Improving Transformer with an Admixture of Attention Heads

Tan M. Nguyen∗
Department of Mathematics
University of California, Los Angeles
tanmnguyen89@ucla.edu

Tam Nguyen∗
FPT Software AI Center
nguyenminhtam9520@gmail.com

Hai Do
FPT Software AI Center
haidn6@fsoft.com.vn

Khai Nguyen
Department of Statistics and Data Sciences
University of Texas at Austin
khainb@utexas.edu

Vishwanath Saragadam
Department of ECE
Rice University
vishwanath.saragadam@rice.edu

Minh Pham
Department of Mathematics
University of California, Los Angeles
minhrose@ucla.edu

Duy Khuong Nguyen
FPT Software AI Center
khuongnd6@fsoft.com.vn

Nhat Ho∗∗
Department of Statistics and Data Sciences
University of Texas at Austin
minhnhat@utexas.edu

Stanley J. Osher∗∗
Department of Mathematics
University of California, Los Angeles
sjo@math.ucla.edu

Abstract

Transformers with multi-head self-attention have achieved remarkable success in sequence modeling and beyond. However, they suffer from high computational and memory complexities for computing the attention matrix at each head. Recently, it has been shown that those attention matrices lie on a low-dimensional manifold and, thus, are redundant. We propose the Transformer with a Finite Admixture of Shared Heads (FiSHformers), a novel class of efficient and flexible transformers that allow the sharing of attention matrices between attention heads. At the core of FiSHformer is a novel finite admixture model of shared heads (FiSH) that samples attention matrices from a set of global attention matrices. The number of global attention matrices is much smaller than the number of local attention matrices generated. FiSHformers directly learn these global attention matrices rather than the local ones as in other transformers, thus significantly improving the computational and memory efficiency of the model. We empirically verify the advantages of the FiSHformer over the baseline transformers in a wide range of practical applications including language modeling, machine translation, and image classification. On the WikiText-103, IWSLT’14 De-En and WMT’14 En-De, FiSHformers use much fewer floating-point operations per second (FLOPs), memory, and parameters compared to the baseline transformers.

∗ Co-first authors. ∗∗ Co-last authors. Please correspond to: tanmnguyen89@ucla.edu

1 Introduction

Transformers have become the state-of-the-art model for solving many challenging problems in natural language processing [75, 1, 17, 80, 21, 10, 35, 61] and computer vision [19, 68, 22, 73]. Transformers learn from unlabeled data effectively and take advantage of the pre-trained models on downstream tasks that involve different data modalities with limited supervision [59, 60, 21, 82, 43]. The success of transformer is credited to the multi-head self-attention (MHA) mechanism as their fundamental building block. For each token in the sequence, self-attention in transformers aggregates information from other tokens by computing a weighted average of their feature representations with a weight proportional to a similarity score between the representations. This attention mechanism allows arbitrary input-dependent interaction between tokens in the sequence where a token can pay attention to other tokens and attain a contextual representation [6, 75, 38]. Multi-head self-attention captures multiple such contextual representations, one at each head, thereby increasing the representation capacity of the self-attention. It has been argued that the representation capacity of the attention mechanism [72] and its flexibility in capturing diverse syntactic and semantic relationships [72, 76, 15, 77, 32] account for the impressive performance of transformers in practice.

1.1 Background: Self-Attention

For a given input sequence $X := [x_1, \cdots, x_N]^T \in \mathbb{R}^{N \times D_x}$ of $N$ feature vectors, self-attention transforms $X$ into the output sequence $H$ in the following two steps:

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

$$Q = X W_Q^T, \quad K = X W_K^T, \quad V = X W_V^T,$$

where $W_Q, W_K \in \mathbb{R}^{D \times D_x}$, and $W_V \in \mathbb{R}^{D_x \times D_v}$ are the weight matrices. We denote $Q := [q_1, \cdots, q_N]^T$, $K := [k_1, \cdots, k_N]^T$, and $V := [v_1, \cdots, v_N]^T$, where the vectors $q_i, k_i, v_i$ for $i = 1, \cdots, N$ are the query, key, and value vectors, respectively.

**Step 2.** The output sequence $H := [h_1, \cdots, h_N]^T$ is then computed as follows

$$H = \text{softmax}(Q K^T / \sqrt{D}) V := \text{softmax}(A / \sqrt{D}) V,$$

where the softmax function is applied to each row of the matrix $A = (Q K^T)$. This matrix $A \in \mathbb{R}^{N \times N}$ and its component $a_{ij}$ for $i, j = 1, \cdots, N$ are called the attention matrix and attention scores, respectively. For each query vector $q_i$ for $i = 1, \cdots, N$, an equivalent form of Eqn. (1) to compute the output vector $h_i$ is given by

$$h_i = \sum_{j=1}^{N} \text{softmax}(q_i^T k_j / \sqrt{D}) v_j.$$

The self-attention computed by Eqn. (1) and (2) is called the scaled dot-product or softmax attention. In our paper, we call a transformer that uses this attention the softmax transformer. The structure
that the attention matrix $A$ learns from training determines the ability of the self-attention to capture contextual representation for each token.

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

$$\text{MultiHead}(\{H_j\}_{j=1}^{H}) = \text{Concat}(H_1, \ldots, H_H)W^D.$$

### 1.2 Eigenvalue Analysis of the Attention Matrices

The multi-head mechanism allows transformers to capture more diverse attention patterns and increase the capacity of the model. However, in many practical tasks, transformers learn redundant heads \([47, 78]\), whose learned attention matrices lie on a low-dimensional manifold \([8]\). To confirm this claim, in Figure 2, we follow the eigenvalue analysis in \([8]\) and investigate the eigenvalues of the covariance matrix of vectorized attention matrices aggregated over each layer of a transformer model trained on the WikiText-103 dataset for language modeling. We observe that this covariance matrix is low rank with top 200 (1.2%) eigenvalues capturing more than 90% of the energy. This result verifies that the variability of learned attention matrices in transformers can be explained by a relatively small number of principal components, and those attention matrices lie on a low-dimensional manifold. Therefore, in multi-head attention, the effective number of heads is much smaller than the actual number of heads, and a more effective way to compute multi-head attention is needed to improve the efficiency of transformers.

### 1.3 Contribution

Leveraging the idea of the finite admixture model (FAM) \([57, 9]\), we propose a new class of efficient transformer architectures, namely the Transformer with a Finite Admixture of Shared Heads (FiSHformer). At the core of FiSHformer is to sample local attention matrices from an admixture of a small number of global attention matrices. This sharing mechanism between heads helps reduce the computational complexity and the model size compared to the MHA softmax and the computational complexity measured by the number of parameters. The advantages of our methods grow with the model/feature dimension $D$ and the input sequence length $N$.

1. We construct an admixture model for shared attention matrices between heads and propose FiSHformer, a novel class of transformers that take advantage of this admixture model to efficiently compute multi-head attention.
2. We introduce a nonlinearity mapping from global heads to local heads into FiSH and propose the Generalized FiSHformer (GFiSHformer). We then explore different possibilities to design FiSHformer and GFiSHformer.
3. We empirically verify that FiSHformer and GFiSHformer achieve similar or even better accuracy but with much less computational cost in terms of FLOPs and smaller model complexity measured by the number of parameters. The advantages of our methods grow with the model/feature dimension $D$ and the input sequence length $N$.

We also show that FiSHformer-based models help reduce head redundancy in our experiments.

**Organization:** We structure this paper as follows: In Section 2, we develop a finite admixture model of shared heads and then present our FiSHformer and its extensions. In Section 2.5, we analyze the reduction in model complexity and computational cost from FiSH. In Section 3 and 4, we validate and empirically analyze the efficiency and accuracy of FiSHformer, as well as conducting ablation studies on the model. We discuss related works in Section 5. The paper ends up with concluding remarks. More results and details are provided in the Appendix.

### 2 Transformer with a Finite Admixture of Shared Heads

We first review the finite admixture model (FAM) and derive a FAM of shared heads for the multi-head self-attention. We then define Transformer with a Finite Admixture of Shared Heads (FiSHformer).

