SPARSER-MOE: BEYOND EXPERT SPARSITY WITH HETEROGENEOUS NEURON-LEVEL SPARSITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have demonstrated strong performance across diverse domains, but their high inference costs remain a challenge. To address this issue, prior work has introduced methods that selectively activate neurons in Gated-MLP blocks based on intermediate activation magnitudes, reducing computational costs while largely preserving model performance. However, these methods do not account for the distinctive characteristics of mixture-of-experts (MoE) models, limiting their ability to preserve performance when applying activation sparsification to MoE. We propose Sparser-MoE, an MoE-tailored neuron sparsification framework featuring dynamic expert-wise sparsity and contribution-scale-aware neuron scoring. Motivated by the observation that higher-ranked experts contain more important neurons, Sparser-MoE performs global neuron selection across the top-K experts, allowing heterogeneous sparsity to emerge naturally across experts. It further incorporates routing weights into neuron scores to better preserve the scale of the MoE layer output. Together, these designs identify a more effective set of active neurons than existing methods and thereby better preserve the original model performance under sparsification. Experiments across diverse models, model scales, and benchmarks show that Sparser-MoE retains more of the dense model's original performance than existing sparsification methods on nearly every individual benchmark at each evaluated sparsity level, while achieving the highest average benchmark performance at every evaluated sparsity level. To our knowledge, Sparser-MoE is the first Gated-MLP sparsification approach to jointly introduce MoE-specific neuron scoring and selection strategies, and we believe it can serve as a starting point for related research.

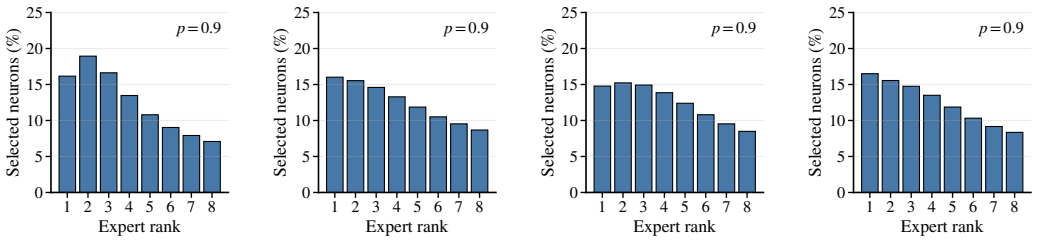
1 INTRODUCTION

Large language models (LLMs) excel at text generation and are now evolving beyond conventional NLP into agents that autonomously select and employ various techniques (Schick et al., 2023; Yao et al., 2022). As the applications of LLMs become increasingly diverse and their capabilities grow, the computational costs required to support them continue to rise (Kim et al., 2026; Jia & Li, 2026). Consequently, techniques that reduce inference costs while maintaining high performance are becoming increasingly important.

Improving LLM performance by increasing model size and computation raises not only training costs but also inference costs (Hoffmann et al., 2022; Sardana et al., 2023). Mixture-of-Experts (MoE) has emerged as an approach to address this challenge (Shazeer et al., 2017; Fedus et al., 2022; Lepikhin et al., 2020). A defining characteristic of MoE is that it maintains multiple FFN or MLP experts—from a few to tens or even hundreds—and uses a network called a router to determine which experts to activate at runtime. This sparse activation allows model capacity and the performance benefits of scaling to increase without a proportional rise in per-token computation, thereby reducing training and inference costs relative to a similarly sized dense model.

Another approach that has attracted attention exploits the prevalence of low-magnitude intermediate activations in and around Gated-MLP blocks. By retaining only activations estimated to contribute most to the output, these methods reduce computation while largely preserving model performance (Lee et al., 2024; Liu et al., 2025; Zhou et al., 2025; Cheon & Kang, 2025). Because MoE models also contain Gated-MLP blocks, as dense models do, the same general sparsification

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) layer0-3 **(b) layer4-7** **(c) layer8-11** **(d) layer12-15**

Figure 1: Distribution of selected neurons across expert ranks within the top-K experts at 90% sparsity. Neurons from higher-ranked experts are generally selected more frequently. Neuron scores are based on the gate projection, with $s_j = |\mathrm{SiLU}((xW_{\mathrm{gate}})_j)|$. Measurements are obtained on OLMoE-1B-7B (Muennighoff et al., 2024) using the C4 (Raffel et al., 2020) calibration set, with results grouped into sets of four layers across all 16 layers.

principle can be applied to them. However, simply applying such sparsification to MoE overlooks its distinctive characteristics, and MoE-specific sparsification remains underexplored.

Our experiments (Figure 1) reveal an interesting pattern: when intermediate activations from the selected top-K experts are jointly sorted and neurons are selected to meet a target sparsity level, more neurons are selected from higher-ranked experts. This observation suggests that enforcing the same sparsity level across all experts is inappropriate and does not accurately reflect their differing activation patterns. Moreover, expert outputs are combined through a weighted sum using routing weights. The contributions of different experts to the MoE layer output therefore have different scales. If these differences are ignored during neuron selection, the output computed using only the selected neurons cannot be expected to faithfully preserve the original output scale. To address these limitations of existing methods, we propose Sparser-MoE, which incorporates these MoE-specific characteristics to effectively apply Gated-MLP sparsification to MoE. Our contributions are as follows:

- We identify the limitations of imposing uniform sparsity across all selected top-K experts. We also point out that existing methods select neurons solely based on activation magnitudes, without accounting for the scale of their contributions to the MoE layer output.
- We propose Sparser-MoE to address these issues. Sparser-MoE dynamically adjusts sparsity across experts while maintaining the overall target sparsity. It also incorporates contribution-scale information into neuron scores, enabling the output of sparse computation to preserve the scale of the original output.
- Experiments across diverse models, model sizes, and benchmarks demonstrate that Sparser-MoE consistently preserves more of the original model performance than existing methods at the same sparsity level. We further show that incorporating sparse kernels achieves lower latency and higher throughput than dense computation, with improvements proportional to the sparsity level.

