AN EMPIRICAL STUDY OF PARAMETER-EFFICIENT FINE-TUNING FOR MIXTURE-OF-EXPERTS LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Parameter-efficient fine-tuning (PEFT) methods are critical for adapting large models under resource-constrained settings. However, as mixture-of-experts (MoE) architectures, which offer efficient inference at massive parameter scales, gain increasing prevalence in the community, a critical gap arising from the lack of systematic recipes for applying PEFT to MoE models. To narrow this gap, we conduct a large-scale systematic study that investigates a set of essential design recipes, including which modules should be adapted, which granularity should be used to adapt MoE modules, which PEFT method to use, and to what extent should the load-balancing objective be used, across three MoE backbones on downstream tasks and general knowledge tasks. Our experiments yield several key insights. First, adapting MoE experts alone achieves the highest downstream performance with competitive general capability retention under a comparable parameter budget. Second, native expert-weight updates offer a better downstream–retention trade-off than adapters that bypass experts or entire MoE modules. Third, LoRA and DoRA together define the strongest operating region for high downstream accuracy among the evaluated PEFT methods. Fourth, no nonzero load-balancing coefficient Pareto-dominates the zero baseline. We finally evaluate a weight-level redesign of MoE^2-LoRA on Qwen3. Moving its module-level bypass pool into native expert weights improves all four upstream endpoints by 2.4 to 17.9 points with 17% fewer trainable parameters and a 3.2-point average downstream cost.

1 INTRODUCTION

Mixture-of-experts architectures (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022; Du et al., 2022) enable extensive scaling of model parameters while activating only a small fraction of parameters per input unit. This sparse activation paradigm offers an outstanding trade-off between performance and computation, and has consequently been adopted by frontier models such as GLM-5.2 (GLM-5 Team, 2026) and Kimi-K3 (Kimi Team, 2026). Despite the increasing prevalence of MoE architectures, most existing parameter-efficient fine-tuning (PEFT) methods (Hu et al., 2022; Houlsby et al., 2019; Li & Liang, 2021) and recipes were originally designed for dense models, and the community still lacks a systematic guideline for applying PEFT methods to MoE structures. Given that MoE models typically comprise a massive number of parameters, it is essential to adopt a scientifically grounded guideline for applying PEFT to such models to achieve efficient adaptation. This gap therefore raises a pressing question: *How can we adapt MoE models to achieve the best performance-efficiency balance with the least forgetting?*

To answer this question, we conduct a large-scale systematic empirical study using representative MoE backbones from different families, including Qwen3-30B-A3B-Base (Qwen Team, 2025), Gemma-4-26B-A4B (Gemma Team, 2026), and GPT-OSS-20B (OpenAI, 2025). Specifically, we structure our study around four main axes: (1) which modules should be targeted for adapting MoE models under a similar trainable parameter budget? (2) should adaptation granularity for MoE modules be set at the module level, expert level, or weight level? (3) which PEFT method generally works best for adapting MoE models? (4) how strong should the auxiliary loss coefficient be set for adapting MoE models? We conduct the study using two distinct downstream tasks and two general tasks. The first downstream task is mathematical reasoning, on which base models typically already possess strong capabilities; it serves to probe whether further leverages well-established knowledge. The second downstream task is an intent classification task, which yields unsatisfactory results for

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.

the three base models and therefore provides a more stringent test of a PEFT method's ability to adapt in under-represented regimes.

Our study therefore yields four instructive insights: (1) applying adaptation solely to experts, achieves the best downstream performance; (2) within the MoE modules, applying adaptation at the weight level is more reliable than at the expert level or module level; (3) among the evaluated PEFT methods, LoRA and DoRA deliver the strongest downstream performance, whereas methods with weaker general capabilities retention can fail to learn the target task properly; (4) auxiliary load balancing may degrade performance on both downstream tasks and general capabilities retention. Guided by these findings, we revisit MoE^2-LoRA, a PEFT method tailred for MoE fine-tuning, which established a learned mapping between the selection of pre-trained MoE experts and that of LoRA experts. Our modification on MoE^2-LoRA improves all four upstream endpoints by 2.4 to 17.9 points with 17% fewer trainable parameters and only a 3.2-point average downstream cost, turning a large task-dependent forgetting into near-complete retention and thereby showing that our insights are effective beyond the controlled grid.

Our contributions are summarized as follows:

- To our knowledge, this is the first large-scale PEFT study conducted directly on pretrained MoE LLMs, covering four design axes, three heterogeneous backbones, and two downstream tasks. Beyond the downstream accuracy that PEFT work usually reports alone, we pair every selected checkpoint with general-capability evaluation on MMLU-Pro and IFEval, so each design choice is judged by what it gains and by what it costs the pretrained model.
- We distill the study into four actionable insights on adaptation position, expert granularity, update form, and load balancing, each stated with the regime in which it holds, providing concrete defaults and open questions for future work on MoE adaptation. Functional and spectral analyses explain how weight-level updates reuse pretrained expert computation.
- We show the insights are practical by refining MoE^2-LoRA, a recent routed-pool method, along the single axis it does not already follow, which improves all four general-capability endpoints with 17% fewer parameters.

2 Related Work

Parameter-efficient Fine-tuning (PEFT). PEFT adapts large models by learning a small set of parameters while keeping pretrained weights largely frozen. Low-rank methods modify pretrained weights through compact updates: LoRA learns a low-rank residual, while DoRA decouples weight magnitude and direction to improve the update (Hu et al., 2022; Liu et al., 2024a). OFT and FourierFT further restrict the update space through orthogonal transformations and frequency-domain parameterization, respectively (Qiu et al., 2024; Gao et al., 2024b), whereas IA3 adapts the model by scaling intermediate activations (Liu et al., 2022). A complementary line introduces conditional computation into PEFT by organizing prompts, adapters, or LoRA modules as routed experts (Dou et al., 2023; Li et al., 2024; Gao et al., 2024a; Tian et al., 2024; Luo et al., 2024). However, these methods are predominantly validated on dense models. It therefore remains unclear how PEFT should interact with the existing experts and routers of an MoE model. In this paper, we systematically study adaptation position, expert granularity, update form, and load balancing to derive principled guidelines for applying PEFT to MoE models.

PEFT for MoE architecture models. PEFT for pretrained MoE models is not a straightforward extension of dense-model PEFT, because sparse routing governs how limited trainable parameters receive supervision. Since each token is dispatched to a few experts, PEFT modules receive uneven, task-dependent signals, making expert selection and update sharing central design choices (Wang et al., 2024c). Conventional top-k routing also provides incomplete gradients for PEFT modules optimization during post-training (Yao et al.). Moreover, load-balancing objectives may enforce overly uniform routing and weaken the specialization induced by expert-specific updates on narrow downstream distributions (Wang et al., 2024a; Guo et al., 2025). Existing PEFT studies address parts of this through task-relevant expert selection, expert-wise allocation, joint router–expert adaptation, or additional routed adaptation modules (Wang et al., 2024c; Bai et al., 2025; Deng et al., 2026; Liu et al., 2024b; Yang et al., 2026). However, they often vary simultaneously in adaptation position,

expert granularity, update form, routing strategy, and balancing objective, making it difficult to determine which choices consistently benefit. We therefore conduct a controlled study of these factors to derive robust principles for MoE adaptation.

3 Background and Study Design

3.1 Background

This work investigates the principled application of PEFT methods to MoE models, with an emphasis on the MoE modules. Formally, given a hidden state x, the computation inside an MoE module can be written as

$$y = \sum_{e \in \mathrm{TopK}(g(x))} p_e(x)\, f_e(x), \qquad (1)$$

where g denotes the router, p_e is the routing probability for expert e, and f_e represents the transformation of expert e.

We model the fine-tuning process as an incremental update of the pre-trained weights:

$$W' = W + \Delta W, \qquad (2)$$

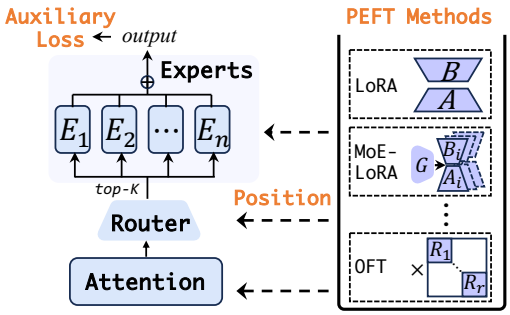


Figure 1: Illustration of Position, PEFT Methods, and Auxiliary Loss.

where $W' \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ is the fine-tuned weight, and W and ΔW correspond to the pre-trained weight and its additive update, respectively. In the case of LoRA, the update is parameterized as $\Delta W = \frac{\alpha}{r} BA$, with low-rank matrices $A \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B \in \mathbb{R}^{d_{\text{out}} \times r}$, while α and r are hyperparameters controlling the update scale and rank. Other PEFT methods evaluated in this study are deferred to Appendix A.

3.2 Study Design

We evaluate Qwen3-30B-A3B-Base, Gemma-4-26B-A4B, and GPT-OSS-20B on two complementary downstream tasks, specifically mathematical reasoning and intent classification. Appendix E details their architectures. For the intent classification task, the models are trained on ESFT_Intent (Wang et al., 2024c) and evaluated by exact match on 500 held-out examples. For the mathematical reasoning task, models are trained on MetaMathQA-R1, derived from MetaMathQA (Yu et al., 2024), and evaluated on MATH-500 (Hendrycks et al., 2021). We use MMLU-Pro (Wang et al., 2024b) and IFEval (Zhou et al., 2023) to measure the general capabilities, including knowledge-based reasoning and instruction-following retained after downstream adaptation. For each configuration and model–task pair in the four design studies, we train three seeds per candidate learning rate and select the rate with the highest mean downstream score. We evaluate the three checkpoints at that rate on MMLU-Pro and IFEval and report three-seed means, without using general scores for selection. **Detailed implementation settings are provided in Appendix B.** Selected learning rates and Qwen3 training throughput are reported in Appendix H.

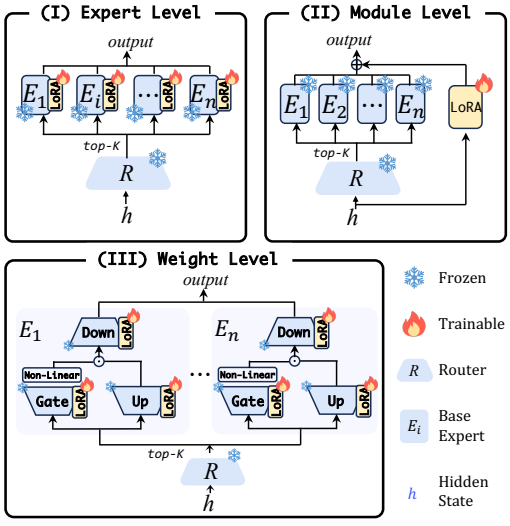


Figure 2: Three evaluated adaptation granularities of the MoE module.

