Dropping Experts, Recombining Neurons: Retraining-Free Pruning for Sparse Mixture-of-Experts LLMs

Yixiao Zhou[1,2], Ziyu Zhao[1,2], Dongzhou Cheng[2,3], Zhiliang Wu[1], Jie Gui[3], Yi Yang[1], Fei Wu[1,4], Yu Cheng[2,5]*, Hehe Fan[1]*

[1]Zhejiang University, [2]Shanghai Innovation Institute, [3]Southeast University, [4]Shanghai AI Laboratory, [5]The Chinese University of Hong Kong

{12421181,ziyuzhao.cs,wu_zhiliang,yangyics,wufei,hehefan}@zju.edu.cn, {230249457,guijie}@seu.edu.cn, {chengyu}@cse.cuhk.edu.hk

Abstract

Sparse Mixture-of-Experts (SMoE) architectures are widely used in large language models (LLMs) due to their computational efficiency. However, though only a few experts are activated for each token, SMoE still requires loading all expert parameters, leading to high memory usage and challenges in deployment. Previous work has tried to reduce the overhead by pruning and merging experts, but primarily focused on expert-level operations, leaving neuron-level structure underexplored. We propose **DERN** (**D**ropping **E**xperts, **R**ecombining **N**eurons), a task-agnostic and retraining-free framework for expert pruning and reconstruction. We observe that experts are often misaligned and contain semantic conflicts at the neuron level, which poses challenges for direct merging. To solve this, DERN works in three steps: it first prunes redundant experts using router statistics; then it decomposes them into neuron-level expert segments, assigning each segment to its most compatible retained expert; and finally, it merges segments within each retained expert to build a compact representation. Experiments on Mixtral, Qwen, and DeepSeek SMoE models show that DERN achieves over a 5% performance gains than previous methods on commonsense reasoning and MMLU benchmarks under 50% expert sparsity, without extra training. It also greatly reduces the number of experts and memory usage, making SMoE LLMs easier to deploy in practice.

1 Introduction

Large Language Models (LLMs) have become the foundation models of modern NLP (Brown et al., 2020), with their application rapidly expanding into diverse domains such as multimodal learning (Zhang et al., 2025; Li et al., 2024b,c,a), scientific research (Wang et al., 2025; Hu et al., 2025), and advanced reasoning (Dong and Fan, 2025).

Table 1: Comparison of our method with other expert pruning methods. E-Merge means expert-level merge, while **N-Merge** means neuron-level merge.

Method	Task-Agnostic	Train-Free	Strategy
TSEP (Chen et al., 2022)	✗	✗	Prune
MoE-I^2 (Yang et al., 2024b)	✓	✗	Prune
NAEE (Lu et al., 2024a)	✓	✓	Prune
HC-SMoE (Chen et al., 2024)	✓	✓	E-Merge
MC-SMoE (Li et al., 2023)	✓	✗	E-Merge
DERN (Ours)	✓	✓	**N-Merge**

This widespread use, however, makes their ever-growing scale a serious challenge for efficient deployment. Sparse Mixture-of-Experts (SMoE) architectures alleviate inference cost by activating only a subset of experts per token (Shazeer et al., 2017; Fedus et al., 2022), matching or surpassing dense models in performance. However, their total parameter footprint remains massive. For instance, DeepSeek-V3 (Liu et al., 2024a) activates only 37B parameters per pass, yet requires storing 671B in memory. Reducing this overhead without compromising performance remains an open challenge.

Recent work has shown that MoE layers exhibit significant redundancy: experts contribute unequally to downstream predictions (Chi et al., 2022), and many share high parameter similarity (Lo et al., 2024), indicating overlapping functional capacity and structural duplication. These findings motivate expert-level sparsification as a promising direction. Early approaches, such as (Chen et al., 2022) prune experts progressively during fine-tuning on a specific task, but suffer from high training overhead. Subsequent approaches shifted toward retraining-free and task-agnostic pruning. For example, (Lu et al., 2024a) searches for optimal expert subsets using the output discrepancy, while (He et al., 2024) leverages routing statistics for large-scale pruning. However, the abrupt removal of expert parameters can significantly impair the model's capabilities.

To mitigate such loss, a parallel line of work explores expert merging as a post-pruning remedy.

*Corresponding authors.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 15169–15186
November 4-9, 2025

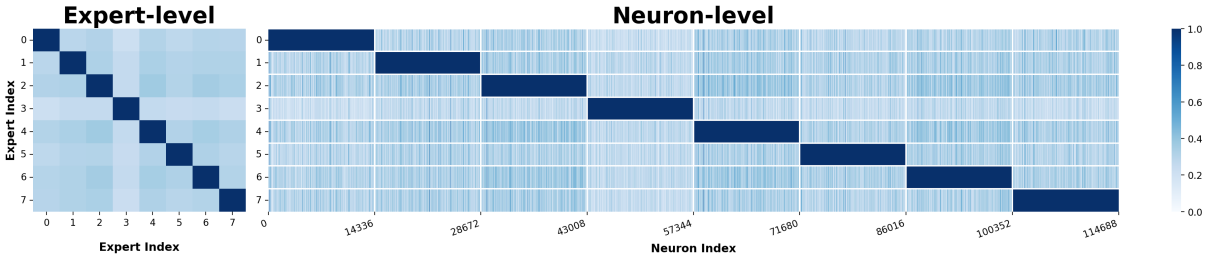

Figure 1: Expert-level and neuron-level cosine similarity in layer 15 of Mixtral-8×7B-Instruct. The left plot shows expert-to-expert similarity; the right plot depicts how strongly a neuron aligns with the most similar neuron in each expert. Despite some global expert alignment (left), clear inconsistencies remain at the neuron level (right).

Representative methods (Li et al., 2023; Chen et al., 2024; Liu et al., 2024b) group or cluster experts followed by weighted averaging. However, these approaches merge at the whole-expert level, often overlooking that neuron arrangements across experts are inherently unaligned (Ainsworth et al., 2022). Even with alignment mechanisms (Li et al., 2023), similar experts may encode distinct or even conflicting internal representations, due to divergent learned semantics (Zhao et al., 2024, 2025; Ruan et al., 2025). Such structural and semantic mismatches limit the effectiveness of direct expert fusion methods and can introduce nontrivial degradation in merged models.

To probe this issue, we visualize the similarity at the expert level and the neuron level for Mixtral (Jiang et al., 2024) in Fig. 1. The results reveal that, while some experts exhibit global alignment in their structural patterns, significant inconsistencies remain at the neuron level. This indicates that expert merging is feasible in principle, but naive whole-expert averaging may fail to preserve neuron-level consistency. Is it possible to enable expert merge through neuron-level, structure-aware recombination of transferable components?

In this paper, we introduce **DERN** (**D**ropping **E**xperts, **R**ecombining **N**eurons), a new paradigm for expert pruning and reconstruction, grounded in segment-based modularity and recomposition. At the core of DERN is the notion of an ***Expert Segment***: a minimal functional triplet composed of corresponding rows from the gate and up-projection matrices and a column from the down-projection matrix, as detailed in Sec. 3.2. Then each expert is viewed as a collection of such segments.

As illustrated in Fig. 2, DERN proceeds in three stages: (1) It identifies and prunes redundant experts based on routing activation statistics; (2) It decomposes pruned experts into segments, pools them, and reassigns each segment to the most compatible retained expert based on local structural similarity; (3) It applies spherical weighted k-Means clustering (Dhillon and Modha, 2001) to merge segments within each retained expert, using cluster centroids to reconstruct more compact yet expressive experts with significantly fewer parameters.

Unlike previous whole-expert weighted averaging (Li et al., 2023; Chen et al., 2024; Liu et al., 2024b), DERN reframes pruning as a segment-based decomposition and recombination problem. This modular view enables flexible, fine-grained knowledge transfer across experts, breaking the rigidity of fixed expert partitions, and enhancing the overall expressiveness of pruned models. Our main contributions are summarized as follows:

- We reformulate expert merging as a segment-based decomposition and recombination problem, enabling structure-aware, neuron-level operation that surpasses conventional coarse-grained averaging.
- We propose **DERN**, a unified, task-agnostic, and retraining-free pruning framework that restructures experts through a multi-stage pipeline of identification, segments decomposition, adaptive recombination, and cluster-based reconstruction.
- We conduct extensive experiments on Mixtral, Qwen, and DeepSeek SMoE models across multiple commonsense reasoning datasets and the comprehensive MMLU benchmark, demonstrating that DERN achieves over a 5% performance gains than previous methods under 50% expert sparsity.

