Diversifying the Expert Knowledge for Task-Agnostic Pruning in Sparse Mixture-of-Experts

Anonymous ACL submission

Abstract

In this work, we address the memory overhead of deploying Mixture-of-Experts (MoE) architectures in Large Language Models (LLMs). While MoE layers improve LLM performance without increasing inference costs, the ever-growing number of experts inflates memory requirements, hindering practical deployment. Our empirical study reveals that some experts encode redundant knowledge during pre-training. We thus propose a method of grouping and pruning similar experts to improve the model's parameter efficiency. We validate the effectiveness of our method by pruning three state-of-the-art MoE architectures, including Mixtral, Deepseek-MoE, and Qwen. The evaluation shows that our method outperforms other model pruning methods on a range of natural language tasks. We will release our code to facilitate future research.

1 Introduction

Large Language Models (LLMs) have achieved outstanding performance across various tasks by learning a large number of model parameters on large amounts of data, as shown by the scaling laws (Kaplan et al., 2020). In addition to increasing the depth of neural network models, widening neural networks by using the sparsely-activated mixture-of-experts (MoE) architecture is also proved effective. MoE widens the feed-forward network (FFN) layer (one expert) by having multiple parallel FFNs (experts). During forward propagation, only a subset of these experts is activated. Thus, compared to dense models, MoE models achieve better end-task performance and generalize better to new tasks without increasing computation costs. Notable examples of MoE models include Switch Transformer (Fedus et al., 2022), Mixtral-MoE (Jiang et al., 2024), and Uni-MoE (Li et al., 2024b).

Despite significant progress in developing wider and deeper MoE LLMs, the increased memory consumption due to larger model sizes (i.e., increased number of experts) poses a substantial challenge to the deployment of these models in real-world settings. For example, storing and loading Mixtral-8x7B, which has 8 experts in each of its 32 layers, requires approximately 88 GB. The MoE layers constitute the majority of the parameters. Adding or removing even one expert in each layer can significantly impact overall memory cost and model performance. For example, (Lu et al., 2024) shows that randomly dropping 2 experts in each MoE layer reduces the memory cost by 21 GB, and decreases model performance by 7% on the MMLU benchmark (Hendrycks et al., 2020). In this study, we strive to seek the best trade-off between memory efficiency and task performance by identifying an optimal set of experts in each MoE layer to prune.

There have been several studies on pruning MoE models. One line of work utilizes task-specific information to prune irrelevant experts. For example, Chen et al. (2022) prune the less frequently visited experts based on experiments on a range of tasks. Chowdhury et al. (2024) find that less important experts usually exhibit smaller changes in routing weights during the fine-tuning stage. Li et al. (2024a) merge experts that are frequently visited by tokens of a fine-tuned dataset for pruning. While effective, these methods depend on knowing the target tasks. In contrast, task-agnostic pruning methods that do not rely on task information are more appealing and useful in real-world applications because they can apply to both seen and unseen tasks. However, this is more challenging since there are no explicit task and data cues to guide which experts are redundant. He et al. (2024) explore pruning experts with less visited frequency in a task-agnostic calibration dataset but report a significant performance drop. In comparison, Lu et al. (2024) enumerate all the combinations of experts and prune some to achieve a minimum loss of

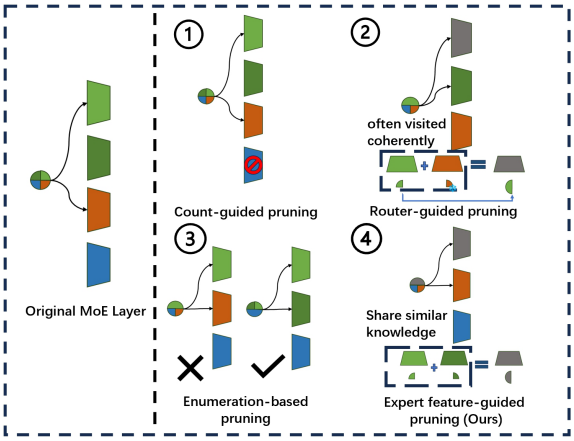

Figure 1: Removing several experts from the original MoE layer would not cause model collapse but improve efficiency. Experts with similar colors share the similar knowledge with each other. Prior works often utilize expert access information to filter out unimportant experts. In our work, we first group different experts with similar knowledge in the feature space, then merge them along with the routers to prune the MoE layer. This post-processing approach allows us to diversify the features of each MoE layer, thereby preserving the knowledge of the original large models as much as possible while reducing computation and storage consumption.

reconstruction, which significantly improves performance. We illustrate the difference among these works in fig. 1. Although pruning MoE models in task-agnostic settings is of great practical value, this area has not been fully explored.

In this work, we explore how to prune MoE models in a task-agnostic fashion. Our study is motivated by the finding that, given the same input, many experts respond similarly, indicating that these experts likely encode similar knowledge, and thus are somewhat redundant. We propose a method to improve model parameter efficiency by pruning redundant experts in two stages. As shown in fig. 3, we first identify and group similar experts in the feature space. Then, for each group, we merge experts in the weight space to diversify the knowledge in different MoE layers. We validate the effectiveness of our method by pruning experts for three state-of-the-art MoE architectures, including Mixtral (Mixtral-8x7B and Mixtral-8x22B), Deepseek-MoE (Deepseek-MoE-16B) and Qwen (Qwen2-57B-14A). The evaluation shows that our method outperforms other model pruning methods on a range of natural language tasks. Our contributions are summarized as follows,

1. We empirically validate that some experts within each well-trained MoE layer encode similar knowledge, making them somewhat redundant.

2. We propose a two-stage, task-agnostic method for grouping and merging redundant experts, which is further divided into data-centric and model-centric implementation strategies.

3. We demonstrate the effectiveness of our method by pruning experts from a series of state-of-the-art MoE models, including Mixtral-MoE, DeepSeek-MoE, and Qwen-MoE. The results from a greedy search for MoE pruning further validate the success of our approach.

2 Related Work

Sparse MoEs. The Mixture-of-Experts (MoE) structure is firstly applied in classical machine learning models by Jacobs et al. (1991) and Jordan and Jacobs (1994), then widely used in various deep learning models (Yuksel et al., 2012; Masoudnia and Ebrahimpour, 2014; Zhang et al., 2023). Recently, some works employ the MoE to scale the capacity of transformer-based models, especially the large language models (Shazeer et al., 2017; Lepikhin et al., 2020; Zoph et al., 2022). It adapts the original large feed-forward network (FFN) in each transformer block into multiple smaller FFNs, forming an expert layer with a router that computes the weighted output of each MoE layer. Sparse MoEs were first proposed by Fedus et al. (2022). In this approach, only a few experts are activated in each layer, accelerating training and inference while significantly increasing the number of parameters for greater model capacity. Many sparse MoEs have been developed and open-sourced within the AI community, such as Switch Transformer (Fedus et al., 2022), Mixtral-8B (Jiang et al., 2024), and Uni-MoE (Li et al., 2024b). Recent studies also indicate that neural networks with the MoE structure exhibit better generalization ability compared to dense models (Zhu et al., 2022; Li et al., 2022).

Model Pruning. Model pruning involves removing unimportant parameters from a well-trained neural network to balance task performance and computational efficiency (Liu et al., 2018; Wang et al., 2020; Liu et al., 2021). Pruning techniques can be categorized into unstructured pruning (Liao et al., 2023; Shi et al., 2024; Mason-Williams and Dahlqvist,

2024), which introduces sparsity in the weight matrix by setting some parameters to zero, and structured pruning (Lemaire et al., 2019; Fang et al., 2023; Shen et al., 2022), which removes entire neurons, layers, or blocks, reducing redundancy and being more suitable for acceleration on GPUs (Choi and Yang, 2021). Many efforts have been made to leverage model pruning techniques to reduce the memory consumption of neural networks, spanning a range of models from conventional architectures like CNNs (Luo et al., 2018), RNNs (Zhu and Gupta, 2017), and LSTMs (Ding et al., 2020) to modern large models such as Llama (Xia et al., 2023) and Stable-Diffusion (Castells et al., 2024).

