RC-GEM: FINE-GRAINED EXPERT AGGREGATION VIA ROUTING-CONDITIONED SUBSPACE MODULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

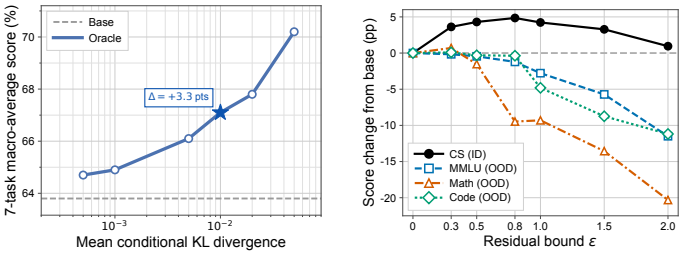
Sparse Mixture-of-Experts (MoE) models use routing scores to both select experts and weight their outputs, assigning each selected expert the same weight across all representation dimensions. We first probe the adaptation potential of this aggregation scheme through two diagnostic studies. On OLMoE, a ground-truth oracle shows that a small redistribution of expert weights (mean conditional KL ≈ 0.01) improves the seven-benchmark average by 3.3 points, while a learned residual aggregation controller shows that larger scalar reallocations eventually reduce in-distribution (ID) gains and degrade out-of-distribution (OOD) performance. Motivated by these findings, we propose RC-GEM (Routing-Conditioned Granular Expert Modulation), a parameter-efficient adaptation framework that augments standard expert-level weighting with fine-grained modulation over learned subspaces. RC-GEM preserves pretrained expert selection and aggregation weights, while using routing signals to condition subspace-wise modulation within each selected expert. Across OLMoE and Qwen1.5-MoE, RC-GEM approaches or surpasses full fine-tuning in average ID performance with fewer than 0.1% trainable parameters, while largely preserving pretrained OOD capabilities. Compared with existing MoE adaptation methods and conventional PEFT baselines, RC-GEM achieves a favorable balance between adaptation performance and generalization, highlighting aggregation granularity as an important direction for efficient MoE adaptation. Our implementation is available at `https://anonymous.4open.science/r/RC-GEM-F5F4`.

1 INTRODUCTION

Sparse Mixture-of-Experts (MoE) has emerged as a key architecture for scaling large language models (LLMs), expanding model capacity without a proportional increase in per-token computation (Shazeer et al., 2017; Fedus et al., 2022). Recent MoE-based LLMs, including Qwen3 MoE (Yang et al., 2025), gpt-oss (OpenAI, 2025), and GLM-5 (GLM-5 Team, 2026), have demonstrated strong performance across a wide range of challenging tasks. The router governs both expert selection and output aggregation, typically using the same scores to select experts and weight their outputs (Shazeer et al., 2017). However, routing scores that effectively identify useful experts may still leave room for improving how their outputs are weighted and combined. This motivates us to investigate whether adapting expert aggregation can better exploit selected experts without modifying expert selection.

Our diagnostic results on OLMoE (Muennighoff et al., 2025) suggest substantial headroom in expert aggregation under fixed expert selection. We design a label-assisted aggregation oracle (Section 4.1) that uses ground-truth labels to optimize aggregation weights for context tokens in the penultimate MoE layer. All pretrained parameters remain frozen, and expert selections stay unchanged except in the final layer, where routing may shift after intervention. As shown in Figure 1, the seven-benchmark average rises from 63.8 to 67.1 at optimization snapshots with mean conditional KL divergence of approximately 0.01. This 3.3-point improvement suggests that modest aggregation shifts, even within a single layer, can yield downstream gains. To examine whether such improvements can be achieved through learnable aggregation, we train a Residual Aggregation Controller (RAC) on Commonsense-15K (Hu et al., 2023) (Section 4.2).

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.



(a) Aggregation Headroom (b) ID–OOD Trade-off

Figure 1: Two aggregation diagnostics on OLMoE. (a) Oracle reweighting reveals aggregation headroom under constrained redistribution, achieving a 3.3-point gain with a mean conditional KL of approximately 0.01. (b) RAC shows an ID–OOD trade-off as residual reweighting becomes increasingly unconstrained.

As shown in Figure 1(b), bounding the log-weight residual amplitude at $\varepsilon = 0.3$ improves ID commonsense reasoning performance from 64.44 to 68.06 while largely preserving OOD performance. Increasing this bound to $\varepsilon = 0.8$ further raises ID performance to 69.29 but degrades OOD performance, most notably lowering the Math score from the pretrained baseline of 27.84 to 18.37. Meanwhile, the additional ID gain is also diminishing. Together, these findings motivate retaining the pretrained allocation as a reference for adaptation, with the aim of improving the aggregated representations for downstream tasks without unduly disrupting learned token–expert allocation patterns.

Retaining the pretrained allocation leaves room to adapt expert aggregation at finer granularity. Existing MoE adaptation methods adjust routing decisions or expert-level contribution weights (Li et al., 2026; 2025; Vahidi et al., 2026), but still assign a single coefficient to each routed expert output, leaving finer-grained control unexplored. Recent studies show that routing signals capture semantic information beyond expert selection and provide complementary information to hidden representations (Li & Zhou, 2025). Further evidence suggests that expert-selection patterns encode input-dependent information even at a discrete level (Nuriyev & Kulp, 2026). Together, these findings suggest that routing signals provide an informative basis for conditioning fine-grained expert adaptation.

To this end, we propose Routing-Conditioned Granular Expert Modulation (RC-GEM), a post-training method that enables fine-grained expert aggregation over learned representation subspaces while preserving the router's original aggregation weights as anchors. As illustrated in Figure 2, RC-GEM learns layer-wise subspaces and uses routing signals to assign bounded token-specific residual weights to expert contributions across subspaces, extending aggregation beyond a single scalar coefficient. Only lightweight modulation heads and subspace bases are optimized, while all pretrained parameters remain frozen. As shown in Table 1, RC-GEM approaches FFT performance on ID commonsense reasoning across OLMoE and Qwen1.5-MoE (Qwen Team, 2024) with only 0.04–0.09% trainable parameters, while consistently outperforming parameter-comparable LoRA (Hu et al., 2022) on three OOD domains, with largest gains on mathematical reasoning.

2 Related Work

Efficient Adaptation of MoE Models. Recent works explore adapting pretrained MoE models by leveraging sparse expert structures. RoMA (Li et al., 2026) aligns routing weights with task manifolds, while DirMoE (Vahidi et al., 2026) separates expert selection from contribution allocation. Other MoE-specific approaches, such as ACE (Lee et al., 2026) and PERFT-E (Liu et al., 2026), introduce expert-aware adaptation mechanisms. Test-time optimization methods such as C3PO (Li et al., 2025) further explore expert re-mixing without parameter updates. Meanwhile, generic PEFT methods such as Prefix Tuning (Li & Liang, 2021) and LoRA (Hu et al., 2022) reduce adaptation costs but do not exploit MoE-specific aggregation mechanisms. However, existing approaches mainly operate at the expert level, leaving finer-grained modulation of expert contributions within learned subspaces unexplored.

