TENP: Trapezoidal Expert Neuron Pruning For Mixture-of-Experts

Jiangyang He, Shaolin Zhu, Deyi Xiong*
TJUNLP Lab, School of Computer Science and Technology, Tianjin University, China
{jiangyanghe, zhushaolin, dyxiong}@tju.edu.cn

Abstract

Mixture-of-Experts large language models (LLMs) scale efficiently through sparse activation, yet their deployment is fundamentally constrained by the large static parameter footprint of experts. Existing compression approaches either remove entire experts, disrupting routing topology and harming performance, or rely on unstructured weight pruning with limited practical efficiency. To address the limitations, we propose TENP, a structured **T**rapezoidal **E**xpert **N**euron **P**runing framework. Using a few samples, we identify and retain important experts, while applying expert neuron pruning (ENP) to less important experts, preserving model parameters in a trapezoidal pattern from shallow to deep layers. When evaluating expert importance, we jointly consider both the magnitude of the expert output and its ability to change the direction of the input vector. For ENP, we measure each neuron's projected contribution to the expert output to identify and retain important neurons. We conduct extensive experiments on the Qwen and DeepSeek models. Under a routing expert sparsity of 40% and an average of 63.76% activated expert parameters, the DeepSeek model suffers only a 1-point drop in accuracy compared to the full-parameter model. Moreover, it outperforms the full-parameter model by 10% on code generation tasks.

1 Introduction

The paradigm shift towards Mixture-of-Experts (MoE) architectures has become a cornerstone for scaling, as it effectively decouples model capacity from computational cost (OpenAI, 2025). By conditionally activating a sparse subset of parameters for each token, models such as DeepSeek-V3.2 (DeepSeek-AI et al., 2025) and Qwen3 (Yang et al., 2025) achieve state-of-the-art performance with significantly reduced FLOPs compared to their dense counterparts (Sun et al., 2024a; Pan et al., 2025; Zhu et al., 2024). However, this efficiency comes with a substantial trade-off that the massive static parameter footprint required to host the full set of experts creates a severe bottleneck for deployment, particularly in memory-constrained environments (Bai et al., 2025).

To address the memory bottleneck, existing compression methodologies primarily bifurcate into expert pruning, weight pruning (Bai et al., 2025) or quantization (Du et al., 2025; Jin et al., 2024). Expert pruning methods attempt to permanently remove less significant experts based on activation frequency or router gradients (Muzio et al., 2024; Dong et al., 2025). However, these approaches are constrained by a fundamental limitation: the coarse-grained removal of entire experts disrupts the model's original routing topology. As indicated in recent analyses (Chen et al., 2025), altering the routing path compels forces tokens to be dispatched to sub-optimal experts, which can precipitate significant performance degradation on domain-specific tasks. Alternatively, unstructured weight pruning methods (Sun et al., 2024b) target individual parameters but consequently yield irregular sparsity patterns that require specialized hardware kernels for acceleration.

Recent empirical analyses indicate that for compressed models, restoring their original routing paths can recover model performance (Chen et al., 2025). However, experts that have been removed can no longer be routed to. We therefore prune redundant parameters within experts while keeping the experts intact and routable. Prior work (Sun et al., 2024b; Cheng et al., 2025) has found that a large amount of redundancy exists at the microscopic, neuron-level within these experts. Furthermore, redundancy is non-uniformly distributed in the layer-wise representational capacity of LLMs. Shallow layers primarily process local syntactic features and exhibit high redundancy, whereas deep layers encapsulate complex semantic reasoning

*Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 20911–20925

(Yang et al., 2024a; Gao et al., 2024). This implies that a uniform pruning ratio is sub-optimal for MoE architectures and motivates a depth-aware allocation of parameter budgets.

To address these limitations, we propose **TENP** (Trapezoidal Expert Neuron Pruning), a structured pruning framework tailored for MoE LLMs. Unlike expert-level pruning, which alters where a token goes, TENP focuses on slimming down what the expert computes by pruning neurons within experts to maintain the validity of the pre-trained router's decisions. In particular, TENP introduces a depth-aware Trapezoidal sparsity strategy. It applies aggressive pruning to shallow, high-redundancy layers and progressively retains more capacity in deep layers to preserve reasoning capabilities. To accurately identify redundant neurons without costly retraining, we design a dual-metric evaluation that combines the magnitude contribution with a directional impact score to distinguish between essential transformations and redundant identity-like mappings.

Our contributions are summarized as follows:

- We propose **TENP**, a **structured expert-neuron pruning** that reduces memory usage while strictly preserving the original MoE routing topology.
- We introducean Trapezoidal sparsity distribution strategy. We empirically demonstrate that allocating high parameter budgets to deep layers while aggressively compressing shallow layers yields a superior trade-off between model size and performance.
- Experiments demonstrate that, under routing expert sparsities of 40% and 70%, our method consistently outperforms existing expert pruning approaches across a wide range of reasoning tasks and benchmarks, achieving an average improvement of approximately 16%.

2 Related Work

MoE LLMs MoE models have gained significant attention in recent years due to their unique capability of expanding model capacity without proportionally increasing computational costs. MoE architectures partition a large neural network (or specific components) into multiple expert sub-networks, where only a subset is activated for each input token based on routing decisions (Shazeer et al., 2017; Fedus et al., 2022). GShard (Lepikhin et al., 2020) pioneers trillion-parameter models by distributing parameters across multiple devices. DeepSeekMoE (Dai et al., 2024) presents a shared expert mechanism to reduce communication overhead and computational cost. The effectiveness of MoE architectures has been validated at the 16 billion parameter scale (Team, 2024; DeepSeek-AI et al., 2024). More recently, Mixtral (Jiang et al., 2024), GPT-OSS (OpenAI, 2025), DeepSeekV3.2 (DeepSeek-AI et al., 2025), and KimiK2 (Team et al., 2025) have demonstrated MoE's efficacy at the hundred-billion parameter scale. Advanced routing strategies have also emerged, with DA-MoE (Yao et al., 2024) and XMoE (Yang et al., 2024b) implementing dynamic expert selection mechanisms that allocate more computational resources to challenging tokens. Gao et al. (2024) propose a pyramid-shaped architecture where layers closer to the output employ more parameters. Meanwhile, GroveMoE (Wu et al., 2025) adopts heterogeneous experts with dynamic parameter activation to optimize performance.

