GRADROUTER: DECOMPOSE GRADIENT BEFORE ASSIGNING LEARNING RESPONSIBILITY IN SPARSE MIXTURE-OF-EXPERTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse Mixture-of-Experts (MoE) models enable efficient scaling by activating only a subset of experts for each token. Their effectiveness largely depends on how experts divide and coordinate their responsibilities. Modern MoEs use routers to determine which experts are activated and how much their outputs contribute to the aggregated output. These forward routing decisions are directly inherited during backpropagation to determine how learning signals are distributed among the activated experts, *i.e.*, implicitly equating an expert's *learning responsibility* with its *prediction responsibility*, which might hinder their functional differentiation. In this paper, we raise the attention that the backward learning signals are high-dimensional, with different experts might be potentially better suited to different directions. This suggests that scalar routing weights alone might not fully characterize how learning responsibility should be divided. Therefore, we propose **GradRouter**, a framework that decomposes backward learning responsibility into differentiated learning directions before assigning to co-activated experts. GradRouter identifies principal directions in the output-gradient space and allocates them among co-activated experts according to their direction-specific capabilities, while preserving the aggregate output gradient. Experiments across diverse sparse MoE architectures with 7B–30B total parameters show that GradRouter generally improves model performance, demonstrating the effectiveness and scalability of directional learning responsibility assignment.

1 INTRODUCTION

Sparse Mixture-of-Experts (MoE) models enable efficient scaling by dynamically activating only a small subset of experts for each input, allowing model capacity to grow without proportional increases in per-token computation (Shazeer et al., 2017; Dai et al., 2024; Qiu et al., 2025). Realizing the benefit of this expanded capacity depends critically on whether experts effectively develop complementary capabilities and coordinate across tokens.

Modern MoEs use token-conditioned *routers* to determine which experts are activated for each token and how much their outputs contribute to the aggregated prediction (Shazeer et al., 2017), as illustrated in the top part of **Fig.** 1. Accordingly, existing methods promote expert specialization primarily by shaping this forward division of labor, *e.g.*, encouraging different experts to process distinct token distributions through routing mechanisms Wang et al. (2026), or explicitly regularizing expert representations to increase diversity among experts Do et al. (2025); Wang et al. (2026). In both cases, specialization is induced mainly by differentiating experts' roles in forward computation.

Forward routing, however, determines how experts divide *prediction responsibility*: which experts should contribute to the current prediction and by how much. During training, there comes a distinct question of *learning responsibility*: given the supervision signal produced by the current prediction, how should the co-activated experts divide what needs to be learned? These two responsibilities need not coincide. An expert's contribution to the current prediction does not by itself determine how much, or along which directions, that expert should change in response to the resulting supervision. Nevertheless, standard backpropagation directly inherits the forward routing weights when assigning learning signals to experts, implicitly tying learning responsibility to prediction responsibility.

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.

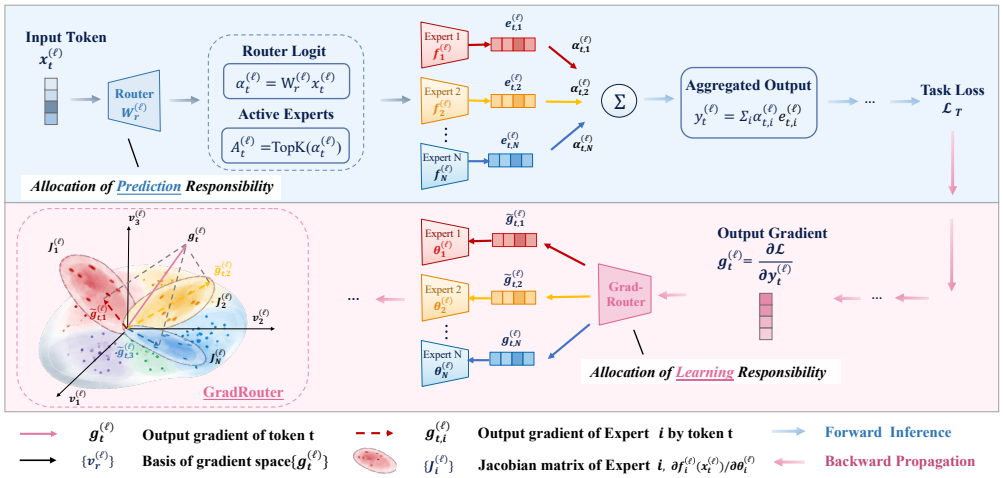

Figure 1: **Prediction and learning responsibility in sparse MoE. Top:** The forward router allocates prediction responsibility by selecting active experts and weighting their outputs. Standard backpropagation inherits these scalar weights, giving co-activated experts collinear output-side gradients. **Bottom:** GradRouter introduces directional learning responsibility assignment by decomposing output gradient according to response capabilities of co-activated experts.

This coupling becomes particularly restrictive when viewed through the geometry of backward learning signals. Under linear output aggregation, the output-side gradients received by co-activated experts differ only by their scalar routing weights and are therefore collinear. Such gradient alignment may induce similar update tendencies across co-activated experts, thereby weakening the differentiation of their learned functions. More fundamentally, backward learning signals are inherently high-dimensional, whereas scalar routing weights provide only direction-agnostic control over their allocation. As a result, the same relative responsibility is imposed across all gradient directions, despite potential heterogeneity in experts' capabilities along different directions. This discrepancy reveals a fundamental limitation of directly inheriting forward routing weights for backward credit assignment and motivates a finer-grained formulation in which learning responsibility is differentiated across gradient directions. Here comes the central question :

> *While forward routing assigns prediction responsibility with scalar weights, how should backward learning responsibility be divided across directions among co-activated experts?*

To address this question, we propose **GradRouter**, a framework for directional learning responsibility assignment among co-activated experts in sparse MoEs, as illustrated in the bottom part of **Fig.** 1. GradRouter leaves forward expert selection and output aggregation unchanged, and operates only during backpropagation to redistribute the output-side gradient within each token's co-activated experts. Specifically, GradRouter first identifies principal directions in the output-gradient space, providing a low-dimensional basis for directional responsibility assignment. It then estimates each expert's capability along these directions and uses local optimal transport to allocate directional credit among the currently activated experts. Finally, it conservatively recombines the assigned and residual components, preserving the aggregate output-side gradient while allowing individual experts to receive differentiated learning signals. Experiments across diverse sparse MoE architectures spanning 7B–30B total parameters show broad performance improvements, indicating the effectiveness of direction-aware learning responsibility assignment.

Our contributions are summarized as follows:

❶ We distinguish forward *prediction responsibility* from backward *learning responsibility*, and highlight that directly inheriting scalar forward routing weights may be insufficient for allocating high-dimensional backward learning signals.

❷ We propose **GradRouter**, a backward responsibility assignment framework that decomposes backward learning signals into principal directions and assigns direction-specific responsibility among the activated experts according to their capabilities, while preserving the aggregate output-side gradient.

❸ We demonstrate performance improvements across sparse MoE architectures spanning 7B–30B total parameters, suggesting the effectiveness of directional learning responsibility assignment.

2 Related Work

Responsibility Assignment in Classical and Modern Mixtures of Experts. Responsibility assignment has been a central problem of mixture-of-experts models since their early formulations. Jacobs et al. (1991) introduced competitive learning, in which experts that better explain an example receive stronger learning signals, while Jordan & Jacobs (1994) further formalized expert responsibility through posterior assignment under an EM framework. In these classical formulations, expert responsibility is closely tied to prediction error and directly governs how strongly each expert learns from an example. While modern sparse MoEs (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022) introduce token-conditioned routers to determine expert selection and output aggregation, thereby assigning experts' relative contributions during the forward pass. Under standard backpropagation, the same routing weights are subsequently inherited to scale the learning signals received by the activated experts. Thus, modern sparse MoEs largely couple backward learning responsibility to forward routing decisions rather than modeling it independently. Our work revisits this distinction and highlights that backward learning responsibility should be allocated at a finer directional granularity within the set of co-activated experts in Modern MoEs.

