Scatterbrain: Unifying Sparse and Low-rank Attention Approximation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent advances in efficient Transformers have exploited either the sparsity or low-rank properties of attention matrices to reduce the computational and memory bottlenecks of modeling long sequences. However, it is still challenging to balance the trade-off between model quality and efficiency to perform a one-size-fits-all approximation for different tasks. To better understand this trade-off, we observe that sparse and low-rank approximations excel in different regimes, determined by the softmax temperature in attention, and sparse + low-rank can outperform each individually. Inspired by the classical robust-PCA algorithm for sparse and low-rank decomposition, we propose Scatterbrain, a novel way to unify sparse (via locality sensitive hashing) and low-rank (via kernel feature map) attention for accurate and efficient approximation. The estimation is unbiased with provably low error. Empirically, we show that Scatterbrain can achieve $2.1\times$ lower error compared to baselines when serving as a drop-in replacement in BigGAN image generation and pre-trained T2T-ViT. On a pre-trained T2T Vision transformer, even without fine-tuning, Scatterbrain can reduce 98% of attention memory at the cost of only 0.8% drop in accuracy. We demonstrate Scatterbrain for end-to-end training with up to 4 points better perplexity and 5 points better average accuracy compared to sparse or low-rank efficient transformers on language modeling and long-range-arena tasks.

1 Introduction

Transformer models [39] have been adapted in a wide variety of applications, including natural language processing [19, 5, 31], image processing [8, 29], and speech recognition [27]. Training large Transformers requires extensive computational and memory resources, especially when modeling long sequences, mainly due to the quadratic complexity (w.r.t. sequence length) in attention layers. Recent advances in efficient transformers [24, 12, 23, 40, 15] leverage attention approximation to overcome the bottleneck by approximating the attention matrices. However, it is still challenging to find a robust approximation method that can balance the efficiency-accuracy trade-off on a wide variety of different tasks [36, 37].

We categorize most of the existing approaches for efficient attention matrix computation into two major groups: exploiting either the sparsity, e.g. Reformer [24], SMYRF [15], or low-rank properties of the attention matrices, e.g. Linformer [40], Linear Transformer [23], and Performer [12]. However, these techniques usually have different strengths and focus on the performance for specific kinds of tasks, so their approximations still cause accuracy degradation on many other tasks. For instance, according to a recent benchmark paper [36] and our experiments, low-rank-based attention might be less effective on hierarchically structured data or language modeling tasks, while sparse-based variants do not perform well on classification tasks.

We observe that sparse and low-rank approximations are complementary for many attention matrices in practice, and sparse+low-rank could outperform each individually (Figure 1 left). We empirically categorize the regimes in which sparse or low-rank approximation achieves better error based on the softmax

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

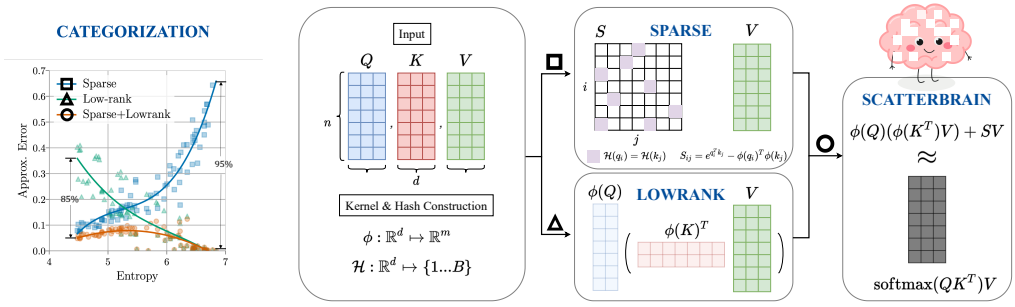

Figure 1: Left: regimes that sparse+low-rank approximation is more accurate, based on the entropy of the attention matrices. Right: Scatterbrain Workflow - For the attention layer in Transformers, after computing Query Q, Key K, and Value V matrices, we approximate $\mathrm{softmax}(QK^\top)V$ with two components: (i) sparse SV (ii) low-rank $\phi(Q)(\phi(K)^\top V)$.

temperature of attention (of which the entropy of softmax distribution can be used as a proxy). We expect that if the attention depends on a few entries (low entropy softmax), sparse methods perform well, while if the attention depends on a mixture of many components (high entropy softmax), low-rank methods do better. This explains the phenomenon that current sparse and low-rank-based approaches excel on different kinds of tasks. A natural question is whether one could unify the strength of both approaches. While it is NP-hard to find the optimal combination of sparse and low-rank approximations, robust PCA [7] is a polynomial-time solution with tight approximation error. We observe that robust PCA achieves lower approximation error than sparse or low-rank alone on attention matrices. The difference is most pronounced for "mid-range" entropy, where we observe that up to 95% error reduction is possible.

The connection between robust PCA and attention matrix estimation provides an opportunity to realize a more robust approximation. Specifically, given an attention matrix, one could adaptively perform sparse+low-rank approximation to obtain a low error. However, it comes with three challenges: (i) How to decompose the attention matrices into sparse and low-rank components and estimate them efficiently and accurately; robust PCA is accurate but slow and requires materializing the full attention, while straightforward addition of sparse and low-rank attention will be inaccurate due to double counting. (ii) It is not clear if there is a theoretical guarantee that sparse + low-rank can improve the approximation in some regimes, though we observe the separation empirically. (iii) How does the lower approximation error transfer to end-to-end performance in real tasks.