While the large amount of parameters in sparse MoEs benefits the model's capacity to achieve good performance at the pre-training stage, the increasing memory consumption causes great challenges to fine-tuning different downstream tasks. In this paper, we work on pruning the sparse MoEs to reduce redundant experts in the task-agnostic setting, which enhances the computational and memory efficiency throughout the fine-tuning process, and scalability of deploying these models.

3 Method

In this section, we present our approach for pruning experts within MoE layers in large language models. Our goal is to identify experts that share highly similar knowledge and merge them, thereby reducing the model size and improving efficiency without significantly degrading performance.

3.1 Notation and Preliminaries

We denote $F(\cdot;\Theta,W,K)$ as an MoE layer. Here, $\Theta=\{\theta_1,\theta_2,...,\theta_N\}$ is the set of parameters for N experts $\{f_n(\cdot;\theta_n)\}_{n=1}^{N}$, where each expert is typically a feed-forward network (FFN). The routing matrix $W\in\mathbb{R}^{N\times d}$ determines which experts are selected for each token. K denotes how many top experts are chosen for each input token.

For a token $x\in\mathbb{R}^d$, we first compute the routing logits to measure how well it matches each expert: $p_n(x)=\frac{e^{W_n x}}{\sum_{t=1}^{N}e^{W_t x}}$. Then, we select top-$K$ experts, $\{f_n(\cdot;\theta_n)\}_{n=i_1}^{i_K}$, based on these logits. Finally, we compute the MoE layer's output as the weighted combination of the chosen experts: $y=\sum_{n=i_1}^{i_K}p_n(x)\cdot f_n(x;\theta_n)$.

3.2 Task Definition

Previous works (Lu et al., 2024; He et al., 2024; Li et al., 2024a) have shown that pruning some experts in MoE layers can improve model efficiency in both inference and fine-tuning without causing model collapse. However, an open question remains: which experts should be removed?

Formally, consider a LLM $\mathcal{M}(\cdot;\mathcal{F})$ comprsing L MoE layers. Each MoE Layer $F^l(\cdot;\Theta^l,W^l,K)$ has N experts. Our objetive is to only remain r experts from each MoE layer:

$$\begin{aligned}\hat{\Theta}^l&=\Theta^l\setminus\{\theta^l_{s^l_1},\theta^l_{s^l_2},\dots,\theta^l_{s^l_{N-r}}\}\\ \hat{W}^l&=W^l\setminus\{W^l_{s^l_1},W^l_{s^l_2},\dots,W^l_{s^l_{N-r}}\}\end{aligned}\tag{1}$$

We aim to find these indices $s^l=\{s^l_1,s^l_2,\dots,s^l_{N-r}\}$ that minimizes loss on a generic dataset D:

$$\min_{(x,y)\sim\mathcal{D}}\mathcal{L}(\hat{\mathcal{M}}(x;\hat{\mathcal{F}}),y).\tag{2}$$

The search space $(C_N^r)^L$ is enormous, making direct combinatorial search intractable.

3.3 Measuring Expert Similarity with CKA

Our key insight is that experts exhibiting similar behaviors likely contain redundant knowledge. By identifying and pruning these redundant experts, we can simplify model and improve its efficiency.

To quantify similarity, we use Centered Kernel Alignment (CKA) (Kornblith et al., 2019) as the criteria to evaluate the similarity between experts in each MoE layer. Intuitively, CKA measures how similarly two experts transform a shared batch of inputs. For any two experts f_i and f_j, given a batch of inputs $x=\{x_1,x_2,...,x_s\}$, the similarity ρ_{ij} is computed as follows,

$$\rho_{ij}=\mathrm{CKA}(K^i,K^j)=\frac{\mathrm{HSIC}(K^i,K^j)}{\mathrm{HSIC}(K^i,K^i)\cdot\mathrm{HSIC}(K^j,K^j)},\tag{3}$$

where K^i and K^j are kernel matrices constructed from experts' outputs on the same input batch x, and HSIC is the Hilbert-Schmidt Independence Criterion (Greenfeld and Shalit, 2020). The definition is as follows: $\mathrm{HSIC}(K^i,K^j)=\frac{1}{(s-1)^2}\mathrm{tr}(K^iHK^jH)$, $K^i_{mn}=k(f_i(x_m),f_i(x_n))$, $H=I-\frac{1}{s}11^T$, and $k(\cdot,\cdot)$ is the kernel function. Notably, all experts are provided with the same input batch—no token distribution by the router function is involved—allowing for a clearer representation of the distinct knowledge each expert acquires within the same layer.

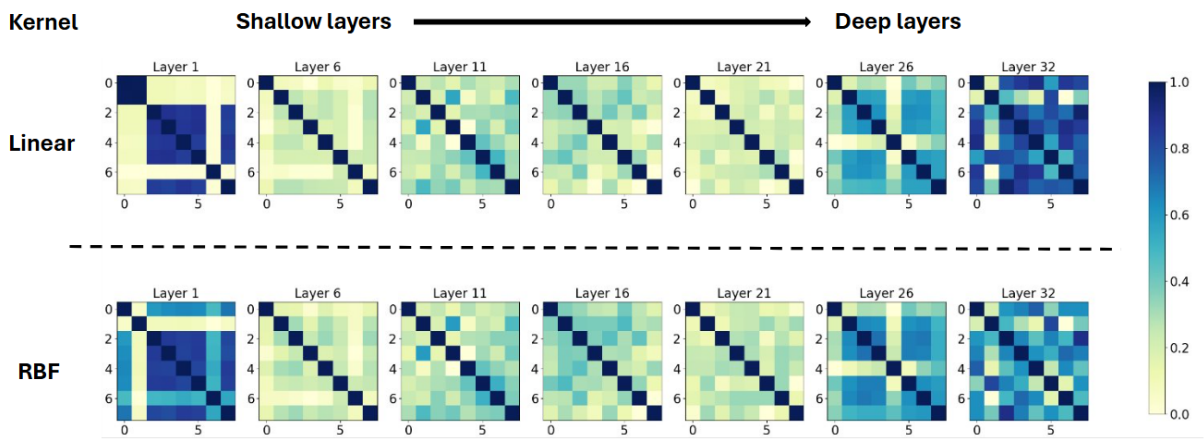

Figure 2: Evaluation of the expert similarity for different MoE layers in Mixtral-8x7B under two kernel-based CKA criteria (Linear and RBF). A darker color indicates a greater similarity between experts.

We evaluate the similarity between experts in Mixtral-8Bx7B using 32 randomly selected samples from the C4 pre-training dataset (Raffel et al., 2020). Following eq. (3), we compute expert similarity with both a linear kernel (equivalent to using a dot product) and an RBF kernel.

As shown in fig. 2, darker cells indicate greater similarity between pairs of experts. Both kernel measures reveal similar patterns across layers, indicating that some experts are moderately to highly similar. This suggests the feasibility of pruning certain MoE layers. For example, in the first layer, experts 2 through 5 consistently show similarity scores above 0.7, suggesting that removing one of these experts would likely have minimal impact on overall model performance. Full evaluation results can be found in appendix B.

3.4 Discovering and Merging Similar Experts

As shown in fig. 3, our pruning strategy consists of two key steps to reduce the number of experts from N to r: (1) identifying groups of similar experts and (2) merging each group into a single expert.

Discovering Similar Experts from the Expert Graph. We construct a unidirectional graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{A}\}$ to cluster similar experts, where $\mathcal{V}$ is the set of nodes (each representing an expert), $\mathcal{E}$ denotes the set of edges connecting experts that exhibit positive similarity, and $\mathcal{A}$ is a weight matrix that encodes these pairwise similarities. The procedures for constructing $\mathcal{A}$ is shown as follows.

