SPECTRAL-MOE: HETEROGENEOUS PER-LAYER EXPERT UTILIZATION IN DENSE-TO-MOE CONVERSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models impose substantial memory demands and arithmetic costs during inference. Dense-to-mixture-of-experts (MoE) conversion is a model optimization technique that aims to reduce these arithmetic costs. Dense-to-MoE conversion partitions pretrained feed-forward network (FFN) neurons into experts and selects a subset for each token. Leading methods fall into two categories: analytic methods produce a converted model quickly using heuristics, while learning methods produce a model much more slowly by optimizing via gradients. Both approaches, however, fix the same active expert count, or "utilization," at every layer. We introduce Spectral-MoE, which learns to allocate a utilization budget heterogeneously across layers. Its name draws a metaphor between separating a dense FFN into experts and separating a light beam into its spectrum. Spectral-MoE consists of two distinct approaches: a static approach that uses a single per-layer scalar which is shared across tokens, and a dynamic approach which emits a per-token schedule. We demonstrate that both allocators improve performance over uniform on both the analytic-based and learning-based SOTA dense-to-MoE methods. At 25% FFN utilization, dynamic allocation reduces WikiText-2 perplexity relative to uniform by 1.04, 2.96, and 2.71 points on the learned base for Llama-2-7B, Llama-3-8B, and Qwen2.5-7B, respectively. The corresponding reductions on the analytic base are 2.45, 6.26, and 2.17 points. On the learned base, dynamic allocation improves six-task mean accuracy over uniform by an average of 1.5, 4.2, and 3.6 percentage points across the three models at FFN utilizations of 12.5%, 25%, and 50%, respectively. These results establish per-layer expert utilization as a novel optimization axis in dense-to-MoE conversion.

1 INTRODUCTION

The scaling of large language models (LLMs) has produced remarkable capabilities in natural language understanding and generation (Brown et al., 2020). Realizing these capabilities, however, requires substantial computation and memory bandwidth, with the latter becoming increasingly problematic for performance (Gholami et al., 2024). Optimizing the transformer (Vaswani et al., 2023), the engine powering nearly all LLMs, has therefore become an extensive area of research. In this domain there exist two primary axes along which compression research is performed: quantization (Frantar et al., 2023; Nagel et al., 2020; Xiao et al., 2024; Lin et al., 2026), and pruning (Frantar & Alistarh, 2023; Sun et al., 2024; Han et al., 2015). A third technique, orthogonal to both quantization and pruning, called dense-to-MoE conversion has also begun to emerge (Zhang et al., 2022).

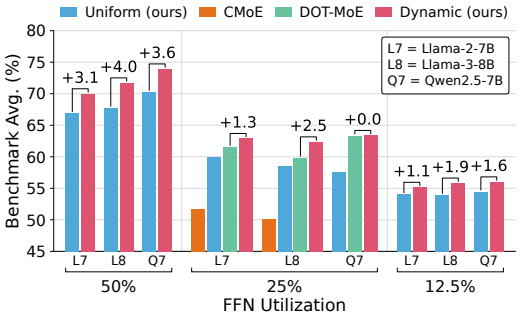


Figure 1: Benchmark accuracy versus utilization. Dynamic, heterogeneous allocation (learned) extends the frontier of dense-to-MoE performance.

A dense FFN evaluates every neuron for every token. Dense-to-MoE conversion groups these neurons into experts and routes each token to a subset, retaining pretrained weights while reducing active

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.

computation. The fraction of experts active for a given token is the "utilization" and the assignment of neurons to expert groups is the neuron-to-expert "partition". Existing methods differ in how they construct the partition and the token-to-expert router (Zhang et al., 2022; Qu et al., 2024; Pei et al., 2026; Bamba et al., 2026; Szatkowski et al., 2024), but all either use a uniform per-layer allocation or a floating utilization budget. However, research on layer-dependent sensitivity to compression in quantization and pruning work (Dong et al., 2019b;a; Yin et al., 2025) motivates investigating heterogeneous per-layer allocation as a potential optimization axis in dense-to-MoE conversion.

To this end, we present Spectral-MoE, which learns per-layer allocation under a fixed utilization budget via an allocator agnostic to the base dense-to-MoE conversion method. Our contributions are as follows:

1. **Heterogeneous per-layer allocation.** We introduce the first formulation of learnable per-layer allocation under a fixed per-token budget for dense-to-MoE conversion. This adds a third optimization axis to dense-to-MoE conversion alongside expert partitioning and routing.
2. **Static and dynamic allocator tiers.** We develop a static allocator that learns one scalar per layer shared across tokens and a dynamic allocator that learns a per-token schedule. Both of these techniques are able to adhere to a user defined utilization budget.
3. **Consistent performance gains.** We evaluate both static and dynamic allocators on the two leading approaches to dense-to-MoE conversion: analytic methods that convert the model quickly using heuristics (Pei et al., 2026), and learning methods that optimize the model through slower gradient-based training (Bamba et al., 2026). Using our approach, we achieve SOTA mean accuracy at 25% utilization and outperform uniform across budgets (Figure 1).

2 Background and Related Work

2.1 Heterogeneous Allocation

Per-Layer Allocation. Prior work on per-layer allocation typically uses pretrained MoEs, with LTE (Zheng et al., 2024) as an exception (Section 2.2). Alloc-MoE's Alloc-L uses dynamic programming over profiled layer sensitivities (Liu et al., 2026), while LExI uses evolutionary search over per-layer output-perturbation estimates (Chitty-Venkata et al., 2025). Both optimize surrogates that predict a schedule's accuracy degradation as a sum of per-layer sensitivities. Spectral-MoE instead learns per-layer utilization directly through gradients.

Per-Token Allocation. Prior work on per-token allocation uses confidence-based routing, null experts, or budget redistribution to vary utilization across tokens. Confidence-based routing sets utilization from the concentration of routing scores (Huang et al., 2024), while AdaMoE introduces null experts to vary utilization (Zeng et al., 2024). SeqTopK and Alloc-MoE's Alloc-T redistribute a layer's utilization budget across tokens (Wen et al., 2026; Liu et al., 2026). Spectral-MoE's dynamic allocator maps hidden states to per-token schedules that conserve a utilization budget across layers.

2.2 Dense-to-MoE Conversion

Analytic Methods. Analytic methods quickly construct neuron-to-expert partitions from heuristics. MoEfication clusters neuron input weights (Zhang et al., 2022), while LLaMA-MoE-v2 uses activation and gradient importance (Qu et al., 2024). CMoE groups neurons by their activation patterns through balanced clustering, then uses a representative neuron from each cluster to construct the router (Pei et al., 2026). Spectral-MoE's analytic base uses CMoE to construct the partition and router, recovers weights with LoRA as CMoE does, then trains the allocator with both frozen.

Learning Methods. Learning methods use slower gradient-based optimization. LLaMA-MoE partitions FFN neurons into experts before continued pretraining (Zhu et al., 2024). LTE jointly trains the router and model weights with an efficiency penalty and heterogeneous utilization follows from thresholding router scores (Zheng et al., 2024). DOT-MoE jointly learns the neuron-to-expert partition and token-to-expert router through differentiable relaxations (Bamba et al., 2026). Spectral-MoE learns per-layer utilization in tandem with this joint optimization, enforcing the same total utilization budget for every token.

