FiSHFormer: Transformer with a Finite Admixture of Shared Heads

Anonymous Authors[1]

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 that they generate. 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 on 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.

1 Introduction

Transformers have become the state-of-the-art model for solving many challenging problems in natural language processing (Vaswani et al., 2017; Al-Rfou et al., 2019a; Dai et al., 2019; Williams et al., 2018; Devlin et al., 2018; Brown & et al., 2020; Howard & Ruder, 2018; Rajpurkar

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

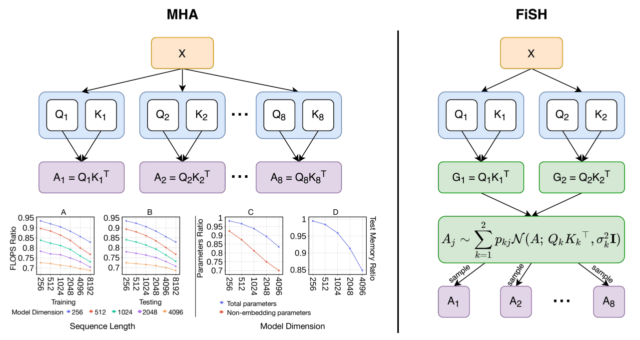

Figure 1. Our proposed finite admixture model of shared heads (FiSH) vs. the standard multi-head (MHA) attention. FiSH samples local attention matrices from a finite admixture of global attention matrices. Compared to MHA, FiSH is more efficient, saving computation and memory (See Fig. 3 and Section 5.1).

et al., 2016) and computer vision (Dehghani et al., 2018; So et al., 2019; Dosovitskiy et al., 2020; Touvron et al., 2020). 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 (Radford et al., 2018; 2019; Devlin et al., 2018; Yang et al., 2019; Liu et al., 2019). 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 (Bahdanau et al., 2014; Vaswani et al., 2017; Kim et al., 2017). 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 (Tenney et al., 2019) and its flexibility in capturing diverse syntactic and semantic relationships (Tenney et al., 2019; Vig & Belinkov, 2019; Clark et al., 2019; Voita et al., 2019a; Hewitt & Liang, 2019) 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 $\mathbf{X}$

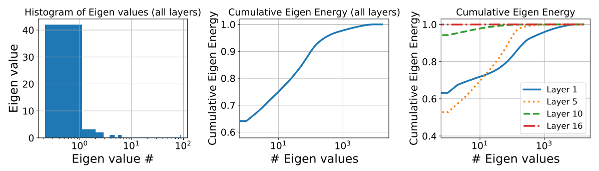

Figure 2. Left: Histogram of the top 50 eigen values of the layer-average attention scores covariance matrix. Middle: Cumulative sum of eigen values of the layer-average attention scores covariance matrix. Right: the layer-average attention scores covariance matrix of different layers.

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

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

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

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

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

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

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 (Michel et al., 2019; Voita et al., 2019b), whose learned attention matrices lie on a low-dimensional manifold (Bhojanapalli et al., 2021). To confirm this claim, in Figure 2, we follow the eigenvalue analysis in (Bhojanapalli et al., 2021) 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) (Pritchard et al., 2000; Blei et al., 2003), 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 3, we analyze the reduction in model complexity and computational cost from FiSH. In Section 4 and 5, 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 6. 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) and discuss its extensions.

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

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

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 $\boldsymbol{x}_j$, $j = 1, 2, \ldots, H$, are drawn from different mixture distributions that share the same M components $f(\boldsymbol{x}; \theta_k)$, $k = 1, 2, \ldots, M$ but with different mixture weights

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

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

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

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 $\{\mathbf{G}_k = \mathbf{Q}_k {\mathbf{K}_k}^\top\}_{k=1,\ldots,M}$ global attention matrices and $\{\mathbf{A}_j\}_{j=1,\ldots,H}$ 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, and thus, FiSH is much more efficient to compute 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:*

$$p(w|d) = \sum_c p(c|d)p(w|c). \tag{8}$$

Comparing Eqn. 7 of FiSH and Eqn. 8 of pLSA, 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: Each Head has a Hierarchical Structure

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 (Kingma & Welling, 2014), the attention matrices $\mathbf{A}_j$ can be written in a differentiable form as follows:

$$\mathbf{A}_j = \sum_{k=1}^{M} p_{kj}(\mathbf{Q}_k\mathbf{K}_k{}^\top + \sigma_k \odot \epsilon_j), \;\; \epsilon_j \sim \mathcal{N}(0, \mathbf{I}),$$
$$\sum_{k=1}^{M} p_{kj} = 1, \;\; p_{kj} \geq 0. \tag{9}$$

FiSHformers use the formulation of local attention heads in Eqn. 9 to implement FiSH.

