From Experts to Sub-experts: Fine-grained Parameter-Efficient Fine-Tuning for MoE LLMs

Zhentao Tan[1*] **Chang Liu**[1,2*] **Yao Liu**[1] **Yue Wu**[1†] **Jieping Ye**[1†]
[1]Alibaba Token Hub, Alibaba Group
[2]Southeast University
matthew.wy@alibaba-inc.com
yejieping.ye@alibaba-inc.com

Abstract

As large language models (LLMs) scale rapidly, dense full-parameter adaptation becomes increasingly expensive, motivating sparse and modular architectures such as Mixture-of-Experts (MoE) models. This shift raises a key question for parameter-efficient fine-tuning (PEFT): at what granularity should parameters be selected and updated? Existing PEFT methods such as LoRA operate on predefined weight matrices, while expert-level sparse tuning methods update entire selected experts. However, we observe that activated experts are internally sparse, with only a small fraction of intermediate channels strongly responding to downstream tasks, indicating that expert-level adaptation is still too coarse. We propose **NSFT** (*Neural Sub-expert Fine-Tuning*), a fine-grained PEFT framework that refines MoE adaptation from experts to sub-experts. NSFT decomposes each expert along the intermediate dimension into structured channel groups and selects task-relevant sub-experts by combining routing importance with intra-expert activation saliency. To optimize sparse partial updates, NSFT further introduces learning-rate scaling and dynamic gradient scaling to compensate for the reduced effective update magnitude. Experiments on OLMoE and Ling-mini-2.0 across challenging domain-specific tasks and general benchmarks show that NSFT consistently outperforms representative PEFT and expert-level sparse tuning baselines, while using substantially fewer trainable parameters and preserving competitive general capability. These results suggest that sub-expert-level adaptation is a more precise and efficient PEFT paradigm for MoE LLMs. The code is publicly available at `https://github.com/aheadformore/NSFT`.

1 Introduction

As large language models (LLMs) scale into the hundreds of billions—and increasingly, trillions—of parameters, the conventional paradigm of dense, full-parameter training faces mounting computational and memory bottlenecks [2, 13]. This pressure has catalyzed a fundamental shift from dense parameterization toward sparse, modular architectures [21, 26, 35, 43]. Along with this architectural transition, parameter-efficient adaptation raises a crucial question: *at what granularity should model parameters be selected and updated for effective downstream learning?* The choice of adaptation granularity not only affects parameter efficiency and training cost, but also relates to broader questions about modularity in Transformers [12], sparsity in knowledge representation [17, 29], and the intrinsic redundancy of model parameters [5, 10].

*These authors contributed equally to this work.
†Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

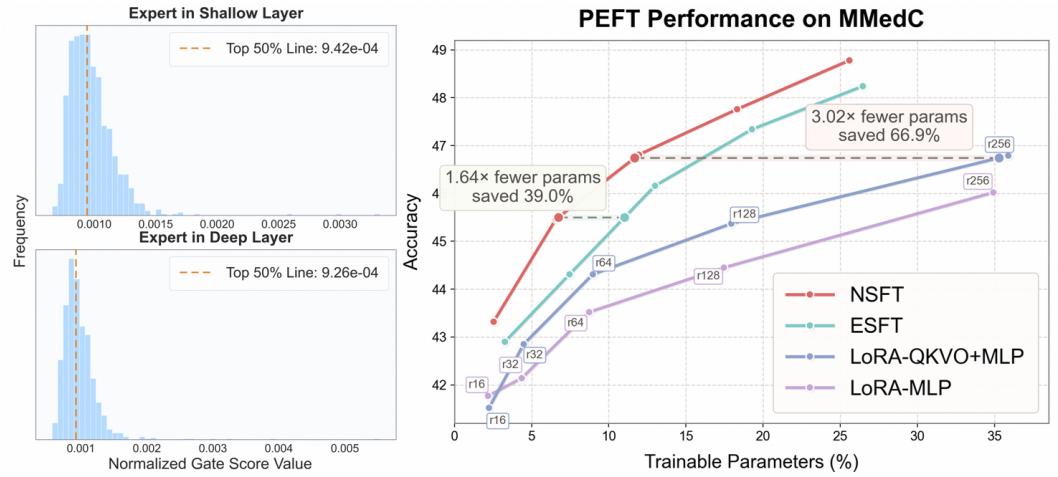

Figure 1: Left: channel-level gate-score distributions of shallow and deep OLMoE experts on MMedC, showing strong intra-expert sparsity. The dashed line marks the top-50% cumulative activation threshold. Right: PEFT performance on MMedC under different trainable-parameter budgets, where NSFT achieves a better accuracy–parameter trade-off than ESFT and LoRA variants.

Historically, adaptation granularity has largely followed architectural topology. In dense LLMs, parameter-efficient fine-tuning (PEFT) methods such as LoRA [17] typically operate on linear weight matrices, approximating task-specific updates with low-rank components. However, such matrix-level adaptation may spread gradients across broadly coupled neurons, making it difficult to precisely target the functional components most relevant to downstream tasks. Mixture-of-Experts (MoE) architectures introduce routed computation paths [9], enabling a coarser form of sparsity at the expert level. Recent methods such as ESFT [39] therefore update only selected experts for modular adaptation. Nevertheless, expert-level selection remains too coarse. As shown in Figure 1 (left), channel activations inside activated experts are highly concentrated near zero, indicating that only a small fraction of internal channels are strongly involved in the target task (refer to the Appendix E for more details). Updating the whole expert therefore introduces substantial redundancy and limits the parameter efficiency of expert-level tuning.

To address this limitation, we propose **NSFT** (*Neural Sub-expert Fine-Tuning*), a fine-grained PEFT method that moves sparse adaptation below the expert level. As illustrated in Figure 2, NSFT decomposes each expert along its intermediate dimension into structured sub-experts and selects the most task-relevant ones for fine-tuning. Inspired by MoNE [5], our method treats sub-experts as compact functional units rather than updating entire experts. By focusing updates on responsive internal channel groups, NSFT constructs sparse and precise adaptation pathways while avoiding redundant updates to weakly activated expert parameters.

