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 [68] have been becoming the method of choice in computer vision and machine learning [1,14,73,17,8,28,52,16,60,18,65]. 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 [50,51,17,74,40]. Lying at the heart 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 [10,46,38,4,68,33]. In particular, self-attention represents each token as the weighted average of the other tokens' feature representation using the similarity scores between the tokens as weight coefficients. The capability of self-attention to attain diverse syntactic and semantic representations accounts for the success of transformers in practice [64,69,13,70,25].

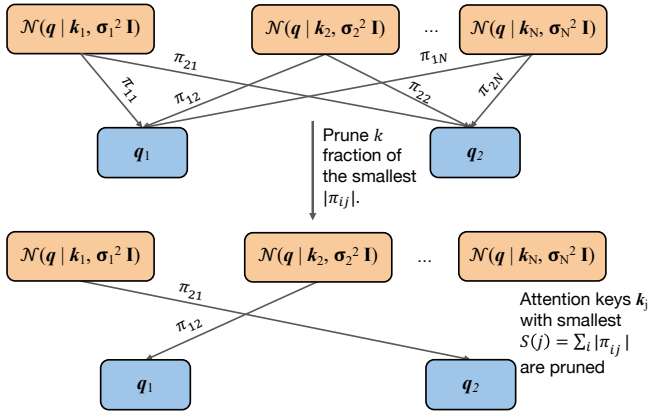

Fig. 1. Our Finite Admixture of Keys (FiAK) models the distribution of the queries $\boldsymbol{q}_i$ in self-attention by an admixture model whose cluster components center around the attention keys $\boldsymbol{k}_j$, i.e. $p(\boldsymbol{q}_i) = \sum_{j=1}^{N} \pi_{ij} \mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_j, \sigma_j^2 \mathbf{I})$, $i, j = 1, \ldots, N$. 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.

1.1 Self-Attention

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

Multi-head Attention Eqn. 3 corresponds to an attention head. In multi-head attention, output sequences $\hat{\boldsymbol{V}}_h$, $h = 1, \ldots, H$ are computed from H attention heads and then concatenated. Let $W^O \in \mathbb{R}^{HD\times HD}$ be the projection matrix for the output. The multi-head attention is defined as

$$\hat{\boldsymbol{Z}} = \text{MultiHead}(\{\hat{\boldsymbol{V}}\}_{h=1}^{H}) = \text{Concatenate}(\hat{\boldsymbol{V}}_1, \ldots, \hat{\boldsymbol{V}}_H)\mathbf{W}^O. \tag{5}$$

The final output of the self-attention layer $T_\ell(\cdot)$ is computed via the following residual connection.

$$T_\ell(\boldsymbol{X}) = f_\ell(\hat{\boldsymbol{Z}} + \boldsymbol{X}), \tag{6}$$

where $f_\ell(\cdot)$ is a function that transforms each feature vector independently and usually chosen to be a feedforward network. In this paper, we call a transformer built with softmax attention softmax transformer or transformer. It is easy to see that both memory and computational complexity of Eqn. 3 are $\mathcal{O}(N^2)$ with N being the length of the input sequence. We can further introduce causal masking into Eqn. 3 for autoregressive applications [68].

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. Particularly, attention heads in transformers tend to learn similar attention patterns. Also, many attention scores and keys within each head explain the same patterns and are not needed [43,71,6]. Such redundancy wastes memory and computation during both training and inference while limiting the representation capacity of the model, posing a challenge to scale up transformers to large-scale tasks.

1.2 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. Also, 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 efficiency of the model.
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 protocols on Imagenet object recognition, COCO object detection, and WikiText-103 language modeling tasks.

1.3 Organization

We structure this paper as follows: In Section 2, we derive the connection between attention scores and posterior distributions from a Gaussian mixture model and then present a finite admixture of keys (FiAK). We formulate our probabilistic pruning framework for transformers via FiAK in Section 3. In Section 4, we validate and empirically analyze the advantages of our FiAK-based pruning framework. We discuss related works in Section 5. The paper ends up with concluding remarks. More results and details are provided in the Appendix.

2 A Finite Admixture of Keys

In this section, we first establish the connection between attention scores in self-attention with the posterior distributions from a Gaussian mixture model (GMM). We then extend this GMM into a finite admixture of keys (FiAK) to capture more diverse patterns of attention and for pruning attention scores and keys later.

2.1 Attention Scores are Posterior Distributions from a Gaussian Mixture Model

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:

$$p(\boldsymbol{q}_i) = \sum_{j=1}^{N} \pi_j p(\boldsymbol{q}_i|\boldsymbol{t}_j = 1) = \sum_{j=1}^{N} \pi_j \mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_j, \sigma_j^2 \mathbf{1}), \tag{7}$$

Following Eqn. 7, 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

$$\begin{aligned} p(\boldsymbol{t}_j = 1|\boldsymbol{q}_i) &= \frac{\pi_j \mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_j, \sigma_j^2)}{\sum_{j'} \pi_{j'} \mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_{j'}, \sigma_{j'}^2)} \\ &= \frac{\pi_j \exp\left(-\|\boldsymbol{q}_i - \boldsymbol{k}_j\|^2 / 2\sigma_j^2\right)}{\sum_{j'} \pi_{j'} \exp\left(-\|\boldsymbol{q}_i - \boldsymbol{k}_{j'}\|^2 / 2\sigma_{j'}^2\right)} \\ &= \frac{\pi_j \exp\left[-\left(\|\boldsymbol{q}_i\|^2 + \|\boldsymbol{k}_j\|^2\right)/2\sigma_j^2\right] \exp\left(\boldsymbol{q}_i^\top \boldsymbol{k}_j / \sigma_j^2\right)}{\sum_{j'} \pi_{j'} \exp\left[-\left(\|\boldsymbol{q}_i\|^2 + \|\boldsymbol{k}_{j'}\|^2\right)/2\sigma_{j'}^2\right] \exp\left(\boldsymbol{q}_i^\top \boldsymbol{k}_{j'} / \sigma_{j'}^2\right)}. \end{aligned} \tag{8}$$

