KDEformer: Accelerating Transformers via Kernel Density Estimation

Amir Zandieh [1] **Insu Han** [*2] **Majid Daliri** [*3] **Amin Karbasi** [2]

Abstract

Dot-product attention mechanism plays a crucial role in modern deep architectures (e.g., Transformer) for sequence modeling, however, naïve exact computation of this model incurs quadratic time and memory complexities in sequence length, hindering the training of long-sequence models. Critical bottlenecks are due to the computation of partition functions in the denominator of softmax function as well as the multiplication of the softmax matrix with the matrix of values. Our key observation is that the former can be reduced to a variant of the kernel density estimation (KDE) problem, and an efficient KDE solver can be further utilized to accelerate the latter via subsampling-based fast matrix products. Our proposed KDEformer can approximate the attention in sub-quadratic time with provable spectral norm bounds, while all prior results merely provide entry-wise error bounds. Empirically, we verify that KDEformer outperforms other attention approximations in terms of accuracy, memory, and runtime on various pre-trained models. On BigGAN image generation, we achieve better generative scores than the exact computation with over $4\times$ speedup. For ImageNet classification with T2T-ViT, KDEformer shows over $18\times$ speedup while the accuracy drop is less than 0.5%.

1. Introduction

Transformers (Vaswani et al., 2017) have been successfully applied to a wide variety of learning tasks in areas such as natural language processing (Devlin et al., 2018; Yang et al., 2019; Brown et al., 2020; Raffel et al., 2020), computer vision (Carion et al., 2020; Dosovitskiy et al., 2021), and time series forecasting (Zhou et al., 2021). Although popular, these models face serious scalability limitations because naïve exact computation of their attention layers incurs quadratic (in sequence length) runtime and memory complexities. This can inhibit the training of large-scale long-sequence models.

Several algorithms have been proposed to improve Transformers' efficiency via approximating the *softmax matrices* in their attention layers with either sparse matrices (Kitaev et al., 2020; Daras et al., 2020; Roy et al., 2021; Sun et al., 2021) or low-rank matrices (Choromanski et al., 2021; Katharopoulos et al., 2020), or a combination of both (Chen et al., 2021b; Zaheer et al., 2020; Chen et al., 2021a; Dass et al., 2022). However, all prior advances solely focused on point-wise approximating the entries of the softmax matrix and fail to provide rigorous approximation guarantees on the final output of the attention mechanism. In this work, we design algorithms to approximate the output matrix of attention layers with provable spectral norm guarantees.

1.1. Problem Formulation and Setting.

Let n be the number of tokens in the input sequence and d be the dimension of latent representations. The *dot-product attention* (Vaswani et al., 2017) is a mapping which takes inputs $\mathrm{Q}, \mathrm{K}, \mathrm{V} \in \mathbb{R}^{n\times d}$ (interpreted as queries, keys, and values of a dictionary) and outputs the following matrix:

$$\mathrm{Att}(\mathrm{Q},\mathrm{K},\mathrm{V}) := \mathrm{D}^{-1}\mathrm{A}\mathrm{V}$$
$$\mathrm{A} := \exp\left(\mathrm{Q}\mathrm{K}^\top/\sqrt{d}\right), \quad \mathrm{D} := \mathtt{diag}(\mathrm{A}\mathbf{1}_n),$$

where $\exp(\cdot)$ is applied in an element-wise manner, $\mathbf{1}_n$ is the ones vector in $\mathbb{R}^n$, and $\mathtt{diag}(\cdot)$ maps its input vector to a diagonal matrix. We refer to $\mathrm{A} \in \mathbb{R}^{n\times n}$ as the *attention matrix* and to $\mathrm{D}^{-1}\mathrm{A}$ as the *softmax matrix*. Exact computation of the attention matrix A takes $\Theta(n^2 d)$ operations and storing it requires $\Theta(n^2)$ memory. Thus, naïve computation of $\mathrm{Att}(\mathrm{Q},\mathrm{K},\mathrm{V})$ requires $\Omega(n^2 d)$ runtime and $\Omega(n^2)$ memory. Our aim is to approximate the output matrix $\mathrm{Att}(\mathrm{Q},\mathrm{K},\mathrm{V})$ efficiently while preserving its spectral structure.

Our approach is based on reducing the number of columns of matrix A using importance sampling. We also devise an efficient estimator for the diagonal scaling matrix D, which bypasses exact and explicit computation of matrix A. Formally, for any given $\varepsilon > 0$ and any $\mathrm{Q},\mathrm{K},\mathrm{V} \in \mathbb{R}^{n\times d}$, we want to quickly find a sampling matrix $\Pi \in \mathbb{R}^{m\times n}$ with a

[*]Equal contribution [1]Max-Planck-Institut für Informatik, Germany [2]Yale University, USA [3]New York University, USA. Correspondence to: Amir Zandieh <amir.zed512@gmail.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

small number $m = n^{1-\Omega(1)}$ of rows along with a diagonal matrix $\widetilde{\mathrm{D}} \in \mathbb{R}^{n\times n}$, such that the following bound on the *operator norm* of the error is satisfied:

$$\left\|\mathrm{Att}(\mathrm{Q},\mathrm{K},\mathrm{V}) - \widetilde{\mathrm{D}}^{-1}\mathrm{A}\Pi^\top \cdot \Pi\mathrm{V}\right\|_{\mathrm{op}} \leq \varepsilon \cdot \|\mathrm{V}\|_{\mathrm{op}}. \quad (1)$$

Note that $\mathrm{D}^{-1}\mathrm{A}$ is a *row-stochastic (transition) matrix*, so its operator norm is $\left\|\mathrm{D}^{-1}\mathrm{A}\right\|_{\mathrm{op}} = 1$, thus the r.h.s. in Eq. (1) is in fact equal to $\varepsilon \cdot \left\|\mathrm{D}^{-1}\mathrm{A}\right\|_{\mathrm{op}} \|\mathrm{V}\|_{\mathrm{op}}$.

Given a sampling matrix Π with m rows, we can compute the matrix product $\mathrm{A}\Pi^\top \cdot \Pi\mathrm{V}$ in $O(nmd)$ total runtime and $O(nm)$ memory because we only need to compute the m sampled columns of A. Therefore, our main goal is to generate a sampling matrix Π with a small number of samples along with a diagonal matrix $\widetilde{\mathrm{D}}$ which satisfy Eq. (1) using a sub-quadratic runtime in n.

All prior approximate attention methods have solely focused on finding an approximate attention matrix $\widetilde{\mathrm{A}}$ such that $\left\|\mathrm{A} - \widetilde{\mathrm{A}}\right\|_F$ is small, even though A is not the ultimate output of attention and the output depends on V in addition to A. In contrast, we propose the first efficient algorithm for approximating the output matrix $\mathrm{Att}(\mathrm{Q},\mathrm{K},\mathrm{V})$ with spectral bounds as per Eq. (1) (see Section 3.3).

1.2. Our Techniques and Results

We leverage the line of work on efficient *Kernel Density Estimation (KDE)* (Schölkopf et al., 2002; Joshi et al., 2011; Charikar & Siminelakis, 2017; Backurs et al., 2018; 2019; Siminelakis et al., 2019). In the KDE problem, we are given a dataset $\mathrm{X} = \{x_1, x_2, \dots x_n\}$ and a kernel function $k(\cdot,\cdot)$ and aim to compute the kernel density $\mu_{\mathrm{X}}(q) = \frac{1}{n}\sum_{i=1}^n k(q, x_i)$ for an arbitrary query point q. The goal of existing methods in the literature is to estimate this value to $(1+\varepsilon)$ relative error in time $O\left(\varepsilon^{-2}d/\widetilde{\mu}^\tau\right)$ for some $\tau > 0$, where $\widetilde{\mu}$ is a lower bound on $\mu_{\mathrm{X}}(q)$. Particularly, the best-known algorithm for the Gaussian kernel, due to Charikar et al. (2020), achieves $\tau = 0.173 + o(1)$.

We show that finding the sampling matrix Π and diagonal scaling $\widetilde{\mathrm{D}}$ which satisfy Eq. (1) can be reduced to a generalization of the KDE problem. First note that the i^{th} diagonal entry of the scaling matrix D is $\mathrm{D}_{i,i} = \sum_{j=1}^n \exp\left(\frac{\langle q_i, k_j\rangle}{\sqrt{d}}\right)$, which is indeed the kernel density corresponding to exponential kernel function $k(x,y) = \exp(\langle x, y\rangle)$ and dataset $\frac{1}{d^{1/4}} \cdot \mathrm{K}$ at query point $\frac{1}{d^{1/4}} \cdot q_i$. Thus, if we had an efficient KDE procedure for estimating the exponential kernel density up to a multiplicative $(1 \pm \varepsilon)$ factor, we could compute a scaling $\widetilde{\mathrm{D}}$ that satisfies the spectral guarantee of Eq. (1).

Additionally, to design an efficient sampling matrix Π that satisfies Eq. (1) with small number of rows, the sampling probabilities need to be proportional to the column norms of the softmax matrix $\mathrm{D}^{-1}\mathrm{A}$ (Zouzias, 2013). One can see that the squared norm of the i^{th} column of $\mathrm{D}^{-1}\mathrm{A}$ is $\sum_{j\in[n]} \mathrm{D}_{j,j}^{-2} \exp\left(\frac{2}{\sqrt{d}}\langle q_j, k_i\rangle\right)$, which is a *weighted* exponential kernel density with weights $\left\{\mathrm{D}_{i,i}^{-2}\right\}_{i\in[n]}$ and dataset $\frac{\sqrt{2}}{d^{1/4}} \cdot \mathrm{Q}$ at query point $\frac{\sqrt{2}}{d^{1/4}} \cdot k_i$. Therefore, if we could estimate this weighted exponential kernel density up to some constant multiplicative factor, we could generate a sampling matrix Π with small number of samples that satisfies Eq. (1).

Thus, having a generalized KDE procedure for efficiently evaluating the weighted exponential kernel density, enables us to approximate $\mathrm{Att}(\mathrm{Q},\mathrm{K},\mathrm{V})$ as per Eq. (1). While there is no prior solution for this problem, we show how to translate it to the Gaussian KDE problem, which has witnessed significant recent progress, by applying appropriate transformations on K and Q (see Algorithm 2 and Theorem 3.4).

Our Theoretical Results. We give an algorithm that outputs a diagonal $\widetilde{\mathrm{D}} \in \mathbb{R}^{n\times n}$ and a sampling matrix $\Pi \in \mathbb{R}^{m\times n}$ with $m = O\left(\varepsilon^{-2}\log n \cdot \mathtt{srank}(\mathrm{D}^{-1}\mathrm{A})\right)$ samples which satisfy the spectral bound of Eq. (1) with high probability in n, where $\mathtt{srank}(\mathrm{D}^{-1}\mathrm{A})$ denotes the *stable rank* of the softmax matrix. Our method reduces the memory of attention layers to $mn = O\left(\varepsilon^{-2}n\log n \cdot \mathtt{srank}(\mathrm{D}^{-1}\mathrm{A})\right)$. Furthermore, if the Gaussian KDE is supported by an algorithm with runtime $O\left(\varepsilon^{-2}d/\widetilde{\mu}^\tau\right)$ for relative error $1+\varepsilon$, and density lower bound $\widetilde{\mu}$, then our algorithm's runtime is bounded by $O\left(\varepsilon^{-2}d \cdot n^{1+\tau}\right)$ for any datasets of queries Q and keys K with diameter $\max_{i,j\in[n]} \|k_i - q_j\|_2^2 = o\left(\sqrt{d}\cdot\log n\right)$, which is strongly sub-quadratic in n. The current best value for τ is $\tau = 0.173 + o(1)$ due to (Charikar et al., 2020) and any future progress on Gaussian density evaluation immediately improves our method's runtime.

This result applies to a wide range of practical scenarios where the dimension d is not too large. To see why, note that entries of K,Q are typically constant, thus, the diameter is $\max_{i,j\in[n]} \|k_i - q_j\|_2^2 = O(d)$. Therefore, for any dimension $d = o(\log^2 n)$, e.g., $d \approx \frac{\log^2 n}{\log\log n}$, our method needs only $O\left(m + \varepsilon^{-2}d \cdot n^{1+\tau}\right)$ operations, which is significantly faster than exact computation of $\mathrm{Att}(\mathrm{Q},\mathrm{K},\mathrm{V})$.

Our Practical Results. Our necessary number m of samples depends on the stable rank of the softmax matrix. To reduce m, we employ Locality Sensitive Hashing (LSH) to extract the heavy elements of $\mathrm{D}^{-1}\mathrm{A}$ and then show that, in practice, the residual has a significantly smaller stable rank than the original matrix (see Section 3.4). With this heuristic improvement, we verify that our proposed algorithm outperforms popular attention approximations. In particular, it can save memory space up to $19.06\times$ when the sequence length n is 16,394. We apply our method to image generation with BigGAN (Brock et al., 2019) and observe that our images, shown in Fig. 1, look more natural

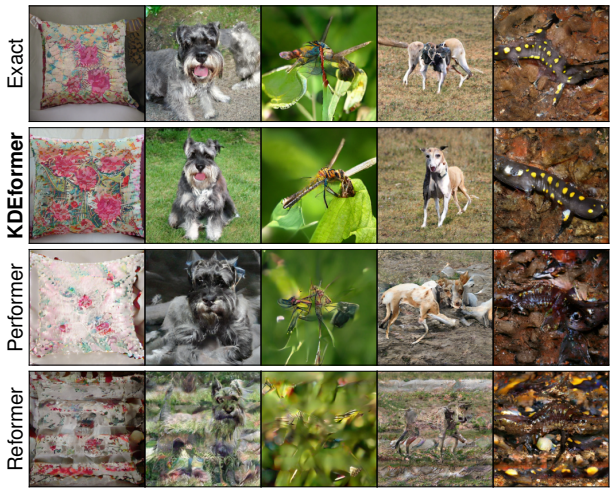


Figure 1. Image generations by the pre-trained BigGAN using exact and approximate attention without fine-tuning.

than others and our generative score is even better than the exact attention. Furthermore, for ImageNet classification with Vision Transformer (Yuan et al., 2021), KDEformer shows $18\times$ speedup and 82.08% accuracy which is only 0.5% lower than the exact attention (see Section 4.3). Finally, we demonstrate our method on end-to-end training under the Long Range Arena benchmark (Tay et al., 2021) and observe up to $8\times$ speedup on wall-clock time than the exact attention (see Section 4.4).

