Analytical FFN-to-MoE Restructuring via Activation Pattern Analysis

Zehua Pei[1], **Hui-Ling Zhen**[2], **Lancheng Zou**[1], **Xianzhi Yu**[2], **Wulong Liu**[2], **Sinno Jialin Pan**[1], **Mingxuan Yuan**[2], **Bei Yu**[1]
[1]The Chinese University of Hong Kong [2]Huawei Technologies Co., Ltd

Abstract

Scaling large language models (LLMs) improves performance but significantly increases inference costs, with feed-forward networks (FFNs) consuming the majority of computational resources. While Mixture-of-Experts (MoE) architectures can reduce this cost through sparse activation, restructuring existing dense models into MoEs typically requires extensive retraining on hundreds of billions of tokens. We propose an analytical post-training framework that rapidly restructures FFNs into sparse MoE architectures using only a small calibration dataset. The method analyzes neuron activation patterns to partition neurons into always-active shared experts and conditionally activated routed experts, then constructs a router analytically from representative neuron statistics, enabling immediate deployment or optional lightweight fine-tuning. This approach applies both to dense models and recursively to existing MoE models for hierarchical sparsity. Experiments demonstrate up to $1.17\times$ speedup in compute-bound scenarios with only minutes of processing and 2k-sample fine-tuning, outperforming methods requiring orders of magnitude more resources.[1]

1 Introduction

Large language models (LLMs) have achieved strong performance on a wide range of tasks (Zhang et al., 2022; Touvron et al., 2023; Liu et al., 2024b,a), but their ever-growing size presents deployment challenges due to high computational demands, especially on resource-constrained hardware or under strict latency budgets. This has spurred the development of various inference acceleration techniques. Among these, the mixture-of-experts (MoE) architecture (Lepikhin et al., 2020; Du et al., 2022; Fedus et al., 2022; Dai et al., 2024) decouples model capacity from computational cost by using a router to dynamically select a sparse subset of parameters for each input token. However, reaping the benefits of MoE models has traditionally required expensive pre-training from scratch, establishing a challenging trade-off between model performance and training cost.

The computational bottleneck in modern transformer architectures is disproportionately located in the feed-forward network (FFN) blocks. Several studies have reported high activation sparsity in FFN neurons (Liu et al., 2023; Zhang et al., 2021; Pei et al., 2024), meaning only a small fraction of neurons activate for any given input. This natural sparsity presents an opportunity to accelerate inference without the cost of pre-training. Prior work has explored restructuring dense models into MoEs, but existing methods either rely on weight-based clustering (Zhang et al., 2021; Qiu et al., 2023) that treats all neurons uniformly regardless of their activation behavior, or require extensive continual training (Zhu et al., 2024; Qu et al., 2024) with up to 200 billion tokens to recover quality. A key observation overlooked by these methods is that neuron activation frequencies exhibit two distinct groups: the majority of neurons activate only for specific inputs, while a subset activates consistently across inputs. By treating all neurons uniformly, prior approaches scatter consistently active neurons across experts, requiring extensive training to learn effective routing. This motivates an analytical approach that explicitly leverages this distinct structure: grouping high-frequency neurons into shared experts and low-frequency neurons into routed experts, with routing derived directly from activation statistics.

To overcome these limitations, we propose an analytical post-training framework that improves the performance-cost trade-off for LLM acceleration. The framework restructures FFNs through a rapid, analytical process using only a tiny calibration dataset. It operates by analyzing neuron acti-

[1]Code: https://github.com/JarvisPei/CMoE

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 4777–4789
July 2-7, 2026

vation patterns to distinguish frequently active neurons (grouped into 'shared' experts) from sparsely active ones. The sparsely active neurons are then clustered into specialized 'routed' experts using a balanced assignment algorithm (Jonker and Volgenant, 1988). This restructuring is broadly applicable: it can transform a dense model's single, large FFN into a sparse MoE architecture, or it can be applied recursively to the individual experts of an existing MoE model to induce a finer-grained hierarchical sparsity. The framework constructs a router analytically from activation statistics, bypassing the need for expensive router training and enabling rapid deployment with a training-free baseline or optional lightweight fine-tuning.

Our contributions are:

- We identify two distinct groups in FFN neuron activation frequencies: a subset activates consistently across inputs while the majority activates conditionally. We show that prior FFN-to-MoE methods overlook this structure.
- We propose an analytical framework that leverages this observation to partition neurons into shared and routed experts, constructing a router directly from representative neuron statistics without training.
- The method outperforms prior MoE restructuring methods in accuracy with only 2k-sample fine-tuning, and achieves up to 1.17× inference speedup. It applies to both dense models and existing MoE architectures for hierarchical sparsity.

2 Related Work

In contrast to pretraining MoE models from scratch, recent research has investigated constructing MoE architectures by repurposing existing dense LLMs. Current methodologies generally follow two paradigms: (1) partitioning FFN parameters while preserving total parameter count (Zuo et al., 2022; Zhang et al., 2021; Yang et al., 2024), or (2) expanding capacity while retaining activation dimensions (Komatsuzaki et al., 2022; Wu et al., 2024). This work prioritizes the former. MoEBERT (Zuo et al., 2022) redistributes top-scoring neurons across experts using an importance-driven strategy; while it also shares important neurons, it duplicates them inside every expert rather than forming structurally separate shared experts, and relies on task-specific gradient-based importance scores rather than task-agnostic activation profiling. MoEfication (Zhang et al., 2021) leverages sparse activation patterns in ReLU-based FFNs, decomposing layers into expert groups with a learned router. G-MoEfication (Lee et al., 2024) generalizes MoEfication to non-ReLU models by retaining representative values for unselected experts. LLaMA-MoE (Zhu et al., 2024) and its successor (Qu et al., 2024) partition FFNs into experts but require extensive continual training (200B and 7B tokens, respectively). EMoE (Qiu et al., 2023) clusters neurons by key vectors during fine-tuning, while Read-ME (Cai et al., 2024) focuses on domain-aware expert construction with system co-design. In contrast, our method analytically restructures FFNs by partitioning neurons into shared and routed experts based on activation patterns, then constructs a router from representative neuron statistics, requiring only 2k-sample fine-tuning.

A parallel line of work studies fully differentiable routing. ReMoE (Wang et al., 2024) replaces hard Top-K with ReLU routing, and Lory (Zhong et al., 2024) performs differentiable expert merging at large token budgets. These methods learn routers during pre-training, whereas our training-light analytical restructuring is complementary.

