A Probabilistic Framework for Pruning Transformers via a Finite Admixture of Keys

Anonymous ECCV submission

Paper ID 6563

Abstract. Pairwise dot product-based self-attention is key to the success of transformers which achieve state-of-the-art performance across a variety of applications in language and vision, but are costly to compute. However, it has been shown that most attention scores and keys in transformers are redundant and can be removed without loss of accuracy. In this paper, we develop a novel probabilistic framework for pruning attention scores and keys in transformers. We first formulate an admixture model of attention keys whose input data to be clustered are attention queries. We show that attention scores in self-attention correspond to the posterior distribution of this model when attention keys admit a uniform prior distribution. We then relax this uniform prior constraint and let the model learn these priors from data, resulting in a new Finite Admixture of Keys (FiAK). The learned priors in FiAK are used for pruning away redundant attention scores and keys in the baseline transformers, improving the diversity of attention patterns that the models capture. We corroborate the efficiency of transformers pruned with FiAK on practical tasks including ImageNet object classification, COCO object detection, and WikiText-103 language modeling. Our experiments demonstrate that transformers pruned with FiAK yield similar or even better accuracy than the baseline dense transformers while being much more efficient in terms of memory and computational cost.

Keywords: pruning, transformer, admixture models

1 Introduction

Transformers [21] have been becoming the method of choice in computer vision and machine learning [1,4,7,19]. Thanks to their ability to learn from unlabeled data and from different data modalities, transformers have achieved state-of-the-art performance on a wide range of tasks and applications, including image recognition, object detection, and language modeling [17,6,14]. At the core of transformers is the self-attention mechanism, which captures the contextual representation of the input sequence by allowing each token in the input sequence to pay attention to other tokens [21,3]. The capability of self-attention to attain diverse syntactic and semantic representations accounts for the success of transformers [18,10].

Self-Attention. Given an input $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D_x}$ of N feature vectors, the self-attention transforms it into sequence $\hat{\boldsymbol{V}} = [\hat{\boldsymbol{v}}_1, \ldots, \hat{\boldsymbol{v}}_N]^\top \in$

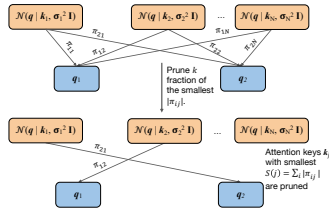

Fig. 1. Our Finite Admixture of Keys (FiAK) models the distribution of the queries $\boldsymbol{q}_i$ in self-attention by an admixture model whose cluster components center around the attention keys $\boldsymbol{k}_j$, i.e. $p(\boldsymbol{q}_i) = \sum_{j=1}^{N} \pi_{ij} \mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_j, \sigma_j^2 \mathbf{I})$, $i, j = 1, \ldots, N$. The prior distributions π_{ij} in the admixture are used to prune redundant attention scores $a_{ij} = \text{softmax}\Big(\frac{\boldsymbol{q}_i^\top \boldsymbol{k}_j}{\sqrt{D}}\Big)$. The scores $S(j) = \sum_i |\pi_{ij}|$ are used to prune redundant keys $\boldsymbol{k}_j$. A fraction of attention scores a_{ij} and keys $\boldsymbol{k}_j$ with the smallest $|\pi_{ij}|$ and $S(j)$, respectively, will be pruned away to save memory and computation.

in $\mathbb{R}^{N \times D}$ as follows

$$\hat{\boldsymbol{v}}_i = \sum_{j=1}^{N} \text{softmax}\Big(\frac{\boldsymbol{q}_i^\top \boldsymbol{k}_j}{\sqrt{D}}\Big)\boldsymbol{v}_j, \text{ for } i = 1, \ldots, N, \tag{1}$$

where the scalar $\text{softmax}((\boldsymbol{q}_i^\top \boldsymbol{k}_j)/\sqrt{D})$ can be understood as the attention $\hat{\boldsymbol{v}}_i$ pays to the input feature $\boldsymbol{x}_j$. The vectors $\boldsymbol{q}_i, \boldsymbol{k}_j$, and $\boldsymbol{v}_j$ are called the query, key, and value vectors, respectively; these vectors are computed as follows

$$\begin{aligned} [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_N]^\top &:= \boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q^\top \in \mathbb{R}^{N \times D}, \\ [\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_N]^\top &:= \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K^\top \in \mathbb{R}^{N \times D}, \\ [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N]^\top &:= \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V^\top \in \mathbb{R}^{N \times D_v}, \end{aligned} \tag{2}$$

where $\boldsymbol{W}_Q, \boldsymbol{W}_K \in \mathbb{R}^{D \times D_x}$, and $\boldsymbol{W}_V \in \mathbb{R}^{D_v \times D_x}$ are the weight matrices. We can further write Eqn. 1 into the following compact form

$$\hat{\boldsymbol{V}} = \text{softmax}\Big(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D}}\Big)\boldsymbol{V} = \mathbf{A}\mathbf{V}, \tag{3}$$

where the softmax function is applied to each row of the matrix $(\boldsymbol{Q}\boldsymbol{K}^\top)/\sqrt{D}$.