Expert Specialization in Modern Sparse MoEs. A major challenge in sparse MoEs is to encourage experts to develop complementary rather than redundant functions Chi et al. (2022); Do et al. (2025). Existing approaches promote expert differentiation of modern MoEs from several perspectives: ***(i)*** define heterogeneous or decomposed expert roles by architectural designs, *e.g.*, distinguishing shared and fine-grained experts Dai et al. (2024); ***(ii)*** specialization-oriented routing that encourages experts to receive distinct token distributions (Eo et al., 2025; Guo et al., 2025); and ***(iii)*** explicit diversity regularization over expert outputs (Do et al., 2025; Guo et al., 2025) and update directions (Liu et al., 2024). Recent analyses further study how specialization relates to routed-token history and representation geometry (Wang et al., 2026; Ahrac et al., 2026), while also suggesting that geometric diversity alone does not necessarily imply task-relevant functional specialization (Kim, 2026). Moreover, auxiliary objectives for encouraging expert differentiation may introduce optimization interactions with the primary task objective (Wang et al., 2024a). Instead of differentiating experts in forward computation, GradRouter directly operates on the task-derived backward learning signal and decomposes the output-gradient into directionally differentiated learning responsibility according to co-activated capabilities, enabling a structural learning responsibility assignment.

3 Preliminary

Forward: Sparse Routing and Expert Aggregation in MoE. We consider a specific MoE layer and omit the layer index throughout for notational clarity. Given the input hidden state $\mathbf{x}_t \in \mathbb{R}^d$ of token t, the router predicts a score $s_{t,i}$ for each expert i. Experts with the Top-K highest scores (this activated expert set is denoted as $\mathcal{A}_t$ in the following) are activated and their outputs are aggregated with normalized routing weight $\alpha_{t,i}$ to construct MoE layer's output:

$$\mathbf{y}_t = \sum_{i \in \mathcal{A}_t} \alpha_{t,i} \cdot \mathbf{e}_{t,i}. \tag{1}$$

where $\mathbf{e}_{t,i} = f_i(\mathbf{x}_t; \boldsymbol{\theta}_i)$ denotes expert i's output for token t. Thus, the forward MoE computation performs *scalar-weighted expert aggregation*: the router controls both expert selection and each activated expert's contribution to the final output. We refer to this forward allocation as the token's *prediction responsibility* across experts.

Backward: Routing-Weighted Expert Credit Assignment in MoE. Let $\mathbf{g}_t := \partial \mathcal{L} / \partial \mathbf{y}_t$ denote the output-side task gradient for token t. Backpropagating through the weighted aggregation in Eq. 1, the output of each activated expert receives the gradient as :

$$\frac{\partial \mathcal{L}}{\partial \mathbf{e}_{t,i}} = \alpha_{t,i} \cdot \mathbf{g}_t, \qquad i \in \mathcal{A}_t. \tag{2}$$

Hence, for the same token, all co-activated experts receive collinear output-space gradients, differing only in magnitude through the routing weight $\alpha_{t,i}$. In this sense, standard sparse MoE training couples forward *prediction responsibility* with backward *learning responsibility*: the relative weighting used to aggregate expert predictions in the forward pass is inherited directly when distributing the output-side learning signal in the backward pass.

4 METHOD

In standard backpropagation (Eq. 2), the activated experts always receive collinear output gradients. To better redistribute learning responsibility, in this paper, we propose **GradRouter**, which decomposes the output-side gradient g_t into several expert-specific directional gradients $g_{i,t}$ to enable a directionally differentiated assignment of learning responsibility among co-activated experts. Realizing such directional learning responsibility mainly requires considering the following 3 questions:

(Q1) *How can we construct a shared directional basis for decomposing high-dimensional output gradients across tokens?*

(Q2) *How can learning responsibility along these directions be assigned among co-activated experts according to their directional capabilities?*

(Q3) *How can we preserve the output-side gradient after directional responsibility assignment?*

In **GradRouter**, we address ***(Q1)*** by computing the principal components of recent output gradients in **Sec.** 4.1, ***(Q2)*** by first estimating expert–direction affinities and solving a local optimal transport problem to enable directional assignment in **Sec.** 4.2, and ***(Q3)*** by redistributing the residual gradient component according to router weights in **Sec.** 4.3. An overview of GradRouter is illustrated in **Fig.** 2, which summarizes its three main stages.

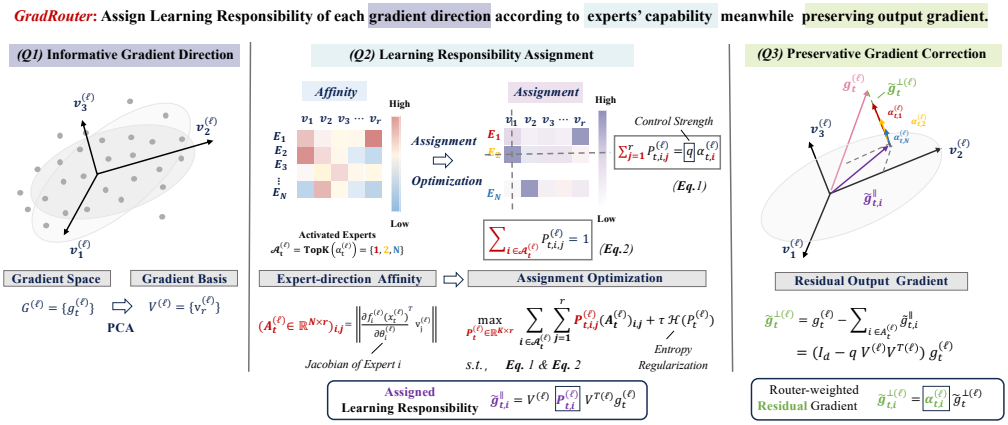

Figure 2: **Overview of GradRouter.** GradRouter assigns learning responsibility to co-activated experts at the level of output-gradient directions while preserving the aggregate output gradient. It addresses **(Q1)** by constructing a shared basis of informative directions, **(Q2)** by estimating expert–direction affinities and assigning directional learning responsibility among the activated experts, and **(Q3)** by distributing the residual gradient according to the original router weights. Consequently, each expert receives a differentiated learning signal while the expert-specific output gradients sum to the original output gradient.

4.1 GRADIENT BASIS CONSTRUCTION

(A1) To identify the informative directions of the output-gradient space for ***(Q1)***, we apply uncentered PCA to a recent collection of token-level output gradients gathered from the training stream, irrespective of which experts were activated. Every T steps, we form the empirical second moment C_g from the most recent n_g gradients and retain its leading r principal components V:

$$C_g := \frac{1}{n_g}\sum_{t=1}^{n_g} \mathbf{g}_t \mathbf{g}_t^\top \in \mathbb{R}^{d\times d} \quad \xrightarrow{\text{PCA}} \quad V := [\mathbf{v}_1, \ldots, \mathbf{v}_r] \in \mathbb{R}^{d\times r}. \tag{3}$$

Equivalently, V contains the top-r right singular vectors of the matrix formed by stacking the recent output gradients. These directions demonstrate the dominant structure of output gradients and provide a shared coordinate for assigning learning responsibility. The layer-wise PCA analysis of output gradients across 4096 tokens can be found in Appendix B.1 and the computational complexity can be found in Appendix E. Generally, each PCA refresh incurs $O(n_g d^2 + d^3 + dr^2)$ arithmetic cost, amortized over T training steps.

4.2 Learning Responsibility Assignment

(A2) For ***(Q2)***, the overall aim is to let each expert take responsibility for different directions spanned by the principal directions. For that, we first quantify each expert's capacity to change its output along each principal direction, yielding an **expert–direction affinity** matrix $A = [A_{ij}] \in \mathbb{R}^{N\times r}$. We then solve a token-local assignment problem that allocates learning responsibility among the K activated experts according to these affinities, while preserving each expert's router-defined *prediction-responsibility* budget.

