ACTIVATION FACTORIZATION FOR DENSE-TO-MOE CONVERSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Converting pretrained dense language models into Mixture-of-Experts (MoE) architectures can enable structured sparse execution without pretraining a new model from scratch. The quality of the converted model, however, depends on how neuron activation structure is represented for expert construction and routing. We propose a dense-to-MoE conversion framework based on non-negative matrix factorization (NMF) of FFN activation magnitudes. The factorization provides complementary neuron-side factor loadings and token-side factor coefficients. We use the neuron-side representation to construct balanced physical experts, while a lightweight predictor estimates token coefficients and maps them to expert-level routing scores through the same factorized activation structure. The original dense-model parameters remain frozen, with only lightweight routing modules adapted after conversion. On LLaMA-3-8B with 25% active FFN capacity, our method achieves an average accuracy of 57.93% across five downstream benchmarks, exceeding the average accuracy of all evaluated CMoE configurations. Controlled ablations show that both NMF-based expert construction and NMF-guided routing contribute to the final performance. On a single NVIDIA RTX 4090, the resulting physical expert execution achieves 1.13–$1.22\times$ end-to-end prefill speedup across sequence lengths from 512 to 2048. These results show that activation factorization provides an effective representation for low-cost dense-to-MoE conversion with structured sparse execution.

1 INTRODUCTION

Large language models (LLMs) rely heavily on feed-forward networks (FFNs), which account for a substantial fraction of Transformer inference computation (Kaplan et al., 2020; Fedus et al., 2022). Mixture-of-Experts (MoE) architectures reduce this cost through conditional computation, activating only a subset of experts for each input token while retaining a large overall model capacity (Lepikhin et al., 2020; Fedus et al., 2022; Shazeer et al., 2017). However, many widely used pretrained models are only released as dense checkpoints (Komatsuzaki et al., 2022). Replacing such models with newly pretrained MoEs can require substantial additional training cost. This motivates dense-to-MoE conversion, where the FFNs of an existing dense model are reorganized into sparsely activated experts while preserving the original FFN parameters.

Dense-to-MoE conversion introduces a structural constraint that is absent from the original dense model. In a dense FFN, every token can access all intermediate neurons. After conversion, these neurons are partitioned into physical experts, and each token can access only a subset of them. The quality of the converted model therefore depends on two closely related components: expert construction, which determines how neurons are assigned to experts, and expert routing, which determines which experts are activated for each token. These two components are structurally coupled because routing operates over the expert partition produced during construction. Once the neuron partition is fixed, the router is restricted to selecting among the resulting expert groups; an unsuitable partition may therefore exclude useful neurons even when the routing decision itself is reasonable.

Recent low-cost dense-to-MoE methods increasingly rely on activation statistics collected from a small calibration set. In this setting, how these activations are represented becomes important for both expert construction and routing: the representation determines how neuron behavior is summarized when forming experts and what token-dependent signal is available for estimating expert

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.

relevance. For example, CMoE (Pei et al., 2026) groups neurons using activation patterns and derives routing scores from representative neurons of the resulting experts. This motivates a broader question: what activation representation can capture the distributed structure of FFN neuron behavior for expert construction while also providing informative token-dependent signals for expert routing?

We address this question through activation factorization. Specifically, we apply non-negative matrix factorization (NMF) (Lee & Seung, 1999; 2000) to FFN activation magnitudes collected from a calibration dataset, decomposing them into token-side factor coefficients and neuron-side factor loadings. The neuron-side representation is used to construct balanced physical experts, while the token-side coefficients are predicted from the FFN input and mapped back to neuron activity scores, which are then aggregated to obtain expert-level routing scores. The factorization therefore exposes complementary neuron-side and token-side views of the same activation structure, providing a richer description of distributed neuron behavior for both expert construction and token-dependent routing. The resulting model executes only a subset of physical experts for each token while keeping the original FFN parameters frozen.

Experiments on LLaMA-3-8B show that our method preserves competitive model quality while activating only 25% of the FFN neurons for each token. Across five downstream benchmarks, it achieves an average accuracy of 57.93%, exceeding the average accuracy of all evaluated CMoE configurations. Compared with the LoRA-adapted CMoE variants, our method uses substantially fewer post-conversion token exposures. Controlled ablations further show that both components are important: replacing NMF-based expert construction with co-activation clustering reduces average accuracy by 5.63 percentage points, while replacing NMF-guided routing with representative-neuron routing reduces it by 3.83 points. The resulting physical expert structure also provides measurable inference benefits, achieving 1.13–$1.22\times$ end-to-end prefill speedup on a single NVIDIA RTX 4090 across sequence lengths from 512 to 2048.

