Scatterbrain: Unifying Sparse and Low-rank Attention Approximation

Beidi Chen[*†], **Tri Dao**[*†], **Eric Winsor** [†], **Zhao Song** [§], **Atri Rudra** [‡], **Christopher Ré** [†]
[†] Department of Computer Science, Stanford University
[§] Adobe Research
[‡] Department of Computer Science and Engineering, University at Buffalo, SUNY
{beidic,trid,winsor,chrismre}@stanford.edu, zsong@adobe.com, atri@buffalo.edu

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. We empirically show that Scatterbrain can achieve $2.1\times$ lower error than 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 1% drop in accuracy. We demonstrate Scatterbrain for end-to-end training with up to 4 points better perplexity and 5 points better average accuracy than sparse or low-rank efficient transformers on language modeling and long-range-arena tasks.

1 Introduction

Transformer models [64] have been adapted in a wide variety of applications, including natural language processing [26, 7, 51], image processing [10, 48], and speech recognition [43]. 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 [36, 17, 35, 66, 22] leverage attention approximation to overcome the bottleneck by approximating the attention matrices. However, it is challenging to find a robust approximation method that balances the efficiency-accuracy trade-off on a wide variety of tasks [58, 59].

We categorize most of the existing approaches for efficient attention matrix computation into two major groups: exploiting either the sparsity, e.g., Reformer [36], SMYRF [22], or low-rank properties of the attention matrices, e.g., Linformer [66], Linear Transformer [35], and Performer [17]. However, these techniques usually have different strengths and focus on the performance of specific tasks, so their approximations still cause accuracy degradation on many other tasks. For instance, according to a recent benchmark paper [58] 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.

[*]Equal contribution. Order determined by coin flip.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

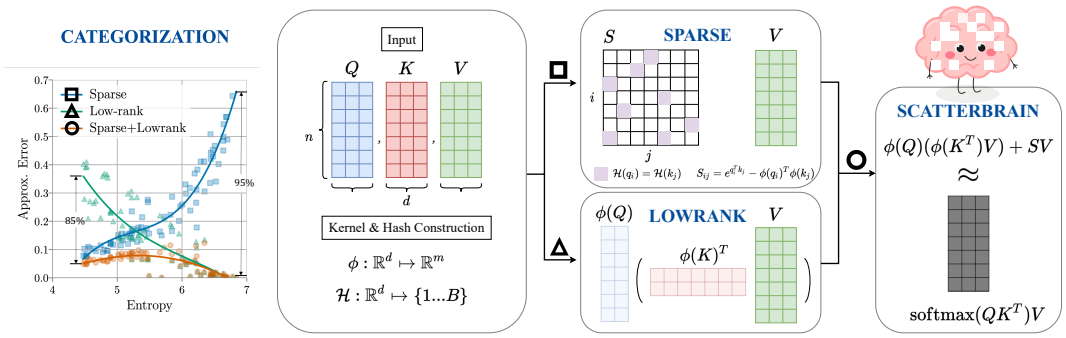

Figure 1: Left: regimes that sparse+low-rank approximation is more accurate, based on the entropy of the attention matrices. Right: Scatterbrain Workflow. For the attention layer in Transformers, after computing Query Q, Key K, and Value V matrices, we approximate $\mathrm{softmax}(QK^\top)V$ with two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

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 temperature of attention (of which the entropy of softmax distribution can be used as a proxy). We expect that sparse methods perform well if the attention depends on a few entries (low entropy softmax). In contrast, low-rank methods do better if the attention depends on a mixture of many components (high entropy softmax). 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 understand and unify the strength of both approaches. While it is NP-hard to find the optimal combination of sparse and low-rank approximations, Robust PCA [9] 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 approximation is strictly better than sparse or low-rank 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]

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, $Q, K, V \in \mathbb{R}^{n \times d}$ to compute $\mathrm{softmax}(QK^\top)V$. We seek to reduce the quadratic complexity of $\mathrm{softmax}(QK^\top)$ (applied row-wise) with low approximation error. More precisely, for 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: Recent work exploits the sparsity patterns or finds a low-rank mapping of the original attention matrices to overcome the computational and memory bottlenecks in Transformers [36, 22, 54, 17, 35, 66]. Generally, we can divide most of the techniques into two categories – sparse and low-rank approximations. Reformer [36] is a representative sparse variant that uses LSH [3] 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 [17] 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. [66] observes that attentions tend to have more low-rank structures in lower layers and Ramsauer et al. [52] 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.

Sparse + Low-rank and Robust PCA: Fortunately, classical Robust PCA [9] presents a polynomial algorithm to find the approximately optimal or good combinations of sparse and low-rank approximation of the matrices. The sparse + low-rank matrix structure has been well studied in statistics and signal processing since the late 2000s [9]. This structure naturally generalizes low-rank [33, 63], and sparse [61] matrices. Scatterbrain is built on a line of work, e.g., Bigbird [71], Longformer [5] with the theme of combining multiple types of attention and another one in the optimal transport setting [37]. However, despite the multitude of papers, this sparse + low-rank matrix approximation has not been rigorously studied in the context of attention matrices. We undertake this study and show how we can relax the sparse + low-rank approximation from robust PCA, making it efficient while still retaining PCA's accuracy. In fact, our results shed further light on why Bigbird or Longformer work, as they are special cases of a single principled structure. 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 [58]. 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] Scatterbrain code is available at `https://github.com/HazyResearch/scatterbrain`

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.

Figure 2: Visualization of the generative process, for three different values of the intra-cluster distance Δ (small, medium, and large). The vectors from the input sequence (rows of Q) form clusters that lie approximately on the unit sphere. Different colors represent different clusters.

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 $\Delta \leq O(1/\log^{1/4} n)$.*

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 a 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.

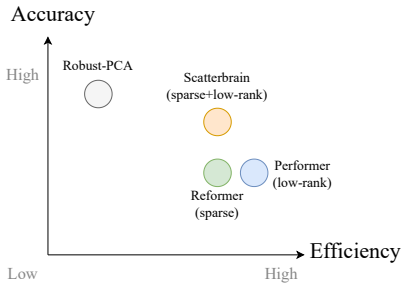

Figure 3: Qualitative comparison of approx. accuracy and efficiency, among Robust PCA, sparse (Reformer) and low-rank (Performer) attention, and Scatterbrain. Scatterbrain is more accurate while being efficient.

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 the 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[3] 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.

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 the 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 requirements.
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 [16], by LSH [36, 22], or by clustering [54]. 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 $O(n^2 m)$ 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.

[3] 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.

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 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. The combination of Reformer and Performer is simply one instance of Scatterbrain. Instead of using Reformer as a sparse component, we could use local attention [5] or random block-sparse attention [16]. Instead of using Performer [17] as a low-rank component, we could also use Linear attention [35] or global tokens as in BigBird [71].

The Scatterbrain method would work exactly the same way. As long as the low-rank component is unbiased (e.g., Performer), its combination with any sparse component in Scatterbrain would yield an unbiased estimator of the attention matrix as shown below.

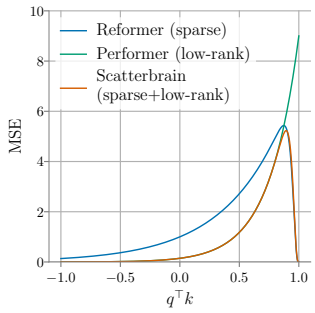

Figure 4: Per-entry MSE for different approximations, across a range of magnitude of $q^\top k$.Scatterbrain has low MSE for both small and large entries, thus outperforming its sparse (Reformer) and low-rank (Performer) counterparts.

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 [17] but with strictly lower variance.

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

Hence Scatterbrain is unbiased, similar to Performer [17], 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 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 $O(n^2 d)$, which is much bigger if $m \ll n$).

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

5 Experiments

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 benchmark datasets.

- 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.
- 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. Scatterbrain 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 lengths (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), our theoretical foundation and oracle (2) compare it with SMYRF[4] [22], Performer [17], which are popular variants of sparse and low-rank approximation to attention respectively and a naive baseline that directly adds SMYRF and Performer, (3) evaluate the inference accuracy when replacing full attention with Scatterbrain

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%
Baseline SMYRF + Performer	79.7%	12.6%
Scatterbrain	**80.7**%	**5.3**%

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

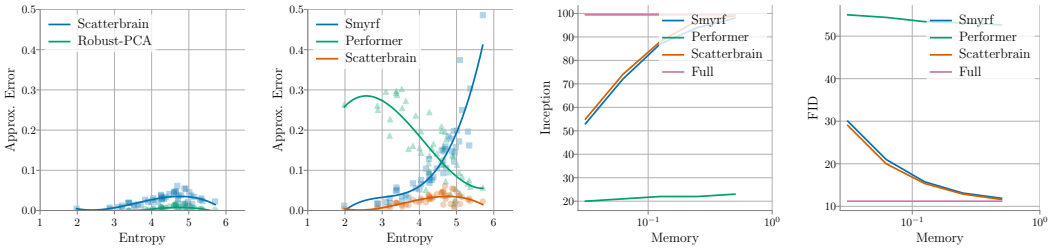

Figure 5: First: approximation comparison between Scatterbrain and its "lowerbound" Robust PCA. Second: comparison of error vs. entropy among SMYRF, Performer and Scatterbrain, three representatives of sparse, low-rank and sparse+low-rank approximations. Third and forth: Inception score (higher is better)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

approximation. Scatterbrain achieves error
within 20% of the oracle robust PCA, and up to $2.1\times$ lower error than SMYRF and Performer. When serving as a drop-in replacement for full attention, even without training, Scatterbrain can reduce the attention memory of Vision Transformer by 98% at the cost of only 0.8% drop of accuracy.

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 64×64 (4096 queries). T2T-ViT has 14 attention layers. Scatterbrain sets the ratio between SMYRF and Performer based on the entropy of an observed subset of attention matrices in different layers. We allocate more memory to the low-rank component compared to the sparse part if the entropy is high.

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 approx. error on the BigGAN image generation task, Scatterbrain achieves $2\times$ lower error compared to Performer.

Drop-in replacement for full attention: 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 [70], which is a token-to-token vision Transformer pre-trained on ImageNet [25]. 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 1% drop from the original 81.7% 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 [46]. Reformer is a representative sparse-approximation-based variant and Performer is a low-rank-approximation-based variant. The base model is vanilla Transformer [64]. We observed that generally allocating more memory budget to sparse tends to perform better, so Scatterbrain sets the ratio to 3:1 (sparse: low-rank component) for simplicity. The statistics of each dataset and model hyper-parameters are in Appendix E. We report the best results of each method in perplexity.