In this paper, we propose Scatterbrain, an accurate and efficient robust estimation of attention matrices with theoretical guarantees to address the above challenges. Specifically:

- In Section 3, we observe that sparse and low-rank approximation are complementary and demonstrate that sparse + low-rank structure arises naturally when elements in the input sequence form clusters. We theoretically characterize and analyze the regimes where sparse, low-rank and sparse+low-rank excel, dictated by the softmax temperature of attention.
- In Section 4, inspired by the classical robust-PCA algorithm, we propose Scatterbrain, which efficiently combines sparse and low-rank matrices to approximate attention. In particular, we use Locality Sensitive Hashing (LSH) to identify large entries of the attention matrix (after softmax) without materializing the full matrix and then leverage kernel approximation to parameterize the low-rank part. We prove that our method has a strictly lower approximation error than the low-rank baseline.
- In Section 5, we empirically validate our theory and the proposed method, showing that Scatterbrain accurately approximates the attention matrix, is memory efficient for long sequences, and works well across different tasks. First, we show that its approximation accuracy is close to our oracle robust-PCA and achieves $2.1\times$ lower error compared to other efficient baselines on real benchmarks. This leads to a direct application of Scatterbrain as a drop-in replacement to pre-trained full attention, thus reducing up to 98% of the memory required for attention computations in pre-trained T2T-ViT and BigGAN while maintaining similar quality. Last we show that its superior accuracy and efficiency can improve the efficiency-accuracy trade-offs of Transformer end-to-end training. On the WikiText-103 language modeling task, Scatterbrain achieves up to 1 point better perplexity compared to Reformer and Performer. On 5 benchmark long-range tasks, Scatterbrain improves the average accuracy by up to 5 points.

2 Problem Setting and Related Work

We first define the approximation problem we aim to solve in this paper. Then we discuss the applications of sparse and low-rank techniques in efficient Transformers and introduce robust-PCA algorithm.

Problem Formulation: In the attention matrix approximation problem, we are given three matrices, query, key, and value, $Q, K, V \in \mathbb{R}^{n \times d}$, and the attention output is $\mathrm{softmax}(QK^\top)V$. We seek to reduce the quadratic complexity of $\mathrm{softmax}(QK^\top)$ (applied row-wise) with low approximation error. More precisely, fro an approximation procedure f, we minimize two objectives, the approximation error $\mathbf{E}\left[\|f(Q,K) - \mathrm{softmax}(QK^\top)\|_F^2\right]$, and the computation/memory cost $\mathcal{C}(f(\cdot))$.

Sparse, Low-rank Approximation for Attention Matrices and Robust PCA: Recent work exploits the sparsity patterns or find a low-rank mapping of the original attention matrices to overcome the computational and memory bottlenecks in Transformers [24, 15, 34, 12, 23, 40]. Generally, we can divide most of the techniques into two categories – sparse and low-rank approximations. Reformer [24] is a representative sparse variant which uses LSH [2] to retrieve or detect the locations of the attention matrices with large values and reduce the computation from $O(n^2)$ to $O(n \log n)$. Performer [12] is an example of the low-rank variant, which uses kernelization to avoid explicit $O(n^2 d)$ computation. One problem of either the sparse or low-rank approximation is that the structure of the attention matrices varies in practice, and it is challenging to perform robust approximation on a wide range of attention matrices. For example, Wang et al. [40] observes that attentions tend to have more low-rank structures in lower layers and Ramsauer et al. [32] shows that they are sparser in the later stage of the training. Ideally, we want to unify the strength of both techniques but it is NP-hard to find the best combination of sparse and low-rank approximation. Fortunately, classical Robust PCA [7] presents a polynomial algorithm to find the approximately optimal or good combinations of sparse and low-rank approximation of the matrices. An extended discussion of related work is in Appendix A.

3 Characterization of Sparse + Low-rank Approx. to Attention Matrices

We motivate the use of sparse + low-rank approximation of the attention matrices with the key observation that for many attention matrices sparse and low-rank approximation are complementary, and their ideal combination (via robust PCA) can outperform both (Section 3.1). Furthermore, we argue that the sparse + low-rank structure can arise naturally when elements in the input sequence form clusters, as dictated by the softmax temperature (Section 3.2).

3.1 Motivating Observations: Low-rank and Sparse Structures of Attention Matrices

We empirically characterize regimes where sparse and low-rank approximation are well-suited, based on the softmax temperature (for which we use the softmax distribution entropy is a proxy). Specifically, in Fig. 1 (left), we present the approximation error of the original attention matrices and the approximation (sparse or low-rank) of matrices sampled from a 4-layer Transformer trained on IMDb reviews classification [36]. We make two observations:

1. Sparse and low-rank approximation are complementary: sparse excels when the softmax temperature scale is low (i.e., low entropy), and low-rank excels when the softmax temperature is high (i.e., high entropy).
2. An ideal combination of sparse and low-rank (orange line in Fig. 1 left), obtained with robust PCA, can achieve lower error than both.

Similar observations on other benchmarks and details are presented in Appendix B.

3.2 A Generative Model of How Sparse + Low-rank Structure Can Arise

Sparse + low-rank parameterization is more expressive than either sparse or low-rank alone. Indeed, in the Appendix, we construct a family of attention matrices to show the separation between the approximation capability of sparse + low-rank vs sparse or low-rank alone: for an $n \times n$ attention matrix, sparse or low-rank alone requires a $O(n^2)$ parameters to get ϵ approximation error in Frobenius norm, while sparse + low-rank only requires $O(n)$ parameters.