First, we represent each expert f_i by its output embedding on a shared calibration dataset (data-centric) or by its own weights (model-centric) if data is unavailable, as $\mathcal{R}(f_i)$.

Second, we calculate the weight on each edge of $\mathcal{E}$ to represent the similarity between experts, *i.e.*, $\mathcal{A}(\mathcal{E})$, using CKA (data-centric) as eq. (3) or cosine similarity (model-centric). Each element $\mathcal{A}(\mathcal{E}_{ij}) = \text{CKA}(\mathcal{R}(f_i), \mathcal{R}(f_j))$ reflects how similar experts f_i and f_j are in the transformed space $\mathcal{R}(f)$ as the previous step.

Last, we split $\mathcal{G}$ into r subgraphs $\{\mathcal{G}_i\}_{i=1}^{r}$ to group similar experts. The experts indexed by $\mathcal{V}_i$ of $\mathcal{G}_i$ share the most similar knowledge with each other and show much difference with experts indexed by $\mathcal{V}_t$ of $\mathcal{G}_t$, $t \neq i$. This can be formulated as the follows,

$$\max \sum_{i=1}^{r} \left(\sum_{j,k \in \mathcal{V}_i} \mathcal{A}(\mathcal{E}_{jk}) - \sum_{t \neq i}^{r} \sum_{j \in \mathcal{V}_i, k \in \mathcal{V}_t} \mathcal{A}(\mathcal{E}_{jk}) \right),$$
$$s.t. \quad \bigcup_{i=1}^{r} \mathcal{V}_i = \mathcal{V}, \quad \mathcal{V}_i \cap \mathcal{V}_t = \emptyset \quad \text{for} \quad i \neq t. \tag{4}$$

Our objective is to minimize the intra-group similarity (first term) and maximize the inter-group difference (second term) based on the pairwise similarity $\mathcal{A}(\mathcal{E}_{ij})$ based on expert i and j.

Merging Similar Experts To diversify the experts and preserve the different knowledge learned by different models clustered in the same group (Wortsman et al., 2022; Wang et al., 2022; Zhao et al., 2024), we merge the clustered experts with their routers on the weight space as follows,

$$\hat{\theta}_n \leftarrow \sum_{i=1}^{|\mathcal{V}_n|} \alpha_i \theta_{\mathcal{V}_n(i)}, \quad \hat{W}_n \leftarrow \sum_{i=1}^{|\mathcal{V}_n|} \alpha_i W_{\mathcal{V}_n(i)}, \tag{5}$$

where $\mathcal{V}_n$ is the set of similar experts in the n-th cluster $\mathcal{G}_n$, and we have $\sum_{i=1}^{|\mathcal{V}_n|} \alpha_i = 1$. We update the MoE layer F by respectively replacing all the

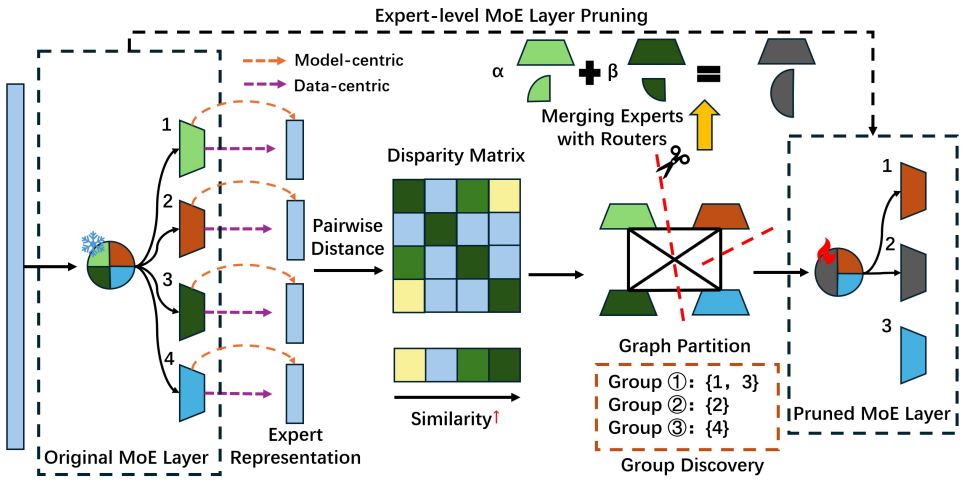

Figure 3: We first leverage model or data-centric strategies to obtain the expert representation, then compute the pairwise distance to get the disparity matrix. Based on the expert similarity matrix, we can group similar experts with shared knowledge in the same cluster, which can be merged on the weight space for pruning.

experts grouped in $\mathcal{G}_n$ by a single FFN expert layer $f(\cdot;\hat{\theta}_n)$ and corresponding routing weights $\hat{W}_n$, $n = 1, 2, ..., N - r$. The pseudo-code is provided in appendix G.

3.5 Practical Considerations

Data-Centric vs. Model-Centric Strategies. We propose two strategies for computing the disparity matrix $\mathcal{A}(\mathcal{E})$: a data-centric strategy and a model-centric strategy.

For the data-centric strategy, since the full pre-training datasets for different models are large and inaccessible, we use the C4 dataset, a commonly used smaller pre-training subset that serves as an effective surrogate for capturing task-agnostic knowledge of experts (Lu et al., 2024). Specifically, by disabling the router function, for the same input, we use the output of each expert as its expert representation $\mathcal{R}(\cdot)$. Then, we use the linear kernel-based CKA metric introduced in eq. (3) to compute the expert similarity and discover the similar experts by eq. (4). To enhance generalization and mitigate overfitting to the selected samples during model pruning, we apply data augmentation by randomly mixing token embeddings during the representation computation at the discovery stage.

For model-centric strategy, we propose two ways to prune models with only expert weights, which encode the dataset information during the training process and can be a good agency for expert representation. One is to leverage the vectorized weights directly (♣). The other is to leverage the local linearity* of neural networks (Zhang and Wu, 2020) to compute the surrogate weight matrix (♠).

Taking the FNN $f(\cdot;\theta)$ in Mixtral-8x7B as an example, it consists of three linear layers, *i.e.*, $\theta = \{\theta_1, \theta_2, \theta_3\}$. For input x, the output is computed as $f(x;\theta) = \theta_2(\sigma(\theta_1 x) \cdot \theta_3 x)$. Then, the vectorized weight (♣) is concat$\{\theta_1, \theta_2, \theta_3\}$, while the surrogate weights can be obtained by $\theta_2(\theta_1 \cdot \theta_3)$ (♠). Compared to vectorized weights, *surrogate weights are more flexible with model size, showing more stable performance* (see appendix D).

Due to the lack of data for computing the CKA for distance evaluation, we replace the K_i and K_j of eq. (3) with the weights of i-th and j-th experts, respectively. This substitution enables us to use the cosine similarity between model weights to estimate the similarity between experts.

Merging Strategies and Trade-offs. We consider three approaches in practice deciding α to merge similar grouped experts. The first is to only maintain the experts with the maximum visiting frequency and drop all the others. The second is to set $\alpha_i = \frac{1}{|\mathcal{V}_n|}$, which uniformly assembles all the experts grouped. The last one is learning α to merge the grouped experts by minimizing the following loss function,

$$\mathcal{L}(\{\alpha_n\}_{n=1}^{|\mathcal{V}_n|}) = \|y - F(x; \hat{\Theta}, \hat{W}, K)\|,$$
$$s.t.\ \hat{\Theta}(n) = \lambda(\sum_{i=1}^{|\mathcal{V}_n|} \alpha_i \theta_{\mathcal{V}_n(i)}), \hat{W}_n = \sum_{i=1}^{|\mathcal{V}_n|} \alpha_i W_{\mathcal{V}_n(i)}, \tag{6}$$

*In short, the 'local linearity' property of neural networks refers to the observation that, despite having multiple hidden layers and non-linear activation functions, neural networks exhibit linear behavior within a local region. This property motivates our design of a single-layer approximation using the surrogate weight.

where the ground-truth is the output of original MoE layer $y = F(x; \Theta, W, K)$, and we jointly optimize λ and α for different merging groups. Among these three strategies, both the first and third require the presence of data, while the second is compatible with both data-centric and model-centric approaches. We empirically find that the uniform souping strategy offers more stable performance and greater efficiency in task-agnostic model pruning (see appendix D). While the learning strategy can yield slightly better performance, it is more time-consuming due to the need for tuning the parameter α.

