GRADIENT REBALANCING: MITIGATING EXPOSURE-INDUCED ROUTING DRIFT IN MOE FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Pretrained mixture-of-experts (MoE) models develop expert specialization across different input patterns. Narrow downstream distributions shift the relative frequencies of these patterns, naturally leading to imbalanced expert utilization under the pretrained router. Enforcing uniform utilization during fine-tuning may therefore disrupt useful pretrained routing preferences. However, simply removing load balancing leaves exposure asymmetry in router training unaddressed. Frequently selected experts receive direct gradient signals from more tokens. Their router updates can therefore have greater influence on routing boundaries than those of rarely selected experts. This imbalance creates a mismatch with the more balanced training conditions of pretraining, which may amplify deviations from pretrained routing behavior. To address this issue, we propose *Gradient Rebalancing* (GR) to mitigate exposure-induced routing drift without imposing load-balancing constraints. GR targets each expert's *self-region tendency*, a first-order estimate of the change in its routing-region mass induced by its own router update. We decompose this tendency into token-level contributions to region expansion and contraction. GR selectively attenuates these contributions to bring each expert's estimated tendency close to zero, regardless of its exposure. This limits exposure-driven asymmetry in boundary adjustments, helping reduce drift from pretrained routing behavior. By preserving the signs of individual gradient signals, GR retains flexibility for task-driven router adaptation. Experiments across diverse tasks and datasets show that GR achieves the best overall performance among the evaluated methods and substantially reduces routing drift with only a small computational overhead.

1 INTRODUCTION

Sparse mixture-of-experts (MoE) models increase capacity by activating only a subset of experts per token (Shazeer et al., 2017; Fedus et al., 2022). During pretraining, the router learns to assign inputs to experts, while experts specialize in the inputs they receive. This joint training establishes a useful correspondence between input patterns and expert capabilities (Muennighoff et al., 2025). Pretrained routing therefore provides a valuable prior for fine-tuning. We aim to preserve this learned correspondence while allowing task-driven adaptation.

Load-balancing strategies commonly used during pretraining can conflict with this objective. When certain input patterns become more common downstream, the experts specialized in those patterns naturally receive more tokens. Imbalanced utilization can therefore reflect appropriate expert assignments. Nevertheless, both auxiliary-loss-based (Fedus et al., 2022) and bias-based balancing (DeepSeek-AI, 2024b) methods encourage more uniform utilization. Applying these interventions during fine-tuning can alter useful assignments to satisfy a load objective, disrupting the routing structure inherited from pretraining.

Simply removing load balancing, however, does not ensure that the pretrained routing behavior is preserved. Under imbalanced utilization, frequently selected experts receive direct task-gradient signals from more tokens for their router updates. Routing boundaries depend on relative expert scores, so an expert's region can change even when its own parameters remain fixed. A rarely selected expert may therefore receive little direct task feedback while its region is reshaped by updates to more frequently selected experts. This gives exposure asymmetry a role in boundary competition, beyond the task preferences expressed by individual gradient signals, as illustrated in Figure 1(I).

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.

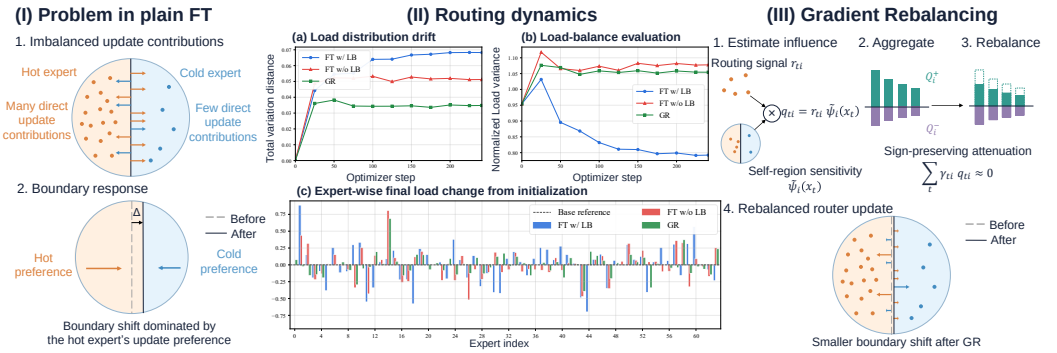


Figure 1: **Exposure-induced routing drift and Gradient Rebalancing (GR). (I)** Imbalanced exposure gives hot experts more direct update contributions, which can dominate boundary movement when hot and cold experts favor opposing changes. **(II)** Routing dynamics with auxiliary load balancing (FT w/ LB), without it (FT w/o LB), and with GR: (a) load-distribution drift from initialization, measured by total variation distance; (b) normalized load variance; and (c) final expert-wise load changes. In the plotted run, GR reduces distribution drift while retaining imbalanced utilization. **(III)** An illustration of the Gradient Rebalancing (GR) algorithm.

We refer to the resulting routing drift as *exposure-induced routing drift*. This motivates our central question: *How can we preserve useful pretrained routing behavior and mitigate exposure-induced routing drift while allowing the router to adapt to downstream tasks?*

To address this question, we propose *Gradient Rebalancing* (GR). GR selectively attenuates task gradients to bring each expert's *self-region tendency* close to zero (Figure 1(III)). This limits the net expansion or contraction driven by each expert's own update, curbing a channel through which exposure asymmetry can influence routing boundaries. It thereby helps preserve pretrained routing behavior without freezing router update (Figure 1(II)). Crucially, a near-zero net tendency does not require all token-level contributions to vanish: expansion and contraction contributions can offset one another. The router therefore remains trainable through selectively attenuated task gradients, retaining flexibility for downstream adaptation.

Our main contributions are:

- We identify exposure-induced routing drift as a challenge in MoE fine-tuning and introduce *self-region tendency* to quantify each expert's own contribution to routing-region mass changes.
- We propose GR, which brings the estimated self-region tendency close to zero through sign-preserving gradient attenuation, without imposing a target load distribution.
- We demonstrate that GR improves performance and reduces routing drift across downstream fine-tuning and continual learning tasks, with low computational overhead. Analyses of routing dynamics further substantiate its effectiveness.

2 Background and Related Work

2.1 Sparse MoE Routing

Consider an MoE layer with E experts. For an input representation $x \in \mathbb{R}^d$, a linear router contains one *router row* for each expert. The router row associated with expert i, parameterized by (w_i, β_i), computes its raw routing logit

$$s_i(x) = w_i^\top x + \beta_i. \tag{1}$$

The logits are converted into routing probabilities through

$$p_i(x) = \frac{\exp(s_i(x))}{\sum_{j=1}^{E} \exp(s_j(x))}. \tag{2}$$

A Top-k router selects the k highest-scoring experts, whose outputs are combined to produce the MoE output.

$$y(x) = \sum_{i \in \mathcal{A}(x)} p_i(x) f_i(x), \tag{3}$$

where $\mathcal{A}(x)$ denotes the indices of the k highest-scoring experts. We refer to the proportion of tokens for which expert i is selected as its *exposure*,

$$\text{Exposure}(i) = \frac{1}{t} \sum_{t} \mathbf{1}[i \in \mathcal{A}(x_t)]. \tag{4}$$

Exposure therefore measures how often an expert directly participates in downstream computation. When a token is routed to expert i, router row i receives a direct gradient signal from that expert's downstream computation. An unselected router row j may still receive a nonzero gradient through the softmax normalization, since its logit participates in the denominator. These gradients reflect competitive pressure induced by selected experts rather than direct evidence of the unselected expert's suitability for the token. Consequently, under imbalanced Top-k exposure, highly exposed experts repeatedly generate direct expert-conditioned signals and can exert greater influence on routing-region evolution.

2.2 MoE Fine-Tuning

We discuss related MoE fine-tuning research from two perspectives on routing. One retains pretraining-style load balancing to encourage broad expert utilization. The other treats pretrained routing as a useful prior and supports adaptation without load-balancing constraints. Within the first framework, *Advancing Expert Specialization for Better MoE* (Guo et al., 2025) supplements the balancing objective with orthogonality and variance losses to promote distinct expert assignments and more discriminative routing. ExpertCondenser (He et al., 2026) combines bias-based load regulation with gated condenser experts to consolidate knowledge from rarely activated experts. Complementary methods provide the router with feedback from experts excluded by Top-k selection. DenseMixer (Yao et al., 2025) computes inactive experts' outputs in an additional forward pass to estimate router gradients more accurately. Default MoE (Panda et al., 2025) instead replaces missing outputs with exponential moving averages of past expert outputs, providing denser router feedback while retaining sparse expert computation. These advances improve specialization and optimization. However, when combined with load-balancing interventions, they still encourage balanced utilization, which can disrupt useful routing preferences on narrow downstream distributions.