For each query vector $\boldsymbol{q}_i$ for $i = 1, \cdots, N$, an equivalent form of Eqn. 3 to compute the output vector $\hat{\boldsymbol{v}}_i$ is given by

$$\hat{\boldsymbol{v}}_i = \sum_{j=1}^{N} \text{softmax}\Big(\frac{\boldsymbol{q}_i^\top \boldsymbol{k}_j}{\sqrt{D}}\Big)\boldsymbol{v}_j := \sum_{j=1}^{N} a_{ij}\boldsymbol{v}_j. \tag{4}$$

The matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ and its component a_{ij} for $i, j = 1, \cdots, N$ are the attention matrix and attention scores, respectively. Eqn. 3 is also called the "scaled dot-product attention" or "softmax attention". The attention matrix $\mathbf{A}$ after training captures the contextual representation of each token.

Despite the success of transformers in capturing the contextual representation of tokens in the input sequence, it has been shown that the contextual

representation learned by the self-attention are redundant and many attention scores and keys explain the same patterns and are not needed [16,22,2]. Such redundancy wastes memory and computation during both training and inference while limiting the model's capacity, posing a challenge to scale up transformers to large-scale tasks.

Contribution. We propose a novel probabilistic model for self-attention, namely the Finite Admixture of Keys (FiAK), that allows pruning attention scores and keys using the prior distributions of attention keys. FiAK models the query distribution $p(\boldsymbol{q}_i)$ as an admixture of Gaussian distributions $\mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_j, \sigma_j^2 \mathbf{I})$ centering around the attention keys $\boldsymbol{k}_j$, $i, j = 1, \ldots, N$. Our admixture approach uses different mixture models to represent the queries $\boldsymbol{q}_i$ and thus helps increase the diversity of attention patterns. Since these mixture models share the same set of component distributions $\mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_j, \sigma_j^2 \mathbf{I})$, FiAK is efficient. The prior distributions of attention keys in FiAK are then used to prune redundant attention scores and keys to improve the memory and computational cost of the model. An illustration of FiAK and our pruning scheme is given in Fig. 1. Our contribution is three-fold:

1. We develop FiAK, a new finite admixture of keys for self-attention that allows key sharing to diversify attention patterns while guaranteeing the model's efficiency.
2. We design a probabilistic framework for pruning transformers that employs the prior distributions of keys in FiAK to remove redundant attention scores and keys.
3. We demonstrate the advantages of our FiAK-based pruning on ImageNet object classification, COCO object detection, and WikiText-103 language modeling tasks.

2 A Finite Admixture of Keys

In this section, we first review the connection between attention scores in self-attention with the posterior distributions from a Gaussian mixture model (GMM) in [**?**]. We then extend this GMM into a finite admixture of keys (FiAK).

2.1 Background: Attention Scores are Posterior Distributions from a GMM

Given a query $\boldsymbol{q}_i \in \mathbf{Q}$ and a key $\boldsymbol{k}_j \in \mathbf{K}$, let $\boldsymbol{t}$ be a K-dimensional binary random variable having a 1-of-K representation in which a particular element $\boldsymbol{t}_j$ is equal to 1 and all other elements are equal to 0. The distribution $p(\boldsymbol{q}_i|\boldsymbol{t}_j = 1)$ is the likelihood of the query $\boldsymbol{q}_i$ belongs to the j-th cluster centering around the key $\boldsymbol{k}_j$. In particular, let $\mathbf{1}$ be an identity matrix and π_j be the prior distribution $p(\boldsymbol{t}_j = 1)$, the distribution $p(\boldsymbol{q}_i)$ is given by the following GMM:

$$p(\boldsymbol{q}_i) = \sum_{j=1}^{N} \pi_j p(\boldsymbol{q}_i|\boldsymbol{t}_j = 1) = \sum_{j=1}^{N} \pi_j \mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_j, \sigma_j^2 \mathbf{1}), \tag{5}$$

Following Eqn. 5, the posterior $p(\boldsymbol{t}_j = 1|\boldsymbol{q}_i)$ captures how much the query $\boldsymbol{q}_i$ matches the key $\boldsymbol{k}_j$ and is computed by

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

Assuming that the query $\boldsymbol{q}_i$ and the key $\boldsymbol{k}_j$ are normalized, the prior π_j is uniform, and let $\sigma_j^2 = \sigma^2$, $j = 1, 2, \ldots, K$, the posterior $p(\boldsymbol{t}_j = 1|\boldsymbol{q}_i)$ can then be written in the following form

$$p(\boldsymbol{t}_j = 1|\boldsymbol{q}_i) = \frac{\exp\left(\boldsymbol{q}_i^\top \boldsymbol{k}_j/\sigma^2\right)}{\sum_{j'} \exp\left(\boldsymbol{q}_i^\top \boldsymbol{k}_{j'}/\sigma^2\right)} = \text{softmax}\Big(\boldsymbol{q}_i^\top \boldsymbol{k}_j/\sigma^2\Big).$$

The equation above becomes Eqn. (4) of the attention score a_{ij} when $\sigma^2 = \sqrt{D}$. Thus, under right assumptions, the attention score a_{ij} between the query $\boldsymbol{q}_i$ and the key $\boldsymbol{k}_j$ in a self-attention layer of a transformer plays the role of the posterior distribution $p(\boldsymbol{t}_j = 1|\boldsymbol{q}_i)$.

