Condense, Don't Just Prune: Enhancing Efficiency and Performance in MoE Layer Pruning

Anonymous authors
Paper under double-blind review

Abstract

Mixture-of-Experts (MoE) has garnered significant attention for its ability to scale up neural networks while utilizing the same or even fewer active parameters. However, MoE does not relieve the massive memory requirements of networks, which limits their practicality in real-world applications, especially in the era of large language models (LLMs). While recent work explores the possibility of removing entire layers of MoE to reduce memory, the performance degradation is still notable. In this paper, we propose ConDense-MoE (CD-MoE) that, instead of dropping the entire MoE layer, condenses the large, sparse MoE layer into a smaller, denser layer with only a few experts activated for all tokens, while maintaining hardware friendliness. Our approach is specifically designed for fine-grained MoE with shared experts, where Feed-Forward Networks are split into many small experts, with certain experts isolated to serve as shared experts that are always activated, such as DeepSeekMoE and QwenMoE. We demonstrate the effectiveness of our method. Specifically, for the DeepSeekMoE-16B model, our approach maintains **90%** of the average accuracy while reducing memory usage by **27.5%** and increasing inference speed to **1.26** times. Moreover, we show that by applying lightweight expert fine-tuning—only to the condensed layers—and using 5 hours on a single 80G A100 GPU, we can successfully recover **98%** of the original performance. Our code has been submitted.

1 Introduction

Large Language Models (LLMs) continue to grow in both size and capability (Brown, 2020; Touvron et al., 2023b;a), fueling demand for architectures that can scale with minimal increases in computational cost. Mixture of Experts (MoE) architectures have emerged as a promising solution to this challenge (Shazeer et al., 2017; Jiang et al., 2024; Team et al., 2024). Unlike standard dense networks, MoEs selectively activate only the most relevant subset of parameters (referred to as "experts") for a given input, enabling substantial growth in model capacity without a proportional escalation in compute overhead. Recent advances in fine-grained expert segmentation (Dai et al., 2024; Team, 2024) push this concept further by splitting feed-forward layers into many small experts, while designating a small set of shared experts that remain active across all tokens. Despite the clear benefits, MoE architectures still face major deployment barriers due to high memory costs. Storing a large number of experts, even if most remain dormant per token, can be prohibitively expensive in real-world systems. This reality highlights the urgent need for approaches that preserve the performance benefits of MoEs while lowering their memory footprint.

In this paper, we present a new framework called **ConDense-MoE** (`CD-MoE`), a novel framework that significantly enhances the efficacy specifically for DeepSeek-style shared-expert-type MoE (Dai et al., 2024; Team, 2024). Unlike prior MoE compression approaches that often remove entire layers or retain the routing mechanism with fewer experts (He et al., 2024b), our method eliminates the routing process at selected layers and condenses them into dense layers. Specifically, we devise a greedy strategy to identify which layers—and which experts within them—can be condensed with minimal impact on model accuracy, leading to significant reductions in memory usage and faster inference.

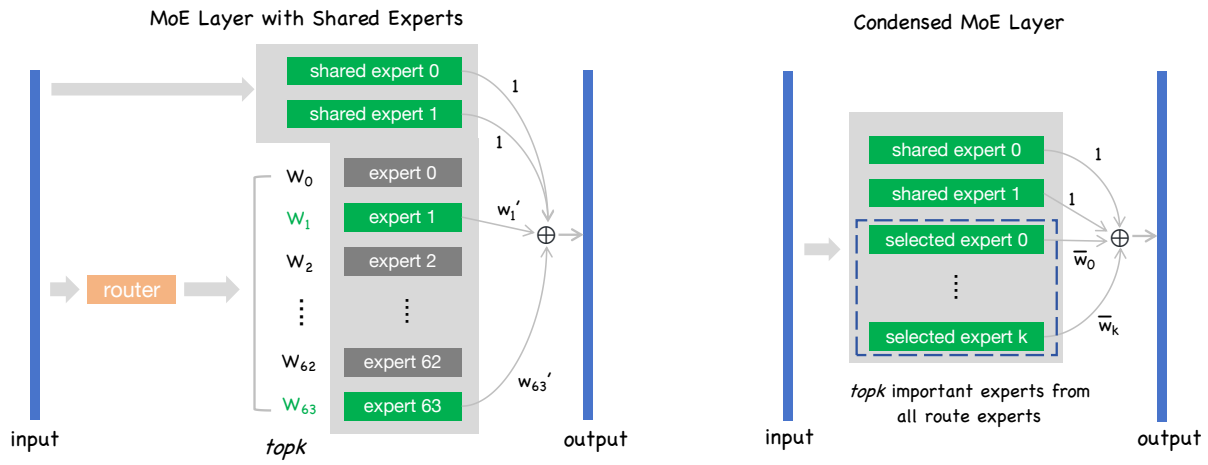

Figure 1: **Left:** The structure of the Deepseek MoE layer. w_i' represents the weights after normalization. **Right:** The structure of the ConDense-MoE layer, where only the most important top-k experts are retained. $\bar{w}_i$ represents the fixed weights that are pre-computed during condensing using the average weight of all calibration tokens.

`CD-MoE` stems from a surprising yet critical insight: in shared-expert–type MoEs, simply removing the routing mechanism while retaining only the shared experts at a given layer generally incurs minor output difference. Building on this, `CD-MoE` preserves these essential shared experts and a small subset of routing experts, thereby condensing originally sparse layers into compact, dense analogs. As these preserved experts account for only a small fraction of overall parameters, we substantially cut memory requirements without sacrificing much performance. Our experiments demonstrate that on *DeepSeekMoE-16B* (Dai et al., 2024), `CD-MoE` reduces memory by 27.5% and maintains nearly 90% of the original accuracy on a suite of downstream tasks. Moreover, our method uncovers additional experts that are beneficial for fine-tuning, enabling up to 98% recovery of the original performance after a lightweight supervised fine-tuning that updates only the condensed layers. In summary, our contributions are:

- **A novel MoE condensation approach** that selectively removes less critical experts and condenses large sparse MoE layers into small, dense structures.
- **An efficient selection algorithm** that identifies which layers to condense and which experts to preserve, achieving a favorable trade-off between memory savings and accuracy.
- **Empirical validation** on *DeepSeekMoE-16B*, demonstrating a 27.5% memory reduction and a 1.26× inference speed, while retaining up to 90% of the zero-shot accuracy. Moreover, lightweight fine-tuning can effectively reclaim 98% of the original performance using only a single A100 GPU in a few hours.