However, fine-grained sub-expert tuning also introduces a new optimization challenge. Since only a small subset of channels inside each expert is updated, the effective update magnitude becomes much smaller than that of full-expert or expert-level tuning. If the original optimization settings are used directly, the selected sub-experts may be under-updated, limiting adaptation effectiveness. To address this issue, we further introduce an adaptive training strategy with two complementary components: learning-rate scaling, which compensates for the reduced global update step under sparse partial updates, and gradient scaling, which redistributes update strength among selected sub-experts according to their relative importance.

We evaluate NSFT on two MoE backbones, **OLMoE** [31] and **Ling-mini-2.0** [20], across a wide range of challenging domain-specific adaptation tasks, including multilingual medicine [33], scientific information extraction [37], domain-specific RAG [30], mathematical reasoning [36], code generation [32], and table question answering [40]. These tasks cover diverse cognitive demands, such as factual recall, logical reasoning, instruction following, and structured output synthesis, providing a comprehensive testbed for evaluating parameter-efficient adaptation.

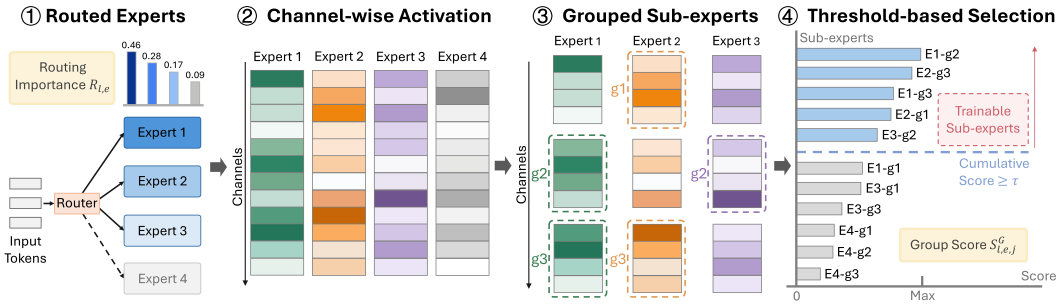

Figure 2: Illustration of sub-expert selection. Routed experts are first identified, and channel-wise activations are computed within each expert. With group size $G = 4$, consecutive channels are partitioned into sub-expert groups, where $E1$-$g1$ denotes the first group of Expert 1. All groups are then scored and ranked, and the groups selected under the threshold τ are used as trainable sub-experts for fine-tuning.

Our results show that conventional PEFT and sparse tuning methods still face clear efficiency–performance limitations. As shown in Figure 1 (right), NSFT achieves a better accuracy–parameter trade-off on MMedC than ESFT and LoRA variants, reaching higher accuracy with substantially fewer trainable parameters. In particular, NSFT saves up to **66.9%** trainable parameters compared with LoRA-QKVO+MLP under comparable or stronger performance, and requires fewer parameters than ESFT to reach the same accuracy level. Across OLMoE and Ling-mini, NSFT consistently improves domain adaptation over ESFT and LoRA while preserving competitive general capability, demonstrating that fine-grained sub-expert selection provides a more effective use of the trainable parameter budget.

Our contributions are threefold:

- Conceptual: We revisit the adaptation granularity of MoE models and show that expert-level sparse tuning remains too coarse for domain-specific fine-tuning. By shifting the update unit from entire experts to sub-experts, we provide a finer-grained perspective on how task-relevant knowledge is localized and adapted within MoE architectures.
- Methodological: We design and implement the Sub-experts framework, enabling fine-grained, structured sparsity through principled decomposition of expert modules, thereby advancing the frontier of efficient adaptation in MoE models.
- Empirical: We demonstrate across diverse real-world domains and two MoE backbones that NSFT consistently outperforms representative PEFT and expert-level sparse tuning baselines, while requiring substantially fewer trainable parameters. These results highlight the effectiveness of fine-grained sub-expert adaptation as a more parameter-efficient and performance-competitive approach for customizing large-scale MoE LLMs.

2 Related Works

Adaptation Granularity in Dense LLMs. The pursuit of efficient LLM adaptation has shifted attention from full-parameter updates to the choice of trainable granularity within model architectures. In dense models, existing methods mainly define update units from three perspectives. First, **externally added modules**, such as Adapters [16, 28, 38] and Soft Prompting [25, 44], introduce learnable auxiliary components while keeping the backbone frozen. Second, reparameterization-based methods operate on **individual linear layers**; LoRA [17], DoRA [29], and their successors [3, 41] approximate weight updates with low-rank decompositions. Third, selective fine-tuning methods directly update subsets of existing parameters [14, 19, 42]. Despite their differences, most dense-model PEFT methods remain closely tied to linear transformations, treating Transformers more as collections of tunable operators than as modular functional units.

Expert-Level Adaptation in Sparse Architectures. MoE architectures introduce structural sparsity through routing and expert modules, naturally enabling coarser-grained adaptation at the expert level [7, 26, 27, 35]. In this setting, the **expert** becomes an intuitive update unit. Expert-Specialized

Fine-Tuning (ESFT) [39] updates only experts that strongly respond to target data. DES-MoE [24] further adjusts expert selection through a multi-stage process for multi-domain adaptation, while DR-LoRA [8] uses non-uniform expert activation to assign different LoRA ranks within experts. These studies show that expert-level adaptation can improve efficiency for sparse MoE models, but they still treat each selected expert as an indivisible unit.

From Full Experts to Sub-Expert Units. Recent evidence suggests that experts themselves remain internally redundant. Mixture of Neuron Experts (MoNE) [5] shows that experts can be decomposed into neuron-level grains, where activating only high-response neurons can preserve inference performance. However, directly tuning at the neuron level can be overly fragmented, leading to unstable optimization and weak update capacity. We therefore introduce *sub-experts* as fine-grained but structured update units. By grouping neurons into cohesive sub-expert units, our method retains the flexibility of neuron-level selection while avoiding the instability of isolated-neuron updates, providing a more robust paradigm for parameter-efficient adaptation of MoE LLMs.