Pruning and Compression of MoE Models As scaling laws continue to drive exponential growth in MoE model sizes, numerous techniques have emerged to reduce parameter counts while preserving performance. SEER-MoE (Muzio et al., 2024) prunes less important experts based on their activation frequency or gating score. Lu et al. (2024) introduces expert-level pruning combined with dynamic skipping mechanisms. Dong et al. (2025) concentrates model capabilities in specific domains through few-shot expert localization. While these methods reduce parameter counts, they generally fail to decrease computational requirements proportionally. Cheng et al. (2025) achieve computation reduction through fine-grained neuron activation within selected experts, without reducing the overall parameter count. More comprehensive approaches include MoE-I2 (Yang et al., 2024a), which proposes a three-stage pruning methodology requiring subsequent fine-tuning to recover performance, and Task-Specific Expert Pruning (Chen et al., 2022), which integrates pruning with task-specific training. SparseGPT (Frantar and Alistarh, 2023), MoE-Pruner (Xie et al., 2024), and Wanda (Sun et al., 2024b) employ an unstructured pruning method that imposes specific hardware requirements. Alternative approaches like (Liu et al., 2024) consolidate important neurons across experts, though this compromises the original routing mech-

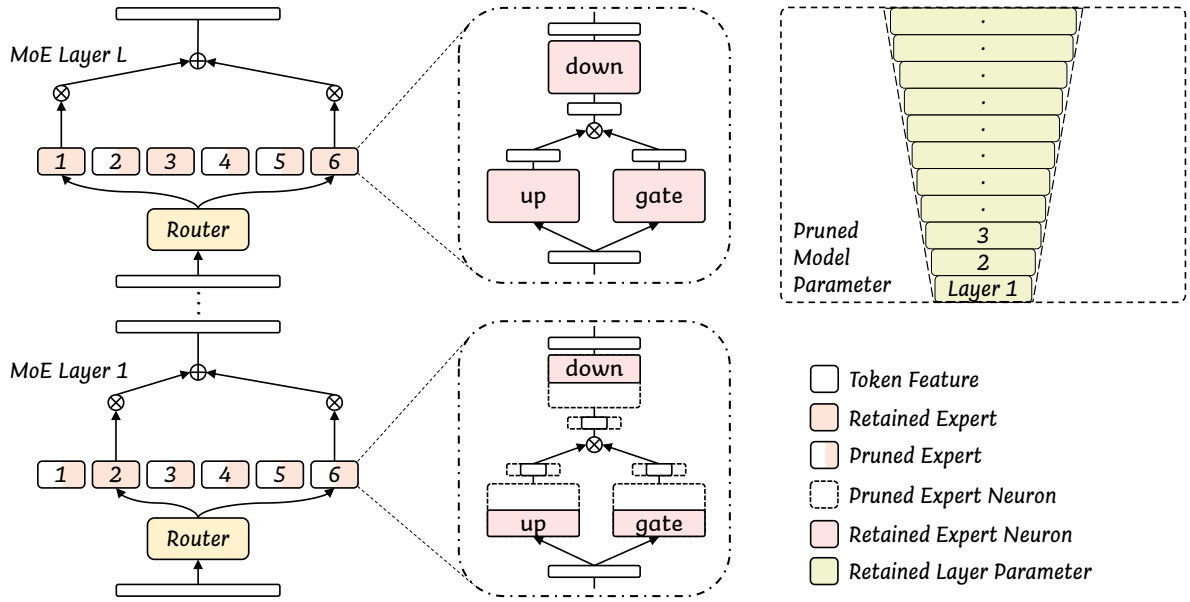

Figure 1: Schematic illustration of the Trapezoidal Expert Neuron Pruning (TENP) framework. (a) TENP preserves the complete routing structure, allowing even pruned experts to remain routable. (b) Comparison between an expert before pruning and after pruning: neurons inside the expert are removed, while the dimensionality of the expert's output remains unchanged. (c) After pruning, the parameter distribution of the model changes from a rectangular structure, where each layer has an identical number of parameters, to a trapezoidal structure with fewer parameters in shallow layers and more parameters in deep layers.

anism. Similarly, Li et al. (2024) and Li et al. (2025) propose expert merging strategies that face routing challenges. The methods compress all experts into a single dense model, although they have reduced many parameters, have changed the architecture of the model (He et al., 2023; Cao et al., 2025).

3 Method

Our approach consists of two stages. In the first stage, we follow and modify EASY-EP to identify important experts. We conduct experiments with different important expert retention ratios, as detailed in Appendix D. Ultimately, we determine that under a routed expert sparsity of 40%, retaining 20%–30% of important experts yields optimal results, while under a routed expert sparsity of 70%, retaining 5% of important experts is sufficient. This approach not only preserves the complete routing topology but also reduces the average number of activated parameters per expert. Moreover, the number of important experts preserved in each layer gradually increases with depth. As illustrated in Figure 1, higher layers retain more parameters, resulting in a trapezoidal parameter distribution. In the second stage, we perform neuron pruning on the less important experts identified in the first stage. Alternatively, the first stage can be skipped, and neuron pruning can be directly applied to all experts; we refer to this approach as Expert Neuron Pruning (ENP). We evaluate the contribution of each intermediate dimension of an expert to the output using the method described below, where each intermediate dimension corresponds to specific rows and columns of the expert's parameters. For dimensions deemed unimportant, we remove the corresponding rows and columns of parameters to eliminate that intermediate dimension.

3.1 Retaining Important Experts

To evaluate expert importance, we compute an importance score from the experts' outputs. Suppose the input to the MoE block at layer l is $\mathbf{h}_t^l$. Let the output of the i-th routed expert E_i^l for token t be $\overline{\mathbf{h}}_{i,t}^l$, and the routing weight be $\mathbf{g}_{i,t}^l$. The MoE output of all routed experts at layer l is denoted as $\tilde{\mathbf{h}}_t^l$, which is the weighted sum of the N experts' outputs, as follows:

$$\overline{\mathbf{h}}_{i,t}^l = \mathrm{E}_i^l(\mathbf{h}_t^l), \tag{1}$$

$$\tilde{\mathbf{h}}_t^l = \sum_{i=1}^{N} \mathbf{g}_{i,t}^l \cdot \overline{\mathbf{h}}_{i,t}^l. \tag{2}$$

We define the length-based contribution of expert outputs as $\mathbf{c}_{i,t}^l$. Using $\|\cdot\|$ to denote the ℓ_2

norm, for each token t, we quantify the contribution of expert i to the layer output by the product of the routing weight and the output norm:

$$\mathbf{c}_{i,t}^{l} = \mathbf{g}_{i,t}^{l} \, \|\overline{\mathbf{h}}_{i,t}^{l}\|, \quad \forall \, \mathbf{g}_{i,t}^{l} > 0. \tag{3}$$

Inspired by Men et al. (2025) and Dong et al. (2025), beyond the output magnitude and routing weight, we consider each expert's ability to alter the direction of the input vector. Men et al. (2025) suggest that when a layer behaves closer to an identity mapping, it tends to be more redundant. However, $\mathbf{c}_{i,t}^{l}$ only reflects the magnitude of an expert's output vector and the weight assigned to the expert by the router; it does not capture the expert's ability to change the direction of the input vector. In other words, relying solely on $\mathbf{c}_{i,t}^{l}$ does not allow us to determine whether an expert is performing an identity mapping, since an expert that implements an identity mapping can also have a large $\mathbf{c}_{i,t}^{l}$ value. Therefore, we introduce $\mathbf{s}_{i,t}^{l}$ to quantify the directional change induced by the expert, defined as one minus the cosine similarity between the expert's input and output. Values close to zero correspond to near-identity behavior, whereas larger values indicate more substantial angular deviations.

$$\mathbf{s}_{i,t}^{l} = 1 - \mathrm{Sim}(\mathbf{h}_{t}^{l}, \overline{\mathbf{h}}_{i,t}^{l}), \tag{4}$$

where $\mathrm{Sim}(\cdot,\cdot)$ denotes cosine similarity.