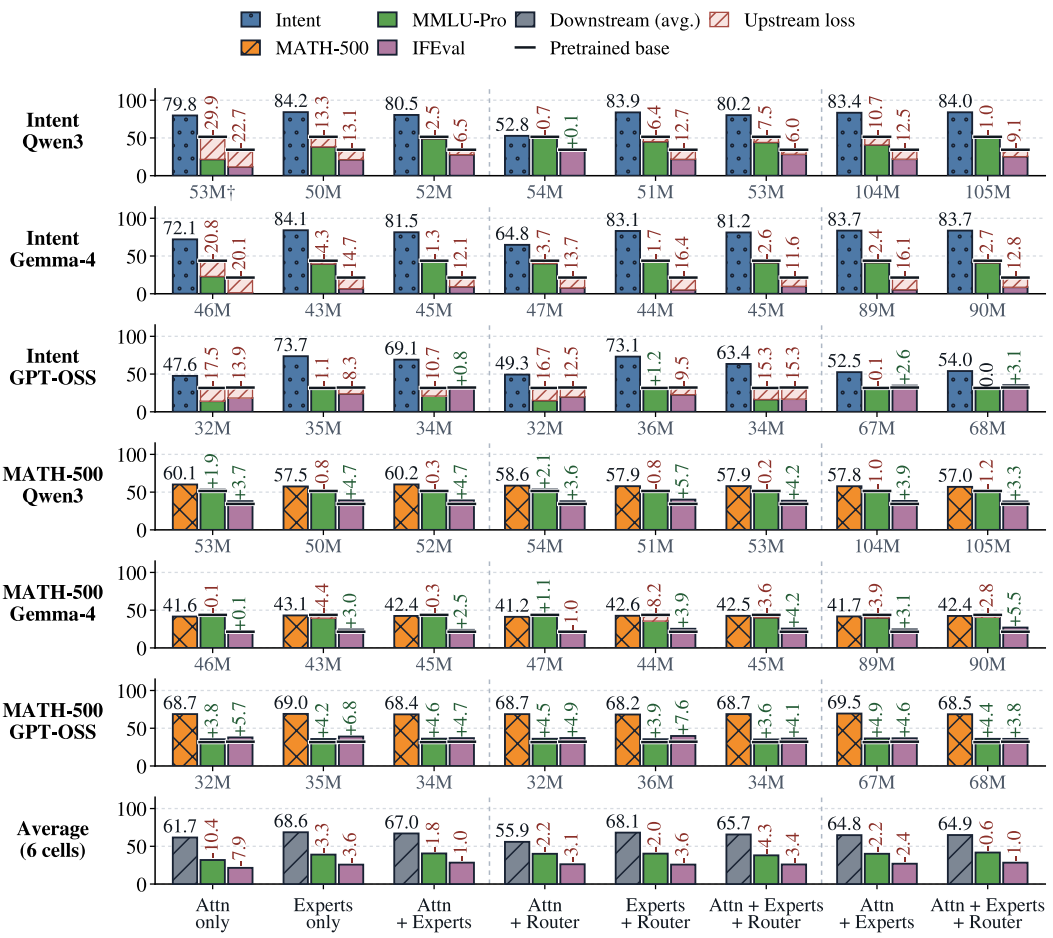


Figure 3: Visualized results of the study of adapted modules. Attn denotes attention, and gray labels report trainable parameters.

4 Empirical Study

4.1 Which Modules Should be Targeted for Adaptation?

We compare attention-only ($r = 64$), expert-only ($r = 256$), and attention+experts ($r = 32$ and 128) under a comparable budget, using one shared branch for the expert block to keep the focus on position. We repeat each with a rank-8 router update and also test the higher-budget attention/expert pair ($r = 64/256$), with and without router adaptation.

Figure 3 summarizes the adaptation-position study. Expert-only adaptation leads across the six model–task cells at 68.6%, ahead of experts+router at 68.1% and attention+experts at 67.0%. It changes MMLU-Pro and IFEval by -3.3 and -3.6 points. Attention-only reaches 61.7%, with changes of -10.4 and -7.9 points. Attention+experts changes the two benchmarks by -1.8 and -1.0 points, while experts+router changes them by -2.0 and -3.6 points. Without and with router adaptation, the higher-budget settings reach 64.8% and 64.9%; MMLU-Pro changes by -2.2 and -0.6 points, and IFEval by -2.4 and -1.0 points. Expert-only is strongest at comparable budget, while attention+experts retains more general capability.

Insight 1.

For MoE fine-tuning, target the experts by default, and adapt the router only when the task calls for routing specialization.

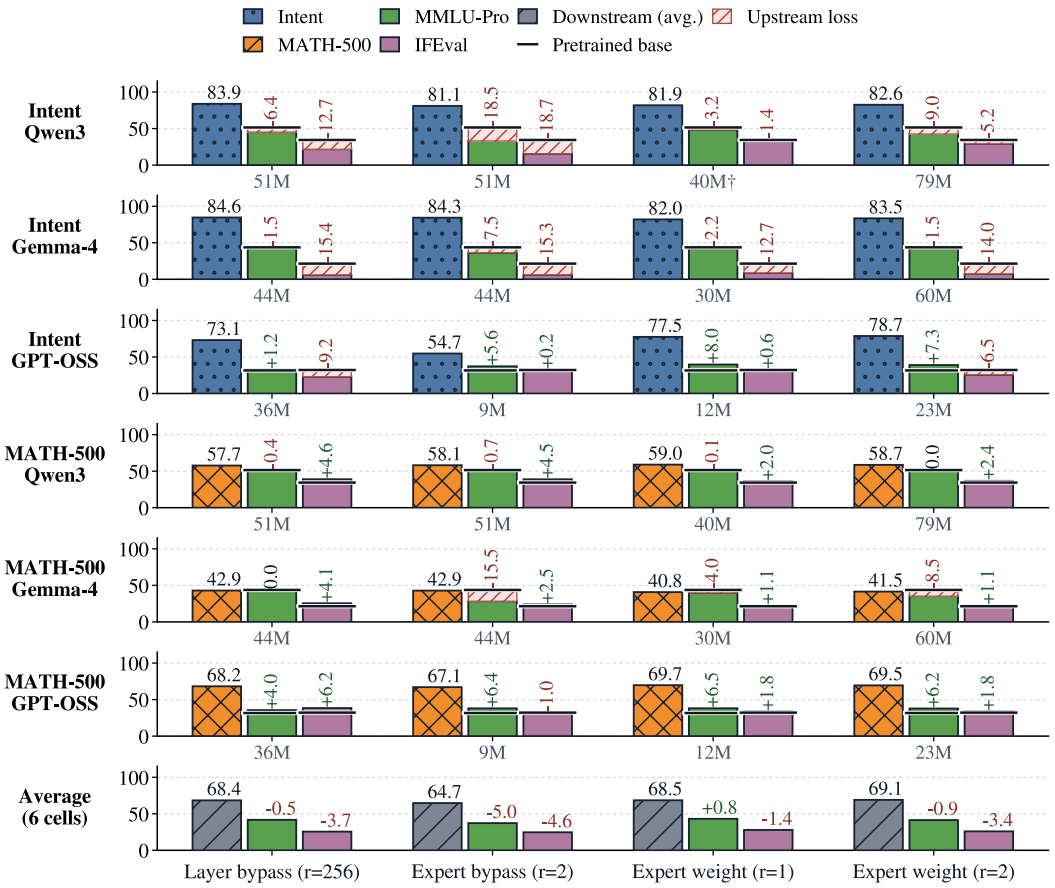


Figure 4: Visualized results of the study of MoE adaptation granularity.

4.2 Which Granularity Should be Used to Adapt MoE Modules?

Figure 2 contrasts a rank-256 module-level bypass, a rank-2 expert-level bypass, and weight-level updates at ranks 1 and 2.

Figure 4 shows weight-level adaptation leading at rank 2 with 69.1%, followed by rank 1 at 68.5%, module-level bypass at 68.4%, and expert-level bypass at 64.7%. Rank 1 has the strongest general-capability profile, changing MMLU-Pro and IFEval by $+0.8$ and -1.4 points; rank 2 changes them by -0.9 and -3.4. Both bypass alternatives reduce both benchmarks. Rank 2 gains 0.6 downstream points at roughly twice the trainable-parameter count.

Insight 2.

Weight-level adaptation provides the best trade-off. Rank-1 is a strong default, achieving near-optimal downstream performance while preserving general capabilities. Increasing the rank brings only marginal gains, whereas bypass-based adaptation is generally less stable and less effective.

4.3 Which PEFT Method Generally Works Best for Adapting MoE Models?

We compare LoRA (Hu et al., 2022), DoRA (Liu et al., 2024a), OFT (Qiu et al., 2024), FourierFT (Gao et al., 2024b), and IA3 (Liu et al., 2022), alongside expert-level LoRA, MoE-LoRA (Luo et al., 2024), module-level LoRA, DAS-LoRA (Tang et al., 2026), and last-layer partial fine-tuning (Ye et al., 2023).

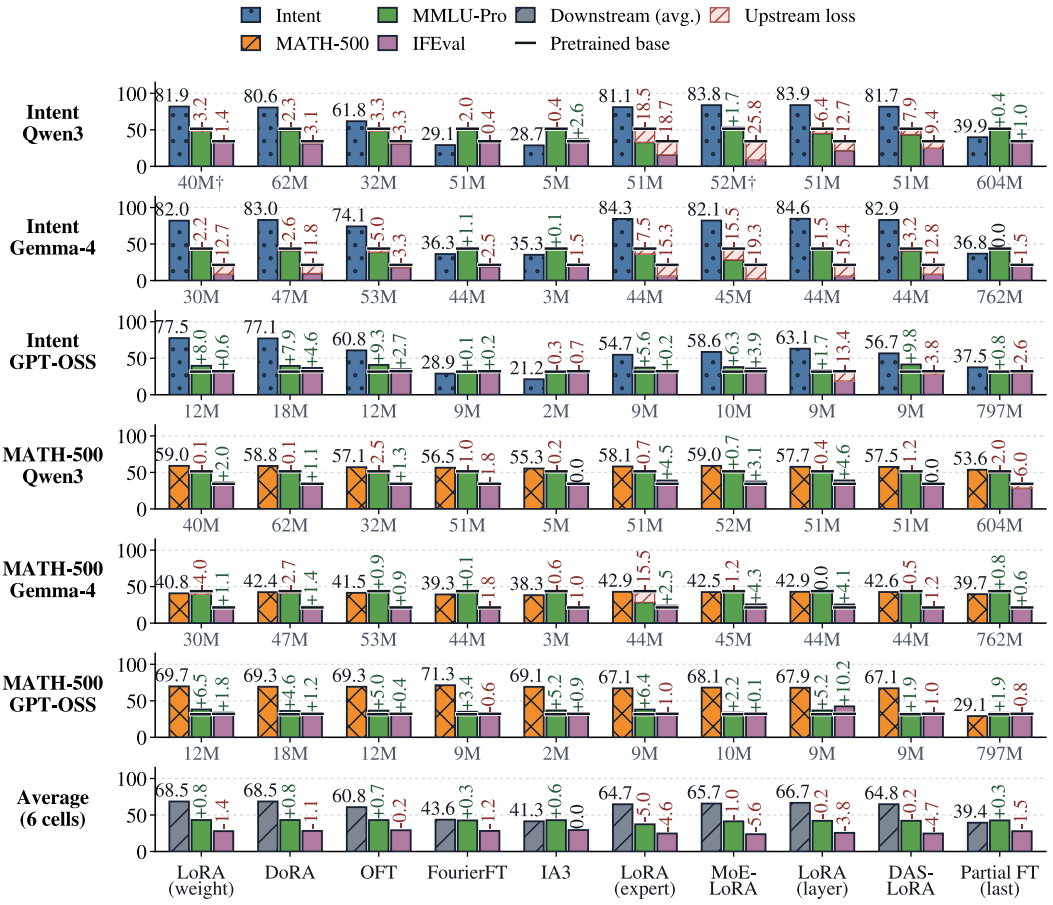


Figure 5: Visualized results of the study of PEFT methods.

Weight-level LoRA and DoRA tie at 68.5% across the six downstream cells. Both raise MMLU-Pro by 0.8 points; DoRA changes IFEval by -1.1, compared with -1.4 for LoRA. Among MoE-specific variants, layer-level LoRA leads at 66.7%, followed by MoE-LoRA at 65.7%, DAS-LoRA at 64.8%, and expert-level LoRA at 64.7%. OFT, FourierFT, IA3, and partial fine-tuning reach 60.8%, 43.6%, 41.3%, and 39.4%. Weight-level LoRA and DoRA provide the strongest downstream accuracy, with DoRA retaining more IFEval performance.