1.3. Prior Work

Several popular methods try to approximate the *heavy* entries of the attention matrix A by restricting the attention to local neighbors of queries using Locality Sensitive Hashing (LSH) (Kitaev et al., 2020; Chen et al., 2020; Sun et al., 2021) or k-means clustering (Daras et al., 2020; Roy et al., 2021). Such approaches, however, only provide error bounds on the attention matrix, e.g., guarantees of the form $\|\mathrm{A} - \widetilde{\mathrm{A}}\|_F < \varepsilon n$, and cannot provide any provable guarantees for the final output matrix $\mathrm{Att}(\mathrm{Q}, \mathrm{K}, \mathrm{V})$. Remarkably, at the core of our algorithm, there are invocations of the Gaussian KDE primitive from Charikar et al. (2020), which heavily employs LSH to estimate kernel densities. In contrast to previous works, our algorithm uses LSH in a more subtle way, that is for estimating the right sampling probabilities in order to generate Π and also to approximate the scaling D. This difference of approach allows us to approximate $\mathrm{Att}(\mathrm{Q}, \mathrm{K}, \mathrm{V})$ with spectral norm guarantees.

Another recent line of work is based on approximating the attention matrix A via random feature maps of the Gaussian or exponential kernels (Choromanski et al., 2021; Katharopoulos et al., 2020). Chen et al. (2021b) has recently shown that using a combination of both LSH-based and random features based methods works better at approximating the attention matrix A. See (Tay et al., 2022) for a survey.

2. Preliminaries and Notations

For any matrix A, we let a_i be its i^{th} row vector and its *stable rank* is defined as $\mathtt{srank}(\mathrm{A}) := \frac{\|\mathrm{A}\|_F^2}{\|\mathrm{A}\|_{\mathrm{op}}^2}$ which is always upper bounded by the algebraic rank. We denote $e_1, e_2, \ldots e_n$ by the standard basis vectors in $\mathbb{R}^n$ and $\mathbf{1}_n$ and $\mathbf{0}_n$ by the all-ones and all-zeros vectors in $\mathbb{R}^n$. For vectors x, y their *direct sum* is denoted by $x \oplus y := [x^\top, y^\top]^\top$.

Gaussian KDE. Our main algorithm is tightly related to the Gaussian KDE, where one is given a dataset $\mathrm{X} \in \mathbb{R}^{n \times d}$ and wants to build a data-structure (DS) such that given this DS one can estimate the following kernel density value up to $(1+\varepsilon)$ relative error for any query point $q \in \mathbb{R}^d$:

$$\mu_{\mathrm{X}}(q) := \frac{1}{n} \sum_{i \in [n]} \exp(-\|q - x_i\|_2^2 / 2). \tag{2}$$

The naïve method without any DS requires $\Theta(nd)$ time and memory complexities. The aim is to minimize the memory needed to store the DS and the query time, ultimately being sublinear in n. The pre-processing time which is needed to construct the DS is also desired to be small. There have been significant advances on this problem and the current best result was proposed by Charikar et al. (2020) as follows:

Theorem 2.1 (Fast Gaussian KDE, Theorem 2 in (Charikar et al., 2020))**.** *Let $\tau = 0.173 + o(1)$. For any dataset $\mathrm{X} \in \mathbb{R}^{n \times d}$ and any $\varepsilon, \widetilde{\mu} \in (0, 1)$, there exist:*

1. *Procedure* PREPROCESSKDE$(\mathrm{X}, \varepsilon, \widetilde{\mu})$ *constructs a data-structure named* $\mathtt{DS}_{\mathtt{kde}}$ *in time* $O\left(\varepsilon^{-2} dn / \widetilde{\mu}^{\tau}\right)$.
2. *Given* $\mathtt{DS}_{\mathtt{kde}}$*, any query $q \in \mathbb{R}^d$, and $\mu_{\mathrm{X}}(q)$ defined as in Eq. (2), procedure* QUERYKDE$(\mathtt{DS}_{\mathtt{kde}}, q)$ *approximates the quantity $\mu_{\mathrm{X}}(q) \cdot \mathbb{1}_{\{\widetilde{\mu} \le \mu_{\mathrm{X}}(q)\}}$ up to $(1+\varepsilon)$ relative error in time $O(\varepsilon^{-2} d \left(\widetilde{\mu} + \mu_{\mathrm{X}}(q)\right)^{-\tau})$.*

The density lower bound $\widetilde{\mu}$ required by Theorem 2.1 is unknown to us in advance and we learn this quantity adaptively in Algorithm 2. We show in Section 3.3 that for datasets with bounded diameter $\widetilde{\mu} = n^{-1-o(1)}$.

3. Efficient Attention with Spectral Bounds

In this section, we design KDEformer which can efficiently compute a sampling matrix Π and a diagonal scaling $\widetilde{\mathrm{D}}$ satisfying Eq. (1). We start by showing that this can be done very efficiently given access to a primitive for estimating the row-norms of the attention matrix A as well as the column-norms of the softmax matrix $\mathrm{D}^{-1}\mathrm{A}$. Next, in Section 3.2, we present a reduction from norm estimators for A and $\mathrm{D}^{-1}\mathrm{A}$ to the Gaussian KDE problem which has an efficient solution. Finally, we prove our main result in Section 3.3

3.1. High-level Architecture of the Algorithm

Here, we assume that we have access to an oracle, which can estimate the *weighted* linear combination of n exponential kernels at arbitrary query points, and given this oracle, we design an algorithm that can output Π and $\widetilde{\mathrm{D}}$ which satisfy Eq. (1). In other words, we translate and reduce the problem of spectrally approximating $\mathrm{Att}(\mathrm{Q},\mathrm{K},\mathrm{V})$ to a weighted KDE problem corresponding to the exponential dot-product kernel. The precise interface and desired properties of this oracle are presented in the following definition,

Definition 3.1 (Weighted Exponential KDE)**.** Let $\mathrm{X},\mathrm{Y}\in\mathbb{R}^{n\times d}$ be arbitrary datasets and let $v\in\mathbb{R}^n_+$ be an arbitrary vector with positive coordinates. For any $\varepsilon>0$, primitive $\textsc{WExpKDE}(\mathrm{X},\mathrm{Y},v,\varepsilon)$ outputs a non-negative vector $\alpha\in\mathbb{R}^n_+$ such that:

$$\alpha_j\in(1\pm\varepsilon)\cdot\sum_{i\in[n]}v_i\exp(\langle x_i,y_j\rangle)\quad\forall j\in[n]. \tag{3}$$

Now we show how to generate Π and $\widetilde{\mathrm{D}}$ that satisfy Eq. (1), given access to $\textsc{WExpKDE}$ as per Definition 3.1.

Estimating $\mathrm{D}=\mathtt{diag}\left(\exp\left(\mathrm{QK}^\top/\sqrt{d}\right)\mathbf{1}_n\right)$**.** One can easily see that the j^{th} diagonal entry of D equals:

$$\mathrm{D}_{j,j}=\sum_{i\in[n]}\exp\left(\langle k_i,q_j\rangle/\sqrt{d}\right)\quad\forall j\in[n]. \tag{4}$$

Therefore, if we let $\alpha=\textsc{WExpKDE}\left(\frac{\mathrm{K}}{d^{1/4}},\frac{\mathrm{Q}}{d^{1/4}},\mathbf{1}_n,\frac{\varepsilon}{3}\right)$ and define $\widetilde{\mathrm{D}}=\mathtt{diag}(\alpha)$, then by Definition 3.1 and using the fact that entries of D are positive, we have $(1-\varepsilon/3)\mathrm{D}\preceq\widetilde{\mathrm{D}}\preceq(1+\varepsilon/3)\mathrm{D}$ where $\preceq$ is the Loewner order. So, using the fact that $\left\|\mathrm{D}^{-1}\mathrm{A}\right\|_{\mathrm{op}}=1$,

$$\left\|\mathrm{Att}(\mathrm{Q},\mathrm{K},\mathrm{V})-\widetilde{\mathrm{D}}^{-1}\mathrm{AV}\right\|_{\mathrm{op}}\le\frac{\varepsilon}{2}\cdot\|\mathrm{V}\|_{\mathrm{op}}. \tag{5}$$

Hence, we can estimate D to sufficient precision by invoking $\textsc{WExpKDE}\left(\frac{\mathrm{K}}{d^{1/4}},\frac{\mathrm{Q}}{d^{1/4}},\mathbf{1}_n,\frac{\varepsilon}{3}\right)$.

Generating the Sampling Matrix Π**.** Given a diagonal matrix $\widetilde{\mathrm{D}}$ which satisfies Eq. (5), by triangle inequality, in order to satisfy the spectral bound of Eq. (1), it suffices to find a sampling matrix for which the following holds,

$$\left\|\widetilde{\mathrm{D}}^{-1}\mathrm{A}\Pi^\top\cdot\Pi\mathrm{V}-\widetilde{\mathrm{D}}^{-1}\mathrm{AV}\right\|_{\mathrm{op}}\le\frac{\varepsilon}{2}\cdot\|\mathrm{V}\|_{\mathrm{op}} \tag{6}$$

So, our goal is to design a sampling matrix $\Pi\in\mathbb{R}^{m\times n}$ with a small number m of rows that satisfies Eq. (6). This problem is in fact well studied in the randomized numerical linear algebra literature and is known as the *Approximate Matrix Multiplication* (AMM) with respect to the spectral norm. It is known how to achieve the above guarantee using a sampling matrix with $m=O\left(\varepsilon^{-2}\log n\cdot(\mathtt{srank}(\mathrm{D}^{-1}\mathrm{A})+\mathtt{srank}(\mathrm{V}))\right)$ i.i.d. rows.

More formally, we have the following result which is a slight modification of Theorem 2.1 from (Zouzias, 2013) and is proved in Appendix B.1.

Lemma 3.2 (AMM)**.** *For any matrices $\mathrm{X}\in\mathbb{R}^{n\times q},\mathrm{Y}\in\mathbb{R}^{n\times d}$ and any probability distribution $\{p_i\}_{i\in[n]}$ satisfying $p_i\ge\frac{1}{4}\cdot\frac{\|x_i\|_2^2+\gamma\cdot\|y_i\|_2^2}{\|\mathrm{X}\|_F^2+\gamma\cdot\|\mathrm{Y}\|_F^2}$ for all $i\in[n]$ and $\gamma=\|\mathrm{X}\|_{\mathrm{op}}^2/\|\mathrm{Y}\|_{\mathrm{op}}^2$, a sampling matrix $\Pi\in\mathbb{R}^{m\times n}$ constructed by first generating m i.i.d. samples $\ell_1,\dots\ell_m\in[n]$ according to $\{p_\ell\}_{\ell\in[n]}$ and then letting the r^{th} row of Π be $\frac{1}{\sqrt{m\cdot p_{\ell_r}}}\cdot e_{\ell_r}^\top$, if $m=\Omega\left(\varepsilon^{-2}\log n\cdot(\mathtt{srank}(\mathrm{X})+\mathtt{srank}(\mathrm{Y}))\right)$ for some $\varepsilon>0$, the following holds,*

$$\Pr\left[\left\|\mathrm{X}^\top\Pi^\top\Pi\mathrm{Y}-\mathrm{X}^\top\mathrm{Y}\right\|_{\mathrm{op}}>\varepsilon\|\mathrm{X}\|_{\mathrm{op}}\|\mathrm{Y}\|_{\mathrm{op}}\right]\le\frac{1}{\mathrm{poly}(n)}.$$

So, by invoking Lemma 3.2 with $\mathrm{X}^\top=\widetilde{\mathrm{D}}^{-1}\mathrm{A}$ and $\mathrm{Y}=\mathrm{V}$ and error parameter $\varepsilon/2$, we can find a random sampling matrix Π which satisfies Eq. (6) with high probability in n, as long as the number of samples is at least $m=\Omega\left(\varepsilon^{-2}\log n(\mathtt{srank}(\widetilde{\mathrm{D}}^{-1}\mathrm{A})+\mathtt{srank}(\mathrm{V}))\right)$. The only catch is that, to apply Lemma 3.2, we need to compute the distribution $\{p_i\}_{i\in[n]}$ as per this lemma. In other words, we need to compute the row norms of V as well as the column norms of $\widetilde{\mathrm{D}}^{-1}\mathrm{A}$. All row norms of V can be computed in $O(nd)$ time. However, naively computing the column norms of $\widetilde{\mathrm{D}}^{-1}\mathrm{A}$ would require $\Theta(n^2d)$ operations. Fortunately, the column norms of $\widetilde{\mathrm{D}}^{-1}\mathrm{A}$ can be approximated via the primitive $\textsc{WExpKDE}$ from Definition 3.1.

The procedure for computing $\widetilde{\mathrm{D}}$ and sampler Π is presented in Algorithm 1. We state the correctness of Algorithm 1 in the following theorem and prove it in Appendix B.2.

Algorithm 1 KDEformer

input: matrices $\mathrm{Q},\mathrm{K},\mathrm{V}\in\mathbb{R}^{n\times d}$, integer m, and $\varepsilon>0$
$\gamma\leftarrow\|\mathrm{V}\|_{\mathrm{op}}^{-2}$ via power method
$\alpha\leftarrow\textsc{WExpKDE}\left(\frac{\mathrm{K}}{d^{1/4}},\frac{\mathrm{Q}}{d^{1/4}},\mathbf{1}_n,\frac{\varepsilon}{3}\right)$ in Definition 3.1
$\beta\leftarrow\textsc{WExpKDE}\left(\frac{\sqrt{2}\cdot\mathrm{Q}}{d^{1/4}},\frac{\sqrt{2}\cdot\mathrm{K}}{d^{1/4}},u,1/3\right)$, where $u_i\leftarrow1/\alpha_i^2$ for every $i\in[n]$
$p_i\leftarrow\beta_i+\gamma\cdot\|v_i\|_2^2$ for every $i\in[n]$ then normalize $p_\ell\leftarrow\frac{p_\ell}{\sum_{j\in[n]}p_j}$ for every $\ell\in[n]$
generate i.i.d. samples $\ell_1,\ell_2,\dots\ell_m\in[n]$ from distribution $\{p_\ell\}_{\ell\in[n]}$
let r^{th} row of Π be $\frac{1}{\sqrt{m\cdot p_{\ell_r}}}\cdot e_{\ell_r}^\top$ for every $r\in[m]$
return $\widetilde{\mathrm{D}}=\mathtt{diag}(\alpha)$ and Π

Theorem 3.3 (Correctness of Algorithm 1)**.** *For any matrices* $\mathrm{Q}, \mathrm{K}, \mathrm{V} \in \mathbb{R}^{n \times d}$*, any* $\varepsilon > 0$*, and number of samples* $m = \Omega\left(\varepsilon^{-2} \log n \cdot (\mathtt{srank}(\mathrm{D}^{-1}\mathrm{A}) + \mathtt{srank}(\mathrm{V}))\right)$*, given access to a primitive* WEXPKDE *as per Definition 3.1, Algorithm 1 outputs a diagonal matrix* $\widetilde{\mathrm{D}} \in \mathbb{R}^{n \times n}$ *and a sampling matrix* $\Pi \in \mathbb{R}^{m \times n}$ *which satisfy Eq. (1) with probability at least* $1 - \frac{1}{\mathrm{poly}(n)}$.