Transformer with a Hard Finite Admixture of Shared Heads (Hard FiSHformer) Hard FiSHformer takes the zero-noise limit of Eqn. 9 to reduce the computational cost. The attention matrices $\mathbf{A}_j$ in Hard FiSHformer are then calculated as

$$\mathbf{A}_j = \sum_{k=1}^{M} p_{kj}\mathbf{Q}_k\mathbf{K}_k{}^\top, \;\; \sum_{k=1}^{M} p_{kj} = 1, \;\; p_{kj} \geq 0. \tag{10}$$

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*

$$\mathbf{A}_j = \sum_{k=1}^{M} p_k(\mathbf{Q}_k\mathbf{K}_k{}^\top + \sigma_k \odot \epsilon_j), \;\; \epsilon_j \sim \mathcal{N}(0, \mathbf{I}),$$
$$\sum_{k=1}^{M} p_k = 1, \;\; p_k \geq 0. \tag{11}$$

An empirical comparison between FiSHformer and MiSHformer is provided in Section 5.4.

2.4 Transformer with a Generalized Finite Admixture of Shared Heads: Each Head has a Nonlinear Hierarchical Structure

In order to increase the representation capacity of attention heads, we follow a common approach in learning representation by replacing the linear mapping in Eqn. 9 and 10 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 follows:

$$\mathbf{A}_j = \sum_{k=1}^{M} \phi(p_{kj}\mathbf{Q}_k\mathbf{K}_k{}^\top). \tag{12}$$

3 Reduction in Model Complexity and Computational Cost from FiSH

In this section, we derive the reduction in computation cost, measured in terms of FLOPs, and model complexity, measured in terms of the number of parameters, achieved by FiSH when compared to its baseline MHA attention. Detailed derivations are provided in Appendix D.

Reduction in Computational Cost Let D be the model dimension, D_x is the input dimension, and N is the length of the input sequence. 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.

Table 1. Perplexity (PPL) on WikiText-103 of (G)FiSHformer 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

Reduction in Model Complexity Let D be the model dimension, D_x is the input dimension, and N is the length of the input sequence. Compared to its H-head MHA counterpart, FiSH attention of M global heads and H local heads saves $2(H-M)DD_x - HM - M$ parameters.

4 Experimental Results

In this section, we empirically study the advantages of FiSHformer on various tasks and benchmarks, including language modeling on WikiText-103 (Section 4.1), machine translation on IWSLT' 14 De-En and WMT'14 (Section 4.2), and image classification on ImageNet (Section 4.3). 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 on state-of-the-art transformer models to improve their performance on large-scale applications. In Section 5, 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.

4.1 WikiText-103 Language Modeling

Models and baselines We compare the 2 and 4-global-head FiSHformers with the 8-head softmax transformers (Vaswani et al., 2017). Each model has 16 layers, and our training follows the setting from (Schlag et al., 2021).

Results Perplexity: Table 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 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 5.1 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.

4.2 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 (Cettolo et al., 2014) and then scale up our experiments to the WMT'14 En-De (Macháček & Bojar, 2014). 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 (Ott et al., 2018).

Results As shown in Table 2 and 3, (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