Our main contributions are as follows:

- **NMF-guided expert construction.** We factorize FFN activation magnitudes and use neuron-side factor-loading profiles to construct balanced, non-overlapping physical experts from a pretrained dense FFN.
- **NMF-guided expert routing.** We use token-side factor coefficients to estimate neuron activity and aggregate it into expert-level routing scores. This allows expert selection to reflect the predicted activity across all neurons within each expert.
- **Low-cost structured sparse execution.** Our conversion keeps the original model parameters frozen and activates only 25% of the FFN neurons per token. With lightweight router alignment, the resulting model achieves competitive downstream accuracy and provides 1.13–$1.22\times$ end-to-end prefill speedup on a single NVIDIA RTX 4090.

2 Related Work

2.1 Dense-to-MoE Conversion

Dense-to-MoE conversion restructures the FFNs of pretrained dense models into sparsely activated experts while reusing the original model parameters. Early approaches such as MoEfication (Zhang et al., 2022) construct experts from parameter similarity or neuron co-activation patterns and train a router for expert selection. MoEBERT (Zuo et al., 2022) instead uses neuron importance to guide expert construction together with task-specific adaptation. More recent methods extend this paradigm to decoder-only LLMs. LLaMA-MoE (Zhu et al., 2024) and LLaMA-MoE-v2 (Qu et al., 2024) convert pretrained LLaMA FFNs into expert structures and rely on additional post-conversion training to recover model quality.

Recent work has focused on reducing this adaptation cost. CMoE (Pei et al., 2026) analyzes neuron activation patterns, separates frequently active neurons into shared experts, clusters the remaining neurons into balanced routed experts, and derives routing scores from representative neurons of each expert. DOT-MoE (Bamba et al., 2026) formulates neuron assignment as a balanced optimal-transport problem and jointly optimizes neuron-to-expert assignment and token-to-expert routing through a differentiable training framework.

Our work differs in how activation information is represented and reused across the two stages of conversion. We factorize activation magnitudes with NMF and use the resulting neuron-side factor loadings for expert construction and token-side factor coefficients for routing. Thus, expert construction and routing are derived from the same factorized activation representation.

2.2 Sparse FFN Inference and Model Compression

A related line of work exploits activation sparsity within FFNs to reduce inference computation. For example, R-Sparse (Zhang et al., 2025) dynamically selects a subset of FFN neurons for each token based on their predicted importance.

Structured and unstructured pruning methods, including SliceGPT (Ashkboos et al., 2024), Wanda (Sun et al., 2024), and LLM-Pruner (Ma et al., 2023), reduce computation by permanently removing model parameters or structured components. In contrast, our method retains the full FFN parameter set and performs token-dependent structured sparse execution through physical experts.

2.3 Non-negative matrix factorization

Non-negative matrix factorization (NMF) decomposes a non-negative matrix into two lower-rank non-negative factors, providing an additive latent representation of the original data (Lee & Seung, 1999; 2000). Unlike unconstrained matrix factorization, the non-negativity constraint often yields components that can be interpreted through their contributions to individual observations and features.

3 Problem Setup

We consider converting the FFNs of a pretrained dense language model into a structured sparse Mixture-of-Experts (MoE) while preserving the original FFN parameters. The conversion requires two components: partitioning the intermediate FFN neurons into physical experts and routing each token to only a subset of these experts.

3.1 Dense Feed-Forward Networks

Consider a bias-free LLaMA-style SwiGLU FFN. Given an input representation $x \in \mathbb{R}^{d_{\text{model}}}$,

$$h(x) = \text{SiLU}(W_g x) \odot (W_u x), \qquad F(x) = W_d h(x), \tag{1}$$

where $h(x) \in \mathbb{R}^{d_{\text{ff}}}$ is the intermediate FFN activation and each dimension corresponds to one intermediate neuron. For LLaMA-3-8B (Grattafiori et al., 2024), $d_{\text{model}} = 4096$ and $d_{\text{ff}} = 14336$. Dense execution evaluates all d_{ff} neurons for every token.

3.2 Physical Experts and Sparse Routing

We partition the d_{ff} intermediate neurons into E balanced, non-overlapping physical experts $\mathcal{S} = \{S_1, \ldots, S_E\}$, where each expert contains $|S_e| = d_{\text{ff}}/E$ neurons and each neuron belongs to exactly one expert. Each expert consists of the corresponding rows and columns of the original FFN weight matrices, so the conversion reorganizes rather than modifies the pretrained FFN parameters.

For each token, the router produces expert scores $q(x) \in \mathbb{R}^E$ and activates the k highest-scoring experts. Since the experts are balanced, the fraction of FFN neurons executed for each token is $\rho = k/E$.

Dense-to-MoE conversion therefore requires constructing a useful physical expert partition and defining a routing function that selects appropriate experts under a fixed sparse-execution budget.