(i) **Expert–direction Affinity.** An expert could take more responsibility for the direction when updating its parameters can effectively move its output along $\mathbf{v}_j$. Therefore, we measure how readily each expert i can change its output along each principal direction $\mathbf{v}_j$, obtaining an expert–direction affinity matrix. Specifically, the Jacobian of each expert $J_i(\mathbf{x}_t) = \partial f_i(\mathbf{x}_t)/\partial \boldsymbol{\theta}_i$ maps a local parameter update to the resulting output change, reflecting the output directions the expert can readily realize. Therefore, the maximum attainable change along $\mathbf{v}_j$ with a unit-norm parameter update of expert E_i is

$$\max_{\|\Delta\boldsymbol{\theta}_i\|_2\le 1} \left|\mathbf{v}_j^\top J_i(\mathbf{x}_t)\Delta\boldsymbol{\theta}_i\right| = \left\|J_i(\mathbf{x}_t)^\top \mathbf{v}_j\right\|_2 . \tag{4}$$

The expert-direction affinity is estimated over a shared calibration set $\mathcal{C}_h$:

$$A := [A_{ij}] \in \mathbb{R}^{N\times r}, \quad \text{where} \quad A_{ij} = \frac{1}{|\mathcal{C}_h|}\sum\nolimits_{\mathbf{x}_t\in\mathcal{C}_h} \left\|J_i(\mathbf{x}_t)^\top \mathbf{v}_j\right\|_2^2 . \tag{5}$$

Explicitly constructing each token-wise Jacobian $J_i(\mathbf{x}_t) \in \mathbb{R}^{d\times|\boldsymbol{\theta}_i|}$ would incur substantial computational cost and require $O(d|\boldsymbol{\theta}_i|)$ storage. Leveraging the structure of bias-free SwiGLU experts, the Jacobian–direction product with respect to the gate and up projections admits an explicit closed-form expression that depends only on cached forward activations and the basis directions projected through the down projection. We therefore use $\|(J_i^{g,u}(\mathbf{x}_t))^\top \mathbf{v}_j\|_2^2$ as a **computationally efficient proxy** for the full expert directional response when estimating the affinity:

$$\|J_i^{g,u}(\mathbf{x}_t)^\top \mathbf{v}_j\|_2^2 = \|\mathbf{x}_t\|_2^2 \left[\|\mathbf{r}_{i,j}\odot \mathrm{SiLU}(\mathbf{g}_{i,t})\|_2^2 + \|\mathbf{r}_{i,j}\odot \mathbf{u}_{i,t}\odot \mathrm{SiLU}'(\mathbf{g}_{i,t})\|_2^2\right] . \tag{6}$$

For notational simplicity, we write J_i for $J_i^{g,u}$ hereafter. In practice, we parallel compute $\mathbf{g}_{i,t} := W_{g,i}\mathbf{x}_t$, $\mathbf{u}_{i,t} := W_{u,i}\mathbf{x}_t$, and $\mathbf{r}_{i,j} := W_{d,i}^\top \mathbf{v}_j$ in parallel, and evaluate the affinity efficiently with batched matrix operations over the calibration tokens. More details about this closed-form derivation and computation complexity are provided in Appendix E.

(ii) **Learning Responsibility Assignment.** Based on the above Expert-direction affinity matrix, for each token $\mathbf{x}_t$, we then assign the directional learning responsibility among the top-K co-activated experts by solving an assignment optimization problem.

Before performing the assignment, we transform the raw affinity matrix A into comparative scores, *i.e.,* apply log compression to compress their dynamic range and then remove expert-wise and direction-wise offsets through double-centering:

$$\widetilde{A}_{ij} = \log\max(A_{ij},\epsilon) - \overline{\log A}_{i,\cdot} - \overline{\log A}_{\cdot,j} + \overline{\log A}. \tag{7}$$

Here, the overlined terms denote the row, column, and global means of the clipped log affinities, respectively. This normalization suppresses systematic expert-wise and direction-wise offsets, allowing the assignment to depend on relative expert–direction affinity rather than raw magnitude.

We then optimize an assignment strategy $P_t = (P_{t,i,j}) \in \mathbb{R}_+^{K\times r}$ for token $\mathbf{x}_t$, where $P_{t,i,j}$ specifies the fraction of direction $\mathbf{v}_j$'s learning responsibility allocated to activated expert i. The objective assigns larger shares to expert–direction pairs with larger $\widetilde{A}_{ij}$, while entropy regularization prevents an excessively sharp allocation. Meanwhile, each direction is fully distributed among the activated experts, and the total responsibility received by expert i is fixed to $r\alpha_{t,i}$. This optimization is

formalized as the following:

$$\begin{aligned} \max_{P_t \geq 0} \quad & \sum_{i \in \mathcal{A}_t} \sum_{j=1}^{r} P_{t,i,j} \widetilde{A}_{ij} + \tau \mathcal{H}(P_t) \\ \text{s.t.} \quad & \sum_{i \in \mathcal{A}_t} P_{t,i,j} = 1 \;\; (j = 1, \ldots, r), \qquad \sum_{j=1}^{r} P_{t,i,j} = r\alpha_{t,i} \;\; (i \in \mathcal{A}_t). \end{aligned} \tag{8}$$

where $\mathcal{H}(P_t) = -\sum_{i,j} P_{t,i,j} \log P_{t,i,j}$. We solve Eq. 8 using log-domain Sinkhorn iterations and apply column normalization to remove numerical drift.

***(iii)* Assigned Learning Responsibility.** The i-th row of above assignment strategy P_t for token t indicates expert E_i's assigned shares across the r principal directions. Then we could assign the learning responsibility by decomposing its output gradient g_t into:

$$\mathbf{g}_{t,i}^{\parallel} = qV \,\mathrm{Diag}(\mathbf{p}_{t,i}) V^T g_t. \tag{9}$$

where $q \in [0, 1]$ is an allocation strength to control the overall strength of this reassignment.

4.3 Residual Responsibility Assignment

(A3) The affinity-guided assignment above redistributes only the gradient component represented in the shared principal subspace $\mathrm{span}(V)$. To preserve the complete task gradient, we retain the complementary component and distribute it according to the original router weights $\alpha_{t,i}$. This preserves the forward-routing prior for gradient information not covered by the directional assignment.

Specifically, the residual gradient is

$$\mathbf{g}_t^{\mathrm{res}} := \mathbf{g}_t - \sum_{i \in \mathcal{A}_t} \mathbf{g}_{t,i}^{\parallel} = \left(I_d - qVV^\top\right) \mathbf{g}_t \tag{10}$$

It contains all gradient components outside $\mathrm{span}(V)$, together with the unassigned fraction $1 - q$ inside this subspace.

Combining the affinity-assigned component with the router-weighted residual gives the complete learning responsibility of expert i:

$$\widetilde{\mathbf{g}}_{t,i} = \underbrace{VM_{t,i}V^\top \mathbf{g}_t}_{\text{Assigned Learning Responsibility}} + \underbrace{\alpha_{t,i}\mathbf{g}_t^{\mathrm{res}}}_{\text{Router-weighted Residual Gradient}}. \tag{11}$$

Since $\sum_{i \in \mathcal{A}_t} M_{t,i} = \sum_{i \in \mathcal{A}_t} q\,\mathrm{Diag}(\mathbf{p}_{t,i}) = qI_r$ and $\sum_{i \in \mathcal{A}_t} \alpha_{t,i} = 1$, the redistributed gradients satisfy $\sum_{i \in \mathcal{A}_t} \widetilde{\mathbf{g}}_{t,i} = \mathbf{g}_t$. Thus, GradRouter changes how learning responsibility is distributed across experts and directions while preserving the aggregate output-space task gradient.

Zero-sum Correction View. The same construction can be expressed as a correction to standard MoE backpropagation. Substituting Eq. 10 into Eq. 11 yields

$$\widetilde{\mathbf{g}}_{t,i} = \alpha_{t,i}\mathbf{g}_t + \underbrace{V\left(M_{t,i} - q\alpha_{t,i} I_r\right) V^\top \mathbf{g}_t}_{\Delta \mathbf{g}_{t,i}}, \qquad \sum_{i \in \mathcal{A}_t} \Delta \mathbf{g}_{t,i} = \mathbf{0}. \tag{12}$$

Equation 12 shows that GradRouter modifies the standard router-weighted gradient only within $\mathrm{span}(V)$. These expert-specific corrections sum to zero, thereby preserving the aggregate task gradient. The correction vanishes when $q = 0$ or when $\mathbf{p}_{t,i} = \alpha_{t,i}\mathbf{1}_r$ for every activated expert, recovering standard MoE backpropagation.

5 Experiments

5.1 Experimental Setup

Models and Datasets. We evaluate on three sparse MoE backbones: **OLMoE-1B-7B** (Muennighoff et al., 2025), **DeepSeek-V2-Lite** (DeepSeek-AI et al., 2024), and **Qwen3-30B-A3B** (Yang et al., 2025). All models are fine-tuned on the same 20K-example instruction mixture, covering general instruction following, mathematics, code, symbolic reasoning, and format following. The mixture is constructed from Tülu 3 (Lambert et al., 2025), FLAN-v2 (Longpre et al., 2023), NuminaMath (Li et al., 2024), OpenMathInstruct-1 (Toshniwal et al., 2024), and Magicoder (Wei et al., 2024). Detailed

Algorithm 1 Training an MoE with GradRouter

Input: MoE parameters $\boldsymbol{\theta}$, optimizer, and batches $\{\mathcal{B}_m\}_{m=1}^{M}$; rank r, allocation strength $q \in [0,1]$, entropy coefficient τ, refresh interval T, buffer capacities n_g, n_c, and numerical floor ϵ.
Output: Updated parameters $\boldsymbol{\theta}$.