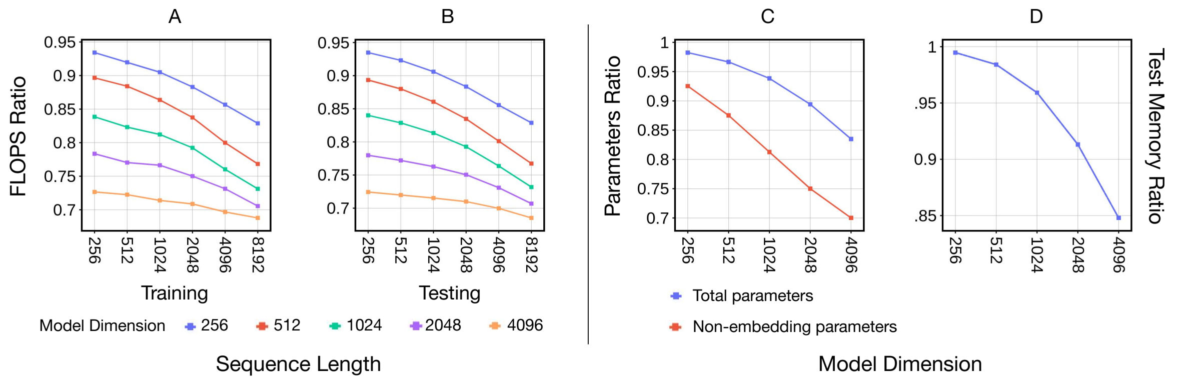

Figure 3. (Left) Training (A) and Inference (B) FLOP ratios between a 2-global-head GFiSHformers with 8-head MHA baselines across different model dimensions D and sequence lengths N trained on the WikiText-103 language modeling task. (Right) Number of parameters (C) and GPU memory usage at test time (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 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. BLEU scores of (G)FiSHformers, with a various number of shared heads compared to the 16-head baseline on the WMT'14 En-De machine translation. Our methods obtain comparable or 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

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.

4.3 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 (Liu et al., 2021), a state-of-the-art vision transformer architecture, for the image classification task on the ImageNet dataset (Deng et al., 2009). 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 model and training follow the settings in (Liu et al., 2021). We summarize our results in Table 4. Our GFiSHformer is only 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 3, these advantages grow with larger D and N.

Table 4. 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 (Liu et al., 2021) are provided in parentheses.

Method	Acc top-1	Acc top-5	FLOPs	Params
Softmax-12/24 (baseline)	**80.3** (81.2)	**95.05** (95.5)	4.51B	28.3M
Hard-GFiSH-6/12	80.14	94.92	4.27B	26.2M

4.4 Beyond Multi-Head Softmax Transformers

In this section, we show that (G)FiSH can be applied on top of a variety of transformer architectures to improve their performance including the linear transformers (Katharopoulos et al., 2020) and the state-of-the-art transformer with noisy back-translation (Edunov et al., 2018).

Applying (G)FiSH on Linear Transformers Linear transformers (Katharopoulos et al., 2020) is a class of efficient transformers that linearize the softmax kernel in Eqn. 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 5. 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.

Table 5. 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

Table 6. Machine translation BLEU scores of Hard GFiSHformer with 8 global heads compared to the SoTA 16-head Noisy Back-Translation on the WMT'14 De-En benchmark. Our method improves the efficiency of the SoTA baseline while obtaining comparable performance.

Method	BLEU score
SoTA Softmax 16 heads	**33.52**
Hard GFiSHformer 8 global heads	33.45

(G)FiSH Improves the State-of-the-Art Noisy Back-Translation We apply Hard GFiSH on the transformers trained with noisy back-translation for the WMT'14 En-De translation task. Table 6 shows that the transformers trained with noisy back-translation equipped with Hard GFiSH obtain comparable accuracy.

5 Empirical Analysis

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

5.1 Efficiency Analysis

We have already shown that on the WikiText-103 language modeling task, the efficiency advantage of 2-global-head GFiSHformers over the 8-head baseline grows with D and N in Fig. 3 and Section 4.1. 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 analysis 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.

5.2 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 on the WikiText-103 language modeling task, we obtain attention matrices of each head in each attention layer. We then compute the pair-wise $\mathcal{L}_2$ distances between heads in the same layer. More precisely, we consider a population of M sequences $\{X\}_{m=1}^{M}$ of length N with a pre-trained transformer of L layers and H heads. Let $l \in [[L]]$ denotes the layer-index and $h \in [[H]]$ denotes the head-index, we obtain the corresponding vectorized attention scores $A_m^{l,h} \in \mathbb{R}^{N^2}$. For each layer l, we compute the average distance between each pair of the head $h, h' \in [[H]]$:

$$d^l = \frac{2}{MH(H-1)} \sum_{m=1}^{M} \sum_{h=1}^{H} \sum_{h'=h+1}^{H} ||A_m^{l,h} - A_m^{l,h'}||_2^2$$

Table 7. Laver-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
GFiSHformer 2 global heads	2.93	2.62
GFiSHformer 4 global heads	**3.59**	**3.95**
GFiSHformer 6 global heads	3.37	2.78

Here, we do the analysis on models trained for the WikiText-103 language modeling task. These models have 16 layers of 8 local heads attention module, $n = 128$, and $M = 10^5$. We show the layer-average mean and variance of distances between heads in GFiSHformers compared with those in the MHA softmax baselines in Table 7. We provide additional results for Hard GFiSHformers and Hard FiSHformer in Table 11 in the Appendix.