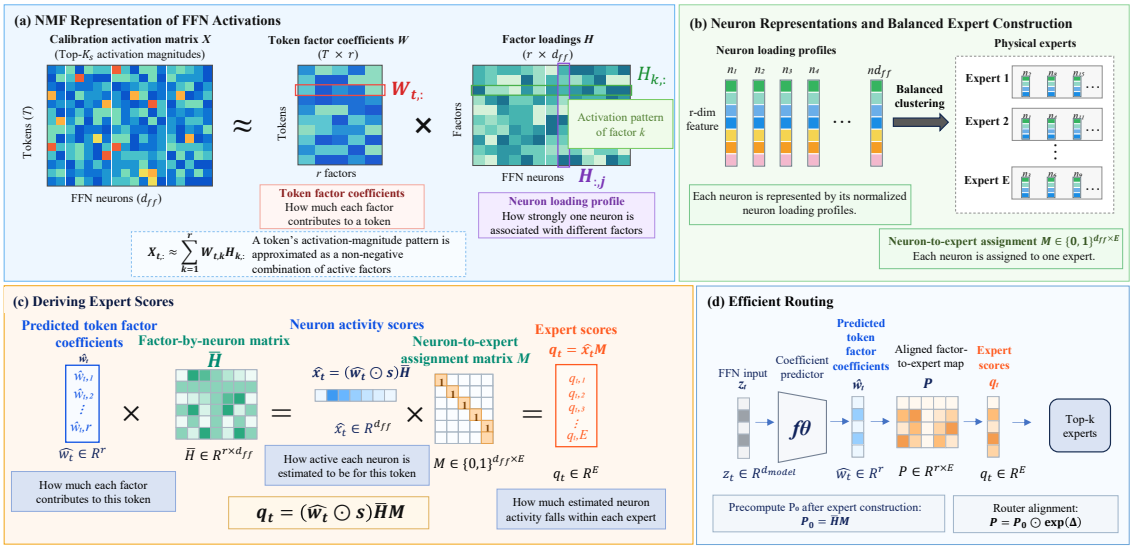

Figure 1: Overview of our dense-to-MoE conversion framework. (a) FFN activation magnitudes are factorized into token-side coefficients and neuron-side factor loadings. (b) Neuron loading profiles are used to construct balanced physical experts. (c) Predicted token factor coefficients are mapped to neuron activity scores and aggregated into expert scores. (d) The factor-to-expert mapping is precomputed and adapted for efficient Top-k expert routing.

4 Methodology

We convert each dense FFN into balanced physical experts using activation statistics from a small calibration set. As illustrated in Figure 1, the method consists of three stages: activation factorization, balanced expert construction, and NMF-guided routing. The factorization provides neuron-side representations for expert construction and token-side coefficients for routing.

4.1 Activation Factorization

For each FFN layer, we collect intermediate SwiGLU activations $h_t \in \mathbb{R}^{d_{\text{ff}}}$ over T calibration tokens. We retain the K_s neurons with the largest activation magnitudes for each token and construct

$$X_{t,j} = |h_{t,j}|\mathbf{1}\left[j \in \text{TopK}(|h_t|, K_s)\right]. \tag{2}$$

We then apply non-negative matrix factorization independently to each layer:

$$X \approx WH, \qquad W \in \mathbb{R}_+^{T \times r}, \quad H \in \mathbb{R}_+^{r \times d_{\text{ff}}}, \tag{3}$$

where r is the NMF rank. For token t,

$$X_{t,:} \approx \sum_{k=1}^{r} W_{t,k} H_{k,:}. \tag{4}$$

Thus, $W_{t,:}$ describes the activation-factor coefficients of token t, while $H_{:,j}$ provides an r-dimensional factor-loading profile for neuron j.

To remove the arbitrary scaling between corresponding factors and coefficients, we normalize

$$s_k = \sum_j H_{k,j}, \qquad \bar{H}_{k,j} = \frac{H_{k,j}}{s_k}, \qquad \widetilde{W}_{t,k} = W_{t,k} s_k, \tag{5}$$

which preserves $WH = \widetilde{W}\bar{H}$. We use $\bar{H}$ in the subsequent construction and routing stages.

4.2 Balanced Expert Construction

We represent neuron j by its normalized factor-loading profile

$$v_j = \frac{\bar{H}_{:,j}}{\|\bar{H}_{:,j}\|_2 + \epsilon}. \tag{6}$$

Balanced clustering of $\{v_j\}_{j=1}^{d_{\text{ff}}}$ produces E physical experts of equal size. We denote the resulting assignment by $M \in \{0,1\}^{d_{\text{ff}} \times E}$, where $M_{j,e} = 1$ if neuron j belongs to expert e. Each neuron is assigned to exactly one expert, yielding a balanced, non-overlapping partition of the original FFN. We provide the details of the clustering method in A.3.