The second perspective treats learned routing patterns as a useful prior for downstream adaptation. Their value has been recognized in different settings: RA-MoE (Deng et al., 2026) uses task-relevant expert assignments to guide multilingual fine-tuning, while SafeMoE (Kim et al., 2025) preserves routing from an initial safety-aligned model through a routing-alignment objective. More directly related to our setting, concurrent work *Router Prior Bias* (Lee et al., 2026) advocates softly anchoring the pretrained router during post-training without reintroducing load balancing. These methods underscore the importance of retaining useful routing information, but differ in how they constrain its evolution. GR controls each expert's self-region tendency through sign-preserving gradient attenuation, without matching a fixed routing target or anchoring router parameters. In our experiments, GR better preserves pretrained routing behavior and achieves stronger downstream performance than the router anchor baseline.

3 Method

3.1 Self-Region Tendency

Consider an MoE layer with E experts and a fixed Top-k routing budget. For router input $x \in \mathbb{R}^d$, expert i receives a score $s_i(x; \theta_i)$, where θ_i denotes its router parameters and $\Theta = (\theta_1, \ldots, \theta_E)$. Assuming a router input representation distribution $p(x)$, we define the routing region $\mathcal{R}$ and its region mass V as

$$\mathcal{R}_i(\Theta) = \left\{ x \in \mathbb{R}^d : i \in \mathcal{A}_\Theta(x) \right\}, \qquad V_i(\Theta) = \int_{\mathcal{R}_i(\Theta)} p(x)\, dx. \tag{5}$$

Next, we study how an expert's own update changes its region mass while holding all other router parameters fixed. For a small update $\Delta\theta_i$, we define the *self-region tendency*

$$S_i = \nabla_{\theta_i} V_i^\top \Delta\theta_i. \tag{6}$$

S_i estimates the first-order change in expert i's routing-region mass induced by its own update, with all other router parameters held fixed. Its sign indicates a tendency toward self-expansion or self-contraction. GR therefore aims to make the self-region tendency of each expert approximately zero:

$$S_i \approx 0, \qquad i = 1, \ldots, E. \tag{7}$$

3.2 Token-Level Self-Region Influence

To realize this objective through task-gradient updates, we decompose the self-region tendency into token-level contributions. For token x_t and router row i, we define the routing signal as the negative derivative of the loss with respect to its logit:

$$r_{ti} = -\frac{\partial L}{\partial s_i(x_t)}. \tag{8}$$

Under SGD, the router parameter update is

$$\Delta\theta_i = -\eta \nabla_{\theta_i} L = \eta \sum_t r_{ti} \nabla_{\theta_i} s_i(x_t). \tag{9}$$

Substituting into Eq. (6) gives

$$S_i = \eta \sum_t r_{ti} \psi_i(x_t), \qquad \psi_i(x) = \nabla_{\theta_i} V_i^\top \nabla_{\theta_i} s_i(x). \tag{10}$$

Hence,

$$q_{ti} = r_{ti} \psi_i(x_t) \tag{11}$$

measures the token-level contribution to self-region tendency. Its sign indicates whether the task gradient of the token favors self-expansion or self-contraction.

Influence estimation. For a general router, obtaining an analytical expression for $\psi_i(x)$ is difficult. For a linear router, however, the influence reduces to a linear function whose coefficients can be estimated. Specifically, for

$$s_i(x) = w_i^\top x + \beta_i, \tag{12}$$

we have

$$\psi_i(x) = a_i^\top x + d_i, \qquad a_i = \nabla_{w_i} V_i, \quad d_i = \frac{\partial V_i}{\partial \beta_i}. \tag{13}$$

Since V_i is defined over the assumed representation distribution and its routing region is determined by discrete Top-k selection, these coefficients are difficult to compute directly. We therefore approximate the router-input distribution using the current batch and replace hard Top-k membership with a smooth function of the score margin. Let

$$m_{ti} = s_i(x_t) - \zeta_{ti}^{(k)}, \qquad \zeta_{ti}^{(k)} = \text{kth-largest}\left(\{s_j(x_t) \mid j \in \{1, \ldots, E\} \setminus \{i\}\}\right), \tag{14}$$

where $\zeta_{ti}^{(k)}$ is the k-th largest competing score. Define

$$\bar{V}_i = \frac{1}{N} \sum_{t=1}^{N} \sigma\left(\frac{m_{ti}}{T}\right), \qquad \kappa_{ti} = \frac{\partial}{\partial m_{ti}} \sigma\left(\frac{m_{ti}}{T}\right) = \frac{1}{T} \sigma\left(\frac{m_{ti}}{T}\right)\left[1 - \sigma\left(\frac{m_{ti}}{T}\right)\right]. \tag{15}$$

Here, σ denotes the sigmoid function and $T > 0$ is the temperature parameter. As $T \to 0^+$, $\bar{V}_i$ converges to the exact routing-region mass on the given batch. Holding competing router rows fixed, differentiating the surrogate yields

$$\hat{a}_i = \frac{1}{N} \sum_t \kappa_{ti} x_t, \qquad \hat{d}_i = \frac{1}{N} \sum_t \kappa_{ti}, \qquad \hat{\psi}_i(x_t) = \hat{a}_i^\top x_t + \hat{d}_i. \tag{16}$$

For bias-free routers, the bias terms are omitted.

3.3 GRADIENT REBALANCING

GR balances the positive and negative self-region contributions of each router row. Given

$$q_{ti} = r_{ti}\hat{\psi}_i(x_t), \tag{17}$$

we aggregate

$$Q_i^+ = \sum_t [q_{ti}]_+, \qquad Q_i^- = \sum_t [-q_{ti}]_+. \tag{18}$$

We attenuate the dominant side to balance the positive and negative self-region contributions. Specifically, we scale each token's contribution to the router-parameter update on the dominant side by

$$\alpha_i = \frac{\min(Q_i^+, Q_i^-)}{\max(Q_i^+, Q_i^-) + \epsilon}, \tag{19}$$

while leaving the other contributions unchanged. The resulting SGD update is

$$\Delta\widetilde{\theta}_i = \eta \sum_t \gamma_{ti} r_{ti} \nabla_{\theta_i} s_i(x_t), \qquad \gamma_{ti} = \begin{cases} \alpha_i, & q_{ti} > 0,\ Q_i^+ > Q_i^-, \\ \alpha_i, & q_{ti} < 0,\ Q_i^- > Q_i^+, \\ 1, & \text{otherwise.} \end{cases} \tag{20}$$

The resulting positive and negative contributions approximately cancel:

$$\sum_t \gamma_{ti} q_{ti} \approx 0, \tag{21}$$

so the resulting rebalanced SGD update has approximately zero first-order self-region tendency under the local surrogate.

Interpretation of GR. GR targets a specific mechanism through which imbalanced exposure affects router geometry. Frequently selected experts receive more direct gradient contributions, which can give their own parameter updates a stronger tendency to expand or contract their routing regions. GR attenuates the dominant side of these contributions so that the estimated self-region tendency satisfies $S_i \approx 0$ for each expert. This reduces the direct region-change advantage associated with imbalanced exposure. Because GR does not reverse the signs of individual task-gradient contributions, the router remains responsive to downstream training signals.

Controlling S_i is a deliberate choice rather than a constraint on the total change in region mass. To first order,

$$\Delta V_i \approx \underbrace{\nabla_{\theta_i} V_i^\top \Delta\theta_i}_{\text{self-induced}} + \underbrace{\sum_{j \neq i} \nabla_{\theta_j} V_i^\top \Delta\theta_j}_{\text{competition-induced}}. \tag{22}$$

The self-induced term measures the effect of expert i's own router update while holding the other router rows fixed. GR controls this term because it is directly affected by the expert's exposure. The competition-induced term reflects how updates to other experts change expert i's region through their relative scores. Leaving this term unconstrained allows routing boundaries to adjust during downstream adaptation; constraining the entire ΔV_i to zero would impose a substantially stronger restriction. Our experiments show reduced overall routing-region changes under GR. This empirical reduction is consistent with suppressing exposure-driven self-region tendencies while retaining competitive adaptation.

Adaptive optimizers. For AdamW (Loshchilov & Hutter, 2019), neutrality must be evaluated on the optimizer-induced parameter update rather than the raw gradient. We retain the same self-region objective and use a sign-preserving correction; the complete formulation is given in Appendix A.

3.4 COMPUTATIONAL COMPLEXITY

Let N, d and E denote the number of router inputs, representation dimension and number of experts, respectively. Compared to plain MoE training, GR additionally estimates the region derivatives of $\hat{\psi}$ and rebalances the router gradients. These operations require $O(NdE)$ computation, matching

Table 1: **Performance across downstream tasks.** Results are reported as percentages. The best and second-best results within each model are shown in bold and underlined, respectively.