#### 2.1 A Probabilistic Viewpoint of Attention Matrices

Let $A_j$ denote the attention matrix at the $j$th head, $j = 1, 2, \ldots, H$. From a probabilistic viewpoint, to have diversity among $A_1, \ldots, A_H$, we can assume that $A_j$ comes from a distribution $P_j$ for
all $j$. Since the distributions $\mathbb{P}_1, \ldots, \mathbb{P}_H$ can have complex forms and be difficult to compute, our approach is to consider approximated distributions of $\mathbb{P}_j$ and these approximated distributions have simple forms. A natural choice for each of these approximated distributions is via a finite mixture of Gaussian distributions, which can be summarized in the following lemma below.

**Lemma 1.** Assume that $P \in \mathbb{R}^{D'}$ is a probability distribution supported on some compact set and admits differentiable and bounded density estimation $p$. Then, for any scale parameter $\sigma > 0$ and for any $\epsilon > 0$, there exists universal constant $C$ and $M \leq (C \log(1/\epsilon))^{D'}$ such that we can find a mixture of $M$ components $\sum_{i=1}^{M} p_i \mathcal{N}(\theta_i, \sigma^2 I_{D'})$ where $p_1, \ldots, p_M$ are weight parameters and $\theta_1, \ldots, \theta_K$ are location parameters that satisfy the following inequality

$$
\sup_{x \in \mathbb{R}^d} |p(x) - \sum_{i=1}^{M} p_i \phi(x|\theta_i, \sigma^2 I_{D'})| \leq \epsilon,
$$

where $\phi(\cdot|\theta, \sigma^2 I)$ is Gaussian density function with location parameter $\theta$ and covariance matrix $\sigma^2 I_{D'}$.

The proof of Lemma 1 is in Appendix E. In light of Lemma 1, for each scale parameter $\sigma > 0$ and for each distribution $P_j$, we can find the corresponding number of components $M_j$, weight parameters $p_{j1}, \ldots, p_{jM_j}$, and location parameters $\theta_{j1}, \ldots, \theta_{jM_j}$ such that the mixtures $P_j' = \sum_{i=1}^{M_j} p_{ij} \mathcal{N}(\theta_{ij}, \sigma^2 I_{D'})$ can approximate the distribution $P_j$ up to a given accuracy $\epsilon$. However, these approximations still involve $\prod_{j=1}^{K} M_j$ number of location parameters, which can be computationally expensive. To overcome this issue, we assume that $M_1 = M_2 = \ldots = M_H$ and the location parameters $(\theta_{11}, \ldots, \theta_{M_j}) = (\theta_1, \ldots, \theta_M)$ for all $j$, i.e., these approximated mixtures share a similar set of location parameters. This sharing information of location parameters has a deep connection to finite admixture models, which we are going to elaborate in the next sections.

### 2.1.1 Background

Finite admixture models (FAM) are extensions of finite mixture models (FMMs), which served as a workhorse in stochastic modeling. A finite mixture distribution of $M$ components for a random array $X \in \mathbb{R}^{N \times J}$ is given by

$$
x_j \sim \sum_{k=1}^{M} p_k f(x; \theta_k), \quad \sum_{k=1}^{M} p_k = 1, \quad p_k \geq 0,
$$

where $x_j \in \mathbb{R}^N$ is the $j$-th row of $X$ randomly sampled from the mixture distribution, $f$ is a chosen probability measure, such as a Gaussian distribution, $p = \{p_1, p_2, \ldots, p_M\}$ are mixture weights, and $\theta_k$ denotes the parameter values for the $k$-th component.

A FAM is a generalization of a FMM where rows $x_j, j = 1, \ldots, H$, are drawn from different mixture distributions that share the components $f(x; \theta_k), k = 1, \ldots, M$ but with different mixture weights

$$
x_j \sim \sum_{k=1}^{M} p_{kj} f(x; \theta_k), \quad \sum_{k=1}^{M} p_{kj} = 1, \quad p_{kj} \geq 0.
$$

### 2.2 Multi-head as a Finite Admixture Model of Shared Heads (FiSH)

As demonstrated in Section 2.1, we propose a Finite Admixture Model of Shared Heads (FiSH), in which $A_j$ follows finite admixture distribution of $M$ components given by

$$
A_j \sim \sum_{k=1}^{M} p_{kj} f(A; \theta_k), \quad \sum_{k=1}^{M} p_{kj} = 1, \quad p_{kj} \geq 0.
$$

Here $M < H$ and $f(A; \theta_k)$ are chosen probability measures. In particular, we choose $f(A; \theta_k)$ to be Gaussian distributions $\mathcal{N}(A; G_k, \Sigma_k)$, where $G_k = Q_k K_k^\top$ and $\Sigma_k = \sigma_k^2 I$ are the mean and covariances, respectively. FiSH is then defined as follows:

**Definition 1** (Finite Admixture Model of Shared Heads). The multi-head attention admits a finite admixture model of shared heads if the attention matrices $A_j$ at the $j$-th head are sampled from the following finite admixture model:

$$
A_j \sim \sum_{k=1}^{M} p_{kj} \mathcal{N}(A; Q_k K_k^\top, \sigma_k^2 I), \quad \sum_{k=1}^{M} p_{kj} = 1, \quad p_{kj} \geq 0.
$$
In FiSH defined in Def. 1, we call \( \{ G_k = Q_k K_k^\top \}_{k=1}^M \) global attention matrices and \( \{ A_j \}_{j=1}^H \) local attention matrices. FiSH computes \( M \) global attention matrices \( G_k \), and \( H \) local attention matrices \( A_j \) are sampled from FiSH as in Eqn. (7) with \( M < H \).

**Remark 1 (FiSH vs. Baseline MHA).** The baseline MHA with \( H \) heads need to compute \( H = 8 \), attention matrices, each of which requires \( O(N^2) \) computational costs where \( N \) is the length of the input sequence. In contrast, FiSH only need to compute \( M < H \), e.g. \( M = 2 \), global attention matrices, each of which also requires \( O(N^2) \) computational and memory costs. Then FiSH combines those global attention matrices to form a FAM from which \( H = 8 \), local attention matrices are sampled as in Eqn. (7). This second step of sampling local attention matrices from a set of global attention matrices in FiSH requires very few computations. Thus, FiSH is more efficient than MHA.

**Remark 2 (Connection to Topic Models).** FiSH can be interpreted as a Probabilistic Latent Semantic Analysis (pLSA) model for topic modeling. Considering the document \( d \) that contains the word \( w \) whose topic is \( c \), pLSA models the occurrence of the word \( w \) in the document \( d \) as a mixture of conditionally independent Multinomial distributions \( p(w|d) = \sum c p(c|d)p(w|c) \). Comparing this pLSA with Eqn. (7) of FiSH, we can associate the mixture weights \( p_{kj} \) and the distribution \( N(A_j; Q_j K_j^\top, \sigma_k^2 I) \) in FiSH with the distributions \( p(c|d) \) and \( p(w|c) \) in pLSA, respectively. Therefore, it can be interpreted that the global attention matrices in FiSH play the role of topics, and the local attention matrices in FiSH are words sampled from those topics. It is interesting to note that pLSA is equivalent to the famous Latent Dirichlet Allocation model under a uniform Dirichlet prior on the per-document topic distribution \( p(c|d) \).

### 2.3 Transformer with a Finite Admixture of Shared Heads

FiSHformers are transformers that use FiSH instead of MHA. FiSH, as defined in Def. 1, is not differentiable, which poses a difficulty in training FiSHformers. Applying the reparameterization trick [39], the attention matrices \( A_j \) can be written in a differentiable form as follows:

\[
A_j = \sum_{k=1}^M p_{kj}(Q_k K_k^\top + \sigma_k \odot \epsilon_j), \quad \epsilon_j \sim N(0, I), \quad \sum_{k=1}^M p_{kj} = 1, \quad p_{kj} \geq 0. \tag{8}
\]

FiSHformers use the formulation of local attention heads in Eqn. (8) to implement FiSH.