So, to spectrally approximate $\mathrm{Att}(\mathrm{Q}, \mathrm{K}, \mathrm{V})$, it is enough to run Algorithm 1. This algorithm relies on the existence of primitive WEXPKDE as per Definition 3.1, therefore, we focus on efficient implementation of WEXPKDE.

3.2. Weighted Exponential KDE

Here, we devise an efficient algorithm that satisfies the desired properties of WEXPKDE as per Definition 3.1. We show that this procedure is tightly related to and can be translated to an instance of the Gaussian KDE. First note that if all data-points in dataset X were on a sphere, i.e., $\|x_i\|_2 = r$ for all $i \in [n]$ and some $r > 0$, then the weighted exponential kernel density corresponding to the weights $v = \frac{1}{n} \cdot \mathbf{1}_n$ would be equal to $e^{(\|q\|_2^2 + r^2)/2} \cdot \mu_{\mathrm{X}}(q)$, where $\mu_{\mathrm{X}}(q)$ is defined as in Eq. (2).

Our proposed WEXPKDE primitive employs a fast Gaussian KDE method as per Theorem 2.1. The weighted exponential kernel density for a query point q and weight vector $v \in \mathbb{R}^n_+$ can be written as,

$$\sum_{i \in [n]} v_i e^{\langle x_i, q \rangle} = e^{\frac{\|q\|_2^2}{2}} \sum_{i \in [n]} v_i e^{\frac{\|x_i\|_2^2}{2}} \cdot e^{-\frac{\|x_i - q\|_2^2}{2}}. \quad (7)$$

Let us define $w_i := \sqrt{2 \log \frac{\sum_{j \in [n]} v_j \exp(\|x_j\|_2^2/2)}{v_i \cdot \exp(\|x_i\|_2^2/2)}}$ for every $i \in [n]$ and define the augmented dataset $\mathrm{X}' \in \mathbb{R}^{n \times (d+1)}$ as $x'_i := x_i \oplus [w_i]$ for every $i \in [n]$. Also let the augmented query point be $q' := q \oplus [0]$. Then, the r.h.s. in Eq. (7) can be written as

$$e^{\frac{\|q\|_2^2}{2}} \sum_{i \in [n]} v_i e^{\frac{\|x_i\|_2^2}{2}} \cdot \exp\left(-\frac{\|x'_i - q'\|_2^2}{2} + \frac{w_i^2}{2}\right)$$
$$= n \cdot e^{\frac{\|q\|_2^2}{2}} \sum_{j \in [n]} v_j e^{\frac{\|x_j\|_2^2}{2}} \cdot \mu_{\mathrm{X}'}(q'). \quad (8)$$

Therefore, the weighted exponential kernel density can be obtained from the Gaussian kernel density corresponding to the augmented dataset X' and augmented query q', i.e., $\mu_{\mathrm{X}'}(q')$. The augmented dataset can be constructed very efficiently in time $O(nd)$, so given a fast Gaussian KDE as per Theorem 2.1, Eq. (8) shows us an efficient way to implement the WEXPKDE procedure. Our proposed procedure is presented in Algorithm 2. Note that, fast Gaussian KDE requires a lower bound $\widetilde{\mu}$ on the kernel density value $\mu_{\mathrm{X}'}(q')$, and we show how to adaptively learn $\widetilde{\mu}$ in Algorithm 2 using the fact that if $\textsc{QueryKDE}(\mathtt{DS}_{\mathtt{kde}}, q')$ outputs zero we can infer that our lower bound was too high. We analyze Algorithm 2 in the following theorem.

Algorithm 2 Weighted Exponential KDE (WEXPKDE)

1: **input**: matrices $\mathrm{X}, \mathrm{Y} \in \mathbb{R}^{n \times d}$, vector $v \in \mathbb{R}^n_+$, error parameter $\varepsilon > 0$, and $\tau > 0$
2: $\mu \leftarrow 1/n$ and $S \leftarrow [n]$ and $\alpha \leftarrow \mathbf{0}_n$
3: $N \leftarrow \sum_{j \in [n]} v_j e^{\frac{\|x_j\|_2^2}{2}}$
4: $w_i \leftarrow \sqrt{2 \log \frac{N}{v_i \cdot \exp(\|x_i\|_2^2/2)}}$ for every $i \in [n]$
5: $\mathrm{X}' \leftarrow [\mathrm{X}; w] \in \mathbb{R}^{n \times (d+1)}$, $\mathrm{Y}' \leftarrow [\mathrm{Y}; \mathbf{0}_n] \in \mathbb{R}^{n \times (d+1)}$
6: **while** $\mu^{-\tau} \leq \varepsilon^2 \cdot |S|$ **do**
7: $\quad \mathtt{DS}_{\mathtt{kde}} \leftarrow \textsc{PreprocessKDE}(\mathrm{X}', \varepsilon, \mu)$
8: $\quad \alpha_i \leftarrow n \cdot N \cdot e^{\frac{\|y_i\|_2^2}{2}} \cdot \textsc{QueryKDE}(\mathtt{DS}_{\mathtt{kde}}, y'_i)$ for every $i \in S$
9: $\quad \mu \leftarrow \mu/2$ and $S \leftarrow \{i \in [n] : \alpha_i = 0\}$
10: **end while**
11: $\alpha_j \leftarrow \sum_{i \in [n]} v_i \cdot \exp(\langle x_i, y_j \rangle)$ for every $j \in S$
12: **return** α

Theorem 3.4 (Analysis of Algorithm 2)**.** *For every matrices* $\mathrm{X}, \mathrm{Y} \in \mathbb{R}^{n \times d}$*, any non-negative vector* $v \in \mathbb{R}^n_+$*, and any* $\varepsilon \in (0, 1)$*, and given a fast Gaussian KDE as per Theorem 2.1, Algorithm 2 outputs a vector* $\alpha \in \mathbb{R}^n$ *which satisfies the desired conditions of Definition 3.1 (i.e., Eq. (3)). Furthermore, this procedure's runtime is* $O\left(nd \cdot \mathcal{C}_{\mathrm{X},\mathrm{Y},v,\varepsilon,\tau}\right)$*, where*

$$\mathcal{C}_{\mathrm{X},\mathrm{Y},v,\varepsilon,\tau} := \quad (9)$$
$$\min_{\mu > 0} \frac{1}{\varepsilon^2 \mu^\tau} + \left| \left\{ i \in [n] : \frac{\sum_{j=1}^n v_j e^{\langle x_j, y_i \rangle}}{\sum_{j=1}^n v_j e^{\frac{\|x_j\|_2^2 + \|y_i\|_2^2}{2}}} < n\mu \right\} \right|$$

Proof. First, we prove the correctness. Let us index the iterations of the algorithm's while loop by $t = 0, 1, 2, \ldots$ and let μ_t, α_t, and S_t denote the value of μ, the vector α, and set S at t^{th} iteration. We have $|S_t| \leq n$ and $\mu_t = \frac{1}{n \cdot 2^t}$ for every t, thus, the algorithm must terminate in $T = O(\log n)$ iterations. Also, by Theorem 2.1, the set S_{t+1} computed in line 9 equals $S_{t+1} = \{i \in [n] : \mu_{\mathrm{X}'}(y'_i) < \mu_t\}$, because the fast Gaussian KDE procedure outputs zero if and only if $\mu_{\mathrm{X}'}(y'_i) < \mu_t$.

Next, we show by induction that at every iteration t, $\alpha_t(i)$ is within $(1 \pm \varepsilon)$ factor of $nNe^{\frac{\|y_i\|_2^2}{2}} \cdot \mu_{\mathrm{X}'}(y'_i)$ for all $i \in [n] \setminus S_t$. **Base of induction** is trivial because $S_0 = [n]$. For proving the **inductive step**, note that in lines 7-8 $\alpha_{t+1}(i)$ is updated for every $i \in S_t$ by invoking the fast Gaussian KDE procedure and $\alpha_{t+1}(i) = \alpha_t(i)$ for $i \in [n] \setminus S_t$. Thus, by the inductive hypothesis and Theorem 2.1 as well as

definition of S_{t+1} in line 9, $\alpha_{t+1}(i)$ is within $(1 \pm \varepsilon)$ factor of $nNe^{\frac{\|y_i\|_2^2}{2}} \cdot \mu_{\mathrm{X}'}(y_i')$ for all $i \in [n] \backslash S_{t+1}$, which completes the inductive proof. Using the definition of N in line 3 and definition of X', Y' in line 5 along with Eq. (8), the invariant that we proved implies that for every $t = 0, 1, \ldots T$, $\alpha_t(i)$ is within $(1 \pm \varepsilon)$ factor of $\sum_{j \in [n]} v_j \cdot \exp(\langle x_j, y_i \rangle)$ for all $i \in [n] \setminus S_t$. After exiting the while loop, $\alpha(i)$ is updated at all $i \in S_{T+1}$ in line 2 as $\alpha(i) = \sum_{j \in [n]} v_j \cdot \exp(\langle x_j, y_i \rangle)$, and $\alpha(i) = \alpha_T(i)$ for every $i \in [n] \setminus S_T$. This proves that the output vector α satisfies Eq. (3), which completes the correctness proof.

Runtime Analysis. The runtime has three components;

1) Time to run PREPROCESSKDE in line 7. The total time of running this primitive in all iterations $t = 0, 1, \ldots T$ is $O\left(\sum_{t=0}^{T} \frac{d \cdot n}{\varepsilon^2} \mu_t^{-\tau}\right)$, by Theorem 2.1. Since $\mu_t = \frac{1}{n \cdot 2^t}$, this runtime is bounded by $O\left(\frac{d \cdot n}{\varepsilon^2} \mu_T^{-\tau}\right)$.

2) Time to run QUERYKDE in line 8. By Theorem 2.1, the total time to run this procedure in all iterations is $O\left(\frac{d}{\varepsilon^2} \cdot \sum_{t=0}^{T} \sum_{i \in S_t} \left(\mu_t + \mu_{\mathrm{X}'}(y_i')\right)^{-\tau}\right)$. Because $|S_t| \leq n$, this runtime complexity is completely dominated by **(1)**.

3) Time to exactly compute the weighted exponential densities of the points with very small $\mu_{\mathrm{X}'}(y_i')$ value in line 10. This runtime is bounded by $O(nd \cdot |S_{T+1}|)$.

Now we combine these bounds. Using the assumption that the algorithm terminated at iteration $t = T$, the while loop condition at iteration $T + 1$ must fail. Therefore, $|S_{T+1}| < \mu_{T+1}^{-\tau}/\varepsilon^2 < 2\mu_T^{-\tau}/\varepsilon^2$. This shows that the first component of the runtime must dominate the third component. So the total time is bounded by $O\left(\frac{d \cdot n}{\varepsilon^2} \mu_T^{-\tau}\right)$.

Recall that the while loop terminates at iteration T meaning that $\varepsilon^{-2} \mu_t^{-\tau} \leq |S_t|$ for every $t = 0, 1, \ldots T$ and $\varepsilon^{-2} \mu_{T+1}^{-\tau} > |S_{T+1}|$. So, T is the largest integer that satisfies $\varepsilon^{-2} \mu_T^{-\tau} \leq |S_T|$. Also recall that $S_t = \{i \in [n] : \mu_{\mathrm{X}'}(y_i') < \mu_{t-1}\}$ and $\mu_t = \frac{1}{n \cdot 2^t}$. Thus, the runtime of the procedure can be expressed as,

$$O(nd) \cdot \min_{\mu > 0} \varepsilon^{-2} \mu^{-\tau} + \left|\{i \in [n] : \mu_{\mathrm{X}'}(y_i') < \mu\}\right|.$$

The definition of X', Y' in line 5 along with Eq. (8) gives the claimed runtime bound in Eq. (9). □

To get a better understanding of the runtime bound in Theorem 3.4, suppose that datasets X, Y are such that cardinality of set $\left\{ i \in [n] : \frac{\sum_{j \in [n]} v_j \exp(\langle x_j, y_i \rangle)}{\sum_{j \in [n]} v_j \exp\left(\frac{\|x_j\|_2^2 + \|y_i\|_2^2}{2}\right)} \leq n^{-o(1)} \right\}$ is upper bounded by $O\left(\varepsilon^{-2} \cdot n^{\tau}\right)$. For such datasets, the runtime of Theorem 3.4 is bounded by $O\left(\varepsilon^{-2} d \cdot n^{1+\tau+o(1)}\right)$, which is strongly sub-quadratic in n.

3.3. Main Result

Now we are in a position to prove our main result, i.e., an efficient algorithm that can approximate the attention mechanism with spectral guarantees as per Eq. (1).