Routing Signals Beyond Expert Selection. Beyond their role in expert selection, recent studies show that MoE routing signals encode token-dependent semantic information. Li & Zhou (2025) demonstrate that routing weights serve as effective off-the-shelf embeddings complementary to hidden representations, while Nuriyev & Kulp (2026) show that discrete expert-selection patterns encode input-dependent information. These findings suggest routing signals as an informative conditioning source for token-specific adaptation.

3 PRELIMINARIES

A sparse Mixture-of-Experts (MoE) layer contains N experts and activates only k of them for each input token (Shazeer et al., 2017; Lepikhin et al., 2021). Given an input representation $x \in \mathbb{R}^d$ and a router matrix $R \in \mathbb{R}^{N \times d}$, the routing weights are computed as

$$\mathbf{w}(x) = \mathrm{TopK}(\mathrm{softmax}(Rx); k)\,, \tag{1}$$

where $\mathrm{TopK}(\cdot; k)$ retains the k largest entries and sets the rest to zero. We denote the corresponding set of selected experts by $S(x)$. Different MoE-based LLMs may use different gating and normalization conventions (Muennighoff et al., 2025; Jiang et al., 2024; Yang et al., 2025).

The MoE output is then formed by aggregating the selected expert outputs:

$$y(x) = \sum_{e \in S(x)} w_e(x)\, E_e(x), \tag{2}$$

where E_e denotes the e-th expert and $w_e(x)$ determines its contribution to the aggregated output.

Standard MoE aggregation uses the same routing scores to both select experts and weight their outputs. Each selected expert is assigned a single scalar weight across all output dimensions, offering no finer control over its internal representation. Consequently, the aggregation mechanism operates only at the expert level.

4 REVISITING EXPERT AGGREGATION

We first probe the adaptation headroom of pretrained expert aggregation under fixed expert selection, and then test whether it can be exploited by a controller shared across examples.

4.1 AGGREGATION HEADROOM UNDER RESTRICTED REWEIGHTING

We design a ground-truth oracle on OLMoE (Muennighoff et al., 2025) to probe aggregation headroom under a restricted intervention. All pretrained parameters are frozen, and only the aggregation weights of context tokens in the penultimate MoE layer are modified, while preserving both their pretrained Top-k expert sets and total aggregation mass. Targeting the penultimate layer keeps all preceding computations identical to the base model, so that only the final Transformer layer receives the modified context representations. For each evaluation example, a single set of reweighting parameters is shared across all answer candidates and optimized using the ground-truth label.

For each context token t, let S_t denote its pretrained Top-k expert set and $p_{t,e}$ the corresponding aggregation weight for expert e. In OLMoE, the selected softmax weights are not renormalized after Top-k. Their total aggregation mass is therefore

$$C_t = \sum_{e \in S_t} p_{t,e}. \tag{3}$$

We preserve C_t and modify only the relative allocation among the selected experts. Collecting their pretrained weights into $\mathbf{p}_t \in \mathbb{R}^k$, we introduce a residual vector $\boldsymbol{\delta}_t \in \mathbb{R}^k$ and define the reweighted allocation as

$$\mathbf{q}_t = C_t\ \mathrm{softmax}(\log \mathbf{p}_t + \boldsymbol{\delta}_t)\,. \tag{4}$$

This construction preserves both the selected expert set and its total aggregation mass, while $\boldsymbol{\delta}_t = \mathbf{0}$ exactly recovers the pretrained allocation.

To measure only the change in relative allocation, we normalize the selected weights as $\bar{\mathbf{p}}_t = \mathbf{p}_t / C_t$ and $\bar{\mathbf{q}}_t = \mathbf{q}_t / C_t$, and compute their conditional KL divergence:

$$D_{\mathrm{KL}}(\bar{\mathbf{q}}_t \,\|\, \bar{\mathbf{p}}_t) = \sum_{e \in S_t} \bar{q}_{t,e} \log \frac{\bar{q}_{t,e}}{\bar{p}_{t,e}}. \tag{5}$$

As shown in Figure 1, the seven-task macro-average increases from 63.8% to 67.1% at snapshots near a mean conditional KL of 0.01, and further to 70.2% near 0.05. Notably, the 3.3-point gain at KL ≈ 0.01 already indicates substantial adaptation headroom in pretrained expert aggregation under this restricted intervention. Protocol details, per-task threshold crossings, and the unconstrained step trajectory are reported in Appendix A.1.

4.2 Learnable Aggregation Reweighting

The oracle above relies on ground-truth labels and cannot be directly deployed. We therefore introduce a Residual Aggregation Controller (RAC), which learns a lightweight controller for each MoE layer while keeping the pretrained model frozen. RAC preserves the Top-k expert set and total selected mass, modifying only the relative allocation among selected experts.

At each MoE layer l, RAC predicts an expert-wise residual score from the router input $x_{l,t}$:

$$a_{l,t} = W_l x_{l,t} + b_l. \tag{6}$$

For each selected expert $e \in S_{l,t}$, the score is converted into a bounded log-weight residual:

$$\delta_{l,t,e} = \varepsilon \tanh(a_{l,t,e}), \tag{7}$$

and the reweighted allocation is $\mathbf{q}_{l,t} = C_{l,t}\,\mathrm{softmax}(\log \mathbf{p}_{l,t} + \boldsymbol{\delta}_{l,t})$, the same residual reallocation as in Equation 4. The controller is zero-initialized, so $\boldsymbol{\delta}_{l,t} = \mathbf{0}$ and thus $\mathbf{q}_{l,t} = \mathbf{p}_{l,t}$. The bound ε controls how far the learned allocation may depart from the pretrained one in log-weight space.

As shown in Figure 1(b), increasing reweighting flexibility leads to a non-monotonic effect. With $\varepsilon = 0.3$, the CS average improves from 64.44% to 68.06% while OOD performance remains close to the base model. Increasing ε to 0.8 further improves CS to 69.29%, but reduces the Math average from 27.84% to 18.37%. Larger residual bounds cause broader OOD degradation, while the ID improvement also diminishes. The full ε sweep and training details are reported in Appendix A.2.

Taken together, the two diagnostics reveal adaptation headroom in pretrained expert aggregation, while showing that aggressive scalar reallocation can compromise generalization. One possible reason is that larger reallocations alter the representations passed to subsequent layers and may consequently perturb downstream routing decisions. These observations motivate preserving the pretrained expert-level allocation while exploring finer-grained forms of expert aggregation.

5 Routing-Conditioned Granular Expert Modulation (RC-GEM)

To address this limitation, we introduce a post-training method for fine-grained expert aggregation, termed **R**outing-**C**onditioned **G**ranular **E**xpert **M**odulation (RC-GEM), which complements expert-level weighting with routing-conditioned modulation over learned representation subspaces. Instead of altering the pretrained expert selection or aggregation weights, RC-GEM assigns distinct, bounded adjustments to different learned subspaces of each selected expert representation. These adjustments are conditioned on the current token's router logits, allowing different subspaces of the same expert to be selectively amplified or attenuated. The resulting expert outputs are still combined using the original pretrained aggregation weights. Only the subspace bases and lightweight modulation heads are trained, while the pretrained backbone, router, and experts remain frozen.