Initialize per-layer gradient/calibration buffers and mark $(V, \widetilde{A})$ unavailable for each MoE layer ℓ.
for $m = 1, \ldots, M$ **do**
  Run standard forward/backward on $\mathcal{B}_m$ and retain $\{\mathbf{x}_t, \mathbf{g}_t, \mathcal{A}_t, \boldsymbol{\alpha}_t\}$.
  for each MoE layer ℓ **do**
    if $(V, \widetilde{A})$ is available **then**
      for each token t **do** ▷ Computed in batches
        $P_t \leftarrow \textsc{Sinkhorn}(\widetilde{A}_{\mathcal{A}_t,:}, \boldsymbol{\alpha}_t, \tau)$; normalize columns of P_t. ▷ Directional allocation; Eq. 8
        $\mathbf{p}_{t,i} \leftarrow P_{t,i,:}^{\top}, \quad \mathbf{g}_{t,i}^{\parallel} \leftarrow qV\,\mathrm{Diag}(\mathbf{p}_{t,i})V^{\top}\mathbf{g}_t, \ \forall i \in \mathcal{A}_t.$ ▷ Directional credit; Eq. 9
        $\mathbf{g}_t^{\mathrm{res}} \leftarrow \mathbf{g}_t - \sum_{i \in \mathcal{A}_t} \mathbf{g}_{t,i}^{\parallel}.$ ▷ Residual credit; Eq. 10
        $\widetilde{\mathbf{g}}_{t,i} \leftarrow \mathbf{g}_{t,i}^{\parallel} + \alpha_{t,i}\mathbf{g}_t^{\mathrm{res}}, \quad \Delta\mathbf{g}_{t,i} \leftarrow \widetilde{\mathbf{g}}_{t,i} - \alpha_{t,i}\mathbf{g}_t, \ \forall i \in \mathcal{A}_t.$ ▷ Conservative correction; Eq. 12
      end for
      Accumulate the parameter-gradient corrections induced by $\{\Delta\mathbf{g}_{t,i}\}$.
    else
      Retain the gradients from standard backpropagation.
    end if
    Update the layer-wise gradient/calibration buffers with detached samples, up to capacities n_g, n_c.
    if $(V, \widetilde{A})$ is unavailable **or** $m \bmod T = 0$, and sufficient samples are available **then**
      Form C_g from buffered gradients and $V \leftarrow \mathrm{TopEig}_r(C_g)$. ▷ Gradient basis; Eq. 3
      $A \leftarrow \textsc{EstimateAffinity}(\mathcal{C}_h, V)$. ▷ Eq. 5
      $\widetilde{A} \leftarrow \textsc{DoubleCenter}(\log\max(A, \epsilon))$. ▷ Eq. 7
      Cache $(V, \widetilde{A})$ for subsequent steps. ▷ Used from step $m+1$
    end if
  end for
  Apply global gradient clipping and update $\boldsymbol{\theta}$ with the optimizer.
end for

model configurations and the composition of the training mixture are provided in Appendix D.1 and Appendix D.2, respectively.

Baselines. We compare with three MoE training methods: ***(i)*** **AuxLoss** (Dai et al., 2024), which regularizes expert utilization through an auxiliary load-balancing loss; ***(ii)*** **Loss-Free Balancing** (Wang et al., 2024a), which adjusts expert-wise routing biases according to recent loads; ***(iii)*** **AFBM** (Guo et al., 2025), which combines auxiliary load balancing with expert-output orthogonality and routing-score variance objectives. Implementation details of these baselines are provided in Appendix D.3.

Evaluation. We evaluate the fine-tuned MoE models on 11 benchmarks covering general knowledge and reasoning, mathematical reasoning, and code generation. Specifically, we use MMLU (Hendrycks et al., 2021), MMLU-Pro (Wang et al., 2024b), BBH (Suzgun et al., 2023), GPQA (Rein et al., 2024), and LiveBench (White et al., 2025) for general knowledge and reasoning; GSM8K (Cobbe et al., 2021) and MATH500 (Lightman et al., 2024) for mathematical reasoning; and HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), HumanEval+, and MBPP+ (Liu et al., 2023) for code generation. We report accuracy or exact match for knowledge, reasoning, and mathematics, the official aggregate score for LiveBench, and pass@1 for code-generation benchmarks. Detailed evaluation protocols are provided in Appendix D.5.

Training Details. We fine-tune all models with LoRA (Hu et al., 2022) for three epochs, using rank $r_L = 32$, LoRA parameter $\alpha = 128$, a maximum sequence length of 1,024, and an effective batch size of 128, at a learning rate of 10^{-4}, batch size 128. More details are provided in Appendix D.4.

Hyper-parameters of GradRouter. GradRouter uses a gradient-subspace rank of $r_G = 512$ and refreshes the gradient basis and affinity estimates every five optimizer steps. We solve the local OT problem with temperature $\tau = 0.1$ for 24 Sinkhorn iterations, and cap each expert's correction-to-

standard-gradient ratio at 0.5. The realized LoRA-gradient correction is controlled toward a mean ratio of 0.10, with all affinity, correction, and publication EMAs disabled.

5.2 Downstream Performance

Tab. 1 evaluates GradRouter across 11 downstream benchmarks and three MoE backbones. While the gains are task dependent, GradRouter shows particularly consistent improvements on code generation: it improves HumanEval over the strongest baseline by 2.44 points on both OLMoE and Qwen3-30B-A3B, and HumanEval+ by 3.65 and 3.66 points, respectively. On DeepSeek-V2-Lite, the gains are more distributed, including a 1.40-point improvement on MATH500 together with improvements on MMLU-Pro, BBH, and LiveBench. These results suggest that directional learning-responsibility assignment can provide complementary benefits across different MoE architectures, with its strongest gains observed on code generation.

Table 1: **Downstream performance of instruction-tuned sparse MoE models.** Fine-tuned models use the shared 20K instruction mixture for three epochs. We report accuracy/exact match for knowledge, reasoning, and mathematics, pass@1 for code generation, and the official aggregate score for LiveBench (all in %; higher is better).

Method	Knowledge		Mathematics		Code generation				Reasoning / General		
	MMLU	MMLU-Pro	GSM8K	MATH500	HumanEval	HumanEval+	MBPP	MBPP+	BBH	GPQA	LiveBench
OLMoE-1B-7B-0125											
AuxLoss (Dai et al., 2024)	45.72	17.42	45.79	16.00	12.20	10.37	28.57	23.02	9.43	**24.75**	**9.66**
Loss-Free (Wang et al., 2024a)	43.93	**17.55**	45.41	17.20	14.63	10.98	26.98	22.22	**9.97**	20.20	9.45
AFBM (Guo et al., 2025)	24.85	16.81	1.36	16.60	13.41	10.37	**30.69**	**25.93**	8.12	13.64	9.44
GradRouter	**47.25**	17.34	**46.93**	**19.00**	**17.07**	**14.63**	**30.69**	25.40	8.65	21.21	9.41
DeepSeek-V2-Lite (16B-A2.4B)											
AuxLoss (Dai et al., 2024)	56.46	24.43	45.26	17.80	**36.59**	30.49	**59.79**	**49.74**	46.89	**30.81**	12.32
Loss-Free (Wang et al., 2024a)	56.29	24.38	**45.94**	19.20	35.37	**31.10**	58.20	48.41	46.92	29.29	12.71
AFBM (Guo et al., 2025)	53.67	12.47	32.30	12.80	33.54	27.44	54.50	45.77	13.50	26.77	10.40
GradRouter	**56.47**	**25.05**	44.96	**20.60**	**36.59**	30.49	59.52	49.21	**47.30**	30.30	**12.86**
Qwen3-30B-A3B-Base											
AuxLoss (Dai et al., 2024)	80.94	61.16	86.43	67.00	81.10	73.78	**85.98**	70.11	71.54	40.40	29.40
Loss-Free (Wang et al., 2024a)	80.59	**61.86**	**87.41**	**70.60**	80.49	72.56	85.71	**70.90**	**79.24**	**41.92**	**33.72**
AFBM (Guo et al., 2025)	79.14	58.48	83.09	52.20	76.22	68.90	77.25	64.02	40.82	37.37	25.79
GradRouter	**81.07**	61.32	86.43	69.80	**83.54**	**77.44**	83.60	67.99	77.44	39.39	32.83

5.3 Training Dynamics

Fig. 3 compares the training LM-loss trajectories across three MoE architectures. We could see that GradRouter trains stably on all three backbones, without the loss spikes or divergence that could arise from modifying the backward signal. On OLMoE and DeepSeek-V2-Lite, its loss broadly tracks the best-performing baselines during the early and middle stages of training and reaches a comparable or lower value near convergence. The separation is more pronounced on Qwen3-30B-A3B-Base, where GradRouter attains a visibly lower late-stage loss. In contrast, all baselines exhibit occasional loss spikes in the unsmoothed curves. These curves collectively demonstrate that directional credit reassignment remains compatible with stable task learning.