Theorem 3.5 (Approximate Attention with Spectral Norm Bound)**.** *For any matrices $\mathrm{Q}, \mathrm{K}, \mathrm{V} \in \mathbb{R}^{n \times d}$, any $\varepsilon > 0$, and given a fast Gaussian KDE as per Theorem 2.1, there exists an algorithm that outputs a diagonal matrix $\widetilde{\mathrm{D}} \in \mathbb{R}^{n \times n}$ and a sampling matrix $\Pi \in \mathbb{R}^{m \times n}$ with $m = O\left(\varepsilon^{-2} \log n \cdot (\mathtt{srank}(\mathrm{D}^{-1}\mathrm{A}) + \mathtt{srank}(\mathrm{V}))\right)$ samples which satisfy Eq. (1) with probability at least $1 - \frac{1}{\mathrm{poly}(n)}$. The runtime of this algorithm is $O\left(m + nd \cdot \left(\mathcal{C}_{\frac{\mathrm{K}}{d^{1/4}}, \frac{\mathrm{Q}}{d^{1/4}}, \mathbf{1}_n, \varepsilon, \tau} + \mathcal{C}_{\frac{\sqrt{2} \cdot \mathrm{Q}}{d^{1/4}}, \frac{\sqrt{2} \cdot \mathrm{K}}{d^{1/4}}, v, 1, \tau}\right)\right)$, where $v_j = \left(\sum_{\ell \in [n]} \exp\left(\frac{1}{\sqrt{d}} \langle q_j, k_\ell \rangle\right)\right)^{-2}$ for $j \in [n]$ and $\mathcal{C}_{\frac{\mathrm{K}}{d^{1/4}}, \frac{\mathrm{Q}}{d^{1/4}}, \mathbf{1}_n, \varepsilon, \tau}, \mathcal{C}_{\frac{\sqrt{2} \cdot \mathrm{Q}}{d^{1/4}}, \frac{\sqrt{2} \cdot \mathrm{K}}{d^{1/4}}, v, 1, \tau}$ are defined as in Eq. (9).*

We prove this theorem in Appendix B.3. The runtime bound in Theorem 3.5 can be simplified for datasets Q, K with bounded diameter as follows,

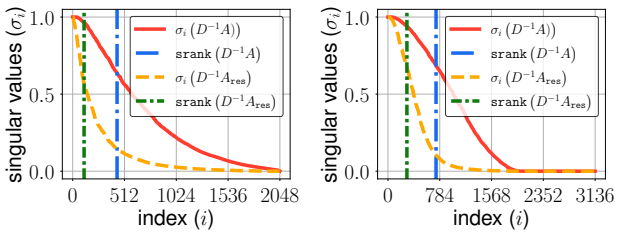
Corollary 3.6 (Simplified Runtime for Bounded Diameter Datasets)**.** *For any datasets Q, K with diameter $\max_{i,j \in [n]} \|k_i - q_j\|_2^2 = \gamma \sqrt{d} \log n$ for some $\gamma > 0$, the runtime of Theorem 3.5 is upper bounded by $O\left(m + nd \cdot \left(n^{\tau(1+\gamma)} + \varepsilon^{-2} n^{\tau(1+\gamma/2)}\right)\right)$, which is strongly sub-quadratic in n. In particular, if $\gamma = o(1)$, the runtime is bounded by $O\left(m + \varepsilon^{-2} d \cdot n^{1+\tau+o(1)}\right)$.*

We prove Corollary 3.6 in Appendix B.4. The current best value for τ is $\tau = 0.173 + o(1)$ due to Charikar et al. (2020), thus, for any datasets of queries Q and keys K with diameter $\max_{i,j \in [n]} \|k_i - q_j\|_2^2 = o(\sqrt{d} \log n)$, our algorithm's runtime is $O\left(m + \varepsilon^{-2} d \cdot n^{1.173+o(1)}\right)$.

3.4. Practical Improvements by Exploiting Sparsity

Our method relies on a sampling-based AMM (Lemma 3.2) and the number of samples m is proportional to $\mathtt{srank}(\mathrm{D}^{-1}\mathrm{A})$ by Theorem 3.5. Here, we propose a practical technique for reducing the stable rank of $\mathrm{D}^{-1}\mathrm{A}$ by finding and subtracting off its "heavy" elements. Specifically, recall that $\mathtt{srank}(\mathrm{D}^{-1}\mathrm{A}) = \frac{\|\mathrm{D}^{-1}\mathrm{A}\|_F^2}{\|\mathrm{D}^{-1}\mathrm{A}\|_{\mathrm{op}}^2}$ and the softmax matrix $\mathrm{D}^{-1}\mathrm{A}$ is dominated by its largest elements which correspond to the nearest pairs of queries q_i and keys k_j. Therefore, subtracting off the heavy elements of $\mathrm{D}^{-1}\mathrm{A}$ reduces $\left\|\mathrm{D}^{-1}\mathrm{A}\right\|_F^2$ which in turn can reduce $\mathtt{srank}(\mathrm{D}^{-1}\mathrm{A})$.

Similar to Reformer (Kitaev et al., 2020), we employ a Locality Sensitive Hashing (LSH) scheme to find dominant entries of the attention matrix A. Specifically, let $\mathcal{H} : \mathbb{R}^d \to [B]$ be an LSH function with B buckets such that the collision probability $\Pr[\mathcal{H}(q_i) = \mathcal{H}(k_j)]$ is "roughly"



(a) GloVe dataset (b) T2T-ViT on ImageNet

Figure 2. Singular values distribution and stable rank of the softmax matrix $\mathrm{D}^{-1}\mathrm{A}$ versus those of the residual $\mathrm{D}^{-1}\mathrm{A}_{\mathtt{res}}$. The stable rank of the residual matrix is significantly smaller.

proportional to $\langle q_i, k_j \rangle$. Given such LSH function, we define the sparse approximation to A as well as the residual attention matrix as:

$$\forall i,j \in [n]: \quad [\mathrm{A}_{\mathtt{spar}}]_{i,j} := e^{\frac{\langle q_i, k_j \rangle}{\sqrt{d}}} \cdot \mathbb{1}_{\{\mathcal{H}(q_i)=\mathcal{H}(k_j)\}}$$
$$\mathrm{A}_{\mathtt{res}} := \mathrm{A} - \mathrm{A}_{\mathtt{spar}}. \tag{10}$$

Intuitively, the stable rank of $\mathrm{D}^{-1}\mathrm{A}_{\mathtt{res}}$ is expected to be smaller than that of $\mathrm{D}^{-1}\mathrm{A}$ because the former has a considerably smaller Frobenius norm. We verify this intuition by plotting the singular values distributions of the softmax matrix $\mathrm{D}^{-1}\mathrm{A}$ and the residual $\mathrm{D}^{-1}\mathrm{A}_{\mathtt{res}}$ for two real-world instances in Fig. 2. Fig. 2(a) corresponds to when keys and queries are the first $n = 2{,}048$ vectors from GloVe word embedding dataset (Pennington et al., 2014). In Fig. 2(b), we focused on the first attention layer in Tokens-to-token Vision Transformer (T2T-ViT) (Yuan et al., 2021) and an arbitrary batch of images from ImageNet dataset. In both instances, the singular values of the residual $\mathrm{D}^{-1}\mathrm{A}_{\mathtt{res}}$ decay faster than that of $\mathrm{D}^{-1}\mathrm{A}$ while the largest singular value (spectral norm) of both matrices are equal to one. Thus, as shown in Fig. 2, subtracting off the sparse component $\mathrm{D}^{-1}\mathrm{A}_{\mathtt{spar}}$ reduces the stable rank significantly.

Building upon this observation, we propose a new version of Algorithm 1 with improved practical performance. We start by using Eq. (10) to write:

$$\mathrm{Att}(\mathrm{Q},\mathrm{K},\mathrm{V}) = \mathrm{D}^{-1}\mathrm{A}_{\mathtt{spar}}\mathrm{V} + \mathrm{D}^{-1}\mathrm{A}_{\mathtt{res}}\mathrm{V}. \tag{11}$$

Given D, the first term above can be computed in time $O(d \cdot \mathtt{nnz}(\mathrm{A}_{\mathtt{spar}}))$, where $\mathtt{nnz}(\cdot)$ denotes the number of nonzero entries of a matrix. By choosing an appropriate LSH we can ensure that $\mathtt{nnz}(\mathrm{A}_{\mathtt{spar}})$ is almost linear in n.

The second term in Eq. (11) can be approximated via AMM, similar to what was done in Algorithm 1, however, we need to be able to estimate the column norms of $\mathrm{D}^{-1}\mathrm{A}_{\mathtt{res}}$. Fortunately, by Eq. (10), we have $\left\|\mathrm{D}^{-1}\mathrm{A}^j_{\mathtt{res}}\right\|_2^2 = \left\|\mathrm{D}^{-1}\mathrm{A}^j\right\|_2^2 - \left\|\mathrm{D}^{-1}\mathrm{A}^j_{\mathtt{sparse}}\right\|_2^2$, where $\mathrm{A}^j_{\mathtt{res}}, \mathrm{A}^j, \mathrm{A}^j_{\mathtt{sparse}}$ denote the j^{th} columns of $\mathrm{A}_{\mathtt{res}}, \mathrm{A}, \mathrm{A}_{\mathtt{spar}}$, respectively. Since we can estimate the column norms of $\mathrm{D}^{-1}\mathrm{A}$ efficiently using WExpKDE and all column norms of $\mathrm{D}^{-1}\mathrm{A}_{\mathtt{spar}}$ can be computed

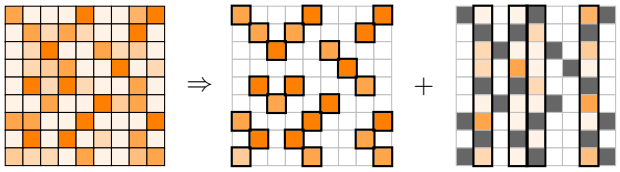


Figure 3. The softmax matrix $\mathrm{D}^{-1}\mathrm{A}$ decomposes into its sparse approximation $\mathrm{D}^{-1}\mathrm{A}_{\mathtt{spar}}$, which captures large entries (coded with darker colors), and the residual $\mathrm{D}^{-1}\mathrm{A}_{\mathtt{res}}$, where black cells represent entries captured by $\mathrm{D}^{-1}\mathrm{A}_{\mathtt{spar}}$. Blank colors in the matrix on the right side represent columns of $\mathrm{D}^{-1}\mathrm{A}_{\mathtt{res}}$ **not** sampled by AMM sampling matrix $\Pi_{\mathtt{res}}$ and the union of colored and blank columns together represent the whole residual matrix $\mathrm{D}^{-1}\mathrm{A}_{\mathtt{res}}$.

Algorithm 3 Practical Improvement of KDEformer

1: **input**: matrices $\mathrm{Q},\mathrm{K},\mathrm{V} \in \mathbb{R}^{n\times d}$, integer m, $\varepsilon > 0$, and LSH function $\mathcal{H}: \mathbb{R}^d \to [B]$
2: compute α, β, γ as per lines 2-4 of Algorithm 1
3: $p_j \leftarrow \beta_j - \sum_{i=1}^n \alpha_j^{-2} e^{\frac{2\langle q_i, k_j\rangle}{\sqrt{d}}} \mathbb{1}_{\{\mathcal{H}(q_i)=\mathcal{H}(k_j)\}} + \gamma \|v_j\|_2^2$ for every $j \in [n]$ then normalize $p_\ell \leftarrow \frac{p_\ell}{\sum_{j\in[n]} p_j}$ for every $\ell \in [n]$
4: generate the sampling matrix $\Pi_{\mathtt{res}}$ as per lines 6-7 of Algorithm 1 using distribution $\{p_j\}_{j\in[n]}$ computed above
5: **return** $\widetilde{\mathrm{D}} = \mathtt{diag}(\alpha)$ and $\Pi_{\mathtt{res}}$

in total $\mathtt{nnz}(\mathrm{A}_{\mathtt{spar}})$ time, the AMM sampling matrix for residual $\Pi_{\mathtt{res}}$ can be generated quickly.

Putting everything together, we first choose an appropriate LSH function $\mathcal{H}$ and compute the sparse approximation to the attention matrix as per Eq. (10). We show how to design a GPU-friendly LSH whose collision probability $\Pr[\mathcal{H}(q_i) = \mathcal{H}(k_j)]$ is roughly proportional to $\langle q_i, k_j \rangle$ in Appendix A. Next, we compute a spectral proxy $\widetilde{\mathrm{D}}$ for D, as was done efficiently in Algorithm 1. Finally, we perform AMM on matrices $\widetilde{\mathrm{D}}^{-1}\mathrm{A}_{\mathtt{res}}$ and V via a sampling matrix $\Pi_{\mathtt{res}}$. The resulting estimator is:

$$\widetilde{\mathrm{Att}} = \widetilde{\mathrm{D}}^{-1}\mathrm{A}_{\mathtt{spar}}\mathrm{V} + \widetilde{\mathrm{D}}^{-1}\mathrm{A}_{\mathtt{res}}\Pi_{\mathtt{res}}^{\top} \cdot \Pi_{\mathtt{res}}\mathrm{V}.$$

We illustrate this procedure in Fig. 3 and present the pseudocode for computing $\widetilde{\mathrm{D}}$ and $\Pi_{\mathtt{res}}$ in Algorithm 3. By an analysis similar to Corollary 3.6, we find that the runtime of Algorithm 3 is $O(m + \varepsilon^{-2}dn^{1+\tau+o(1)} + \mathtt{nnz}(\mathrm{A}_{\mathtt{spar}}))$ with some $m = O\left(\varepsilon^{-2}\log n \cdot \mathtt{srank}(\mathrm{D}^{-1}\mathrm{A}_{\mathtt{res}})\right)$.

4. Experiments

4.1. Single Self-attention Layer Approximation

We first benchmark our algorithm on approximating a single self-attention layer, i.e., $\mathrm{Att}(\mathrm{Q},\mathrm{K},\mathrm{V})$. We randomly select a pair of matrices $\mathrm{Q},\mathrm{V} \in \mathbb{R}^{n\times d}$ from the GloVe word

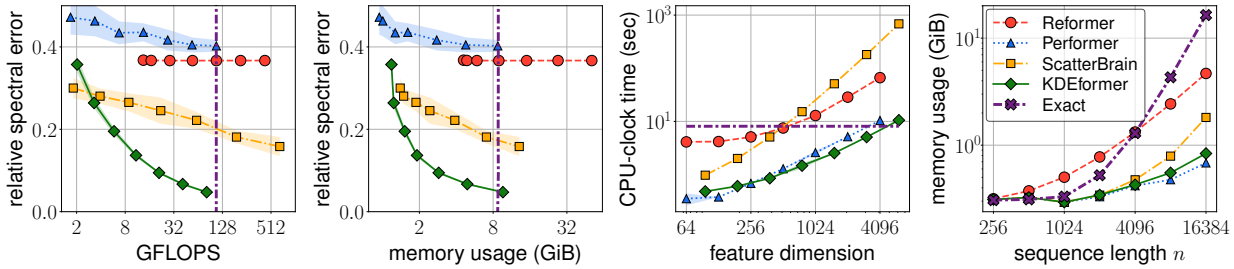


Figure 4. Performance evaluations of various self-attention approximations on approximating under the GloVe word embeddings.

embeddings (Pennington et al., 2014) with sequence length $n = 8{,}192$ and dimension $d = 100$ and set $\mathrm{K} = \mathrm{Q}$. We compare our KDEformer to other attention approximations including Reformer (Kitaev et al., 2020), Performer (Choromanski et al., 2021), and ScatterBrain (Chen et al., 2021b). We compute the relative error with respect to the operator norm, i.e., $\frac{\|\mathrm{Att}(\mathrm{Q},\mathrm{K},\mathrm{V})-\widetilde{\mathrm{Att}}\|_{\mathrm{op}}}{\|\mathrm{Att}(\mathrm{Q},\mathrm{K},\mathrm{V})\|_{\mathrm{op}}}$ where $\widetilde{\mathrm{Att}} \in \mathbb{R}^{n\times d}$ is an approximate attention. Additionally, we measure the peak memory usage, FLOP count, and CPU-clock time of approximation methods while varying hyperparameters of algorithms that affect runtime and memory.