Method	Mathematical Reasoning			Medical QA			Text-to-SQL		
	AddSub	GSM8K	AQuA	MedMCQA	MedQA	MMLU-Med.	WikiSQL	Spider	BIRD
DeepSeek-V2-Lite									
Frozen Router	71.14	53.83	41.73	49.80	52.30	64.83	83.90	23.40	7.43
FT w/o LB	73.16	54.59	38.19	50.20	52.30	64.00	85.48	23.02	5.35
FT w/ LB	72.41	53.22	39.37	50.00	50.10	62.83	82.40	19.73	4.17
Router Anchor	73.16	54.74	37.80	50.25	**53.30**	65.67	85.65	23.02	5.35
Spec. Loss	70.63	55.42	42.52	50.05	52.60	63.00	84.73	20.12	4.69
Expert Cond.	73.67	55.27	40.94	49.35	50.60	61.50	85.35	24.76	6.06
GR (Ours)	**75.44**	**58.38**	**45.67**	**52.15**	53.10	**67.33**	**88.83**	**25.82**	**8.60**
OLMoE									
Frozen Router	47.09	25.32	16.14	47.05	43.90	60.00	83.45	20.31	5.41
FT w/o LB	49.11	26.00	14.96	48.05	45.60	62.17	83.68	20.21	5.54
FT w/ LB	40.51	19.48	12.99	46.70	44.70	59.83	80.23	19.15	3.98
Router Anchor	48.35	25.47	16.93	48.90	46.10	61.83	83.45	20.50	5.08
Spec. Loss	48.10	23.50	18.11	48.30	44.70	61.33	82.33	19.34	5.15
Expert Cond.	48.35	25.47	16.93	47.85	41.30	61.83	83.13	**20.60**	5.67
GR (Ours)	**51.39**	**28.81**	**20.47**	**50.55**	**47.10**	**64.33**	**86.58**	19.83	**6.84**
Qwen3									
Frozen Router	95.19	91.05	72.83	72.45	80.00	86.83	88.15	58.32	24.38
FT w/o LB	94.43	91.21	74.41	73.00	80.20	87.00	88.30	59.48	26.47
FT w/ LB	92.41	86.88	68.11	71.70	71.20	85.67	86.25	56.48	25.88
Router Anchor	95.44	90.52	74.02	73.90	79.60	86.17	88.13	58.03	24.90
Spec. Loss	95.70	86.05	72.05	71.65	73.00	86.50	87.18	56.67	24.90
Expert Cond.	95.70	90.14	74.80	73.05	79.00	85.17	88.60	55.80	26.79
GR (Ours)	**96.46**	**92.72**	**77.95**	**75.35**	**83.50**	**88.33**	**90.65**	**60.64**	**28.81**

the asymptotic complexity of score computation in a linear router. GR introduces no additional parameters or expert forward/backward passes; its overhead consists only of router-scale matrix operations and reductions. Because MoE training is dominated by expert FFN computation, GR adds only a small fraction to the overall training cost, as confirmed empirically in Section 4.

4 Experiments

We evaluate GR across five complementary perspectives: downstream adaptation, routing preservation, mechanism validation, replay-free continual learning, and computational efficiency. Additional experiments are reported in the Appendix.

4.1 Experimental Setup

Models. We consider three pretrained MoE models: OLMoE-1B-7B-0924 (Muennighoff et al., 2025), DeepSeek-V2-Lite (DeepSeek-AI, 2024a), and Qwen3-30B-A3B-Base (Yang et al., 2025). Due to computational constraints, OLMoE is fully fine-tuned, whereas DeepSeek-V2-Lite and Qwen3 use rank-32 LoRA (Hu et al., 2022), with their router parameters fully trainable to allow direct routing adaptation. More details are provided in Appendix B.1.

Tasks. We evaluate our method on three challenging task families spanning 12 benchmarks. For the mathematical reasoning task, we fine-tune on Math14K (Hu et al., 2023) and evaluate on SingleEq (Koncel-Kedziorski et al., 2015), MultiArith (Roy & Roth, 2015), AddSub (Hosseini et al., 2014), GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), and AQuA (Ling et al., 2017). For medical question answering task, we train on MedMCQA (Pal et al., 2022) and evaluate on 2,000 examples sampled from its official validation split, MedQA-USMLE (Jin et al., 2021), and MMLU Medical (Hendrycks et al., 2020). For Text-to-SQL task, we train on WikiSQL (Zhong et al., 2017)

and evaluate on WikiSQL-test, Spider (Yu et al., 2018), and BIRD (Li et al., 2023). More details are provided in Appendix B.2.

Baselines. We compare GR with plain fine-tuning without load balancing (FT w/o LB), fine-tuning with an auxiliary load-balancing objective (FT w/ LB) (Fedus et al., 2022), and the frozen-router baseline. We additionally include Router Anchor (Lee et al., 2026), Specialization Loss (Guo et al., 2025) and Expert Condenser (He et al., 2026). More details are provided in Appendix B.3.

Implementation details. All methods use AdamW with a linear learning-rate schedule. The learning rate is 10^{-4} for DeepSeek-V2-Lite and Qwen3-30B-A3B, and 2×10^{-5} for OLMoE-1B-7B; router parameters use the same learning rate as the other trainable parameters. Each model is trained for one epoch over the processed training set. More details are provided in Appendix B.4.

4.2 Main Results

Table 1 reports three mathematical benchmarks in the main text due to space constraints; results for all six benchmarks appear in Table 6. GR achieves the highest accuracy in 25 of the 27 model–benchmark combinations in the main table. Averaged equally across these combinations, it exceeds Router Anchor, the strongest baseline overall, by 2.57 percentage points.

FT w/ LB generally performs poorly, with particularly large drops for OLMoE. This pattern is consistent with load balancing disrupting useful pretrained routing preferences when downstream expert utilization is imbalanced. Specialization Loss and Expert Condenser improve on standard FT w/ LB in many settings, but neither closes the overall gap to GR. Frozen Router does not outperform the stronger adaptive baselines overall, suggesting that fixing the router can limit downstream adaptation. FT w/o LB and Router Anchor are the strongest baselines, although their relative performance varies across models and tasks. GR improves on both while permitting task-driven router updates and controlling their exposure-dependent effect on routing regions.

On a commonsense task, however, FT w/ LB outperforms GR (Appendix B.12). This result indicates that GR is not uniformly preferable across fine-tuning settings and suggests that its benefit may depend on the extent of downstream load imbalance. This observation is consistent with our original motivation of addressing imbalanced training during downstream adaptation.

4.3 Routing Preservation and Self-Region Control

We evaluate routing behavior on a fixed set of examples shared across all checkpoints and methods. Our primary diagnostics are token-level KL divergence between the pretrained and fine-tuned router distributions, Top-k expert overlap, and total variation distance between their aggregate expert-load distributions. These metrics measure changes in routing behavior rather than router parameters, aligning with our goal of preserving pretrained routing behavior. The definitions of these metrics are provided in Appendix B.9

Table 2: Routing diagnostics on Math14K. **(a)** Comparison with baseline methods. **(b)** Comparison with GR ablation variants.

(a) Baseline comparison				**(b) Ablation comparison**			
Method	**KL ↓**	**Overlap ↑**	**TV ↓**	**Method**	**KL ↓**	**Overlap ↑**	**TV ↓**
Frozen Router	**1.6163**	**0.8859**	**0.0461**	FT w/o LB	1.9495	0.7957	0.0838
FT w/o LB	1.9495	0.7957	0.0838	GR w/o ψ	1.7355	0.8357	0.0511
FT w/ LB	2.2101	0.7582	0.0908	Layer-global	1.9766	0.7907	0.0871
Router Anchor	1.8634	0.8278	0.0755	Projection	1.6614	0.8576	0.0493
Spec. Loss	2.7804	0.6980	0.1091	Norm-matched FT	1.9464	0.8052	0.0813
				Exposure scaling	1.9409	0.7964	0.0840
GR (Ours)	1.6477	0.8826	0.0474	**GR (Ours)**	**1.6477**	**0.8826**	**0.0474**

Note: KL, Overlap, and TV denote router KL divergence, Top-6 expert overlap, and load total variation distance, respectively.

Table 2(a) shows that FT w/ LB exhibits greater routing drift than FT w/o LB across all three diagnostics. Specialization Loss exhibits even greater drift, consistent with its additional router-variance and expert-orthogonality objectives placing further pressure on pretrained routing. Parameter constraints offer only partial protection: Frozen Router still changes its assignments as input representations evolve, while Router Anchor reduces drift relative to FT w/o LB but remains substantially less effective than GR.

Among methods that update the router, GR achieves the lowest router KL and load TV, and the highest Top-k overlap. Relative to FT w/o LB, it reduces router KL by 15.5% and load TV by 43.4%, while increasing overlap by 8.69 percentage points. Its routing diagnostics are close to those of Frozen Router, yet GR retains the ability to adapt the router and achieves stronger downstream performance. We exclude Expert Condenser from this comparison because it replaces cold experts with always-active condenser experts and selects only $k - c$ sparse experts per token, making its routing decisions structurally different from those of the other methods.

4.4 Ablation Study

We perform the principal ablation on DeepSeek-V2-Lite with Math14K. In addition to FT w/o LB and GR, we evaluate:

1. **GR w/o ψ**: balance positive and negative logit signals without accounting for each token's influence on routing-region mass;
2. **Layer-global GR**: use one dominant side and attenuation coefficient for the entire MoE layer;
3. **Projection GR**: project each router-row parameter update onto the subspace orthogonal to the estimated region-gradient direction;
4. **Norm-matched FT w/o LB**: retain the FT w/o LB direction while matching the norm of the GR update;
5. **Exposure-only scaling**: scale each router-row gradient based on exposure alone.