Insight 3.

Start expert-weight adaptation with standard LoRA or DoRA. Favor DoRA when preserving instruction-following performance is a priority.

4.4 How Strong Should the Auxiliary Loss Coefficient Be Set For Adapting MoE Models?

We vary the auxiliary loss coefficient from zero to 10^{-1} while holding the adaptation method and trainable-parameter count fixed, as illustrated in Figure 1.

Figure 6 shows a 68.5% downstream average without auxiliary loss; coefficients of 10^{-4}, 10^{-3}, 10^{-2}, and 10^{-1} yield 68.1%, 66.8%, 64.6%, and 58.6%. At 10^{-3}, MMLU-Pro rises 1.2 points, IFEval falls 2.5, and the downstream average falls 1.7 relative to zero. At 10^{-1}, MMLU-Pro and IFEval fall 0.9 and 8.5 points. Larger coefficients therefore reduce downstream performance and IFEval retention.

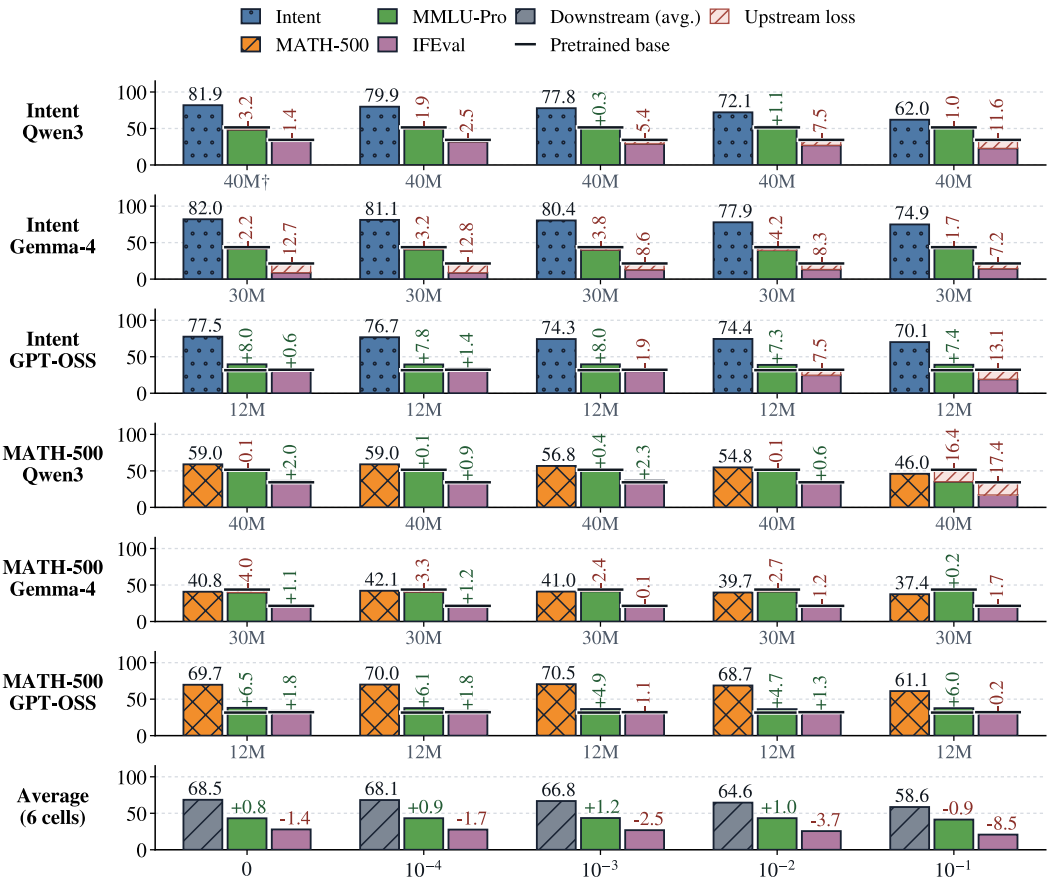


Figure 6: Visualized results of the study of the strength of auxiliary loss.

Insight 4.

The auxiliary loss should be disabled or assigned a very small coefficient during MoE adaptation. A coefficient around 10^{-4} is a safe choice when routing regularization is required, whereas larger values can substantially degrade downstream performance.

5 Design Implications

To test whether the insights transfer to a composite method, we refine MoE^2-LoRA (Yang et al., 2026). It follows Insight 1 by adapting only MoE modules with frozen attention and follows Insight 4 by disabling auxiliary load balancing. It also uses the native router to select entries from a shared pool without updating router weights. Its remaining mismatch is Insight 2: the pool is a module-level bypass rather than an update to native expert projections. We therefore replace each of the 128 rank-32 pool entries with a low-rank update to native expert weights while retaining the routed pool, router-driven selection, frozen attention, and zero balancing coefficient. Appendix A.1 gives both update equations.

Checkpoint	Benchmark	Original	Weight Level
Trainable params		47.71M	**39.58M**
Math	MATH-500	**60.0**	55.8
	MMLU-Pro	48.19	**50.63**
	IFEval	35.12	**40.30**
Intent	Intent	**84.0**	81.8
	MMLU-Pro	45.26	**51.20**
	IFEval	18.11	**36.04**

Table 1: Comparison of the original MoE^2-LoRA and the insight-guided expert-weight redesign on Qwen3.

Table 1 shows this change reduces the parameter count by 17.0%, from 47.71M to 39.58M. It improves MMLU-Pro and IFEval by 2.44 and 5.18 points on Math, and by 5.94 and 17.93 points on Intent. The mean absolute difference from the base falls from 6.64 to 2.19 points, and the worst loss from 16.27 to 0.87. Intent remains near base on MMLU-Pro at -0.30 and exceeds base on IFEval by 1.66 points; empty responses stay low and truncation falls from 49.2% to 11.1%. The redesign costs 3.2 downstream points while preserving high downstream performance. Appendix D reports additional MoE^2-LoRA experiments showing that adding attention adaptation or auxiliary load balancing does not consistently improve performance across tasks.

6 Analysis of Adaptation Granularity

We explain the advantage of weight-level adaptation in Insight 2 through functional response diversity and consistency with pretrained expert structure. We use RMSNorm-scaled probes to analyze functional response diversity and pretrained spectral consistency in Qwen3 Math and Intent adapters.

6.1 Functional Response Diversity

For a layer with base output F_0 and adapted output F_a, collect the changes on shared probes as $Y = [F_a(x_j) - F_0(x_j)]_{j=1}^{n}$. We measure how broadly response energy spans output directions using the functional effective rank

$$r_{\text{eff}}(Y) = \frac{(\operatorname{tr} K)^2}{\|K\|_F^2 + \varepsilon}, \qquad K = \frac{1}{n} Y Y^\top. \tag{3}$$

Equal energy in k orthogonal directions gives $r_{\text{eff}} = k$ up to the numerical stabilizer, making the four granularities comparable.

Weight-level updates reuse the frozen expert's nonlinear features. For a gated expert $f_e(x) = D_e[\sigma(G_e x) \odot U_e x]$, a rank-one up-projection update $\Delta U_e = ab^\top$ gives the exact correction

$$\delta_U f_e(x) = (b^\top x) D_e \operatorname{Diag}(a) \sigma(G_e x). \tag{4}$$

The frozen nonlinear features $\sigma(G_e x)$ let a rank-one update span multiple output directions, while routing combines expert-specific responses. Appendix F gives the derivation, rank bounds, and probe protocol.

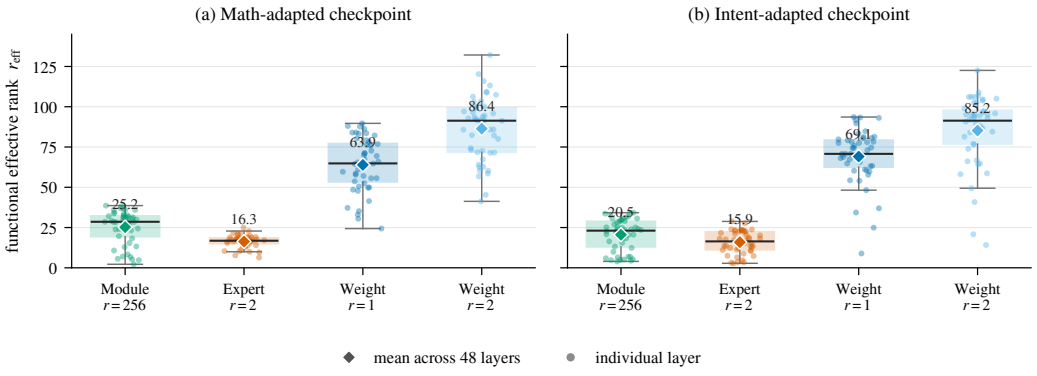


Figure 7: Functional response diversity across all 48 Qwen3 MoE layers on synthetic probes. Points denote layers, boxes show medians and interquartile ranges, and diamonds mark means. Weight-level rank-one updates produce broad layer responses on both adapted checkpoints.

Figure 7 shows mean effective ranks of 63.90 and 69.08 for weight-level $r = 1$ on Math and Intent, compared with 16.31 and 15.91 for expert-level $r = 2$. The module-level $r = 256$ update reaches 25.25 and 20.54, while weight-level $r = 2$ reaches 86.37 and 85.22. Native weight updates therefore produce broad responses at low rank, with additional rank expanding coverage.

6.2 Pretrained Spectral Consistency

To test alignment with the pretrained expert, let $u_{e,i}$ and $\sigma_{e,i}$ denote the left singular vectors and descending singular values of its frozen down projection. For perturbations Y_e, define the cumulative energy S_k and pretrained profile $S^0_{k,e}$ as

$$S_k = \frac{\sum_{i\le k}\|u_{e,i}^\top Y_e\|_2^2}{\sum_{i\le m}\|u_{e,i}^\top Y_e\|_2^2+\varepsilon}, \qquad S^0_{k,e} = \frac{\sum_{i\le k}\sigma_{e,i}^2}{\sum_{i\le m}\sigma_{e,i}^2}. \tag{5}$$

At cutoffs $\mathcal{K}=\{16,64,256\}$ and expert width $m=768$, the spectral-consistency score is

$$C_{\text{spec}} = 1 - \frac{\sum_{k\in\mathcal{K}}|S_k - S^0_{k,e}|}{\sum_{k\in\mathcal{K}}|k/m - S^0_{k,e}|+\varepsilon}. \tag{6}$$

A score of one matches the pretrained profile, while zero denotes uniform energy. Appendix G gives the derivation and full spectral and output-subspace results.

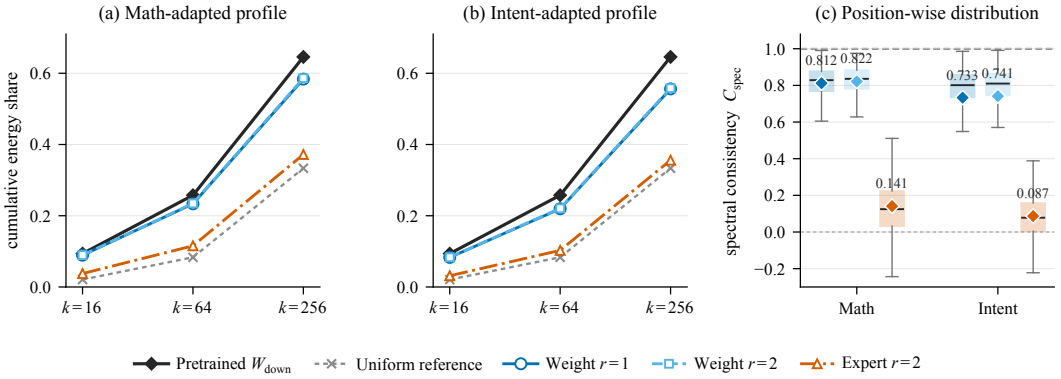


Figure 8: Pretrained spectral consistency on Qwen3: Math and Intent energy profiles and score distributions across layer–expert positions. Weight-level updates follow the pretrained spectrum.