2 Related Work

Mixture-of-Experts (MoE) architectures have become a powerful framework for scaling neural networks without linearly increasing computational burden (Shazeer et al., 2017; Jiang et al., 2024; Dai et al., 2024; Team, 2024). Early methods such as Switch Transformer (Fedus et al., 2022) demonstrated that sparse activation can substantially boost model capacity while maintaining tractable compute. Further work has refined routing policies (Jiang et al., 2024) and introduced a fine-grained segmentation of feed-forward layers into many small experts, often augmented with "shared" experts that remain active for every token (Dai et al., 2024; Team, 2024). These advances collectively enhance training stability and generalization, yet the growing pool of experts poses serious deployment challenges, chiefly due to large memory demands.

A range of pruning and compression strategies have emerged to address these overheads. Some works prune experts by identifying those that are minimally activated or less essential for downstream tasks (Chi et al., 2022;

Sun et al., 2024; Muzio et al., 2024), with solutions ranging from exhaustive searches over all expert subsets (Lu et al., 2024) to filtering experts by their activation frequency. While these methods typically preserve routing, more aggressive approaches prune entire MoE layers, achieving considerable memory reductions but at the risk of pronounced accuracy losses (He et al., 2024b). In contrast, our technique selectively eliminates the routing mechanism in particular layers and retains only a small subset of experts (including the shared experts), thus reducing memory demands while sustaining most of the model's representational power.

3 ConDense-MoE (CD-MoE)

In this section, we present a comprehensive explanation of the CD-MoE process, which follows a sequential workflow comprising several critical steps. First, we delve into *Expert Selection and Condensing* (Section 3.2), where we detail the methods used to identify and condense the most essential experts from all available experts within a layer. Second, we explore *Layer Selection and Condensing* (Section 3.3), focusing on selecting the most suitable layers for condensation to preserve as much of the LLM's inherent capabilities as possible. Last but not least, we highlight that only performing *lightweight fine-tuning* (Section 4.3) on condensed layers, can almost recover the original performance of MoE.

3.1 Preliminaries of MoE

Here we define the basic structure of an MoE layer. All experts in an MoE layer are represented as $\{E_1, E_2, \ldots, E_n\}$. For most fine-grained MoE models, in addition to routing experts, the shared experts E_s is used. Any token will activate the shared experts and several routing experts, as shown in Figure 1 left. For the input data X, the output is computed as follows:

$$\mathbf{h}_t = \sum_{i=1}^{N} \left(g_{i,t}\, E_i\left(\mathbf{x}_t\right)\right) + E_s\left(\mathbf{x}_t\right) + \mathbf{x}_t, \tag{1}$$

where

$$g_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \mathrm{TopK}\left(\{s_{j,t} \mid 1 \le j \le N\}, K\right), \\ 0, & \text{otherwise}, \end{cases}$$

$$s_{i,t} = \mathrm{Softmax}_i\left(\mathbf{x}_t^T \mathbf{e}_i\right).$$

Here, N denotes the total number of route experts, $E_i(\cdot)$ represents the i-th expert, E_s is the shared expert, $g_{i,t}$ is the gate value for the i-th expert, $s_{i,t}$ denotes the token-to-expert similarity, $\mathrm{TopK}(\cdot, K)$ represents the set consisting of the K highest similarity scores among those computed for the t-th token and all N experts, and $\mathbf{e}_i$ is the centroid of the i-th expert.

3.2 Expert Selection and Condensing

Our CD-MoE framework removes the standard routing mechanism during inference. For each expert E_i, we use open-source C4 calibration data to compute the mean of $g_{i,t}$ when that expert is activated, which is then used as the fixed gate value g_i for that expert during the inference phase:

$$g_i = \left(1/|t \mid g_{i,t} \neq 0|\right) \sum_{t \mid g_{i,t} \neq 0} g_{i,t} \tag{2}$$

For the condense layer, we only retain K routing experts including the shared experts, which is consistent with the number of active experts during the training phase. Additionally, benefiting from sufficient training during the training stage, the shared expert is retained, as shown in Figure 1 right. After condensation, the MoE layer is represented as:

$$\mathbf{h}_t = \sum_{i=1}^{K} \left(g_{i,t}\, E_i\left(\mathbf{x}_t\right)\right) + E_s\left(\mathbf{x}_t\right) + \mathbf{x}_t \tag{3}$$

Algorithm 1: Condense Experts Selection

Input: Calibration data input X, routing expert set $E_{\text{route}} = \{E_1, \ldots, E_n\}$, number of selected experts K
Output: Condense expert set $E_{\text{condense}} = \{E_{\text{shared}}\}$
Initialize $E_{\text{condense}} \leftarrow \{E_{\text{shared}}\}$, $O_{\text{ref}} \leftarrow \text{Layer}_{\text{route}}(X)$ ▷ //Compute reference output with all routing experts
for $k \leftarrow 1$ **to** K do
  for each expert $E_i \in E_{\text{route}}$ do
    $E_{\text{condense}} \leftarrow E_{\text{condense}} \cup \{E_i\}$ ▷ //Temporarily add expert to condense set
    $O_{\text{tmp}} \leftarrow \text{Layer}_{\text{condense}}(X)$ ▷ //Compute output with current condense experts
    $loss_i \leftarrow \text{JS}(O_{\text{ref}}, O_{\text{tmp}})$ ▷ //Compute JS divergence as loss for E_i
    $E_{\text{condense}} \leftarrow E_{\text{condense}} \setminus \{E_i\}$ ▷ //Remove expert after computing loss
  $E_{\text{best}} \leftarrow \text{argmin}_{E_i \in E_{\text{route}}} loss_i$ ▷ //Select the expert with minimum loss
  $E_{\text{condense}} \leftarrow E_{\text{condense}} \cup \{E_{\text{best}}\}$ ▷ //Add best expert to condense set
  $E_{\text{route}} \leftarrow E_{\text{route}} \setminus \{E_{\text{best}}\}$ ▷ //Remove selected expert from routing set

Next, we describe how to select the most crucial experts. Lu et al. (2024) explore all possible expert combinations and choose the combination of experts that makes the layer outputs before and after pruning as close as possible, while pruning the rest. However, for fine-grained MoE models like *DeepSeekMoE-16B* (Dai et al., 2024) and *Qwen1.5-MoE-A2.7B* (Team, 2024), where the number of experts often exceeds 60, testing all permutations of experts requires significant computational resources and time. Therefore, we propose using a greedy search approach Algorithm 1 to select the most critical experts.

In which, K is consistent with the number of experts activated in the original model, and $JS(.)$ is the Jensen-Shannon divergence used to measure the difference between two outputs. Zhang et al. (2024) shown that in model pruning, this metric is better than angular distance and Euclidean distance. We also demonstrated its superiority over other metrics such as Kullback–Leibler (KL) divergence and perplexity in Table 5.

