Transformer with a Mixture of Gaussian Keys

Tam Nguyen[†,⋆] Tan M. Nguyen[‡,⋆] Dung Le[†] Khuong Nguyen[†] Anh Tran[†]
Richard G. Baraniuk[◇] Nhat Ho[○,⋆⋆] Stanley J. Osher[‡,⋆⋆]

FPT Software, Vietnam[†]
University of California, Los Angeles, USA[‡]
Rice University, Houston, USA[◇]
University of Texas, Austin, USA[○]
October 17, 2021

Abstract

Multi-head attention is a driving force behind state-of-the-art transformers which achieve remarkable performance across a variety of natural language processing (NLP) and computer vision tasks. It has been observed that for many applications, those attention heads learn redundant embedding, and most of them can be removed without degrading the performance of the model. Inspired by this observation, we propose Transformer with a Mixture of Gaussian Keys (Transformer-MGK), a novel transformer architecture that replaces redundant heads in transformers with a mixture of keys at each head. These mixtures of keys follow a Gaussian mixture model and allow each attention head to focus on different parts of the input sequence efficiently. Compared to its conventional transformer counterpart, Transformer-MGK accelerates training and inference, has fewer parameters, and requires less FLOPs to compute while achieving comparable or better accuracy across tasks. Transformer-MGK can also be easily extended to use with linear attentions. We empirically demonstrate the advantage of Transformer-MGK in a range of practical applications including language modeling and tasks that involve very long sequences. On the Wikitext-103 and Long Range Arena benchmark, Transformer-MGKs with 4 heads attain comparable or better performance to the baseline transformers with 8 heads.

1 Introduction

Transformers [65] have become the state-of-the-art model for sequence processing tasks, solving many challenging problems in natural language processing and computer vision [1, 13, 70, 16, 7, 25, 50, 15, 56, 17, 63]. These models can also transfer the learned knowledge from a pre-trained model to task that involves different data modalities and has limited supervision [48, 49, 16, 72, 34]. The success of transformers is rooted in the self-attention mechanism as their fundamental building blocks for modeling [9, 43, 32]. For each token, self-attention computes a weighted average of the feature representations of other tokens where the weight is proportional to a similarity score between each pair of tokens. This mechanism allows a token to pay attention to other tokens in the sequence and attain a contextual representation [4, 65, 28]. It has been shown that the representation capacity of the attention mechanism [62] and its capability of capturing diverse syntactic and semantic relationships [62, 66, 12, 67, 23] is keyed to the impressive performance of transformers in practice.

⋆ Tan M. Nguyen and Tam Nguyen are co-first authors of this paper. Please correspond to: tanmnguyen89@ucla.edu and nguyenminhtam9520@gmail.com.

⋆⋆ Nhat Ho and Stanley J. Osher are co-last authors of this paper.

1.1 Self-Attention

For a given input sequence $\mathbf{X} := [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D_x}$ of N feature vectors, self-attention transforms $\mathbf{X}$ into the output sequence $\mathbf{H}$ in the following two steps:

Step 1. The input sequence $\boldsymbol{X}$ is projected into the query matrix $\mathbf{Q}$, the key matrix $\mathbf{K}$, and the value matrix $\mathbf{V}$ via three linear transformations

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q^\top; \mathbf{K} = \mathbf{X}\mathbf{W}_K^\top; \mathbf{V} = \mathbf{X}\mathbf{W}_V^\top,$$

where $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{D \times D_x}$, and $\mathbf{W}_V \in \mathbb{R}^{D_v \times D_x}$ are the weight matrices. We denote $\boldsymbol{Q} := [\boldsymbol{q}_1, \cdots, \boldsymbol{q}_N]^\top, \mathbf{K} := [\boldsymbol{k}_1, \cdots, \boldsymbol{k}_N]^\top$, and $\mathbf{V} := [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N]^\top$, where the vectors $\boldsymbol{q}_i, \boldsymbol{k}_i, \boldsymbol{v}_i$ for $i = 1, \cdots, N$ are the query, key, and value vectors, respectively.

Step 2. The output sequence $\mathbf{H} := [\boldsymbol{h}_1, \cdots, \boldsymbol{h}_N]^\top$ is then computed as follows

$$\mathbf{H} = \text{softmax}\Big(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D}}\Big)\mathbf{V} := \mathbf{A}\mathbf{V}, \tag{1}$$

where the softmax function is applied to each row of the matrix $(\mathbf{Q}\mathbf{K}^\top)/\sqrt{D}$. For each query vector $\boldsymbol{q}_i$ for $i = 1, \cdots, N$, an equivalent form of Eqn. (1) to compute the output vector $\boldsymbol{h}_i$ is given by

$$\boldsymbol{h}_i = \sum_{j=1}^{N} \text{softmax}\Big(\frac{\boldsymbol{q}_i^\top \boldsymbol{k}_j}{\sqrt{D}}\Big)\boldsymbol{v}_j := \sum_{j=1}^{N} a_{ij}\boldsymbol{v}_j. \tag{2}$$

The matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ and its component a_{ij} for $i, j = 1, \cdots, N$ are the attention matrix and attention scores, respectively. The self-attention computed by Eqn. (1) and (2) is called the scaled dot-product attention or softmax attention. In our paper, we call a transformer that uses this attention the softmax transformer. The structure that the attention matrix $\mathbf{A}$ learns from training determines the ability of the self-attention to capture contextual representation for each token.

Multi-head Attention Each output sequence $\mathbf{H}$ forms an attention head. In multi-head attention, multiple heads are concatenated to compute the final output. Let H be the number of heads and $\mathbf{W}^O \in \mathbb{R}^{HD_v \times HD_v}$ be the projection matrix for the output. The multi-head attention is defined as

$$\text{MultiHead}(\{\mathbf{Q}\}_{i=1}^H, \{\mathbf{K}\}_{i=1}^H, \{\mathbf{V}\}_{i=1}^H) = \text{Concat}(\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_H)\mathbf{W}^O. \tag{3}$$

Even though multi-head attention extends single-head attention to capture diverse attention patterns and improve the performance of transformers, it has been shown that transformers for practical tasks including sequence classification and language modeling learn redundant heads [38]. These redundant heads compute similar attention mappings. Having many of them in the model limits the representation capacity of the transformer while wasting parameters, memory and computation, impeding the application of transformers to many important large-scale tasks.

1.2 Contribution

We establish the correspondence between self-attention in transformer and a Gaussian mixture model (GMM) and propose Transformer with a Mixture of Gaussian Keys (Transformer-MGK), a novel class of transformers that can avoid the head redundancy. At the core of Transformer-MGK is replacing the attention key $\boldsymbol{k}_j$ in each head by a GMM to allow the query $\boldsymbol{q}_i$, as well as its

associated token, to attend to more diverse positions in the input sequence, thereby increasing the representation of each attention head and reducing the chance of learning redundant heads. In summary, our contribution is four-fold:

1. We construct a GMM and show that attention scores in self-attention match posterior distribution in our model, providing a probabilistic framework to study self-attention in transformers.
2. Under our probabilistic framework for self-attention, we introduce an additional mixture of Gaussian to model each attention key. We empirically show that this mixture of Gaussian keys (MGK) can capture a diversity of attention patterns, thus alleviating head redundancy.
3. We extend our MGK to use with linear attentions and propose the mixture of linear keys (MLK) for efficient computation and better memory footprint.
4. We empirically show that Transformer-MGK and Transformer-MLK are comparable or better than the corresponding baseline transformers with softmax and linear attentions while only using half the number of attention heads and reducing both model complexity measured by the number of parameters and computational cost in terms of FLOPs.

1.3 Organization

We structure this paper as follows: In Section 2, we establish the connection between GMM and self-attention and then present our Transformer-MGK and its extensions including Transformer-MLK. In Section 3, we validate and empirically analyze the efficiency and accuracy of Transformer-MGK/MLK. We discuss related works in Section 4. The paper ends up with concluding remarks. More experimental details are provided in the Appendix.

