which fails all baseline models.

ListOps (Nangia and Bowman, 2018): This 10-label classification task requires the models to parse a sequence of length $2k$ of numbers and operators and evaluates their capacity of modeling hierarchically structured long sequences.

Text Classification on IMDb review dataset(Maas et al., 2011): This byte-Level binary classification task requires the model to analyze the sentiment of a sequence of length $4k$ by composing the unsegmented characters into higher-level meaningful units.

Document Retrieval on AAN dataset (Radev et al., 2013): This byte-Level binary classification task requires the model to compress long sequences of length $4k$ into representations for similarity score calculation in a two-tower setup without cross-attention.

Pathfinder on CIFAR-10 dataset(Linsley et al., 2018): This binary classification task requires the model to decide whether two points are connected by a dashed path on an image represented as a pixel sequence of length $4k$, and exams their capacity to capture long-range spatial dependency.

Image Classification (Krizhevsky et al., 2009): This 10-label classification task requires the models to learn the spatial relations between the flattened input pixels of length $1k$.

A.3 Validation Loss

We present the loss decay plots on all tasks in Figure 2. In the first subplot for the text classification task, we note all the methods quickly overfit the dataset. In all the other plots, our methods show the ability to both efficiently converge to the long-time limit and find better local minima with lower validation loss.

A.4 FLOPs

We conclude in this subsection the floating point operations (FLOPs) of each candidate model (excluding the ablation models). To ease the notation, given the sequence length n, we fix $p = 32, d = 256$. Assuming the matrices $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are given and omitting the non-leading term, we report the FLOPs of each model in Table 3. We additionally comment that Reformer is excluded from the table since its FLOPs are not fixed and depend on the frequency of collision after hashing of tokens, which changes with the input sequence.

A.5 Hyper-parameter Sensitivity

Figure 3 shows the accuracy and training time for Skeinformer using different batch sizes (64,128) and learning rates ($1e-3$,$1e-4$,$1e-5$) on text

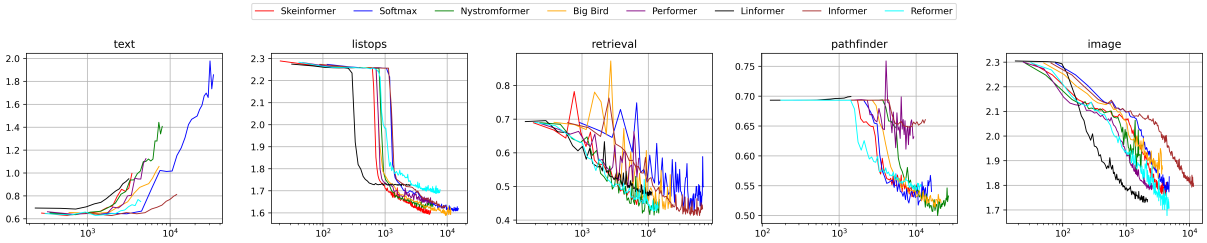

Figure 2: Validation loss (Y axis) changes with regard to training time (second, X axis).

Models	**FLOPs**
Standard	$2n^2p$
Big Bird	$5ndp$
Performer	$3ndp$
Nystromformer	$4ndp$
Linformer	$4ndp$
Informer	$3ndp$
Skeinformer	$4ndp$

Table 3: The leading terms of FLOPs in computing attention.

classification. The results are averaged across random trials. We observe that smaller learning rate offers slower convergence but to a better point.