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**

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 points 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 [58] that kernel or low-rank based approximations are less effective for hierarchical 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 classifications: byte-level IMDb reviews text classification, image classification on sequences of pixels, a text retrieval, and pathfinder tasks. The datasets are obtained from the Long Range Arena (LRA) Benchmark [58], 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 [58]. 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

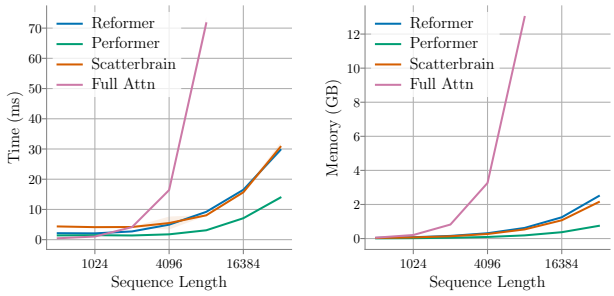

Figure 6: Speed and memory required by different efficient attention methods. Scatterbrain is competitive with SMYRF (sparse baseline) and Performer (low-rank baseline), while up to $3\times$ faster and $12\times$ more memory efficient than full attention for sequence length 4096.

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.

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 [31, 29].

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 [6], it does not address other issues such as unfathomable training data [6].

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.

Acknowledgments

We thank Xun Huang, Sarah Hooper, Albert Gu, Ananya Kumar, Sen Wu, Trenton Chang, Megan Leszczynski, and Karan Goel for their helpful discussions and feedback on early drafts of the paper.

We gratefully acknowledge the support of NIH under No. U54EB020405 (Mobilize), NSF under Nos. CCF1763315 (Beyond Sparsity), CCF1563078 (Volume to Velocity), and 1937301 (RTML); ONR under No. N000141712266 (Unifying Weak Supervision); ONR N00014-20-1-2480: Understanding and Applying Non-Euclidean Geometry in Machine Learning; N000142012275 (NEPTUNE); the Moore Foundation, NXP, Xilinx, LETI-CEA, Intel, IBM, Microsoft, NEC, Toshiba, TSMC, ARM, Hitachi, BASF, Accenture, Ericsson, Qualcomm, Analog Devices, the Okawa Foundation, American Family Insurance, Google Cloud, Salesforce, Total, the HAI-AWS Cloud Credits for Research program, the Stanford Data Science Initiative (SDSI), and members of the Stanford DAWN project: Facebook, Google, and VMWare. The Mobilize Center is a Biomedical Technology Resource Center, funded by the NIH National Institute of Biomedical Imaging and Bioengineering through Grant P41EB027060. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of NIH, ONR, or the U.S. Government. Atri Rudra's research is supported by NSF grant CCF-1763481.

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. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1225–1233. 2015.

[3] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal LSH for angular distance. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 1225–1233, 2015.

[4] R Artusi, P Verderio, and E Marubini. Bravais-pearson and spearman correlation coefficients: meaning, test of hypothesis and confidence interval. *The International journal of biological markers*, 17(2):148–151, 2002.

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

[6] 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*, New York, NY, USA, 2021.

[7] 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.

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

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

[10] 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.

[11] Beidi Chen and Anshumali Shrivastava. Densified winner take all (wta) hashing for sparse datasets. In *Uncertainty in artificial intelligence*, 2018.

[12] Beidi Chen, Anshumali Shrivastava, and Rebecca C Steorts. Unique entity estimation with application to the syrian conflict. *The Annals of Applied Statistics*, 12(2):1039–1067, 2018.

[13] Beidi Chen, Yingchen Xu, and Anshumali Shrivastava. Fast and accurate stochastic gradient estimation. 2019.

[14] Beidi Chen, Tharun Medini, James Farwell, Charlie Tai, Anshumali Shrivastava, et al. SLIDE: In defense of smart algorithms over hardware acceleration for large-scale deep learning systems. *Proceedings of Machine Learning and Systems*, 2:291–306, 2020.

[15] 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*, 2021.

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

[17] 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.

[18] Shabnam Daghaghi, Tharun Medini, Nicholas Meisburger, Beidi Chen, Mengnan Zhao, and Anshumali Shrivastava. A tale of two efficient and informative negative sampling distributions. In *International Conference on Machine Learning*, pages 2319–2329. PMLR, 2021.

[19] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] Yihe Dong, Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. Learning space partitions for nearest neighbor search. In *International Conference on Learning Representations (ICLR)*, 2019.

[28] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

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

[30] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, pages 518–529, 1999.

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

[32] 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.

[33] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[34] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

[35] 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.

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

[37] Johannes Klicpera, Marten Lienen, and Stephan Günnemann. Scalable optimal transport in high dimensions for graph distances, embedding alignment, and more. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 2021.

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

[39] 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.

[40] 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.

[41] Drew Linsley, Junkyung Kim, Vijay Veerabadran, and Thomas Serre. Learning long-range spatial dependencies with horizontal gated-recurrent units. *arXiv preprint arXiv:1805.08315*, 2018.

[42] Zichang Liu, Zhaozhuo Xu, Alan Ji, Jonathan Li, Beidi Chen, and Anshumali Shrivastava. Climbing the wol: Training for cheaper inference. *arXiv preprint arXiv:2007.01230*, 2020.

[43] 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.

[44] Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. Luna: Linear unified nested attention. *arXiv:2106.01540*, 2021.

[45] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 2011.

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

[47] Nikita Nangia and Samuel R Bowman. Listops: A diagnostic dataset for latent tree learning. *arXiv preprint arXiv:1804.06028*, 2018.

[48] 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.

[49] Dragomir R Radev, Pradeep Muthukrishnan, Vahed Qazvinian, and Amjad Abu-Jbara. The acl anthology network corpus. *Language Resources and Evaluation*, 47(4):919–944, 2013.

[50] 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.

[51] 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.

[52] 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.

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

[54] 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.

[55] Anshumali Shrivastava and Ping Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2321–2329, 2014.

[56] Vikas Sindhwani, Tara N. Sainath, and Sanjiv Kumar. Structured transforms for small-footprint deep learning. In *Advances in Neural Information Processing Systems*, pages 3088–3096, 2015.

[57] 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.

[58] 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.

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

[60] Richard Taylor. Interpretation of the correlation coefficient: a basic review. *Journal of diagnostic medical sonography*, 6(1):35–39, 1990.

[61] Reginald P Tewarson and Reginald P Tewarson. *Sparse matrices*, volume 69. Academic Press New York, 1973.

[62] 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.

[63] Madeleine Udell and Alex Townsend. Why are big data matrices approximately low rank? *SIAM Journal on Mathematics of Data Science*, 1(1):144–160, 2019.

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

[65] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

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

[67] 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.

[68] 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.

[69] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

[70] 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.

[71] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33, 2020.

[72] Chen Zhu, Wei Ping, Chaowei Xiao, Mohammad Shoeybi, Tom Goldstein, Anima Anandkumar, and Bryan Catanzaro. Long-short transformer: Efficient transformers for language and vision. *arXiv preprint arXiv:2107.02192*, 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]

A Extended Related Work

A.1 Robust PCA

Robust Principle Component Analysis (robust PCA) is the problem of finding a composition of a matrix M into a sum of sparse and low-rank components: $M = S + L$. It is a modification of PCA to accommodate corrupted observations (aka, noise). The sparse part covers the noise, while the low-rank part recovers the principle components. The most popular method to solve the problem is convex relaxation [8], where one minimizes the error $\|M - S - L\|_F^2$ subject to ℓ_1 constraint on $\|S\|_1$ and nuclear norm constraint on $\|L\|_*$, in order to promote the sparsity of S and the low-rankness of L. This convex problem can be solved with a variety of methods, such as interior point methods or the method of Augmented Lagrange Multipliers.

In our context, to find a sparse + low-rank decomposition of the attention matrix, one can also heuristically "peel off" the sparse part by finding the large entries of the attention matrix, then find a low-rank decomposition of the remainder. To avoid materializing the full attention matrix, one can use LSH to find potential locations of large entries, and use matrix completion [53] to find a low-rank decomposition. Gradient descent can find global optimum for this matrix completion problem [23]. However, it still requires too many iterations to be used in each training step.

A.2 Efficient Transformers

Sparse, Low-rank Approx.: Transformer-based model such as BERT [39] has achieved unprecedented performance in natural language processing. Recently, Vision Transformers [28, 70] has also achieved comparable performance to the traditional convolutional neural network in computer vision tasks [67]. However, the quadratic computation of the attention layers constrains the scalability of Transformers. There are many existing directions to overcome this bottleneck, including attention matrix approximation such as Reformer [36], Performer [17], leveraging a side memory module that can access multiple tokens at once [57, 40, 39] such as Longformer [5] and BigBird [71], segment-based recurrence such as Transformer-XL [19] and Compressive Transformer [50]. Please refer to a recent survey [59] for more details. In this paper, we mainly explore within the scope of approximating dense or full attention matrices.

Existing combination of Sparse and Low-rank Attention: Our focus on the classical and well-defined problem of matrix approximation, as opposed to simply designing an efficient model that performs well on downstream tasks (e.g., Longformer, Luna, Long-short transformer, etc.) affords us several advantages: (i) Easier understanding and theoretical analysis (Section 3, 4). We see that Scatterbrain yields an unbiased estimate of the attention matrix, and we can also understand how its variance changes. (ii) Clear-cut evaluation based on approximation error, as well as the ability to directly replace a full attention layer with Scatterbrain attention without re-training (Section 5). This setting is increasingly important as transformer models are getting larger and training them from scratch has become prohibitively costly. Other methods such as Luna and Long-short transformer are not backward compatible with pre-trained models.

Here we compare Scatterbrain with other work mentioned by the reviewer, showing how most of them are special cases of Scatterbrain. We will also add this discussion in the updated version of the manuscript.