4.3 NMF-Guided Expert Routing

The offline coefficients $W_{t,:}$ are unavailable for unseen tokens, so we train a lightweight predictor

$$\hat{w}_t = f_\theta(z_t) \in \mathbb{R}^r_+, \tag{7}$$

where $z_t \in \mathbb{R}^{d_{\text{model}}}$ is the FFN input.

Deriving expert scores. Given the predicted factor coefficients, we estimate the neuron activity profile as

$$\hat{x}_t = (\hat{w}_t \odot s)\bar{H}. \tag{8}$$

Aggregating these neuron activity scores according to the physical expert assignment gives

$$q_t = \hat{x}_t M = (\hat{w}_t \odot s)\bar{H}M, \tag{9}$$

where $q_t \in \mathbb{R}^E$ contains the expert scores. Thus, each expert score reflects the predicted activity of all neurons assigned to that expert.

Efficient routing and alignment. Since $\bar{H}$ and M are fixed after expert construction, we precompute

$$P_0 = \bar{H}M. \tag{10}$$

Routing can therefore be performed directly as

$$q_t = (\hat{w}_t \odot s)P_0, \tag{11}$$

without explicitly reconstructing the full neuron activity vector.

For lightweight model-level alignment, we introduce

$$P = P_0 \odot \exp(\Delta), \qquad q_t = (\hat{w}_t \odot s)P, \tag{12}$$

with Δ initialized to zero. During alignment, only f_θ and Δ are updated; the expert partition, NMF factors, and original model parameters remain fixed. At inference time, the router computes q_t and activates the Top-k physical experts.

5 EXPERIMENTAL SETUP

5.1 MODEL AND CONVERSION CONFIGURATION

We conduct our experiments on LLaMA-3-8B.

For each FFN layer, we construct $E = 112$ balanced physical experts, with 128 neurons assigned to each expert. Every FFN neuron is assigned to exactly one expert, with no overlap or shared experts. During sparse inference, the router activates $k = 28$ experts for each token, corresponding to an active FFN-neuron fraction of

$$\rho = \frac{k}{E} = \frac{28}{112} = 25\%. \tag{13}$$

For the NMF representation, we retain the $K_s = 2500$ neurons with the largest activation magnitudes for each calibration token. NMF is fitted independently for every FFN layer with rank $r = 128$. Both K_s and r are kept fixed in the main experiments.

The original attention and FFN parameters of LLaMA-3-8B remain frozen throughout conversion and router training. Only the parameters associated with the routing mechanism are optimized.

5.2 CALIBRATION AND ROUTER TRAINING

We use 64 sequences of length 512 sampled from Dolmino-Mix (Olmo et al., 2025), corresponding to 32,768 calibration tokens. Intermediate SwiGLU activations from the frozen dense model are used to fit a separate NMF model for each FFN layer. The resulting factorization provides both the neuron-side representation for expert construction and token-side supervision for initializing the routing mechanism. Each layer uses an independent $4096 \to 256 \to 128$ coefficient predictor with SiLU and Softplus activations. Detailed NMF and router-initialization settings are provided in Appendix A.

After this offline calibration stage, we perform model-level router alignment using the original dense LLaMA-3-8B model as the teacher. Only the coefficient predictors and factor-to-expert mapping adapters are updated; the expert partition, NMF factors, and original model parameters remain fixed. Router alignment uses approximately 0.13M input-token exposures. The objective combines full-vocabulary KL distillation and next-token cross-entropy with router-logit and load-balancing regularization. Additional optimization details are provided in Appendix A.

5.3 BASELINES

We compare our method with baselines from three families, in addition to the original dense model.

Dense model. We report the original LLaMA-3-8B(Grattafiori et al., 2024) model as the dense reference.

Dense-to-MoE conversion. We compare against CMoE (Pei et al., 2026), which constructs MoE experts from neuron activation statistics and derives an analytical routing mechanism, and LLaMA-MoE (Zhu et al., 2024), which converts a pretrained dense LLaMA model into an MoE architecture followed by additional training.

Activation sparsity. We include R-Sparse (Zhang et al., 2025) as a neuron-level activation sparsity baseline. Unlike structured MoE conversion, R-Sparse dynamically selects individual FFN neurons rather than predefined physical experts.

Pruning methods. We compare against SliceGPT (Ashkboos et al., 2024), Wanda (Sun et al., 2024), and LLM-Pruner (Ma et al., 2023). These approaches reduce model computation by removing weights or structured model components rather than performing token-dependent expert selection.