3.3 Layer Selection and Condensing

Algorithm 2: Condense Layers Selection

Input: Data input X, routing layer set $L_{\text{route}} = \{L_1, \ldots, L_n\}$, number of condensed layers N
Output: Condense layer set $L_{\text{condense}} = \{\}$
Initialize $L_{\text{condense}} \leftarrow \{\}$, $O_{\text{ref}} \leftarrow \text{Model}(X)$ ▷ //Compute reference output with all routing layers
for $n \leftarrow 1$ **to** N do
  for each layer $L_i \in L_{\text{route}}$ do
    $L_{\text{condense}} \leftarrow L_{\text{condense}} \cup \{L_i\}$ ▷ //Temporarily add layer to condense set
    $O_{\text{tmp}} \leftarrow \text{Model}(X)$ ▷ //Compute output with current condense layers
    $loss_i \leftarrow \text{JS}(O_{\text{ref}}, O_{\text{tmp}})$ ▷ //Compute JS divergence as loss for L_i
    $L_{\text{condense}} \leftarrow L_{\text{condense}} \setminus \{L_i\}$ ▷ //Remove layer after computing loss
  $L_{\text{best}} \leftarrow \text{argmin}_{L_i \in L_{\text{route}}} loss_i$ ▷ //Select the layer with minimum loss
  $L_{\text{condense}} \leftarrow L_{\text{condense}} \cup \{L_{\text{best}}\}$ ▷ //Add best layer to condense set
  $L_{\text{route}} \leftarrow L_{\text{route}} \setminus \{L_{\text{best}}\}$ ▷ //Remove selected layer from routing set

Clark et al. (2019b) have confirmed that different layers in language models exhibit distinct functionalities. While our objective is to retain the most impactful experts during the condensing process of each layer, it is inevitable that the outputs will deviate from their original values due to reduced capacity. Indiscriminately condensing all layers can thus lead to significant degradation in model performance and the loss of essential features. Therefore, akin to our approach in expert selection, we prioritize condensing those layers that exert minimal impact on model output changes post-pruning.

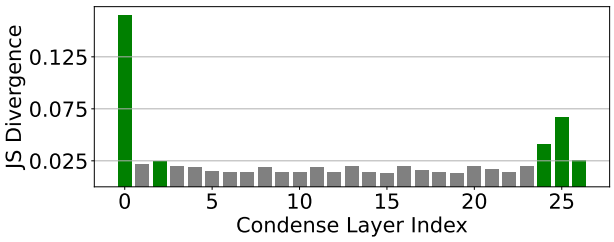

Figure 2: Fluctuations in the JS divergence between the the outputs of the condensed model and the original dense model across different layers.

To systematically quantify the impact of condensing individual layers, we conducted preliminary experiments assessing the output changes using the Jensen-Shannon (JS) divergence between the pruned and unpruned model outputs as our evaluation metric. As illustrated in Figure 2, condensing layer 0 results in a significant shift in the output distribution, indicating its critical role in the model's performance. While the middle layers exhibit some fluctuation, condensing the last three layers also causes noticeable deviations, suggesting that these layers are essential for maintaining output fidelity. These observations indicate that condensing different layers introduces varying degrees of change in the JS metric, reflecting their varying importance to the overall model performance.

Motivated by these findings, we adopt a greedy search strategy analogous to that used in expert selection to determine the optimal layers for condensing. Specifically, we compute the JS divergence before and after condensing based on the output of the final layer, iteratively adding layers to the $Layer_{condense}$ set that can minimize the distance between the model's final layer output and the output before condensing, as detailed in Algorithm 2. For layers that are not condensed, the standard token routing mechanisms are employed during forward propagation to maintain their full expressive power. For the layers being condensed, the routing mechanism is removed, and the MOE is compressed into a dense structure. The number of condensing layers **N** in can be chosen based on goals (e.g., memory budget vs. desired accuracy). This selective condensing approach allows us to effectively reduce computational complexity and memory usage while minimizing degradation in model performance.

3.4 Time Complexity Analysis

The computational efficiency of our greedy search algorithms stems from their polynomial time complexity relative to key model parameters:

- **Greedy Expert Selection:** For a layer with E experts and target selection of K experts, each iteration evaluates all remaining candidates ($O(E)$ experts). The total complexity is:

$$O(K \cdot E) \tag{4}$$

  This improves upon exhaustive search's $O(2^E)$ complexity (Lu et al., 2024).

- **Greedy Layer Selection:** With L candidate layers and N target condensed layers, each step evaluates up to L layers. The complexity becomes:

$$O(N \cdot L) \tag{5}$$

  This avoids $O(L!)$ brute-force layer permutations

For *DeepSeekMoE-16B* ($E = 66$, $L = 27$), selecting $K = 6$ experts per condensed layer requires only 369 inferences, demonstrating practical scalability. Comprehensive experimental results are presented in Table A.1. The linear dependence on E and L makes our approach feasible for billion-parameter MoEs.

Table 1: Comparison on *DeepSeekMoE-16B*. The underlined variables are tried to be kept consistent for fair comparison, and the **bolds** represent the best results. '*#L*' represents the number of pruned layers; 'ACT.' represents the number of activated parameters; and 'MEM.' is the remaining memory ratio compared to the original model. `CD-MoE-S` refers to retaining only the **S**hared expert after condensing, while `CD-MoE-SR` means selecting additional 6 **R**outed experts and keeping as dense. 6 route experts are consistent with the number of experts activated in the dense model.

method	**#L**	**ACT.**	**MEM.**	**Speedup**	ARC-C	BOOLQ	HELLA	MMLU	OBQA	PIQA	RTE	WINO	**AVG.**
GPT-NEO-2.7B	-	2.7B	-	-	29.8	61.7	55.2	25.0	34.6	72.9	53.4	57.8	48.8
OPT-2.7B	-	2.7B	-	-	31.2	59.9	60.6	25.4	35.2	74.7	54.2	60.5	50.2
BLOOM-3B	-	3B	-	-	30.4	61.4	54.5	25.9	32.4	70.7	56.0	58.6	48.7
OPENLLAMA-3B	-	3B	-	-	36.1	67.1	64.4	23.9	38.6	75.1	54.2	62.3	52.7
UNPRUNED	-	2.8B	100%	1.0x	48.3	72.7	77.4	38.3	44.2	78.7	63.2	70.1	61.6
						W/O SFT							
LAYERDROP	8	2.4B	72.2%	1.34x	35.3	65.4	57.5	29.8	32.6	62.9	48.0	63.6	49.4
BLOCKDROP	8	2.3B	71.3%	1.42x	36.2	62.2	57.2	29.0	34.6	70.2	63.2	64.7	52.2
M-SMOE	9	2.8B	70.0%	1.0x	37.1	71.1	64.0	28.0	36.4	75.5	59.9	67.2	54.9
`CD-MoE-S`	8	2.4B	73.1%	1.34x	37.8	70.6	65.9	28.8	38.6	75.0	58.8	65.0	55.1
`CD-MoE-SR`	9	2.8B	72.5%	1.26x	37.9	72.3	64.4	27.1	37.6	75.9	61.7	67.0	**55.5**
						W/ LIGHTWEIGHT SFT							
`CD-MoE-S`	8	2.4B	73.1%	1.34x	41.3	72.1	65.9	39.7	40.0	76.2	67.9	69.7	59.1
`CD-MoE-SR`	9	2.8B	72.5%	1.26x	42.6	79.2	66.9	38.6	38.6	75.7	66.8	75.2	**60.5**

4 Experiments

4.1 Experiment Setup

We adopt the open-source models *DeepSeekMoE-16B* and *Qwen1.5-MoE-A2.7B* (Team, 2024) (with 16B and 14.3B parameters, respectively) as representative high-capacity examples, and we present results on the larger model *Qwen2-MoE-57B-A14B* in A.5. We randomly sample 100 examples from the open-source C4 dataset to serve as calibration data during the condensing phase of our model, due to its widespread adoption in the LLM community (Sun et al., 2024; Frantar & Alistarh, 2023; Yin et al., 2024).