4 Evaluation

4.1 Experiment Setup

Studied Models. We take pruning three MoE-based architectures as an example, including the Mixtral, Deepseek, and Qwen. For the Mixtral architecture, the Mixtral-8x7B has 32 sparse MoE-involved layers, in each there are 8 experts. The Mixtral-8x22B is similar to Mixtral-8x7B but with 56 sparse MoE layers. During the inference, each token will select 2 experts in each MoE layer. The deepseek model has 28 layers, and there are 64 experts in each layer. Each token will pass 2 shared experts and select 6 experts during the inference. The Qwen model also has 28 MoE layers with 64 experts in each layer but will activate 8 experts during the inference. We include more experimental details in appendix A.

Pruning Methods. We take three advanced MoE pruning methods (He et al., 2024; Li et al., 2024a; Lu et al., 2024) as our baseline for comparison. Among them, *router-guided* merging (Li et al., 2024a) is initially designed for task-specific MoE pruning, where we set the target dataset as the samples from the pre-training dataset. For task-agnostic methods, we select the frequency-based pruning method Expert Trimming (Lu et al., 2024), also the *count-guided* strategy and loss-based pruning method (Lu et al., 2024), namely the *Enumerate* in our paper. Under the task-agnostic pruning setting, we disable all the fine-tuning stage of all methods for fair comparison.

For our method, we respectively report the best results of *our data-centric* and *model-centric* methods in pruning models. Following the previous setting (Lu et al., 2024), we use 128 samples in C4 for computation in data-centric pruning methods, while the model-centric method doesn't rely on the data. More detailed results on different implementations of discovery and merging steps (*e.g.*, vectorized and surrogate weight strategy at the discovery step, max or learning strategy at the merging step) are provided in appendix D.

Evaluation Datasets. The open-sourced Language Model Evaluation Harness library (Gao et al., 2021) is used to evaluate the performance. We select four tasks, including MMLU (Hendrycks et al., 2020), BoolQ (Clark et al., 2019), OpenBookQA (Mihaylov et al., 2018), and RTE (Bentivogli et al., 2009). Among these tasks, MMLU is the most challenging one, which consists of 57 subtasks, where we present four groups, namely the humanities, social science, stem and other.

4.2 Pruning the Mixtral Architecture

We present the results of the pruning of the Mixtral architecture, including Mixtral-8x7B and Mixtral-8x22B. Both have 8 experts in each MoE layer. We apply different methods to prune them from 8 experts to 6 and 4 experts in each layer, respectively. The results of the two models are shown in table 1 and table 2 respectively.

Results on Mixtral-8x7B. We can see that all our proposed four strategies surpass the related works, with a clear margin performance improvement of 1.5% on average. Compared with count-guided strategy (He et al., 2024) which just drops the experts less visited, router-guided strategy (Li et al., 2024a) has a large improvement of 3.7% on average by merging these experts, showing that *the merging operation plays a crucial role in preserving the expert knowledge*. Besides, compared with count-guided and enumerate strategies (Lu et al., 2024) which all adopt the dropping strategy, we can see that *directly leveraging the expert feedback rather than the routing frequency is more suitable for task-agnostic pruning in MoE layers*. We can also notice that the model-centric method surpasses all the other data-involved pruning baseline methods. This suggests that *weights already encode fruitful data information and can be deployed to group experts for pruning*.

Results on Mixtral-8x22B. In this experiment, the model-centric approach achieves the best result, with only a minor performance drop of 2.8% on average compared to the full model. Our proposed data-centric method ranks second to last, with a performance gap of 0.8% compared to the runner-up method. This suggests that model-centric approaches exhibit better robustness when pruning

Table 1: Results on pruning the Mixtral-8x7B from 8 experts to 6 and 4 experts in each MoE layer. The first and second columns respectively indicate the results of the pruned model with 6 and 4 experts.

Dataset / Method		MMLU humanities	MMLU social science	MMLU stem	MMLU Other	BoolQ	OpenBookQA	RTE	Average
Mixtral-8x7B (Jiang et al., 2024)		60.5	77.8	58.9	74.2	85.4	34.4	71.1	66.0
Router-guided (Li et al., 2024a)		51.8/24.8	60.5/26.5	46.9/24.7	60.5/25.0	82.6/39.9	32.0/11.6	70.4/50.9	57.8/29.1
Count-guided (He et al., 2024)		49.2/36.9	59.7/45.6	45.0/35.1	58.2/43.4	77.2/76.6	**33.0**/26.4	56.6/55.9	54.1/45.7
Enumerate (Lu et al., 2024)		52.4/43.5	66.4/52.7	49.0/40.4	63.7/43.5	84.0/80.8	32.6/28.8	**71.1**/66.4	59.9/50.8
Ours	**Model-centric**	54.4/**48.1**	70.2/**58.5**	51.8/**45.2**	66.8/**55.2**	85.6/83.7	31.4/26.2	68.9/62.4	61.3/54.2
	Data-centric	56.0/48.0	**73.1**/57.0	**52.4**/43.3	**68.2**/54.6	**86.4**/83.3	31.4/28.5	**69.3/67.1**	**62.4/54.5**

Table 2: Results on pruning the Mixtral-8x22B from 8 experts to 6 and 4 experts in each MoE layer. The first and second columns respectively indicate the results of the pruned model with 6 and 4 experts.

Dataset / Method		MMLU humanities	MMLU social science	MMLU stem	MMLU Other	BoolQ	OpenBookQA	RTE	Average
Mixtral-8x22B (Jiang et al., 2024)		68.6	84.1	67.1	78.7	87.9	0.358	71.2	70.4
Router-guided (Li et al., 2024a)		27.3/22.7	25.4/25.8	24.4/24.0	27.9/23.4	62.8/62.7	12.8/13.0	54.2/49.5	33.5/31.6
Count-guided (He et al., 2024)		58.0/45.7	74.9/57.7	54.1/42.0	70.2/45.7	81.5/74.4	35.2/27.0	69.3/57.4	63.3/50.0
Enumerate (Lu et al., 2024)		60.4/53.9	78.0/67.2	59.5/52.3	73.0/64.2	87.4/80.5	35.0/31.1	70.1/67.9	66.2/59.6
Ours	**Model-centric**	**63.7/58.1**	**80.0/72.5**	**62.1/54.3**	**75.6/68.3**	88.0/**85.2**	34.6/31.2	69.0/**68.6**	**67.6/62.6**
	Data-centric	62.3/57.8	78.5/69.7	60.2/51.3	73.4/64.2	87.6/83.1	**35.8/33.2**	**71.1**/68.1	67.0/61.1

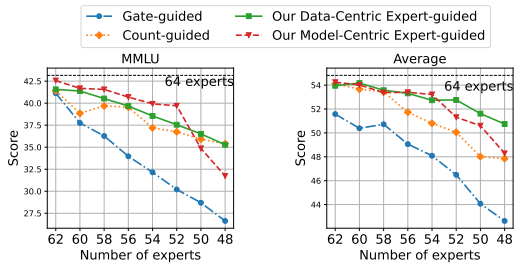

Figure 4: Results on pruning the Deepseek-MoE-16B with different strategies.

models of different scales, while data-centric methods are more prone to overfitting on small calibration datasets (as evidenced by the collapse of the route-guided method in table 2).