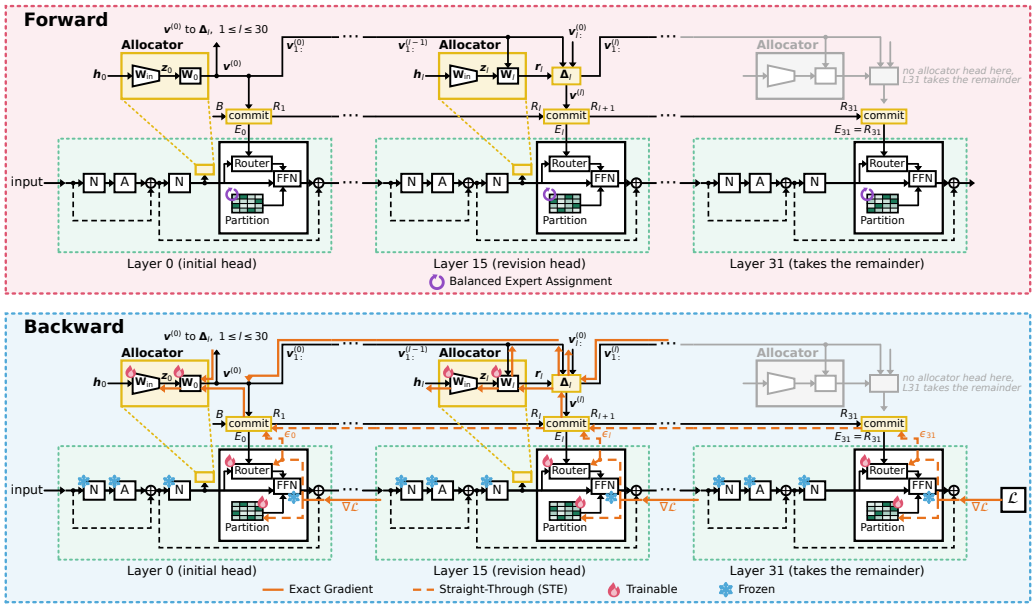


Figure 2: Spectral-MoE learned-base dynamic allocator architecture. During the forward pass, the allocator uses hidden states to initialize and revise per-token schedules, layers commit budget-constrained expert counts, and the final layer consumes the remainder. During the backward pass, loss gradients pass through selections and budget commitments via STEs to train only the allocator, routers, and partition affinities.

3 METHODS

Spectral-MoE learns per-layer allocation under a budget derived from a user-specified target utilization. This requires straight-through estimators (STE) (Bengio et al., 2013) and budget-preserving allocation. We develop these components for static allocation, then extend them to a dynamic allocator that produces token-dependent schedules (Figure 2). Appendix A summarizes the notation.

3.1 UTILIZATION BUDGET

Consider L FFN layers indexed by $l \in \{0, \ldots, L-1\}$. Each layer contains E_r routed and E_s always-active shared experts, for $E_t = E_r + E_s$ experts in total. Each expert contains m neurons, giving FFN width $d_{\text{ffn}} = E_t m$. We enforce a minimum number of routed experts $E_{\min}$ at each layer to ensure all experts are reachable. We use $E_s = E_{\min} = 1$. For each token, the routed count E_l is the number of routed experts selected at layer l, with $E_{\min} \le E_l \le E_r$. The free count $f_l = E_l - E_{\min}$ is the number selected above the minimum, so per-layer FFN utilization is

$$u_l = \frac{E_s + E_l}{E_r + E_s} = \frac{E_s + E_{\min} + f_l}{E_r + E_s}.$$

The user specifies a nominal FFN utilization u_{target}. This determines the uniform initial routed count $\bar{E}$ and total routed budget B:

$$\bar{E} = \lfloor u_{\text{target}}(E_r + E_s) - E_s \rceil, \qquad B = L\bar{E}, \qquad E_{\min} \le \bar{E} \le E_r. \tag{1}$$

Here $\lfloor \cdot \rceil$ rounds to the nearest integer, with ties rounded up. The target must yield a count within the stated bounds. Both allocators enforce $\sum_l E_l = B$ for every token.

With mean free count $\bar{f} = \bar{E} - E_{\min}$, the re-distributable budget and realized utilization satisfy

$$\sum_l f_l = L\bar{f}, \qquad u = \frac{1}{L}\sum_l u_l = \frac{\bar{E} + E_s}{E_r + E_s}.$$

Shared experts and minimum routed counts account for $L(E_s + E_{\min})$ expert activations per token, leaving Lf to allocate.

Note that not all target utilizations are realizable. For Llama-2-7B, $L = 32$, $E_r = 85$, and $E_s = 1$, so a target of 0.25 gives $\bar{E} = 21$ and $B = 672$. The realized utilization is $22/86 \approx 25.58\%$ (Appendix B.1). Both allocators conserve this same integer budget.

3.2 Static Per-Layer Allocation

Static allocation learns one continuous routed count ϵ_l per layer, shared across tokens. We initialize all counts to $\bar{E}$ for a uniform allocation, but optimizer updates can violate the budget or count bounds. After each update, we therefore adjust the counts to satisfy $\sum_l \epsilon_l = B$ and $\frac{1}{2} \le \epsilon_l \le E_r - \frac{1}{2}$. Appendix B.2 details how we enforce these bounds and resolve any budget excess or deficit.

Hard Expert Selection. Let Γ_B denote largest-remainder rounding with integer-bound repair, producing the forward schedule $\boldsymbol{E} = \Gamma_B(\boldsymbol{\epsilon})$. We floor each count, then add one to the $B - \sum_l \lfloor \epsilon_l \rfloor$ layers with the largest fractional parts. If integer clipping changes the sum, we adjust eligible counts to restore B (Appendix B.2). Each router selects its top-E_l experts per token, allowing expert identities to vary under the shared schedule.

Gradients Through Hard Selection. Let $\mathrm{rank}_l(t, e) \in \{0, \dots, E_r - 1\}$ rank expert e for token t at layer l, with zero assigned to the highest router score. Treating ranks as constants during backpropagation, we define the soft inclusion

$$\tilde{R}_{te}^{(l)} = \sigma\left(\frac{\epsilon_l - \mathrm{rank}_l(t, e) - \frac{1}{2}}{T}\right), \tag{2}$$

where σ is the logistic sigmoid and $T > 0$ the inclusion temperature (Appendix F). This makes soft inclusion most sensitive to ϵ_l for experts near the selection boundary. Soft inclusions serve only to compute count gradients and need not sum to ϵ_l. The forward pass selects exactly E_l routed experts at each layer, with the integer allocation rule enforcing $\sum_l E_l = B$.

For n tokens, let $\mathbf{R} \in \{0, 1\}^{n \times E_r}$ indicate routed selections, with $R_{te} = 1$ when token t selects expert e, and let $\mathbf{P}$ contain router softmax probabilities. With sg denoting stop-gradient, we add the count-gradient term to the inherited router STE (Bamba et al., 2026):

$$\mathbf{R}_{\mathrm{STE}} = \mathbf{R} + (\mathbf{P} - \mathrm{sg}[\mathbf{P}]) + (\tilde{\mathbf{R}}^{(l)} - \mathrm{sg}[\tilde{\mathbf{R}}^{(l)}]). \tag{3}$$

Its forward value is $\mathbf{R}$, while router and count gradients flow through $\mathbf{P}$ and $\tilde{\mathbf{R}}^{(l)}$, respectively.

Integration with the Conversion Base. The learned-base Spectral-MoE jointly trains partition affinities, router weights, and the allocator while keeping pretrained weights frozen. The analytic-base Spectral-MoE first recovers weights with LoRA, then freezes weights, partition, and router while training the allocator. Both bases retain an always-active shared expert, selected by activation rate on the analytic base and calibrated output contribution on the learned base. The analytic base omits the router STE term in Equation 3. Inherited scalar routing coefficients scale expert outputs and count gradients. Training also evaluates the next-ranked expert, if available, with zero forward contribution. Its soft-inclusion gradient helps assess the benefit of selecting another expert. Both learned and analytic bases use an objective which combines language-modeling cross-entropy with dense-teacher distillation. The learned base also retains router regularization and load balancing.

3.3 Dynamic Per-Layer Allocation

Dynamic allocation revises each token's remaining schedule from the current hidden state. A schedule predicted only at the first layer cannot respond to later hidden states. A shared projection feeds an initial output head and subsequent revision heads. The final layer has no revision head because it consumes the remainder.