5.1 Routing-Conditioned Subspace Modulation

Learned representation subspaces. To enable fine-grained modulation within expert representations, RC-GEM learns a layer-specific low-dimensional representation basis

$$A_l \in \mathbb{R}^{r \times d}, \qquad r \ll d, \tag{8}$$

where d is the expert output dimension and r is the dimension of the learned representation space. The basis A_l is shared across all experts within the same layer, providing a common decomposition of their output representations.

We partition the rows of A_l into G equal-sized blocks,

$$A_l = \begin{bmatrix} A_{l,1} \\ A_{l,2} \\ \vdots \\ A_{l,G} \end{bmatrix}, \qquad A_{l,g} \in \mathbb{R}^{r_g \times d}, \qquad r_g = \frac{r}{G}, \tag{9}$$

where each block spans a learned representation subspace

$$\mathcal{U}_{l,g} = \mathrm{span}(\mathrm{rows}(A_{l,g})) \subseteq \mathbb{R}^d. \tag{10}$$

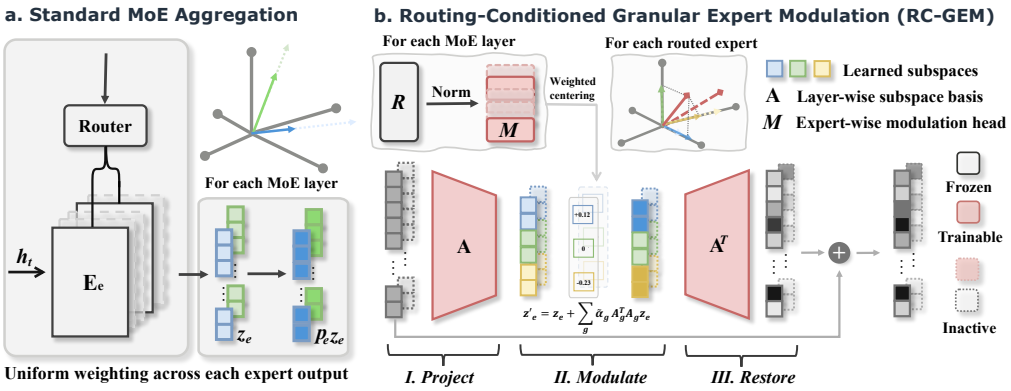


Figure 2: Overview of RC-GEM. (a) Standard MoE aggregates selected expert outputs z_e with router weights p_e. (b) RC-GEM preserves pretrained aggregation and performs routing-conditioned subspace modulation through learned subspaces A_g and centered coefficients $\tilde{\alpha}_g$.

For an expert output $z_{l,t,e} \in \mathbb{R}^d$, we denote its low-dimensional representation associated with subspace g as

$$u_{l,t,e,g} = A_{l,g} z_{l,t,e} \in \mathbb{R}^{r_g}. \tag{11}$$

Together, the G subspaces define the units over which RC-GEM performs fine-grained modulation. Each subspace is modulated independently, allowing different components of the same expert representation to be adjusted separately. Here, G determines the modulation granularity, whereas r controls the capacity of the learned representation space.

Routing-conditioned modulation. Motivated by prior findings that MoE routing signals encode high-level semantic information (Li & Zhou, 2025), we use the full router-logit vector as a compact conditioning signal to guide modulation over the learned subspaces, with each logit reflecting the affinity between the current token and the corresponding expert. For token t at layer l, let $r_{l,t} \in \mathbb{R}^N$ denote the router logits. We obtain the modulation signal as

$$s_{l,t} = \mathcal{N}(r_{l,t}), \tag{12}$$

where $\mathcal{N}$ denotes the normalization applied to the routing profile. For each selected expert $e \in S_{l,t}$, a lightweight expert-specific head produces G modulation coefficients, one for each learned subspace,

$$\delta_{l,t,e} = f_{l,e}(s_{l,t}) \in \mathbb{R}^G. \tag{13}$$

We parameterize the modulation as a bounded residual over the pretrained contribution:

$$\alpha_{l,t,e,g} = \varepsilon \tanh(\delta_{l,t,e,g}), \tag{14}$$

where ε bounds the residual magnitude before centering.

A common component in the residuals of the selected experts would uniformly amplify or attenuate the same subspace across all experts, changing its overall aggregation scale rather than how its contribution is distributed among them. We therefore remove this shared component using the pretrained aggregation weights:

$$\bar{\alpha}_{l,t,g} = \frac{\sum_{j \in S_{l,t}} p_{l,t,j} \alpha_{l,t,j,g}}{\sum_{j \in S_{l,t}} p_{l,t,j}}, \qquad \tilde{\alpha}_{l,t,e,g} = \alpha_{l,t,e,g} - \bar{\alpha}_{l,t,g}. \tag{15}$$

By construction,

$$\sum_{e \in S_{l,t}} p_{l,t,e} \tilde{\alpha}_{l,t,e,g} = 0, \tag{16}$$

for every subspace g. We then define the multiplicative modulation weight as

$$m_{l,t,e,g} = 1 + \tilde{\alpha}_{l,t,e,g}. \tag{17}$$

It follows that

$$\sum_{e \in S_{l,t}} p_{l,t,e} m_{l,t,e,g} = \sum_{e \in S_{l,t}} p_{l,t,e}. \tag{18}$$

Thus, centering preserves the total aggregation weight assigned to each subspace, while allowing its allocation across the selected experts to vary.

Project–Modulate–Restore. For each selected expert, RC-GEM scales the representation associated with subspace g by the multiplicative weight $m_{l,t,e,g} = 1 + \tilde{\alpha}_{l,t,e,g}$ (Equation 17). The modulated representation is then mapped back through $A_{l,g}^{\top}$. Applying this operation across all learned subspaces and aggregating the resulting expert outputs with the pretrained weights $p_{l,t,e}$ gives

$$y_{l,t} = \sum_{e \in S_{l,t}} p_{l,t,e} \left(I_d + \sum_{g=1}^{G} \tilde{\alpha}_{l,t,e,g} A_{l,g}^{\top} A_{l,g} \right) z_{l,t,e}. \tag{19}$$

Here, $p_{l,t,e}$ retains the pretrained expert-level weighting, while $m_{l,t,e,g}$ provides a finer-grained multiplicative weight for each learned subspace. By Equation 16, the residual weights $\tilde{\alpha}_{l,t,e,g}$ have zero weighted mean across the selected experts for every subspace, allowing the subspace-level weights to vary across experts while preserving the total aggregation weight assigned to that subspace (Equation 18). At initialization, the modulation heads yield near-zero residuals, such that $\tilde{\alpha}_{l,t,e,g} \approx 0$ and hence $m_{l,t,e,g} \approx 1$, keeping the model close to the pretrained MoE at the start of training.

5.2 Orthogonal Subspace Regularization

Each basis A_l is initialized with orthonormal rows. As A_l is optimized jointly with the modulation heads, we regularize its row Gram matrix toward the identity to keep the learned basis close to row-orthonormal throughout training. For the L MoE layers equipped with learned bases, we define

$$\mathcal{L}_{\text{orth}} = \frac{1}{Lr} \sum_{l=1}^{L} \left\| A_l A_l^{\top} - I_r \right\|_F^2. \tag{20}$$

The overall training objective is

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_{\text{orth}} \mathcal{L}_{\text{orth}}. \tag{21}$$

This regularizer plays two complementary roles.