4.3 Pruning the Deepseek Model

We also evaluate the effectiveness of our proposed pruning method on compressing the DeepSeek architecture. Unlike Mixtral-MoE, DeepSeek-MoE features a shared expert and incorporates more fine-grained experts at each MoE layer. Specifically, we pruned the *non-shared experts* of DeepSeek-MoE-16B, reducing the number of experts from 64 to 48 using various model pruning strategies. The results are illustrated in fig. 4[†].

Notably, even after pruning one-third of the experts in Deepseek-MoE-16B, our data-centric strategy maintains an impressive average performance of 50.9%, with only a 3.1% reduction in performance compared to the full model. In the evaluation of the most challenging MMLU task, our model-centric strategy demonstrates superior performance in most cases, particularly when reducing the number of experts from 62 to 52. It consistently outperforms the runner-up baseline method, achieving a clear performance advantage of 2.6%.

4.4 Pruning the Qwen Model

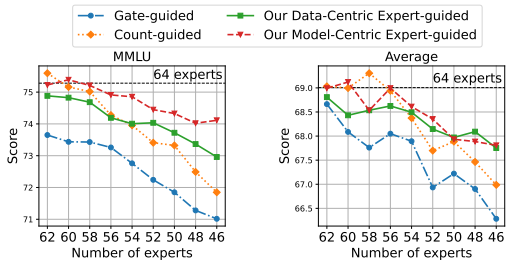

Figure 5: Results on pruning the Qwen2-57B-14A with different strategies.

The Qwen architecture is also utilized in our experiments. We study the Qwen2-57B-14A to evaluate the performance of different model pruning strategies. We prune the experts from 64 experts to 48 experts in each layer and evaluate the performance on different tasks.

Reported in fig. 5, when pruning $\frac{1}{3}$ experts of the full model, the data-centric strategy delivers the best performance, outperforming both our model-centric method and the count-guided base-

[†]We only show the pruning results of the Deepseek and Qwen models on the most challenging MMLU task and the average performance across MMLU, BoolQ, OpenBookQA, and RTE. Full results can be found in supplementary.

Table 3: Evaluation results when ranging the number of samples for expert similarity estimation. The augmentation is disabled in this experiment.

# of samples	MMLU	BoolQ	OpenBookQA	RTE	Avg.
128	61.5	85.6	32.8	68.6	62.1
256	61.4	85.4	33.2	69.7	62.4
512	61.7	85.4	33.4	70.4	62.7

line method. On the MMLU task, our model-centric strategy achieves approximately 1.5% better performance compared to the runner-up one.

4.5 Ablation Study and Discussion

Comparison with the greedy search. To further demonstrate the superiority of our proposed expert pruning strategy, we employ a greedy search approach to identify the optimal candidates in each MoE layer for pruning and compare the results with those obtained using our method. Specifically, we enumerate all possible combinations of removing two experts in each MoE layer, evaluate the pruned models on the MMLU task, and record the best combination for each layer. We then merge these results to prune the model across all layers. The model pruned using the greedy search approach achieves a performance of 62.22%, while our data-centric strategy achieves 61.19%, showing a comparable result. We provide the full enumeration results in appendix C.

On the used samples. We conduct experiments to study the effect of both sample size used for the calibration dataset and augmentation in data-centric pruning methods. As shown in table 3, increasing the number of samples from 128 to 512 *without augmentation* leads to a noticeable performance improvement of up to 0.6%.In contrast, when comparing this result with table 1, where 128 samples were used with augmentation, we observe an average performance improvement of 0.3%. This demonstrates the importance of both sample size and augmentation in enhancing generalization during the pruning process, as discussed in section 3.5.

On merging the routing policy.The motivation of our work is to diversify expert knowledge by merging similar experts. A key distinction of our approach is the merging of routing policies, which potentially directs more tokens to the resulting merged expert. To highlight the importance of merging routers, we evaluate the performance of our pruning method without merging the routing policies. As shown in table 4, compared with the results in table 1 and table 2,this results in performance degradation across all tasks and different numbers of experts, underscoring the crucial role of simultaneously merging both routers and experts.

Table 4: Evaluation results of our data-centric pruning method without merging the routers.

# of experts	MMLU	BoolQ	OpenBookQA	RTE	Avg.
4	49.1	83.3	27.4	66.4	56.5
6	60.4	84.7	31.2	67.5	60.9

Table 5: Computation cost of evaluating Mixtral-8x7B and DeepSeek-MoE-16B on the MMLU task. Experiment settings are consistent with our paper.

Model	# Experts	Time (s)	Mem (GB)
Mixtral-8x7B	8 (Ori.)	281	125
	6	241	104
	4	223	83
DeepSeek-MoE-16B	64 (Ori.)	457	64
	60	446	61
	56	429	59
	52	413	57
	48	386	55

Efficiency improvement after MoE pruning To demonstrate the benefits of MoE pruning, we provide a statistical efficiency analysis for Mixtral-8x7B and DeepSeek-MoE-16B. We set the batch size to 8 and evaluate the performance on the MMLU task using the lm_eval library (Gao et al., 2021). Other settings are consistent with the previous experiment. As shown in table 5, pruning two experts from each MoE layer in the Mixtral model achieves a 1.17× speedup and a 16.8% reduction in GPU memory usage. Pruning 16 experts from each MoE layer in the Deepseek model reduces inference time by 15.5% and GPU memory usage by 14.1%. These results highlight the computational benefits of expert pruning across different model scales. More analysis can be found in appendix F.

5 Conclusion and Future Work

In this paper, we work on the task-agnostic pruning of sparse MoEs. We propose discovering similar experts at the feature level and then merging them in the weight space for MoE pruning while preserving as much original expert knowledge as possible. This approach allows the MoE layer to maintain diverse experts with different knowledge, thereby efficiently reducing redundancy.

However, we didn't account for the uniqueness of different MoE layers during the pruning process, often setting the same number of experts to be reduced across all layers. Our analysis reveals that some experts have low visitation frequencies, indicating a significant potential for further pruning.

6 Limitations

Several unexplored questions remain in our project. First, we designed various strategies to prune the MoE, and we observed that different models require different strategies to achieve optimal post-pruning performance. It remains unclear what causes these performance differences across strategies. Second, while the learning strategy at the merging step can bring slightly performance improvement, the cost is also large. The question of how to efficiently find the optimal merging coefficients remains. Third, in our work, we prune the same experts across different MoE layers, despite each layer having varying levels of redundancy. A key question remains: how can we push MoE compression to its limits while maintaining acceptable performance?

References

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. *TAC*, 7(8):1.

Thibault Castells, Hyoung-Kyu Song, Bo-Kyeong Kim, and Shinkook Choi. 2024. Ld-pruner: Efficient pruning of latent diffusion models using task-agnostic insights. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 821–830.

Tianyu Chen, Shaohan Huang, Yuan Xie, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. 2022. Task-specific expert pruning for sparse mixture-of-experts. *arXiv preprint arXiv:2206.00277*.

Kyusik Choi and Hoeseok Yang. 2021. A gpu architecture aware fine-grain pruning technique for deep neural networks. In *European Conference on Parallel Processing*, pages 217–231. Springer.

Mohammed Nowaz Rabbani Chowdhury, Meng Wang, Kaoutar El Maghraoui, Naigang Wang, Pin-Yu Chen, and Christopher Carothers. 2024. A provably effective method for pruning experts in fine-tuned sparse mixture-of-experts. *arXiv preprint arXiv:2405.16646*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936.

Guiguang Ding, Shuo Zhang, Zizhou Jia, Jing Zhong, and Jungong Han. 2020. Where to prune: Using lstm to guide data-dependent soft pruning. *IEEE Transactions on Image Processing*, 30:293–304.

Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. 2023. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16091–16101.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. 2021. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*, 10:8–9.

Daniel Greenfeld and Uri Shalit. 2020. Robust learning with the hilbert-schmidt independence criterion. In *International Conference on Machine Learning*, pages 3759–3768. PMLR.

Shwai He, Daize Dong, Liang Ding, and Ang Li. 2024. Demystifying the compression of mixture-of-experts through a unified framework. *arXiv preprint arXiv:2406.02500*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

R Jacobs, MI Jordan, GE Hinton, et al. 1991. Mixtures of expert networks. *Neural Comput*, 3:79–87.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Michael I Jordan and Robert A Jacobs. 1994. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR.

Carl Lemaire, Andrew Achkar, and Pierre-Marc Jodoin. 2019. Structured pruning of neural networks with budget-aware regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9108–9116.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Bo Li, Yifei Shen, Jingkang Yang, Yezhen Wang, Jiawei Ren, Tong Che, Jun Zhang, and Ziwei Liu. 2022. Sparse mixture-of-experts are domain generalizable learners. *arXiv preprint arXiv:2206.04046*.

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. 2024a. Merge, then compress: Demystify efficient smoe with hints from its routing policy. In *The Twelfth International Conference on Learning Representations*.

Yunxin Li, Shenyuan Jiang, Baotian Hu, Longyue Wang, Wanqi Zhong, Wenhan Luo, Lin Ma, and Min Zhang. 2024b. Uni-moe: Scaling unified multimodal llms with mixture of experts. *arXiv preprint arXiv:2405.11273*.

Zhu Liao, Victor Quétu, Van-Tam Nguyen, and Enzo Tartaglione. 2023. Can unstructured pruning reduce the depth in deep neural networks? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1402–1406.

Jing Liu, Bohan Zhuang, Zhuangwei Zhuang, Yong Guo, Junzhou Huang, Jinhui Zhu, and Mingkui Tan. 2021. Discrimination-aware network pruning for deep model compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4035–4051.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2018. Rethinking the value of network pruning. In *International Conference on Learning Representations*.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. 2024. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. *arXiv preprint arXiv:2402.14800*.

Jian-Hao Luo, Hao Zhang, Hong-Yu Zhou, Chen-Wei Xie, Jianxin Wu, and Weiyao Lin. 2018. Thinet: Pruning cnn filters for a thinner net. *IEEE transactions on pattern analysis and machine intelligence*, 41(10):2525–2538.

Gabryel Mason-Williams and Fredrik Dahlqvist. 2024. What makes a good prune? maximal unstructured pruning for maximal cosine similarity. In *The Twelfth International Conference on Learning Representations*.

Saeed Masoudnia and Reza Ebrahimpour. 2014. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42:275–293.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

Andrew Ng, Michael Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems 14*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Maying Shen, Pavlo Molchanov, Hongxu Yin, and Jose M Alvarez. 2022. When to prune? a policy towards early structural pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12247–12256.

Xinyu Shi, Jianhao Ding, Zecheng Hao, and Zhaofei Yu. 2024. Towards energy efficient spiking neural networks: An unstructured pruning framework. In *The Twelfth International Conference on Learning Representations*.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan, and Jianfeng Gao. 2022. Adamix: Mixture-of-adaptations for parameter-efficient model tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5744–5760.

Yulong Wang, Xiaolu Zhang, Lingxi Xie, Jun Zhou, Hang Su, Bo Zhang, and Xiaolin Hu. 2020. Pruning from scratch. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 12273–12280.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*.

Seniha Esen Yuksel, Joseph N Wilson, and Paul D Gader. 2012. Twenty years of mixture of experts. *IEEE*

transactions on neural networks and learning systems, 23(8):1177–1193.

Xiao Zhang and Dongrui Wu. 2020. Empirical studies on the properties of linear regions in deep neural networks. *arXiv preprint arXiv:2001.01072*.

Yihua Zhang, Ruisi Cai, Tianlong Chen, Guanhua Zhang, Huan Zhang, Pin-Yu Chen, Shiyu Chang, Zhangyang Wang, and Sijia Liu. 2023. Robust mixture-of-expert training for convolutional neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 90–101.

Xinyu Zhao, Guoheng Sun, Ruisi Cai, Yukun Zhou, Pingzhi Li, Peihao Wang, Bowen Tan, Yexiao He, Li Chen, Yi Liang, Beidi Chen, Binhang Yuan, Hongyi Wang, Ang Li, Zhangyang Wang, and Tianlong Chen. 2024. Model-glue: Democratized llm scaling for a large model zoo in the wild. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Jinguo Zhu, Xizhou Zhu, Wenhai Wang, Xiaohua Wang, Hongsheng Li, Xiaogang Wang, and Jifeng Dai. 2022. Uni-perceiver-moe: Learning sparse generalist models with conditional moes. *Advances in Neural Information Processing Systems*, 35:2664–2678.

Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*.

A Experiment details

We conducted our experiments using four 80GB NVIDIA A100 GPUs.

In our learning strategy for expert merging, we used SGD as the optimizer to learn the coefficients for expert merging, with the learning rate set to 1×10^{-3}. We randomly sample 128 samples from the calibration set (C4) and partition them into a $3:1$ ratio as the training and evaluation sets, respectively. The coefficients were initialized as identity matrices and optimized over 50 epochs until they converged.

For baseline methods and models, all usage and distribution comply with the terms of their license, *i.e.*, Mixtral (Apache License 2.0), Deepseek (MIT license), Qwen (Tongyi Qianwen license), lm evaluation harness (MIT license). We use Copilot to help with debugging and coding.

B Evaluation on the expert similarity

Following the same setting in section 3.3, we conduct experiments on evaluating the expert similarity in all MoE layers of Mixtral-8x7B and Mixtral-8x22B. The results are respectively depicted in fig. D1 and fig. D2. We summarize the observation as follows,

- Most MoE layers in the two Mixtral models contain significant expert redundancy.
- The most redundant MoE layers are located in the first and last several layers, while the experts in the intermediate MoE layers learn more diverse features.

C Enumeration on the expert pruning

We present the full greedy search result on Mixtral-8x7B in fig. D3. In detail, we first enumerate all the possible combinations of dropping 2 experts layer by layer, and then evaluate the model on the MMLU task. For example, 66.19 in the first row and third column in layer 0 indicates performance while dropping the first and third expert in layer 0 of the Mixtral-8x7B model.

Although dropping most of the combinations on different layers only leads to a minor performance drop, we can notice that it could cause the model to crash when the fourth expert in layer 1 is involved during the pruning process.

Table B1: Results on pruning the Mixtral-8x7B and Mixtral-8x22B from 8 experts to 6 and 4 experts in each MoE layer. We present the results of our four strategies, namely 1) Vectorized and surrogate θ: prunable experts discovery using vectorized or surrogate weights and merging with uniform coefficients; 2) Learn: prunable experts discovery using vanilla data and learn coefficients to merge based on eq. (6); 3) Max: maintaining the expert in each discovered group with the maximum visiting frequency. The first and second columns respectively indicate the results on pruned model with 6 and 4 experts.