2 Related Work

Activation Sparsification. Prior work has observed that a substantial fraction of the output activations in $xW = o$ are zero or close to zero. Motivated by this observation, activation sparsification deactivates a subset of activations and retains only nonzero or large-magnitude activations, enabling sparse computation in the subsequent weight multiplication to reduce computational costs while closely approximating the original results (Li et al., 2022; Liu et al., 2023; Mirzadeh et al., 2024; Liu et al., 2025). For Gated-MLP sparsification, CATS (Lee et al., 2024) selects neurons based on the magnitudes of activations obtained after applying SiLU to the gate-projection outputs. TEAL (Liu et al., 2025) instead applies magnitude-based sparsification to hidden states throughout the model; when restricted to MLP activations, it follows the same general activation-magnitude principle as CATS. FloE (Zhou et al., 2025) and M-COUNTDOWN (Cheon & Kang, 2025) use the magnitudes

of the up-projection outputs. WINA (Chen et al., 2026) estimates a neuron's influence on the output of the subsequent operation by considering both its activation and the ℓ_2 norm of the corresponding weight vector used in that operation.

Parameter Merging & Pruning. Unlike activation sparsification, which dynamically selects neurons to execute based on the input, static parameter merging and pruning keep the parameter configuration or sparsity pattern fixed after compression. These methods use information such as calibration statistics and gradients to assess importance and redundancy at the weight, neuron, or expert level. MC-SMoE (Li et al., 2024) and HC-SMoE (Chen et al., 2024) merge multiple experts into fewer experts using routing statistics and clustering based on expert outputs, respectively. SparseGPT (Frantar & Alistarh, 2023) and Wanda (Sun et al., 2024) perform weight-level pruning, whereas LLM-Pruner (Ma et al., 2023) performs structured pruning by removing interconnected structures based on gradient information. MoNE (Zhang et al., 2026) evaluates expert redundancy using access frequency and output variance and replaces pruned experts with lightweight novices. Each novice is an input-independent constant vector given by an unbiased estimate of the corresponding expert's mean output over calibration inputs for which that expert is selected.

Although existing sparsification methods can be directly applied to MoE, they do not account for the aforementioned MoE-specific characteristics, such as differences in sparsity across expert ranks and the distinct scales at which expert outputs contribute to the MoE layer output. Furthermore, parameter merging and pruning irreversibly alter or remove parameters, requiring more parameters to be retained or post-training to be performed to preserve performance. In this paper, we propose a novel, training-free Gated-MLP sparsification method tailored to MoE that neither modifies parameters nor permanently removes them.

3 SPARSER-MOE

3.1 BACKGROUND

Gated-MLP in MoE. An MoE FFN consists of N experts, denoted by E_i, and a router parameterized by $W_{\text{router}} \in \mathbb{R}^{h \times N}$, where h is the model's hidden dimension. Given an input vector $x \in \mathbb{R}^{1 \times h}$, the router selects the top-K experts, each of which performs a Gated-MLP computation. Their outputs are then combined through a weighted sum using the corresponding routing weights. Each expert contains three weight matrices: $W^{(i)}_{\text{gate}}, W^{(i)}_{\text{up}} \in \mathbb{R}^{h \times h_{\text{mlp}}}$ and $W^{(i)}_{\text{down}} \in \mathbb{R}^{h_{\text{mlp}} \times h}$, where h_{mlp} is the intermediate dimension of the Gated-MLP. In this paper, we use SiLU as the activation function σ following the gate projection, as it is commonly adopted in recent pretrained models (Shazeer, 2020; Grattafiori et al., 2024; Yang et al., 2025). The main computations are expressed as follows:

$$y(x) = \sum_{i \in \mathcal{I}_K(x)} r_i(x) E_i(x), \tag{1}$$

$$r(x) = \text{Softmax}(\text{TopK}(xW_{\text{router}}, K)), \tag{2}$$

$$E_i(x) = \left(\sigma\left(xW^{(i)}_{\text{gate}}\right) \odot \left(xW^{(i)}_{\text{up}}\right)\right) W^{(i)}_{\text{down}}. \tag{3}$$

Here, $\mathcal{I}_K(x)$ denotes the set of selected expert indices, and $\odot$ denotes element-wise multiplication. The TopK operator retains the K largest logits and masks the remaining entries with $-\infty$, The subsequent softmax therefore assigns positive routing weights, normalized to sum to one, only to the selected experts and zero weight to all others.

Gated-MLP Sparsification. For our comparison, we consider three Gated-MLP neuron-scoring variants derived from prior activation-sparsification designs: scoring based on the gate projection, as in CATS (Lee et al., 2024); scoring based on the up projection, as in FloE and M-COUNTDOWN (Zhou et al., 2025; Cheon & Kang, 2025); and a gate-based score augmented with the ℓ_2 norm of the corresponding down-projection weight vector, inspired by WINA (Chen et al., 2026). For neuron j in expert i, the three scoring variants used in our comparison are defined as

follows:

$$s_{i,j}(x) = \begin{cases} \left|\mathrm{SiLU}\Big((xW^{(i)}_{\mathrm{gate}})_j\Big)\right|, & \text{gate-based,} \\ \left|(xW^{(i)}_{\mathrm{up}})_j\right|, & \text{up-based,} \\ \left|\mathrm{SiLU}\Big((xW^{(i)}_{\mathrm{gate}})_j\Big)\right| \left\|(W^{(i)}_{\mathrm{down}})_{j,:}\right\|_2, & \text{gate + down-norm.} \end{cases} \tag{4}$$

For each expert i, we estimate the neuron score distribution through offline calibration. Given a target sparsity p, we set the threshold τ_i to the p-quantile of this distribution and deactivate neurons whose scores fall below τ_i during inference. This procedure targets the removal of the bottom p fraction of scores under the calibration distribution, with the sparsity on actual inputs expected to be close to p. The binary selection mask for expert i, denoted by $\mathbf{S}_i(x)$, is defined element-wise as

$$S_{i,j}(x) = \mathbf{1}\{s_{i,j}(x) \geq \tau_i\}\,. \tag{5}$$

When the realized sparsity is close to p, the number of retained neurons across the selected top-K experts is approximately

$$\sum_{i \in \mathcal{I}_K(x)} \|\mathbf{S}_i(x)\|_0 \approx K h_{\mathrm{mlp}}(1-p). \tag{6}$$