Evaluation: To provide a comprehensive assessment of our model's performance, we follow the protocol established by He et al. (2024b). We report zero-shot accuracies on eight diverse tasks selected from the EleutherAI Language Model Evaluation Harness (Gao et al., 2024), which includes the 7 open-source commonsense reasoning benchmark datasets: *ARC-Challenge* (Clark et al., 2018), *BoolQ* (Clark et al., 2019a), *HellaSwag* (Zellers et al., 2019), *OpenBookQA* (Mihaylov et al., 2018), *PIQA* (Bisk et al., 2019), *RTE* (Dagan et al., 2005), *WinoGrande* (Sakaguchi et al., 2019), and a more challenging dataset *MMLU* (Hendrycks et al., 2021).

Baseline: For comparative analysis, we primarily benchmark our approach against the *Block Drop* and *Layer Drop* (He et al., 2024b) and *M-SMoE* (Li et al., 2024b). These methods serve as strong baselines for pruning MoE models and are directly relevant to our study. *Block Drop* and *Layer Drop* prune all experts and the routing process in the layer, which is more aggressive than our method, while *M-SMoE* more conservatively retains the routing process and only merges the experts into a smaller number. We also compared the results with dense models that have similar activations, such as GPT-Neo-2.7B (Gao et al., 2020), OPT-2.7B (Zhang et al., 2022), BLOOM-3B (Press et al., 2022), and OpenLLaMA-3B (Touvron et al., 2023a).

4.2 Zero-Shot Evaluation

To assess the effectiveness of our condensation approach in a purely zero-shot setting, we evaluate `CD-MoE-S` and `CD-MoE-SR` on eight downstream tasks without any additional fine-tuning. As shown in Table 1, we report each model's performance alongside its memory reduction ratio. Because pruning different layers can

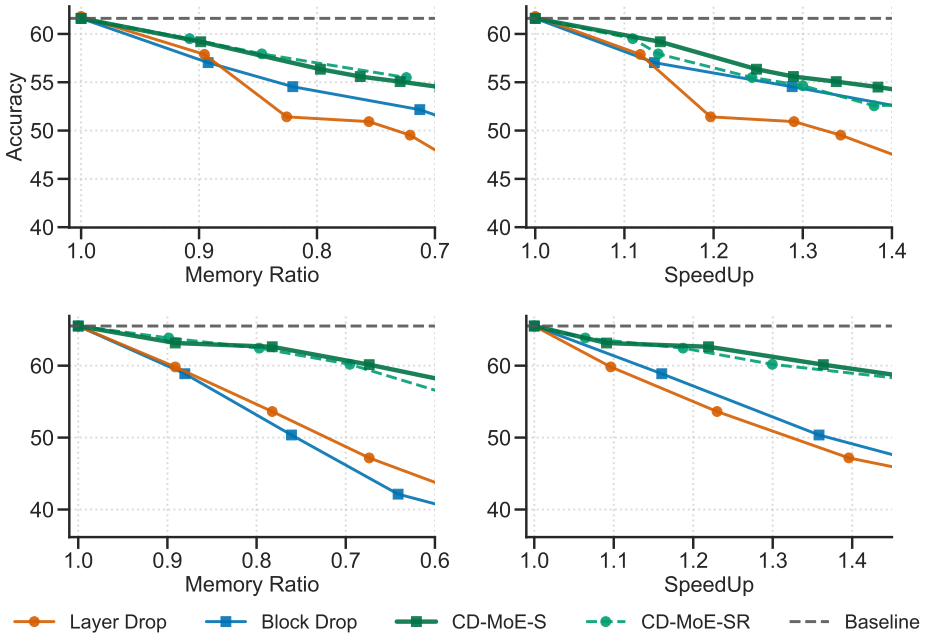

Figure 3: `CD-MoE` against baselines on zero-shot tasks w/o fine-tuning. Left: Average accuracy with varying Memory Ratio against the original model. Right: Average accuracy with varying SpeedUp against the original model. The gray dotted line is the original model result. `CD-MoE-S` represents the shared experts and no routing experts, and `CD-MoE-SR` represents shared with routing experts. Baseline indicated performance of the dense model.

lead to varying memory footprints, we normalize the number of pruned layers so that all methods operate under comparable memory budgets.

Specifically, ❶ **Comparison to Baselines:** Under similar memory constraints, both `CD-MoE-S` and `CD-MoE-SR` consistently outperform the baselines—*Block Drop*, *Layer Drop* and *M-SMoE*—highlighting the importance of preserving the most informative MoE experts. Moreover, Fig 3 plots the average accuracy against increasing pruning rates and memory savings, illustrating that the performance gap between `CD-MoE` and conventional pruning techniques widens as pruning becomes more aggressive. ❷ **Comparison to Small Dense LLMs:** Notably, even without fine-tuning, `CD-MoE-S` and `CD-MoE-SR` often surpass smaller dense LLMs of comparable size (e.g., BLOOM-3B, OpenLLaMA-3B). This suggests that training a dense model from scratch at a reduced scale does not match the performance of selectively condensed MoE architectures that retain high-value parameters. ❸ **`CD-MoE-S` vs. `CD-MoE-SR`:** In zero-shot mode, `CD-MoE-S` exhibits comparable or slightly superior performance to `CD-MoE-SR`. However, as detailed in Section 4.3, the routed experts in `CD-MoE-SR` confer a substantial advantage during fine-tuning: they enable nearly 98% of the original performance to be recovered in just a few hours on a single 80GB A100 GPU. The results for *Qwen1.5-MoE-A2.7B* are reported in A.4.

Overall, these results confirm that our condensation strategy preserves core model capacities more effectively than layer-level pruning alone. By selectively retaining expert parameters, `CD-MoE` consistently delivers strong zero-shot accuracy while substantially reducing memory and computational overhead.