Allocator Architecture. Let d_{model} be the residual-stream width and $q_l = L - l$ the number of uncommitted layers, including the current layer. The shared projection $\mathbf{W}_{\text{in}}$, initial head $\mathbf{W}_0$, and

revision heads $\mathbf{W}_l$ have shapes

$$\mathbf{W}_{\text{in}} \in \mathbb{R}^{L\times(d_{\text{model}}+1)}, \qquad \mathbf{W}_0 \in \mathbb{R}^{L\times L}, \qquad \mathbf{W}_l \in \mathbb{R}^{q_l\times(L+q_l)} \quad (1 \le l \le L-2),$$

where the additional input dimension of $\mathbf{W}_{\text{in}}$ carries the log-RMS of the unnormalized FFN input. Biases are omitted. Including biases, the allocator adds $158{,}942$ parameters for $L = 32$ and $d_{\text{model}} = 4096$, negligible compared to the billion-scale models being converted.

For hidden state input $\boldsymbol{h}_l$, define its root-mean-square (RMS) normalized form $\boldsymbol{x}_l = \boldsymbol{h}_l / \operatorname{RMS}(\boldsymbol{h}_l)$. The schedule $\boldsymbol{v}^{(l)} \in \mathbb{R}^{q_l}$ contains logits for layers $l, \dots, L-1$, with entry zero corresponding to the current layer. Let $\boldsymbol{r}_l$ be a proposed revision and $\boldsymbol{\Delta}_l$ the accumulated correction to the initial schedule. The $\tanh$ operations bound each proposed logit revision and each logit's accumulated deviation from the initial schedule. Square brackets concatenate features:

$$\begin{aligned}
\boldsymbol{z}_l &= \operatorname{GELU}\left(\mathbf{W}_{\text{in}}[\boldsymbol{x}_l, \log \operatorname{RMS}(\boldsymbol{h}_l)]\right), & \boldsymbol{v}^{(0)} &= \mathbf{W}_0 \boldsymbol{z}_0, \\
\boldsymbol{r}_l &= \tanh\left(\mathbf{W}_l[\boldsymbol{z}_l, \boldsymbol{v}_{1:}^{(l-1)}]\right), & \boldsymbol{\Delta}_l &= \tanh\left((\boldsymbol{\Delta}_{l-1})_{1:} + \boldsymbol{r}_l\right), \\
\boldsymbol{v}^{(l)} &= \boldsymbol{v}_{l:}^{(0)} + \boldsymbol{\Delta}_l, & \boldsymbol{\Delta}_0 &= \mathbf{0}.
\end{aligned}$$

Concatenating the projected hidden state $\boldsymbol{z}_l$ with the remaining schedule $\boldsymbol{v}_{1:}^{(l-1)}$ lets each head correct the same hidden state differently depending on the allocation already proposed. The log-RMS feature restores the scale removed by normalization. Small nonzero head weights produce uniform integer counts on initialization while preserving gradients into the shared projection.

Exact Budget Conservation. Let $S_l = \sum_{j<l} E_j$ be the routed budget committed at earlier layers j, $R_l = B - S_l$ its remainder, and $\pi_l = \operatorname{softmax}(\boldsymbol{v}^{(l)})_0$ the current layer's share. After reserving $E_{\min}$ experts for each uncommitted layer, we assign share π_l of the surplus to the current layer. Let a_l and b_l denote the minimum and maximum feasible routed counts at the current layer, ensuring the remainder fits within later layers' count bounds. We round and clip the continuous proposal ϵ_l to these bounds:

$$\begin{aligned}
\epsilon_l &= E_{\min} + \max(R_l - q_l E_{\min}, 0)\pi_l, \\
a_l &= \max\{E_{\min}, R_l - (q_l - 1)E_r\}, \qquad b_l = \min\{E_r, R_l - (q_l - 1)E_{\min}\}, \\
E_l &= \operatorname{clip}_{[a_l, b_l]}\left(\lfloor \epsilon_l \rceil\right).
\end{aligned}$$

Here $\lfloor \cdot \rceil$ rounds ties to even, and clipping preserves feasibility under any tie rule.

For a feasible initial budget, the bounds ensure $(q_l - 1)E_{\min} \le R_l - E_l \le (q_l - 1)E_r$ at every layer. At the final layer, $q_l - 1 = 0$, forcing $E_{L-1} = R_{L-1}$ and hence $\sum_l E_l = B$ for every token.

Gradients Through the Utilization Schedule. The soft inclusion in Equation 2 supplies gradients through the continuous proposal ϵ_l to both schedule shares and the remaining budget. For positive surplus with R_l fixed, the derivative with respect to schedule logit $v_j^{(l)}$ is

$$\frac{\partial \epsilon_l}{\partial v_j^{(l)}} = (\epsilon_l - E_{\min})\left(\delta_{0j} - \operatorname{softmax}(\boldsymbol{v}^{(l)})_j\right),$$

Here schedule index j refers to layer $l + j$, and δ_{0j} is one for $j = 0$ and zero otherwise. Increasing a later layer's logit reduces the current layer's surplus share, coupling allocation decisions across the remaining layers. We evaluate soft inclusion using ϵ_l clipped to $[a_l, b_l]$. During backpropagation, we treat clipping as the identity, allowing gradients to reach the continuous proposal even when a bound is active.

Earlier integer commitments also affect later proposals by changing the remaining budget. To preserve this gradient path, we compute $B - \widehat{S}_l$ using an STE over earlier layers j:

$$\widehat{S}_l = \sum_{j<l}\left(E_j + \epsilon_j - \operatorname{sg}[\epsilon_j]\right), \qquad \frac{\partial \epsilon_l}{\partial \widehat{S}_l} = -\pi_l.$$

The derivative assumes positive surplus and holds the current schedule logits constant. The proposal terms cancel in the forward pass, giving $\widehat{S}_l = S_l$ and preserving the exact budget. During backpropagation, E_j is held constant and each summand passes gradients to the proposal ϵ_j before rounding or clipping. Increasing the committed budget therefore reduces the current proposal at rate π_l, allowing later selections, including the final layer's, to influence earlier allocations.

Method		PPL (↓)	BoolQ [32, acc]	SciQ [0, acc]	PIQA [0, acc]	WinoG. [5, acc]	ARC-C [25, acc-n]	HellaS. [10, acc-n]	Avg.
Llama-2-7B									
Dense		5.47	82.0	94.0	78.1	74.3	52.5	78.9	76.6
LLaMA-MoE†		–	37.8	20.0	49.7	50.1	25.8	26.2	34.9
LLaMA-MoE-v2†		–	51.3	67.0	56.6	52.9	25.7	35.1	48.1
CMoE†		–	55.0	77.5	57.1	54.1	**27.6**	38.8	51.7
Spectral-MoE Analytic	Uniform	21.86	59.2	84.0	59.0	52.8	26.1	39.1	53.4
	Static	20.54	61.7	**85.0**	59.6	**56.1**	25.0	39.9	**54.6**
	Dynamic	**19.41**	**61.8**	83.1	**60.2**	54.8	26.5	**41.5**	**54.6**
DOT-MoE‡		–	66.4	89.6	67.2	57.9	34.0	53.9	61.5
Spectral-MoE Learned	Uniform	11.61	63.2	88.9	65.0	57.3	32.3	52.7	59.9
	Static	11.50	64.8	**89.8**	65.3	58.4	33.6	52.8	60.8
	Dynamic	**10.57**	**67.1**	88.8	**68.4**	**60.5**	**35.8**	**56.5**	**62.8**
Llama-3-8B									
Dense		6.14	83.2	96.2	79.6	77.3	58.3	82.1	79.4
LLaMA-MoE†		–	37.8	20.3	49.0	50.7	25.7	25.8	34.9
CMoE†		–	56.9	72.0	57.3	51.9	**25.8**	36.7	50.1
Spectral-MoE Analytic	Uniform	45.24	60.9	82.2	58.9	**52.5**	23.7	36.8	52.5
	Static	40.48	61.9	82.1	**59.7**	51.1	24.1	36.5	52.6
	Dynamic	**38.99**	**62.3**	**82.6**	58.5	**52.5**	25.3	**37.2**	**53.1**
DOT-MoE‡		–	62.3	90.9	65.0	56.0	34.1	50.1	59.8
Spectral-MoE Learned	Uniform	19.40	59.1	87.2	64.5	**58.2**	32.3	49.3	58.5
	Static	18.19	63.1	89.4	65.7	57.5	32.7	49.5	59.7
	Dynamic	**16.44**	**66.9**	**91.8**	**67.6**	**58.2**	**35.7**	**53.7**	**62.3**
Qwen2.5-7B									
Dense		6.85	88.0	96.6	78.6	76.2	64.1	80.3	80.6
Spectral-MoE Analytic	Uniform	47.07	59.4	**78.3**	**58.7**	53.0	25.6	35.9	51.8
	Static	45.61	**60.3**	76.8	58.2	52.7	25.3	36.0	51.5
	Dynamic	**44.90**	59.4	77.4	57.8	**53.4**	**27.6**	**36.4**	**52.0**
DOT-MoE‡		–	64.3	**93.9**	68.5	**58.7**	39.6	55.0	**63.3**
Spectral-MoE Learned	Uniform	16.92	43.7	89.4	66.9	58.0	35.4	51.9	57.6
	Static	15.99	**65.9**	90.8	65.6	57.8	35.2	51.5	61.1
	Dynamic	**14.21**	65.2	92.6	**69.0**	57.5	**39.8**	**55.8**	**63.3**