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

$$p(\boldsymbol{t}_j = 1|\boldsymbol{q}_i) = \frac{\exp\left(\boldsymbol{q}_i^\top \boldsymbol{k}_j / \sigma^2\right)}{\sum_{j'} \exp\left(\boldsymbol{q}_i^\top \boldsymbol{k}_{j'} / \sigma^2\right)} = \text{softmax}\Big(\boldsymbol{q}_i^\top \boldsymbol{k}_j / \sigma^2\Big). \tag{9}$$

The right-hand side of Eqn. (9) is the formula for the attention score a_{ij} as shown in Eqn. (4) 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 $p(\boldsymbol{t}_j = 1|\boldsymbol{q}_i)$, which indicates the *responsibility* that the key $\boldsymbol{k}_j$ takes for 'explaining' the query $\boldsymbol{q}_i$. In other words, the attention score a_{ij}, as a posterior distribution, implies how much a token at position i pays attention to a token at position j in the input sequence.

Remark 1. The assumption that the query $\boldsymbol{q}_i$ and the key $\boldsymbol{k}_j$ are normalized is from the observation that in many applications, those two vectors are normalized. [57] suggests that such normalization is to stabilize the training. Also, uniform prior is commonly used in practice when no prior knowledge is provided.

2.2 FiAK: A Finite Admixture of Keys

The GMM in Eqn. 7 does not take into account the temporal order of the queries $\boldsymbol{q}_i$. In particular, all queries $\boldsymbol{q}_i$ are from the same mixture distribution. In many practical settings such as in the autoregressive tasks, the temporal order of the queries $\boldsymbol{q}_i$ plays an important role. We extend the GMM of keys for self-attention in Eqn. 7 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 We first review the finite mixture models (FMMs), such as the GMM in Eqn. 7 above, which served as a workhorse in stochastic modeling, and then discuss the finite admixture models (FAM). A finite mixture distribution of N components for a random array $\mathbf{X} \in \mathbb{R}^{M \times D}$ is given by

$$\boldsymbol{x}_i \sim \sum_{j=1}^{N} p_j f(\boldsymbol{x}; \theta_j), \quad \sum_{j=1}^{N} p_j = 1, \;\; p_j \geq 0, \tag{10}$$

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. 7, $p = \{p_1, p_2, \ldots, p_N\}$ are mixture weights that correspond to the prior π_j in Eqn. 7, and θ_j denotes the parameter values for the k-th component.

A FAM is a generalization of a FMM, in which rows $\boldsymbol{x}_i$, $i = 1, 2, \ldots, M$, are drawn from different mixture distributions that share the same N components $f(\boldsymbol{x}; \theta_j)$, $j = 1, 2, \ldots, N$ but with different mixture weights

$$\boldsymbol{x}_i \sim \sum_{j=1}^{N} p_{ij} f(\boldsymbol{x}; \theta_j), \;\; \sum_{j=1}^{N} p_{ij} = 1, \;\; p_{ij} \geq 0. \tag{11}$$

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.

Finite Admixture of Keys We propose the finite admixture of keys (FiAK) for the queries in self-attention. In Eqn. 11, 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 follows:

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:*

$$\boldsymbol{q}_i \sim \sum_{j=1}^{N} \pi_{ij} p(\boldsymbol{q}_i | \boldsymbol{t}_j = 1) = \sum_{j=1}^{N} \pi_{ij} \mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_j, \sigma_j^2 \mathbf{I}),$$
$$\sum_{j=1}^{N} \pi_{ij} = 1, \;\; \pi_{ij} \geq 0. \tag{12}$$

Note that the difference between FiAK and the Gaussian mixture of keys in Eqn. 7 is that the prior distribution π_{ij} are different for each query $\boldsymbol{q}_i$, i.e. $\boldsymbol{q}_i$ are sampled from different mixtures that share the same N components $\mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_j, \sigma_j^2 \mathbf{I})$, $i, j = 1, \ldots, N$.

Remark 2 (FiAK as a Topic Model). FiAK can be connected to the Probabilistic Latent Semantic Analysis (pLSA) model for topic modeling [27]. Given the document d that contains the word w sampled from the topic c, pLSA models the occurrence of the word w in the document d as a mixture of conditionally independent Multinomial distributions:

$$p(w|d) = \sum_{c} p(c|d) p(w|c). \tag{13}$$

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 corresponding 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.

Comparing Eqn. 12 of FiAK and Eqn. 13 of pLSA, we can interpret the mixture weights π_{ij} and the distribution $\mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{k}_j, \sigma_j^2 \mathbf{I})$ in FiAK as the distributions $p(c|d)$ and $p(w|c)$ in pLSA, respectively. As a result, the attention keys in FiAK correspond to the topics, and the queries are words sampled from those topics. pLSA and thus FiAK are also equivalent to the well-known Latent Dirichlet Allocation model under a uniform Dirichlet prior on the per-document topic distribution $p(c|d)$ [7].

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. Here, attention score pruning removes the redundant attention scores a_{ij}, key pruning removes the redundant attention keys $\boldsymbol{k}_j$ together with its corresponding value vectors $\boldsymbol{v}_j$, and mixed pruning removes both attention scores and keys, as well the corresponding value vectors $\boldsymbol{v}_j$ as in key pruning. 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.

1. Attention Score Pruning via FiAK 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 $a_{ij} = \text{softmax}\left(\frac{\boldsymbol{q}_i^\top \boldsymbol{k}_j}{\sqrt{D}}\right)$, thus saving memory and computation when computing the self-attention. The pruning algorithm via FiAK is given in Algorithm 1.

2. Mixed Pruning via FiAK To further reduce the computation 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 complete 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.

3. Key Pruning via Gaussian Mixture Model For a comparison between admixture-based pruning and mixture-based pruning, we introduce key pruning via GMM (Algorithm 3), which uses the learned prior $|\pi_j|$ in the GMM defined by Eqn. 7 as importance weights to prune the pairs $(\boldsymbol{k}_j, \boldsymbol{v}_j)$.