2 Transformer with a Mixture of Gaussian Keys

2.1 Attention Score as a Posterior Distribution

We first consider a query $\boldsymbol{q}_i \in \mathbf{Q}$ and a key $\boldsymbol{k}_j \in \mathbf{K}$. Let $\boldsymbol{t}$ be a K-dimensional binary random variable having a 1-of-K representation in which a particular element $\boldsymbol{t}_j$ is equal to 1 and all other elements are equal to 0. We use $\boldsymbol{t}_j$ to indicate the position j of the key $\boldsymbol{k}_j$. In particular, let $\mathbf{I}$ be the unit matrix, we model the distribution $\mathbb{P}(\boldsymbol{q}_i|\boldsymbol{t}_j = 1)$ by the following Gaussian distribution:

$$\mathbb{P}(\boldsymbol{q}_i|\boldsymbol{t}_j = 1) = \mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_j, \sigma_j^2\mathbf{I}). \tag{4}$$

Let π_j be the prior $\mathbb{P}(\boldsymbol{t}_j = 1)$. Given the query $\boldsymbol{q}_i$, how likely $\boldsymbol{q}_i$ matches the key $\boldsymbol{k}_j$ is given by posterior $\mathbb{P}(\boldsymbol{t}_j = 1|\boldsymbol{q}_i)$. This posterior is computed as follows

$$\begin{aligned}
\mathbb{P}(\boldsymbol{t}_j = 1|\boldsymbol{q}_i) &= \frac{\pi_j \mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_j, \sigma_j^2)}{\sum_{j'} \pi_{j'} \mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_{j'}, \sigma_{j'}^2)} \\
&= \frac{\pi_j \exp\left(-\|\boldsymbol{q}_i - \boldsymbol{k}_j\|^2/2\sigma_j^2\right)}{\sum_{j'} \pi_{j'} \exp\left(-\|\boldsymbol{q}_i - \boldsymbol{k}_{j'}\|^2/2\sigma_{j'}^2\right)} \\
&= \frac{\pi_j \exp\left[-\left(\|\boldsymbol{q}_i\|^2 + \|\boldsymbol{k}_j\|^2\right)/2\sigma_j^2\right] \exp\left(\boldsymbol{q}_i\boldsymbol{k}_j^\top/\sigma_j^2\right)}{\sum_{j'} \pi_{j'} \exp\left[-\left(\|\boldsymbol{q}_i\|^2 + \|\boldsymbol{k}_{j'}\|^2\right)/2\sigma_{j'}^2\right] \exp\left(\boldsymbol{q}_i\boldsymbol{k}_{j'}^\top/\sigma_{j'}^2\right)}.
\end{aligned} \tag{5}$$

We further assume that the query $\boldsymbol{q}_i$ and the key $\boldsymbol{k}_j$ are normalized, and the prior π_j is uniform. We will justify these assumptions in our Remarks at the end of this section. We also let $\sigma_j^2 = \sigma^2$, $j = 1, 2, \ldots, K$. Then the posterior $\mathbb{P}(\boldsymbol{t}_j = 1|\boldsymbol{q}_i)$ can be written as

$$\mathbb{P}(\boldsymbol{t}_j = 1|\boldsymbol{q}_i) = \frac{\exp\left(\boldsymbol{q}_i \boldsymbol{k}_j^\top / \sigma^2\right)}{\sum_{j'} \exp\left(\boldsymbol{q}_i \boldsymbol{k}_{j'}^\top / \sigma^2\right)}. \tag{6}$$

The right-hand side of Eqn. (6) matches the attention score given in Eqn. (2) when $\sigma^2 = \sqrt{D}$. Thus, we show that under right assumptions, the attention score between the query $\boldsymbol{q}_i$ and the key $\boldsymbol{k}_j$ in an attention unit of a transformer is the posterior $p(\boldsymbol{t}_j = 1|\boldsymbol{q}_i)$, which indicates the *responsibility* that the key $\boldsymbol{k}_j$ takes for 'explaining' the query $\boldsymbol{q}_i$, which in turn decides, for example, how much a token at position i pays attention to a token at position j in the input sequence.

Remark. *The assumption that the query $\boldsymbol{q}_i$ and the key $\boldsymbol{k}_j$ are normalized is realistic and not artificial. In many applications, those two vectors are normalized. [54] points out that such normalization is to avoid instability occurring during the training.*

Remark. *In practice, the prior is chosen to be uniform when there is no prior knowledge available.*

2.2 Transformer with a Mixture of Gaussian Keys: Each Key is Again a Gaussian Mixture Model

As we have seen from Eqn. (6), the key k_j is used to explain the query q_i via the posterior $\mathbb{P}(t_j = 1|q_i)$. Via this simple connection, each key k_j at position j is modelled as a Gaussian distribution $\mathcal{N}(k_j, \sigma_j^2 \mathbf{I})$. To further improve the explanation power of each key k_j, increase the representation of each attention head, and reduce the chance of learning redundant heads, we would like to model it as mixture of Gaussian distributions. We refer to this model as *Transformer with a Mixture of Gaussian Keys* (Transformer-MGK). In particular, in Transformer-MGK we model each key $\boldsymbol{k}_j$ at position j as a mixture of M Gaussians $\mathcal{N}(\boldsymbol{k}_{jr}, \sigma_{jr}^2 \mathbf{I})$, $r = 1, 2, \ldots, M$. Here we are abusing the notation a little bit and use $\boldsymbol{k}_{jr}$ and $\sigma_{jr}^2 \mathbf{I}$ to denote the mean and covariance matrix of the r^{th} Gaussian at position j. Let $\boldsymbol{z}$ be a M-dimensional binary random variable having a 1-of-M representation. We use $\boldsymbol{z}_r$ to indicate the r^{th} Gaussian in the mixture. Let $\pi_{jr} \equiv \mathbb{P}(\boldsymbol{z}_r = 1|\boldsymbol{t}_j = 1)$, our MGK can be written as

$$\mathbb{P}(\boldsymbol{q}_i|\boldsymbol{t}_j = 1) = \sum_r \mathbb{P}(\boldsymbol{z}_r = 1|\boldsymbol{t}_j = 1)\mathbb{P}(\boldsymbol{q}_i|\boldsymbol{z}_r = 1, \boldsymbol{t}_j = 1) = \sum_r \pi_{jr} \mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_{jr}, \sigma_{jr}^2 \mathbf{I}). \tag{7}$$

Similar to the derivation above, the posterior $p(\boldsymbol{t}_j = 1|\boldsymbol{q}_i)$ in Transformer-MGK can be written as

$$\mathbb{P}(\boldsymbol{t}_j = 1|\boldsymbol{q}_i) = \frac{\sum_r \pi_{jr} \exp\left(\boldsymbol{q}_i \boldsymbol{k}_{jr}^\top / \sigma_{jr}^2\right)}{\sum_{j'} \sum_r \pi_{j'r} \exp\left(\boldsymbol{q}_i \boldsymbol{k}_{j'r}^\top / \sigma_{j'r}^2\right)}. \tag{8}$$

Furthermore, in Transformer-MGK, we relax the assumption that the queries and keys are normalized. Thus, when computing $\mathbb{P}(\boldsymbol{t}_j = 1|\boldsymbol{q}_i)$, we compute the Gaussian kernels between the queries and keys instead of their dot products. The posterior $\mathbb{P}(\boldsymbol{t}_j = 1|\boldsymbol{q}_i)$ in Transformer-MGK is then given by

$$\mathbb{P}(\boldsymbol{t}_j = 1|\boldsymbol{q}_i) = \frac{\sum_r \pi_{jr} \exp\left(-\|\boldsymbol{q}_i - \boldsymbol{k}_{jr}\|^2 / 2\sigma_{jr}^2\right)}{\sum_{j'} \sum_r \pi_{j'r} \exp\left(-\|\boldsymbol{q}_i - \boldsymbol{k}_{j'r}\|^2 / 2\sigma_{j'r}^2\right)}. \tag{9}$$