Table 1: Dense-to-MoE conversion at 25% FFN utilization. Headers give [shots, metric]. Downstream benchmarks use the evaluation harness from Gao et al. (2024). Analytic and Learned use CMoE and DOT-MoE bases, respectively. Uniform denotes our custom implementation of the bases. Uniform, static, and dynamic runs are seed matched (Appendix D). †: published Pei et al. (2026). ‡: published Bamba et al. (2026).

4 Results

4.1 Experimental Settings

Implementation. We implement Spectral-MoE using PyTorch (Paszke et al., 2019) and Hugging Face Transformers (Wolf et al., 2020), with learned-base conversion run on four NVIDIA H100 GPUs and analytic-base conversion run on one. All converted models use 128 neurons per expert, following Bamba et al. (2026), giving 86, 112, and 148 experts per layer for Llama-2-7B, Llama-3-8B, and Qwen2.5-7B, respectively. On the analytic base, we construct the partition and router following CMoE (Pei et al., 2026), recover weights with LoRA (Hu et al., 2021), then freeze the recovered weights, partition, and router while training only the allocator. On the learned base, we extend DOT-MoE's joint partition and router optimization (Bamba et al., 2026) to train partition affinities, router weights, and allocator parameters in tandem while keeping pretrained weights frozen. We optimize trainable parameters with AdamW (Loshchilov & Hutter, 2019). Appendix C details the full testing and training configurations.

Models and Evaluation. We evaluate Llama-2-7B (Touvron et al., 2023), Llama-3-8B (Grattafiori et al., 2024), and Qwen2.5-7B (Qwen et al., 2025). We measure language-modeling performance using WikiText-2 test perplexity (Merity et al., 2016). Following the canonical benchmark selection of dense-to-MoE methods (Pei et al., 2026; Bamba et al., 2026), we evaluate downstream benchmark performance using the harness provided by Gao et al. (2024) on BoolQ (Clark et al., 2019), SciQ (Welbl et al., 2017), PIQA (Bisk et al., 2019), WinoGrande (Sakaguchi et al., 2019), ARC-Challenge (Clark et al., 2018), and HellaSwag (Zellers et al., 2019).

Baselines. CMoE (Pei et al., 2026) and DOT-MoE (Bamba et al., 2026) represent the current SOTA in analytic and learned conversion, respectively, while LLaMA-MoE (Zhu et al., 2024) and LLaMA-MoE-v2 (Qu et al., 2024) are representative of earlier work. Given the substantial improvement of CMoE and DOT-MoE over previous work (Table 1), we build Spectral-MoE on these two methods and include the earlier methods as reference points. We were unable to locate a public implementation of DOT-MoE during our experiments. As a result, the reported uniform numbers are from our own implementation of DOT-MoE based on the details provided in the paper. Given our baseline implementation falls short of the published accuracy, we report both sets of results. Appendix C.4 describes our implementations of the conversion bases.

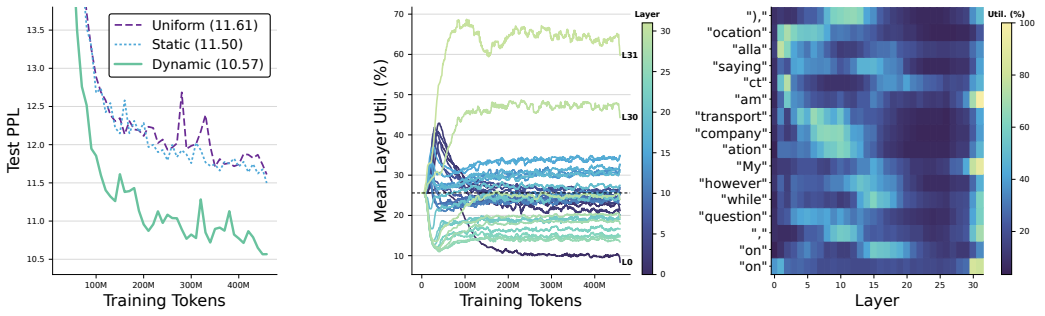
4.2 Benchmark Performance

Comparison with Previous Works. Table 1 compares Spectral-MoE with uniform allocation and published methods at 25% FFN utilization. On the analytic base, dynamic allocation reduces perplexity by 2.17–6.26 points and improves six-task mean accuracy by 0.2–1.3 percentage points over uniform, exceeding published CMoE mean accuracy by 3.0 points on both Llama-2-7B and Llama-3-8B. On the learned base, dynamic allocation reduces perplexity by 1.04–2.96 points and improves mean accuracy by 3.0–5.8 percentage points over uniform. It exceeds published DOT-MoE mean accuracy by 1.3 and 2.5 points on Llama-2-7B and Llama-3-8B, respectively, and matches Qwen2.5-7B at 63.3% at the reported precision. Both allocators lower perplexity across all models and bases, demonstrating robust benefits across different partition and router construction techniques.

Utilization Sweep. Table 2 compares Spectral-MoE's dynamic allocator with uniform allocation on the learned base at 12.5%, 25%, and 50% FFN utilization. Perplexity and six-task mean accuracy improve for every model–budget pair, with accuracy gains averaging 1.5, 4.2, and 3.6 percentage points across models, respectively. Perplexity falls by 1.06–2.46, 1.04–2.96, and 0.91–2.55 points at