Maintaining subspace structure. The diagonal entries of $A_l A_l^{\top}$ measure the squared norms of the basis directions, while its off-diagonal entries measure their pairwise correlations. Driving this matrix toward I_r therefore keeps the learned directions close to unit norm while discouraging overlap between them. As a consequence, $A_{l,g}^{\top} A_{l,g}$ in Equation 19 remains an approximate orthogonal projector onto $\mathcal{U}_{l,g}$, and different learned subspaces remain approximately separated during training.

Controlled modulation scale. The bounded residuals remain controlled after weighted centering. Since $|\alpha_{l,t,e,g}| \le \varepsilon$ and the weighted mean in Equation 15 also lies in $[-\varepsilon, \varepsilon]$, we have $|\tilde{\alpha}_{l,t,e,g}| \le 2\varepsilon$. The corresponding update to expert e is

$$\Delta z_{l,t,e} = \sum_{g=1}^{G} \tilde{\alpha}_{l,t,e,g} A_{l,g}^{\top} A_{l,g} z_{l,t,e}. \tag{22}$$

Bounding the coefficients alone is insufficient if the learned basis can grow arbitrarily in scale. By keeping $A_l A_l^{\top} \approx I_r$, the regularizer also keeps $\|A_l\|_2 \approx 1$, yielding

$$\|\Delta z_{l,t,e}\|_2 \lesssim 2\varepsilon \, \|z_{l,t,e}\|_2. \tag{23}$$

Thus, weighted centering preserves a bounded residual after centering, while orthogonal regularization prevents the learned basis from introducing additional amplification. Together, they keep the representation perturbation controlled by ε.

Table 1: Evaluation results of RC-GEM and baselines on two MoE backbones across eight ID commonsense reasoning benchmarks and five OOD benchmarks spanning general knowledge, mathematical reasoning, and code generation. Avg denotes the macro-average within each task group, while Avg* denotes the sample-size-weighted average across MMLU subjects. **Bold** and underlined numbers indicate the best and second-best results, respectively. Blue/red shading denotes ID/OOD summary columns. See Section 6.1 for details of the baselines and benchmarks.

	Commonsense									Math			Code			MMLU
	BoolQ	PIQA	SIQA	HellaS	WinoG	ARC-e	ARC-c	OBQA	Avg	GSM8K	MATH	Avg	HEval	MBPP	Avg	Avg*
OLMoE-1B-7B-0125																
Base	70.18	79.60	47.29	78.12	68.82	76.64	49.40	44.40	64.31	44.88	10.80	27.84	11.59	30.95	21.27	53.47
FFT	**85.93**	**84.93**	59.11	**88.59**	**79.01**	86.20	70.14	**65.20**	**77.39**	14.71	5.00	9.85	11.59	23.81	17.70	50.53
LoRA	84.10	84.22	58.85	86.86	77.35	86.20	70.56	64.20	76.54	14.40	3.20	8.80	9.15	9.79	9.47	51.79
PREFIX	84.80	83.68	58.80	88.36	77.58	87.42	**72.44**	61.60	76.84	12.74	3.60	8.17	6.10	2.12	4.11	51.37
ACE	83.33	80.79	57.52	80.84	71.43	84.01	63.99	51.40	71.66	44.35	**11.20**	27.78	12.20	30.42	21.31	**53.73**
RoMA	73.00	79.71	47.65	78.62	69.53	79.38	51.96	46.80	65.83	**47.84**	8.20	**28.02**	11.59	**32.80**	22.19	53.60
PERFT-E	81.90	84.22	58.03	83.43	77.98	85.94	70.73	64.80	75.88	12.96	5.40	9.18	13.41	19.31	16.36	41.28
Router-FT	83.30	80.09	53.02	79.82	70.01	81.90	59.22	48.60	69.49	31.16	8.00	19.58	13.41	26.98	20.20	50.33
RC-GEM	84.89	84.66	**60.08**	88.33	77.66	**89.06**	72.35	59.60	77.08	43.59	10.80	27.20	**14.02**	30.69	**22.36**	53.44
Qwen1.5-MoE-A2.7B																
Base	79.69	80.25	49.54	77.37	68.98	69.28	44.37	44.20	64.21	52.69	23.00	37.85	38.41	48.94	43.68	60.96
FFT	84.71	84.93	54.66	89.62	81.69	84.26	72.27	58.80	76.37	4.55	5.40	4.97	25.61	38.89	32.25	56.12
LoRA	**86.51**	**86.18**	**61.98**	**91.42**	81.45	86.99	**73.55**	**61.80**	**78.74**	6.97	5.60	6.29	20.73	40.21	30.47	55.90
PREFIX	86.42	85.36	58.50	91.35	79.16	86.87	71.50	60.60	77.47	1.21	1.40	1.31	12.20	5.03	8.61	56.36
ACE	84.34	81.39	59.62	79.05	69.30	83.59	60.67	47.80	70.72	**53.37**	**23.80**	**38.59**	**41.46**	51.32	**46.39**	**61.55**
RoMA	80.58	80.63	48.46	77.38	69.93	69.87	45.22	43.80	64.48	52.77	22.60	37.68	34.15	47.35	40.75	60.85
PERFT-E	85.72	82.70	60.03	85.95	77.27	85.35	70.14	61.40	76.07	25.17	10.60	17.89	31.71	52.65	42.18	58.07
Router-FT	81.90	80.25	49.59	77.53	67.17	77.86	53.24	39.60	65.89	32.22	7.20	19.71	34.15	46.83	40.49	57.92
RC-GEM	86.18	85.64	58.55	89.52	**82.24**	**88.01**	72.35	59.80	77.79	51.63	20.40	36.02	32.93	**54.50**	43.72	58.96

6 EXPERIMENTS

6.1 EXPERIMENTAL SETTINGS

LLM backbones. We evaluate RC-GEM on two open-source MoE LLMs: OLMoE-1B-7B-0125 (Muennighoff et al., 2025) and Qwen1.5-MoE-A2.7B (Qwen Team, 2024), which represent different expert configurations and routing sparsities. Further details are provided in Appendix A.3.

Baselines. We compare RC-GEM with full fine-tuning (FFT), general parameter-efficient fine-tuning methods, and MoE-specific adaptation methods. FFT updates all parameters except token embeddings and the language modeling head. For general PEFT, we include LoRA (Hu et al., 2022) and Prefix Tuning (Li & Liang, 2021). We apply LoRA to the query and value projections in each self-attention layer under a parameter-matched setting. MoE-aware adaptation methods include Router Tuning (Router-FT), RoMA (Li et al., 2026), ACE (Lee et al., 2026), and PERFT-E (Liu et al., 2026). Router Tuning directly updates pretrained routers; RoMA further regularizes router adaptation through routing-manifold alignment; ACE consolidates expert-specific adapters into shared adapters; and PERFT-E introduces routed PEFT experts following pretrained routing patterns. Implementation details are provided in Appendix A.4.

Training data. All methods are trained on the Commonsense15K dataset (Hu et al., 2023) under the same training setting. Dataset details are provided in Appendix A.5.

