MoE-Pruner: Pruning Mixture-of-Experts Large Language Model Using the Hints from Its Router

Anonymous ACL submission

Abstract

Mixture-of-Experts (MoE) architectures face challenges such as high memory consumption and redundancy in experts. Pruning MoE can reduce network weights while maintaining model performance. Motivated by the recent observation of emergent large magnitude features in Large Language Models (LLM) and MoE routing policy, we propose MoE-Pruner, a method that prunes weights with the smallest magnitudes multiplied by the corresponding input activations and router weights. Our pruning method is one-shot, requiring no retraining or weight updates. Furthermore, our pruned MoE models can benefit from a pre-trained teacher model through expert-wise knowledge distillation, improving performance post-pruning. We evaluate our method on various MoE models, such as Mixtral and DeepSeek, across multiple zero-shot evaluation benchmarks. Experimental results demonstrate that our pruning method significantly outperforms state-of-the-art LLM pruning methods. The pruned model with 50% sparsity maintains 99% of the performance of the original model after the expert-wise knowledge distillation.

1 Introduction

Mixture-of-Experts (MoE) architectures (Jacobs et al., 1991; Shazeer et al., 2017) have been proposed to reduce the computing cost while enabling efficient scaling of network capacity. It has been successfully employed to scale both vision (Ruiz et al., 2021; Shen et al., 2023) and language (Lepikhin et al., 2021; Fedus et al., 2022) models. In addition, MoE provides other advantages, such as sparsity that can mitigate catastrophic forgetting in continual learning (Collier et al., 2020; Komatsuzaki et al., 2023). Overall, MoE has proven to be a promising strategy for scaling deep learning models across various domains.

However, several crucial limitations persist in MoE for expanding its capacity. First of all, the static parameters, particularly those required for constructing the MoE architecture, introduce substantial memory overheads and constraints for deployment. For example, Mixtral-8x7B (Jiang et al., 2024) expert layers account for 96% of model parameters (45B out of 47B), which demands considerable memory and storage during inference. Moreover, MoE has a poor utilization of its experts. The conventional learning-based routing policy for MoE suffers from representation collapse issues since it encourages token embeddings to be clustered around expert centroids (Chi et al., 2022) and results in redundant experts (Mittal et al., 2022; Chen et al., 2022).

One possible solution to address those drawbacks and fully unleash the power of MoE is consolidating information from insignificant experts, aiming to establish a more compact MoE without hurting performance. Another solution is pruning experts that yield the lowest token reconstruction loss. Nevertheless, naively combining existing model merging mechanisms or expert pruning leads to performance degradation in the MoE architectures (Lu et al., 2024). We raise the following pivotal question for MoE LLM pruning: *How can we formulate and devise comprehensive pruning metrics tailored for MoE Large Language Models without degrading model performance?*

In this paper, we systematically explore MoE LLM pruning and target a high-quality compressed MoE model in downstream fine-tuning scenarios. Specifically, we first analyze the open-source MoE model's expert activation frequency and observe that different MoE expert initialization methods result in different expert activation frequencies and expert similarities. We leverage existing LLM pruning methods such as SparseGPT (Frantar and Alistarh, 2023) and Wanda (Sun et al., 2024), and design a novel pruning metric that incorporates MoE router weights information to identify and remove unimportant weights in expert layers. Since the pruning process is one-shot and only requires a small set of calibration data, the MoE model suffers from performance degradation. To recover MoE model performance, we further propose an expert-wise knowledge distillation method that utilizes the pre-trained model as a teacher model, facilitating the recovery of the pruned model's performance.

Our main contributions can be summarized as follows:

- We propose MoE-Pruner which is efficient and effective for pruning MoE models with minimal performance degradation.
- We design an innovative expert-wise knowledge distillation method that leverages the pre-trained MoE model as a teacher model to recover pruned MoE student model performance.
- Experimental results on various MoE models, such as Mixtral and DeepSeek, across nine zero-shot evaluation benchmarks demonstrate the effectiveness of our MoE-Pruner algorithm. MoE-Pruner achieves minimal performance drop even at 50% sparsity using only a small set of calibration data, outperforming existing pruning methods. Furthermore, the pruned model maintains 99% of the performance of the original model after the expert-wise knowledge distillation.

2 Preliminaries

Mixture-of-Experts (MoE). Scaling model size increases learning capacity and enhances generalization (Kaplan et al., 2020; Brown et al., 2020; Hoffmann et al., 2022). MoE (Jacobs et al., 1991; Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022) is an efficient approach that enables significantly more compute-efficient pretraining and inference. It replaces the feed-forward network (FFN) layers in Transformers (Vaswani et al., 2017) with mixture-of-expert layers, where different experts are activated for different input tokens instead of utilizing the full network parameters. Sparse MoE architecture can dramatically scale the model size with the same compute budget as a dense model.

MoE Architecture. A router or a gating network is trained to select a subset of experts for each input token based on its routing policy. Given n experts in a layer, the output of the expert layer is given by:

$$y = \sum_{k=1}^{n} Gate(x)_k \cdot E_k(x), \tag{1}$$

where the $Gate(x)_k$ is the router weights from the gating network assigned to the k-th expert, and $E_k(x)$ is the output of k-th expert. The router weights can be formulated as softmax over the Top-K logits:

$$Gate(x) = \text{Softmax}(\text{TopK}(x \cdot W_g)), \tag{2}$$

where W_g is the weight of the router or gating network, and $\text{TopK}(X)_k = l_k$ if k is in the top-K coordinates of logits l and $\text{TopK}(X)_k = -\infty$ otherwise.