FFN Util.	Model	Allocator	PPL (↓)	BoolQ [32, acc]	SciQ [0, acc]	PIQA [0, acc]	WinoG. [5, acc]	ARC-C [25, acc-n]	HellaS. [10, acc-n]	Avg.
0.125	Llama-2-7B	Uniform	15.93	55.4	84.7	**60.9**	**54.3**	**27.0**	41.4	54.0
		Dynamic	**14.87**	**60.5**	**85.2**	60.3	53.0	26.9	**44.6**	**55.1**
	Llama-3-8B	Uniform	27.69	59.9	84.6	60.8	52.5	25.9	39.5	53.9
		Dynamic	**25.46**	**62.1**	**86.3**	**62.6**	**54.5**	**28.2**	**40.9**	**55.8**
	Qwen2.5-7B	Uniform	23.60	53.5	84.2	**63.1**	**55.6**	28.5	41.6	54.4
		Dynamic	**21.14**	**62.0**	**85.3**	61.8	54.1	**29.2**	**43.4**	**56.0**
0.25	Llama-2-7B	Uniform	11.61	63.2	**88.9**	65.0	57.3	32.3	52.7	59.9
		Dynamic	**10.57**	**67.1**	88.8	**68.4**	**60.5**	**35.8**	**56.5**	**62.8**
	Llama-3-8B	Uniform	19.40	59.1	87.2	64.5	**58.2**	32.3	49.3	58.5
		Dynamic	**16.44**	**66.9**	**91.8**	**67.6**	**58.2**	**35.7**	**53.7**	**62.3**
	Qwen2.5-7B	Uniform	16.92	43.7	89.4	66.9	**58.0**	35.4	51.9	57.6
		Dynamic	**14.21**	**65.2**	**92.6**	**69.0**	57.5	**39.8**	**55.8**	**63.3**
0.50	Llama-2-7B	Uniform	8.26	68.3	90.9	70.5	64.6	40.6	65.9	66.8
		Dynamic	**7.35**	**75.7**	**92.7**	**73.6**	**65.0**	**42.9**	**69.7**	**69.9**
	Llama-3-8B	Uniform	12.20	71.4	93.5	71.3	63.1	41.7	65.2	67.7
		Dynamic	**9.64**	**73.1**	**94.8**	**75.4**	**67.6**	**47.7**	**71.4**	**71.7**
	Qwen2.5-7B	Uniform	11.34	75.6	94.8	72.4	66.1	46.8	65.6	70.2
		Dynamic	**9.12**	**79.4**	**95.4**	**75.8**	**67.1**	**54.5**	**71.0**	**73.9**

Table 2: Spectral-MoE learned-base utilization sweep. Dynamic allocation improves perplexity and six-task mean accuracy over uniform across all tested models and utilizations. Metrics follow Table 1.



(a) Perplexity convergence (b) Per-layer convergence (c) Per-token schedules

Figure 3: Spectral-MoE on the Llama-2-7B learned base at a nominal 25% FFN utilization. (a) WikiText-2 test perplexity. (b) Each layer's mean utilization under dynamic allocation. The dashed line marks the uniform start (25.6%). (c) Final-step schedules of 16 sampled tokens. The two unique schedules for the token "on" showcase the allocator adapting to context.

12.5%, 25%, and 50% utilization, respectively. These gains demonstrate that the benefits of dynamic allocation extend across the tested utilization range and motivate systematic utilization sweeps in future dense-to-MoE work.

4.3 Learned Heterogeneity

Perplexity Convergence. Figure 3(a) compares perplexity convergence on the learned Llama-2-7B base at 25% FFN utilization. Dynamic allocation finishes at 10.57 perplexity, versus 11.61 for uniform and 11.50 for static allocation. Static allocation is initialized at uniform (Section 3.2), but ultimately learns a nonuniform schedule. Its small final gain suggests a slightly improved allocation, although training fluctuations prevent a firm conclusion. Spectral-MoE's dynamic allocator, however, maintains a significant lead throughout. Appendix E provides the experimental specification and per-layer profiles across models.

Per-Layer Average Utilization. Figure 3(b) tracks each layer's mean utilization during dynamic allocation training. The initially uniform schedule develops into a heterogeneous layer profile whose averages stabilize late in training. Together with the perplexity gains, this stabilization supports the assertion that Spectral-MoE learns a useful allocation. The resulting profile also offers a unique perspective on the relative importance of each layer during dense-to-MoE conversion.

Per-Token Schedule Samples. Figure 3(c) shows sampled per-token schedules from the final training step. The schedules vary substantially across tokens while preserving the same total utilization budget. Even positions sharing a token ID receive different allocations, consistent with the allocator responding to changing hidden states. Spectral-MoE's dynamic allocator can revise each token's remaining schedule, combining stable layer averages with context-dependent allocation. Appendix E provides the experimental specification.

4.4 Training Stability Considerations

Neuron-to-Expert Assignment Turnover During Training. Figure 4 measures changes in neuron-to-expert assignments during the dense-to-MoE conversion of a learned-base, dynamic Llama-2-7B model at 25% FFN utilization. Turnover is the fraction of routed neurons assigned to a different expert between consecutive training steps, averaged across 32 layers and excluding shared neurons.

Turnover averages 8.92% over the final 500 steps but falls to 0.33% at the final step. The sharp drop at the end is consistent with the learning-rate decay toward zero, which eventually prevents updates to the partition affinities. Prior to the sharp drop, however, the turnover rate appears to hold relatively constant despite the decreasing learning rate. One issue sustained turnover introduces is that the router is forced to adapt to a constantly changing expert composition, potentially resulting in a partially adapted router at the end of joint training. Despite this issue, learned dense-to-MoE conversion still performs quite well.



Figure 4: Fraction of routed neurons assigned to a different expert between consecutive training steps, measured on a sample learned-base, dynamic dense-to-MoE conversion.

This observation motivates us to test whether additional router optimization after freezing expert assignments boosts performance. We leave the investigation of learning the partition itself in a more stable fashion to future work.

Frozen Checkpoint Router Refitting. To limit additional token use, we refit routers by closed-form ridge regression (Hoerl & Kennard, 1970). We pass $32{,}768$ training tokens through each frozen checkpoint from the end of joint training, recording router inputs and computing an activation-based importance proxy I_{te} at each layer. We then use these inputs to fit router weights and expert biases to normalized regression labels Y_{te}, keeping all other parameters frozen. For token t and routed expert e,

$$Y_{te} = \frac{I_{te} - \mu_t}{\max(s_t, 10^{-6})}, \qquad I_{te} = \sum_{j:M_{je}=1} |H_{tj}| \, \|\mathbf{W}_{\text{down}}[:, j]\|_1, \tag{4}$$

where $M_{je} = 1$ assigns neuron j to expert e, H_{tj} is the neuron's FFN activation, and μ_t, s_t are the mean and sample standard deviation of I_{te} across routed experts.

We compare each router's expert selections before and after refitting with the expert ranking given by I_{te}. All selections use the same pre-refit hidden states and expert counts. "Overlap" is the fraction of shared selections. "Coverage" is the selected experts' summed importance divided by the reference total. Both metrics improve for both checkpoints, but six-task mean accuracy changes by $+1.0$ and -0.4 percentage points for seeds *A* and *B* (Table 3). While the naive ridge regression router refitting yields mixed results, there may exist additional methods which improve accuracy with minimal token overhead. We leave the exploration of such approaches to future work.

Seed	Router	Overlap	Coverage	BoolQ	SciQ	PIQA	WinoG.	ARC-C	HellaS.	Avg.
A	Original	50.11	87.66	69.5	89.5	65.8	57.4	32.0	56.2	61.74
	Refit	69.42	95.62	66.8	89.7	67.5	60.9	34.9	56.6	62.73
B	Original	50.02	87.66	67.1	88.8	68.4	60.5	35.8	56.5	62.84
	Refit	69.53	95.66	65.4	88.4	67.8	60.8	35.8	56.6	62.48

Table 3: Router refitting on learned dynamic Llama-2-7B at 25% FFN utilization. Original and Refit denote checkpoint and ridge-regression routers. Entries are percentages; metrics follow Table 1.

5 Conclusion

Spectral-MoE learns per-layer expert utilization during dense-to-MoE conversion under a user-specified budget. Static allocation learns a schedule shared across tokens, while dynamic allocation revises each token's schedule and conserves the same total budget. Dynamic allocation lowers perplexity on both conversion bases across all three models at nominal 25% utilization, with learned-base perplexity and mean-accuracy gains extending to 12.5% and 50%. It also exceeds published DOT-MoE mean accuracy on Llama-2-7B and Llama-3-8B and matches it on Qwen2.5-7B. The learned schedules combine stable layer averages with context-dependent variation while preserving the same routed budget for every token. Future work should examine whether stabilizing expert assignments improves the joint adaptation of routers and allocators.