Orthogonal to FFN-to-MoE conversion, other efficiency techniques include structured pruning methods such as SliceGPT (Ashkboos et al., 2024) and SLEB (Song et al., 2024), which remove model components statically. Activation sparsity methods exploit natural sparsity in FFN hidden states: DejaVu (Liu et al., 2023) leverages contextual sparsity, while TEAL (Liu et al., 2024c) and WINA (Chen et al., 2025) use magnitude- or weight-informed thresholds. Learn-To-be-Efficient (Zheng et al., 2024) trains models to activate fewer neurons. These approaches operate at different granularities and can be complementary to our MoE restructuring.

3 Motivation: Activation Patterns

Before presenting our method, we analyze the activation patterns in FFN layers. These observations inform the design of our analytical restructuring approach.

3.1 High Activation Sparsity

Consider an FFN layer that takes input $\mathbf{x} \in \mathbb{R}^d$ and produces output $F(\mathbf{x}) \in \mathbb{R}^d$. The computation involves a hidden state $\mathbf{h} \in \mathbb{R}^{d_h}$, where d_h is

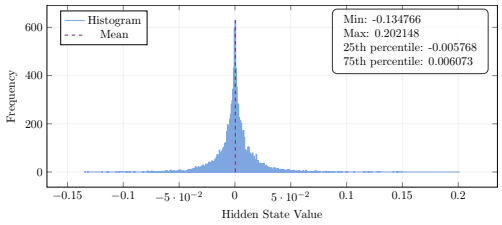

Figure 1: Distribution of FFN hidden state $\mathbf{h}$ (Llama-2-7B). Most activations concentrate near zero.

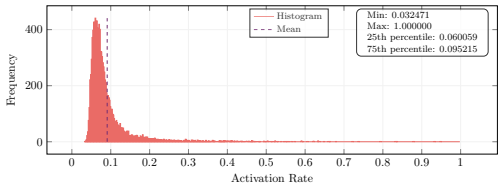

Figure 2: Distribution of neuron activation rates $\boldsymbol{\mu}$ (Llama-2-7B, $K_a = 1000$ for visualization; similar patterns hold at smaller K_a). Most neurons have low activation rates, while a subset is consistently active.

the hidden dimension (typically $d_h \gg d$). Each neuron's contribution to the output can be written as:

$$F(\mathbf{x}) = \sum_{i=1}^{d_h} h_i \mathbf{w}_i, \tag{1}$$

where h_i is the i-th hidden activation and $\mathbf{w}_i \in \mathbb{R}^d$ is the corresponding column of the output projection.

Figure 1 shows the distribution of hidden activations for a representative layer. The distribution is sharply peaked at zero, indicating that most neurons contribute negligibly to the output for any given input. This sparsity suggests that selectively activating neurons could reduce computation while preserving output quality. However, to exploit this, we need to understand *which* neurons are important for *which* inputs.

3.2 Diverse Activation Patterns

To characterize neuron behavior across inputs, we compute activation rates over a calibration set. For each neuron i, we define its activation rate μ_i as the fraction of tokens for which it ranks among the top-K_a activations by magnitude, where K_a is a hyperparameter specifying how many top activations to consider.

Figure 2 reveals two distinct groups. The majority of neurons exhibit low activation rates (peak near 0.07), meaning they activate only for specific inputs. However, a subset of neurons shows consistently high activation rates approaching 1, indicating they contribute to nearly all inputs. This distinction suggests a natural partition: neurons that are always important versus neurons that are conditionally important.

This observation has implications for MoE design. Prior methods (Zhang et al., 2021; Qiu et al., 2023) treat all neurons uniformly when constructing experts, ignoring this bimodal pattern. Since these methods do not distinguish consistently active neurons from conditionally active ones, high-frequency neurons may be scattered across different routed experts. This forces the router to activate most experts regardless of input, undermining the sparsity that MoE relies on for efficiency. In contrast, explicitly separating high-frequency neurons into always-active shared experts and grouping low-frequency neurons into conditionally activated routed experts yields a more structured architecture where the router only needs to select among genuinely input-dependent experts.

3.3 Problem Formulation

Based on these observations, we formulate the restructuring objective. Given the original FFN output $F(\mathbf{x})$, we seek an MoE architecture that minimizes reconstruction error:

$$\min \mathbb{E}_{\mathbf{x}} \left[\|F_{MoE}(\mathbf{x}) - F(\mathbf{x})\|^2 \right]. \tag{2}$$

The MoE output is defined as $F_{MoE}(\mathbf{x}) = E^s(\mathbf{x}) + \sum_{i=1}^{N_r} g_i \cdot E_i^r(\mathbf{x})$, where E^s is a shared expert that is always active, $\{E_i^r\}_{i=1}^{N_r}$ are N_r routed experts, and $g_i \in \{0, 1\}$ are gate values produced by a router.

Since experts are constructed by partitioning the original neurons without adding parameters, the objective reduces to determining which routed experts to deactivate. When some routed experts are deactivated (receiving $g_i = 0$), the reconstruction error equals the sum of their outputs. To minimize this error, we should construct the expert partition and router such that deactivated experts have minimal expected contribution.

The observations above suggest that shared experts should contain high-frequency neurons, routed experts should group co-activated neurons, and the router can be derived from activation patterns. The following section presents our analytical solution.

4 Methodology

Building on the problem formulation in Section 3.3, we now present our analytical solution. The framework operates in three stages: (A) profiling neuron

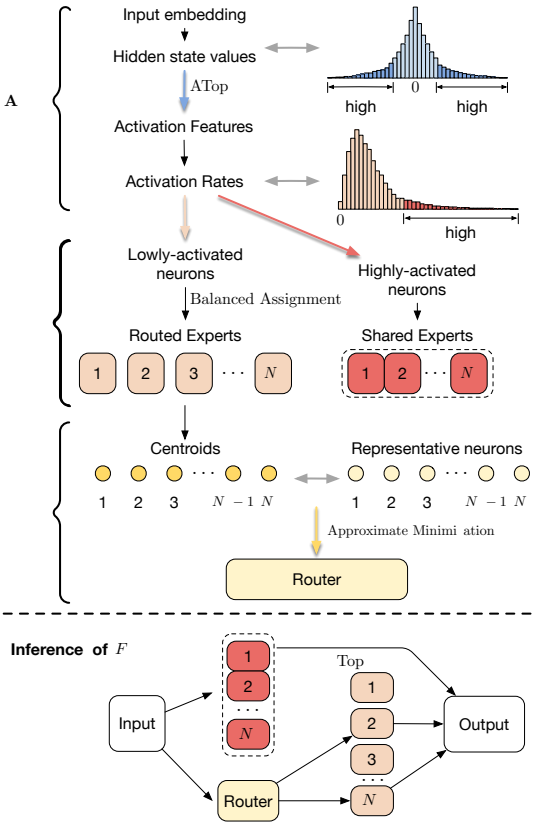

Figure 3: Overview of the proposed analytical FFN-to-MoE restructuring framework.

activation patterns, (B) partitioning neurons into shared and routed experts, and (C) constructing an analytical router. Figure 3 illustrates the overall pipeline.

