STUCK IN COLLAPSE: UNDERSTANDING ROUTING COLLAPSE IN SPARSE MIXTURE-OF-EXPERTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-Experts is a key architecture for scaling model capacity with limited computation. However, *routing collapse*, where token assignments concentrate on a small subset of experts, remains a long-standing failure mode that degrades both model performance and computational efficiency. To understand how routing collapse emerges and persists, we conduct a controlled pretraining study across two model scales and eight model variants. We find that routing collapse is not a gradual accumulation of imbalance, but a failure to correct routing perturbations, with correction dynamics that differ across routing operators. Tracing the underlying optimization mechanism, we observe that these perturbations coincide with rapid early reorganization of hidden representations, which can act as a common input-side source. Under sparse routing, selection-dependent asymmetry in router gradients can fail to correct the resulting bias. For unselected experts, these gradients either vanish entirely or lose expert-specific information and share a common scalar component. To validate this mechanism, we introduce a targeted router gradient intervention that restores expert-specific information for unselected experts while preserving sparse forward computation. It substantially alleviates routing imbalance without an auxiliary loss, providing evidence that router gradient dynamics for unselected experts play a key role in sustaining routing collapse.

1 INTRODUCTION

Mixture-of-Experts (MoE) (Jacobs et al., 1991; Eigen et al., 2013) is a prominent sparse architecture that substantially expands model capacity while limiting computation by activating only a subset of experts for each token. This property has led to its widespread adoption in recent large-scale frontier models (Team et al., 2026; Yang et al., 2025a; Comanici et al., 2025). However, sparse computation inherently partitions experts into selected and unselected sets for each token, making model training and computational efficiency heavily dependent on the router that determines this selection. When routing becomes concentrated on a subset of experts, infrequently selected experts participate less in forward computation and receive fewer parameter updates. Concurrently, the resulting workload imbalance across experts can create hardware bottlenecks and severely degrade system efficiency in parallel computing environments (Yan et al., 2026; Rajbhandari et al., 2022; Hwang et al., 2023).

Routing collapse refers to a common failure mode in MoE models where the router repeatedly assigns most tokens to a small subset of experts, leaving the remaining experts rarely selected. This phenomenon has been consistently reported from early studies (Shazeer et al., 2017; Lepikhin et al., 2020; Lewis et al., 2021) to recent work (Lv et al., 2026; Do et al., 2025; Li et al., 2026). To mitigate this issue, many approaches introduce auxiliary losses that encourage balanced token assignment across experts (Yang et al., 2025b; Qiu et al., 2025; Omi et al., 2025), and such explicit balancing constraints have become standard practice in MoE training. However, while these approaches can effectively alleviate routing collapse, they do not explain why it arises, leaving its underlying mechanism poorly understood. Without understanding these dynamics, existing solutions offer limited principled guidance for designing routing mechanisms that address the causes of collapse.

Motivated by this gap, prior work has proposed two main hypotheses for routing collapse. One is a positive-feedback mechanism, where experts favored early by chance receive more parameter updates and improve more rapidly, further increasing the router's preference for them and amplifying routing imbalance (Shazeer et al., 2017; Qiu et al., 2025). The other is a structural explanation, where routing pulls token representations into a subspace spanned by expert-specific routing directions, thereby collapsing representation diversity (Chi et al., 2022; Do et al., 2023; 2025). However, while

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.

these accounts provide plausible explanations, it remains empirically unclear how these dynamics actually manifest during training and ultimately lead to collapse.

In this work, we systematically investigate how routing collapse emerges during training and uncover its underlying mechanism. To this end, we conduct a controlled study while strictly holding data, architecture, and optimization settings fixed. At two scales, 0.68B and 1.92B, we pretrain eight model variants from scratch on 100B tokens. Across the MoE variants, we vary the routing operator, which determines the selected expert set and routing probabilities, as well as the use of an auxiliary loss, with a Dense model matched in active parameters per token. We identify two key empirical findings: (1) *Perturbation and divergent trajectories:* Rather than gradually intensifying throughout training, routing imbalance arises from sharp perturbations, most prominently early in training, after which models diverge into distinct trajectories, including persistent collapse, rapid rebound, and gradual recovery toward high expert utilization. (2) *Operator dependency:* Routing collapse is strongly operator-dependent, with its severity, layer-wise pattern, and response to auxiliary loss differing qualitatively across routing operators. Sparse routing can exhibit severe collapse, whereas dense routing, which routes each token to all experts, remains largely free of imbalance.

Guided by these observations, we trace the optimization dynamics underlying the emergence and persistence of routing collapse, examining how routing imbalance arises and how routing operators shape the subsequent router response. We find that early in training, the geometry of hidden representations shifts rapidly across all models, coinciding with substantial fluctuations in routing probabilities. Crucially, the router's ability to adapt to these early shifts depends on the gradient signals governing updates to the router weights. Under sparse routing, expert selection makes these signals fundamentally selection-dependent. Specifically, depending on the operator, the corresponding gradients for unselected experts either vanish entirely or lose expert-specific information and share a common scalar component. Consequently, once an representation shift biases routing toward a subset of experts, the router gradients associated with underselected experts may fail to provide an effective corrective signal for counteracting this bias, leaving the resulting imbalance uncorrected and thereby sustaining routing collapse.

To validate this insight, we introduce a targeted router gradient intervention that restores expert-specific information for unselected experts while preserving sparse forward computation. This intervention substantially alleviates routing imbalance without relying on an explicit balancing objective. By restoring this information to the router gradient pathway for unselected experts, it enables continued router adaptation to emerging imbalance, providing empirical evidence that these gradient dynamics play a key role in sustaining routing collapse.

2 PRELIMINARIES

Mixture-of-Experts. In Transformer-based MoE architectures, the feed-forward network (FFN) in each block is replaced by an MoE layer, which consists of N expert networks $\{E_i\}_{i=1}^{N}$ and a router R. Let $\boldsymbol{h} \in \mathbb{R}^{D_{\text{hidden}}}$ denote the hidden representation of a token produced by the self-attention sublayer, which serves as the input to both the router and the experts. Given $\boldsymbol{h}$, the router $R(\boldsymbol{h}; \boldsymbol{\theta}_R)$ determines a set of selected experts $\mathcal{S} \subseteq [N]$ and a softmax-normalized routing probability vector $\boldsymbol{p} = [p_1, \ldots, p_N]^\top$ satisfying $\sum_{i=1}^{N} p_i = 1$. Each selected expert $i \in \mathcal{S}$ produces $\boldsymbol{y}_i = E_i(\boldsymbol{h}; \boldsymbol{\theta}_{E_i}) \in \mathbb{R}^{D_{\text{hidden}}}$, and the final MoE output is computed as a weighted sum over the selected experts:

$$\boldsymbol{y} = \sum_{i \in \mathcal{S}} p_i \boldsymbol{y}_i. \tag{1}$$

Here, $\boldsymbol{\theta}_R$ and $\boldsymbol{\theta}_{E_i}$ denote the router and expert parameters, respectively. The selection of $\mathcal{S}$ and the computation of $\boldsymbol{p}$ depend on the routing operator and are specified in Section 3.1.

Auxiliary Loss. MoE models are typically trained with auxiliary losses in addition to the language modeling loss $\mathcal{L}_{\text{LM}}$ to encourage balanced expert utilization. Although numerous auxiliary losses have been proposed (Guo et al., 2026; Zoph et al., 2022; Li et al., 2025), the standard load balancing loss (Fedus et al., 2022) remains the de facto choice for MoE training:

$$\mathcal{L}_{\text{aux}} = N \sum_{i=1}^{N} f_i \bar{p}_i, \tag{2}$$

where f_i denotes the fraction of tokens dispatched to expert i, and $\bar{p}_i$ denotes the average routing probability for expert i over the batch. When the auxiliary loss is used, the training objective is $\mathcal{L} = \mathcal{L}_{\text{LM}} + \lambda \mathcal{L}_{\text{aux}}$, where λ is commonly set to 0.01 (Lv et al., 2026; Panda et al., 2026).

MODEL	# Params.	# Tokens	# Layers	N	D_{hidden}	D_{inter}	# Heads
0.68B / 0.34B							
MoE	0.68B/0.34B	100B	12	8	768	2048	12
Dense	0.34B	100B	12	-	768	4096	12
1.92B / 0.51B							
MoE	1.92B/0.51B	100B	12	16	1024	2730	16
Dense	0.51B	100B	12	-	1024	5460	16

Table 1: Model sizes and architectures of the evaluated models. For sparse routing MoE models, model sizes are reported as Total/Active parameters. Dense denotes the corresponding dense counterpart with the same active parameter budget as the sparse MoE model.

3 EXPERIMENTAL SETUP

To systematically investigate the mechanisms underlying routing collapse, we establish a controlled experimental setup. We train all model variants from scratch under identical training conditions, varying only the routing operator and the use of auxiliary loss in MoE models.

3.1 MODEL ARCHITECTURES AND ROUTING OPERATORS

We adopt a decoder-only Transformer where each FFN is replaced by an MoE layer consisting of SwiGLU experts with intermediate dimension D_{inter} (Liu et al., 2024; Yang et al., 2025a). For routing, we use a standard linear router (Fedus et al., 2022; Dai et al., 2024). Let $\boldsymbol{W}_R = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N]^\top \in \mathbb{R}^{N \times D_{\text{hidden}}}$ denote the router weight matrix, where $\boldsymbol{w}_i$ is the routing vector for expert i. Given $\boldsymbol{h}$, the router computes the logit $z_i = \boldsymbol{w}_i^\top \boldsymbol{h}$ for each expert, yielding $\boldsymbol{z} = [z_1, \ldots, z_N]^\top$.

We consider three representative routing operators: two sparse variants, *Softmax-TopK* (Dai et al., 2024; Lepikhin et al., 2020) and *TopK-Softmax* (Jin et al., 2025; Jiang et al., 2024), and dense *Softmax* (Li et al., 2025; Dou et al., 2023). Softmax-TopK first computes softmax probabilities from the logits over all experts and then selects the top-K experts according to these probabilities, whereas TopK-Softmax first selects the top-K experts directly from the logits and normalizes only within the selected set. Softmax uses all N experts without selection. Formally,

$$\text{Softmax-TopK:} \quad \mathcal{S} = \mathrm{TopK}(\boldsymbol{p}, K), \qquad p_i = \mathrm{softmax}(\boldsymbol{z})_i, \tag{3}$$

$$\text{TopK-Softmax:} \quad \mathcal{S} = \mathrm{TopK}(\boldsymbol{z}, K), \qquad p_i = \begin{cases} \frac{\exp(z_i)}{\sum_{j \in \mathcal{S}} \exp(z_j)}, & i \in \mathcal{S}, \\ 0, & i \notin \mathcal{S}, \end{cases} \tag{4}$$

$$\text{Softmax:} \quad \mathcal{S} = [N], \qquad p_i = \mathrm{softmax}(\boldsymbol{z})_i. \tag{5}$$

Thus, over the selected expert set $\mathcal{S}$, TopK-Softmax ensures $\sum_{i \in \mathcal{S}} p_i = 1$, whereas Softmax-TopK generally has $\sum_{i \in \mathcal{S}} p_i < 1$. We fix $K = 2$ for both sparse routing operators across all experiments.

3.2 TRAINING SETTINGS

Model configurations are given in Table 1. We consider two model scales, each paired with a Dense model matched in active parameter count. Since Softmax activates all N experts, its active and total parameter counts are identical. Throughout the paper, MoE models are denoted by their routing operator, with the subscript "aux" indicating the use of $\mathcal{L}_{\text{aux}}$. We omit this loss for Softmax because it is constant by construction.[1] At each model scale, we report eight variants: Dense, Softmax, Softmax-TopK, Softmax-TopK$_{\text{aux}}$, TopK-Softmax, TopK-Softmax$_{\text{aux}}$, Ours, and Ours$_{\text{aux}}$, where Ours denotes the router gradient intervention that will be introduced in Section 6. All variants are pretrained from scratch on 100B tokens from the FineWeb dataset (Penedo et al., 2024), with all other settings held fixed across variants. Additional details are provided in Section C.1.

3.3 EVALUATION METRIC

To quantify routing imbalance, prior MoE studies commonly use MaxVio (Wang et al., 2024), which considers the single most overloaded expert. However, this worst-case statistic fails to capture how routing mass is distributed across the full set of experts. To measure overall expert utilization, we use the entropy-based effective number of experts, $N_{\text{eff}} \triangleq \exp(H(\bar{\boldsymbol{p}})) = \exp(-\sum_{i=1}^{N} \bar{p}_i \log \bar{p}_i)$, computed separately for each MoE layer. By expressing the entropy, $H(\cdot)$, of the aggregate routing distribution on an expert-count scale ($1 \le N_{\text{eff}} \le N$), N_{eff} provides an intuitive measure of how many experts are effectively utilized. A more detailed discussion is provided in Section C.2.

[1] Under Softmax, $f_i = 1$ for all experts, making $\mathcal{L}_{\text{aux}}$ constant ($\mathcal{L}_{\text{aux}} = N$) regardless of $\bar{p}_i$.

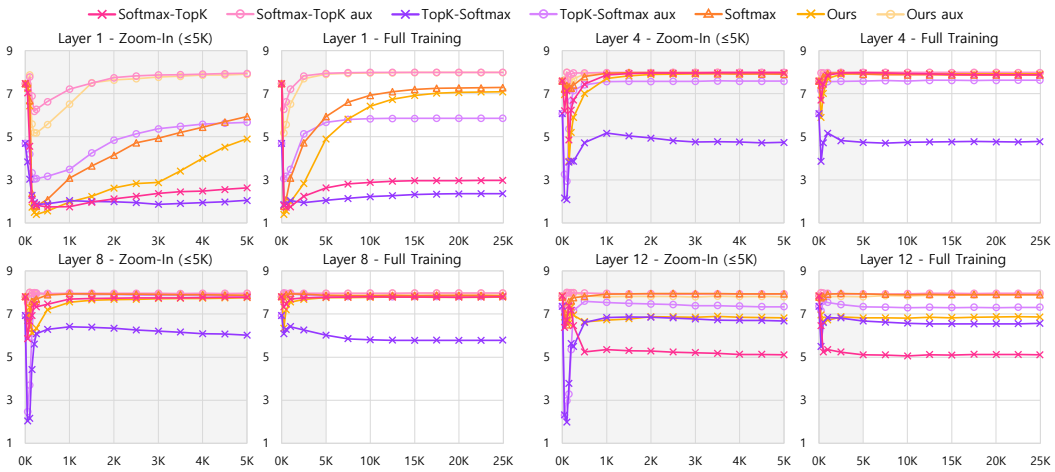

Figure 1: Evolution of the effective number of experts for 0.68B-scale models during training. We show Layers 1, 4, 8, and 12 spanning the model depth. Left panels zoom into the first 5K steps, while right panels show the full trajectory, with the same interval shaded in gray.

	0.68B / 0.34B		1.92B / 0.51B	
METHODS	$N_{\text{eff}}^{\text{avg}}$	$N_{\text{eff}}^{\text{worst}}$	$N_{\text{eff}}^{\text{avg}}$	$N_{\text{eff}}^{\text{worst}}$
Softmax-TopK	7.03	2.96	13.79	6.58
TopK-Softmax	5.48	2.34	8.09	2.08
Softmax	7.87	7.25	15.56	14.16
Ours	7.46	6.56	14.92	13.12
Softmax-TopK$_{\text{aux}}$	7.95	7.86	15.86	15.72
TopK-Softmax$_{\text{aux}}$	7.46	5.85	14.54	10.61
Ours$_{\text{aux}}$	7.94	7.79	15.81	15.56

Table 2: Model-level effective number of experts across model scales, reporting the layer average $N_{\text{eff}}^{\text{avg}}$ and the minimum across layers $N_{\text{eff}}^{\text{worst}}$.

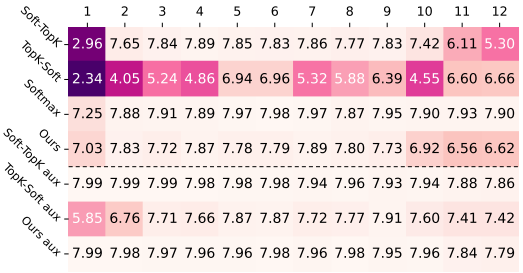

Figure 2: Layer-wise effective number of experts for the 0.68B-scale models. "Soft" abbreviates Softmax in the sparse routing operator labels.

4 ANATOMY OF ROUTING COLLAPSE

4.1 PERTURBATION AND DIVERGENCE IN ROUTING DYNAMICS

Figure 1 shows the evolution of layer-wise N_{eff} during training for the 0.68B models, while complete results across all layers and both model scales are provided in Section E.1. Although final expert utilization varies substantially across routing operators, auxiliary loss usage, and layers, a common pattern emerges very early in training. Within the first few hundred steps, N_{eff} often drops sharply or fluctuates substantially, after which the routing dynamics diverge into distinct trajectories.

These post-perturbation trajectories broadly fall into three patterns. In the first, the reduced N_{eff} fails to recover and stabilizes at a low or intermediate level for the remainder of training. In the second, N_{eff} rapidly rebounds from the initial perturbation and remains at a high level of expert utilization thereafter. In the third, N_{eff} recovers more gradually, steadily increasing over an extended period toward higher expert utilization. Routing collapse therefore does not appear to arise from a gradual accumulation of imbalance. Rather, routing preferences are abruptly perturbed, most prominently early in training, and the subsequent dynamics determine whether the resulting imbalance persists.

4.2 OPERATOR-DEPENDENT PATTERNS OF ROUTING COLLAPSE

Table 2 summarizes model-level expert utilization using the layer-wise average $N_{\text{eff}}^{\text{avg}}$ and the worst-layer value $N_{\text{eff}}^{\text{worst}}$, while Figure 2 shows layer-wise profiles for the 0.68B models. A clear distinction emerges between dense (Softmax) and sparse routing. Without explicit balancing constraints, both sparse routing operators exhibit severe expert underutilization in some layers, whereas dense Softmax routing maintains high expert utilization across layers. This contrast is consistent across both model scales. Importantly, routing collapse also differs substantially between the two sparse routing operators in its severity, layer-wise pattern, and response to auxiliary loss.