**Transformer with a Hard Finite Admixture of Shared Heads (Hard FiSHformer)** Hard FiSHformer takes the zero-noise limit of Eqn. (8) to reduce the computational cost. The attention matrices \( A_j \) in Hard FiSHformer are then calculated as

\[
A_j = \sum_{k=1}^M p_{kj} Q_k K_k^\top, \quad \sum_{k=1}^M p_{kj} = 1, \quad p_{kj} \geq 0. \tag{9}
\]

**Remark 3 (Discriminative Relaxation).** To take the advantage of learning from data, the convex combination condition of \( p_{kj} \), i.e. \( \sum_{k=1}^M p_{kj} = 1, \quad p_{kj} \geq 0 \), can be relaxed, and those mixing coefficients are made learnable parameters that are learned from data during training.

**Remark 4 (Transformers with a Mixture of Shared Heads (MiSHformer)).** A transformer with a mixture of shared heads (MiSHformer) can be used to reduce the amount of computation with the cost of accuracy reduction. MiSH is a special case of FiSH when the mixture weights \( p_{kj} \) are the same for all \( j \). The local attention matrices \( A_j \) in MiSHformer are given by

\[
A_j = \sum_{k=1}^M p_k(Q_k K_k^\top + \sigma_k \odot \epsilon_j), \quad \epsilon_j \sim N(0, I), \quad \sum_{k=1}^M p_k = 1, \quad p_k \geq 0. \tag{10}
\]

An empirical comparison between FiSHformer and MiSHformer is provided in Section 4.

### 2.4 Transformer with a Generalized Finite Admixture of Shared Heads

In order to increase the representation capacity of attention heads, we follow a common approach in learning representation by replacing the linear mapping in Eqns. (8) and (9) by a nonlinear mapping such as a neural network with the rectified linear units (ReLU). The Transformer with a Generalized Finite Admixture of Shared Heads (GFiSHformer) is then formulated as

\[
A_j = \sum_{k=1}^M \phi(p_{kj}(Q_k K_k^\top + \sigma_k \odot \epsilon_j)), \quad \epsilon \sim N(0, I),
\]

where \( \phi \) is a nonlinear mapping and \( p_{kj} \) are relaxed to be learnable parameters. Similarly, we formulate local attention matrices \( A_j \) in the Transformer with a Generalized Hard Finite Admixture of Shared Heads (Hard GFiSHformer) as

\[
A_j = \sum_{k=1}^M \phi(p_{kj} Q_k K_k^\top).
\]
2.5 Reduction in Model Complexity and Computational Cost from FiSH
Compared to its $H$-head MHA counterpart, a FiSH attention of $M$ global heads and $H$ local heads saves $\left[2(H-M)D - 2MH\right]N^2 + 2(H-M)D\left(2D_x - 1\right)N$ FLOPs in a forward pass and $2(H-M)DD_x - HM - M$ parameters. Detailed derivations are provided in Appendix D.

3 Experimental Results
In this section, we empirically study the advantages of FiSHformer on various tasks and benchmarks, including language modeling on WikiText-103 (Section 3.1), machine translation on IWSLT’14 De-En and WMT’14 (Section 3.2), image classification on ImageNet (Section 3.3), time series classification on the UEA benchmark (Section 3.4), and reinforcement learning on the D4RL Benchmark (Section 3.5). We aim to show that: (i) FiSHformers improve the efficiency and accuracy upon the MHA baseline; (ii) FiSH is a universal method that can be applied to state-of-the-art transformer models to improve their performance on large-scale applications. In Section 4, we also show that FiSH helps reduce the redundancy between attention heads.

We compare FiSHformers, Hard FiSHformers, GFiSHformers, and Hard GFiSHformers with the baseline MHA softmax transformers. In our experiments, we apply the discriminative relaxation explained in Remark 3 on our FiSHformers to make the mixture weights $p_{kj}$ learnable parameters. For GFiSHformers/Hard GFiSHformers, we choose the nonlinear mapping $\phi$ to be a ReLU followed by a linear neural network. All of our results are averaged over 5 runs with different seeds. More details on datasets, models, and training are provided in Appendix A. Our PyTorch code with documentation can be found at https://github.com/minhtannguyen/FishFormer.

3.1 WikiText-103 Language Modeling
Models and baselines We compare the 2 and 4-global-head FiSHformers with the 8-head softmax transformers [75]. Each model has 16 layers, and our training follows the setting from [66].

Results Perplexity: Table 3.1 demonstrates that our 2/4-global-head (G)FiSHformers and their hard versions obtain comparable or better PPLs than the corresponding 8 head MHA baseline on
Table 2: Machine translation BLEU scores of 2-global-head (G)FiSHformers compared to the 4-head baseline on the IWSLT14 De-En dataset. Our methods perform on par or better than the baseline while being more efficient.

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax 4 heads</td>
<td>34.42</td>
</tr>
<tr>
<td>Hard Fishformer 2 global heads</td>
<td>34.31</td>
</tr>
<tr>
<td>FiSHformer 2 global heads</td>
<td>34.38</td>
</tr>
<tr>
<td>Hard GFiSHformer 2 global heads</td>
<td>34.54</td>
</tr>
<tr>
<td>GFiSHformer 2 global heads</td>
<td>34.71</td>
</tr>
</tbody>
</table>

Table 3: Machine translation BLEU scores of 2-global-head (G)FiSH + linear transformers compared to the 4-head baseline linear on the IWSLT14 De-En benchmark. Our methods significantly outperform the linear baseline.

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear 4 heads</td>
<td>28.22</td>
</tr>
<tr>
<td>Hard LFiSHformer 2 global heads</td>
<td>30.24</td>
</tr>
<tr>
<td>LFiSHformer 2 global heads</td>
<td>29.63</td>
</tr>
<tr>
<td>Hard GLFiSHformer 2 global heads</td>
<td>29.20</td>
</tr>
<tr>
<td>GLFiSHformer 2 global heads</td>
<td>29.46</td>
</tr>
</tbody>
</table>

WikiText-103. Interestingly, the 2-global-head (G)FiSHformers perform on par with the 8-head baseline even though only 2 global attention matrices are used to span all local attention matrices, indicating that the attention matrices in MHA are indeed redundant and the representation capacity of local attention matrices in (G)FiSH, though being generated from only 2 global bases, is comparable to those in the 8-head MHA.

Efficiency: In Fig. 3A and 3B, we presents the reduction ratio of train and test FLOPS, respectively, of our 2-global-head GFiSHformer vs. the baseline 8-head MHA transformer as functions of model dimension $D$ and sequence length $N$. In Fig. 3C and 3D, we show the reduction ratio of model size and GPU memory usage at test time, respectively, of our 2-global-head GFiSHformer vs. the same baseline. We observe that the efficiency advantage of GFiSHformer over the baseline grows with $D$ and $N$, making it more suitable and superior for large-scale applications. Note that the model size in terms of the number of parameters does not depend on the sequence length $N$, and from our experiments, we observe that the GPU memory usage reduction ratio is almost the same for different sequence lengths. More efficiency analysis results on this language modeling task are provided in Section 4 and Appendix C. Also, Figure 5 in Appendix B.2 shows the train and test PPL of (G)FiSHformers and the MHA softmax transformers. In our experiment, 4-global-head GFiSHformers obtain the best validation PPL.

3.2 Machine Translation

In this section, we examine the performance of (G)FiSHformer on the neural machine translation task, an important task in natural language processing in which the sequence lengths of the input are not the same. We first compare (G)FiSHformers with the baselines MHA softmax transformers on the IWSLT’14 De-En [11] and then scale up our experiments to the WMT’14 En-De [45]. On these tasks, we calculate the BLEU scores for evaluation.

Models and baselines For the IWSLT’14 De-En task, we compare 2-global-heads (G)FiSHformers with the baseline 4-head softmax transformer. Each model consists of 12 layers, 6 layers for an encoder and the other 6 layers for a decoder. Our experiments follow the setting on fairseq. For the WMT’14 En-De task, we use similar models as in the IWSLT’14 De-En task. However, we compare (G)FiSHformers of 8 and 4 global heads with the 16-head MHA softmax baseline. Our training and model setting are the same as those in [53].