4.1 Expert Construction

For an FFN with hidden dimension d_h, we construct N experts of size $m = d_h/N$, comprising N_s shared experts and N_r routed experts ($N_s + N_r = N$). For LLaMA-style models with SwiGLU activation, the FFN computes:

$$\begin{aligned} \mathbf{h} &= \text{Swish}(\mathbf{W}_{\text{gate}}^{\top}\mathbf{x}) \odot (\mathbf{W}_{\text{up}}^{\top}\mathbf{x}), \\ F(\mathbf{x}) &= \mathbf{W}_{\text{down}}^{\top}\mathbf{h}, \end{aligned} \tag{3}$$

with $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{W}_{\text{up}}, \mathbf{W}_{\text{gate}} \in \mathbb{R}^{d \times d_h}$, and $\mathbf{W}_{\text{down}} \in \mathbb{R}^{d_h \times d}$.

Activation Profiling. Using a small calibration dataset, we compute hidden states $\mathbf{H} \in \mathbb{R}^{q \times d_h}$ for q tokens. For each token, we identify the top-K_a neurons by activation magnitude, i.e., an absolute top-K (ATopK) selection over $|h_i|$, yielding a binary activation matrix $\mathbf{A} \in \{0,1\}^{q \times d_h} = [\mathbf{c}_1\ \mathbf{c}_2\ \cdots\ \mathbf{c}_{d_h}]$. Each column $\mathbf{c}_i$ represents neuron i's activation pattern across the calibration set, and the activation rate $\mu_i = \text{mean}(\mathbf{c}_i)$ measures how frequently neuron i is active (see Appendix A.2 for the complete pipeline).

Shared Experts. Based on the diverse activation pattern observed in Section 3.2, we select the $N_s \cdot m$ neurons with highest activation rates to form the shared expert E^s. These neurons are consistently active across inputs and capture common knowledge. The shared expert weights are constructed by slicing the original FFN matrices according to the selected neuron indices.

Routed Experts. The remaining neurons are partitioned into N_r routed experts. Since neurons with similar functions tend to co-activate, we cluster them based on the similarity of their activation feature vectors $\mathbf{c}_i$. We employ a balanced assignment algorithm that groups neurons into N_r equal-sized clusters while minimizing intra-cluster distance. Details are provided in Appendix A.3.

The resulting MoE architecture computes:

$$F_{MoE}(\mathbf{x}) = E^s(\mathbf{x}) + \sum_{i=1}^{N_r} g_i \cdot E_i^r(\mathbf{x}), \tag{4}$$

where $g_i \in \{0,1\}$ are gate values and only the top-N_k routed experts (by router score) are activated, with N_k denoting the number of routed experts selected per input.

4.2 Analytical Router Construction

From the problem formulation in Section 3.3, minimizing reconstruction error requires that deactivated experts contribute minimally to the output. Let S_{de} denote the set of deactivated routed experts. The reconstruction error is:

$$F_{MoE}(\mathbf{x}) - F(\mathbf{x}) = -\sum_{i \in S_{de}} E_i^r(\mathbf{x}). \tag{5}$$

Each expert's output can be decomposed as a sum over its neurons' contributions. Under the sparsity observation from Section 3.1, neurons with small hidden activations contribute negligibly (see Appendix A.1 for the formal hypothesis). This suggests that the L_1 magnitude of an expert's hidden state $\|\mathbf{h}_i^r\|_1$ serves as a proxy for its output magnitude. Thus, minimizing reconstruction error approximately reduces to (full derivation in Appendix A.4):

$$\min_G \mathbb{E}_{\mathbf{x}} \left[\sum_{i \in S_{de}} \|\mathbf{h}_i^r\|_1 \right], \tag{6}$$

where G is the router that determines S_{de} via top-N_k selection.

This objective is minimized when the router scores $\mathbf{s} = [s_1, \ldots, s_{N_r}]$ rank experts by their expected hidden state magnitude, ensuring that experts with larger contributions are activated while those with smaller contributions are deactivated.

Representative Neuron Selection. To construct such a router analytically, we identify a **representative neuron** R_j for each expert j as the neuron whose activation pattern is closest to the cluster centroid $\hat{\mathbf{c}}_j$:

$$R_j = \underset{i \in \text{cluster } j}{\operatorname{argmin}} \|\mathbf{c}_i - \hat{\mathbf{c}}_j\|_2, \tag{7}$$

where $\hat{\mathbf{c}}_j$ is the centroid of cluster j obtained from the balanced clustering step. Since R_j best represents the expert's typical activation behavior, its hidden activation h_{R_j} approximates the expert's overall contribution.

The router is then constructed using only the representative neurons' parameters:

$$G(\mathbf{x}) = \text{Swish}(\mathbf{W}_{\text{gate}}^{R\top}\mathbf{x}) \odot (\mathbf{W}_{\text{up}}^{R\top}\mathbf{x}), \tag{8}$$

where $\mathbf{W}_{\text{gate}}^{R}, \mathbf{W}_{\text{up}}^{R} \in \mathbb{R}^{d \times N_r}$ contain only the columns corresponding to representative neurons. This yields router scores $\mathbf{s} = [s_1, \ldots, s_{N_r}] = G(\mathbf{x})$ that approximate each expert's expected contribution, enabling effective routing.

4.3 Fine-tuning Enhancements

The analytical router provides a training-free baseline. For further improvement, we introduce two enhancements for optional lightweight fine-tuning.

Learnable Scaling. The initial gate values are binary ($g_i \in \{0, 1\}$). To enable gradient-based optimization, we introduce learnable scaling parameters $\mathbf{u} = [u_1, \ldots, u_{N_r}]$, initialized to zero. For selected experts, the gate becomes $g_i = 1 + s'_i \cdot u_i$, where $\mathbf{s}' = \text{Softmax}(\mathbf{s})$.

Load Balancing. To ensure balanced expert utilization without auxiliary losses, we introduce adaptive bias terms $\mathbf{b} = [b_1, \ldots, b_{N_r}]$ added to scores before top-N_k selection (Liu et al., 2024a). The final gating logic is:

$$g_i = \begin{cases} 1 + s'_i \cdot u_i, & \text{if } s'_i + b_i \in \text{Top-}N_k, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The biases are updated after each step: if expert i is overloaded ($p_i > p^*$), we decrease b_i by γ; if underloaded ($p_i < p^*$), we increase b_i by γ, where p_i is expert i's utilization fraction, $p^* = 1/N_r$ is the uniform target, and $\gamma = 10^{-3}$.