Since current LLMs mostly adopt SwiGLU (Shazeer, 2020) architecture for the FFN, and MoE LLM such as Mixtral-8x7B (Jiang et al., 2024) uses a top-2 to select experts, we can derive the output of an expert layer as:

$$y = \sum_{k=1}^{n} \text{Softmax}(\text{Top2}(x \cdot W_g))_k \cdot \text{SwiGLU}_k(x). \tag{3}$$

Some recent MoE LLMs, such as DeepSeek-MoE (Dai et al., 2024), adopt shared experts that are always activated, aiming at capturing and consolidating common knowledge across varying contexts.

MoE Expert Initialization. MoE expert initialization uses different strategies, which can be classified into two categories: **sparse upcycling** (Komatsuzaki et al., 2023) and **training from scratch**. The sparse upcycling method starts from a dense model checkpoint and copies all parameters, except the MoE router, which does not exist in the original dense model. In particular, each expert in the new MoE layer is an identical copy of the original MLP layer that is replaced. Some open-source MoE models such as Mixtral (Jiang et al., 2024), Qwen1.5-MoE-A2.7B (Team, 2024), and MiniCPM-MoE (Hu et al., 2024) all employ the sparse upcycling approach to reduce the total training costs. While some MoE models like OLMoE (Muennighoff et al., 2024), DeepSeek-V2 (Liu et al., 2024a), and DeepSeek-V3 (DeepSeek-AI et al., 2024) are trained from scratch to help expert diversification.

Large Language Model Pruning. Magnitude pruning (Han et al., 2016) is a standard approach to induce sparsity in neural networks. It removes individual weights with magnitudes below a certain threshold. However, magnitude pruning fails dramatically on LLMs even with relatively low levels of sparsity (Frantar and Alistarh, 2023). SparseGPT (Frantar and Alistarh, 2023) proposes a one-shot, post-training pruning method that prunes LLM weights and uses Hessian matrix and calibration data to update the remaining weights without any retraining. Wanda (Sun et al., 2024) is a simple method that prunes LLM weights with the smallest magnitudes multiplied by the corresponding input activations without any additional weight update.

Pruning for MoE Models. Most of the works for MoE pruning focus on structured expert pruning (Yang et al., 2024b; Lee et al., 2024). Chen et al. (2022) and Koishekenov et al. (2023) prune experts based on their utilization to save memory. However, this usually leads to degraded performance. Lu et al. (2024) enumerates expert combinations based on the required expert number and uses calibration data to find a set of remaining experts that has the minimum reconstruction loss, but this method cannot scale to MoE LLMs with 32 or more experts. Chowdhury et al. (2024) prunes experts based on the change in the router's norm and proves that the generalization accuracy can be preserved. However, expert pruning sometimes removes experts with certain knowledge and results in the loss of model performance. Therefore, Li et al. (2024), Zhang et al. (2024b), Liu et al. (2024b) all leverage expert merging techniques to compress the expert layer while also preserving expert knowledge.

3 Methodology

3.1 Pruning Metric

MoE Expert Activation Frequency. We first use a subset of the C4 (Raffel et al., 2020) dataset and collect the activation frequency of MoE experts. Motivated by the load balancing loss (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022), we propose to use the coefficient of variation of expert activation frequency in each layer to represent the load balancing score, where a lower score represents more balanced loads. Given n experts and l layers and a batch $\mathcal{B}$ with T tokens, the load balancing score for one layer is:

$$
\begin{aligned}
s &= \frac{\sigma}{\mu} = \frac{\sqrt{\frac{1}{n}\sum_{k=1}^{n}(f_k-\mu)^2}}{\mu}, \\
\mu &= \frac{1}{n}\sum_{i=1}^{n} f_k,
\end{aligned} \tag{4}
$$

where f_k is the number of tokens dispatched to k-th expert:

$$
f_k = \sum_{x\in\mathcal{B}} \mathbb{1}\{\operatorname{argmax} p(x) = k\}. \tag{5}
$$

We can derive the load balancing score by calculating the mean of scores across all l MoE layers, such that we can use this score to compare with various MoE models with different numbers of experts.

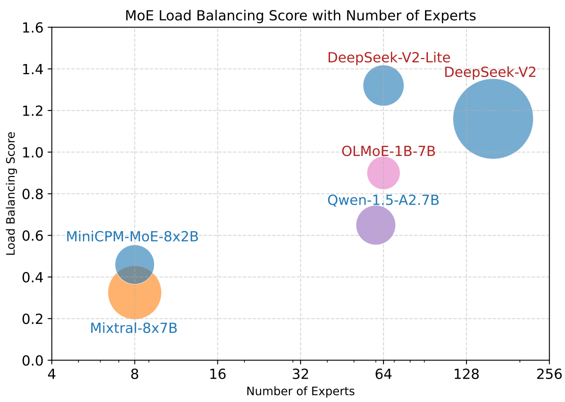


Figure 1: Load balancing score of MoE models. We collect the expert activation frequency of MoE models and calculate the load balancing score (lower is better). The circle area represents the model size. MoE models trained from scratch are marked with red, while MoE models that use upcycling are marked with blue. MoE models trained from scratch usually have more experts and imbalanced loads. MoE models initialized with sparse upcycling tend to have more balanced loads and less number of experts. The only exception is Qwen-1.5-A2.7B, which is initialized with upcycling. But according to the report (Yang et al., 2024a), its expert parameters are shuffled along the intermediate dimension to guarantee that each fine-grained expert exhibits unique characteristics and therefore exhibits more like trained from scratch MoE models.