Table 1: Downstream accuracy (%) on LLaMA-3-8B under a 25% FFN activation or remaining-capacity setting. For MoE methods, k/n denotes activating k out of n experts per token. For pruning methods, 25% denotes the remaining FFN capacity after 75% FFN pruning. "LoRA-4M" denotes CMoE with LoRA fine-tuning using approximately 4M token exposures, while "Router-0.13M" denotes model-level router alignment with approximately 0.13M input-token exposures. Higher is better for all downstream tasks.

Method	Regime	Conversion Training	Active FFN	PIQA	ARC-C	SciQ	WinoGrande	BoolQ
Dense	Dense	–	100%	78.40	57.21	96.52	77.35	81.07
Wanda	Unstructured Pruning	No	25%	58.11	19.36	71.41	50.04	61.45
LLM-Pruner	Structured Pruning	No	25%	36.51	17.94	43.58	36.81	29.14
SliceGPT	Structured Pruning	No	25%	50.71	22.01	53.80	50.59	37.83
R-Sparse	Activation Sparsity	No	25%	66.97	26.58	85.49	58.64	59.13
CMoE	Dense-to-MoE	No	4/16 (25%)	50.87	24.23	53.10	53.19	45.08
CMoE	Dense-to-MoE	No	2/8 (25%)	52.56	23.72	47.70	50.67	47.43
CMoE	Dense-to-MoE	LoRA-4M	2/8 (25%)	57.34	25.77	72.00	51.93	56.88
CMoE	Dense-to-MoE	LoRA-4M	4/16 (25%)	59.14	27.73	72.20	51.14	62.14
Ours	**Dense-to-MoE**	**Router-0.13M**	**25%**	**68.45**	**29.01**	**82.54**	**53.25**	**56.38**

For reimplemented baselines, we use LLaMA-3-8B as the backbone. Table 1 reports the execution or pruning configuration used for each method.

5.4 Evaluation Benchmarks

We use the `lm-evaluation-harness` (Gao et al., 2024) to evaluate on five standard downstream benchmarks: BoolQ (32-shot)(Clark et al., 2019), SciQ (0-shot) (Welbl et al., 2017), PIQA (0-shot) (Bisk et al., 2020), WinoGrande (5-shot) (Sakaguchi et al., 2021), and ARC-Challenge (25-shot) (Clark et al., 2018). Accuracy is reported for all benchmarks.

5.5 Overall Performance

Table 1 compares our method with dense-to-MoE, activation-sparsity, and pruning baselines under comparable sparse-execution settings. Our method achieves an average accuracy of 57.93%, exceeding the five-task average of all evaluated pruning baselines. Compared with CMoE, our method achieves higher average performance: it improves over the best LoRA-adapted CMoE configuration by 3.46 percentage points on average, with particularly large gains on PIQA and SciQ.

R-Sparse achieves a slightly higher average accuracy of 59.36%, but operates at a finer granularity by dynamically selecting individual FFN neurons for each token. In contrast, our method restricts execution to fixed physical expert blocks. Despite this stronger structural constraint, our method remains close in overall accuracy and performs better on PIQA and ARC-Challenge. These results indicate that NMF-guided expert construction and routing preserve competitive model quality while enabling structured sparse execution with physical experts.

For additional context, we compare different post-conversion adaptation regimes, including released LLaMA-MoE checkpoints, in Appendix A.2.

5.6 Inference Efficiency

We evaluate inference efficiency from both FFN computational cost and measured wall-clock performance. All runtime measurements use LLaMA-3-8B in BF16 on a single NVIDIA RTX 4090. Each FFN layer is partitioned into 112 physical experts with 128 neurons per expert, and 28 experts are activated for each token, corresponding to 25% active FFN capacity.

Activating 28 of 112 experts reduces the FFN arithmetic cost from 176.16M to approximately 45.14M MACs per token per layer, corresponding to a 74.4% reduction after accounting for the main router computation. This reduction translates into measurable GPU acceleration: FFN execution reaches 1.17× speedup at 512 tokens and 1.75× at 2048 tokens. The stronger gain at larger

Table 2: FFN computational and execution efficiency. MACs are reported per token per layer. FFN speedups are averaged over six paired layer/input measurements.

Method	Active FFN	MACs / Token / Layer	512 Tokens	2048 Tokens
Dense	100%	176.16M	1.00×	1.00×
Ours	25%	≈45.14M	**1.17**×	**1.75**×

The MAC estimate for our method includes the main matrix multiplications of the routing predictor and factor-to-expert mapping. Routing, dispatch, sorting, and output aggregation overheads are included in the measured wall-clock speedups.

Table 3: End-to-end prefill performance of LLaMA-3-8B on a single RTX 4090.