2.2 FiAK: A Finite Admixture of Keys

We extend the GMM of keys for self-attention in Eqn. 5 into a finite admixture of keys so that the attention score a_{ij} can capture more diverse attention patterns and provide a probabilistic framework for pruning transformers.

Finite Admixture Models A finite mixture distribution of N components for a random array $\mathbf{X} \in \mathbb{R}^{M \times D}$ is given by

$$\boldsymbol{x}_i \sim \sum_{j=1}^{N} p_j f(\boldsymbol{x}; \theta_j), \;\; \sum_{j=1}^{N} p_j = 1, \;\; p_j \geq 0, \tag{6}$$

where $\boldsymbol{x}_i \in \mathbb{R}^D$ is the i-th row of $\mathbf{X}$ randomly sampled from the mixture distribution. f is a chosen probability measure, such as a Gaussian distribution as in Eqn. 5, $p = \{p_1, \ldots, p_N\}$ are mixture weights that correspond to the prior π_j, and θ_j denotes the parameter values for the k-th component.

A finite admixture models (FAM) is a generalization of a FMM, in which rows $\boldsymbol{x}_i$, $i = 1, \ldots, M$, are drawn from different mixture distributions that share N components $f(\boldsymbol{x}; \theta_j)$, $j = 1, \ldots, N$ with different mixture weights

$$\boldsymbol{x}_i \sim \sum_{j=1}^{N} p_{ij} f(\boldsymbol{x}; \theta_j), \;\; \sum_{j=1}^{N} p_{ij} = 1, \;\; p_{ij} \geq 0. \tag{7}$$

Comparing to FMM, FAM has better representation capacity thanks to its flexibility in choosing the mixture components. Since all components are shared between mixtures in FAM, FAM is efficient in term of the model size and computational cost for sampling samples from the model.

Algorithm 1 Attention Score Pruning via FiAK

Hyperparameter $0 < k < 1$: k fraction of the attention scores a_{ij} to be pruned.
Step 1 Incorporate parameters π_{ij} into the self-attentions.
Step 2 Train the transformer with the additional parameters π_{ij} until convergence.
Step 3 Prune k fraction of the attention scores a_{ij} whose learned coefficients $|\hat{\pi}_{ij}|$ are the smallest.
Step 4 Set the remaining $\hat{\pi}_{ij} = 1$, which corresponds to uniform prior, and finetune the pruned network.

Finite Admixture of Keys We propose the finite admixture of keys (FiAK) for the queries in self-attention. In Eqn. 7, let the function $f(\boldsymbol{x}; \theta_j) = p(\boldsymbol{q}_i | \boldsymbol{t}_j = 1) = \mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_j, \sigma_j^2 \mathbf{I})$ and $p_{ij} = \pi_{ij} = p_i(\boldsymbol{t}_j = 1)$ where $\pi_{ij} = p_i(\boldsymbol{t}_j = 1)$ is the prior distribution $p(\boldsymbol{t}_j = 1)$ of the mixture corresponding to the query $\boldsymbol{q}_i$. FiAK is defined as:

Definition 1 (Finite Admixture of Keys). *Given a set of queries $\boldsymbol{q}_i$ and keys $\boldsymbol{k}_j$ in self-attention, $i, j = 1, \ldots, N$, the queries $\boldsymbol{q}_i$ admit a finite admixture of keys if $\boldsymbol{q}_i$ are sampled from the following finite admixture model:*

$$\boldsymbol{q}_i \sim = \sum_{j=1}^{N} \pi_{ij} \mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_j, \sigma_j^2 \mathbf{I}), \;\; \sum_{j=1}^{N} \pi_{ij} = 1, \;\; \pi_{ij} \geq 0. \tag{8}$$

3 Prior-based Pruning via FiAK

Using the prior π_{ij} in FiAK, we propose two novel pruning methods: 1) attention score pruning via FiAK and 2) mixed pruning via FiAK. For comparison with the GMM of keys in Section 2.1, we also derive 3) key pruning via GMM. In all of our proposed methods, attention scores and keys with the smallest importance weights, i.e. $|\hat{\pi}_{ij}|$, $\hat{S}(j)$, and $|\hat{\pi}_j|$ in Algorithm 1, 2, and 3 are pruned away.

Attention Score Pruning. The magnitude of the prior, $|\pi_{ij}|$, in FiAK implies how much the key $\boldsymbol{k}_j$ is needed to explain the query $\boldsymbol{q}_i$. These priors act as importance weights of the keys $\boldsymbol{k}_j$ given the query $\boldsymbol{q}_i$ and can be used to prune away the attention score a_{ij}, thus saving memory and computation when computing the self-attention (see Algorithm 1).

Mixed Pruning. To further reduce the computational complexity of the model, we introduce mixed pruning via FiAK in Algorithm 2. In addition to pruning the attention score a_{ij}, we derive the importance weights of the keys $\boldsymbol{k}_j$ and remove the pairs $(\boldsymbol{k}_j, \boldsymbol{v}_j)$ whose importance weights are the smallest. This strategy enables the pruned model to save computation not only at the attention calculation step, but also removes the key vector $\boldsymbol{k}_j$ and the value vector $\boldsymbol{v}_j$, as well as other computations related to these vectors in Eqn. 4.