2 Related Work

2.1 Sparse Mixture-of-Experts

Sparse Mixture-of-Experts architectures have emerged as a key paradigm for scaling large lan-

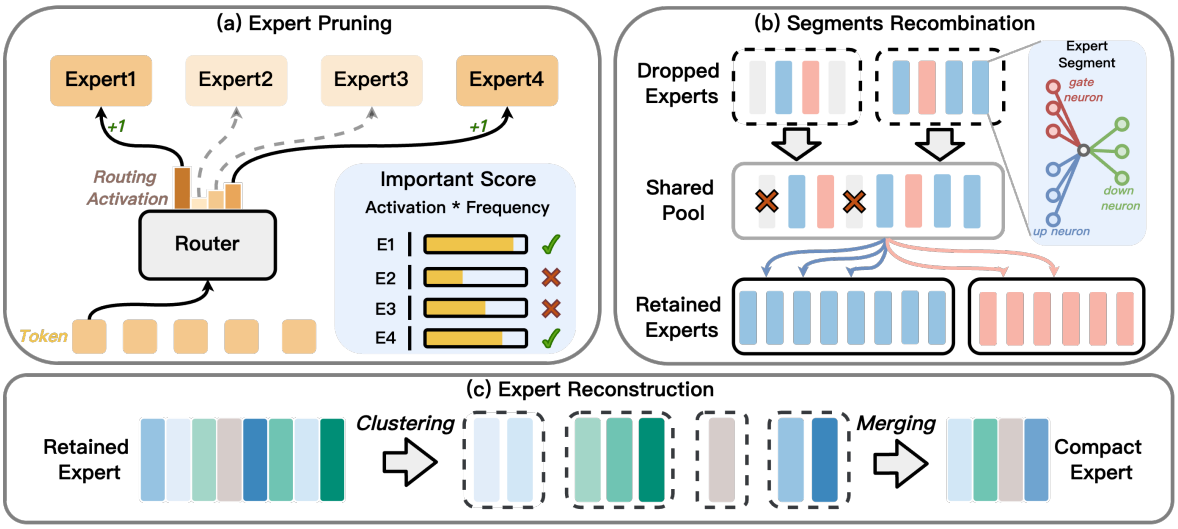

Figure 2: Overview of the **DERN** framework. (a) Redundant experts are pruned based on routing activation statistics. (b) Dropped experts are decomposed into neuron-level segments, each represented by a triplet of gate, up, and down projection vectors. These segments are then reassigned to retained experts based on local structural similarity. (c) Within each retained expert, reassigned and original segments are clustered via spherical weighted k-means to form a compact, performance-preserving expert.

guage models efficiently, offering reduced inference cost by activating only a small subset of subnetworks (experts) per token (Shazeer et al., 2017; Fedus et al., 2022). In SMoEs, the dense feedforward layers in Transformers are replaced with multiple parallel experts, typically feedforward networks (FFNs), governed by a trainable gating mechanism that dynamically selects a small number of experts for each token. This enables a substantial increase in total model capacity while maintaining inference cost proportional to the number of active experts. SMoEs have been adopted in many recent LLMs, including Mixtral (Jiang et al., 2024), DeepSeek-MoE (Dai et al., 2024; Liu et al., 2024a), and Qwen-MoE (Yang et al., 2024a).

2.2 Expert Pruning and Merging

Expert pruning and merging, as a SMoE compression technique, have gained growing interest. Expert pruning is motivated by observations of expert redundancy and contribution imbalance (Chi et al., 2022; Lo et al., 2024). Early methods such as (Chen et al., 2022) involve fine-tuning and incur high training cost. More recent work explores retraining-free, task-agnostic approaches based on reconstruction loss (Lu et al., 2024a), routing statistics (He et al., 2024), or changes in router norms (Chowdhury et al., 2024). (Yang et al., 2024b) further considers internal sparsity within experts. However, pruning entire experts can result in irreversible loss of important representations.

Expert merging, a technique within model merging (Yang et al., 2025), retains the knowledge of pruned experts to mitigate performance loss. Representative methods typically perform expert grouping or clustering followed by weighted expert averaging. For example, (Chen et al., 2024) uses hierarchical clustering based on expert output similarity; (Liu et al., 2024b) employs evolutionary strategies to iteratively merge similar experts; and (Li et al., 2023) identifies dominant experts based on routing patterns to guide grouping and merging. However, most existing merging strategies operate at the expert level, overlooking neuron-level misalignment and potential semantic conflicts. This motivates finer-grained recomposition strategies, which we address with our proposed DERN framework. Tab. 1 summarizes the differences between our method and existing approaches.

Other SMoE compression approaches include weight pruning within experts (Xie et al., 2024; Liang et al., 2025), layer compression (Cao et al., 2024; Lu et al., 2024b; Li et al., 2024d), low-rank decomposition (Gu et al., 2025), and quantization (Huang et al., 2024), alongside general pruning methods (Ma et al., 2023; Lu et al., 2024c).

3 Proposed Method

Our DERN framework performs expert pruning and merge through a three-stage process, as illus-

trated in Fig. 2. First, we identify and retain the top-k experts based on routing activations (Sec. 3.1). Next, pruned experts are decomposed into neuron-level segments and reassigned to retained experts (Sec. 3.2). Finally, each expert merges its segments through clustering to reconstruct a compact structure (Sec. 3.3). The overall procedure is summarized in Alg. 1.

3.1 Expert Pruning via Routing Behaviors

We identify redundant experts in SMoE layers based on routing behaviors observed over a calibration set, and select them as pruning candidates.

Let an SMoE layer consist of N experts $\mathcal{E} = \{E_1, \ldots, E_N\}$ and a routing network $G : \mathbb{R}^d \to \mathbb{R}^N$, which outputs soft routing weights $G(x) = \big(G_1(x), \ldots, G_N(x)\big)$ for each token feature x. Given a calibration set $\mathcal{C} = \{x_m\}_{m=1}^M$, let $\mathcal{A}(x_m) \subseteq \{1, \ldots, N\}$ denote the index set of the top-k experts selected for x_m by the router.

We define the ***importance score*** for E_i as:

$$S_i = \mathbb{E}_{x_m \sim \mathcal{C}} \left[\frac{\mathbb{I}[i \in \mathcal{A}(x_m)] \cdot G_i(x_m)}{\sum_{j \in \mathcal{A}(x_m)} G_j(x_m)} \right], \quad (1)$$

where $\mathbb{I}[\cdot]$ is the indicator function. This score reflects both the frequency and the strength of activation, effectively measuring each expert's utility under realistic input distributions.

After computing $\{S_i\}_{i=1}^N$, we identify the top-k experts with the highest importance scores as retained experts, while designating the remaining $N - k$ as pruning candidates.

3.2 Expert Decomposition and Segment Recombination

We reformulate the pruning task as a modular decomposition and recombination problem. Instead of discarding entire experts, we decompose them into smaller functional units called segments, which can be selectively reassigned to compatible retained experts. Segments from pruned experts are collected into a shared pool and reallocated based on local structural similarity. This approach facilitates neuron-level knowledge transfer while preserving the functional coherence of each expert.

Expert-to-Segments Decomposition. We begin by formalizing the decomposition of a standard SMoE expert. The current mainstream LLMs adopt the Gated Linear Unit (GLU) architecture for MLP layers, where each expert is represented by three weight matrices: $W_g \in \mathbb{R}^{h \times d}$, $W_u \in \mathbb{R}^{h \times d}$, and $W_d \in \mathbb{R}^{d \times h}$, where d is the input dimension and h the intermediate hidden size. Given an input token $x \in \mathbb{R}^d$, the forward computation is expressed as:

$$f(x) = W_d \left(\sigma(W_g x) \odot (W_u x) \right), \quad (2)$$

which can be rewritten as a sum of independent contributions from each hidden dimension:

$$f(x) = \sum_{i=1}^{h} w_{d,i} \cdot \left[\sigma(w_{g,i}^\top x) \cdot (w_{u,i}^\top x) \right], \quad (3)$$

where $\sigma(\cdot)$ is the activation function, $w_{g,i}^\top$ and $w_{u,i}^\top$ represent the i-th rows of W_g and W_u respectively, and $w_{d,i}$ represents the i-th column of W_d.

As shown in Eq. (3), the expert output is a sum of functionally independent, low-rank transformations. This decomposition motivates our definition of an ***Expert Segment***: a minimal self-contained unit governed by the parameter triplet $(w_{g,i}, w_{u,i}, w_{d,i})$. Specifically, the i-th segment is:

$$\text{seg}_i = (w_{g,i}, w_{u,i}, w_{d,i}). \quad (4)$$

Segments exhibit three key properties that make them well-suited for recombination. First, they are functionally independent, each contributing to a single output dimension. Second, they maintain gradient locality, with gradients depending solely on their own parameters. Third, their structural regularity allows each triplet to be flattened into a fixed-dimensional vector, enabling efficient similarity comparison in parameter space. Leveraging these properties, we decompose both retained and pruned experts into segments, aggregating those from pruned experts into a global segment pool $\mathcal{P}$ for reassignment.

Segment Recombination via local structural similarity. To reassign segments from $\mathcal{P}$ to appropriate retained experts, we use a similarity-based matching scheme. Let $\text{seg}_i \in \mathcal{P}$ be a candidate segment and E_r a retained expert with segment set $\mathcal{S}_r$. Each segment is vectorized as:

$$v(\text{seg}_i) = \begin{bmatrix} w_{g,i} \\ w_{u,i} \\ w_{d,i} \end{bmatrix} \in \mathbb{R}^{3d}, \quad (5)$$

where each component corresponds to a row or column from the expert's gate, up, and down projection matrices, respectively.

