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

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

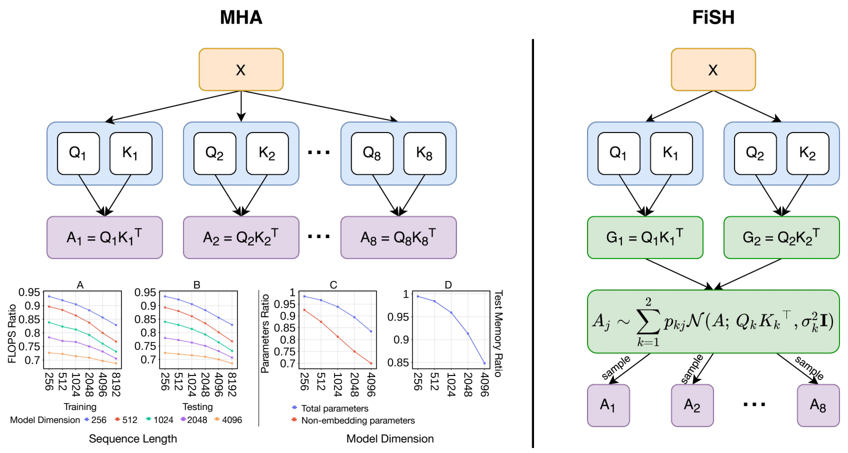

Figure 1: Our proposed finite admixture model of shared heads (FiSH) vs. the standard multi-head (MHA) attention. FiSH samples local attention matrices from a finite admixture of global attention matrices. Compared to MHA, FiSH is more efficient, saving computation and memory (See Fig. 3 and Section 4).

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 $\mathbf{X} := [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N\times D_x}$ 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 $\mathbf{X}$ 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

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

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

$$\boldsymbol{h}_i = \sum_{j=1}^{N} \mathrm{softmax}\Big(\boldsymbol{q}_i^\top \boldsymbol{k}_j/\sqrt{D}\Big)\boldsymbol{v}_j. \tag{2}$$

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 $\mathbf{A}$ learns from training determines the ability of the self-attention to capture contextual representation for each token.

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

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

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.

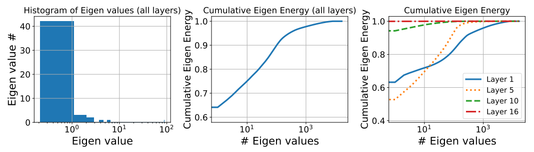

Figure 2: (Left) Histogram of the top 50 eigen values and (Middle, Right) cumulative sum of eigen values of the layer-average attention scores covariance matrix.

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 transformer. Our contribution is three-fold:

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 $\mathbf{A}_j$ denote the attention matrix at the j^{th} head, $j = 1, 2, \ldots, H$. From a probabilistic viewpoint, to have diversity among $\mathbf{A}_1, \ldots, \mathbf{A}_H$, we can assume that $\mathbf{A}_j$ comes from a distribution $\mathbb{P}_j$ 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 $\sum_{i=1}^{M} p_i \mathcal{N}(\theta_i, \sigma^2 \mathbf{I}_{D'})$ where $p_1, \ldots, p_K$ are weight parameters and $\theta_1, \ldots, \theta_K$ are location parameters that satisfy the following inequality*

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

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

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 $p_{1j}, \ldots, p_{M_j j}$, and location parameters $\theta_{1j}, \ldots, \theta_{M_j j}$ such that the mixtures $\mathbb{P}'_j = \sum_{i=1}^{M_j} p_{ij} \mathcal{N}(\theta_{ij}, \sigma^2 \mathbf{I}_{D'})$ can approximate the distribution P_j up to a given accuracy ϵ. However, these approximations still involve $\prod_{j=1}^{H} M_j$ 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 $(\theta_{1j}, \ldots, \theta_{M_j j}) = (\theta_1, \ldots, \theta_M)$ 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 $\mathbf{X} \in \mathbb{R}^{N \times J}$ is given by

$$\boldsymbol{x}_j \sim \sum_{k=1}^{M} p_k f(\boldsymbol{x}; \theta_k), \ \ \sum_{k=1}^{M} p_k = 1, \ \ p_k \geq 0, \tag{4}$$

where $\boldsymbol{x}_j \in \mathbb{R}^N$ is the j-th row of $\mathbf{X}$ 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 θ_k denotes the parameter values for the k-th component.