Finally, we jointly consider the magnitude of the expert output vector, the weight assigned by the router, and the expert's ability to alter the vector direction, and average these factors over all tokens T to comprehensively evaluate the importance of each expert, as shown in the following formula:

$$\mathbf{I}(\mathrm{E}_{i}^{l}) = \sum_{t=1}^{T} \mathbf{c}_{i,t}^{l} \cdot \mathbf{s}_{i,t}^{l}. \tag{5}$$

For each domain, we use 128 validation samples to evaluate expert and neuron importance. We also investigate the effect of different numbers of samples on the results, as reported in 4.8. For aggregation across domains τ, we apply an ℓ_2-norm-based normalization (regularization) as follows:

$$\mathbf{I}_{\mathrm{mix}}(\mathrm{E}_{i}^{l}) = \sum_{\tau \in \mathcal{T}} \frac{\mathbf{I}_{\tau}(\mathrm{E}_{i}^{l})}{\sqrt{\sum_{j=1}^{N} \mathbf{I}_{\tau}(\mathrm{E}_{j}^{l})^{2}}}. \tag{6}$$

3.2 Expert Neuron Pruning

After selecting important experts, we perform neuron pruning on the remaining experts. As noted above, neuron pruning can also be applied to all experts directly. For a single expert, its output can be written as:

$$\overline{\mathbf{h}}_{t}^{l} = \mathbf{W}_{\mathrm{down}} \, \mathrm{SwiGLU}(\mathbf{W}_{\mathrm{up}}\mathbf{h}_{t}^{l}, \; \mathbf{W}_{\mathrm{gate}}\mathbf{h}_{t}^{l}), \tag{7}$$

where $\mathbf{W}_{\mathrm{gate}}$ is the gating matrix, $\mathbf{W}_{\mathrm{up}}$ is the first linear projection, $\mathbf{W}_{\mathrm{down}}$ is the second linear projection, and $\mathrm{SwiGLU}(\cdot,\cdot)$ denotes the SiLU-gated activation.*

We extract the k-th row of $\mathbf{W}_{\mathrm{up}}$ and $\mathbf{W}_{\mathrm{gate}}$, and the k-th column of $\mathbf{W}_{\mathrm{down}}$, denoted by $\mathbf{w}_{\mathrm{up},k}$, $\mathbf{w}_{\mathrm{gate},k}$, and $\mathbf{w}_{\mathrm{down},k}$, respectively. Substituting them into the above equation yields the expert output when only the k-th neuron is retained:

$$\overline{\mathbf{h}}_{t,k}^{l} = \mathbf{w}_{\mathrm{down},k} \, \mathrm{SwiGLU}(\mathbf{w}_{\mathrm{up},k}\mathbf{h}_{t}^{l}, \; \mathbf{w}_{\mathrm{gate},k}\mathbf{h}_{t}^{l}). \tag{8}$$

Both $\overline{\mathbf{h}}_{t,k}^{l}$ and $\overline{\mathbf{h}}_{t}^{l}$ share the same output dimension, i.e., $\overline{\mathbf{h}}_{t,k}^{l}, \; \overline{\mathbf{h}}_{t}^{l} \in \mathbb{R}^{d}$. We then quantify the importance of neuron k by either the magnitude of its projection onto the full expert output or by the ℓ_2 norm of $\overline{\mathbf{h}}_{t,k}^{l}$. Using the projection magnitude, we define:

$$\mathbf{p}_{k} = \frac{\langle \overline{\mathbf{h}}_{t,k}^{l}, \; \overline{\mathbf{h}}_{t}^{l} \rangle}{\|\overline{\mathbf{h}}_{t}^{l}\|}. \tag{9}$$

A larger projection magnitude (or ℓ_2 norm) indicates a more important neuron. We aggregate neuron importance by averaging across tokens for the k-th neuron in expert i at layer l (As in Appendix in Section B):

$$\mathbf{P}_{i,k}^{l} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{P}_{i,k}^{l}. \tag{10}$$

We prune each expert by keeping its top-K most important neurons. Let $\mathrm{TopK}(\mathbf{P}_{i}^{l})_{\mathrm{idx}}$ denote the indices of the top-K values in $\mathbf{P}_{i}^{l}$. For the i-th expert at layer l, the pruned parameters are:

$$\tilde{\mathbf{W}}_{i,\mathrm{up}}^{l} = \mathbf{W}_{i,\mathrm{up}}^{l}\big[\mathrm{TopK}(\mathbf{P}_{i}^{l})_{\mathrm{idx}}, :\big], \tag{11}$$

$$\tilde{\mathbf{W}}_{i,\mathrm{gate}}^{l} = \mathbf{W}_{i,\mathrm{gate}}^{l}\big[\mathrm{TopK}(\mathbf{P}_{i}^{l})_{\mathrm{idx}}, :\big], \tag{12}$$

$$\tilde{\mathbf{W}}_{i,\mathrm{down}}^{l} = \mathbf{W}_{i,\mathrm{down}}^{l}\big[:, \mathrm{TopK}(\mathbf{P}_{i}^{l})_{\mathrm{idx}}\big]. \tag{13}$$

By replacing the original expert's weight matrix $\mathbf{W}_{i,\mathrm{up}}^{l}, \mathbf{W}_{i,\mathrm{gate}}^{l}, \mathbf{W}_{i,\mathrm{down}}^{l}$ with the pruned $\tilde{\mathbf{W}}_{i,\mathrm{up}}^{l}, \tilde{\mathbf{W}}_{i,\mathrm{gate}}^{l} \tilde{\mathbf{W}}_{i,\mathrm{down}}^{l}$, we obtain the neuron-pruned expert. The forward computation of the pruned expert can be formally formulated as:

*If using a different FFN variant, the formulation can be adjusted accordingly.

Model	Method	E↓	A↓	GSM8K	MBPP	Humaneval	ARC-E	ARC-C	Avg.
Qwen1.5MoE -A2.7B	Full	100%	100%	61.5	47.6	34.2	86.4	76.1	61.16
	Random	30%	100%	1.7	0.0	0.0	25.8	25.7	10.64
	Frequency	30%	100%	1.6	0.0	0.0	52.6	44.1	19.66
	Gating Score	30%	100%	2.3	0.4	1.2	56.3	43.2	20.68
	EASY-EP	30%	100%	3.4	0.4	1.8	55.3	44.8	21.14
	ENP(Ours)	30%	**30%**	23.1	25.1	17.1	78.2	66.0	41.90
	TENP(Ours)	30%	34.51%	**25.7**	**25.3**	**18.9**	**78.2**	**66.9**	**43.00**
	Random	60%	100%	19.6	1.2	0.0	77.8	64.5	32.62
	Frequency	60%	100%	30.9	14.6	6.7	80.0	66.7	39.78
	Gating Score	60%	100%	30.8	18.9	8.5	84.5	73.0	43.14
	EASY-EP	60%	100%	36.8	35.4	19.5	80.1	68.3	48.02
	ENP(Ours)	60%	**60%**	51.3	40.6	28.0	84.6	74.9	55.88
	TENP(Ours)	60%	61.38%	**58.3**	**45.7**	**31.1**	**85.1**	**75.1**	**59.06**
DeepSeek -V2-Lite	Full	100%	100%	41.1	43.2	26.2	84.1	70.3	52.98
	Random	30%	100%	1.1	0.0	0.0	24.5	24.2	9.96
	Frequency	30%	100%	1.9	0.0	0.0	24.3	24.0	10.04
	Gating Score	30%	100%	1.9	0.0	0.0	26.7	27.8	11.28
	EASY-EP	30%	100%	2.8	4.3	1.2	38.4	29.6	15.26
	ENP(Ours)	30%	**30%**	3.7	9.1	0.0	59.6	48.4	24.16
	TENP(Ours)	30%	36.38%	**21.1**	**33.1**	**14.0**	**73.3**	**57.1**	**39.72**
	Random	60%	100%	1.8	0.0	0.0	34.3	31.7	13.56
	Frequency	60%	100%	32.8	24.4	11.6	75.3	61.8	41.18
	Gating Score	60%	100%	21.5	30.7	11.6	74.0	57.5	39.06
	EASY-EP	60%	100%	34.8	42.1	14.6	76.7	62.7	46.18
	ENP(Ours)	60%	**60%**	22.1	33.9	19.5	77.8	65.7	43.80
	TENP(Ours)	60%	63.76%	**38.4**	**45.7**	**29.9**	**79.1**	**66.8**	**51.98**