The local structural similarity between seg_i and E_r is defined as the maximum cosine similarity

Algorithm 1 The Overall Procedure of DERN

Input: $\mathcal{E} = \{E_1, \ldots, E_N\}$, $\mathcal{C}$, k, α
Output: $\mathcal{E}_r$

Stage 1: Expert Pruning
for $i = 1$ **to** N **do**
  $S_i \leftarrow \mathbb{E}_{x_m \sim \mathcal{C}} \left[\frac{\mathbb{I}[i \in \mathcal{A}(x_m)] \cdot g_i(x_m)}{\sum_{j \in \mathcal{A}(x_m)} g_j(x_m)} \right]$
end for
$\mathcal{E}_r \leftarrow \{E_i \mid S_i \in \text{Top-}k(S_1, \ldots, S_N)\}$
Stage 2: Decomposition, Recombination
$\mathcal{P} \leftarrow \bigcup_{E_i \in \mathcal{E}_r} \text{Decompose}(E_i)$
for $\text{seg}_i \in \mathcal{P}$ **do**
  $E_r^* \leftarrow \arg\max_{E_r \in \mathcal{E}_r} \text{sim}(\text{seg}_i, E_r)$
  if $\text{sim}(\text{seg}_i, E_r^*) > \alpha$ **then**
    $E_r^* \leftarrow E_r^* \cup \text{seg}_i$
  end if
end for
Stage 3: Expert Reconstruction
for each $E_r \in \mathcal{E}_r$ **do**
  $E_r' \leftarrow \text{Cluster}(\mathcal{S}_r)$
end for
return $\mathcal{E}_r$

with any segment already in E_r:

$$\text{sim}(\text{seg}_i, E_r) = \max_{s' \in \mathcal{S}_r} \frac{\langle v(\text{seg}_i), v(s') \rangle}{\|v(\text{seg}_i)\| \|v(s')\|}. \quad (6)$$

For each segment $\text{seg}_i \in \mathcal{P}$, we first identify the most similar retained expert $E_r^* = \arg\max_{E_r \in \mathcal{E}_r} \text{sim}(\text{seg}_i, E_r)$. The segment is then reassigned to this expert only if the similarity score exceeds the threshold α:

$$E_r^* \leftarrow E_r^* \cup \{\text{seg}_i\} \quad \text{if} \quad \text{sim}(\text{seg}_i, E_r^*) > \alpha. \quad (7)$$

This local structural similarity criterion serves as a soft gating mechanism, filtering out structurally incompatible segments while maintaining coherent feature composition within each expert.

To preserve routing semantics, we softly transfer the routing contribution from E_o to E_r, scaled by the number of segments in E_o:

$$\text{G}_{E_r} \leftarrow \text{G}_{E_r} + \frac{1}{n} \cdot \text{G}_{E_o}, \quad (8)$$

where $n = |\mathcal{S}_{E_o}|$ is the number of original segments in expert E_o (*i.e.*, its intermediate size).

3.3 Expert Reconstruction via Segment Clustering

After reassignment, each retained expert E_r is associated with a unified set of neuron segments:

$$E_r \to \mathcal{S}_r = \mathcal{S}_r^{\text{int}} \cup \mathcal{S}_r^{\text{ext}}, \quad (9)$$

where $\mathcal{S}_r^{\text{int}}$ contains the expert's original segments, and $\mathcal{S}_r^{\text{ext}}$ includes segments reassigned from pruned experts based on local structural similarity. To reduce redundancy while preserving diversity, we apply spherical weighted k-Means clustering to compress this combined segment set into a smaller, semantically coherent set of representative neurons.

We minimize a weighted cosine-based clustering objective over the segment set, using cluster centers $\mathcal{C} = \{c_1, \ldots, c_k\}$, where k is the target hidden dimension of the reconstructed expert:

$$\min_{\mathcal{C}} \sum_{j=1}^{k} \sum_{\text{seg}_i \in \mathcal{S}_j} w_i \left(1 - \frac{v(\text{seg}_i)^\top c_j}{\|v(\text{seg}_i)\| \|c_j\|} \right), \quad (10)$$

where $v(\text{seg}_i)$ is the vectorized form of seg_i in Eq. (5), and w_i reflects the importance of seg_i, derived from its source expert's importance score.

To initialize cluster centers, we select the top-k segments with the highest estimated activation bounds, measured by the maximum absolute values of their gate projection vectors $w_{g,i}$. This leverages the gating mechanism's implicit bias toward highly responsive neurons, promoting faster convergence and more coherent clusters.

To stabilize iteration and prevent segments with extremely large norms from dominating, we apply norm equalization before computing each cluster center. For each segment $\text{seg}_i \in S_j$, we scale its vector as:

$$\hat{v}_i = v(\text{seg}_i) \cdot \frac{\bar{r}_j}{\|v(\text{seg}_i)\|},$$
$$\bar{r}_j = \frac{1}{|\mathcal{S}_j|} \sum_{\text{seg}_k \in \mathcal{S}_j} \|v(\text{seg}_k)\|. \quad (11)$$

Then, each cluster center c_j is computed via normalized weighted averaging:

$$c_j \leftarrow \frac{\sum_{\text{seg}_i \in \mathcal{S}_j} \tilde{w}_i \hat{v}_i}{\left\| \sum_{\text{seg}_i \in \mathcal{S}_j} \tilde{w}_i \hat{v}_i \right\|},$$
$$\tilde{w}_i = \frac{w_i}{\sum_{\text{seg}_k \in \mathcal{S}_j} w_k}. \quad (12)$$

Finally, the resulting cluster centers $\{c_1, \ldots, c_k\}$ form the compressed expert:

$$E_r' \leftarrow \{c_1, c_2, \ldots, c_k\}, \quad (13)$$

offering a more compact and efficient representation while retaining the expert's functional diversity and expressiveness.

Table 2: Performance for Mixtral-8x7b-Instruct and Qwen2-57B-A14B-Instruct. Our method is highlighted in gray.

Model	Method	PIQA	BoolQ	HellaS.	ARC-e	ARC-c	OBQA	OBQA-F	WinoG.	MMLU	Avg.
					Mixtral-8x7b-Instruct						
8*7B	None	82.75	78.69	80.68	92.24	87.46	83.20	89.40	64.48	69.01	80.88
6*7B	LLM-Pruner	75.63	77.49	69.53	85.71	76.27	71.80	89.00	57.30	59.30	73.56
	M-SMoE	68.66	80.24	57.40	85.36	70.17	70.80	86.80	55.17	52.44	69.67
	NAEE	**82.59**	77.22	74.71	88.01	80.68	77.00	87.40	63.61	64.00	77.25
	DERN	81.72	**86.33**	**78.48**	**91.71**	**82.03**	**80.20**	**90.20**	**64.63**	**65.10**	**80.04**
4*7B	LLM-Pruner	59.96	71.47	51.60	64.90	51.86	49.40	68.20	39.94	40.73	55.34
	M-SMoE	49.56	62.11	26.01	47.62	35.93	35.20	48.80	48.86	28.29	42.49
	NAEE	69.59	71.83	**63.31**	77.78	67.46	61.00	74.40	53.91	51.89	65.69
	DERN	**70.08**	**82.57**	62.73	**83.42**	**73.56**	**71.80**	**83.60**	**54.46**	**54.91**	**70.79**
					Qwen2-57B-A14B-Instruct						
57.4B	None	84.00	89.24	87.08	97.18	91.19	87.60	93.60	69.85	75.54	86.14
45.1B	LLM-Pruner	76.93	85.99	82.38	93.12	86.10	83.20	91.80	66.77	70.64	81.88
	M-SMoE	82.70	87.09	85.00	94.53	89.83	83.40	90.20	64.80	70.65	83.13
	NAEE	80.09	87.19	85.05	94.36	88.14	82.20	92.00	67.09	72.09	83.13
	DERN	**84.60**	**88.10**	**86.02**	**95.94**	**90.17**	**88.40**	**93.60**	**70.64**	**74.06**	**85.73**
33.3B	LLM-Pruner	73.78	81.25	76.18	76.19	67.46	67.00	83.80	60.77	63.21	72.18
	M-SMoE	70.89	83.88	73.76	84.83	69.83	68.00	82.20	60.69	58.33	72.49
	NAEE	68.34	79.24	63.58	86.24	70.85	64.80	81.60	62.12	59.61	70.71
	DERN	**84.44**	**87.03**	**85.67**	**92.77**	**87.12**	**85.60**	**91.20**	**67.01**	**71.17**	**83.56**

4 Experiments

4.1 Experimental Settings

Model Settings. We evaluate our method on Mixtral-8×7B-Instruct (Jiang et al., 2024), Qwen2-57B-A14B-Instruct (Yang et al., 2024a), and DeepSeek-MoE-16B-Chat (Dai et al., 2024). For Mixtral (46.7B), we evaluate 6/8 (35.4B) and 4/8 (24.2B) expert configurations. For Qwen2 (57.4B), we use 48/64 (45.1B) and 32/64 (33.3B) expert configurations. For DeepSeek (16.4B), we use 56/64 (14.5B) and 48/64 (12.7B) configurations. These settings cover diverse architectures and compression ratios. All experiments were conducted on 2×NVIDIA H100 GPUs.