AI USE STATEMENT

The main use of generative AI in this work was for code generation. Specifically, we created a software specification manually and used generative AI to implement that specification. The generated code was then reviewed manually to ensure correctness. Generative AI was also used for anonymizing the code for the initial submission. During the drafting and editing process of the manuscript generative AI was used to reword sentences, make certain paragraphs more concise, format equations, and identify related works for manual review. The authors audited all AI generated artifacts and take full responsibility for the content of this work.

REPRODUCIBILITY STATEMENT

In order to ensure the reproducibility of our work we provide detailed training parameters and evaluation procedures in appendix C. Furthermore we also provide code which fully implements the techniques described in this paper.

REFERENCES

Udbhav Bamba, Arnav Chavan, Aryamaan Thakur, Steve Teig, and Deepak Gupta. Dot-moe: Differentiable optimal transport for moefication. 2026. URL https://arxiv.org/abs/2606.01666.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. URL https://arxiv.org/abs/1308.3432.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. 2019. URL https://arxiv.org/abs/1911.11641.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020. URL https://arxiv.org/abs/2005.14165.

Krishna Teja Chitty-Venkata, Sandeep Madireddy, Murali Emani, and Venkatram Vishwanath. Lexi: Layer-adaptive active experts for efficient moe model inference. 2025. URL https://arxiv.org/abs/2509.02753.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. 2019. URL https://arxiv.org/abs/1905.10044.

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. 2018. URL https://arxiv.org/abs/1803.05457.

Zhen Dong, Zhewei Yao, Yaohui Cai, Daiyaan Arfeen, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. 2019a. URL https://arxiv.org/abs/1911.03852.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. 2019b. URL https://arxiv.org/abs/1905.03696.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. 2023. URL https://arxiv.org/abs/2301.00774.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. 2023. URL https://arxiv.org/abs/2210.17323.

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.

Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W. Mahoney, and Kurt Keutzer. AI and memory wall. *IEEE Micro*, 44(3):33–39, 2024. doi: 10.1109/MM.2024.3373763. URL https://doi.org/10.1109/MM.2024.3373763.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models. 2024. URL https://arxiv.org/abs/2407.21783.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. 2015. URL https://arxiv.org/abs/1506.02626.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. doi: 10.1080/00401706.1970.10488634. URL https://doi.org/10.1080/00401706.1970.10488634.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. 2021. URL https://arxiv.org/abs/2106.09685.

Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. Harder tasks need more experts: Dynamic routing in moe models. 2024. URL https://arxiv.org/abs/2403.07652.

R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987. doi: 10.1007/BF02278710. URL https://doi.org/10.1007/BF02278710.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. 2026. URL https://arxiv.org/abs/2306.00978.

Baihui Liu, Kaiyuan Tian, Wei Wang, Zhaoning Zhang, Linbo Qiao, and Dongsheng Li. Alloc-moe: Budget-aware expert activation allocation for efficient mixture-of-experts inference. 2026. URL https://arxiv.org/abs/2604.08133.

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28 (2):129–137, 1982. doi: 10.1109/TIT.1982.1056489. URL https://doi.org/10.1109/TIT.1982.1056489.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. 2019. URL https://arxiv.org/abs/1711.05101.

Mikko I. Malinen and Pasi Fränti. Balanced K-means for clustering. In *Structural, Syntactic, and Statistical Pattern Recognition*, volume 8621 of *Lecture Notes in Computer Science*, pp. 32–41. Springer, 2014. doi: 10.1007/978-3-662-44415-3_4. URL https://doi.org/10.1007/978-3-662-44415-3_4.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. 2016. URL https://arxiv.org/abs/1609.07843.

Markus Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. 2020. URL https://arxiv.org/abs/2004.10568.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 olmo 2 furious. 2025. URL https://arxiv.org/abs/2501.00656.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. 2019. URL https://arxiv.org/abs/1912.01703.

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. 2026. URL https://arxiv.org/abs/2502.04416.

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. 2024. URL https://arxiv.org/abs/2411.15708.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. 2025. URL https://arxiv.org/abs/2412.15115.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. 2019. URL https://arxiv.org/abs/1907.10641.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. 2024. URL https://arxiv.org/abs/2306.11695.

Filip Szatkowski, Bartosz Wójcik, Mikołaj Piórczyński, and Simone Scardapane. Exploiting activation sparsity with dense to dynamic-k mixture-of-experts conversion, 2024. URL https://arxiv.org/abs/2310.04361.

Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open foundation and fine-tuned chat models. 2023. URL https://arxiv.org/abs/2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2023. URL https://arxiv.org/abs/1706.03762.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. 2017. URL https://arxiv.org/abs/1707.06209.

Tiansheng Wen, Yifei Wang, Aosong Feng, Long Ma, Xinyang Liu, Yifan Wang, Lixuan Guo, Bo Chen, Stefanie Jegelka, and Chenyu You. Route experts by sequence, not by token. 2026. URL https://arxiv.org/abs/2511.06494.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing. 2020. URL https://arxiv.org/abs/1910.03771.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. 2024. URL https://arxiv.org/abs/2211.10438.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Jaiswal, Mykola Pechenizkiy, Yi Liang, Michael Bendersky, Zhangyang Wang, and Shiwei Liu. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. 2025. URL https://arxiv.org/abs/2310.05175.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? 2019. URL https://arxiv.org/abs/1905.07830.

Zihao Zeng, Yibo Miao, Hongcheng Gao, Hao Zhang, and Zhijie Deng. Adamoe: Token-adaptive routing with null experts for mixture-of-experts language models. 2024. URL https://arxiv.org/abs/2406.13233.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Moefication: Transformer feed-forward layers are mixtures of experts. 2022. URL https://arxiv.org/abs/2110.01786.

Haizhong Zheng, Xiaoyan Bai, Xueshen Liu, Z. Morley Mao, Beidi Chen, Fan Lai, and Atul Prakash. Learn to be efficient: Build structured sparsity in large language models. 2024. URL https://arxiv.org/abs/2402.06126.

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. 2024. URL https://arxiv.org/abs/2406.16554.

A NOTATION

Table 4: Notation used in the main text and appendices. Routed counts measure selected routed experts, whereas FFN utilization includes the shared expert. Bold italic symbols denote vectors and bold upright symbols denote matrices.