- Longformer [5]: a special case of Scatterbrain where the sparse component is local attention, and the low-rank component is the global tokens. Global tokens can be considered a restricted form of low-rank approximation.
- BigBird [71]: a special case of Scatterbrain where the sparse component is local + random sparse attention, and the low-rank component is the global tokens. The use of global tokens makes the model unsuited for autoregressive modeling. On the other hand, Scatterbrain's generality allows it to use other kinds of low-rank attention (e.g., Performer), and thus Scatterbrain works on both the causal/autoregressive and the bidirectional/non-causal attention settings. BigBird's motivation is also quite different from ours: they aim to design efficient attention such that the whole Transformer model is still a universal approximator and is Turing complete. Our goal is more concrete and easier to evaluate: we approximate the attention matrices, to get a small Frobenius error between the Scatterbrain attention and the full attention matrices.
- Luna [44] (concurrent work): they use a fixed-length extra sequence and two consecutive attention steps: the context sequence attends to the extra sequence, and then the query sequence attends to the extra sequence. This is similar in spirit to low-rank attention (Linformer) and global tokens, but it is

not a low-rank approximation due to the non-linearity between the two attention steps. It is not clear to us that it combines different kinds of attention.
- Long-short transformer[72] (concurrent work): a special case of Scatterbrain where the sparse component is local attention and the low-rank component is Linformer.

A.3 Locality Sensitive Hashing for Efficient Neural Network Training

Locality Sensitive Hashing (LSH) has been well-studied in approximate nearest-neighbor search [30, 34, 55, 2, 27, 11]. Since the brute-force approach for similarity search is computationally expensive, researchers have come up with various indexing structures to expedite the search process. Usually this comes with trade-offs on the search quality. Based on these indexing structures, one can achieve sub-linear search time. LSH has been used in estimation problem as well [13, 12].

Recently, there has been several work taking advantage of LSH data structures for efficient neural network training. During training process, the weight matrices are slowly modified via gradients derived from objective functions. If we consider the weights as the search data and input as queries, we can view neural network training as a similarity search problem. For example, [14, 18, 42] proposes an algorithm which performs sparse forward and backward computations via maximum inner product search during training. It is based on the observation that the model is usually over-parameterized so the activation for a given input could be sparse and LSH is used to find or impose the sparse structure. Similarly, LSH based algorithms have also been used in Transformers [14, 15], where LSH is used to capture the sparse structure of the attention matrices. They can largely reduce the memory bottleneck of self-attention modules especially over long sequences in Transformer. Though [15] has done some exploration to improve LSH accuracy-efficiency trade-offs through learnable LSH, most of the above works have limited understanding on when and where LSH can perform well.

A.4 Structured Matrices for Efficient Machine Learning Models

Sparse + low-rank is an example of a class of structured matrices: those with asymptotically fast matrix-vector multiplication algorithm ($o(n^2)$ time complexity) and few parameters ($o(n^2)$ space complexity). Common examples include sparse, low-rank matrices, and matrices based on fast transforms (e.g., Fourier transform, circulant, Toeplitz, Legendre transform, Chebyshev transform, and more generally orthogonal polynomial transforms). These classes of matrices, and their generalization, have been used in machine learning to replace dense matrices in fully connected, convolutional, and recurrent layers [56, 62, 32]. De Sa et al. [24] shows that any structured matrix can be written as product of sparse matrices, and products of sparse matrices even with fixed sparsity pattern have been shown to be effective at parameterizing compressed models [20, 1, 21].

In our setting, it remains difficult to approximate the attention matrix with these more general classes of structured matrices. This is because many of them are fixed (e.g., Fourier transform, orthogonal polynomial transforms), and there lacks efficient algorithms to find the closest structured matrix to a given attention matrix.

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

We aim to build a deeper understanding of sparse and low-rank structures in real attention matrices: where each of them excel, and the potential for their combination. Specifically, we

- show that sparse and low-rank approximation errors are negatively correlated (through statistical tests),
- characterize regimes where each of sparse and low-rank approximation are well-suited, as dictated by the entropy of the softmax attention distribution, and
- demonstrate that sparse + low-rank has the potential to achieve better approximation than either.

B.1 Setup

Denote M as the attention matrix (after softmax) and $\mathcal{H}$ as entropy. We measure approximation error by the Frobenius norm or the original matrix and the approximation (sparse or low-rank). All the observed attention matrices in this section are from (1) a 4-layer vanilla Transformer trained from scratch on char-level IMDb reviews classification [58] (2) a 16-layer vanilla Transformer trained from scratch on WikiText103 [46] (3) a 1-layer (attention) pre-trained BigGAN on ImageNet [25]. To collect attention matrices for IMDb and WikiText103, we first save checkpoint of the models in every epoch; then evaluate 100 samples from validate data for each checkpoint and collect attention matrices from each layer each head. Note we take the median of the stats (error) for those 100 samples if it is difficult to visualize. To collect attention matrices for BigGAN, we generate 100 samples and collect the attention on the fly.

B.2 Observation 1: Sparse and low-rank approximation errors are negatively correlated

Table 3: The Spearman's rank, Pearson and Kendall's Tau correlation coefficients between Sparse and Low-rank approx. error on IMDb, WikiText-103, and BigGAN-ImageNet. P-values of < 0.05 indicate statistical significance. The two errors are negatively correlated.

	IMDb		WikiText103		BigGAN-ImageNet	
	Coef	p-value	Coef	p-value	Coef	p-value
Spearman's rank	-0.89	$<$ .00001	-0.63	$<$ .00001	-0.21	$<$ .00001
Pearson	-0.78	$<$ .00001	-0.61	$<$ .00001	-0.31	$<$ .00001
Kendall's Tau	-0.74	$<$ .00001	-0.51	$<$ .00001	-0.18	$<$ .00001

We fixed the number of parameters, K, allowed for each attention matrix approximation and collect the errors from ideal sparse and low-rank approximations: top$-K$ entries for each row of the matrix for sparse and top$-K$ eigenvalues for low-rank. Then we run three standard statistical correlation tests [4, 60], Spearman, Pearson and Kendall's Tau on sparse and low-rank approximation error for all the matrices. We can see from Table 3 that errors are significantly negatively correlated (p-value < 0.05). Further more, the left three plots on Figure 7 visualizes the correlation between the two errors on three datasets.

This negative correlation suggests that there is some property of the softmax attention distribution which determines when sparse or low-rank excels. We validate this claim in the next observation.

B.3 Observation 2: Sparse approximation error is lower when softmax entropy is low and low-rank approximation error is lower error when entropy is high

We visualize the sparse and low-rank approximation error against the entropy of attention matrices $\mathcal{H}(M)$ (applied to each row, then averaged) on the right plot in Figure 7. The attention matrices are $\in \mathbb{R}^{1024\times 1024}$ (padded) so the x-axis has range from $[0, \ln(1024)]$. For high-entropy distributions (more diffused) low-rank matrices approximates the attention matrix well. For low-entropy distributions (more peaked), sparse matrices are better-suited.

This implies that sparse and low-rank approximations could be complementary: if we can combine the strength of both, it is possible to come up with a better approximation across more general scenarios. Therefore, in the next observation, we try to combine sparse and low-rank approximations.

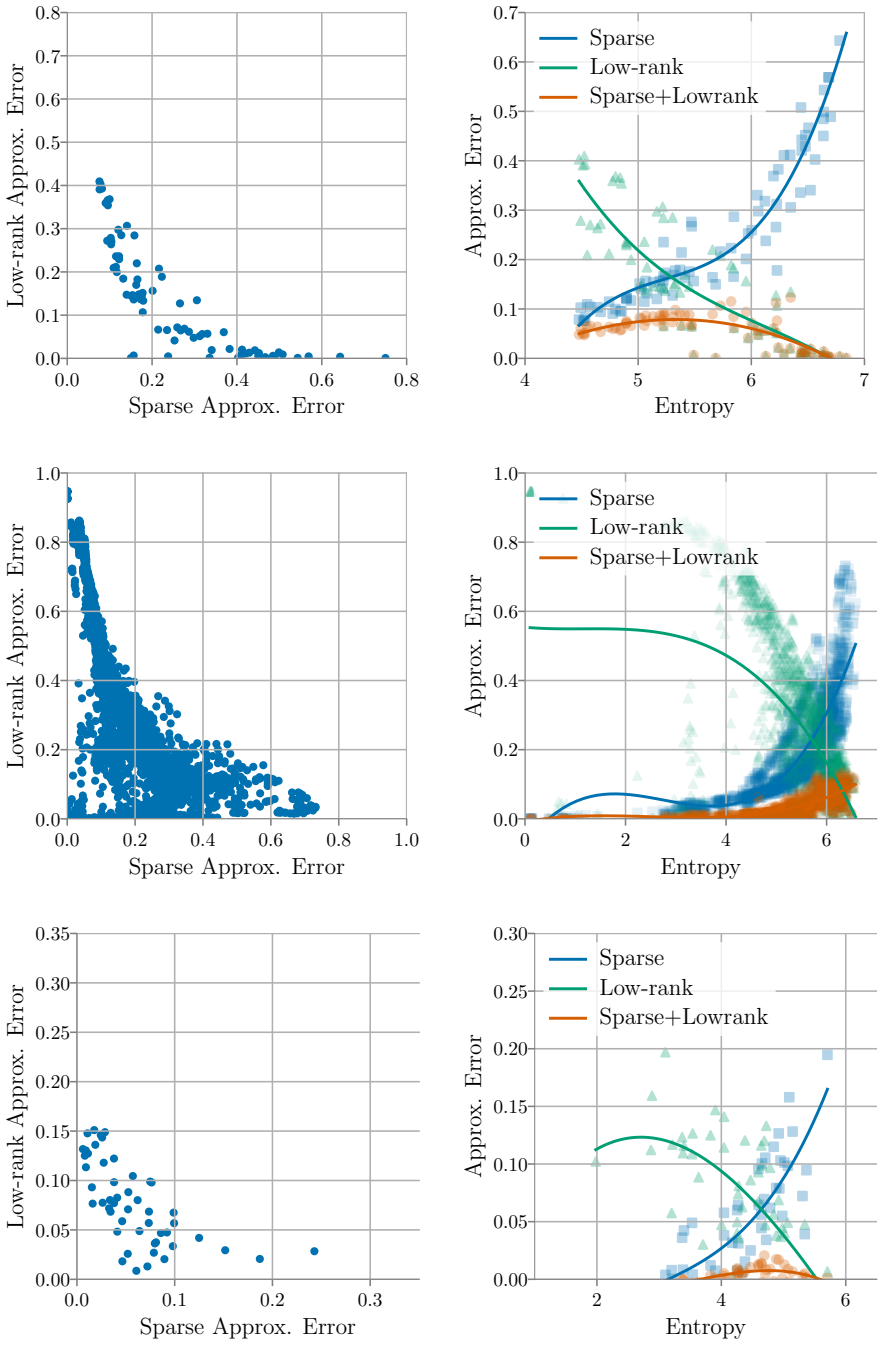

Figure 7: Characterization of the relationship between the softmax distribution of each attention matrix row and approximation error of sparse, low-rank and sparse+low-rank. The top, middle and bottom plots are for IMDb, WikiText103 and BigGAN-ImageNet respectively. Left: The approximation error of sparse and low-rank are negatively correlated. Sparse performs well when low-rank does not, and vice versa. Right: Entropy of the softmax attention distribution (i.e., scale of logits) determines the regimes where sparse, low-rank, or sparse + low-rank perform well. Sparse + low-rank yields better approximation than sparse or low-rank alone, across the board.