Table 3: Downstream accuracy of GR variants on Math14K using DeepSeek-V2-Lite. The best and second-best results are shown in bold and underlined, respectively.

Method	SingleEq	MultiArith	AddSub	GSM8K	SVAMP	AQuA	Avg.
FT w/o LB	88.56	88.10	73.16	54.59	62.90	38.19	67.58
GR w/o ψ	89.55	**88.79**	<u>74.94</u>	55.04	63.40	<u>42.13</u>	68.98
Layer-global	89.55	87.59	71.65	53.83	63.90	36.61	67.19
Projection	**90.14**	87.41	73.92	<u>56.86</u>	64.50	41.34	<u>69.03</u>
Norm-matched FT	88.17	87.76	73.16	54.21	<u>64.60</u>	35.43	67.22
Exposure scaling	88.76	87.07	71.14	54.97	63.10	39.76	67.47
GR (Ours)	<u>89.74</u>	<u>88.62</u>	**75.44**	**58.38**	**65.30**	**45.67**	**70.53**

As shown in Table 3, GR achieves the highest average accuracy of 70.53%, exceeding FT w/o LB by 2.95 percentage points. Layer-global GR, Norm-matched FT, and Exposure scaling offer little or no improvement over FT w/o LB. Their routing diagnostics in Table 2(b) also remain close to plain fine-tuning. These results suggest that neither a shared correction across experts, a change in gradient magnitude alone, nor scaling by exposure count adequately controls routing drift.

The two variants that account for the direction of routing change are more effective. GR w/o ψ balances the raw score signals r_{ti} and improves average accuracy to 68.98%, while substantially reducing routing drift. This is plausible because increasing or decreasing an expert's score is often associated with self-expansion or self-contraction of its routing region. Nevertheless, the complete GR formulation improves accuracy by another 1.55 points and better preserves routing, indicating that score signals alone do not capture each token's influence on region mass. GR and Projection method yield similar routing diagnostics, but GR achieves 1.50 percentage points higher average accuracy. One possible explanation is that scaling selected token contributions preserves useful task-gradient signals more effectively than projecting the router gradient. Together, the results favor controlling self-region tendency through expert-specific, sign-preserving attenuation.

4.5 Continual Learning

We further evaluate GR in a replay-free continual-learning setting using DeepSeek-V2-Lite on NuminaMath-1.5 (AI-MO, 2025). Using its native domain annotations, we construct a seven-stage sequence comprising Algebra, Geometry, Number Theory, Combinatorics, Calculus, Inequalities, and Logic and Puzzles, with one domain introduced at each stage. We report cumulative average accuracy (CAA), final average accuracy (FAA), trajectory-averaged forgetting, and routing forgetting. The complete protocol and metric definitions are provided in Appendix B.7.

Table 4: Replay-free continual-learning results on seven NuminaMath-1.5 domains using DeepSeek-V2-Lite. All values are reported as percentages.

Method	CAA ↑	FAA ↑	Forgetting ↓	Routing Forgetting ↓
Frozen Router	17.03	17.64	1.00	**4.75**
FT w/o LB	<u>17.57</u>	<u>18.73</u>	<u>0.64</u>	11.84
FT w/ LB	15.84	16.35	1.15	14.05
Router Anchor	17.34	18.27	0.82	11.03
GR (Ours)	**20.85**	**21.14**	**0.35**	<u>6.26</u>

Recent work (Zhao et al., 2026; Xie et al., 2026) has identified routing forgetting—changes in routing behavior on previous-task data after learning new tasks—as a key challenge in MoE continual learning. By mitigating exposure-induced routing drift while retaining the router's ability to adapt to sequential tasks, GR provides a natural means of addressing this problem. As shown in Table 4, GR achieves the highest CAA and FAA, supporting this hypothesis. Although freezing the router yields the lowest routing forgetting, this rigid constraint limits adaptation to new tasks and consequently produces a less favorable overall accuracy–stability trade-off.

4.6 Efficiency Analysis

We compare plain fine-tuning and GR on a single NVIDIA RTX PRO 6000 GPU (96 GB), using a batch size of 32 without gradient accumulation. As shown in Table 5, GR introduces little computational overhead. Compared with standard fine-tuning, GR increases the training time from 32.14 to 33.03 minutes, corresponding to an overhead of only 2.77%, while maintaining a comparable throughput (6.514 vs. 6.693 samples/s). These results show that the additional router-scale computations required by GR account for only a small fraction of the overall MoE fine-tuning cost.

Table 5: Training efficiency on Math14K.

Method	Time	Overhead	Samples/s
FT w/o LB	32.14 min	–	6.693
GR (Ours)	33.03 min	+2.77%	6.514

5 Conclusion

We study how imbalanced expert exposure affects router adaptation during MoE fine-tuning. Downstream load imbalance can reflect useful pretrained specialization, yet it also gives frequently selected experts more direct task-gradient contributions. Their updates may consequently exert greater influence on routing boundaries, contributing to drift from pretrained routing behavior. We address this issue with Gradient Rebalancing (GR). GR uses each expert's self-region tendency as a local control target and selectively attenuates token-level gradient contributions to bring the estimated tendency close to zero. This limits self-induced region expansion or contraction without imposing a target load distribution or preventing competitive, task-driven changes in routing. The same principle extends to adaptive optimizers. Across downstream and continual-learning evaluations, GR improves performance and reduces routing drift with little computational overhead. These results suggest that controlling the effect of imbalanced exposure on router updates can help preserve pretrained routing structure while allowing downstream adaptation.

AI USE STATEMENT

We used generative AI tools to discuss and refine research hypotheses, provide feedback on the formulation of our method and the experimental design, and assist in interpreting experimental results. We also used these tools to draft and edit portions of the manuscript, improve clarity and organization, and format tables and figures. The authors reviewed AI-assisted text, mathematical arguments, and reported results against their derivations and experimental records, and made the final decisions about the method, experiments, and conclusions. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

AI-MO. NuminaMath-1.5 dataset. Hugging Face dataset repository, 2025. URL https://huggingface.co/datasets/AI-MO/NuminaMath-1.5.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. arXiv preprint arXiv:2405.04434, 2024a. URL https://arxiv.org/abs/2405.04434.

DeepSeek-AI. DeepSeek-V3 technical report. arXiv preprint arXiv:2412.19437, 2024b. URL https://arxiv.org/abs/2412.19437.

Guanzhi Deng, Kuan Wu, Haibo Wang, Shing Yin Wong, Sichun Luo, and Linqi Song. Routing-aligned fine-tuning for multilingual downstream tasks in mixture-of-experts models. arXiv preprint arXiv:2605.28306, 2026. URL https://arxiv.org/abs/2605.28306.

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.

Hongcan Guo, Haolang Lu, Guoshun Nan, Bolun Chu, Jialin Zhuang, Yuan Yang, Wenhao Che, Xinye Cao, Sicong Leng, Qimei Cui, and Xudong Jiang. Advancing expert specialization for better MoE. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1629. URL https://papers.nips.cc/paper_files/paper/2025/hash/4598de7d243d528e38eb0c5d8155fb52-Abstract-Conference.html.

Haoze He, Xingyuan Ding, Xuan Jiang, Xinkai Zou, Alex Cheng, Yibo Zhao, Juncheng Billy Li, and Heather Miller. Preserving long-tailed expert information in mixture-of-experts tuning. In *Proceedings of the Third Conference on Language Modeling*, 2026. URL https://arxiv.org/abs/2604.23036.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. arXiv preprint arXiv:2009.03300, 2020. URL https://arxiv.org/abs/2009.03300.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 523–533, 2014. doi: 10.3115/v1/D14-1058. URL https://aclanthology.org/D14-1058/.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5254–5276. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.319. URL https://aclanthology.org/2023.emnlp-main.319/.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021. doi: 10.3390/app11146421. URL https://doi.org/10.3390/app11146421.

Jaehan Kim, Minkyoo Song, Seungwon Shin, and Sooel Son. Defending MoE LLMs against harmful fine-tuning via safety routing alignment. arXiv preprint arXiv:2509.22745, 2025. URL https://arxiv.org/abs/2509.22745.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597, 2015. doi: 10.1162/tacl_a_00160. URL https://aclanthology.org/Q15-1042/.

Jaedeok Lee, Keonwoo Kim, Dongyoon Han, Sangdoo Yun, Yera Choi, and Haanju Yoo. Router prior bias: Preserving base routing structure in MoE post-training. arXiv preprint arXiv:2609.08115, 2026. URL https://arxiv.org/abs/2609.08115.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM already serve as a database interface? a BIg bench for large-scale database grounded text-to-SQLs. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/83fc8fab1710363050bbd1d4b8cc0021-Abstract-Datasets_and_Benchmarks.html.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pp. 158–167, 2017. URL https://aclanthology.org/P17-1015/.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah Smith, Pang Wei Koh, Amanpreet Singh, and Hanna Hajishirzi. OLMoE: Open mixture-of-experts language models. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/9b224ace8963c9385ad5e2b5c9039b97-Abstract-Conference.html.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. MedMCQA: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pp. 248–260, 2022. URL https://proceedings.mlr.press/v174/pal22a.html.