Moreover, we argue here that sparse + low-rank is a natural candidate to approximate generic attention matrices. We describe a generative model of how the sparse + low-rank structure in attention matrices could arise, when the elements of the input sequence form clusters. Under this process, we characterize how the softmax temperature dictates when we would need sparse, low-rank, or sparse + low-rank matrices to approximate the attention matrix. This result corroborates the observation in Section 3.1.

Generative process of clustered elements in input sequence We describe here a generative model of an input sequence to attention, parameterized by the inverse temperature $\beta \in \mathbb{R}$ and the intra-cluster distance $\Delta \in \mathbb{R}$.

Process 1. *Let $Q \in \mathbb{R}^{n \times d}$, where $d \geq \Omega(\log^{3/2}(n))$, with every row of Q generated randomly as follows:*

1. *For $C = \Omega(n)$, sample C number of cluster centers $c_1, \ldots, c_C \in \mathbb{R}^d$ independently from $\mathcal{N}(0, I_d/\sqrt{d})$.*
2. *For each cluster around c_i, sample $n_i = O(1)$ number of elements around c_i, of the form $z_{ij} = c_i + r_{ij}$ for $j = 1, \ldots, n_i$ where $r_{ij} \sim \mathcal{N}(0, I_d \Delta/\sqrt{d})$. Assume that the total number of elements is $n = n_1 + \cdots + n_C$ and $\Delta \leq O(1/\log^{1/4} n)$.*

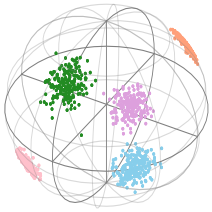

Figure 2: Visualization of the generative process. The vectors from the input sequence (rows of Q) form clusters that lie approximately on the unit sphere.

Let Q be the matrix whose rows are the vectors z_{ij} where $i = 1, \ldots, C$ and $j = 1, \ldots, n_i$. Let $A = QQ^\top$ and let the attention matrix be $M_\beta = \exp(\beta \cdot A)$.

We visualize this generative process in Fig. 2.

Softmax temperature and approx. error We characterize when to use sparse, low-rank, or sparse + low-rank to approximate the attention matrices in Process 1, depending on the inverse temperature β. The intuition here is that the inverse temperature corresponds to the strength of interaction between the clusters. If β is large, intra-cluster interaction dominates the attention matrix, the softmax distribution is peaked, and so we only need a sparse matrix to approximate the attention. If β is small, then the inter-cluster attention is similar to intra-cluster attention, the softmax distribution is diffuse, and we can approximate it with a low-rank matrix. In the middle regime of β, we need the sparse part to cover the intra-cluster attention and the low-rank part to approximate the inter-cluster attention.

We formalize this intuition in Theorem 1 (in bounds below we think of ϵ as a constant). All the proofs are in Appendix D.

Theorem 1. *Let M_β, be the attention matrix in Process 1. Fix $\epsilon \in (0,1)$. Let $R \in \mathbb{R}^{n \times n}$ be a matrix. Consider low-rank, sparse, and sparse + low-rank approximations to M_β.*

1. ***High temperature**: Assume $\beta = o(\log n / \log d)$.*
   - *(a) **Low-rank**: There exists R with $n^{o(1)}$ rank (and hence $n^{1+o(1)}$ parameters) such that $\|M_\beta - R\|_F \leq \epsilon n$.*
   - *(b) **Sparse**: If R has sparsity $o(n^2)$, then $\|M_\beta - R\|_F \geq \Omega(n)$.*
2. ***Mid temperature**: Assume $(1-\Delta^2)\log n \leq \beta \leq O(\log n)$.*
   - *(a) **Sparse + low-rank**: There exists a sparse + low-rank R with $n^{1+o(1)}$ parameters with $\|M_\beta - R\|_F \leq \epsilon n$.*
   - *(b) **Low-rank**: If R is such that $n - \mathrm{rank}(R) = \Omega(n)$, then $\|M_\beta - R\|_F \geq \Omega(n)$.*
   - *(c) **Sparse**: If R has sparsity $o(n^2)$, then $\|M_\beta - R\|_F \geq \Omega(n)$.*
3. ***Low temperature**: Assume $\beta = \Omega(\log n)$.*
   - *(a) **Low-rank**: If R is such that $n - \mathrm{rank}(R) = \Omega(n)$, then $\|M_\beta - R\|_F \geq \Omega(e^{\beta(1-\Delta^2)})$.*
   - *(b) **Sparse**: There exists R with sparsity $O(n)$ such that $\|M_\beta - R\|_F \leq \epsilon \cdot e^{\beta(1-\Delta^2)}$*

4 Scatterbrain: Unifying Sparse and Low-rank Attention

We present Scatterbrain, and show that it approximates attention accurately and efficiently. Section 4.1 describes the challenges of designing an accurate and efficient approximation, and how obvious baselines such as robust PCA or simple combination of sparse attention and low-rank attention fail to meet both criteria. Section 4.2 demonstrates how Scatterbrain address the challenges (Fig. 1 contains a schematic of Scatterbrain). In Section 4.3, we show that Scatterbrain is unbiased with provably lower variance than low-rank baselines such as Performer.

Fig. 3 shows a qualitative comparison between different methods of approximating the attention matrix: robust PCA is accurate but slow, sparse (e.g., Reformer) and low-rank (e.g., Performer) attention are fast and memory efficient but may not be very accurate, while Scatterbrain is more accurate than its sparse and low-rank counterparts while remaining just as efficient.

More details about efficient implementation of Scatterbrain are in Appendix C.