B.4 Observation 3: Sparse + Low-rank achieves better approximation error than sparse or low-rank alone

We find an approximation of the attention matrix of the form $S+L$, where S is sparse and L is low-rank. This problem has a rich history and is commonly solved with Robust PCA. As shown in 7, across the range of entropy, sparse + low-rank approximation can achieve lower error than either sparse or low-rank when choosing the correct mix ratio of sparse and low rank approximation ideally (with robust-PCA).

Motivated by the fact that sparse and low-rank approximations of attention matrices have complementary strengths (Observations 1 and 2), one might want to combine them (Observation 3) in hope of yielding a more robust approximation that works well across different kinds of attention matrices. The above introduces three main challenges that we have addressed in the main paper:

- how to find sparse + low-rank decomposition of an attention matrix that is compute efficient (the most studied algorithm, robust PCA, is orders of magnitude too slow to be done at each training iteration) and memory efficient (i.e., without materializing the full matrix) (Section 4),
- if we can find such a sparse + low-rank decomposition, how accurate is the approximation (Section 4.3),
- how expressive is the sparse + low-rank parameterization, i.e., are there natural classes of matrices where sparse + low-rank yields asymptotically better approximation than sparse or low-rank alone) (Section 3)?

C Scatterbrain Algorithm and Implementation Details

Let $Q,K \in \mathbb{R}^{n\times d}$ be the query and key matrices respectively, and $V \in \mathbb{R}^{n\times d}$ be the value matrix. Let the rows of Q be $q_1,...,q_n$, and the rows of K be $k_1,...,k_n$. The attention computes:

$$\mathrm{softmax}(QK^\top)V,$$

with $\mathrm{softmax}$ applied row-wise, where for each vector $v \in \mathbb{R}^n$, $\mathrm{softmax}(v) = \frac{1}{\sum_{j=1}^n e^{v_j}}[e^{v_1},...,e^{v_n}]^\top$.

Here we omit the usual scaling of $\frac{QK^\top}{\sqrt{d}}$ for simplicity since that could be folded into Q or K. Note that $\mathrm{softmax}(QK^\top) = D^{-1}\exp(QK^\top)$, where the exponential function is applied element-wise and D is a diagonal matrix containing the softmax normalization constants ($D_{i,i} = \sum_{j=1}^n \exp(q_i^\top k_j)$). Then attention has the form $D^{-1}\exp(QK^\top)V$.

We describe the Scatterbrain approximation algorithm in Algorithm 1. This includes the normalization step.

Algorithm 1 Scatterbrain Approximation of Attention

Input: $Q,K,V \in \mathbb{R}^{n\times d}$, hyper-parameters m,k,l
procedure INIT(m,k,l)
  Sample $W \in \mathbb{R}^{m\times d}$ where $W_i \sim \mathcal{N}(0,1)$ i.i.d.
  Kernels $\phi\colon \mathbb{R}^d \mapsto \mathbb{R}^m$, $\phi(x) = \frac{\exp\left(Wx - \|x\|^2/2\right)}{\sqrt{m}}$
  Hash $\forall l \in [L]$, $\mathcal{H}_l = \{h_{l,k}\}_{k\in[K]}$, $\mathcal{H} = \cup_{l\in[L]}\mathcal{H}_l$
end procedure
procedure LOWRANKAPPROX(Q,K,V,ϕ)
  $\widetilde{Q} = \phi(Q)$, $\widetilde{K} = \phi(K)$ ▷ applied to each row
  return $\widetilde{Q}(\widetilde{K}^\top V)$, $\widetilde{Q}(\widetilde{K}^\top)1_n$.
end procedure
procedure SPARSEAPPROX($Q,K,V,\phi,\mathcal{H}$)
  $\mathcal{S} = \{(i,j) | \mathcal{H}(Q_i) = \mathcal{H}(K_j)\}$
  $S \leftarrow$ sparse matrix whose support is $\mathcal{S}$
  for $(i,j) \in \mathcal{S}$ **do**
    $S_{ij} = \exp(q_i^\top k_j) - \phi(q_i)^\top \phi(k_j)$.
  end for
  return SV, $S1_n$.
end procedure
procedure SCATTERBRAINAPPROX(Q,K,V)
  $\phi, h \leftarrow$ INIT(m,k,l).
  $O_{\mathrm{lr}}, D_{\mathrm{lr}} \leftarrow$ LOWRANKAPPROX(Q,K,V,ϕ).
  $O_{\mathrm{s}}, D_{\mathrm{s}} \leftarrow$ SPARSEAPPROX(Q,K,V,ϕ,h).
  return $\mathrm{diag}(D_{\mathrm{lr}} + D_{\mathrm{s}})^{-1}(O_{\mathrm{lr}} + O_{\mathrm{s}})$.
end procedure

Autoregressive / Causal / Unidirectional Attention To approximate autoregressive attention, we simply use the autoregressive variant of low-rank attention, and apply the autoregressive mask to the sparse attention. In particular, let $M \in \mathbb{R}^{n\times n}$ be the autoregressive mask, whose lower triangle is all ones and the rest of the entries are zero. The unnormalized attention matrix is $\exp((QK^\top)\odot M)$, and the unnormalized output is $\exp((QK^\top)\odot M)V$, where $\odot$ is elementwise multiplication.

The low-rank autoregressive variant computes $((\widetilde{Q}\widetilde{K}^\top)\odot M)V$, though with a custom GPU kernel / implementation so as not to materialize the $n\times n$ matrix. For the sparse component, we simply mask out locations S_{ij} where $i > j$. That is, we can perform $S \odot M$ efficiently. As a result, we can compute the Scatterbrain output $((\widetilde{Q}\widetilde{K}^\top)\odot M)V + (S\odot M)V$ efficiently.

D Proofs

D.1 Expressiveness of Sparse + Low-rank Matrices

To motivate the use of sparse + low-rank matrices, we describe a family of attention matrices where sparse + low-rank matrices need asymptotically fewer parameters to approximate the attention matrix, compared to sparse or low-rank matrices alone. For there cases, either sparse or low-rank alone requires a quadratic number of parameters ($O(n^2)$, where $n \times n$ is the dimension of the attention matrix) to get ϵ approximation error in Frobenius norm, while sparse + low-rank only requires $O(n)$ parameters.

We construct a matrix family that shows the separation between the approximation capability of sparse + low-rank vs. sparse or low-rank alone. More specifically, we will use diagonal + low-rank (a special case of sparse + low-rank).

Example 1. *Let ϵ denote a parameter that satisfies $\epsilon \in (0,1/2]$. Consider the following randomized construction of a matrix $Q \in \mathbb{R}^{n \times d}$ with $d \geq 6\epsilon^{-2}\log n$ and $d = \Theta(\epsilon^{-2}\log n)$, where each entry of Q is picked independently and uniformly at random from $\{\pm 1/\sqrt{d}\}$. Let $M = \sigma(QQ^\top)$ where σ is the elementwise exponential function (we first ignore the normalization term of softmax here).*

It can be shown (e.g. by Hoeffding's inequality) that with high probability

$$(QQ^\top)_{i,j} = \begin{cases} 1, & \text{if } i = j; \\ \in [-\epsilon, \epsilon], & \text{otherwise.} \end{cases}$$

Since $M = \sigma(QQ^\top)$ where σ is the elementwise exponential function,

$$M_{i,j} = \begin{cases} e, & \text{if } i = j; \\ \in [1 - O(\epsilon), 1 + O(\epsilon)], & \text{otherwise.} \end{cases}$$

Intuitively, as the attention matrix M has large diagonal entries, low-rank matrices will not be able to approximate it well. However, the off-diagonals are also of reasonable size, thus making sparse approximation difficult. With sparse + low-rank, we can use the sparse part to represent the diagonal, and the low-rank part to represent the remaining elements, allowing it to approximate this matrix well. We formalize this separation in the theorem below.

Theorem 3. *Let M be the attention matrix from Example 1. For any $\gamma \in [0,1]$, with probability at least $1 - n^{-1}$, there exists a sparse + low-rank estimator with $O(\gamma^{-1} n^{3/2} \log n)$ parameters that achieve $\gamma\sqrt{n}$ Frobenius error. For any matrix $R \in \mathbb{R}^{n \times n}$ with rank such that $n - \mathrm{rank} = \Omega(n)$ (e.g., R has $o(n^2)$ parameters), with probability at least $1 - n^{-1}$, we have $\|M - R\|_F \geq \Omega(\sqrt{n})$. Moreover, any matrix E_{S} that has row sparsity k (each row has less than k non-zeros) such that $n - k = \omega(1)$ (e.g., E_{S} has $o(n^2)$ parameters) will have error $\|M - E_S\|_F \geq \Omega(\sqrt{n})$ with probability at least $1 - n^{-1}$.*

We see that for any $\gamma \in [0,1]$, any low-rank or sparse estimator for M with (n^2) parameters has $\Omega(\gamma^{-1})$ times the error of the sparse + low-rank estimator with $O(\gamma^{-1} n^{1.5} \log n)$ parameters.

Proof of Theorem 3. For each $i \in [n]$, let q_i denote the i-th row of $Q \in \mathbb{R}^{n \times d}$. Define $J \in \mathbb{R}^{n \times n}$ to be the all 1s matrix. Define $T = M - J - QQ^\top$. Therefore,

$$T_{i,j} = \begin{cases} e - 2 & \text{if } i = j \\ e^{q_i^\top q_j} - 1 - q_i^\top q_j & \text{otherwise} \end{cases}.$$

By Hoeffding's inequality, for a pair $i \neq j$, we have that

$$\mathbb{P}\left(\left|q_i^\top q_j - \mathbb{E}[q_i^\top q_j]\right| \geq \epsilon\right) \leq 2\exp\left(-\frac{2\epsilon^2}{\left(\frac{1}{\sqrt{d}} - \frac{-1}{\sqrt{d}}\right)^2}\right) = 2\exp(-d\epsilon^2/2).$$

Note that $\mathbb{E}[q_i^\top q_j] = 0$.