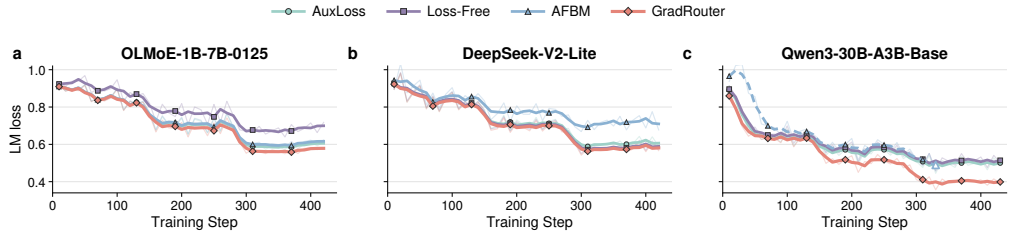

Figure 3: **Training Language-modeling (LM) loss trajectories of different training algorithms during fine-tuning across 3 MoE architectures.** Solid curves show a centered moving average over 30 training steps, while faint curves denote unsmoothed 10-step-bin averages. GradRouter converges stably across all backbones and achieves the lowest final training LM loss.

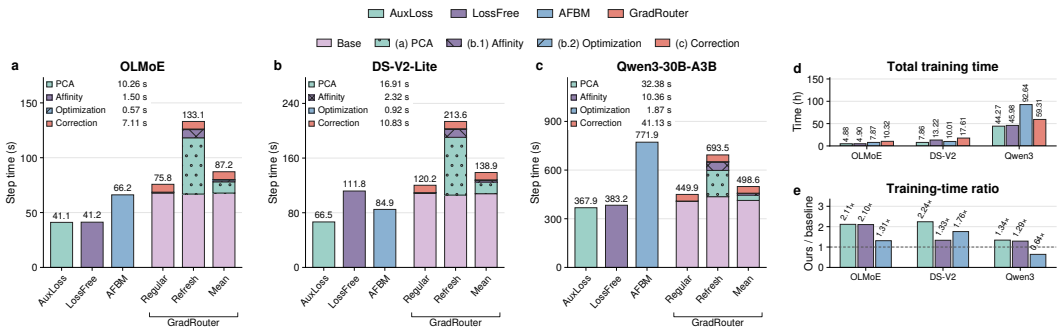

Figure 4: **Training-time overhead of GradRouter.** (a–c) Per-step time cost on 3 MoE architectures. **Regular** denotes standard GradRouter steps without computing new PCA basis and Expert-direction Affinity Matrix, **Refresh** denotes periodic calibration steps of updating the gradient basis and Expert-direction Affinity Matrix; **Mean** is the amortized per-step time over one refresh cycle ($T = 5$). Colors decompose the additional cost of GradRouter into PCA, affinity estimation, gradient correction, and the remaining base computation. (d) End-to-end training time. (e) GradRouter-to-baseline training-time ratio, with the dashed line indicating an equal training time.

5.4 Training-Time Overhead

Fig. 4 quantifies the computational cost of GradRouter at both the step and full-training levels. As demonstrated in sub-figures (a)-(c), the step-wise time decomposition attributes most of the additional cost to the two operations that act on high-dimensional gradients: estimating the PCA basis and applying the gradient correction. Affinity estimation and the local optimal-transport optimization are the least expensive additional component across the three backbones. Refreshing the basis and affinity every $T = 5$ steps amortizes the calibration cost, but refresh steps remain substantially slower than regular GradRouter steps, which take almost 2x longer time. Further efficiency improvements could therefore focus on lightweight or incremental basis estimation, adaptive refresh schedules, and more efficient gradient-correction implementations. As demonstrated in sub-figure (d) and (e), GradRouter is approximately $1.3\times$–$2.2\times$ end-to-end training time compared with the baselines. The additional training overhead remains a practical limitation of GradRouter. As an initial exploration of directional learning responsibility assignment, the current design prioritizes establishing the effectiveness of this mechanism and leaves substantial room for computational optimization. More efficient approximations and implementations of these components constitute an important direction for future work.

5.5 Supplementary Experiments and Analyses

We provide the following experiments and analyses for completeness in Appendix: (i) downstream performance of combining AuxLoss and GradRouter in Appendix C.1; (ii) a hyper-parameter sensitivity analysis of GradRouter in Appendix C.2.

6 Conclusion

In this work, we revisit expert specialization in sparse MoEs from the perspective of backward learning responsibility. We show that conventional routing directly couples forward contribution weights with backward credit assignment, causing co-activated experts to receive collinear output-side gradients and limiting their ability to specialize along different learning directions. To address this issue, we propose **GradRouter**, which decomposes the output-gradient space into principal directions and assigns directional learning responsibility to co-activated experts according to their capabilities, while preserving the original gradient on the aggregated expert output. Experiments demonstrate its effectiveness and highlight the directional learning responsibility assignment as an effective and scalable complement to existing forward-side expert specialization mechanisms.

AI USE STATEMENT

In this work, we used generative AI tools primarily for improving the grammar, clarity, and readability of text drafted by the authors, and were occasionally used to troubleshoot implementation issues and suggest possible fixes. We also used these tools to find and organize relevant background materials. Any literature suggestions or summaries served only as starting points, and the authors checked the relevant references and technical claims against the original sources.

The research questions, core ideas, proposed method, experimental design, experiments, interpretation of the results, and scientific conclusions were developed and determined by the authors. Generative AI was not used to produce experimental results or make scientific decisions. All AI-assisted material was reviewed and revised by the authors, who take full responsibility for the final manuscript and its associated code, figures, and other research artifacts.

BIBLIOGRAPHY

Sagi Ahrac, Noya Hochwald, and Mor Geva. Routers learn the geometry of their experts: Geometric coupling in sparse mixture-of-experts. *arXiv preprint arXiv:2605.12476*, 2026.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

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 *Advances in Neural Information Processing Systems*, volume 35, pp. 34600–34613, 2022.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, et al. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

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 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1280–1297, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.70.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui

Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

Giang Do, Hung Le, and Truyen Tran. Simsmoe: Toward efficient training mixture of experts via solving representational collapse. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 2012–2025, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-naacl.107.

Sugyeong Eo, Jung Jun Lee, Chanjun Park, and Heuiseok Lim. Mixture-of-clustered-experts: Advancing expert specialization and generalization in instruction tuning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 14201–14212. Association for Computational Linguistics, 2025.

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*, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

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.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994. doi: 10.1162/neco.1994.6.2.181.

Hyunjun Kim. Geometric regularization in mixture-of-experts: The disconnect between weights and activations. *arXiv preprint arXiv:2601.00457*, 2026.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025.

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. In *International Conference on Learning Representations*, 2021.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. NuminaMath. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/AI-MO/NuminaMath-CoT. CoT and TIR training data; TIR: https://huggingface.co/datasets/AI-MO/NuminaMath-TIR.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, 2024.

Boan Liu, Liang Ding, Li Shen, Keqin Peng, Yu Cao, Dazhao Cheng, and Dacheng Tao. Diversifying the mixture-of-experts representation for language models with orthogonal optimizer. In *European Conference on Artificial Intelligence*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, pp. 2966–2973. IOS Press, 2024.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 21558–21572. Curran Associates, Inc., 2023.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. The Flan collection: Designing data and methods for effective instruction tuning. In *Proceedings of the International Conference on Machine Learning*, 2023.

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*, volume 2025, pp. 62061–62121, 2025.

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 Annual Meeting of the Association for Computational Linguistics*, pp. 5005–5018. Association for Computational Linguistics, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, et al. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

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*, 2024a.

Xi Wang, Soufiane Hayou, and Eric Nalisnick. The myth of expert specialization in MoEs: Why routing reflects geometry, not necessarily domain expertise. *arXiv preprint arXiv:2604.09780*, 2026.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, pp. 95266–95290. Curran Associates, Inc., 2024b.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Empowering code generation with OSS-Instruct. In *Proceedings of the International Conference on Machine Learning*, 2024.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Sandha, Siddartha Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. Livebench: A challenging, contamination-limited llm benchmark. In *International Conference on Learning Representations*, volume 2025, 2025.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

APPENDIX

A NOTATION DEFINITION

Unless stated otherwise, we describe a single MoE layer and suppress the layer index ℓ. Table A1 lists only the notation used repeatedly across the formulation; local indices and one-off auxiliary quantities are defined at first appearance. Throughout the equations, $\boldsymbol{\alpha}_t$ denotes the normalized routing weights. Accordingly, the router-logit vector annotated as $\boldsymbol{\alpha}_t^{(\ell)}$ in Fig. 1 should be read as $\mathbf{z}_t^{(\ell)}$.