As shown by the experimental results in Section 4, the three scoring approaches exhibit only minor differences in accuracy, with gate-projection-based scoring performing slightly better on average. We therefore base neuron scoring in Sparser-MoE on the gate projection. The sparse Gated-MLP computation is

$$\begin{aligned} \mathbf{g}^{\mathrm{full}}_i(x) &= \mathrm{SiLU}\Big(xW^{(i)}_{\mathrm{gate}}\Big)\,, \\ E_{i,\mathrm{sparse}}(x) &= \Big([\mathbf{g}^{\mathrm{full}}_i(x)]_{\mathbf{S}_i(x)} \odot x[W^{(i)}_{\mathrm{up}}]_{:,\mathbf{S}_i(x)}\Big)\,[W^{(i)}_{\mathrm{down}}]_{\mathbf{S}_i(x),:}. \end{aligned} \tag{7}$$

The subscript $\mathbf{S}_i(x)$ denotes Boolean indexing that gathers only entries for which $S_{i,j}(x) = 1$. Thus, the gate projection is evaluated in full, while only the selected up-projection columns and down-projection rows are multiplied; the method does not perform dense up/down projections followed by masking.

3.2 Methodology

This section presents the motivation and key components of Sparser-MoE. We introduce MoE-tailored neuron scoring and selection strategies, an optimized two-stage procedure for conventional threshold-based selection, and sparse kernels that efficiently execute the resulting sparse computation. In Section 4, experiments across diverse models and model scales show that Sparser-MoE outperforms the baselines on most benchmarks and improves throughput over dense computation.

Intrinsic Expert-Rank-Dependent Sparsity. Figure 1 shows that, under gate-projection-based neuron scoring, higher-ranked experts contain more neurons with large activation magnitudes. Consequently, assigning the same sparsity to all top-K experts can disproportionately restrict neuron selection in higher-ranked experts. A neuron in a higher-ranked expert may therefore be excluded even when its score exceeds that of a selected neuron in a lower-ranked expert. This observation motivates allowing sparsity to vary across expert ranks in Sparser-MoE.

A straightforward solution is to select scored neurons globally across all top-K experts rather than locally within each expert. This allows neurons from higher-ranked experts to account for a larger proportion of the selected neurons. Accordingly, Sparser-MoE adopts global rather than per-expert local neuron selection.

Neuron Scoring Based on Contribution to the FFN Output Scale. Unlike dense models, MoE organizes its FFN into multiple experts and ranks them through routing. Neuron selection must account for this structure, which motivates the global selection strategy described above. Another defining characteristic of an MoE FFN is that the outputs of the selected experts are combined through a weighted sum using their routing weights. Each expert's contribution to the final FFN output is therefore scaled by its routing weight, with higher-ranked experts receiving larger weights.

Table 1: Comparison of benchmark performance across different neuron scoring and selection methods on OLMoE-1B-7B at 80% sparsity. Full benchmark results and an additional routing-weight-aware local-selection setting are provided in Appendix A.

	BoolQ	SciQ	Arc-C	HellaSwag
dense	**74.3**	**91.0**	**54.9**	**79.3**
w/o r, local selection	65.1	86.9	45.5	71.1
w/o r, global selection	66.0	88.1	45.1	73.2
w/ r, global selection	**69.1**	**90.8**	**49.5**	**76.0**

Neurons in higher-ranked experts are thus more likely to contribute at a larger scale, and excluding these neurons from sparse computation can prevent the sparse output from preserving the original FFN output scale.

Existing approaches that rely solely on activation magnitudes for neuron scoring select neurons without considering routing weights. Such selection can therefore fail to preserve the final output scale when computation is restricted to the selected neurons. An intuitive remedy is to favor neurons from experts with larger routing weights. Accordingly, in addition to global selection, Sparser-MoE determines the final neuron score by multiplying the magnitude of each neuron's intermediate activation by its expert's routing weight. This allows neurons from experts with larger routing weights to be selected more frequently.

Table 1 compares benchmark performance under local and global selection, and further compares global selection with and without routing-weight-based scoring. The full results are provided in the Appendix A. Average benchmark performance is highest for global selection with routing weights, followed by global selection without routing weights and then local selection without routing weights. These results support the hypothesis that allowing heterogeneous sparsity to emerge through global selection yields better performance than imposing uniform sparsity across all top-K experts. Incorporating routing weights further concentrates neuron selection in higher-ranked experts compared with global selection alone. In other words, neurons from higher-ranked experts account for an even larger proportion of the selected neurons. The additional performance gains from routing-weight-based scoring suggest that accounting for the scale of each neuron's contribution to the final output is important, consistent with the preceding motivation.

Neuron Selection Optimization. To select the highest-scoring neurons at a target sparsity, one can sort all neuron scores and retain the TopK neurons. However, sorting all M scores incurs $O(M \log M)$ complexity, whereas existing approaches use threshold-based selection with $O(M)$ complexity. Here, $M = Kh_{\mathrm{mlp}}$ denotes the total number of neurons across the routed top-K experts. Although threshold-based selection is faster, it does not guarantee the target sparsity: it may retain either fewer or more neurons than required. Sparser-MoE employs a two-stage selection procedure to enforce the target sparsity, as in TopK selection, while targeting latency close to that of threshold-based selection.

First, we set a preliminary sparsity target $p' < p$ and determine a corresponding threshold τ_i' for each expert. The preliminary target p' is a hyperparameter; based on the results in Figure 2, we set $p' = p - 0.1$ in all experiments. Applying these thresholds reduces the number of candidate neurons from M to M', where $M' < M$ and $M' \approx M(1-p')$. We then apply TopK selection to the reduced set of M' candidates to meet the target sparsity.

However, Sparser-MoE combines global selection with routing-weight-based scoring, so the fraction of retained neurons varies across experts. In particular, the retention ratio of a higher-ranked expert may exceed the overall retention ratio $1 - p$, and may even exceed the preliminary retention ratio $1 - p'$. Consequently, calibrating every expert's threshold to the same preliminary sparsity target p' can discard neurons from higher-ranked experts that would otherwise be selected. Conversely, it can retain too many neurons from lower-ranked experts that would ultimately be discarded. Expert-specific preliminary sparsity targets are therefore needed.