Softmax-TopK exhibits a depth-dependent pattern, maintaining well-balanced routing in intermediate layers while showing substantially lower N_{eff} in early and late layers, consistent with the

METHODS	ARC-E	WG	PIQA	SIQA	HS	PMQA	SciQ	Avg.↑	GFLOPs↓
0.68B / 0.34B									
Dense	45.24	52.09	65.07	37.82	34.87	42.80	68.40	49.47	0.556
Softmax-TopK	45.88	50.12	67.68	39.41	39.36	44.00	67.90	50.62	0.556
Softmax-TopK$_{\text{aux}}$	48.36	52.33	68.50	38.84	40.46	51.80	71.30	53.08	0.556
TopK-Softmax	46.42	51.07	68.61	38.33	39.70	49.20	69.90	51.89	0.556
TopK-Softmax$_{\text{aux}}$	46.63	52.80	69.53	38.18	39.82	48.40	68.60	52.00	0.556
Softmax	49.03	51.30	68.39	39.61	41.80	53.40	72.20	53.68	1.235
Ours	47.81	52.72	67.90	38.54	39.63	54.40	71.50	53.21	0.556
Ours$_{\text{aux}}$	47.98	51.22	68.55	38.54	40.24	51.80	71.60	52.85	0.556
1.92B / 0.51B									
Dense	48.48	50.67	67.63	38.69	38.44	54.80	71.80	52.93	0.867
Softmax-TopK	50.04	52.57	70.73	38.74	46.27	49.60	72.50	54.35	0.867
Softmax-TopK$_{\text{aux}}$	51.30	50.43	71.38	40.02	48.00	52.00	74.10	55.32	0.867
TopK-Softmax	51.64	51.78	71.71	39.15	46.38	49.20	74.30	54.88	0.867
TopK-Softmax$_{\text{aux}}$	50.97	53.67	70.84	41.20	47.68	50.40	75.20	55.71	0.867
Softmax	54.34	52.01	72.25	40.38	51.14	54.80	76.70	57.37	3.685
Ours	50.08	52.41	70.29	39.15	46.74	54.40	75.10	55.45	0.867
Ours$_{\text{aux}}$	51.64	51.46	71.60	39.76	47.95	54.20	73.20	55.69	0.867

Table 3: Comparison of zero-shot performance and forward computation across model variants, including accuracy on ARC-Easy (ARC-E), WinoGrande (WG), PIQA, SocialIQA (SIQA), HellaSwag (HS), PubMedQA (PMQA), and SciQ, with average accuracy and GFLOPs per token.

observation of Soboleva (2025). In contrast, TopK-Softmax exhibits low N_{eff} across multiple layers without a similarly localized depth pattern, resulting in a lower overall $N_{\text{eff}}^{\text{avg}}$. The two operators also respond differently to the same auxiliary balancing loss. For Softmax-TopK, the auxiliary loss largely eliminates routing imbalance, keeping N_{eff} close to N across all layers. For TopK-Softmax, overall expert utilization improves substantially, but several layers retain comparatively low N_{eff}, leaving noticeable residual imbalance.

4.3 RELATION BETWEEN ROUTING COLLAPSE AND DOWNSTREAM PERFORMANCE

We additionally compare zero-shot benchmark performance in Table 3 to examine how routing collapse relates to downstream performance. Despite severe collapse, sparse MoE models consistently outperform the parameter-matched Dense baseline in average accuracy, indicating that substantial routing imbalance does not erase the advantage of MoE. Within each sparse operator, however, alleviating collapse tends to improve downstream performance. Applying the auxiliary loss raises average accuracy for both operators at both scales, alongside substantially higher expert utilization.

Softmax maintains high expert utilization and the highest accuracy, but activates all experts per token. Given this different compute regime, we use Softmax not as a direct performance baseline, but as a reference showing that high expert utilization can arise without explicit balancing constraints, depending on the routing operator. Ours, the targeted router gradient intervention, preserves sparse forward computation while alleviating collapse. The reduced routing imbalance is accompanied by higher average accuracy at both scales, comparable to that obtained with the auxiliary loss. We introduce and analyze this intervention in Section 6.

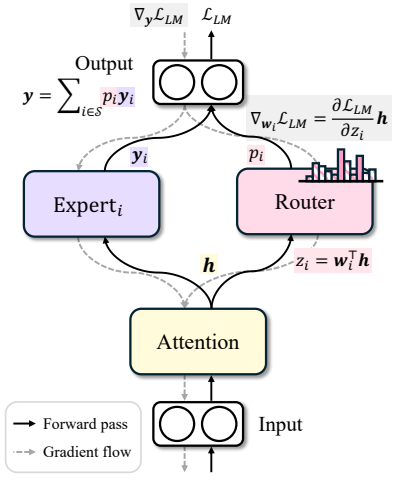

Figure 3: Routing computation and gradient flow in an MoE layer. Solid and dashed arrows denote forward and backward paths, respectively.

5 UNDERSTANDING ROUTING COLLAPSE

To mechanistically understand the routing perturbation and the operator-dependent patterns of collapse observed in Section 4, we analyze the optimization dynamics of the router. As illustrated in Figure 3, the hidden representation $\boldsymbol{h}$ produced by the attention sublayer serves as a shared input to the experts and the router. For expert i, the routing logit is given by $z_i = \boldsymbol{w}_i^\top \boldsymbol{h}$, so routing preferences are governed by the geometric relationship between $\boldsymbol{h}$ and the routing vectors $\{\boldsymbol{w}_i\}_{i=1}^N$. Under this parameterization, the gradient of $\mathcal{L}_{\text{LM}}$ with respect to $\boldsymbol{w}_i$ is

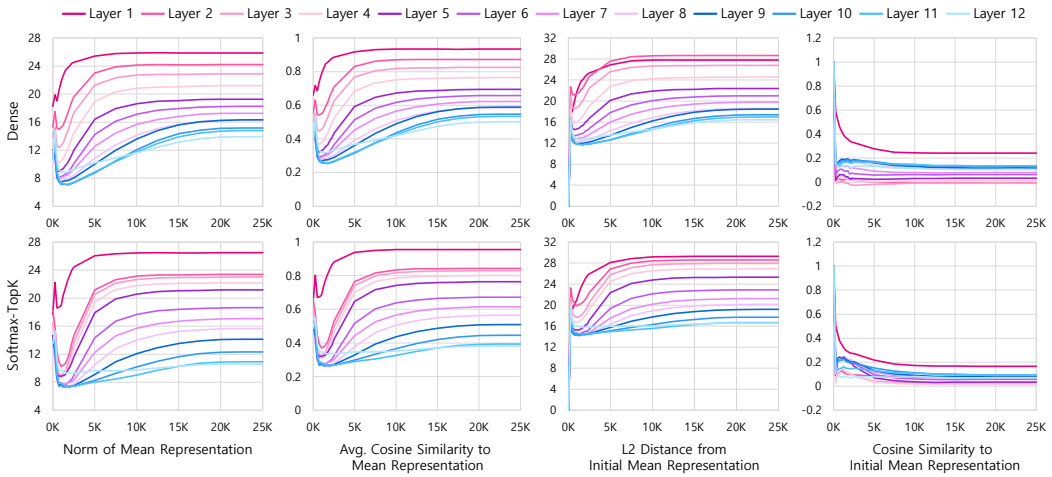

Figure 4: Evolution of $\boldsymbol{h}$ geometry during training for the 0.68B-scale Softmax-TopK and its Dense counterpart. The left two columns characterize representation geometry relative to the current mean, while the right two track the displacement and directional change of the mean from initialization.

$$\nabla_{\boldsymbol{w}_i}\mathcal{L}_{\text{LM}} = \left(\frac{\partial z_i}{\partial \boldsymbol{w}_i}\right)^{\top} \frac{\partial \mathcal{L}_{\text{LM}}}{\partial z_i} = \frac{\partial \mathcal{L}_{\text{LM}}}{\partial z_i}\boldsymbol{h}. \tag{6}$$

This decomposition separates the router gradient into two components: $\boldsymbol{h}$ determines the direction of the gradient induced by each token, while the scalar logit gradient $\frac{\partial \mathcal{L}_{\text{LM}}}{\partial z_i}$ determines its sign and magnitude. Because all MoE variants share the same router parameterization, routing operators differ only in the scalar logit gradient.

Building on this decomposition, we trace the optimization dynamics underlying the emergence and persistence of routing collapse in three steps. First, in Section 5.1, we examine how the geometry of $\boldsymbol{h}$ evolves and how these changes relate to the routing perturbation observed in Section 4.1. Next, in Section 5.2, we analyze the operator-dependent structure of $\frac{\partial \mathcal{L}_{\text{LM}}}{\partial z_i}$, focusing on the gradient signals for selected and unselected experts. Finally, in Section 5.3, we synthesize these observations into a mechanism for the emergence and persistence of routing collapse by tracing how the gradient structures shape the evolution of the routing vectors $\{\boldsymbol{w}_i\}_{i=1}^{N}$.

5.1 Early Representation Reorganization and Routing Perturbation

Figure 4 shows the evolution of the batch-level representation geometry of $\boldsymbol{h}$ for the 0.68B-scale Softmax-TopK and Dense models. Complete results across both model scales and all model variants are provided in Section E.2. The first two columns characterize the directional concentration of representations, while the last two track how the mean representation departs from its initialization.

Across all layers, the geometry of $\boldsymbol{h}$ changes sharply early in training and then stabilizes, coinciding with the early N_{eff} fluctuations. During this period, the directional concentration of the representations changes substantially, and the mean representation moves away from its initial state, as indicated by increasing displacement and decreasing cosine similarity to the initial mean. Importantly, this reorganization occurs across all variants, including the Dense model. This indicates that the reorganization does not require routing, but reflects a general feature of early training dynamics.

The pronounced early reorganization of $\boldsymbol{h}$ can perturb routing through its effect on the routing logits. Even with fixed routing vectors $\{\boldsymbol{w}_i\}_{i=1}^{N}$, changes in $\boldsymbol{h}$ alter the routing logits $z_i = \boldsymbol{w}_i^{\top}\boldsymbol{h}$ and can therefore change their relative ordering and the resulting routing preferences. Thus, this reorganization acts as a common input-side source of routing perturbation.

Takeaway 1. The geometry of $\boldsymbol{h}$ can reorganize substantially during training across all model variants. Since routing logits are computed from $\boldsymbol{h}$, such reorganization can directly alter routing preferences, even when the router itself remains unchanged.

5.2 Operator-Dependent Router Response

We next examine how each routing operator shapes the logit gradient from $\mathcal{L}_{\text{LM}}$, focusing on the distinction between selected and unselected experts. Detailed derivations are provided in Section A.

5.2.1 SOFTMAX: EXPERT-SPECIFIC GRADIENT SIGNAL

For a Softmax router with the upstream gradient from subsequent layers $\nabla_{\boldsymbol{y}}\mathcal{L}_{\mathrm{LM}}$, it becomes

$$\frac{\partial \mathcal{L}_{\mathrm{LM}}}{\partial z_i} = p_i(\boldsymbol{y}_i - \boldsymbol{y})^\top \nabla_{\boldsymbol{y}}\mathcal{L}_{\mathrm{LM}}. \tag{7}$$

The inner product in Equation (7) is the directional derivative of the loss when $\boldsymbol{y}$ is infinitesimally shifted toward $\boldsymbol{y}_i$. Thus, when $\frac{\partial \mathcal{L}_{\mathrm{LM}}}{\partial z_i} < 0$, increasing z_i increases expert i's contribution and locally decreases the loss by shifting $\boldsymbol{y}$ toward $\boldsymbol{y}_i$. Accordingly, each logit gradient retains expert-specific information about whether increasing the contribution of expert i would locally improve the loss, and this pathway remains available even when p_i is small.

5.2.2 SOFTMAX-TOPK: LOSS OF EXPERT-SPECIFIC SIGNAL FOR UNSELECTED EXPERTS

Under Softmax-TopK, routing probabilities are normalized over all experts, while only the top-K experts contribute to the MoE output. The logit gradient therefore depends on expert selection:

$$\frac{\partial \mathcal{L}_{\mathrm{LM}}}{\partial z_i} = \begin{cases} p_i(\boldsymbol{y}_i - \boldsymbol{y})^\top \nabla_{\boldsymbol{y}}\mathcal{L}_{\mathrm{LM}}, & i \in \mathcal{S}, \\ -p_i \boldsymbol{y}^\top \nabla_{\boldsymbol{y}}\mathcal{L}_{\mathrm{LM}}, & i \notin \mathcal{S}. \end{cases} \tag{8}$$

For $i \in \mathcal{S}$, the logit gradient has the same form as under Softmax and retains an expert-specific signal. For $i \notin \mathcal{S}$, however, the $\boldsymbol{y}_i$ term disappears, eliminating this expert-specific signal. Instead, the corresponding logit gradients for all unselected experts share the same scalar component $-\boldsymbol{y}^\top \nabla_{\boldsymbol{y}}\mathcal{L}_{\mathrm{LM}}$, which is determined by the current selected mixture $\boldsymbol{y}$ and the upstream gradient, and differ only by the scaling factor $p_i > 0$.

5.2.3 TOPK-SOFTMAX: ZERO GRADIENT FOR UNSELECTED EXPERTS

In TopK-Softmax, softmax normalization is applied only over the selected set, yielding

$$\frac{\partial \mathcal{L}_{\mathrm{LM}}}{\partial z_i} = \begin{cases} p_i(\boldsymbol{y}_i - \boldsymbol{y})^\top \nabla_{\boldsymbol{y}}\mathcal{L}_{\mathrm{LM}}, & i \in \mathcal{S}, \\ 0, & i \notin \mathcal{S}. \end{cases} \tag{9}$$

For $i \in \mathcal{S}$, the logit gradient has the same expert-specific signal as under Softmax. In contrast, for $i \notin \mathcal{S}$, the gradient from $\mathcal{L}_{\mathrm{LM}}$ vanishes entirely. Thus, once an expert falls outside the selected set for a given token, that token provides no direct gradient pathway from $\mathcal{L}_{\mathrm{LM}}$ for adjusting the expert's routing preference until it is selected again.

Takeaway 2. Sparse routing operators induce selection-dependent asymmetry in router logit gradients $\partial \mathcal{L}_{\mathrm{LM}}/\partial z_i$. For unselected experts, the corresponding logit gradients either lose expert-specific information and share a common scalar component, or vanish entirely.

5.3 FROM GRADIENT STRUCTURE TO ROUTING-VECTOR DYNAMICS

The preceding analyses identify two ingredients of the mechanism: representation reorganization can perturb routing preferences, while sparse selection induces asymmetric router logit gradients for selected and unselected experts. We now synthesize these observations into a mechanism for persistent routing collapse by tracing their consequences for routing vector dynamics.

Figure 5 shows routing vector trajectories across routing operators. We fit a two-dimensional PCA basis to the mean-centered $\{\boldsymbol{w}_i\}_{i=1}^N$ trajectories over training and project both $\{\boldsymbol{w}_i\}_{i=1}^N$ and $\boldsymbol{h}$ onto this basis without subtracting the PCA mean, preserving the feature-space origin. We visualize layers with relatively high PCA explained variance, where the projection captures a larger fraction of the original routing vector variation. Projection details and additional visualizations are provided in Sections C.3 and E.4.

5.3.1 SOFTMAX-TOPK: SHARED UPDATES AND COHERENT DRIFT

Under Softmax-TopK, the logit gradients of unselected experts share a scalar component determined by the selected expert set, with no expert-specific dependence beyond the positive scaling factor p_i. Consequently, their per-token routing vector updates are aligned along $\boldsymbol{h}$ with the same sign, which can accumulate across tokens into coherent routing vector motion (circles in Fig. 5).

The sign of this common update determines whether all unselected routing vectors are pulled toward or pushed away from $\boldsymbol{h}$. Either direction may fortuitously improve the routing competitiveness of an expert, but the direction is not chosen according to the correction required by that expert. The shared update can therefore exacerbate, rather than correct, geometry-induced routing imbalances.

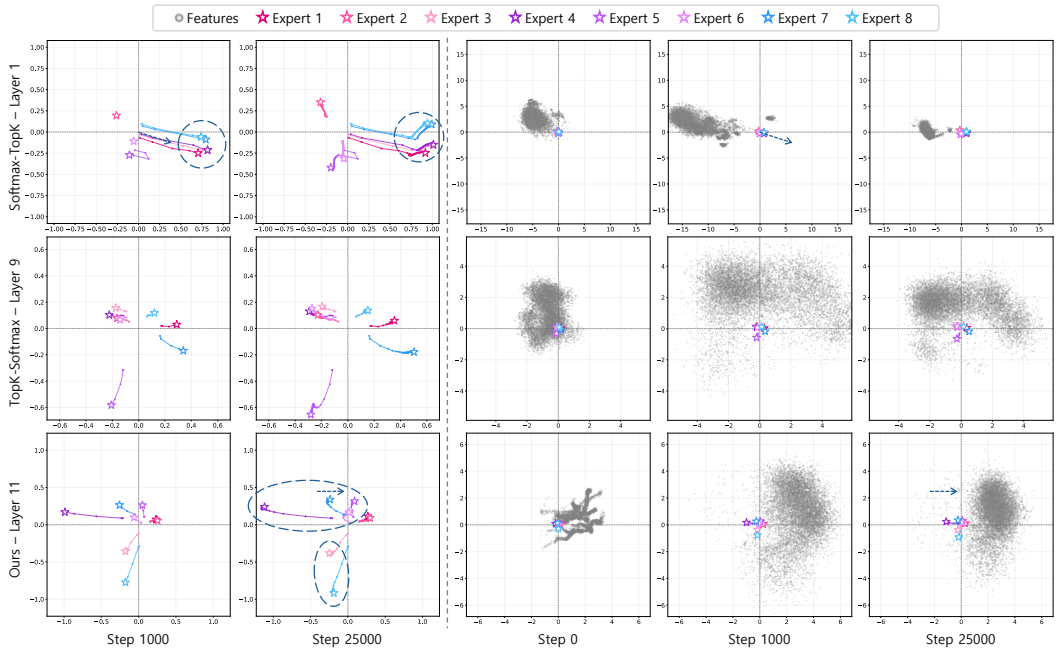

Figure 5: PCA projections of routing vector and hidden representation dynamics across routing operators in a shared, origin preserving coordinate system. Left columns zoom in on routing-vector trajectories, while right columns jointly show hidden representations and routing vectors over training. Dashed arrows and circles mark representative movements.