4.4 Application to Existing MoE Models

The framework applies not only to dense FFNs but also to existing MoE models. For an MoE layer with experts $\{E_i\}$, we apply our restructuring to each expert individually, transforming it into a hierarchical structure with shared and routed sub-experts:

$$E_i(\mathbf{x}) \to E_i^s(\mathbf{x}) + \sum_{j=1}^{N'_r} g'_{i,j} \cdot E_{i,j}^r(\mathbf{x}). \tag{10}$$

This creates a two-level hierarchy: the top-level router selects primary experts, and within each activated expert, a sub-router selects specialized sub-experts. This induces finer-grained sparsity for further acceleration.

5 Experiments

We evaluate the proposed framework as a post-training sparsification method for inference acceleration on large language models.

5.1 Experimental Settings

Models and Implementation. We evaluate on Llama-2 7B, Llama-2 70B, Qwen-2.5-7B, and Qwen-3-30B-A3B using Hugging Face Transformers (Wolf, 2019) and PyTorch (Paszke et al., 2019). Qwen-2.5 72B is additionally used in Table 9 for the industrial speedup evaluation.

Calibration. We use 8 examples (2048 tokens each) from WikiText-2 (Merity et al., 2016) for activation profiling, with $K_a = 10$ for top-K activation selection.

Fine-tuning. We apply LoRA (Hu et al., 2021) (rank 8, alpha 32) on 2,048 WikiText-2 samples for 1 epoch using Adam (Kingma, 2014) ($\beta_1 = 0.9$, $\beta_2 = 0.95$), with learning rates 0.001 for router scaling and 5.95e-5 for LoRA. Load balancing uses $\gamma = 0.001$.

Baselines. We compare against: (1) *Structured Pruning*: SliceGPT (Ashkboos et al., 2024) and SLEB (Song et al., 2024) at 20% reduction; (2) *MoE Restructuring*: LLaMA-MoE (Zhu et al., 2024), LLaMA-MoE-v2 (Qu et al., 2024), and EMoE (Qiu et al., 2023). All baselines are re-implemented by us and fine-tuned with LoRA under matched data budget (2k samples) for fair com-

Table 1: Zero-shot accuracy (%) at 25% sparsity (Sp.). All sparsified methods are fine-tuned with 2k samples; the dense baseline is not fine-tuned.

Method	Sp.	PIQA	WinoG.	ARC-E	ARC-C	HellaS.
Llama-2 7B						
Dense	0%	78.78	69.06	74.58	46.16	76.00
SliceGPT	20%	65.71	62.88	59.76	33.21	51.34
SLEB	20%	73.13	58.98	57.90	33.02	62.47
LLaMA-MoE	25%	49.35	50.28	54.04	26.37	25.77
LLaMA-MoE-v2	25%	63.55	59.35	63.77	34.81	54.89
EMoE	25%	72.47	64.48	58.63	35.75	60.80
Ours	25%	**74.34**	**65.77**	**67.09**	**40.35**	**69.36**
Llama-2 70B						
Dense	0%	82.70	77.98	80.98	57.34	83.84
SliceGPT	20%	68.91	70.06	64.56	41.14	56.26
SLEB	20%	77.39	65.55	62.37	40.11	68.39
LLaMA-MoE	25%	51.95	56.50	59.09	32.40	27.57
LLaMA-MoE-v2	25%	66.79	66.57	68.94	42.38	59.57
EMoE	25%	76.34	72.33	63.47	43.62	66.19
Ours	25%	**78.49**	**73.49**	**73.32**	**49.86**	**76.12**
Qwen-2.5-7B						
Dense	0%	79.82	73.16	77.36	51.02	78.86
SliceGPT	20%	66.19	66.51	61.88	36.69	53.21
SLEB	20%	74.95	61.76	59.95	35.80	64.41
LLaMA-MoE	25%	49.63	53.21	57.05	28.64	25.65
LLaMA-MoE-v2	25%	64.25	62.71	65.77	37.59	56.06
EMoE	25%	73.98	65.41	60.63	38.48	62.71
Ours	25%	**75.93**	**69.36**	**70.59**	**43.86**	**72.21**
Qwen-3-30B-A3B						
Dense	0%	84.51	79.18	84.43	57.88	87.44
SliceGPT	20%	70.60	71.58	66.88	41.85	58.41
SLEB	20%	79.16	66.01	70.08	42.11	71.74
LLaMA-MoE	25%	52.18	54.48	62.50	30.77	28.32
LLaMA-MoE-v2	25%	65.54	67.24	71.27	41.99	62.78
EMoE	25%	74.76	70.50	65.78	43.12	70.62
Ours	25%	**80.23**	**74.84**	**76.75**	**48.80**	**80.71**

parison. Structured pruning baselines use 20% reduction because they reduce both FFN and attention parameters, making their effective FFN sparsity comparable to our 25% FFN-only sparsity.

Configuration. Unless otherwise noted, we use 25% sparsity (i.e., 75% of FFN neurons are activated per token) with S3A3E8 (3 shared + 3 active routed / 8 total experts). All MoE methods use 8 experts for fair comparison.

5.2 Main Results

Zero-Shot Downstream Tasks. Table 1 presents results on five benchmarks: PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2021), ARC-Easy, ARC-Challenge (Clark et al., 2018), and HellaSwag (Zellers et al., 2019). At 25% sparsity, our method consistently outperforms all baselines across four models spanning 7B to 30B parameters. On Llama-2 7B, it achieves 74.34% PIQA and 69.36% HellaSwag, exceeding both structured pruning and MoE restructuring approaches. The improvements scale consistently: on Llama-2 70B we observe similar gains, and on the Qwen-3-30B-A3B, our method achieves 80.23% PIQA and 80.71% HellaSwag.

Table 2: Broader evaluation on Llama-2 7B at 25% sparsity (S3A3E8).

Method	MMLU (%)	HumanEval (%)	GSM8K (%)
LLaMA-MoE	35.09	7.58	7.41
LLaMA-MoE-v2	38.02	9.32	10.09
EMoE	43.11	10.29	12.55
Ours	**44.02**	**11.22**	**13.01**

Table 3: Training-free vs fine-tuned on Llama-2 7B (25% sparsity).

Method	Regime	MMLU (%)	PPL Wiki	PPL C4
LLaMA-MoE-v2	Training-free	30.33	>10k	>7k
LLaMA-MoE-v2	Fine-tuning	38.02	8.68	19.76
Ours	Training-free	42.50	7.32	11.98
Ours	Fine-tuning (2k)	**44.02**	**5.92**	**11.21**

Broader Evaluation on Knowledge, Coding, and Math. Beyond these five zero-shot tasks, we also evaluate Llama-2 7B at 25% sparsity (S3A3E8) on MMLU-5shot, HumanEval pass@1, and GSM8K-8shot to cover knowledge-intensive and reasoning benchmarks. As summarized in Table 2, our analytical MoE restructuring achieves 44.02% MMLU-5shot and competitive coding/math accuracy, outperforming LLaMA-MoE variants and EMoE across all three benchmarks.