To address this issue, Sparser-MoE adjusts each expert's preliminary candidate-retention ratio according to its routing weight, allowing experts with larger routing weights to retain more candidates. Higher-ranked experts consequently receive more permissive thresholds, whereas lower-ranked ex-

perts receive stricter thresholds. The threshold phase is thus designed to retain neurons that would survive the final selection while filtering out those that would not, yielding a reduced candidate set of size M'. The main algorithmic equations of Sparser-MoE are summarized as follows:

$$
\begin{aligned}
s_{i,j}(x) &= \left| \mathrm{SiLU}\Big(x W^{(i)}_{\mathrm{gate}}\Big)_j \right|, \\
p'_i &= \frac{r_i(x)}{\sum_{n=1}^{N} r_n(x)}(1-p'), \\
\tau'_i &= \mathrm{Quantile}(p'_i, \mathrm{Calib}), \\
\mathcal{C}(x) &= \{(i,j) \mid i \in \mathcal{I}_K(x),\ s_{i,j}(x) \geq \tau'_i\}, \\
K_{\mathrm{neuron}} &= K h_{\mathrm{mlp}}(1-p), \\
S_{i,j}(x) &= \mathbf{1}\{(i,j) \in \mathrm{TopKIdx}(\{r_u(x) s_{u,v}(x) \mid (u,v) \in \mathcal{C}(x)\}, K_{\mathrm{neuron}})\}, \\
E_{i,\mathrm{sparse}}(x) &= \Big([\mathbf{g}^{\mathrm{full}}_i(x)]_{\mathbf{s}_i(x)} \odot x[W^{(i)}_{\mathrm{up}}]_{:,\mathbf{s}_i(x)}\Big)\,[W^{(i)}_{\mathrm{down}}]_{\mathbf{s}_i(x),:}.
\end{aligned}
\tag{8}
$$

Here, TopKIdx selects K_{neuron} expert–neuron pairs from $\mathcal{C}(x)$, and their indicator directly forms the same binary mask $\mathbf{S}_i(x)$ used in Equation 7.

3.3 Sparse Kernel

The ultimate goal of Gated-MLP sparsification is to preserve model accuracy while improving throughput through reduced computation. Achieving this goal requires sparse kernels designed to exploit the reduction in FLOPs enabled by sparsity. Sparser-MoE performs GEMV for each token by compacting the indices of the selected neurons into a contiguous list rather than processing neurons separately for each expert. This avoids per-expert kernel launches and reduces unnecessary computation within tiles. To expose parallelism, we implement GEMV using block-based Triton kernels inspired by the custom sparse kernel design of CATS (Lee et al., 2024; Tillet et al., 2019). For the down projection, we additionally employ Split-K (NVIDIA Corporation, 2026; Tillet & Cox, 2017), allowing K_{split} workers to process neuron blocks concurrently. To avoid the overhead of atomic additions, each worker writes its partial results to a separate buffer. Once all workers have completed their down-projection computations, a reduction combines their partial results. The number of workers is chosen to improve streaming multiprocessor (SM) utilization for the given workload, and the reduction is skipped when $K_{\mathrm{split}} = 1$.

4 Experiments

4.1 Setup

Models. We evaluate Sparser-MoE on OLMoE-1B-7B (Muennighoff et al., 2024), DeepSeek-V2-Lite-16B (Liu et al., 2024), and Qwen3.5-35B-A3B (Qwen Team, 2026), covering diverse architectures and model sizes. All three models use SiLU as their activation function.

Datasets. We evaluate downstream task performance on eight datasets: WinoGrande (Sakaguchi et al., 2021), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), OpenBookQA (Mihaylov et al., 2018), HellaSwag (Zellers et al., 2019), BoolQ (Clark et al., 2019), ARC-Easy, and ARC-Challenge (Clark et al., 2018). The composition of this evaluation suite was informed by the benchmark configurations used in prior activation-sparsification studies (Lee et al., 2024; Liu et al., 2025; Zhou et al., 2025; Cheon & Kang, 2025; Chen et al., 2026), and the selected tasks collectively cover diverse domains. To determine the thresholds in Sparser-MoE, we perform calibration using C4 (Raffel et al., 2020), following the calibration settings used in SparseGPT (Frantar & Alistarh, 2023) and Wanda (Sun et al., 2024) as closely as possible. All latency and throughput measurements are conducted with a sequence length of 512, repeated five times, and reported as the mean.

Baselines. We consider three baselines distinguished by their neuron scoring strategies: `gate`, `up`, and `gate+l2_down`. The `gate` baseline scores neurons using the intermediate activations obtained after the gate projection and the subsequent activation function, as motivated by CATS (Lee

Table 2: Downstream task performance across various benchmarks at 90% sparsity. The corresponding results at 80% sparsity are provided in Appendix B. WG, OBQA, and HS denote WinoGrande, OpenBookQA, and HellaSwag, respectively.

Setting	Model / Method	WG	PIQA	SciQ	OBQA	HS	BoolQ	Arc-E	Arc-C	Avg.
dense	OlMoE-1B-7B	71.9	80.5	91.0	45.4	79.3	74.3	76.1	54.9	71.7
sparsity-90%	gate	55.4	71.3	73.8	35.2	59.8	62.5	54.2	37.8	56.3
	up	57.9	68.3	74.9	35.0	56.0	63.1	54.2	36.0	55.7
	gate+l2_down	55.4	70.9	73.5	35.8	59.2	61.7	53.5	37.6	56.0
	Sparser-MoE	**58.6**	**74.8**	**83.8**	**39.0**	**68.3**	**63.1**	**64.2**	**43.9**	**62.0**
dense	DeepSeek-V2-Lite-16B	76.0	80.4	93.4	43.6	80.7	80.6	76.3	56.5	73.4
sparsity-90%	gate	65.7	77.9	85.5	39.4	73.0	64.1	69.7	47.2	65.3
	up	64.5	75.2	89.9	38.4	70.5	70.9	63.4	42.1	64.4
	gate+l2_down	65.4	77.9	85.0	39.0	73.5	63.6	69.7	47.2	65.2
	Sparser-MoE	**69.6**	**78.7**	**87.8**	**43.0**	**77.1**	**71.1**	**71.3**	**49.6**	**68.5**
dense	Qwen3.5-35B-A3B	76.3	83.4	95.0	43.4	82.2	90.1	79.1	72.9	77.8
sparsity-90%	gate	69.6	80.0	93.5	42.6	79.4	86.1	78.5	68.9	74.8
	up	67.9	80.0	95.1	40.6	75.5	87.5	79.5	65.9	74.0
	gate+l2_down	70.6	79.8	93.1	43.0	79.4	86.6	78.6	68.8	75.0
	Sparser-MoE	**70.9**	**82.3**	**94.8**	**45.2**	**81.4**	**88.5**	**78.6**	**70.6**	**76.5**