Figure 5 illustrates this adverse case in a layer with severe collapse. Several routing vectors exhibit coherent drift away from the dominant representation region. As these vectors become less competitive for tokens in that region, their re-entry into the top-K set becomes less likely. They are therefore unselected for more tokens and continue to follow the same non-expert specific update pathway. This feedback can reinforce the routing bias and sustain persistent routing collapse.

5.3.2 TopK-Softmax: Gradient Blockage and Geometric Lock-In

Under TopK-Softmax, the router logit gradient associated with an unselected expert is exactly zero. Each routing vector is therefore updated only by tokens that select its expert. Consistent with this isolated update structure, Figure 5 shows distinct routing vector trajectories and an early separation structure that persists throughout training.

This gradient blockage can turn a routing bias into geometric lock-in. As reorganized representations concentrate in regions dominated by a subset of experts, some routing vectors may fall outside the selection boundary of a dominant representation region. Once this happens, they stop receiving updates from the very tokens needed to restore their routing preference in that region. Updates from tokens elsewhere may still move these vectors, but they cannot recover the missing region-specific corrective signal. The resulting geometric misalignment can therefore persist, preserving the biased selection pattern and sustaining routing collapse.

In addition, this gradient blockage helps explain TopK-Softmax's limited response to auxiliary loss. Because $\bar{p}_i$ is computed from probabilities normalized within the selected set, tokens for which an expert is unselected contribute nothing to its auxiliary gradient. Balancing pressure can thus act only indirectly through selected logits, consistent with the residual imbalance in several layers.

5.3.3 Softmax: Persistent Corrective Pathway

Unlike sparse routing, dense Softmax applies no expert selection, so every expert retains an expert-specific router logit gradient for every token. Thus, even a less competitive expert continues to receive token-specific corrective signals.

Without selection-dependent asymmetry, these signals allow each routing vector to continuously adapt to the evolving representation geometry. Consistent with this, Figure 1 shows that Softmax in Layer 1 gradually recovers expert utilization after the initial drop in N_{eff}, with similar recovery observed in other layers and at the 1.92B scale (Section E.1). Given its similar representation reorganization (Section 5.1), this recovery is consistent with the view that whether collapse persists depends on the availability of corrective signals, which we directly test in Section 6.

Takeaway 3. Persistent routing collapse reflects a failure of router adaptation: once routing is biased, selection-dependent sparse updates may fail to provide the corrective signals needed to realign routing preferences with the evolving representation geometry.

6 Validating the Mechanism through Gradient Intervention

Section 5.3 suggests that persistent collapse arises when selection-dependent sparse gradients fail to provide effective corrective signals for unselected experts once routing is biased. To validate this mechanism, we introduce a targeted router gradient intervention that restores expert-specific information for unselected experts while preserving sparse forward computation. The resulting intervention substantially alters the routing dynamics and alleviates routing imbalance without an explicit balancing objective, providing empirical evidence that router gradient dynamics for unselected experts play a key role in sustaining routing collapse.

6.1 Restoring Expert-Specific Router Gradients

We instantiate the intervention with Softmax-TopK, whose nonzero pre-selection probabilities provide a gradient pathway for unselected experts. Recovering expert-specific information for unselected experts would require evaluating their responses for every token, resulting in dense MoE computation. We therefore approximate these responses at the cluster level. Since the routing logits $z_i = \boldsymbol{w}_i^\top \boldsymbol{h}$ partition the representation space by routing preference, we group tokens assigned to the same top-1 expert into a cluster c and use their mean representation, $\hat{\boldsymbol{h}}_c$, as a proxy. Evaluating all experts on $\hat{\boldsymbol{h}}_c$ yields counterfactual responses $\hat{\boldsymbol{y}}_{i,c} = E_i(\hat{\boldsymbol{h}}_c; \theta_{E_i})$, forming an $N \times N$ response table with limited additional computation. For each token in c, we construct $\hat{\boldsymbol{y}} = \sum_{i \in \mathcal{S}} p_i \boldsymbol{y}_i + \sum_{i \notin \mathcal{S}} p_i \hat{\boldsymbol{y}}_{i,c}$ and recompute the router logit gradient vector as

$$\frac{\partial \mathcal{L}_{\text{LM}}}{\partial z_i} = \begin{cases} p_i(\boldsymbol{y}_i - \hat{\boldsymbol{y}})^\top \nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}}, & i \in \mathcal{S}, \\ p_i(\hat{\boldsymbol{y}}_{i,c} - \hat{\boldsymbol{y}})^\top \nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}}, & i \notin \mathcal{S}. \end{cases} \tag{10}$$

Using $\hat{\boldsymbol{y}}$ as a common reference gives each unselected expert an expert-specific corrective signal through its counterfactual response. The same reference is applied to selected experts to retain the zero-sum structure of router logit gradients, as further analyzed in Section D.3. Importantly, the actual MoE forward output remains $\boldsymbol{y} = \sum_{i \in \mathcal{S}} p_i \boldsymbol{y}_i$, while $\hat{\boldsymbol{y}}$ is used only for router gradients. We detach $\hat{\boldsymbol{y}}_{i,c}$ so that the counterfactual branch introduces no additional expert parameter updates.

6.2 Empirical Validation of Router Adaptation

Geometry Adaptation. For Ours, Figure 5 visualizes routing vector trajectories relative to the evolving representation geometry. Early in training, routing vectors follow distinct trajectories. Later, however, several vectors appear to move toward regions where their experts are unselected, indicating continued adaptation beyond their currently selected token regions.

Gradual Recovery. In Layer 1, Figure 1 shows that Ours exhibits a sustained, gradual recovery in N_{eff} after the early drop. Unlike the sparse baselines, this pattern resembles the gradual recovery of dense Softmax. The continued increase in N_{eff} suggests that initially underselected experts progressively regain routing mass, correcting the early imbalance over training.

Model-Level Recovery. As shown in Table 2, across both model scales, the intervention substantially improves average and worst-layer expert utilization relative to the sparse baselines, even without an explicit balancing constraint. The layer-wise profiles in Figure 2 further show that Ours consistently mitigates the severe layer-wise collapse observed in the sparse baselines.

7 Conclusion

We systematically investigated how routing collapse emerges and persists in MoE models across model scales and routing operators. Our results suggest that reorganization of hidden representations, most pronounced early in training, acts as an input-side source of routing perturbation, while sparse routing provides limited corrective signals for underselected experts, allowing the resulting bias to persist. Restoring expert-specific information to the router gradients of unselected experts substantially alleviates routing imbalance without an auxiliary loss and promotes continued routing vector adaptation and expert utilization recovery. Together, these results identify router gradient dynamics associated with unselected experts as a key mechanism sustaining routing collapse. However, one limitation of our study is that the observed depth-dependent collapse pattern of Softmax-TopK remains unexplained. Identifying the factors underlying these layer-wise differences in collapse dynamics remains an important direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools to discuss and refine the conceptual framing and hypotheses, provide feedback on experimental methodology and analysis, and assist with the interpretation of results. Additionally, we used generative AI tools for brainstorming and for drafting and editing portions of the manuscript to improve clarity and readability. All research decisions, experimental design, implementation, analyses, and conclusions were made, conducted, or verified by the authors, and all AI-assisted content was reviewed by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the model architectures, routing operators, training setup, and evaluation protocol in Section 3, with full training configurations in Section C. Detailed router-gradient derivations are provided in Section A, and the implementation of our proposed gradient intervention, including PyTorch-style pseudocode, is provided in Section D. Additional experimental analyses and results are provided in Section E.

REFERENCES

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

Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, Xian-Ling Mao, Heyan Huang, and Furu Wei. On the representation collapse of sparse mixture of experts. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=mWaYC6CZf5.

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.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Damai Dai, Chengqi Deng, Chenggang Zhao, R.X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y.K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1280–1297, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.70. URL https://aclanthology.org/2024.acl-long.70/.

Larry Dial. Mixture of experts quantile balancing: Validated at 32b-a5b (1e22 flops) scale. https://www.openathena.ai/blog/quantile-balancing/, apr 2026. Open Athena Blog.

Giang Do, Hung Le, and Truyen Tran. SimSMoE: Toward efficient training mixture of experts via solving representational collapse. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 2012–2025, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.107. URL https://aclanthology.org/2025.findings-naacl.107/.

Truong Giang Do, Le Khiem, Quang Pham, Trungtin Nguyen, Thanh-Nam Doan, Binh Nguyen, Chenghao Liu, Savitha Ramasamy, Xiaoli Li, and Steven Hoi. Hyperrouter: Towards efficient training and inference of sparse mixture of experts. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 5754–5765, 2023.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, et al. Loramoe: Alleviate world knowledge forgetting in large language models via moe-style plugin. *arXiv preprint arXiv:2312.09979*, 2023.

Dheeru Dua, Shruti Bhosale, Vedanuj Goswami, James Cross, Mike Lewis, and Angela Fan. Tricks for training sparse translation models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3340–3345, 2022.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013.

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`.

Sam Gross, Marc'Aurelio Ranzato, and Arthur Szlam. Hard mixtures of experts for large scale weakly supervised vision. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5085–5093. IEEE, 2017.

Hongcan Guo, Haolang Lu, Guoshun Nan, Bolun Chu, Jialin Zhuang, Yuan Yang, Wenhao Che, Xinye Cao, Sicong Leng, Qimei Cui, et al. Advancing expert specialization for better moe. *Advances in Neural Information Processing Systems*, 38:48767–48809, 2026.

Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, et al. Tutel: Adaptive mixture-of-experts at scale. *Proceedings of Machine Learning and Systems*, 5:269–287, 2023.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, pp. 79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Peng Jin, Bo Zhu, Yuan Li, and Shuicheng Yan. Moe++: Accelerating mixture-of-experts methods with zero-computation experts. In *International Conference on Learning Representations*, volume 2025, pp. 50832–50856, 2025.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 2567–2577, 2019.

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.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pp. 6265–6274. PMLR, 2021.

Dengchun Li, Naizheng Wang, Zihao Zhang, Haoyang Yin, Lei Duan, Meng Xiao, and Mingjie Tang. Dynmole: Boosting mixture of lora experts fine-tuning with a hybrid routing mechanism. *arXiv preprint arXiv:2504.00661*, 2025.

Houyi Li, Ka Man Lo, Shijie Xuyang, Ziqi Wang, Wenzhen Zheng, Haocheng Zhang, Zhao Li, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Mixture-of-experts can surpass dense LLMs under strictly equal resource. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=oIdzliJAeA.

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

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Ang Lv, Jin Ma, Yiyuan Ma, and Siyuan Qiao. Coupling experts and routers in mixture-of-experts via an auxiliary loss. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=MpeyjgWbKt.

Nabil Omi, Siddhartha Sen, and Ali Farhadi. Load balancing mixture of experts with similarity preserving routers. *arXiv preprint arXiv:2506.14038*, 2025.

Antonio Orvieto and Robert Gower. In search of adam's secret sauce. *Advances in Neural Information Processing Systems*, 38:63404–63442, 2026.

Ashwinee Panda, Vatsal Baherwani, Zain Sarwar, Benjamin Therien, Sambit Sahu, Tom Goldstein, and Supriyo Chakraborty. Dense backpropagation improves training for sparse mixture-of-experts. *Advances in Neural Information Processing Systems*, 38:168625–168660, 2026.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=n6SCkn2QaG.

Zihan Qiu, Zeyu Huang, Bo Zheng, Kaiyue Wen, Zekun Wang, Rui Men, Ivan Titov, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Demons in the detail: On implementing load balancing loss for training specialized mixture-of-expert models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5005–5018, 2025.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 18332–18346. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/rajbhandari22a.html.

Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. Hash layers for large sparse models. *advances in neural information processing systems*, 34:17555–17566, 2021.

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.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.

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. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Varun Singh, Lucas Krauss, Sami Jaghouar, Matej Sirovatka, Charles Goddard, Fares Obied, Jack Min Ong, Jannik Straube, Aria Harley, Conner Stewart, et al. Arcee trinity large technical report. *arXiv preprint arXiv:2602.17004*, 2026.

Daria Soboleva. Router wars: Which moe routing strategy actually works, 2025. URL https://www.cerebras.ai/blog/moe-guide-router.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *ArXiv*, abs/2104.09864, 2021. URL https://api.semanticscholar.org/CorpusID:233307138.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In Leon Derczynski, Wei Xu, Alan Ritter, and Tim Baldwin (eds.), *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4413. URL https://aclanthology.org/W17-4413/.

Zijie Yan, Hongxiao Bai, Xin Yao, Dennis Liu, Tong Liu, Hongbin Liu, Pingtian Li, Evan Wu, Shiqing Fan, Li Tao, et al. Scalable training of mixture-of-experts models with megatron core. *arXiv preprint arXiv:2603.07685*, 2026.

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

Longrong Yang, Dong Shen, Chaoxiang Cai, Fan Yang, Tingting Gao, Di Zhang, and Xi Li. Solving token gradient conflict in mixture-of-experts for large vision-language model. In *International Conference on Learning Representations*, volume 2025, pp. 33561–33587, 2025b.

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

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

A MoE Gradient Analysis

A.1 Attention

We first consider the gradient propagated from the MoE layer to the preceding attention sublayer. Let $\boldsymbol{\theta}_A$ denote the parameters involved in producing the hidden representation $\boldsymbol{h}$ that serves as the input to the MoE layer. For notational simplicity, we suppress the sequence-level dependencies of the attention computation and focus on the resulting representation $\boldsymbol{h}$ and its upstream gradient.

Dependencies.

- Attention parameters: $\boldsymbol{\theta}_A$
- Attention output (MoE input): $\boldsymbol{h}$
- Upstream gradient to the attention output: $\nabla_{\boldsymbol{h}} \mathcal{L}_{\text{LM}}$

Forward Computation (Parameter $\rightarrow$ Loss).

$$\boldsymbol{\theta}_A \rightarrow \boldsymbol{h} \rightarrow \begin{cases} \boldsymbol{z} \xrightarrow{\text{routing operator}} \boldsymbol{p}, \\ \{\boldsymbol{y}_i\}_{i \in \mathcal{S}} \end{cases} \rightarrow \boldsymbol{y} \rightarrow \mathcal{L}_{\text{LM}} \tag{11}$$

Gradient Derivation. Applying the chain rule, the gradient with respect to the attention parameters is

$$\nabla_{\boldsymbol{\theta}_A} \mathcal{L}_{\text{LM}} = \left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{\theta}_A} \right)^{\top} \nabla_{\boldsymbol{h}} \mathcal{L}_{\text{LM}}, \tag{12}$$

where $\nabla_{\boldsymbol{h}} \mathcal{L}_{\text{LM}}$ denotes the gradient propagated to the attention output $\boldsymbol{h}$. Since $\boldsymbol{h}$ serves as the shared input to both the router and expert modules, the gradient with respect to $\boldsymbol{h}$ is obtained by summing the gradients propagated through the two computational branches:

$$\nabla_{\boldsymbol{h}} \mathcal{L}_{\text{LM}} = \underbrace{\nabla_{\boldsymbol{h}}^{(\boldsymbol{\theta}_E)} \mathcal{L}_{\text{LM}}}_{\text{Expert path}} + \underbrace{\nabla_{\boldsymbol{h}}^{(\boldsymbol{\theta}_R)} \mathcal{L}_{\text{LM}}}_{\text{Router path}}, \tag{13}$$

where $\nabla_{\boldsymbol{h}}^{(\boldsymbol{\theta}_E)} \mathcal{L}_{\text{LM}}$ and $\nabla_{\boldsymbol{h}}^{(\boldsymbol{\theta}_R)} \mathcal{L}_{\text{LM}}$ represent the gradients propagated through the expert and router branches, respectively. Their derivations are presented in the following subsections. Substituting the above decomposition into the chain rule yields the final gradient with respect to the attention parameters:

$$\boxed{\nabla_{\boldsymbol{\theta}_A} \mathcal{L}_{\text{LM}} = \left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{\theta}_A} \right)^{\top} \left(\nabla_{\boldsymbol{h}}^{(\boldsymbol{\theta}_E)} \mathcal{L}_{\text{LM}} + \nabla_{\boldsymbol{h}}^{(\boldsymbol{\theta}_R)} \mathcal{L}_{\text{LM}} \right)} \tag{14}$$

A.2 Expert

Dependencies.