Figure 1 shows the load balancing scores of Mixtral-8x7B (Jiang et al., 2024), Qwen-1.5-A2.7B (Team, 2024), DeepSeek-V2 and DeepSeeek-V2-Lite (Liu et al., 2024a), MiniCPM-MoE-8x2B (Hu et al., 2024), and OLMoE (Muennighoff et al., 2024). We find that different MoE expert initialization methods result in different expert activation frequencies and expert similarities, which will impact the MoE pruning strategies. For instance, the MoE model initialized with upcycling can take advantage of the dense model and reduce training costs. The final MoE model exhibits higher expert similarity and more balanced expert activation frequency. MoE model trained from scratch might yield better performance as it avoids the limitations of starting with a group of identical experts, which can hinder diversification (Wei et al., 2024).

Problem Formulation. Post-training pruning for LLMs can be decomposed into layer-wise subproblems (Lu et al., 2022; Frantar and Alistarh, 2023). Given a sparsity ratio and a linear layer with weight $\mathbf{W}$, the pruning algorithm tries to find a sparsity mask $\mathbf{M}$ that minimizes reconstruction loss:

$$\underset{\mathbf{M}}{\text{argmin}} \|\mathbf{W}\mathbf{X} - (\mathbf{M} \odot \mathbf{W})\mathbf{X}\|. \tag{6}$$

Optimal Brain Damage (OBD) (LeCun et al., 1989) first sets up a pioneering framework for neural network pruning. It uses second-order information without off-diagonal elements in the Hessian matrix for faster approximation. Optimal Brain Surgeon (OBS) (Hassibi et al., 1993) develops upon OBD partly by taking into account the off-diagonal elements. SparseGPT (Frantar and Alistarh, 2023) revisits the OBS, computes the inverse Hessian only once, and reuses to update weight in the remaining rows that are also in the mask to mitigate reconstruction loss. The pruning metric $\mathcal{S}_{ij}$ in SparseGPT is:

$$\mathcal{S}_{ij} = [|\mathbf{W}|^2/\text{diag}(\mathbf{H}^{-1})]_{ij}, \tag{7}$$

where $\mathbf{H}$ is the Hessian matrix, i and j stands for output feature and input feature dimension, respectively.

Wanda (Sun et al., 2024) further simplifies the pruning metric to the following form without the need to compute the inverse of the Hessian matrix:

$$\begin{aligned} \mathcal{S}_{ij} &= [|\mathbf{W}|^2/\text{diag}((\mathbf{X}^T\mathbf{X})^{-1})]_{ij} \\ &\approx [|\mathbf{W}|^2/(\text{diag}(\mathbf{X}^T\mathbf{X})^{-1})]_{ij} \\ &= (|\mathbf{W}_{ij}| \cdot \|\mathbf{X}_j\|_2)^2, \end{aligned} \tag{8}$$

where $\mathbf{X}$ is the corresponding input activations, i and j stands for output feature and input feature dimension, respectively.

When it comes to pruning MoE, the expert layers constitute the majority of model parameters. For example, the Mixtral-8x7B has a total of 47B parameters where 1.3B belongs to attention modules and 45B is used for expert layers (2 out of 8 experts are activated, 12.5B active parameters during inference). Only a subset of experts are activated for different input tokens, so there is a large space of expert redundancy.

Motivation. Consider a simple Mixture-of-Experts model with two experts and each with only one weight: $y = Gate(x)_1 \cdot E_1(x) + Gate(x)_2 \cdot E_2(x) = Gate_1 \cdot w_1 \cdot x + Gate_2 \cdot w_2 \cdot x$, where $|w_1| \leq |w_2|$. If we want to remove one weight without incurring significant change on the output, traditional magnitude pruning (Han et al., 2016) will remove weight $|w_1|$. However, in MoE architecture, the router weights $Gate_k$ is an important part as it assigns different values to different experts. Especially when we consider a top-k setting that only a subset of experts are activated, the router weights $Gate_1$ could be a large value close to 1, while router weights $Gate_2$ could be 0 if it is not activated. As a results, $|Gate_1 \cdot w_1 \cdot x| \gg |Gate_2 \cdot w_2 \cdot x|$, and therefore we should remove weight w_2 instead to minimize change on the output.

This motivating example shows that for MoE architecture, we need to consider the importance of router weights. Previous pruning methods for LLMs do not consider the router weights which only exist in MoE architecture and may result in lower performance after pruning MoE. We propose a pruning metric designed explicitly for MoE LLMs to handle such a limitation while maintaining the simplicity of Wanda's pruning metric.

Router Tells It All. Motivated by the pruning metric in Wanda and the MoE routing policy, our approach, MoE-Pruner, prunes weights using the local relative importance (Zhang et al., 2024a) of weight, which compares against the ℓ_2-norm of its corresponding column and row, multiplied by the scaled input activations and router weights, on each output neuron:

$$\mathcal{S}_{ij} = \Big(\frac{|\mathbf{W}_{ij}|}{\sqrt{\sum_i |\mathbf{W}_{ij}|^2}} + \frac{|\mathbf{W}_{ij}|}{\sqrt{\sum_j |\mathbf{W}_{ij}|^2}}\Big) \cdot (\|\mathbf{X}_j \cdot \mathbf{Gate}_k\|_2)^a, \tag{9}$$

where $\mathbf{Gate}_k$ is the router weights for the k-th expert, a is a scale to control the strength of activations and router weights, and i and j stands for output feature and input feature dimension. In experiments, we find that $a = 0.1$ or 0.5 works generally well, but has slight difference for MoE models with different initialization methods. We use $a = 0.5$ for sparse upcycled models and $a = 0.1$

for models trained from scratch for better performance following our experiments in Table 7.

Table 1: Comparison of different pruning methods including magnitude pruning, SparseGPT, Wanda, and MoE-Pruner.