Figure 8 shows mean C_{spec} values of 0.812 and 0.822 on Math and 0.733 and 0.741 on Intent for weight-level ranks 1 and 2, versus 0.141 and 0.087 for expert-level rank 2. Rank-one weight updates lead across all 48 layer means and at 99.8% of Math and 96.3% of Intent positions. Measurements across gate, up, and down projections confirm spectral alignment; rank two broadens responses while retaining it. These results explain the low-rank efficiency behind Insight 2.

7 Conclusion

MoE-PEFT is not only a choice of update parameterization; placement, granularity, optimization, and routing objectives materially change its reliability. Across three backbones and two tasks, expert FFNs are the downstream-robust placement, native expert weights are the most reliable evaluated granularity, LoRA/DoRA lead when a high downstream floor is enforced, and zero auxiliary load balancing is the only no-tuning joint default. None is universal: mixed placement can drift less, low-drift update forms can adapt poorly, and a moderate auxiliary coefficient can trade downstream accuracy for better retention of some general capabilities. Our Qwen3 analyses show that weight-level adaptation combines broad functional responses with strong reuse of pretrained spectral structure. These defaults also carry over beyond our own grid: a weight-level redesign of MoE^2-LoRA that moves its module-level bypass pool into native expert weights improves all four upstream endpoints by 2.4 to 17.9 points with 17% fewer trainable parameters at a small downstream cost. We therefore offer these defaults as a scoped but directionally reliable starting point for MoE adaptation. We hope our findings and analyses can stimulate further research on parameter-efficient adaptation of MoE models, and encourage future methods to report what each design choice costs the pretrained model alongside what it gains.

AI USE STATEMENT

In this work, we used generative AI tools solely to aid and polish the writing, including improving grammar, clarity, concision, and presentation of author-written text. We did not use generative AI for literature retrieval or discovery, research ideation or execution, or generating synthetic data. All AI-assisted edits were reviewed and revised by the authors, who verified the technical accuracy and consistency of the final manuscript. We take full responsibility for the final content of this work.

REFERENCES

Jun Bai, Minghao Tong, Yang Liu, Zixia Jia, and Zilong Zheng. Understanding and leveraging the expert specialization of context faithfulness in mixture-of-experts LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21927–21942. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1114.

Guanzhi Deng, Bo Li, Ronghao Chen, Xiujin Liu, Zhuo Han, Huacan Wang, Lijie Wen, and Linqi Song. DR-LoRA: Dynamic rank LoRA for fine-tuning mixture-of-experts models, 2026.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. LoRAMoE: Alleviate world knowledge forgetting in large language models via MoE-style plugin, 2023.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V. Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. GLaM: Efficient scaling of language models with mixture-of-experts. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 5547–5569, 2022.

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.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and V. S. Subrahmanian. Higher layers need more LoRA experts, 2024a.

Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. Parameter-efficient fine-tuning with discrete fourier transform, 2024b.

Gemma Team. Gemma 4 technical report, 2026.

GLM-5 Team. GLM-5: From vibe coding to agentic engineering, 2026.

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. URL https://papers.nips.cc/paper_files/paper/2025/file/4598de7d243d528e38eb0c5d8155fb52-Paper-Conference.pdf.

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 Datasets and Benchmarks Track*, 2021.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799, 2019.

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.

Kimi Team. Kimi K3: Open frontier intelligence, 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.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, and Mingjie Tang. MixLoRA: Enhancing large language models fine-tuning with LoRA-based mixture of experts, 2024.

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*, pp. 4582–4597, 2021. doi: 10.18653/v1/2021.acl-long.353.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation, 2024a.

Yilun Liu, Yunpu Ma, Shuo Chen, Zifeng Ding, Bailan He, Zhen Han, and Volker Tresp. PERFT: Parameter-efficient routed fine-tuning for mixture-of-expert model, 2024b.

Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. MoELoRA: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models, 2024.

OpenAI. gpt-oss-20b model card. Hugging Face model card: `https://huggingface.co/openai/gpt-oss-20b`, 2025. Accessed 2026-07-24.

Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. Controlling text-to-image diffusion by orthogonal finetuning, 2024.

Qwen Team. Qwen3 technical report, 2025.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 3505–3506, 2020.

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.

Yiru Tang, Kun Zhou, Xin Zhao, Jing Sha, Zhichao Sheng, and Shijin Wang. Exploring expert concentration for parameter-efficient fine-tuning of mixture-of-expert LLMs, 2026. URL `https://openreview.net/forum?id=zBgjWTWgCh`.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. HydraLoRA: An asymmetric LoRA architecture for efficient fine-tuning, 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. doi: 10.48550/arXiv.2408.15664. URL `https://arxiv.org/abs/2408.15664`.

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 Z. Wang, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark, 2024b.

Zihan Wang, Deli Chen, Damai Dai, Runxin Xu, Zhuoshu Li, and Yu Wu. Let the expert stick to his last: Expert-specialized fine-tuning for sparse architectural large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 784–801. Association for Computational Linguistics, 2024c. doi: 10.18653/v1/2024.emnlp-main.46.

Qingyu Yang, Haonan He, Minglei Li, Jingqi Ye, Tao Chen, Lei Bai, and Peng Ye. MoE^2-LoRA: When MoE models meet MoE-style low-rank adaptation, 2026.

Feng Yao, Junxia Cui, Ruohan Zhang, Liyuan Liu, Shibo Hao, Li Zhang, Chengyu Dong, Shuohang Wang, Jianfeng Gao, Jingbo Shang, et al. Densemixer: Improving moe post-training with precise router gradient. In *NeurIPS 2025 Workshop on Structured Probabilistic Inference {\&} Generative Modeling*.

Peng Ye, Yongqi Huang, Chongjun Tu, Minglei Li, Tao Chen, Tong He, and Wanli Ouyang. Partial fine-tuning: A successor to full fine-tuning for vision transformers, 2023. URL `https://arxiv.org/abs/2312.15681`.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations*, 2024.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, and Denny Zhou. Instruction-following evaluation for large language models, 2023.

An Empirical Study of Parameter-Efficient Fine-Tuning for Mixture-of-Experts LLMs

Supplementary Material

The supplementary material is organized as follows:

- Appendix A defines the MoE and PEFT updates, including MoE^2-LoRA and its redesign.
- Appendix B describes the datasets, checkpoint selection, and evaluation protocol.
- Appendix C gives the settings for the four design studies.
- Appendix D reports additional MoE^2-LoRA placement and balancing experiments.
- Appendix E compares the backbones' attention and expert architectures.
- Appendix F derives and measures functional response diversity across adaptation granularities.
- Appendix G analyzes pretrained spectral structure and expert output-subspace alignment.
- Appendix H reports selected learning rates and Qwen3 training throughput.

A Overview and Notation

Let $x \in \mathbb{R}^d$ denote the token representation entering an MoE layer. Each routed expert is a gated feed-forward network. Using the fused gate/up projection employed by the evaluated backbones, expert e is

$$\begin{bmatrix} a_e(x) \\ u_e(x) \end{bmatrix} = W_{\mathrm{gu},e} x, \qquad W_{\mathrm{gu},e} \in \mathbb{R}^{2d_{\mathrm{ff}} \times d}, \tag{7}$$

$$h_e(x) = \phi(a_e(x)) \odot u_e(x), \tag{8}$$

$$f_e(x) = W_{\mathrm{down},e} h_e(x), \qquad W_{\mathrm{down},e} \in \mathbb{R}^{d \times d_{\mathrm{ff}}}, \tag{9}$$

where ϕ is the backbone activation function; for a SwiGLU expert, ϕ is SiLU. Equivalently, $W_{\mathrm{gu},e}$ may be written as separate gate and up matrices $W_{\mathrm{gate},e}$ and $W_{\mathrm{up},e}$. Biases are omitted for clarity.

Let $g(x) \in \mathbb{R}^E$ denote the native router logits over E routed experts. For a router that renormalizes its selected probabilities, define

$$\bar{p}_e(x) = \frac{\exp(g_e(x))}{\sum_{j=1}^{E} \exp(g_j(x))}, \tag{10}$$

$$\mathcal{S}(x) = \mathrm{TopK}(\bar{p}(x), k)\,, \tag{11}$$

$$p_e(x) = \frac{\bar{p}_e(x)}{\sum_{j \in \mathcal{S}(x)} \bar{p}_j(x)}, \qquad e \in \mathcal{S}(x). \tag{12}$$

The routed MoE output is

$$y_{\mathrm{routed}}(x) = \sum_{e \in \mathcal{S}(x)} p_e(x) f_e(x). \tag{13}$$

If a backbone contains an always-on shared expert f_{sh}, the complete MoE output may be written abstractly as

$$y_{\mathrm{MoE}}(x) = y_{\mathrm{routed}}(x) + c_{\mathrm{sh}}(x) f_{\mathrm{sh}}(x), \tag{14}$$

where $c_{\mathrm{sh}}(x)$ is the native shared-expert gate, or one for an ungated shared expert. This term is omitted for backbones without a shared expert. Router normalization, shared-expert gating, residual placement, and normalization layers otherwise remain unchanged from the pretrained backbone.

Each routed expert contains fused gate/up and down projections. We use *Weight Level* for directly adapting these expert weights, *Expert Level* for applying one adaptation branch to each routed expert, and *Module Level* for applying one shared adaptation branch to the complete MoE module in a layer. All unlisted pretrained parameters remain frozen.

DoRA. DoRA (Liu et al., 2024a) separates the magnitude and direction of an adapted weight. Under the convention $W_0 \in \mathbb{R}^{d_{\mathrm{out}} \times d_{\mathrm{in}}}$ and $y = W_0 x$, define

$$V = W_0 + \frac{\alpha}{r} BA, \tag{15}$$

$$m_0 = \|W_0\|_{\mathrm{row}}, \qquad m \in \mathbb{R}^{d_{\mathrm{out}}}, \tag{16}$$

where m is trainable and initialized as m_0. The adapted weight is

$$W' = \mathrm{Diag}(m \oslash \|V\|_{\mathrm{row}})\, V. \tag{17}$$

Here $\oslash$ denotes elementwise division. This row-wise expression is the transpose-equivalent form of the column-wise normalization used in the original DoRA notation.

The memory-efficient implementation replaces the normalizer used for gradient computation by

$$c(V) = \mathrm{stopgrad}(\|V\|_{\mathrm{row}})\,, \qquad W' = \mathrm{Diag}(m \oslash c(V))\, V. \tag{18}$$

The stop-gradient operation changes the backward graph, not the numerical forward value. Weight-level DoRA is applied independently to the fused gate/up and down projections of each adapted routed expert.

OFT. OFT (Qiu et al., 2024) applies the same orthogonal transformation to the weight vector of every output neuron in a frozen projection. For a skew-symmetric parameter Q,

$$Q^\top = -Q, \qquad R = (I+Q)(I-Q)^{-1}, \qquad R^\top R = I. \tag{19}$$

Under the convention $y = Wx$, with $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, the adapted projection is

$$W' = WR^\top, \qquad y' = WR^\top x, \tag{20}$$

Here $R \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$ acts on the input dimension, equivalently transforming each row weight vector by the same orthogonal map. For block-diagonal OFT,

$$R = \mathrm{Diag}(R_1, \ldots, R_b), \tag{21}$$

and the block size must divide the input width d_{in}. The skew-symmetric parameters are initialized so that $R = I$ at initialization.