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

$$\boldsymbol{x}_j \sim \sum_{k=1}^{M} p_{kj} f(\boldsymbol{x}; \theta_k), \ \ \sum_{k=1}^{M} p_{kj} = 1, \ \ p_{kj} \geq 0. \tag{5}$$

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 $\mathbf{A}_j$ follows finite admixture distribution of M components given by

$$\mathbf{A}_j \sim \sum_{k=1}^{M} p_{kj} f(\mathbf{A}; \theta_k), \ \ \sum_{k=1}^{M} p_{kj} = 1, \ \ p_{kj} \geq 0. \tag{6}$$

Here $M < H$ and $f(\mathbf{A}; \theta_k)$ are chosen probability measures. In particular, we choose $f(\mathbf{A}; \theta_k)$ to be Gaussian distributions $\mathcal{N}(\mathbf{A}; \mathbf{G}_k, \mathbf{\Sigma}_k)$, where $\mathbf{G}_k = \mathbf{Q}_k {\mathbf{K}_k}^\top$ and $\mathbf{\Sigma}_k = \sigma_k^2 \mathbf{I}$ are the cluster means and coveriances, 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 $\mathbf{A}_j$ at the j^{th} head are sampled from the following finite admixture model:*

$$\mathbf{A}_j \sim \sum_{k=1}^{M} p_{kj} \mathcal{N}(\mathbf{A};\ \mathbf{Q}_k {\mathbf{K}_k}^\top, \sigma_k^2 \mathbf{I}), \ \ \sum_{k=1}^{M} p_{kj} = 1, \ \ p_{kj} \geq 0. \tag{7}$$

In FiSH defined in Def. 1, we call $\{\mathbf{G}_k = \mathbf{Q}_k\mathbf{K}_k{}^\top\}_{k=1,\dots,M}$ global attention matrices and $\{\mathbf{A}_j\}_{j=1,\dots,H}$ local attention matrices. FiSH computes M global attention matrices $\mathbf{G}_k$, and H local attention matrices $\mathbf{A}_j$ 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, e.g. $H = 8$, attention matrices, each of which requires $\mathcal{O}(N^2)$ 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 $\mathcal{O}(N^2)$ computational and memory costs. Then FiSH combines those global attention matrices to form a FAM from which H, e.g. $H = 8$, 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 $\mathcal{N}(\mathbf{A};\, \mathbf{Q}_k\mathbf{K}_k{}^\top, \sigma_k^2\mathbf{I})$ 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 $\mathbf{A}_j$ can be written in a differentiable form as follows:

$$\mathbf{A}_j = \sum_{k=1}^{M} p_{kj}(\mathbf{Q}_k\mathbf{K}_k{}^\top + \sigma_k \odot \epsilon_j),\ \ \epsilon_j \sim \mathcal{N}(0, \mathbf{I}),\ \ \sum_{k=1}^{M} p_{kj} = 1,\ \ p_{kj} \geq 0. \tag{8}$$

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 $\mathbf{A}_j$ in Hard FiSHformer are then calculated as

$$\mathbf{A}_j = \sum_{k=1}^{M} p_{kj}\mathbf{Q}_k\mathbf{K}_k{}^\top,\ \ \sum_{k=1}^{M} p_{kj} = 1,\ \ p_{kj} \geq 0. \tag{9}$$

Remark 3 (Discriminative Relaxation)**.** *To take the advantage of learning from data, the convex combination condition of p_{kj}, i.e. $\sum_{k=1}^{M} p_{kj} = 1,\ \ p_{kj} \geq 0$, 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)**.** *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 $\mathbf{A}_j$ in MiSHformer are given by*

$$\mathbf{A}_j = \sum_{k=1}^{M} p_k(\mathbf{Q}_k\mathbf{K}_k{}^\top + \sigma_k \odot \epsilon_j),\ \ \epsilon_j \sim \mathcal{N}(0, \mathbf{I}),\ \ \sum_{k=1}^{M} p_k = 1,\ \ p_k \geq 0. \tag{10}$$

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

$$\mathbf{A}_j = \sum_{k=1}^{M} \phi(p_{kj}(\mathbf{Q}_k\mathbf{K}_k{}^\top + \sigma_k \odot \epsilon_j)),\ \ \epsilon \sim \mathcal{N}(0, \mathbf{I}),$$

where ϕ is a nonlinear mapping and p_{kj} are relaxed to be learnable parameters. Similarly, we formulate local attention matrices $\mathbf{A}_j$ in the Transformer with a Generalized Hard Finite Admixture of Shared Heads (Hard GFiSHformer) as $\mathbf{A}_j = \sum_{k=1}^{M} \phi(p_{kj}\mathbf{Q}_k\mathbf{K}_k{}^\top)$.