4.1 Challenges of Designing an Accurate and Efficient Sparse + Low-rank Approximation

We seek a sparse + low-rank approximation of the attention matrix[1] A that is *both* accurate and efficient. The natural theoretical baseline of robust PCA is too slow and requires too much memory, while the most straightforward way of combining sparse attention and low-rank attention fails due to double counting on the support of the sparse attention.

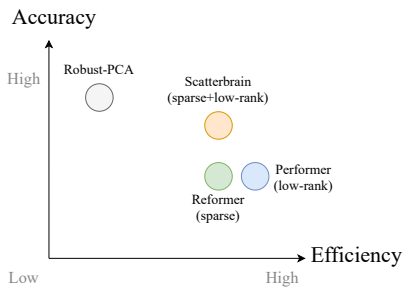

Figure 3: Qualitative comparison, in terms of approx. accuracy and efficiency, between robust PCA, sparse (Reformer) and low-rank (Performer) attention, and Scatterbrain. Scatterbrain is more accurate than sparse and low-rank attention while being just as efficient.

1. If the goal is accuracy, robust PCA is the most studied algorithm to find a sparse + low-rank approximation to a given matrix. It relaxes the NP-hard problem of finding the best sparse + low-rank approximation into a convex optimization problem, with nuclear norm and ℓ_1 constraints. Even though it can be solved in polynomial time, it is orders of magnitude too slow to be used in each iteration of a training loop. Moreover, it requires materializing the attention matrix, which defeats the main purpose of reducing compute and memory requirement.
2. On the other hand, one efficient way to get sparse + low-rank approximation of an attention matrix is to simply add the entries of a sparse approximation S (say, from Reformer) and a low-rank approximation $\widetilde{Q}\widetilde{K}^\top$ for $\widetilde{Q},\widetilde{K}\in\mathbb{R}^{n\times m}$ (say, from Performer). The sparse matrix S typically has support determined randomly [11], by LSH [24, 15], or by clustering [34]. On the support of S, which likely includes the locations of the large entries of the attention matrix A, the entries of S match those of A. One can multiply $(S+\widetilde{Q}\widetilde{K}^\top)V = SV+\widetilde{Q}(\widetilde{K}^\top V)$ efficiently because S is sparse, and grouping $\widetilde{Q}(\widetilde{K}^\top V)$ reduces the matrix multiplication complexity when $m\ll n$, from $O(n^2m)$ to $O(nmd)$. The approximation $S+\widetilde{Q}\widetilde{K}^\top$ matches $\widetilde{Q}\widetilde{K}^\top$ outside $\mathrm{supp}(S)$, hence it could be accurate there if $\widetilde{Q}\widetilde{K}^\top$ is accurate. However, $S+\widetilde{Q}\widetilde{K}^\top$ will not be accurate on the support of S due to the contributions from both S and from $\widetilde{Q}\widetilde{K}^\top$. Adjusting $\widetilde{Q}\widetilde{K}^\top$ to discount the contribution from S is difficult, especially if we want to avoid materializing $\widetilde{Q}\widetilde{K}^\top$ for efficiency.

4.2 Scatterbrain: Algorithm Intuition and Description

The simple insight behind our method is that on the support of the sparse matrix S, instead of trying to match the entries of the attention matrix A, we can set the entries of S to discount the contribution from the low-rank part $\widetilde{Q}\widetilde{K}^\top$. This way, the approximation $S+\widetilde{Q}\widetilde{K}^\top$ will match A exactly on the support of S, and will match $\widetilde{Q}\widetilde{K}^\top$ outside $\mathrm{supp}(S)$, which means it will still be accurate there if $\widetilde{Q}\widetilde{K}^\top$ is accurate. We do not need to materialize the full matrix $\widetilde{Q}\widetilde{K}^\top$ as need a subset of its entries is required, hence our approximation will be compute and memory efficient.

Scatterbrain thus proceeds in three steps: we construct a low-rank approximation $\widetilde{Q}\widetilde{K}^\top\approx A$, and construct a sparse matrix S such that $S+\widetilde{Q}\widetilde{K}^\top$ matches A on the support of S, then finally multiply SV and $\widetilde{Q}(\widetilde{K}^\top V)$ and combine the result. More specifically:

1. **Low-rank Approximation**. We define a procedure LowRank that returns two matrices $\widetilde{Q},\widetilde{K}\in\mathbb{R}^{n\times m}$ such that $\widetilde{Q}\widetilde{K}^\top$ approximates A. In particular, we use a randomized kernel feature map $\phi\colon\mathbb{R}^d\to\mathbb{R}^m$ where $\phi(x)=\frac{1}{\sqrt{m}}\exp(Wx-\|x\|^2/2)$ with $W\in\mathbb{R}^{m\times d}$ randomly sampled, entry-wise, from the standard normal distribution $\mathcal{N}(0,1)$. We apply ϕ to each row vector of Q,K matrices, and denote $\widetilde{Q}=\phi(Q)$ and $\widetilde{K}=\phi(K)$ (row-wise). Note that we do not materialize $\widetilde{Q}\widetilde{K}^\top$.
2. **Sparse Approximation**. We define a procedure Sparse that returns a sparse matrix S that matches $A-\widetilde{Q}\widetilde{K}^\top$ on $\mathrm{supp}(S)$. In particular, using a family of locality sensitive hash functions, compute the hash codes of each query and key vectors in Q,K matrices (row-wise). Let $\mathcal{S}$ be the set of locations (i,j) where q_i and k_j have the same hash codes (i.e, fall into the same hash bucket). Let S

[1] For simplicity of discussion, we consider the unnormalized attention matrix $A=\exp(QK^\top)$, omitting the usual scaling of $\sqrt{d}$ and the softmax normalization constant.

be the sparse matrix whose support is $\mathcal{S}$, and for each $(i,j)\in\mathcal{S}$, define

$$S_{i,j}=\exp(q_i^\top k_j)-\phi(q_i)^\top\phi(k_j)=\exp(q_i^\top k_j)-\widetilde{q}_i^\top\widetilde{k}_j, \tag{1}$$

where $q_i,k_j,\widetilde{q}_i,\widetilde{k}_j$ are the i-th and j-th rows of $Q,K,\widetilde{Q},\widetilde{K}$ respectively. Note that we do not materialize $\widetilde{Q}\widetilde{K}^\top$.