Seq. Length	Dense Lat.	Ours Lat.	Dense Thru.	Ours Thru.	Speedup
512	59.58 ms	52.87 ms	8,594 tok/s	9,684 tok/s	**1.13**×
1024	142.34 ms	117.79 ms	7,194 tok/s	8,694 tok/s	**1.21**×
2048	347.00 ms	284.48 ms	5,902 tok/s	7,199 tok/s	**1.22**×

token batches indicates that routing and dispatch overheads are increasingly amortized as token-level parallelism grows.

The FFN-level savings translate to consistent end-to-end prefill acceleration. Speedup increases from 1.13× at 512 tokens to 1.22× at 2048 tokens, where throughput improves from approximately 5,902 to 7,199 tokens/s. The smaller end-to-end gain relative to the FFN-level speedup is expected because attention and other non-FFN operations remain unchanged.

Autoregressive decoding does not show consistent acceleration under the tested small-batch settings, where limited token-level parallelism makes routing and dispatch overheads harder to amortize. Detailed decoding results are reported in the appendix.

5.7 Contribution of Expert Construction and Routing

We isolate the contributions of expert construction and routing under the same LLaMA-3-8B backbone, 112-expert structure, Top-28 routing budget, and matched router-training budget.

For the construction ablation, we replace only the neuron-to-expert partition. Our NMF-guided router first estimates neuron activity scores and then aggregates them according to the assignment matrix M:

$$\mathbf{q}_t = (\hat{\mathbf{w}}_t \odot \mathbf{s})\bar{\mathbf{H}}M.$$

Thus, the co-activation baseline uses M_{coact}, while our method uses M_{NMF}, with the same routing formulation.

With the NMF-guided routing method held fixed, NMF-based expert construction improves average accuracy from 52.30% to 57.93% (+5.63 pp). With the NMF expert partition fixed, NMF-guided routing improves average accuracy from 54.10% to 57.93% (+3.83 pp).

These results show that expert construction and routing make distinct contributions to the final performance.

6 Discussion and Limitations

A central property of our framework is that activation factorization exposes complementary neuron-side and token-side views of FFN activation structure. The neuron-side factor loadings describe how individual neurons participate across latent activation factors, while the token-side coefficients indicate which factors are relevant to a particular token. This allows expert construction to exploit multi-factor neuron structure and routing to estimate expert relevance by aggregating predicted ac-

Table 4: Controlled analysis of expert construction and routing. All variants use the same 25% active FFN budget.

Expert Construction	Routing	Avg. Accuracy
Co-activation clustering	NMF-guided	52.30
NMF construction	Representative-neuron	54.10
NMF construction	**NMF-guided**	**57.93**

tivity across all neurons assigned to an expert. The controlled component analysis shows that both uses of the factorized representation contribute to the final model quality.

NMF provides a soft representation in which a neuron may have non-zero loadings on multiple factors, whereas our deployed MoE uses balanced, non-overlapping physical experts. The expert-construction stage therefore converts this overlapping latent structure into executable expert blocks, while the routing mechanism continues to use the factorized representation to estimate token-dependent activity over the full membership of each expert. This separation between a soft activation representation and a discrete execution structure is a key aspect of the proposed conversion framework.

Our evaluation has several limitations. First, the current study focuses on LLaMA-3-8B, and the behavior of the method across other model families and scales remains to be evaluated. Second, the NMF rank, activation sparsification parameter K_s, and expert size are fixed in the main experiments rather than systematically explored. Third, the current implementation provides its clearest runtime benefits for prefill and other token-parallel workloads; small-batch autoregressive decoding does not show consistent acceleration under the tested settings.

7 Conclusion

We presented an activation-factorization framework for converting pretrained dense FFNs into structured sparse MoE layers. Non-negative matrix factorization provides neuron-side factor-loading profiles for balanced expert construction and token-side coefficients for estimating token-dependent expert relevance, allowing routing decisions to reflect distributed neuron activity within each expert. On LLaMA-3-8B, the resulting model operates with 25% active FFN capacity while keeping the original model parameters frozen and achieves competitive downstream performance with lightweight router alignment. The physical expert implementation further provides 1.13–$1.22\times$ end-to-end prefill speedup on a single RTX 4090. These results support activation factorization as a practical representation for low-cost dense-to-MoE conversion and structured sparse execution.

Generative AI Disclosure

In this work, we used generative AI tools to assist with experimental planning, implementation and debugging of research code, interpretation of experimental results, and drafting and editing parts of the manuscript. We also used generative AI tools to help organize related literature and prepare scientific figures.

All AI-assisted outputs were reviewed by the authors. Generated code was inspected and tested through execution of the corresponding experiments, and reported results were obtained from our own experimental runs. Claims and citations were checked against the relevant sources. We take responsibility for the final content of this work.