5.3 Eigen Analysis: FiSHformer vs. GFiSHformer

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 on the WikiText-103 language modeling task, we first compute the covariance matrix of the vectorized attention scores of the l-th layer (see Section 5.2):

$$\mathcal{C}^l = \frac{1}{M \cdot H} \sum_{m=1}^{M} \sum_{h=1}^{H} (A_m^{l,h})(A_m^{l,h})^\top \tag{13}$$

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 8. Interestingly, Table 8 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. Additional results for Hard GFiSHformer and Hard FiSHformer are provided in Table 12 in the Appendix.

Table 8. Layer-average number of principle components for 95% variance explained of the covariance of attention matrices in models trained for the WikiText-103 language modeling task.

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

Table 9. Perplexity (PPL) on WikiText-103 of 2-global-head FiSHformer vs. 2-global-head MiSHformer in comparison with the 8-head baseline. MiSHformer attains worse PPL than FiSHformer but can potentially be more efficient.

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 10. 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

5.4 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 9.

5.5 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 10. Ablation results on efficiency are summarized in Fig. 7, 8 in Appendix C, and ablation results on representation capacity are provided in Table 7, 8.

6 Related Work

Efficient Transformers Efficient transformers have been developed to reduce the quadratic computational and memory cost of transformers (Roy et al., 2021). A class of efficient transformers are sparse transformers which design the attention matrix to have sparse structure (Parmar et al., 2018; Liu et al., 2018; Qiu et al., 2019; Child et al., 2019; Beltagy et al., 2020). Another class of efficient transformers are models that integrate different access patterns for better coverage (Child et al., 2019; Ho et al., 2019). These access patterns can also be learned from the data (Kitaev et al., 2020; Roy et al., 2021; Tay et al., 2020). In other works, a side memory module is used to access multiple tokens simultaneously (Lee et al., 2019; Sukhbaatar et al., 2019; Asai & Choi, 2020; Beltagy et al., 2020). Recently, low-rank and kernelization methods have been proposed to improve the computational and memory efficiency of computing self-attention (Tsai et al., 2019; Wang et al., 2020; Katharopoulos et al., 2020; Choromanski et al., 2021; Shen et al., 2021; Nguyen et al., 2021b; Peng et al., 2021). Multi-query attention that shares keys and values between different attention heads (Shazeer, 2019) has also been studied to reduce the memory-bandwidth cost and increase the speed for incremental transformer inference. Finally, methods that use auxiliary losses (Al-Rfou et al., 2019b) and adaptive input embedding (Baevski & Auli, 2019) have been studied to accelerate the training transformers. Our (G)FiSHformers are orthogonal and can be easily combined with these methods.

Redundancy in Transformers It has been shown that most of the neurons and heads in the pre-trained transformer are redundant and can be pruned when applied on a downstream task (Dalvi et al., 2020; Michel et al., 2019; Durrani et al., 2020). The contextualized embeddings in pre-trained networks under this redundancy due to overparameterization have also been studied to demonstrate that the representations learned within these models are highly anisotropic (Mu & Viswanath, 2018; Ethayarajh, 2019). Knowledge distillation and sparse approximation have also been used to enhance the efficiency of transformers, including (Sanh et al., 2019; Sun et al., 2019; Voita et al., 2019b; Sajjad et al., 2020). Our FiSH-based approach can be used along these methods to improve their performance.

Mixture Models for Transformers Recently, mixture models have been employed to study and enhance transformers. Among these works is switch transformers (Fedus et al., 2021) that take advantage of the routing algorithm in Mixture of Experts (MoE) to decrease the communication and computational costs in transformers. (Nguyen et al., 2021a) derives a connection between self-attention and Gaussian mixture models and proposes to use a mixture of attention keys to improve the efficiency of transformers. Other works that combine mixture models with transformers include (Cho et al., 2020; Guo et al., 2019; Jiang et al., 2020).

7 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 interesting to impose additional structures such as low-rank and sparsity into the global attention matrices. Also, neural architecture search (Zoph & Le, 2017) can be used to select structures for the global attention matrices.

References

Al-Rfou, R., Choe, D., Constant, N., Guo, M., and Jones, L. Character-level language modeling with deeper self-attention. In *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019a. URL https://arxiv.org/abs/1808.04444.

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

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

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

Baevski, A. and Auli, M. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByxZX20qFQ.

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

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

Bhojanapalli, S., Chakrabarti, A., Jain, H., Kumar, S., Lukasik, M., and Veit, A. Eigen analysis of self-attention and its reconstruction from partial computation. *arXiv preprint arXiv:2106.08823*, 2021.

Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *the Journal of machine Learning research*, 3: 993–1022, 2003.

Brown, T. and et al. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Cettolo, M., Niehues, J., Stüker, S., Bentivogli, L., and Federico, M. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*, volume 57, 2014.