In Fig. 4, we observe that our proposed algorithm achieves the lowest error with minimal FLOP count and memory usage. In particular, our approximation error can be about 9% with 3.06× memory reduction and 5.11× FLOPS reduction. In addition, we plot CPU-clock time for various choices of hyperparameters that determine peak memory usage. Specifically, if the approximation method requires at most nk memory usage for computing $\widetilde{\mathrm{Att}}$, we refer to k as the feature dimension. Having equal feature dimensions, our algorithm and Performer are the fastest methods, but Performer has significantly larger errors than the others. We fix the feature dimension $k = 128$ and measure the peak memory usage while the sequence length n is changing from 256 and 16,384. For $n = 16{,}384$, our method can save up to 19.62× memory space compared to the exact computation.

4.2. Image generation with pre-trained BigGAN

We next apply the above-mentioned attention approximations to the pre-trained BigGAN model (Brock et al., 2019) to generate synthetic images. The model contains a single attention layer where the corresponding inputs have different dimensions: $\mathrm{Q} \in \mathbb{R}^{4,096\times 64}$, $\mathrm{K} \in \mathbb{R}^{1,024\times 64}$ and $\mathrm{V} \in \mathbb{R}^{1,024\times 256}$. Following the experiments in (Chen et al., 2021b), we use the pre-trained BigGAN on ImageNet at 512×512 resolution and replace the attention layer with its approximations. Note that we do not perform training or fine-tuning at all. We generate 5,000 fake images and compute the Frechet Inception Distance (FID) with ImageNet validation set as ground truth and Inception Scores (IS) (Salimans et al., 2016). Lower FID and higher IS values imply better generation quality. We also measure the number of FLOPS in the attention layer. We set the hyperparameters (i.e., feature dimensions) so that all approximation methods have the same peak memory usage. The results are reported in Table 1. Interestingly, our algorithm shows a lower FID value than the exact attention with 4.14× fewer FLOPs. Although Performer is the fastest algorithm, its generated images look unnatural while our attention can generate more realistic images. A number of generated images by various methods can be found in Appendix C.1.

Table 1. Results on image generation using BigGAN with the exact attention and its approximations. Bold values indicate the best within the standard deviation.

Method	FID (↓)	IS (↑)	GFLOPS	
Exact	32.17	**58.38** ± 4.23	10.738	–
Reformer	72.39	19.04 ± 2.32	10.872	(0.99×)
Performer	33.39	37.32 ± 2.91	**1.682**	(6.38×)
ScatterBrain	38.55	36.43 ± 3.34	2.891	(3.71×)
KDEformer	**31.41**	**58.16** ± 4.04	2.596	(4.14×)

Table 2. Results on ImageNet classification using T2T-ViT with the exact attention and its approximations.

Method	Top-1 Accuracy (%)	GFLOPS	
Exact	**82.55**	161.10	–
Reformer	81.44	11.71	(13.75 ×)
Performer	80.50	**5.06**	(31.87 ×)
ScatterBrain	81.95	7.18	(22.43 ×)
KDEformer	82.08	8.80	(18.30 ×)

4.3. ImageNet classification with Vision Transformer

Next, we evaluate the attention approximations in the context of image classification with pre-trained Tokens-to-Token Vision Transformer (T2T-ViT) (Yuan et al., 2021). The model consists of Tokens-to-Token (T2T) module and the Vision Transformer (ViT) backbone. The T2T module consists of 2 attention layers with sequence lengths 3136 and 784, respectively, and the ViT module consists of 24 attention layers all of which have token lengths 197 (in addition, the embedding dimension of attention layers in the ViT is 512, larger than the sequence length). Hence, the two attention layers in T2T module dominate the overall computational and memory complexities because their sequence lengths are significantly larger. So, we exactly compute the

Table 3. Results on end-to-end training on 5 Long Range Arena (LRA) benchmark datasets.

	ListOps	Text	Image	Retrieval	Pathfinder	Average
Exact	33.32	60.22	37.41	81.07	70.25	56.45
Reformer	36.74	61.39	43.59	78.15	66.25	**57.22**
Performer	37.75	58.81	35.74	80.39	62.84	55.11
KDEformer	36.64	62.00	45.45	73.52	68.13	57.15

(a) Test accuracy (%)

	ListOps	Text	Image	Retrieval	Pathfinder	Average
Exact	6.53	16.71	9.41	8.72	4.70	9.21
Reformer	1.59	3.18	6.36	2.94	3.18	3.45
Performer	1.07	2.13	4.28	2.15	2.14	2.35
KDEformer	1.02	2.03	4.08	2.38	1.87	**2.28**

(b) Peak memory (GB)

	ListOps	Text	Image	Retrieval	Pathfinder	Average
Exact	0.133	0.479	0.276	0.478	0.141	0.301
Reformer	0.041	0.081	0.155	0.092	0.082	0.090
Performer	0.036	0.067	0.127	0.074	0.068	0.074
KDEformer	0.034	0.058	0.110	0.073	0.063	**0.068**

(c) Wall-clock time (sec) per batch

pre-trained model with 24 layers in ViT backbone and only apply the approximation methods to the 2 attention layers in the T2T module as a drop-in replacement. The dimensions of $\mathrm{Q}, \mathrm{K}, \mathrm{V}$ are all the same, $n = 3{,}136, d = 64$ in the first layer and $n = 784, d = 64$ in the second layer.

We compute top-1 accuracy on the ImageNet validation dataset and measure FLOPS in the first attention layer, which requires the most resources. The results are shown in Table 2. Observe that our method is the best among all approximate methods with 82.08% test accuracy. In particular, it leads to less than 1% performance drop compared to the exact computation but the required operations are $18.3\times$ fewer. Such performance gains would increase when token sequence lengths are larger.

4.4. End-to-end Training with Long Range Arena Benckmark

Finally, to demonstrate the power of our method in reducing the training time of transformer models, we run end-to-end training on the Long Range Arena benchmark (Tay et al., 2021), which contains 5 classification datasets, i.e., ListOps, Text, Image, Retrieval and Pathfinder. The maximum sequence lengths of these datasets are $2{,}048$, $4{,}096$, $1{,}024$, $4{,}096$ and $1{,}024$, respectively. We follow the same settings from (Chen et al., 2021c); model is a 2-layer transformer with 64 embedding dimension, 128 hidden dimension, 2 attention heads, and mean pooling is used for the classification task. Learning rate is set to 10^{-4} for Text, ListOps, Image and 2×10^{-4} for the rest. All models are trained for $50{,}000$ steps. Similar to Section 4.1, we choose hyperparameters of all methods having equal feature dimensions to 128.

In Table 3, we provide results on (a) test accuracy, (b) peak memory and (c) wall-clock time per batch of single training step (including forward and backward propagations). As a result, we observe that the proposed KDEformer achieves the second-best test accuracy in average followed by Reformer, but it requires much less memory as well as faster wall-clock time than other competitors. For example, KDEformer with Text dataset runs about $8\times$ faster than the exact attention.

5. Conclusion

We propose a fast attention approximation based on recent advances in KDE solvers. The proposed algorithm can run in strongly sub-quadratic time in sequence length and provide an error bound under the spectral norm. It shows promising performances under various practical applications involving long-sequence attention. We believe this can have a significant impact on other practical problems as well.

Acknowledgements

Insu Han and Amin Karbasi acknowledge funding in direct support of this work from NSF (IIS-1845032), ONR (N00014- 19-1-2406), and the AI Institute for Learning-Enabled Optimization at Scale (TILOS).

References

Backurs, A., Charikar, M., Indyk, P., and Siminelakis, P. Efficient density evaluation for smooth kernels. In *Foundations of Computer Science (FOCS)*, 2018. 2

Backurs, A., Indyk, P., and Wagner, T. Space and time efficient kernel density estimation in high dimensions. *Neural Information Processing Systems (NeurIPS)*, 2019. 2

Brock, A., Donahue, J., and Simonyan, K. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations (ICLR)*, 2019. 2, 8

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Neural Information Processing Systems (NeurIPS)*, 2020. 1

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision(ECCV)*, 2020. 1

Charikar, M. and Siminelakis, P. Hashing-based-estimators for kernel density in high dimensions. In *Foundations of Computer Science (FOCS)*, 2017. 2

Charikar, M., Kapralov, M., Nouri, N., and Siminelakis, P. Kernel density estimation through density constrained near neighbor search. In *Foundations of Computer Science (FOCS)*, 2020. 2, 3, 6

Chen, B., Liu, Z., Peng, B., Xu, Z., Li, J. L., Dao, T., Song, Z., Shrivastava, A., and Re, C. MONGOOSE: A learnable LSH framework for efficient neural network training. In *International Conference on Learning Representations (ICLR)*, 2020. 3

Chen, B., Dao, T., Liang, K., Yang, J., Song, Z., Rudra, A., and Re, C. Pixelated Butterfly: Simple and Efficient Sparse training for Neural Network Models. In *International Conference on Learning Representations (ICLR)*, 2021a. 1

Chen, B., Dao, T., Winsor, E., Song, Z., Rudra, A., and Re, C. Scatterbrain: Unifying sparse and low-rank attention. *Neural Information Processing Systems (NeurIPS)*, 2021b. 1, 3, 8, 18

Chen, Y., Zeng, Q., Ji, H., and Yang, Y. Skyformer: Remodel self-attention with gaussian kernel and nystr\" om method. *Neural Information Processing Systems (NeurIPS)*, 2021c. 9

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., et al. Rethinking Attention with Performers. In *International Conference on Learning Representations (ICLR)*, 2021. 1, 3, 8, 18

Daras, G., Kitaev, N., Odena, A., and Dimakis, A. G. Smyrf-efficient attention using asymmetric clustering. *Neural Information Processing Systems (NeurIPS)*, 2020. 1, 3

Dass, J., Wu, S., Shi, H., Li, C., Ye, Z., Wang, Z., and Lin, Y. Vitality: Unifying low-rank and sparse approximation for vision transformer acceleration with a linear taylor attention. *arXiv preprint arXiv:2211.05109*, 2022. 1

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Association for Computational Linguistics (NAACL)*, 2018. 1

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations (ICLR)*, 2021. 1

Hagerup, T., Mehlhorn, K., and Munro, J. I. Maintaining discrete probability distributions optimally. In *International Colloquium on Automata, Languages, and Programming*, 1993. 17

Joshi, S., Kommaraji, R. V., Phillips, J. M., and Venkatasubramanian, S. Comparing distributions and shapes using the kernel distance. In *Symposium on Computational Geometry (SOCG)*, 2011. 2

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning (ICML)*, 2020. 1, 3

Kitaev, N., Kaiser, L., and Levskaya, A. Reformer: The Efficient Transformer. In *International Conference on Learning Representations (ICLR)*, 2020. 1, 3, 6, 8, 12, 18

Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014. 7, 8

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research (JMLR)*, 2020. 1

Roy, A., Saffar, M., Vaswani, A., and Grangier, D. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics (ACL)*, 2021. 1, 3

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. In *Neural Information Processing Systems (NeurIPS)*, 2016. 8

Schölkopf, B., Smola, A. J., Bach, F., et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002. 2

Siminelakis, P., Rong, K., Bailis, P., Charikar, M., and Levis, P. Rehashing kernel evaluation in high dimensions. In *International Conference on Machine Learning (ICML)*, 2019. 2

Sun, Z., Yang, Y., and Yoo, S. Sparse Attention with Learning to Hash. In *International Conference on Learning Representations (ICLR)*, 2021. 1, 3

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena: A benchmark for efficient transformers. *International Conference on Learning Representations (ICLR)*, 2021. 3, 9

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. Efficient transformers: A survey. *ACM Computing Surveys*, 2022. 3

Tropp, J. A. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 2015. 14

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *Neural Information Processing Systems (NeurIPS)*, 2017. 1

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. *Neural Information Processing Systems (NeurIPS)*, 2019. 1

Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z.-H., Tay, F. E., Feng, J., and Yan, S. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *International Conference on Computer Vision (ICCV)*, 2021. 3, 7, 8, 18

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. Big bird: Transformers for longer sequences. *Neural Information Processing Systems (NeurIPS)*, 2020. 1

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Conference on Artificial Intelligence (AAAI)*, 2021. 1

Zouzias, A. *Randomized primitives for linear algebra and applications*. University of Toronto, 2013. 2, 4

A. Practical Angular LSH with Fixed Bucket Sizes

The practical version of our algorithm that we presented in Section 3.4 requires a locality sensitive hashing $\mathcal{H} : \mathbb{R}^d \to [B]$ for identifying the dominant entries of the attention matrix A, which correspond to pairs of keys and queries whose "angular distances" are small. In this section, we develop a simple yet effective and practical LSH function whose collision probability is related to the angular distance between hashed points.

While the lsh allows computing a very sparse approximation to the attention matrix, uneven bucket sizes hinder batching of the computations across lsh buckets. In fact, if we parallelize the computation across buckets, the largest bucket determines the runtime (Kitaev et al., 2020). Our proposed lsh function has equal-sized buckets, thus, it aligns with modern hardware's block-memory access and can be efficiently parallelized by batching across buckets.

We start by defining a simple LSH function whose collision probability is *roughly* proportional to the angle between the hashed points.