- Output of the i-th expert: $\boldsymbol{y}_i = E_i(\boldsymbol{h}; \boldsymbol{\theta}_{E_i})$
- MoE output: $\boldsymbol{y} = \sum_{i \in \mathcal{S}} p_i \boldsymbol{y}_i$
- Upstream gradient to the MoE output: $\nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}}$

Forward Computation (Parameter $\rightarrow$ Loss). $\boldsymbol{\theta}_{E_i} \rightarrow \boldsymbol{y}_i \rightarrow \boldsymbol{y} \rightarrow \mathcal{L}_{\text{LM}}$

Gradient Derivation. Applying the chain rule, the gradient with respect to the parameters of expert i is

$$\nabla_{\boldsymbol{\theta}_{E_i}} \mathcal{L}_{\text{LM}} = \left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}_{E_i}} \right)^{\top} \nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}}. \tag{15}$$

Since the routing probabilities p_j are independent of the expert parameters $\boldsymbol{\theta}_{E_i}$ and the MoE output is defined as $\boldsymbol{y} = \sum_{j \in \mathcal{S}} p_j \boldsymbol{y}_j$, we obtain

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\theta}_{E_i}} = \sum_{j \in \mathcal{S}} p_j \frac{\partial \boldsymbol{y}_j}{\partial \boldsymbol{\theta}_{E_i}} = p_i \frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{\theta}_{E_i}}, \tag{16}$$

where the second equality follows from the fact that $\boldsymbol{y}_j$ depends only on $\boldsymbol{\theta}_{E_j}$. Substituting the above result into the chain rule yields the final gradient with respect to the parameters of expert i:

$$\boxed{\nabla_{\boldsymbol{\theta}_{E_i}} \mathcal{L}_{\mathrm{LM}} = p_i \left(\frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{\theta}_{E_i}} \right)^\top \nabla_{\boldsymbol{y}} \mathcal{L}_{\mathrm{LM}}} \tag{17}$$

The corresponding expert-branch component of the upstream gradient defined in Appendix A.1 is given by

$$\nabla_{\boldsymbol{h}}^{(\boldsymbol{\theta}_E)} \mathcal{L}_{\mathrm{LM}} = \sum_{i \in \mathcal{S}} p_i \left(\frac{\partial \boldsymbol{y}_i}{\partial \boldsymbol{h}} \right)^\top \nabla_{\boldsymbol{y}} \mathcal{L}_{\mathrm{LM}}. \tag{18}$$

A.3 Softmax Router

In this subsection, we consider the softmax router, where the routing probabilities are obtained by applying the softmax function to the router logits. Unlike sparse routing strategies, all experts contribute to the final MoE output without any masking or top-K selection. For consistency with the notation used for sparse routing, we set $\mathcal{S} = [N]$.

Dependencies.

- Router logits: $\boldsymbol{z} = \boldsymbol{W}_R \boldsymbol{h}$
- Routing probabilities: $\boldsymbol{p} = \mathrm{Softmax}(\boldsymbol{z})$
- Selected expert set: $\mathcal{S} = [N]$
- MoE output: $\boldsymbol{y} = \sum_{i \in \mathcal{S}} p_i \boldsymbol{y}_i = \sum_{i=1}^{N} p_i \boldsymbol{y}_i$
- Upstream gradient to the router logits: $\nabla_{\boldsymbol{z}} \mathcal{L}_{\mathrm{LM}}$
- Upstream gradient to the MoE output: $\nabla_{\boldsymbol{y}} \mathcal{L}_{\mathrm{LM}}$

Forward Computation (Parameter $\to$ Loss). $\boldsymbol{\theta}_R \to \boldsymbol{z} \xrightarrow{\mathrm{Softmax}} \boldsymbol{p} \to \boldsymbol{y} \to \mathcal{L}_{\mathrm{LM}}$

Gradient Derivation. Applying the chain rule, the gradient with respect to the router parameters is

$$\nabla_{\boldsymbol{\theta}_R} \mathcal{L}_{\mathrm{LM}} = \left(\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\theta}_R} \right)^\top \nabla_{\boldsymbol{z}} \mathcal{L}_{\mathrm{LM}}. \tag{19}$$

The local gradient $\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\theta}_R}$ depends only on the router architecture. Therefore, it remains to derive the upstream gradient $\nabla_{\boldsymbol{z}} \mathcal{L}_{\mathrm{LM}}$, which is propagated along the computation path $\boldsymbol{z} \to \boldsymbol{p} \to \boldsymbol{y} \to \mathcal{L}_{\mathrm{LM}}$:

$$\nabla_{\boldsymbol{z}} \mathcal{L}_{\mathrm{LM}} = \left(\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{z}} \right)^\top \nabla_{\boldsymbol{p}} \mathcal{L}_{\mathrm{LM}}. \tag{20}$$

To compute the upstream gradient with respect to the router logits, we first derive the gradient with respect to the routing probabilities:

$$\frac{\partial \mathcal{L}_{\mathrm{LM}}}{\partial p_i} = \left(\frac{\partial \boldsymbol{y}}{\partial p_i} \right)^\top \nabla_{\boldsymbol{y}} \mathcal{L}_{\mathrm{LM}} = \boldsymbol{y}_i^\top \nabla_{\boldsymbol{y}} \mathcal{L}_{\mathrm{LM}} \tag{21}$$

Since the MoE output is defined as $\boldsymbol{y} = \sum_{j=1}^{N} p_j \boldsymbol{y}_j$, it follows that $\frac{\partial \boldsymbol{y}}{\partial p_i} = \boldsymbol{y}_i$. Collecting the gradients with respect to all routing probabilities gives

$$\nabla_{\boldsymbol{p}} \mathcal{L}_{\mathrm{LM}} = \begin{bmatrix} \boldsymbol{y}_1^\top \nabla_{\boldsymbol{y}} \mathcal{L}_{\mathrm{LM}} \\ \vdots \\ \boldsymbol{y}_N^\top \nabla_{\boldsymbol{y}} \mathcal{L}_{\mathrm{LM}} \end{bmatrix}. \tag{22}$$

Next, we derive $\nabla_{\boldsymbol{z}} \mathcal{L}_{\mathrm{LM}}$ using the Jacobian of the softmax function. The Jacobian is given by

$$\frac{\partial p_j}{\partial z_i} = p_j(\mathbb{1}[j = i] - p_i), \tag{23}$$

Substituting the above results into the chain rule yields

$$\frac{\partial \mathcal{L}_{\text{LM}}}{\partial z_i} = \sum_{j=1}^{N} \frac{\partial p_j}{\partial z_i} \frac{\partial \mathcal{L}_{\text{LM}}}{\partial p_j} \tag{24}$$

$$= \sum_{j=1}^{N} p_j(\mathbb{1}[j=i] - p_i) \frac{\partial \mathcal{L}_{\text{LM}}}{\partial p_j} \tag{25}$$

$$= p_i(1-p_i)\frac{\partial \mathcal{L}_{\text{LM}}}{\partial p_i} - p_i \sum_{j \neq i}^{N} p_j \frac{\partial \mathcal{L}_{\text{LM}}}{\partial p_j} \tag{26}$$

$$= p_i(1-p_i)\frac{\partial \mathcal{L}_{\text{LM}}}{\partial p_i} - p_i \left(\sum_{j=1}^{N} p_j \frac{\partial \mathcal{L}_{\text{LM}}}{\partial p_j} - p_i \frac{\partial \mathcal{L}_{\text{LM}}}{\partial p_i} \right) \tag{27}$$

$$= p_i \left(\frac{\partial \mathcal{L}_{\text{LM}}}{\partial p_i} - \sum_{j=1}^{N} p_j \frac{\partial \mathcal{L}_{\text{LM}}}{\partial p_j} \right) \tag{28}$$

$$= p_i \left(\boldsymbol{y}_i^\top \nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}} - \sum_{j=1}^{N} p_j \boldsymbol{y}_j^\top \nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}} \right) \qquad \text{(by Equation (21))} \tag{29}$$

$$= p_i \left(\boldsymbol{y}_i - \sum_{j=1}^{N} p_j \boldsymbol{y}_j \right)^\top \nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}} \tag{30}$$

$$= p_i(\boldsymbol{y}_i - \boldsymbol{y})^\top \nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}} \qquad (\text{since } \boldsymbol{y} = \sum_{j=1}^{N} p_j \boldsymbol{y}_j). \tag{31}$$

Collecting all components, the upstream gradient with respect to the router logits can be written as

$$\boxed{\nabla_{\boldsymbol{z}} \mathcal{L}_{\text{LM}} = \begin{bmatrix} p_1(\boldsymbol{y}_1 - \boldsymbol{y})^\top \nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}} \\ \vdots \\ p_N(\boldsymbol{y}_N - \boldsymbol{y})^\top \nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}} \end{bmatrix}} \tag{32}$$

Substituting the above result into the chain rule yields the final gradient with respect to the router parameters:

$$\boxed{\nabla_{\boldsymbol{\theta}_R} \mathcal{L}_{\text{LM}} = \left(\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\theta}_R} \right)^\top \nabla_{\boldsymbol{z}} \mathcal{L}_{\text{LM}}} \tag{33}$$

The corresponding router-branch component of the upstream gradient defined in Section A.1 is therefore

$$\nabla_{\boldsymbol{h}}^{(\boldsymbol{\theta}_R)} \mathcal{L}_{\text{LM}} = \left(\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{h}} \right)^\top \nabla_{\boldsymbol{z}} \mathcal{L}_{\text{LM}}. \tag{34}$$

A.4 Softmax-TopK Router

We next consider the Softmax-TopK router. The router first computes dense routing probabilities over all experts using the softmax function and subsequently retains only the top-K probabilities. Let

$$\boldsymbol{p} = \text{Softmax}(\boldsymbol{z}), \qquad \mathcal{S} = \text{TopK}(\boldsymbol{p}, K), \tag{35}$$

where $\mathcal{S}$ denotes the set of selected experts. We define the corresponding binary selection mask as

$$m_i = \mathbb{1}[i \in \mathcal{S}]. \tag{36}$$

The masked routing probabilities are obtained by applying the binary mask to the dense routing probabilities,

$$\tilde{\boldsymbol{p}} = \boldsymbol{m} \odot \boldsymbol{p}, \qquad \tilde{p}_i = m_i p_i, \tag{37}$$

where $\odot$ denotes element-wise multiplication. The final MoE output is then given by

$$\boldsymbol{y} = \sum_{i=1}^{N} \tilde{p}_i \boldsymbol{y}_i = \sum_{i=1}^{N} m_i p_i \boldsymbol{y}_i = \sum_{i \in \mathcal{S}} p_i \boldsymbol{y}_i. \tag{38}$$

Dependencies.

- Router logits: $\boldsymbol{z} = \boldsymbol{W}_R \boldsymbol{h}$
- Dense routing probabilities: $\boldsymbol{p} = \mathrm{Softmax}(\boldsymbol{z})$
- Top-K selection mask: $m_i = \mathbb{1}[i \in \mathcal{S}]$
- Masked routing probabilities: $\tilde{\boldsymbol{p}} = \boldsymbol{m} \odot \boldsymbol{p}$
- MoE output: $\boldsymbol{y} = \sum_{i=1}^{N} \tilde{p}_i \boldsymbol{y}_i = \sum_{i=1}^{N} m_i p_i \boldsymbol{y}_i = \sum_{i \in \mathcal{S}} p_i \boldsymbol{y}_i$
- Upstream gradient to the router logits: $\nabla_{\boldsymbol{z}} \mathcal{L}_{\mathrm{LM}}$
- Upstream gradient to the MoE output: $\nabla_{\boldsymbol{y}} \mathcal{L}_{\mathrm{LM}}$

Forward Computation (Parameter $\to$ Loss). $\boldsymbol{\theta}_R \to \boldsymbol{z} \xrightarrow{\mathrm{Softmax}} \boldsymbol{p} \xrightarrow{\mathrm{TopK}} \tilde{\boldsymbol{p}} \to \boldsymbol{y} \to \mathcal{L}_{\mathrm{LM}}$

Gradient Derivation. The top-K selection mask $\boldsymbol{m} = \boldsymbol{m}(\boldsymbol{p})$ is a piecewise constant function of the routing probabilities $\boldsymbol{p}$. Except at the discrete boundaries where the ranking of elements changes, $\boldsymbol{m}(\boldsymbol{p})$ is locally invariant to small perturbations in $\boldsymbol{p}$. Following the standard automatic differentiation convention for piecewise constant operations, we treat $\boldsymbol{m}$ as constant during the backward pass, i.e., $\frac{\partial m_j}{\partial p_i} = 0$ almost everywhere. Under this locally fixed-mask assumption,

$$\frac{\partial \tilde{\boldsymbol{p}}}{\partial \boldsymbol{p}} = \mathrm{Diag}(\boldsymbol{m}). \tag{39}$$

Applying the chain rule, the gradient with respect to the router parameters is

$$\nabla_{\boldsymbol{\theta}_R} \mathcal{L}_{\mathrm{LM}} = \left(\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\theta}_R} \right)^{\top} \nabla_{\boldsymbol{z}} \mathcal{L}_{\mathrm{LM}}. \tag{40}$$

Compared to the Softmax (Section A.3), the Softmax-TopK router introduces an additional masking operation. We therefore first derive the gradient with respect to the dense routing probabilities $\boldsymbol{p}$. Since only the selected probabilities contribute to the MoE output,

$$\frac{\partial \mathcal{L}_{\mathrm{LM}}}{\partial p_i} = \begin{cases} \boldsymbol{y}_i^{\top} \nabla_{\boldsymbol{y}} \mathcal{L}_{\mathrm{LM}}, & i \in \mathcal{S}, \\ 0, & i \notin \mathcal{S}. \end{cases} \tag{41}$$

Using the softmax Jacobian in Equation (23), the upstream gradient with respect to the router logits is

$$\frac{\partial \mathcal{L}_{\mathrm{LM}}}{\partial z_i} = \sum_{j=1}^{N} \frac{\partial p_j}{\partial z_i} \frac{\partial \mathcal{L}_{\mathrm{LM}}}{\partial p_j} \tag{42}$$

$$= p_i \left(\frac{\partial \mathcal{L}_{\mathrm{LM}}}{\partial p_i} - \sum_{j=1}^{N} p_j \frac{\partial \mathcal{L}_{\mathrm{LM}}}{\partial p_j} \right) \tag{43}$$

$$= p_i \left(\frac{\partial \mathcal{L}_{\mathrm{LM}}}{\partial p_i} - \sum_{j \in \mathcal{S}} p_j \boldsymbol{y}_j^{\top} \nabla_{\boldsymbol{y}} \mathcal{L}_{\mathrm{LM}} \right) \tag{44}$$

$$= p_i \left(\frac{\partial \mathcal{L}_{\mathrm{LM}}}{\partial p_i} - \boldsymbol{y}^{\top} \nabla_{\boldsymbol{y}} \mathcal{L}_{\mathrm{LM}} \right), \tag{45}$$

where the last equality follows from $\boldsymbol{y} = \sum_{j \in \mathcal{S}} p_j \boldsymbol{y}_j$. Substituting Equation (41) gives

$$\boxed{\frac{\partial \mathcal{L}_{\mathrm{LM}}}{\partial z_i} = \begin{cases} p_i (\boldsymbol{y}_i - \boldsymbol{y})^{\top} \nabla_{\boldsymbol{y}} \mathcal{L}_{\mathrm{LM}}, & i \in \mathcal{S}, \\ -p_i \boldsymbol{y}^{\top} \nabla_{\boldsymbol{y}} \mathcal{L}_{\mathrm{LM}}, & i \notin \mathcal{S}. \end{cases}} \tag{46}$$

For selected experts ($i \in \mathcal{S}$), the gradient contains the expert-specific term $\boldsymbol{y}_i$. In contrast, for unselected experts ($i \notin \mathcal{S}$), this expert-specific information is absent, and the gradient depends only on the shared MoE output $\boldsymbol{y}$, scaled by p_i. The final gradient with respect to the router parameters is obtained by substituting the derived upstream gradient into Equation (33).

A.5 TopK-Softmax Router

We finally consider the TopK-Softmax. Unlike the Softmax-TopK in Section A.4, which masks the routing probabilities after softmax normalization, the TopK-Softmax first selects the top-K router logits and then normalizes only over the selected set. Consequently, the resulting routing probabilities satisfy $\sum_{i\in\mathcal{S}} p_i = 1$. Let

$$\mathcal{S} = \mathrm{TopK}(\boldsymbol{z}, K), \tag{47}$$

where $\mathcal{S}$ denotes the set of selected experts. Using the selected set $\mathcal{S}$, the masked logits are defined as

$$\tilde{z}_i = \begin{cases} z_i, & i \in \mathcal{S}, \\ -\infty & i \notin \mathcal{S}. \end{cases} \tag{48}$$

The routing probabilities are then obtained by applying the softmax function to the masked logits:

$$\boldsymbol{p} = \mathrm{Softmax}(\tilde{\boldsymbol{z}}). \tag{49}$$

Equivalently, each routing probability can be written as

$$p_i = \begin{cases} \frac{\exp(z_i)}{\sum_{j\in\mathcal{S}} \exp(z_j)}, & i \in \mathcal{S}, \\ 0, & i \notin \mathcal{S}. \end{cases} \tag{50}$$

The final MoE output is therefore

$$\boldsymbol{y} = \sum_{i=1}^{N} p_i \boldsymbol{y}_i = \sum_{i\in\mathcal{S}} p_i \boldsymbol{y}_i. \tag{51}$$

Dependencies.