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 $(\boldsymbol{k}_j, \boldsymbol{v}_j)$:

$$\hat{S}(j) = \sum_i |\hat{\pi}_{ij}|, \tag{14}$$

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, where:

$$\hat{k}_1 = 1 - \frac{1 - k_1}{1 - k_2} \tag{15}$$

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 $\hat{\pi}_j = 1$, which corresponds to uniform prior, and finetune the pruned network.

Finetuning the Pruned Network FiAK (Eqn.12) introduces additional priors π_{ij} to capture the importance of the attention score a_{ij}. After the attention scores are pruned, those extra parameters can be removed by setting them to 1, which corresponds to using uniform priors. The network is then finetuned for more epochs to obtain competitive accuracy compared to the dense baseline network. The same finetuning strategy is used for key pruning via GMM.

Computational Complexity Reduction from Pruning Table 1 shows the computational saving from the multi-head attention pruned by our methods compared to the dense baseline softmax attention. Here, H is the number of attention heads, and the computational cost is computed as the total number of additions and multiplications needed. Our analysis results in Table 1 indicate that our pruning methods save more computations as we scale up the model for longer sequences, i.e. the sequence length N is large. Note that the saving in computational cost is quadratic in terms of N. The details of our derivation are given in the Appendix.

4 Experimental Results

In this section, we empirically corroborate the advantages of the models pruned via our proposed FiAK-based pruning methods over the dense baseline model on various benchmarks, including ImageNet object classification, COCO object detection, and WikiText-103 language modeling. We aim to show that: (i)

Table 1. Computational saving achieved by using our proposed pruning methods to prun a dense H-head softmax attention. Here, H, N, D, D_x denote the number of attention heads, sequence length, head dimension, and model/input dimension, respectively. Parameters k and (k_1, k_2) are the fraction to be pruned as explained in Algorithm 1, 2 and 3. As demonstrated, the advantages of our pruned models increase when we scale up the model for longer sequences.

Method	Hyper-parameters	Computational Saving for H-head Attention
Attention Score Pruning via FiAK	k	$kHN^2(2D-1)$
Mixed-Pruning via FiAK	k_1, k_2	$2[(k_1+k_2)D-k_1]HN^2+(2D_x-3)k_2HDN$
Key Pruning via GMM	k	$kHN^2(4D-1)+(2D_x-3)kHDN$

the FiAK-based pruned models are more efficient than the dense baseline in term of memory and computation cost while achieving comparable or better accuracy; (ii) the FiAK-based pruning methods, i.e. attention score pruning and mixed pruning via FiAK, are more effective than the GMM-based pruning method, i.e. key pruning via GMM, resulting in more efficient and accurate pruned models.

Throughout this section, we refer to tranformers that use FiAK-based attention defined by Eqn. 12 as FiAKformer and transformers that use GMM-based attention defined by Eqn. 7 as GMMformer. Except for the ImageNet image classification task, in other experiments in this section, we use the attention score pruning via FiAK to study the FiAK-based pruning. More results on the mixed pruning via FiAK, as well as details on datasets, models, and training are provided in the Appendix.

4.1 Image classification on Imagenet

Datasets and metrics The ImageNet dataset [54] consists of $1.28M$ training images and $50K$ validation images. For this benchmark, the model learns to predict which category among 1000 categories the input image belongs to. Top-1 and top-5 classification accuracies are reported.

Model and setting We use the DeiT-tiny model [66] with 12 transformer 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 [66], 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 length of the input sequence.

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 2).* 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. 3). These results show the effectiveness of our pruning schemes for the image classification task. Section 4.4 provides details on the efficiency analysis of our pruned models on this task.

Table 2. 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 [66]. We also show the top-1 and top-5 accuracy (%) of the pruned model from the key pruning via GMM on the same task for comparison. Here, pruning fractions of attention scores via FiAK and keys via GMM are k. For mixed pruning via FiAK, pruning fractions for attention scores and keys are k_1 and k_2, respectively. FiAK-based pruning schemes result in pruned models with much better accuracies than the dense baseline DeiT-tiny and are more effective than the GMM-based pruning scheme.

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

Comparison with GMM-based pruning: Table 2 shows that while the GMMformer yields better accuracy than the baseline, by pruning 30% of attention keys (or equivalently key-value pairs), the pruned model performs worse than the baseline. *This results show the advantage of the FiAK-based pruning over the GMM-based pruning and validate the need of using admixture, such as FiAK, to model the self-attention and design its effective pruning schemes.*

Visualizing the pruning masks In Figure 2, we visualize the magnitude of the priors π_{ij} at each of the 4 heads in layers {1, 3, 5, 7, 9, 11} (Left), as well as the binary pruning masks when we prune 70% of the attention scores with attention score pruning via FiAK (Right). We observe that the matrices of the priors π_{ij} (Left) are sparse with large values of π_{ij} are near the diagonal. This suggests that the queries $\boldsymbol{q}_i$ pay attention to the keys $\boldsymbol{k}_j$ in its local neighborhood. Therefore, when pruning is applied, we can remove many redundant attention scores a_{ij} where i and j are far away from each other.

4.2 Object Detection on COCO with a Pre-trained Model.

Pruning methods via FiAK are universal and can also be applied on pre-trained transformers finetuned for downstream tasks. In this section, we demonstrate the effectiveness our FiAK-based pruning schemes for a pre-trained Swin transformer finetuned for the object detection task on the COCO dataset [37].

Datasets and metrics Our experiments are conducted on the COCO 2017 [37]. We use Mask R-CNN [24] as our object detection framework. Trainings are done on 118K training images, and box mean average precisions (box mAP) on 5K validation images are reported.