Key Pruning. We introduce key pruning via GMM (Algorithm 3), which uses the learned prior $|\pi_j|$ in the GMM defined by Eqn. 5 as importance weights to prune the pairs $(\boldsymbol{k}_j, \boldsymbol{v}_j)$.

Finetuning the Pruned Network. FiAK introduces additional priors π_{ij} to capture the importance of the attention score a_{ij}. After pruning, those extra parameters can be removed by setting them to 1, which corresponds to

Algorithm 2 Mixed Pruning via FiAK

Hyperparameters $0 < k_1, k_2 < 1$: k_1 fraction of the total attention scores a_{ij} to be pruned; k_2 fraction of pairs (key, value) to be pruned.
Step 1 and **Step 2** Same as **Step 1** and **Step 2** of Algorithm 1.
Step 3 Calculate the importance score $\hat{S}(j)$ of each pair $(\boldsymbol{k}_j, \boldsymbol{v}_j)$:

$$\hat{S}(j) = \sum_i |\hat{\pi}_{ij}|, \text{ or } \frac{1}{N-j+1} \sum_i |\hat{\pi}_{ij}| \text{ for autoregressive tasks.}$$

Then prune k_2 fraction of the pairs $(\boldsymbol{k}_j, \boldsymbol{v}_j)$ with the smallest scores $\hat{S}(j)$.
Step 4 Prune $\hat{k}_1$ fraction of the remain unpruned a_{ij} whose corresponding $|\hat{\pi}_{ij}|$ are the smallest $\hat{k}_1 = 1 - \frac{1-k_1}{1-k_2}$.
Step 5 Follow **Step 4** of Algorithm 1.

Algorithm 3 Key Pruning via GMM

Hyperparameter $0 < k < 1$: k fraction of the keys to be pruned.
Step 1 Incorporate parameters π_j into the self-attentions.
Step 2 Train the transformer with the additional parameters π_j until convergence.
Step 3 Prune k fraction of the key-value pairs $(\mathbf{k_j}, \mathbf{v_j})$, whose corresponding learned mixing-coefficients $|\hat{\pi}_j|$ are the smallest.
Step 4 Set the remaining $\hat{\pi}_j = 1$, i.e. uniform prior, and finetune the pruned network.

using uniform priors. The network is then finetuned for more epochs to obtain competitive accuracy compared to the dense baseline network.

4 Experimental Results

We empirically corroborate the advantages of the models pruned via our proposed FiAK-based pruning methods over the dense baseline model on the ImageNet object classification task. We refer to tranformers that use FiAK-based attention defined by Eqn. 8 as FiAKformer and transformers that use GMM-based attention defined by Eqn. 5 as GMMformer.

Model and setting. We use the DeiT-tiny model [20] with 12 layers and 4 attention heads per layer. The model dimension is 192. To train the models, we follow the same setting and configuration as for the baseline [20], with the initialization of the learnable priors π_{ij} and π_j set to be $\frac{1}{\sqrt{N}}$ and $\frac{1}{N}$, respectively, where N is the input sequence's length.

Results. *Pruned models from attention score and mixed pruning via FiAK attain much better accuracy than the DeiT-tiny baseline while being significantly more efficient (See Table 1).* Attention score pruning via FiAK at different pruning fractions $k = 50\%, 60\%$ and 70% result in the highest accuracies. In particular, at the pruning fractions $k = 50\%$ and 60%, we observe substantial accuracy improvement over the dense baseline (1.33% and 1.44% in top-1 accuracy, respectively). These two pruned models also outperform the dense FiAKformer. On the other hand, mixed pruning with the same attention score pruning fraction, $k_1 = 70\%$ and different key pruning fractions, $k_2 = 15\%$ and 20%, gain better accuracy compared to the baseline while obtaining the most computation and memory reduction (See Fig. 2). Table 1 also shows the advantage of the FiAK-

Table 1. Top-1 and top-5 accuracy (%) of the pruned models from the attention score and mixed pruning via FiAK on the Imagenet dataset compared to the dense baseline DeiT-tiny [20].

Method	Top-1 Acc	Top-5 Acc
Baseline DeiT-tiny	72.23	91.13
GMMformer	72.96	91.64
Key pruning $k = 30\%$	71.57	90.80
FiAKformer	73.50	91.90
Attention-score pruning$k = 50\%$	73.56	**91.95**
Attention-score pruning $k = 60\%$	**73.67**	91.91
Attention-score pruning $k = 70\%$	73.09	91.57
Mixed pruning $k_1 = 70\%$, $k_2 = 15\%$	72.78	91.38
Mixed pruning $k_1 = 70\%$, $k_2 = 20\%$	72.25	91.14

Table 2. Comparison to other pruning methods on Imagenet task.