- Router logits: $\boldsymbol{z} = \boldsymbol{W}_R \boldsymbol{h}$
- Selected expert set: $\mathcal{S} = \mathrm{TopK}(\boldsymbol{z}, K)$
- Masked router logits: $\tilde{z}_i = \begin{cases} z_i, & i \in \mathcal{S} \\ -\infty, & i \notin \mathcal{S} \end{cases}$
- Routing probabilities: $\boldsymbol{p} = \mathrm{Softmax}(\tilde{\boldsymbol{z}})$
- MoE output: $\boldsymbol{y} = \sum_{i=1}^{N} p_i \boldsymbol{y}_i = \sum_{i\in\mathcal{S}} p_i \boldsymbol{y}_i$
- Upstream gradient to the router logits: $\nabla_{\boldsymbol{z}} \mathcal{L}_{\mathrm{LM}}$
- Upstream gradient to the MoE output: $\nabla_{\boldsymbol{y}} \mathcal{L}_{\mathrm{LM}}$

Forward Computation (Parameter $\to$ Loss). $\boldsymbol{\theta}_R \to \boldsymbol{z} \xrightarrow{\mathrm{TopK}} \tilde{\boldsymbol{z}} \xrightarrow{\mathrm{Softmax}} \boldsymbol{p} \to \boldsymbol{y} \to \mathcal{L}_{\mathrm{LM}}$

Gradient Derivation. Applying the chain rule, the gradient with respect to the router parameters is

$$\nabla_{\boldsymbol{\theta}_R} \mathcal{L}_{\mathrm{LM}} = \left(\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\theta}_R} \right)^{\top} \nabla_{\boldsymbol{z}} \mathcal{L}_{\mathrm{LM}}. \tag{52}$$

As in the Softmax-TopK case, we treat the top-K selection set as locally fixed during the backward pass. Since the softmax function is applied only to the masked logits $\tilde{\boldsymbol{z}}$, its Jacobian with respect to the original logits has the standard softmax form within $\mathcal{S}$ and vanishes outside the selected set.

$$\frac{\partial p_j}{\partial z_i} = \begin{cases} p_j(\mathbb{1}[j=i] - p_i), & i, j \in \mathcal{S}, \\ 0, & \text{otherwise.} \end{cases} \tag{53}$$

From the definition of the MoE output (Equation (51)),

$$\frac{\partial \mathcal{L}_{\mathrm{LM}}}{\partial p_i} = \boldsymbol{y}_i^{\top} \nabla_{\boldsymbol{y}} \mathcal{L}_{\mathrm{LM}}, \qquad i \in \mathcal{S}. \tag{54}$$

The upstream gradient with respect to the router logits is then obtained by applying the chain rule,

$$\nabla_{\boldsymbol{z}} \mathcal{L}_{\text{LM}} = \left(\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{z}} \right)^{\top} \nabla_{\boldsymbol{p}} \mathcal{L}_{\text{LM}}. \tag{55}$$

Applying the same derivation as in Section A.3, together with the restricted Jacobian above, yields

$$\boxed{\frac{\partial \mathcal{L}_{\text{LM}}}{\partial z_i} = \begin{cases} p_i(\boldsymbol{y}_i - \boldsymbol{y})^{\top} \nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}}, & i \in \mathcal{S}, \\ 0, & i \notin \mathcal{S} \end{cases}} \tag{56}$$

Unlike the Softmax-TopK router, the gradient is identically zero for all unselected experts because both the routing probabilities and the corresponding Jacobian entries vanish outside the selected set. Consequently, once an expert is unselected for a token, its corresponding router logit receives no gradient signal from that token unless the expert becomes selected again. The final gradient with respect to the router parameters is obtained by substituting the derived upstream gradient into Eq. 33.

B Related Work

Sparse MoE Routing and Load Balancing. Sparse MoEs scale model capacity efficiently by activating only a small subset of experts per token, but imbalanced routing can leave some experts under-trained and create bottlenecks in expert-parallel execution. A common remedy is to augment the task objective with auxiliary routing losses. Fedus et al. (2022) introduced a load balancing loss that aligns token dispatch with routing probabilities. Subsequent work has extended this auxiliary objective paradigm to encourage other desirable routing properties, including router stability (Zoph et al., 2022), routing consistency (Omi et al., 2025), expert specialization (Guo et al., 2026), and routing sparsity (Li et al., 2025).

In parallel, several approaches control expert assignment without auxiliary objectives. Early methods fixed or constrained routing through clustering (Gross et al., 2017) or hashing (Dua et al., 2022; Roller et al., 2021), or formulated token–expert assignment as a balanced allocation problem (Lewis et al., 2021). More recently, motivated by the potential interference between auxiliary losses and the training objective, approaches have been proposed to directly control expert selection while retaining a learned router. Wang et al. (2024) adjusts expert-specific biases based on observed loads to favor under-utilized experts, while Singh et al. (2026) extends this approach with soft clamping and momentum. Quantile Balancing (Dial, 2026; Team et al., 2026) estimates balancing biases directly from the previous step's routing probability distributions.

Our work differs from these approaches in that our primary goal is not to directly control expert utilization, but to understand the optimization mechanism underlying routing collapse. We focus on how sparse expert selection shapes router gradient dynamics, particularly for unselected experts, and how these dynamics can sustain routing imbalance. Based on this analysis, we introduce a targeted gradient intervention that modifies only the router gradients for unselected experts while preserving sparse forward computation, substantially alleviating routing imbalance without an explicit balancing objective.

Understanding Routing Collapse in MoEs. Prior work has explained routing collapse from several perspectives. Early studies attributed imbalanced expert utilization to a self-reinforcing positive-feedback loop. If some experts happen to be selected more frequently early in training, they receive more parameter updates and improve more rapidly, which in turn makes them increasingly preferred by the router and gradually concentrates routing on a small subset of experts (Shazeer et al., 2017; Qiu et al., 2025).

A separate line of work on representation collapse has examined how routing structurally shapes the geometry of hidden representations. Chi et al. (2022); Do et al. (2023; 2025) showed that routing gradients lie within the subspace spanned by expert embeddings and can pull token representations assigned to the same expert toward a shared expert direction. Based on this observation, these studies suggested that the gradient structure induced by routing can reduce representation diversity and lead to representation collapse. However, how such representation contraction leads to concentrated expert utilization, and whether it explains persistent routing collapse in actual training dynamics, remains unclear.

Our work extends these perspectives by systematically examining how routing collapse emerges and persists across routing operators. We find that rapid reorganization of hidden representations early in training perturbs routing preferences, while subsequent routing dynamics depend on how the router adapts to these shifts. Specifically, under sparse routing, expert selection induces asymmetric router gradient structures, where gradients associated with unselected experts either lose expert-specific information or vanish entirely depending on the routing operator. These selection-dependent signals can limit the router's ability to realign routing preferences with the evolving representation geometry, allowing early routing bias to persist. A targeted intervention that restores expert-specific information to the router gradients of unselected experts substantially alleviates routing imbalance, providing intervention-based evidence that these gradient dynamics play a key role in sustaining routing collapse.

C Experimental Setup

C.1 Training Details

All models are trained from scratch using the same training pipeline, with architecture and optimization settings held fixed across routing variants within each model scale. Each model is trained on 100B tokens with a sequence length of 2,048. We use an effective batch size of 2,048 sequences, corresponding to approximately 4.19M tokens per optimization step, and train for 25,000 optimization steps. We fix the random seed to 0 for all experiments to ensure consistent initialization and training data ordering across variants.

For optimization, we use AdamW (Loshchilov & Hutter, 2017) with $\beta_1 = \beta_2 = 0.95$ (Orvieto & Gower, 2026) and no weight decay. We set the learning rate to 5×10^{-4} and use a cosine learning-rate schedule with linear warmup over the first 500 steps. The same optimization hyperparameters and learning rate schedule are used across all training configurations.

We use standard multi-head attention in all models. All linear and embedding parameters are initialized from a Gaussian distribution with mean 0 and standard deviation 0.02. Biases in linear layers are initialized to zero, and the embedding vector corresponding to the padding index, when present, is also initialized to zero. We use RMS normalization with $\epsilon = 10^{-6}$ and SiLU activations for both the attention and expert FFN modules, with no attention dropout. For rotary positional embeddings (Su et al., 2021), we set $\theta = 10{,}000$. All models are trained in bfloat16 precision.

The 0.68B/0.34B models are trained on a single NVIDIA B200 GPU, while the 1.92B/0.51B models are trained on a single NVIDIA RTX PRO 6000 Blackwell GPU. For sparse MoE models, pretraining on approximately 100B tokens takes roughly 7 and 20 days on a single GPU for the two model scales, respectively.

C.2 Evaluation Metrics

Expert Utilization. We quantify overall expert utilization using the effective number of experts, N_{eff}, defined in Section 3.3. For each MoE layer, let $\bar{\boldsymbol{p}} = [\bar{p}_1, \ldots, \bar{p}_N]^\top$ denote the aggregate routing distribution obtained by averaging the softmax-normalized routing probabilities over tokens. We compute

$$N_{\text{eff}} = \exp\left(-\sum_{i=1}^{N} \bar{p}_i \log \bar{p}_i\right). \tag{57}$$

This quantity is the exponentiated Shannon entropy of the aggregate routing distribution and satisfies $1 \leq N_{\text{eff}} \leq N$. It equals 1 when all routing probability mass is concentrated on a single expert and N when the routing distribution is uniform across experts. Thus, N_{eff} expresses the overall concentration of routing preferences directly on an expert-count scale.

To enable consistent comparison across sparse and dense routing operators, we compute N_{eff} from routing probabilities rather than expert loads, i.e., token dispatch fractions. Under Softmax, every token is dispatched to every expert, making expert loads uniform by construction even when routing probabilities are highly concentrated. Load-based statistics therefore cannot capture routing imbalance for this operator. We consequently use probability-based N_{eff} throughout our analyses. Detailed

utilization statistics across routing variants and layers, including both probability- and load-based measures, are provided in Section E.

Interpreting the Effective Number of Experts. To illustrate the intuitive interpretation of N_{eff}, Figure 6 shows the aggregate routing probabilities $\bar{p}_i$ for layer 1 of the 0.68B-scale models without an auxiliary loss. For Softmax-TopK and TopK-Softmax, the routing mass is concentrated on roughly three experts, with one expert particularly dominant under TopK-Softmax, corresponding to $N_{\text{eff}} = 2.96$ and 2.34, respectively. In contrast, Softmax and our intervention distribute routing mass across nearly all eight experts, with only moderate imbalance, yielding $N_{\text{eff}} = 7.25$ and 7.03. These examples illustrate that N_{eff} provides an intuitive expert-count scale summary of how broadly routing probability mass is distributed across experts.

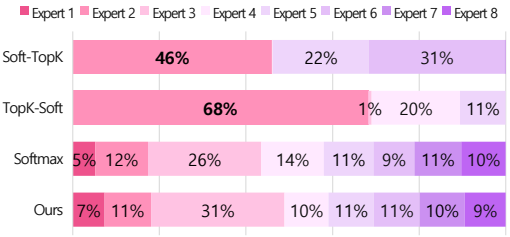

Figure 6: Aggregate routing probabilities $\bar{p}_i$ across experts for layer 1 of the 0.68B-scale models. "Soft" abbreviates Softmax in the sparse routing operator labels.

Importantly, $K = 2$ constrains only the number of experts selected per token, not the number of experts utilized across tokens. Thus, if routing preferences vary across tokens, the aggregate routing distribution can still be spread over many experts. Under routing collapse, however, these preferences concentrate on a small subset of experts, leading to a correspondingly small N_{eff}.

Why Use the Effective Number of Experts? Prior MoE studies commonly use metrics such as MaxVio (Wang et al., 2024) to quantify expert load imbalance:

$$\text{MaxVio} = \frac{\max \text{Load}_i - \overline{\text{Load}}}{\overline{\text{Load}}} \tag{58}$$

Here, Load_i denotes the load of expert i, and $\overline{\text{Load}}$ denotes the expected expert load under perfectly balanced routing. In our notation, Load_i corresponds to the token dispatch fraction f_i defined in Section 2. This metric measures the relative overload of the most heavily utilized expert compared with perfectly balanced routing. While useful for characterizing worst-case load imbalance, it depends only on the maximum dispatch fraction and therefore does not capture how utilization is distributed across the remaining experts. For example, with five experts, the dispatch distributions $[0.4, 0.4, 0.2, 0.0, 0.0]$ and $[0.4, 0.15, 0.15, 0.15, 0.15]$ have the same MaxVio, although the latter assigns nonzero load to all five experts whereas the former utilizes only three.

In contrast, N_{eff} depends on the full aggregate routing distribution and summarizes its concentration on an intuitive expert-count scale. It therefore distinguishes routing states with the same maximum utilization but different degrees of dispersion across experts, while remaining directly interpretable as the effective number of utilized experts.

Downstream Performance. We evaluate downstream performance using zero-shot accuracy on ARC-Easy (ARC-E) (Clark et al., 2018), WinoGrande (WG) (Sakaguchi et al., 2021), PIQA (Bisk et al., 2020), SocialIQA (SIQA) (Sap et al., 2019), HellaSwag (HS) (Zellers et al., 2019), PubMedQA (PMQA) (Jin et al., 2019), and SciQ (Welbl et al., 2017). All evaluations are conducted using the lm-evaluation-harness (Gao et al., 2024). To isolate the relationship between routing dynamics during pretraining and the resulting model performance, we evaluate all models directly after pretraining, without any additional instruction tuning or task-specific fine-tuning.

C.3 Representation Visualization

Since directly analyzing routing-vector trajectories in a high-dimensional representation space is difficult, we use principal component analysis (PCA) to visualize their trajectories in a low-dimensional space. Let $\boldsymbol{w}_i^{(s)}$ denote the routing vector of the i-th expert at the s-th training checkpoint, where $s = 1, \ldots, S$. We define the set of routing vectors collected across training steps as

$$\mathcal{W} = \left\{ \boldsymbol{w}_i^{(s)} \mid i = 1, \ldots, N,\ s = 1, \ldots, S \right\}. \tag{59}$$

The mean routing vector is given by

$$\bar{\boldsymbol{w}} = \frac{1}{SN} \sum_{s=1}^{S} \sum_{i=1}^{N} \boldsymbol{w}_i^{(s)}. \tag{60}$$

We denote the first two principal directions by $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2] \in \mathbb{R}^{D_{\text{hidden}} \times 2}$. As in standard PCA, $\boldsymbol{V}$ is computed from the mean-centered routing vectors $\boldsymbol{w}_i^{(s)} - \bar{\boldsymbol{w}}$ in $\mathcal{W}$. For visualization, however, we use

$$\hat{\boldsymbol{w}}_i^{(s)} = \boldsymbol{V}^\top \boldsymbol{w}_i^{(s)} \tag{61}$$

instead of the standard PCA coordinates $\boldsymbol{V}^\top(\boldsymbol{w}_i^{(s)} - \bar{\boldsymbol{w}})$. The two coordinate representations satisfy $\boldsymbol{V}^\top \boldsymbol{w}_i^{(s)} = \boldsymbol{V}^\top(\boldsymbol{w}_i^{(s)} - \bar{\boldsymbol{w}}) + \boldsymbol{V}^\top \bar{\boldsymbol{w}}$, and therefore differ only by a constant translation of $\boldsymbol{V}^\top \bar{\boldsymbol{w}}$ applied to all projected routing vectors. Thus, the relative displacements and trajectories among the projected routing vectors are preserved. We use this coordinate representation so that the same linear projection, $\hat{\boldsymbol{h}} = \boldsymbol{V}^\top \boldsymbol{h}$, can also be applied to hidden representations, allowing routing vectors and token representations to be analyzed in a common projected coordinate system while preserving the origin of the feature space

D Additional Details of Our Approach

D.1 Implementation Details

PyTorch-style pseudocode for the actual implementation is provided in Algorithms 1 and 2. In the forward pass of the MoE layer, we detach the softmax-normalized routing probabilities $\boldsymbol{p}$, thereby blocking the gradient path from the routing operator to the router logits $\boldsymbol{z}$. Since this stop gradient operation does not alter the forward values, expert selection and the sparse MoE output $\boldsymbol{y}$ are computed identically to the original routing procedure. During the backward pass, a custom autograd function passes the upstream gradient with respect to $\boldsymbol{y}$ unchanged through the sparse expert path, while returning the surrogate gradient for the router logits $\boldsymbol{z}$ defined in Equation (10). Consequently, the expert parameters receive gradients solely through the original sparse expert path, whereas only the gradient used to optimize the router is replaced by our proposed gradient.

D.2 Preserving Sparse Computation

Ours preserves the sparse forward computation of the original MoE layer. The counterfactual table is computed once per layer for each batch. Since tokens are grouped into at most N clusters according to their top-1 routing decisions and each cluster representative is evaluated by all N experts, the resulting expert evaluation cost is $\mathcal{O}(BTK + N^2)$, compared with $\mathcal{O}(BTN)$ for dense Softmax routing. Under the typical regime $BT \gg N$, the additional $\mathcal{O}(N^2)$ cost is amortized over the batch, effectively preserving the $\mathcal{O}(BTK)$ scaling of sparse routing.

The corresponding forward computation is reported in Table 3. For the 0.68B model, standard sparse routing requires 0.55576 GFLOPs per token, compared with 1.23524 GFLOPs for Softmax routing and 0.55576 GFLOPs for Ours. For the 1.92B model, the corresponding costs are 0.86694, 3.68483, and 0.86696 GFLOPs per token, respectively. The additional counterfactual computation of Ours amounts to only 0.00173 and 0.01229 MFLOPs per token for the 0.68B and 1.92B models, corresponding to approximately 0.0003% and 0.0014% overhead over standard sparse routing, respectively.