3 Method

3.1 Preliminaries

Mixture-of-Experts Architecture. We first revisit the standard MoE architecture. Let $x \in \mathbb{R}^{d_{\text{model}}}$ denote the input hidden state to layer l. An MoE layer contains N_E experts $\{E_{l,e}\}_{e=1}^{N_E}$ and a router that assigns routing weights to experts. For a GLU-style expert, we write

$$E_{l,e}(x) = \mathbf{W}_{\text{down}}^{l,e}\left(\text{SiLU}(g_{l,e}(x)) \odot \mathbf{W}_{\text{up}}^{l,e}x\right), \tag{1}$$

where $g_{l,e}(x) = \mathbf{W}_{\text{gate}}^{l,e}x$. Here, $\mathbf{W}_{\text{gate}}^{l,e}, \mathbf{W}_{\text{up}}^{l,e} \in \mathbb{R}^{d_{\text{expert}} \times d_{\text{model}}}$ and $\mathbf{W}_{\text{down}}^{l,e} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{expert}}}$.

The router produces expert weights $\mathbf{p}_l(x) \in \mathbb{R}^{N_E}$ and selects the top-K experts. Let $\mathcal{E}_l(x)$ denote the selected expert set. The MoE output is then

$$\text{MoE}_l(x) = \sum_{e \in \mathcal{E}_l(x)} p_{l,e}(x) E_{l,e}(x). \tag{2}$$

Neuron Expert Decomposition. Recent studies such as MoNE [5] show that each MoE expert can be decomposed into finer-grained *neuron experts*. For a GLU-style expert, the output of expert e in layer l can be expressed as $E_{l,e}(x) = \sum_{m=1}^{d_{\text{expert}}} a_{l,e,m}(x)\mathbf{A}_{l,e,m}x$, where $a_{l,e,m}(x) = \text{SiLU}(g_{l,e}(x))_m$ is the activation of the m-th intermediate channel, and $\mathbf{A}_{l,e,m} = \mathbf{W}_{\text{down}}^{l,e}[:,m]\mathbf{W}_{\text{up}}^{l,e}[m,:]$ is the rank-one transformation induced by this channel. This formulation reveals that an expert is a superposition of neuron-wise rank-one transformations, motivating us to select and update task-relevant neuron groups as sub-experts rather than treating the entire expert as an indivisible unit.

3.2 Fine-grained Sub-expert Selection

Existing expert-level sparse adaptation methods, such as ESFT, improve parameter efficiency by updating only a small subset of experts during supervised fine-tuning. However, they still treat each selected expert as an indivisible unit. In practice, the parameter space inside an activated expert remains highly sparse: only a small portion of its intermediate channels are consistently responsive to the downstream task, while many others contribute marginally. As a result, even after expert selection, updating the full expert may still introduce substantial redundancy.

This motivates us to refine the granularity of sparse adaptation from the *expert level* to the *sub-expert level*. Instead of treating each expert as a monolithic MLP block, we decompose it along the intermediate dimension into multiple finer-grained sub-experts. Such a decomposition is natural for MoE feed-forward blocks, where each expert is implemented as a gated MLP and its intermediate channels already exhibit highly non-uniform activation patterns. As shown in Figure 2, we organize consecutive intermediate channels into fixed-size groups, where each group contains G channels and serves as a candidate sub-expert. This grouping provides a structured adaptation unit that is finer than a full expert but less fragmented than individual neurons. Compared with expert-level selection,

this finer-grained formulation further removes weakly activated channels inside selected experts and yields a more faithful approximation to the effective task-specific computation path.

Formally, for layer l and expert e, let $\mathcal{T}_{l,e}$ denote the set of tokens routed to this expert, and let $m \in \{1, \ldots, d_{\text{expert}}\}$ index the intermediate channels of the expert. We first accumulate the channel-wise gate responses over all routed tokens:

$$M_{l,e,m} = \sum_{x \in \mathcal{T}_{l,e}} \left| \text{SiLU}(g_{l,e}(x))_m \right|. \tag{3}$$

To characterize the relative importance distribution inside each expert, we normalize the accumulated responses as $\hat{M}_{l,e,m} = \frac{M_{l,e,m}}{\sum_{m'} M_{l,e,m'}}$. In addition, let $N_{l,e}$ denote the number of times expert e is selected by the router in layer l. We define the expert-level routing importance as $R_{l,e} = \frac{N_{l,e}}{\sum_{e'} N_{l,e'}}$.

Based on both expert-level activity and intra-expert channel saliency, we define the channel-level sub-expert importance score as:

$$S_{l,e,m} = R_{l,e} \hat{M}_{l,e,m}. \tag{4}$$

To obtain structured sub-experts, we further aggregate channel scores within each group. Let $\mathcal{G}_{l,e,j}$ denote the j-th group of G consecutive channels in expert e of layer l. Its group-level importance score is computed as $S^G_{l,e,j} = \sum_{m \in \mathcal{G}_{l,e,j}} S_{l,e,m}$. When $G = 1$, the selection reduces to individual channel selection; larger G values produce more structured but coarser sub-expert units.

This score jointly captures whether an expert is globally important for the current task and whether a specific sub-expert group is locally important within that expert. After obtaining $S^G_{l,e,j}$, we perform layer-wise selection by flattening all sub-expert groups in the same layer and ranking them in descending order. For layer l, we select the smallest subset whose cumulative score exceeds a threshold τ: $\sum_{i=1}^{K_l} \left(S^G_l\right)^{(i)} \geq \tau$, where $\{(S^G_l)^{(i)}\}$ denotes the sorted group-level sub-expert scores in layer l. The selected subset is then used as the sub-experts for downstream tuning. In this way, our method preserves the most task-relevant internal channel groups of activated experts while filtering out low-contribution parameters.

3.3 Adaptive Training for Sub-expert Fine-tuning

After identifying the selected sub-experts via Section 3.2, fine-tuning only a small subset of channels within each expert creates two optimization issues. For expert e in layer l, let $\mathcal{A}_{l,e}$ denote the set of selected intermediate channels induced by the selected sub-expert groups, and let $K_{l,e} = |\mathcal{A}_{l,e}|$ be the number of selected channels. First, since only $K_{l,e}$ out of d_{expert} channels are updated, the effective update magnitude becomes much smaller than in full-expert tuning, leading to *step-size underestimation*. Second, even among the selected channels, their task relevance is heterogeneous, while standard optimization would treat them uniformly. We address these two issues with learning-rate scaling and gradient scaling, respectively.