Evaluation and Datasets. To support task-agnostic pruning, we calibrate pruning-related statistics using 128 sequences (each with 2048 tokens) randomly sampled from the C4 corpus (Raffel et al., 2020), following Wanda (Sun et al., 2023).

We adopt the evaluation protocol of LLM-Pruner (Ma et al., 2023), conducting zero-shot evaluations on a suite of commonsense reasoning datasets: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), ARC-e and ARC-c (Clark et al., 2018), OpenbookQA (Mihaylov et al., 2018), and WinoGrande (Sakaguchi et al., 2021). In addition, we evaluate multi-domain reasoning via few-shot prompting on MMLU (Hendrycks et al., 2020).

All tasks are framed as generation-based evaluations, where the instruction-tuned model directly outputs the answer, and correctness is determined by template-based string matching. We follow OpenCompass (Contributors, 2023) for prompt formatting and matching criteria, and conduct all inference using the vLLM (Kwon et al., 2023). For more details, see App. B.

Baselines. We compare against representative structured SMoE pruning methods: NAEE (expert pruning) (Lu et al., 2024a), MC-SMoE (expert merging) (Li et al., 2023), and LLM-Pruner (general structured pruning) (Ma et al., 2023). For NAEE on Qwen2 and DeepSeek, we approximate expert selection by sampling 10k combinations per layer and choosing the one with minimal output deviation. For MC-SMoE, to ensure fairness under the retraining-free constraint, we disable intra-expert compression and denote it as M-SMoE. For LLM-Pruner, we evaluate on the taylor version, which is data-aware and gradient-based.

4.2 Main Results

Performance on Commonsense QA and Multidomain Benchmarks. This section presents a comprehensive evaluation of our method against

Table 3: Performance for DeepSeek-MoE-16b-Chat. Our method is highlighted in gray.

Model	Method	PIQA	BoolQ	HellaS.	ARC-e	ARC-c	OBQA	OBQA-F	WinoG.	MMLU	Avg.
16.4B	None	67.57	75.75	54.00	70.55	49.83	47.40	68.80	56.99	48.32	59.91
14.5B	LLM-Pruner	62.73	67.86	40.37	54.85	31.53	35.80	53.20	48.93	41.13	48.49
	M-SMoE	60.28	61.19	43.90	50.62	37.63	37.80	52.80	51.22	39.13	48.29
	NAEE	**64.42**	69.51	48.36	68.78	51.53	45.20	**68.80**	53.75	41.06	56.82
	DERN	61.75	**73.67**	**52.75**	**76.37**	**55.93**	**51.80**	68.40	**53.91**	**47.40**	**60.22**
12.7B	LLM-Pruner	54.68	26.36	26.59	34.74	21.02	26.00	36.20	47.59	34.05	34.14
	M-SMoE	50.38	15.87	33.20	29.45	18.64	27.60	34.00	42.78	31.79	31.52
	NAEE	56.37	52.14	34.35	55.73	36.27	39.80	59.40	51.38	34.60	46.67
	DERN	**60.83**	**68.29**	**44.47**	**70.90**	**48.14**	**49.00**	**60.20**	**53.51**	**44.17**	**55.50**

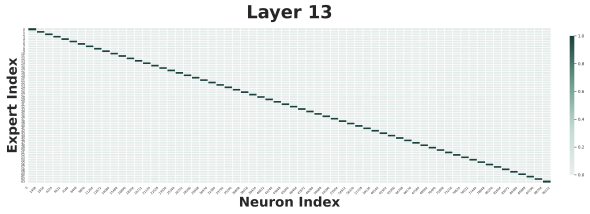

Figure 3: Neuron-level similarity heatmap among experts in Layer 13 of the DeepSeek-MoE-16b-Chat model. The predominantly light-colored off-diagonal regions highlight a high degree of independence and specialization among experts.

leading pruning baselines across three SMoE LLMs: Mixtral-8x7B-Instruct, Qwen2-57B-A14B-Instruct, and DeepSeek-MoE-16B-Chat.

Across all models and sparsity levels, DERN consistently outperforms existing techniques such as LLM-Pruner, M-SMoE, and NAEE, with particularly strong gains in commonsense reasoning and multidomain generalization. On Mixtral-8x7B-Instruct (Tab. 2), DERN achieves leading performance on most Commonsense QA tasks, in some cases matching or surpassing the original dense model. With Qwen2-57B-A14B-Instruct, DERN maintains this advantage, outperforming all baselines on MMLU while preserving high accuracy on QA benchmarks, demonstrating strong resilience under compression.

For DeepSeek-MoE-16B-Chat (Tab. 3), DERN yields consistent gains at both 56 and 48 expert settings, outperforming baseline methods. However, as the pruning ratio increases, performance degradation becomes more pronounced across all methods. To better understand this behavior, we visualize expert similarity in Fig. 3. The resulting heatmap reveals low inter-expert similarity, suggesting that DeepSeek's experts are more independent and thus more vulnerable to pruning. This indicates that general-purpose expert pruning methods may require architectural adaptation to fully accommodate highly decoupled expert designs.

In summary, DERN delivers robust and scalable performance across diverse backbones and compression settings. Its superior knowledge preservation enables effective retention of both commonsense reasoning and cross-domain capabilities.

Table 4: Inference efficiency on Mixtral before and after expert pruning, evaluated on 1,024 C4 samples with 64-way concurrency using vLLM.

Model	Mem. (GB)	Tok/s ↑	TTFT (ms) ↓	Avg.
8×7B	88.7	7962.9	3349.5	80.88
6×7B	66.0	9395.0	2850.0	80.04
4×7B	45.0	10990.3	2443.8	70.60

Inference Speedup and Memory Usage. Tab. 4 reports inference efficiency on the vLLM before and after expert pruning. Reducing the number of experts from 8 to 6 and 4 yields notable improvements in both throughput and latency: token throughput increases by up to 38%, while the time-to-first-token (TTFT) drops by over 900 ms. This acceleration primarily stems from improved kernel-level parallelism—fewer experts lead to more tokens routed to each active expert, resulting in larger and more GPU-efficient matrix multiplications. Meanwhile, memory usage is substantially reduced, enabling larger batch sizes and better hardware utilization. Notably, the 6-expert variant achieves nearly the same accuracy as the original model while offering 25% lower memory consumption and 18% higher throughput, demonstrating that DERN can effectively trade off redundancy for speed while maintaining competitive accuracy.

4.3 Ablation Study

Ablation on Segment Retention Ratio. Fig. 4 illustrates the effect of the similarity threshold α used in the second stage of DERN. Notably, $\alpha = 1$ disables merging entirely and serves as a no-merge baseline, while $\alpha = 0$ merges all pruned segments

Table 5: Ablation of different segment components used for similarity computation during segment reassignment. **DERN** uses only the up and down vectors; **triplet** includes gate, up, and down vectors; **w/o up** removes the up component; **w/o down** removes the down component.

Model	Method	PIQA	BoolQ	HellaS.	ARC-e	ARC-c	OBQA	OBQA-F	WinoG.	MMLU	Avg.
Mixtral-4×7B	**DERN**	70.08	82.57	62.73	83.42	73.56	71.8	83.6	54.46	55.01	70.80
	triplet	70.57	79.39	59.43	82.89	72.88	71.2	84.2	56.27	55.11	70.22
	w/o up	71.98	79.42	58.9	83.42	72.54	69.8	83.2	55.33	54.74	69.93
	w/o down	70.57	81.96	59.49	82.36	71.86	69.4	85.2	55.8	55.51	70.24

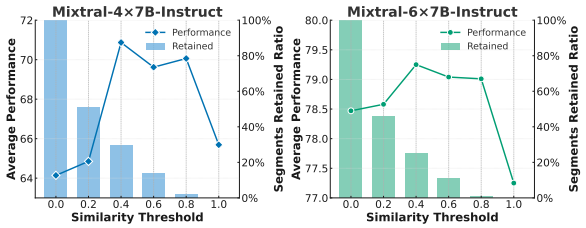

Figure 4: Effect of the similarity threshold α during segment reassignment. Line plots indicate average performance, and bars show the ratio of segments retained in last layer. Results are averaged over all benchmarks for Mixtral-4×7B (left) and Mixtral-6×7B (right).

regardless of compatibility.

We observe a clear pattern across both Mixtral-4×7B and Mixtral-6×7B: performance improves as α increases from 0 to around 0.6, then drops sharply beyond that point. This reveals a fundamental trade-off: lower thresholds allow excessive reuse, introducing semantically mismatched segments; higher thresholds, while stricter, lead to underutilization. Best performance is achieved when only a moderate fraction of segments are reassigned, highlighting a "less is more" phenomenon where selective reuse outperforms indiscriminate merging. For more details, see App. C.2.