Results As shown in Table 2 and 4, (G)FiSHformers outperform or at least are on par with the baseline MHA softmax transformers. Again, these results indicate rich representations of the local attention matrices generated by (G)FiSH. Furthermore, (G)FiSHformer outperforming Hard (G)FiSHformer in all settings suggests the positive value of adding noise into the models to turn them into a proper probabilistic model. Nevertheless, it is worth noticing that Hard (G)FiSHformer is more efficient than (G)FiSHformer. Fig. 4 in Appendix B.1 summarizes the advantage in efficiency of 2-global-head GFiSHformer over the 4-head baseline on the IWSLT’14 De-En task. These advantages of GFiSHformer grow with the model dimension $D$.

3.3 Image Classification on ImageNet

The advantages of (G)FiSHformers also hold across different data modalities. To illustrate this point, in this section, we apply (G)FiSH to Swin transformer [44], a state-of-the-art vision transformer architecture, for the image classification task on the ImageNet dataset [20]. The baseline Swin-T we use has a total of 12 layers, across 4 stages of transformer blocks with 3, 6, 12, and 24 heads each. Our GFiSH Swin-T uses 6 and 12 global attention heads at the last two stages. The model and training follow the settings in [44]. We summarize our results in Table 5. Our GFiSHformer is only...
Table 4: BLEU scores of (G)FiSHformers, with various numbers of shared heads compared to the 16-head baseline on the WMT’14 En-De machine translation. Our methods obtained comparable/better results than the baseline while being more efficient.

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax 16 heads</td>
<td>29.38</td>
</tr>
<tr>
<td>Hard FiSHformer 8 global heads</td>
<td>29.32</td>
</tr>
<tr>
<td>FiSHformer 8 global heads</td>
<td>29.57</td>
</tr>
<tr>
<td>Hard GFISHformer 8 global heads</td>
<td>29.27</td>
</tr>
<tr>
<td>GFISHformer 8 global heads</td>
<td>29.42</td>
</tr>
<tr>
<td>Hard GFISHformer 4 global heads</td>
<td>28.97</td>
</tr>
<tr>
<td>GFISHformer 4 global heads</td>
<td>29.34</td>
</tr>
</tbody>
</table>

Table 5: ImageNet Image Classification accuracy scores, FLOPs, and number of parameters on Swin Transformer, comparing between baseline Swin-T and our GFISH Swin-T. Baseline results from [44] are provided in parentheses. The Swin-T baseline uses 12 and 24 attention heads in the last two stages while our GFISH Swin-T uses only 6 and 12 global attention heads in the last two stages.

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc top-1</th>
<th>Acc top-5</th>
<th>FLOPs (10^9)</th>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax-12/24</td>
<td>81.20</td>
<td>95.50</td>
<td>4618.24</td>
<td>28.3M</td>
</tr>
<tr>
<td>Hard-GFISH-6/12</td>
<td>81.11</td>
<td>95.44</td>
<td>4372.48</td>
<td>26.2M</td>
</tr>
</tbody>
</table>

Table 6: The GFISHformer vs. the baseline softmax transformer on the UEA Time Series Classification Archive benchmark [5]. The GFISHformer performs on par with the baseline while being more efficient. We also include the reported results from [83] and [81] (in parentheses) in addition to our reproduced results. The experiment setups and configurations for the baseline and our GFISHformer are the same as in [81] (for the PEMS-SF, SelfRegulationSCP2, UWaveGestureLibrary datasets) and [83] (for other tasks).

<table>
<thead>
<tr>
<th>Dataset/Model</th>
<th>BLEU score</th>
<th>GFiSHformer</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETHANOL CONCENTRATION</td>
<td>32.08 (33.70)</td>
<td>33.70</td>
</tr>
<tr>
<td>FACIAL DETECTION</td>
<td>68.70 (68.10)</td>
<td>68.57</td>
</tr>
<tr>
<td>HANDWRITING</td>
<td>32.08 (30.50)</td>
<td>31.55</td>
</tr>
<tr>
<td>HEARTBEAT</td>
<td>75.77 (77.60)</td>
<td>76.10</td>
</tr>
<tr>
<td>JAPANESE VOWELS</td>
<td>99.46 (99.40)</td>
<td>99.37</td>
</tr>
<tr>
<td>EMNLP ROTATION</td>
<td>82.66 (82.10)</td>
<td>82.66</td>
</tr>
<tr>
<td>SELFREGULATIONSCP1</td>
<td>91.46 (92.50)</td>
<td>90.56</td>
</tr>
<tr>
<td>SELFREGULATIONSCP2</td>
<td>54.72 (53.90)</td>
<td>54.81</td>
</tr>
<tr>
<td>SPOKEN ARABIC DIGITS</td>
<td>99.33 (99.30)</td>
<td>99.34</td>
</tr>
<tr>
<td>UWAVEGESTURELIBRARY</td>
<td>84.45 (85.60)</td>
<td>85.01</td>
</tr>
<tr>
<td>AVERAGE ACCURACY</td>
<td>72.07 (72.27)</td>
<td>72.17</td>
</tr>
</tbody>
</table>

slightly more efficient than the baseline in this case because the sequence length N per window for this task is small, i.e. $N = 49$. However, like with the previous language tasks and as pointed by formula of computational cost and model complexity reduction in Section 2.5, these advantages grow with larger $D$ and $N$.

3.4 UEA Time Series Classification

We compare the accuracy of the GFISHformer and the baseline softmax transformers trained on the UEA Time Series Classification Archive benchmark [5]. In Table 6, we show that GFISHformers perform on par with the baselines. For each classification task in this benchmark, the number of GFISHformer’s global heads is half the number of heads in the baseline softmax transformers. The experiment setups and configurations for the baseline and our GFISHformer are the same as in [81] (for the PEMS-SF, SelfRegulationSCP2, UWaveGestureLibrary datasets) and [83] (for other tasks).

3.5 Reinforcement Learning on the D4RL Benchmark

We also demonstrate the benefits of GFISHformers on reinforcement learning. In Table 7, we report the results of the GFISHformer and the softmax transformer trained for the continuous control tasks from D4RL benchmark [29]. The GFISHformer yields comparable results to the baseline while being more efficient. We also include the reported results from [81] (in parentheses) in addition to our reproduced results.

<table>
<thead>
<tr>
<th>Environment/Model</th>
<th>Baseline softmax</th>
<th>GFiSHformer</th>
</tr>
</thead>
<tbody>
<tr>
<td>HALFCEETAH</td>
<td>91.03 (83.80)</td>
<td>90.25</td>
</tr>
<tr>
<td>HOPPER</td>
<td>110.30 (104.40)</td>
<td>110.60</td>
</tr>
<tr>
<td>WALKER</td>
<td>108.70 (107.70)</td>
<td>108.30</td>
</tr>
<tr>
<td>MEDIUM-REPLAY</td>
<td>HOPPER</td>
<td>85.61 (79.70)</td>
</tr>
<tr>
<td>MEDIUM</td>
<td>HOPPER</td>
<td>63.44</td>
</tr>
<tr>
<td>MEDIUM-EXPERT</td>
<td>HOPPER</td>
<td>81.19 (79.00)</td>
</tr>
</tbody>
</table>

3.6 GFiSHformer is more effective than other methods for head-redundancy reduction

To further demonstrate the effectiveness of our method, we compare FiSHformers against the head-redundancy reduction method in [16] on the WMT’14 machine translation task. [16] proposes the adaptively sparse transformer (AST), reducing redundancy within each head by zeroing out low-attention scores. In comparison, the results further confirm the effectiveness of our method since the
Table 8: Layer-Average mean and variance of $L_2$ distances between heads of models trained for the WikiText-103 language modeling task.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax 8 heads</td>
<td>1.62</td>
<td>0.66</td>
</tr>
<tr>
<td>Linear 8 heads</td>
<td>1.90</td>
<td>0.06</td>
</tr>
<tr>
<td>GFiSHformer 2 global heads</td>
<td>2.93</td>
<td>2.62</td>
</tr>
<tr>
<td>GFiSHformer 4 global heads</td>
<td>3.59</td>
<td>3.95</td>
</tr>
<tr>
<td>GFiSHformer 6 global heads</td>
<td>3.37</td>
<td>2.78</td>
</tr>
</tbody>
</table>

Table 9: Layer-average number of principal components for 95% variance explained of the covariance of attention matrices (WikiText-103 models).