FourierFT. FourierFT (Gao et al., 2024b) represents a dense weight update using a small number of trainable Fourier coefficients. Let Ω be a fixed random support in the two-dimensional frequency domain, and let $S_\Omega(c)$ scatter the trainable real coefficient vector c into the corresponding frequency locations. The update is

$$\Delta W = \gamma \, \mathrm{Re}[\mathrm{IDFT}_2(S_\Omega(c))], \qquad W' = W + \Delta W. \tag{22}$$

The original method samples the support once and uses randomly initialized coefficients. In our benchmark implementation, the support is fixed with seed 2024, the coefficients are instead initialized to zero so that $\Delta W = 0$ initially, and the update scale is set to $\gamma = 150$. These are experimental implementation choices rather than defining properties of FourierFT.

IA3. IA3 (Liu et al., 2022) learns multiplicative activation vectors. For routed expert e, let

$$h_e(x) = \phi(W_{\text{gate},e}x) \odot W_{\text{up},e}x. \tag{23}$$

The feed-forward IA3 vector $\ell_{\text{ff},e} \in \mathbb{R}^{d_{\text{ff}}}$ modifies the expert as

$$f_e^{\text{IA3}}(x) = W_{\text{down},e}\left(\ell_{\text{ff},e} \odot h_e(x)\right). \tag{24}$$

All IA3 vectors are initialized to one. Canonical IA3 additionally scales attention key and value activations, but attention adaptation is disabled in the final PEFT method comparison.

Where router IA3 is enabled in our benchmark, it is a MoE-specific extension rather than part of canonical IA3. It is applied before top-k selection:

$$g^{\text{IA3}}(x) = \ell_{\text{router}} \odot g(x), \qquad \ell_{\text{router},0} = \mathbf{1}, \tag{25}$$

and $\mathcal{S}(x)$ and $p_e(x)$ are then computed from $g^{\text{IA3}}(x)$.

DAS-LoRA. DAS-LoRA (Tang et al., 2026) performs an offline analysis of native router statistics on downstream-domain and reference data. It computes the paper's cumulative Domain Advantage Score (CDAS) for each routed expert and selects a fixed budget of task-relevant experts in each layer:

$$\mathcal{E}_\ell^{\text{DAS}} = \mathrm{TopB}\Big(\{\mathrm{CDAS}_{\ell,e}\}_{e=1}^{E}, b_\ell\Big), \tag{26}$$

where b_ℓ is the selected-expert budget. We do not replace the paper's CDAS procedure by a simple difference of mean routing probabilities.

LoRA is attached only to the gate/up/down weights of the selected experts. Writing $\Delta f_{\ell,e}^{\text{LoRA}}(x)$ for the change induced by those weight-level low-rank updates, the layer output is

$$y'_\ell(x) = \sum_{e \in \mathcal{S}_\ell(x)} p_{\ell,e}(x)\left[f_{\ell,e}(x) + \mathbb{I}\big(e \in \mathcal{E}_\ell^{\text{DAS}}\big)\,\Delta f_{\ell,e}^{\text{LoRA}}(x)\right]. \tag{27}$$

Thus an adapted expert contributes only when it is both selected offline by DAS-LoRA and dispatched to the current token by the native router.

A.1 MoE^2-LoRA and the Insight-Guided Redesign

Canonical MoE^2-LoRA. MoE^2-LoRA (Yang et al., 2026) uses both native router information and a low-dimensional hidden-state channel to route over a global pool of LoRA experts. For MoE layer ℓ, let

$$z_\ell(x) = g_\ell(x) \in \mathbb{R}^E, \tag{28}$$

$$h^c_\ell(x) = W^h_\ell x \in \mathbb{R}^{d_b}. \tag{29}$$

The layer-specific LoRA-pool router is

$$q_\ell(x) = \mathrm{softmax}\left(W^q_\ell \begin{bmatrix} z_\ell(x) \\ h^c_\ell(x) \end{bmatrix}\right), \qquad W^q_\ell \in \mathbb{R}^{N_L \times (E+d_b)}. \tag{30}$$

Let

$$\mathcal{T}_\ell(x) = \mathrm{TopK}(q_\ell(x), K_L)\,. \tag{31}$$

The selected pool probabilities are renormalized as

$$\widetilde{q}_{\ell,j}(x) = \frac{q_{\ell,j}(x)}{\sum_{i\in\mathcal{T}_\ell(x)} q_{\ell,i}(x)}, \qquad j \in \mathcal{T}_\ell(x). \tag{32}$$

A single LoRA expert pool $\{(A_j, B_j)\}_{j=1}^{N_L}$ is shared across all MoE layers, while the routing projections are layer-specific. The canonical module-level update is

$$y'_\ell(x) = y_{\mathrm{MoE},\ell}(x) + \sum_{j\in\mathcal{T}_\ell(x)} \widetilde{q}_{\ell,j}(x)\frac{\alpha}{r} B_j A_j x. \tag{33}$$

The published Qwen3-30B-A3B configuration uses $N_L = 256$, rank $r = 32$, adapter top-6 routing, auxiliary dimension $d_b = 32$, and two hierarchical routing groups.

Weight-level redesign. Our redesign retains the native-router-conditioned pool selection but places the resulting low-rank residuals directly in the expert weights rather than applying a module-level adaptation after the complete MoE output.

Let

$$\pi_\ell(j) = (e, t) \tag{34}$$

map pool entry j to routed expert e and target projection $t \in \{\mathrm{gu}, \mathrm{down}\}$ in layer ℓ. For an input u to that projection, define the token-conditioned effective update

$$\Delta W_{\ell,e,t}(x) = \sum_{j\in\mathcal{T}_\ell(x)} \mathbb{I}[\pi_\ell(j) = (e,t)]\, \widetilde{q}_{\ell,j}(x)\frac{\alpha_t}{r} B_j A_j. \tag{35}$$

The corresponding expert projection is evaluated as

$$z'_{\ell,e,t}(u; x) = (W_{\ell,e,t} + \Delta W_{\ell,e,t}(x))\, u. \tag{36}$$

Replacing the relevant projections in Equations 7–9 yields the adapted expert $\widehat{f}_{\ell,e}(x)$, and the redesigned routed output is

$$y'_\ell(x) = \sum_{e\in\mathcal{S}_\ell(x)} p_{\ell,e}(x)\widehat{f}_{\ell,e}(x), \tag{37}$$

with the backbone's shared-expert term added separately where applicable.

B Datasets, Evaluation, and Training Protocol

B.1 Downstream and General-Capability Evaluation

The mathematical-reasoning experiments fine-tune on MetaMathQA-R1 (huggingface oumi-ai/MetaMathQA-R1), a pre-constructed synthetic dataset whose problems are drawn from MetaMathQA (Yu et al., 2024) and whose responses are generated by DeepSeek-R1. We use only the

Setting	Math	Intent
Training data	MetaMathQA-R1	ESFT_Intent
Maximum samples	80,000	80,000
Epochs	1	3
Sequence cutoff	2,048	4,096
Packing	default	disabled
Evaluation set size	500	500
Evaluation metric	accuracy	exact match
Maximum new tokens	512	512
Evaluation batch size	32	8

Table 2: Task-specific training and downstream-evaluation settings.

final solution portion of each response for training, excluding the intermediate think content. Mathematical reasoning is evaluated on the 500 examples in MATH-500 (Hendrycks et al., 2021). The intent experiments fine-tune on ESFT_Intent and report exact-match accuracy on a held-out set of 500 examples. We assess general-capability retention on MMLU-Pro (Wang et al., 2024b) and IFEval (Zhou et al., 2023).

For each model–task–configuration in the four design studies, we train every candidate learning rate with three random seeds and select one rate by mean downstream score. We then evaluate the three checkpoints at that rate on MMLU-Pro and IFEval. Downstream and general-capability scores are averaged over the same three seeds; general scores do not affect learning-rate selection. Table 2 summarizes the task-specific training and downstream-evaluation settings.

B.2 Common Optimization Settings

All reported fine-tuning runs use four GPUs, BF16 computation, gradient checkpointing, and DeepSpeed ZeRO-2 (Rasley et al., 2020). Optimization uses fused AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$; the learning-rate schedule is cosine with a 0.1 warmup ratio. For every experiment, the learning rate is selected by grid search over candidates drawn from $\{10^{-5}, 10^{-4}, 10^{-3}\}$. No adapter dropout was configured. All backbones are loaded with remote model code enabled.

The per-device batch size is 6 with six accumulation steps on four GPUs, giving an effective batch size of $6 \times 6 \times 4 = 144$. Gemma-4 Intent uses a per-device batch size of 4 and nine accumulation steps, preserving the same effective batch size of $4 \times 9 \times 4 = 144$. GPT-OSS Intent explicitly disables FlashAttention.

C Experiment-Specific Settings

C.1 Adaptation Position

The position study uses one shared low-rank adaptation branch alongside the complete expert block when experts are adapted, standard LoRA for attention, and an optional LoRA branch on the native router. All target layers are enabled, pool and weight sharing are per layer, and the auxiliary coefficient is zero. The scale is $\alpha = 2r$ throughout. Each of the first three base configurations is evaluated both without and with router LoRA ($r = 8$, $\alpha = 16$); the higher-budget configuration is evaluated in the same two variants, yielding eight arms. Table 3 summarizes these adaptation-position configurations.

For Qwen3, the estimated trainable counts before adding the optional router branch are approximately 53.48M for attention $r = 64$, 50.33M for expert adaptation with $r = 256$, 51.90M for attention $r = 32$ plus expert adaptation with $r = 128$, and 103.81M for the higher-budget arm.

C.2 Adaptation Granularity

The final granularity experiment isolates the expert-adaptation design while retaining the router LoRA anchor. Attention is disabled. Every arm targets all MoE layers, uses per-layer pool/weight sharing, router LoRA $r = 8$, $\alpha = 16$, and zero auxiliary coefficient.

Base arm	Attention	Expert adaptation	Router variant
Attention only	$r64/a128$	–	off or $r8/a16$
Experts only	–	$r256/a512$	off or $r8/a16$
Attention+experts	$r32/a64$	$r128/a256$	off or $r8/a16$
Higher budget	$r64/a128$	$r256/a512$	off or $r8/a16$

Table 3: Adaptation-position arms. Expert adaptation uses one independently routed top-1 branch per layer ($N_L = 1$), shared by the complete expert block.

Granularity	Routing and placement	Expert setting
Module Level	Independent, $N_L = 1$, top-1	$r256/a512$
Expert Level	Follow native router; one branch per expert	$r2/a4$
Weight Level	Follow native router; gate/up/down weights	$r1/a2$
Weight Level	Follow native router; gate/up/down weights	$r2/a4$

Table 4: The four final granularity arms. Native routing uses 128 experts and top-8 for Qwen3/Gemma-4, and 32 experts and top-4 for GPT-OSS.

C.3 PEFT Method Comparison

The ten-method comparison uses the same experts+router, no-attention protocol as the final granularity study. For LoRA-structured methods, the router anchor is LoRA $r = 8$, $\alpha = 16$. For DoRA, OFT, FourierFT, and IA3, the router uses the same update family as the weight-level expert adaptation rather than a LoRA adapter. All methods target every MoE layer and use zero auxiliary loss. The exact method-specific settings are listed in Tables 5 and 6.

C.4 Auxiliary Load-Balancing Loss

The auxiliary study uses weight-level LoRA with $r = 1$, $\alpha = 2$, router LoRA $r = 8$, $\alpha = 16$, no attention, and the same learning-rate grid as the granularity study. For a batch-level routing frequency f_e and mean dense router probability $\bar{p}_e$, the per-layer balancing term has the form

$$\mathcal{L}_{\text{bal}} = E \sum_{e=1}^{E} f_e \bar{p}_e, \qquad \mathcal{L} = \mathcal{L}_{\text{task}} + \lambda \mathcal{L}_{\text{bal}}. \tag{38}$$

The schedule is constant and $\lambda \in \{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. The zero-coefficient arm reuses the corresponding no-attention granularity baseline. The downstream averages reported in the main paper are 68.5, 68.1, 66.8, 64.6, and 58.6 percent for increasing λ, respectively.