Reproducibility Statement

The methodology and algorithmic details of our approach are described in Section 4. Section 5 reports the main model, data, evaluation, and inference configurations, while Appendix A provides additional implementation and optimization details.

REFERENCES

Saleh Ashkboos, Maximilian Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations*, volume 2024, pp. 11682–11701, 2024.

Udbhav Bamba, Arnav Chavan, Aryamaan Thakur, Steve Teig, and Deepak Gupta. Dot-moe: Differentiable optimal transport for moefication. *arXiv preprint arXiv:2606.01666*, 2026.

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.

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.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL `https://zenodo.org/records/12608602`.

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.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

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

Daniel Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13, 2000.

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *nature*, 401(6755):788–791, 1999.

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.

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.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Zehua Pei, Hui-Ling Zhen, Lancheng Zou, Xianzhi Yu, Wulong Liu, Sinno Jialin Pan, Mingxuan Yuan, and Bei Yu. Analytical ffn-to-moe restructuring via activation pattern analysis. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4777–4789, 2026.

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

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.

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.

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.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Moefication: Transformer feed-forward layers are mixtures of experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 877–890, 2022.

Zhenyu Zhang, Zechun Liu, Yuandong Tian, Harshit Khaitan, Zhangyang Wang, and Steven Li. R-sparse: Rank-aware activation sparsity for efficient llm inference. In *International Conference on Learning Representations*, volume 2025, pp. 77452–77466, 2025.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. 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*, pp. 15913–15923, 2024.

Simiao Zuo, Qingru Zhang, Chen Liang, Pengcheng He, Tuo Zhao, and Weizhu Chen. Moebert: from bert to mixture-of-experts via importance-guided adaptation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1610–1623, 2022.

A Appendix

A.1 NMF Implementation Details

We fit NMF independently for each FFN layer using the scikit-learn implementation. The input matrix is constructed from the absolute SwiGLU activations over 32,768 calibration tokens sampled from Dolmino-Mix, retaining the $K_s = 2500$ neurons with the largest activation magnitudes for each token. We use rank $r = 128$ with the scikit-learn nndsvdar initialization and coordinate-descent optimization under the Frobenius reconstruction objective. The optimization tolerance is set to 10^{-4} with a maximum of 1500 iterations and random seed 0. No regularization is applied to either factor matrix ($\alpha_W = \alpha_H = 0$, `l1_ratio`=0).

A.2 Performance–Adaptation Trade-off

The main comparison in Table 1 focuses on low-cost sparsification and dense-to-MoE conversion. We further compare our method with CMoE and the released LLaMA-MoE checkpoints across different adaptation regimes in Figure 2.

Our method occupies the low-cost end of this spectrum. With router-only adaptation, it achieves substantially higher average downstream accuracy than CMoE, including the LoRA-adapted variants, while using substantially fewer alignment exposures than the LoRA-adapted CMoE variants.

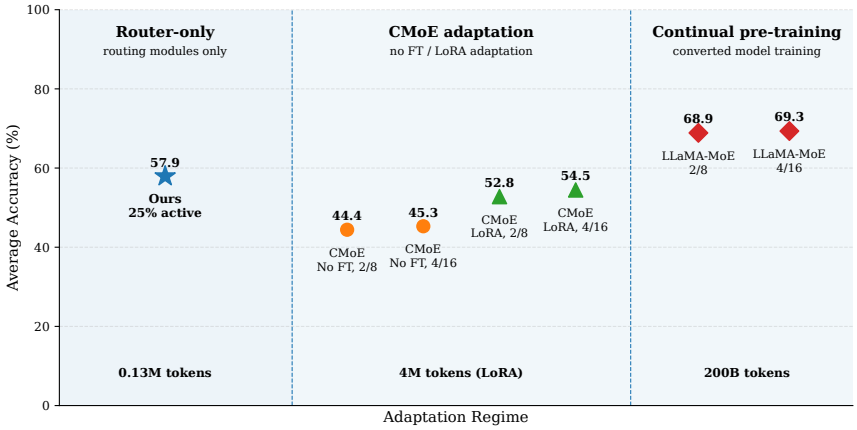

Figure 2: Average downstream accuracy across different post-conversion adaptation regimes and scales. Our method uses router-only adaptation, CMoE uses analytical conversion and is evaluated both without fine-tuning and with additional LoRA adaptation, and LLaMA-MoE uses large-scale continual pre-training of the converted model. The annotations indicate the corresponding post-conversion adaptation scale for each regime. Higher is better.

The released LLaMA-MoE checkpoints achieve higher downstream accuracy, but rely on substantially larger-scale continual pre-training. These results highlight the different quality–adaptation trade-offs of the three dense-to-MoE conversion regimes.