Definition A.1 (Angular LSH)**.** For positive integers d, r, let $w_1, w_2, \dots w_r$ be i.i.d. random samples from the tropical Gaussian distribution $\mathcal{N}(0, I_d)$. We define the *rank-r angular LSH* $h : \mathbb{R}^d \to \{0,1\}^r$ as follows:

$$h(x) := \left(\mathbb{1}_{\{w_1^\top x\}}, \mathbb{1}_{\{w_2^\top x\}}, \dots \mathbb{1}_{\{w_r^\top x\}}\right) \qquad \text{for any } x \in \mathbb{R}^d.$$

Note that the buckets are labeled by r-bit binary numbers and if $r \leq d$ then almost surely the total number of buckets is 2^r.

It is easy to calculate the collision probability of the angular lsh defined in Definition A.1.

Claim 1. *For positive integers r, d let $h(\cdot)$ be an instance of rank-r angular LSH as per Definition A.1. For any $x, y \in \mathbb{R}^d$ the collision probability of $h(x)$ and $h(y)$ is:*

$$\Pr[h(x) = h(y)] = \left(1 - \frac{\theta_{x,y}}{\pi}\right)^r,$$

where $\theta_{x,y} = \cos^{-1}\left(\frac{x^\top y}{\|x\| \cdot \|y\|}\right)$ denotes the angle between x and y.

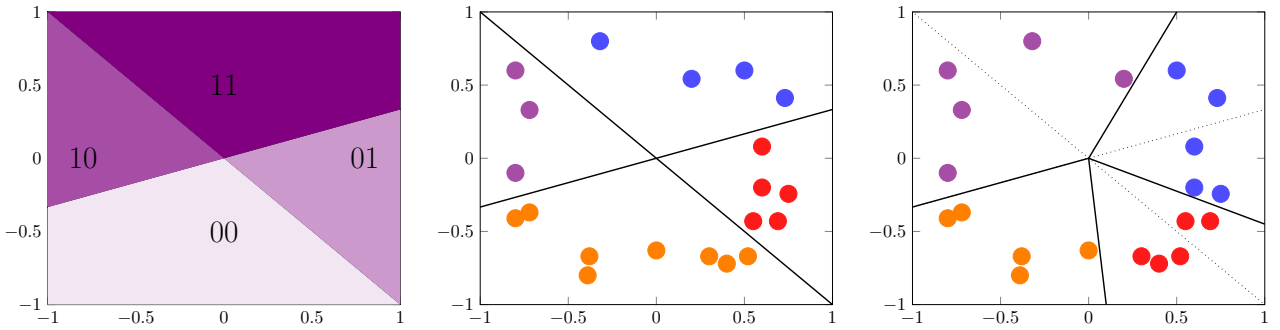
Therefore, the points with small angular distances are likely to be hashed to the same buckets while points with large angular distances are unlikely to be hashed to the same buckets.

So, if we hash keys k_j and queries q_i using the angular lsh given in Definition A.1 then the entries of the attention matrix A which correspond to colliding pairs of keys and queries will likely have very large values. As we mentioned earlier, the main efficiency bottleneck in this lsh-based approach for computing the dominant entries of the attention matrix is the unevenness of hash bucket sizes. If we try to compute the sparse approximation to A, as defined in Eq. (10), using the lsh function from Definition A.1 by parallelizing the computation across buckets, the runtime will be dominated by the time to compute entries in the largest bucket.

One solution for increasing efficiency, which was proposed in (Kitaev et al., 2020), is to truncate the lsh buckets and force them to contain equal number of keys and queries. However, truncation can degrade the quality of approximation drastically because there will be *spillover* from one bucket to another, and some points can be forced into far-away buckets. The reason for this spillover effect is the fact that consecutive buckets in a hash table do not necessarily represent areas of the $\mathbb{R}^d$ space which are geometrically close to each other.

We show that in fact, it is possible to sort the buckets of the angular lsh from Definition A.1 such that the order of buckets reflects their geometrical position, thus, consecutive buckets actually represent neighboring partitions of $\mathbb{R}^d$. It turns out that the geometric distance between two buckets of this lsh function translates into the Hamming distance between their binary labels.

To be precise, for any binary numbers $b_1, b_2 \in \{0,1\}^r$ let $d_H(b_1, b_2) \in [r+1]$ represent the *Hamming distance* between the two, i.e., the number of bits where b_1 and b_2 differ. Now note that the lsh buckets in Definition A.1 are labeled with r-bit binary numbers. Each bit in the binary representations of buckets corresponds to a partitioning of the $\mathbb{R}^d$ into two sides of a random hyperplane whose normal vector is sampled from a tropical Gaussian. Therefore, if we have two buckets b_1 and b_2 with hamming distance $d_H(b_1, b_2) = 1$ then these buckets are positioned on the same sides of all random hyperplanes except for one, thus, they represent neighboring regions in $\mathbb{R}^d$ and the hyperplanes corresponding to the differing bit of b_1 and b_2 is the boundary between two regions.



(a) Space partitions by angular LSH (b) Hashing an example dataset (c) Buckets truncation in Hamming distance order

Figure 5. Rank-2 Angular LSH in action (in dimension $d = 2$). The space partitions corresponding to buckets with unit Hamming distance are neighbors in $\mathbb{R}^d$. In Fig. 5(b) we hash an example dataset and we get uneven buckets. Fig. 5(c) show that if we order the dataset according to the Hamming distance of their buckets and then truncate the buckets we get new equal-sized buckets with minimal spillover effect.

We show this fact in Fig. 5(a), which illustrates the space partitions corresponding to the buckets of a rank-2 angular lsh in dimension $d = 2$. It is clearly visible that the bucket labels of neighboring partitions have unit Hamming distance. In Fig. 5(b) we hash an example dataset using this LSH function and as can be seen, the buckets have uneven sizes. Because of the relationship between the Hamming distance of bucket labels and the distance between space partitions, if we order the dataset according to the Hamming ordering of their buckets and then truncate them we get new buckets with even sizes and minimal spillover effect. In particular, in Fig. 5(c) we order the dataset such that the points from buckets $00, 01, 11, 10$ come in this specific order and then we bin the data points by partitioning the ordered dataset into equal-sized parts. The resulting bins show no spillover effect.

In the following lemma we show how to order r-bit binary numbers $\{0, 1\}^r$ such that all consecutive numbers have unit Hamming distance:

Lemma A.2 (Ordering of binary numbers according to their Hamming distance)**.** *For any positive integer r it is possible to order the set of binary numbers $\{0, 1\}^r$ as a sequence $b_1, b_2, \dots b_{2^r}$ such that for any $j \in [2^r - 1]$:*

$$d_H(b_j, b_{j+1}) = 1.$$

Proof. The proof is by induction. For $r = 1$ the base of induction follows trivially. Now suppose that we have the sequence of $(r-1)$-bit numbers $b'_1, b'_2, \dots b'_{2^{r-1}}$ such that $d_H(b'_j, b'_{j+1}) = 1$ for any $j \in [2^{r-1} - 1]$. Then the sequence of r-bit numbers will be as follows:

$$b_j := \begin{cases} (b'_j,\ 0) & \text{if } j \le 2^{r-1} \\ (b'_{2^r+1-j},\ 1) & \text{if } j > 2^{r-1} \end{cases} \qquad \text{for } j \in [2^r].$$

One can verify that this sequence satisfies the desired property and the proof is complete. □

Therefore, we can use the angular LSH together with the ordering of binary numbers from Lemma A.2 to construct an effective hash function with equal-sized buckets.

Definition A.3 (Equal-sized LSH with Minimal Spillover)**.** Suppose that we want to hash a dataset $x_1, x_2, \dots x_n \in \mathbb{R}^d$.

1. Hash these points using a rank-r Angular LSH $h(\cdot)$ as per Definition A.1.
2. Then, using Lemma A.2, produce an ordering of r-bit binary numbers such that consecutive numbers have unit Hamming distance; let $b_1, b_2, \dots b_{2^r}$ be such ordering.
3. Next, define a permutation $\mathcal{P} \in \mathtt{Sym}(n)$ which orders the dataset according to the Hamming ordering of their buckets. More specifically, $\mathcal{P}$ satisfies:

$$\mathcal{P}(i) < \mathcal{P}(j) \quad \text{iff } h(x_i) \le_* h(x_j), \text{ where the inequality } \le_* \text{ acts with respect to the ordering } b_1, b_2, \dots b_{2^r}.$$

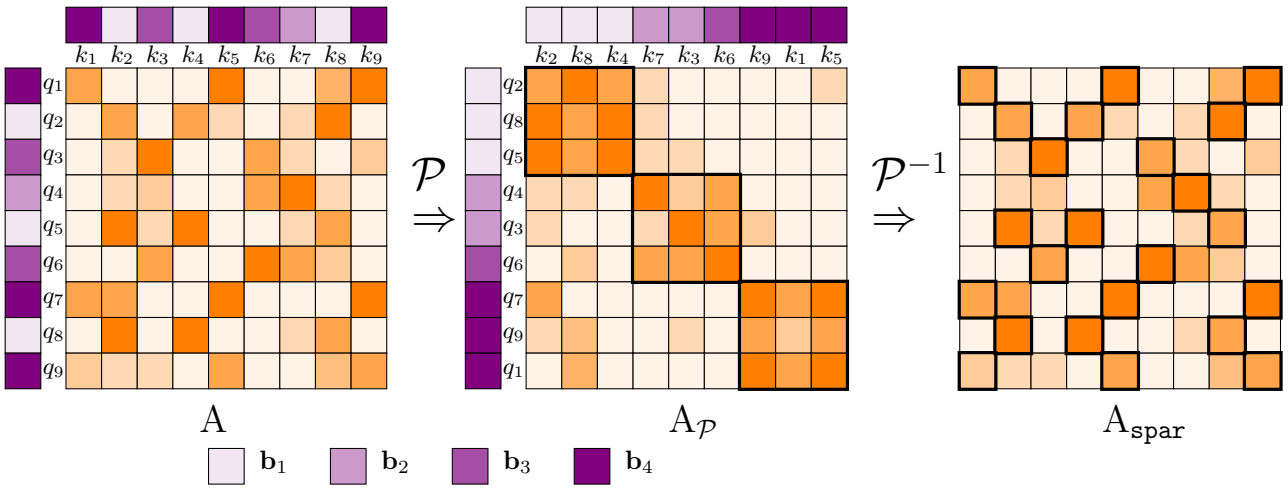


Figure 6. An example of how $\mathrm{A}_{\mathtt{spar}}$ can be computed efficiently. (Left) keys and queries are hashed using the angular lsh function. buckets are represented by shades of violet. (Middle) keys and queries are permuted such that their buckets are sorted according to the Hamming distance ordering. Large entries of the permuted attention matrix $\mathrm{A}_{\mathcal{P}}$ are concentrated around the diagonal blocks, so we compute the diagonal blocks. (Right) the block diagonal approximation to $\mathrm{A}_{\mathcal{P}}$ is reverse permuted to obtain $\mathrm{A}_{\mathtt{spar}}$.

4. Permute $x_1, x_2, \ldots x_n$ according to $\mathcal{P}$ and then partition the sequence into equal-sized chunks. These chunks are the buckets.

Now we explain how we can use the lsh procedure given in Definition A.3 to compute $\mathrm{A}_{\mathtt{spar}}$ as per Eq. (10) through an example shown in Fig. 6. We first hash keys k_j and queries q_i via the angular lsh. We represent the buckets of this hashing via different shades of violet in Fig. 6. Clearly, the bucket sizes are uneven. Then we permute keys and queries via $\mathcal{P}$ which orders the points such that their buckets are sorted according to the ordering b_1, b_2, b_3, b_4 obtained from Lemma A.2. Then we truncate the sorted points which is in fact equivalent to selecting blocks along the diagonal of the permuted attention matrix. The selected diagonal blocks in Fig. 6 illustrate this. Finally, we can reverse the permutation on the rows and columns of the block diagonal attention which gives us the final $\mathrm{A}_{\mathtt{spar}}$.

B. Omitted Proofs

B.1. Proof of Lemma 3.2: Approximate Matrix Multiplication via Sampling

In this section, we analyze the random sampling method for approximately computing the product of two rectangular matrices, presented in Lemma 3.2. The proof of this lemma is based on the following version of the matrix Bernstein inequality.

Lemma B.1 (Matrix Approximation by Random Sampling, Corollary 6.2.1 from (Tropp, 2015))**.** *Let* B *be a fixed* $q \times d$ *matrix. Construct a* $q \times d$ *random matrix* R *that satisfies*

$$\mathbb{E}[\mathrm{R}] = \mathrm{B}, \quad \textit{and} \quad \|\mathrm{R}\|_{\mathrm{op}} \leq L.$$

Compute the per-sample second moment:

$$m_2(\mathrm{R}) = \max\{\|\mathbb{E}[\mathrm{R}^*\mathrm{R}]\|_{\mathrm{op}}, \|\mathbb{E}[\mathrm{R}\mathrm{R}^*]\|_{\mathrm{op}}\}.$$

Form the matrix sampling estimator

$$\overline{\mathrm{R}}_m = \frac{1}{m}\sum_{i=1}^{m} \mathrm{R}_i \qquad \textit{where each } \mathrm{R}_i \textit{ is an independent copy of } \mathrm{R}.$$

Then for every $t > 0$*, the estimator satisfies*

$$\Pr\left[\left\|\overline{\mathrm{R}}_m - \mathrm{B}\right\|_{\mathrm{op}} \geq t\right] \leq (q+d) \cdot \exp\left(\frac{-mt^2/2}{m_2(\mathrm{R}) + 2Lt/3}\right).$$

Now we prove Lemma 3.2 by invoking the above matrix Bernstein inequality.