As proven in Section 2.1, this posterior corresponds to the attention score. Thus, Eqn. (9) is the formula for computing the attention score in Transformer-MGK. We compute the output vector $\boldsymbol{h}_i$ of the self-attention in Transformer-MGK as follows

$$\boldsymbol{h}_i = \sum_j \left(\frac{\sum_r \pi_{jr} \exp\left(-\|\boldsymbol{q}_i - \boldsymbol{k}_{jr}\|^2/2\sigma_{jr}^2\right)}{\sum_{j'}\sum_r \pi_{j'r} \exp\left(-\|\boldsymbol{q}_i - \boldsymbol{k}_{j'r}\|^2/2\sigma_{j'r}^2\right)} \right) \boldsymbol{v}_j. \tag{10}$$

2.3 Inference and Learning via the Expectation Maximization Algorithm

Let $\gamma_{ir} \equiv \mathbb{P}(\boldsymbol{z}_r = 1|\boldsymbol{q}_i, \boldsymbol{t}_j = 1)$, in MGK, we apply the E-step inference in the Expectation-Maximization (EM) algorithm to estimate this posterior given the query $\boldsymbol{q}_i$. The posterior γ_{ir} is also known as the *responsibility* that the component $\mathcal{N}(\boldsymbol{k}_{jr}, \sigma_{jr}^2\mathbf{I})$ takes to account for the observation, which in MGK is the query $\boldsymbol{q}_i$. Below we propose two approaches to estimate this responsibility.

Soft E-step Using soft E-step inference, the EM algorithm makes a soft assignment, in which each query is associated with all clusters. The responsibilities are then given by

$$\gamma_{ir} = \frac{\pi_{jr} \exp\left(-\|\boldsymbol{q}_i - \boldsymbol{k}_{jr}\|^2/2\sigma_{jr}^2\right)}{\sum_{r'} \pi_{jr'} \exp\left(-\|\boldsymbol{q}_i - \boldsymbol{k}_{jr'}\|^2/2\sigma_{jr'}^2\right)}. \tag{11}$$

At learning, the responsibilities estimated by Eqn. (11) are used to update the prior π_{jr}, i.e. $\pi_{jr} = N_{jr}/N$, where N is the number of queries and $N_{jr} = \sum_{i=1}^N \gamma_{ir}$. These updated priors π_{jr} are then used in Eqn. (9) to compute attention scores.

Hard E-step Hard E-step performs a hard assignment of queries to key clusters, in which each query is associated uniquely with one cluster. This is similar to the K-means algorithm [35] and corresponds to the MGK at the limit when the variance parameter σ_{jr}^2 goes to 0. Following the derivation of K-means from a GMM in [6], Eqn. (9) becomes

$$\mathbb{P}(\boldsymbol{t}_j = 1|\boldsymbol{q}_i) = \frac{\max_r \exp\left(-\|\boldsymbol{q}_i - \boldsymbol{k}_{jr}\|^2/2\sigma_{jr}^2\right)}{\sum_{j'} \max_r \exp\left(-\|\boldsymbol{q}_i - \boldsymbol{k}_{j'r}\|^2/2\sigma_{j'r}^2\right)}. \tag{12}$$

Remark. *The hard E-step inference allows the attention score to be computed more efficiently because the priors π_{jr} no longer play an active role in the algorithm and can be completely ignored.*

Learning via Stochastic Gradient Descent (SGD) In the M-step of the EM algorithm, the cluster mean $\boldsymbol{k}_{jr}$ and variance σ_{jr}^2 can be updated as follows:

$$\boldsymbol{k}_{jr}^{\text{new}} = \frac{1}{N_{jr}} \sum_{i=1}^N \gamma_{ir} \boldsymbol{q}_i, \;\; \sigma_{jr}^{2\,\text{new}} = \frac{1}{N_{jr}} \sum_{i=1}^N \gamma_{ir} (\boldsymbol{q}_i - \boldsymbol{k}_{jr}^{\text{new}})^\top (\boldsymbol{q}_i - \boldsymbol{k}_{jr}^{\text{new}}). \tag{13}$$

The M-step update in Eqn. (13) is computationally costly when using with transformers. In order to increase the efficiency of the model, in MGK, we fix the variance parameter σ_{jr}^2 to be $\sqrt{D}$ as in the standard softmax attention and make only the keys $\boldsymbol{k}_{jr}$ learnable. We also make the prior π_{jr} learnable parameters as one of the design options. In that case, both $\boldsymbol{k}_{jr}$ and π_{jr} are learned via SGD. This update via SGD can be considered as a generalized M-step [6].

Design Options for Keys We follow the standard setting in the softmax transformer and make the keys $\boldsymbol{k}_{jr}$ a linear projection of the input $\boldsymbol{x}$, i.e. $\boldsymbol{k}_{jr} = \boldsymbol{x}\mathbf{W}_{k_{jr}}$. Alternatively, we also make the keys $\boldsymbol{k}_{jr}$ shifted version of each other to save computation, i.e. $\boldsymbol{k}_{jr} = \boldsymbol{x}\mathbf{W}_{k_j} + \boldsymbol{b}_r$.

2.4 Transformer with a Mixture of Linear Keys

The MGK can be easily extended to use with linear attentions. We call that model Transformer with a Mixture of Linear Keys (Transformer-MLK). In this section, we adopt the formulation of linear attentions from [27] to derive Transformer-MLK. Similar approach can be taken to derive Transformer-MLK when using with other linear attentions such as those in performers [11] and fast-weight transformers [54]. In Transformer-MLK, the Gaussian kernel in Eqn. (10) is linearized as the product of feature maps $\phi(\cdot)$ on the vectors $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$. The associative property of matrix multiplication is then utilized to derive the following efficient computation of the attention map

$$\boldsymbol{h}_i = \frac{\sum_j \sum_r \pi_{jr} \phi(\boldsymbol{q}_i)^\top \phi(\boldsymbol{k}_{jr}) \boldsymbol{v}_j}{\sum_j \sum_r \pi_{jr} \phi(\boldsymbol{q}_i)^\top \phi(\boldsymbol{k}_{jr})} = \frac{\phi(\boldsymbol{q}_i)^\top \sum_j \sum_r \pi_{jr} \phi(\boldsymbol{k}_{jr}) \boldsymbol{v}_j^\top}{\phi(\boldsymbol{q}_i)^\top \sum_j \sum_r \pi_{jr} \phi(\boldsymbol{k}_{jr})}. \tag{14}$$

Replacing $\sum_j \sum_r \pi_{jr} \phi(\boldsymbol{q}_i)^\top \phi(\boldsymbol{k}_{jr}) \boldsymbol{v}_j$ with $\phi(\boldsymbol{q}_i)^\top \sum_j \sum_r \pi_{jr} \phi(\boldsymbol{k}_{jr}) \boldsymbol{v}_j^\top$, as in linear transformers, reduces the memory and computational cost of computing the attention map in Transformer-MLK from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$, making Transformer-MLK scalable to very long sequences.

3 Experimental Results

In this section, we numerically justify the efficiency of Transformer-MGK/MLK and empirically study the advantage of using mixture of keys on various benchmarks, including different tasks in the Long Range Arena (LRA) [61] (Section 3.1) and language modeling on Wikitext-103 [37] (Section 3.2). We aim to show that: (i) Transformer-MGK/MLK with half the number of heads is comparable or better than the baseline softmax and linear transformers with full the number of heads while being more efficient in both computational cost and memory footprints; (ii) Mixture of keys helps reduce the redundancy in multi-head transformers and benefits learning of the long-term dependency in long input sequences; (iii) Using the same number of heads, Transformer-MGK/MLK significantly outperforms the baseline softmax and linear transformers. Especially in the case of Transformer-MLK, it helps reduce the performance gap between softmax and linear transformers while still maintaining linear memory and computational complexities.