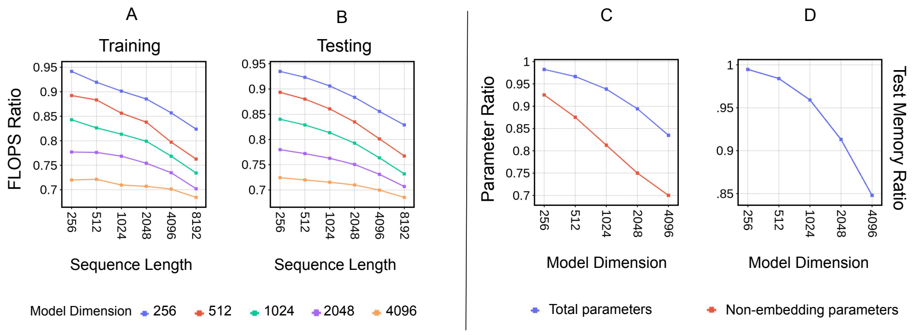

Figure 3: (Left) Training (A) and Inference (B) FLOP ratios between a 2-global-head GFiSHformers with 8-head MHA baselines across different model dimensions D and sequence lengths N trained on the WikiText-103 language modeling task. (Right) Number of parameters (C) and GPU memory usage at test time (D) ratios between 2-global-head GFiSHformers with 8-head MHA baselines across different model dimensions D. 2-global-head GFiSHformers are significantly more efficient than the baseline as D and N increase, indicating the benefits of our method for long-range and large-scale tasks.

Table 1: Perplexity (PPL) on WikiText-103 compared to the baselines.

Method	Valid PPL	Test PPL
Softmax 8 heads	33.15	34.29
Softmax 4 heads	34.80	35.85
Hard FiSHformer 4 global heads	33.10	34.11
Hard FiSHformer 2 global heads	34.14	35.24
FiSHformer 4 global heads	33.15	34.16
FiSHformer 2 global heads	34.01	34.96
Hard GFiSHformer 4 global heads	32.70	33.75
Hard GFiSHformer 2 global heads	33.31	34.63
GFiSHformer 4 global heads	**32.68**	**33.71**
GFiSHformer 2 global heads	33.21	34.48

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 $[2(H-M)D-2MH)]N^2+2(H-M)D(2D_x-1)N$ 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 ϕ 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.

Method	BLEU score
Softmax 4 heads	34.42
Hard Fishformer 2 global heads	34.31
FiSHformer 2 global heads	34.38
Hard GFiSHformer 2 global heads	34.54
GFiSHformer 2 global heads	**34.71**

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.

Method	BLEU score
Linear 4 heads	28.22
Hard LFiSHformer 2 global heads	**30.24**
LFiSHformer 2 global heads	29.63
Hard GLFiSHformer 2 global heads	29.20
GLFiSHformer 2 global heads	29.46

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 obtain comparable/better results than the baseline while being more efficient.

Method	BLEU score
Softmax 16 heads	29.38
Hard FiSHformer 8 global heads	29.32
FiSHformer 8 global heads	**29.57**
Hard GFiSHformer 8 global heads	29.27
GFiSHformer 8 global heads	29.42
Hard GFiSHformer 4 global heads	28.97
GFiSHformer 4 global heads	29.34

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.

Method	Acc top-1	Acc top-5	FLOPs (10^9)	Params
Softmax-12/24	**81.20**	**95.50**	4618.24	28.3M
Hard-GFiSH-6/12	81.11	95.44	4372.48	26.2M

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).

Dataset/Model	*Baseline softmax*	GFishformer
EthanolConcentration	32.08 (33.70)	**33.70**
FaceDetection	**68.70 (68.10)**	68.57
HandWriting	**32.08 (30.50)**	31.55
HeartBeat	75.77 (77.60)	**76.10**
JapaneseVowels	**99.46 (99.40)**	99.37
PEMS-SF	**82.66 (82.10)**	**82.66**
SelfRegulationSCP1	**91.46 (92.50)**	90.56
SelfRegulationSCP2	54.72 (53.90)	**54.81**
SpokenArabicDigits	99.33 (99.30)	**99.34**
UWaveGestureLibrary	84.45 (85.60)	**85.01**
Average Accuracy	72.07 (72.27)	**72.17**

Table 7: The GFiSHFormer vs. the baseline softmax transformer on 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.

Environment/Model	*Baseline softmax*	GFiSHFormer
Medium-Expert		
HalfCheetah	**91.03 (83.80)**	90.25
Hopper	110.30 (104.40)	**110.60**
Walker	**108.70 (107.70)**	108.30
Medium-Replay		
Hopper	85.61 (79.70)	**85.89**
Medium		
HalfCheetah	**42.28 (42.40)**	41.35
Hopper	61.47 (64.20)	**63.44**
Walker	**68.68 (70.60)**	67.07
Avg Reward	**81.19 (79.00)**	80.99

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] to evaluate the model performance on the offline reinforcement learning. On average, the 2-global-heads GFiSHformers perform comparably with the 4-head transformer baselines. For this benchmark, we follow the architecture and training configuration from [81].