Lemma 3.2 (Approximate Matrix Multiplication (AMM))**.** *For any matrices* $\mathrm{X} \in \mathbb{R}^{n \times q}, \mathrm{Y} \in \mathbb{R}^{n \times d}$ *and any probability distribution* $\{p_i\}_{i \in [n]}$ *which satisfies* $p_i \geq \frac{1}{4} \cdot \frac{\|x_i\|_2^2 + \gamma \cdot \|y_i\|_2^2}{\|\mathrm{X}\|_F^2 + \gamma \cdot \|\mathrm{Y}\|_F^2}$ *for all* $i \in [n]$ *and* $\gamma = \|\mathrm{X}\|_{\mathrm{op}}^2 / \|\mathrm{Y}\|_{\mathrm{op}}^2$*, a sampling matrix* $\Pi \in \mathbb{R}^{m \times n}$ *constructed by first generating* m *i.i.d. samples* $\ell_1, \ell_2, \ldots \ell_m \in [n]$ *according to* $\{p_\ell\}_{\ell \in [n]}$ *and then letting the* r^{th} *row of* Π *be* $\frac{1}{\sqrt{m \cdot p_{\ell_r}}} \cdot e_{\ell_r}^\top$*, if* $m = \Omega\left(\varepsilon^{-2} \log n \cdot (\mathtt{srank}(\mathrm{X}) + \mathtt{srank}(\mathrm{Y}))\right)$ *for some* $\varepsilon > 0$*, the following holds,*

$$\Pr\left[\left\|\mathrm{X}^\top \Pi^\top \Pi \mathrm{Y} - \mathrm{X}^\top \mathrm{Y}\right\|_{\mathrm{op}} > \varepsilon \|\mathrm{X}\|_{\mathrm{op}} \|\mathrm{Y}\|_{\mathrm{op}}\right] \leq \frac{1}{\mathrm{poly}(n)}.$$

Proof. First we let $\mathrm{B} := \mathrm{X}^\top \mathrm{Y}$. Then we let the random matrix R have the following distribution

$$\Pr\left[\mathrm{R} = \frac{x_i^\top \cdot y_i}{p_i}\right] = p_i \quad \text{for } i \in [n]$$

where x_i and y_i are i^{th} row vector in X and Y, respectively. With this definition we have,

$$\mathbb{E}[\mathrm{R}] = \sum_{i \in [n]} \frac{x_i^\top \cdot y_i}{p_i} \cdot p_i = \sum_{i \in [n]} x_i^\top \cdot y_i = \mathrm{X}^\top \mathrm{Y} = \mathrm{B}.$$

Furthermore, we can bound the operator norm of R as follows,

$$\begin{aligned}
\|\mathrm{R}\|_{\mathrm{op}} &\leq \max_{i \in [n]} \frac{\left\|x_i^\top \cdot y_i\right\|_{\mathrm{op}}}{p_i} \\
&= \max_{i \in [n]} \frac{\|x_i\|_2 \|y_i\|_2}{p_i} \\
&\leq 4 \cdot \max_{i \in [n]} \frac{\|x_i\|_2 \|y_i\|_2 \cdot \left(\|\mathrm{X}\|_F^2 + \gamma \cdot \|\mathrm{Y}\|_F^2\right)}{\|x_i\|_2^2 + \gamma \cdot \|y_i\|_2^2} \\
&\leq 2 \cdot \max_{i \in [n]} \frac{1}{\sqrt{\gamma}} \cdot \|\mathrm{X}\|_F^2 + \sqrt{\gamma} \cdot \|\mathrm{Y}\|_F^2 \\
&= 2 \|\mathrm{X}\|_{\mathrm{op}} \cdot \|\mathrm{Y}\|_{\mathrm{op}} \cdot (\mathtt{srank}(\mathrm{X}) + \mathtt{srank}(\mathrm{Y})) \equiv L,
\end{aligned}$$

where the third line above follows from the precondition of Lemma 3.2 about the distribution $\{p_i\}_{i \in [n]}$ and the fourth line follows from AM-GM inequality. The last line follows from the definition of γ and definition of stable rank. Next, we will compute the per-sample second moment as follows,

$$\begin{aligned}
\mathbb{E}[\mathrm{R}^* \mathrm{R}] &= \sum_{i \in [n]} \|x_i\|_2^2 \cdot \frac{y_i^\top \cdot y_i}{p_i^2} \cdot p_i = \sum_{i \in [n]} \|x_i\|_2^2 \cdot \frac{y_i^\top \cdot y_i}{p_i} \\
&\preceq 4 \cdot \left(\|\mathrm{X}\|_F^2 + \gamma \cdot \|\mathrm{Y}\|_F^2\right) \cdot \sum_{i \in [n]} \frac{\|x_i\|_2^2}{\|x_i\|_2^2 + \gamma \cdot \|y_i\|_2^2} \cdot y_i^\top y_i \\
&\preceq 4 \cdot \left(\|\mathrm{X}\|_F^2 + \gamma \cdot \|\mathrm{Y}\|_F^2\right) \cdot \sum_{i \in [n]} y_i^\top y_i = 4 \cdot \left(\|\mathrm{X}\|_F^2 + \gamma \cdot \|\mathrm{Y}\|_F^2\right) \cdot \mathrm{Y}^\top \mathrm{Y}.
\end{aligned}$$

Similarly,

$$\mathbb{E}[\mathrm{R}\mathrm{R}^*] \preceq 4 \cdot \left(\|\mathrm{X}\|_F^2 / \gamma + \|\mathrm{Y}\|_F^2\right) \cdot \mathrm{X}^\top \mathrm{X}.$$

In summary,

$$\begin{aligned}
m_2(\mathrm{R}) &= \max\{\|\mathbb{E}[\mathrm{R}^* \mathrm{R}]\|_{\mathrm{op}}, \|\mathbb{E}[\mathrm{R}\mathrm{R}^*]\|_{\mathrm{op}}\} \\
&\leq 4 \cdot \max\left\{\left(\|\mathrm{X}\|_F^2 + \gamma \cdot \|\mathrm{Y}\|_F^2\right) \cdot \left\|\mathrm{Y}^\top \mathrm{Y}\right\|_{\mathrm{op}}, \left(\|\mathrm{X}\|_F^2 / \gamma + \|\mathrm{Y}\|_F^2\right) \cdot \left\|\mathrm{X}\mathrm{X}^\top\right\|_{\mathrm{op}}\right\} \\
&= 4 \cdot \|\mathrm{X}\|_{\mathrm{op}}^2 \|\mathrm{Y}\|_{\mathrm{op}}^2 \cdot (\mathtt{srank}(\mathrm{X}) + \mathtt{srank}(\mathrm{Y})).
\end{aligned}$$

Finally, we note that, from the way the sampling matrix was constructed we have $\mathrm{X}^\top \Pi^\top \Pi \mathrm{Y} = \frac{1}{m}\sum_{r\in[m]} \frac{x_{\ell_r}\cdot y_{\ell_r}}{p_{i_r}} = \overline{\mathrm{R}}_m$. Thus, by invoking Lemma B.1 we find that for $t = \varepsilon \cdot \|\mathrm{X}\|_{\mathrm{op}} \|\mathrm{Y}\|_{\mathrm{op}}$ we have,

$$\Pr\left[\left\|\overline{\mathrm{R}}_m - \mathrm{B}\right\|_{\mathrm{op}} \geq \varepsilon \cdot \|\mathrm{X}\|_{\mathrm{op}} \|\mathrm{Y}\|_{\mathrm{op}}\right] \leq (q+d)\cdot \exp\left(\frac{-mt^2/2}{m_2(\mathrm{R}) + 2Lt/3}\right) \leq \frac{1}{\mathrm{poly}(n)}.$$

This completes the proof of Lemma 3.2. □

B.2. Proof of Theorem 3.3

Theorem 3.3 (Correctness of Algorithm 1)**.** *For any matrices $\mathrm{Q}, \mathrm{K}, \mathrm{V} \in \mathbb{R}^{n\times d}$, any $\varepsilon > 0$, and number of samples $m = \Omega\left(\varepsilon^{-2}\log n \cdot (\mathtt{srank}(\mathrm{D}^{-1}\mathrm{A}) + \mathtt{srank}(\mathrm{V}))\right)$, given access to a primitive* WEXPKDE *as per Definition 3.1, Algorithm 1 outputs a diagonal matrix $\widetilde{\mathrm{D}} \in \mathbb{R}^{n\times n}$ and a sampling matrix $\Pi \in \mathbb{R}^{m\times n}$ which satisfy Eq. (1) with probability at least $1 - \frac{1}{\mathrm{poly}(n)}$.*

Proof. First, note that all entries of $\mathrm{D}^{-1}\mathrm{A}$ are positive and the sum of entries of each row of this matrix equals 1, so by the Gershgorin circle theorem $\left\|\mathrm{D}^{-1}\mathrm{A}\right\|_{\mathrm{op}} \leq 1$. On the other hand, $\mathrm{D}^{-1}\mathrm{A}\cdot \mathbf{1}_n = \mathbf{1}_n$, so we have $\left\|\mathrm{D}^{-1}\mathrm{A}\right\|_{\mathrm{op}} = 1$. We will use this fact in the rest of the proof.

Now note that Algorithm 1 computes $\alpha = \text{WEXPKDE}\left(\frac{\mathrm{K}}{d^{1/4}}, \frac{\mathrm{Q}}{d^{1/4}}, \mathbf{1}_n, \frac{\varepsilon}{3}\right)$ in line 3 and lets $\widetilde{\mathrm{D}} = \mathtt{diag}(\alpha)$. Thus, as we showed earlier, by Definition 3.1 and using the fact that entries of D are positive, we have $(1-\varepsilon/3)\mathrm{D} \preceq \widetilde{\mathrm{D}} \preceq (1+\varepsilon/3)\mathrm{D}$. So, using this inequality along with the fact that $\left\|\mathrm{D}^{-1}\mathrm{A}\right\|_{\mathrm{op}} = 1$, the diagonal matrix $\widetilde{\mathrm{D}}$ satisfies Eq. (5).

Next, let us consider the vector $\beta = \text{WEXPKDE}\left(\frac{\sqrt{2}\cdot\mathrm{Q}}{d^{1/4}}, \frac{\sqrt{2}\cdot\mathrm{K}}{d^{1/4}}, u, 1/3\right)$ computed in line 4. For ease of notation, let $\mathrm{X}^\top := \widetilde{\mathrm{D}}^{-1}\mathrm{A}$. By Definition 3.1 and using the definition of $u_i = 1/\alpha_i^2$ in line 3, we have,

$$\beta_j \in (1\pm 1/3)\cdot \sum_{i\in[n]} u_i \cdot \exp\left(\frac{2}{\sqrt{d}}\langle q_i, k_j\rangle\right) = (1\pm 1/3)\cdot \|x_j\|_2^2 \quad \text{for any } j\in[n].$$

Also, note that γ which is computed in line 2 of the algorithm is equal to $\gamma = \frac{\left\|\mathrm{D}^{-1}\mathrm{A}\right\|_{\mathrm{op}}^2}{\|\mathrm{V}\|_{\mathrm{op}}^2}$. Because $(1-\varepsilon/3)\mathrm{D} \preceq \widetilde{\mathrm{D}} \preceq (1+\varepsilon/3)\mathrm{D}$, we have $\gamma \in (1\pm\varepsilon/3)^{-1}\cdot\tilde{\gamma}$, where $\tilde{\gamma} := \left\|\widetilde{\mathrm{D}}^{-1}\mathrm{A}\right\|_{\mathrm{op}}^2 / \|\mathrm{V}\|_{\mathrm{op}}^2$. Therefore, the distribution $\{p_i\}_{i\in[n]}$ computed in line 5 satisfies,

$$p_\ell = \frac{\beta_\ell + \gamma\cdot\|v_\ell\|_2^2}{\sum_{j\in[n]}\beta_j + \gamma\cdot\|\mathrm{V}\|_F^2} \geq \frac{1}{4}\cdot\frac{\|x_\ell\|_2^2 + \tilde{\gamma}\cdot\|v_\ell\|_2^2}{\|\mathrm{X}\|_F^2 + \tilde{\gamma}\cdot\|\mathrm{V}\|_F^2}.$$

Furthermore, note that $\mathtt{srank}(\widetilde{\mathrm{D}}^{-1}\mathrm{A}) \leq 2\cdot\mathtt{srank}(\mathrm{D}^{-1}\mathrm{A})$. Therefore, we can invoke the AMM result from Lemma 3.2 with matrices $\mathrm{X}^\top = \widetilde{\mathrm{D}}^{-1}\mathrm{A}$ and $\mathrm{Y} = \mathrm{V}$ and use the precondition of Theorem 3.3 about the number of samples $m = \Omega\left(\varepsilon^{-2}\log n\cdot(\mathtt{srank}(\mathrm{D}^{-1}\mathrm{A}) + \mathtt{srank}(\mathrm{V}))\right) = \Omega\left(\varepsilon^{-2}\log n\cdot(\mathtt{srank}(\widetilde{\mathrm{D}}^{-1}\mathrm{A}) + \mathtt{srank}(\mathrm{V}))\right)$ to conclude that the sampling matrix Π computed in lines 6-7 satisfies the following with high probability in n:

$$\left\|\widetilde{\mathrm{D}}^{-1}\mathrm{A}\Pi^\top\cdot\Pi\mathrm{V} - \widetilde{\mathrm{D}}^{-1}\mathrm{A}\mathrm{V}\right\|_{\mathrm{op}} \leq \frac{\varepsilon}{4}\left\|\widetilde{\mathrm{D}}^{-1}\mathrm{A}\right\|_{\mathrm{op}}\|\mathrm{V}\|_{\mathrm{op}} \leq \frac{\varepsilon}{2}\left\|\mathrm{D}^{-1}\mathrm{A}\right\|_{\mathrm{op}}\|\mathrm{V}\|_{\mathrm{op}} = \frac{\varepsilon}{2}\|\mathrm{V}\|_{\mathrm{op}},$$

where the second inequality above follows from the fact that $\left\|\widetilde{\mathrm{D}}^{-1}\mathrm{A}\right\|_{\mathrm{op}} \leq 2\cdot\left\|\mathrm{D}^{-1}\mathrm{A}\right\|_{\mathrm{op}}$. The above inequality shows that Eq. (6) holds with high probability in n. Thus the theorem follows from combining Eq. (5) and Eq. (6) using triangle inequality. □

B.3. Proof of Theorem 3.5

Theorem 3.5 (Approximate Attention with Spectral Norm Bound)**.** *For any matrices $\mathrm{Q}, \mathrm{K}, \mathrm{V} \in \mathbb{R}^{n\times d}$, any $\varepsilon > 0$, and given a fast Gaussian KDE as per Theorem 2.1, there exists an algorithm that outputs a diagonal matrix $\widetilde{\mathrm{D}} \in \mathbb{R}^{n\times n}$ and a*

sampling matrix $\Pi \in \mathbb{R}^{m\times n}$ *with* $m = O\left(\varepsilon^{-2}\log n \cdot (\texttt{srank}(\mathrm{D}^{-1}\mathrm{A}) + \texttt{srank}(\mathrm{V}))\right)$ *samples which satisfy Eq. (1) with probability at least* $1 - \frac{1}{\mathrm{poly}(n)}$. *The runtime of this algorithm is* $O\left(m + nd \cdot \left(\mathcal{C}_{\frac{\mathrm{K}}{d^{1/4}}, \frac{\mathrm{Q}}{d^{1/4}}, \mathbf{1}_n, \varepsilon, \tau} + \mathcal{C}_{\frac{\sqrt{2}\cdot\mathrm{Q}}{d^{1/4}}, \frac{\sqrt{2}\cdot\mathrm{K}}{d^{1/4}}, v, 1, \tau}\right)\right)$, *where* $v_j = \left(\sum_{\ell\in[n]} \exp\left(\frac{1}{\sqrt{d}}\langle q_j, k_\ell\rangle\right)\right)^{-2}$ *for* $j \in [n]$ *and* $\mathcal{C}_{\frac{\mathrm{K}}{d^{1/4}}, \frac{\mathrm{Q}}{d^{1/4}}, \mathbf{1}_n, \varepsilon, \tau}, \mathcal{C}_{\frac{\sqrt{2}\cdot\mathrm{Q}}{d^{1/4}}, \frac{\sqrt{2}\cdot\mathrm{K}}{d^{1/4}}, v, 1, \tau}$ *are defined as in Eq. (9).*

Proof. It suffices to run Algorithm 1 with some $m = O\left(\varepsilon^{-2}\log n(\texttt{srank}(\mathrm{D}^{-1}\mathrm{A}) + \texttt{srank}(\mathrm{V}))\right)$ samples and invoke Algorithm 2 for the calls to WEXPKDE made in lines 3-4. By Theorem 3.3 and Theorem 3.4 along with union bound, the outputs Π and $\widetilde{\mathrm{D}}$ of this procedure satisfy the desired condition of Eq. (1) with probability $\geq 1 - \frac{1}{\mathrm{poly}(n)}$.