3. **Scatterbrain Approximation**. With $\widetilde{Q},\widetilde{K}$ returned from LowRank and S from Sparse, we compute the (unnormalized) attention output with

$$\widetilde{O}=(\widetilde{Q}\widetilde{K}^\top+S)V=\widetilde{Q}(\widetilde{K}^\top V)+SV. \tag{2}$$

The precise algorithm, including the normalization step, as well as the causal/unidirectional variant, is described in Appendix C. We also note Scatterbrain's flexibility: it can use different kinds of low-rank and sparse approximation as its sub-components.

4.3 Scatterbrain: Analysis

Our method combines a low-rank approximation $\widetilde{Q}\widetilde{K}^\top$ (which has rank $m\ll n$) with a sparse approximation S. We argue that it is accurate (lower approximation error than baselines) and efficient (scaling the same as sparse or low-rank alone). The main insight of the analysis is that our approximation is exact for entries on the support of S (picked by LSH), which are likely to be large. For entries not in the support of S (likely to be small), our approximation matches the low-rank part (Performer) $\widetilde{Q}\widetilde{K}^\top$, which is unbiased and has low variance for these entries. As a result, Scatterbrain retains the unbiasedness of Performer [12] but with strictly lower variance.

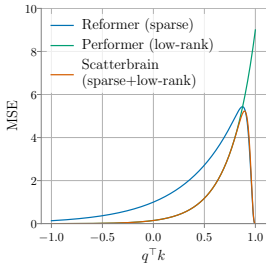

Figure 4: Per-entry MSE for different approximations, across a range of magnitude of $q^\top k$.Scatterbrain has low MSE for both small and large entries, thus outperforming its sparse (Reformer) and low-rank (Performer) counterparts.

We compare Scatterbrain to its low-rank baseline (Performer) and sparse baseline (Reformer). Performer is also based on the kernel approximation ϕ, and simply uses $\widetilde{Q}\widetilde{K}^\top$ to approximate the attention matrix A. Reformer uses LSH to identify large entries of A, then compute a sparse matrix S such that $S_{ij}=\exp(q_i^\top k_j)$ for $ij\in\mathrm{supp}(S)$.

Accuracy: Because of the way S is defined in Eq. (1), $\widetilde{Q}\widetilde{K}^\top+S$ matches $A=\exp(QK^\top)$ exactly on locations $(i,j)\in\mathcal{S}$, which are locations with likely large values. This addresses a weakness of low-rank methods (e.g., Performer) where the low-rank estimate is not accurate for locations with large values. We analyze the expectation and variance per entry of our estimator below (proof in Appendix D).

Theorem 2. *Define $\sigma(q,k)=\exp(q^\top k)$, $\widehat{\sigma}^{\mathsf{pfe}}$ as Performer's estimator and $\widehat{\sigma}^{\mathsf{sbe}}$ as Scatterbrain estimator. Denote $\mathcal{S}^{d-1}\subset\mathbb{R}^d$ as the unit sphere. Suppose $q,k\in S^{d-1}$ are such that $\|q-k\|<\tau$. Then:*

$$\mathbb{E}[\widehat{\sigma}^{\mathsf{sbe}}(q,k)]=\sigma(q,k),\quad \mathrm{Var}[\widehat{\sigma}^{\mathsf{sbe}}(q,k)]=(1-p)\cdot\mathrm{Var}[\widehat{\sigma}^{\mathsf{pfe}}(q,k)]<\mathrm{Var}[\widehat{\sigma}^{\mathsf{pfe}}(q,k)] \tag{3}$$

where $p=\exp(-\frac{\tau^2}{4-\tau^2}\ln d-O_\tau(\ln\ln d))$.

Hence Scatterbrain is unbiased, similar to Performer [12], but with strictly lower variance. The variance is small if $\exp(q^\top k)$ is small (since $\mathrm{Var}(\widehat{\sigma}^{\mathsf{pfe}}(q,k))$ will be small), or if $\exp(q^\top k)$ is large (since the probability of not being selected by LSH, $1-p$, will be small). In Fig. 4, we plot the per-entry MSE of different methods from Theorem 2 when approximating the unnormalized softmax attention $\exp(QK^\top)$. Scatterbrain can approximate well both small entries (similar to the low-rank baseline, Performer), as well as large entries (similar to the sparse baseline, Reformer). Thus Scatterbrain has much lower MSE than Performer for large entries, and much lower MSE than Reformer for small entries.

Efficiency: In Eq. (2), the computation SV is efficient because S is sparse, and $\widetilde{Q}(\widetilde{K}^\top V)$ is efficient because of the way we associate matrix multiplication (scaling as $O(nmd)$ instead of $O(n^2d)$, which is much bigger if $m\ll n$).

We validate these two properties of our approach in Section 5.

5 Experiments

In this section, we validate three claims that suggest Scatterbrain provides an accurate and efficient approximation to attention matrices, allowing it to outperform its sparse and low-rank baselines on various benchmark datasets.