5.3 Ablation Studies

Training-Free vs Fine-Tuned Performance. Figure 4 shows data efficiency at 25% sparsity. The method achieves reasonable performance immediately after construction with zero fine-tuning, indicating that the analytical router initialization is effective. Performance plateaus quickly with additional data: as few as 1,024 samples reach near-optimal results.

Table 3 compares with LLaMA-MoE-v2 on Llama-2 7B. Our training-free model achieves 42.50% MMLU-5shot (vs 38.02% for LLaMA-MoE-v2 after fine-tuning). With 2k samples, we reach 44.02%, indicating most gains come from analytical restructuring rather than fine-tuning.

Calibration Sensitivity. Table 4 varies calibration source (WikiText-2 and C4) and size on Llama-2 7B. For WikiText-2, increasing from 8 to 64 sam-

Table 4: Calibration sensitivity on Llama-2 7B (25% sparsity). Perplexity is consistent across calibration sources and sizes.

Source	n	MMLU (%)	PPL Wiki	PPL C4
WikiText-2	8	44.02	5.92	11.21
WikiText-2	32	44.63	5.72	11.15
WikiText-2	64	**44.89**	**5.69**	**10.98**
C4	8	42.31	7.04	9.17
C4	32	43.25	6.92	9.07
C4	64	**43.39**	**6.78**	**9.02**

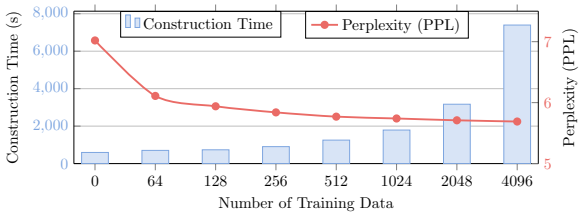

Figure 4: Data efficiency: performance and construction time vs fine-tuning samples (25% sparsity).

ples yields modest MMLU gains (44.02→44.89) and small perplexity reductions. Perplexity is consistent across both calibration sources, confirming the diverse activation pattern (Section 3.2) is intrinsic to pre-trained FFNs rather than data-specific. The low sample requirement (8 examples) makes domain-specific calibration practical when needed.

To further verify domain invariance, we profile shared-expert neurons separately on math and science data (Nemotron-Post-Training-Dataset-v1 (Nathawani et al., 2025)) and code data (OpenCoder (Huang et al., 2024)), then measure pairwise overlap. The overlap is 84% (math/science), 86% (math/code), and 80% (science/code), confirming that shared-expert selection reflects intrinsic model structure rather than domain-specific artifacts.

Clustering and Routing. To isolate the contributions of clustering and routing, Table 5 compares methods under identical settings (25% sparsity, 2k fine-tuning). MoEfication (Zhang et al., 2021), G-MoEfication (Lee et al., 2024) (its non-ReLU generalization), and Read-ME (Cai et al., 2024) achieve only 35.17%, 36.37%, and 31.24% MMLU respectively, while our method reaches 44.02%. Replacing each method's router with our analytical router improves results by +2 to 6 pp, and further switching to our activation-based clustering with shared experts yields an additional +5 to 7 pp, confirming both components contribute independently.

Weight-based methods assume neurons with similar parameters serve similar functions, but ignore input-dependent behavior. Our approach groups neurons that co-activate, aligning expert structure with actual usage patterns. Notably, MoEfication also explored co-activation graph clustering, but their parameter-based method performed better. Our approach differs: (1) we explicitly separate high-frequency neurons into shared experts, and (2) we derive the router analytically from representative neurons. This combination yields nearly +9 pp over MoEfication's best method.

Table 5: Ablation: clustering and routing on Llama-2 7B (MMLU). All use 25% sparsity and 2k fine-tuning. G-MoEfication (Lee et al., 2024) extends MoEfication to non-ReLU models.

Method	Expert grouping	Router	MMLU (%)
MoEfication	Param. K-means	MLP	35.17
G-MoEfication	Param. K-means	MLP	36.37
READ-ME	Domain-aware	Global	31.24
MoEfication + ours	Param. K-means	Analytical	37.33
G-MoEfication + ours	Param. K-means	Analytical	39.21
READ-ME + ours	Domain-aware	Analytical	36.79
Ours	Activation + shared	Analytical	**44.02**

Table 6: Token budget and conversion time comparison on Llama-2 7B. E2E: end-to-end time including fine-tuning; Construct: restructuring only.

Method	Token budget	E2E time	Construct time
Ours	**4M**	**46min**	**4.5min**
LLaMA-MoE-v1	200B	Weeks	6min†
LLaMA-MoE-v2	7B	Days	8min†

† Split-only; training time not included.

Table 7: Efficiency: FLOPs, MACs, and throughput.

Model	Method	FLOPs (T/G)	MACs (G)	Thru. (tok/s)
Llama-2 7B	Dense	1.69T	845.7	45.9
	Ours (25%)	1.41T -16.6%	707.4 -16.3%	52.7 +14.8%
Qwen3-30B-A3B	Dense	778.7G	389.3	1.19
	Ours (Hier.)	634.9G -18.5%	331.3 -14.9%	1.36 +14.3%

5.4 Analysis and Discussion

Efficiency Comparison. Table 6 compares token budgets and conversion times. Note that LLaMA-MoE performs continual pre-training while ours focuses on post-training restructuring; nonetheless, practitioners choosing between approaches may consider total cost. LLaMA-MoE-v1/v2 require 200B/7B tokens (weeks/days); our method uses only 4M tokens (2k samples × 2048 tokens) and completes in 46 minutes end-to-end (4.5 minutes

Table 8: Orthogonality of neuron-level activation sparsity (WINA) and our expert-level Dense-to-MoE restructuring on Llama-2 7B at 25% sparsity. WINA operates at the finer neuron level, while our method restructures FFNs into routed experts; combining them yields additive efficiency gains.

Method	TFLOPs (↓)	GMACs (↓)	tokens/s (↑)
Dense (baseline)	1.69	845.71	45.88
WINA (25% sparsity)	1.31 (-22.5%)	691.19 (-18.3%)	51.76 (+12.8%)
Ours (25% sparsity)	1.41 (-16.6%)	707.36 (-16.3%)	52.67 (+14.8%)
Ours + WINA	**1.23** (-27.2%)	**625.53** (-26.0%)	**55.97** (+22.0%)

Table 9: Inference speedup (25% sparsity) on Qwen-2.5 72B at different context lengths and scenarios. SxAyEz: x shared + y active routed / z total experts.