Learning-rate scaling. We first compensate for the loss of global update magnitude caused by partial updates. We scale the base learning rate according to the inverse active ratio:

$$\eta_{l,e} = \eta_{\text{base}} \times \min\left(\frac{d_{\text{expert}}}{K_{l,e}},\ \alpha_{\max} \right), \tag{5}$$

where $\alpha_{\max}$ is a clipping threshold for stability. Intuitively, when only a fraction $K_{l,e}/d_{\text{expert}}$ of channels is trainable, each selected channel should receive a proportionally larger update so that the overall optimization progress remains comparable to full-expert tuning.

Gradient scaling. Learning-rate scaling restores the *global* update magnitude, but it does not distinguish the relative importance of different selected channels. We therefore further modulate the gradient inside each selected expert by applying a channel-group-level scaling mask during the backward pass. Let $\mathcal{G}_{l,e}$ denote the set of selected channel groups. For each parameter tensor, we define a mask $\mathbf{m}_{l,e}$ whose entries are zero on frozen channels and equal to a group-specific scaling coefficient σ_g on channels belonging to group $g \in \mathcal{G}_{l,e}$. The resulting masked gradient is $\tilde{\mathbf{g}}_\theta = \nabla_\theta \mathcal{L} \odot \mathbf{m}_{l,e}$.

The scaling coefficient σ_g is determined by the relative importance of each selected group. We first compute the captured importance mass of the selected channels: $\rho_{l,e} = \sum_{m \in \mathcal{A}_{l,e}} \hat{M}_{l,e,m}$. For each selected group g, we define its activation energy as $g^{(g)}_{\text{energy}} = \sum_{m \in g} \hat{M}_{l,e,m}$, and compute the mean group energy: $\bar{g}_{\text{energy}} = \rho_{l,e}/|\mathcal{G}_{l,e}|$. The group-specific scaling coefficient is then given by

$$\sigma_g = \text{clip}\left(\left(\frac{g^{(g)}_{\text{energy}}}{\bar{g}_{\text{energy}}}\right)^{\gamma},\ 1.0,\ \sigma_{\max}\right). \tag{6}$$

This design amplifies gradients for groups whose activation energy is higher than the average selected group, while avoiding excessive scaling through clipping. In this way, gradient scaling redistributes update strength among selected sub-experts according to their relative task relevance.

Entropy-adaptive modulation. To avoid overly sharp or overly flat gradient allocation, we further control the contrast among groups with an entropy-adaptive exponent:

$$\gamma = 0.5 + 0.5 \cdot \frac{H(\mathcal{G}_{l,e})}{H_{\max}}, \quad H(\mathcal{G}_{l,e}) = -\sum_{g} \hat{p}_g \log \hat{p}_g, \tag{7}$$

where

$$\hat{p}_g = \frac{g^{(g)}_{\text{energy}}}{\rho_{l,e}}, \quad H_{\max} = \log |\mathcal{G}_{l,e}|. \tag{8}$$

When the group energy distribution is uniform, γ approaches 1, preserving inter-group contrast. When it is highly concentrated, γ moves closer to 0.5, reducing extreme disparities and improving training stability.

Static and dynamic gradient scaling. A straightforward implementation is to compute the scaling coefficients once from pre-collected importance statistics and keep them fixed throughout fine-tuning. However, such static scaling can be partially absorbed by Adam-style adaptive optimization. Adam maintains the first-order moment m_t and second-order moment v_t of gradients, and updates parameters using the normalized direction $m_t/\sqrt{v_t}$. Therefore, when gradients are multiplied by a fixed coefficient, both m_t and v_t are scaled accordingly, making the normalized update relatively insensitive to this constant factor. As a result, the intended effect of static gradient scaling may be weakened during training.

To address this issue, we further introduce dynamic gradient scaling. Instead of using a fixed scaling mask, we update channel activation statistics online and periodically refresh the scaling coefficients with EMA smoothing. This allows the gradient allocation to track the evolving importance of selected sub-experts while avoiding overly noisy updates. In this way, dynamic scaling acts as an adaptive refinement of static scaling rather than a separate equivalent variant.

Overall, learning-rate scaling and gradient scaling play complementary roles: the former restores the global step size under sparse updates, while the latter allocates gradient strength among selected channels according to their relative task relevance, with dynamic scaling further improving this allocation during training.

4 Experiments

4.1 Implementation Details

Fine-tuning datasets. We evaluate our method on multiple domain-specific supervised fine-tuning datasets covering five application domains: mathematics, medicine, scientific reasoning, retrieval-augmented question answering, and code generation. Specifically, they include **MATH**, **MMedC** [33], **SciRIFF** [37], **RAGQA**, and **Code**. For RAGQA, we include both the **PubMedQA** [22] and **Health** [23] subsets. The reported MATH score is the average over **MATH500** [15] and **GSM8K** [6], while the reported Code score is the average over **HumanEval** [4] and **MBPP** [1]. These datasets are chosen to cover diverse expert activation patterns and task-specific adaptation requirements.

Method	MMedC		SciRiFF			RAG-Instruct		MATH		Code		TableQA		Avg.	General Avg.
	Train.	Domain	Train.	4k	8k	Train.	Domain	Train.	Domain	Train.	Domain	Train.	Domain		
Base-OLMoE	-	35.09	-	27.67	25.87	-	63.11	-	56.74	-	39.19	-	7.08	36.39	47.49
Full FT	100%	52.12	100%	47.14	41.95	100%	72.92	100%	59.76	100%	44.77	100%	30.70	49.91	35.98
LoRA-r32	4.49%	42.84	4.49%	35.60	35.36	4.49%	64.56	4.49%	55.68	4.49%	40.67	4.49%	15.15	41.41	44.13
LoRA-r64	8.97%	44.31	8.97%	38.56	38.07	8.97%	65.62	8.97%	55.70	8.97%	42.13	8.97%	19.26	43.38	43.41
ESFT-OLMoE	5.64%	44.31	7.46%	40.15	38.08	11.37%	64.47	5.00%	56.28	4.00%	41.19	5.55%	19.47	43.42	**46.24**
NSFT-OLMoE	4.87%	**45.81**	6.57%	**44.38**	**42.05**	10.35%	**70.36**	4.29%	**57.36**	3.21%	**42.24**	4.75%	**24.13**	**46.62**	45.96

Table 1: In-domain fine-tuning results on OLMoE under threshold $\tau = 0.2$. Each domain column reports the result after fine-tuning on the corresponding domain dataset, while **Train.** denotes the percentage of trainable parameters. Avg. and General Avg. denote the average in-domain performance and the average general-benchmark performance, respectively.