Table 1: Comparison of our method with other expert-pruning approaches across all benchmarks. E denotes the equivalent total parameter count of the routed experts, and A denotes the average activated parameter count of the routed experts.

$$\overline{\mathbf{h}}_t^l = \tilde{\mathbf{W}}_{\text{down}} \, \text{SwiGLU}(\tilde{\mathbf{W}}_{\text{up}} \mathbf{h}_t^l, \; \tilde{\mathbf{W}}_{\text{gate}} \mathbf{h}_t^l), \tag{14}$$

Neuron pruning provides an additional benefit: it not only reduces the total number of parameters, but also decreases the number of parameters that are activated in the routed experts.

4 Experiment

We conducted experiments on two MoE models with distinct architectures: Qwen1.5-MoE-A2.7B (Team, 2024) and DeepSeek-V2-Lite (DeepSeek-AI et al., 2024). Detailed model descriptions are provided in Appendix A. We also report experiments on Qwen3-Next-80B-A3B-Instruct(Yang et al., 2025) in Appendix F.

4.1 Experimental Setup

Evaluation. We evaluated the proposed method on a diverse set of benchmarks spanning multiple domains. For challenging mathematical reasoning, we used GSM8K (Cobbe et al., 2021) with 8-shot prompting. For code generation, we reported results on MBPP (Austin et al., 2021) with 3-shot prompting and HumanEval (Chen et al., 2021) under zero-shot evaluation. For science question answering, we used ARC-Easy and ARC-Challenge (Clark et al., 2018), both under 25-shot prompting. Following standard protocols, we used dataset-specific few-shot settings on open-source datasets and reported results averaged over three runs.

Baselines. We compared against four representative expert-pruning approaches for MoE models. As a lower bound, we included a random expert selection baseline to quantify performance when no preference is given to expert selection. We further evaluated the frequency-based pruning and gating-score-based pruning strategies proposed in SEER-MoE (Muzio et al., 2024). In addition, we compared with the recently proposed EASY-EP

Method	E↓	A↓	MBPP	Humaneval	ARC-E	ARC-C	Avg.
Full	100%	100%	43.2	26.2	84.2	70.3	55.98
Random	60%	100%	0.0	0.0	34.3	31.7	16.50
TENP w/o Both (Random Select Both)	60%	60.73%	1.5	0.0	41.0	37.0	19.88
TENP w/o ENP (Random Select ENP)	60%	60.87%	6.7	0.6	59.8	45.7	28.20
TENP w/o TE (Random Select TE)	60%	61.00%	34.7	20.7	76.8	64.4	49.15
Only EP	60%	100%	45.7	8.5	77.8	63.6	48.90
Only ENP-L2	60%	**60%**	32.3	18.9	77.5	65.9	48.65
Only ENP-COS	60%	**60%**	41.7	17.7	78.7	65.9	51.00
TENP	60%	63.55%	**47.2**	**29.8**	**79.4**	**66.7**	**55.78**

Table 2: Results of the ablation study. Only EP denotes applying expert pruning only. ENP-L2 and ENP-COS denote neuron pruning based on neuron importance measured by the ℓ_2 norm or by the projection length, respectively. Random denotes randomly selected groups used as a control baseline.

(Dong et al., 2025). These baselines retained only the experts that are ranked highest according to statistics estimated from a small number of samples per dataset.[†]

4.2 Main Results

Table 1 reported a comprehensive comparison between our approach and a variety of baselines across multiple datasets, model backbones, and pruning ratios. When we applied TENP to prune DeepSeek-V2-Lite, the resulting model achieves better performance on both mathematical reasoning and knowledge-intensive QA tasks than other expert-pruning methods such as SEER-MoE and EASY-EP. On code-generation benchmarks (e.g., MBPP and HumanEval), the pruned model yields more than one point improvement over the full model. Similar trends are observed on Qwen1.5MoE-A2.7B, where our method consistently surpassed competing approaches across all evaluated domains. For relatively simple QA-style tasks such as ARC, pruned models generally preserved their original performance well. In contrast, on more challenging reasoning-heavy tasks (e.g., GSM8K), pruning induced some degradation; nevertheless, our approach still outperforms baselines. Notably, for code generation, the performance drop is often small and can even exceed the full model, as highlighted by the underlined entries in Table 1. We also conducted experiments at a higher sparsity level (70%), where our method consistently outperformed other approaches across all benchmarks. Additional comparisons under different sparsity settings are provided in Appendix C.

Activated-Parameter Efficiency. Most existing expert-pruning methods can remove experts, yet keep the number of experts selected by the router unchanged. As a result, although the total parameter count decreases, the number of activated parameters remained the same (normalized to 100%). In contrast, our approach pruned neurons within experts, thereby reducing not only the number of experts but also the activated parameters of routed experts accordingly. With neuron pruning alone, our ENP variant achieved the lowest activated-parameter footprint and, in most cases, still outperforms prior expert-pruning methods.

As indicated by metric A in Table 1, if the router selected a preserved (unpruned) expert, its activated parameters remain at 100%. If it selected a neuron-pruned expert, the activated parameters are reduced to 50% or lower of the original. Since the routing probability of preserved experts is higher than that of neuron-pruned experts, the overall activated parameters are only slightly higher than the fraction of preserved experts. Consequently, the activated-parameter cost of our routed experts is substantially lower than that of existing expert-pruning approaches.

4.3 Ablation Study

As shown in Table 2, applying either expert pruning alone (Only EP) or expert neuron pruning alone (ENP) already yields a certain level of pruning effectiveness to validate the effectiveness of the two core components of our method.

When ENP is applied, measuring neuron importance by the projection length of a neuron's output vector onto the final output vector of the cor-

[†]We do not included comparisons to methods that require unstructured pruning (e.g., Wanda), additional fine-tuning (e.g., MoE-I2), or approaches that modify the model architecture.