Throughout this section, we compare Transformer-MGK/MLK with the softmax and linear transformers that have the same or double the number of attention heads. Among the design options for Transformer-MGK mentioned in Section 2.3, we use the one with Soft-E step but make the parameter π_{jr} and $\boldsymbol{k}_{jr}$ learnable and fix the variance σ_{jr}^2 to be constants. We study both implementations for keys: (A) $\boldsymbol{k}_{jr}$ is a linear projection of the input $\boldsymbol{x}$, i.e., $\boldsymbol{k}_{jr} = \boldsymbol{x}\mathbf{W}_{k_{jr}}$ and (B) $\boldsymbol{k}_{jr}$ are shifted version of each other, i.e., $\boldsymbol{k}_{jr} = \boldsymbol{x}\mathbf{W}_{k_j} + \boldsymbol{b}_r$.

In this section, we refer to the Transformer-MGK/MLK whose keys are implemented by (A) as Transformer-MGK/MLK, and whose keys are implemented by (B) as Transformer-sMGK/sMLK. We empirically compare these models with other design options for Transformer-MGK in Section 3.3. All experiments are conducted on a server with 4 NVIDIA A100 GPUs. Details on datasets, models, and training are provided in the Appendix. Code and instructions to reproduce the experiments are available at https://github.com/minhtannguyen/transformer-mgk.

3.1 Long Range Arena (LRA) Benchmark

Datasets and metrics We consider the following tasks in the LRA benchmark: Listops [40], byte-level IMDb reviews text classification [36], and byte-level document retrieval [47]. These tasks involve long sequences of length $2K$, $4K$, and $4K$, respectively. We follow the setup/evaluation protocol in [61] and report the test accuracy for individual task and the average result across all tasks.

Table 1: Test Accuracy (%) of Transformer-MGK compared with the baseline softmax transformer on the LRA benchmark. Our Transform-MGKs outperform softmax transformers while using half the number of heads, having less parameters, and requiring less FLOPs (see Figure 3 for more details). Results are averaged over 5 runs with different random seeds.

Model	ListOps	Text	Retrieval	Average
Softmax 8 heads	37.03	**65.71**	81.74	61.49
Transformer-sMGK 4 heads	**37.25**	65.51	**82.79**	**61.85**
Transformer-MGK 4 heads	36.98	65.69	82.23	61.63
Softmax 4 heads	36.89	65.26	81.54	61.23
Transformer-sMGK 2 heads	**37.35**	65.17	**82.20**	**61.57**
Transformer-MGK 2 heads	36.88	**65.37**	81.83	61.36
Softmax 2 heads	36.76	64.90	79.1	60.25
Transformer-sMGK 1 head	**37.31**	65.04	**81.23**	**61.19**
Transformer-MGK 1 head	37.13	**65.40**	80.63	61.05
Softmax 1 head	36.81	64.48	77.9	59.73

Table 2: Test Accuracy (%) of Transformer-MLK compared with the linear transformer on the LRA benchmark. Our Transform-MLKs achieve comparable or better accuracy than the baselines while using half the number of heads, having less parameters, and requiring less FLOPs (see Figure 3 for more details). Results are averaged over 5 runs with different random seeds.

Model	ListOps	Text	Retrieval	Average
Linear 8 heads	19.17	**65.85**	81.18	55.40
Transformer-sMLK 4 heads	**20.11**	65.74	**81.53**	**55.79**
Transformer-MLK 4 heads	20.06	65.7	81.34	55.7
Linear 4 heads	19.37	**65.81**	81.65	55.61
Transformer-sMLK 2 heads	19.88	65.61	**81.66**	**55.71**
Transformer-MLK 2 heads	**20.12**	65.72	80.80	55.54
Linear 2 heads	18.35	**65.94**	80.94	**55.07**
Transformer-sMLK 1 head	**18.87**	65.57	80.37	54.93
Transformer-MLK 1 head	18.34	65.70	**81.09**	55.04
Linear 1 head	18.60	65.70	80.6	54.96

Models and baselines We compare our 1-head, 2-head, 4-head Transformer-MGK and MLK with the baseline softmax [65] and linear transformers [27] that have 1 head, 2 heads, 4 heads, and 8 heads. Each model consists of two layers, and we adopt the model and training setting from [71] in our experiments.

Results We summarize our results in Table 1. Transformer-MGKs with half the number of heads consistently achieve better test accuracy than the baseline softmax attention across tasks. Since fewer heads are needed, transformer-MGKs use less parameters and need less FLOPs to compute than the baselines. We provide a detailed efficiency analysis for Transformer-MGKs in Figure 3. More interestingly, these efficiency advantages of Transformer-MGK over the baseline become more significant as the number of heads in the baseline model grows. Also, when using the same number of heads as the baseline models, Transformer-MGKs further improve over those baselines. Among the models, Transformer-sMGK performs the best across LRA tasks.

We also compare the performance of Transformer-MLK with the baseline linear transformers in Table 2. Like Transformer-MGK, Transformer-MLK yields comparable or better results than the baseline using only half the number of heads with less parameters and FLOPs. When using the same number of heads, Transformer-MLK helps improve the linear transformer further.

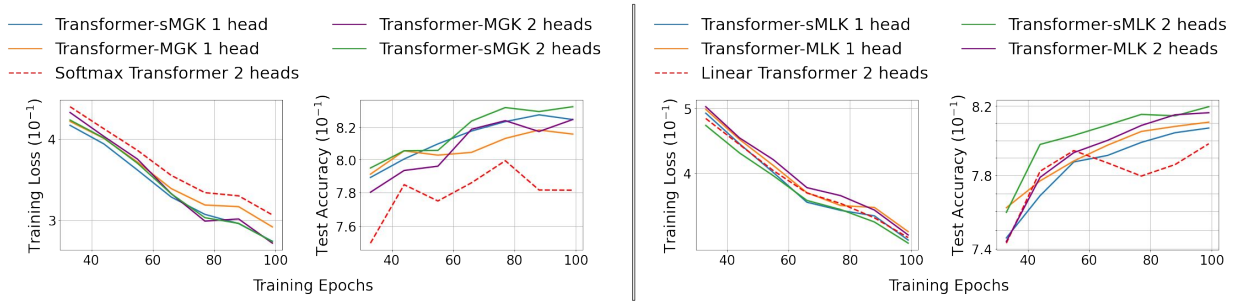

Figure 1: Training loss and test accuracy of Transformer-MGK vs. softmax transformer (Left) and of Transformer-MLK vs. linear transformer (Right) on the retrieval task, which has the longest average sequence-length and attention span among the LRA tasks [61]. The impressive performance of Transformer-MGK/MLK on this challenging task validates the capability of our models to capture long-range dependencies via learning a diversity of attention patterns.

In Figure 1, we compare the training loss and test accuracy curves of our 1-head and 2-head Transformer-MGK/MLK with the 2-head softmax and 2-head linear transformers on the document retrieval task. This retrieval task has the longest average sequence-length and attention span among the LRA tasks [61]. On this task, as shown in Figure 1, our Transformer-MGKs/MLKs are always better than the baseline models throughout the training. This observation corroborates our models's capability of capturing long-range dependencies in very long input sequences.

3.2 Language Modeling on WikiText-103

Next we confirm the advantage of our models on a large-scale application. We consider the word-level language modeling task on WikiText-103 [37] for our experiments in this section..

Datasets and metrics. WikiText-103 consists of articles from Wikipedia and is a dataset with long contextual dependencies. The training set is made up of about $28K$ articles containing $103M$ running words; this corresponds to text blocks of about 3600 words. The validation and test sets are composed of $218K$ and $246K$ running words, respectively. Each of them contains 60 articles and about $268K$ words. Our experiment follows the standard setting [37, 54] and splits the training data into L-word independent long segments. For evaluation, we use a batch size of 1, and go through the text sequence with a sliding window of size L. We consider only the last position for computing perplexity (PPL) except in the first segment, where all positions are evaluated as in [2, 54].