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

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

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

Clark, K., Khandelwal, U., Levy, O., and Manning, C. D. 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*, pp. 276–286, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4828. URL https://www.aclweb.org/anthology/W19-4828.

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

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

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

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

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

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

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

Edunov, S., Ott, M., Auli, M., and Grangier, D. Understanding back-translation at scale. In *Proceedings of* *the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 489–500, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1045. URL https://aclanthology.org/D18-1045.

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

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

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

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

Hewitt, J. and Liang, P. 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)*, pp. 2733–2743, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1275. URL https://www.aclweb.org/anthology/D19-1275.

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

Howard, J. and Ruder, S. 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)*, pp. 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031. URL https://www.aclweb.org/anthology/P18-1031.

Jiang, J., Xia, G. G., Carlton, D. B., Anderson, C. N., and Miyakawa, R. H. 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)*, pp. 516–520. IEEE, 2020.

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

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

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

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

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

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

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

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

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10012–10022, October 2021.

Macháček, M. and Bojar, O. Results of the wmt14 metrics shared task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pp. 293–301, 2014.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. 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. URL https://openreview.net/forum?id=Byj72udxe.

Michel, P., Levy, O., and Neubig, G. Are sixteen heads really better than one? In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/2c601ad9d2ff9bc8b282670cdd54f69f-Paper.pdf.

Mu, J. and Viswanath, P. All-but-the-top: Simple and effective postprocessing for word representations. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HkuGJ3kCb.

Nguyen, T., Nguyen, T. M., Le, D., Nguyen, K., Tran, A., Baraniuk, R. G., Ho, N., and Osher, S. J. Transformer with a mixture of gaussian keys. *arXiv preprint arXiv:2110.08678*, 2021a.

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

Ott, M., Edunov, S., Grangier, D., and Auli, M. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 1–9, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6301. URL https://aclanthology.org/W18-6301.

Papineni, K., Roukos, S., Ward, T., and jing Zhu, W. Bleu: a method for automatic evaluation of machine translation. pp. 311–318, 2002.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D. Image transformer. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4055–4064. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/parmar18a.html.

Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N., and Kong, L. Random feature attention. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=QtTKTdVrFBB.

Pritchard, J. K., Stephens, M., and Donnelly, P. Inference of population structure using multilocus genotype data. *Genetics*, 155(2):945–959, 2000.

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

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

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

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://www.aclweb.org/anthology/D16-1264.

Roy, A., Saffar, M., Vaswani, A., and Grangier, D. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021. doi: 10.1162/tacl_a_00353. URL https://www.aclweb.org/anthology/2021.tacl-1.4.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015.

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

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

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

Shazeer, N. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

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

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

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

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

Tay, Y., Bahri, D., Yang, L., Metzler, D., and Juan, D.-C. Sparse Sinkhorn attention. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9438–9447. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/tay20a.html.

Tenney, I., Das, D., and Pavlick, E. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4593–4601, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1452. URL https://www.aclweb.org/anthology/P19-1452.

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

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

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

Vig, J. and Belinkov, Y. 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*, pp. 63–76, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4808. URL https://www.aclweb.org/anthology/W19-4808.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. 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*, pp. 5797–5808, Florence, Italy, July 2019a. Association for Computational Linguistics. doi: 10.18653/v1/P19-1580. URL https://www.aclweb.org/anthology/P19-1580.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. 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*, pp. 5797–5808, 2019b.

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

Williams, A., Nangia, N., and Bowman, S. 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)*, pp. 1112–1122, June 2018. doi: 10.18653/v1/N18-1101. URL https://www.aclweb.org/anthology/N18-1101.

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

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.

Supplementary materials for

FiSHFormer: Transformer with a Finite Admixture of Shared Heads

A Experiment Details

In this section, we provide model and training details for experiments in Section 4. 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 (Merity et al., 2017; Schlag et al., 2021) 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 (Al-Rfou et al., 2019b; Schlag et al., 2021).

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 (Papineni et al., 2002) is used to measure the performance of the trained model.

A.3 Image Classification

Datasets and metrics The ImageNet dataset (Russakovsky et al., 2015) 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 (Liu et al., 2021) 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.

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.