Method	Weight Update	Calib -ration	Pruning metric $\mathcal{S}_{ij}$	Complexity
Magnitude	✗	✗	$\lvert\mathbf{W}\rvert$	$\mathcal{O}(1)$
SparseGPT	✔	✔	$[\lvert\mathbf{W}\rvert^2/\mathrm{diag}(\mathbf{H}^{-1})]_{ij}$	$\mathcal{O}(d^3_{hidden})$
Wanda	✗	✔	$\lvert\mathbf{W}_{ij}\rvert \cdot \lVert\mathbf{X}_j\rVert$	$\mathcal{O}(d^2_{hidden})$
MoE-Pruner	✗	✔	$\left(\frac{\lvert\mathbf{W}_{ij}\rvert}{\lVert\mathbf{W}_{*j}\rVert_2}+\frac{\lvert\mathbf{W}_{ij}\rvert}{\lVert\mathbf{W}_{i*}\rVert_2}\right)\cdot(\lVert\mathbf{X}_j\cdot\mathbf{Gate}\rVert_2)^a$	$\mathcal{O}(d^2_{hidden})$

Table 1 summarizes pruning methods, including magnitude pruning, SparseGPT, Wanda, and MoE-Pruner and their pruning metric and complexity. Algorithm 1 presents the unstructured sparsity version of our MoE-Pruner algorithm, which is efficient and does not require a sophisticated weight update procedure.

Algorithm 1 The MoE-Pruner algorithm. We prune each expert layer weight matrix $\mathbf{W}$ to $p\%$ sparsity.

Initialize: A MoE model $\mathcal{M}$ with l MoE layers, where each MoE layer has n experts. Let $\mathbf{X} \in \mathbb{R}^{b\times d_{\text{col}}}$ and $\mathbf{Gate} \in \mathbb{R}^{b\times n}$ denote the *calibration samples* and *router weights* respectively.
for layer $t = 1, \ldots, l$ **do**
  $\mathbf{X}', \mathbf{Gate} \leftarrow \texttt{forward}(\text{layer}_t, \mathbf{X})$
  for expert $k = 1, \ldots, n$ **do**
    $\mathbf{M} \leftarrow \mathbf{1}_{d_{\text{row}}\times d_{\text{col}}}$
    $\mathcal{S}_{ij} \leftarrow (\frac{\lvert\mathbf{W}_{ij}\rvert}{\lVert\mathbf{W}_{*j}\rVert_2}+\frac{\lvert\mathbf{W}_{ij}\rvert}{\lVert\mathbf{W}_{i*}\rVert_2})\cdot(\lVert\mathbf{X}_j\cdot\mathbf{Gate}\rVert_2)^a$
    $idx \leftarrow \texttt{sort}(\mathcal{S}_{ij}, dim = 1)$
    $idx \leftarrow idx_{:,d_{\text{col}}*p\%}$
    $\mathbf{M} \leftarrow \texttt{scatter}(0, idx_{:,d_{\text{col}}*p\%})$
    $\mathbf{W}' \leftarrow \mathbf{M} \odot \mathbf{W}$
  end for
  $\mathbf{X}, \mathbf{Gate} \leftarrow \texttt{forward}(\text{layer}'_t, \mathbf{X})$
end for
Return: A pruned MoE model $\mathcal{M}'$.

Structured N:M Sparsity. Structured N:M sparsity (Mishra et al., 2021) can leverage NVIDIA's sparse tensor cores to accelerate matrix multiplication. MoE-Pruner can be easily extended from unstructured sparsity to structured N:M sparsity, where we compare weights using the same metric among every M consecutive weights, and remove N weights with lowest scores.

3.2 Expert-Wise Knowledge Distillation

Expert-Wise Knowledge Distillation. MoE models can preserve most of their capacity after pruning but still suffer from performance degradation. To recover MoE LLM performance, we fine-tune the model by leveraging the unpruned pre-trained model as a teacher model in an expert-wise knowledge distillation (KD) manner. The pre-trained model is a natural teacher model for the pruned model since they share exactly the same number of layers, experts, and dimensions. The loss function for expert-wise knowledge distillation is formulated as follows:

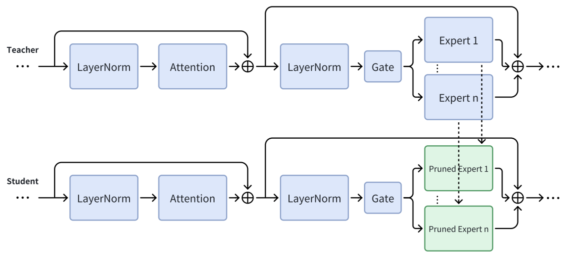


Figure 2: Expert-wise knowledge distillation for the pruned MoE model using the pre-trained MoE model as the teacher to recover the performance of the pruned model.

$$\begin{aligned}\mathcal{L}_{KD} &= \mathcal{L}_{CE} + \lambda \times \mathcal{L}_{expert} \\ &= \mathcal{L}_{CE} + \lambda \times \sum_{j=1}^{l}\sum_{k=1}^{n} \mathrm{MSE}(E^j_{kt}, E^j_{ks}),\end{aligned} \tag{10}$$

where $\mathcal{L}_{CE}$ is the cross entropy loss, MSE is the mean squared error calculated as $\mathrm{MSE}(X, Y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2$ for N-dimensional vectors X and Y. λ is a weighting coefficient and initialized based on the strength of cross entropy loss and expert-wise knowledge distillation loss: $\frac{\mathcal{L}_{CE}}{\mathcal{L}_{expert}}$. We sum up all the differences between teacher experts and student experts. Figure 2 illustrates the expert-wise knowledge distillation for pruned models. The corresponding expert in the pre-trained teacher model will be used to distill the expert in the pruned student model.