Models and baselines We compare the 4 and 8-head Transformer-MGK/MLK with the 8-head softmax [65] and linear transformers [27]. Each model consists of 16 layers. Our experiments follow the setting from [54].

Results As shown in Table 3, our Transformer-MGKs outperform the baseline softmax transformers. Even when using only half the number of attention heads (i.e., 4 heads vs. 8 heads as in the baselines), the Transformer-MGK still achieves better test perplexities than the baseline. Adding more heads into our Transformer-MGKs improves their performance. Similarly, Transformer-MLKs attain comparable test and validation perplexities to the baseline linear transformers when using half the number of attention heads. When using the same number of attention heads as in the baseline, Transformer-MLKs consistently achieve better performance.

3.3 Empirical Analysis

In this section, we conduct empirical analysis based on the Transformer-MGK trained for the document retrieval tasks. Empirical analysis results for Transformer-MLKs and the language modeling task are provided in the Appendix.

Table 3: Perplexity (PPL) on WikiText-103 of Transformer-MGK and MLK compared to the baselines. Both Transformer-MGK and MLK achieve comparable or better PPL than the baselines while using only half the number of heads. When using the same number of heads, our models significantly improve the baselines.

Method	Valid PPL	Test PPL
Softmax 8 heads	33.15	34.29
Transformer-MGK 4 heads	33.28	**34.21**
Transformer-sMGK 8 heads	32.92	**33.99**
Transformer-MGK 8 heads	32.74	33.93
Linear 8 heads	38.07	39.08
Transformer-MLK 4 heads	38.49	39.46
Transformer-MLK 8 heads	37.78	**38.99**

Transformer-MGK helps avoid learning redundant heads We visually compare attention matrices learned by Transformer-MGKs and the baseline softmax transformer on the document retrieval task in Figure 2. In particular, we randomly select an attention matrix at each head in each layer and visualize that attention matrix for each model in comparison. Figure 2(Left) shows that the queries in Transformer-MGKs can attend to a variety of keys and equivalently to other tokens at different positions in the input sequence. This diversity in attention pattern helps reduce the chance that the model learns similar and redundant attention matrices at different heads significantly.

Another metric to measure the representation capacity of an attention matrix is its rank. Attention matrices with high ranks can capture more diverse attention patterns compared to those with low ranks [42]. We study the rank of the attention matrix from the Transformer-MGK and the softmax transformer trained for the document retrieval task. In particular, we randomly select 1000 different attention matrices at each layer from each model. Then, we perform singular value decomposition (SVD) to compute the rank of each matrix and threshold the singular values smaller than 10^{-6}. Figure 2(Right) presents the distribution of the rank of attention matrices at each layer of the Transformer-MGK and the softmax transformer. We observe that attention matrices in Transformer-MGK has higher rank than those in the softmax transformer. Thus, our attention with MGK is capable of capturing more diverse and complex attention patterns than the baseline softmax attention.

Transformer-MGK/MLK reduces model complexity and computational cost We empirically analyze the efficiency of Transformer-MGK/MLK. Figure 3 compares the computational cost, measured in FLOPS, and model complexity, measured in the number of parameters, between our Transformer-MGK/MLK that has half the number of heads and the full-head softmax/linear transformer. The more heads being used, the more advantage Transformer-MGK/MLK has over the softmax/linear transformer. For much larger transformer models, this saving is significant.

Comparing different inference and learning techniques Table 4 compares the performance of Transformer-MGKs using different design options mentioned in Section 2.3. In particular, we consider the following three design options: A) Soft-E step, parameters π_{jr} and $\boldsymbol{k}_{jr}$ are learnable via SGD, and variance σ^2_{jr} are constants, B) Soft-E step, parameter π_{jr} is updated according to the M-step update, $\boldsymbol{k}_{jr}$ are learnable via SGD, and variance σ^2_{jr} are constants, and C) Hard-E step, π_{jr} and $\boldsymbol{k}_{jr}$ are learnable via SGD, and variance σ^2_{jr} are constants. Note that Transformer-MGKs with setting A are the default models we use in all experiments above. Table 4 summarizes our comparison results on tasks in the LRA benchmark. In Table 4, Transformer-MGK + Hard-E is the Transformer-MGK with setting C, Transformer-MGK + Soft-E is the Transformer-MGK

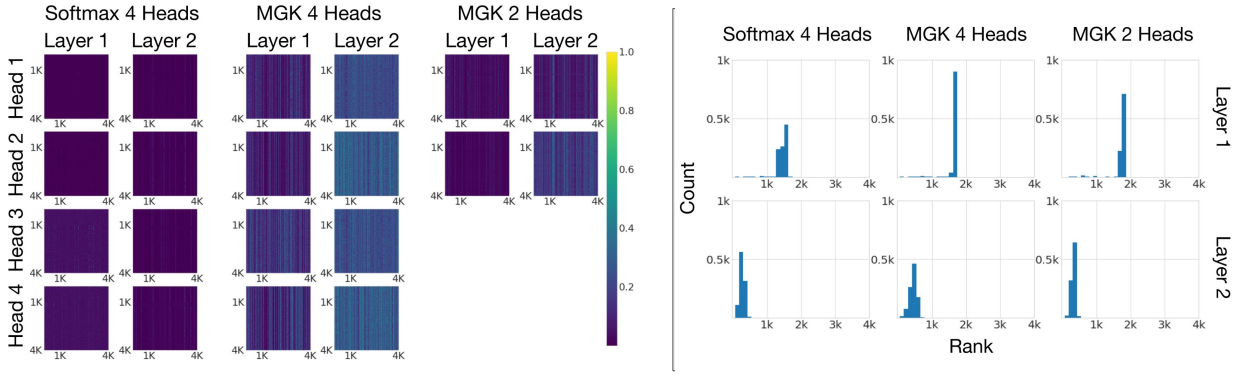

Figure 2: (Left) Visualization of attention matrices in the baseline 4-head softmax transformer (left), 4-head Transformer-MGK (middle), and 2-head Transformer-MGK (right) trained on the document retrieval task. Attention matrices from our Transformer-MGKs have more diverse patterns than those from the baseline softmax transformer. This diversity implies that queries in Transformer-MGKs can attend to more positions in the input sequence, reducing the risk of learning redundant heads. (Right) Rank distributions of attention matrices in softmax transformer and Transformer-MGK. These distributions show that attention matrices in Transformer-MGK have higher ranks than those in the softmax transformer and thus can capture more diverse attention patterns.

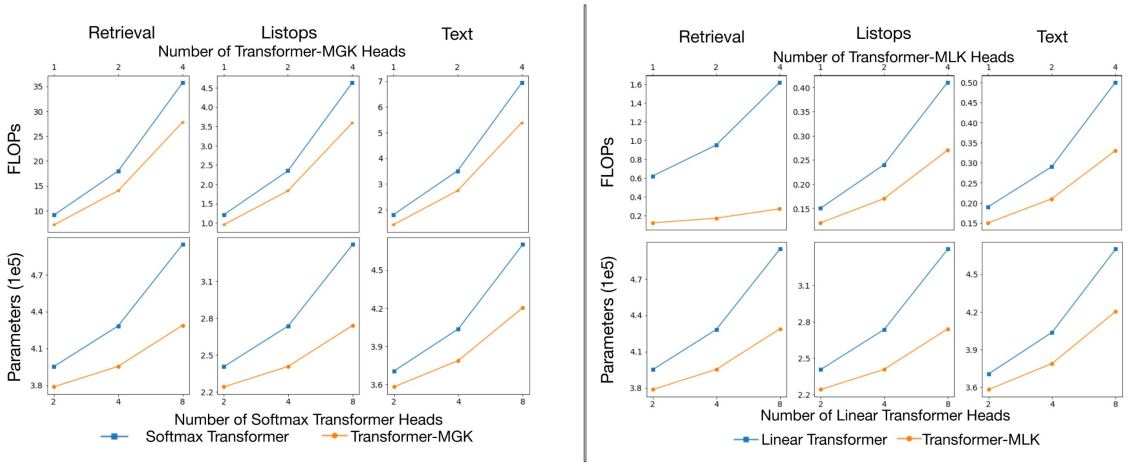

Figure 3: Computational cost (FLOPs) and the number of parameters of Transformer-MGK vs the baseline softmax transformer (Left) and Transformer-MLK vs. the baseline linear transformer (Right). Transformer-MGK/MLK is more efficient in both metrics than the softmax and linear transformer, and this advantage of Transformer-MGK/MLK grows with the number of heads.

with setting B, and Transformer-MGK only is the Transformer-MGK with setting A. It is worth noting that Transformer-sMGK + Hard-E obtains comparable results to the models with the best performance in each task even though it is the most efficient model in our study.