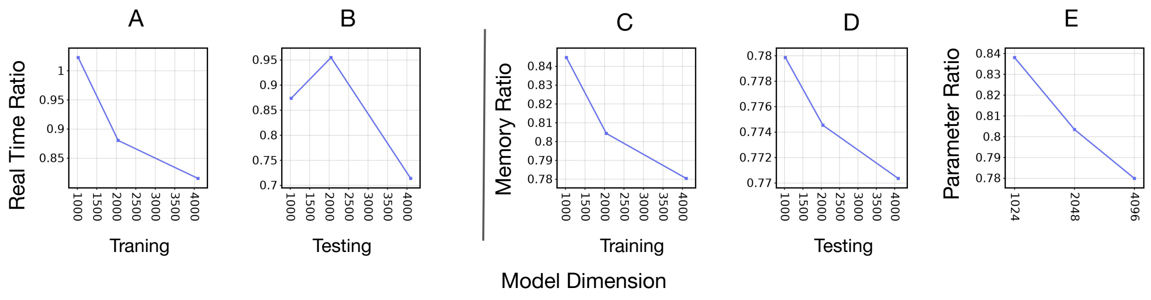

Figure 4. (Left) Training (A) and Inference (B) real time ratios between a 2-global-head GFiSHformers with 4-head MHA baselines across different model dimensions D trained on the IWSLT14 De-En machine translation task. (Right) GPU memory usage at train time (C) and test time (D) and number of parameters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

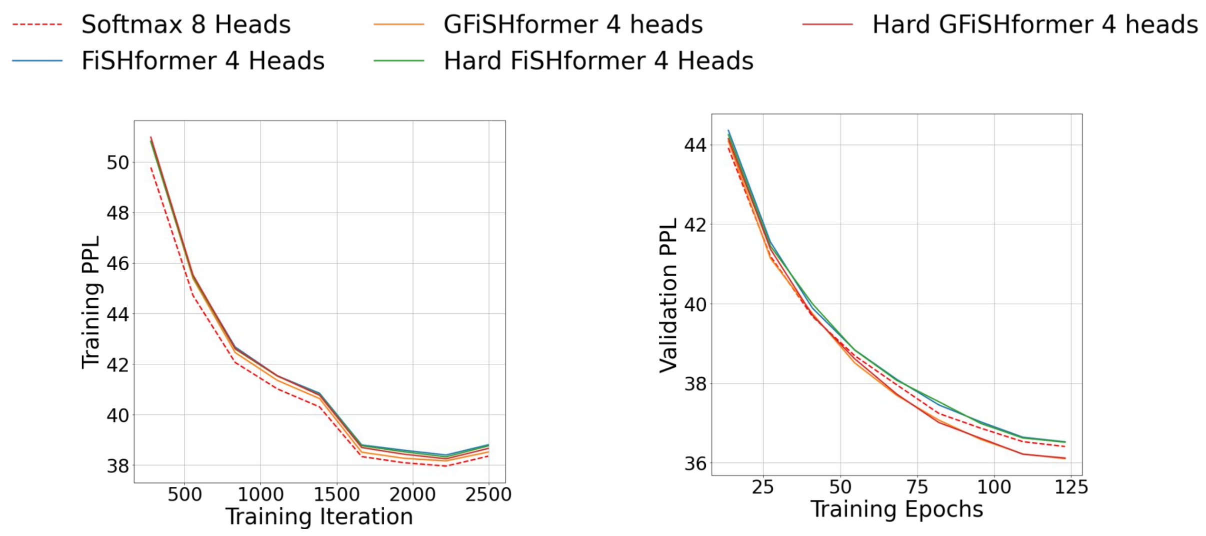

Figure 5. Train and validation PPL of 4-global-head FiSH-based models vs . 8-head MHA Transformer trained for the WikiText-103 language modeling task.

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 11 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.

Table 11. Laver-Average mean and variance of $\mathcal{L}_2$ distances between heads

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

B.4 More Results on Eigen Analysis

Table 12 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

We present the efficiency improvement of (G)FiSHformers over the MHA baseline. 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

Table 12. Layer-average number of principle components for 95% variance explained.

Method	Mean
Softmax 8 heads	296
Hard FiSHformer 4 global heads	302.125
Hard GFiSHformer 2 global heads	1161
Hard GFiSHformer 4 global heads	1317
Hard GFiSHformer 6 global heads	1102