4 Experiments

Models, Datasets, and Evaluation. We conduct pruning experiments across various MoE models with different initialization methods to validate the effectiveness of MoE-Pruner, including Mixtral-8x7B (Jiang et al., 2024), MiniCPM-8x2B (Hu et al., 2024), DeepSeek-V2-Lite (Liu et al., 2024a), and Qwen1.5-MoE-A2.7B (Team, 2024). We use samples from the pretraining dataset C4 (Raffel et al., 2020) as calibration data for one-shot pruning since pretraining datasets are often more comprehensive and not dominated by knowledge specific to any particular domain. We use the exact same

Table 2: WikiText Perplexity↓ against other one-shot pruning methods, including SparseGPT, Wanda, NAEE, and MoE-Pruner, with 50% unstructured sparsity or 2:4 structured sparsity.

Method	50% Unstructured				2:4 Structured	
	Mixtral -8x7B	Mixtral -8x7B -Instruct	Mixtral -8x22B	Mixtral -8x22B -Instruct	Mixtral -8x7B	Mixtral -8x7B -Instruct
Pre-trained	3.84	4.14	2.83	2.89	3.84	4.14
SparseGPT	5.02	5.20	4.19	4.27	7.09	7.19
Wanda	4.97	5.16	3.97	4.06	6.98	6.92
NAEE (r=4)	-	-	-	-	6.49	6.42
MoE-Pruner	**4.68**	**4.94**	**3.64**	**3.72**	**5.60**	**5.69**

Table 3: Comparison against other one-shot pruning methods, including SparseGPT, Wanda, NAEE, and MoE-Pruner, about memory reduction and wall-clock time inference speedup on A100. The 2:4 sparsity is supported by the cuSPARSELt backend, while the previous CUTLASS backend only shows 1.14× speedup.

Model	Method	Sparsity	Average	Memory	Speedup
Mixtral -8x7B	Pre-trained	-	69.16	87.49	1.00×
	SparseGPT	2:4	54.73	50.74	**1.31**×
	Wanda	2:4	59.95	50.74	**1.31**×
	NAEE	r=4	61.70	**45.49**	1.01×
	MoE-Pruner	2:4	**64.58**	50.74	**1.31**×
	MoE-Distilled	2:4	**67.07**	50.74	**1.31**×

128 sequences of calibration data for all one-shot pruning experiments to control this variable factor. We evaluate the perplexity on the WikiText (Merity et al., 2017) validation set. Our expert-wise knowledge distillation method uses a subset of the C4 as the training set. We measure the average performance of pruned models on zero-shot tasks and language modeling, using nine popular tasks from EleutherAI LM Harness (Gao et al., 2023): ARC-easy, ARC-challenge (Clark et al., 2018), Boolq (Clark et al., 2019), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021), OpenBookQA (OBQA) (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), RTE (Wang et al., 2018), and WinoGrande (Sakaguchi et al., 2021).

Baselines and Experiments Setup. We compare MoE-Pruner with prior pruning methods, including SparseGPT (Frantar and Alistarh, 2023), Wanda (Sun et al., 2024) and NAEE (Lu et al., 2024). Similarly, our pruning algorithm is implemented in a layer-wise reconstruction manner. All pruning experiments are conducted on a server with 8 NVIDIA H100-80GB GPU. The fine-tuning experiments use the pruned model as a starting point and perform full-parameter fine-tuning to preserve the sparsity mask. We implement the expert-wise knowledge distillation method in Llama-Factory (Zheng et al., 2024) and conduct experiments on 2 servers, each with 8 NVIDIA H100-80GB GPUs. We fine-tune the pruned student model for three epochs, using a learning rate of 2e-5 with the cosine learning rate scheduler. All experimental results are after one-shot pruning, unless otherwise specified as “MoE-Distilled”.

4.1 One-Shot Pruning

Table 2 shows the one-shot pruning model perplexity on WikiText with both 50% unstructured and 2:4 structured sparsity. Across all tested models and sparsity, MoE-Pruner outperforms SparseGPT, Wanda, and NAEE. For the Mixtral-8x7B models, MoE-Pruner reduces perplexity by up to 0.31 compared to SparseGPT and Wanda. This gap expands when the MoE model scales to the Mixtral-8x22B model When applying the 2:4 structured sparsity, MoE-Pruner’s advantage is even more pronounced, achieving improvements of 1.49, 1.38, and 0.89 in perplexity over SparseGPT, Wanda, and NAEE, respectively.

Table 3 presents the memory reduction and inference speedup of MoE-Pruner compared with NAEE. MoE-Pruner at the structured 2:4 sparsity pattern outperforms NAEE in terms of average performance and shows a 1.31× inference speedup, while incurring only a small memory overhead for storing sparse tensor indices.

Table 4 shows the average zero-shot performance on nine zero-shot tasks for MoE models with 50% unstructured sparsity. Table 5 demonstrates the average zero-shot performance for MoE models at the structured 2:4 sparsity or 50% expert pruning. MoE-Pruner outperforms all the state-of-the-art pruning approaches by a large margin. Please note that these are the one-shot pruning results and no fine-tuning takes place at this stage.

4.2 Expert-Wise Knowledge Distillation Performance

The gap between the pruned MoE model and the pre-trained MoE model can be largely mitigated via expert-wise knowledge distillation. We only need 1000 training samples from C4, and training can be done in 1 hour. Table 6 shows the average zero-shot accuracy of the pruned and distilled Mixtral-8x7B MoE models with 50% unstructured sparsity. Our

Table 4: Average zero-shot performance on 9 evaluation tasks of pruned models using SparseGPT, Wanda, and MoE-Pruner, with 50% unstructured sparsity.