By a union bound over all pairs $i \neq j$ (there are $n(n-1)/2$ such pairs), with probability at least $1 - n^2\exp\left(-d\epsilon^2/2\right)$, we have that

$$q_i^\top q_j \in [-\epsilon, \epsilon] \quad \text{for all } i \neq j.$$

Since we assume that $d \geq 6\epsilon^{-2}\log n$, we have that

$$n^2\exp(-d\epsilon^2/2) \leq n^2\exp(-3\log n) = n^{-1}.$$

Hence $q_i^\top q_j \in [-\epsilon, \epsilon]$ for all $i \neq j$ with probability at least $1 - n^{-1}$. For the rest of the proof, we only consider this case (where $q_i^\top q_j \in [-\epsilon, \epsilon]$ for all $i \neq j$).

Since $1+x \le e^x \le 1+x+x^2$ for $|x|<1$, we can bound the off diagonal elements $|T_{i,j}| \le \epsilon^2$. In particular, for all $i \neq j$,

$$|T_{ij}| = \left| e^{q_i^\top q_j} - 1 - q_i^\top q_j \right| \le \left(q_i^\top q_j \right) \le \epsilon^2. \tag{4}$$

Sparse + low-rank estimator: We use the following sparse + low-rank estimator:

$$E_{\mathrm{SL}} = \underbrace{(e-2)\cdot I}_{\text{sparse}} + \underbrace{J + QQ^\top}_{\text{low-rank}},$$

where $(e-2)I$ has row sparsity 1 and $\mathrm{rank}(J+QQ^\top) \le d+1 = O\left(\epsilon^{-2}\log n\right)$.

Notice that the E_{SL} estimate matches M exactly on the diagonal, and on the off-diagonal it differs from M by T_{ij}. Thus, the Frobenious error of the sparse + low-rank estimator is

$$\|M - E_{\mathrm{SL}}\|_F \le \epsilon^2 \sqrt{n(n-1)} \le \epsilon^2 n.$$

Set $\epsilon = \frac{\sqrt{\gamma}}{n^{1/4}}$ for $0 \le \gamma \le 1$, Then

(i) The sparse + low-rank parameter count is $n + n \cdot \mathrm{rank} \le n \cdot O(\epsilon^{-2}\log n) \le O(\gamma^{-1} n^{1.5} \log n)$.

(ii) The Frobenius error is $\le \gamma\sqrt{n}$.

Low-rank estimator: We want to argue that low-rank approximation would require more parameters. If we approximate the matrix $(e-2)I$ by a matrix R with rank r, then the difference matrix will have at least $n-d$ singular values of magnitude $e-2 \ge 1/2$. As a result, by the Eckart–Young–Mirsky theorem,

$$\|(e-2)\cdot I - R\|_F \ge \frac{1}{2}\sqrt{n-r}.$$

Define $T' = T - (e-2)\cdot I$, then T' is all 0 on the diagonal and has absolute value $\le \epsilon^2$ on off-diagonal entries. Thus $\|T'\|_F \le \epsilon^2 n = \gamma\sqrt{n}$.

We want to show that if $R^{'}$ is a rank r' matrix, then $\|M - R'\|_F \ge \frac{1}{2}\sqrt{n-r'-d-1} - \|T'\|_F$. We argue by contradiction. Suppose that there exists some matrix R' with rank r' such that

$$\|M - R'\|_F \le \frac{1}{2}\sqrt{n-r'-d-1} - \|T'\|_F.$$

Define $R = R' - J - QQ^\top$, so $M - R' = (e-2)\cdot I - R + T'$. We see that:

$$\begin{aligned}
\|(e-2)\cdot I - R\|_F &= \|M - R' - T'\|_F \\
&\le \|M - R'\|_F + \|T'\|_F \\
&\le \frac{1}{2}\sqrt{n-r'-d-1} \\
&\le \frac{1}{2}\sqrt{n - \mathrm{rank}(R)}.
\end{aligned}$$

This contradicts the result above, which states that $\|(e-2)\cdot I - R\|_F \ge \frac{1}{2}\sqrt{n-\mathrm{rank}(R)}$.

Therefore any low-rank estimator with rank r such that $n - r = \Omega(n)$, which has $\Omega(n^2)$ parameters, will have error at least $\Omega(\sqrt{n-r-d-1}) - \|T'\|_F = \Omega(\sqrt{n})$, which is $\Omega(\gamma^{-1})$ times the error of the sparse + low-rank estimator above.

Sparse estimator: For our sparse estimator, it is easy to see that for any $E_{\mathrm{S}} \in \mathbb{R}^{n\times n}$ that has row sparsity k (each row has fewer than k non-zeros),

$$\|M - E_{\mathrm{S}}\|_F \ge \Omega(\sqrt{n(n-k)}).$$

This implies that in order to achieve error $O(\sqrt{n})$, we would need $n - k = O(1)$, which requires $\Omega(n^2)$ parameters.

□

Now we construct a matrix that shows better separation between the approximation capability of sparse + low-rank vs sparse or low-rank alone.

Example 2. *Consider the following randomized construction of matrix $Q \in \mathbb{R}^{n\times d}$ with $d \ge 6\epsilon^{-2} r \log n$ and $d = \Theta(\epsilon^{-2} r \log n)$ ($\epsilon \in (0,1]$ and close to 0 and r is $\Theta(\log n)$): each entry of Q is picked independently and uniformly at random from $\{\pm\sqrt{r/d}\}$. Let $M = \sigma(QQ^\top)$ where σ is the elementwise exponential function.*

Similar to Example 1, with high probability, we have:

$$(QQ^\top)_{i,j} = \begin{cases} r, & \text{if } i=j; \\ \in[-\epsilon,\epsilon], & \text{otherwise.} \end{cases}$$

We also have:

$$M_{i,j} = \begin{cases} e^r, & \text{if } i=j; \\ \in[1-O(\epsilon),1+O(\epsilon)], & \text{otherwise.} \end{cases}$$

By setting r appropriately, we can formalize the separation between the approximation ability of sparse, low-rank, and sparse + low-rank matrices:

Theorem 4. *Let M be the attention matrix from Example 2. Any sparse or low-rank estimator of M needs $\Omega(n^2)$ parameters for $O(n)$ error with probability at least $1-n^{-1}$ while a sparse + low-rank estimator needs $O(n)$ parameters for $O(n)$ error with probability at least $1-n^{-1}$.*

Proof of Theorem 4. Similar to the proof of Theorem 3, by Hoeffding's inequality, for a pair $i \neq j$, we have that

$$\mathbb{P}\left(\left|q_i^\top q_j - \mathbb{E}[q_i^\top q_j]\right| \geq \epsilon\right) \leq 2\exp\left(-\frac{2\epsilon^2}{\left(\frac{r}{\sqrt{d}} - \frac{-r}{\sqrt{d}}\right)^2}\right) = 2\exp\left(-\frac{d\epsilon^2}{2r}\right).$$

Note that $\mathbb{E}[q_i^\top q_j] = 0$. By a union bound over all pairs $i \neq j$ (there are $n(n-1)/2$ such pairs), with probability at least $1-n^{-1}$ (since $d \geq 6\epsilon^{-2} r \log n$), we have that

$$q_i^\top q_j \in [-\epsilon,\epsilon] \quad \text{for all } i \neq j.$$

Since we assume that $d \geq 6\epsilon^{-2}\log n$, we have that For the rest of the proof, we only consider this case (where $q_i^\top q_j \in [-\epsilon,\epsilon]$ for all $i \neq j$).

Let $T = M - (e^r - 1)\cdot I + J$, where J is the all one matrix. We see that T is zero on the diagonal. Moreover, using the fact that $e^x \leq 1 + 2|x|$ for all $x \in [-1,1]$, the off-diagonal entries of T have of magnitude at most 2ϵ.

We consider 3 different estimators.

Sparse + low-rank estimator: Our estimator is

$$E_{\mathrm{SL}} = \underbrace{(e^r-1)\cdot I}_{\text{sparse}} + \underbrace{J}_{\text{low-rank}},$$

where $(e-1)I$ has row sparsity 1 and $\mathrm{rank}(J) = 1$.

The Frobenious error of sparse + low-rank approximation is

$$\|M - E_{\mathrm{SL}}\|_F \leq O(\sqrt{\epsilon^2 n(n-1)}) \leq O(\epsilon n).$$

We have that:

(i) Sparse + low-rank parameter count is $n\cdot(1+1) \leq O(n)$.

(ii) Its Frobenius error is $\leq O(n)$.

Low-rank estimator: We want to argue that low-rank approximation would require more parameters. From a similar observation that any matrix R with rank that $n - \mathrm{rank} = \Omega(1)$,

$$\|(e^r-1)I - R\|_F \geq \Omega(e^r),$$

(by Eckart–Young–Mirsky theorem), we obtain a similar result to the proof of Theorem 3.

If R' is a matrix with rank such that $n - \mathrm{rank} = \Omega(1)$, then $\|M - R'\|_F \geq \Omega(n) - \|T\|_F \geq \Omega(n) - O(\epsilon n) \geq \Omega(n)$. Hence any low-rank matrix with $O(n^2)$ parameters would have error $\Omega(n)$.

Sparse estimator: Similar to the proof of Theorem 3, for our sparse estimator, it is easy to see that for any $E_{\mathrm{S}} \in \mathbb{R}^{n\times n}$ that has row sparsity k (each row has fewer than k non-zeros),

$$\|M - E_{\mathrm{S}}\|_F \geq \Omega(\sqrt{n(n-k)}).$$

This implies that to get $O(n)$ error, we would need $\Omega(n^2)$ parameters. □

D.2 Generative Model, Softmax Temperature, and Matrix Approximation

Here we show 3 cases where depending on the softmax temperature, either we'll need low-rank, low-rank + sparse, or sparse to approximate the attention matrix.

We start with some notation first. Given a matrix B, let $B[i,j]$ be the entry in the ith row and jth column. For a range $[l,r]$, we define a matrix $B_{[l,r]}$ where $B_{[l,r]}[i,j] = B[i,j]$ if $B[i,j] \in [l,r]$ and $B_{[l,r]} = 0$ otherwise (that is, $B_{[l,r]}$ only keep entries for B that are in the range $[l,r]$, with other entries zeroed out). We write $\mathrm{supp}(C)$ for the set of locations of non-zeros in C. We let $\lambda_i(D)$ be the i-th largest (in absolute value) eigenvalue of D.

To prove Theorem 1, we first define a more general matrix class, prove that the attention matrix in Process 1 is a subset of this class (with high probability), and then show that Theorem 1 holds for this more general class. We introduce an extra parameter $l \in \mathbb{R}$, in addition to the inverse temperature β and the intro-cluster distance Δ.