4 Related Work

Efficient Transformers Efficient transformers can be classified into several categories, as summarized in [51]. Among these categories are models with fixed patterns, which design the attention matrix to have sparse structure [44, 33, 46, 8, 5]. Another category includes models that combine two or more different access patterns to improve the coverage [8, 24]. Access patterns can also be

Table 4: Performance of Transformer-MGK using different inference and learning techniques on LRA benchmark. Transformer-sMGK + Hard-E is the most efficient model but also has competitive performance to other models in this experiment.

Model	ListOps	Text	Retrieval	Average
Transformer-sMGK + Hard-E 1 head	37.25	64.7	81.29	61.08
Transformer-sMGK + Soft-E 1 head	37.05	64.68	**81.44**	61.05
Transformer-sMGK 1 head	**37.31**	**65.04**	81.23	**61.19**
Transformer-MGK + Hard-E 1 head	19.40	**65.40**	80.72	55.17
Transformer-MGK + Soft-E 1 head	33.85	65.25	**80.73**	59.94
Transformer-MGK 1 head	**37.13**	**65.40**	80.63	**61.05**

made learnable in a data-driven fashion [30, 51, 60]. Other efficient transformers take advantage of a side memory module to access multiple tokens at once [31, 58, 3, 5]. Finally, low-rank and kernelization approximation are utilized to enhance the memory and computational efficiency of computing self-attention, see e.g., [64, 69, 27, 11, 55, 42]. Our MLK approach is complementery to those efficient transformers and can be easily incorporated into those architectures to further improve their accuracy and efficiency.

Redundancy in Transformers Latest works suggest that most of the neurons and heads in the pre-trained transformer are redundant and can be removed when optimzing towards a downstream task [14, 38, 18]. Some other works also study the contextualized embeddings in pretrained networks under this redundancy due to overparameterization and show that the representations learned within these models are highly anisotropic [39, 19]. From these observation, an emerging body of work is proposed to either distill or prune the model, including [53, 59, 68, 52]. Our MGK/MLK approch can be combined with these distilling and pruning methods to improve their accuracy and efficiency.

Mixture Models for Transformers Recently, several works have used mixture models to study and enhance transformers. Switch transformers [20] employ the routing algorithm in Mixture of Experts (MoE) to reduce the communication and computational costs in transformers. [41, 45] derive a probablistic framework based on GMMs for deep neural networks that can be extended to study transformers and attention-based architectures. Other works that use mixture models with transformers include [10, 21, 26].

5 Concluding Remarks

In this paper, we proposed Transformer-MGK, a class of transformers that use GMM to represent the key vectors in self-attention. Transformer-MGK reduces the redundancy among heads in transformer. Furthermore, attention heads in the Transformer-MGK have better representation capability than those in the baseline, allowing the Transformer-MGK to achieve comparable or better performance than the baseline softmax transformer while using only half of the number of heads. As a result, comparing to the baseline, the Transformer-MGK uses less parameters and requires the smaller amount of FLOPs. Furthermore, we extend the Transformer-MGK into the Transformer-MLK to use linear attentions for better efficiency. We empirically validate the advantage of the Transformer-MGK/MLK over the baseline softmax and linear transformers on various benchmarks including tasks in the LRA benchmark and WikiText-103 language modeling. In our work, we make the means and the variance of the cluster learnable variables and constants, respectively. It is interesting to explore how to leverage the M-step update in the EM algorithm to update those parameters. Furthermore, we leave the application of Transformer-MGK/MLK for improving the vision transformer [17, 63] as future work.

References

[1] R. Al-Rfou, D. Choe, N. Constant, M. Guo, and L. Jones. Character-level language modeling with deeper self-attention. In *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[2] R. Al-Rfou, D. Choe, N. Constant, M. Guo, and L. Jones. Character-level language modeling with deeper self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3159–3166, 2019.

[3] A. Asai and E. Choi. Challenges in information seeking qa: Unanswerable questions and paragraph retrieval. *arXiv preprint arXiv:2010.11915*, 2020.

[4] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

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

[6] C. M. Bishop. Pattern recognition. *Machine learning*, 128(9), 2006.

[7] T. Brown and et al. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, 2020.

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

[9] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.

[10] S. M. Cho, E. Park, and S. Yoo. Meantime: Mixture of attention mechanisms with multi-temporal embeddings for sequential recommendation. In *Fourteenth ACM Conference on Recommender Systems*, pages 515–520, 2020.

[11] K. M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, D. B. Belanger, L. J. Colwell, and A. Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

[12] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy, Aug. 2019. Association for Computational Linguistics.

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

[14] F. Dalvi, H. Sajjad, N. Durrani, and Y. Belinkov. Analyzing redundancy in pretrained transformer models. *arXiv preprint arXiv:2004.04010*, 2020.

[15] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

[18] N. Durrani, H. Sajjad, F. Dalvi, and Y. Belinkov. Analyzing individual neurons in pre-trained language models. *arXiv preprint arXiv:2010.02695*, 2020.

[19] K. Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*, 2019.

[20] W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021.

[21] M. Guo, Y. Zhang, and T. Liu. Gaussian transformer: a lightweight approach for natural language inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6489–6496, 2019.

[22] S. J. Hanson. A stochastic version of the delta rule. *Physica D: Nonlinear Phenomena*, 42(1-3):265–272, 1990.

[23] J. Hewitt and P. Liang. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[24] J. Ho, N. Kalchbrenner, D. Weissenborn, and T. Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

[25] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[26] J. Jiang, G. G. Xia, D. B. Carlton, C. N. Anderson, and R. H. Miyakawa. Transformer vae: A hierarchical model for structure-aware and interpretable music representation learning. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 516–520. IEEE, 2020.

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

[28] Y. Kim, C. Denton, L. Hoang, and A. M. Rush. Structured attention networks. *arXiv preprint arXiv:1702.00887*, 2017.

[29] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] N. Kitaev, L. Kaiser, and A. Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[31] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.

[32] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130, 2017.

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

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

[35] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[36] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[37] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[38] P. Michel, O. Levy, and G. Neubig. Are sixteen heads really better than one? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[39] J. Mu and P. Viswanath. All-but-the-top: Simple and effective postprocessing for word representations. In *International Conference on Learning Representations*, 2018.

[40] N. Nangia and S. Bowman. ListOps: A diagnostic dataset for latent tree learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 92–99, New Orleans, Louisiana, USA, June 2018. Association for Computational Linguistics.

[41] T. Nguyen, N. Ho, A. Patel, A. Anandkumar, M. I. Jordan, and R. G. Baraniuk. A Bayesian perspective of convolutional neural networks through a deconvolutional generative model. *arXiv preprint arXiv:1811.02657*, 2018.

[42] T. M. Nguyen, V. Suliafu, S. J. Osher, L. Chen, and B. Wang. Fmmformer: Efficient and flexible transformer via decomposed near-field and far-field attention. *arXiv preprint arXiv:2108.02347*, 2021.

[43] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas, Nov. 2016. Association for Computational Linguistics.

[44] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran. Image transformer. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4055–4064. PMLR, 10–15 Jul 2018.

[45] A. B. Patel, M. T. Nguyen, and R. Baraniuk. A probabilistic framework for deep learning. *Advances in neural information processing systems*, 29:2558–2566, 2016.