Dataset		MMLU				BoolQ	OpenBookQA	RTE	Average
Method		humanities	social science	stem	Other				
Mixtral-8x7B		60.5	77.8	58.9	74.2	85.4	34.4	71.1	66.0
Model-centric	**Vectorized**	54.4/**48.1**	70.2/**58.5**	51.8/**45.2**	66.8/**55.2**	85.6/83.7	31.4/26.2	68.9/62.4	61.3/54.2
	Surrogate	56.8/47.1	69.7/56.4	50.9/42.2	66.0/55.0	**86.9/83.8**	32.6/27.0	68.5/64.6	61.6/53.7
Data-centric	**Learn**	56.0/48.0	**73.1**/57.0	**52.4**/43.3	**68.2**/54.6	86.4/83.3	31.4/28.5	**69.3/67.1**	62.4/**54.5**
	Max	56.4/47.6	71.9/58.6	52.0/42.9	66.9/55.7	85.1/82.8	**35.2/28.8**	70.4/66.8	**62.6**/53.7
	Uniform	56.2/47.8	72.7/57.2	52.1/42.7	68.1/54.0	85.6/83.2	32.8/28.6	68.6/66.5	62.3/54.3
Mixtral-8x22B		68.6	84.1	67.1	78.7	87.9	0.358	71.2	70.4
Model-centric	**Vectorized**	26.7/24.2	28.5/21.7	26.9/21.3	31.7/23.8	62.0/53.6	19.6/11.4	52.0/53.1	35.3/29.9
	Surrogate	**63.7/58.1**	**80.0/72.5**	**62.1/54.3**	**75.6/68.3**	88.0/**85.2**	34.6/31.2	69.0/**68.6**	**67.6/62.6**
Data-centric	**Learn**	61.4/57.9	78.3/70.1	61.2/51.4	72.8/65.0	**88.2**/84.9	35.6/32.7	70.5/67.3	66.9/61.3
	Max	56.8/47.1	69.7/56.4	50.9/42.2	66.0/55.0	86.9/83.8	32.6/27.0	68.5/64.6	61.6/53.7
	Uniform	62.3/57.8	78.5/69.7	60.2/51.3	73.4/64.2	87.6/83.1	**35.8/33.2**	71.1/68.1	67.0/61.1

D Results on different strategies for pruning

While we report the best performance using our data-centric and model-centric strategies to prune the MoE models, we detail more results on pruning the Mixtral-8x7B and Mixtral-8x22B.

1. For model-centric stratgies, we show the results with **vectorized** weight and **surrogate** weight strategies to discover the similar experts. We uniformly merge the experts and their routers for model pruning. In other words, *the model-centric strategies differ at the discovery stage*.

2. For data-centric strategy, after we discover similar experts, we respectively use the learning strategy to merge the experts with weights (**Learn**), only maintain the expert with the maximum visiting frequency (**Max**), or uniformly merging different experts (**Uniform**). Thus, in this experiment, *the data-centric strategies differ at the merging stage*.

From the results of our strategies, we can observe the following:

1. *Both the Vectorized and Surrogate strategies surpass all other data-involved pruning baseline methods.* This suggests that *weights already encode valuable data information, which can be utilized to group experts for pruning*.

2. *When pruning relatively small models (Mixtral-8x7B), the inclusion of data in the pruning process improves candidate selection for merging and pruning, leading to better performance compared to using only model weights.* While the learning strategy offers a slight improvement in average performance, it comes at a higher computational cost compared to using uniform coefficients for merging.

3. *For larger models (Mixtral-8x22B), the model-centric method outperforms the data-centric method.* We argue that this is due to overfitting to the small calibration dataset during pruning, especially when dealing with a large number of parameters. The small calibration dataset cannot approximate the distribution of the pre-training dataset effectively. This is evident from the significant performance drop observed when using the Max strategy for pruning Mixtral-8x22B.

E Full results on pruning Deepseek and Qwen

We present the full results of pruning Deepseek-MoE-16 and Qwen2-57B-14A in fig. E4 and fig. E5. It is evident that our proposed data-centric and model-centric strategies outperform all baseline methods in most test cases. Additionally, when examining different evaluation tasks, the results on MMLU show a more reliable and consistent trend

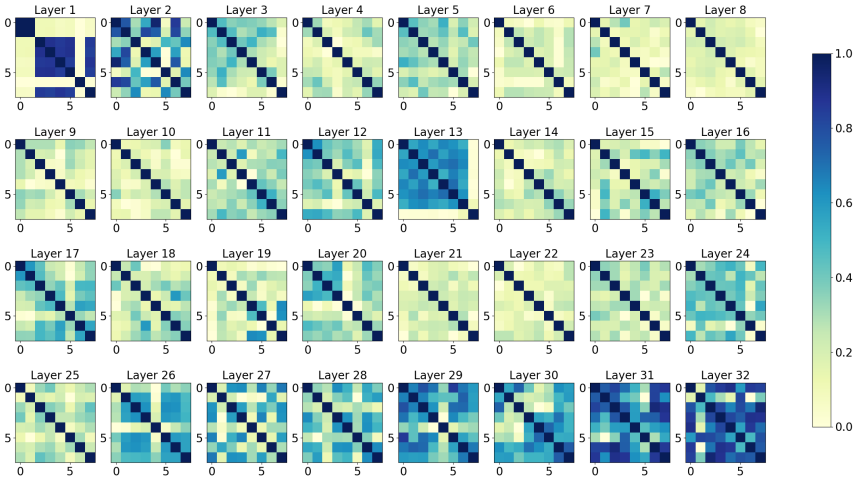

Figure D1: Evaluation of the expert similarity for different MoE layers in Mixtral-8x7B under the linear kernel-based CKA criteria.

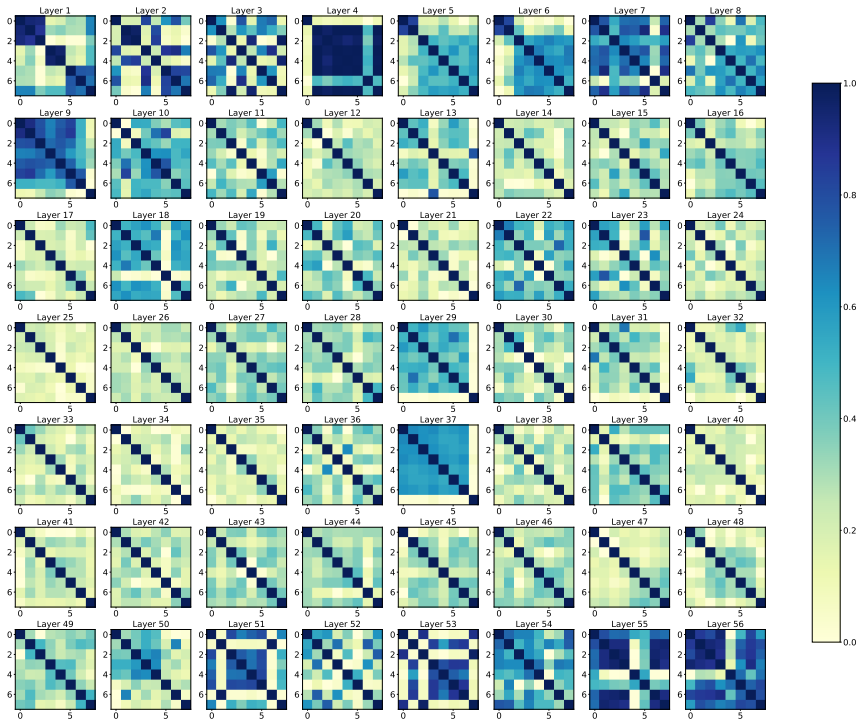

Figure D2: Evaluation of the expert similarity for different MoE layers in Mixtral-8x22B under the linear kernel-based CKA criteria.