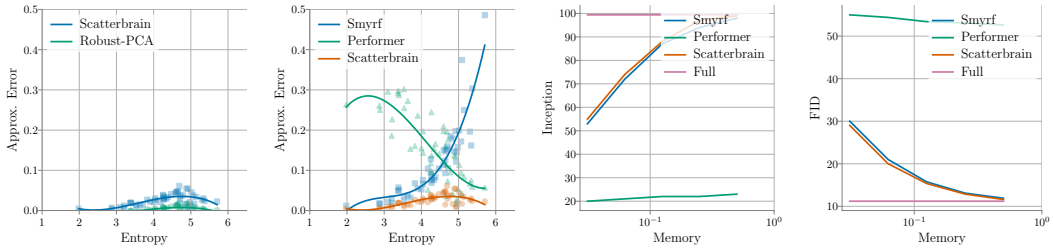

Figure 5: First plot: approximation comparison between Scatterbrain and its "lowerbound" robust-PCA. Second: comparison of error vs. entropy among SMYRF, Performer and Scatterbrain, three representatives of sparse, low-rank and sparse+low-rank approximations. Third and forth: Inception score (higher is better) and FID score (lower is better) of different attention variants for pretrained BigGAN.

- In Section 5.1, we evaluate the approximation error and testing accuracy of different approximation methods on pre-trained models such as BigGAN and Vision Transformer. We show that the approximation by Scatterbrain is close to the robust-PCA oracle and up to $2.1\times$ lower approximation error than other efficient baselines by .
- In Section 5.2, we validate that when trained end-to-end, Scatterbrain outperforms baselines (sparse or low-rank attention) on a wide variety of benchmark tasks, including language modeling, classification, and the Long-range Arena (LRA) benchmarks. It achieves up to 5 points higher average accuracy on the LRA benchmark compared to Performer and Reformer.
- In Section 5.3, we demonstrate the scalability of Scatterbrain, showing that it has comparable memory and time usage with simpler baselines (sparse or low-rank alone) across a range of input sequence length (Section 5.3), while requiring up to $12\times$ smaller memory than full attention.

All details (hyperparameters, data splits, etc.), along with additional experiments, are in Appendix E.

5.1 Scatterbrain's Approximation Accuracy

We evaluate Scatterbrain's approximation accuracy in three steps: (1) compare it with of robust-PCA (sparse+low-rank), which is our theoretical foundation and oracle (2) compare it with SMYRF[2] [15] and Performer [12], which are popular variants of sparse and low-rank approximation to attention respectively. Scatterbrain achieves error within 20% of the oracle robust PCA, and up 2x lower error than SMYRF and Performer. Hence when serving as a drop-in replacement for full attention, without training, Scatterbrain's can reduce the attention memory of Vision Transformer by 98% at the cost of only 0.8% drop of accuracy.

Table 1: Top-1 Accuracy of pre-trained T2T Vision Transformer on ImageNet with different attention replacements. Error represents the average normalized approximation error to full attention.

Attention	Top-1 Acc	Error (avg)
Full Attention	81.7%	-
SMYRF	79.8%	11.4%
Performer	80.1%	7.5%
Scatterbrain	**80.7**%	**5.3**%

Setup: We use the attention matrices from pre-trained BigGAN and T2T-ViT. BigGAN is a state-of-the-art model in Image Generation for ImageNet. BigGAN has a single attention layer at resolution 64 $\times$ 64 (4096 queries). T2T-ViT has 14 attention layers.

Scatterbrain and Robust-PCA: We first show that Scatterbrain approximates pre-trained attention matrices $10^5\times$ faster while its approximation error is within 20% on average. We also provide an example visualization on 100 attention matrices from the BigGAN generation process in Figure 5 (left).

Scatterbrain vs. SMYRF and Performer We show that Scatterbrain approximates pre-trained dense attention matrices with very low error compared to sparse (reformer) or low-rank (performer). Measuring Frobenius approximation error directly on the BigGAN image generation task, Scatterbrain achieves 2x lower error compared to Performer. When used as a drop-in replacement for full-attention during inference (with no additional training), it achieves a memory savings of $1.3\times$ compared to SMYRF (sparse baseline).

Impact and potential applications: We show that accurate approximation directly leads to efficient Inference. We replace BigGAN's dense attention with a Scatterbrain layer without other modifications. In 5 (right two), we show Inception and FID scores for Scatterbrain and other baselines under different memory budgets. Similarly, we use T2T-ViT [43], which is a token-to-token vision Transformer

[2] SMYRF is a variant of Reformer that does not require the key and query to be the same, necessary for experiments in this section.

Table 2: The performance of Scatterbrain, REFORMER, PERFORMER and Full-Attention on Long-Range-Arena benchmarks and 2 popular language modeling tasks. We fix the same number of parameters ($1/8$ of the full) used for approximating the attention matrix for each method.

Attention	Copy (ppl)	WikiText-103 (ppl)
Full Attention	1	25.258
Reformer	6.8	27.68
Performer	49	66
Scatterbrain	**2.58**	**26.72**

Attention	ListOps	Text	Retrieval	Image	Pathfinder	Avg
Full Attention	38.2	63.29	80.85	41.78	73.98	59.62
Reformer	36.85	58.12	78.36	28.3	67.95	53.9
Performer	35.75	62.36	78.83	39.71	68.6	57.05
Scatterbrain	**38.6**	**64.55**	**80.22**	**43.65**	**69.91**	**59.38**

pre-trained on ImageNet [18]. In Table 1, we show the average approximation error of Scatterbrain for each layer and the end-to-end testing accuracy after replacing full attention with Scatterbrain and other baselines. Notably, Scatterbrain achieves 80.7% Top-1 accuracy, which is only 0.8% drop from the original 81.5% by full attention reducing up to 98% of the memory usage.