Model and baselines *Pruning via FiAK methods are easy to apply on the pre-trained models.* Following Algorithm 1, we apply the attention score pruning via FiAK on the pre-trained tiny Swin Transformer [41] for the object detection

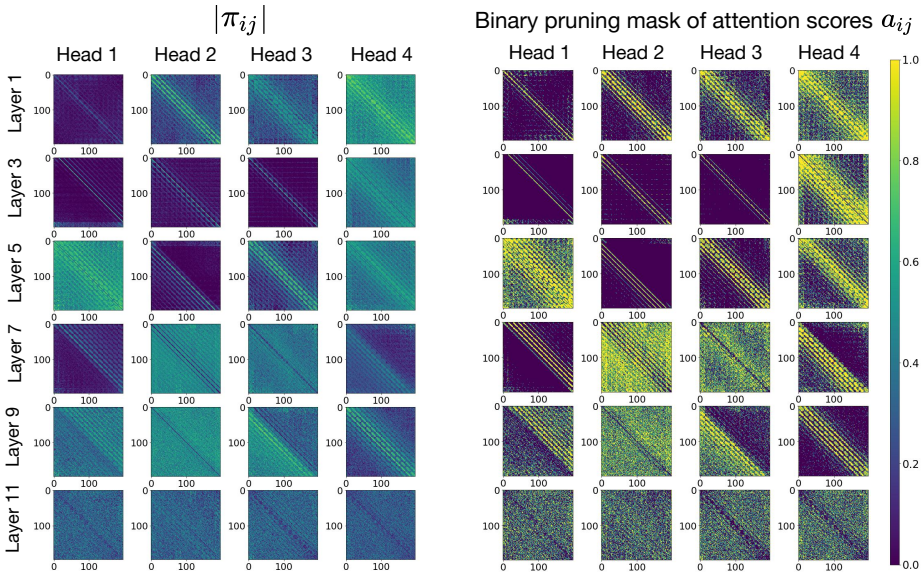

Fig. 2. (Left) The magnitude of the priors, $|\pi_{ij}|$, learned from the ImageNet object classification task, and (Right) the binary pruning masks of the attention scores for attention score pruning via FiAK with the pruning fraction $k = 70\%$. We plot these prior matrices and pruning masks for all 4 heads in layer $\{1, 3, 5, 7, 9, 11\}$. The visualization of $|\pi_{ij}|$ shows that the prior matrices are sparse, allowing most attention scores to be pruned.

task on the COCO dataset. We first introduce additional parameters for the priors π_{ij} into self-attentions of the pre-trained model and then learn these priors by finetuning the model for 12 epochs. After pruning the attention scores a_{ij}, we do another finetuning round with uniform priors as in Algorithm 1. The entire process takes less than one-tenth of the time used for training a Swin transformer from scratch, indicating the efficiency of applying our method. We follow the same configuration and setting for the baseline as provided in [41].

Results Table 3 indicates that at pruning fraction $k = 60\%$, Swin transformers pruned with attention score pruning via FiAK perform competitively with the dense baseline. The simple usage and competitive performance of pruning via FiAK on this challenging object detection task demonstrate the universal applicability of our methods.

4.3 Language Modeling on WikiText-103

To further examine the effectiveness of our pruning methods across different data modalities, we experiment with the word-level language modeling task on WikiText-103 [42] in this section.

Datasets and metrics. WikiText-103 is a dataset with long contextual dependencies and is made up of articles from Wikipedia. The training set consists of around $28K$ articles containing $103M$ running words corresponding to text blocks of about 3600 words. There are $218K$ and $246K$ running words in the validation and test sets, respectively. Each of them contains 60 articles and about $268K$

Table 3. Box mean average precision (box mAP) of Swin tranformers pruned by attention score pruning via FiAK compared to dense baseline Swin Transformer on the COCO validation set. The pruned model with the pruning fraction $k = 60\%$ achieves comparable results with the dense baseline. The box mAP of the pretrained baseline is reported at https://github.com/SwinTransformer/Swin-Transformer-Object-Detection.

Method	Box mAP
Baseline Swin transformer	43.7
FiAKformer	**44.7**
Attention-score pruned FiAKformer 60%	43.3

Table 4. Test perplexity of pruned FiAKformer for the language modeling task on Wikitext-103 dataset. We apply attention score pruning via FiAK and prune 40% and 50% of the attention scores. The results show that the pruned models after finetuning can reach comparable or better accuracy than the dense baseline.

Method	Perplexity (PPL)
Baseline softmax transformer	34.29
FiAKformer	**33.69**
Attention score pruned FiAKformer 40%	33.88
Attention score pruned FiAKformer 50%	34.28

words. In our experiment, we split the training data into L-word independent long segments, following the standard setting [42,57]. 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 to compute the perplexity (PPL) except in the first segment, where all positions are evaluated as in [1,57].

Models and baselines The baseline model we use for all of our experiments is the 8-head softmax baseline, which consists of 16 transformers layers [68]. We follow the same experiment settings from [57].

Results We summarize our results in Table 4. Same as the vision tasks above, attention score pruning via FiAK yields more efficient language models with competitive or even better performance than the dense baseline.

4.4 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. For attention score pruning via FiAK (Algorithm 1), we study the pruned model with pruning fraction $k = 70\%$. For mixed pruning via FiAK (Algorithm 2), we study the pruned model with pruning fractions $k_1 = 70\%$ and $k_2 = 20\%$. For comparison, we also analyze key pruning via GMM (Algorithm 3) using a pruned model with pruning fraction $k = 30\%$. As shown in Table 2, model pruned by key pruning via GMM with pruning fraction $k = 30\%$ already yields worse accuracy than models pruned via FiAK.