D Additional MoE^2-LoRA Validation of Placement and Balancing

In this section, we test the adaptation-position conclusion (Insight 1 in the main paper) and the auxiliary-load-balancing conclusion (Insight 4).

D.1 Experimental Controls

The experts-only $r32$ reference uses the reproduced MoE^2-LoRA configuration: two layer groups, a 128-entry rank-32 adaptation pool per group (256 entries in total), $\alpha = 64$, learned top-8 pool routing, a 128-dimensional routing bottleneck, and no attention or router LoRA. Its load-balancing coefficient is zero. The budget-matched placement arm reduces the expert-adaptation pool to $r = 18$,

Method	Expert update	Router update	Backbone-specific values
Weight-level LoRA	Expert gate/up/down weights, $r1/a2$	LoRA $r8/a16$	Same on all three backbones
Weight-level DoRA	Expert gate/up/down weights, $r1/a2$, plus magnitude	DoRA $r8/a16$	Same ranks/scales on all three backbones
Weight-level OFT	Blockwise Cayley transform	Blockwise OFT	Expert/router blocks: Qwen3 $4/128$; Gemma-4 $8/128$; GPT-OSS $6/32$
FourierFT	Fixed-support spectral delta	FourierFT	Expert/router frequencies: Qwen3 $4096/17408$; Gemma-4 $5632/23552$; GPT-OSS $5760/23296$; scale 150, seed 2024
IA3	Per-expert down-input scale	Router-logit output scale	One-initialized vectors; no rank

Table 5: Weight-level methods in the final PEFT method comparison. Attention is disabled, so attention-specific capacity values are intentionally omitted.

Method	Expert update	Router update	Backbone-specific values
Expert-level LoRA	One adaptation branch per routed expert, follows the native router's top-k, $r2/a4$	LoRA $r8/a16$	$E/k = 128/8$ for Qwen3 and Gemma-4; $32/4$ for GPT-OSS
MoE-LoRA	Independent adapter router, $N_L = 8$, adapter top-2	LoRA $r8/a16$	FFN $r32/a64$ on Qwen3/Gemma-4; $r8/a16$ on GPT-OSS
Module-level LoRA	One independent top-1 adaptation branch per layer	LoRA $r8/a16$	FFN $r256/a512$ on Qwen3/Gemma-4; $r64/a128$ on GPT-OSS
DAS-LoRA	Highest-DAS expert subset per layer	Enabled common LoRA router anchor	Select 8 experts with $r32/a64$ on Qwen3/Gemma-4; 4 experts with $r16/a32$ on GPT-OSS
Partial FT	Full expert weights in last decoder layer	Full router in last layer	No adapter and no parameter matching

Table 6: MoE-structured and partial-fine-tuning methods in the final PEFT method comparison. MoE^2-LoRA is excluded from this controlled ten-method grid and analyzed separately in Section A.1.

$\alpha = 36$, and adds standard rank-18, $\alpha = 36$ LoRA to the Q, K, V, and output projections in every attention layer. This changes the trainable count by only 0.76%, from 47.71M to 48.07M parameters.

The balancing arms keep the experts-only $r32$ architecture and trainable count fixed, leave attention and router adaptation disabled, and apply a constant $\lambda \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. The downstream-selected learning rate is 10^{-3} for Intent and 10^{-4} for Math. Intent is trained for three epochs and Math for one epoch; both use the common 80,000-sample training and downstream-selected checkpoint protocol. MMLU-Pro and IFEval are evaluated on those same checkpoints. Tables 7 and 8 report full-set results.

D.2 Evidence for the Two Design Defaults

Adaptation position. Starting from the experts-only reference, adding attention adaptation at an effectively matched parameter budget consistently weakens downstream performance: Intent decreases from 84.00 to 81.60 and MATH-500 from 60.00 to 56.60. For each task, the joint Atten-

Configuration	Trainable params	Intent	MMLU-Pro	IFEval
Experts only	47.71M	84.00	45.26	18.11
Attention+experts	48.07M	81.60	47.93	14.23
Balance 10^{-4}	47.71M	85.00	47.46	17.56
Balance 10^{-3}	47.71M	82.20	45.18	5.91
Balance 10^{-2}	47.71M	80.80	22.37	7.76
Balance 10^{-1}	47.71M	67.80	16.89	13.31

Table 7: Additional Qwen3 MoE^2-LoRA validation on Intent. All metrics are percentages. Intent uses 500 held-out examples, MMLU-Pro uses 12,032 examples, and IFEval is prompt-level accuracy on 541 prompts.

Configuration	Trainable params	MATH-500	MMLU-Pro	IFEval
Experts only	47.71M	60.00	48.19	35.12
Attention+experts	48.07M	56.60	47.56	36.23
Balance 10^{-4}	47.71M	56.40	47.27	40.67
Balance 10^{-3}	47.71M	55.80	46.69	43.81
Balance 10^{-2}	47.71M	55.60	44.83	45.47
Balance 10^{-1}	47.71M	53.40	42.50	46.58

Table 8: Additional Qwen3 MoE^2-LoRA validation on Math. All metrics are percentages.

tion+experts configuration also degrades at least one upstream benchmark. On Intent, IFEval falls from 18.11 to 14.23 despite the MMLU-Pro improvement; on Math, MMLU-Pro falls from 48.19 to 47.56 despite the IFEval improvement. Adding attention therefore fails to provide a consistent benefit and instead introduces simultaneous downstream and upstream degradation on both tasks. These results reinforce Insight 1: for MoE fine-tuning, target the experts by default and unfreeze shared attention modules cautiously, since adapting more modules does not necessarily yield better transfer and may amplify general-capability forgetting.

Auxiliary load balancing. On Intent, the smallest coefficient, 10^{-4}, remains competitive: it improves Intent from 84.00 to 85.00 and MMLU-Pro from 45.26 to 47.46, with only a 0.55-point decrease on IFEval. This local benefit does not transfer to Math, where the same coefficient reduces MATH-500 from 60.00 to 56.40 and MMLU-Pro from 48.19 to 47.27, although IFEval improves. The other nonzero coefficients are generally more damaging. They all lower downstream performance on both tasks; on Intent, the larger coefficients also cause substantial upstream degradation, while on Math the IFEval gains are consistently accompanied by losses on both MATH-500 and MMLU-Pro. Thus, auxiliary balancing does not yield a task-stable improvement. These results support Insight 4: the auxiliary loss should be disabled or assigned a very small coefficient during MoE adaptation. A coefficient around 10^{-4} is a safe choice when routing regularization is required and may benefit particular datasets, whereas larger values can substantially degrade downstream performance.

E Backbone Architectures

The experiments use the text-generation pathways of Qwen3-30B-A3B-Base (Qwen Team, 2025), Gemma-4-26B-A4B (Gemma Team, 2026), and GPT-OSS-20B (OpenAI, 2025). Gemma-4 is distributed through a multimodal wrapper, but no visual inputs or vision-tower adaptation are used. The architectural values below follow the official reports, model cards, and released checkpoint configurations. Tables 9 and 10 summarize the attention and MoE architectures, respectively.

These backbones are not minor variants of one MoE template. They differ in expert cardinality (32 versus 128), token-level fan-out (top-4 versus top-8), routed width (704, 768, or 2880), the presence of an always-active shared expert, grouped-KV geometry, and attention locality. In particular, Gemma-4 tests routed-expert adaptation in the presence of a frozen shared path, while Qwen3 and GPT-OSS expose no explicit shared-expert branch; Qwen3 uses full attention throughout, whereas

Backbone	Layers	Hidden	Q heads	KV heads	Head dim	Attention pattern
Qwen3-30B-A3B-Base	48	2048	32	4	128	Full attention; no sliding window
Gemma-4-26B-A4B	30	2816	16	8 local / 2 global	256 local / 512 global	Five sliding layers followed by one global layer; window 1024; global K=V and p-RoPE
GPT-OSS-20B	24	2880	64	8	64	Alternating sliding and full layers; window 128

Table 9: Text-backbone and attention differences. "Q heads" and "KV heads" distinguish query heads from grouped key/value heads. Gemma-4 changes both KV-head count and head dimension in its global-attention layers and ties their key and value projections.

Backbone	Routed experts	Top-k	Shared/always-on expert path	Routed width	Expert adaptation target
Qwen3-30B-A3B-Base	128	8	No explicit shared expert	768	Routed `experts` host
Gemma-4-26B-A4B	128	8	One always-active shared expert; dense width 2112 in config	704	The 128 routed `experts`; shared path frozen
GPT-OSS-20B	32	4	No explicit shared expert	2880	Routed `experts` host

Table 10: MoE structure and the actual expert-adaptation boundary. "No explicit shared expert" means that no such branch is defined in the released checkpoint configuration or reference implementation; it is not a claim about all models in that family. The Gemma-4 shared path is always active, while the repository adapts only its separate routed-expert container.

the other two mix local and global attention with different schedules. Against this architectural variation, the recurring trends behind the main paper's placement, granularity, PEFT method, and balancing insights are observed across the three-backbone grid, even though individual endpoints need not share an identical ordering. This makes the findings less likely to be an artifact of one expert count, router fan-out, or attention topology, while remaining empirical rather than universal claims.

F Functional Geometry of Adaptation Granularity

This appendix gives the response definitions, rank derivations, measurement protocol, and detailed numerical results for the functional analysis of adaptation granularity. The module-level rank constrains one shared branch, the expert-level rank constrains each routed branch, and the weight-level rank constrains individual expert projections. We compare these configurations through the output of the full MoE computation, including nonlinear expert features and routed aggregation.

F.1 Layer-Level Response and Functional Effective Rank

For MoE layer ℓ, let $F_{0,\ell}(x)$ denote the frozen base-model block output and $F_{a,\ell}(x)$ the output after loading adapter configuration a. The adapter-induced functional perturbation is

$$\delta F_{a,\ell}(x) = F_{a,\ell}(x) - F_{0,\ell}(x). \tag{39}$$

For a probe set $\mathcal{X}_\ell = \{x_1, \ldots, x_n\}$, we collect these perturbations as columns,

$$Y_{a,\ell} = [\delta F_{a,\ell}(x_1) \quad \cdots \quad \delta F_{a,\ell}(x_n)] \in \mathbb{R}^{d \times n}. \tag{40}$$

Unlike the rank of an individual LoRA factor or weight matrix, $Y_{a,\ell}$ includes the effect of pretrained expert computation, routing weights, and aggregation across selected experts.

We define the uncentered second-moment matrix

$$K_{a,\ell} = \frac{1}{n} Y_{a,\ell} Y_{a,\ell}^{\top}. \tag{41}$$

If its eigenvalues are $\lambda_1, \ldots, \lambda_d \geq 0$, the layer-level functional effective rank is the participation ratio

$$r_{\text{eff}}(Y_{a,\ell}) = \frac{\big(\operatorname{tr} K_{a,\ell}\big)^2}{\|K_{a,\ell}\|_F^2 + \varepsilon} = \frac{\big(\sum_i \lambda_i\big)^2}{\sum_i \lambda_i^2 + \varepsilon}, \tag{42}$$

where ε is used only for numerical protection. For a nonzero response in the $\varepsilon \to 0$ limit, concentrating all energy in one direction gives $r_{\text{eff}} = 1$, whereas equal energy in k orthogonal directions gives $r_{\text{eff}} = k$. More generally,

$$r_{\text{eff}}(Y_{a,\ell}) \leq \operatorname{rank}(Y_{a,\ell}) \leq \min(d, n). \tag{43}$$

The measure is invariant to a common nonzero rescaling of the response, up to the numerical stabilizer. It therefore describes how response energy is distributed across directions, rather than the perturbation magnitude. A larger value indicates a higher-dimensional realized response on the probe set. Table 11 reports this geometric quantity alongside the measured downstream and retention outcomes.