Symbol	Definition	Symbol	Definition
L, l	FFN layers and zero-based layer index.	q_l, π_l	Uncommitted layers $L-l$, and the current layer's softmax share.
n, t, e	Token count, token index, and routed-expert index.	$\boldsymbol{v}^{(l)}$, $\boldsymbol{r}_l$, $\boldsymbol{\Delta}_l$	Vector of schedule logits, proposed revisions, and accumulated corrections in $\mathbb{R}^{q_l}$.
m, d_{ffn}, d_{model}	Neurons per expert, FFN width, and residual-stream width.	S_l, R_l, $\widehat{S}_l$	Committed budget, remainder $B-S_l$, and the STE for S_l.
E_t, E_r, E_s	Total, routed, and shared experts per layer, with $E_t = E_r + E_s$.	a_l, b_l	Feasible integer bounds on the current commitment.
$E_{\min}$, E_l	Minimum routed count and selected routed count at layer l.	$\boldsymbol{h}_l$, $\boldsymbol{x}_l$, $\boldsymbol{z}_l$	FFN input, its RMS-normalized vector, and the allocator's shared-projection output.
f_l, $\bar{f}$	Free count $E_l - E_{\min}$ and its layer mean.	$\mathbf{W}_{\text{in}}$, $\mathbf{W}_0$, $\mathbf{W}_l$	Matrices for the shared projection, initial output head, and revision output head.
B, $\bar{E}$	Per-token routed budget and the uniform count B/L.	$\mathbf{H}$, H_{tj}	FFN activation matrix and its scalar entry for token t, neuron j.
u_{target}, u, u_l	User-specified utilization, realized mean, and per-layer utilization.	$\mathbf{W}_{\text{down}}$	FFN output projection matrix.
ϵ_l, $\boldsymbol{\epsilon}$	Scalar continuous count or proposal for layer l, and the static count vector.	$\mathbf{M}$, M_{je}	Hard neuron-to-expert partition matrix. $M_{je}=1$ assigns neuron j to expert e.
$\boldsymbol{E}$, Γ_B	Integer count vector and largest-remainder rounding operator with bound repair.	$\mathbf{A}$	Soft neuron-to-expert partition affinity matrix.
T, σ	Inclusion temperature and the logistic sigmoid.	$\mathbf{W}_r$, τ	Router weight matrix and partition transport temperature.
$\text{rank}_l(t,e)$	Rank of expert e by token t's router score at layer l.	$\mathcal{L}$, $\nabla\mathcal{L}$	Training objective and its gradient.
$\mathbf{R}$, $\tilde{\mathbf{R}}^{(l)}$, $\mathbf{P}$	Matrices of hard selection, soft inclusion, and router probabilities.	p_{S}	Student next-token distribution (Appendix C.1).
$\text{sg}[\cdot]$	Stop-gradient operation.	I_{te}	Activation-based expert importance (Section 4.4).
$\boldsymbol{g}$	Objective gradient with respect to the static count vector (Appendix B.2).	$\mathcal{S}_{l,t}$, $\mathcal{O}_{l,t}$	Selected and reference expert sets (Section 4.4).
x_i, $x_{<i}$	Target token and preceding context within its block.	$\mathbf{R}_{\text{STE}}$	STE for hard routed selection, used for router and count gradients.
δ_{0j}	Kronecker delta: one when $j=0$, zero otherwise.		

B METHODS IMPLEMENTATION DETAILS

B.1 REAL-MODEL UTILIZATION NUMBERS

Equation 1 rounds $u_{\text{target}}(E_r+E_s)-E_s$ to an integer, restricting realized utilization to multiples of $1/E_t$. A requested utilization is exact when it is such a multiple and is otherwise rounded to the nearest realizable value. Llama-3-8B has $E_t = 112$, making all three reported tiers exact. Llama-2-7B and Qwen2.5-7B have 86 and 148 experts, respectively. They realize one and two tiers exactly, with the remaining tiers within 0.6 percentage points of the requested utilization.

Table 5 reports the geometry, realized utilization, and routed budget $B = L\bar{E}$ for every model and tier. Within each model and tier, all allocation methods hold B fixed and activate the same total number of FFN neurons for every token. The per-layer average is $(\bar{E} + E_s)m$, or 2,816 neurons on Llama-2-7B at the nominal 25% tier. Exact agreement with the requested utilization is unnecessary for comparing allocation methods, provided their realized budgets are matched.

Model	L	E_t	d_{ffn}	Nominal u	$\bar{E}$	Realized u	B
Llama-2-7B	32	86	11,008	12.5%	10	12.79%	320
				25%	21	25.58%	672
				50%	42	50.00%	1,344
Llama-3-8B	32	112	14,336	12.5%	13	12.50%	416
				25%	27	25.00%	864
				50%	55	50.00%	1,760
Qwen2.5-7B	28	148	18,944	12.5%	18	12.84%	504
				25%	36	25.00%	1,008
				50%	73	50.00%	2,044

Table 5: Expert geometry, the uniform routed count derived from each requested utilization, the utilization it realizes, and the conserved routed budget. All models use $m = 128$ neurons per expert, one shared expert per layer, and $E_{\min} = 1$.

B.2 Enforcing Bounds

Utilization Update Stability. Static training constrains the continuous counts before and after each optimizer step. Before the step, we center the count gradient, $\boldsymbol{g} \leftarrow \boldsymbol{g} - (\mathbf{1}^\top \boldsymbol{g}/L)\mathbf{1}$, so that the gradient is tangent to the budget hyperplane $\{\mathbf{1}^\top \boldsymbol{\epsilon} = B\}$. AdamW rescales each coordinate separately, so the resulting update can still change the sum. After the step, we therefore restore the sum by projecting:

$$\boldsymbol{\epsilon} \leftarrow \boldsymbol{\epsilon} - \frac{\mathbf{1}^\top \boldsymbol{\epsilon} - B}{L}\mathbf{1},$$

which removes drift introduced by adaptive scaling, momentum, and finite precision.

Enforcing Count Bounds. Centering the count gradient and restoring the sum does not enforce $\frac{1}{2} \le \epsilon_l \le E_r - \frac{1}{2}$. To mitigate this, violating coordinates are clamped and the projection over the remaining free coordinates is repeated. The bounded fixed-sum set is acceptable when $L/2 \le B \le L(E_r - \frac{1}{2})$. This formula holds for every budget in Table 5.

Integer Repair. Flooring the continuous counts removes $B - \sum_l \lfloor \epsilon_l \rfloor$ units. These units are assigned back to the layers with the largest fractional parts. This process restores $\sum_l E_l = B$ and gives $|E_l - \epsilon_l| < 1$ before bound repair. Integer clipping can change the sum, so a seeded repair adds or removes one expert at an eligible layer until the budget is restored. Such a layer exists whenever $LE_{\min} \le B \le LE_r$. Continuous counts persist between steps, allowing sub-integer updates to accumulate before the hard schedule changes.

C Experimental Protocol

C.1 Perplexity

Protocol. We measure WikiText-2 test perplexity (Merity et al., 2016) using each model's tokenizer and complete, nonoverlapping blocks of 2,048 tokens:

$$\text{PPL} = \exp\left(-\frac{1}{\sum_b |b|}\sum_b \sum_{i \in b} \log p_{\text{S}}(x_i \mid x_{<i})\right),$$

Here b contains the scored next-token targets of a complete block, i indexes these targets, and $x_{<i}$ contains context only from that block. Preprocessing retains empty rows and special tokens, and the trailing partial block is discarded. The Llama-2-7B stream contains 341,469 tokens. Its 166

complete blocks contain $339{,}968$ input tokens and $339{,}802$ scored targets. Every final perplexity in this paper is the corresponding run's end-of-training evaluation under this protocol, while training curves and step-0 values are labeled where they appear.

Data Separation. All reported perplexities use the WikiText-2 test split.

C.2 Benchmark Evaluation

Benchmarks. Evaluation uses the harness provided by (Gao et al., 2024) with the settings in Table 6. The six-task mean weights tasks equally, and reported cross-model gains average the three model-specific gains equally. Averages and differences are computed from unrounded scores, then rounded for display.

Task	Shots	Metric	Examples	Split
BoolQ (Clark et al., 2019)	32	accuracy	3,270	validation
SciQ (Welbl et al., 2017)	0	accuracy	1,000	test
PIQA (Bisk et al., 2019)	0	accuracy	1,838	validation
WinoGrande (Sakaguchi et al., 2019)	5	accuracy	1,267	validation
ARC-Challenge (Clark et al., 2018)	25	length-normalized accuracy	1,172	test
HellaSwag (Zellers et al., 2019)	10	length-normalized accuracy	10,042	validation

Table 6: Evaluation settings, identical for every model and allocation method.

Data Separation. Training and calibration use Dolmino (OLMo et al., 2025). Evaluation uses the WikiText-2 test split and the benchmarks in Table 6.

C.3 Training Configurations

Learned Base Each learned-base run processes $3{,}500 \times 64 \times 2{,}048 = 458{,}752{,}000$ Dolmino tokens from one tokenized cache. Training uses bfloat16 on four H100 GPUs with sharded dense weights and a shared data seed. The runs in Table 1 use initialization seed ***B***, and Table 8 lists the others. Partition affinities and router weights use a peak learning rate of 5×10^{-4} and weight decay 10^{-4}. Allocator parameters use a separate optimizer group with zero weight decay, since decay on continuous counts would bias them toward zero routed experts.