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

To futher 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 $\mathcal{L}_2$ distances between heads of models trained for the WikiText-103 language modeling task.

Method	Mean	Variance
Softmax 8 heads	1.62	0.66
Linear 8 heads	1.90	0.06
GFiSHformer 2 global heads	2.93	2.62
GFiSHformer 4 global heads	**3.59**	**3.95**
GFiSHformer 6 global heads	3.37	2.78

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

Method	Mean
Softmax 8 heads	296
Linear 8 heads	436
GFiSHformer 2 global heads	895
GFiSHformer 4 global heads	**1408**
GFiSHformer 6 global heads	1228

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 Eqns. 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. Fig. 6, 7, 8 and details on our setting are provided in Appendix C. 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 $\mathcal{L}_2$ 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: $\mathcal{C}^l = \frac{1}{M \cdot H}\sum_{m=1}^{M}\sum_{h=1}^{H}(A_m^{l,h})(A_m^{l,h})^\top$. We use spectral decomposition to represent $\mathcal{C}^l$ in terms of eigenvalues and eigenvectors, namely, $\mathcal{C}^l = \sum_{i=1}^{n^2}\lambda_i v_i v_i^\top$. 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.

Method	Valid PPL	Test PPL
Softmax 8 heads	**33.15**	**34.29**
FiSHformer 2 global heads	34.01	34.96
MiSHformer 2 global heads	35.11	36.28

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

Method	Valid PPL	Test PPL
Softmax 8 heads	33.15	34.29
GFiSHformer 6 global heads	32.80	33.80
GFiSHformer 4 global heads	**32.68**	**33.71**
GFiSHformer 2 global heads	33.21	34.48

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, 62, 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 currenly 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

[1] Rami Al-Rfou, DK Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. In *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[2] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion 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] Akari Asai and Eunsol Choi. Challenges in information seeking qa: Unanswerable questions and paragraph retrieval. *arXiv preprint arXiv:2010.11915*, 2020.

[4] A. G. Bacharoglou. Approximation of probability distributions by convex mixtures of Gaussian measures. *Proceedings of the American Mathematical Society*, 138:2619–2628, 2010.

[5] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

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

[8] Srinadh Bhojanapalli, Ayan Chakrabarti, Himanshu Jain, Sanjiv Kumar, Michal Lukasik, and Andreas Veit. Eigen analysis of self-attention and its reconstruction from partial computation. *arXiv preprint arXiv:2106.08823*, 2021.

[9] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[10] Tom 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.

[11] Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*, volume 57, 2014.

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

[13] Sung Min Cho, Eunhyeok Park, and Sungjoo Yoo. Meantime: Mixture of attention mechanisms with multi-temporal embeddings for sequential recommendation. In *Fourteenth ACM Conference on Recommender Systems*, pages 515–520, 2020.

[14] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

[15] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher 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, August 2019. Association for Computational Linguistics.

[16] Gonçalo M Correia, Vlad Niculae, and André FT Martins. Adaptively sparse transformers. *arXiv preprint arXiv:1909.00015*, 2019.

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

[18] Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. Analyzing redundancy in pretrained transformer models. *arXiv preprint arXiv:2004.04010*, 2020.

[19] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

[20] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

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

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

[23] Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and Yonatan Belinkov. Analyzing individual neurons in pre-trained language models. *arXiv preprint arXiv:2010.02695*, 2020.

[24] R. Dwivedi, N. Ho, K. Khamaru, M. J. Wainwright, M. I. Jordan, and B. Yu. Sharp analysis of expectation-maximization for weakly identifiable models. *AISTATS*, 2020.

[25] R. Dwivedi, N. Ho, K. Khamaru, M. J. Wainwright, M. I. Jordan, and B. Yu. Singularity, misspecification, and the convergence rate of EM. *Annals of Statistics*, 44:2726–2755, 2020.

[26] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

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

[28] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021.

[29] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[30] S. Ghosal and A. van der Vaart. Entropies and rates of convergence for maximum likelihood and bayes estimation for mixtures of normal densities. *Ann. Statist.*, 29:1233–1263, 2001.

[31] Maosheng Guo, Yu Zhang, and Ting 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.

[32] John Hewitt and Percy 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, November 2019. Association for Computational Linguistics.

[33] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

[34] Nhat Ho, Tongzheng Ren, Sujay Sanghavi, Purnamrita Sarkar, and Rachel Ward. An exponentially increasing step-size for parameter estimation in statistical models. *arXiv preprint arXiv: 2205.07999*, 2022.