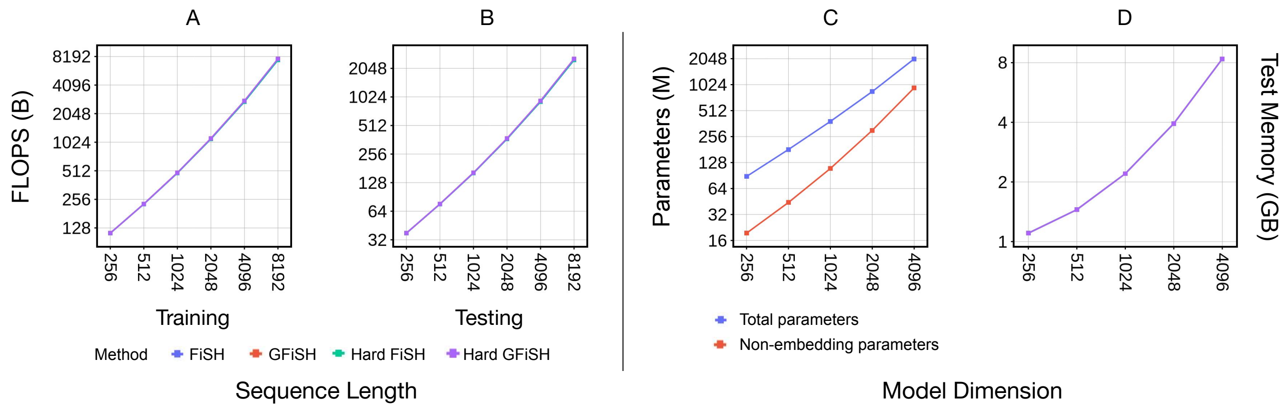

Figure 6. Training (Left) and Inference (Right) FLOPS (B) of 2-global-head FiSHformers, Hard FiSHformers, GFiSHformers, and Hard GFiSHformer. FiSH-based models have comparable computational costs. Here the model size is 1024.

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.

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 2-global-head GFISHformers compared to the 8-head MHA Transformer. Figure 8 shows the number-of-parameter-ratio (Left) and the memory-ratio (Right) of 2-global-head GFiSHformers and the 8-head baseline. At a fixed sequence length N, 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.

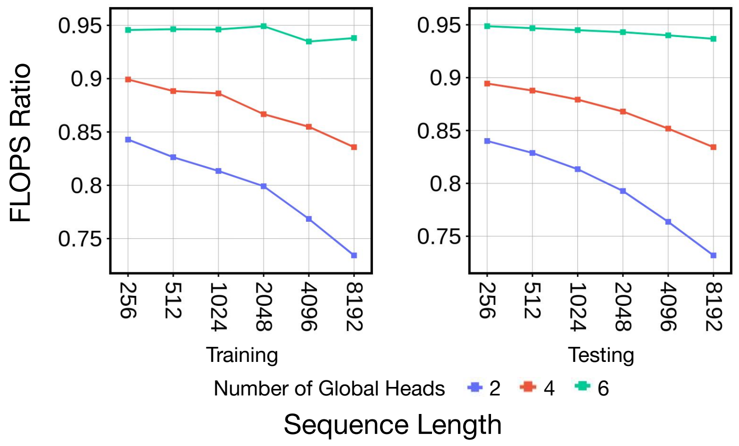

Figure 7. Training (Left) and Inference (Right) FLOP ratio between GFiSHformers and 8-head MHA baseline as the number of heads vary in $\{2, 4, 6\}$ A decrease in the number of heads leads to a significant reduction in computational cost at each sequence length. Here, the model dimension is 1024.