Ashwinee Panda, Vatsal Baherwani, Zain Sarwar, Benjamin Thérien, Sambit Sahu, Tom Goldstein, and Supriyo Chakraborty. Dense backpropagation improves training for sparse mixture-of-experts. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-5619. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/f69707de866eb0805683d3521756b73f-Abstract-Conference.html.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094, 2021. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168/.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1743–1752, 2015. doi: 10.18653/v1/D15-1202. URL https://aclanthology.org/D15-1202/.

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. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454/.

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

Zhen-Hao Xie, Jun-Tao Tang, Yu-Cheng Shi, Han-Jia Ye, De-Chuan Zhan, and Da-Wei Zhou. Same: Stabilized mixture-of-experts for multimodal continual instruction tuning. In *ICML*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025. URL https://arxiv.org/abs/2505.09388.

Feng Yao, Junxia Cui, Ruohan Zhang, Liyuan Liu, Shibo Hao, Li Zhang, Chengyu Dong, Shuohang Wang, Yelong Shen, Jianfeng Gao, and Jingbo Shang. DenseMixer: Improving MoE post-training with precise router gradient. Feng Yao's Notion, June 2025. URL https://fengyao.notion.site/moe-posttraining.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale

human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3911–3921, 2018. doi: 10.18653/v1/D18-1425. URL https://aclanthology.org/D18-1425/.

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. Association for Computational Linguistics, 2019. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Chongyang Zhao, Mingsong Li, Haodong Lu, and Dong Gong. On token's dilemma: Dynamic moe with drift-aware token assignment for continual learning of large vision language models. *arXiv preprint arXiv:2603.27481*, 2026.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2SQL: Generating structured queries from natural language using reinforcement learning. arXiv preprint arXiv:1709.00103, 2017. URL https://arxiv.org/abs/1709.00103.

A AdamW-Compatible Gradient Rebalancing

The main text presents GR under SGD, where each token's contribution to self-region tendency is linear in its routing signal. Under AdamW, momentum and adaptive preconditioning change the parameter update induced by the current gradient. We therefore evaluate self-region tendency on the candidate AdamW update. For clarity, we consider a bias-free linear router; the biased case follows by adding a bias coordinate to the router row and its region-influence direction.

A.1 AdamW Update as the Controlled Object

Let t index optimizer steps and n index tokens in the current batch. The descent direction for router row i is

$$g_i = \sum_n r_{ni} x_n, \qquad r_{ni} = -\frac{\partial \mathcal{L}}{\partial s_i(x_n)}. \tag{23}$$

We use the descent convention for Adam's moments throughout this section. Given previous moments $m_{i,t-1}$ and $v_{i,t-1}$, a candidate descent direction g induces

$$m_{i,t}(g) = \beta_1 m_{i,t-1} + (1-\beta_1) g, \tag{24}$$

$$v_{i,t}(g) = \beta_2 v_{i,t-1} + (1-\beta_2) g^{\odot 2}, \tag{25}$$

$$\hat{m}_{i,t}(g) = \frac{m_{i,t}(g)}{1-\beta_1^t}, \qquad \hat{v}_{i,t}(g) = \frac{v_{i,t}(g)}{1-\beta_2^t}. \tag{26}$$

With zero weight decay and omitting the common learning-rate factor, the resulting router parameter increment is

$$u_{i,t}(g) = \frac{\hat{m}_{i,t}(g)}{\sqrt{\hat{v}_{i,t}(g)} + \epsilon}, \tag{27}$$

where division and square root act elementwise.

Let $\hat{a}_{i,t}$ be the batch estimate of the region-influence direction from Eq. (16). In training, we smooth this estimate across steps:

$$\bar{a}_{i,t} = \gamma \bar{a}_{i,t-1} + (1-\gamma)\hat{a}_{i,t}. \tag{28}$$

The optimizer-consistent self-region target is

$$\bar{a}_{i,t}^{\top} u_{i,t}(g) \approx 0. \tag{29}$$

The EMA reduces fluctuations in the estimated direction across batches. Unlike the SGD target, Eq. (29) also depends on the optimizer's previous moments.

A.2 SIGN-PRESERVING CORRECTION

We use a reference preconditioner based on the previous second moment to assign tokens to two groups:

$$\bar{D}_{i,t} = \operatorname{diag}\left(\frac{1}{\sqrt{v_{i,t-1}} + \epsilon}\right), \qquad \rho_{ni} = r_{ni}\bar{a}_{i,t}^{\top}\bar{D}_{i,t}x_n. \tag{30}$$

The corresponding descent contributions are

$$g_i^{+} = \sum_{n:\rho_{ni}\geq 0} r_{ni}x_n, \qquad g_i^{-} = \sum_{n:\rho_{ni}<0} r_{ni}x_n, \qquad g_i = g_i^{+} + g_i^{-}. \tag{31}$$

Assigning a zero proxy to the positive group makes the partition exhaustive without affecting its interpretation.

To choose which group to attenuate, evaluate their approximate contributions using the preconditioner induced by the original direction g_i:

$$D_{i,t}(g_i) = \operatorname{diag}\left(\frac{1}{\sqrt{\hat{v}_{i,t}(g_i)} + \epsilon}\right), \qquad C_i^{\pm} = \bar{a}_{i,t}^{\top}D_{i,t}(g_i)g_i^{\pm}. \tag{32}$$

We select the group with the larger absolute contribution,

$$h_i = \begin{cases} g_i^{+}, & |C_i^{+}| \geq |C_i^{-}|, \\ g_i^{-}, & |C_i^{+}| < |C_i^{-}|, \end{cases} \tag{33}$$

and form the candidate direction

$$\tilde{g}_i(\alpha) = g_i - (1-\alpha)h_i, \qquad \alpha \in [0,1]. \tag{34}$$

Thus, GR attenuates one group of existing token contributions without reversing their routing signals.

A.3 SELF-CONSISTENT NEUTRALITY

The candidate direction changes both Adam moments. We therefore evaluate its effect through the candidate optimizer update:

$$F_i(\alpha) = \bar{a}_{i,t}^{\top}u_{i,t}(\tilde{g}_i(\alpha))\,. \tag{35}$$

We measure neutrality using the normalized residual

$$\mathrm{Res}_i(\alpha) = \frac{|F_i(\alpha)|}{\|\bar{a}_{i,t}\|_2\|u_{i,t}(\tilde{g}_i(\alpha))\|_2 + \epsilon_{\mathrm{GR}}}. \tag{36}$$

An update satisfies the neutrality tolerance when $\mathrm{Res}_i(\alpha) \leq \tau_{\mathrm{neut}}$. This criterion measures approximate orthogonality between the estimated region-gradient direction and the candidate AdamW update.

If $\mathrm{Res}_i(1) \leq \tau_{\mathrm{neut}}$, we retain the original gradient. Otherwise, when $F_i(0)F_i(1) \leq 0$, we search for a root of F_i by bisection, stopping once the normalized residual satisfies the tolerance or the iteration limit is reached. If the limit is reached without satisfying the tolerance, we select the evaluated candidate with the smallest normalized residual, including the interval endpoints. We then pass the gradient corresponding to $\tilde{g}_i(\alpha_i^{\star})$ to AdamW.

If the endpoints do not bracket a root, we retain the original gradient. This preserves the sign structure of the task signal without extending the attenuation coefficient beyond $[0,1]$.

A.4 RELATION TO SGD

With zero momentum and a constant scalar preconditioner, $u_{i,t}(g)$ is proportional to g. The AdamW construction then reduces to balancing the two groups' self-region contributions as in the SGD formulation. The control target remains near-zero self-region tendency; AdamW changes how a candidate gradient maps to the parameter update.

Algorithm 1 AdamW-Compatible Gradient Rebalancing

Require: Batch $\{x_n\}_{n=1}^{N}$; previous Adam moments; EMA directions $\{\bar{a}_{i,t-1}\}$
Compute task loss and score signals $r_{ni} = -\partial\mathcal{L}/\partial s_i(x_n)$
for each router row i **do**
  Estimate $\hat{a}_{i,t}$ using Eq. (16); update $\bar{a}_{i,t}$ using Eq. (28)
  Compute g_i and partition it into g_i^+, g_i^- using Eq. (31)
  Select the dominant contribution h_i using Eq. (33)
  Define $\tilde{g}_i(\alpha) = g_i - (1-\alpha)h_i$ and $F_i(\alpha) = \bar{a}_{i,t}^{\top} u_{i,t}(\tilde{g}_i(\alpha))$
  if $F_i(1)$ satisfies the neutrality tolerance **then**
    $\alpha_i \leftarrow 1$
  else if $F_i(0)F_i(1) \le 0$ **then**
    Find α_i with $F_i(\alpha_i) \approx 0$ by bisection
  else
    $\alpha_i \leftarrow 1$
  end if
  Replace the router-row loss gradient with $-\tilde{g}_i(\alpha_i)$
end for
Apply the standard AdamW step

A.5 Algorithm

Algorithm 1 summarizes one training step. The function $u_{i,t}(g)$ includes the candidate first and second moments defined above.