Method	FLOPS reduced (%)	Acc-1 (%)
DeiT-tiny	0.00	72.23
Head pruning [16]	**23.69**	68.59
S^2ViTE [**?**]	**23.69**	70.12
Attention-score pruning $k = 70\%$	8.50	**73.09**
Mixed pruning $k_1 = 70\%$, $k_2 = 20\%$	13.00	72.25
Mixed pruning $k_1 = 70\%$, $k_2 = 20\%$ + S^2ViTE [**?**]	22.76	72.24

based pruning over the GMM-based pruning and validate the need of using admixture to model the self-attention and design its effective pruning schemes.

Comparison to Other Pruning Methods. We compare our FiAK-based pruning schemes with other pruning methods for transformers on the ImageNet task (see Table 2 below). Compared to the head pruning [16] and S^2ViTE [**?**], our schemes prune the model less but increase its accuracy. Combining with the S^2ViTE [**?**], mixed FiAK pruning can increase the FLOPs reduction up to 22.76% while maintaining similar advantage in accuracy on the ImageNet task.

Other Tasks: Language Modeling on WikiText-103. To examine the effectiveness of our pruning methods across different data modalities, we experiment with the word-level language modeling task on WikiText-103 [15]. We summarize our results in Table 3. Same as the vision tasks above, attention score pruning via FiAK and mixed pruning via FiAK yield more efficient language models with competitive or even better performance than the dense baseline.

Efficiency Analysis. We investigate the improvement in efficiency of transformers pruned via FiAK-based and GMM-based approach over the baseline. In particular, we analyze the computation and memory complexity of the pruned models trained for the ImageNet object classification task. We summarize our results in Fig. 2. We observe that the *efficiency advantage of models pruned via FiAK over the baseline model grows with the sequence length.* FiAK-based pruning also wins in real time. On the ImageNet task, the latency for the dense baseline and our attention-score pruned FiAKformer, $k = 70\%$, are 508 and 649 images/second (on GPU) and 76 and 95 images/second (on CPU), respectively.

Table 3. Test perplexity of pruned FiAKformer for the language modeling task on Wikitext-103 dataset.

Method	Perplexity (PPL)
Baseline softmax transformer	34.29
FiAKformer	**33.69**
Attention score pruning 40%	33.88
Attention score pruning 50%	34.28
Mixed pruning $k_1 = 40\%$, $k_2 = 10\%$	34.21

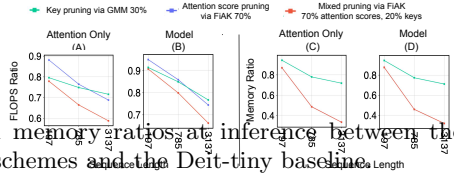

Fig. 2. FLOPS and memory ratios at inference between the models pruned with FiAK/GMM-based schemes and the Deit-tiny baseline.

5 Related Work

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 [5,16,8]. Works in pruning transformers can be categorized into two groups: 1) head pruning and 2) token pruning. An early work in head pruning calculates the head sensitivity to decide to prun a head or not [16]. [23] employs layerwise relevance propagation to decide the head importance. The head importance can also be learned in a data-driven manner as in [13]. For token pruning, [9] computes a token's importance score as average attention score of other tokens to that token. A dropout-based approach that stochastically determines a sequence length at each layer has also been used to prune redundant tokens [11]. [12] learns an attention mask for token pruning adaptively. Our FiAK-based approach is complementary to these methods.

6 Concluding Remarks

In this paper, we propose FiAK, a novel finite admixture of keys for self-attention, that model the distribution of queries $\boldsymbol{q}_i$ in self-attention as an admixture of Gaussian distributions $\mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_j, \sigma_j^2 \mathbf{I})$ whose centers are the attention keys $\boldsymbol{k}_j$, $i, j = 1, \ldots, N$. Using the prior distributions of the attention keys in FiAK, we propose a probabilistic pruning framework to remove redundant attention scores and keys in transformers. We verify that models pruned by our FiAK-based pruning methods improve the memory and computational cost over the baseline dense transformers while achieving comparable or better accuracy. Admixture models are equivalent to Latent Dirichlet Allocation (LDA) models under a uniform Dirichlet prior. Extending FiAK into an LDA-based framework for pruning transformers is an interesting research direction.