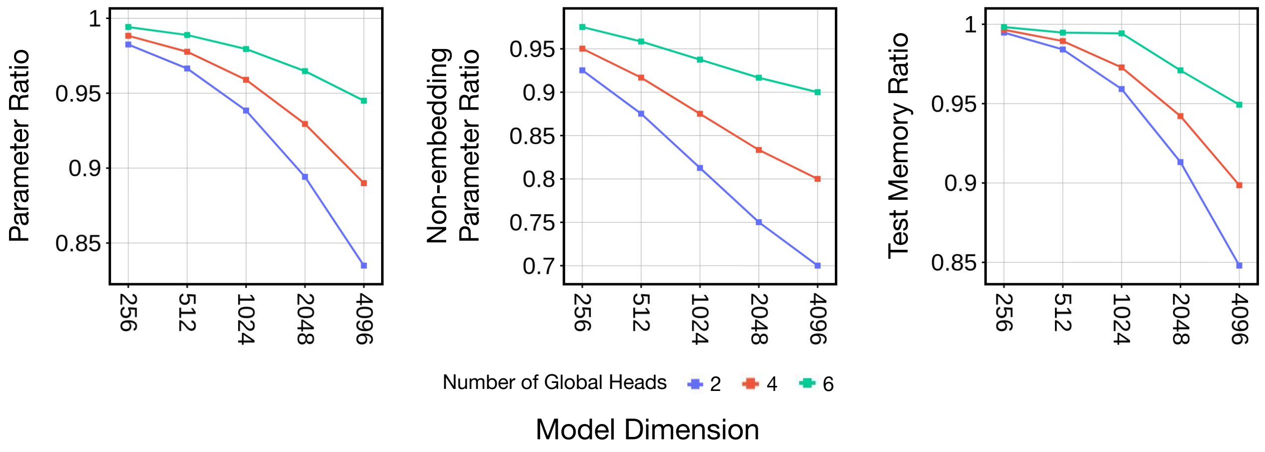

Figure 8. Number-of-parameter ratio (Left) and the memory ratio (at test time) (Right) between the 2-global-head GFiSHformers and the 8-head MHA Transformer. Our method achieve significant reduction in memory cost and model complexity over the baseline as we scale up the model dimension, $D \in \{256, 512, 1024, 2048 \text{ to } 4096\}$. Here, the sequence length is 1024.

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 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 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 $\mathbf{Q}$, the key matrix $\mathbf{K}$, and the value matrix $\mathbf{V}$ 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 $\mathbf{QK}^\top$ 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 $2ND(2D_x-1)+N^2D+N^2(D-1)=N^2(2D-1)+2ND(2D_x-1)$ computations. The total computation for all 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 Bacharoglou (2010), the space of Gaussian mixtures is dense in the space of continuous probability measures. Therefore, we can find probability distribution G_1 such that

$$\sup_{x\in\mathbb{R}^d}|p(x)-p_{G_1}(x)|\leq\frac{\epsilon}{2}. \tag{14}$$

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

$$\sup_{x \in \mathbb{R}^d} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{\epsilon}{2}. \tag{15}$$

Our technique for proving the above approximation bound relies on Lemma A.1 in (Ghosal & van der Vaart, 2001). 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

$$\int \theta^\alpha d(G_1 - G_2)(\theta) = 0, \tag{16}$$

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

$$\begin{aligned} \sup_{\|x\| > M} |p_{G_1}(x) - p_{G_2}(x)| &= \sup_{\|x\| > M} \left| \int f(x-\theta) d(G_1 - G_2)(\theta) \right| \\ &\leq \sup_{\|x\| > M} \int \frac{1}{(\sqrt{2\pi}\sigma)^d} \exp\left(-\frac{\|x-\theta\|^2}{2\sigma^2}\right) d(G_1 + G_2)(\theta) \\ &\leq \frac{2}{(\sqrt{2\pi}\sigma)^d} \exp\left(-\frac{M^2}{8\sigma^2}\right). \end{aligned} \tag{17}$$

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

$$\begin{aligned} \sup_{\|x\| \leq M} |p_{G_1}(x) - p_{G_2}(x)| &= \sup_{\|x\| \leq M} \left| \int f(x-\theta) d(G_1 - G_2)(\theta) \right| \\ &\leq \sup_{\|x\| \leq M} \left| \int \left(f(x-\theta) - \sum_{j=0}^{k-1} \frac{(-1)^j \|x-\theta\|^{2j}}{(\sqrt{2\pi})^d \sigma^{d+2j} j!} \right) d(G_1 - G_2)(\theta) \right|, \end{aligned} \tag{18}$$

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. (16).

To further bound the right-hand-side (RHS) of Eqn. (18), 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

$$\left| \exp(y) - \sum_{j=0}^{k-1} (y)^j / j! \right| \leq \frac{|ye|^k}{k^k}. \tag{19}$$

Further simplification of Eqn. (18) 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 (19) 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

$$\sup_{\|x\|\leq M} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{(9e)^k M^{2k}}{(8\sigma^2 k)^k}. \tag{20}$$

By choosing $M^2 = 8\sigma^2 \log(1/\epsilon')$ for some $\epsilon' > 0$, the bounds in Eqns. (17) and (20) become

$$\sup_{\|x\|\leq M} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{2}{(\sqrt{2\pi}\sigma)^d}\epsilon',$$
$$\sup_{\|x\|> M} |p_{G_1}(x) - p_{G_2}(x)| \leq \frac{(9e)^k (\log(1/\epsilon'))^k}{k^k}. \tag{21}$$

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

$$\sup_{\|x\|> M} |p_{G_1}(x) - p_{G_2}(x)| \leq e^{-k} = e^{-9e^2\log(1/\epsilon')} = (\epsilon')^{9e^2} \leq \epsilon'. \tag{22}$$

By choosing $\epsilon' = \frac{\epsilon}{2\max\{\frac{2}{(\sqrt{2\pi}\sigma)^d}, 1\}}$, the results from Eqns. (21) and (22) 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 (15) 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.