Method	Train. #Params	TableQA Subtasks				PoT Avg.	General Avg.
		Instruction Type	Fact Checking	Numerical Reasoning	Data Analysis & Visualization		
Base-Ling	–	32.29	29.72	7.65	12.00	19.79	66.45
Full FT	100%	48.96	53.40	26.07	30.00	39.34	62.59
LoRA-r32	7.45%	39.58	40.30	16.39	20.00	28.71	64.09
LoRA-r64	14.84%	43.75	43.83	**19.64**	16.00	32.00	63.90
ESFT-Ling	3.35%	37.50	21.91	9.76	**24.00**	17.68	**67.03**
NSFT-Ling	2.86%	**48.96**	**50.88**	16.34	18.00	**34.44**	66.18

Table 2: TableQA results on Ling-mini-2.0. The table reports subtask performance, the average TableQA result (PoT Avg.), and the average general-benchmark performance after fine-tuning.

Evaluation benchmarks. To evaluate general capability after fine-tuning, we use four general-purpose benchmarks: **GPQA** [34], **MMLU-Redux** [11], **C-Eval** [18], and **IFEval** [45]. These benchmarks evaluate scientific reasoning, broad knowledge understanding, Chinese knowledge and reasoning, and instruction following, respectively. In all tables, **General Avg.** denotes the average score over these four benchmarks.

Baselines. We compare our method against three categories of baselines. **Base**: the original pretrained model without any downstream fine-tuning. **Full FT**: full fine-tuning of all model parameters, serving as an upper-capacity baseline. **LoRA-r16** and **LoRA-r32**: parameter-efficient fine-tuning with LoRA of rank $r = 16$ and $r = 32$, respectively. **ESFT**: expert-level sparse fine-tuning, which updates only a subset of experts during adaptation. These baselines allow us to compare our method against dense adaptation, parameter-efficient tuning, and expert-level sparse tuning under a unified setting.

We conduct experiments on two representative MoE language models of different scales: **OLMoE-7B** [31] and **Ling-mini-2.0-16B** [20]. Using two models of substantially different sizes allows us to verify that the proposed method is not tied to a specific parameter scale and can generalize across different MoE architectures. We provide more details regarding the experiment in the Appendix A.

4.2 Main Results

We evaluate whether NSFT offers a better balance among domain adaptation, parameter efficiency, and general capability preservation than PEFT and expert-level sparse tuning baselines. Results are reported on both OLMoE and Ling-mini across domain-specific and general benchmarks. Unless otherwise specified, NSFT refers to the full method with selection threshold $\tau = 0.2$, learning-rate scaling, and dynamic gradient scaling.

NSFT significantly improves in-domain adaptation on OLMoE while preserving competitive general capability. Table 1 summarizes the main results on OLMoE across multiple domain-

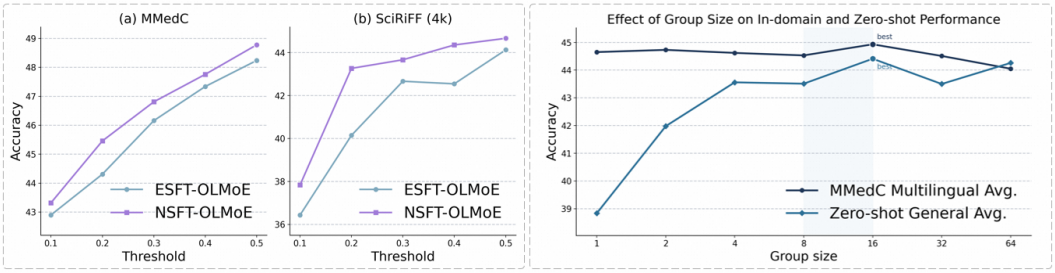

Figure 3: Threshold and group-size ablations. Left: performance under different selection thresholds on MMedC and SciRiFF. The NSFT results correspond to NSFT-SG, which uses learning-rate scaling and static gradient scaling. Right: effect of group size on in-domain (MMedC) and zero-shot general performance.

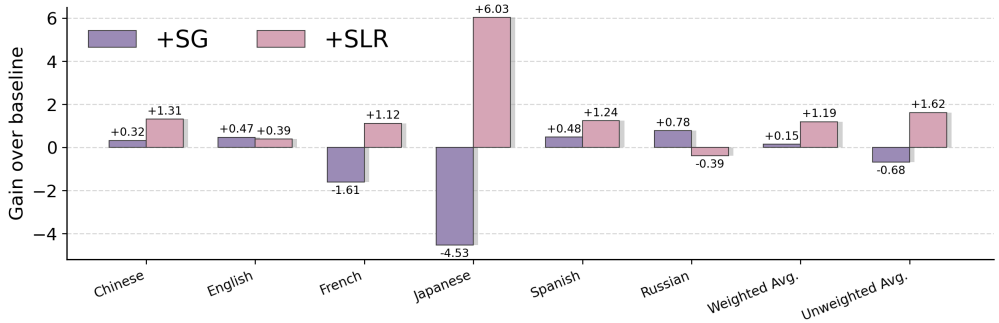

Figure 4: Ablation of gradient scaling (+SG) and learning-rate scaling (+SLR) on MMedC multilingual subtasks using OLMoE. Bars show gains over the NSFT baseline, i.e., the variant with only sub-expert selection and no adaptive training strategy.

specific fine-tuning datasets. NSFT-OLMoE achieves the best in-domain performance among efficient adaptation baselines, with an average score of 46.62, substantially outperforming ESFT-OLMoE and both LoRA variants. Full FT obtains a higher in-domain average of 49.91, but its general-benchmark average drops sharply to 35.98. In contrast, NSFT-OLMoE maintains a general average of 45.96, nearly matching ESFT-OLMoE's 46.24, while requiring only **5.67%** average trainable parameters. These results show that sub-expert-level adaptation effectively strengthens domain-specific learning without the severe generalization loss caused by full fine-tuning.