References

1. Al-Rfou, R., Choe, D., Constant, N., Guo, M., Jones, L.: Character-level language modeling with deeper self-attention. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 3159–3166 (2019)
2. Bhojanapalli, S., Chakrabarti, A., Jain, H., Kumar, S., Lukasik, M., Veit, A.: Eigen analysis of self-attention and its reconstruction from partial computation. arXiv preprint arXiv:2106.08823 (2021)
3. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734. Association for Computational Linguistics, Doha, Qatar (Oct 2014). https://doi.org/10.3115/v1/D14-1179, https://www.aclweb.org/anthology/D14-1179
4. Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q.V., Salakhutdinov, R.: Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860 (2019)
5. Dalvi, F., Sajjad, H., Durrani, N., Belinkov, Y.: Analyzing redundancy in pretrained transformer models. arXiv preprint arXiv:2004.04010 (2020)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
7. 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)
8. Durrani, N., Sajjad, H., Dalvi, F., Belinkov, Y.: Analyzing individual neurons in pre-trained language models. arXiv preprint arXiv:2010.02695 (2020)
9. Goyal, S., Choudhury, A.R., Raje, S., Chakaravarthy, V., Sabharwal, Y., Verma, A.: Power-bert: Accelerating bert inference via progressive word-vector elimination. In: International Conference on Machine Learning. pp. 3690–3699. PMLR (2020)
10. Hewitt, J., 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. Association for Computational Linguistics, Hong Kong, China (Nov 2019). https://doi.org/10.18653/v1/D19-1275, https://www.aclweb.org/anthology/D19-1275
11. Kim, G., Cho, K.: Length-adaptive transformer: Train once with length drop, use anytime with search. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 6501–6511. Association for Computational Linguistics, Online (Aug 2021). https://doi.org/10.18653/v1/2021.acl-long.508, https://aclanthology.org/2021.acl-long.508
12. Kim, S., Shen, S., Thorsley, D., Gholami, A., Kwon, W., Hassoun, J., Keutzer, K.: Learned token pruning for transformers. arXiv preprint arXiv:2107.00910 (2021)
13. Li, J., Cotterell, R., Sachan, M.: Differentiable subset pruning of transformer heads. Transactions of the Association for Computational Linguistics **9**, 1442–1459 (2021)
14. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)

15. Merity, S., Xiong, C., Bradbury, J., 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), https://openreview.net/forum?id=Byj72udxe
16. Michel, P., Levy, O., Neubig, G.: Are sixteen heads really better than one? In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019), https://proceedings.neurips.cc/paper/2019/file/2c601ad9d2ff9bc8b282670cdd54f69f-Paper.pdf
17. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training. OpenAI report (2018)
18. Tenney, I., Das, D., Pavlick, E.: BERT rediscovers the classical NLP pipeline. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 4593–4601. Association for Computational Linguistics, Florence, Italy (Jul 2019). https://doi.org/10.18653/v1/P19-1452, https://www.aclweb.org/anthology/P19-1452
19. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. arXiv preprint arXiv:2012.12877 (2020)
20. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. CoRR **abs/2012.12877** (2020), https://arxiv.org/abs/2012.12877
21. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
22. Voita, E., Talbot, D., Moiseev, F., Sennrich, R., Titov, I.: Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. arXiv preprint arXiv:1905.09418 (2019)
23. Voita, E., Talbot, D., Moiseev, F., Sennrich, R., 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. Association for Computational Linguistics, Florence, Italy (Jul 2019). https://doi.org/10.18653/v1/P19-1580, https://aclanthology.org/P19-1580

Supplement to "A Probabilistic Framework for Pruning Transformers via a Finite Admixture of Keys"

In this appendix, we include experimental details, additional experiments/visualization of pruning via FiAK, and the detailed derivation of the computational saving in Table **??**. We also provide code to reproduce our results in a separate folder in the supplementary material.

A Experiment details

In this section, we provide model and training details of our experiments in Section 4.

A.1 Object classification on Imagenet

Our baseline for the object classification task is the DeiT-tiny, a 4-head dense softmax transformer with 12 layers. In this baseline, the model dimension is of size 192, the feed-forward layer is of size 768, and the patch size is 16, which is equivalent to the sequence length of 197. Our FiAKformers and GMMformers have the same architecture configuration as the baseline with additional parameters π_{ij} and π_j as in Eqn.8 and 5, respectively.

All models are trained for 300 epochs, and the pruned models are fine-tuned on the Imagenet dataset for additional 100 epochs, using 4 A100 GPUs, 40 GB each, with batch size of 256. The initial learning rates for training and fine-tuning are 5×10^{-4} and 5×10^{-5}, respectively.

A.2 Object Detection on COCO

The pretrained baseline for the object detection task is the Swin-Transformer-tiny provided at https://github.com/SwinTransformer/Swin-Transformer-Object-Detection. All models have 12 attention layers with windows of size 7, which is equivalent to the sequence length of 49 patches per window.

The pruned models are fine-tuned for 12 epochs, using 8 A100 GPUs, 40GB each, with the initial learning rate is 10^{-4}. The weights are updated by an AdamW optimizer with the weight decay coefficient of 0.05.

A.3 Language Modeling on WikiText-103

For the language modeling task, we use the dense softmax transformer as our baseline. For all experiments, we use a transformer model that has 16 layers, 8 heads, feed-forward layer dimension of size 2048, embedding dimension of size 128, and hidden dimension of size 128. The context length for training and evaluation is set to 256.

We train our models using 2 A100 GPUs, 40GB each. We set the batch size to 96 and train our models for 120 epochs. We also apply dropout with dropout rate 10%. To optimize our models, we use Adam optimizer and Cosine annealing scheduler with initial learning rate 0.00025.

After the training phase, we prune the resulting models using one of our pruning schemes. Then, we finetune the pruned models for 30% time of the training phase, or equivalently 36 epochs.