F.2 Why Local Rank Does Not Determine Layer-Level Rank

Module- and expert-level adaptation. A module-level adaptation produces the linear correction

$$\delta F_M(x) = Cx, \qquad \operatorname{rank}(C) \leq r_M. \tag{44}$$

Writing the probe inputs as $X = [x_1, \ldots, x_n]$ gives

$$\operatorname{rank}(Y_M) = \operatorname{rank}(CX) \leq \min\big(r_M, \operatorname{rank}(X)\big). \tag{45}$$

Thus, $r_M = 256$ is an upper bound for the single shared map, not its realized effective rank. A concentrated learned spectrum or a restricted probe span can yield a substantially smaller value. Moreover, this local bound cannot be converted directly into the rank of a configuration distributed across many experts and projections.

For expert-level adaptation, the correction from expert e is

$$\delta f_{E,e}(x) = C_e x, \qquad \operatorname{rank}(C_e) \leq r_E. \tag{46}$$

Multiplication by the input-dependent routing weight $p_e(x)$ does not move a branch response outside $\operatorname{col}(C_e)$. The complete layer, however, aggregates several selected experts whose column spaces may differ; its effective rank can therefore exceed r_E. The value $r_E = 2$ constrains each branch, not the routed layer response.

Weight-level adaptation. Write one gated expert as

$$f_e(x) = D_e\big[\sigma(G_e x) \odot U_e x\big], \tag{47}$$

where G_e, U_e, and D_e are the gate, up, and down projections. Consider only a rank-one update to the up projection, $\Delta U_e = ab^{\top}$. Its exact output correction is

$$\delta_U f_e(x) = (b^{\top} x) D_e \operatorname{Diag}(a) \sigma(G_e x). \tag{48}$$

For n inputs, define

$$S_{\mathcal{X}} = \begin{bmatrix} \sigma(G_e x_1) & \cdots & \sigma(G_e x_n) \end{bmatrix}, \qquad \zeta_j = b^{\top} x_j. \tag{49}$$

The resulting sample-response matrix is

$$\begin{aligned} Y_U &= D_e \operatorname{Diag}(a) S_{\mathcal{X}} \operatorname{Diag}(\zeta), \\ \operatorname{rank}(Y_U) &\leq \min(d, m, n), \end{aligned} \tag{50}$$

where m is the expert intermediate width. When $D_e \operatorname{Diag}(a) S_{\mathcal{X}}$ attains the corresponding rank and all ζ_j are nonzero, this upper bound is attainable. Hence, even a rank-one update to a single expert projection can reuse the frozen gate's input-dependent nonlinear features to produce a multi-directional sample response. The measured weight-level configurations update gate, up, and down projections across all 128 experts and then aggregate routed responses. Equation 50 establishes the rank-expansion mechanism for an individual up-projection update. The full measurements include all adapted projections and routed aggregation.

Granularity	Local r	Math	Intent	Mean	Down. avg.	General-capability observation
Module Level	256	25.25	20.54	22.90	68.4	MMLU-Pro -0.5; IFEval -3.7
Expert Level	2	16.31	15.91	16.11	64.7	MMLU-Pro -5.0; IFEval -4.6; unstable across backbones
Weight Level	1	63.90	69.08	66.49	68.5	MMLU-Pro $+0.8$; IFEval -1.4; best retention
Weight Level	2	**86.37**	**85.22**	**85.80**	**69.1**	MMLU-Pro -0.9; IFEval -3.4; nearly twice the trainable parameters

Table 11: Functional effective rank and the corresponding granularity results. Math, Intent, and Mean report Qwen3 layer-level effective ranks; Mean averages the 96 task–layer observations. Downstream and general-capability results are the main-paper aggregates over three backbones and two task endpoints. The local r values constrain different objects and are included only to identify each configuration.

F.3 Measurement Protocol

We measure the four Qwen3 granularity configurations selected for the Math and Intent experiments. The protocol is as follows.

- We scan all 48 MoE layers and all 128 routed experts, without expert subsampling.
- Each layer uses $n = 384$ deterministic synthetic probes, $x \sim \mathcal{N}\big(0, \mathrm{diag}(\gamma_\ell^2)\big)$, where γ_ℓ is the pretrained pre-MoE RMSNorm gain for that layer.
- We estimate a checkpoint-specific expert-selection distribution from the recorded training-time activation counts. For each synthetic token, we sample eight experts without replacement and assign each selected expert an aggregation weight of $1/8$.
- The expert-level and two weight-level configurations share the same synthetic routing realization. The module-level adaptation is applied once per token and does not pass through expert routing.
- We evaluate Equation 42 for each task–configuration–layer combination, giving $2 \times 4 \times 48 = 384$ layer-level observations.

Math and Intent identify the trained adapters and their recorded routing statistics. All reported functional ranks characterize these checkpoints on the RMSNorm-scaled synthetic probe distribution specified above.

F.4 Functional Effective Rank Across Granularities

Table 11 reports the checkpoint-specific means and their average over 96 task–layer observations. Each layer uses 384 probes, so the response matrix has rank at most 384. The reported participation ratios further account for the concentration of energy within that response span.

F.5 Configuration-Level Comparisons

Within weight-level adaptation. Holding the expert and projection set fixed, increasing r from one to two raises the mean effective rank from 66.49 to 85.80, a 29.0% increase. The task-specific increases are 22.47 on Math (35.2%) and 16.14 on Intent (23.4%). The corresponding three-backbone downstream average rises from 68.5 to 69.1, while rank one uses approximately half the trainable parameters and retains more general capability. These measurements identify the additional functional coverage supplied by a larger local rank and its associated task, parameter, and retention costs.

Across adaptation positions. The expert-level responses have mean effective ranks near 16 on both checkpoints. Their fixed branch column spaces are combined through routing, as characterized

in Appendix F.2. The shared module-level branch reaches 25.25 on Math and 20.54 on Intent, showing different energy concentration under the same local rank of 256. Weight-level rank one reaches 63.90 and 69.08 through the nonlinear mechanism in Equation 50.

Interpreting the reported quantities. Functional ranks are measured on the selected Qwen3 checkpoints, while the downstream and retention columns summarize the three-backbone study. Effective rank characterizes response diversity; MMLU-Pro and IFEval directly measure general-capability retention. Together with the expert-local spectral measurements below, these quantities describe complementary aspects of the adaptation trade-off.

G Pretrained Spectral Consistency Across Adaptation Granularities

This appendix provides the expert-local spectral definitions, the mechanism of spectral inheritance, and the full measurement protocol and numerical profiles. We express each expert-output perturbation in the left singular basis of its corresponding pretrained down projection and compare its cumulative energy distribution with that projection's spectrum.

G.1 Spectral Profile and Consistency Score

Let the pretrained down projection of expert e be $D_e \in \mathbb{R}^{d\times m}$, with singular value decomposition

$$\begin{aligned} D_e &= U_e \Sigma_e V_e^\top \\ &= \sum_{i=1}^{m} \sigma_{e,i} u_{e,i} v_{e,i}^\top. \end{aligned} \tag{51}$$

The left singular vectors $u_{e,i}$ form the pretrained expert's output coordinates, with $\sigma_{e,1} \geq \cdots \geq \sigma_{e,m} \geq 0$, ordered from spectrally strong to weak directions. For adapter configuration a and probes $\mathcal{X} = \{x_1, \ldots, x_n\}$, define the expert-output perturbation matrix

$$\begin{aligned} Y_{a,e} &= \left[\delta f_{a,e}(x_1) \quad \cdots \quad \delta f_{a,e}(x_n)\right], \\ \delta f_{a,e}(x) &= f_{a,e}(x) - f_{0,e}(x). \end{aligned} \tag{52}$$

For weight-level adaptation, this is the change induced jointly by the gate, up, and down LoRA updates. For expert-level adaptation, it is the output of the new expert-local branch. Measuring experts separately ensures that every response is compared with its corresponding pretrained D_e.

The cumulative fraction of perturbation energy carried by the first k pretrained output directions is

$$S_k(Y_{a,e}) = \frac{\sum_{i\leq k} \|u_{e,i}^\top Y_{a,e}\|_2^2}{\sum_{i\leq m} \|u_{e,i}^\top Y_{a,e}\|_2^2 + \varepsilon}. \tag{53}$$

We compare this profile with the down projection's own cumulative spectrum,

$$S_{k,e}^0 = \frac{\sum_{i\leq k} \sigma_{e,i}^2}{\sum_{i\leq m} \sigma_{e,i}^2}, \qquad S_k^{\mathrm{uni}} = \frac{k}{m}, \tag{54}$$

where S_k^{uni} is the reference for energy distributed uniformly over the m pretrained directions. A profile close to $S_{k,e}^0$ follows the pretrained strong-to-weak hierarchy, whereas a profile close to k/m is approximately uniform in the same singular coordinates.

To summarize the three measured cutoffs $\mathcal{K} = \{16, 64, 256\}$, we define the pretrained spectral-consistency score

$$C_{\mathrm{spec}}(Y_{a,e}) = 1 - \frac{\sum_{k\in\mathcal{K}} \left| S_k(Y_{a,e}) - S_{k,e}^0 \right|}{\sum_{k\in\mathcal{K}} \left| k/m - S_{k,e}^0 \right| + \varepsilon}. \tag{55}$$

This normalization assigns $C_{\mathrm{spec}} = 1$ to an exact match with the pretrained cumulative profile and $C_{\mathrm{spec}} = 0$ to the uniform reference; negative values indicate a larger aggregate deviation from the

Checkpoint	Configuration	S_{16}	S_{64}	S_{256}	C_{spec}
Reference	Pretrained down spectrum	0.09368	0.25758	0.64587	1.000
Reference	Uniform directions	0.02083	0.08333	0.33333	0.000
Math	Weight Level, $r = 1$	0.08921	0.23353	0.58366	**0.812**
Math	Weight Level, $r = 2$	0.08950	0.23470	0.58642	**0.822**
Math	Expert Level, $r = 2$	0.03762	0.11561	0.37202	0.141
Intent	Weight Level, $r = 1$	0.08308	0.21957	0.55615	**0.733**
Intent	Weight Level, $r = 2$	0.08338	0.22072	0.55822	**0.741**
Intent	Expert Level, $r = 2$	0.03151	0.10275	0.35596	0.087

Table 12: Mean cumulative spectral profiles and pretrained spectral-consistency scores. Each adapted row averages 6,144 layer–expert positions. The reference rows average the corresponding pretrained expert spectra and use $m = 768$ for the uniform profile.

pretrained profile than the uniform reference. Because common rescaling of $Y_{a,e}$ leaves the score unchanged, it isolates the relative spectral organization of the realized perturbation.

The pretrained expert computation provides a direct mechanism for spectral inheritance. For a perturbation component that passes through the frozen down projection, $Y^{\|}_{a,e} = D_e H_{a,e}$, Equation 51 gives

$$S_k(Y^{\|}_{a,e}) = \frac{\sum_{i \le k} \sigma^2_{e,i} \|v^\top_{e,i} H_{a,e}\|_2^2}{\sum_{i \le m} \sigma^2_{e,i} \|v^\top_{e,i} H_{a,e}\|_2^2 + \varepsilon}. \tag{56}$$

Thus, when the latent perturbation energy is broadly distributed in the right singular coordinates, the pretrained singular values shape the output-energy profile. Weight-level gate/up updates traverse this frozen projection and can inherit its hierarchy. By contrast, expert-level adaptation produces its output through a newly learned map, without the same architectural coupling to the expert's pretrained down spectrum. The measured score tests the resulting placement-dependent distinction directly.

G.2 Measurement Protocol

We evaluate the Qwen3 weight-level $r = 1$, weight-level $r = 2$, and expert-level $r = 2$ checkpoints selected for the Math and Intent experiments. The protocol is as follows.