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax 8 heads</td>
<td>296</td>
</tr>
<tr>
<td>Linear 8 heads</td>
<td>436</td>
</tr>
<tr>
<td>GFiSHformer 2 global heads</td>
<td>895</td>
</tr>
<tr>
<td>GFiSHformer 4 global heads</td>
<td>1408</td>
</tr>
<tr>
<td>GFiSHformer 6 global heads</td>
<td>1228</td>
</tr>
</tbody>
</table>

BLEU score of FiSHformer and GFiSHformer are 27.26 and 27.67, respectively, better than that of AST, which is 26.93. All models share the same architecture, with 12 transformer layers, 6 encoder, and 6 decoder layers. Our FiSHformers have 4 global heads and 8 local heads per layer while 8 attention heads are used in each AST layer.

3.7 Beyond Multi-Head Softmax Transformers

We show that (G)FiSH can be applied on top of many transformer architectures to improve their performance including the linear transformers [37] and the SoTA transformer with noisy back-translation [26]. More results of combining (G)FiSH with efficient transformers are in the Appendix.

Applying (G)FiSH on Linear Transformers

Linear transformers [37] is a class of efficient transformers that linearize the softmax kernel in Eqs. 1 and 2 when computing attention matrices. We apply (G)FiSH on linear transformers trained for the IWSLT14 De-En machine translation task and summarize the results in Table 3. The empirical results verify that applying (G)FiSH using only 2-global heads on a 4-head linear transformer improves the accuracy of the baseline model.

(G)FiSH Improves the State-of-the-Art Noisy Back-Translation

We apply an 8-global-head Hard GFiSH on the transformers trained with noisy back-translation [26] for the WMT’14 En-De translation task and obtain the BLEU score of 33.45. This result is comparable to the SoTA result of 33.52 from the transformers trained with noisy back-translation but our model is more efficient.

4 Empirical Analysis

We study models trained for the WikiText-103 language modeling task in this section.

Efficiency Analysis

In this section, we further investigate the efficiency reduction of 2-global-head GFiSHformers over the 8-head baseline as a function of the number of heads in Fig. 7 and 8 and compare the efficiency of our FiSH-based models in Fig. 6. From Fig. 7 and 8, we observe that when using fewer number of global heads, GFiSHformers achieve significantly more computation reduction (in both training and inference). Furthermore, Fig. 6 shows that the efficiency measures, i.e. FLOPs, model size, and GPU memory usage, of all FiSH-based models we study in this paper follow similar patterns.

FiSHformer Helps Reducing Head Redundancy

We show that (G)FiSHformers attain more significant distances between heads than the baseline. Thus, our models capture more diverse patterns across heads than the baseline. For a given pre-trained model, we compute the pair-wise $L_2$ distances between heads in the same layer. We show the layer-average mean and variance of distances between heads in GFiSHformers compared with those in the MHA softmax baselines in Table 8. We provide additional results for Hard GFiSHformers and Hard FiSHformer in Table 12 in the Appendix.

Eigen Analysis

We show that heads in GFiSHformer lie on a higher-dimensional subspace compared to those in FiSHformer. This justifies our use of nonlinearity mapping to generate local heads from global heads. Using a pre-trained model, we first compute the covariance matrix of the vectorized attention scores of the $l$-th layer: $C_l = \frac{1}{N_T} \sum_{t=1}^{N_T} \sum_{h=1}^{H} (A_{l,m}^{i,h})(A_{l,m}^{i,h})^\top$. We use spectral decomposition to represent $C_l$ in terms of eigenvalues and eigenvectors, namely, $C_l = \sum_{i=1}^{n^2} \lambda_i v_i v_i^\top$. Without losing generality, we assume that eigenvalues are sorted in descending order. We illustrate the layer-average number of principle components that are needed to explain 95% variance in Table 9. Interestingly, Table 9 shows that attention matrices in all of our proposed FiSH-based models lie on higher-dimensional subspace than those in the baseline MHA softmax transformers, which indicates that our models achieve better representational capacity than the baseline, confirming the advantage of (G)FiSH over MHA. Table 13 in the Appendix provide additional results for Hard (G)FiSHformer.
Table 10: Perplexity (PPL) on WikiText-103 of 2-global-head FiSHformer vs. 2-global-head MiSHformer compared to the 8-head baseline. MiSHformer attains worse PPL than FiSHformer.

<table>
<thead>
<tr>
<th>Method</th>
<th>Valid PPL</th>
<th>Test PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax 8 heads</td>
<td>33.15</td>
<td>34.29</td>
</tr>
<tr>
<td>FiSHformer 2 global heads</td>
<td>34.01</td>
<td>34.96</td>
</tr>
<tr>
<td>MiSHformer 2 global heads</td>
<td>35.11</td>
<td>36.28</td>
</tr>
</tbody>
</table>

Table 11: Perplexity on WikiText-103 of GFiSHformer with various number of global heads compared with the 8-head baseline.

<table>
<thead>
<tr>
<th>Method</th>
<th>Valid PPL</th>
<th>Test PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax 8 heads</td>
<td>33.15</td>
<td>34.29</td>
</tr>
<tr>
<td>GFiSHformer 6 global heads</td>
<td>32.80</td>
<td>33.80</td>
</tr>
<tr>
<td>GFiSHformer 4 global heads</td>
<td>32.68</td>
<td>33.71</td>
</tr>
<tr>
<td>GFiSHformer 2 global heads</td>
<td>33.21</td>
<td>34.48</td>
</tr>
</tbody>
</table>

Admixture vs. Mixture of Heads We compare the transformer with a mixture of heads and the transformer with an admixture of heads. We show that the transformer with a mixture of heads yields worse accuracy. We summarize our results on the WikiText-103 language modeling task in Table 10.

Ablation Study on the Effect of the Number of Global Heads on FiSH-based Models We investigate the accuracy, efficiency, and representation capacity of FiSH-based models under different numbers of global heads on the WikiText-103 language modeling task. Since GFiSH obtains the best accuracy on this task, we use GFiSH as a study case in this section and report our results on accuracy in Table 11. Ablation results on efficiency are summarized in Fig. 7, 8 in Appendix C, and ablation results on representation capacity are provided in Table 8, 9.

5 Related Work

Efficient Transformers Efficient transformers have been developed to reduce the quadratic computational and memory cost of transformers [62]. A class of efficient transformers are sparse transformers which design the attention matrix to have sparse structure [55, 42, 58, 12, 7]. Another class of efficient transformers are models that integrate different access patterns for better coverage [12, 33]. These access patterns can also be learned from the data [40, 71]. In other works, a side memory module is used to access multiple tokens simultaneously [41, 69, 3, 7]. Recently, low-rank and kernelization methods have been proposed to improve the computational and memory efficiency of computing self-attention [74, 79, 37, 14, 50, 67, 52, 56]. Our (G)FiSHformers are complementary to these methods.

Redundancy in Transformers Most of the neurons and heads in the pre-trained transformer are redundant and can be pruned when applied on downstream tasks [18, 47, 23]. The contextualized embeddings in pre-trained networks under this redundancy have also been studied to demonstrate that the representations learned within these models are highly anisotropic [48, 27]. Knowledge distillation and sparse approximation have also been used to enhance the efficiency of transformers, including [65, 70, 78, 64]. Our FiSH-based approach are complementary to these methods.

Mixture Models for Transformers Recently, mixture models have been employed to study and enhance transformers. Among these works is switch transformers [28] that uses the routing algorithm in Mixture of Experts (MoE) to decrease the communication and computational costs in transformers. [49] derives a GMM for each attention head. Other works that combine mixture models with transformers include [51, 13, 31, 36].

6 Concluding Remarks

In this paper, we proposed the FiSHformer, a class of transformers that samples local attention matrices from a finite admixture of global attention matrices. FiSHformers and their generalized version GFiSHformers attain better computational cost and model complexity than their baseline MHA softmax transformer counterparts. Furthermore, (G)FiSHformers help increase the diversity between attention heads. It is worth noting that there is no potential negative societal impacts of FiSHformers. Also, global attention matrices in FiSHformers currently do not have any structure, and this is a limitation of our model. It is interesting to impose additional structures such as low-rank and sparsity into the global attention matrices. Finally, establishing theoretical guarantee for optimization algorithms [25, 24, 34] for solving FiSHformers is also an important future direction to improve the efficiency of FiSHformers.