Matrix Class 1. *Let $Q \in \mathbb{R}^{n \times d}$ with every row of Q having ℓ_2-norm in $[1-O(\Delta), 1+O(\Delta)]$, and let $A = QQ^\top$. Further:*

1. *Let $H = A_{[1/l, 2-1/l]}$ for some $l \geq \Omega(1)$. Assume that H is block diagonal with $\Omega(n)$ blocks, and $\mathrm{supp}(H)$ is $o(n^2)$. That is, the large entries of $QQ^\top$ form a block diagonal matrix.*
2. *Let $L = A - H$ then $L = A_{[-\Delta,\Delta]}$ where $\Delta = o(1/\log d)$. Assume that there is a constant fraction of elements in $\mathrm{supp}(L)$ falling in $[0,\Delta]$. Assume that $\mathrm{supp}(A_{[0,\Delta]})$ is $\Omega(n^2)$.*

Let $M_\beta = \exp(\beta \cdot A)$.

We now show that Process 1 is a subset of Matrix Class 1, with high probability.

Lemma 5. *The matrix M_β in Process 1 is a subset of Matrix Class 1, where $l = \frac{1}{1-\Delta^2}$.*

Proof. We first bound the norm of each row in Q in Process 1. For any i,j, we have

$$\|z_{ij}\|^2 = \|c_i + r_{ij}\|^2 = \|c_i\|^2 + 2c_i^\top r_{ij} + \|r_{ij}\|^2.$$

Since $c_i \sim \mathcal{N}(0, I_d/\sqrt{d})$, $\|c_i\|^2 \in [1-\Delta^2, 1+\Delta^2]$ with probability at least $1 - 2e^{-d\Delta^2/8}$ (by the standard argument using the fact that χ^2-random variables are sub-exponential). Similarly, $\|r_{ij}\|^2 \in [\Delta^2 - \Delta^4, \Delta^2 + \Delta^4]$ with probability at least $1 - 2e^{-d\Delta^2/8}$. By concentration of measure, we can also bound $2c_i^\top r_{ij} \in [2\Delta - 2\Delta^3, 2\Delta + 2\Delta^3]$ as well. Therefore, we have that $\|z_{ij}\|^2 \in [1-O(\Delta), 1+O(\Delta)]$.

Now we show that the large entries of $QQ^\top$ form a block diagonal matrix. With high probability, the large entries come from intra-cluster dot product, and the small entries come from inter-cluster dot product.

We bound the intra-cluster dot product:

$$\begin{aligned} z_{ij}^\top z_{ik} &= (c_i + r_{ij})^\top (c_i + r_{ik}) \\ &= \|c_i\|^2 + c_i^\top r_{ij} + c_i^\top r_{ik} + r_{ij}^\top r_{ik}. \end{aligned}$$

Similar to the argument above, by concentration of measure, $\|c_i\|^2 \in [1+\epsilon\Delta, 1-\epsilon\Delta]$ with high probability (we will pick $\epsilon = \theta(\Delta)$). The cross terms $c_i^\top r_{ij}$ and $c_i^\top r_{ik}$ can be bounded using Cauchy-Schwarz inequality to be in $[-\epsilon\Delta, \epsilon\Delta]$ with high probability. And the fourth term $r_{ij}^\top r_{ik}$ is in $[-\epsilon\Delta^2, \epsilon\Delta^2]$ with high probability. Therefore, the inner product is in $1 \pm O(\epsilon\Delta)$ with high probability. This satisfies the first condition in Matrix Class 1, for $l = \frac{1}{1-\Delta^2}$, assuming $\epsilon \leq \Delta$.

We use a similar argument to bound the inter-cluster dot product. For $i \neq i'$

$$\begin{aligned} z_{ij}^\top z_{i'k} &= (c_i + r_{ij})^\top (c_{i'} + r_{i'k}) \\ &= c_i^\top c_{i'}^\top + c_i^\top r_{i'k} + c_{i'}^\top r_{ij} + r_{ij}^\top r_{i'k}. \end{aligned}$$

By concentration of measure, $c_i^\top c_{i'} \in [-\epsilon, \epsilon]$. Similar to the argument in the intra-cluster case, we can bound the other three terms, so this dot product is in $[-O(\epsilon), O(\epsilon)]$. This satisfies the second condition in Matrix Class 1.

□

To prove Theorem 1 for Matrix Class 1, we start with some technical lemmas.

Lemma 6. *Let $F \in \mathbb{R}_{\geq 0}^{N \times N}$ be a symmetric matrix. Let $\lambda_{\max}$ be the largest eigenvalue of F. Assuming $N \geq 2$, we have that*

$$\lambda_{\max} \geq \min_{i \neq j} F[i,j].$$

Proof. Since F is symmetric, $\lambda_{\max}$ is real and

$$\lambda_{\max} = \max_{u \neq 0} \frac{u^\top F u}{u^T u}.$$

Let u be the all 1's vector, then

$$\begin{aligned}
\lambda_{\max} &\geq \frac{1}{N} \sum_{i=j} F[i,j] \\
&\geq \frac{1}{N} \sum_{i \neq j} F[i,j] \\
&\geq \frac{1}{N} \cdot N(N-1) \min_{i \neq j} F[i,j] \\
&\geq \min_{i \neq j} F[i,j],
\end{aligned}$$

where the second step follows from all the diagonal entries are non-negative, the last step follows from $N \geq 2$ □

The above implies the following result:

Corollary 7. *Let $F \in \mathbb{R}_{\geq 0}^{N \times N}$ be a block diagonal matrix. Let r be the number of $m \times m$ blocks in F for some $m \geq 2$. The $\lambda_r(F)$ is at least the smallest non-diagonal element in any $m \times m$ block ($m \geq 2$) in F.*

Proof. By Lemma 6, each $m \times m$ block B ($m \geq 2$) by itself has max eigenvalue at least $\min_{i \neq j \in [m]} B[i,j]$. The claim then follows from the fact that any eigenvalue of B is also an eigenvalue of F. □

We'll need the following function for our low-rank argument:

$$f_k(x) = \sum_{i=0}^{k} \frac{x^i}{i!}.$$

Note that $f_\infty(x) = e^x$.

Definition 1. *Let $\epsilon \in (0,1/10)$ and $L > 0$. We say a function $f : \mathbb{R} \to \mathbb{R}$ is (ϵ, L)-close to e^y if*

$$|e^y - f(y)| \leq \epsilon \quad \text{for any } y \in [-L, L].$$

Lemma 8. *For any $\epsilon \in (0,1/10)$ and $L > 0$. If*

$$D \geq 10(L + \log(1/\epsilon))$$

then function $f_D(y)$ is (ϵ, L)-close to e^y.

Proof. Recall the definition of function f_D,

$$e^x = f_D(x) + \sum_{i=D+1}^{\infty} \frac{x^i}{i!},$$

It is sufficient to show that $|e^y - f(y)| < \epsilon$ if we have

$$\frac{x^{D+1}}{(D+1)!} \leq \frac{\epsilon}{2},$$

We can show that

$$\begin{aligned}
\frac{y^D}{D!} &\leq \frac{L^D}{D!} \\
&\leq \frac{L^D}{(D/4)^D} \\
&= (\frac{4L}{D})^D \\
&\leq (1/2)^D \\
&\leq \epsilon/10
\end{aligned}$$

where the first step follows from $|y| \leq L$, the second step follows $n! \geq (n/4)^n$, the forth step follows from $D \geq 10L$, the last step follows $D \geq 10\log(1/\epsilon)$ and $\epsilon \in (0,1/10)$.

□

We'll also use the following fact:

Lemma 9. *For any $D = o(\log n/\log d)$, we have*

$$\mathrm{rank}(f_D) \leq n^{o(1)}.$$

Proof. We can upper bound $\mathrm{rank}(f_D(A))$ in the following sense:

$$\begin{aligned}\mathrm{rank}(f_D(A)) &\leq (\mathrm{rank}(A))^D \\ &\leq d^D \\ &= 2^{D\cdot\log d} \\ &= 2^{o(\log n)} \\ &= n^{o(1)}.\end{aligned}$$

where the second step follows from $\mathrm{rank}(A) \leq d$, the forth step follows from $D = o(\frac{\log n}{\log d})$. □

Finally we're ready to prove the theorem:

Proof. The basic idea is: (i) Use $f_{k^*}(b\cdot A)$ to get the low-rank approximation (ii) Use $\exp(b\cdot H)$ to get the sparse part.

Small β range, i.e., β is $o\left(\frac{\log n}{\log d}\right)$.

Low rank approximation: $R = f_{k^*}(b\cdot A)$.

Since each entry of A is in $[-1,1]$, each entry of $\beta\cdot A$ is in $[-\beta,\beta]$. But note that β in this case is $o\left(\frac{\log n}{d}\right) = O(\log n\cdot\Delta)$. By the definition of k^*, each entry of $\exp(\beta\cdot A) - f_{k^*}(\beta\cdot A)$ has absolute value $\leq \epsilon$. Therefore the overall error is $\leq \epsilon n$.

For sparse only: By assumption, $m = \Omega(\|L\|_0)$ entries in A are ≥ 0, which are exactly the entries in $\exp(\beta\cdot A)$ that are ≥ 1. Hence any (say) $\frac{m}{2}$ sparse approximation has error $\geq \sqrt{\frac{m}{2}} \geq \Omega(\sqrt{\|L\|_0})$. By our assumption, $\|L\|_0 = \Omega(n^2)$.

Mid-range β, i.e., $\beta \geq \frac{1}{l}\cdot\log n$ and β is $O(\log n)$.

Sparse only: the argument is the same as in the low β range.

Sparse + low-rank: The low-rank part $R = fst(\beta\cdot A)$. By Lemma 9, this has rank $n^{o(1)}$, so it has $n^{(1+o(1))}$ parameters.

The sparse part is $S = e^{\beta\cdot H} - R_{\mathrm{supp}(H)}$. Clearly this needs $|\mathrm{supp}(H)|$ parameters.

Let $E = M_\beta - (S+R)$. Then (i) in $\mathrm{supp}(H)$, E is all 0. (ii) output of $\mathrm{supp}(H)$, by definition, entries of $\beta\cdot A$ are in $[-\beta\Delta,\beta\Delta]$, which in the current range of β is $[-O(\log n\Delta),O(\log n\Delta)]$. Therefore all the entries of E have absolute value $\leq \epsilon$. By the definition of k^*, we have that $\|E\|_F \leq \epsilon n$.