B Experimental Details and Additional Results

B.1 Models and Trainable Parameters

We initialize our experiments from local copies of `deepseek-ai/DeepSeek-V2-Lite`, `Qwen/Qwen3-30B-A3B-Base`, and `allenai/OLMoE-1B-7B-0924`. All three are base checkpoints rather than instruction-tuned variants. We fully fine-tune OLMoE-1B-7B. For DeepSeek-V2-Lite and Qwen3-30B-A3B, we train the routers and LoRA adapters with rank 32, scaling parameter 32, and dropout 0.1, while freezing all other model parameters.

For DeepSeek-V2-Lite, LoRA targets `q_proj`, `kv_a_proj_with_mqa`, `kv_b_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`. For Qwen3-30B-A3B, it targets `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`. Matching modules in all layers receive LoRA adapters. Sparse router matrices are trained directly rather than through LoRA, except in the Frozen Router baseline.

GR modifies the gradients of every trainable Top-k router in the sparse MoE layers: 26 routers in DeepSeek-V2-Lite, 48 in Qwen3-30B-A3B, and 16 in OLMoE-1B-7B. It does not directly modify expert gradients, dense feed-forward layers, attention parameters, or projections such as `gate_proj` that do not determine Top-k expert assignments. DeepSeek-V2-Lite's always-active shared experts do not participate in this Top-k routing and are likewise outside GR's scope.

B.2 Datasets and Preprocessing

Mathematical reasoning. We train on Math-14K and evaluate on SingleEq, MultiArith, AddSub, GSM8K, SVAMP, and AQuA. The archived training file contains 13,921 records. After removing invalid records, records with conflicting answers, and examples whose responses do not explicitly contain the numerical target, 13,910 examples remain. We split these deterministically into 12,910 training and 1,000 validation examples. All three models use the same processed training set, and no training examples are subsequently removed by the sequence-length filter.

The evaluation sets contain 507 SingleEq, 580 MultiArith, 395 AddSub, 1,319 GSM8K, 1,000 SVAMP, and 254 AQuA examples. For each example, the task input is the original problem text, without an additional mathematical-reasoning instruction. In AQuA, the answer choices are already

included in the problem text. DeepSeek-V2-Lite serializes the input with its tokenizer's template; Qwen3-30B-A3B-Base uses its native chat template with `enable_thinking=False`. Because OLMoE-1B-7B-0924 has no chat template, we append a blank line followed by `Answer:` to the problem text.

Medical question answering. We construct a 30,000-example training subset from the MedMCQA training split. Before sampling, we remove duplicate records with identical labels and question groups with conflicting labels. Sampling is stratified by subject, target answer, and choice type. A separate 2,000-example validation set is drawn from the remaining MedMCQA training data. Evaluation uses 2,000 sampled examples from the official MedMCQA validation split, 1,000 examples from the four-option MedQA-USMLE test split, and 100 test examples from each of six MMLU Medical subjects. We refer to the sampled MedMCQA validation examples as the *MedMCQA evaluation set* to distinguish them from our training-validation set. An audit based on canonicalized questions and choices found no overlap between the training set and these evaluation sets.

We format each medical question with the following task prompt, where the placeholders are replaced by the question and its four answer choices:

```
Answer the following medical multiple-choice question.
Select exactly one option from A, B, C, and D.

Question: {question}
A. {choice_A}
B. {choice_B}
C. {choice_C}
D. {choice_D}

Return your answer exactly as: The correct answer is X.
```

The training response is `The correct answer is X.`, with `X` replaced by the target option.

Text-to-SQL. We use WikiSQL as the training source. From its 56,355-example training split, we select 14,000 examples by stratifying on the aggregation operator, number of conditions, and multiset of condition operators. We similarly select 2,000 examples from the WikiSQL development split for validation and 4,000 examples from its test split for evaluation. We additionally evaluate on the complete Spider development set (1,034 examples) and BIRD development set (1,534 examples). An exact comparison of the complete question-and-schema prompts found no overlap between the WikiSQL training subset and these evaluation sets. Some WikiSQL questions have identical wording but refer to different tables; these are retained because their complete inputs differ.

All three models receive the same task prompt, with the database identifier, schema, and question substituted for the placeholders:

```
Database: {database_id}

Schema:
{CREATE TABLE statements}

Question: {natural_language_question}

Return only the SQL query.
```

The training response contains only the target SQL query, without an explanation or a Markdown code block. Model-specific input serialization is applied outside the task prompt.

B.3 Baselines and Hyperparameter Selection

Within each model–task setting, methods share the same initial checkpoint, processed data, data order, prompts, optimization settings, sequence limits, decoding configuration, and evaluation procedure. They differ in their specified routing or expert-training intervention.

Baselines. FT w/o LB performs standard fine-tuning with the auxiliary load-balancing objective disabled. The task loss updates the router and all other designated trainable parameters. FT w/ LB follows the same procedure but enables the model's auxiliary load-balancing objective. Following the original pretraining settings, its coefficient is 0.001 for DeepSeek-V2-Lite and Qwen3-30B-A3B, and 0.01 for OLMoE-1B-7B.

Frozen Router excludes router parameters from optimization while updating the remaining designated parameters. It measures how much routing changes as the representations received by a fixed router evolve. Router Anchor keeps the router trainable but adds an L2 penalty on its deviation from the pretrained parameters, with a coefficient of 0.001. In our experiments, this coefficient provides a balance between routing stability and evaluation accuracy.

Specialization Loss adds auxiliary load-balancing, router-variance, and expert-output orthogonality objectives to the task loss (Guo et al., 2025). Following the original paper, we set their coefficients to 0.001, 0.0002, and 0.0001, respectively. Expert Condenser introduces two condenser experts per layer to incorporate information from infrequently selected experts and regulates load through routing biases (He et al., 2026). GR disables the auxiliary load-balancing objective and adjusts router gradients as described in Section 3.

B.4 Training and GR Implementation

All main experiments use PyTorch AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, and zero weight decay. The learning rate is 10^{-4} for DeepSeek-V2-Lite and Qwen3-30B-A3B, and 2×10^{-5} for OLMoE-1B-7B. Router parameters use the same learning rate as the other trainable parameters within each model. We use a linear learning-rate decay with a warmup ratio of 0.03, BF16 precision, and gradient checkpointing. Gradient clipping is disabled. Training lasts one epoch over the processed training set.

Each training run uses a single NVIDIA RTX PRO 6000 GPU with 96 GB of memory and a batch size of 32, without gradient accumulation or distributed training. The per-device batch size and effective batch size are therefore both 32.

Maximum training sequence lengths are 2048, 512, and 1024 tokens for mathematical reasoning, medical QA, and Text-to-SQL, respectively. The corresponding evaluation input limits are 2048, 1152, and 4096 tokens, and the generation limits are 1536, 32, and 1024 tokens. For OLMoE-1B-7B on Text-to-SQL, the evaluation input limit is 3072 tokens so that the input and maximum generation fit its 4096-token context window.

GR uses a region-surrogate temperature of $T = 0.05$ and a numerical stability constant of $\epsilon_{\mathrm{GR}} = 10^{-12}$. The region-gradient EMA has decay 0.9 and is updated every optimizer step. We set the relative neutrality tolerance in Eq. (36) to $\tau_{\mathrm{neut}} = 0.02$ and allow at most 12 bisection iterations.

For AdamW, the correction is calculated with respect to the subsequent parameter update. The calculation includes the previous first and second moments, the candidate gradient's contribution to the current second moment, and Adam bias correction. GR adjusts the gradient before the standard AdamW step, which then updates the parameters and optimizer moments. GR aggregates token-level contributions over the full training batch and applies one correction per router row before each optimizer step. In the DeepSeek-V2-Lite Math14K run, the no-feasible-root fallback described in Appendix A.3 was invoked for only 0.0167% of router-row update attempts. Thus, the absence of a feasible sign-preserving correction was rare in this setting.

B.5 Evaluation Protocol

All models use greedy decoding without sampling.

Mathematical reasoning and medical QA. For mathematical reasoning, automatic answer extraction compares normalized final numerical answers with their targets; AQuA instead compares the final A–E choice. For medical QA, it compares the final A–D choice.

Text-to-SQL. We report execution accuracy using local implementations of the evaluation rules specified by the respective benchmarks. For WikiSQL, we follow the DBEngine execution criterion

and evaluate on the sampled test split.[1] For Spider, we follow the `result_eq` comparison rules and evaluate on the canonical development databases.[2] For BIRD, we follow its execution-accuracy criterion and evaluate on the development split.[3].