4.3 Lightweight Fine-tuning Evaluation

We further experimented with expert fine-tuning and improving performance on specific tasks. Since the condensed model inevitably loses part of its language modeling capability, further fine-tuning can restore the model's abilities (Frantar & Alistarh, 2023; Shi et al., 2023; Sanh et al., 2020). Our goal is to demonstrate that fine-tuning can restore the capabilities of the condensed layers to their pre-condensation state, so we

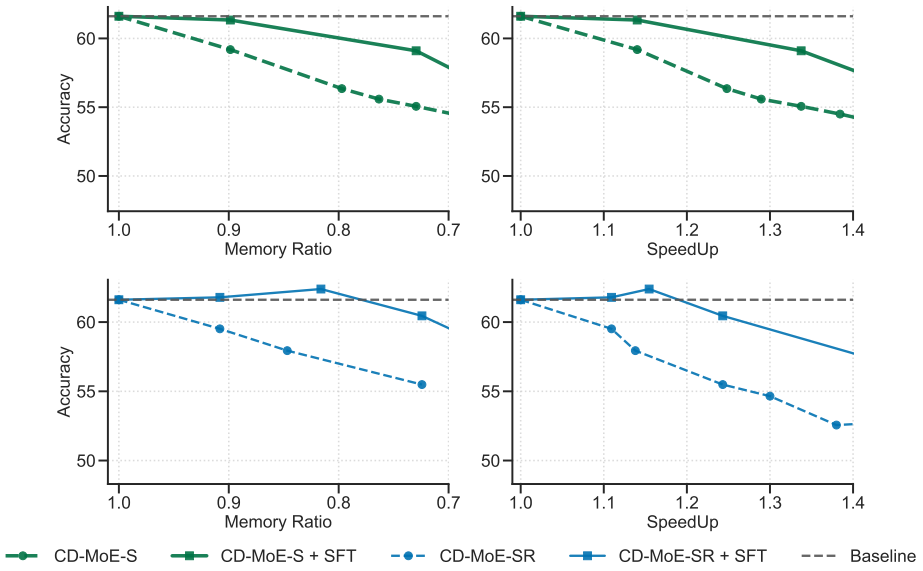

Figure 4: `CD-MoE` with lightweight fine-tuning. Left: SFT results on `CD-MoE-S` with increasing number of condensed layers. Right: SFT results on `CD-MoE-SR` with increasing number of condensed layers. Baseline indicated performance of the dense model.

propose a lightweight fine-tuning approach. Specifically, only the parameters in the condensed layers are updated to conduct efficient fine-tuning. For example, with `CD-MoE-SR`, only 3.8% of the parameters update gradients, and it takes just 2 hours on a single A100 GPU to perform SFT on the MMLU training set. For SFT, following the approach of Li et al. (2024a), we fine-tuned using commonsense170k and MMLU training set respectively.

Table 1 (bottom panel) summarizes the performance gains attributable to lightweight SFT. `CD-MoE-S` improves from an average score of 55.1 to 59.1, while `CD-MoE-SR` jumps from 55.5 to 60.5—nearly matching the original uncondensed performance of 61.6. Notably, ❶ **Commonsense Reasoning:** Task like *BoolQ* shows marked improvements. The `CD-MoE-SR + SFT` lifts performance by up to 7 points in compared to its counterpart before fine-tuning. ❷ **Complex Knowledge Tasks:** On *MMLU*, we observe significant gains (e.g., from 27.1 to 38.6 with `CD-MoE-SR + SFT`), indicating that even after pruning, the model retains foundational knowledge that can be effectively recovered through selective fine-tuning. ❸ **Efficiency:** Despite the notable performance boost, the computational overhead remains low because only a fraction of parameters ($\sim 3\%$) is updated. This design ensures training remains practical for resource-constrained settings, aligning with real-world demands.

Fig 4 shows how the average accuracy changes after SFT as more layers are condensed. Compared to `CD-MoE-S`, `CD-MoE-SR` delivers consistently higher accuracy with lower memory usage, suggesting that selecting additional routing experts via greedy search preserves more of the model's foundational knowledge. Moreover, `CD-MoE-SR` can restore and even surpass the original model's performance through lightweight SFT, reducing memory usage by up to 20%. These findings indicate that our condensation strategy retains recoverable knowledge despite significant pruning. With targeted lightweight SFT, the condensed experts regain most of their original performance, and in certain tasks, `CD-MoE-SR + SFT` even exceeds the unpruned baseline.

4.4 Experiment Analysis

Expert selection algorithms. We evaluate several strategies for selecting the most critical experts at each layer. Our primary baselines rely on easily computable statistical properties of the expert parameters:

- *Random*: Randomly select K experts.

Table 2: Comparison of expert selection methods. L_n denotes condensing n layers. Each entry is the average accuracy (%) on eight downstream tasks.

Methods	$\boldsymbol{L_6}$	$\boldsymbol{L_9}$	$\boldsymbol{L_{12}}$	$\boldsymbol{L_{15}}$
Random	54.6	52.7	50.1	46.8
`PL_Alpha_Hill`	**56.8**	52.5	50.3	47.3
L_1	56.5	53.2	50.4	47.5
Greedy Search	56.5	**55.5**	**52.8**	**49.0**

Table 3: Time cost comparison for expert and layer selection. '**-**' indicates that the method does not require the step, '*****' denotes estimates.

Method	**Expert Selection**		**Layer Selection**
	Inferences (Counts)	**Time (Hours)**	**Time (Hours)**
(Lu et al., 2024)	75,041,808	2500*	-
CD-MoE-S	-	-	0.12
CD-MoE-SR	369	0.13	0.15

- L_1: Calculate the sum of the L_1 norms of each expert's parameters, sort them, and select the K experts with the lowest L_1 norms.
- *PL_Alpha_Hill*: Following heavy-tailed self-regularization (Martin & Mahoney, 2019; 2020; 2021), we measure each layer's spectral density via the Hill estimator (Hill, 1975; Xiao et al., 2023):

$$\texttt{PL_Alpha_Hill}_\ell = 1 + k \Big/ \sum_{i=1}^{k} \ln\Big(\lambda_{n-i+1}^{\ell} \big/ \lambda_{n-k}^{\ell}\Big), \tag{6}$$

  where $\{\lambda_i^\ell\}$ are sorted eigenvalues and k is chosen via the Fix-Finger method Yang et al. (2023). A lower $\texttt{PL_Alpha_Hill}_\ell$ indicates a more heavy-tailed layer, so it is less likely to be pruned.

We also experimented with selecting experts having the highest L_1-norms, but observed substantially worse performance, potentially because lower norms often indicate better-converged parameters. In Table 2, we compare these methods to our Greedy Search, which iteratively selects experts that minimize the deviation between the post-condensation and pre-condensation outputs of the layer. Notably, none of the statistical baselines (*Random*, L_1, or *PL_Alpha_Hill*) outperform Greedy Search. We hypothesize that the latter's layer-level output fidelity objective directly retains the most impactful experts.