Efficiency Advantage of Models Pruned via FiAK over the Baseline Model Grows with the Sequence Length. In Fig. 3, we compare the floating point operations

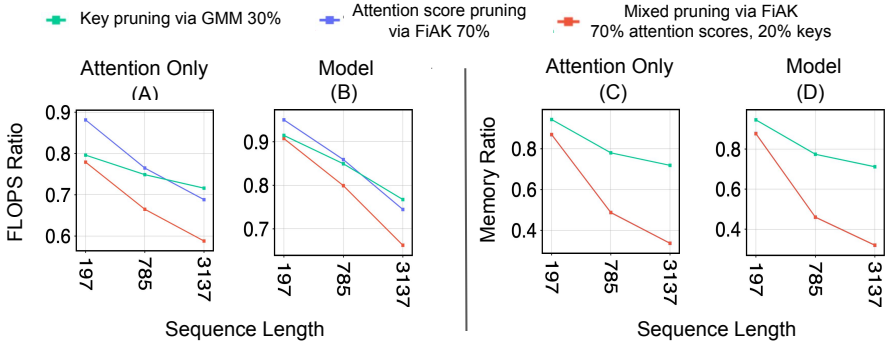

Fig. 3. FLOPS and memory ratios at inference between the models pruned with FiAK-based/GMM-based pruning schemes and the dense Deit-tiny baseline. For a thourough analysis, we show a comparison at attention block only ((A) and (C)) and for the entire model ((B) and (D)). Note that in (C) and (D), the memory ratios for attention score and mixed pruning via FiAK are the same. The advantage of the FiAK-based pruning grows with the sequence length. Compared to the GMM-based pruning, FiAK-based pruning schemes are more effective.

per second (FLOPS) ratios and the memory ratios at inference time between the models pruned via FiAK and the dense baseline for multiple sequence lengths {197, 785, 3137}. In Fig. 3, (A) and (C) shows the FLOPS and memory ratios for the attention blocks, respectively, while (B) and (D) shows these ratios for the whole model, respectively. In (C) and (D), the memory ratios for attention score and mixed pruning via FiAK are the same. Overall, we observe that our pruning schemes result in much more efficient models for long sequences in terms of computation and memory compared to the dense baseline, and this advantage becomes more significant for longer sequences.

Mixed-pruning via FiAK gains the most advantage in FLOPS among different pruning methods while still achieving competitive performance to the dense baseline. This trend continues as the sequence length increases. Attention score and mixed pruning via FiAK share the same benefits of substantial memory reduction. As shown in Fig. 3(C) and (D), at sequence length $N = 785$ and 3137, the pruned models via FiAK save more than 60% of the baseline memory, both for each multi-head attention block and the whole model.

FIAK-based pruning methods result in more efficient models with better accuracy than the GMM-based pruning (See Table 2). This again proves the advantage of modeling self-attention as an admixture model rather than a mixture model.

5 Related Work

Pruning and Reducing 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 [15,43,19]. 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 [43]. [70] employs layerwise relevance propagation to decide the head importance. The head importance can also be learned in

a data-driven manner as in [36]. For token pruning, [22] 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 [31]. In addition, [32] proposes an adaptive approach that learns an attention mask for token pruning. 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 [44,20]. Knowledge distillation and sparse approximation have also been used to enhance the efficiency of transformers, including [56,62,71,55]. Our FiAK-based approach can be used along these methods to improve their performance.

Efficient Transformers To lower the quadratic computational and memory cost of transformers, efficient transformers have been studied [53]. Among them are sparse transformers which incorporate sparse structures into the attention matrix [47,39,49,9,5]. Another class of efficient transformers are models that aim to have better coverage by integrating different access patterns [9,26], which can also be learned from the data [34,53,63]. In other works, a side memory module is utilized in order to access multiple tokens simultaneously [35,61,2,5]. In another line of work, low-rank and kernelization methods have recently been proposed to improve the computational and memory efficiency of self-attention calculation [67,72,30,12,59,45,48]. Multi-query attention, which shares keys and values between different attention heads [58], has also been studied in order to reduce the memory-bandwidth cost and speed up incremental inference in transformers. Finally, approaches that leverage auxiliary losses [1] and adaptive input embedding [3] have been studied to accelerate the training process of transformers. Our FiAK are orthogonal and complementary to these methods.

Mixture Models for Transformers Recently, mixture models have been employed to study and improve transformers. Among these works is switch transformers [21] that make use of the routing algorithm in Mixture of Experts (MoE) to reduce the communication and computational costs in transformers. Other works that combine mixture models with transformers include [11,23,29].

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. As mentioned in Remark 2, 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 for future work.

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. Asai, A., Choi, E.: Challenges in information seeking qa: Unanswerable questions and paragraph retrieval. arXiv preprint arXiv:2010.11915 (2020)
3. Baevski, A., Auli, M.: Adaptive input representations for neural language modeling. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=ByxZX20qFQ
4. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
5. Beltagy, I., Peters, M.E., Cohan, A.: Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150 (2020)
6. 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)
7. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of machine Learning research **3**(Jan), 993–1022 (2003)
8. Brown, T., et al.: Language models are few-shot learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 1877–1901 (2020), https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf
9. Child, R., Gray, S., Radford, A., Sutskever, I.: Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509 (2019)
10. 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
11. Cho, S.M., Park, E., 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)
12. 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., Weller, A.: Rethinking attention with performers. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=Ua6zuk0WRH
13. Clark, K., Khandelwal, U., Levy, O., 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. Association for Computational Linguistics, Florence, Italy (Aug 2019). https://doi.org/10.18653/v1/W19-4828, https://www.aclweb.org/anthology/W19-4828
14. 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)
15. Dalvi, F., Sajjad, H., Durrani, N., Belinkov, Y.: Analyzing redundancy in pretrained transformer models. arXiv preprint arXiv:2004.04010 (2020)

16. Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., Kaiser, L.: Universal transformers. arXiv preprint arXiv:1807.03819 (2018)
17. 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)
18. 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)
19. Durrani, N., Sajjad, H., Dalvi, F., Belinkov, Y.: Analyzing individual neurons in pre-trained language models. arXiv preprint arXiv:2010.02695 (2020)
20. Ethayarajh, K.: How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. arXiv preprint arXiv:1909.00512 (2019)
21. Fedus, W., Zoph, B., Shazeer, N.: Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. arXiv preprint arXiv:2101.03961 (2021)
22. 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)
23. Guo, M., Zhang, Y., Liu, T.: Gaussian transformer: a lightweight approach for natural language inference. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 6489–6496 (2019)
24. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
25. 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
26. Ho, J., Kalchbrenner, N., Weissenborn, D., Salimans, T.: Axial attention in multidimensional transformers. arXiv preprint arXiv:1912.12180 (2019)
27. Hofmann, T.: Probabilistic latent semantic analysis. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence. p. 289–296. UAI'99, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999)
28. Howard, J., 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. Association for Computational Linguistics, Melbourne, Australia (Jul 2018). https://doi.org/10.18653/v1/P18-1031, https://www.aclweb.org/anthology/P18-1031
29. Jiang, J., Xia, G.G., Carlton, D.B., Anderson, C.N., 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)
30. Katharopoulos, A., Vyas, A., Pappas, N., Fleuret, F.: Transformers are rnns: Fast autoregressive transformers with linear attention. In: International Conference on Machine Learning. pp. 5156–5165. PMLR (2020)
31. 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 Interna-

tional 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
32. 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)
33. Kim, Y., Denton, C., Hoang, L., Rush, A.M.: Structured attention networks. arXiv preprint arXiv:1702.00887 (2017)
34. Kitaev, N., Kaiser, L., Levskaya, A.: Reformer: The efficient transformer. arXiv preprint arXiv:2001.04451 (2020)
35. Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., 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)
36. Li, J., Cotterell, R., Sachan, M.: Differentiable subset pruning of transformer heads. Transactions of the Association for Computational Linguistics **9**, 1442–1459 (2021)
37. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)
38. Lin, Z., Feng, M., dos Santos, C.N., Yu, M., Xiang, B., Zhou, B., Bengio, Y.: A structured self-attentive sentence embedding. CoRR **abs/1703.03130** (2017), http://arxiv.org/abs/1703.03130
39. Liu, P.J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., Shazeer, N.: Generating wikipedia by summarizing long sequences. arXiv preprint arXiv:1801.10198 (2018)
40. 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)
41. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. CoRR **abs/2103.14030** (2021), https://arxiv.org/abs/2103.14030
42. 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
43. 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
44. Mu, J., Viswanath, P.: All-but-the-top: Simple and effective postprocessing for word representations. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=HkuGJ3kCb
45. Nguyen, T.M., Suliafu, V., Osher, S.J., Chen, L., Wang, B.: Fmmformer: Efficient and flexible transformer via decomposed near-field and far-field attention. arXiv preprint arXiv:2108.02347 (2021)
46. Parikh, A., Täckström, O., Das, D., Uszkoreit, J.: A decomposable attention model for natural language inference. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. pp. 2249–2255. Association for Computational Linguistics, Austin, Texas (Nov 2016). https://doi.org/10.18653/v1/D16-1244, https://www.aclweb.org/anthology/D16-1244

47. Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., Tran, D.: Image transformer. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 4055–4064. PMLR (10–15 Jul 2018), http://proceedings.mlr.press/v80/parmar18a.html
48. Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N., Kong, L.: Random feature attention. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=QtTKTdVrFBB
49. Qiu, J., Ma, H., Levy, O., Yih, S.W.t., Wang, S., Tang, J.: Blockwise self-attention for long document understanding. arXiv preprint arXiv:1911.02972 (2019)
50. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training. OpenAI report (2018)
51. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
52. Rajpurkar, P., Zhang, J., Lopyrev, K., 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. Association for Computational Linguistics, Austin, Texas (Nov 2016). https://doi.org/10.18653/v1/D16-1264, https://www.aclweb.org/anthology/D16-1264
53. Roy, A., Saffar, M., Vaswani, A., Grangier, D.: Efficient content-based sparse attention with routing transformers. Transactions of the Association for Computational Linguistics **9**, 53–68 (2021), https://www.aclweb.org/anthology/2021.tacl-1.4
54. 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)
55. Sajjad, H., Dalvi, F., Durrani, N., Nakov, P.: Poor man's bert: Smaller and faster transformer models. arXiv e-prints pp. arXiv–2004 (2020)
56. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108 (2019)
57. Schlag, I., Irie, K., Schmidhuber, J.: Linear transformers are secretly fast weight programmers. In: International Conference on Machine Learning. pp. 9355–9366. PMLR (2021)
58. Shazeer, N.: Fast transformer decoding: One write-head is all you need. arXiv preprint arXiv:1911.02150 (2019)
59. Shen, Z., Zhang, M., Zhao, H., Yi, S., 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)
60. So, D.R., Liang, C., Le, Q.V.: The evolved transformer. arXiv preprint arXiv:1901.11117 (2019)
61. Sukhbaatar, S., Grave, E., Lample, G., Jegou, H., Joulin, A.: Augmenting self-attention with persistent memory. arXiv preprint arXiv:1907.01470 (2019)
62. Sun, S., Cheng, Y., Gan, Z., Liu, J.: Patient knowledge distillation for bert model compression. arXiv preprint arXiv:1908.09355 (2019)
63. Tay, Y., Bahri, D., Yang, L., Metzler, D., Juan, D.C.: Sparse Sinkhorn attention. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 9438–9447. PMLR (13–18 Jul 2020), http://proceedings.mlr.press/v119/tay20a.html
64. 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
65. 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)
66. 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
67. Tsai, Y.H.H., Bai, S., Yamada, M., Morency, L.P., Salakhutdinov, R.: Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. arXiv preprint arXiv:1908.11775 (2019)
68. 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)
69. Vig, J., 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. Association for Computational Linguistics, Florence, Italy (Aug 2019). https://doi.org/10.18653/v1/W19-4808, https://www.aclweb.org/anthology/W19-4808
70. 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
71. 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)
72. Wang, S., Li, B., Khabsa, M., Fang, H., Ma, H.: Linformer: Self-attention with linear complexity. arXiv preprint arXiv:2006.04768 (2020)
73. Williams, A., Nangia, N., 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 (Jun 2018). https://doi.org/10.18653/v1/N18-1101, https://www.aclweb.org/anthology/N18-1101
74. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., Le, Q.V.: Xlnet: Generalized autoregressive pretraining for language understanding. arXiv preprint arXiv:1906.08237 (2019)

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

In this supplementary material, we include experimental details, additional experiments/visualization of pruning via FiAK, and the detailed derivation of the computational saving in Table 1. 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.12 and 7, 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.