- We scan all 48 MoE layers, $\ell \in \{0, \dots, 47\}$, and all 128 routed experts in each layer, giving 6,144 layer–expert positions per checkpoint and configuration.
- Each layer uses $n = 768$ deterministic Gaussian probes scaled by the pretrained pre-MoE RMSNorm gain.
- We compute the expert-output perturbation before cross-expert aggregation and express it in the left singular basis of the corresponding pretrained down projection.
- For Qwen3, $m = 768$; we record S_{16}, S_{64}, and S_{256} and evaluate Equation 55 at every layer–expert position before averaging the scores.

The diagnostic is expert-indexed. The module-level configuration produces one shared post-MoE response rather than an expert-specific response, and its geometry is therefore characterized by the layer-level analysis in Section F.

G.3 Spectral Consistency Across Granularities

Table 12 reports the three cumulative energy cutoffs and the mean consistency score. Scores are computed separately for each layer–expert position before averaging, preserving each expert's own pretrained reference. Weight-level rank one exceeds expert-level adaptation by 0.671 on Math and 0.646 on Intent in mean C_{spec}. The layer-mean difference is positive in all 48 layers on both checkpoints; paired differences are positive at 99.8% of Math positions and 96.3% of Intent positions. The two weight-level ranks produce closely matched cumulative profiles at all three cutoffs.

Configuration	C_{spec} (M / I)	Down. avg.	MMLU Δ	IFEval Δ
Weight Level $r = 1$	**0.812 / 0.733**	68.5	+0.8	−1.4
Weight Level $r = 2$	**0.822 / 0.741**	**69.1**	−0.9	−3.4
Expert Level $r = 2$	0.141 / 0.087	64.7	−5.0	−4.6

Table 13: Pretrained spectral consistency alongside the main granularity results. M/I denotes Math-/Intent, and MMLU denotes MMLU-Pro. Spectral scores are measured on Qwen3; downstream and general-capability values are the main-paper aggregates over three backbones and two task endpoints.

G.4 Alignment with the Main Granularity Result

Table 13 places the Qwen3 spectral measurements beside the three-backbone granularity results. Both weight-level configurations combine high spectral consistency with better downstream accuracy and retention than the expert-level configuration. Within weight-level adaptation, rank two increases functional response diversity while rank one gives the best retention and near-best downstream accuracy. The detailed measurements thus connect update placement to both response diversity and the reuse of pretrained spectral structure.

G.5 Energy in the Pretrained Output Subspace

Weight-level adaptation also concentrates more perturbation energy in the pretrained expert's output subspace. To quantify this concentration, we measure the fraction of the expert-output perturbation carried by $\mathrm{col}(D_e)$:

$$\eta_{\text{col}}(Y_{a,e}) = \frac{\|U_e^\top Y_{a,e}\|_F^2}{\|Y_{a,e}\|_F^2 + \varepsilon}. \tag{57}$$

Here U_e contains the $m = 768$ left singular directions of the pretrained down projection. We measure this share on RMSNorm-scaled Gaussian probes across all 48 layers and 128 experts for each Math and Intent checkpoint. The projection energy is estimated using $D_e(D_e^\top D_e + \lambda_e I)^{-1} D_e^\top$, with $\lambda_e = 10^{-6}\,\mathrm{tr}(D_e^\top D_e)/m$. An isotropic output perturbation has expected share $m/d = 768/2048 = 0.375$.

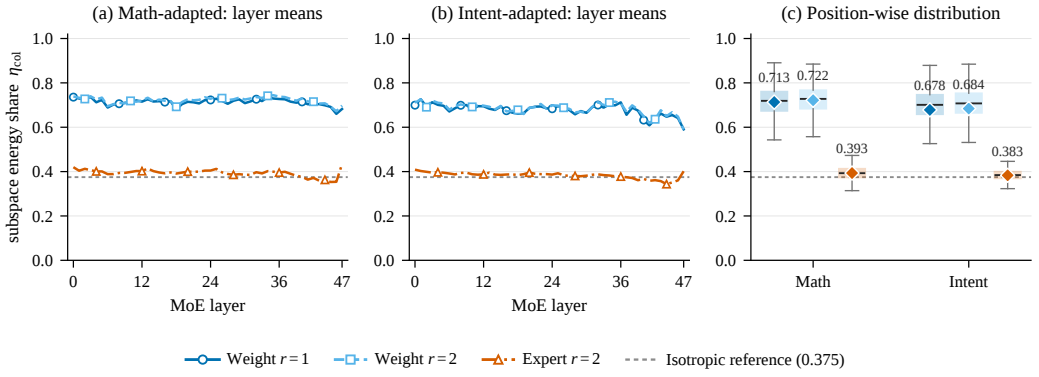


Figure 9: Perturbation energy in the pretrained expert output subspace on Qwen3 synthetic probes. The left and middle panels show layer means over 128 experts; the right panel shows distributions over 6,144 layer–expert positions per configuration. Boxes show medians and interquartile ranges, whiskers extend to the most extreme observations within 1.5 interquartile ranges, and diamonds mark means. The dashed line is the isotropic reference of 0.375. Weight-level updates concentrate more energy in pretrained output subspaces across both checkpoints.

Figure 9 shows mean η_{col} values of 0.713 and 0.722 for weight-level ranks one and two on Math, and 0.678 and 0.684 on Intent. Expert-level rank two reaches 0.393 and 0.383, respectively. Rank-one

weight updates increase the in-subspace energy share by 32.0 percentage points on Math and 29.5 points on Intent, with a positive difference in all 48 layer means on both checkpoints. The rank-one and rank-two results show that this concentration is already established at rank one. Together, η_{col} and C_{spec} characterize two complementary aspects of pretrained structure reuse: the energy retained in the expert's output subspace and its distribution along the pretrained spectral hierarchy.

H Selected Learning Rates and Training Throughput on Qwen3

Table 14 complements the four design studies with the selected learning rates and training throughput on Qwen3-30B-A3B-Base. Each configuration uses the learning rate selected by mean downstream performance across three seeds from candidates drawn from $\{10^{-5}, 10^{-4}, 10^{-3}\}$; the same rate is used for all three runs. Throughput is measured under the four-GPU training protocol in Appendix B, with an effective batch size of 144. Samples/s measures training examples processed per second, and steps/s measures optimizer updates per second.

The throughput results support the practical appeal of expert-weight LoRA. At the selected rates of 10^{-3} for Intent and 10^{-4} for Math, weight-level LoRA processes 28.27 and 19.97 samples/s, respectively, giving $1.31\times$ and $1.45\times$ the throughput of DoRA in these settings. It also achieves $1.53\times$ and $1.41\times$ the throughput of expert-level LoRA on the two tasks. These comparisons support standard expert-weight LoRA as an efficient training recipe for both Qwen3 tasks.

	Intent			Math		
Configuration	LR	Samples/s	Steps/s	LR	Samples/s	Steps/s
Adaptation position						
A_{64}	10^{-3}	72.034 ± 3.596	0.504 ± 0.025	10^{-5}	30.903 ± 0.298	0.215 ± 0.002
E_{256}	10^{-4}	44.719 ± 1.206	0.313 ± 0.008	10^{-5}	29.615 ± 0.124	0.206 ± 0.001
$A_{32} + E_{128}$	10^{-4}	38.396 ± 1.004	0.269 ± 0.007	10^{-5}	27.960 ± 0.317	0.194 ± 0.002
$A_{64} + R$	10^{-4}	70.154 ± 1.695	0.491 ± 0.012	10^{-5}	30.835 ± 0.061	0.214 ± 0.001
$E_{256} + R$	10^{-4}	43.071 ± 1.033	0.302 ± 0.007	10^{-5}	29.129 ± 0.508	0.202 ± 0.003
$A_{32} + E_{128} + R$	10^{-4}	38.363 ± 0.822	0.269 ± 0.006	10^{-5}	27.360 ± 0.102	0.190 ± 0.001
$A_{64} + E_{256}$	10^{-4}	38.008 ± 0.723	0.266 ± 0.005	10^{-5}	27.951 ± 0.361	0.194 ± 0.002
$A_{64} + E_{256} + R$	10^{-4}	37.875 ± 0.893	0.265 ± 0.006	10^{-5}	27.189 ± 0.108	0.189 ± 0.001
Adaptation granularity						
Module Level ($r = 256$)	10^{-4}	43.664 ± 1.077	0.306 ± 0.008	10^{-5}	29.242 ± 0.683	0.203 ± 0.005
Expert Level ($r = 2$)	10^{-3}	18.471 ± 0.243	0.129 ± 0.002	10^{-4}	14.123 ± 0.708	0.098 ± 0.005
Weight Level ($r = 1$)	10^{-3}	28.266 ± 1.027	0.198 ± 0.007	10^{-4}	19.969 ± 0.070	0.139 ± 0.001
Weight Level ($r = 2$)	10^{-3}	28.986 ± 0.543	0.203 ± 0.004	10^{-4}	19.811 ± 0.138	0.138 ± 0.001
PEFT method						
Weight-level LoRA	10^{-3}	28.266 ± 1.027	0.198 ± 0.007	10^{-4}	19.969 ± 0.070	0.139 ± 0.001
DoRA	10^{-3}	21.630 ± 0.995	0.152 ± 0.007	10^{-4}	13.736 ± 0.060	0.096 ± 0.001
OFT	10^{-3}	20.519 ± 0.270	0.144 ± 0.002	10^{-3}	3.690 ± 0.011	0.026 ± 0.000
FourierFT	10^{-3}	8.490 ± 0.019	0.060 ± 0.001	10^{-3}	7.250 ± 0.015	0.050 ± 0.000
IA3	10^{-3}	62.046 ± 0.887	0.435 ± 0.006	10^{-3}	28.988 ± 0.137	0.201 ± 0.001
Expert-level LoRA	10^{-3}	18.471 ± 0.243	0.129 ± 0.002	10^{-4}	14.123 ± 0.708	0.098 ± 0.005
MoE-LoRA	10^{-3}	45.963 ± 0.405	0.322 ± 0.003	10^{-5}	30.232 ± 0.298	0.210 ± 0.002
Module-level LoRA	10^{-4}	43.664 ± 1.077	0.306 ± 0.008	10^{-5}	29.242 ± 0.683	0.203 ± 0.005
DAS-LoRA	10^{-3}	26.508 ± 0.604	0.186 ± 0.004	10^{-4}	19.298 ± 0.122	0.134 ± 0.001
Partial FT (last layer)	10^{-4}	23.357 ± 1.326	0.164 ± 0.009	10^{-3}	19.746 ± 0.480	0.137 ± 0.004
Auxiliary load-balancing loss						
$\lambda = 0$	10^{-3}	28.266 ± 1.027	0.198 ± 0.007	10^{-4}	19.969 ± 0.070	0.139 ± 0.001
$\lambda = 10^{-4}$	10^{-3}	25.090 ± 0.752	0.176 ± 0.005	10^{-4}	20.116 ± 0.061	0.140 ± 0.001
$\lambda = 10^{-3}$	10^{-3}	24.806 ± 0.355	0.174 ± 0.003	10^{-4}	20.072 ± 0.142	0.139 ± 0.001
$\lambda = 10^{-2}$	10^{-3}	24.216 ± 0.679	0.170 ± 0.005	10^{-4}	19.773 ± 0.695	0.138 ± 0.005
$\lambda = 10^{-1}$	10^{-3}	24.337 ± 0.288	0.170 ± 0.002	10^{-3}	19.703 ± 0.321	0.137 ± 0.002

Table 14: Selected learning rates and training throughput on Qwen3-30B-A3B-Base. Throughput is reported as mean $\pm$ sample standard deviation across the runs used for each reported configuration. In the position block, A_r and E_r denote attention and expert-block adaptation at rank r; R adds router adaptation.