On TableQA, NSFT achieves a markedly better efficiency–performance trade-off than ESFT and LoRA. As shown in Table 2, NSFT-Ling consistently outperforms ESFT-Ling on fine-grained TableQA subtasks. It achieves a PoT average of **34.44**, substantially higher than ESFT-Ling's 17.68, while using only 2.86% trainable parameters. Compared with LoRA, NSFT-Ling is also much more parameter-efficient and surpasses the strongest LoRA variant, LoRA-r64. Moreover, it preserves stronger general capability, reaching a general average of 66.18, which is higher than all LoRA variants and clearly above Full FT. These results show that fine-grained sub-expert adaptation provides a better efficiency–performance trade-off for TableQA fine-tuning. For more detailed results and discussions on this part, please refer to the Appendix B.

These results show that both ESFT and LoRA are suboptimal for domain-specific MoE adaptation. ESFT is too coarse because it updates entire experts, while LoRA applies generic low-rank updates without explicitly targeting task-relevant expert components. In contrast, NSFT selects responsive sub-experts inside activated experts, enabling more precise parameter updates and a better efficiency–generalization trade-off.

4.3 Ablation Analysis

To investigate the key factors behind NSFT, we conduct ablations on both sub-expert selection and adaptive optimization. We analyze the sensitivity to the selection threshold, the effect of group size G,

Method	MMedC		SciRiFF			RAG-Instruct	
	Avg.	General Avg.	Avg_{4096}	Avg_{8192}	General Avg.	Avg.	General Avg.
NSFT-SG	45.46	45.43	43.27	39.99	**45.55**	68.80	**45.47**
NSFT-DSG_iter1	**45.89**	**45.75**	42.70	39.83	45.05	**70.36**	44.80
NSFT-DSG_iter5	45.81	45.51	**44.38**	**42.05**	45.46	**70.36**	45.46

Table 3: Static vs. dynamic gradient scaling using OLMoE. NSFT-SG uses a fixed scaling mask, while NSFT-DSG_iter1 and NSFT-DSG_iter5 update the dynamic scaling mask every one and five training steps, respectively. Results are reported on MMedC, SciRiFF, and RAG-Instruct, including in-domain and general-benchmark averages.

the contribution of learning-rate and gradient scaling, and the difference between static and dynamic scaling. These studies show how fine-grained selection and adaptive training jointly contribute to the effectiveness and stability of NSFT. Additional ablation results, including detailed threshold robustness, group granularity and dynamic scaling analyses, are provided in Appendix C D F.

Threshold Robustness. Figure 3 (left) compares ESFT and NSFT under different selection thresholds on MMedC and SciRiFF-4k. NSFT consistently outperforms ESFT across thresholds, showing that sub-expert-level selection is more robust and effective than expert-level sparse tuning under different sparsity budgets. This advantage mainly comes from allocating updates to task-relevant internal channels rather than entire experts. For a fair comparison with ESFT and to balance performance with training cost, we use $\tau = 0.2$ for both ESFT and NSFT in the main experiments, following the original ESFT setting.

Choosing the Sub-expert Group Granularity. In our implementation, each expert is partitioned into contiguous channel groups along the intermediate dimension, and each group serves as a candidate sub-expert. Figure 3 (right) shows the effect of group size on MMedC multilingual and zero-shot general performance. As G increases from 1 to 16, the zero-shot general average improves steadily, while the in-domain MMedC performance remains stable. The best overall balance is achieved at $G = 16$, where the general performance peaks and the in-domain score stays among the highest. This indicates a trade-off between flexibility and expressiveness. Overly small groups fragment the trainable subspace and reduce effective update capacity, while overly large groups introduce redundant channels and weaken fine-grained selection. A moderate group size, especially $G = 16$, provides the best balance.

Effect of Gradient and Learning-rate Scaling. Figure 4 compares the effect of gradient scaling (+SG) and learning-rate scaling (+SLR) on multilingual MMedC tasks. We observe that +SLR consistently improves performance on most languages and yields clear gains on both weighted and unweighted averages, indicating that compensating for the reduced effective step size is critical under partial updates. +SG also improves several languages and the weighted average, although its effect is less stable. This is because, when only a subset of sub-experts is updated, the effective update magnitude becomes significantly smaller; learning-rate scaling directly compensates for this effect at the optimizer level, while gradient scaling further enhances the selected sparse updates. These results confirm the effectiveness of our adaptive scaling strategy.

Static vs. Dynamic Scaling. To evaluate the effect of update frequency, we compare static gradient scaling with dynamic gradient scaling updated every step or every five steps, as shown in Table 3. Dynamic scaling generally improves in-domain adaptation over static scaling, showing that refreshing the scaling mask during training better captures the evolving importance of selected sub-experts. The best update interval is task-dependent: iter1 performs best on MMedC, while iter5 achieves stronger results on SciRiFF and better general retention on RAG-Instruct. This suggests that dynamic scaling improves over static scaling, but its update frequency should balance rapid adaptation with noise suppression.

5 Conclusion

We present NSFT, a fine-grained PEFT framework for MoE LLMs that refines sparse adaptation from whole experts to sub-experts. By selecting task-relevant internal channel groups inside activated experts, NSFT enables more precise updates than expert-level sparse tuning. To make such partial updates effective, we further introduce learning-rate scaling and dynamic gradient scaling to compensate for the reduced effective update magnitude. Experiments on OLMoE-7B and Ling-mini-2.0-16B show that NSFT consistently outperforms representative PEFT and expert-level sparse tuning baselines, while using substantially fewer trainable parameters and preserving competitive general capability. These results indicate that expert-level sparsity remains too coarse, and that exploiting fine-grained sub-expert structure is a promising direction for efficient adaptation of MoE LLMs.

Acknowledgments and Disclosure of Funding

This work was supported by the Alibaba Research Intern Program. We would like to thank the Qwen Team at Alibaba Token Hub (ATH), Alibaba Group, for providing the computational resources and foundation models (Qwen) used in this research.

References

[1] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[2] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[3] Chang Che, Ziqi Wang, Pengwan Yang, Qi Wang, Hui Ma, and Zenglin Shi. Lora in lora: Towards parameter-efficient architecture expansion for continual visual instruction tuning. *arXiv preprint arXiv:2508.06202*, 2025.