Benchmarks. We evaluate ID performance on eight commonsense reasoning benchmarks: BoolQ, PIQA, Social IQA, HellaSwag, WinoGrande, ARC-Easy, ARC-Challenge, and OpenBookQA. We further evaluate OOD generalization on MMLU, GSM8K, MATH-500, HumanEval, and MBPP. Detailed benchmark settings are provided in Appendix A.6.

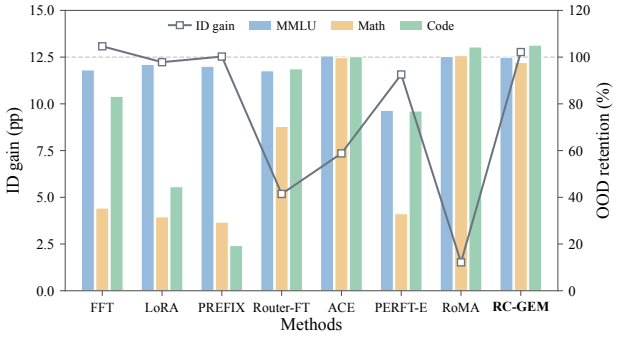


Figure 3: The curve shows ID gains over the pretrained model (left), and the bars show OOD retention on MMLU, Math, and Code (right). The dashed line denotes the pretrained model's OOD performance.

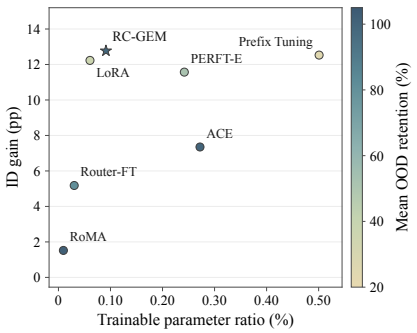


Figure 4: Points represent adaptation methods, with position indicating trainable parameter ratio and ID gain, and color indicating mean OOD retention.

Evaluation protocol. We use established evaluation frameworks, including LM Evaluation Harness (Gao et al., 2024) for commonsense benchmarks and MMLU, Math-Verify (Kydlíček, 2025) for GSM8K and MATH-500, and EvalPlus (Liu et al., 2023) for HumanEval and MBPP. All benchmarks follow zero-shot evaluation, with pass@1 reported for code generation. Further details are provided in Appendix A.7.

Training setup. RC-GEM freezes the pretrained model, router, and experts, and only trains the subspace bases and modulation heads. Baseline configurations follow their original papers or official implementations. Detailed optimization settings are provided in Appendix A.8.

6.2 Main Results

Advantage of RC-GEM over different adaptation methods. Table 1 compares RC-GEM with full fine-tuning, generic PEFT, and MoE-aware adaptation methods on OLMoE and Qwen1.5-MoE. FFT, LoRA, and Prefix Tuning achieve strong ID adaptation, but with larger declines on Math, Code, and MMLU. **By contrast, our method RC-GEM reaches comparable ID performance while better retaining these pretrained capabilities.** On OLMoE, RC-GEM improves the ID average from 64.31 to 77.08, only 0.31 points below FFT, while keeping MMLU, Math, and Code close to the base model (53.44 vs. 53.47, 27.20 vs. 27.84, and 22.36 vs. 21.27, respectively). On Qwen1.5-MoE, it reaches 77.79, within 0.95 points of the best generic PEFT baseline, LoRA, while largely preserving Math and Code performance; in comparison, FFT, LoRA, and Prefix Tuning retain only 3.5–16.6% of the base Math performance and 19.7–73.8% of the base Code performance. Among MoE-aware baselines, RoMA keeps a slightly higher average OOD retention and ACE stays near the base scores, but both recover at most 58% of RC-GEM's ID gain, whereas PERFT-E reaches 87–91% of its ID gain with larger drops on Math and MMLU. Overall, RC-GEM achieves the highest ID average among the evaluated MoE-aware methods on both backbones while keeping OOD performance close to the pretrained models. Figure 3 summarizes this adaptation–preservation trade-off.

Parameter efficiency. Figure 4 compares the trainable parameter ratios on OLMoE. RC-GEM updates 0.0913% of the backbone parameters. RoMA (0.0095%) and Router-FT (0.0303%) use fewer parameters, but achieve only 12% and 41% of RC-GEM's ID gain, respectively. In contrast, ACE and PERFT-E use 3.0× and 2.6× more trainable parameters than RC-GEM, yet achieve only 58% and 91% of its ID gain, with PERFT-E also showing larger Math and MMLU degradation. Taken together, RC-GEM combines low parameter overhead with the largest ID gain among these MoE-aware baselines while keeping OOD performance close to the base model.

6.3 Ablation Study

Effect of Subspace Learning. We investigate whether the performance gain of RC-GEM comes from learning an effective subspace decomposition rather than merely increasing the number of modulation units. We compare RC-GEM with Contiguous grouping, which partitions the original

Table 2: Ablation study on subspace learning.

Method	Decomposition	Trainable	CS
Base	—	—	64.31
Contiguous	Coordinate grouping	×	68.32
Random	Random basis	×	66.47
RC-GEM	**Learned basis**	✓	**77.08**

Table 3: Ablation of weighted centering.

Variant	CS	Math	Code	MMLU
RC-GEM	**77.08**	**27.20**	**22.36**	**53.44**
w/o WC	76.69	15.99	6.98	52.66

Table 4: Ablation of modulation conditioning.

Variant	%train.	CS	Math	Code	MMLU
RC-GEM	**0.0913**	**77.08**	**27.20**	**22.36**	**53.44**
w/ RR	0.0913	65.12	27.89	22.30	53.52
w/ HS	1.0205	76.87	27.53	22.53	53.68

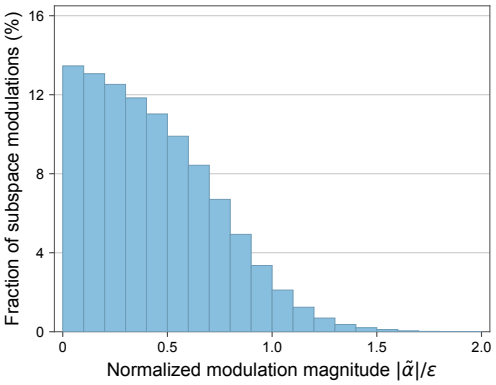


Figure 5: Distribution of normalized modulation magnitudes under $\epsilon = 0.3$. The histogram aggregates weighted-centered subspace modulation coefficients across all MoE layers with respect to $|\tilde{\alpha}|/\epsilon$.

coordinates into groups, and Random, which uses a fixed random orthogonal basis. As shown in Table 2, both fixed decompositions provide only limited gains over the base model ($+4.01$ and $+2.16$ points), whereas learning the basis enables a substantially larger improvement ($+12.77$ points). This demonstrates that effective expert modulation relies on learning suitable directions in the expert output space rather than applying predefined decompositions.

Effect of Weighted Centering. We remove weighted centering and directly apply the predicted modulation coefficients. As shown in Table 3, the degradation of this variant suggests that weighted centering helps suppress shared scaling effects and enables subspace-wise redistribution among selected experts.