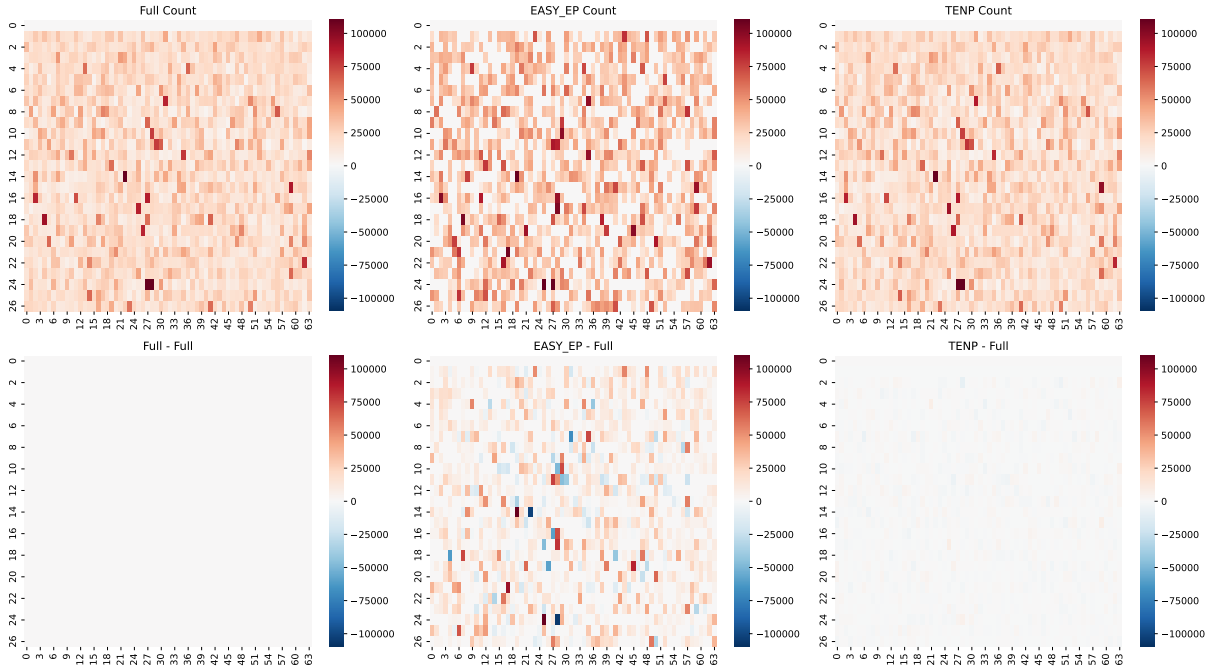

Figure 2: Expert selection frequencies under different pruning methods, and their differences compared to the expert selection frequencies of the full-parameter model.

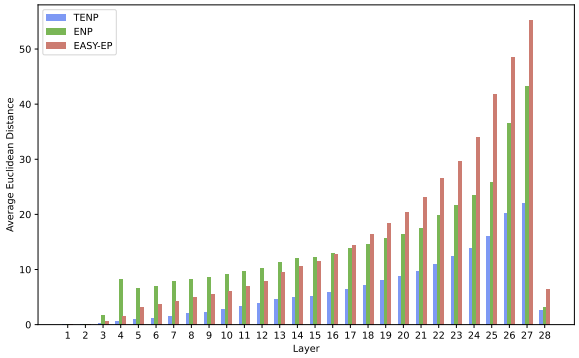

Figure 3: Layer-wise error of the pruned models, measured as the Euclidean distance between the output vectors of each layer and those of the full-parameter model.

responding expert consistently outperformed the method that used the ℓ_2 norm of the neuron output vector. This result indicated that projection-based importance better captured a neuron's contribution to the expert output. When experts are selected randomly, the model performance is noticeably affected. This observation demonstrated the validity of the experts we have retained in a trapezoidal structure. Furthermore, when both experts and neurons are selected randomly, our method still outperformed the baseline that randomly selected experts (Random), even though both configurations have the same number of parameters. This advantage arose from the different parameter distributions: our method allocated more parameters to higher layers through the trapezoidal structure and preserved the complete routing architecture. We also compare other layer selection methods in Appendix E. By combining important experts with neurons selected based on projection-length importance, we obtained TENP, which achieved the best overall performance. Notably, TENP incurred almost no accuracy loss compared to the full-parameter model.

4.4 Routing Analysis

We characterized routing changes by measuring the difference in expert selection counts before and after pruning. As shown in Figure 2, we analyzed and compared the expert selection statistics of the full-parameter model, the expert-pruned model, and our proposed pruning method on DeepSeek. We focus on the changes in expert selection frequencies induced by pruning. When we retained only a subset of experts, it induced substantial changes in expert selection patterns. Notably, the selection frequency of remaining experts does not uniformly increase after pruning. Instead, while some experts experience a significant increase in selection frequency, others are selected less frequently, or even less than before pruning. This observation indicates that routing behavior changes drastically. More importantly, some experts that were not pruned and were previously routable are no longer selected by

Method	Data	GSM8K	MBPP	Humaneval	ARC-E	ARC-C	InAvg.	OutAvg	InAvg.	OutAvg
EASY-EP	Math	38.3	28.0	13.4	64.4	55.6	38.30	**40.35**	47.45	31.81
	Code	33.0	43.7	18.9	56.3	44.2	31.30	44.50		
	Science	23.6	6.3	1.8	79.1	66.4	72.75	10.57		
TENP	Math	40.9	20.5	10.4	66.8	51.2	**40.90**	37.23	**51.37**	**33.06**
	Code	29.5	50.8	27.4	69.9	54.1	**39.10**	**51.17**		
	Science	20.2	9.8	2.4	80.7	67.5	**74.10**	**10.80**		

Table 3: Generalization performance of the pruned models. Data denotes the domain of the data used for pruning.

Model	Method	E	MMLU
Qwen1.5MoE -A2.7B	Full	100%	61.05
	GatingScore	60%	49.13
	EASY-EP	60%	47.49
	TENP(Ours)	60%	**54.81**

Table 4: The generalization performance of three pruning methods on the MMLU dataset, where no MMLU data is used during the pruning process.

the router after pruning. This led to a significant shift in the output representations. Furthermore, we observed a trend that the magnitude of changes in expert selection frequency increased in deep layers, suggesting that routing behavior in deep layers is more severely affected by expert pruning. In contrast, under our proposed method, the selection frequency of each expert remained almost unchanged, as illustrated in the bottom-right figure in Figure 2. This indicated that our approach largely preserved the original routing behavior. Maintaining stable expert routing is one of the key reasons for the effectiveness of our method.

4.5 Error Analysis

Pruning the FFN layers inevitably caused discrepancies between the outputs of the pruned model and those of the full-parameter model. In general, pruning more parameters led to larger output deviations. Since the output of one layer serves as the input to the next, these deviations further influence subsequent routing decisions, causing errors to accumulate progressively across layers, as shown in Figure 3. To quantify this effect, we computed the difference between the output vectors of the pruned model and the full model at each layer, and used the ℓ_2 norm (i.e., Euclidean distance) of the difference vector as a measure of layer-wise error. Our method exhibited a similar overall error trend to the expert pruning method EASY-EP. When ENP is applied in isolation, relatively large errors can be observed even in the shallow layers. This behavior arose because certain experts play a disproportionately important role; uniform neuron pruning removes neurons indiscriminately from both critical and less critical experts, which introduced a substantial error at early stages. Nevertheless, because ENP preserved the original routing structure, error accumulation across layers proceeds at a slower rate than with direct expert pruning. Consequently, in the middle and deep layers of the model, the error introduced by expert pruning exceeded that caused by neuron pruning. Interestingly, although the error gradually accumulates across layers, it drops sharply at the final layer. This observation highlights the strong representational capacity of high-level experts. Motivated by this phenomenon, our model adopts a trapezoidal parameter distribution, allocating more parameters to higher layers.