Acknowledgements

This material is based on research sponsored by the AFOSR MURI FA9550-18-1-0502, the ONR grant N00014-20-1-2093, the MURI N00014-20-1-2787, and the NSF under Grant# 2030859 to the Computing Research Association for the CIFellows Project (CIF2020-UCLA-38). NH acknowledges support from the NSF IFML 2019844 and the NSF AI Institute for Foundations of Machine Learning.
References


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]
Supplement to "Improving Transformer with an Admixture of Attention Heads"

A Experiment Details

In this section, we provide model and training details for experiments in Section 3. In our experiments, we consider the number of global attention matrices as a hyper parameter to finetune. We observe that in all of our experiments, choosing the global attention matrices in FiSH to be 1/4 and 1/2 of the number attention heads in the original MHA results in models with good accuracy and efficiency. All of our experiments are conducted on a server with 4 NVIDIA A100 GPUs.

A.1 Language Modeling

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 [46, 66] 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, 66].

A.2 Machine Translation

Datasets and metrics The dataset IWSLT’14 De-En contains about $170K$ training sentence pairs, $7K$ validation pairs, and $7K$ test pairs. In this task, the model does the translation from German to English. The WMT dataset is a rich-resource English-German machine translation dataset, containing about $4.5M$ training sentence pairs. Validation and test data are from the corresponding newest data. The BLEU score [54] is used to measure the performance of the trained model.

A.3 Image Classification

Datasets and metrics The ImageNet dataset [63] contains about $1.281M$ training images and $50K$ validation images, the model learns to predict which one of 1000 classes an image belongs to. Our Swin Transformer [44] experiments are based on the public code https://github.com/microsoft/Swin-Transformer, we implemented our Hard GFiSH models with the Swin-T version. We add our global heads to the last 8 of the total 12 layers of the model, on each layer we set the number of global heads to half the number of heads, which are 6 and 12 global heads for layers with 12 and 24 heads, respectively. Our experiments were conducted on a server with 1 NVIDIA RTX 3090. We set the batch size to 128 and the learning rate to 1.25e-4, all models are trained with single precision.

A.4 UEA Time Series Classification

Datasets and metrics There are 30 datasets in the benchmark [5]. Following [81], to evaluate our models on temporal sequences, we choose 10 datasets, which vary in input sequence lengths, the number of classes, and dimensionality. We report the test accuracy as an evaluation for the benchmark.

Models and baseline The experiment setups and configurations for the softmax and our models are the same as in [81] 1 (for the PEMS-SF, SelfRegulationSCP2, UWaveGestureLibrary datasets) and [83] 2 (for the other tasks). In all models, the number of heads is 8, whereas the model dimension and number of transformer layers are varied. Our GFiSHformer uses 4 global heads to produce 8 local heads.

A.5 Reinforcement learning on the D4RL benchmark

Datasets and metrics The D4RL benchmark [29] contains the continuous control tasks for offline reinforcement learning. We adapt the selection from [81], including HalfCheetah, Hopper, and Walker as experiment environments and Medium-Expert, Medium, and Medium-Replay as behavior policies.

Models and baseline The softmax baseline trained on this benchmark adopts the configuration from [81], with 3 transformer layers and 4 heads per layer. The 2-global-head GFiSHformer also shares the same configuration and training set-ups.

---

1 Implementation available at https://github.com/thuml/Flowformer.
A.6 Applying (G)FiSH on linear transformers

Here we provide the detailed implementation of (G)LFiSHformer, i.e. (G)FiSH + linear transformer, discussed in Section 3.7 and Table 3 in the main text. The linear transformer reduces the quadratic computational cost of self-attention to linear complexity, in terms of the sequence length, by linearizing the softmax kernel [37]. We combine (G)FiSHformer with linear transformer by generating the global $K^T V$ and then sampling the local $K^T V$ from the global ones, resulting in the Transformer with a Linear Finite Admixture of Shared Heads (LFiSHformer). Similar to (G)FiSHformer, we derive four different LFiSH-based transformers: LFiSHformer, Hard LFiSHformer, Generalized LFiSHformer (GLFiSHformer), and Hard Generalized LFiSHformer (Hard GLFiSHformer). Our LFiSH-variants improve the performance of the linear baseline, as demonstrated in Table 3.

B Additional Experimental Results

B.1 A Comparison on the Model Efficiency for the IWSLT14 De-En Machine Translation Task

Fig. 4 summarizes the advantage in efficiency of 2-global-head GFiSHformer over the 4-head baseline on the IWSLT’14 De-En task.

![Figure 4: (Left) Training (A) and Inference (B) real time ratios between a 2-global-head GFiSHformers with 4-head MHA baselines across different model dimensions $D$ trained on the IWSLT14 De-En machine translation task. (Right) GPU memory usage at train time (C) and test time (D) and number of parameters (E) ratios between 2-global-head GFiSHformers with 4-head MHA baselines across different model dimensions $D$. 2-global-head GFiSHformers are significantly more efficient than the baseline as $D$ increase, indicating the benefits of our method for long-range and large-scale tasks. Note that the ratios do not change much when $N$ increase for this task.](image)

B.2 Train and validation PPL of models trained for the WikiText-103 language modeling task

Figure 5 shows the train and valid PPL of 4-global-head FiSH-based models vs. 8-head MHA Transformer trained for the WikiText-103 language modeling task.

B.3 More Results to Show that FiSH Former Helps Reducing Head Redundancy

Table 12 presents the layer-average mean and variance of distances between heads in Hard FiSHformers and Hard GFiSHformers compared with those in the MHA softmax baselines. Models are trained for the WikiText-103 language modeling task.

![Figure 5: Train and validation PPL of 4-global-head FiSH-based models vs. 8-head MHA Transformer trained for the WikiText-103 language modeling task.](image)
Table 12: Layer-Average mean and variance of $L_2$ distances between heads of models trained for the WikiText-103 language modeling task.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax 8 heads</td>
<td>1.62</td>
<td>0.66</td>
</tr>
<tr>
<td>Hard FiSHformer 4 global heads</td>
<td>1.75</td>
<td>1.38</td>
</tr>
<tr>
<td>Hard GFiSHformer 2 global heads</td>
<td>2.99</td>
<td>2.79</td>
</tr>
<tr>
<td>Hard GFiSHformer 4 global heads</td>
<td>3.58</td>
<td>3.01</td>
</tr>
<tr>
<td>Hard GFiSHformer 6 global heads</td>
<td>2.90</td>
<td>1.71</td>
</tr>
</tbody>
</table>

Table 13: Layer-average number of principle components for 95% variance explained of models trained for the WikiText-103 language modeling task.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax 8 heads</td>
<td>296</td>
</tr>
<tr>
<td>Hard FiSHformer 4 global heads</td>
<td>302</td>
</tr>
<tr>
<td>Hard GFiSHformer 2 global heads</td>
<td>1161</td>
</tr>
<tr>
<td>Hard GFiSHformer 4 global heads</td>
<td>1317</td>
</tr>
<tr>
<td>Hard GFiSHformer 6 global heads</td>
<td>1102</td>
</tr>
</tbody>
</table>

B.4 More Results on Eigen Analysis

Table 13 presents the layer-average number of principle components for 95% variance explained of the covariance of attention matrices in Hard FiSHformers and Hard GFiSHformers trained for the WikiText-103 language modeling task compared with those in the MHA softmax baselines. Models are trained for the WikiText-103 language modeling task.

C Efficiency Analysis

In this section, we present the efficiency improvement of (G)FiSHformers over the MHA baseline on the WikiText-103 language modeling task. We show that the advantage in the efficiency of (G)FiSHformers increases significantly as the model dimension $D$ and sequence length $N$ grows, making (G)FiSHformers more suitable and superior for large-scale applications. In our analysis, we report the number of FLOPs, the number of model parameters, and memory usage (bytes) as the measures of model efficiency.