Table A1: Core notation for sparse-MoE routing and GradRouter. Symbols used only locally in a derivation, implementation detail, or complexity analysis are omitted.

Symbol	Definition
Sparse-MoE forward and backward pass	
N, K	Numbers of routed experts and experts activated per token, respectively.
d, d_{ff}	Hidden/output width and expert intermediate width.
$\mathbf{x}_t \in \mathbb{R}^d$	Input hidden state of token t.
$\mathcal{A}_t,\ \alpha_{t,i}$	Top-K active-expert set and the normalized routing weight of expert $i \in \mathcal{A}_t$, with $\sum_{i\in\mathcal{A}_t} \alpha_{t,i} = 1$.
$f_i(\cdot;\boldsymbol{\theta}_i),\ \mathbf{e}_{t,i}$	Expert i and its output $\mathbf{e}_{t,i} = f_i(\mathbf{x}_t;\boldsymbol{\theta}_i)$.
$\mathbf{y}_t$	Aggregated MoE output $\mathbf{y}_t = \sum_{i\in\mathcal{A}_t} \alpha_{t,i}\mathbf{e}_{t,i}$.
$\mathcal{L},\ \mathbf{g}_t$	Task loss and output-side gradient $\mathbf{g}_t = \partial\mathcal{L}/\partial\mathbf{y}_t$.
GradRouter	
$r,\ r_{\text{L}}$	Gradient-basis rank and LoRA rank, respectively.
$\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_r]$	Orthonormal basis of the leading output-gradient directions, $\mathbf{V}^\top\mathbf{V} = \mathbf{I}_r$.
$A_{ij},\ \widetilde{A}_{ij}$	Expert–direction affinity and its log-compressed, double-centered score. The affinity averages $\|\mathbf{J}_i(\mathbf{x})^\top\mathbf{v}_j\|_2^2$ over the calibration set, where $\mathbf{J}_i = \partial f_i/\partial\boldsymbol{\theta}_i$.
$\mathbf{P}_t,\ \mathbf{p}_{t,i}$	Token-local assignment matrix and its row for expert i (written as a column vector); columns of $\mathbf{P}_t$ sum to one and row i sums to $r\alpha_{t,i}$.
$q,\ \tau,\ T$	Directional-allocation strength, assignment temperature, and basis/affinity refresh interval.
$\mathbf{M}_{t,i}$	Directional credit matrix $\mathbf{M}_{t,i} = q\,\mathrm{Diag}(\mathbf{p}_{t,i})$.
$\mathbf{g}_{t,i}^{\parallel},\ \mathbf{g}_t^{\text{res}}$	Assigned principal-subspace gradient $\mathbf{V}\mathbf{M}_{t,i}\mathbf{V}^\top\mathbf{g}_t$ and residual gradient $(\mathbf{I}_d - q\mathbf{V}\mathbf{V}^\top)\mathbf{g}_t$.
$\widetilde{\mathbf{g}}_{t,i}$	Final output-side gradient assigned to expert i, $\widetilde{\mathbf{g}}_{t,i} = \mathbf{g}_{t,i}^{\parallel} + \alpha_{t,i}\mathbf{g}_t^{\text{res}}$, satisfying $\sum_{i\in\mathcal{A}_t}\widetilde{\mathbf{g}}_{t,i} = \mathbf{g}_t$.

B DETAILS OF GRADROUTER

B.1 LAYER-WISE PCA ANALYSIS OF MOE OUTPUT GRADIENTS

Setup. We analyze the pretrained base checkpoints OLMoE-1B-7B-0125, DeepSeek-V2-Lite, and Qwen3-30B-A3B-Base, all with hidden dimension $d = 2,048$. We select the first 16 documents in a fixed, shuffled FineWeb shard that contain at least 257 tokens under all three native tokenizers. Each model receives the first 257 tokens of each document, yielding 256 valid next-token prediction positions per document and 4,096 gradient vectors per layer. We inspect layers $\{4, 8, 12, 16\}$, $\{7, 13, 20, 27\}$, and $\{12, 24, 36, 48\}$, respectively (one-based), spanning approximately quarter-depth to the final layer. With weights frozen and dropout disabled, FP32 backpropagation computes $g_t = \partial\mathcal{L}/\partial y_t$ at the aggregated MoE output, where $\mathcal{L}$ sums causal cross-entropies without auxiliary losses.

Energy spectrum. Stacking these gradients gives $G \in \mathbb{R}^{4096\times 2048}$. We use uncentered PCA, with neither mean subtraction nor per-token normalization, and report

$$E(r) = \frac{\sum_{j=1}^{r}\lambda_j(G^\top G)}{\|G\|_F^2}, \qquad \lambda_1 \geq \cdots \geq \lambda_d \geq 0. \tag{A1}$$

The denominator includes the full spectrum. Figure A1 shows that rank 512 (25% of d) retains 82.0–91.1%, 75.2–83.9%, and 83.4–87.9% of the energy on OLMoE, DeepSeek-V2-Lite, and Qwen3-30B-A3B, respectively. Across the inspected layers, retaining 90% requires 463–899 directions: the gradients are compressible, but a few directions do not suffice.

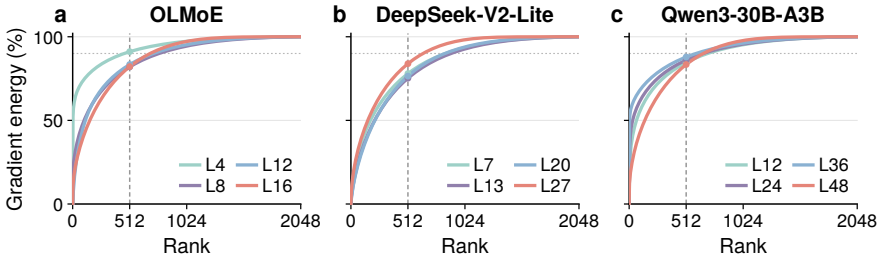

Figure A1: **Layer-wise output-gradient energy.** Each curve uses 4,096 valid FineWeb token positions; legends give one-based layer indices. Dashed vertical lines mark rank 512; dotted horizontal lines mark 90% energy.

Scope and relation to GradRouter. These are sample-energy measurements at pretrained checkpoints, not a test of temporal stability or held-out reconstruction. Large-norm tokens can strongly affect the spectrum: removing the largest-norm 1% lowers OLMoE layer 4's $E(512)$ from 91.1% to 76.7%. The figure uses raw output gradients. For GradRouter's actual credit-basis input $\tilde{g}_t = m_t g_t$, with m_t the sum of selected routing weights, rank-512 coverage is 83.0–89.5%, 80.3–84.3%, and 83.4–87.9%, respectively; Qwen's normalized top-k weights make $m_t \simeq 1$.

C Additional Experiments

C.1 Combination with Auxiliary Load Balancing Loss

Fig. A2 compares AuxLoss, GradRouter, and GradRouter+AuxLoss on OLMoE, using an auxiliary-loss coefficient of 10^{-3}. The combined objective consistently improves over using AuxLoss alone, showing that GradRouter remains effective when conventional load-balancing regularization is present. However, the combination generally underperforms GradRouter alone across the evaluated benchmarks, indicating that the two objectives are not fully complementary. One possible reason is that AuxLoss primarily encourages balanced expert utilization, whereas GradRouter reallocates directional credit according to expert–direction affinity. It suggests that more compatible load-balancing strategies could better complement directional credit assignment.

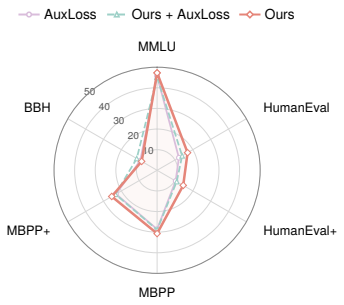

Figure A2: **GradRouter with auxiliary load balancing on OLMoE.**

C.2 Hyperparameter sensitivity of GradRouter.