[35] Jeremy Howard and Sebastian 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.

[36] Junyan Jiang, Gus G Xia, Dave B Carlton, Chris N Anderson, and Ryan 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.

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

[38] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. Structured attention networks. *arXiv preprint arXiv:1702.00887*, 2017.

[39] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

[40] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[41] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.

[42] Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018.

[43] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[44] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, October 2021.

[45] Matouš Macháček and Ondřej Bojar. Results of the wmt14 metrics shared task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 293–301, 2014.

[46] Stephen Merity, Caiming Xiong, James Bradbury, and Richard 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.

[47] Paul Michel, Omer Levy, and Graham 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.

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

[49] Tam M Nguyen, Tan M Nguyen, Dung D Le, Duy-Khuong Nguyen, Viet-Anh Tran, Richard G Baraniuk, Nhat Ho, and Stanley J Osher. Improving transformers with probabilistic attention keys. In *International Conference on Machine Learning*, pages 16595–16621. PMLR, 2022.

[50] Tan M Nguyen, Richard Baraniuk, Robert Kirby, Stanley Osher, and Bao Wang. Momentum transformer: Closing the performance gap between self-attention and its linearization. In *Mathematical and Scientific Machine Learning*, pages 189–204. PMLR, 2022.

[51] Tan M Nguyen, Minh Pham, Tam Nguyen, Khai Nguyen, Stanley J Osher, and Nhat Ho. Fourierformer: Transformer meets generalized fourier integral theorem. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[52] Tan M Nguyen, Vai Suliafu, Stanley Osher, Long Chen, and Bao Wang. Fmmformer: Efficient and flexible transformer via decomposed near-field and far-field attention. In *Advances in neural information processing systems*, volume 34, pages 29449–29463, 2021.

[53] Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[54] Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. Bleu: a method for automatic evaluation of machine translation. pages 311–318, 2002.

[55] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In Jennifer Dy and Andreas 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.

[56] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations*, 2021.

[57] Jonathan K Pritchard, Matthew Stephens, and Peter Donnelly. Inference of population structure using multilocus genotype data. *Genetics*, 155(2):945–959, 2000.

[58] Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding. *arXiv preprint arXiv:1911.02972*, 2019.

[59] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI report*, 2018.

[60] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[61] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy 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, November 2016. Association for Computational Linguistics.

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

[63] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[64] Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. Poor man's bert: Smaller and faster transformer models. *arXiv e-prints*, pages arXiv–2004, 2020.

[65] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[66] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pages 9355–9366. PMLR, 2021.

[67] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3531–3539, 2021.

[68] David R So, Chen Liang, and Quoc V Le. The evolved transformer. *arXiv preprint arXiv:1901.11117*, 2019.

[69] Sainbayar Sukhbaatar, Edouard Grave, Guillaume Lample, Herve Jegou, and Armand Joulin. Augmenting self-attention with persistent memory. *arXiv preprint arXiv:1907.01470*, 2019.

[70] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*, 2019.

[71] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse Sinkhorn attention. In Hal Daumé III and Aarti 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.

[72] Ian Tenney, Dipanjan Das, and Ellie 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.

[73] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[74] Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. *arXiv preprint arXiv:1908.11775*, 2019.

[75] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[76] Jesse Vig and Yonatan 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, August 2019. Association for Computational Linguistics.

[77] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan 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.

[78] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan 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, 2019.

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

[80] Adina Williams, Nikita Nangia, and Samuel 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.

[81] Haixu Wu, Jialong Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Flowformer: Linearizing transformers with conservation flows. In *International Conference on Machine Learning*, 2022.

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

[83] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2114–2124, 2021.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

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.

[2] Implementation available at https://github.com/gzerveas/mvts_transformer.

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.

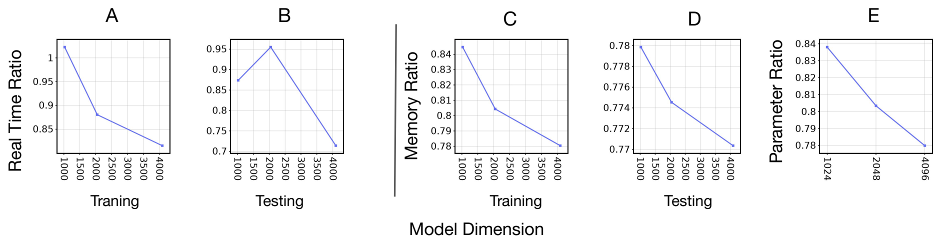

Figure 4: (Left) Training (A) and Inference (B) real time ratios between a 2-global-head GFiSHformers with 4-head MHA baselines across different model dimensions D trained on the IWSLT14 De-En machine translation task. (Right) GPU memory usage at train time (C) and test time (D) and number of parameters (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.