We also measure the time cost of our Greedy Search on the DeepSeekMoE-16B model and compare it with the exhaustive traversal in (Lu et al., 2024), which enumerates all expert combinations. Each experiment uses 100 C4 calibration samples with maximum sequence length 512 and batch size 256, running on an 80G A100 GPU. As shown in Table 3, the exhaustive approach requires an intractably large number of inferences and is therefore infeasible. By contrast, Greedy Search requires only a few hundred inferences per layer, striking a favorable balance between accuracy preservation and runtime efficiency.

Layer selection algorithms. We compare our proposed *Greedy Search (Layer)* method against two baselines that rank layers by measuring the divergence between the layer outputs before and after condensation:

- *Layer Rank*: For each layer, we compute the JS distance between its output before and after condensation. We then sort the layers and choose the one with the smallest divergence, indicating minimal impact on the layer-level output.
- *Global Layer Rank*: For each layer, we apply condensation and evaluate the JS distance between the model's final outputs before and after condensation. Layers are then sorted accordingly, and the layer that most preserves the final output distribution is preferred.

Table 4: Comparison of layer selection methods. L_n indicates the condensation of n layers. Entries are average accuracies (%) on eight test datasets.

Methods	L_6	L_9	L_{12}	L_{15}
Layer Rank	45.5	44.2	42.8	42.2
Global Layer Rank	55.7	53.1	48.8	45.9
Greedy Search	**56.5**	**55.5**	**52.8**	**49.0**

Table 5: Performance of different metrics for layer selection. L_n indicates the number of condensed layers. The values shown are average accuracies (%) on eight test datasets.

Metrics	$\boldsymbol{L_6}$	$\boldsymbol{L_9}$	$\boldsymbol{L_{12}}$	$\boldsymbol{L_{15}}$
PPL	**58.7**	55.4	50.9	47.2
KL divergence	57.1	54.7	52.2	48.5
JS divergence	56.5	**55.5**	**52.8**	**49.0**

Table 4 shows a comparison of these methods in the context of `CD-MoE-SR`. Notably, our Greedy Search consistently surpasses the baselines, underscoring its effectiveness in identifying layers that minimally disrupt the overall model output. We also measure the time needed to condense 15 of the 27 MoE layers using the same setup as in expert selection (see Table 3). This operation requires only 0.15 hours, further validating the computational efficiency of the `CD-MoE` approach.

Searching metrics. To quantify how well each condensed model approximates the original model's output, we also compare several metrics during the Greedy Search process. Following Zhang et al. (2024), we adopt JS divergence as our primary measure, as it has been shown to be more sensitive to output changes than angular distance or Euclidean distance. We compare JS against two alternative metrics:

- *Perplexity (PPL)*: For each calibration sample, we compute the change in PPL and select layers that minimize the average difference in PPL between the original and condensed models.
- *KL Divergence*: We measure the shift between the two output distributions using KL divergence and select layers with minimal KL distance.

Table 5 presents the results. While PPL achieves a marginally better score for L_6, JS divergence excels as more layers are condensed (L_9, L_{12}, L_{15}). These findings substantiate the choice of JS divergence as a robust metric for maintaining the model's overall quality under heavier compression.

Explanation of shared expert dominance. Our analysis of Table 1 reveals a noteworthy finding: `CD-MoE-S`, while retaining only shared experts in condensed layers, achieves comparable zero-shot performance

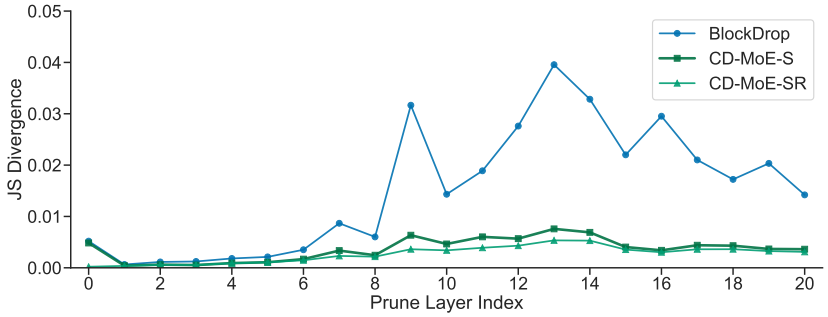

Figure 5: Layer-wise JS curves. The three curves represent the JS divergence between the outputs of a pruned layer and the outputs of the original layer, using different pruning methods.

to `CD-MoE-SR`, which preserves both shared and selected routed experts, significantly outperforming *Block Drop*. This suggests that shared experts alone may sufficiently preserve model capability.

To investigate this phenomenon, we quantitatively evaluated layer-wise output deviations between pruned and original models using JS divergence (Fig 5). The results demonstrate that `CD-MoE-S` induces minimal output distortion, closely approximating `CD-MoE-SR`'s behavior. In contrast, *Block Drop*'s aggressive layer removal causes substantially greater output deviation, explaining its inferior performance.

5 Conclusion

We have presented `CD-MoE`, a novel framework for compressing large-scale Mixture-of-Experts (MoE) models by selectively condensing them into denser layers. Our approach removes the token routing mechanism and uses a greedy search algorithm to prune the majority of less significant experts, allowing all tokens to activate only a handful of highly impactful experts. Extensive experiments on DeepSeekMoE-16B and other fine-grained MoE architectures show that `CD-MoE` can retain up to 90% of the original model's accuracy while enjoy 27.5% memory reduction and a 1.26× inference speed. Moreover, we demonstrate that lightweight expert fine-tuning, focused solely on the condensed layers, can further reclaim the original model's performance within a few hours on a single GPU. These results highlight the practical benefits of `CD-MoE` for resource-constrained scenarios, where memory and computational efficiency are critical. Future directions include exploring the synergy of `CD-MoE` with quantization and knowledge distillation, and extending the method to more diverse MoE structures lacking shared experts.

References

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020. URL https://arxiv.org/abs/2006.11477.

Abhinav Bandari, Lu Yin, Cheng-Yu Hsieh, Ajay Kumar Jaiswal, Tianlong Chen, Li Shen, Ranjay Krishna, and Shiwei Liu. Is c4 dataset optimal for pruning? an investigation of calibration data for llm pruning. *arXiv preprint arXiv:2410.07461*, 2024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

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. On the representation collapse of sparse mixture of experts, 2022. URL https://arxiv.org/abs/2204.09179.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions, 2019a. URL https://arxiv.org/abs/1905.10044.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does bert look at? an analysis of bert's attention, 2019b. URL https://arxiv.org/abs/1906.04341.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognizing textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment*, 2005.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. 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. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *CoRR*, abs/2401.06066, 2024. URL https://arxiv.org/abs/2401.06066.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2022. URL https://arxiv.org/abs/2101.03961.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot, 2023. URL https://arxiv.org/abs/2301.00774.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

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. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

Haorui He, Zengqiang Shang, Chaoren Wang, Xuyuan Li, Yicheng Gu, Hua Hua, Liwei Liu, Chen Yang, Jiaqi Li, Peiyang Shi, Yuancheng Wang, Kai Chen, Pengyuan Zhang, and Zhizheng Wu. Emilia: An extensive, multilingual, and diverse speech dataset for large-scale speech generation, 2024a. URL https://arxiv.org/abs/2407.05361.