5.2 End-to-end Training Performance

Scatterbrain's accurate approximation of attention matrices allows it to outperform other efficient Transformer methods on benchmark tasks. Across a range of diverse tasks, both commonly used autoregressive tasks (sequence modeling) and benchmark long-range classification tasks (Long-Range Arena), Scatterbrain outperforms Performer (low-rank baseline) and Reformer (sparse baseline) by up-to 4 points.

5.2.1 Auto-regressive Tasks

On the standard language modeling task of Wikitext-103, Scatterbrain obtains 1 point better perplexity than Reformer (sparse baseline), coming within 1.5 points of full attention.

Settings: We compare the performance of Scatterbrain against REFORMER and PERFORMER on one popular synthetic task, Copy, and one large language modeling task: WikiText103[28]. REFORMER is a representative sparse-approximation-based variant and PERFORMER is a low-rank-approximation-based variant. The base model is vanilla Transformer [39]. The statistics of each dataset and model hyper-parameters are in Appendix E. We report the best results of each method in perplexity.

Results: Table 2 shows the testing perplexity for Scatterbrain and other baselines under the same parameter budget (each approximation is only allowed to compute $\frac{1}{8}$ of the full computation). Scatterbrain achieves comparable perplexity compared to the full attention Transformer model on Copy, and WikiText-103. Notably, Scatterbrain achieves 4 point lower perplexity on Copy and 1 point lower on WikiText-103 compared to REFORMER, while PERFORMER does not train stably on auto-regressive tasks (loss does not go down).

Analysis: We also analyze the results by visualizing the error of REFORMER (sparse), PERFORMER (low-rank), and Scatterbrain (sparse + low-rank) given the same number of parameters when approximating the full attention matrices for each attention layer during training (Appendix E). The conclusion is for language modeling tasks, sparse+low-rank has the smallest approximation error in most of the cases and sparse has the largest error, which matches with the end-to-end results. It also confirms the observation in the popular benchmark paper [36] that kernel or low-rank based approximations is less effective for hierachical structured data.

5.2.2 Classification Tasks

On a suite of long-range benchmark tasks (Long Range Area), Scatterbrain outperforms Reformer (sparse baseline) and Performer (low-rank baseline) by up-to 5 points on average.

Settings: We compare the performance of Scatterbrain against REFORMER and PERFORMER on ListOps, two classification: byte-level IMDb reviews text classification, image classification on sequences of pixels, a text retrievel and pathfinder tasks. The datasets are obtained from the Long Range Arena (LRA) Benchmark [36], which is a recent popular benchmark designed for testing efficient Transformers. Similar to the auto-regressive tasks above, we use REFORMER and PERFORMER as baselines. The base model is also a vanilla Transformer. We follow the evaluation protocol from [36]. We report the best accuracy of each method.

Results: Table 2 shows the individual and average accuracy of each task for Scatterbrain and other baselines under the same parameters budget. Specially, each approximation is only allowed to use 12.5% of the full computation. We can see Scatterbrain is very close to full attention even with a large reduction in computation and memory. Further more, it outperforms all the other baselines consistently on every task and achieves more than 5 point average accuracy improvement than sparse-based approximation REFORMER and more than 2 point average accuracy improvement than low-rank-based variant PERFORMER.

Analysis: Similarly, in order to analyze the performance of REFORMER, PERFORMER and Scatterbrain, we visualize their approximation error given the same number of parameters when approximating the full attention matrices for each attention layer during training (Appendix E). We again find that Scatterbrain has the smallest approximation error, while PERFORMER is the worst on ListOps and REFORMER has the largest error on classification tasks, which matches with the end-to-end results and confirms our observations earlier (sparse and low-rank approximation excel in different regimes).

5.3 Scatterbrain's Efficiency, Scaling with Input Sequence Length

We include ablation studies on the scalability of Scatterbrain in Fig. 6, showing that it is as computation and memory-efficient as simpler baselines such as SMYRF and Performer, while up to $3\times$ faster and $12\times$ more memory efficient than full attention for sequence length 4096. This demonstrates that our combination of sparse and low-rank inherits their efficiency.

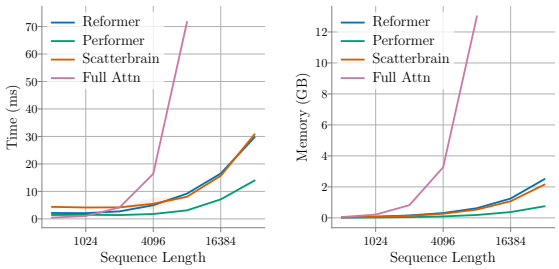

Figure 6: Speed and memory required by different efficient attention methods. Scatterbrain is competitive with SMYRF (sparse baseline) and Performer (low-rank baseline), while up to $3\times$ faster and $12\times$ more memory efficient than full attention for sequence length 4096.

We report run times and memory consumption of the sequence lengths ranging from 512 to 32768. We use a batch size of 16 for all runs and conduct experiments a V100 GPU. Since the efficiency would be largely conditioned on hardware and implementation details, we perform best-effort fair comparisons. We adapt the Pytorch implementation from `pytorch-fast-transformers` library for our baselines and implement Scatterbrain similarly without any customized cuda kernels.