D.3 Analysis of the Zero-Sum Property

We use $\hat{\boldsymbol{y}}$ as the common reference in Equation (10) to preserve the zero-sum property of the Softmax router gradient. Since $\sum_{i=1}^{N} p_i = 1$ and

$$\hat{\boldsymbol{y}} = \sum_{i \in \mathcal{S}} p_i \boldsymbol{y}_i + \sum_{i \notin \mathcal{S}} p_i \hat{\boldsymbol{y}}_{i,c}, \tag{62}$$

```python
class MoELayer(nn.Module):
    def forward(self, h):
        B, T, D = h.shape
        N, K = self.num_experts, self.top_k
        # Sparse routing
        z = self.router(h)                                  # [B,T,N]
        p = torch.softmax(z, dim=-1).detach()
        topk_p, topk_idx = p.topk(K, dim=-1)                # [B,T,K]
        # Standard sparse MoE forward
        y, expert_outputs = sparse_expert_forward(  # [B,T,D], [B,T,K,D]
            h, topk_p, topk_idx, self.experts
        )
        # Counterfactual expert outputs
        proxy_idx = p.argmax(dim=-1)                        # [B,T]
        proxy = self.compute_proxy(h, proxy_idx)     # [N,D]
        with torch.no_grad():
            proxy_table = torch.stack(                      # [N,N,D]
                [expert(proxy) for expert in self.experts],
                dim=1,
            )
        # Replace only the router gradient
        y = ProxyRouterGradient.apply(
            y, z, p, topk_idx, expert_outputs, proxy_table, proxy_idx
        )
        return y

    @torch.no_grad()
    def compute_proxy(self, h, top1_idx):
        B, T, D = h.shape
        N = self.num_experts
        # Group tokens by top-1 expert
        h = h.reshape(-1, D)                                # [BT,D]
        idx = top1_idx.reshape(-1)                          # [BT]
        # Aggregate representations by cluster
        proxy = torch.zeros(N, D, device=h.device, dtype=h.dtype)
        proxy.index_add_(0, idx, h)
        # Cluster-wise mean
        count = torch.bincount(idx, minlength=N).clamp_min(1)
        return proxy / count[:, None]                       # [N,D]
```

Algorithm 1: PyTorch-style pseudocode of the sparse MoE forward pass and proxy construction.

the proposed router gradients satisfy

$$\sum_{i=1}^{N} \frac{\partial \mathcal{L}_{\text{LM}}}{\partial z_i} = \left[\sum_{i \in \mathcal{S}} p_i (\boldsymbol{y}_i - \hat{\boldsymbol{y}}) + \sum_{i \notin \mathcal{S}} p_i (\hat{\boldsymbol{y}}_{i,c} - \hat{\boldsymbol{y}}) \right]^{\top} \nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}} \tag{63}$$

$$= \left[\sum_{i \in \mathcal{S}} p_i \boldsymbol{y}_i + \sum_{i \notin \mathcal{S}} p_i \hat{\boldsymbol{y}}_{i,c} - \hat{\boldsymbol{y}} \sum_{i=1}^{N} p_i \right]^{\top} \nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}} \tag{64}$$

$$= (\hat{\boldsymbol{y}} - \hat{\boldsymbol{y}})^{\top} \nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}} \tag{65}$$

$$= 0. \tag{66}$$

Thus, introducing counterfactual responses for unselected experts does not alter the zero-sum structure of the original Softmax router gradient. In contrast, suppose that the actual sparse output $\boldsymbol{y} = \sum_{i \in \mathcal{S}} p_i \boldsymbol{y}_i$ is used as the common reference while retaining the counterfactual responses for

```python
class ProxyRouterGradient(torch.autograd.Function):
    @staticmethod
    def forward(
        ctx, y, z, p, topk_idx, expert_outputs, proxy_table, proxy_idx
    ):
        ctx.save_for_backward(
            p, topk_idx, expert_outputs, proxy_table, proxy_idx
        )
        return y

    @staticmethod
    def backward(ctx, grad_y): # grad_y: upstream gradient
        (
            p, topk_idx, expert_outputs, proxy_table, proxy_idx
        ) = ctx.saved_tensors
        # Counterfactual scores for all experts
        proxy = proxy_table[proxy_idx]                  # [B,T,N,D]
        s = torch.einsum(                               # [B,T,N]
            "btd,btnd->btn", grad_y, proxy
        )
        # Exact scores for selected experts
        selected_s = torch.einsum(                      # [B,T,K]
            "btd,btkd->btk", grad_y, expert_outputs
        )
        s.scatter_(-1, topk_idx, selected_s)
        # Probability-weighted mean baseline
        baseline = (p * s).sum(dim=-1, keepdim=True)
        grad_z = p * (s - baseline)
        return grad_y, grad_z, None, None, None, None, None
```

Algorithm 2: PyTorch-style pseudocode of our proxy-based router gradient.

unselected experts. In this case,

$$\sum_{i=1}^{N} \frac{\partial \mathcal{L}_{\text{LM}}}{\partial z_i} = \left[\sum_{i \in \mathcal{S}} p_i (\boldsymbol{y}_i - \boldsymbol{y}) + \sum_{i \notin \mathcal{S}} p_i (\hat{\boldsymbol{y}}_{i,c} - \boldsymbol{y}) \right]^{\top} \nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}} \tag{67}$$

$$= \left[\sum_{i \in \mathcal{S}} p_i \boldsymbol{y}_i + \sum_{i \notin \mathcal{S}} p_i \hat{\boldsymbol{y}}_{i,c} - \boldsymbol{y} \sum_{i=1}^{N} p_i \right]^{\top} \nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}} \tag{68}$$

$$= \left[\boldsymbol{y} + \sum_{i \notin \mathcal{S}} p_i \hat{\boldsymbol{y}}_{i,c} - \boldsymbol{y} \right]^{\top} \nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}} \tag{69}$$

$$= \left(\sum_{i \notin \mathcal{S}} p_i \hat{\boldsymbol{y}}_{i,c} \right)^{\top} \nabla_{\boldsymbol{y}} \mathcal{L}_{\text{LM}}, \tag{70}$$

$$\tag{71}$$

which is generally nonzero. Since $\frac{\partial z_i}{\partial \boldsymbol{w}_i} = \boldsymbol{h}$, a nonzero sum of router logit gradients also yields a nonzero aggregate routing vector gradient,

$$\sum_{i=1}^{N} \frac{\partial \mathcal{L}_{\text{LM}}}{\partial \boldsymbol{w}_i} = \left(\sum_{i=1}^{N} \frac{\partial \mathcal{L}_{\text{LM}}}{\partial z_i} \right) \boldsymbol{h}. \tag{72}$$

Thus, when the zero-sum property is violated, a shared gradient component along the current representation $\boldsymbol{h}$ no longer cancels across experts. We also evaluated this alternative formulation using the actual sparse output $\boldsymbol{y}$ as the reference and observed substantially less stable training. The loss increased sharply during training, followed by non-finite hidden activations that terminated the run. In contrast, the proposed formulation using $\hat{\boldsymbol{y}}$ preserves the zero-sum structure and trains stably.

METHODS	$N_{\text{eff}}^{\text{avg}}$	$N_{\text{eff}}^{\text{worst}}$	ARC-E	WG	PIQA	SIQA	HS	PMQA	SciQ	Avg.↑
0.68B / 0.34B										
Cluster Mean†	7.46	6.56	47.81	52.72	67.90	38.54	39.63	54.40	71.50	53.21
Cluster Mean + Scale	7.59	7.34	45.20	52.17	67.25	38.33	37.66	38.00	67.50	49.44
Routing Vector	6.92	2.97	47.22	52.72	68.55	39.46	39.00	52.20	66.00	52.16
Routing Vector + Scale	7.16	4.15	46.93	52.96	68.61	38.02	39.02	56.20	69.60	53.05
1.92B / 0.51B										
Cluster Mean†	14.92	13.12	50.08	52.41	70.29	39.15	46.74	54.40	75.10	55.45
Cluster Mean + Scale	14.84	13.93	49.87	50.67	69.64	39.15	42.77	44.60	74.50	53.03
Routing Vector	13.89	7.49	51.05	52.41	70.02	38.95	46.41	56.20	76.80	55.98
Routing Vector + Scale	13.75	6.59	49.45	49.72	70.40	39.15	44.28	56.80	74.00	54.83

†Cluster Mean is used in Section 6.

Table 4: Ablation study on proxy-vector construction. We report $N_{\text{eff}}^{\text{avg}}$, $N_{\text{eff}}^{\text{worst}}$, and zero-shot accuracy on ARC-Easy (ARC-E), WinoGrande (WG), PIQA, SocialIQA (SIQA), HellaSwag (HS), PubMedQA (PMQA), SciQ, and the average.

These observations further motivate preserving the zero-sum property when constructing the surrogate router gradient.

D.4 Ablation on Proxy-Vector Construction

We ablate the choice of proxy vector used to construct counterfactual expert responses. Since tokens are clustered by their top-1 routing decision, each cluster requires a single representative vector that is evaluated by all experts. We compare four choices.

Cluster Mean. Our default choice uses the mean hidden representation of tokens within each cluster, as described in Section 6.1.

Cluster Mean + Scale. We rescale the cluster mean to match the average norm of hidden representations while preserving its direction. This controls for the reduction in norm that can arise from averaging multiple token representations.

Routing Vector. Instead of using a representative hidden state, we use the routing vector $\boldsymbol{w}_i$ associated with the expert defining each top-1 cluster as the proxy vector.

Routing Vector + Scale. We additionally rescale the routing vector to match the average hidden-representation norm, isolating the effect of its direction from differences in vector magnitude.

Results are reported in Table 4. The cluster mean provides consistently balanced expert utilization while maintaining competitive downstream performance across both model scales. Using the routing vector as the proxy maintains comparable average accuracy, but results in substantially lower worst-layer expert utilization. Meanwhile, rescaling either proxy to match the average hidden representation norm provides no consistent improvement in either expert utilization or downstream performance. These results suggest that the cluster mean provides a simple and stable representative of each routing cluster, and we therefore use the unscaled cluster mean as our default proxy.

E Additional Experimental Results

E.1 Routing Dynamics during Training

Figures 19 and 20 present the full layer-wise evolution of the effective number of experts for the 0.68B- and 1.92B-scale models, respectively, extending the representative-layer results shown in Section 4.1.

E.2 Early Representation Reorganization Perturbs Routing

Figures 21 and 22 present the full hidden-representation geometry analysis for the 0.68B- and 1.92B-scale models, respectively. Across model variants, the representations undergo pronounced geometric changes early in training, extending the representative Softmax-TopK and Dense examples shown in Section 5.1.

METHODS	0.68B / 0.34B $N_{\text{eff}}^{\text{avg}}$	0.68B / 0.34B $N_{\text{eff}}^{\text{worst}}$	1.92B / 0.51B $N_{\text{eff}}^{\text{avg}}$	1.92B / 0.51B $N_{\text{eff}}^{\text{worst}}$
Softmax-TopK	7.03	2.96	13.79	6.58
TopK-Softmax	5.48	2.34	8.09	2.08
Softmax	7.87	7.25	15.56	14.16
Ours	7.46	6.56	14.92	13.12
Softmax-TopK$_{\text{aux}}$	7.95	7.86	15.86	15.72
TopK-Softmax$_{\text{aux}}$	7.46	5.85	14.54	10.61
Ours$_{\text{aux}}$	7.94	7.79	15.81	15.56

Table 5: Model-level effective number of experts based on routing probabilities. We report the layer average $N_{\text{eff}}^{\text{avg}}$ and the minimum across layers $N_{\text{eff}}^{\text{worst}}$ for both model scales.

METHODS	0.68B / 0.34B $N_{\text{eff}}^{\text{avg}}$	0.68B / 0.34B $N_{\text{eff}}^{\text{worst}}$	1.92B / 0.51B $N_{\text{eff}}^{\text{avg}}$	1.92B / 0.51B $N_{\text{eff}}^{\text{worst}}$
Softmax-TopK	6.37	2.71	10.80	2.00
TopK-Softmax	5.64	2.95	8.31	3.25
Softmax	8.00	8.00	16.00	16.00
Ours	6.56	5.08	11.83	6.21
Softmax-TopK$_{\text{aux}}$	7.93	7.72	15.83	15.45
TopK-Softmax$_{\text{aux}}$	7.35	5.70	14.48	10.78
Ours$_{\text{aux}}$	7.85	7.48	15.73	15.11

Table 6: Model-level effective number of experts based on dispatched-token fractions. We report the layer average $N_{\text{eff}}^{\text{avg}}$ and the minimum across layers $N_{\text{eff}}^{\text{worst}}$ for both model scales.

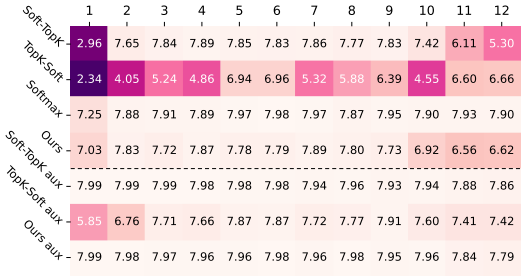

Figure 7: Layer-wise effective number of experts based on routing probabilities for the 0.68B-scale models.

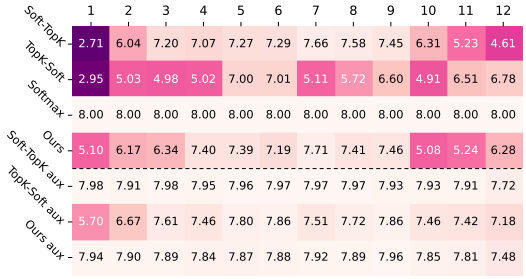

Figure 8: Layer-wise effective number of experts based on dispatched-token fractions for the 0.68B-scale models.

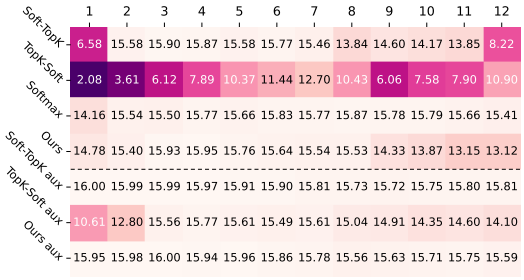

Figure 9: Layer-wise effective number of experts based on routing probabilities for the 1.92B-scale models.

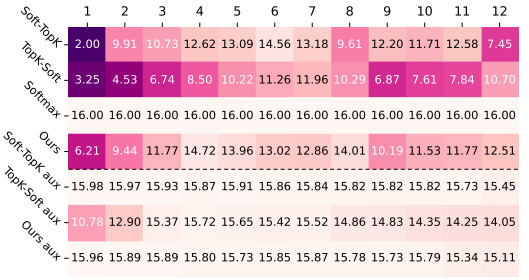

Figure 10: Layer-wise effective number of experts based on dispatched-token fractions for the 1.92B-scale models.

E.3 ADDITIONAL ANALYSIS OF OPERATOR-DEPENDENT ROUTING COLLAPSE

We report complementary routing imbalance statistics based on both routing probabilities and dispatched-token fractions. The probability-based metric characterizes how routing preference is distributed across experts and enables a consistent comparison across all routing operators, including dense Softmax routing. In contrast, the dispatched-token fraction directly reflects the distribution of actual expert computation under sparse routing.

For dense Softmax routing, however, every token is evaluated by every expert, so the dispatched-token fraction is $f_i = 1$ for all experts. Accordingly, any load-based effective-number statistic becomes trivially maximal and does not provide a meaningful measure of routing imbalance for Softmax. For this reason, we use probability-based N_{eff} in the main text for cross-operator comparison, while reporting both probability- and load-based statistics here for completeness.

Model-level probability-based results are summarized in Table 5, while the corresponding dispatched-token-based results are shown in Table 6. The full layer-wise results are provided in Figures 7 and 8 for the 0.68B-scale models and in Figures 9 and 10 for the 1.92B-scale models.

METHODS	PC1	PC2	PC3	PC1	PC2	PC3	PC1	PC2	PC3
	Layer 1			**Layer 2**			**Layer 3**		
Softmax-TopK	**55.08**	9.95	8.25	17.73	16.65	15.84	18.45	16.75	15.32
TopK-Softmax	27.88	18.07	14.17	**29.83**	17.72	15.53	23.51	19.94	16.70
Softmax	**35.47**	15.19	12.68	23.00	16.98	15.65	21.39	18.14	15.92
Ours	**24.58**	18.53	13.80	19.92	17.51	14.75	19.92	17.77	14.86
	Layer 4			**Layer 5**			**Layer 6**		
Softmax-TopK	18.65	16.53	16.06	20.08	17.76	14.63	20.54	18.58	16.88
TopK-Softmax	22.23	19.56	16.08	26.29	21.84	15.89	27.22	19.79	15.74
Softmax	20.17	17.87	15.44	21.24	17.68	16.25	23.25	19.71	14.57
Ours	20.61	18.75	17.30	19.62	18.15	15.66	23.68	20.03	14.67
	Layer 7			**Layer 8**			**Layer 9**		
Softmax-TopK	20.74	18.41	16.75	24.98	20.89	13.76	29.12	22.09	14.79
TopK-Softmax	21.12	20.02	18.30	24.68	20.27	17.07	**29.54**	22.99	14.03
Softmax	22.61	19.81	16.09	30.84	21.91	12.28	23.19	18.36	16.35
Ours	22.84	17.80	14.64	24.02	22.16	18.18	25.65	21.91	17.84
	Layer 10			**Layer 11**			**Layer 12**		
Softmax-TopK	28.79	24.75	15.75	**41.32**	22.15	17.72	**31.14**	23.02	15.67
TopK-Softmax	**34.60**	18.44	13.04	25.81	23.44	18.65	26.82	23.31	19.45
Softmax	22.30	21.89	16.61	25.62	21.14	16.25	**41.90**	17.82	13.42
Ours	24.87	22.87	19.04	**26.25**	24.08	15.74	**32.65**	21.63	20.15

Table 7: Layer-wise explained variance ratios of the top three principal components for the 0.68B-scale models. Bold values indicate the largest PC1 explained variance ratio within each layer and are used to identify representative layers for PCA visualization.

E.4 PCA EXPLAINED VARIANCE AND ADDITIONAL VISUALIZATIONS

We report the explained variance ratios of the first three principal components for all layers and routing operators in Tables 7 and 8, providing a quantitative characterization of the PCA projections used in Section 5.3. The explained variance ratios vary substantially across model scales, routing operators, and layers. PC1 often captures a relatively large fraction of the variance in the 0.68B models, while the corresponding ratios are generally lower in the 1.92B models. The latter operate in a higher-dimensional hidden space and contain more expert routing vectors per layer (1024 dimensions and 16 experts, compared with 768 dimensions and 8 experts for the 0.68B models), making low-dimensional projections less descriptive of the full variation in the original space.