4.6 Generalization Ability

In this section, we designed a set of generalization experiments and compared our method with other approaches. We performed pruning using data from a single domain,e.g., mathematics, code, or science, and then evaluated the pruned models on both in domain and out of domain benchmarks. The experimental results are summarised in Table 3 and Table 4. For in-domain pruning, our method consistently outperforms expert pruning approaches. More importantly, our method achieves the highest average performance among the compared methods on out of domain evaluation. We attribute this strong generalization capability to the fact that our approach better preserves the original routing behavior and the overall structural integrity of the model.

4.7 Static Memory Consumption and Throughput

We further compare our ENP method with expert pruning approaches in terms of memory consumption and throughput. As shown in Table 5, under

Method	Param	Mem(GB)	Input	Output	Total	Rate
Full	100%	32.95	2790.43	2793.08	5583.51	1.0
Expert Pruning	50%	18.55	3941.07	3944.81	7885.88	1.41
ENP(Ours)	50%	**16.37**	**4094.03**	**4097.91**	**8191.94**	**1.47**

Table 5: A comparison of throughput and static memory consumption among the full-parameter model, expert pruning methods, and our method. Param denotes the routed expert retention ratio, and Mem represents the static GPU memory consumption of the Qwen model on a single A100-SXM-80GB GPU. Input denotes the input token throughput (tok/s), Output denotes the output token throughput (tok/s), Total denotes the total token throughput (tok/s), and Rate represents the ratio of the total throughput of each method relative to that of the full-parameter model.

the same parameter scale (50% routed expert sparsity), ENP yields smaller individual experts, making it more friendly to GPU memory allocation, and significantly reduces static memory consumption from 32.95 GB to 16.37 GB. With fewer activated parameters, ENP achieves higher input, output, and total throughput, increasing the total throughput to 147% relative to the full-parameter model, and outperforming the expert pruning model with the same parameter scale by 6%.

Unlike unstructured pruning methods, both ENP and TENP do not impose any special hardware requirements. TENP requires modifications to the inference framework (e.g., vLLM, SGLang) to accommodate experts of different sizes, whereas ENP can be deployed without modifying the inference framework.

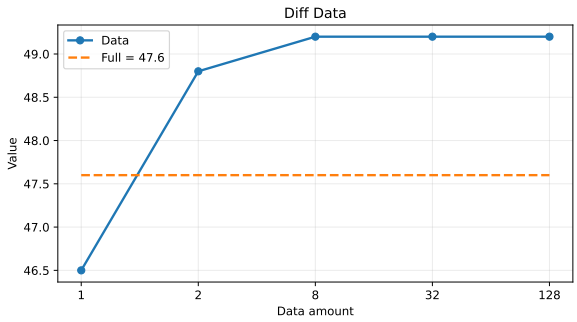

Figure 4: The impact of different data scales on pruning performance.

4.8 The Impact of Different Data Scales on Pruning Performance

Following the data selection strategy in EASY-EP and considering practical scenarios where sufficient samples may not be available, we use only the prompts from the dataset and the responses generated by the full-parameter model itself, thereby minimizing human involvement. We further investigate the impact of different data scales on model performance, as shown in Figure 4. For a single application scenario, under a routed expert sparsity of 60%, using only one sample is sufficient for the model to retain most of its performance. With two samples, the pruned model can already surpass the performance of the full-parameter model. Performance plateaus at eight samples, and further scaling brings no significant improvement.

5 Conclusion

In this paper, we have presented TENP, a pruning method for MoE models that preserves important experts while structurally pruning unimportant neurons within experts. Extensive experiments demonstrate that our approach better maintains the original routing behavior of the model, induces smaller intermediate-layer errors, and achieves superior generalization performance. Moreover, TENP consistently performs well across different sparsity levels, model architectures, and benchmarks.

Limitations

Although TENP has been shown to be highly effective on Qwen and DeepSeek models, and larger models are expected to contain more redundant parameters suggesting that TENP could yield even greater pruning benefits we have not yet conducted experiments on extremely large scale mixture-of-experts models such as DeepSeek-V3.2. We leave the evaluation of TENP on such large-scale models to future work.

Acknowledgments

The present research was supported by the National Key Research and Development Program of China (Grant No. 2023YFE0116400). We would like to thank the anonymous reviewers for their insightful comments.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Sikai Bai, Haoxi Li, Jie Zhang, Zicong Hong, and Song Guo. 2025. Diep: Adaptive mixture-of-experts compression through differentiable expert pruning. *Preprint*, arXiv:2509.16105.

Mingyu Cao, Gen Li, Jie Ji, Jiaqi Zhang, Xiaolong Ma, Shiwei Liu, and Lu Yin. 2025. Condense, don't just prune: Enhancing efficiency and performance in moe layer pruning. *Preprint*, arXiv:2412.00069.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

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. *Preprint*, arXiv:2206.00277.

Yuanteng Chen, Yuantian Shao, Peisong Wang, and Jian Cheng. 2025. EAC-MoE: Expert-selection aware compressor for mixture-of-experts large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12942–12963, Vienna, Austria. Association for Computational Linguistics.

Runxi Cheng, Yuchen Guan, Yucheng Ding, Qingguo Hu, Yongxian Wei, Chun Yuan, Yelong Shen, Weizhu Chen, and Yeyun Gong. 2025. Mixture of neuron experts. *Preprint*, arXiv:2510.05781.

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. *Preprint*, arXiv:1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

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. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *Preprint*, arXiv:2401.06066.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, and 138 others. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *Preprint*, arXiv:2405.04434.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, and 245 others. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *Preprint*, arXiv:2512.02556.

Zican Dong, Han Peng, Peiyu Liu, Wayne Xin Zhao, Dong Wu, Feng Xiao, and Zhifeng Wang. 2025. Domain-specific pruning of large mixture-of-experts models with few-shot demonstrations. *Preprint*, arXiv:2504.06792.

Jiangcun Du, Renren Jin, Wuwei Huang, Wei Liu, Jian Luan, and Deyi Xiong. 2025. Optimize quantization for large language models via progressive training. In *Companion Proceedings of the ACM on Web Conference 2025*, WWW '25, page 2474–2483, New York, NY, USA. Association for Computing Machinery.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Preprint*, arXiv:2101.03961.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. *Preprint*, arXiv:2301.00774.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and VS Subrahmanian. 2024. Higher layers need more lora experts. *Preprint*, arXiv:2402.08562.

Shwai He, Run-Ze Fan, Liang Ding, Li Shen, Tianyi Zhou, and Dacheng Tao. 2023. Merging experts into one: Improving computational efficiency of mixture of experts. *Preprint*, arXiv:2310.09832.

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, and 7 others. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Renren Jin, Jiangcun Du, Wuwei Huang, Wei Liu, Jian Luan, Bin Wang, and Deyi Xiong. 2024. A comprehensive evaluation of quantization strategies for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages

12186–12215, Bangkok, Thailand. Association for Computational Linguistics.

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

Muqing Li, Ning Li, Xin Yuan, Wenchao Xu, Quan Chen, Song Guo, and Haijun Zhang. 2025. Comoe: Collaborative optimization of expert aggregation and offloading for moe-based llms at edge. *Preprint*, arXiv:2508.09208.

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. 2024. Merge, then compress: Demystify efficient smoe with hints from its routing policy. *Preprint*, arXiv:2310.01334.

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

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

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2025. ShortGPT: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20192–20204, Vienna, Austria. Association for Computational Linguistics.