We vary the target correction ratio, direction-refresh interval T, and OT temperature τ individually on OLMoE (**Fig.** A3). All configurations use three training epochs, learning rate 10^{-4}, subspace rank 512, and $q = 0.09876371$. We test target correction ratios $\{0.05, 0.10, 0.20\}$, refresh intervals $H \in \{5, 10, 20\}$, and OT temperatures $\tau \in \{0.05, 0.10, 0.20\}$. Across the six alternatives, changes from the default remain within 2.43 percentage points on the four plotted benchmarks. Increasing H from 5 to 20 improves GSM8K and MBPP+ by 0.68 and 1.58 points, respectively, while reducing HumanEval+ by 2.43 points. No alternative improves all four plotted benchmarks simultaneously, indicating task-dependent hyperparameter preferences.

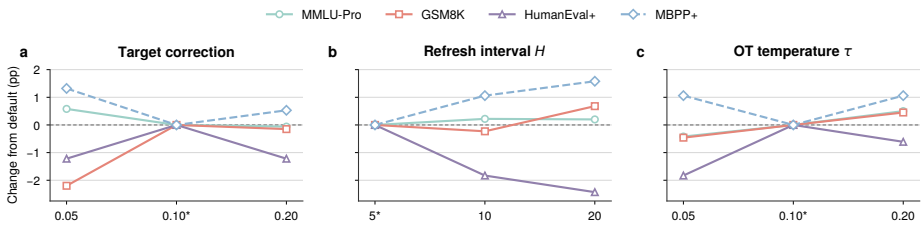

Figure A3: **Sensitivity to directional-credit hyperparameters on OLMoE.** Each panel varies one hyperparameter and reports the change from the default configuration in percentage points. Defaults are target correction 0.10, refresh interval $H = 5$, and OT temperature $\tau = 0.10$, marked by an asterisk. All panels share the same vertical scale; the dashed horizontal line denotes the default score. Curves show MMLU-Pro, GSM8K, HumanEval+, and MBPP+. Each point is a single-run estimate.

Table A2: Specifications of the pretrained MoE backbones used in our experiments. The checkpoint names link to the corresponding Hugging Face repositories.

Model	Total / Active	Routed	Shared	Top-k	Checkpoint
OLMoE-1B-7B	7B / 1B	64	0	8	`OLMoE-1B-7B-0125`
DeepSeek-V2-Lite	15.7B / 2.4B	64	2	6	`DeepSeek-V2-Lite@604d5664`
Qwen3-30B-A3B-Base	30.5B / 3.3B	128	0	8	`Qwen3-30B-A3B-Base@1b75feb7`

D Experimental Settings

D.1 Models

We use the official pretrained checkpoints summarized in Table A2. For each backbone, we retain its native expert architecture and routing top-k configuration. To ensure reproducibility, the DeepSeek-V2-Lite and Qwen3-30B-A3B-Base checkpoints are pinned to revisions `604d5664` and `1b75feb7`, respectively.

OLMoE-1B-7B (Muennighoff et al., 2025) is a fully open sparse Mixture-of-Experts language model with 7B total parameters and approximately 1B parameters activated for each token. Each MoE layer contains 64 routed experts, of which the top eight are selected for each token, combining a large model capacity with comparatively low inference cost. We use the January 2025 base release, OLMoE-1B-7B-0125, which provides access to the model weights and the associated training artifacts.

DeepSeek-V2-Lite (DeepSeek-AI et al., 2024) is a 15.7B-parameter MoE language model that activates approximately 2.4B parameters for each token. It combines Multi-head Latent Attention with the DeepSeekMoE architecture; each MoE layer contains 64 routed experts and two shared experts, with six routed experts selected per token. This design reduces both activation cost and key–value cache overhead while retaining the capacity of a substantially larger model. We use the official base checkpoint pinned to revision `604d5664`.

Qwen3-30B-A3B (Yang et al., 2025) is a pretrained MoE backbone with 30.5B total parameters and 3.3B parameters activated for each token. The model comprises 48 layers and 128 routed experts, with the top eight experts selected at every MoE layer, and supports a native context length of 32,768 tokens. Its sparse activation provides high model capacity while limiting the computation required for each token. We use the official base checkpoint pinned to revision `1b75feb7`.

D.2 Training Data

We construct `balanced_20k` from the training splits of Tulu 3 SFT mixture, NuminaMath-CoT, NuminaMath-TIR, OpenMathInstruct-1, and Magicoder-OSS-Instruct-75K (Lambert et al., 2025; Li

et al., 2024; Toshniwal et al., 2024; Wei et al., 2024). FLAN-v2 (Longpre et al., 2023) is included through the `flan_v2_converted` subset of Tulu 3. Table A3 lists the domain allocation and source composition.

Table A3: Composition of the 20K instruction mixture before model-specific tokenization and length filtering.

Domain	Examples	Sources
General instruction	6,000	Tülu 3 general instruction subsets
Mathematics	6,000	NuminaMath-CoT (5,517), NuminaMath-TIR (421), OpenMathInstruct-1 (62)
Code	5,000	Magicoder (4,207), Tülu 3 code subsets (793)
Symbolic reasoning	2,000	Tülu 3: FLAN-v2 (667), SciRIFF (667), TableGPT (666)
Format following	1,000	Tülu 3: PersonaHub instruction-following subset

Mixture sampling uses seed 42. Examples are serialized with the model-specific training template, and the loss is computed on the final assistant response. Sequences longer than $1{,}024$ tokens are discarded. The resulting training sets contain $18{,}183$ examples for OLMoE, $18{,}065$ for DeepSeek-V2-Lite, and $18{,}315$ for Qwen3.

D.3 Baselines

AuxLoss (Dai et al., 2024) augments router learning with an expert load-balancing loss. The auxiliary coefficient is 10^{-3} for OLMoE and DeepSeek-V2-Lite, and 10^{-4} for Qwen3.

Loss-Free Balancing (Wang et al., 2024a) adds expert-wise biases to the scores used for top-k selection. The biases are updated using the sign of the discrepancy between the target and observed expert loads. We use a bias update rate of 10^{-3} and an auxiliary-loss coefficient of zero for all three backbones.

AFBM (Guo et al., 2025) combines auxiliary load balancing, expert-output orthogonality, and routing-score variance objectives. The auxiliary coefficient is 10^{-3}. For DeepSeek-V2-Lite and Qwen3, the variance and orthogonality calibration targets are both 10^{-3}. For OLMoE, the variance and orthogonality coefficients are fixed to 1 and 10^{-4}, respectively; the orthogonality computation uses FP32 and the combined auxiliary gradient is scaled by 0.5. The implementation of this method is referred to code released by authors: Auxloss-For-Advancing-Expert-Specialization.

Our method. The local assignment configuration uses rank 512, a refresh interval of 5 steps, affinity temperature 0.1, and 24 assignment iterations. The correction target is 0.10 and the correction-ratio cap is 0.5. The assignment parameter q is 0.09876371 for OLMoE and Qwen3, and 0.06 for DeepSeek-V2-Lite.

D.4 Training Configuration

All runs use three epochs, a maximum sequence length of $1{,}024$, and an effective batch size of 128. LoRA (Hu et al., 2022) uses rank $r = 32$, parameter $\alpha = 128$, and dropout 0.1. Adapters are applied to the configured attention and feed-forward projections, with router LoRA enabled. AdamW uses $(\beta_1, \beta_2) = (0.9, 0.999)$ and $\epsilon = 10^{-8}$, with 20 warmup steps and gradient clipping at 1.0.

D.5 Evaluation Protocols

Knowledge and reasoning. We use MMLU (Hendrycks et al., 2021) with 5-shot multiple-choice scoring, MMLU-Pro (Wang et al., 2024b) with 5-shot chain-of-thought (CoT) prompting, BBH (Suzgun et al., 2023) with 3-shot CoT prompting, and the 198-question Diamond subset of GPQA (Rein et al., 2024) with 5-shot CoT prompting. LiveBench (White et al., 2025) covers reasoning, mathematics, coding, language, instruction following, and data analysis; we aggregate scores by task, then by category, and finally average over the six categories.

Mathematics. GSM8K (Cobbe et al., 2021) uses 8-shot prompting and exact-match scoring on 1,319 test questions. MATH500 (Lightman et al., 2024) uses the 500-question MATH subset, 0-shot CoT prompting, and final-answer correctness.

Code generation. HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) are evaluated with EvalPlus (Liu et al., 2023), using 164 HumanEval problems and 378 MBPP problems. We report pass@1 on both the original and extended test suites, yielding HumanEval, HumanEval+, MBPP, and MBPP+ scores.

Input formatting. Fine-tuned OLMoE and DeepSeek-V2-Lite use model-specific SFT templates. Qwen3-30B-A3B-Base uses raw-completion prompts. The original pretrained checkpoints use raw-completion prompts. Few-shot generative evaluation with SFT templates represents demonstrations as separate dialogue turns.