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 FiSHformer 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.

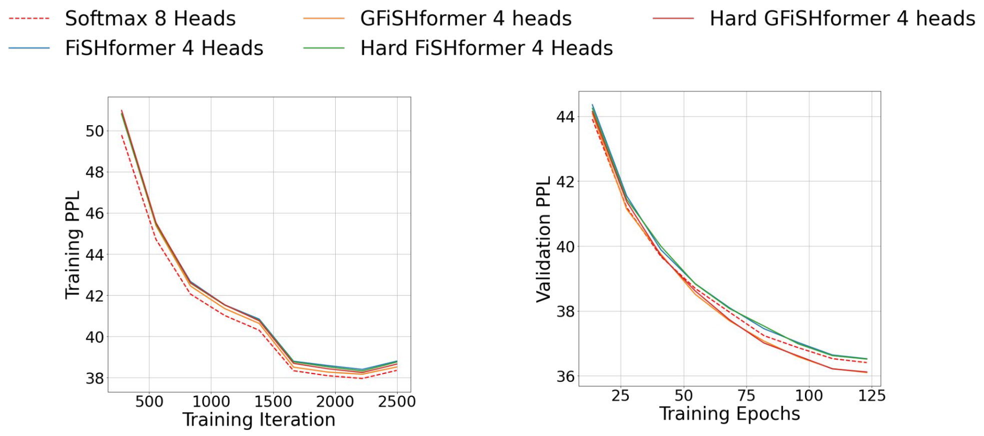

Figure 5: Train and validation PPL of 4-global-head FiSH-based models vs . 8-head MHA Transformer trained for the WikiText-103 language modeling task.

Table 12: Layer-Average mean and variance of $\mathcal{L}_2$ distances between heads of models trained for the WikiText-103 language modeling task.

Method	Mean	Variance
Softmax 8 heads	1.62	0.66
Hard FiSHformer 4 global heads	1.75	1.38
Hard GFiSHformer 2 global heads	2.99	2.79
Hard GFiSHformer 4 global heads	3.58	3.01
Hard GFiSHformer 6 global heads	2.90	1.71

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

Method	Mean
Softmax 8 heads	296
Hard FiSHformer 4 global heads	302
Hard GFiSHformer 2 global heads	1161
Hard GFiSHformer 4 global heads	1317
Hard GFiSHformer 6 global heads	1102

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.

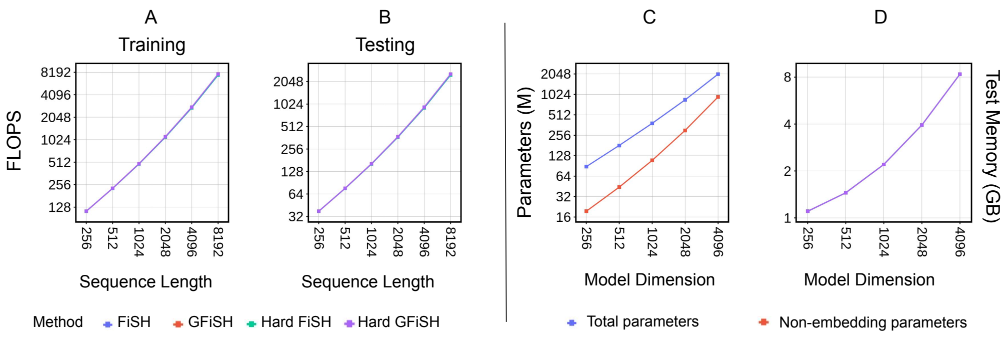

Figure 6: Training (Left) and Inference (Right) FLOPS (B) of 2-global-head FiSHformers, Hard FiSHformers, GFiSHformers, and Hard GFiSHformer. FiSH-based models have comparable computational costs trained on the WikiText-103 language modeling task. Here the model size is 1024.

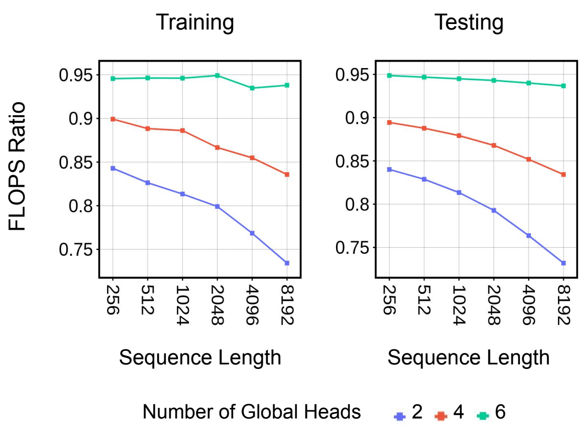

Figure 7: Training (Left) and Inference (Right) FLOP ratio between the GFiSHformers and the 8-head MHA baseline trained on the WikiText-103 language modeling task, as the number of heads vary in {2, 4, 6}. A decrease in the number of heads leads to a significant reduction in computational cost at each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