Config	Mem-Bound		Compute-Bound	
	4k ctx	32k ctx	4k ctx	32k ctx
S1A5E8	1.08×	1.15×	1.12×	**1.17**×
S3A3E8	1.06×	1.13×	1.11×	1.15×
S2A4E8	1.05×	1.12×	1.10×	1.12×
S4A8E16	1.02×	1.10×	1.08×	1.11×
S6A6E16	1.03×	1.08×	1.07×	1.10×
S3A9E16	1.02×	1.05×	1.05×	1.09×

for analytical construction).

Table 7 reports FLOPs and throughput. On Llama-2 7B, 25% sparsity reduces FLOPs by 16.6% and increases throughput by 14.8%. Hierarchical application to Qwen3-30B-A3B yields 18.5% FLOPs savings and 14.3% throughput gains.

Load Balancing. Figure 5 shows expert utilization. Without load balancing, the final layer exhibits activation skew. Our adaptive bias mechanism redistributes load uniformly, which helps achieve the full speedup potential in deployment.

Orthogonality with Activation Sparsity. Table 8 shows that our expert-level restructuring is orthogonal to neuron-level activation sparsity (WINA (Chen et al., 2025)). Combining both yields 27.2% TFLOPs reduction and 22.0% throughput gain on Llama-2 7B, confirming they target different inefficiencies.

Industrial Speedup. Table 9 shows inference speedups on Qwen-2.5 72B across memory- and compute-bound (Batch Size > 400) scenarios. In compute-bound settings with 32k context, S1A5E8 configuration achieves up to 1.17× speedup.

Perplexity-Sparsity Trade-off. Table 10 studies WikiText-2 perplexity on Llama-2 7B as we vary sparsity with 16 total experts. Interestingly, at 0.125 sparsity (87.5% neurons active), our converted model slightly outperforms the dense baseline (5.25 vs 5.27), suggesting the restructuring may provide implicit regularization.

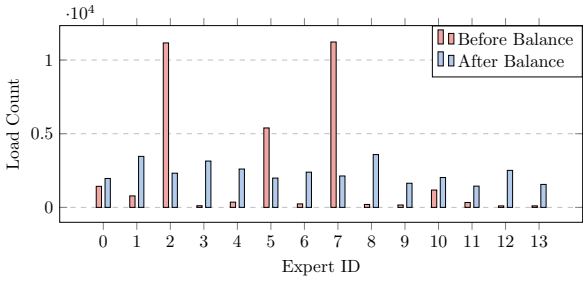

Figure 5: Load balancing: expert utilization before (left) and after (right).

Table 10: Perplexity vs sparsity on Llama-2 7B (16 experts). Lower is better.

Sparsity	Dense	0.75	0.625	0.5	0.375	0.25	0.125
PPL ↓	5.27	12.73	9.56	7.71	6.55	5.78	**5.25**

Table 11: Effect of k-sample self-consistency at 25% sparsity. Accuracy (%).

Method	k	PIQA	ARC-E	ARC-C	Avg
Llama-2 7B					
Dense	1	78.78	74.58	46.16	66.51
Dense	5	79.21	75.29	46.75	67.08
Ours	1	74.34	67.09	40.35	60.59
Ours	5	**77.52**	**73.88**	**44.54**	**65.31**
Qwen3-30B-A3B					
Dense	1	84.51	84.43	57.88	75.61
Dense	5	85.11	85.33	58.12	76.19
Ours	1	80.23	76.75	48.80	68.59
Ours	5	**84.56**	**84.75**	**57.19**	**75.50**

Expert Configuration Impact. Figure 6 compares different expert configurations at 25% sparsity. S6A6E16 achieves highest performance on PIQA and ARC-Easy, while S3A9E16 performs best on WinoGrande, indicating that optimal configuration depends on task characteristics.

Self-Consistency Benefits. Table 11 evaluates k-sample self-consistency (voting) at 25% sparsity. Interestingly, sparse models benefit more from self-consistency than dense ones: on Llama-2 7B, increasing k from 1 to 5 yields +4.72 pp for our method vs only +0.57 pp for dense. On Qwen3-30B-A3B, the gap nearly closes entirely (+6.91 pp vs +0.58 pp), suggesting that variance from sparse routing can be effectively averaged out at deployment time.

6 Further Discussion

Broader Impact and Future Directions. Our work presents an analytical post-training frame-

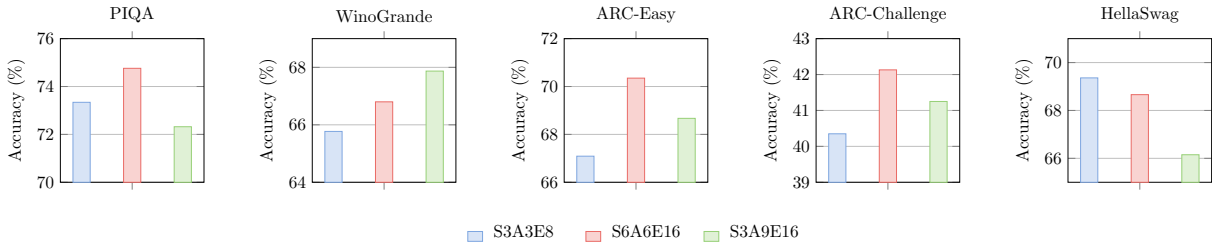

Figure 6: Impact of expert configuration at 25% sparsity. SxAyEz denotes x shared experts + y active routed experts out of z total experts.

work for reducing the significant computational overhead of LLM inference, thereby making powerful models more accessible for research and deployment in resource-constrained settings. Beyond a pure acceleration technique, the analytical nature of the method offers a new lens for interpreting the internal workings of FFNs. The distinct grouping of neurons into 'shared' and 'routed' experts based on activation statistics provides empirical evidence for functional specialization within these layers. Future research could leverage this methodology to analyze how knowledge is encoded and processed within LLMs. For future work, extending this analytical restructuring approach to other parts of the transformer, such as attention heads, is a promising direction. Additionally, exploring more sophisticated analytical techniques for router construction could potentially close the remaining gap with fully trained routers, without sacrificing the efficiency of the post hoc approach.

Compatibility with Other Efficiency Techniques. The analytical restructuring is orthogonal to most system- and model-level efficiency methods and can be composed with them. In practice, FFN restructuring integrates well with post-training quantization (e.g., AWQ/QAT) because the operation preserves layer interfaces; it can be applied either before or after quantization with a small calibration pass to maintain accuracy. Similarly, attention-side optimizations (KV-cache compression, speculative decoding, and attention sparsity) target different bottlenecks and are complementary. Structured pruning (e.g., SliceGPT, SLEB) and our dynamic expert routing address different regimes: pruning induces static capacity reduction across all inputs, while our method activates capacity conditionally per token. Similarly, training-free activation sparsity methods (e.g., TEAL, WINA) operate at the finer neuron level and can be applied within our routed experts to further reduce FLOPs. In deployment, load-balancing and batching policies remain important to realize end-to-end speedups; our built-in bias adaptation mitigates expert hotspotting and improves utilization on both memory-bound and compute-bound settings. Overall, after a lightweight calibration and restructuring step, the framework serves as an FFN replacement that composes with quantization, caching, pruning, and serving optimizations to widen the practical acceleration envelope.