Accordingly, we use the PCA projections primarily to visualize routing vector dynamics, with the understanding that they capture only a subset of the variation in the original high-dimensional space. For the additional visualizations, we focus on layers with relatively large PC1 explained variance ratios, where the projection preserves a larger fraction of the variation in the original space. These cases are highlighted in bold in Tables 7 and 8. Additional visualizations for each routing operator are provided in Figures 11 to 14 for the 0.68B models and Figures 15 to 18 for the 1.92B models.

METHODS	PC1	PC2	PC3	PC1	PC2	PC3	PC1	PC2	PC3
	Layer 1			**Layer 2**			**Layer 3**		
Softmax-TopK	9.11	7.96	7.38	8.03	7.45	7.31	8.83	8.37	7.84
TopK-Softmax	13.11	11.03	6.69	**20.07**	9.65	7.94	14.38	10.31	8.29
Softmax	13.08	9.66	8.55	13.22	11.10	8.80	10.35	9.55	8.27
Ours	**19.23**	10.04	7.69	8.38	8.15	7.52	8.00	7.80	7.50
	Layer 4			**Layer 5**			**Layer 6**		
Softmax-TopK	9.02	8.06	7.62	10.94	10.05	9.44	11.27	10.19	9.09
TopK-Softmax	10.54	9.70	8.71	11.01	9.45	8.76	12.29	10.40	8.95
Softmax	11.09	9.24	8.75	11.84	11.10	8.32	11.80	10.96	8.85
Ours	9.17	9.00	8.49	9.96	9.45	9.05	10.45	10.15	8.92
	Layer 7			**Layer 8**			**Layer 9**		
Softmax-TopK	13.45	11.97	11.02	15.50	11.87	10.64	17.53	14.60	12.00
TopK-Softmax	12.16	10.52	9.33	13.69	11.39	10.01	16.19	12.54	11.45
Softmax	12.94	10.45	9.67	**15.83**	9.46	9.27	13.56	10.78	8.59
Ours	12.28	10.68	9.64	14.18	11.78	11.05	15.98	13.63	10.36
	Layer 10			**Layer 11**			**Layer 12**		
Softmax-TopK	**18.51**	14.96	11.81	**23.18**	12.41	10.29	**28.19**	13.86	11.44
TopK-Softmax	11.82	11.04	10.45	13.28	10.56	9.55	**17.12**	13.69	12.58
Softmax	12.95	12.13	9.81	14.50	11.39	10.46	**21.49**	11.82	9.38
Ours	**18.62**	15.59	9.82	**26.44**	15.44	10.53	18.05	14.73	11.23

Table 8: Layer-wise explained variance ratios of the top three principal components for the 1.92B-scale models. Bold values indicate the largest PC1 explained variance ratio within each layer and are used to identify representative layers for PCA visualization.

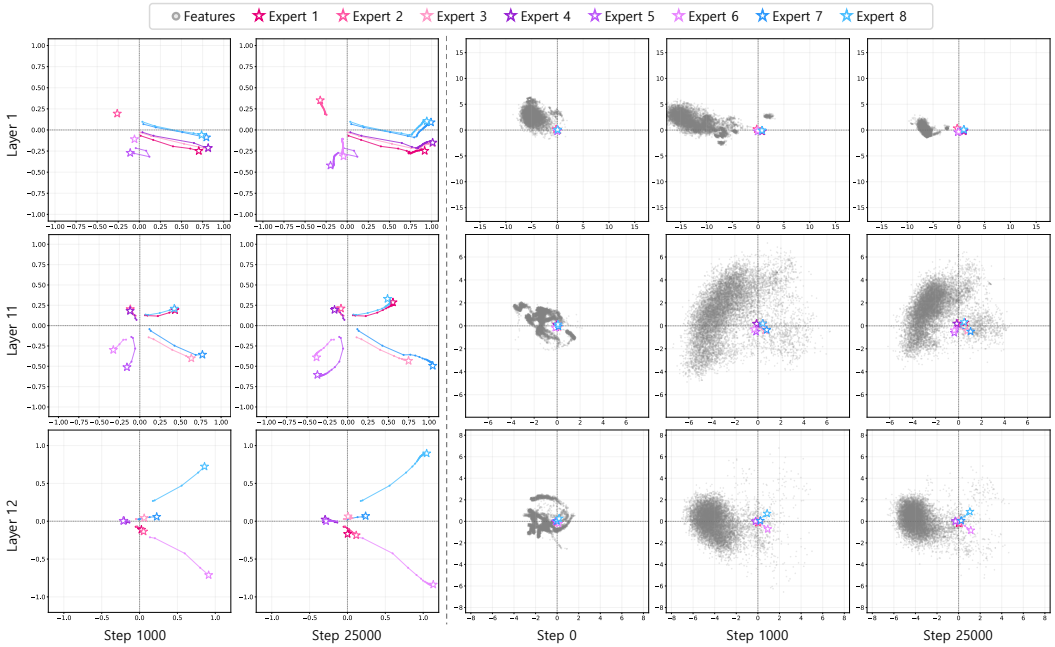

Figure 11: PCA visualization of routing dynamics for the 0.68B-scale Softmax-TopK model. Rows show representative layers selected based on relatively large PC1 explained variance ratios.

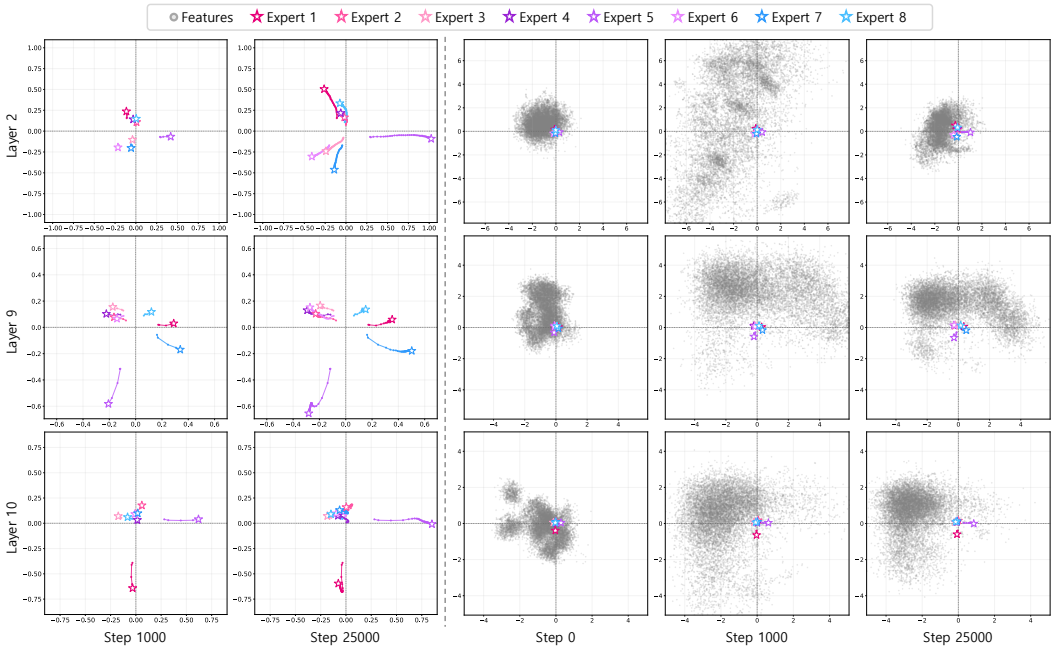

Figure 12: PCA visualization of routing dynamics for the 0.68B-scale TopK-Softmax model. Rows show representative layers selected based on relatively large PC1 explained variance ratios.

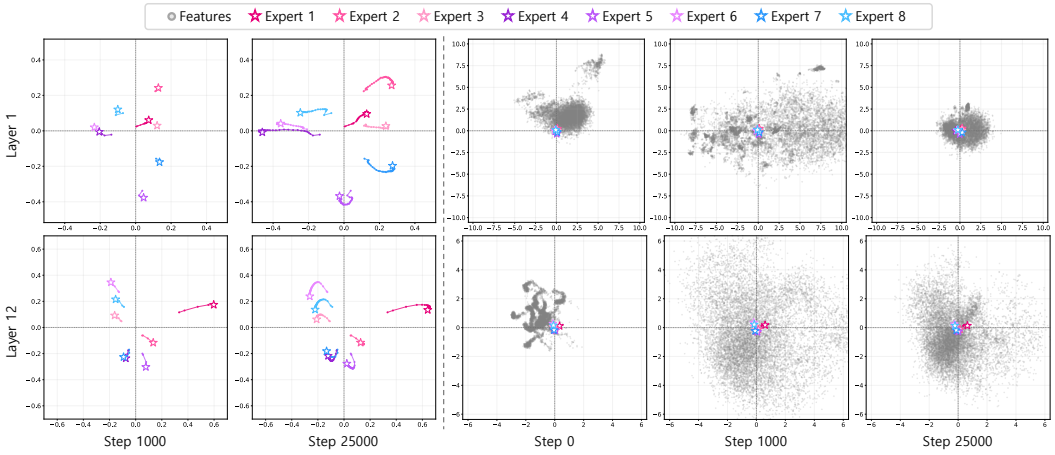

Figure 13: PCA visualization of routing dynamics for the 0.68B-scale Softmax model. Rows show representative layers selected based on relatively large PC1 explained variance ratios.

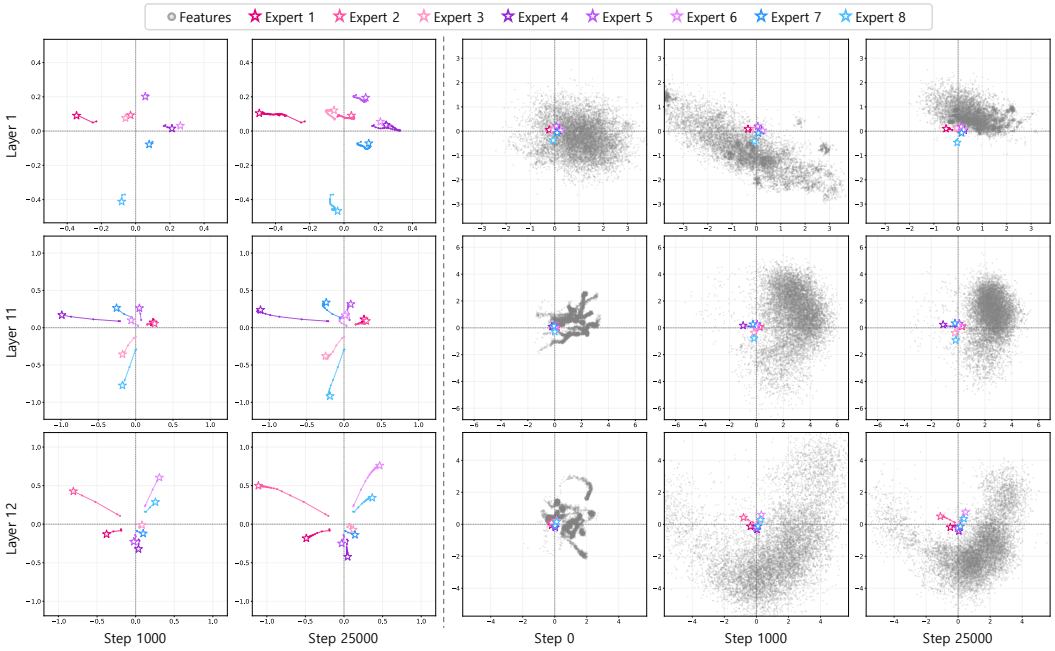

Figure 14: PCA visualization of routing dynamics for the 0.68B-scale Ours model. Rows show representative layers selected based on relatively large PC1 explained variance ratios.

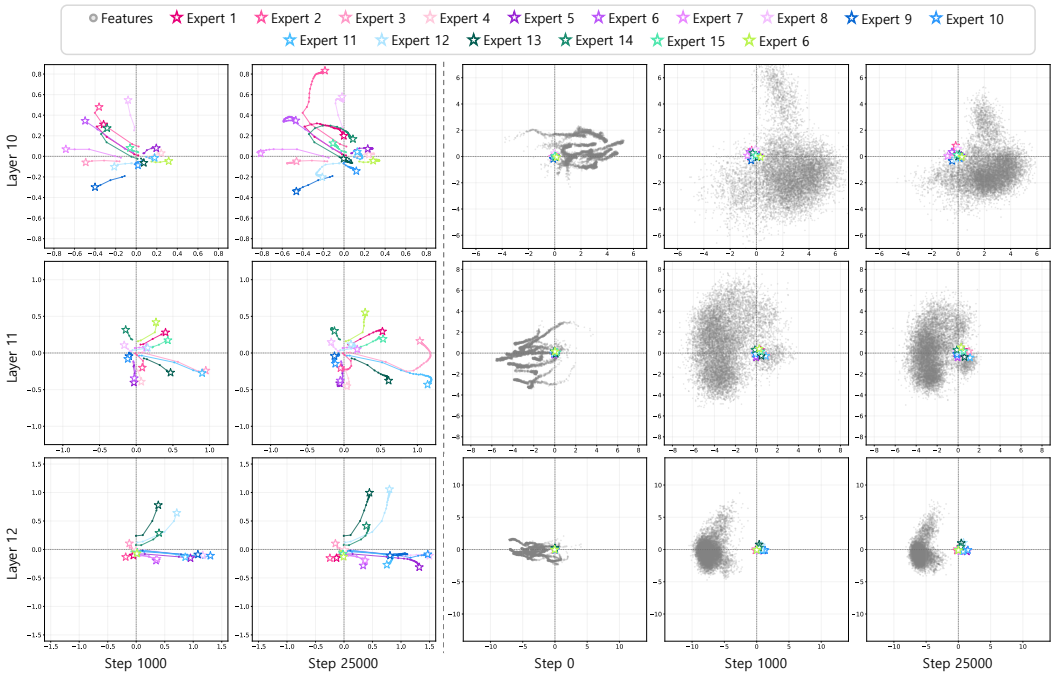

Figure 15: PCA visualization of routing dynamics for the 1.92B-scale Softmax-TopK model. Rows show representative layers selected based on relatively large PC1 explained variance ratios.

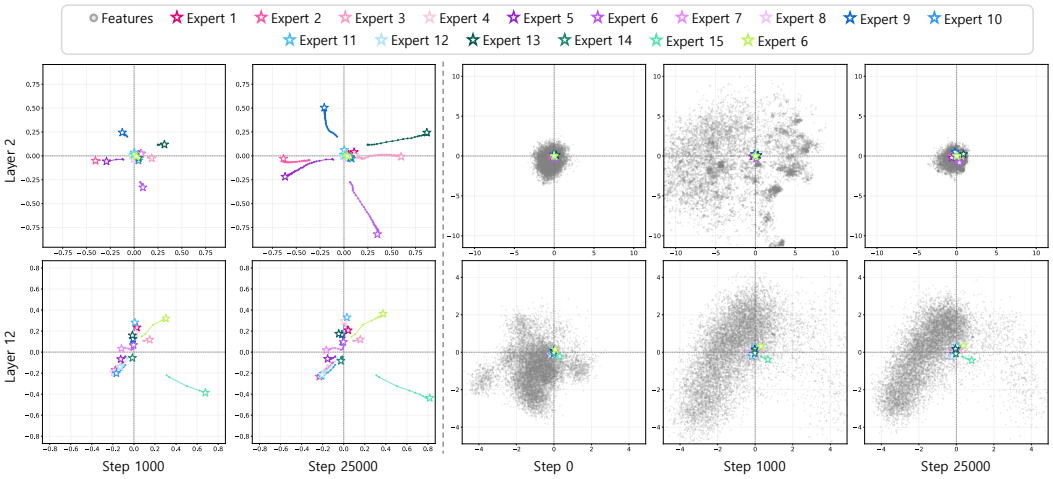

Figure 16: PCA visualization of routing dynamics for the 1.92B-scale TopK-Softmax model. Rows show representative layers selected based on relatively large PC1 explained variance ratios.

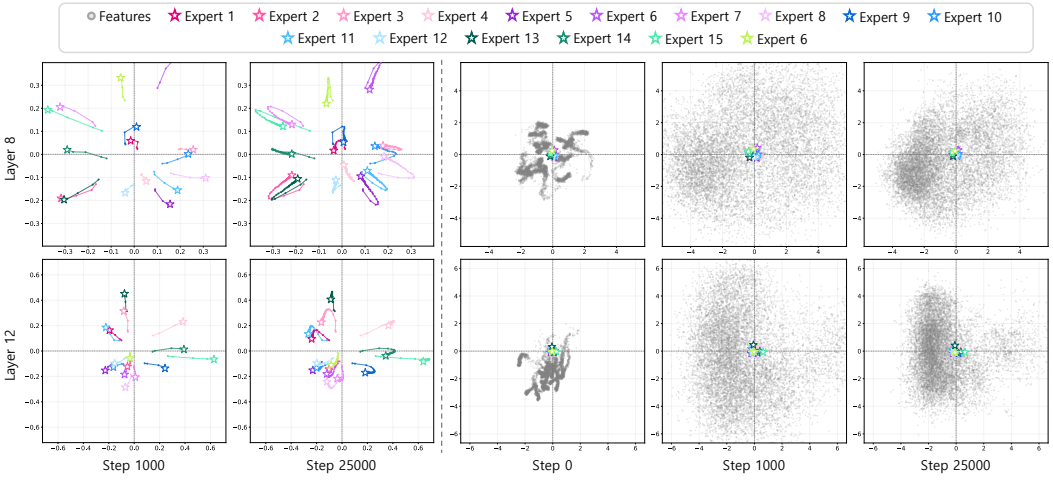

Figure 17: PCA visualization of routing dynamics for the 1.92B-scale Softmax model. Rows show representative layers selected based on relatively large PC1 explained variance ratios.