Alexandre Muzio, Alex Sun, and Churan He. 2024. Seer-moe: Sparse expert efficiency through regularization for mixture-of-experts. *Preprint*, arXiv:2404.05089.

OpenAI. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

Leiyu Pan, Bojian Xiong, Lei Yang, Renren Jin, Shaowei Zhang, Yue Chen, Ling Shi, Jiang Zhou, Junru Wu, Zhen D. Wang, Jianxiang Peng, Juesi Xiao, Tianyu Dong, Zhuowen Han, Zhuo Chen, Yuqi Ren, and Deyi Xiong. 2025. Advancing large language models for tibetan with curated data and continual pre-training. *ArXiv*, abs/2507.09205.

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. *Preprint*, arXiv:1701.06538.

Haoran Sun, Renren Jin, Shaoyang Xu, Leiyu Pan, Supryadi, Menglong Cui, Jiangcun Du, Yikun Lei, Lei Yang, Ling Shi, Juesi Xiao, Shaolin Zhu, and Deyi Xiong. 2024a. FuxiTranyu: A multilingual large language model trained with balanced data. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1499–1522, Miami, Florida, US. Association for Computational Linguistics.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024b. A simple and effective pruning approach for large language models. *Preprint*, arXiv:2306.11695.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, and 150 others. 2025. Kimi k2: Open agentic intelligence. *Preprint*, arXiv:2507.20534.

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

Haoyuan Wu, Haoxing Chen, Xiaodong Chen, Zhanchao Zhou, Tieyuan Chen, Yihong Zhuang, Guoshan Lu, Zenan Huang, Junbo Zhao, Lin Liu, Zhenzhong Lan, Bei Yu, and Jianguo Li. 2025. Grove moe: Towards efficient and superior moe llms with adjugate experts. *Preprint*, arXiv:2508.07785.

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. *Preprint*, arXiv:2410.12013.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Cheng Yang, Yang Sui, Jinqi Xiao, Lingyi Huang, Yu Gong, Yuanlin Duan, Wenqi Jia, Miao Yin, Yu Cheng, and Bo Yuan. 2024a. Moe-i^2: Compressing mixture of experts models through inter-expert pruning and intra-expert low-rank decomposition. *Preprint*, arXiv:2411.01016.

Yuanhang Yang, Shiyi Qi, Wenchao Gu, Chaozheng Wang, Cuiyun Gao, and Zenglin Xu. 2024b. Xmoe: Sparse models with fine-grained and adaptive expert selection. *Preprint*, arXiv:2403.18926.

Zelin Yao, Chuang Liu, Xianke Meng, Yibing Zhan, Jia Wu, Shirui Pan, and Wenbin Hu. 2024. Da-moe: Addressing depth-sensitivity in graph-level analysis through mixture of experts. *Preprint*, arXiv:2411.03025.

Shaolin Zhu, Supryadi, Shaoyang Xu, Haoran Sun, Leiyu Pan, Menglong Cui, Jiangcun Du, Renren Jin,

António Branco, and Deyi Xiong. 2024. Multilingual large language models: A systematic survey. *ArXiv*, abs/2411.11072.

A Model Detail

Qwen1.5-MoE-A2.7B contains 14 billion parameters and 24 layers. Each layer consists of 60 routed experts and 4 shared experts. For each token, the router selects the top 4 experts with the highest scores in each layer to perform the forward computation.DeepSeek-V2-Lite contains 16 billion parameters and 27 layers, where the first layer is a dense layer. Starting from the second layer, each layer includes 64 routed experts and 2 shared experts. For each token, the router selects the top 6 experts with the highest scores in each layer to perform the forward computation.

B Expert Neuron Importance Algorithm

Algorithm 1 describes in detail our method for selecting important neurons within an expert. Given only the parameters of a specific expert and the input vectors, we can compute the importance of each neuron based on this information. By leveraging PyTorch's broadcasting mechanism in matrix multiplication, we are able to compute the average importance of all neurons in an expert across all tokens using a single forward pass.

The algorithm illustrates that neuron importance is determined by computing the projection length of the output vector produced independently by each neuron onto the expert output, which is the superposition of the outputs of all neurons. A simpler alternative is to directly evaluate the magnitude of each neuron's output using its L_2 norm. In this case, it suffices to directly compute the L_2 norm of C in the algorithm.

C Pruning at different levels of sparsification

To more comprehensively reflect the effectiveness of our method under different sparsity levels of routed experts, we design experiments with scales of 15%, 30%, 45%, 60%, 75%, and 90%. As shown in Figure 5, the experimental curves indicate that our method achieves the best performance on average across different sparsity levels. At the scale of 90%, our method outperforms the full-parameter model on nearly all benchmarks, with particularly significant improvements on code generation and mathematical reasoning tasks. At the scale of 75%, our method performs on par with the full-parameter model. At the scale of 60%, the model performance is slightly lower than that of the full-parameter model, yet still superior to other expert pruning methods. At the scale of 45%, 30%, and 15%, our method continues to preserve the core capabilities of the model.

Algorithm 1 Expert Neuron Importance Algorithm

Require: $x \leftarrow$ *Input hidden states* $\triangleright\, x : L \times d$
Require: $K \leftarrow$ *Neuron Number Of One Expert*
Require: $W_G \leftarrow$ *Gate Matrix* $\triangleright\, W_G : K \times d$
Require: $W_U \leftarrow$ *Up Matrix* $\triangleright\, W_U : K \times d$
Require: $W_D \leftarrow$ *Down Matrix* $\triangleright\, W_D : d \times K$
Ensure: $y \leftarrow$ *Output hidden states* $\triangleright\, y : L \times d$
Ensure: $P \leftarrow$ *Neuron Importance* $\triangleright\, P : K$
$m \leftarrow \mathrm{act}(W_G x \odot W_U x)$ $\triangleright\, m : L \times K$
$y \leftarrow W_D m$
$M \leftarrow m^\top.\mathrm{unsqueeze}(-1)$ $\triangleright\, M : K \times L \times 1$
$\mathbf{W}_{D2} \leftarrow W_D^\top.\mathrm{unsqueeze}(1)$ $\triangleright$
$\mathbf{W}_{D2} : K \times 1 \times d$
$C \leftarrow M \mathbin{@} \mathbf{W}_{D2}$ $\triangleright\, C : K \times L \times d$
$Y \leftarrow y.\mathrm{unsqueeze}(0)$ $\triangleright\, Y : 1 \times L \times d$
$s \leftarrow (C \odot Y).\mathrm{sum}(\mathrm{dim} = -1)$ $\triangleright\, s : K \times L$
$r \leftarrow Y.\mathrm{norm}(p = 2, \mathrm{dim} = -1)$ $\triangleright\, r : K \times L$
$\varepsilon \leftarrow 10^{-8}$
$P \leftarrow s \,/\, (r + \varepsilon)$ $\triangleright\, P : K \times L$
$P \leftarrow P.\mathrm{mean}(\mathrm{dim} = 1)$ $\triangleright\, P : K$
return y, P

D The Impact of Different Important Expert Retention Ratios on Pruning Performance