Shwai He, Daize Dong, Liang Ding, and Ang Li. Demystifying the compression of mixture-of-experts through a unified framework, 2024b. URL https://arxiv.org/abs/2406.02500.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Bruce M Hill. A simple general approach to inference about the tail of a distribution. *The annals of statistics*, pp. 1163–1174, 1975.

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. Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

Pengxiang Li, Lu Yin, Xiaowei Gao, and Shiwei Liu. Owlore: Outlier-weighed layerwise sampled low-rank projection for memory-efficient llm fine-tuning, 2024a. URL https://arxiv.org/abs/2405.18380.

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. Merge, then compress: Demystify efficient smoe with hints from its routing policy, 2024b. URL https://arxiv.org/abs/2310.01334.

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

Charles H Martin and Michael W Mahoney. Traditional and heavy-tailed self regularization in neural network models. *arXiv preprint arXiv:1901.08276*, 2019.

Charles H Martin and Michael W Mahoney. Heavy-tailed universality predicts trends in test accuracies for very large pre-trained deep neural networks. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pp. 505–513. SIAM, 2020.

Charles H Martin and Michael W Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *Journal of Machine Learning Research*, 22(165): 1–73, 2021.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260.

Alexandre Muzio, Alex Sun, and Churan He. Seer-moe: Sparse expert efficiency through regularization for mixture-of-experts, 2024. URL https://arxiv.org/abs/2404.05089.

Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation, 2022. URL https://arxiv.org/abs/2108.12409.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in neural information processing systems*, 33:20378–20389, 2020.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL https://arxiv.org/abs/1701.06538.

Ensheng Shi, Yanlin Wang, Hongyu Zhang, Lun Du, Shi Han, Dongmei Zhang, and Hongbin Sun. Towards efficient fine-tuning of pre-trained code models: An experimental study and beyond, 2023. URL https://arxiv.org/abs/2304.05216.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models, 2024. URL https://arxiv.org/abs/2306.11695.

Gemini Team, M Reid, N Savinov, D Teplyashin, Lepikhin Dmitry, T Lillicrap, JB Alayrac, R Soricut, A Lazaridou, O Firat, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. in arxiv [cs. cl]. arxiv, 2024.

Qwen Team. Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters", February 2024. URL https://qwenlm.github.io/blog/qwen-moe/.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023b. URL https://arxiv.org/abs/2302.13971.

Xuanzhe Xiao, Zeng Li, Chuanlong Xie, and Fengwei Zhou. Heavy-tailed regularization of weight matrices in deep neural networks. In *International Conference on Artificial Neural Networks*, pp. 236–247. Springer, 2023.

Yaoqing Yang, Ryan Theisen, Liam Hodgkinson, Joseph E Gonzalez, Kannan Ramchandran, Charles H Martin, and Michael W Mahoney. Test accuracy vs. generalization gap: Model selection in nlp without accessing training or testing data. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3011–3021, 2023.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Jaiswal, Mykola Pechenizkiy, Yi Liang, Michael Bendersky, Zhangyang Wang, and Shiwei Liu. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity, 2024. URL https://arxiv.org/abs/2310.05175.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022. URL https://arxiv.org/abs/2205.01068.

Yang Zhang, Yawei Li, Xinpeng Wang, Qianli Shen, Barbara Plank, Bernd Bischl, Mina Rezaei, and Kenji Kawaguchi. Finercut: Finer-grained interpretable layer pruning for large language models, 2024. URL https://arxiv.org/abs/2405.18218.

A Appendix

A.1 Fluctuations in Metrics across Condensing Different Layers

In Fig 1, we show that the JS divergence between the model's final output and the output before condensing varies substantially across layers. Fig A.1 further demonstrates how KL divergence and perplexity (PPL) change when pruning different layers. Notably, condensing layer 0 continues to produce a significant shift in the model's outputs, while the middle layers exhibit fluctuations, and the final layers show an increase. Across all layers, the trends in JS, KL, and PPL remain similar.

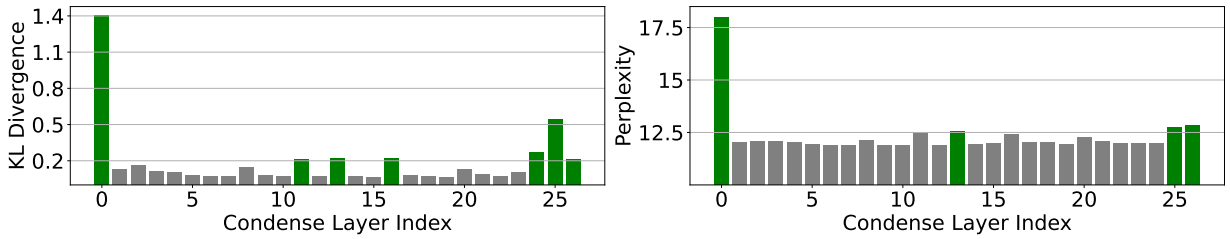

Figure A.1: Left: fluctuations in the KL divergence. Right: fluctuations in the perplexity.

A.2 Implementation Details

We utilize Hugging Face and PyTorch for the implementation of our work [1]. All inference and fine-tuning operations are executed using bf16 (Brain Floating Point 16) precision on NVIDIA A100 GPU equipped with 80GB of memory. During the fine-tuning phase, we employ an initial maximum learning rate of 1×10^{-4}, implement a warm-up ratio of 10% to gradually ramp up the learning rate, and utilize cosine annealing as the learning rate scheduler to ensure smooth convergence and prevent potential overfitting. Due to the large computational requirements and our limited resources, all experiments were conducted in a single run.

The expert configurations of *DeepSeekMoE-16B*, *Qwen1.5-MoE-A2.7B* and *Qwen2-MoE-57B-A14B*is represented at Table A.1.