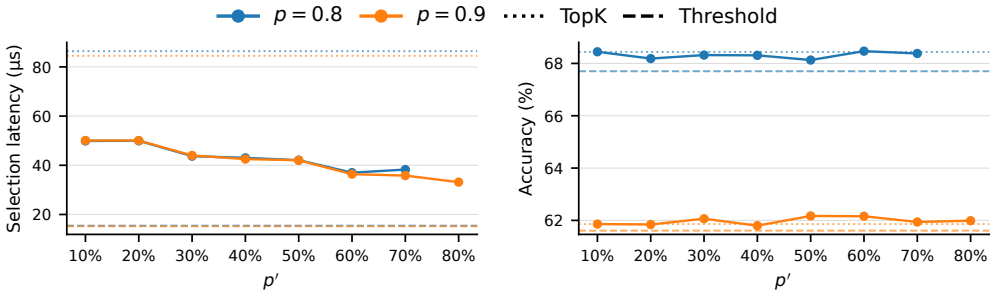
et al., 2024). It also reflects the general activation-magnitude principle that TEAL applies within MLP layers, although TEAL itself is not restricted to gate outputs (Liu et al., 2025). The `up` baseline uses the up-projection outputs for scoring, as motivated by FloE and M-COUNTDOWN (Zhou et al., 2025; Cheon & Kang, 2025). Finally, the `gate+l2_down` baseline combines the intermediate activations obtained after the gate projection and activation function with the ℓ_2 norms of the corresponding down-projection weight vectors. This design adopts WINA's principle of combining activation magnitudes with weight norms but is not a direct implementation of WINA (Chen et al., 2026). A faithful adaptation of WINA to a Gated-MLP would require computing the complete up-projection output for all neurons before selection, introducing substantial overhead and undermining the computational benefit of sparsification. We therefore construct this baseline using only the gate-projection activations together with the down-projection weight norms. In all experiments, Sparser-MoE applies sparsification only to routed experts and leaves shared experts dense. Shared experts tend to concentrate relatively important information, and we found that sparsifying them resulted in poorer performance retention.

Hardware & Environment. All experiments are conducted on a single RTX PRO 6000 GPU with 96 GB of memory. The main libraries used for implementation and execution are Triton 3.6.0 (Tillet et al., 2019), PyTorch 2.11.0+cu128 (Paszke et al., 2019), and Transformers 5.17.0 (Wolf et al., 2020).

4.2 Downsteam Task Performance

Table 2 and Table 5 in Appendix B compare the downstream task performance of Sparser-MoE and the baselines at sparsity levels of 90% and 80%, respectively. Sparser-MoE outperforms all baselines on nearly all benchmarks. In terms of average benchmark performance, Sparser-MoE outperforms the `up` baseline (FloE and M-COUNTDOWN) by up to 11.32% on OLMoE-1B-7B at 90% sparsity. Among the evaluated models, OLMoE-1B-7B also exhibits the largest performance variation across sparsity levels and methods. We attribute this greater sensitivity to its smaller scale, since smaller models generally exhibit less intrinsic sparsity (Li et al., 2022; Szatkowski et al., 2026).

These gains primarily stem from two aspects of Sparser-MoE. First, global selection allows more neurons to be selected from higher-ranked experts rather than restricting this natural tendency through uniform per-expert sparsity. Second, incorporating routing weights into neuron scoring favors neurons with larger contribution scales, helping preserve the scale of the final FFN output as closely as possible. In contrast, existing methods overlook these MoE-specific characteristics and cannot fully prioritize neurons from higher-ranked experts or those with larger contribution scales under a given sparsity constraint. Consequently, neurons from lower-ranked experts or those with smaller contribution scales may be selected instead, resulting in less effective preservation of model performance. Additionally, we show in Appendix C that applying global selection alone to the baselines can improve performance.



(a) Latency **(b)** Accuracy

Figure 2: Tuning the hyperparameter p'. (a) The combined latency of neuron scoring and selection is measured for every layer and reported as the mean across layers. (b) All benchmarks are evaluated, and the mean accuracy is reported. Experiments are conducted on OLMoE-1B-7B with a batch size of 32. OLMoE-1B-7B is selected for this analysis because the results in Table 2 and Appendix B show that it is the most sensitive to sparsification among the evaluated models.

4.3 Hyper-Parameter Tuning

As described in Section 3.2, neuron selection in Sparser-MoE consists of two stages. In the threshold phase, using a threshold τ_i calibrated to the target sparsity p can discard important neurons. To avoid this issue, we use a lower preliminary sparsity target $p' < p$ and the corresponding threshold τ_i'. Setting p' too close to p can prevent the selection of the ideal neuron set. Conversely, setting p' too far below p makes M' approach M, offering little advantage over applying TopK selection to all neurons. We therefore treat p' as a hyperparameter and evaluate the latency–accuracy trade-off in Figure 2 to identify its optimal value.

Figure 2(a) reports the combined latency of neuron scoring and selection. The results show that the TopK pipeline incurs approximately $5.58\times$ the latency of the threshold-based pipeline on average, while the scoring-and-selection latency of our two-stage procedure generally decreases as p' approaches p. The accuracy results in Figure 2(b) show that accuracy is preserved across all evaluated values of p', even when the gap is reduced to $p - p' = 0.1$. Accordingly, we set $p' = p - 0.1$ in all experiments to preserve accuracy while achieving scoring-and-selection latency close to that of threshold-based selection.