Effect of Modulation Conditioning. Table 4 shows that random routing (w/ RR) degrades performance, indicating that routing signals provide token-dependent information for adaptive modulation. Hidden-state conditioning (w/ HS) achieves slightly better OOD performance with over $10\times$ more parameters, likely because hidden states provide richer token representations for modulation. This demonstrates that router logits offer a compact and effective conditioning signal for RC-GEM.

Effect of Modulation Bound. We investigate how the modulation bound ε balances adaptation flexibility and preservation of pretrained aggregation. As shown in Table 5, a small bound ($\varepsilon = 0.1$) limits modulation capacity, while $\varepsilon = 0.3$ achieves strong CS gains with better OOD preservation. Larger bounds provide limited additional ID improvements but degrade OOD performance, especially at $\varepsilon = 0.8$. Figure 5 further shows that under $\varepsilon = 0.3$, RC-GEM learns mostly small modulations, with only 4.79% exceeding ε, suggesting selective subspace-level refinement rather than large-scale rescaling.

Table 5: Sensitivity to modulation bound.

ε	CS	Math	Code	MMLU
0.1	73.40	28.02	17.30	52.87
0.3	**77.08**	**27.20**	**22.36**	**53.44**
0.5	77.48	24.49	21.03	53.08
0.8	77.18	20.63	20.50	52.48

7 Conclusion

This work introduces RC-GEM, a parameter-efficient adaptation framework for sparse MoE LLMs that refines expert aggregation beyond expert-level weighting. Our diagnostic studies reveal that pretrained aggregation leaves room for adaptation: while scalar reallocation can improve performance under moderate changes, excessive deviations introduce an adaptation–generalization tradeoff. RC-GEM addresses this limitation by preserving pretrained expert selection and aggregation weights while enabling routing-conditioned subspace modulation of each selected expert's contribution. With fewer than 0.1% trainable parameters, RC-GEM approaches full fine-tuning performance while largely preserving pretrained OOD capabilities, highlighting aggregation granularity as an important direction for parameter-efficient MoE adaptation.

AI USE STATEMENT

The authors used ChatGPT (OpenAI, 2026) for grammatical refinement and improving the clarity of the manuscript. All generated suggestions were manually reviewed and revised by the authors.

ETHICS STATEMENT

This work studies efficient adaptation of pretrained Mixture-of-Experts (MoE) language models. Our experiments use publicly available open-source models and benchmark datasets, and do not involve the collection or release of private data. We conduct all experiments in accordance with the ethical guidelines of ICLR.

REPRODUCIBILITY STATEMENT

We release the implementation of RC-GEM, including training and evaluation code, to facilitate reproduction of our method and experimental results. Detailed descriptions of model configurations, training settings, and evaluation protocols are provided in the paper and appendix. The source code is available at: `https://anonymous.4open.science/r/RC-GEM-F5F4`

REFERENCES

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. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

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

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. URL https://jmlr.org/papers/v23/21-0998.html.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness. Zenodo, 2024. URL https://zenodo.org/records/12608602. Version v0.4.3.

GLM-5 Team. GLM-5: from vibe coding to agentic engineering. arXiv preprint arXiv:2602.15763, 2026. URL https://arxiv.org/abs/2602.15763.

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*, 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems*, 2021b. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

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://arxiv.org/abs/2106.09685.

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, 2023. URL https://aclanthology.org/2023.emnlp-main.319/.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of Experts. arXiv preprint arXiv:2401.04088, 2024. URL https://arxiv.org/abs/2401.04088.

Hynek Kydlíček. Math-Verify: Math verification library. GitHub repository, 2025. URL https://github.com/huggingface/Math-Verify.

Ahin Lee, Sehyun Yun, Joonha Park, and Taesik Gong. ACE: Adapter consolidation across experts for parameter-efficient fine-tuning of MoE LLMs. arXiv preprint arXiv:2609.06072, 2026. URL https://arxiv.org/abs/2609.06072. Accepted to EMNLP 2026.

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. URL https://arxiv.org/abs/2006.16668.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021. URL https://aclanthology.org/2021.acl-long.353/.

Zhongyang Li, Ziyue Li, and Tianyi Zhou. C3PO: Critical-layer, core-expert, collaborative pathway optimization for test-time expert re-mixing. In *Conference on Language Modeling (COLM)*, 2025. URL https://openreview.net/forum?id=lqC5J7pBP9.

Zhongyang Li, Ziyue Li, and Tianyi Zhou. Routing manifold alignment improves generalization of mixture-of-experts LLMs. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3lskwxB653.

Ziyue Li and Tianyi Zhou. Your mixture-of-experts LLM is secretly an embedding model for free. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=eFGQ97z5Cd.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

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*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

Yilun Liu, Yunpu Ma, Yuetian Lu, Shuo Chen, Zifeng Ding, and Volker Tresp. Parameter-efficient routed fine-tuning: Mixture-of-experts demands mixture of adaptation modules. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 4439–4457, 2026. URL https://aclanthology.org/2026.findings-eacl.232/.

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, 2018. 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 A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. OLMoE: Open mixture-of-experts language models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2409.02060.

Amir Nuriyev and Gabriel Kulp. Expert selections in MoE models reveal (almost) as much as text. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=BABF0WBRfe.

OpenAI. gpt-oss-120b & gpt-oss-20b model card. arXiv preprint arXiv:2508.10925, 2025. URL https://arxiv.org/abs/2508.10925.

OpenAI. ChatGPT, 2026. URL https://chatgpt.com.

Qwen Team. Qwen1.5-MoE: Matching 7B model performance with 1/3 activated parameters. Blog post, 2024. URL https://qwenlm.github.io/blog/qwen-moe/.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. URL https://doi.org/10.1609/aaai.v34i05.6399.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019. 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. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1701.06538.

Amirhossein Vahidi, Hesam Asadollahzadeh, Navid Akhavan Attar, Marie Moullet, Kevin Ly, Xingyi Yang, and Mohammad Lotfollahi. DirMoE: Dirichlet-routed mixture of experts. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=a15cDnzr6r.

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.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019. URL https://aclanthology.org/P19-1472/.

A Appendix

A.1 Aggregation oracle

The oracle intervenes at decoder layer index 14 (zero-based) of the 16-layer OLMoE model. For each example, it assigns a separate residual to each context token and selected expert slot, and all answer candidates share these residuals. Given candidate scores $s_c(\delta)$ and correct candidate c^*, the objective is

$$\mathcal{L}_{\mathrm{MC}} = -\log \frac{\exp(s_{c^*}(\delta)/\tau)}{\sum_c \exp(s_c(\delta)/\tau)}. \tag{24}$$

We use $\tau = 1$. Candidate scores follow the task's evaluation rule, using raw or length-normalized continuation log-likelihood as appropriate. The recorded runs use Adam with learning rate 0.01, `steps=100`, and `opt_batch_size=32`, starting from zero residuals. Hard KL constraints are disabled, so every Adam proposal is accepted; the logged mean conditional KL is recorded only as a diagnostic of how far the allocation has moved from the pretrained weights.