6 Discussion

Limitations. As Scatterbrain has sparse attention as a component, it is not yet as hardware friendly (on GPUs and TPUs) as the low-rank component, which uses the very optimized dense matrix multiplication. This is the same limitation suffered by other sparse attention methods, but we are excited that more efficient sparse GPU kernels are being developed [21, 20].

Potential negative societal impacts. Our work seeks to understand the role of matrix approximation (and potentially energy savings) in the attention layer, which may improve a wide range of applications, each with their own potential benefits and harms. For example, making it language modeling more compute and memory efficient might facilitate spreading misinformation, and better image and video processing may make automatic surveillance easier. To mitigate these risks, one needs to address application-specific issues such as privacy and fairness, going beyond the error metrics we considered. Specially, for language models (LMs), while our work partially addresses the issue of environmental cost of LMs raised in [4], it does not address other issues such as unfathomable training data [4].

Discussion and future work. In this work, we make an observation on the sparse + low-rank structure of the attentions in Transformer models and theoretically characterize the regimes where sparse, low-rank and sparse + low-rank excel, based on the softmax temperature of the attention matrices. Motivated by this observation, we present Scatterbrain, a novel way to unify the strengths of both sparse and low-rank methods for accurate and efficient attention approximation with provable guarantees. We empirically verify the effectiveness of Scatterbrain on pretrained BigGAN, vision transformers, as well as end-to-end training of vanilla transformer. We anticipate that the study of this core approximation problem can prove useful in other contexts, such as generalized attention layers with other non-linearity beside softmax, and wide output layer in language modeling or extreme-classification.

References

[1] Keivan Alizadeh, Ali Farhadi, and Mohammad Rastegari. Butterfly transform: An efficient FFT based neural architecture design. In *The Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[2] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal lsh for angular distance. *arXiv preprint arXiv:1509.02897*, 2015.

[3] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[4] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Margaret Mitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383097. doi: 10.1145/3442188.3445922. URL https://doi.org/10.1145/3442188.3445922.

[5] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[6] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

[7] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):1–37, 2011.

[8] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[9] Beidi Chen, Tharun Medini, James Farwell, Sameh Gobriel, Charlie Tai, and Anshumali Shrivastava. Slide: In defense of smart algorithms over hardware acceleration for large-scale deep learning systems. *arXiv preprint arXiv:1903.03129*, 2019.

[10] Beidi Chen, Zichang Liu, Binghui Peng, Zhaozhuo Xu, Jonathan Lingjie Li, Tri Dao, Zhao Song, Anshumali Shrivastava, and Christopher Ré. Mongoose: A learnable lsh framework for efficient neural network training. In *The International Conference on Learning Representations (ICLR)*, 2021.

[11] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[12] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[13] Tri Dao, Albert Gu, Matthew Eichhorn, Atri Rudra, and Christopher Ré. Learning fast algorithms for linear transforms using butterfly factorizations. In *The International Conference on Machine Learning (ICML)*, 2019.

[14] Tri Dao, Nimit Sohoni, Albert Gu, Matthew Eichhorn, Amit Blonder, Megan Leszczynski, Atri Rudra, and Christopher Ré. Kaleidoscope: An efficient, learnable representation for all structured linear maps. In *The International Conference on Learning Representations (ICLR)*, 2020.

[15] Giannis Daras, Nikita Kitaev, Augustus Odena, and Alexandros G Dimakis. Smyrf: Efficient attention using asymmetric clustering. *arXiv preprint arXiv:2010.05315*, 2020.

[16] Christopher De Sa, Christopher Re, and Kunle Olukotun. Global convergence of stochastic gradient descent for some non-convex matrix problems. In *International Conference on Machine Learning*, pages 2332–2341. PMLR, 2015.

[17] Christopher De Sa, Albert Gu, Rohan Puttagunta, Christopher Ré, and Atri Rudra. A two-pronged progress in structured dense matrix vector multiplication. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1060–1079. SIAM, 2018.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[20] Trevor Gale, Matei Zaharia, Cliff Young, and Erich Elsen. Sparse GPU kernels for deep learning. In *Supercomputing*, 2020.

[21] Scott Gray, Alec Radford, and Diederik P Kingma. GPU kernels for block-sparse weights. *arXiv preprint arXiv:1711.09224*, 3, 2017.

[22] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. In *Advances in neural information processing systems (NeurIPS)*, 2020.

[23] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

[24] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *The International Conference on Machine Learning (ICML)*, 2020.

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

[26] Valerii Likhosherstov, Krzysztof Choromanski, Jared Davis, Xingyou Song, and Adrian Weller. Sub-linear memory: How to make performers slim. *arXiv preprint arXiv:2012.11346*, 2020.

[27] Haoneng Luo, Shiliang Zhang, Ming Lei, and Lei Xie. Simplified self-attention for transformer-based end-to-end speech recognition. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 75–81. IEEE, 2021.

[28] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[29] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018.

[30] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. In *The International Conference on Learning Representations (ICLR)*, 2020.

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[32] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.

[33] Benjamin Recht. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12(12), 2011.

[34] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

[35] Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive attention span in transformers. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2019.

[36] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

[37] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020.

[38] Anna Thomas, Albert Gu, Tri Dao, Atri Rudra, and Christopher Ré. Learning compressed transforms with low displacement rank. In *Advances in neural information processing systems (NeurIPS)*, pages 9052–9060, 2018.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[40] Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[41] Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In *The International Conference on Learning Representations (ICLR)*, 2019.

[42] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nystromformer: A Nystrom-based algorithm for approximating self-attention. *arXiv preprint arXiv:2102.03902*, 2021.

[43] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]