Table 5. Box mean average precision (box mAP) on the COCO validation set of the Swin Tranformer [41] pruned with mixed pruning via FiAK in Algorithm 2 compared to the baseline dense Swin transformer. The mixed pruned model with the pruning fraction $k_1 = 70\%$ and $k_2 = 15\%$ achieves a comparable result with the dense baseline while outperforming the attention-score pruned model that prunes the model less with a smaller pruning fraction $k = 60\%$. The box mAP of the pretrained baseline is reported at https://github.com/SwinTransformer/Swin-Transformer-Object-Detection.

Method	Box mAP
Baseline Swin transformer	43.7
FiAKformer	**44.7**
Attention-score pruned FiAKformer 60%	43.3
Mixed pruned FiAKformer $k_1 = 70\%, k_2 = 15\%$	43.6

B Additional Results on Mixed Pruning via FiAK

B.1 Object Detection on COCO

In this section, we provide additional object detection results for the pruned Swin Transformer [41] with mixed pruning via FiAK (see Algorithm 2). Table 5 shows that the mixed pruned model via FiAK with pruning fractions $k_1 = 70\%$ and $k_2 = 15\%$ has comparable performance with the baseline dense softmax model. It is interesting to notice that the mixed pruned model also outperforms the attention-score pruned model while allowing larger pruning fraction, 70% vs. 60%. Also, mixed pruned models have smaller computational complexity than attention-score pruned models as explained in Section 3 and Table 1.

B.2 Language Modeling on WikiText-103

For autoregressive tasks like language modeling, the importance scores $\hat{S}(j) = \sum_i |\hat{\pi}_{ij}|$ in Algorithm 2 of mixed pruning via FiAK are biased for different time steps j of the keys. In particular, the smaller j, the more $|\hat{\pi}_{ij}|$ will be added into the sum since $|\hat{\pi}_{ij}| = 0$ for $i < j$, $i, j = 1, \ldots, N$, i.e. a query can only attend to the current and previous keys, but not the future keys. Thus, $\hat{S}(j)$ in mixed pruning via FiAK need to be normalized when used in an autoregressive task. We will study the normalization of $\hat{S}(j)$ for autoregressive tasks in our future work.

C 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 4 depicts the pruning mask obtained with mixed pruning via FiAK for the ImageNet object classification task. Figure 5 visualizes the pruning masks obtained with attention score pruning via FiAK for the WikiText-103 language modeling task.

In Figure 6, 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

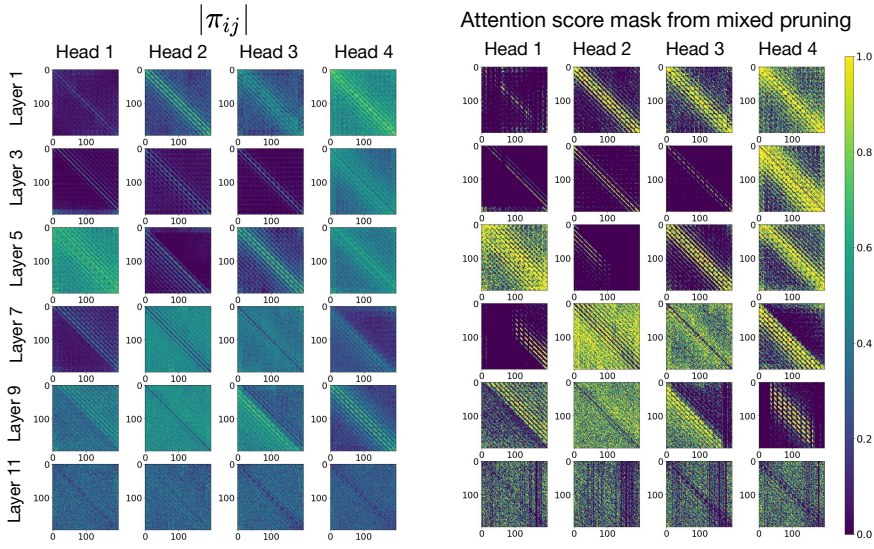

Fig. 4. (Left) The magnitude of the priors $|\pi_{ij}|$ learned from the ImageNet image classification task. (Right) The binary attention score masks from mixed pruning via FiAK with the pruning fraction $k_1 = 70\%$ and $k_2 = 15\%$. We plot these prior matrices and pruning masks for all 4 heads in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

grid corresponds to the pruning mask applied to each query $\boldsymbol{q}_i$. From Figure 6, 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.

D Pruning Masks Obtained via FiAK are Training-Agnostic

This section shows that the pruning masks learned via FiAK can still capture positions of important attention scores in the baseline model trained without additional prior parameters π_{ij} in Eqn. 12. This result suggests that the pruning masks learned via FiAK are training-agnostic. After learned, these masks can be applied on the baseline models trained with different training schemes.

We conduct our experiment on the ImageNet object classification task in Section 4.1. In particular, we mask the attention matrices in the pretrained DeiT-tiny provided in [66] by the binary pruning masks obtained from the attention score pruning via FiAK with the pruning fraction $k = 60\%$. The masked model is then fine-tuned till convergence. Table 6 shows that the masked DeiT-tiny transformer achieves better performance than the dense baseline, demonstrating that the pruning masks learned by FiAK capture relevant attention scores and meaningful connections between tokens at each heads in the baseline DeiT-tiny trained with the setting in [66].