Model	Method	ARC-c	ARC-e	Boolq	HellaSwag	MMLU	OBQA	PIQA	RTE	WinoGrande	Average
Mixtral -8x7B	Pre-trained	56.91	84.47	85.29	64.78	67.03	35.0	82.43	70.4	76.16	69.16
	SparseGPT	50.43	80.68	84.62	60.20	61.79	32.8	81.12	**68.59**	**76.16**	66.27
	Wanda	51.02	80.89	85.08	60.45	62.73	32.6	80.90	64.64	74.82	65.90
	MoE-Pruner	**53.33**	**81.86**	**86.02**	**62.29**	**64.76**	**33.6**	**81.61**	66.06	75.53	**67.23**
MiniCPM -8x2B	Pre-trained	42.75	76.22	77.28	56.49	52.63	29.0	77.48	75.81	66.61	61.58
	SparseGPT	39.25	73.44	**76.36**	53.19	48.35	**28.0**	76.22	64.62	**64.96**	58.26
	Wanda	40.44	72.73	74.71	51.70	45.78	25.8	76.06	71.84	61.48	57.84
	MoE-Pruner	**40.87**	**74.92**	74.74	**54.59**	**48.89**	**28.0**	**76.61**	**72.56**	64.56	**59.53**
DeepSeek -V2-Lite	Pre-trained	46.67	78.28	79.88	58.65	54.94	34.2	80.03	61.37	71.35	62.81
	SparseGPT	40.36	73.70	73.27	50.37	39.85	29.0	76.66	58.12	67.25	56.51
	Wanda	41.64	73.44	71.83	51.36	39.83	29.0	77.53	**63.90**	66.93	57.27
	MoE-Pruner	**44.62**	**76.30**	**78.56**	**55.92**	**49.72**	**31.2**	**78.62**	60.29	**70.32**	**60.62**
Qwen1.5 -MoE -A2.7B	Pre-trained	41.81	73.32	79.88	57.98	61.29	30.0	80.09	69.31	68.98	62.58
	SparseGPT	34.81	68.90	76.24	49.86	51.55	25.2	77.09	55.96	**67.32**	56.33
	Wanda	33.02	67.30	75.11	48.26	50.35	26.8	75.35	62.09	65.82	56.01
	MoE-Pruner	**39.68**	**72.60**	**78.44**	**54.88**	**57.63**	**30.4**	**78.73**	**72.92**	66.93	**61.36**

Table 5: Average zero-shot performance on 9 evaluation tasks of pruned models using SparseGPT, Wanda, NAEE, and MoE-Pruner, at the structured 2:4 sparsity or 50% expert pruning.

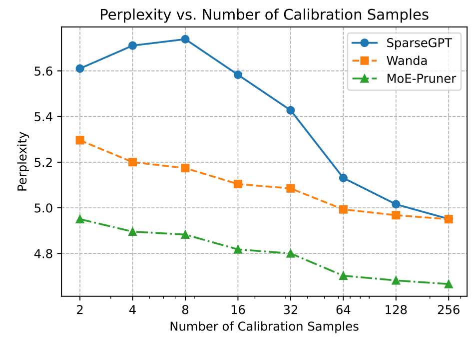
Model	Method	ARC-c	ARC-e	Boolq	HellaSwag	MMLU	OBQA	PIQA	RTE	WinoGrande	Average
Mixtral -8x7B	Pre-trained	56.91	84.47	85.29	64.78	67.03	35.0	82.43	70.4	76.16	69.16
	SparseGPT (2:4)	41.72	74.96	76.85	53.26	52.86	28.6	78.35	66.43	72.38	54.73
	Wanda (2:4)	41.55	74.12	76.61	53.19	52.26	27.8	77.04	63.90	70.48	59.95
	NAEE (r=4)	**48.38**	77.99	**80.52**	57.81	47.68	28.6	78.67	62.45	73.16	61.70
	MoE-Pruner (2:4)	47.87	**79.00**	79.54	**58.86**	**62.17**	**31.8**	**79.49**	**68.23**	**74.27**	**64.58**
MiniCPM -8x2B	Pre-trained	42.75	76.22	77.28	56.49	52.63	29.0	77.48	75.81	66.61	61.58
	SparseGPT (2:4)	33.36	69.07	70.80	47.96	37.96	21.4	73.99	57.76	60.06	52.48
	Wanda (2:4)	33.11	63.34	66.30	42.31	27.23	19.6	69.59	59.57	55.41	48.50
	NAEE (r=4)	33.28	57.87	67.25	42.04	23.39	18.0	68.34	56.68	56.83	47.08
	MoE-Pruner (2:4)	**37.71**	**71.04**	**72.54**	**51.66**	**42.42**	**24.2**	**75.08**	**70.40**	**60.62**	**56.19**
DeepSeek -V2-Lite	Pre-trained	46.67	78.28	79.88	58.65	54.94	34.2	80.03	61.37	71.35	62.81
	SparseGPT (2:4)	33.19	66.67	66.15	44.16	26.65	24.6	74.32	51.26	62.75	49.97
	Wanda (2:4)	31.31	63.97	65.44	41.85	30.53	23.2	72.69	48.01	61.72	48.75
	NAEE (r=32)	22.87	41.33	62.26	36.20	29.89	20.6	62.79	53.07	54.14	42.57
	MoE-Pruner (2:4)	**40.02**	**71.89**	**76.61**	**50.94**	**43.85**	**27.2**	**76.22**	**55.96**	**67.64**	**56.70**
Qwen1.5 -MoE -A2.7B	Pre-trained	41.81	73.32	79.88	57.98	61.29	30.0	80.09	69.31	68.98	62.58
	SparseGPT (2:4)	33.62	67.05	71.01	43.87	42.29	26.0	74.10	62.45	65.51	53.98
	Wanda (2:4)	30.29	62.12	64.59	40.68	37.63	23.4	72.14	57.40	64.48	50.30
	NAEE (r=30)	32.25	59.34	67.28	46.74	38.08	21.2	73.50	64.26	60.46	51.46
	MoE-Pruner (2:4)	**39.93**	**71.21**	**71.53**	**52.73**	**56.31**	**29.4**	**78.18**	**70.04**	**67.80**	**59.68**

Table 6: Average zero-shot performance after pruning and expert-wise knowledge distillation.