To evaluate the effect of different important expert retention ratios on pruning performance, we conduct experiments under a routed expert sparsity of 60% with varying retention ratios to determine the optimal setting. As shown in Table 7, the optimal important expert retention ratio for the Qwen model is 30%, corresponding to a retention ratio of unimportant expert neurons of $(60\% - 30\%)/(100\% - 30\%) \approx 42.86\%$. For the DeepSeek model, the optimal important-expert retention ratio is 20%, with the corresponding retention ratio of unimportant expert neurons being $(60\% - 20\%)/(100\% - 20\%) = 50.00\%$. When the important-expert retention ratio is 0%, the TENP method degenerates into the ENP method. From the table, we observe that as the retention ratio increases, accuracy first improves and then degrades, with the optimal retention ratios concentrated in the middle range. By default, setting the parameter budget of important experts equal to that of unimportant experts yields favorable perfor-

Model	Retain	GSM8K	MBPP	Humaneval	ARC-E	ARC-C	Avg.
Qwen1.5MoE-A2.7B	0%	51.3	40.6	28.0	84.6	74.9	55.88
	10%	52.9	42.3	32.3	85.1	75.0	57.52
	20%	53.8	42.9	**32.3**	85.1	74.6	57.74
	30%	**58.3**	**45.7**	31.1	**85.1**	**75.1**	**59.06**
	40%	57.5	44.5	30.5	83.0	73.5	57.80
	50%	54.3	42.3	26.8	81.2	71.0	55.12
DeepSeek-V2-Lite	0%	22.1	33.9	19.5	77.8	65.7	43.80
	10%	36.8	**48.0**	28.7	78.1	64.7	51.26
	20%	38.4	45.7	**29.9**	**79.1**	**66.8**	**51.98**
	30%	**38.7**	44.5	26.2	78.5	66.7	50.92
	40%	38.6	40.9	23.8	76.4	62.6	48.46
	50%	25.3	34.6	13.4	70.9	56.2	40.08

Table 6: The performance of our TENP method on different datasets under varying important expert retention ratios, where Retain denotes the important expert retention ratio.

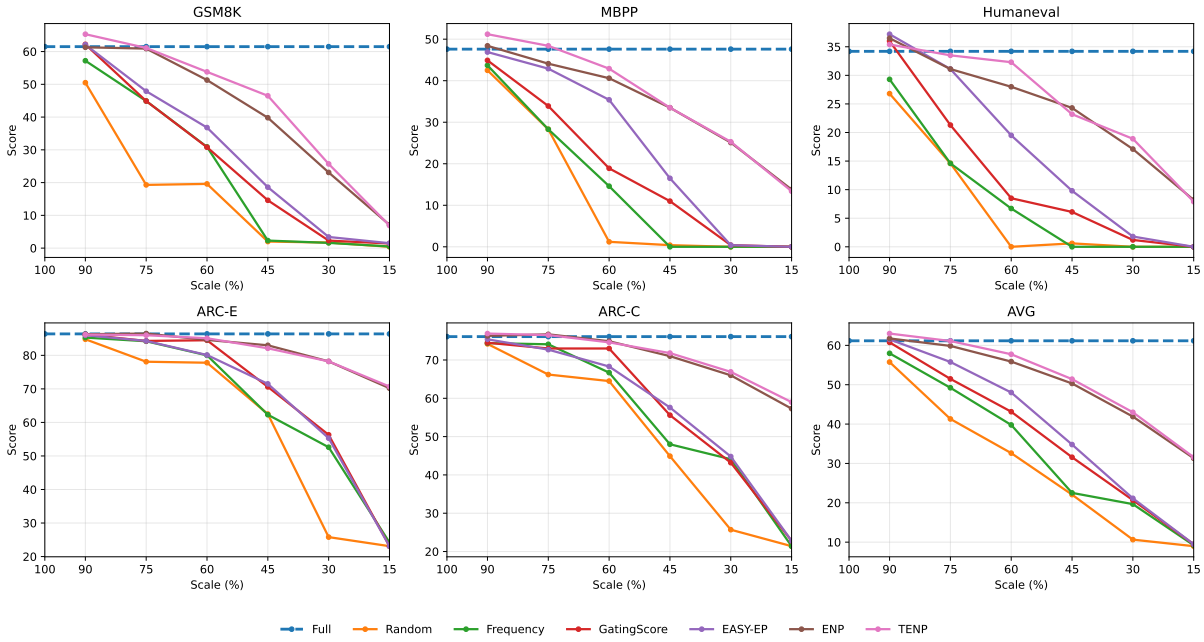

Figure 5: The performance of different model pruning methods across various benchmarks under different sparsity levels.

mance.

E Other layer selection methods

We reproduced both methods on our model using their official open-source implementations. For MoDeGPT (BI), we computed BI scores with the released code and allocated the retained experts to each layer proportionally; when the allocated number exceeded a layer's capacity, the overflow experts were re-assigned to the layers with the smallest retention. For ShortGPT ranking, we followed the open-source procedure to estimate layer importance and then used the resulting ranking to reorder our trapezoidal per-layer expert allocation. The results indicate that, although layer evaluation criteria developed for dense models can affect expert allocation in MoE models, they do not lead to consistent gains under the same sparsity budget. Specifically, the BI and ShortGPT based variants show slight advantages on language understanding tasks such as ARC-E and ARC-C, but perform noticeably worse on reasoning and code generation benchmarks, including GSM8K, MBPP, and HumanEval. By contrast, our allocation strat-

Model	Method	E	Total Sparsity	GSM8K	MBPP	HumanEval	ARC-E	ARC-C	Avg.
Qwen1.5MoE-A2.7B	ENP	60%	35%	51.3	40.6	28.0	84.6	74.9	55.88
	TENP+BI(MoDeGPT)	60%	35%	49.1	39.4	30.5	85.2	**75.3**	55.90
	TENP(Rectange)	60%	35%	51.1	39.1	29.2	85.2	**75.3**	55.98
	TENP+Rank(ShortGPT)	60%	35%	49.7	41.3	28.0	**85.7**	**75.3**	56.00
	TENP (Ours)	60%	35%	**58.3**	**45.7**	**31.1**	85.1	75.1	**59.06**

Table 7: A comparison of other layer selection methods with our method

Model	Method	E	GSM8K	MBPP	HumanEval	ARC-E	ARC-C	Avg.
Qwen3-Next-80B-A3B-Instruct	Full	100.00%	93.7	76.7	84.1	94.8	93.9	88.64
	Random	60.00%	85.8	63.3	64.6	89.9	88.1	78.34
	Frequency	60.00%	87.8	72.2	73.1	94.0	91.5	83.72
	GatingScore	60.00%	92.6	74.0	79.8	**94.5**	93.1	86.80
	EASY-EP	60.00%	93.5	73.3	**81.1**	**94.5**	93.2	87.12
	TENP	60.00%	**93.6**	**75.1**	**81.1**	94.4	**93.3**	**87.50**

Table 8: The evaluation results of our method on Qwen3-Next-80B-A3B-Instruct at 60% expert sparsity.

egy achieves the best overall average performance, while also delivering the strongest results on reasoning and code generation under the same sparsity constraint.

F Performance under the 80B parameter setting

In addition to the two 14B and 16B models used in the main experiments, we also conducted experiments on the Qwen3-Next-80B-A3B-Instruct model. Table 8 further supplements the validation of the effectiveness of our method on large-scale models. As can be observed from the experimental results, as the number of model parameters increases, the level of redundancy also grows, leading to a smaller loss in model accuracy after pruning. Notably, even random pruning achieves relatively strong performance, with an average score decrease of only about 10%. In contrast, our methods, TENP and EASY-EP, incur almost no degradation in model accuracy.