Ablation on Neuron Types Used in Similarity Estimation. Tab. 5 examines how neuron types affect similarity measurement in clustering. Each segment consists of a gate vector (activation), an up-projection vector (input transformation), and a down-projection vector (output transformation). Excluding the gate component slightly improves performance, suggesting dynamic activation signals may introduce noise. In contrast, removing either up or down leads to consistent drops, highlighting their complementary roles in capturing transformation behavior. The best results are achieved using only the up and down components—structural features that govern information flow. These findings support prioritizing weight geometry over activation dynamics in expert reconstruction.

Table 6: Ablation on clustering initialization. **DERN** uses top-k segments ranked by gate vector weights; **Random** samples seeds uniformly; **Equidistant** selects evenly from the similarity range.

Strategy	CQA	MMLU	Average
DERN	72.78	55.01	63.89
Random	58.93	38.29	48.61
Equidistant	72.08	55.03	63.56

Table 7: Ablation on weighting mechanism in clustering. **DERN** applies segment importance during centroid updates; **w/o weighting** treats all segments equally.

Strategy	CQA	MMLU	Average
DERN	72.78	55.01	63.89
w/o weighting	71.23	54.14	62.69

Ablation on Clustering Initialization Strategy. Tab. 6 compares three initialization methods for segment clustering: DERN (gate-based), which selects top-k segments with the highest gate vector weights; Equidistant, which samples evenly across the similarity distribution; and Random choice. Gate-based initialization outperforms the others, showing that highly activated segments provide a more meaningful starting point for clustering. For more details, see App. C.3.

Ablation on Weighting Mechanism in Clustering. We evaluate whether weighting segments by their original expert importance improves cluster quality. As shown in Tab. 7, applying importance-based weighting during cluster center updates leads to a noticeable gain, suggesting that segments from more influential experts should be given higher priority in the reconstruction process.

5 Conclusion

We propose **DERN**, a task-agnostic and retraining-free framework for compressing SMoE LLMs. By treating experts as modular assemblies of functional segments, DERN performs decomposition and reconstruction based on neuron-level similarity. This enables compact expert representations

that preserve model performance while reducing memory and latency. More broadly, DERN highlights a core insight: structural reorganization at the sub-expert level offers a robust and adaptable path to efficiency in SMoE.

Limitations

While DERN performs well across various SMoE models, it measures segment similarity in parameter space using cosine distance. This may be insufficient for capturing functional alignment, especially in models with highly specialized experts. Notably, we observe that existing pruning and merging techniques perform suboptimally on DeepSeekMoE under high sparsity settings, where expert representations are more diverse and less interchangeable. This highlights the need for further investigation into pruning and recombination strategies tailored to highly specialized expert models.

Acknowledgments

This work was supported in part by the National Science and Technology Major Project (2023ZD0120803), the National Natural Science Foundation of China (62472381), the Fundamental Research Funds for the Central Universities (226-2025-00055), the Fundamental Research Funds for the Zhejiang Provincial Universities (226-2024-00208), the "Pioneer" and "Leading Goose" R&D Program of Zhejiang (2025C02032) and the Earth System Big Data Platform of the School of Earth Sciences, Zhejiang University.

References

Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. 2022. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Mingyu Cao, Gen Li, Jie Ji, Jiaqi Zhang, Xiaolong Ma, Shiwei Liu, and Lu Yin. 2024. Condense, don't just prune: Enhancing efficiency and performance in moe layer pruning. *arXiv preprint arXiv:2412.00069*.

I Chen, Hsu-Shen Liu, Wei-Fang Sun, Chen-Hao Chao, Yen-Chang Hsu, Chun-Yi Lee, and 1 others. 2024. Retraining-free merging of sparse mixture-of-experts via hierarchical clustering. *arXiv preprint arXiv:2410.08589*.

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, and 1 others. 2022. On the representation collapse of sparse mixture of experts. *Advances in Neural Information Processing Systems*, 35:34600–34613.

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

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

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*.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, and 1 others. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

Inderjit S Dhillon and Dharmendra S Modha. 2001. Concept decompositions for large sparse text data using clustering. *Machine learning*, 42:143–175.

Yubo Dong and Hehe Fan. 2025. Enhancing large language models through structured reasoning. *arXiv preprint arXiv:2506.20241*.

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.

Hao Gu, Wei Li, Lujun Li, Qiyuan Zhu, Mark Lee, Shengjie Sun, Wei Xue, and Yike Guo. 2025. Delta decompression for moe-based llms compression. *arXiv preprint arXiv:2502.17298*.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Zhaolin Hu, Yixiao Zhou, Zhongan Wang, Xin Li, Weimin Yang, Hehe Fan, and Yi Yang. 2025. Osda agent: Leveraging large language models for de novo design of organic structure directing agents. In *The International Conference on Learning Representations*.

Wei Huang, Yue Liao, Jianhui Liu, Ruifei He, Haoru Tan, Shiming Zhang, Hongsheng Li, Si Liu, and Xiaojuan Qi. 2024. Mc-moe: Mixture compressor for mixture-of-experts llms gains more. *arXiv preprint arXiv:2410.06270*.

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, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. 2023. Merge, then compress: Demystify efficient smoe with hints from its routing policy. *arXiv preprint arXiv:2310.01334*.

Wei Li, Hehe Fan, Yongkang Wong, Mohan Kankanhalli, and Yi Yang. 2024a. Cat-llm: Context-aware training enhanced large language models for multimodal contextual image retrieval.

Wei Li, Hehe Fan, Yongkang Wong, Mohan Kankanhalli, and Yi Yang. 2024b. Topa: Extending large language models for video understanding via text-only pre-alignment. *Advances in Neural Information Processing Systems*, 37:5697–5738.

Wei Li, Hehe Fan, Yongkang Wong, Yi Yang, and Mohan Kankanhalli. 2024c. Improving context understanding in multimodal large language models via multimodal composition learning. In *Forty-first International Conference on Machine Learning*.

Yuqi Li, Yao Lu, Zeyu Dong, Chuanguang Yang, Yihao Chen, and Jianping Gou. 2024d. Sglp: A similarity guided fast layer partition pruning for compressing large deep models. *arXiv preprint arXiv:2410.14720*.

Xun Liang, Hanyu Wang, Huayi Lai, Simin Niu, Shichao Song, Jiawei Yang, Jihao Zhao, Feiyu Xiong, Bo Tang, and Zhiyu Li. 2025. Seap: Training-free sparse expert activation pruning unlock the brainpower of large language models. *arXiv preprint arXiv:2503.07605*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

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*.

Ka Man Lo, Zeyu Huang, Zihan Qiu, Zili Wang, and Jie Fu. 2024. A closer look into mixture-of-experts in large language models. *arXiv preprint arXiv:2406.18219*.

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

Yao Lu, Hao Cheng, Yujie Fang, Zeyu Wang, Jiaheng Wei, Dongwei Xu, Qi Xuan, Xiaoniu Yang, and Zhaowei Zhu. 2024b. Reassessing layer pruning in llms: New insights and methods. *arXiv preprint arXiv:2411.15558*.

Yao Lu, Yutao Zhu, Yuqi Li, Dongwei Xu, Yun Lin, Qi Xuan, and Xiaoniu Yang. 2024c. A generic layer pruning method for signal modulation recognition deep learning models. *IEEE Transactions on Cognitive Communications and Networking*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720.

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. *arXiv preprint arXiv:1809.02789*.

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.

Wei Ruan, Tianze Yang, Yifan Zhou, Tianming Liu, and Jin Lu. 2025. From task-specific models to unified systems: A review of model merging approaches. *arXiv preprint arXiv:2503.08998*.

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

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Chao Wang, Hehe Fan, Ruijie Quan, Lina Yao, and Yi Yang. 2025. Protchatgpt: Towards understanding proteins with hybrid representation and large language models. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1076–1086.

Yanyue Xie, Zhi Zhang, Ding Zhou, Cong Xie, Ziang Song, Xin Liu, Yanzhi Wang, Xue Lin, and An Xu. 2024. Moe-pruner: Pruning mixture-of-experts large language model using the hints from its router. *arXiv preprint arXiv:2410.12013*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024a. Qwen2 technical report. *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. *arXiv preprint arXiv:2411.01016*.

Jinluan Yang, Dingnan Jin, Anke Tang, Li Shen, Didi Zhu, Zhengyu Chen, Ziyu Zhao, Daixin Wang, Qing Cui, Zhiqiang Zhang, and 1 others. 2025. Mix data or merge models? balancing the helpfulness, honesty, and harmlessness of large language model via model merging. *arXiv preprint arXiv:2502.06876*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Yue Zhang, Hehe Fan, Wei Ji, Yongkang Wong, Roger Zimmermann, and Yi Yang. 2025. Prompt-aware adapter: Learning adaptive visual tokens for multimodal large language models. *IEEE Transactions on Artificial Intelligence*.

Ziyu Zhao, Tao Shen, Didi Zhu, Zexi Li, Jing Su, Xuwu Wang, Kun Kuang, and Fei Wu. 2024. Merging loras like playing lego: Pushing the modularity of lora to extremes through rank-wise clustering. *arXiv preprint arXiv:2409.16167*.