Analysis setting We investigate the benefits of our model computation and memory reduction through different $D \in \{256, 512, 1024, 2048, 4096\}$ and $N \in \{128, 256, 512, 1024, 2048, 4096, 8192\}$. For the FLOP calculation, we use fvcore. Another notice is that, for model complexity, we distinguish between input-embedding parameters and non-embedding parameters. Input-embedding parameters are used to represent the inputs before sending them to the model. Non-embedding parameters are the parameters of the main model. Since our method aims at reducing the size of the transformer model, it is important to compare the reduction in non-embedding parameters. Hence, we report both model’s total parameters and non-embedding parameters in this analysis. All measurements are calculated when running the model through data of batch size 1.

FiSH-based models significantly improves computational cost

GFiSHformer benefits computation in large-scale tasks. By showing the FLOP ratios between 2-global-head GFiSHformers with the 8-head MHA baseline across different model dimensions and sequence lengths, Figure 3 indicates that our model requires significantly less computation than the baseline. Especially for both training and inference, this substantially grows with the increases of $D$ and $N$ (up to more than 30% reduction), which makes our method more preferable for long-range and large-scale tasks.

FiSH-based variants share comparable advantages in computation saving. For further analysis, we compare the FLOP of FiSHformers, Hard FiSHformers, GFiSHformers, and Hard GFiSHformer. Figure 6 shows that given a $D$, when $N$ is increased, FiSH-based models have comparable training and inference computation (FLOP). Since we have shown that GFiSHformer benefits computation, the FLOP comparison among FiSH-based models further confirms that our methods have a substantial advantage in computational saving.
Figure 6: Training (Left) and Inference (Right) FLOPS (B) of 2-global-head FiSHformers, Hard FiSHformers, GFiSHformers, and Hard GFiSHformer. FiSH-based models have comparable computational costs trained on the WikiText-103 language modeling task. Here the model size is 1024.

Figure 7: Training (Left) and Inference (Right) FLOP ratio between the GFiSHformers and the 8-head MHA baseline trained on the WikiText-103 language modeling task, as the number of heads vary in \{2, 4, 6\}. A decrease in the number of heads leads to a significant reduction in computational cost at each sequence length. Here, the model dimension is 1024.

GFiSHformer computation efficiency rapidly increases as the number of global heads decreases. We compare the computational-cost reduction of GFiSHformer as the global heads vary. Figure 7 shows the FLOP ratio of GFiSHformer with 2/4/6-global-head versus the 8-head baseline for a given \(D\) and various sequence lengths. As the number of heads decreases, GFiSHformers achieve significant computation reduction (in both training and inference) (Figure 7).

(G)FiSHformers improves memory usage and model complexity. In addition to the computational saving, our method achieves significant benefits in memory cost (at test time) and model complexity (total/non-embedding parameters) over the baseline. Following the computation analysis, we first present the advantage of the GFiSHformers compared to the 8-head MHA Transformer. Figure 8 shows the number-of-parameter-ratio (Left) and the memory-ratio (Right) of the GFiSHformers with 2/4/6 global heads and the 8-head baseline. At a fixed sequence length \(N = 1024\), as we scale up the model dimensions, our method becomes significantly more beneficial than MHA Transformer, indicating the advantage of GFiSHformer in large-scale applications.

Secondly, we show that FiSH-based models are comparable in model size and memory saving, which further indicates that all variants of FiSHformers benefit space complexity. Figure 6 shows that 2-head FiSHformers, Hard FiSHformers, GFiSHformers, and Hard GFiSHformer have a comparative number of total/non-embedding parameters and memory usage at different model dimensions, for a given sequence length.

Finally, we examine the model space complexity of a Fishformer variant as the number of global heads vary. Figure 8 shows a significant increase in total (Left)/non-embedding parameters (Middle) and memory usage (Right) reduction of 2/6/8-global heads GFiSHformers, as we increase \(D\). While only having the fewest heads, 2-head GFiSHformer is the most efficient model (lowest metric ratio
Figure 8: Number-of-parameter ratio (Left) and the memory ratio (at test time) (Right) between the GFiSH- formers and the 8-head MHA Transformer trained on the WikiText-103 language modeling task. Our method achieve significant reduction in memory cost and model complexity over the baseline as we scale up the model dimension. $D \in \{256, 512, 1024, 2048 \text{ to } 4096\}$. Here, the sequence length is 1024, and the number of global heads is chosen to be 2, 4 and 6.

with the 8-head MHA baseline) and achieves comparable performance with the baseline, for model dimension, and sequence length are 128 and 256 respectively, as indicated in Table 3.1

D An Analysis on the Computational Complexity and the Number of Parameters in FiSH and the Softmax Attention

In this section, we compare the computational complexity and the number of parameters in the FiSH with $M$ global attention heads and $H$ local attention heads to the $H$-head baseline MHA softmax transformer. Following the same notation in Section 1.1 in the main text, we let $D_x$, $N$, and $D$ be the input dimension, the input length, and the model/feature dimension, respectively. To simplify the computation, we also do not take the softmax operator into account.

D.1 Computational Complexity

(i) Softmax $H$-head attention: The number of computations needed to compute attention matrices in a softmax $H$-head attention is $N^2 H (2D - 1) + 2 NHD (2D_x - 1)$.

Explanation: To calculate the query matrix $Q$ and the key matrix $K$ in Step 1 in Section 1.1 at each head, we need $2NDD_x$ multiplications and $2ND(D_x - 1)$ additions. In total, these need $2NDD(2D_x - 1)$ computations. Next, to compute the product $QK^\top$ in Eqn. (1), we need $N^2D$ multiplications and $N^2(D - 1)$ additions. In total, computing an attention matrix in Eqn. (1) at each head requires $2NDD(2D_x - 1) + N^2D + N^2(D - 1) = N^2(2D - 1) + 2NDD(2D_x - 1)$ computations. The total computation needed to compute attention matrices at $H$ heads is then $N^2H(2D - 1) + 2NHD(2D_x - 1)$.

(ii) FiSH with $M$ global attention heads and $H$ local attention heads: The number of computations needed to compute attention matrices in a FiSH with $M$ global attention heads and $H$ local attention heads is $2(D + H)M - H]N^2 + 2NMD(2D_x - 1)$.

Explanation: Similar to the above derivation, $M$ global attention matrices need $N^2M(2D - 1) + 2NMDD_x$ computations. The $H$ local attention matrices need $H(MN^2 + (M - 1)N^2) = H(2M - 1)N^2$. There are also $MN^2$ computations needed to add noise to $M$ global attention matrices. Thus, the number of computations needed to compute attention matrices in a FiSH with $M$ global attention heads and $H$ local attention heads is $N^2M(2D - 1) + 2NMD(2D_x - 1) + H(2M - 1)N^2 + MN^2 = [2(D + H)M - H]N^2 + 2NMD(2D_x - 1)$.

Soft-max H-head attention versus FiSH with $M$ global attention heads and $H$ local attention heads: Given the results in (i) and (ii), when compared to the baseline softmax $H$-head attention, our FiSH with $M$ global attention heads and $H$ local attention heads saves $[2(H - M)D - 2MH]N^2 + 2(H - M)D(2D_x - 1)N$ computations in a forward pass. When $N$ is large, this difference is significant.

D.2 The Number of Parameters

(iii) Softmax $H$-head attention: The number of parameters needed to compute the attention matrices in a softmax $H$-head attention is $2HDD_x$.
We denote FiSH with \( M \) global attention heads and \( H \) local attention heads: The number of parameters in a FiSH with \( M \) global attention heads and \( H \) local attention heads is \( 2MDD_x + HM + M \).

Explanation: \( 2MDD_x \) parameters is from the linear projects to calculate the query matrix \( Q \) and the key matrix \( K \) in Step 1 in Section 1.1.

\( \text{(iv)} \) FiSH with \( M \) global attention heads and \( H \) local attention heads: Given the results in (iii) and (iv), when compared to the baseline softmax attention heads: 

Softmax H-head attention versus FiSH with \( M \) global attention heads and \( H \) local attention heads: Given the results in (iii) and (iv), when compared to the baseline softmax H-head attention, our FiSH with \( M \) global attention heads and \( H \) local attention heads saves \( 2(H - M)DD_x - HM - M \) parameters. When \( D \) is large, this saving is significant.