Analytic Base. The base for the analytic approach is CMoE (Pei et al., 2026). Specifically, balanced k-means (Lloyd, 1982; Malinen & Fränti, 2014) with a linear-assignment step is used (Jonker & Volgenant, 1987) on $16{,}384$ Dolmino calibration tokens. The constructed Llama-2-7B model has perplexity 735.85. Following expert assignment, weight recovery trains LoRA adapters (Hu et al., 2021) of rank 8, scaling $\alpha = 32$, and dropout 0.1 on all attention and MLP projections using $2{,}048$ samples of $2{,}048$ tokens. The final perplexity is 21.86.

The allocator then trains for $1{,}024$ steps over the same token budget, with recovered weights, partition, and router frozen. The analytic uniform baseline is the recovered checkpoint before allocator training and therefore has zero allocator-training steps. A comprehensive list of training parameters can be seen in Table 7.

C.4 Existing Method Implementations

The analytic base uses CMoE's published partition and router construction (Pei et al., 2026), unmodified. The learned base, however, reimplements DOT-MoE due to a lack of available public

	Learned base	Analytic base
Trained tensors	$\mathbf{A}$, $\mathbf{W}_r$, allocator	allocator only
Frozen	pretrained FFN weights	weights, partition, router
Steps / tokens	3,500 / 458.8M	1,024 / 4.19M
Preceding stage	–	LoRA recovery, 4.19M tokens
$\mathbf{A}$, $\mathbf{W}_r$ peak rate	5×10^{-4}	–
Static ϵ peak rate	$5 \times 10^{-3}\bar{E}/10$	$0.875\ \bar{E}/512$
Dynamic network peak rate	$5 \times 10^{-5}\bar{E}/10$	$5 \times 10^{-5}\bar{E}/10$
Inclusion temperature T	1.5 static, 1.0 dynamic	1.0
Transport temperature τ	$1.0 \rightarrow 0.1$ over warmup, then flat	–
Optimizer	AdamW	AdamW
Rate schedule	warmup over the first fifth, cosine to zero	as learned
Evaluation / checkpoint cadence	every 10M / 50M tokens	end of run

Table 7: Training configurations. The analytic static rate is set from the step count, $0.875\bar{E}$ spread over half the run, rather than from the learned base's constant.

code at the time of writing. One modification was made to the original DOT-MoE work which was the inclusion of a shared expert for the purposes of training stability.

D Run-to-Run Variation

Dense-to-MoE conversion is sensitive to initialization meaning identical conversion procedures can yield different perplexities and benchmark scores across seeds. Estimating this variation across models, budgets, and allocation methods requires many repeated experiments, whose computational cost is prohibitive. Despite this cost, we did run a small number of different seeds in order to provide some indication of this variation. Table 8 reports representative statistics from the learned Llama-2-7B base training run at 25% FFN utilization.

Allocation	Initialization	Step-0 PPL	Final PPL	Six-task avg. (%)
Dynamic	seed *C*	1115.01	10.72	62.2
	seed *A*	1279.96	10.59	61.7
	seed *B* (Table 1)	1365.54	10.57	62.8

Table 8: Seed variation on the learned Llama-2-7B base at 25% utilization ($B = 672$). Metrics follow Table 1.

While this table only presents a small number of seeds run on a single model with a single allocation approach, it still shows that between runs down-stream accuracy can differ significantly. We therefore match initialization seeds across the uniform, static, and dynamic arms in Table 1 to control initialization differences within each comparison.

E Additional Layer Utilization Heterogeneity Experiments and Information

E.1 Detailed Experiment Information

Figure 3 uses learned Llama-2-7B conversion at 25% target utilization ($B = 672$), seed *B*, and $3{,}500$ steps ($458{,}752{,}000$ tokens), differing only in allocation. Panel (a) evaluates WikiText-2 test perplexity every 10M tokens, giving 46 evaluations after initialization. Panel (b) applies a centered 61-step rolling median to per-step mean counts. Panel (c) shows 16 final-step schedules. Between steps $2{,}501$–$3{,}000$ and $3{,}001$–$3{,}500$, no layer's mean count changes by more than 0.63 experts.

E.2 Per-Layer Profiles Across Models

Despite the differences in methodology, static and dynamic allocation both assign more capacity to late layers on all three models (Figure 5). In this figure *Static* is the final schedule of the static run in Table 1 and *Dynamic* is the per-layer mean over the 4,096 tokens of the final batch.

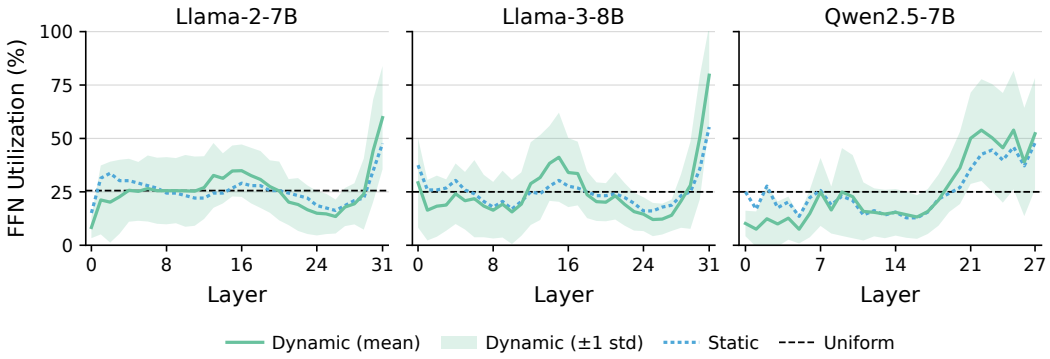


Figure 5: Learned per-layer FFN utilization at 25% utilization on the three main-body models.

F Inclusion Temperature Selection

Table 9 varies inclusion temperature in matched static Llama-2-7B runs initialized with 10 routed experts per layer. This data shows the temperature value can have a significant effect on the overall perplexity. This, in turn, means that selecting a suitable temperature value is imperative to the overall performance of the algorithm.

Inclusion temperature	Final PPL
Uniform control	16.30
$T = 0.25$	16.57
$T = 1$	16.15
$T = 1.5$	15.89

Table 9: Inclusion temperature for learned static allocation on Llama-2-7B at $\bar{E} = 10$ ($B = 320$).

Temperature Selection. For an expert of rank $r = \mathrm{rank}_l(t, e)$, Equation 2 defines the soft inclusion as

$$\tilde{R}_{te}^{(l)} = \sigma\left(\frac{\epsilon_l - r - \frac{1}{2}}{T}\right).$$

Its sensitivity to ϵ_l peaks at the selection boundary $r = \epsilon_l - \frac{1}{2}$ and decreases with distance from this boundary.

To illustrate how T controls the range of sensitive ranks, consider soft inclusions between 0.05 and 0.95. Since $\sigma(-\log 19) = 0.05$ and $\sigma(\log 19) = 0.95$, we have

$$0.05 \le \tilde{R}_{te}^{(l)} \le 0.95 \quad \Longleftrightarrow \quad \left|r - (\epsilon_l - \tfrac{1}{2})\right| \le T \log 19.$$

This interval extends $T \log 19$ ranks on either side of the selection boundary. For the initialization $\epsilon_l = 10$ in Table 9, the boundary lies at rank 9.5. At $T = 1.5$, $T \log 19 \approx 4.42$, so the interval includes ranks 6 through 13. At $T = 0.25$, $T \log 19 \approx 0.74$, leaving only ranks 9 and 10 within the interval. The interval moves with ϵ_l, so ranks outside it at initialization can enter the transition region during training. We select T empirically, with $T = 1.5$ giving the lowest perplexity among the values tested. There may be superior values for T, however, we leave searching for such values for future work.