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

[47] D. R. Radev, P. Muthukrishnan, V. Qazvinian, and A. Abu-Jbara. The acl anthology network corpus. *Language Resources and Evaluation*, 47(4):919–944, 2013.

[48] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. *OpenAI report*, 2018.

[49] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[50] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, Nov. 2016. Association for Computational Linguistics.

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

[52] H. Sajjad, F. Dalvi, N. Durrani, and P. Nakov. Poor man's bert: Smaller and faster transformer models. *arXiv e-prints*, pages arXiv–2004, 2020.

[53] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[54] I. Schlag, K. Irie, and J. Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pages 9355–9366. PMLR, 2021.

[55] Z. Shen, M. Zhang, H. Zhao, S. Yi, and H. Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3531–3539, 2021.

[56] D. R. So, C. Liang, and Q. V. Le. The evolved transformer. *arXiv preprint arXiv:1901.11117*, 2019.

[57] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[58] S. Sukhbaatar, E. Grave, G. Lample, H. Jegou, and A. Joulin. Augmenting self-attention with persistent memory. *arXiv preprint arXiv:1907.01470*, 2019.

[59] S. Sun, Y. Cheng, Z. Gan, and J. Liu. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*, 2019.

[60] Y. Tay, D. Bahri, L. Yang, D. Metzler, and D.-C. Juan. Sparse Sinkhorn attention. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9438–9447. PMLR, 13–18 Jul 2020.

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

[62] I. Tenney, D. Das, and E. Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy, July 2019. Association for Computational Linguistics.

[63] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[64] Y.-H. H. Tsai, S. Bai, M. Yamada, L.-P. Morency, and R. Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. *arXiv preprint arXiv:1908.11775*, 2019.

[65] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[66] J. Vig and Y. Belinkov. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy, Aug. 2019. Association for Computational Linguistics.

[67] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics.

[68] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.

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

[70] A. Williams, N. Nangia, and S. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, June 2018.

[71] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh. Nyströmformer: A Nyström-based Algorithm for Approximating Self-Attention. 2021.

[72] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

Supplement to "Transformer with a Mixture of Gaussian Keys"

In this supplementary material, we provide experimental details and additional experiments of Transformer-MGK and Transformer-MLK.

A Appendix

A.1 Experiment details

In this section, we provide model and training details for experiments in Section 3.

A.1.1 Long Range Arena Benchmark

We use the softmax transformer [65] and linear transformer [27] as our baselines. All models have 2 layers, 64 embedding dimension, and 128 hidden dimension. The number of heads in each layer are set to 1, 2, 4, and 8. For Transformer-MGK/MLKs and their shifted versions, we share π_{jr} for all position j and learn it for each head. The initial value for each π_{jr} is set to 0.5. For Transformer-sMGK, we learn $\mathbf{b_r}$ and initialize its elements from a standard normal distribution. Each σ_{jr} is a constant with value $\sqrt{D}$ where D is the dimension of each head, which is the same as in the baselines models

Details about the Long Range Arena (LRA) benchmarks can be found in the original paper [61]. Our implementation is based on the public code by [71], and we follow their training procedures. The training setting and additional baseline model details are provided in the configuration file used in [71] and available at https://github.com/mlpen/Nystromformer/blob/main/LRA/code.

A.1.2 Language Modeling on WikiText-103

Our language modeling implementation is based on the public code by [54]; we use their small-model configuration for all models in our experiments. In particular, we set the key, value, and query dimension to 128, and the training and evaluation context length to 256. We also set the number of heads to 8, the feed-forward layer dimension to 2048, and the number of layers to 16. For Transformer-MGK/MLK, we use our 4 and 8-head versions to compare with the 8-head baselines. π_{ir}, $\mathbf{b_r}$ and σ_{jr} for this task follow our setings for LRA experiments. Other than those, our language modeling share the same configurations as the baselines.

We train our models for language modeling on 2 A100, 40GB each with a batch size of 96, and each model is trained for 120 epochs. We apply 10% dropout [22, 57] and use the Adam optimizer [29] with an initial learning rate of 0.00025 and 2000 steps for learning rate warm-up.

A.2 More Empirical Analysis on Transformer-MGKs/MLKs trained for Language Modeling

In Table 3 in the main text, we show the improvements in valid and test perplexity of our Transformer-MGKs/MLKs compared with the baseline softmax and linear transformers. In particular, Transformer-MGK/MLKs with the same number of heads as the baselines, e.g. 8 heads, significantly improve the baselines during training while Transformer-MGK/MLKs with 4 head achieve comparable or better performance than the 8-head baselines. In this section, we provide more empirical analysis to shed light on those results. Figure 4 shows the validation perplexity curve of our models versus the softmax and linear transformers.

Figure 5 visualizes the attention matrices from a randomly selected sample for the trained softmax transformer with 8 heads and Transformer-MGKs with 8 heads and 4 heads. These visualizations show that Transformer-MGKs attend to more diverse positions in all heads and layers than the softmax attention baseline. We also compare the rank distributions of these attention matrices

computed from 1000 samples at each layer in the model. Figure 6 presents the rank histograms of the 8-head softmax attention and 4-head and 8-head MGK attentions for the 1^{st} and 5^{th} layer. It is clear that attention matrices from the Transformer-MGKs have higher ranks than those in the softmax transformer, which implies that Transformer-MGK can attend to more diverse regions than the softmax transformer without the need of using more attention heads.

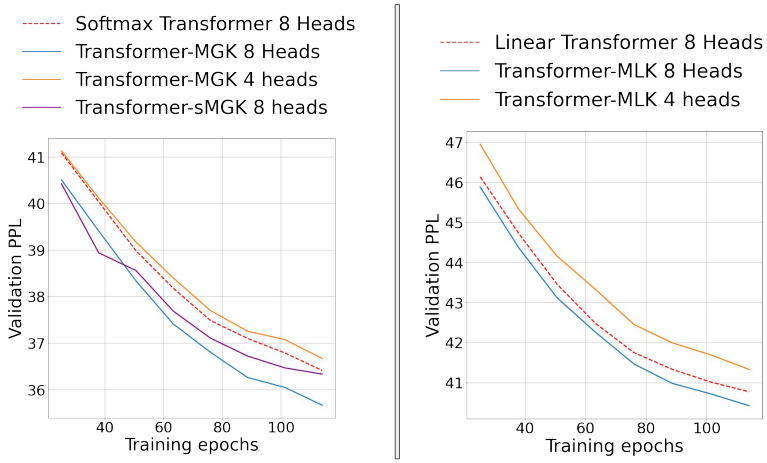

Figure 4: Validation perplexity of Transformer-MGK vs. the softmax transformer (Left) and Transformer-MLK vs. the linear transformer (Right) for language modeling on WikiText-103. While 8-head Transformer-MGK/sMGK and 8-head Transformer-MLK outperform softmax and linear transformer, 4-head Transformer-MGK/MLK perform on par with the baselines.

.

A.3 Additional training results for LRA

In this section, we provide additional experimental results on the LRA benchmark. Figure 7 compares the computational cost measured by FLOPs and model complexity in term of the number of parameters of different inference and learning methods for Transformer-MGK. The computational costs of Transformer-MGK/sMGK and Transformer-MGK Hard-E/Soft-E are on par with each other, while Transformer-sMGK uses fewer parameters than the other without trade-off in performance 4 for all tasks. The naming is as explained in Section 3.3 in the main text. In addition, Figure 8 visualizes the attention matrices in the 4-head linear transformer baseline, 4-head Transformer-MLK, and 2-head Transformer-MLK trained on the document retrieval task.