[4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[5] Runxi Cheng, Yuchen Guan, Yucheng Ding, Qingguo Hu, Yongxian Wei, Chun Yuan, Yelong Shen, Weizhu Chen, and Yeyun Gong. Mixture of neuron experts. *arXiv preprint arXiv:2510.05781*, 2025.

[6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[7] Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, 2024.

[8] Guanzhi Deng, Bo Li, Ronghao Chen, Huacan Wang, Lijie Wen, and Linqi Song. Dr-lora: Dynamic rank lora for mixture-of-experts adaptation. *arXiv preprint arXiv:2601.04823*, 2026.

[9] 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.

[10] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[11] Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, et al. Are we done with mmlu? In *Proceedings of the 2025 Conference of the Nations of*

the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 5069–5096, 2025.

[12] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, 2021.

[13] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[14] Haoyu He, Jianfei Cai, Jing Zhang, Dacheng Tao, and Bohan Zhuang. Sensitivity-aware visual parameter-efficient fine-tuning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11825–11835, 2023.

[15] 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. *arXiv preprint arXiv:2103.03874*, 2021.

[16] 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 *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

[17] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1 (2):3, 2022.

[18] Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in neural information processing systems*, 36: 62991–63010, 2023.

[19] Tingfeng Hui, Zhenyu Zhang, Shuohuan Wang, Weiran Xu, Yu Sun, and Hua Wu. Hft: Half fine-tuning for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12791–12819, 2025.

[20] inclusionAI. Ling-mini-2.0. `https://huggingface.co/inclusionAI/Ling-mini-2.0`, 2025. Hugging Face model repository.

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

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

[23] Neema Kotonya and Francesca Toni. Explainable automated fact-checking for public health claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7740–7754, 2020.

[24] Junzhuo Li, Bo Wang, Xiuze Zhou, and Xuming Hu. Dynamic expert specialization: Towards catastrophic forgetting-free multi-domain moe adaptation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 18489–18504, 2025.

[25] 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)*, pages 4582–4597, 2021.

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

[27] Chang Liu, Boyu Shi, Xu Yang, and Xin Geng. Xpert: Expert knowledge transfer for effective training of language models. *arXiv preprint arXiv:2605.08842*, 2026.

[28] Chang Liu, Qiufeng Wang, Xin Geng, et al. Inheriting generalizable knowledge from llms to diverse vertical tasks. In *International Conference on Learning Representations*, volume 2026, pages 114341–114359, 2026.

[29] 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. In *Forty-first International Conference on Machine Learning*, 2024.

[30] Wanlong Liu, Junying Chen, Ke Ji, Li Zhou, Wenyu Chen, and Benyou Wang. Rag-instruct: Boosting llms with diverse retrieval-augmented instructions. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3888, 2025.

[31] Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024.

[32] Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

[33] Pengcheng Qiu, Chaoyi Wu, Xiaoman Zhang, Weixiong Lin, Haicheng Wang, Ya Zhang, Yanfeng Wang, and Weidi Xie. Towards building multilingual language model for medicine. *Nature Communications*, 15(1):8384, 2024.

[34] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

[35] Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

[36] Xiaoyu Tian, Yunjie Ji, Haotian Wang, Shuaiting Chen, Sitong Zhao, Yiping Peng, Han Zhao, and Xiangang Li. Not all correct answers are equal: Why your distillation source matters, 2025. URL `https://arxiv.org/abs/2505.14464`.

[37] David Wadden, Kejian Shi, Jacob Morrison, Alan Li, Aakanksha Naik, Shruti Singh, Nitzan Barzilay, Kyle Lo, Tom Hope, Luca Soldaini, et al. Sciriff: A resource to enhance language model instruction-following over scientific literature. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6083–6120, 2025.

[38] Yaqing Wang, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. Adamix: Mixture-of-adapter for parameter-efficient tuning of large language models. *arXiv preprint arXiv:2205.12410*, 1(2):4, 2022.

[39] 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*, pages 784–801, 2024.

[40] Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xeron Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, et al. Tablebench: A comprehensive and complex benchmark for table question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25497–25506, 2025.

[41] Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. S-lora: Scalable low-rank adaptation for class incremental learning. *arXiv e-prints*, pages arXiv–2501, 2025.

[42] Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. Raise a child in large language model: Towards effective and generalizable fine-tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 9514–9528, 2021.

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

[44] Zhen-Ru Zhang, Chuanqi Tan, Haiyang Xu, Chengyu Wang, Jun Huang, and Songfang Huang. Towards adaptive prefix tuning for parameter-efficient language model fine-tuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1239–1248, 2023.

[45] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

A Experimental Setup and Dataset Details

A.1 Training and Evaluation Setup

We conduct experiments on two Mixture-of-Experts (MoE) models: Ling-mini-2.0 and OLMoE, across seven downstream tasks including TableQA, SciRIFF, Math, Code, RAGQA, MMedC, and Multilingual. All methods are trained for 2 epochs with a global batch size of 8 and a per-device batch size of 1 on 8 GPUs (NVIDIA A100 80G). We use the AdamW optimizer with β_1 = 0.9, β_2 = 0.95, and ϵ = 1e-8. The peak learning rate is set to 3e-5 with a polynomial (linear) decay schedule to a minimum learning rate of 3e-6, preceded by a warmup phase covering 2% of total training steps. The maximum sequence length is 8,192 tokens, and we employ a bin-packing strategy to concatenate multiple samples into a single sequence for efficient training. All experiments are conducted in BFloat16 precision with FlashAttention-2 and gradient checkpointing enabled. The random seed is fixed at 1,234 for reproducibility.

We set the learning-rate clipping threshold α_{max} = 5.0 and the gradient-scaling upper bound σ_{max} = 5.0 in all experiments, which prevents the inverse-active-ratio scaling in Eq. (5) and the group-level gradient coefficient in Eq. (6) from producing excessively large updates and thus stabilizes training.