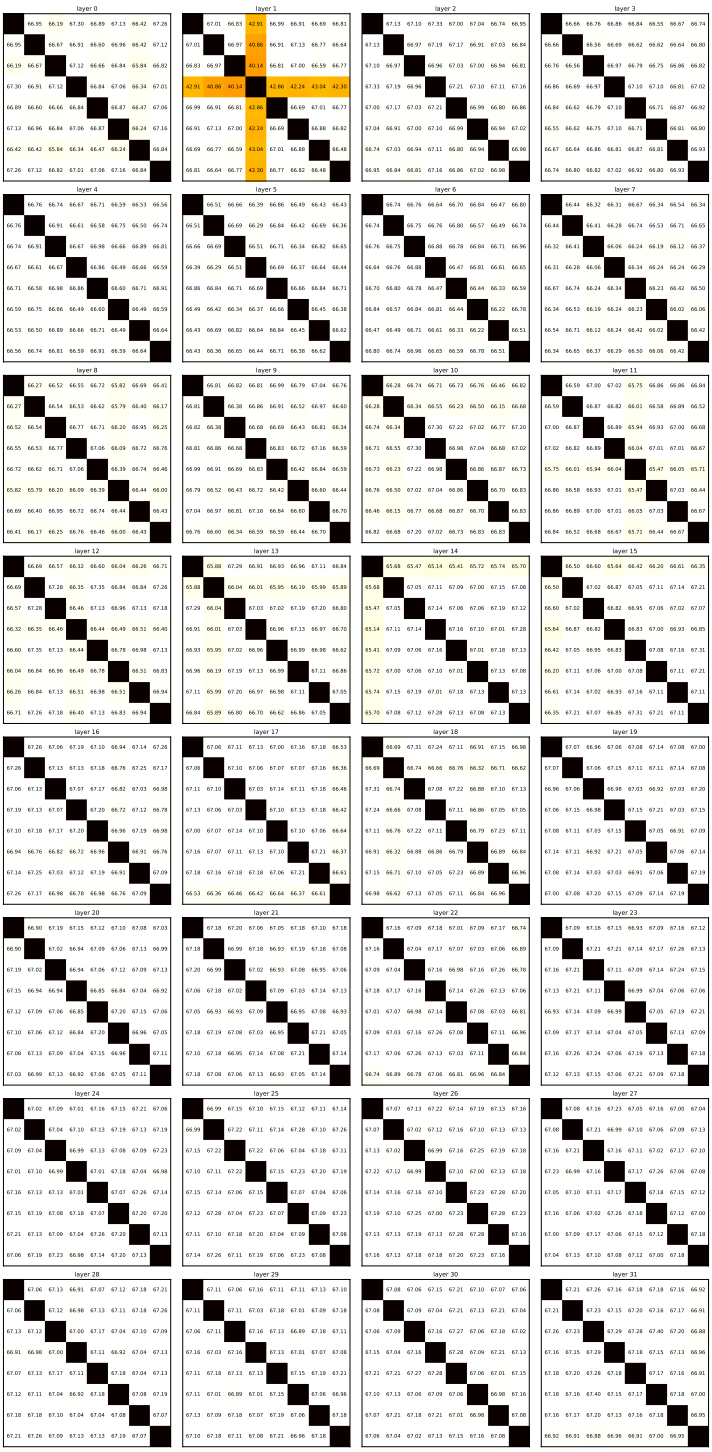

Figure D3: Enumeration on dropping two experts.

as the number of pruned experts increases. However, for smaller tasks such as the RTE dataset, we observe some randomness in the evaluation results due to the limited dataset size.We did not include the enumeration-based method in our comparison, as it is time-consuming and difficult to complete within a limited timeframe, especially when the number of experts is large in these models.

F Empirical Analysis on Expert Hints

We analyze the change of expert hints on the calibration dataset C4, where the expert hint refers to the visiting frequency of the expert on the calibration set.

The statistic results are shown in fig. F6. We can see that in most MoE layers in Mixtral-8x7B, many pairs of experts have similar hints. In contrast, pruning differentiates the hints of experts. Compared to directly using hints as the pruning goal, using expert knowledge as the pruning criterion results in more significant changes in hints. Additionally, the merging operation on the gate aggregates the new expert (last column) more tokens, further increasing the hint differences.

G Pseudo-code implementation

It is an NP-hard problem to solve eq. (4), where we can leverage the spectral clustering method (Ng et al., 2001) to get the approximate solution.

Algorithm 1 Pruning algorithm for given MoE layer

Input: The set of experts $\mathcal{F} = \{f_1, f_2, ..., f_N\}$ in the given MoE layer to prune, each expert f_i consisting of three layers parameterized by $\theta_1^i, \theta_2^i, \theta_3^i$, respectively, the number of experts to reserve r, the calibration dataset $\mathcal{D}$, strategy s.

Output: Pruned MoE layer with reduced experts $\{f_1, f_2, ..., f_r\}$.

function MoE PRUNING(MoE layer $\mathcal{F}$, number of experts r to reserve)
    Initialize $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{A}\}$.
    for each i in N **do**
        if s==Data-centric **then**
            $\mathcal{R}(f_i) = f_i(\text{Mixup}(\mathcal{D}))$.
        else if s==Vectorized weights **then**
            $\mathcal{R}(f_i) = \{\theta_1^i, \theta_2^i, \theta_3^i\}$
        else if s==Surrogate weights **then**
            $\mathcal{R}(f_i) = \theta_2^i(\theta_1^i \cdot \theta_3^i)$
    for each i in N **do**
        for each j in N **do**
            $\mathcal{A}(\mathcal{E}_{ij}) = \text{CKA}(\mathcal{R}(f_i), \mathcal{R}(f_j))$
    Optimize eq. (4) on $\mathcal{G}$ to find r subgroup of experts.
    Merge experts with their routers clustered within the same subgroup based on eq. (5).
    return Pruned $\mathcal{F}$

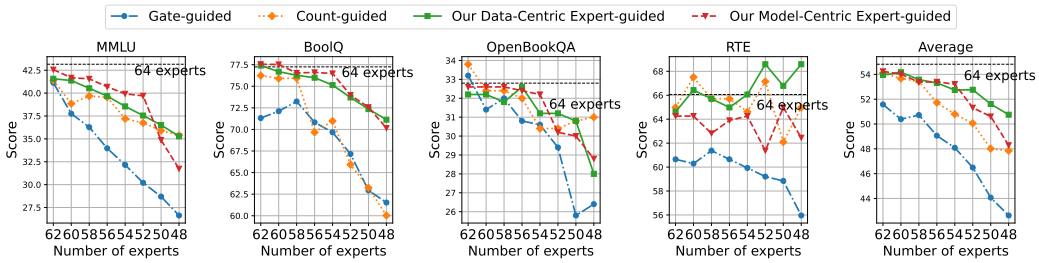

Figure E4: Results on pruning Deepseek-MoE-16B.

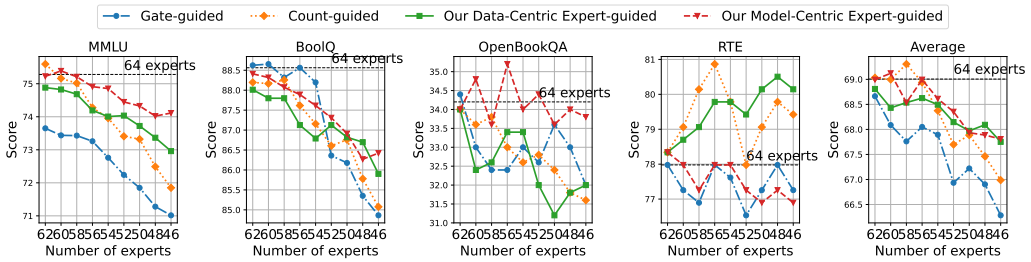

Figure E5: Results on pruning Qwen2-57B-14A.

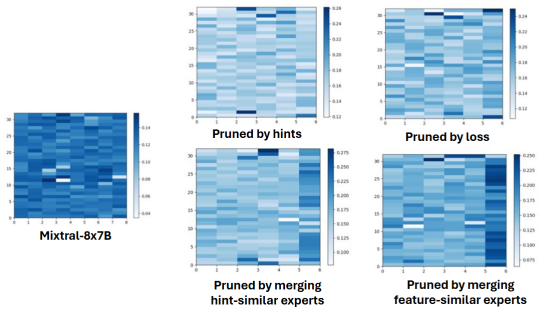

Figure F6: Statistics of the visiting frequency for all experts in different MoE layers.