B.6 VALIDATION

The fixed validation sets contain 1000 Math14K, 2,000 MedMCQA-training, and 2,000 WikiSQL-development examples. Validation loss is used for training diagnostics and early hyperparameter research. The main experiments do not use early stopping, select a checkpoint by validation loss, or restore an intermediate checkpoint. All reported main evaluations use the checkpoint at the end of the single training epoch.

B.7 CONTINUAL-LEARNING PROTOCOL

We evaluate replay-free sequential adaptation using DeepSeek-V2-Lite Base and seven NuminaMath-1.5 domains: Algebra, Geometry, Number Theory, Combinatorics, Calculus, Inequalities, and Logic and Puzzles. The domains are learned in this fixed order, with one domain introduced at each stage. Each domain contains 4,000 training examples, 500 validation examples, and 1,000 test examples.

At each stage, we add a rank-8 LoRA adapter with scaling parameter 8. Earlier adapters remain active but frozen. The router is carried forward and remains trainable. No examples from previous domains are replayed. The optimizer and GR state are reset at each stage boundary.

Let $A_{t,i}$ be the percentage accuracy on domain i after completing stage t, and let $T = 7$. Cumulative average accuracy (CAA) averages performance over all stages, giving equal weight to each stage. Final average accuracy (FAA) averages performance across all domains after the last stage:

$$\mathrm{CAA} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{t}\sum_{i=1}^{t} A_{t,i}, \qquad \mathrm{FAA} = \frac{1}{T}\sum_{i=1}^{T} A_{T,i}. \tag{37}$$

Forgetting measures the change in accuracy on an earlier domain relative to its best result before the current stage:

$$\mathrm{Forgetting} = \frac{1}{T-1}\sum_{i=1}^{T-1}\frac{1}{T-i}\sum_{t=i+1}^{T}\left(\max_{k=i,\ldots,t-1} A_{k,i} - A_{t,i}\right). \tag{38}$$

Thus, we first average across subsequent stages for each earlier domain and then across domains. We retain negative differences when the current accuracy exceeds all earlier measurements. Like CAA and FAA, Forgetting is reported in percentage points.

Routing Forgetting measures changes in expert selection on fixed, identically tokenized probes from each domain. For a probe token n from domain i, let $\mathcal{E}^{(\ell)}_{t,i,n}$ be its Top-K expert set in sparse MoE layer ℓ after stage t. We compare each later stage with the expert set recorded immediately after learning domain i. Let N_i denote the number of probe tokens and M the number of sparse MoE layers. The routing change at stage $t > i$ is

$$R_{t,i} = 1 - \frac{1}{M}\sum_{\ell=1}^{M}\frac{1}{N_i}\sum_{n=1}^{N_i}\frac{\left|\mathcal{E}^{(\ell)}_{i,i,n} \cap \mathcal{E}^{(\ell)}_{t,i,n}\right|}{K}. \tag{39}$$

We aggregate $R_{t,i}$ using the same stage-then-domain averaging rule as Forgetting and convert the result to a percentage:

$$\text{Routing Forgetting} = \frac{1}{T-1}\sum_{i=1}^{T-1}\frac{1}{T-i}\sum_{t=i+1}^{T} R_{t,i}. \tag{40}$$

A fixed router can still have nonzero Routing Forgetting because updates to preceding model components change the representations it receives.

[1] `https://github.com/salesforce/WikiSQL`

[2] `https://github.com/taoyds/test-suite-sql-eval`

[3] `https://github.com/AlibabaResearch/DAMO-ConvAI/tree/main/bird`

B.8 Complete Mathematical Reasoning Results

Table 6: Complete Math14K evaluation (accuracy, %). The best and second-best scores within each model and benchmark are bold and underlined, respectively. Averages are recomputed from the displayed benchmark scores.

Method	SingleEq	MultiArith	AddSub	GSM8K	SVAMP	AQuA	Avg.
DeepSeek-V2-Lite							
Frozen Router	89.55	86.38	71.14	53.83	63.20	41.73	67.64
FT w/o LB	88.56	88.10	73.16	54.59	62.90	38.19	67.58
FT w/ LB	88.17	87.07	72.41	53.22	59.80	39.37	66.67
Router Anchor	89.35	87.07	73.16	54.74	64.60	37.80	67.79
Spec. Loss	89.15	87.41	70.63	55.42	61.70	42.52	67.81
Expert Cond.	88.76	88.45	73.67	55.27	64.50	40.94	68.60
GR (Ours)	**89.74**	**88.62**	**75.44**	**58.38**	**65.30**	**45.67**	**70.53**
OLMoE							
Frozen Router	63.12	73.79	47.09	25.32	31.50	16.14	42.83
FT w/o LB	**64.10**	74.31	49.11	26.00	31.50	14.96	43.33
FT w/ LB	47.34	65.17	40.51	19.48	26.50	12.99	35.33
Router Anchor	62.52	72.93	48.35	25.47	32.30	16.93	43.08
Spec. Loss	57.20	71.55	48.10	23.50	31.90	18.11	41.73
Expert Cond.	60.36	73.28	48.35	25.47	29.90	16.93	42.38
GR (Ours)	63.71	**75.00**	**51.39**	**28.81**	**32.80**	**20.47**	**45.36**
Qwen3							
Frozen Router	99.21	98.97	95.19	91.05	91.70	72.83	91.49
FT w/o LB	99.01	99.14	94.43	91.21	90.90	74.41	91.52
FT w/ LB	98.42	98.97	92.41	86.88	89.80	68.11	89.10
Router Anchor	98.82	99.14	95.44	90.52	**91.80**	74.02	91.62
Spec. Loss	97.04	98.62	95.70	86.05	88.70	72.05	89.69
Expert Cond.	**99.41**	**99.31**	95.70	90.14	90.50	74.80	91.64
GR (Ours)	99.21	99.14	**96.46**	**92.72**	91.20	**77.95**	**92.78**

Table 6 expands the mathematical reasoning columns of Table 1 to all six evaluation benchmarks. The Avg. column is the unweighted mean of these six displayed scores.

B.9 Routing Diagnostics

We analyze routing changes for DeepSeek-V2-Lite fine-tuned on Math14K. Within each comparison, all methods use the same fixed 256-example routing probe, example order, tokenizer, chat template, sequence limit, and pretrained routing reference. The probe is randomly drawn from the Math14K training set. We record routing decisions for every non-padding token in the complete SFT sequence, including prompt and response tokens. Pretrained and fine-tuned traces are paired by token position within each MoE layer.

Let $\mathcal{A}^0_{\ell,t}$ and $\mathcal{A}^f_{\ell,t}$ denote the Top-k expert sets for token t in layer ℓ before and after fine-tuning. Each diagnostic below is averaged equally over the L sparse MoE layers and the N valid probe tokens.

Routing-probability KL. Let $p_\ell(x_t)$ denote the routing probability distribution over all routed experts for token x_t in layer ℓ. Superscripts 0 and f denote the pretrained and fine-tuned checkpoints, respectively. We average the KL divergence over L MoE layers and N probe tokens:

$$\text{RoutingKL} = \frac{1}{LN} \sum_{\ell=1}^{L} \sum_{t=1}^{N} D_{\text{KL}} \left(p_\ell^0(x_t) \,\middle\|\, p_\ell^f(x_t) \right). \tag{41}$$

Top-K overlap. We compare selected expert sets without considering their rank or routing weights:

$$\text{Overlap} = \frac{1}{LN}\sum_{\ell=1}^{L}\sum_{t=1}^{N}\frac{\left|\mathcal{A}^0_{\ell,t}\cap\mathcal{A}^f_{\ell,t}\right|}{K}. \tag{42}$$

Expert-load total variation. Load is based on hard Top-K assignments. Let

$$P_{\ell,e} = \frac{1}{NK}\sum_{t=1}^{N}\mathbf{1}[e\in\mathcal{A}_{\ell,t}] \tag{43}$$

be the normalized assignment frequency of expert e in layer ℓ. Thus, $\sum_e P_{\ell,e}=1$. We report

$$\text{LoadTV} = \frac{1}{L}\sum_{\ell=1}^{L}\frac{1}{2}\sum_{e=1}^{E}\left|P^f_{\ell,e}-P^0_{\ell,e}\right|. \tag{44}$$

Load TV measures aggregate changes in expert utilization.

B.10 Ablation Implementations

All ablations use DeepSeek-V2-Lite on Math14K. They share the same training and evaluation settings; each variant changes the GR correction as specified below. The corresponding accuracy and routing diagnostics are reported in Tables 3 and 2(b).

GR without ψ. This variant balances positive and negative raw score-gradient contributions without estimating their effects on routing-region mass. For $r_{ti} = -\partial\mathcal{L}/\partial s_i(x_t)$, define

$$P_i = \sum_t [r_{ti}]_+, \qquad N_i = \sum_t [-r_{ti}]_+, \qquad \alpha_i = \frac{\min(P_i,N_i)}{\max(P_i,N_i)+\epsilon}. \tag{45}$$

For each expert, contributions on the side with the larger aggregate magnitude are multiplied by α_i; contributions on the other side are unchanged.