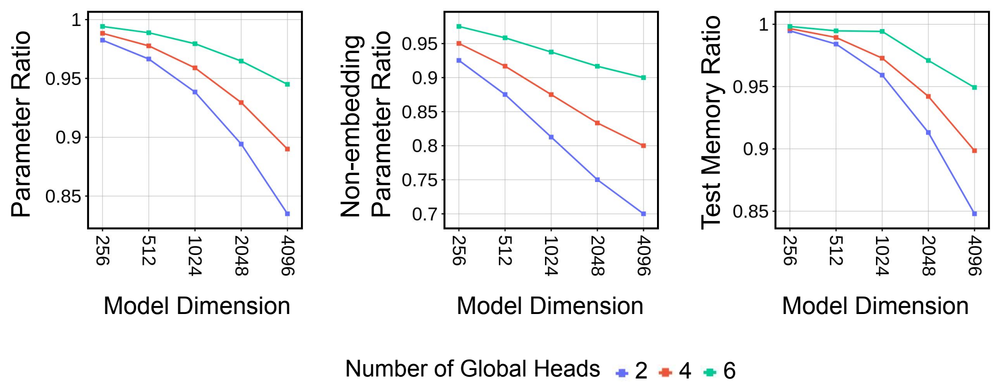

Figure 8: Number-of-parameter ratio (Left) and the memory ratio (at test time) (Right) between the GFiSH-formers and the 8-head MHA Transformer trained on the WikiText-103 language modeling task. Our method achieve significant reduction in memory cost and model complexity over the baseline as we scale 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 $N^2H(2D-1) + 2NHD(2D_x - 1)$.

Explanation: To calculate the query matrix $\mathbf{Q}$ and the key matrix $\mathbf{K}$ 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 $2ND(2D_x - 1)$ computations. Next, to compute the product $\mathbf{Q}\mathbf{K}^\top$ 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 $2ND(2D_x - 1) + N^2D + N^2(D-1) = N^2(2D-1) + 2ND(2D_x - 1)$ 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 $N^2M(2D-1) + 2NMD(2D_x - 1)$ 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 $2HDD_x$.

Explanation: $2HDD_x$ parameters is from the linear projects to calculate the query matrix $\mathbf{Q}$ and the key matrix $\mathbf{K}$ in Step 1 in Section 1.1.

(iv) 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 M query matrices $\mathbf{Q}$ and M key matrices $\mathbf{K}$, which are used to compute M global attention matrices. The extra HM parameters is from the linear mapping for computing H local attention matrices from M global attention matrices, and the extra M parameters are the M $\{\sigma_k\}_{k=1}^{M}$ for M global 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

$$p_G(x) := \int f(x-\theta)dG(\theta) = \int \phi(x|\theta, \sigma^2\mathbf{I})dG(\theta),$$

for all $x \in \mathbb{R}^d$ where $f(x) = \frac{1}{(\sqrt{2\pi}\sigma)^d}\exp\left(-\frac{\|x\|^2}{2\sigma^2}\right)$ 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

$$\sup_{x\in\mathbb{R}^d} |p(x) - p_{G_1}(x)| \leq \frac{\epsilon}{2}. \tag{11}$$

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

$$\sup_{x\in\mathbb{R}^d} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{\epsilon}{2}. \tag{12}$$

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

$$\begin{aligned}
\sup_{\|x\|>M} |p_{G_1}(x) - p_{G_2}(x)| &= \sup_{\|x\|>M} \left|\int f(x-\theta)d(G_1-G_2)(\theta)\right| \\
&\leq \sup_{\|x\|>M} \int \frac{1}{(\sqrt{2\pi}\sigma)^d}\exp\left(-\frac{\|x-\theta\|^2}{2\sigma^2}\right)d(G_1+G_2)(\theta) \\
&\leq \frac{2}{(\sqrt{2\pi}\sigma)^d}\exp\left(-\frac{M^2}{8\sigma^2}\right).
\end{aligned} \tag{14}$$

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

$$\begin{aligned}
\sup_{\|x\|\leq M} |p_{G_1}(x) - p_{G_2}(x)| &= \sup_{\|x\|\leq M} \left|\int f(x-\theta)d(G_1-G_2)(\theta)\right| \\
&\leq \sup_{\|x\|\leq M} \left|\int \left(f(x-\theta) - \sum_{j=0}^{k-1}\frac{(-1)^j\|x-\theta\|^{2j}}{(\sqrt{2\pi})^d\sigma^{d+2j}j!}\right)d(G_1-G_2)(\theta)\right|,
\end{aligned} \tag{15}$$

where the final inequality stems from

$$\int \sum_{j=0}^{k-1} \frac{(-1)^j \|x-\theta\|^{2j}}{(\sqrt{2\pi})^d \sigma^{d+2j} j!} d(G_1 - G_2)(\theta) = 0,$$

which is due to Eqn. (13).