4.4 FFN Latency Breakdown

Figure 3 compares the FFN latency breakdown of Sparser-MoE and the dense baseline. The dense implementation consists of three stages. First, the gate and up projections are executed jointly in a fused kernel, as they can be computed concurrently. This is followed by the activation function and, finally, the down projection. In contrast, the sparse implementation of Sparser-MoE first performs the gate projection using dense computation, followed sequentially by neuron scoring, threshold-based filtering, and TopK selection. In the breakdown, the latencies of these three operations are aggregated into a single *Selection* component. Sparse up and down projections are then performed using the selected neurons. The latency of the sparse up and down projections decreases linearly as sparsity increases. Consequently, Sparser-MoE achieves lower latency than the dense baseline across all evaluated sparsity levels, reducing latency by up to 51.21% at 90% sparsity.

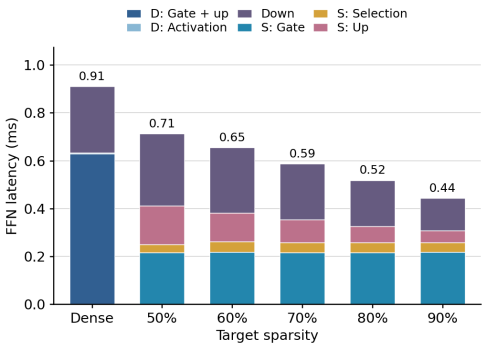
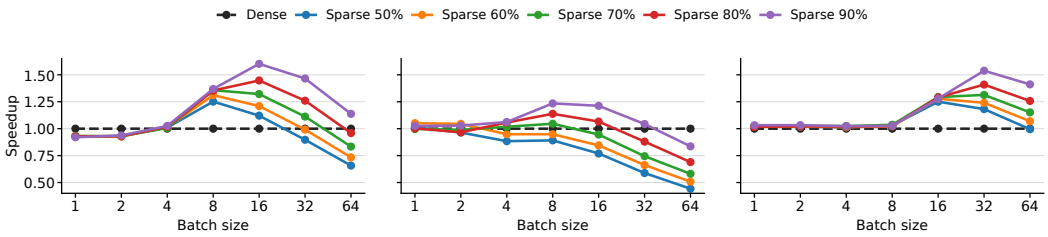


Figure 3: FFN latency breakdown by major kernel. The reduction kernel introduced by Split-K is included in the down-projection latency. Experiments are conducted on Qwen3.5-35B-A3B across different sparsity levels with a batch size of 32.



(a) OlMoE-1B-7B **(b)** DeepSeek-V2-Lite-16B **(c)** Qwen3.5-35B-A3B

Figure 4: Throughput comparison between Sparser-MoE and the dense baseline across different batch sizes and sparsity levels. Throughput improvements at each sparsity level are reported as speedups relative to the dense baseline. Prefill time is excluded from the measurements, and throughput in tokens per second (TPS) is computed using decoding time only.

4.5 Throughput

Finally, Figure 4 reports the throughput speedup of Sparser-MoE over the dense baseline across different batch sizes. Overall, little speedup is observed at small batch sizes. This is because the computational workload is small, allowing kernel overhead and neuron selection to dominate the total FFN latency. Although both models route each token to eight experts, Qwen3.5-35B-A3B has a smaller intermediate dimension, $h_{\text{mlp}} = 512$, than OLMoE-1B-7B, which has $h_{\text{mlp}} = 1024$. This appears to explain why the benefits of sparsification emerge at relatively larger batch sizes for Qwen3.5-35B-A3B.

DeepSeek-V2-Lite exhibits smaller overall performance gains than the other models because sparsification applies to a more limited portion of its computation. Unlike the other models, DeepSeek-V2-Lite contains one dense layer, and each of its MoE layers uses two shared experts and six routed experts per token. In comparison, Qwen3.5-35B-A3B uses one shared expert and eight routed experts per token, whereas OLMoE-1B-7B has no shared experts.

As the batch size increases, the speedup reaches approximately $1.6\times$. This is because larger batch sizes saturate the computational resources, allowing sparse kernels to benefit from their reduced computational workload. However, beyond a certain batch size, the speedup gradually declines, and throughput can even decrease. This behavior is attributable to differences in weight reuse. The dense implementation can group tokens routed to the same expert and launch a GEMM, increasing weight reuse with the number of tokens assigned to that expert. In contrast, the sparse implementation selects a different neuron subset for each token and therefore does not benefit from the same weight-reuse mechanism. Nevertheless, Sparser-MoE achieves throughput improvements at batch sizes ranging from 4 to 32 and, depending on the model, at substantially larger batch sizes.

5 Conclusion

In this paper, we propose Sparser-MoE, which, to our knowledge, is the first Gated-MLP sparsification approach to jointly introduce MoE-specific neuron scoring and selection strategies that incorporate routing weights and allocate neurons across selected experts. Sparser-MoE performs global neuron selection based on the observation that higher-ranked experts contain a greater number of important neurons. It also incorporates routing weights into neuron scoring, accounting for the larger scale at which neurons from experts with higher routing weights contribute to the final FFN output. Together, these strategies allow different proportions of neurons to be selected across experts, with more neurons retained in higher-ranked experts, helping preserve the original FFN output scale as closely as possible. Experiments demonstrate that Sparser-MoE consistently outperforms the baselines across diverse models, model sizes, and datasets. Furthermore, optimizations to neuron selection and sparse kernels reduce FFN latency and improve decoding throughput. Building on these strong results, we believe that Sparser-MoE's MoE-specific sparsification framework can serve as a starting point for future research in this area.

AI USE STATEMENT

AI was used in two main aspects of this work: writing and experimentation. All original content and mathematical formulations in this paper were written by the authors, while AI was used to assist with translation. AI was also used for tasks such as correcting LaTeX syntax and arranging tables and figures. All experimental ideas and the proposed method originated from the authors, while AI assistance was used for parts of the experimental implementation and figure design. The core logic of the implementation was designed by the authors, and any externally assisted ideas or experimental settings relevant to the work are explicitly described in the main paper, including the sparse kernel design, calibration dataset, and benchmark datasets (see Section 3.3 and Section 4.1). AI-generated code was manually inspected by examining both the code and execution logs. The experiments were further validated by directly tuning and verifying the detailed experimental configurations.