E Proofs

In this appendix, we provide proof for Lemma 1.

E.1 Proof of Lemma 1

We denote 

\[
\piG(x) := \int f(x - \theta)dG(\theta) = \int \phi(x|\theta, \sigma^2)I) dG(\theta),
\]

for all \( x \in \mathbb{R}^d \) where \( f(x) = \frac{1}{(2\pi)^{d/2}} \exp \left(-\frac{|x|^2}{2\sigma^2}\right) \) for given \( \sigma > 0 \). From the work of [4], the space of Gaussian mixtures is dense in the space of continuous probability measures. Therefore, we can find probability distribution \( G_1 \) such that

\[
\sup_{x \in \mathbb{R}^d} |p(x) - pG_1(x)| \leq \epsilon/2. \tag{11}
\]

To obtain the conclusion of the lemma, it is sufficient to prove that we can find a probability measure \( G_2 \) with at most \( K \) supports where \( K \leq (C \log(1/\epsilon))^d \) for some universal constant \( C \) such that

\[
\sup_{x \in \mathbb{R}^d} |pG_1(x) - pG_2(x)| \leq \epsilon/2. \tag{12}
\]

Our technique for proving the above approximation bound relies on Lemma A.1 in [30]. In particular, that lemma entails that for any \( k \geq 1 \) there exists a probability distribution \( G_2 \) with at most \((2k - 2)^d\) supports such that

\[
\int \theta^\alpha d(G_1 - G_2)(\theta) = 0, \tag{13}
\]

for any \( \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_d) \in \mathbb{N}^d \) such that \( 0 \leq |\alpha| = \sum_{j=1}^d \alpha_j \leq 2k - 2 \). Here, \( \theta^\alpha = \prod_{j=1}^d \theta_j^{\alpha_j} \).

Now, for any \( M \geq 2a\sqrt{d} \), we have \( |x - \theta| \geq |x| - |\theta| > M - a\sqrt{d} > M/2 \) as long as \( |x| > M \) and \( \theta \in [-a, a]^d \). It indicates that

\[
\sup_{\|x\| > M} |pG_1(x) - pG_2(x)| = \sup_{\|x\| > M} \left| \int f(x - \theta)d(G_1 - G_2)(\theta) \right|
\leq \sup_{\|x\| > M} \int \frac{1}{(2\pi)^{d/2}} \exp \left(-\frac{|x - \theta|^2}{2\sigma^2}\right) d(G_1 + G_2)(\theta)
\leq \frac{2}{(\sqrt{2\pi})^d} \exp \left(-\frac{M^2}{8\sigma^2}\right). \tag{14}
\]

On the other hand, for any \( k \geq 1 \) we also have that

\[
\sup_{\|x\| \leq M} |pG_1(x) - pG_2(x)| = \sup_{\|x\| \leq M} \left| \int f(x - \theta)d(G_1 - G_2)(\theta) \right|
\leq \sup_{\|x\| \leq M} \left| \int \left( f(x - \theta) - \sum_{j=0}^{k-1} \frac{(-1)^j |x - \theta|^{2j}}{(\sqrt{2\pi})^d \sigma^{d+2j} j!} \right) d(G_1 - G_2)(\theta) \right|, \tag{15}
\]
where the final inequality stems from
\[
\int \sum_{j=0}^{k-1} \frac{(-1)^j \|x - \theta\|^2j}{(\sqrt{2\pi})^d \sigma^d + 2j!} d(G_1 - G_2)(\theta) = 0,
\]
which is due to Eqn. (13).

To further bound the right-hand-side (RHS) of Eqn. (15), we use the following inequality:
\[
\exp(y) - \sum_{j=0}^{k-1} (y/j) \leq |y|^k/k!
\]
for any \( y \in \mathbb{R} \). Since \( k! \geq (k/e)^k \) for any \( k \geq 1 \), the above bound can be rewritten as
\[
\exp(y) - \sum_{j=0}^{k-1} (y/j) \leq \frac{|ye|^k}{k^k}.
\]
Further simplification of Eqn. (15) leads to
\[
\sup_{\|x\| \leq M} |pG_1(x) - pG_2(x)| \leq \sup_{\|x\| \leq M} \int \left| \frac{(-1)^j \|x - \theta\|^2j}{(\sqrt{2\pi})^d \sigma^d + 2j!} \right| d(G_1 + G_2)(\theta)
\]
\[
\leq 2 \sup_{\|x\| \leq M, \theta \in [-a, a]^d} \left| \frac{(-1)^j \|x - \theta\|^2j}{(\sqrt{2\pi})^d \sigma^d + 2j!} \right|
\]
\[
\leq \sup_{\|x\| \leq M, \theta \in [-a, a]^d} \frac{e^k \|x - \theta\|^{2k}}{\sigma^{2k}(2k)^k},
\]
where the final inequality is based on an application of inequality (16) with \( y = -\|x - \theta\|^2/(2\sigma^2) \).

For \( \|x\| \leq M \) and \( \theta \in [-a, a]^d \), we have \( |x - \theta| \leq \|x\| + \|\theta\| \leq M + a\sqrt{d} \). Therefore, we further have
\[
\sup_{\|x\| \leq M} |pG_1(x) - pG_2(x)| \leq \sup_{\|x\| \leq M, \theta \in [-a, a]^d} \frac{e^k \|x - \theta\|^{2k}}{\sigma^{2k}(2k)^k} \leq \frac{e^k(M + a\sqrt{d})^{2k}}{\sigma^{2k}(2k)^k}.
\]
When \( M \geq 2a\sqrt{d} \), we have \( M + a\sqrt{d} \leq \frac{3M}{2} \) and the above bound leads to
\[
\sup_{\|x\| \leq M} |pG_1(x) - pG_2(x)| \leq \frac{(9e)^kM^{2k}}{(8\sigma^2)^k}.
\]
By choosing \( M^2 = 8\sigma^2 \log(1/\epsilon') \) for some \( \epsilon' > 0 \), the bounds in Eqns. (14) and (17) become
\[
\sup_{\|x\| \leq M} |pG_1(x) - pG_2(x)| \leq \frac{2}{(\sqrt{2\pi\sigma})^d} \epsilon',
\]
\[
\sup_{\|x\| > M} |pG_1(x) - pG_2(x)| \leq \frac{(9e)^k(\log(1/\epsilon'))^k}{k^k}.
\]
As long as we choose \( k = 9e^2 \log(1/\epsilon') \) and \( \epsilon' \leq 1 \), we have
\[
\sup_{\|x\| > M} |pG_1(x) - pG_2(x)| \leq e^{-k} = e^{-9e^2 \log(1/\epsilon')} = (\epsilon')^{9e^2} \leq \epsilon'.
\]
By choosing \( \epsilon' = \frac{\epsilon}{2\max\left(\frac{\epsilon}{\sqrt{2\pi\sigma}}\right)^d} \), the results from Eqns. (18) and (19) indicate that
\[
\sup_{\|x\| \leq M} |pG_1(x) - pG_2(x)| \leq \frac{\epsilon}{2}, \quad \text{and} \quad \sup_{\|x\| > M} |pG_1(x) - pG_2(x)| \leq \frac{\epsilon}{2}.
\]
Therefore, if we choose \( M = 8\sigma^2 \log\left(\frac{2\max\left(\frac{\epsilon}{\sqrt{2\pi\sigma}}\right)^d}{\epsilon}\right) \) and \( k = 9e^2 \log\left(\frac{2\max\left(\frac{\epsilon}{\sqrt{2\pi\sigma}}\right)^d}{\epsilon}\right) \), we have
\[
\sup_{x \in \mathbb{R}^d} |pG_1(x) - pG_2(x)| \leq \frac{\epsilon}{2}.
\]
It indicates that we obtain the conclusion of claim (12) by choosing \( K = (2k - 2)^d \leq (18e^2 \log\left(\frac{2\max\left(\frac{\epsilon}{\sqrt{2\pi\sigma}}\right)^d}{\epsilon}\right))^d \). As a consequence, we obtain the conclusion of the lemma.