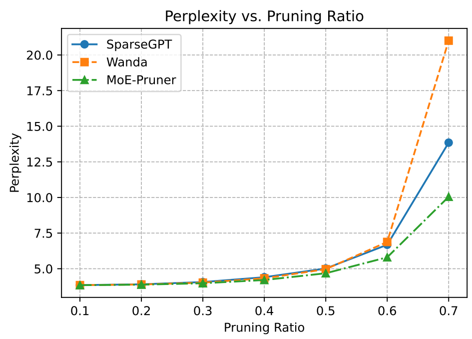
Model	Method	ARC-c	ARC-e	Boolq	HellaSwag	MMLU	OBQA	PIQA	RTE	WinoGrande	Average
Mixtral -8x7B	Pre-trained	56.91	84.47	85.29	64.78	67.03	35.0	82.43	70.4	76.16	69.16
	Wanda	51.02	80.89	85.08	60.45	62.73	32.6	80.90	64.64	74.82	65.90
	Wanda Fine-tuned	53.74	81.25	85.46	64.47	64.73	33.0	80.98	66.74	75.23	67.29
	MoE-Pruned	53.33	**81.86**	**86.02**	62.29	64.76	33.6	81.61	66.06	75.53	67.23
	MoE-Distilled	**54.35**	81.19	85.26	**68.77**	**65.59**	**36.0**	**82.48**	**68.23**	**75.72**	**68.40**

distilled MoE model could achieve a 68.40 average performance on nine zero-shot tasks, beating model fine-tuned by Wanda. The performance is very close to the pre-trained Mixtral-8x7B MoE model, which demonstrates a 69.16 average performance.

4.3 Ablation Studies



(a) Perplexity with different number of calibration samples at 50% sparsity.



(b) Perplexity over different pruning ratios with 128 calibration samples.

Figure 3: Ablation studies on calibration samples and pruning ratios.

Ablation on Different Number of Calibration Samples. We use different number of calibration samples ranging from 2 to 256 to prune the Mixtral-8x7B model. Perplexity results after one-shot pruning with unstructured 50% sparsity are summarized in Figure 3a. We see a clear difference in trend as the number of calibration samples changes. MoE-Pruner is much more robust than SparseGPT when there are few calibration samples and performs the same trend but better perplexity over Wanda. Notably, even with just two calibration samples, pruned networks obtained by MoE-Pruner have a perplexity of just 4.95. This may be because input norm statistics could be much easier to estimate than the full inverse Hessian of the local layer-wise reconstruction problem.

Ablation on Different Sparsity Ratios. We also change the pruning ratio using the same 128 calibration samples. Figure 3b shows that at lower pruning ratios, such as 10% to 40%, all pruning methods achieve good perplexity. When the pruning ratio increases, the Wanda pruned model perplexity changes dramatically and fails at 70%. MoE-Pruner shows better and more stable pruning results especially at higher pruning ratios. This demonstrates that router weights preserve important information when selecting experts and provide a clear hint for pruning unimportant weights.

5 Conclusion

We propose an efficient and effective pruning method for MoE models, MoE-Pruner. We prune weights with the smallest magnitudes multiplied by the corresponding input activations and router weights. Our pruning method is one-shot and fast, without any retraining or weight update procedures. Pruning MoE LLM with high sparsity will incur performance degradation, so we also propose an expert-wise knowledge distillation method that leverages the unpruned pre-trained MoE model as a teacher to guide the pruned student model to recover performance. Extensive experimental results across various MoE models validate the effectiveness of our algorithm and MoE-Pruner outperforms all one-shot pruning methods. The pruned model with 50% sparsity maintains 99% of the performance of the original model after the expert-wise knowledge distillation.

Limitations

Our method can reduce memory usage and improve inference speed for more efficient deployment of MoE LLMs. Despite its advancements, there are still some limitations. We conduct experiments across various MoE models, but not those largest MoE models which has over 300B total parameters, as it is impossible to load these large MoE models on one machine without the help of quantization. We use float16 datatype in our experiments to guarantee numerical precision. We will carry out experiments on these large MoE LLMs in the future using more computation resources and exploring quantized MoE models to give a more comprehensive analysis of the scalability and generalizability of our method.

References

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, pages 7432–7439.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

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*.

Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, Xian-Ling Mao, Heyan Huang, and Furu Wei. 2022. On the representation collapse of sparse mixture of experts. In *Advances in Neural Information Processing Systems*.

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. In *International Conference on Machine Learning*.

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)*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Mark Collier, Efi Kokiopoulou, Andrea Gesmundo, and Jesse Berent. 2020. Routing networks with co-training for continual learning. *arXiv preprint arXiv:2009.04381*.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, Zhenda Xie, Y.K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

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.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Song Han, Huizi Mao, and William J Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations*.

Babak Hassibi, David G Stork, and Gregory J Wolff. 1993. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):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, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

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*.