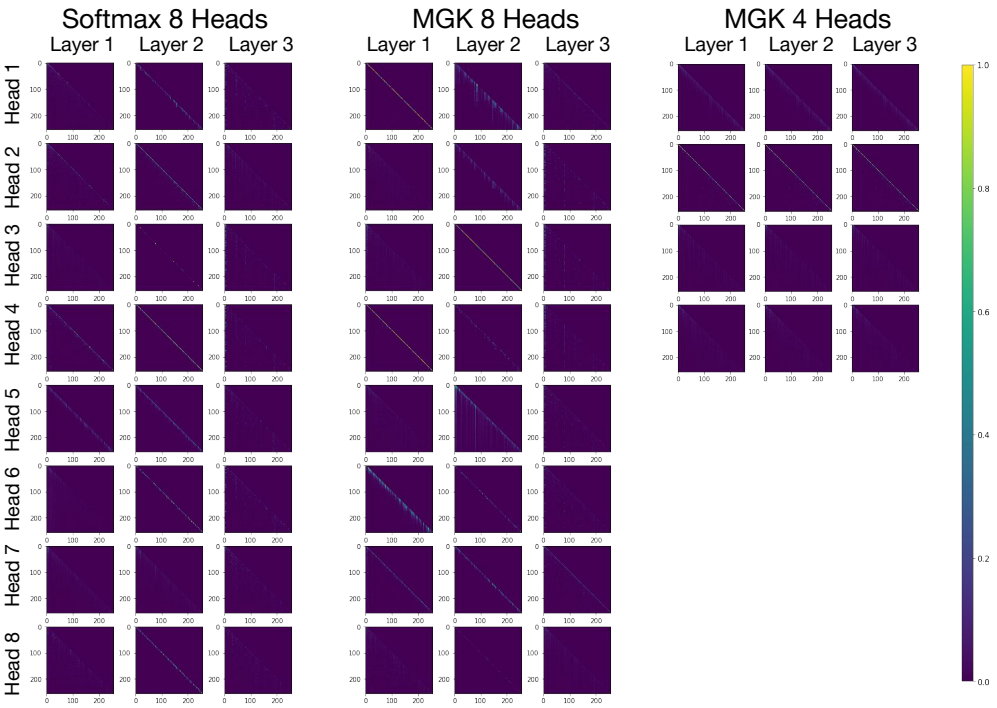

Figure 5: Visualization of attention matrices from 8-head softmax transformer (Left), 8-head Transformer-MGK (Middle), and 4-head Transformer-MGK (Right) trained on WikiText-103 language modeling. The queries from our models can attend to more diverse patterns than those from the softmax baseline, which enables Transformer-MGK to have less heads while achieve competitive performance. Here, we plot the attention matrices for all heads and layers in the models. The sequence length is 256, and the size of each matrix is 256×256.

.

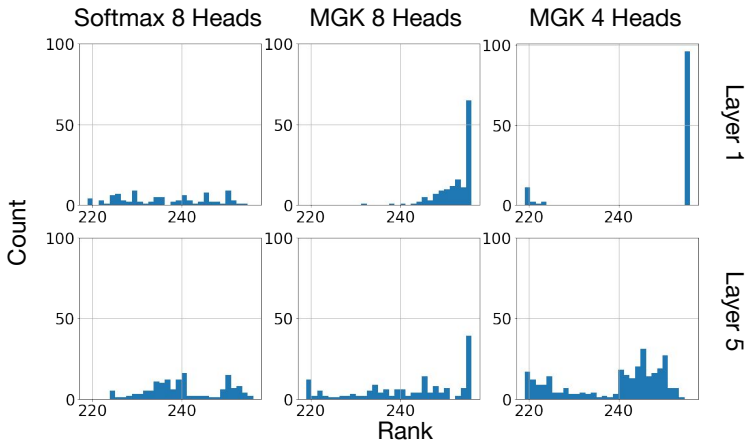

Figure 6: Rank distributions of attention matrices from 8-head softmax transformer (Left), 8-head Transformer-MGK (Middle), and 4-head Transformer-MGK (Right) trained on WikiText-103 language modeling. The rank histograms are computed from 1000 attention matrices at each layer. The attention matrices from Transformer-MGKs have higher ranks than those from softmax transformer. This implies that Transformer-MGKs have more diverse attention patterns, which allows us to reduce the number of heads in Transformer-MGKs.

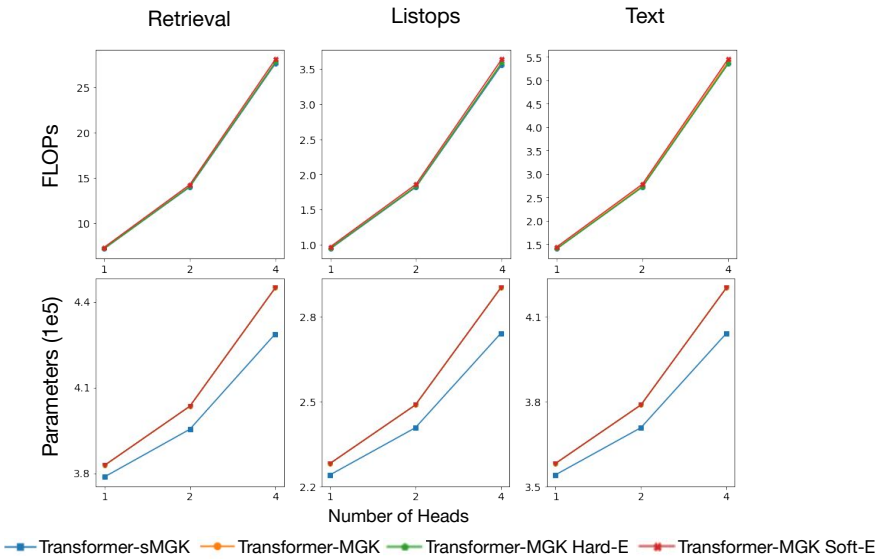

Figure 7: Computational cost (FLOPs) (Top) and model complexity (Bottom) of different inference and learning techiques for Transformer-MGK trained on the document retrieval task. While computational costs are almost the same, Transformer-sMGK has more advantage in model size, comparing to Transformer-MGK, Transformer-MGK Hard-E/Soft-E. The naming is as explained in Section 3.3 in the main text.

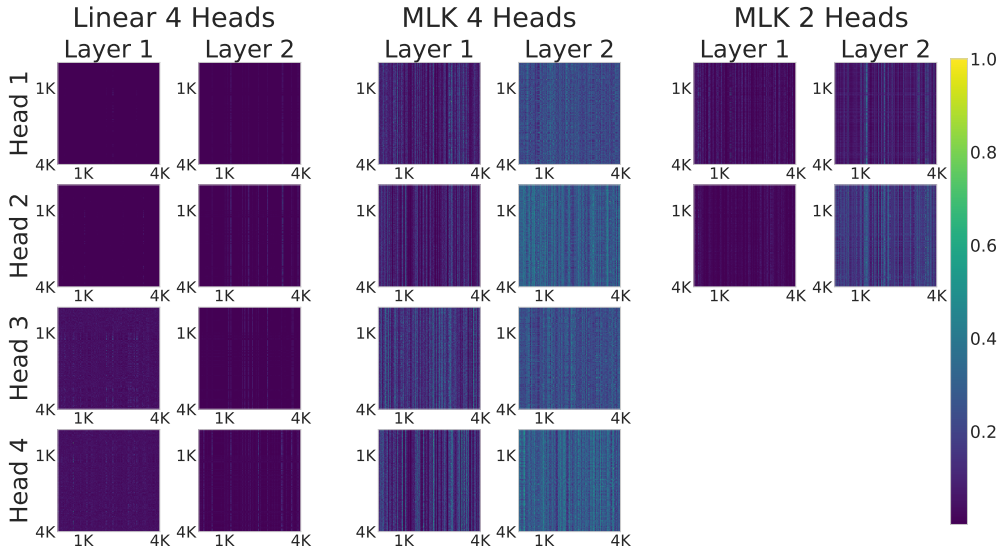

Figure 8: Visualization of attention matrices in the 4-head linear transformer baseline (Left), 4-head Transformer-MLK (Middle), and 2-head Transformer-MLK (Right) trained on the document retrieval task. Here, the sequence length is 4000, and the size of each matrix is 4000×4000.