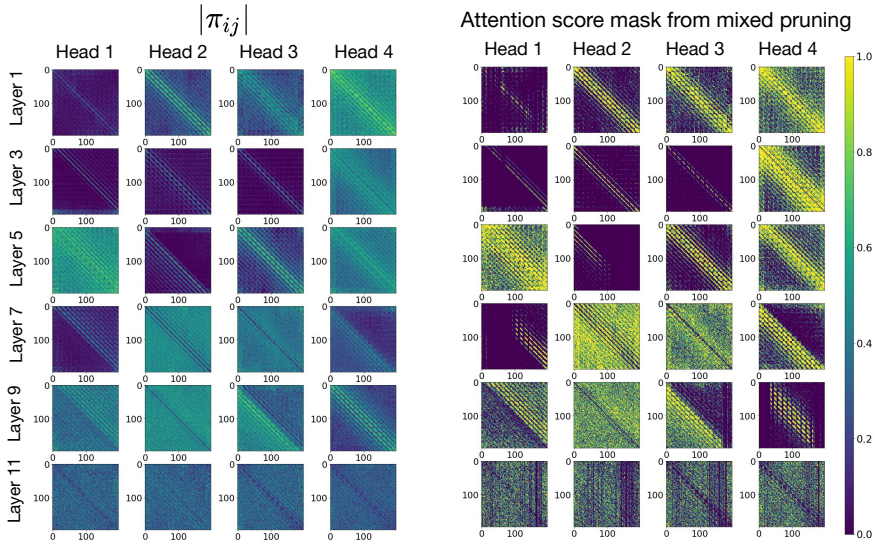

Fig. 3. (Left) The magnitude of the priors $|\pi_{ij}|$ learned from the ImageNet object classification task. (Right) The binary attention score masks from mixed pruning via FiAK with the pruning fraction $k_1 = 70\%$ and $k_2 = 15\%$. We plot these prior matrices and pruning masks for all 4 heads in layers {1, 3, 5, 7, 9, 11}.

B Additional Results on Visualizing the Pruning Masks

In this section, we provide additional results on visualizing the pruning masks learned from pruning via FiAK. Figure 3 depicts the pruning mask obtained with mixed pruning via FiAK for the ImageNet object classification task.

In Figure 4, we provide a detailed visual analysis of the parameters π_{ij} learned from the ImageNet object classification task. We obtained the query-centered mean values (Left) for each attention head by taking the mean of all π_{ij} values corresponding to the relative difference in position between the key and query. This results in the aggregation of the pattern learned by π_{ij}. We also show detailed patterns for the first 5 queries and keys (Right). Each cell in the 5×5 grid corresponds to the pruning mask applied to each query $\boldsymbol{q}_i$. From Figure 4, we observe that the pruning masks in early layers capture local/short-range attentions while the pruning masks in later layers capture non-local/long-range attentions.

C Additional Efficiency Analysis

In this section, we provide additional efficiency analysis for our pruning methods on the WikiText-103 language modeling task. In particular, we study the pruned models using attention score pruning via FiAK with the pruning fraction $k = 50\%$. We compare the FLOPS and memory ratios between the pruned model and the dense softmax transformer baseline at various sequence lengths {128, 256, 512, 1024, 2048, 4096}. As shown in Fig. 5, as the sequence

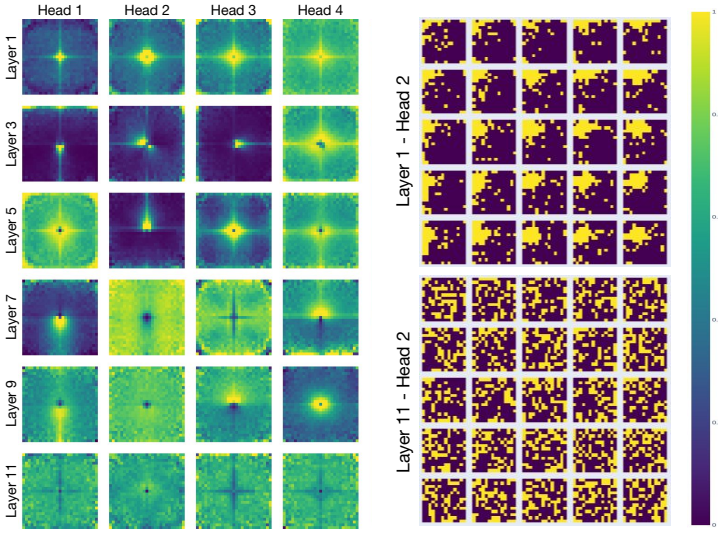

Fig. 4. (Left) Query-centered mean of the magnitude of the priors $|\pi_{ij}|$ for each head in layers {1, 3, 5, 7, 9, 11}, learned from the ImageNet task. (Right) The binary pruning masks of the attention scores for attention score pruning via FiAK with pruning fraction 70%, showing the differences of the pruning masks between layer 1 and layer 11. Pruning masks in early layers capture local/short-range attentions while those in later layers capture non-local/long-range attentions.