Yeskendir Koishekenov, Alexandre Berard, and Vassilina Nikoulina. 2023. Memory-efficient nllb-200: Language-specific expert pruning of a massively multilingual machine translation model. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. 2023. Sparse upcycling: Training mixture-of-experts from dense checkpoints. In *International Conference on Learning Representations*.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. In *Advances in Neural Information Processing Systems*.

Jaeseong Lee, Aurick Qiao, Daniel F Campos, Zhewei Yao, Yuxiong He, et al. 2024. Stun: Structured-then-unstructured pruning for scalable moe pruning. *arXiv preprint arXiv:2409.06211*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. {GS}hard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*.

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

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. 2024a. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.

Enshu Liu, Junyi Zhu, Zinan Lin, Xuefei Ning, Matthew B Blaschko, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. 2024b. Efficient expert pruning for sparse mixture-of-experts language models: Enhancing performance and reducing inference costs. *arXiv preprint arXiv:2407.00945*.

Miao Lu, Xiaolong Luo, Tianlong Chen, Wuyang Chen, Dong Liu, and Zhangyang Wang. 2022. Learning pruning-friendly networks via frank-wolfe: One-shot, any-sparsity, and no retraining. 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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6159–6172.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

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*.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. 2021. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*.

Sarthak Mittal, Yoshua Bengio, and Guillaume Lajoie. 2022. Is a modular architecture enough? In *Advances in Neural Information Processing Systems*.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. 2024. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*.

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.

Carlos Riquelme Ruiz, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. 2021. Scaling vision with sparse mixture of experts. In *Advances in Neural Information Processing Systems*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

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. In *International Conference on Learning Representations*.

Sheng Shen, Zhewei Yao, Chunyuan Li, Trevor Darrell, Kurt Keutzer, and Yuxiong He. 2023. Scaling vision-language models with sparse mixture of experts. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*.

Qwen Team. 2024. Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters".

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: a multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Tianwen Wei, Bo Zhu, Liang Zhao, Cheng Cheng, Biye Li, Weiwei Lü, Peng Cheng, Jianhao Zhang, Xiaoyu Zhang, Liang Zeng, et al. 2024. Skywork-moe: A deep dive into training techniques for mixture-of-experts language models. *arXiv preprint arXiv:2406.06563*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Cheng Yang, Yang Sui, Jinqi Xiao, Lingyi Huang, Yu Gong, Yuanlin Duan, Wenqi Jia, Miao Yin, Yu Cheng, and Bo Yuan. 2024b. Moe-i^2: Compressing mixture of experts models through inter-expert pruning and intra-expert low-rank decomposition. In *Findings of the Association for Computational Linguistics: EMNLP 2024*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. 2024a. Plug-and-play: An efficient post-training pruning method for large language models. In *The Twelfth International Conference on Learning Representations*.

Zeliang Zhang, Xiaodong Liu, Hao Cheng, Chenliang Xu, and Jianfeng Gao. 2024b. Diversifying the expert knowledge for task-agnostic pruning in sparse mixture-of-experts. *arXiv preprint arXiv:2407.09590*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Appendix

A Choice of Scale a for MoE

Table 7: WikiText Perplexity↓ of 2:4 structured pruned MoE models with different initialization method, including Mixtral-8x7B (sparse upcycled), MiniCPM-8x2B (sparse upcycled), and DeepSeek-V2-Lite (train from scratch). a is the scale to control the strength of activations and router weights in our pruning metric.

Model	$a = 0.5$	$a = 0.1$
Mixtral-8x7B	**5.60**	5.72
MiniCPM-8x2B	**8.78**	8.92
DeepSeek-V2-Lite	10.04	**9.76**

B Open-Source MoE Models

Table 8: Open-Source MoE Models List (Released after Jan. 2024).

Name	Active Parameters	Total Parameters	# Experts	Routing Policy	Initialized Method	MMLU*
OLMoE	1B	7B	64	top-8	train from scratch	54.1
MiniCPM-MoE-8x2B	4B	13.6B	8	top-2	sparse upcycled	58.9
Qwen1.5-MoE-A2.7B	2.7B	14.3B	4(shared)+60	4+top-4	sparse upcycled	62.5
Deepseek-V2-Lite	2.4B	16B	2(shared)+64	2+top-6	train from scratch	58.3
Yuan2.0-M32	3.7B	40B	32	top-2	train from scratch	72.2
GRIN-MoE	6.6B	41.9B	16	top-2	sparse upcycled	79.4
Mixtral-8x7B	12.5B	47B	8	top-2	sparse upcycled	70.4
Jamba	12B	52B	16	top-2	unknown	67.4
Qwen2-57B-A14B	14B	57.4B	8(shared)+64	8+top-8	sparse upcycled	76.5
DBRX	36B	132B	16	top-4	unknown	73.7
Mixtral-8x22B	39B	141B	8	top-2	sparse upcycled	77.8
Skywork-MoE	22B	146B	16	top-2	sparse upcycled	77.4
Deepseek-V2	21B	236B	2(shared)+160	2+top-6	train from scratch	78.5
grok-1	80B	314B	8	top-2	unknown	73.0
Hunyuan-A52B	52B	389B	1(shared)+16	1+top-1	unknown	88.4
MiniMax-Text-01	45.9B	456B	32	top-2	unknown	88.5
Snowflake Arctic	17B	480B	128	top-2	unknown	67.3
DeepSeek-V3	37B	671B	1(shared)+256	1+top-8	train from scratch	88.5

*Note: This table presents a subset of open-source MoE models and is not exhaustive. The list is sorted by total parameters. MMLU scores are extracted from original papers or reports and may not reflect model real performance.