To further bound the right-hand-side (RHS) of Eqn. (15), we use the following inequality:

$$\left| \exp(y) - \sum_{j=0}^{k-1} (y)^j / j! \right| \leq |y|^k / k!$$

for any $y \in \mathbb{R}$. Since $k! \geq (k/e)^k$ for any $k \geq 1$, the above bound can be rewritten as

$$\left| \exp(y) - \sum_{j=0}^{k-1} (y)^j / j! \right| \leq \frac{|ye|^k}{k^k}. \tag{16}$$

Further simplification of Eqn. (15) leads to

$$\begin{aligned} \sup_{\|x\| \leq M} |p_{G_1}(x) - p_{G_2}(x)| &\leq \sup_{\|x\| \leq M} \int \left| f(x-\theta) - \sum_{j=0}^{k-1} \frac{(-1)^j \|x-\theta\|^{2j}}{(\sqrt{2\pi})^d \sigma^{d+2j} j!} \right| d(G_1 + G_2)(\theta) \\ &\leq 2 \sup_{\|x\| \leq M, \theta \in [-a,a]^d} \left| f(x-\theta) - \sum_{j=0}^{k-1} \frac{(-1)^j \|x-\theta\|^{2j}}{(\sqrt{2\pi})^d \sigma^{d+2j} j!} \right| \\ &\leq \sup_{\|x\| \leq M, \theta \in [-a,a]^d} \frac{e^k \|x-\theta\|^{2k}}{\sigma^{2k} (2k)^k}, \end{aligned}$$

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

$$\sup_{\|x\| \leq M} |p_{G_1}(x) - p_{G_2}(x)| \leq \sup_{\|x\| \leq M, \theta \in [-a,a]^d} \frac{e^k \|x-\theta\|^{2k}}{\sigma^{2k} (2k)^k} \leq \frac{e^k (M + a\sqrt{d})^{2k}}{\sigma^{2k} (2k)^k}.$$

When $M \geq 2a\sqrt{d}$, we have $M + a\sqrt{d} \leq \frac{3M}{2}$ and the above bound leads to

$$\sup_{\|x\| \leq M} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{(9e)^k M^{2k}}{(8\sigma^2 k)^k}. \tag{17}$$

By choosing $M^2 = 8\sigma^2 \log(1/\epsilon')$ for some $\epsilon' > 0$, the bounds in Eqns. (14) and (17) become

$$\sup_{\|x\| \leq M} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{2}{(\sqrt{2\pi}\sigma)^d} \epsilon',$$

$$\sup_{\|x\| > M} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{(9e)^k (\log(1/\epsilon'))^k}{k^k}. \tag{18}$$

As long as we choose $k = 9e^2 \log(1/\epsilon')$ and $\epsilon' \leq 1$, we have

$$\sup_{\|x\| > M} |p_{G_1}(x) - p_{G_2}(x)| \leq e^{-k} = e^{-9e^2 \log(1/\epsilon')} = (\epsilon')^{9e^2} \leq \epsilon'. \tag{19}$$

By choosing $\epsilon' = \frac{\epsilon}{2 \max\{\frac{2}{(\sqrt{2\pi}\sigma)^d}, 1\}}$, the results from Eqns. (18) and (19) indicate that

$$\sup_{\|x\| \leq M} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{\epsilon}{2}, \quad \text{and} \quad \sup_{\|x\| > M} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{\epsilon}{2}.$$

Therefore, if we choose $M = 8\sigma^2 \log\left(\frac{2\max\{\frac{2}{(\sqrt{2\pi}\sigma)^d}, 1\}}{\epsilon}\right)$ and $k = 9e^2 \log\left(\frac{2\max\{\frac{2}{(\sqrt{2\pi}\sigma)^d}, 1\}}{\epsilon}\right)$, we have

$$\sup_{x \in \mathbb{R}^d} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{\epsilon}{2}.$$

It indicates that we obtain the conclusion of claim (12) by choosing $K = (2k-2)^d \leq \left(18e^2 \log\left(\frac{2\max\{\frac{2}{(\sqrt{2\pi}\sigma)^d}, 1\}}{\epsilon}\right)\right)^d$. As a consequence, we obtain the conclusion of the lemma.