A.3 Balanced Clustering Details

We use a capacity-constrained spherical clustering procedure to convert the neuron representations defined in Section 4.2 into balanced physical experts. For LLaMA-3-8B, each layer contains 14,336 FFN neurons, which are partitioned into 112 experts of exactly 128 neurons each. Since the neuron features v_j are ℓ_2-normalized, clustering is based on cosine similarity.

We initialize the expert centers using a k-means++-style procedure with cosine distance and random seed 42. Given the current centers $\{c_e\}_{e=1}^{E}$, we compute the neuron-to-center similarities $A_{j,e} = v_j^\top c_e$. To enforce equal expert capacity, all neuron–expert pairs are considered in descending similarity order. A pair is accepted when the neuron has not yet been assigned and the corresponding expert contains fewer than 128 neurons. Ties are resolved deterministically by neuron index and then expert index. This assignment procedure ensures that every neuron belongs to exactly one expert and that all experts have the same size.

After each assignment step, the center of an expert is updated as the normalized mean of the neuron features assigned to that expert. Assignment and center updates are repeated for at most 20 iterations and terminate early when the assignment no longer changes. We retain the visited partition with the highest total neuron-to-center cosine similarity. For numerical stability, we use $\epsilon = 10^{-30}$ when normalizing feature vectors and expert centers. The main clustering settings are summarized in Table 5.

Table 5: Settings for the NMF-based expert partition.

Setting	Value
Neurons per layer	14,336
Feature dimension	128
Feature normalization	ℓ_2 per neuron
Similarity	Cosine
Experts per layer	112
Neurons per expert	128 (exact)
Initialization	k-means++-style
Random seed	42
Assignment	Global greedy, exact capacity
Maximum iterations	20
Early stopping	Unchanged assignment
Output selection	Best visited cosine score

A.4 Router Optimization Details

Each FFN layer uses an independent coefficient predictor with architecture $4096 \to 256 \to 128$, a SiLU hidden activation, and a Softplus output. The predictor is first trained using the offline NMF coefficients as supervision. We use AdamW with a learning rate of 10^{-3}, weight decay of 10^{-4}, and gradient clipping at 1.0.

After the physical expert partition is constructed, we further warm up the coefficient predictor to match the token-specific expert distributions induced by the offline NMF coefficients. During this stage, the expert partition, NMF factors, and factor-to-expert mapping are fixed, and only the predictor parameters are updated. We use AdamW with a learning rate of 10^{-4} and weight decay of 10^{-4}.

We then perform model-level router alignment using the original dense LLaMA-3-8B model as the teacher. During alignment, only the coefficient predictors and the multiplicative factor-to-expert mapping adjustment Δ are optimized; the expert partition, NMF factors, attention parameters, and original FFN parameters remain frozen. The alignment objective is

$$\mathcal{L} = 2\mathcal{L}_{\mathrm{KL}} + \mathcal{L}_{\mathrm{CE}} + 0.001\mathcal{L}_z + 0.01\mathcal{L}_{\mathrm{balance}}, \tag{14}$$

where $\mathcal{L}_{\mathrm{KL}}$ matches the full-vocabulary output distribution of the dense teacher, $\mathcal{L}_{\mathrm{CE}}$ is the next-token cross-entropy loss, and $\mathcal{L}_z$ and $\mathcal{L}_{\mathrm{balance}}$ regularize the router logits and expert utilization, respectively.

For alignment, we use AdamW with weight decay 10^{-4} and gradient clipping at 1.0. The learning rate follows a warmup and cosine-decay schedule with a peak learning rate of 5×10^{-5} and a final learning rate of 5×10^{-6}. The final model-level router alignment uses approximately 0.13M input-token exposures.

A.5 Autoregressive Decoding Results

We additionally evaluate autoregressive decoding on LLaMA-3-8B using a prompt length of 512 tokens and generating 128 new tokens. All measurements use BF16 on a single NVIDIA RTX 4090. We evaluate batch sizes of 1, 4, and 8 using the same physical expert implementation as in the prefill experiments.

Table 6: Autoregressive decoding performance with a 512-token prompt and 128 generated tokens.

Batch Size	Dense Decode Throughput	Ours Decode Throughput	Decode Speedup	End-to-End Speedup
1	51.12 tok/s	46.04 tok/s	0.90×	0.91×
4	175.18 tok/s	172.59 tok/s	0.99×	1.01×
8	305.43 tok/s	290.48 tok/s	0.95×	0.99×

Unlike prefill, autoregressive decoding does not show consistent acceleration under these small-batch settings. Since each decoding step processes substantially fewer tokens in parallel, the computation saved by sparse FFN execution is less able to amortize routing, dispatch, sorting, and expert-launch overheads. The batch-size-4 setting reaches approximately equal end-to-end performance, while batch sizes 1 and 8 remain slightly slower than dense execution.