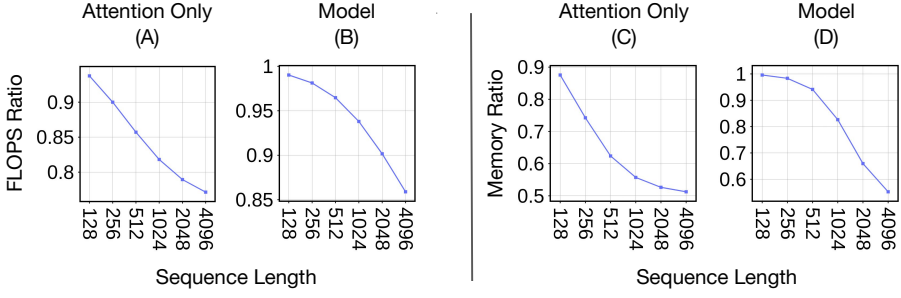

Fig. 5. FLOPS and memory ratios at inference time on the WikiText-103 language modeling task for model pruned using attention score pruning via FiAK with the pruning fraction $k = 50\%$, compared to the baseline dense softmax transformer model. For a thorough analysis, we show a comparison at attention block only ((A) and (C)) and for the entire model ((B) and (D)). The advantage of the FiAK-based pruning grows with the sequence length.

length grows, our pruned model becomes significantly more efficient in both memory and computations than the baseline.

D An Analysis on Computational Complexity of the Pruned Models vs. the Dense Model

In this section, we compare the computational complexity of models pruned by our pruning methods with the dense softmax baseline. Following the same notation in Section 3 in the main text, we denote H, D_x, N, and D as the number of attention heads at each layer, the input dimension, the input length, and the model/feature dimension, respectively. To simplify the notation and computation, without loss of generality, we assume that $D_v = D$, i.e., the values have the same feature dimension as the queries and the keys. We also do not take the softmax operator into account. Since the linear projection of the H-head concatenated outputs is the same for the baseline and our pruned models, its computation is discarded for simplification.

(i) Dense softmax attention: The computational complexity for an H-head attention matrix is $N^2H(4D-1) + NHD(6D_x - 4)$.

Explanation: The output of a self-attention block at each head is computed via the following three steps (See Sec. **??**).

- **Step 1** *Compute the matrices* $\mathbf{Q}$, $\mathbf{K}$ *and* $\mathbf{V}$ *via the linear transformations* $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, *and* $\boldsymbol{W}_V$. Since this step needs $3NDD_x$ multiplications and $3ND(D_x - 1)$ additions, the total computation is $3ND(2D_x - 1)$.
- **Step 2** *Calculate* $\mathbf{QK}^\top$. This needs N^2D multiplications and $N^2(D-1)$ additions, thus $N^2(2D-1)$ in total.
- **Step 3** *Compute the product* $\mathbf{AV}$. This requires N^2D multiplications and $N(N-1)D$ additions.

Hence the total amount of computation for an H-head attention is $N^2H(4D - 1) + NHD(6D_x - 4)$.

(ii) Computation reduction of attention score pruning via FiAK: Attention score pruned model via FiAK with the pruning fraction k (See Algo. 1) has $kHN^2(2D-1)$ less computations than the dense softmax attention.

Explanation: Attention score pruning via FiAK reduces the number of computation at step 2, i.e. calculating $\mathbf{QK}^\top$. Computations in other steps remain unchanged. Attention score pruned model with fraction of k does not calculate the dot product of k fraction of $(\boldsymbol{q}_i, \boldsymbol{k}_j)$ pairs, consequently saving $kHN^2(2D-1)$ computations.

(iii) Computation reduction of mixed pruning via FiAK: Mixed pruned model via FiAK with the pruning fraction k_1, k_2 (See Algo. 2) saves $2[(k_1 + k_2)D - k_1]HN^2 + (2D_x - 3)k_2HDN$ computations.

Explanation: Similar to attention score pruned model via FiAK, at each attention head, mixed pruned model via FiAK with total pruning fraction k_1 saves

$k_1N^2(2D-1)$ computation at step 2 above, i.e. calculating $\mathbf{QK}^\top$. Additionally, pruning k_2 fraction of $(\boldsymbol{k}_j, \boldsymbol{v}_j)$ pairs reduces computation at both step 1 and 3. At step 1, since matrix K and V accounts for $ND(D_x-1)$ computations per head each, pruning k_2 fraction of the pairs saves a total of $2k_2ND(D_x-1)$ computations. Meanwhile cutting off k_2 fraction of $\boldsymbol{v}_j$ leads to $k_2[N^2D+N(N-1)D]$ computations for each head. As a result, mixed pruned model via FiAK saves a total of $2[(k_1+k_2)D-k_1]HN^2+(2D_x-3)k_2HDN$ computations for an H-head attention.

(iv) Computation reduction of key pruning via GMM: Key pruned model via GMM with the key pruning fraction k (see Algo. 3) saves a total of $kHN^2(4D-1)+(2D_x-3)kHDN$ computations.

Explanation: As in mixed pruned model via FiAK, pruning k fraction of $(\boldsymbol{k}_j, \boldsymbol{v}_j)$ via GMM saves $kND(D_x-1)$ and $k[N^2D+N(N-1)D]$ computations at step 1 and 3, respectively. Moreover, for each head, pruning k fraction of keys also saves $kHN^2(2D-1)$ computations at step 2. In total, key pruned model via GMM needs $kHN^2(4D-1)+(2D_x-3)kHDN$ computations less than the dense softmax baseline.

Notice that the computation reduction is quadratic in the sequence length N. Therefore, when N is large, i.e. long input sequences, the computational reduction achieved from using our FiAK-based pruning methods significantly increases.