REPRODUCIBILITY STATEMENT

For reproducibility, all key information required to reproduce our experiments, including the models, datasets, environment, and experimental configurations, is provided in Section 4.1. The main algorithmic formulations are presented in Section 3.2, and the key design details of the sparse kernel are described in Section 3.3. All experiments in this paper can be reproduced on a single RTX PRO 6000 GPU within a few days. To facilitate direct reproduction, we will provide the anonymized code and scripts used in our experiments as supplementary material.

REFERENCES

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

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

Sihan Chen, Dan Zhao, Jongwoo Ko, Colby Banbury, Huiping Zhuang, Luming Liang, Pashmina Cameron, and Tianyi Chen. WINA: Weight informed neuron activation for accelerating large language model inference. In *International Conference on Learning Representations*, volume 2026, pp. 122858–122880, 2026.

Jaewon Cheon and Pilsung Kang. COUNTDOWN: Contextually sparse activation filtering out unnecessary weights in down projection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 28381–28397, 2025.

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

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

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

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pp. 10323–10337. PMLR, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Jingyi Jia and Qinbin Li. AutoTool: Efficient tool selection for large language model agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 31265–31273, 2026.

Jiin Kim, Byeongjun Shin, Jinha Chung, and Minsoo Rhu. The cost of dynamic reasoning: Demystifying AI agents and test-time scaling from an AI infrastructure perspective. In *2026 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 1–16. IEEE, 2026.

Donghyun Lee, Je-Yong Lee, Genghan Zhang, Mo Tiwari, and Azalia Mirhoseini. CATS: Contextually-aware thresholding for sparsity in large language models. *arXiv preprint arXiv:2404.08763*, 2024.

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

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. Merge, then compress: Demystify efficient SMoE with hints from its routing policy. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 14234–14256, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/3d09a88c3372cdb79401fde592ca4db8-Paper-Conference.pdf.

Zonglin Li, Chong You, Srinadh Bhojanapalli, Daliang Li, Ankit Singh Rawat, Sashank J Reddi, Ke Ye, Felix Chern, Felix Yu, Ruiqi Guo, et al. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. *arXiv preprint arXiv:2210.06313*, 2022.

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

James Liu, Pragaash Ponnusamy, Tianle Cai, Yoon Kim, Ben Athiwaratkun, et al. Training-free activation sparsity in large language models. In *International Conference on Learning Representations*, volume 2025, pp. 98302–98322, 2025.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. Deja vu: Contextual sparsity for efficient LLMs at inference time. In *International Conference on Machine Learning*, pp. 22137–22176. PMLR, 2023.

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

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2381–2391, 2018.

Iman Mirzadeh, Keivan Alizadeh-Vahid, Sachin Mehta, Carlo C Del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. ReLU strikes back: Exploiting activation sparsity in large language models. In *International Conference on Learning Representations*, volume 2024, pp. 25378–25400, 2024.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. OLMoE: Open mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2409.02060.

NVIDIA Corporation. Efficient GEMM in CUDA. https://docs.nvidia.com/cutlass/latest/media/docs/cpp/efficient_gemm.html, 2026. NVIDIA CUTLASS Documentation. Last updated September 21, 2026; accessed September 24, 2026.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

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

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

Nikhil Sardana, Jacob Portes, Sasha Doubov, and Jonathan Frankle. Beyond Chinchilla-optimal: Accounting for inference in language model scaling laws. *arXiv preprint arXiv:2401.00448*, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

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

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

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024.

Filip Szatkowski, Patryk Będkowski, Alessio Devoto, Jan Dubiński, Pasquale Minervini, Mikolaj Piórczynski, Simone Scardapane, and Bartosz Wójcik. Universal properties of activation sparsity in modern large language models. In *International Conference on Learning Representations*, volume 2026, pp. 146247–146269, 2026.

Philippe Tillet and David Cox. Input-aware auto-tuning of compute-bound HPC kernels. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pp. 1–12, 2017.

Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019.

Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, 2017.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Geng Zhang, Han Yuxuan, Yuxuan Lou, Yiqi Zhang, Wangbo Zhao, and Yang You. MoNE: Replacing redundant experts with lightweight novices for structured pruning of MoE. In *International Conference on Learning Representations*, volume 2026, pp. 39983–40009, 2026.

Yuxin Zhou, Zheng Li, Jun Zhang, Jue Wang, Yiping Wang, Zhongle Xie, Ke Chen, and Lidan Shou. FloE: On-the-fly MoE inference on memory-constrained GPU. *arXiv preprint arXiv:2505.05950*, 2025.

A Performance across Different Neuron Scoring and Selection Methods

Tables 3 and 4 provide the full benchmark results at 90% and 80% sparsity, respectively, for the gate-based baseline described in Section 4.1 and the scoring and selection variants discussed in Section 3.2. We also include w/ r, local selection, a setting omitted from the main-text comparison. It distributes the total neuron budget among routed experts in proportion to their routing weights while maintaining the target sparsity, then selects the TopK neurons independently within each expert. Unlike Sparser-MoE (w/ r, global selection), it does not rank neurons across expert boundaries.

This setting tests whether the benefit of global selection can be explained solely by routing-weight-dependent allocation. As motivated by Figure 1, global selection allows higher-ranked experts to retain more neurons without imposing a uniform expert-wise budget. However, weighting neuron scores by routing weights may itself induce heterogeneous sparsity across experts. Routing-weight-proportional local selection separates this allocation effect from global ranking: if it performed similarly to Sparser-MoE, global selection would offer little additional benefit over assigning different expert-wise budgets.

Table 3: Benchmark results for OLMoE-1B-7B at 90% sparsity under different neuron scoring and selection methods. WG, OBQA, and HS denote WinoGrande, OpenBookQA, and HellaSwag, respectively.