For each optimization snapshot, the logged mean conditional KL averages over valid context tokens within a batch and then over batches. To connect this trajectory to the small-KL operating points in Section 4.1, we select, separately for each task, the first step at which the logged mean exceeds a given threshold. Thus, neither the realized KL values nor the optimization steps are identical across tasks. Table 6 reports the corresponding in-loop accuracies at selected thresholds, including the 0.01 and 0.05 operating points highlighted in the main text. WinoGrande is excluded because its evaluation protocol inserts each candidate into the input, rather than providing the shared context required by this intervention.

Figure 6 and Table 7 report the complementary view along fixed optimization steps. After every 10 Adam steps we freeze the current residuals and re-evaluate with LM Evaluation Harness. Relative to step 0, all seven tasks improve by step 100, but the rates differ: BoolQ rises sharply early, whereas SocialIQA, ARC-Challenge, and OpenBookQA continue to gain through later steps. The seven-task macro-average increases from 63.7% at step 0 to 80.3% at step 100. At step 100, the logged mean conditional KL ranges from approximately 0.64 to 1.41 across tasks, so these late-step scores are not the small-KL operating points reported in the main text. Small differences between Table 6 and Table 7 at matched settings arise because the former uses optimization-loop accuracy at KL crossings, whereas the latter uses LM Evaluation Harness checkpoints.

Table 6: Per-task oracle scores at selected mean-KL threshold crossings, taken from the optimization-loop accuracy at the first step where the logged mean conditional KL exceeds each threshold. Scores follow each benchmark's accuracy metric; the seven-task average excludes WinoGrande.

Threshold	ARC-C	ARC-E	HellaSwag	SIQA	BoolQ	PIQA	OBQA	Avg.
Base	49.9	76.7	78.1	47.3	70.2	79.7	44.4	63.8
0.01	52.4	78.7	78.9	50.5	82.1	80.2	46.6	67.1
0.05	56.0	81.4	80.0	53.1	91.9	80.7	48.2	70.2
0.20	61.3	84.6	81.6	57.4	98.8	81.9	51.8	73.9
0.30	63.7	85.9	82.2	60.3	99.2	82.4	53.4	75.3
0.50	66.3	87.7	83.1	63.5	99.9	82.9	56.4	77.1

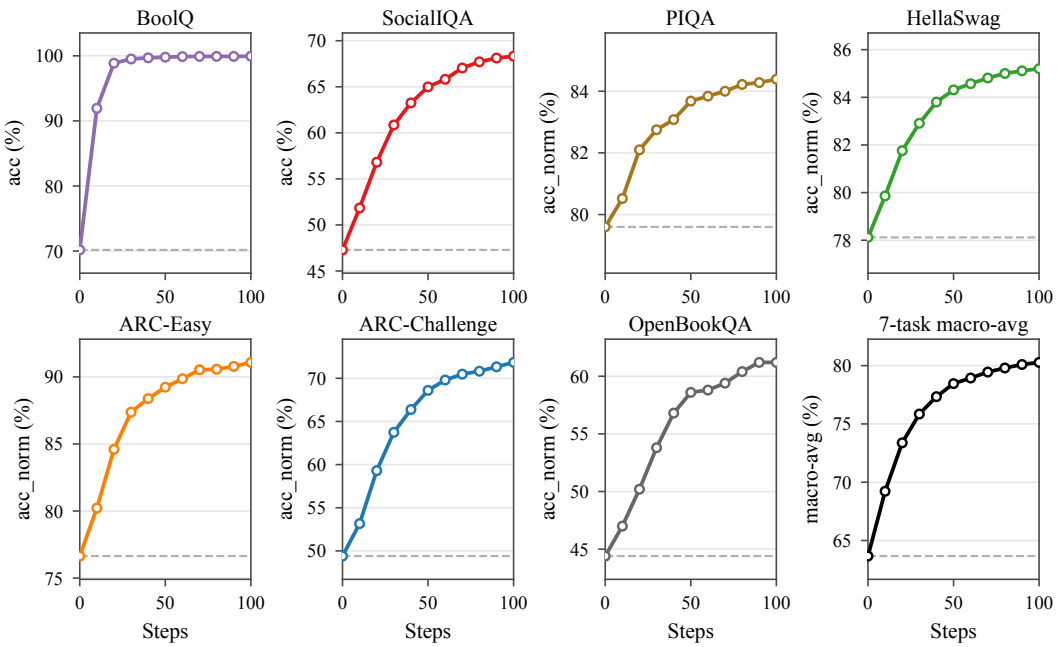


Figure 6: Per-task oracle scores versus unconstrained optimization steps on OLMoE, together with the seven-task macro-average. Step 0 recovers the pretrained baseline. Dashed horizontal lines mark each panel's baseline score. WinoGrande is omitted for the reason noted above.

A.2 Residual aggregation controller

RAC uses a separate linear controller at every MoE layer of OLMoE, with router-input hidden states as inputs and zero-initialized weights and biases. Only controller parameters are trained. Training uses candidate ranking on Commonsense-15K with AdamW at learning rate 3×10^{-4}, cosine decay with warmup ratio 0.03, effective batch size 32, and sequence cutoff 384. All settings in Table 8 use the checkpoint after one epoch. The controller predicts residuals for the experts selected by the frozen router for the current layer input; earlier interventions can still affect later routing decisions.

Since $|\delta_{l,t,e}| < \varepsilon$, the reallocation formula gives, for any selected experts i, j,

$$\frac{q_{l,t,i}/q_{l,t,j}}{w_{l,t,i}/w_{l,t,j}} = \exp(\delta_{l,t,i} - \delta_{l,t,j}) \in (e^{-2\varepsilon}, e^{2\varepsilon}). \tag{25}$$

Thus, ε bounds the change in relative odds, rather than imposing a KL budget. RAC uses no KL loss. Its logged allocation KL is a diagnostic from the last layer, with direction $D_{\mathrm{KL}}(q/C \| w/C)$; its aggregation scope differs from the oracle's context-only statistic. The oracle and RAC also differ in task coverage, intervention scope, and label access, so their gains are not directly comparable.

Table 7: Per-task LM Evaluation Harness oracle scores every 10 unconstrained Adam steps on OLMoE. Step 0 is the pretrained baseline. Scores are percentages under each task's accuracy metric; Avg. is the unweighted seven-task mean. WinoGrande is excluded as above.

Step	ARC-C	ARC-E	HellaSwag	SIQA	BoolQ	PIQA	OBQA	Avg.
0	49.4	76.6	78.1	47.3	70.2	79.6	44.4	63.7
10	53.2	80.2	79.9	51.8	91.9	80.5	47.0	69.2
20	59.3	84.6	81.8	56.8	98.9	82.1	50.2	73.4
30	63.7	87.4	82.9	60.9	99.5	82.8	53.8	75.8
40	66.4	88.4	83.8	63.2	99.7	83.1	56.8	77.3
50	68.6	89.2	84.3	65.0	99.8	83.7	58.6	78.5
60	69.8	89.9	84.6	65.8	99.9	83.8	58.8	78.9
70	70.5	90.5	84.8	67.0	99.9	84.0	59.4	79.5
80	70.8	90.6	85.0	67.7	99.9	84.2	60.4	79.8
90	71.3	90.8	85.1	68.1	99.9	84.3	61.2	80.1
100	71.8	91.1	85.2	68.3	99.9	84.4	61.2	80.3

Table 8: Complete RAC residual-bound sweep on OLMoE. CS is the ID score; MMLU, Math, and Code are OOD scores.