Ziyu Zhao, Yixiao Zhou, Zhi Zhang, Didi Zhu, Tao Shen, Zexi Li, Jinluan Yang, Xuwu Wang, Jing Su, Kun Kuang, and 1 others. 2025. Each rank could be an expert: Single-ranked mixture of experts lora for multi-task learning. *arXiv preprint arXiv:2501.15103*.

Appendix

A Implementation Details

Model Configuration. Our method is applied to three representative SMoE LLMs: Mixtral-8×7B-Instruct, Qwen2-57B-A14B-Instruct, and DeepSeek-MoE-16B-Chat. Each expert in these models is a gated MLP using the GLU architecture, parameterized by three projection matrices $W_g \in \mathbb{R}^{h\times d}$, $W_u \in \mathbb{R}^{h\times d}$, and $W_d \in \mathbb{R}^{d\times h}$, where d and h denote the input and hidden dimensions respectively.

For Mixtral, we evaluate configurations with 8, 6, and 4 experts per MoE layer, using top-2 routing. Qwen2 and DeepSeek follow a similar sparsification scheme, with expert counts pruned from 64 to 48 or 32, and from 64 to 56 or 48 respectively. Unless otherwise noted, all models adopt top-k routing with $k = 2$. A summary of expert-related configuration parameters is shown in Tab. 8.

Table 8: Comparison of expert configurations in Mixtral-8×7B-Instruct, Qwen2-57B-A14B-Instruct, and DeepSeek-MoE-16B-Chat.

Property	Mixtral	Qwen2	DeepSeek
hidden_size	4096	3584	2048
moe_intermediate_size	14336	2560	1408
num_hidden_layers	32	28	28
num_experts	8	64	64
experts_per_token	2	8	6
shared_experts	0	1	2
activation	SiLU	SiLU	SiLU
dtype	bfloat16	bfloat16	bfloat16

Routing Statistics Collection. To evaluate expert importance, we support both parameter-based and data-driven strategies. In the data-driven setting, routing statistics are collected from a held-out calibration set consisting of 128 sequences sampled from the C4 corpus, each containing 2048 tokens. For each token, we record the top-k experts selected by the gating mechanism along with their activation weights. These statistics are aggregated to reflect both the frequency and strength of each expert's participation, forming a routing-aware importance score as defined in Equation 1. This calibration-aware pruning process enables adaptive selection of essential experts under realistic usage scenarios.

Segment Representation. Each expert is decomposed into a set of structurally independent segments, each corresponding to a triplet of projection vectors from the gate, up, and down components of the MLP block. These vectors are flattened and concatenated into fixed-dimensional representations, enabling efficient comparison across experts. To facilitate flexible merging, we define a similarity metric over segments based on weighted cosine distance, where the contribution of each component can be tuned. This design supports ablation studies over different segment configurations and allows fine-grained control over expert recombination.

Clustering Settings. Once segments are reassigned to retained experts based on local similarity, we apply spherical weighted k-means clustering (Dhillon and Modha, 2001) to reduce redundancy and form compact expert representations. The number of output segments is determined proportionally to the original expert size, allowing for flexible compression. Cluster initialization prioritizes highly activated segments to promote stability and convergence. Segment representations are normalized before aggregation, and weighted updates are applied to emphasize contributions from more influential experts. This clustering process ensures both expressiveness and efficiency in the reconstructed model.

Hardware and Inference Setup. All experiments are performed on 2×NVIDIA H100 GPUs using the vLLM serving framework (Kwon et al., 2023). Inference is conducted in float16 precision with a batch size of 128. Inference efficiency is measured with 1,024 samples under 64-way concurrency. For evaluation, we follow the OpenCompass (Contributors, 2023) protocol, using generation-based decoding with prompt templates and string-matching for answer accuracy.

B Evaluation Details

Benchmarks. We evaluate our method on a diverse suite of benchmarks covering commonsense reasoning, scientific QA, and multi-domain knowledge understanding. Specifically, we include PIQA, BoolQ, HellaSwag, ARC-Easy, ARC-Challenge, OpenBookQA, and WinoGrande. For multi-domain evaluation, we adopt the MMLU benchmark, which contains 57 subjects ranging from STEM to humanities and professional domains.

Evaluation Setting. Commonsense QA tasks are conducted in the zero-shot setting, where no in-context examples are provided. For MMLU, we

adopt a fixed 5-shot setting using a static in-context example retriever. Across all tasks, model predictions are evaluated via template-matching, where string-based rules extract the first valid option or capitalized answer token from the generated output.

Prompt and Evaluation Protocol. We use OpenCompass as the unified evaluation framework. Each dataset is paired with a custom prompt template designed for multiple-choice answering. Inference is conducted via autoregressive generation using a standard decoding configuration. Model outputs are post-processed by rule-based functions to extract answers in a structured and comparable format. Evaluation is performed using accuracy-based metrics, optionally supporting detailed per-category breakdowns.

PIQA (Zero-shot Template)

```
{goal}
A. {sol1}
B. {sol2}
Answer:
```

BoolQ (Zero-shot Template)

```
{passage}
Question: {question}
A. Yes
B. No
Answer:
```

HellaSwag (Zero-shot Template)

```
{ctx}
Question: Which ending makes the most sense?
A. {A}
B. {B}
C. {C}
D. {D}
Answer:
```

ARC (Easy / Challenge) (Zero-shot Template)

```
Question: {question}
A. {textA}
B. {textB}
C. {textC}
D. {textD}
Answer:
```

OpenBookQA (Zero-shot Template)

```
Question: {question_stem}
A. {A}
B. {B}
C. {C}
D. {D}
Answer:
```

OpenBookQA-Fact (Zero-shot Template)

```
Given the fact: {fact1}
Question: {question_stem}
A. {A}
B. {B}
C. {C}
D. {D}
Answer:
```

WinoGrande (Zero-shot Template)

```
Question: {prompt}
A. {only_option1}
B. {only_option2}
Answer:
```

MMLU (5-shot Template)

```
</E>
There is a single choice question about
{subject}. Answer the question by replying
A, B, C or D.
Question: {example_input_1}
A. {A1}
B. {B1}
C. {C1}
D. {D1}
Answer: {label1}
. . .
Question: {example_input_5}
A. {A5}
B. {B5}
C. {C5}
D. {D5}
Answer: {label5}
</E>
There is a single choice question about
{subject}. Answer the question by replying
A, B, C or D.
Question: {input}
A. {A}
B. {B}
C. {C}
D. {D}
Answer:
```

Dataset Configuration in OpenCompass. We conduct all evaluations using the standardized dataset configurations provided by the OpenCompass benchmark suite, which ensures consistency across data splits, prompt formats, and evaluation protocols. Specifically, we follow the officially released validation or test splits and adopt the corresponding input-output schema defined by OpenCompass for each task. This ensures compatibility with its prompt-based inference engine and allows for seamless integration with multiple large language models.

To maintain reproducibility and fairness, we preserve the default prompt templates and postprocessing routines associated with each dataset. These include input field extraction, candidate option for-

Table 9: Input and output field configurations for each dataset.

Dataset	Input Columns	Output Column
PIQA	goal, sol1, sol2	answer
BoolQ	passage, question	label
HellaSwag	ctx, A, B, C, D	label
ARC-E / C	question, textA–D	answerKey
OBQA	question_stem, A–D	answerKey
OBQA-F	question_stem, A–D, fact1	answerKey
WinoGrande	prompt, only_option1/2	answer
MMLU	input, A–D	target

Table 10: Task types, evaluation strategies, and output postprocessing rules.

Dataset	Task Type	Eval Type	Postproc
PIQA	Commonsense QA	Accuracy	A/B
BoolQ	Boolean QA	Accuracy	Yes/No
HellaSwag	Sentence Completion	Accuracy	A–D
ARC-E / C	Science MCQ	Accuracy	A–D
OBQA	Open-book QA	Accuracy	A–D
OBQA-F	Fact-augmented QA	Accuracy	A–D
WinoGrande	Coreference Reasoning	Accuracy	A/B
MMLU	Multidomain MCQ	Accuracy	A–D

matting (where applicable), and answer string normalization during evaluation. Tables 9 and 10 provide a detailed summary of the structural mappings and evaluation strategies employed for each benchmark.

The specific dataset variants used in our experiments follow the official OpenCompass configuration identifiers, which define the exact prompt structure, evaluation scripts, and scoring logic. The datasets and their corresponding OpenCompass IDs are listed below:

- **PIQA**: `piqa_gen_1194eb`
- **BoolQ**: `SuperGLUE_BoolQ_gen_883d50`
- **HellaSwag**: `hellaswag_gen_6faab5`
- **ARC-easy**: `ARC_e_gen_1e0de5`
- **ARC-challenge**: `ARC_c_gen_1e0de5`
- **OpenBookQA**: `obqa_gen_9069e4`
- **WinoGrande**: `winogrande_gen_458220`
- **MMLU**: `mmlu_gen_4d595a`

These configurations reflect the latest stable task definitions in OpenCompass and have been widely adopted in prior benchmarking studies. By adhering to these official variants without modification, our evaluations remain fully reproducible and aligned with best practices in LLM benchmarking.