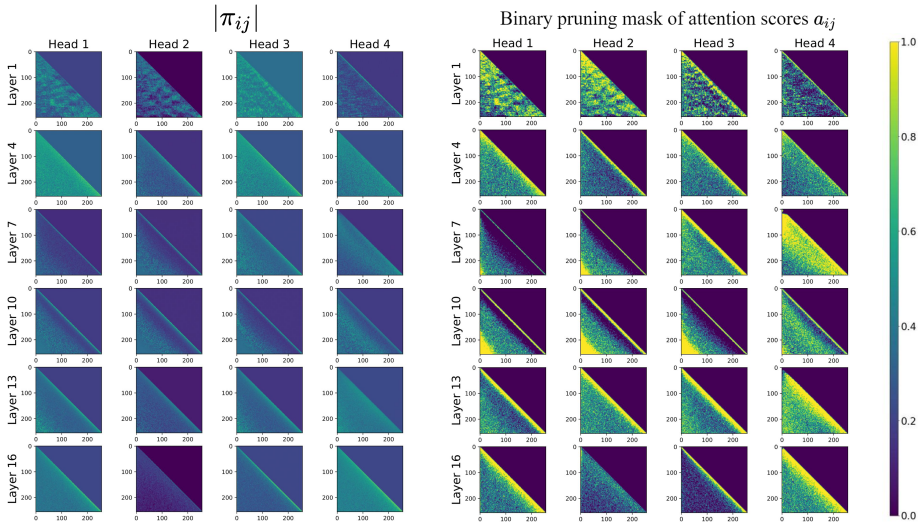

Fig. 5. (Left) Magnitude of the priors $|\pi_{ij}|$ learned from the language modeling task. (Right) The binary pruning masks from attention score pruning via FiAK with the pruning fraction $k = 50\%$. We plot the prior matrices and pruning masks for the first 4 heads in layers {1, 4, 7, 10, 13, 16}.

Table 6. Top-1 and top-5 accuracy (%) of the masked vs dense DeiT-tiny [66]. We mask the attention matrices of the trained baseline with the binary pruning masks learned from attention score pruning via FiAK with the pruning fraction $k = 60\%$ and then fine-tune the masked model. The masked DeiT-tiny achieves better performance than the dense baseline, indicating that the pruning masks successfully capture important attention scores at each head.

Method	Acc Top-1	Acc Top-5
Baseline softmax transformer	72.23	91.13
Masked softmax transformer $k = 60\%$	**72.41**	**91.30**

E 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. 7, as the sequence length grows, our pruned model becomes significantly more efficient in both memory and computations than the baseline.

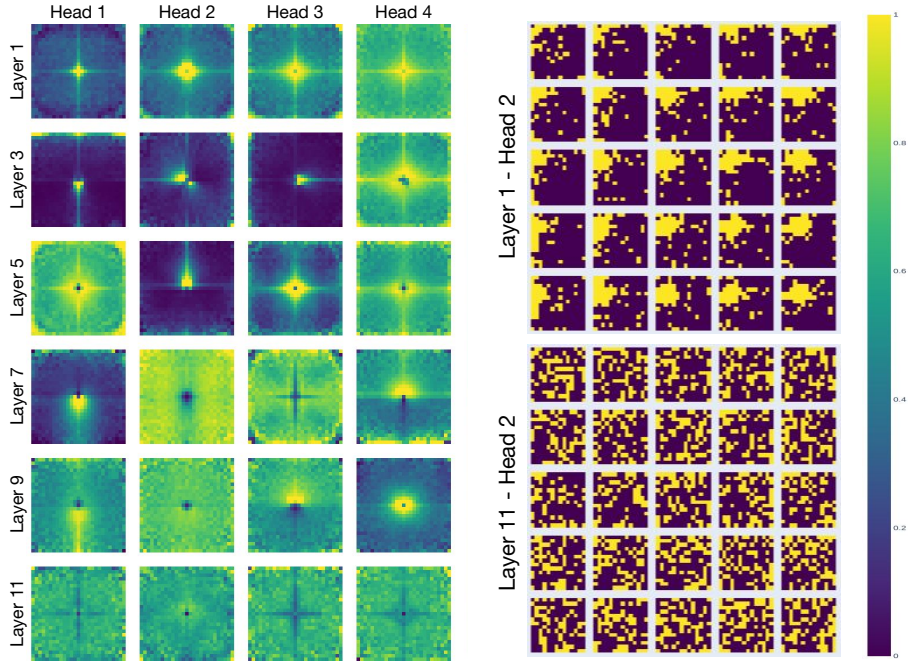

Fig. 6. (Left) Query-centered mean of the magnitude of the priors $|\pi_{ij}|$ for each head in layers $\{1, 3, 5, 7, 9, 11\}$, learned from the ImageNet task. (Right) The binary pruning masks of the attention scores for attention score pruning via FiAK with pruning fraction 70%, showing the differences 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.

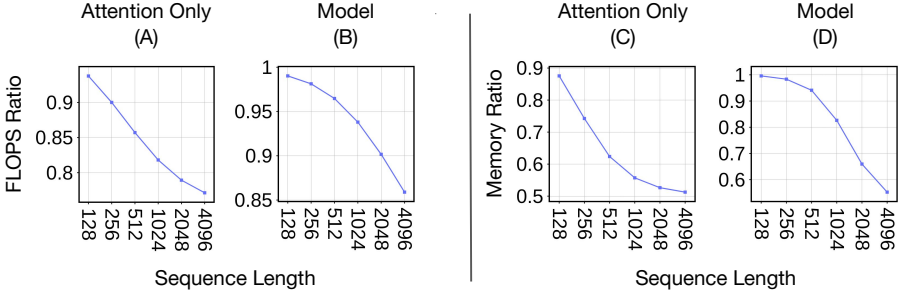

Fig. 7. FLOPS and memory ratios at inference time on the WikiText-103 language modeling task for model pruned using attention score pruning via FiAK with the pruning fraction $k = 50\%$, compared to the baseline dense softmax transformer model. For a thorough analysis, we show a comparison at attention 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.

F 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. 1.1).

- **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* $\mathbf{QK}$. 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 $\mathbf{QK}$. 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 **QK**. 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.