	WG	PIQA	SciQ	OBQA	HS	BoolQ	Arc-E	Arc-C	Avg.
dense	**71.9**	**80.5**	**91.0**	**45.4**	**79.3**	**74.3**	**76.1**	**54.9**	**71.7**
w/o r, local selection	55.4	71.3	73.8	35.2	59.8	62.5	54.2	37.8	56.3
w/o r, global selection	57.6	72.6	76.1	36.2	62.6	62.1	56.6	40.1	58.0
w/ r, local selection	56.5	74.5	79.6	38.0	63.8	62.9	58.4	41.5	59.4
w/ r, global selection	**58.6**	**74.8**	**83.8**	**39.0**	**68.3**	**63.1**	**64.2**	**43.9**	**62.0**

Table 4: Benchmark results corresponding to Table 1 for OLMoE-1B-7B at 80% sparsity, including routing-weight-aware local selection. WG, OBQA, and HS denote WinoGrande, OpenBookQA, and HellaSwag, respectively.

	WG	PIQA	SciQ	OBQA	HS	BoolQ	Arc-E	Arc-C	Avg.
dense	**71.9**	**80.5**	**91.0**	**45.4**	**79.3**	**74.3**	**76.1**	**54.9**	**71.7**
w/o r, local selection	62.6	75.1	86.9	41.8	71.1	65.1	66.2	45.5	64.3
w/o r, global selection	62.8	76.4	88.1	41.0	73.2	66.0	67.3	45.1	65.0
w/ r, local selection	64.6	76.7	87.7	42.0	74.0	67.0	69.4	49.5	66.4
w/ r, global selection	**65.0**	**78.6**	**90.8**	**44.0**	**76.0**	**69.1**	**71.9**	**49.5**	**68.1**

At both sparsity levels, w/ r, global selection achieves higher average accuracy than w/ r, local selection (68.1% versus 66.4% at 80% sparsity and 62.0% versus 59.4% at 90% sparsity). The largest benchmark-level gain occurs on ARC-Easy at 90% sparsity, where accuracy increases from 58.4% to 64.2%, a 9.9% relative improvement. These results suggest that routing-weight-aware expert-wise allocation is useful, but global selection provides an additional benefit by allowing high-scoring neurons to compete across experts rather than restricting them to predetermined local budgets.

B Downstream Task Performance at 80% Sparsity

Table 5: Downstream task performance across various benchmarks at 80% sparsity. WG, OBQA, and HS denote WinoGrande, OpenBookQA, and HellaSwag, respectively.

Setting	Model / Method	WG	PIQA	SciQ	OBQA	HS	BoolQ	Arc-E	Arc-C	Avg.
dense	OlMoE-1B-7B	71.9	80.5	91.0	45.4	79.3	74.3	76.1	54.9	71.7
sparsity-80%	gate	62.6	75.1	86.9	41.8	71.1	65.1	66.2	45.5	64.3
	up	63.5	74.0	85.2	38.6	69.8	63.8	66.3	43.5	63.1
	gate+l2_down	58.7	75.0	85.3	39.6	68.6	64.0	63.4	44.5	62.4
	Sparser-MoE	**65.0**	**78.6**	**90.8**	**44.0**	**76.0**	**69.1**	**71.9**	**49.5**	**68.1**
dense	DeepSeek-V2-Lite-16B	76.0	80.4	93.4	43.6	80.7	80.6	76.3	56.5	73.4
sparsity-80%	gate	66.9	78.6	85.9	40.4	76.0	70.3	70.3	49.1	67.2
	up	70.6	77.5	91.9	41.4	76.8	75.4	70.0	48.0	69.0
	gate+l2_down	66.7	78.2	86.3	40.4	75.7	68.5	70.4	48.7	66.9
	Sparser-MoE	**71.4**	**80.3**	**90.4**	**42.4**	**79.5**	**76.1**	**73.3**	**51.7**	**70.6**
dense	Qwen3.5-35B-A3B	76.3	83.4	95.0	43.4	82.2	90.1	79.1	72.9	77.8
sparsity-80%	gate	70.5	81.3	93.5	45.0	81.3	87.7	79.3	71.0	76.2
	up	71.0	81.9	95.3	41.0	79.4	89.2	80.9	70.1	76.1
	gate+l2_down	70.7	80.9	94.0	45.6	80.8	87.9	79.0	71.2	76.3
	Sparser-MoE	**74.3**	**82.3**	**94.7**	**42.6**	**82.2**	**89.3**	**78.4**	**72.2**	**77.0**

C Global Selection Ablation

Table 6: Effect of global neuron selection on downstream task performance for OLMoE-1B-7B at 90% sparsity. Avg. (w/o) reports the corresponding average without global selection. WG, OBQA, and HS denote WinoGrande, OpenBookQA, and HellaSwag, respectively.

Model / Method	WG	PIQA	SciQ	OBQA	HS	BoolQ	Arc-E	Arc-C	Avg.	Avg. (w/o)
OLMoE-1B-7B (dense)	71.9	80.5	91.0	45.4	79.3	74.3	76.1	54.9	71.7	71.7
gate (90%)	58.9	73.4	74.5	36.2	64.5	62.3	57.0	40.4	58.4	56.3
up (90%)	57.4	69.1	76.6	36.0	58.8	62.7	56.1	36.3	56.6	55.7
gate+l2_down (90%)	56.8	71.8	74.8	35.8	62.2	62.2	55.8	37.2	57.1	56.0
Sparser-MoE (90%)	**58.6**	**74.8**	**83.8**	**39.0**	**68.3**	**63.1**	**64.2**	**43.9**	**62.0**	–

Table 6 reports results obtained by adding only global selection to the baselines. Although these variants still underperform Sparser-MoE, suggesting that they do not fully capture the characteristics of MoE, global selection alone provides a meaningful performance improvement. This result demonstrates that, rather than enforcing the same sparsity level across all experts, allowing neurons from higher-ranked experts to be selected more naturally through global selection can improve performance.

D Limitation and Future Work

Sparser-MoE still has several limitations. First, our evaluation is restricted to models with up to tens of billions of parameters and single-GPU workloads. In addition, sparsification is not applied to shared experts in our experiments. Although Sparser-MoE achieves clear throughput gains through sparsification, these gains are limited to a certain range of batch sizes; outside this range, throughput may remain unchanged or even decrease. Future work will therefore focus on extending Sparser-MoE to settings such as distributed inference for trillion-parameter models, applying sparsification to shared experts and a broader set of model parameters, and sustaining throughput gains across a wider range of batch sizes.