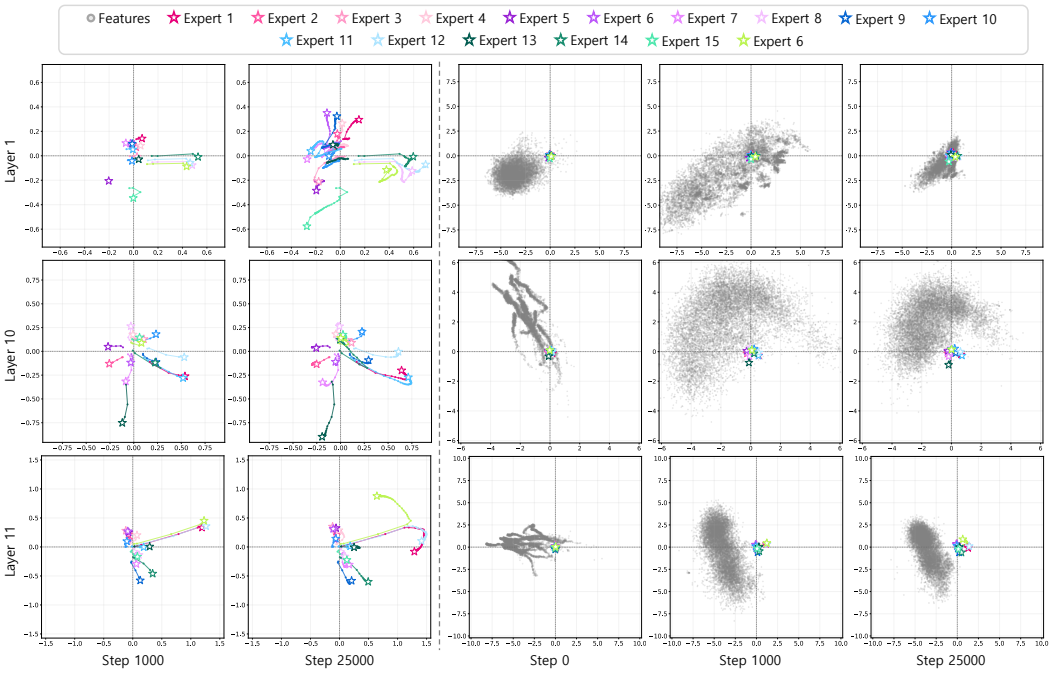

Figure 18: PCA visualization of routing dynamics for the 1.92B-scale Ours model. Rows show representative layers selected based on relatively large PC1 explained variance ratios.

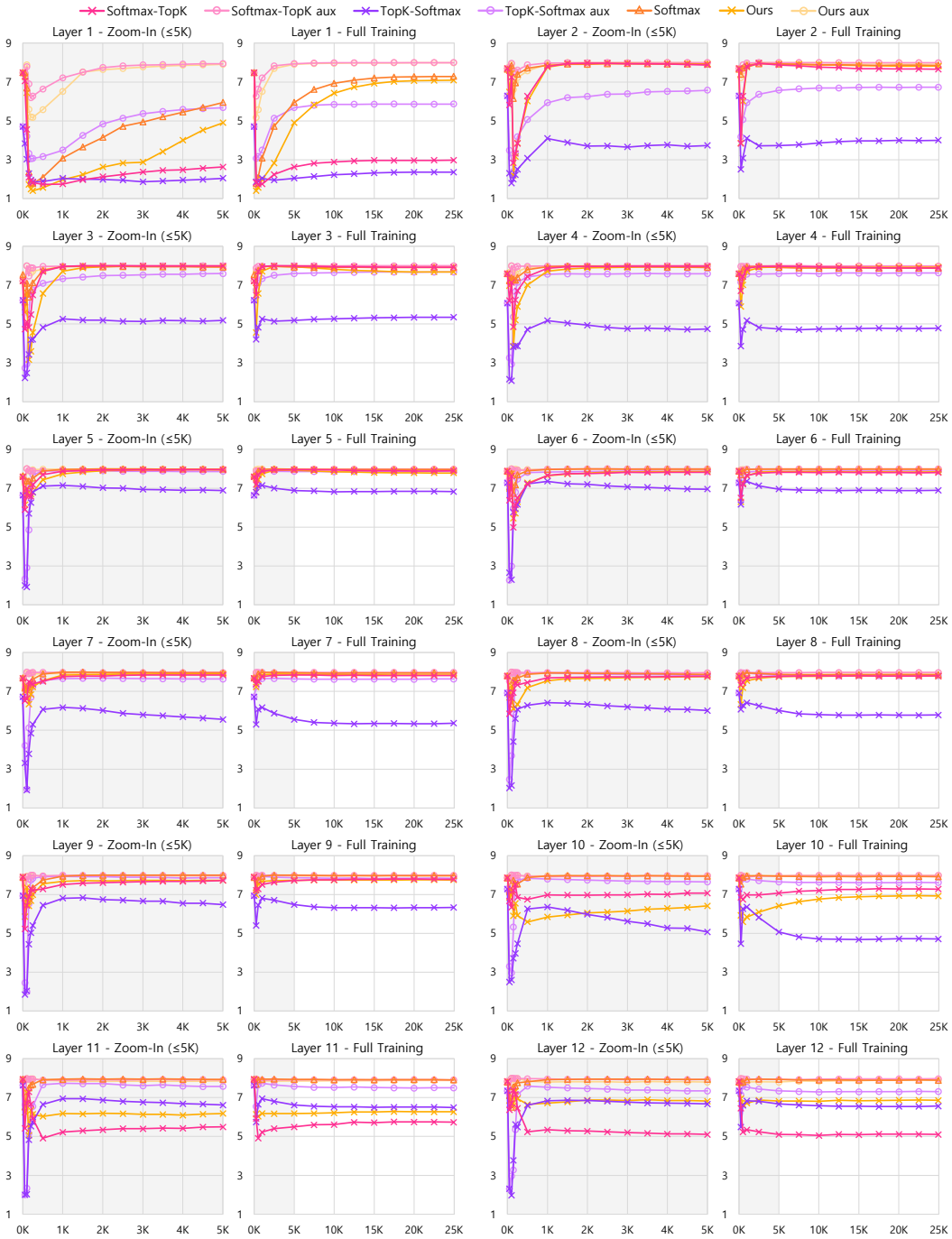

Figure 19: Layer-wise evolution of the effective number of experts for the 0.68B-scale models during training. We report all MoE layers across model variants. For each layer, the left panel zooms into the first 5K training steps, while the right panel shows the full training trajectory, with the same interval shaded in gray.

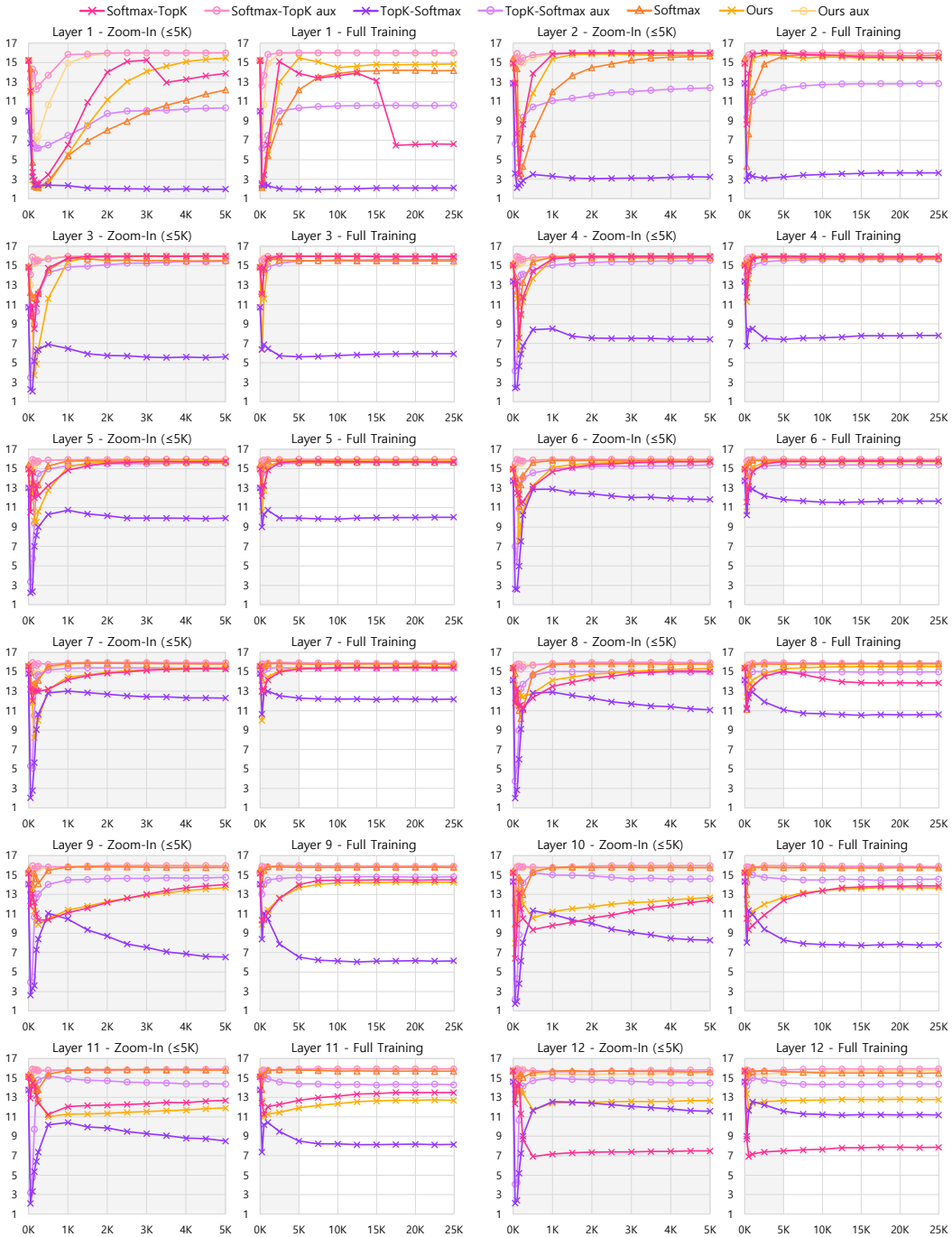

Figure 20: Layer-wise evolution of the effective number of experts for the 1.92B-scale models during training. We report all MoE layers across model variants. For each layer, the left panel zooms into the first 5K training steps, while the right panel shows the full training trajectory, with the same interval shaded in gray.

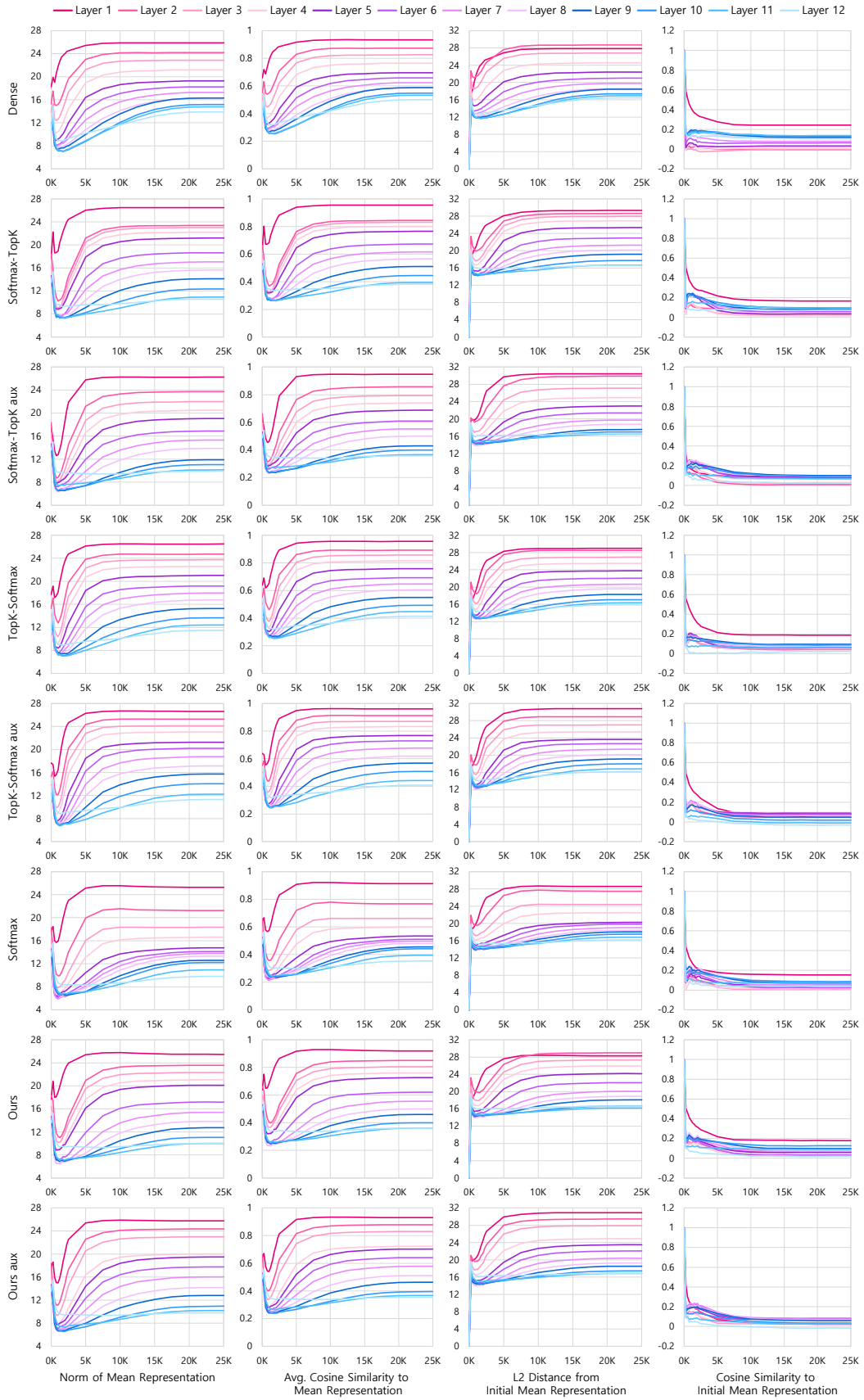

Figure 21: Evolution of hidden-representation geometry during training for the 0.68B-scale models.

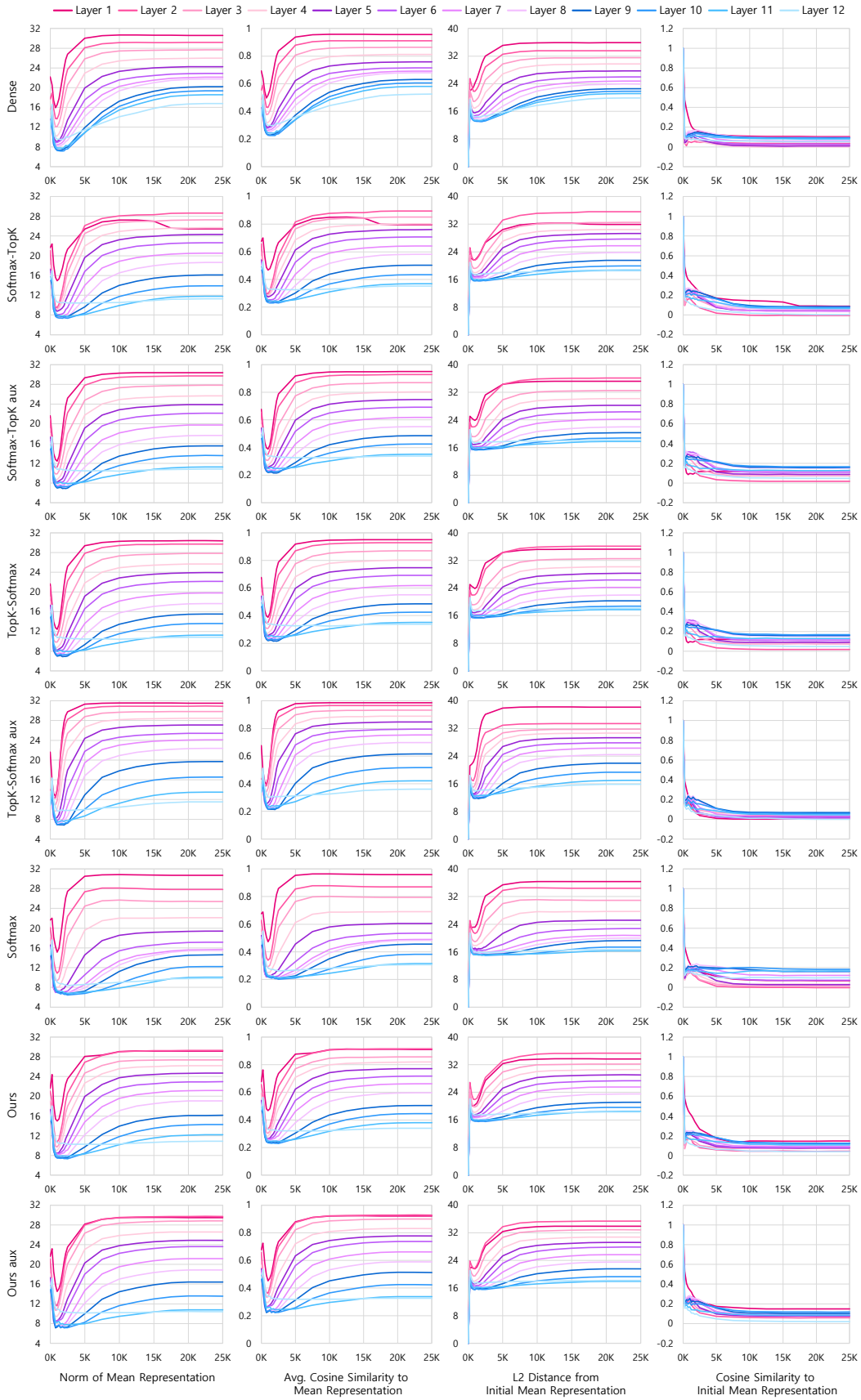

Figure 22: Evolution of hidden-representation geometry during training for the 1.92B-scale models.