Inference with vLLM. All evaluations are conducted using the vLLM serving framework, which enables high-throughput and memory-efficient inference via paged attention and continuous batching. In OpenCompass, we adopt two integration modes with vLLM:

- **Direct backend mode**: By specifying `-a vllm` in the evaluation command, OpenCompass directly loads and executes models using vLLM as a backend engine. No additional serving is required.
- **Remote serving mode**: For scalable or distributed evaluation, vLLM is launched as a RESTful API server via `vllm serve`, and OpenCompass connects using the `OpenAISDK` interface. This enables model-as-a-service deployment and decouples evaluation from inference hosting.

Each task is evaluated with a maximum sequence length of 256 tokens, batch size of 128, and deterministic decoding (`temperature = 0.0001`). Additional runtime parameters (e.g., tensor parallelism, GPU memory usage) are configured via model-level arguments. The following illustrates a typical vLLM server launch and OpenCompass API configuration:

Launching vLLM server:

```
vllm serve mistralai/Mixtral-8x7B-
    Instruct --host 0.0.0.0 --port 8000
```

OpenCompass model config:

```
dict(
    abbr='mixtral-vllm-api',
    type=OpenAISDK,
    openai_api_base='http://localhost
        :8000/v1',
    path='mistralai/Mixtral-8x7B-
        Instruct',
    tokenizer_path='mistralai/Mixtral-8
        x7B-Instruct',
)
```

All evaluations are fully automated and seamlessly integrated with the pruning pipeline. Once a model is compressed, its checkpoint is registered into the evaluation system and executed using vLLM without manual intervention. This ensures consistency across settings and enables scalable assessment of compression-quality trade-offs.

C Further Experimental Analyses

This section provides additional results and ablation studies that complement the main findings. It includes full benchmark breakdowns, detailed analysis of design choices, and supporting visualizations.

C.1 More Visualization of Neuron-level Similarity

To better understand the structural differences across experts in different layers, we visualize both expert-level and neuron-level similarity for several representative layers of the Mixtral-8×7B-Instruct model. As shown in Figures 5, the left subplots represent cosine similarity between expert weight matrices, while the right subplots reflect neuron-level alignment, where each row shows the maximal similarity between neurons in a source expert and neurons in a target expert.

We observe several key trends:

- In early layers (e.g., Layer 0), experts are highly distinct, as evidenced by strong diagonal blocks in expert-level similarity and sparse off-diagonal interactions at the neuron level.
- As depth increases (e.g., Layer 7 and 15), some inter-expert similarity begins to emerge. Neuron-level heatmaps reveal increasing degrees of overlap, suggesting redundancy or convergence in learned representations.
- In deeper layers (e.g., Layer 23 and 31), experts exhibit diffuse structural boundaries and higher degrees of entanglement at the neuron level, complicating naive expert merging.

These observations support our core motivation: expert-level similarity alone is insufficient for reliable merging, and neuron-level structural alignment is essential for preserving functionality. DERN's segment-based recombination mechanism addresses this by explicitly matching and merging structurally compatible components.

C.2 Full Results of Similarity Threshold α Ablation

Table 11 provides the complete results of our ablation study on the similarity threshold α, which controls segment reassignment during expert merging. We report results on Mixtral-8×7B-Instruct under two configurations: 6-expert (35.4B) and 4-expert (24.2B).

We observe a consistent pattern across both settings: performance improves as α increases from 0 to around 0.4–0.6, but then sharply drops as α approaches 1. This trend reveals a key trade-off: lower thresholds enable more segment reuse, but risk semantic mismatches due to indiscriminate merging; higher thresholds impose stricter filtering, but lead to underutilization of transferable structure. Notably, the best performance is achieved when only a moderate fraction of segments are reused, confirming that *selective reuse is superior to exhaustive merging*.

For the 4-expert setting, accuracy peaks at $\alpha = 0.4$ (70.87 average), significantly outperforming the baseline (64.14 at $\alpha = 0$), while reducing the active segment ratio to approximately 30%. Similarly, in the 6-expert setting, $\alpha = 0.4$ achieves the best trade-off between quality and compression, improving average performance from 78.47 (baseline) to 79.25. These results highlight a "less is more" phenomenon: *reusing only the most structurally compatible segments suffices to reconstruct performant and efficient experts*.

This behavior aligns with our broader insight: DERN benefits from structure-aware modularity. By leveraging local segment similarity, the method avoids the pitfalls of whole-expert averaging while effectively capturing reusable computation patterns. As a design recommendation, we find that $\alpha \in [0.4, 0.6]$ offers a robust operational range across SMoE backbones.

C.3 Initialization Strategy

To support the segment clustering process in our expert reconstruction framework, we visualize the ℓ_∞ norm of each **row vector** in the gate projection matrix W_g across four different Transformer layers. Each row vector in W_g corresponds to the gating component of a segment and determines its activation magnitude via the term $\sigma(w_{g,i}^\top x)$, as formulated in Eq. (3) of the main paper. This activation controls the contribution of the segment to the final expert output, modulating the up-projection vector.

The ℓ_∞ norm of each row provides a conservative upper bound on that segment's gating strength over all possible inputs. Segments with higher ℓ_∞ norms are more likely to be activated with greater magnitude and thus play a more prominent role in computation.

As illustrated in Figure 6, different layers exhibit distinct distributions of segment-wise activation po-

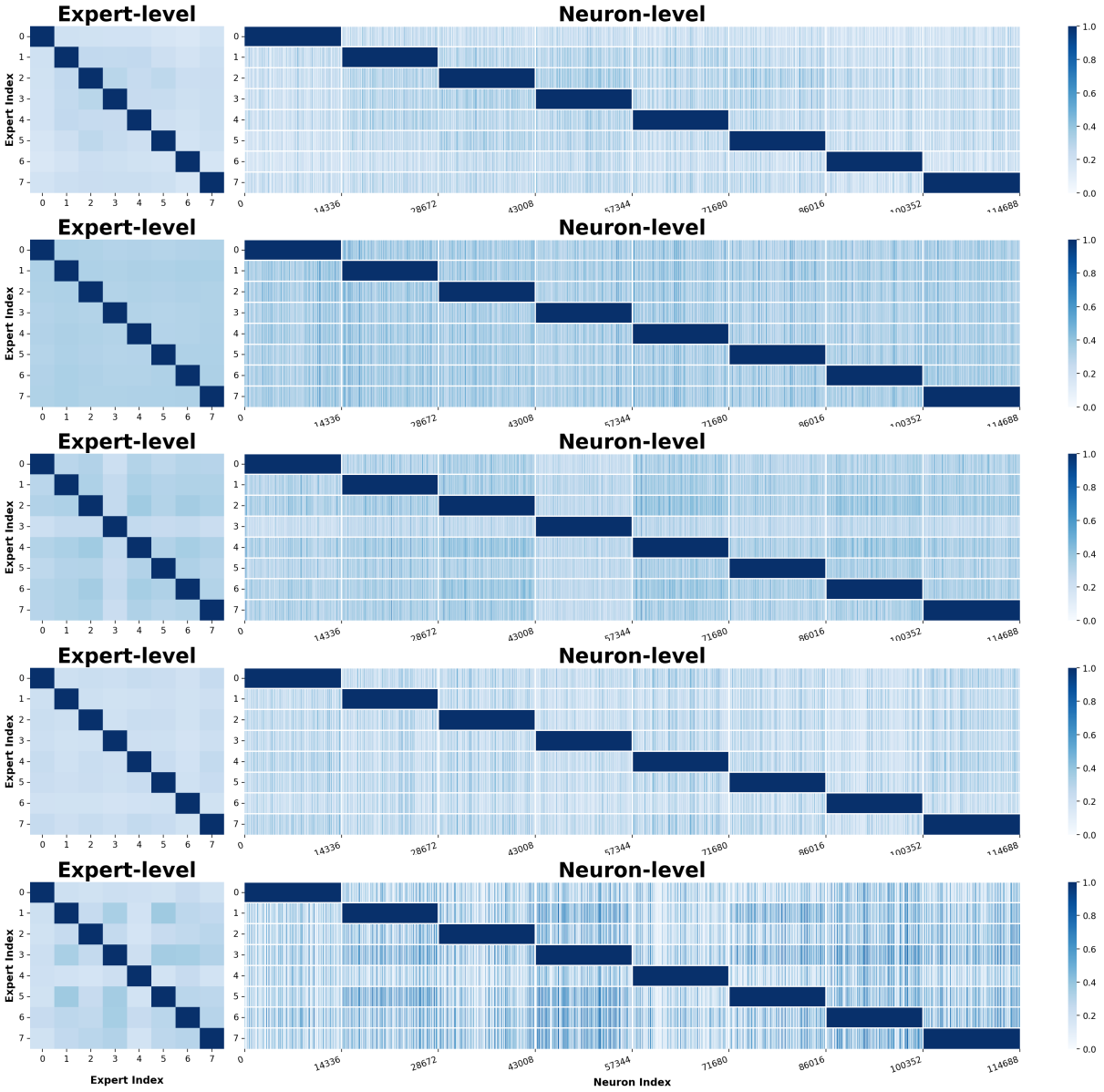

Figure 5: Neuron-level and expert-level similarity heatmaps across Layers 0, 7, 15, 23, and 31. Left: expert similarity; Right: neuron alignment.