7 Conclusion

We introduced a post-training framework that analytically restructures FFNs into sparse MoE architectures using only a small calibration dataset and minutes of computation (4.5 minutes on Llama-2 7B). By leveraging the diverse distribution of neuron activation frequencies, the method partitions neurons into shared and routed experts and constructs a router from representative neuron statistics, enabling strong performance with optional lightweight fine-tuning. It applies to both dense and existing MoE models for hierarchical sparsity, and is orthogonal to techniques such as quantization and neuron-level activation sparsity, offering a practical, training-light path to deploying performant sparse LLMs.

8 Limitations

Our framework has three main limitations. First, activation profiling quality depends on the calibration dataset; performance is best when the data is representative of the target domain, though the method is relatively robust to calibration set size. Second, the discrete nature of sparse routing introduces higher generation variance, which can be mitigated via self-consistency. Third, our evaluation focuses on English benchmarks and decoder-only transformers (Llama-2, Qwen); extension to multilingual and encoder-decoder or state-space architectures is left to future work.

References

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*.

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

Ruisi Cai, Yeonju Ro, Geon-Woo Kim, Peihao Wang, Babak Ehteshami Bejnordi, Aditya Akella, Zhangyang Wang, and 1 others. 2024. Read-me: Refactorizing llms as router-decoupled mixture of experts with system co-design. *Advances in Neural Information Processing Systems*, 37:116126–116148.

Sihan Chen, Dan Zhao, Jongwoo Ko, Colby Banbury, Huiping Zhuang, Luming Liang, and Tianyi Chen. 2025. Wina: Weight informed neuron activation for accelerating large language model inference. *arXiv preprint arXiv:2505.19427*.

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

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

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, and 1 others. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Siming Huang, Tianhao Cheng, Jason Klein Liu, Jiaran Hao, Liuyihan Song, Yang Xu, J. Yang, J. H. Liu, Chenchen Zhang, Linzheng Chai, Ruifeng Yuan, Zhaoxiang Zhang, Jie Fu, Qian Liu, Ge Zhang, Zili Wang, Yuan Qi, Yinghui Xu, and Wei Chu. 2024. Opencoder: The open cookbook for top-tier code large language models.

Roy Jonker and Ton Volgenant. 1988. A shortest augmenting path algorithm for dense and sparse linear assignment problems. In *DGOR/NSOR: Papers of the 16th Annual Meeting of DGOR in Cooperation with NSOR/Vorträge der 16. Jahrestagung der DGOR zusammen mit der NSOR*, pages 622–622. Springer.

Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. 2022. Sparse upcycling: Training mixture-of-experts from dense checkpoints. *arXiv preprint arXiv:2212.05055*.

Jaeseong Lee, Seung-won Hwang, Wonpyo Park, and Mingi Ji. 2024. Breaking relu barrier: Generalized moefication for dense pretrained models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10097–10107.

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. *arXiv preprint arXiv:2006.16668*.

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

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024b. Visual instruction tuning. *Advances in neural information processing systems*, 36.

James Liu, Pragaash Ponnusamy, Tianle Cai, Han Guo, Yoon Kim, and Ben Athiwaratkun. 2024c. Training-free activation sparsity in large language models. *arXiv preprint arXiv:2408.14690*.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, and 1 others. 2023. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Dhruv Nathawani, Igor Gitman, Somshubra Majumdar, Evelina Bakhturina, Ameya Sunil Mahabaleshwarkar, , Jian Zhang, and Jane Polak Scowcroft. 2025. Nemotron-Post-Training-Dataset-v1.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library.

Advances in neural information processing systems, 32.

Zehua Pei, Hui-Ling Zhen, Xianzhi Yu, Sinno Jialin Pan, Mingxuan Yuan, and Bei Yu. 2024. Fusegpt: Learnable layers fusion of generative pre-trained transformers. *arXiv preprint arXiv:2411.14507*.

Zihan Qiu, Zeyu Huang, and Jie Fu. 2023. Unlocking emergent modularity in large language models. *arXiv preprint arXiv:2310.10908*.

Xiaoye Qu, Daize Dong, Xuyang Hu, Tong Zhu, Weigao Sun, and Yu Cheng. 2024. Llama-moe v2: Exploring sparsity of llama from perspective of mixture-of-experts with post-training. *arXiv preprint arXiv:2411.15708*.

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

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. 2024. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. *arXiv preprint arXiv:2402.09025*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ziteng Wang, Jun Zhu, and Jianfei Chen. 2024. Remoe: Fully differentiable mixture-of-experts with relu routing. *arXiv preprint arXiv:2412.14711*.

T Wolf. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Haoyuan Wu, Haisheng Zheng, Zhuolun He, and Bei Yu. 2024. Parameter-efficient sparsity crafting from dense to mixture-of-experts for instruction tuning on general tasks. *arXiv preprint arXiv:2401.02731*.

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

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

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, and 1 others. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2021. Moefication: Transformer feed-forward layers are mixtures of experts. *arXiv preprint arXiv:2110.01786*.

Haizhong Zheng, Xiaoyan Bai, Xueshen Liu, Z Morley Mao, Beidi Chen, Fan Lai, and Atul Prakash. 2024. Learn to be efficient: Build structured sparsity in large language models. *arXiv preprint arXiv:2402.06126*.

Zexuan Zhong, Mengzhou Xia, Danqi Chen, and Mike Lewis. 2024. Lory: Fully differentiable mixture-of-experts for autoregressive language model pre-training. *arXiv preprint arXiv:2405.03133*.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. 2024. Llama-moe: Building mixture-of-experts from llama with continual pre-training. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15913–15923.

Simiao Zuo, Qingru Zhang, Chen Liang, Pengcheng He, Tuo Zhao, and Weizhu Chen. 2022. Moebert: from bert to mixture-of-experts via importance-guided adaptation. *arXiv preprint arXiv:2204.07675*.

A Detailed Mathematical Derivations

This appendix provides the detailed mathematical derivations and algorithmic analysis that support the core concepts presented in the main manuscript.

A.1 Sparsity Hypothesis

Building on the activation sparsity observation in Section 3.1, we formalize the hypothesis that justifies our router construction. Given input $\mathbf{x} \in \mathbb{R}^d$, each neuron's contribution to the FFN output can be analyzed independently. The FFN output decomposes as:

$$F(\mathbf{x}) = \sum_{i=1}^{d_h} h_i \mathbf{w}_{down,i} \tag{11}$$

where h_i is the hidden activation and $\mathbf{w}_{down,i}$ is the corresponding output projection column.