E COMPUTATIONAL COST OF GRADROUTER

We follow Table A1 and the three stages of the method figure. Costs are additional to standard backpropagation, per layer ℓ and worker. Unmarked rows in Table A4 are per optimizer step ($n_s = an$ tokens); † denotes a refresh, whose cost is amortized over T steps. Defaults are $(r, r_{\mathrm{L}}, n_g^{\mathrm{local}}, n_c, T, n_{\mathrm{SK}}) = (512, 32, 4096, 16, 5, 24)$ for a full history buffer.

Table A4: GradRouter computation, grouped by Q1–Q3. Costs marked † occur once per refresh; all other costs are per optimizer step.

Stage	Computation	Arithmetic cost
Q1	Historical moment $\mathbf{G}^\top\mathbf{G}$ †	$O(n_g^{\mathrm{local}} d^2)$
	Full eigensolve and top-r QR †	$O(d^3 + dr^2)$
Q2	Effective weights / calibration †	$O(Ndd_{\mathrm{ff}}(r_{\mathrm{L}} + n_c))$
	Analytic affinity: $\mathbf{W}_{\mathrm{down},i}^\top\mathbf{V}$ and squared-response aggregation †	$O(Nd_{\mathrm{ff}}r(d + n_c))$
	Scores $\mathbf{S}$ and active-row gather	$O(aNr + n_sKr)$
	Sinkhorn / column normalization	$O(n_{\mathrm{SK}}n_sKr)$
Q3	Coordinates $\boldsymbol{\xi}_t = \mathbf{V}^\top\mathbf{g}_t$	$O(n_sdr)$
	Residual subtraction: $\boldsymbol{\delta}_{t,i}$	$O(n_sKr)$
	Fused LoRA correction, norm control, and writeback	$O(n_sKr_{\mathrm{L}}(d_{\mathrm{ff}} + r) + Ndrr_{\mathrm{L}} + Nr_{\mathrm{L}}(d + d_{\mathrm{ff}}))$

VJP-Based Simplification of Affinity Here we show how to use vector–Jacobian product (VJP) and bias-free SwiGLU architecture to avoid the explicit calculation of Jacobian. For expert i and a basis direction $\mathbf{v}_k \in \mathbb{R}^d$, let $\mathbf{y}_i(\mathbf{x})$ denote the expert output for input $\mathbf{x}$, and let $\boldsymbol{\theta}_i^{g,u}$ denote the parameters of MoE's gate and up projections. With Jacobian $J_i^{g,u}(\mathbf{x}) = \frac{\partial \mathbf{y}_i(\mathbf{x})}{\partial \boldsymbol{\theta}_i^{g,u}}$, the product can be represented as:

$$J_i^{g,u}(x)^T v_k = \nabla_{\boldsymbol{\theta}_i^{g,u}} \left(\mathbf{v}_k^\top \mathbf{y}_i(\mathbf{x})\right). \tag{A2}$$

Computing Eq. A2 does not require explicitly materializing the Jacobian. For the bias-free SwiGLU experts considered in our model, Eq. A2 admits an exact closed-form expression, allowing the affinity to be evaluated using only standard matrix and element-wise operations.

Closed-form Equivalence. For clarity, we omit the expert index i. A bias-free SwiGLU MoE expert is given by

$$\begin{aligned} \mathbf{g} &= W_g\mathbf{x}, \qquad \mathbf{u} = W_u\mathbf{x}, \\ \mathbf{s} &= \mathrm{SiLU}(\mathbf{g}), \qquad \mathbf{h} = \mathbf{s} \odot \mathbf{u}, \\ \mathbf{y} &= W_d\mathbf{h}, \end{aligned} \tag{A3}$$

where $W_g, W_u \in \mathbb{R}^{f\times d}$ and $W_d \in \mathbb{R}^{d\times f}$.

Consider an output direction $\mathbf{v} \in \mathbb{R}^d$ and propagate it through the down projection:

$$\mathbf{r} := W_d^\top \mathbf{v} \in \mathbb{R}^f. \tag{A4}$$

Using the chain rule, the derivatives of $\ell_\mathbf{v} := \mathbf{v}^\top \mathbf{y}$ with respect to the two intermediate states $\mathbf{u}$ and $\mathbf{g}$ are

$$\begin{aligned} \frac{\partial \ell_v}{\partial \mathbf{u}} &= \mathbf{r} \odot \mathbf{s}, \\ \frac{\partial \ell_v}{\partial \mathbf{g}} &= \mathbf{r} \odot \mathbf{u} \odot \mathrm{SiLU}'(\mathbf{g}). \end{aligned} \tag{A5}$$

Since $\mathbf{u} = W_u\mathbf{x}$ and $\mathbf{g} = W_g\mathbf{x}$, the gradients with respect to the up and gate projection matrices are outer products:

$$\begin{aligned} \nabla_{W_u} \ell_v &= (\mathbf{r} \odot \mathbf{s})\, \mathbf{x}^\top, \\ \nabla_{W_g} \ell_v &= \big(\mathbf{r} \odot \mathbf{u} \odot \mathrm{SiLU}'(\mathbf{g})\big)\, \mathbf{x}^\top. \end{aligned} \tag{A6}$$

The key observation is that these parameter gradients are rank-one outer products. Using the identity of Frobenius outer products:

$$\|\mathbf{a}\mathbf{b}^T\|_F^2 = \|\mathbf{a}\|_2^2 \|\mathbf{b}\|_2^2, \tag{A7}$$

the squared norm of the vector–Jacobian product $\nabla_{\boldsymbol{\theta}_i^{g,u}} \ell_\mathbf{v} = \nabla_{\boldsymbol{\theta}_i^{g,u}} (\mathbf{v}^\top \mathbf{y})$ can be written as

$$\begin{aligned} \|J^{g,u}(\mathbf{x})^\top \mathbf{v}\|_2^2 &= \|\nabla_{\boldsymbol{\theta}^{g,u}} \ell_\mathbf{v}\|_2^2 && \text{(A8)} \\ &= \|\nabla_{W_u} \ell_v\|_F^2 + \|\nabla_{W_g} \ell_v\|_F^2 \\ &= \|\mathbf{x}\|_2^2 \left[\|\mathbf{r} \odot \mathbf{s}\|_2^2 + \|\mathbf{r} \odot \mathbf{u} \odot \mathrm{SiLU}'(\mathbf{g})\|_2^2\right] \\ &= \|\mathbf{x}\|_2^2 \sum_{j=1}^{f} r_j^2 \left[s_j^2 + \big(u_j\, \mathrm{SiLU}'(g_j)\big)^2\right]. && \text{(A9)} \end{aligned}$$

Eq. A9 is a proxy to the Jacobian-based quantity in Eq. 4 under the considered bias-free SwiGLU architecture. In other words, the Jacobian notation is only required for defining the geometric quantity; its value can be evaluated analytically without explicitly constructing $J^{g,u}$ or performing an individual backward pass for each direction.

This formulation directly corresponds to our implementation. The calibration inputs are first projected through W_g and W_u, while all basis directions are projected simultaneously through $W_d^\top V$. The element-wise factor $\mathbf{s}^{\odot 2} + (\mathbf{u} \odot \mathrm{SiLU}'(\mathbf{g}))^{\odot 2}$ is then contracted with $(W_d^\top \mathbf{V})^{\odot 2}$ along the intermediate dimension, followed by multiplication with $\|\mathbf{x}\|_2^2$ and averaging over calibration tokens. Thus, the complete affinity matrix is obtained using batched matrix operations without storing per-token parameter gradients or materializing the potentially very large Jacobian.

Memory and execution. Persistent device storage is $O(n_s(d + Kd_{\mathrm{ff}}) + dr + Nr + nN)$ for retained inputs/gradients, router scores, basis, and affinity; LoRA buffers add $O(Nr_{\mathrm{L}}(r + d + d_{\mathrm{ff}}))$. OT workspace is $O(nKr)$, independent of n_{SK}; affinity workspace is $O(dd_{\mathrm{ff}} + d_{\mathrm{ff}}r + n_c d_{\mathrm{ff}})$ when experts are processed sequentially. PCA uses $O(n_g^{\mathrm{local}} d + d^2 + dr)$ device workspace and $O(d^2)$ CPU workspace, with $O(n_g^{\mathrm{local}} d)$ host history. The current dense credit check adds $O(nKdr)$ work and $O(nKd + nKr)$ workspace for the first microbatch. History collection transfers $O(n_s d)$ values to the host before subsampling. Measurements include CPU eigensolves, transfers, and diagnostics.