Runtime Analysis. By Theorem 3.4, the time to compute $\widetilde{\mathrm{D}}$ through invoking WEXPKDE (i.e., Algorithm 2) in line 3 of Algorithm 1 is $O\left(nd \cdot \mathcal{C}_{\frac{\mathrm{K}}{d^{1/4}}, \frac{\mathrm{Q}}{d^{1/4}}, \mathbf{1}_n, \varepsilon, \tau}\right)$. Furthermore, time to run WEXPKDE in line 4 is $O\left(nd \cdot \mathcal{C}_{\frac{\sqrt{2}\cdot\mathrm{Q}}{d^{1/4}}, \frac{\sqrt{2}\cdot\mathrm{K}}{d^{1/4}}, u, 1, \tau}\right)$, where u is the vector computed in lines 3-4 of Algorithm 1. On the other hand, by Theorem 3.4, vector u satisfies $\frac{1}{2}v_j \leq u_j \leq \frac{3}{2}v_j$ for all $j \in [n]$ with probability at least $1 - \frac{1}{\mathrm{poly}(n)}$, where v is the vector defined in the theorem statement. Thus, using the definition of $\mathcal{C}_{\frac{\sqrt{2}\cdot\mathrm{Q}}{d^{1/4}}, \frac{\sqrt{2}\cdot\mathrm{K}}{d^{1/4}}, u, 1, \tau}$ in Eq. (9) we can show that the aforementioned runtime is bounded by $O\left(nd \cdot \mathcal{C}_{\frac{\sqrt{2}\cdot\mathrm{Q}}{d^{1/4}}, \frac{\sqrt{2}\cdot\mathrm{K}}{d^{1/4}}, v, 1, \tau}\right)$.

Finally, the time to generate m samples in line 6 of Algorithm 1 is $O(m+n)$, using the sampling method developed by Hagerup et al. (1993). The total runtime is obtained by summing up these terms. □

B.4. Proof of Corollary 3.6

Corollary 3.6 (Simplified Runtime for Bounded Diameter Datasets)**.** *For any datasets* Q, K *with diameter* $\max_{i,j\in[n]} \|k_i - q_j\|_2^2 = \gamma\sqrt{d}\log n$ *for some* $\gamma > 0$, *the runtime of Theorem 3.5 is upper bounded by* $O\left(m + nd \cdot \left(n^{\tau(1+\gamma)} + \varepsilon^{-2}n^{\tau(1+\gamma/2)}\right)\right)$, *which is strongly sub-quadratic in* n. *In particular, if* $\gamma = o(1)$, *the runtime is bounded by* $O\left(m + \varepsilon^{-2}d \cdot n^{1+\tau+o(1)}\right)$.

Proof. First recall that the diameter of the datasets Q, K is $\max_{i,j\in[n]} \|k_i - q_j\|_2^2 = \gamma\sqrt{d}\log n$ for some $\gamma > 0$. For any $i, j \in [n]$, using the fact that $\|k_i - q_j\|_2^2 \leq \gamma\sqrt{d}\log n$, we have,

$$\begin{aligned}\exp\left(\frac{1}{\sqrt{d}}\langle k_j, q_i\rangle\right) &= \exp\left(\frac{-1}{2\sqrt{d}}\|k_j - q_i\|_2^2\right) \cdot \exp\left(\frac{\|k_j\|^2 + \|q_i\|^2}{2\sqrt{d}}\right) \\ &\geq n^{-\gamma/2} \cdot \exp\left(\frac{\|k_j\|^2 + \|q_i\|^2}{2\sqrt{d}}\right).\end{aligned}$$

Therefore, summing the above inequality over all $j \in [n]$ gives,

$$\sum_{j\in[n]} \exp\left(\frac{1}{\sqrt{d}}\langle k_j, q_i\rangle\right) \geq n^{-\gamma/2} \cdot \sum_{j\in[n]} \exp\left(\frac{\|k_j\|^2 + \|q_i\|^2}{2\sqrt{d}}\right).$$

The above inequality holds for every $i \in [n]$. This inequality implies that the following set is empty for any $\mu \leq n^{-1-\gamma/2}$,

$$\left\{i \in [n] : \frac{\sum_{j\in[n]} \exp\left(\frac{1}{\sqrt{d}}\langle k_j, q_i\rangle\right)}{\sum_{j\in[n]} \exp\left(\frac{\|k_j\|^2 + \|q_i\|^2}{2\sqrt{d}}\right)} < n \cdot \mu\right\} = \emptyset.$$

Thus, $\mathcal{C}_{\frac{\mathrm{K}}{d^{1/4}},\frac{\mathrm{Q}}{d^{1/4}},\mathbf{1}_n,\varepsilon,\tau}$ defined as per Eq. (9) is bounded as follows,

$$\begin{aligned}\mathcal{C}_{\frac{\mathrm{K}}{d^{1/4}},\frac{\mathrm{Q}}{d^{1/4}},\mathbf{1}_n,\varepsilon,\tau} &= \min_{\mu>0} \varepsilon^{-2}\mu^{-\tau} + \left|\left\{ i \in [n] : \frac{\sum_{j\in[n]} \exp\left(\frac{1}{\sqrt{d}}\langle k_j, q_i\rangle\right)}{\sum_{j\in[n]} \exp\left(\frac{\|k_j\|^2+\|q_i\|^2}{2\sqrt{d}}\right)} < n\mu \right\}\right| \\ &\leq \varepsilon^{-2}\cdot n^{\tau(1+\gamma/2)}.\end{aligned}$$

Similarly, because $v_j > 0$ for every $j \in [n]$, we can show that, for any $i \in [n]$,

$$\sum_{j\in[n]} v_j \exp\left(\frac{2}{\sqrt{d}}\langle q_j, k_i\rangle\right) \geq n^{-\gamma}\cdot \sum_{j\in[n]} v_j \exp\left(\frac{\|q_j\|^2+\|k_i\|^2}{\sqrt{d}}\right).$$

As a result, the following set is empty for any $\mu \leq n^{-1-\gamma}$,

$$\left\{ i \in [n] : \frac{\sum_{j\in[n]} v_j \cdot \exp\left(\frac{2}{\sqrt{d}}\langle q_j, k_i\rangle\right)}{\sum_{j\in[n]} v_j \exp\left(\frac{\|q_j\|^2+\|k_i\|^2}{\sqrt{d}}\right)} < n\cdot\mu \right\} = \emptyset.$$

So, $\mathcal{C}_{\frac{\sqrt{2}\cdot\mathrm{Q}}{d^{1/4}},\frac{\sqrt{2}\cdot\mathrm{K}}{d^{1/4}},v,1,\tau}$ defined as per Eq. (9) is bounded as follows,

$$\begin{aligned}\mathcal{C}_{\frac{\sqrt{2}\cdot\mathrm{Q}}{d^{1/4}},\frac{\sqrt{2}\cdot\mathrm{K}}{d^{1/4}},v,1,\tau} &= \min_{\mu>0} \mu^{-\tau} + \left|\left\{ i \in [n] : \frac{\sum_{j\in[n]} v_j \cdot \exp\left(\frac{2}{\sqrt{d}}\langle q_j, k_i\rangle\right)}{\sum_{j\in[n]} v_j \exp\left(\frac{\|q_j\|^2+\|k_i\|^2}{\sqrt{d}}\right)} < n\cdot\mu \right\}\right| \\ &\leq n^{\tau(1+\gamma)}.\end{aligned}$$

Therefore, the total runtime of Theorem 3.5 is bounded by

$$O\left(m + nd\cdot\left(\mathcal{C}_{\frac{\mathrm{K}}{d^{1/4}},\frac{\mathrm{Q}}{d^{1/4}},\mathbf{1}_n,\varepsilon,\tau} + \mathcal{C}_{\frac{\sqrt{2}\cdot\mathrm{Q}}{d^{1/4}},\frac{\sqrt{2}\cdot\mathrm{K}}{d^{1/4}},v,1,\tau}\right)\right) = O\left(m + nd\cdot\left(n^{\tau(1+\gamma)} + n^{\tau(1+\gamma/2)}/\varepsilon^2\right)\right),$$

which completes the proof. □

C. Additional Experiments

C.1. BigGAN Image Generations

Images in Fig. 7 are randomly subset from $2,000$ generations from BigGAN (Yuan et al., 2021)[1] with the exact attention computation and its various approximations including KDEformer (our), Performer (Choromanski et al., 2021), Reformer (Kitaev et al., 2020) and ScatterBrain (Chen et al., 2021b). One can observe that our KDEformer generates more natural and realistic images than other methods by a large margin, and in many cases it is even better than the exact computation. This means that it has much less running time and memory, but it has produced a higher quality and more realistic image in the end. Also, note that the hyperparameters of our approach were not fine-tuned.

[1] https://github.com/huggingface/pytorch-pretrained-BigGAN

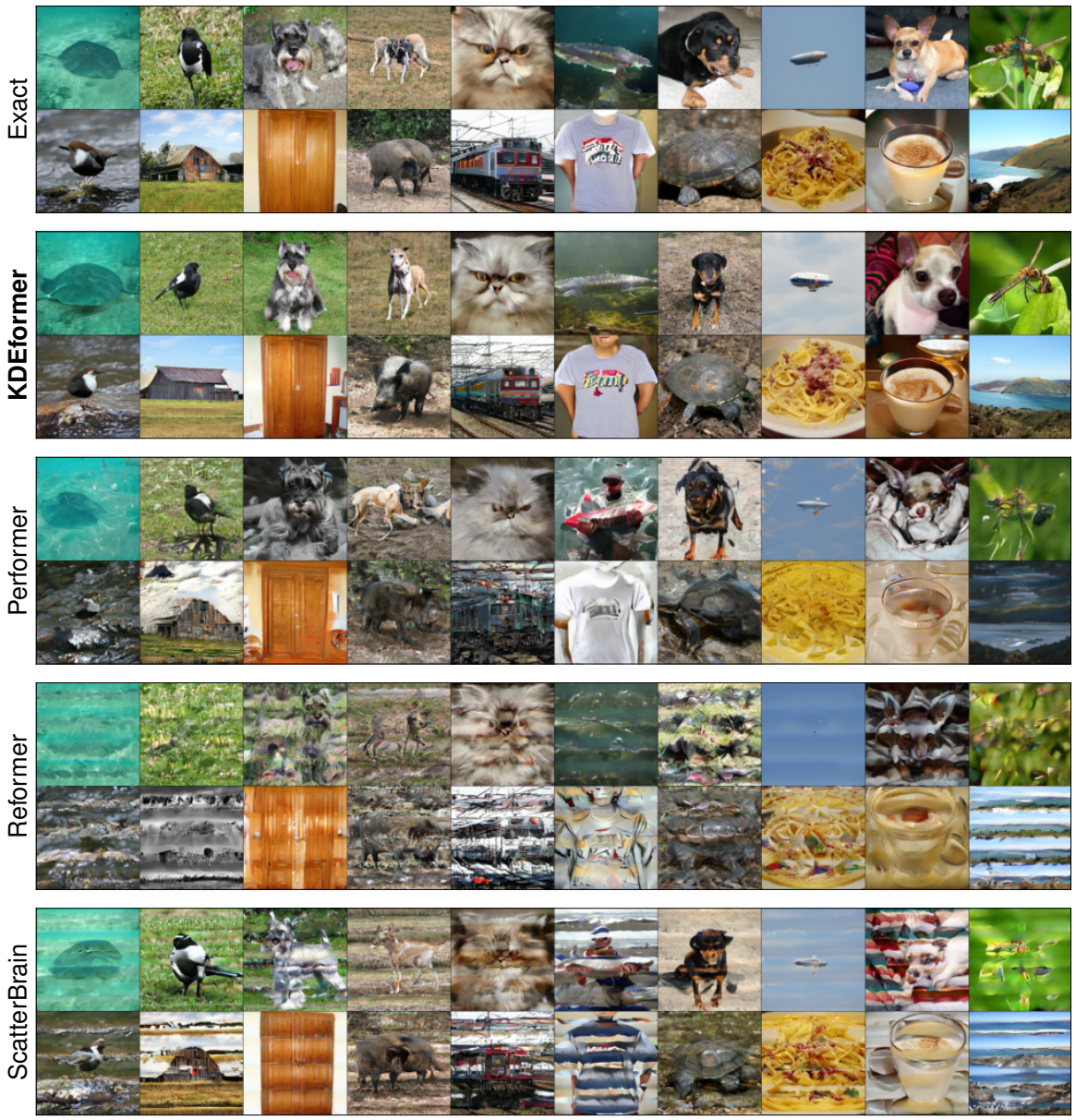


Figure 7. Images generations from the pre-trained BigGAN with the exact attention (top) and drop-in replacement with its approximations including our KDEformer (second row), Performer (third row), Reformer (fourth row) and ScatterBrain (bottom).