Each h_i acts as a gating score for $\mathbf{w}_{down,i}$. Since structured pruning research shows that $\|F(\mathbf{x})\|$ is typically small due to residual connections, FFN activations exhibit high sparsity. This leads to our central hypothesis:

$$\arg\min_i |h_i \mathbf{w}_{down,i}| \approx \arg\min_i |h_i| \tag{12}$$

This approximation is justified because when h_i is extremely small, the product $h_i \mathbf{w}_{down,i}$ vanishes regardless of the magnitude of $\mathbf{w}_{down,i}$.

A.2 Complete Activation Analysis Pipeline

We provide the complete mathematical pipeline for neuron activation profiling.

Step 1: Hidden State Computation. Given input $\mathbf{X} \in \mathbb{R}^{b\times s\times d}$, we reshape to $\mathbf{X}' \in \mathbb{R}^{q\times d}$ where $q = b \cdot s$. The hidden states are:

$$\mathbf{H} = \text{Swish}(\mathbf{X}'\mathbf{W}_{gate}) \odot (\mathbf{X}'\mathbf{W}_{up}) \tag{13}$$

where $\mathbf{H} \in \mathbb{R}^{q\times d_h}$.

Step 2: Activation Matrix Construction (ATopK). For each token, we identify the top-K_a neurons by activation magnitude, i.e., an absolute top-K (ATopK) selection over $|h_i|$:

$$a_i = \begin{cases} 1, & |h_i| \in \text{TopK}(\{|h_j| : 1 \le j \le d_h\}, K_a), \\ 0, & \text{otherwise}, \end{cases} \tag{14}$$

yielding a binary activation matrix $\mathbf{A} \in \{0,1\}^{q\times d_h} = [\mathbf{c}_1\ \mathbf{c}_2\ \cdots\ \mathbf{c}_{d_h}]$, where each column $\mathbf{c}_i \in \{0,1\}^q$ is the activation feature vector for neuron i.

Step 3: Activation Rate Computation.

$$\mu_i = \frac{1}{q}\sum_{t=1}^{q} \mathbf{A}[t,i], \quad \boldsymbol{\mu} = [\mu_1, \cdots, \mu_{d_h}] \tag{15}$$

Step 4: Shared Expert Selection. We select the $N_s \cdot m$ neurons with highest μ_i:

$$S_{N_s} = \{i : \mu_i \in \text{TopK}(\boldsymbol{\mu}, N_s \cdot m)\} \tag{16}$$

A.3 Balanced Clustering Algorithm for Routed Experts

We employ a constrained balanced K-means algorithm on the activation feature vectors $\mathbf{c}_i$.

Centroid Initialization. We select N_r centroids from remaining neurons with highest activation rates:

$$C = \{\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_{N_r}\} \tag{17}$$

Distance Matrix. We construct $\mathbf{D} \in \mathbb{R}^{N_r\cdot m\times N_r}$ where:

$$d_{i,j} = \|\mathbf{c}_i - \hat{\mathbf{c}}_j\|_2 = \sqrt{\sum_{k=1}^{q}(c_{k,i} - \hat{c}_{k,j})^2} \tag{18}$$

Equivalence of L_2 and Hamming Distance. Since $\mathbf{c}_i \in \{0,1\}^q$ are binary, L_2 distance relates directly to Hamming distance:

$$\|\mathbf{c}_i - \mathbf{c}_j\|_2^2 = \sum_{k=1}^{q}(c_{k,i} - c_{k,j})^2 = d_H(\mathbf{c}_i, \mathbf{c}_j) \tag{19}$$

where d_H counts positions where vectors differ. Minimizing L_2 is equivalent to minimizing Hamming distance.

Balanced Assignment. The algorithm iteratively solves:

$$\min_T \sum_{i,j} T_{i,j} \cdot d_{i,j} \tag{20}$$
$$\text{s.t.} \sum_i T_{i,j} = m,\ \sum_j T_{i,j} = 1,\ T_{i,j} \ge 0$$

Cluster Update:

$$\hat{\mathbf{c}}_j^{t+1} = \begin{cases} \frac{\sum_i T_{i,j}^t \cdot \mathbf{c}_i}{\sum_i T_{i,j}^t}, & \text{if } \sum_i T_{i,j}^t > 0, \\ \hat{\mathbf{c}}_j^t, & \text{otherwise.} \end{cases} \tag{21}$$

Since this is an unbalanced assignment (mN_r neurons to N_r clusters of size m), we extend $\mathbf{D}$ by repeating each column m times, then solve the balanced assignment using the Jonker-Volgenant algorithm (Jonker and Volgenant, 1988) with complexity $O(n^3)$.

A.4 Router Construction Derivation

This section provides the complete derivation for the analytical router.

Problem. Given input $\mathbf{x}$, the reconstruction error is:

$$\begin{aligned} &\arg\min_{G} |F_{MoE}(\mathbf{x}) - F(\mathbf{x})| \\ &= \arg\min_{G} |\sum_{i \in S_{de}} E_i^r(\mathbf{x})| \end{aligned} \tag{22}$$

where S_{de} is the set of deactivated experts.

Reduction. Using the sparsity hypothesis (Equation (12)), the reconstruction error can be bounded by the sum of hidden-state magnitudes of deactivated experts:

$$\begin{aligned} &\arg\min_{G} \left| \sum_{i \in S_{de}} E_i^r(\mathbf{x}) \right| \\ &\approx \arg\min_{G} \mathbb{E}_{\mathbf{x}} \left[\sum_{i \in S_{de}} \|\mathbf{h}_i^r\|_1 \right]. \end{aligned} \tag{23}$$

Optimal Solution. The optimal router matches sorting of expert scores $\{s_i\}$ with expected activations $\bar{\mathbf{h}}_i^r = \mathbb{E}[\|\mathbf{h}_i^r\|_1]$:

$$s_{\sigma(1)} \leq \cdots \leq s_{\sigma(N_r)}, \quad \bar{\mathbf{h}}_{\sigma(1)}^r \leq \cdots \leq \bar{\mathbf{h}}_{\sigma(N_r)}^r \tag{24}$$

Representative Neuron. For each expert j, the representative neuron is:

$$R_j = \underset{i \in \text{cluster } j}{\operatorname{argmin}} \|\mathbf{c}_i - \hat{\mathbf{c}}_j\|_2. \tag{25}$$

The router uses these neurons:

$$G(\mathbf{x}) = \text{Swish}(\mathbf{W}_{\text{gate}}^{R\top}\mathbf{x}) \odot (\mathbf{W}_{\text{up}}^{R\top}\mathbf{x}) \tag{26}$$

yielding scores that approximate expert contributions.