Layer-global GR. GR chooses a dominant side and attenuation coefficient separately for each expert. This variant first computes the positive and negative tendency contributions for each expert, then aggregates them across all experts in a layer. It selects one dominant side for the entire layer and uses one shared attenuation coefficient for all its router rows. The coefficient is found by scalar bisection to bring the *aggregate* estimated tendency across the layer close to zero. Individual experts therefore need not have near-zero tendencies.

Projection. This baseline projects the candidate AdamW parameter update onto the subspace orthogonal to the estimated region-gradient direction. Let $\Delta\theta_i^{\text{AdamW}}$ denote the candidate update computed from the unmodified task gradient. We apply

$$\Delta\theta_i^{\text{Proj}} = \Delta\theta_i^{\text{AdamW}} - \frac{\bar{a}_i^\top \Delta\theta_i^{\text{AdamW}}}{\|\bar{a}_i\|_2^2}\bar{a}_i. \tag{46}$$

When $\bar{a}_i = 0$, the candidate update is left unchanged. AdamW's moment estimates are updated using the unmodified task gradient, while the router parameters are updated directly by $\theta_i \leftarrow \theta_i + \Delta\theta_i^{\text{Proj}}$. Thus, the applied update satisfies $\bar{a}_i^\top \Delta\theta_i^{\text{Proj}} = 0$ up to numerical precision, enforcing zero estimated first-order self-region tendency.

Norm-matched fine-tuning. At each optimizer step, we compute the plain and GR-corrected candidate updates for each router row from the same parameters and optimizer state, denoted by $\Delta\theta_i^{\text{FT}}$ and $\Delta\theta_i^{\text{GR}}$. This baseline applies

$$\Delta\theta_i^{\text{NM}} = \frac{\|\Delta\theta_i^{\text{GR}}\|_2}{\|\Delta\theta_i^{\text{FT}}\|_2}\Delta\theta_i^{\text{FT}}.$$

Thus, the baseline matches the GR candidate's parameter-update norm at the same state without constraining the update's self-region tendency.

Exposure-only scaling. Let c_i be the number of probe-batch tokens that select expert i under hard Top-K routing, and let $\bar{c} = NK/E$ be the mean assignment count in that layer. We scale the router-gradient row associated with expert i by

$$g_i^{\exp} = \alpha_i g_i, \qquad \alpha_i = \begin{cases} \bar{c}/c_i, & c_i > \bar{c}, \\ 1, & c_i \le \bar{c}. \end{cases} \tag{47}$$

This attenuates above-average exposure without amplifying less-exposed experts. It does not use ψ, distinguish contributions to expansion from those to contraction, or solve for self-region neutrality. The scaled quantity is the *router-gradient row*, not the parameters or gradients of the expert network.

B.11 Selective Application of GR

We observe substantial variation in load imbalance across MoE layers: some layers have highly uneven expert utilization, whereas others are relatively balanced. This motivates a selective variant that applies GR only when a layer's load imbalance exceeds a threshold. For each layer ℓ and training batch, we measure imbalance by the normalized variance of expert assignment frequencies:

$$V_\ell = \frac{1}{E} \sum_{i=1}^{E} \left(\frac{f_{\ell i} - 1/E}{1/E} \right)^2, \tag{48}$$

where $f_{\ell i}$ is expert i's share of the Top-k assignments in layer ℓ. Selective-GR applies GR when $V_\ell > \tau$ and uses the unmodified task gradient otherwise.

We calibrate τ on Math14K batches using the pretrained router. The labels 20%, 40%, 60%, and 80% denote target trigger rates over layer–batch instances during calibration; the thresholds remain fixed during fine-tuning. All variants use DeepSeek-V2-Lite and the same Math14K training and evaluation protocol described in Section 4.1.

Table 7: Accuracy (%) after Math14K fine-tuning. Avg. is the unweighted mean across the six benchmarks.

Method	SingleEq	MultiArith	AddSub	GSM8K	SVAMP	AQuA	Avg.
FT w/o LB	88.56	88.10	73.16	54.59	62.90	38.19	67.58
Selective-GR 20%	**89.74**	**89.83**	<u>75.19</u>	53.83	<u>64.30</u>	39.76	68.78
Selective-GR 40%	87.38	**89.83**	72.15	54.51	64.10	38.19	67.69
Selective-GR 60%	87.18	87.93	74.18	57.92	63.90	39.76	68.48
Selective-GR 80%	<u>89.35</u>	88.45	74.43	**58.83**	63.10	<u>41.34</u>	<u>69.25</u>
GR	**89.74**	<u>88.62</u>	**75.44**	<u>58.38</u>	**65.30**	**45.67**	**70.53**

Table 8: Calibrated thresholds, realized trigger rates, and routing diagnostics after Math14K fine-tuning.

Method	τ	Trigger rate	KL ↓	Overlap ↑	Load TV ↓
FT w/o LB	–	0%	1.9495	0.7957	0.0838
Selective-GR 20%	0.5488	23.73%	1.9118	0.8324	0.0796
Selective-GR 40%	0.3282	59.19%	1.9156	0.8335	0.0791
Selective-GR 60%	0.2310	82.64%	1.8788	0.8353	0.0746
Selective-GR 80%	0.1692	92.08%	<u>1.7564</u>	<u>0.8421</u>	<u>0.0618</u>
GR	–	100%	**1.6477**	**0.8826**	**0.0474**

As the selective trigger rate increases, routing preservation generally improves: routing KL and load TV decrease, while Top-k overlap increases. Average accuracy, however, is non-monotonic across the selective variants. Selective-GR 80% reaches a realized trigger rate of 92.08% and the highest average accuracy among the selective variants, 69.25%. Full GR still achieves a higher average accuracy of 70.53%, together with lower routing KL and load TV and higher overlap. Thus, correcting most layer–batch instances does not recover the effect of applying GR throughout fine-tuning.

These results indicate that the fraction of corrected updates alone does not determine the outcome. One plausible explanation is that the remaining uncorrected updates occur at consequential layers or

training steps. An earlier layer's router affects the representations passed to later layers, changing their routing decisions and gradient signals. Switching GR on and off within this coupled system could therefore make its updates less consistent across layers and steps. The present experiment establishes the advantage of full GR over this load-variance-based selection rule.

B.12 COMMONSENSE REASONING

We evaluate GR on commonsense reasoning by fine-tuning DeepSeek-V2-Lite Base for one epoch on 14,823 processed Commonsense-15K examples (Hu et al., 2023). We report accuracy on BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), Social IQa (SIQA) (Sap et al., 2019), HellaSwag (Zellers et al., 2019), ARC-Easy and ARC-Challenge (Clark et al., 2018), and OpenBookQA (OBQA) (Mihaylov et al., 2018). Avg. is the unweighted mean of these seven benchmarks.

Table 9: Commonsense reasoning accuracy (%) after fine-tuning on Commonsense-15K. Avg. is the unweighted mean of the seven benchmarks.

Method	**BoolQ**	**PIQA**	**SIQA**	**HellaSwag**	**ARC-E**	**ARC-C**	**OBQA**	**Avg.**
Frozen Router	75.87	81.45	75.18	87.29	90.15	75.94	81.40	81.04
Plain FT w/o LB	76.82	82.10	75.23	87.37	89.81	76.11	81.40	81.26
Plain FT w/ LB	<u>80.35</u>	81.83	**75.79**	87.74	**90.61**	**77.13**	**83.60**	**82.44**
Router Anchor	**80.75**	**82.37**	75.33	87.49	89.86	**77.13**	<u>83.40</u>	<u>82.33</u>
GR	80.04	81.12	<u>75.59</u>	87.91	89.94	75.34	81.60	81.65
Specialization Loss	62.23	80.14	72.93	83.73	86.03	71.50	77.40	76.28
Expert Condenser	78.42	81.23	75.38	**88.05**	89.94	75.94	82.40	81.62

Table 9 shows that GR improves the seven-benchmark Avg. accuracy from 81.26% with Plain FT w/o LB to 81.65%. Its largest gain is on BoolQ, while accuracy decreases on PIQA and ARC-C. Plain FT w/ LB achieves the highest Avg. accuracy of 82.44%, followed by Router Anchor at 82.33%. GR therefore provides a modest improvement over Plain FT w/o LB, but does not lead this comparison on commonsense reasoning.

To examine this difference across task settings, we randomly sample inputs from each downstream task and run zero-shot inference with the pretrained model. We measure expert assignment frequencies using the normalized load variance in Eq. 48, averaged across sampled batches and MoE layers. Commonsense-15K has an average variance of 0.237, compared with 0.475 for Math14K, 0.586 for MedMCQA, and 0.693 for WikiSQL. Thus, commonsense inputs induce substantially more balanced expert utilization before fine-tuning than the tasks in our main experiments. GR is designed to limit routing drift driven by uneven expert exposure; when initial exposure is relatively balanced, there may be less imbalance-induced drift for GR to correct. This is consistent with its smaller accuracy gain on commonsense reasoning.