Table A.1: Expert configurations in MoE models

Model	**DeepSeekMoE-16B**	**Qwen1.5-MoE-A2.7B**	**Qwen2-MoE-57B-A14B**
Parameters	16,375,728,128	14,315,784,192	57,421,636,608
Activated Parameters	2,828,650,496	2,689,177,536	14,249,346,048
Number of Experts	66	64	72
Number of Activated Experts	8	8	16
Ratio of Activated Experts	12.1%	12.5%	22.2%
Number of Shared Experts	2	4	8
Ratio of Shared Experts in Activated Experts	25%	50%	50%

A.3 Calibration Data Analysis

C4 dataset commonly serves as the calibration benchmark. In our study, we aim to ensure that the performance of the condensed model aligns closely with specific task requirements. To this end, we experiment by utilizing questions extracted from multiple test sets as calibration data, thereby tailoring the language modeling capabilities of the condensed model to the tasks at hand. Specifically, we sampled 20 questions from each of eight distinct test sets, resulting in a total of 160 questions.

Table A.2: Performance comparison across different calibration data across different layer pruning indexes. L_n indicates the condensation of n layers. Entries represent evaluation metrics (%).

Methods	L_3	L_6	L_9	L_{12}	L_{15}
DownStream Data	59.5	56.1	55.3	51.9	48.7
Audio Token	59.0	**56.7**	55.0	52.0	**49.1**
C4	**59.5**	56.5	**55.5**	**52.8**	49.0

As demonstrated in Fig A.2, it is noteworthy that across varying numbers of condensed layers, the C4 data consistently achieves superior performance compared to the downstream tasks data. This suggests that the diversity and comprehensiveness of the C4 dataset provide a more robust foundation for calibrating the condensed model. Consequently, we adopt C4 as the calibration data in all experiments. This finding

[1]Our repository is built on top of Transformers: https://github.com/huggingface/transformers

Table A.3: Comparison on Qwen1.5-MoE-A2.7B. The underlined variables are tried to be kept consistent for fair comparison, and the **bolds** represent the best results. '#*L*' represents the number of pruned layers; 'ACT.' represents the number of activated parameters; and 'MEM.' is the remaining memory ratio compared to the original model. `CD-MoE-S` refers to retaining only the **S**hared expert after condensing, while `CD-MoE-SR` means selecting additional 4 **R**outed experts and keeping as dense.

method	**#L**	**ACT.**	**MEM.**	**Speedup**	ARC-C	BOOLQ	HELLA	MMLU	OBQA	PIQA	RTE	WINO	**AVG.**
GPT-NEO-2.7B	-	2.7B	-	-	29.8	61.7	55.2	25.0	34.6	72.9	53.4	57.8	48.8
OPT-2.7B	-	2.7B	-	-	31.2	59.9	60.6	25.4	35.2	74.7	54.2	60.5	50.2
BLOOM-3B	-	3B	-	-	30.4	61.4	54.5	25.9	32.4	70.7	56.0	58.6	48.7
OPENLLAMA-3B	-	3B	-	-	36.1	67.1	64.4	23.9	38.6	75.1	54.2	62.3	52.7
UNPRUNED		2.7B	100%	1.0X	45.0	79.5	77.3	61.2	43.6	80.3	67.9	69.3	65.5
						W/O SFT							
LAYERDROP	6	2.3B	78.3%	1.23X	34.4	62.6	63.1	45.4	32.8	73.6	54.5	62.6	53.6
BLOCKDROP	6	2.2B	76.1%	1.36X	31.3	62.2	57.4	38.4	32.0	70.7	52.7	58.2	50.4
`CD-MoE-S`	6	2.5B	78.3%	1.22X	41.1	75.8	72.9	56.0	41.0	78.9	67.2	68.1	**62.6**
`CD-MoE-SR`	6	2.7B	79.7%	1.19X	40.2	77.2	72.1	54.9	39.4	78.0	71.1	66.4	62.4

is consistent with the work by Bandari et al. (2024), where they perform unstructured pruning of LLM parameters, while we focus on layer condensation.

Additionally, we probed more extreme domain shifts by using small audio-domain samples from the open-source Emilia dataset(He et al., 2024a), processed through wav2vec(Baevski et al., 2020) for calibration. The accuracy remained broadly comparable to the C4-based results, as reported for the average accuracy of DeepSeekMoE-16B on commonsense reasoning and MMLU. This demonstrates that `CD-MoE` is not highly dependent on a specific calibration dataset.

A.4 Zero-shot Evaluation on Qwen1.5-MoE-A2.7B

Table A.3 presents the results of `CD-MoE` on the Qwen1.5-MoE-A2.7B model. Under similar memory costs, the advantages of `CD-MoE` over baseline methods are more pronounced, and with more than a 20% reduction in memory usage, it retains approximately 96% of the original model's performance. We attribute this to the fact that Qwen1.5-MoE-A2.7B has a larger proportion of shared experts, as shown in Table 5. Our method preserves these experts, enabling the condensed model to retain more knowledge acquired during the pre-training phase.

It should be noted that although `CD-MoE-S` slightly outperforms `CD-MoE-SR` in a zero-shot setting, after lightweight fine-tuning, `CD-MoE-SR` better unlocks the model's inherent capabilities, as illustrated in Fig 4. This suggests that expert selection remains meaningful in the Qwen1.5-MoE-A2.7B model.

Table A.4: Comparison on Qwen2-MoE-57B-A14B. The underlined variables are tried to be kept consistent for fair comparison, and the **bolds** represent the best results. '#L' represents the number of pruned layers; 'ACT.' represents the number of activated parameters; and 'MEM.' is the remaining memory ratio compared to the original model. `CD-MoE-S` refers to retaining only the **S**hared expert after condensing, while `CD-MoE-SR` means selecting additional 8 **R**outed experts and keeping as dense.

method	**#L**	**ACT.**	**MEM.**	**Speedup**	ARC-C	BOOLQ	HELLA	OBQA	PIQA	RTE	**AVG.**
UNPRUNED	-	14.2B	100%	1.0x	55.6	88.2	82.3	45.6	82.0	76.9	71.8
					W/O SFT						
LayerDrop	6	11.6B	<u>79.3%</u>	1.24x	42.9	77.3	69.0	37.6	76.0	69.7	62.1
BlockDrop	6	11.4B	<u>79.0%</u>	1.35x	28.3	61.3	51.6	32.6	62.7	59.9	49.7
`CD-MoE-S`	6	12.9B	<u>81.6%</u>	1.22x	54.0	86.7	79.1	43.4	81.5	78.5	**70.5**
`CD-MoE-SR`	6	14.2B	<u>83.9%</u>	1.20x	54.1	86.6	78.6	43.0	80.7	80.2	**70.5**

A.5 Zero-shot Evaluation on Qwen2-MoE-57B-A14B

Table A.4 presents the results of `CD-MoE` on the Qwen2-MoE-57B-A14B model. This model has a structure similar to Qwen1.5-MoE-A2.7B but has four times the number of parameters and active parameters. Due to the larger size of this model, which is constrained by computational resources, we used a 4-bit version for inference and evaluated it on six test sets. `CD-MoE` was able to maintain an average accuracy of **98%** while saving approximately **20%** of the memory, demonstrating that our method can generalize to larger models.