ε	CS (ID)	MMLU	Math	Code
Base	64.44	53.62	27.84	21.27
0.3	68.06	53.45	28.54	21.36
0.5	68.74	53.17	26.31	20.96
0.8	69.29	52.40	18.37	20.90
1.0	68.67	50.84	18.54	16.46
1.5	67.72	47.90	14.28	12.53
2.0	65.39	42.12	7.52	10.08

A.3 Backbone details

OLMoE-1B-7B-0125 has approximately 6.92B parameters, 16 Transformer layers, all equipped with MoE blocks. Each block contains 64 routed experts, activates 8 experts per token, and uses a hidden size of 2048. It has no shared expert. Qwen1.5-MoE-A2.7B has approximately 14.32B parameters and 24 Transformer layers, likewise all MoE. Each block contains 60 routed experts, activates 4 experts per token, uses the same hidden size of 2048, and adds one fused shared expert per layer. The two backbones therefore differ in depth, routing sparsity, and whether a shared expert is present.

A.4 Baseline implementations

FFT updates every parameter except the token embeddings and the language-modeling head. LoRA is inserted on W_q and W_v in each self-attention layer; the runs reported in Table 1 use rank 32 and scaling $\alpha = 32$. Prefix Tuning uses a prefix of length 10 at each layer and trains only the prompt encoder. Router-FT updates all pretrained router parameters. RoMA updates the routers in the last five MoE layers. ACE replaces groups of expert-specific adapters with shared higher-rank adapters, and PERFT-E adds PEFT experts that reuse the pretrained routing pattern. Optimization settings recorded for these runs are listed in Appendix A.8.

A.5 Training data

All methods train on the public Commonsense-15K release (Hu et al., 2023). The training split used in our runs contains about $15{,}100$ multiple-choice examples drawn from BoolQ, PIQA, Social IQA, HellaSwag, WinoGrande, ARC-Easy, ARC-Challenge, and OpenBookQA, with BoolQ passages retained. Training uses a candidate-ranking objective over answer continuations.

A.6 BENCHMARKS

The ID suite is the eight-task commonsense split listed in Section 6.1. The reported ID score is their unweighted macro-average. The OOD suite covers general knowledge (MMLU), mathematical reasoning (GSM8K and MATH-500), and code generation (HumanEval and MBPP). Math and Code scores are unweighted averages within each domain. MMLU is reported separately. We briefly summarize each benchmark below.

ID benchmarks. BoolQ (Clark et al., 2019) is a yes/no reading-comprehension dataset of naturally occurring questions paired with Wikipedia passages. PIQA (Bisk et al., 2020) tests physical commonsense by asking which of two candidate solutions better achieves a stated everyday goal. Social IQA (Sap et al., 2019) probes social and emotional commonsense with multiple-choice questions about everyday social situations. HellaSwag (Zellers et al., 2019) is a commonsense sentence-completion benchmark that selects the most plausible continuation among adversarially filtered endings. WinoGrande (Sakaguchi et al., 2020) is a large-scale, adversarially filtered Winograd-style pronoun-resolution challenge requiring commonsense coreference. ARC-Easy and ARC-Challenge (Clark et al., 2018) are grade-school science multiple-choice splits from the AI2 Reasoning Challenge; the Challenge subset contains questions that defeat simple retrieval and word-co-occurrence baselines. OpenBookQA (Mihaylov et al., 2018) asks elementary science questions that combine a provided core fact with additional common knowledge.

OOD benchmarks. MMLU (Hendrycks et al., 2021a) measures multitask accuracy across 57 academic and professional subjects spanning STEM, humanities, social sciences, and related domains. GSM8K (Cobbe et al., 2021) contains linguistically diverse grade-school math word problems that typically require multi-step elementary arithmetic. MATH-500 is a 500-problem evaluation subset of the MATH competition-problem dataset (Hendrycks et al., 2021b), following the held-out split released with process-supervised verification (Lightman et al., 2023). HumanEval (Chen et al., 2021) evaluates functional correctness of Python programs synthesized from docstrings against unit tests. MBPP (Austin et al., 2021) (Mostly Basic Programming Problems) contains short Python tasks designed to be solvable by entry-level programmers, each paired with a natural-language description and tests.

A.7 EVALUATION DETAILS

All reported scores use open-source evaluators at their default zero-shot settings, rather than custom scoring rules. Commonsense tasks and MMLU are evaluated with LM Evaluation Harness (Gao et al., 2024); we report accuracy, and normalized accuracy when that is the task's preferred metric. GSM8K and MATH-500 are evaluated with Math-Verify (Kydlíček, 2025). HumanEval and MBPP are evaluated with EvalPlus (Liu et al., 2023), and we report pass@1.

Table 9: RC-GEM hyperparameters for the reported OLMoE and Qwen1.5-MoE runs.

Hyperparameter	OLMoE	Qwen1.5-MoE
Learning rate	3×10^{-4}	3×10^{-4}
Subspace rank r	128	64
Number of groups G	32	32
Residual bound ε	0.3	0.25
Orthogonal-regularization weight	4.0	4.0
Router-logit normalization	LayerNorm	LayerNorm
Shared-expert modulation	–	yes
Evaluation checkpoint (epochs)	2.0	2.0

A.8 OPTIMIZATION CONFIGURATIONS

RC-GEM keeps the pretrained model frozen and trains only the subspace bases and modulation heads. Table 9 summarizes the RC-GEM hyperparameters for the reported runs. Shared optimization defaults are an effective batch size of 32, a sequence cutoff of 384, cosine learning-rate decay with warmup ratio 0.03, and weight decay 10^{-2} on the modulation-head weight matrices. On

Qwen1.5-MoE, RC-GEM also modulates the frozen shared expert. For both backbones, we evaluate the 2.0-epoch checkpoint with the lowest validation loss. We select the RC-GEM learning rate by sweeping over $\{5 \times 10^{-5}, 1 \times 10^{-4}, 2 \times 10^{-4}, 3 \times 10^{-4}, 4 \times 10^{-4}\}$ and retaining the setting with the lowest validation loss.

For all baselines, method-specific hyperparameters follow the recommendations in the original papers or the defaults of the official implementations, while the remaining training and evaluation settings are kept consistent with RC-GEM. Table 10 lists the learning rates and evaluation epochs used in Table 1; each reported epoch is the checkpoint with the lowest validation loss.

Table 10: Learning rates and evaluation epochs for the baselines in Table 1. Each epoch is the checkpoint with the lowest validation loss.

	OLMoE		Qwen1.5-MoE	
Method	LR	Epoch	LR	Epoch
FFT	3×10^{-5}	2.0	1×10^{-5}	1.0
LoRA	1×10^{-4}	2.0	2×10^{-4}	2.0
Prefix Tuning	5×10^{-5}	2.0	5×10^{-5}	2.0
Router-FT	2×10^{-4}	2.0	3×10^{-4}	2.0
RoMA	7×10^{-5}	2.75	7×10^{-5}	3.0
ACE	1×10^{-5}	2.25	1×10^{-5}	3.0
PERFT-E	1×10^{-5}	1.0	1×10^{-5}	1.0