Table 11: Full Results of Similarity Threshold α Ablation on Mixtral-4×7B-Instruct and Mixtral-6×7B-Instruct.

Setting	piqa	BoolQ	hellaswag	ARC-e	ARC-c	openbookqa	openbookqa_fact	winogrande	mmlu	Avg.	ratio
Mixtral-4×7B-Instruct											
sim_0	71.38	53.67	63.82	80.95	64.07	65	78.8	50.12	49.49	64.14	100%
sim_0.2	71.82	52.17	64.31	80.78	64.41	66.2	81.8	50.91	51.22	64.85	50.77%
sim_0.4	73.88	81.01	59.86	85.19	72.88	69.4	85	55.88	54.74	70.87	29.79%
sim_0.6	71.00	79.69	58.69	83.42	71.86	69	84.6	53.20	55.16	69.62	13.70%
sim_0.8	71.16	80.03	60.07	83.42	72.88	68.6	84.4	55.17	54.83	70.06	2.08%
sim_1.0	69.59	71.83	63.31	77.78	67.46	61	74.4	53.91	51.89	65.69	0%
Mixtral-6×7B-Instruct											
sim_0	81.77	84.98	77.63	90.48	80.00	77.8	89.0	61.88	62.68	78.47	100%
sim_0.2	78.94	85.26	77.61	90.48	80.34	79.8	89.2	62.51	63.12	78.58	45.99%
sim_0.4	80.20	86.76	78.31	91.18	81.36	79.4	89.2	62.27	64.53	79.25	25.15%
sim_0.6	79.98	85.32	78.08	91.89	82.71	78.2	88.4	62.27	64.50	79.04	10.79%
sim_0.8	80.69	84.77	78.00	91.18	82.37	78.2	88.6	62.67	64.60	79.01	1.10%
sim_1.0	82.59	77.22	74.71	88.01	80.68	77.0	87.4	63.61	64.00	77.25	0%

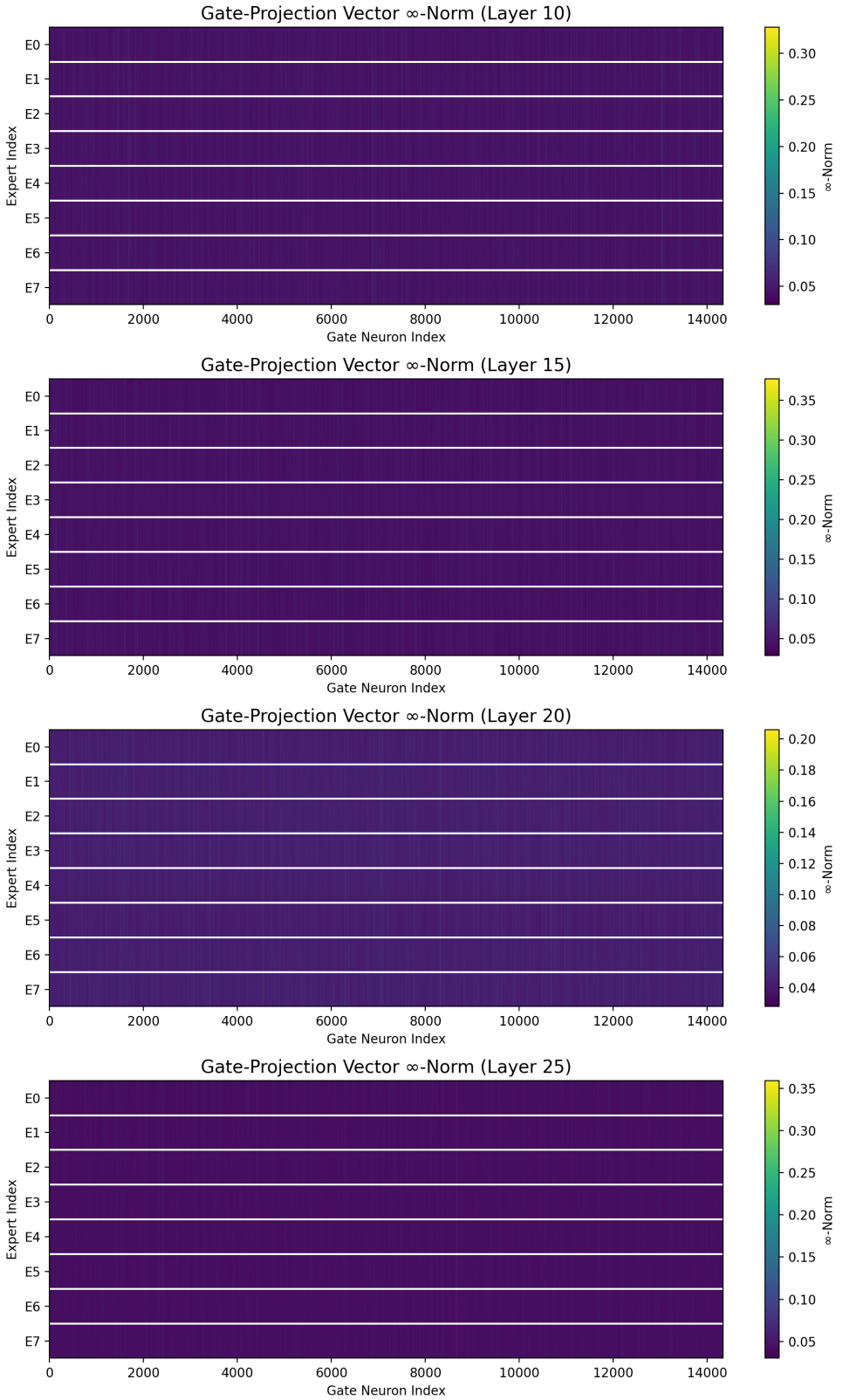

Figure 6: ℓ_∞ norm of row vectors W_g across Layers 10, 15, 20, and 25.

Table 12: Full Results on Ablation of different clustering initialization strategies on Mixtral-4×7B-Instruct.

Strategy	PIQA	BoolQ	HellaSwag	ARC-e	ARC-c	OpenBookQA	OBQA-Fact	WinoGrande	MMLU	Avg.
Gate-based	70.08	82.57	62.73	83.42	73.56	71.80	83.60	54.46	55.01	70.80
Random	60.01	54.28	46.30	73.02	58.31	55.00	75.00	49.49	38.29	57.74
Equidistant	71.38	81.53	61.56	82.72	73.22	69.00	83.20	54.06	55.03	70.30

tential. Some segments are consistently dominant, while others have negligible norms, indicating a disparity in their functional contribution. We leverage this observation to guide the initialization of our clustering algorithm by selecting high-norm segments as cluster centroids. This ensures that structurally and functionally salient components are retained during expert merging, which is crucial for maintaining model performance after compression.

To further validate this design choice, we present the full results of different clustering initialization strategies in Table 12. Among the three evaluated strategies—gate-based, random, and equidistant—the gate-based method consistently outperforms others across tasks. This supports the intuition that segment saliency, as reflected by gating magnitude, serves as an effective heuristic for initialization during expert reconstruction.

D Link to Models and Datasets

All models and datasets used in this work are publicly available and sourced from established repositories to ensure reproducibility and transparency. We rely on widely adopted pretrained SMoE language models including Mixtral-8×7B-Instruct, Qwen2-57B-A14B-Instruct, and DeepSeek-MoE-16B-Chat, all hosted on Hugging Face. These models represent a diverse set of sparse expert architectures and serve as representative backbones for evaluating the generality of our proposed method.

For dataset selection, we follow common benchmarks used in LLM pruning and evaluation literature. Specifically, we include the C4 corpus for calibration, and a suite of commonsense and multi-domain reasoning datasets such as BoolQ, PIQA, HellaSwag, ARC, OpenBookQA, WinoGrande, and MMLU. All datasets are retrieved through the Hugging Face Datasets Hub, ensuring consistent preprocessing and accessibility.

A full list of links to model checkpoints and dataset resources is provided below for reference.

D.1 Models

Mixtral-8×7B-Instruct https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1

Qwen2-57B-A14B-Instruct https://huggingface.co/Qwen/Qwen2-57B-A14B-Instruct

DeepSeek-MoE-16B-Chat https://huggingface.co/deepseek-ai/deepseek-moe-16b-chat

D.2 Datasets

C4 https://huggingface.co/datasets/allenai/c4

BoolQ https://huggingface.co/datasets/google/boolq

PIQA https://huggingface.co/datasets/ybisk/piqa

HellaSwag https://huggingface.co/datasets/Rowan/hellaswag

ARC https://huggingface.co/datasets/allenai/ai2_arc

OpenBookQA https://huggingface.co/datasets/allenai/openbookqa

WinoGrande https://huggingface.co/datasets/allenai/winogrande

MMLU https://huggingface.co/datasets/cais/mmlu