Low-rank only: Let $\widetilde{R}$ be rank $r - n^{o(1)} - 1$ that approximates M_β. Then using the same argument as our existing lower bound argument, we get that $\widetilde{R} - R \approx_E S$ (this means that the error $\leq \|E\|_F + \left\|M_\beta - \widetilde{R}\right\|_F$). Now note that $S = e^{\beta\cdot H} - (f_{k^*}(\beta\cdot A))_{\mathrm{supp}H}$ is a symmetric, block diagonal matrix with $r = \Omega(n)$ blocks. Corollary 7 implies that $\lambda_r(S)$ is at least the smallest non-diagonal value in S. Now the smallest non-diagonal value in $e^{\beta\cdot H}$ is $\geq e^{\frac{1}{l}\log n} = n$. On the other hand, the largest value in

$(f_{k^*}(\beta\cdot A))_{\mathrm{supp}H}$ is

$$\begin{aligned}
&\leq k^*\frac{\beta^{k^*}}{k^*!}\leq\beta\cdot\left(\frac{e\beta}{k^*-1}\right)^{k^*-1}\\
&\lesssim\log n\left(\frac{e\cdot\log n}{\log n\cdot\Delta}\right)^{O(\log n\cdot\Delta)}\\
&\lesssim\log n e^{O(\log n\cdot\Delta\cdot\log\frac{1}{\Delta})}\\
&\lesssim\log n\cdot n^{o(1)}\\
&=n^{o(1)}.
\end{aligned}$$

Hence $\lambda_r(S)$ is $\Omega(n)$. The claimed result then follows since $\|E\|_F\leq\epsilon n$ and $\mathrm{rank}\widetilde{R}-R\leq r-1$ (Eckart-Young-Mirsky theorem).

Large β range, i.e., $\beta\geq\omega(\log n)$.

Sparse only: $S=e^{\beta\cdot H}$. Note that each entry in $E=M_\beta-S$ is upper bounded by $e^{\Delta\cdot\beta}\leq e^{o\left(\frac{\beta}{\log d}\right)}$. Then

$$\begin{aligned}
\|E\|_F&\leq n\cdot e^{o\left(\frac{\beta}{\log d}\right)}\\
&\leq\epsilon\cdot e^{\log\frac{n}{\epsilon}+o\left(\frac{\beta}{\log d}\right)}\\
&\leq\epsilon\cdot e^{o(\beta)+o\left(\frac{\beta}{\log d}\right)}\\
&\leq\epsilon\cdot e^{o(\beta)}\\
&\leq\epsilon\cdot e^{\beta/l}.
\end{aligned}$$

Low-rank only: since $\|E\|_F$ is $\leq\epsilon e^{\beta/l}$, it is enough to argue that any rank r-approximation to S has error $\geq e^{\beta/l}$. But the latter follows since $\lambda_r(S)\geq e^{\beta/l}$. This is because $e^{b\cdot H}$ is symmetric and each entry in H is $\geq\frac{1}{\lambda}$. Then we can use Corollary 7. Eckart-Young-Mirsky then completes the proof. □

D.3 Scatterbrain: Analysis

Here we prove Theorem 2, which shows that Scatterbrain approximation is unbiased and analyses its variance. We restate the theorem here for the reader's convenience.

Theorem. *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:*

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

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

Proof. Let $A_{ij}=\exp(q_k^\top k_j)$ be ij-entry of the unnormalized attention matrix, $A^{\mathrm{lr}}_{ij}=\phi(q_i)^\top\phi(k_j)$ the entry of the low-rank approximation (Performer), and let A^{sb}_{ij} be the entry of the Scatterbrain (sparse + low-rank) approximation. By the construction of the Scatterbrain attention matrix (Eq. (1)), if $ij\in\mathcal{S}$, where $\mathcal{S}$ is the set of indices selected by the LSH, then:

$$A^{\mathrm{sb}}_{ij}=(\widetilde{Q}\widetilde{K}^\top+S)_{ij}=\phi(q_i)^\top\phi(k_j)+\exp(q_i^\top k_j)-\phi(q_i)^\top\phi(k_j)=\exp(q_i^\top k_j).$$

If $ij\notin\mathcal{S}$, then

$$A^{\mathrm{sb}}_{ij}=(\widetilde{Q}\widetilde{K}^\top+S)_{ij}=\phi(q_i)^\top\phi(k_j)+0=\phi(q_i)^\top\phi(k_j).$$

In other words, A^{sb} matches A on the indices in $\mathcal{S}$, and matches A^{lr} on the indices not in $\mathcal{S}$.

To show that A^{sb} is an unbiased estimator of A, we simply use the fact that A^{lr} is also an unbiased estimator of A [17, Lemma 1]:

$$\begin{aligned}
\mathbb{E}[A^{\mathrm{sb}}_{ij}]&=\mathbb{P}(ij\in\mathcal{S})\mathbb{E}[A_{ij}\,|\,ij\in\mathcal{S}]+\mathbb{P}(ij\notin\mathcal{S})\mathbb{E}[A^{\mathrm{lr}}_{ij}\,|\,ij\notin\mathcal{S}]\\
&=\mathbb{P}(ij\in\mathcal{S})A_{ij}+\mathbb{P}(ij\notin\mathcal{S})A_{ij}\\
&=A_{ij}.
\end{aligned}$$

In other words, $\mathbb{E}[\widehat{\sigma}^{\mathsf{sbe}}(q,k)]=\sigma(q,k)$.

Now we analyze the per-entry variance of A^{sb}. Since A^{sb} is an unbiased estimator of A, by the law of total variance,

$$\begin{aligned}\mathbb{V}\partial\!\!\!/(A^{\mathrm{sb}}_{ij}) &= \mathbb{P}(ij\in\mathcal{S})\mathbb{V}\partial\!\!\!/(A_{ij}\,|\,ij\in\mathcal{S}) + \mathbb{P}(ij\notin\mathcal{S})\mathbb{V}\partial\!\!\!/(A^{\mathrm{lr}}_{ij}\,|\,ij\notin\mathcal{S}) \\ &= \mathbb{P}(ij\in\mathcal{S})\cdot 0 + \mathbb{P}(ij\notin\mathcal{S})\mathbb{V}\partial\!\!\!/(A^{\mathrm{lr}}_{ij}) \\ &= \mathbb{P}(ij\notin\mathcal{S})\mathbb{V}\partial\!\!\!/(A^{\mathrm{lr}}_{ij}).\end{aligned}$$

To compute the probability that the index ij is not in $\mathcal{S}$ (i.e., not selected by LSH), we use the standard bound on cross-polytope LSH [3, Theorem 1]:

$$p := \mathbb{P}(ij\in\mathcal{S}) = \exp(-\frac{\tau^2}{4-\tau^2}\ln d - O_\tau(\ln\ln d)).$$

Therefore,

$$\mathbb{V}\partial\!\!\!/(A^{\mathrm{sb}}_{ij}) = (1-p)\mathbb{V}\partial\!\!\!/(A^{\mathrm{lr}}_{ij}) < \mathbb{V}\partial\!\!\!/(A^{\mathrm{lr}}_{ij}).$$

In other words, $\mathbb{V}\partial\!\!\!/[\widehat{\sigma}^{\mathsf{sbe}}(q,k)] = (1-p)\cdot\mathbb{V}\partial\!\!\!/[\widehat{\sigma}^{\mathsf{pfe}}(q,k)] < \mathbb{V}\partial\!\!\!/[\widehat{\sigma}^{\mathsf{pfe}}(q,k)]$.

More explicitly, by plugging in the variance of A^{lr} [17, Lemma 2], we have

$$\mathbb{V}\partial\!\!\!/(A^{\mathrm{sb}}_{ij}) = (1-p)\frac{1}{m}\exp\Big(\|q_i+k_j\|^2\Big)\exp(2q_i^\top k_j)\Big(1-\exp\Big(-\|q_i+k_j\|^2\Big)\Big),$$

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

□

E Additional Experiments and Details

E.1 Datasets

ImageNet [25]: ImageNet is one of the most widely-used image classification benchmarks. In our experiments in Section 5.1 of evaluating the approximation accuracy of Scatterbrain, both BigGAN and Vision Transformer are pre-trained on this dataset. It has roughly 1.2 million training images and 50,000 validation images.

WikiText103 [46] and Copy [36]: WikiText103 is a popular dataset for auto-regressive models. It is from a collection of over 100 million tokens extracted from the set of verified good and featured articles on Wikipedia. It has 28,475 training articles, 60 for validation and 60 for testing.

Copy is a synthetic a synthetic sequence duplication task where inputs are of the form $0w0w$ and $w \in \{0,...,N\}^*$. It is previously used in [36, 15]. This task is useful for demonstrating the effectiveness of long range attention: it requires non-local attention lookups. It cannot be solved by any model relying on sparse attention with a limited range such as, local attention.

Long Range Arena (LRA) [58]: This is a recent benchmark for evaluating efficient transformers with long input sequence. We used ListOps [47], byte-level IMDb reviews text classification [45], byte-level document retrieval [49], image classification on sequences of pixels [38] and Pathfinder [41]. We follow the same evaluation mechanism from [58] but implement our own version in Pytorch (like data loader).

GlUE [65]: GLUE is a standard multi-task benchmark in NLP. It has single-sentence tasks, CoLA and SST-2; similarity and paraphrase tasks, MRPC, STS-B, QQP; and inference tasks, MNLI, QNLI, RTE and WNLI. For our additional experiments below (not enough space to be included in the main paper), we follow the tradition from [26, 69, 22] and truncate all the input sequences to 128 tokens.

E.2 Settings

BigGAN: We adapt the same pre-trained BigGAN model from [22] with no additional training. The model has a single attention layer at resolution 64×64 (4096). Similar to the prior work, we also replace its full attention layer with Scatterbrain at the same resolution. Figure 5 in the main paper shows the best-effort comparison with [1/32, 1/16, 1/8, 1/4, 1/2] of the parameter budget. For example, if given parameter budget 1/2, we report the best performance of Smyrf from choice of 32/64/128 hash round 64/32/16 cluster size.

T2-ViT: We use the pre-trained vision transformer model T2T-ViT-14 from [70] with 224×224 image size. Without any additional training, we just replace the attention layer with Scatterbrain and other baselines and evaluate the approximation error and classification accuracy on ImageNet testings. Again, we report the best-effort best performance of each approximation given the certain parameter budget.