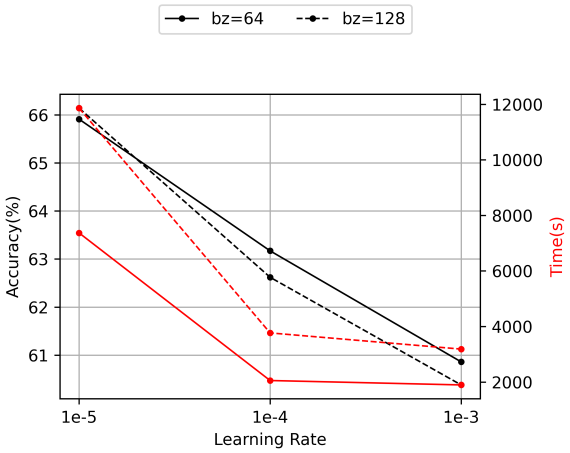

Figure 3: Testing accuracy and training time (Y axis) changes with regard to learning rate (X axis).

B Proof of Lemma 1

Proof. For each column $\boldsymbol{B}^{(i)}$, we first define a discrete random variable X_i, that with probability $\frac{1}{n}$, $X_i = b_{ji}, \forall j \in [n]$, where b_{ji} is the j-th element in $\boldsymbol{B}^{(i)}$. Since all the elements in $\boldsymbol{B}$ are bounded (within the range $[0,1]$) due to the row normalization in softmax, we infer that for any $i \in [n], X_i^2 \in [0,1]$ is a sub-Gaussian random variable with parameter $\sigma = \frac{1}{2}$ (Wainwright, 2019). Combine the conclusion with the assumption that $\mathbb{E}\, X_i^2 \leq C$, we have

$$\frac{X_i^2}{\mathbb{E}\, X_i^2} \sim \text{sub-Gaussian}\left(\sigma^2 = \frac{1}{4C^2}\right). \quad (7)$$

Then we uniformly sample d indexes $\{j_k\}_{k=1}^d$'s with replacement, and we estimate the squared norm of each column with the unbiased estimator $Y_i = \frac{n}{d}\sum_{k=1}^d b_{j_k i}^2$. We remark Y_i has the same distribution as $\frac{n}{d}\sum_{k=1}^d X_{i,(k)}^2$, where $X_{i,(k)}$'s are d i.i.d. copies of X_i. Therefore through a linear transform of Equation (7) we can derive that

$$\frac{Y_i}{n\,\mathbb{E}\, X_i^2} \sim \text{sub-Gaussian}\left(\sigma^2 = \frac{1}{4C^2 d}\right). \quad (8)$$

Notice different Y_i's may not be independent since they all rely on the same d rows in $\boldsymbol{B}$. However, we can still apply the maximal sub-Gaussian inequality (Boucheron et al., 2013) to have:

$$\mathbb{P}\left\{\max_{i\in[n]}\left|\frac{Y_i}{n\,\mathbb{E}\, X_i^2} - 1\right| > \frac{1}{2}\right\} \leq 2ne^{-\frac{C^2 d}{2}}. \quad (9)$$

If the high probability bound holds that $\max_{i\in[n]}\left|\frac{Y_i}{n\,\mathbb{E}\, X_i^2} - 1\right| \leq \frac{1}{2}$, we directly have that our estimators $Y_i \in [\frac{1}{2}\|\boldsymbol{B}^{(i)}\|^2, \frac{3}{2}\|\boldsymbol{B}^{(i)}\|^2], \forall i \in [n]$. In that case, the estimated sub-sampling probabilities satisfy that

$$\hat{p}_i = \frac{Y_i^{\frac{1}{2}}\|V_{(i)}\|}{\sum_{i'=1}^n Y_{i'}^{\frac{1}{2}}\|V_{(i')}\|} \geq \frac{\sqrt{1/2}\|\boldsymbol{B}^{(i)}\|\|V_{(i)}\|}{\sqrt{3/2}\|\boldsymbol{B}^{(i)}\|\|V_{(i')}\|} = \sqrt{\frac{1}{3}}p_i, \forall i \in [n],$$

where p_i's are the optimal probabilities defined in the main paper.

In that case, to prove the lemma it suffices to pick a big enough sub-sample size d such that the right-hand side of Inequality (9) is smaller than δ. Simply solving the inequality leads to the desired result $d \geq \frac{2}{C^2}\log(\frac{2n}{\delta})$. ♢