Auto-regressive Model: We follow the settings from the popular repo `https://github.com/NVIDIA/DeepLearningExamples` for training vanilla Transformer from scratch on WikiText103, except for chunking WikiText103 into sequence length 1024 in order to simulate long input sequences. The model is 16 layer with 8 head and 512 model dimension. We train all the models for 30 epochs and report the best Testing Perplexity. The model we use for Copy task is simply a 2-layer-4-head transformer and sequence length is also 1024. We make 5 runs and report average. Table 4 presents the results with standard deviation.

Classification Model: We follow the model setting from [58, 68]. We share the same finding with [68] that the acuracy for the Retrieval tasks is actually higher than reported in [58].

Ratio between Sparse and Low-rank components: There are some rules that we used in our experiments to set this ratio. For inference, we set this ratio based on the entropy of an observed subset of attention matrices in different layers: we allocate more memory to the low-rank component compared to the sparse component if the entropy is high. For training, generally allocating more memory budget to sparse tends to perform better, so in the experiment, we set the ratio to 3:1 (sparse: low-rank component) for simplicity. Moreover, in future work, it could be useful to make this ratio adaptive during training. For example, in the early stage of the training and early layers, attention matrices are usually more uniform (higher entropy). Thus, the approximation error could be even lower if the ratio favors low-rank-based components. One approach could be to monitor the approximation error of sparse and low-rank components compared to full attention regularly and adjust the memory budget accordingly. We will add the above discussion to the updated manuscript.

Table 4: 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±0.37
Reformer	6.8±0.64	27.68±0.53
Performer	49±2.7	66±5.8
Scatterbrain	**2.58**±0.21	**26.72**±0.44

Attention	ListOps	Text	Retrieval	Image	Pathfinder	Avg
Full Attention	38.2±0.17	63.29±0.38	80.85±0.12	41.78±0.26	73.98±0.31	59.62
Reformer	36.85±0.37	58.12±0.42	78.36±0.29	28.3±0.39	67.95±0.28	53.9
Performer	35.75±0.29	62.36±0.49	78.83±0.33	39.71±0.48	68.6±0.36	57.05
Scatterbrain	**38.6**±0.22	**64.55**±0.34	**80.22**±0.31	**43.65**±0.46	**69.91**±0.25	**59.38**

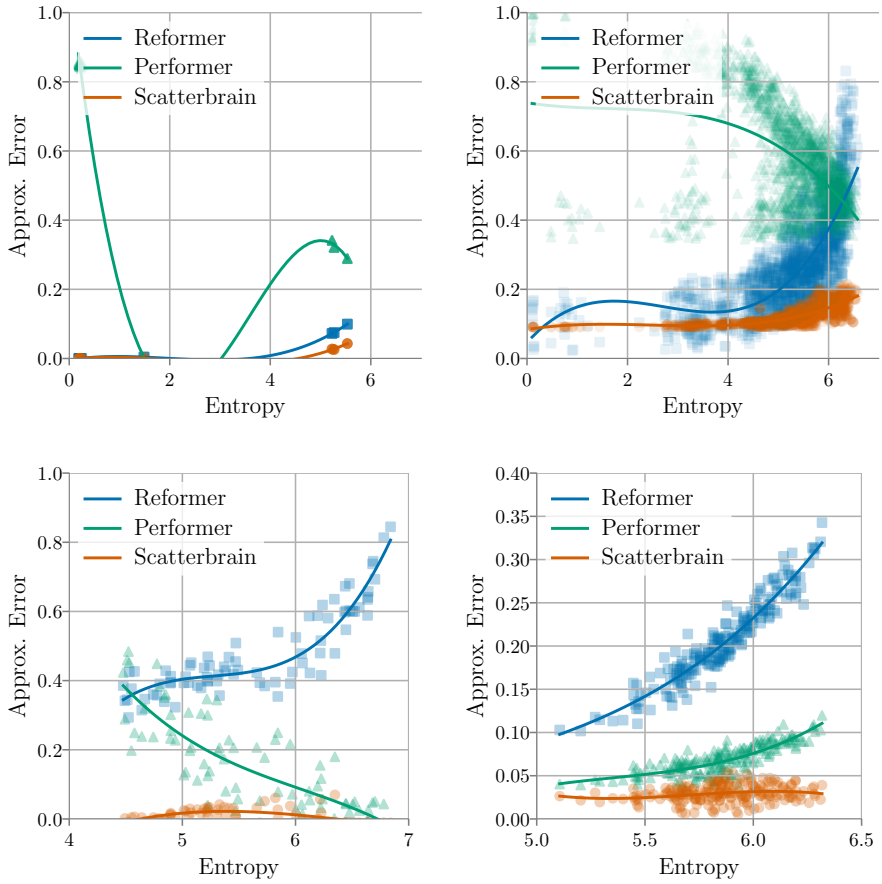

Figure 8: Top two plots present Approximation Error vs. Entropy of attention matrices for REFORMER, PERFORMER and Scatterbrain on Copy (left) and WikiText103 (right). Bottom two plots present Approximation Error vs. Entropy of attention matrices for REFORMER, PERFORMER and Scatterbrain on Text-IMDb (left) and Image-Cifar10 (right). Recall we observe that entropy of the softmax attention distribution (i.e., scale of logits) determines the regimes where sparse, low-rank, or sparse + low-rank perform well. Scatterbrain yields better approximation than REFORMER or PERFORMER in most of the cases; PERFORMER performs the worst on language modeling tasks while REFORMER performs the worst on classification tasks. These plots for approximation error analysis match with their performance on downstream tasks.

E.3 More Ablation Studies

E.3.1 Memory Budget

We present an ablation study on the parameter budget for the WikiText-103 language modeling task. We show that Scatterbrain outperforms its sparse and low-rank baselines across a range of parameter budgets. The results are presented in Table 5.

Analysis: We have observed that Scatterbrain outperforms its sparse and low-rank baselines under different memory budgets. Similar to what we found in Section 5.2, Performer does not train stably even with $\frac{1}{4}$ of the full attention memory. However, under the Scatterbrain framework, Performer can be combined with Reformer in an elegant way to achieve the same accuracy while using only half of the memory and faster than Reformer by exploiting the sparse+low-rank structure in attention matrices.

Table 5: We run WikiText-103 LM with a sweep of 1/4, 1/8, 1/16 memory budget. We show the validation perplexity and speed-up with respect to the full attention with different efficient Attention layers.

	$\frac{1}{4}$ Mem	$\frac{1}{8}$ Mem	$\frac{1}{16}$ Mem
	Perplexity (Speed-up)	Perplexity	Perplexity
SMYRF	26.76 (1.6×)	27.68 (1.39×)	28.7(1.85×)
PERFORMER	58(2.13×)	66 (2.01×)	85(1.77×)
Scatterbrain	**26.26(1.58×)**	**26.72 (1.87×)**	**27.74(2.03×)**

E.3.2 Different Sparse and Low-rank baselines

Scatterbrain is general enough to accommodate different kinds of sparse and low-rank approximations as its sub-components. In particular, we can combine Local attention or block sparse (from Sparse Transformer and BigBird) + Performer (instead of Reformer + Performer) in a similar fashion. The support of the sparse matrix S will thus be fixed and not adaptive to input, but all the other steps are exactly the same.

We have run additional experiments on the Local attention + Performer combination and BigBird. Recall that in Appendix E, we have shown Scatterbrain can reduce the attention memory of Vision Transformer by 98% at the cost of only 0.8% drop of accuracy when serving as a drop-in replacement for full attention without training on ImageNet. We show the results for local+performer variation with the same memory budget in Table 6.

We have also run additional experiments on Local attention on Copy and Wikitext-103 language modeling task (Table 7). We see that Local attention is reasonably competitive on Wikitext-103 but does not perform well on Copy. The results are not surprising as noted in the Reformer paper that Copy requires non-local attention lookups.

E.3.3 Different Sparse and Low-rank baselines

E.4 Analysis

Recall in Section 5, we have reported the analysis after 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. In Figure 8, we present the visualization.

Table 6: 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
Full Attention	81.7%
SMYRF	79.8%
Local	79.6%
Performer	80.1%
BigBird	80.3%
Scatterbrain (Local + Performer)	80.3%
Scatterbrain (SMYRF + Performer)	**80.7**%

Table 7: Additional experiments for Local attention on the Copy and Wikitext-103 language modeling task.

Attention	Copy (ppl)	WikiText-103 (ppl)
Full Attention	1	25.258
Reformer	6.8	27.68
Performer	49	66
Local	53	30.72
Scatterbrain	**2.58**	**26.72**

The conclusion for language modeling tasks is that 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 [58] that kernel or low-rank based approximations are less effective for hierarchical structured data. For classification tasks, 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).

E.5 Additional Experiments of Fine-tuning Bert on GLUE

We provide additional experiments of fine-tuning Bert on GLUE in Table 8. We follow the similar setting as [22]. We replace all the attention layers in Bert base model with Scatterbrain and other baselines. Then we fine-tune Bert on 9 downstream tasks for 3 epochs with batch size 32 and learning rate 3e-5. The parameter budget is 1/2 of the full attention because sequence length 128 is not very long. We can see Scatterbrain outperforms all the other baselines in most of the downstream tasks.

Table 8: Results of GLUE when replacing dense attention matrices with SMYRF, PERFORMER and Scatterbrain in BERT base model. We fix the same number ofparameters (1/2 of the full) used for approximating the attention matrix for each method.

	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	WNLI
	mcc	acc	acc	corr	acc	acc	acc	acc	acc
FULL	0.576	0.934	0.874	0.879	0.905	0.813	0.916	0.668	0.43
SMYRF	0.538	0.912	0.833	0.856	0.898	0.775	0.879	**0.626**	0.412
PERFORMER	0.508	0.838	0.782	0.203	0.831	0.563	0.763	0.556	**0.449**
Scatterbrain	**0.569**	**0.927**	**0.863**	**0.867**	**0.902**	**0.813**	**0.893**	0.619	0.428

F Further Discussions and Future Work

In this paper, we present Scatterbrain, unifying the strength of sparse and low-rank approximation. It is inspired by the observations on the attention matrix structures induced by the data and softmax function as well as the classical robust-PCA algorithm. In our implementation and analysis, we have REFORMER/Smyrf and PERFORMER as the back-bone for sparse and low-rank approximations because of their properties, e.g. Performer is unbiased. Scatterbrain is fundamentally a framework for combining the strength of sparse and low-rank variants, so it can be easily extended to other variants, such as Routing Transformer [54] or Nystromformer [68]. Further more, our observations on the connection between entropy and low-rank/sparse approximation error also provide an opportunity for efficiently detecting the approximation or compression method to choose for different architectures or benchmarks.