We compare four fine-tuning methods under this unified setting. Full SFT updates all model parameters using DeepSpeed ZeRO Stage 3 with a weight decay of 0.1. ESFT selects task-relevant experts via a gate score threshold of 0.2 and fine-tunes only those experts, using DeepSpeed ZeRO Stage 1 with a weight decay of 0.1. LoRA applies low-rank adaptation with rank 32, α = 64, and dropout 0.05, targeting the MLP layers (gate_proj, up_proj, down_proj), trained with DeepSpeed ZeRO Stage 1 and a weight decay of 0.1. NSFT performs fine-grained sub-expert updates with dynamic gradient scaling guided by pre-computed gate energy distributions. NSFT uses DeepSpeed ZeRO Stage 1 with weight decay set to 0 to prevent frozen parameters from drifting, and employs MoE-aware learning rate scaling with an EMA decay factor of 0.9 and a scale update interval of 5 steps. By default, NSFT uses a group size of 16 and a gate score threshold of 0.2; we also ablate on group sizes of 1, 2, 4, 8, 32 and thresholds of 0.2, 0.3, 0.4.

A.2 Datasets

A.2.1 Dataset Details

We evaluate our method across seven diverse and challenging downstream tasks spanning distinct domains:

- TableQA (TableInstruct), a table understanding benchmark requiring structured reasoning over tabular data;
- SciRIFF, a scientific instruction-following dataset demanding domain-specific reasoning across scientific literature;

- Math (GSM8k and MATH500), evaluated on two complementary mathematical reasoning benchmarks and reported as their average score;
- Code (HumanEval and MBPP), evaluated on two widely-used code generation benchmarks and reported as their average score;
- RAG (PubMedQA and Health_Claims), a retrieval-augmented QA task evaluated on two benchmarks and reported as their average, requiring the model to synthesize answers from long retrieved contexts in the biomedical domain.

All datasets are formatted in the standard chat-based message format and pre-tokenized offline with a maximum sequence length of 8,192 tokens. We adopt a bin-packing strategy to concatenate multiple samples into packed sequences for efficient training.

A.2.2 On the choice of evaluation datasets and benchmarks

Our benchmark suite is deliberately chosen to be complex and challenging — spanning structured reasoning (TableQA), domain-specific scientific and medical knowledge (SciRIFF, MMedC), long-context retrieval comprehension (RAGQA), multi-step mathematical and code reasoning (Math, Code), and cross-lingual generalization (Multilingual). These tasks demand deep adaptation of model knowledge and are far more likely to expose the limitations of parameter-efficient approaches when compared to full fine-tuning. By evaluating under such demanding conditions, we aim to provide a more rigorous and convincing assessment of our proposed NSFT method, demonstrating that it can maintain competitive performance even in scenarios where conventional PEFT methods typically fall short.

B Additional Results on TableQA

Table 4 provides a more detailed comparison on TableQA using Ling-mini-2.0. We include LoRA variants with different ranks and target modules, ESFT under different selection thresholds, and NSFT under different thresholds and group sizes. These results further validate the advantage of fine-grained sub-expert adaptation.

NSFT achieves the strongest efficiency–performance trade-off on TableQA. Compared with LoRA and ESFT, NSFT obtains competitive or stronger TableQA performance while using substantially fewer trainable parameters. In particular, NSFT-G4-th0.2 reaches a PoT average of 34.44 with only 2.86% trainable parameters, outperforming all ESFT variants and approaching the strongest LoRA variant, LoRA-128 MLP+QKVO, which obtains a PoT average of 35.35 but requires 29.69% trainable parameters. This indicates that updating fine-grained sub-experts can recover most of the task-specific adaptation ability of much larger PEFT configurations with nearly an order of magnitude fewer trainable parameters.

NSFT is consistently stronger than expert-level sparse tuning. Across different thresholds, NSFT clearly outperforms ESFT on TableQA. The best ESFT variant reaches a PoT average of 21.65, whereas multiple NSFT variants exceed this score by a large margin. For example, NSFT-G16-th0.2, NSFT-G8-th0.2, NSFT-G4-th0.2, NSFT-G2-th0.2, and NSFT-G1-th0.2 all outperform ESFT-th0.3. This confirms that expert-level selection is too coarse for TableQA: updating whole experts still includes many task-irrelevant internal channels, while sub-expert selection can allocate the update budget to more relevant parts inside experts.

NSFT better preserves general capability than full fine-tuning and large LoRA variants. Although Full FT obtains the highest PoT average, it significantly reduces the general average from 66.45 to 62.59. Large LoRA variants also tend to underperform the base model on general benchmarks. In contrast, NSFT maintains competitive general capability, with several variants achieving general averages around 66.0–66.5 while substantially improving TableQA performance. This shows that fine-grained sub-expert tuning provides a better balance between domain specialization and general capability retention.

Discussion on ESFT degradation. Table 4 shows that ESFT underperforms the Ling-mini-2.0 base model on TableQA, while LoRA variants that only update MoE-related modules also provide

Method	Train. #Params	TableQA Subtasks				PoT Avg.	General Avg.
		Instruction Type	Fact Checking	Numerical Reasoning	Data Analysis & Vis.		
Ling-mini-2.0 (Base)	–	32.29	29.72	7.65	12.00	19.79	66.45
Full FT	100%	48.96	53.40	26.07	30.00	**39.34**	62.59
LoRA-r32 MLP	7.38%	9.38	8.56	3.99	16.00	6.41	63.63
LoRA-r32 MLP+QKVO	7.45%	39.58	40.30	16.39	20.00	28.71	64.09
LoRA-r64 MLP	14.77%	11.46	12.09	7.45	18.00	9.55	63.38
LoRA-r64 MLP+QKVO	14.84%	43.75	43.83	19.64	16.00	32.00	63.90
LoRA-r128 MLP	29.54%	20.83	24.18	13.75	24.00	18.43	62.89
LoRA-r128 QKVO	0.15%	27.08	15.87	7.72	18.00	13.04	61.06
LoRA-r128 MLP+QKVO	29.69%	48.96	48.61	21.30	26.00	35.35	63.09
ESFT-th0.2	8.92%	37.50	21.91	9.76	24.00	17.68	67.03
ESFT-th0.3	5.75%	34.38	32.75	8.37	12.00	21.65	67.08
ESFT-th0.4	3.35%	25.00	24.18	7.12	16.00	16.31	66.29
NSFT-G16-th0.2	3.05%	42.71	39.40	11.53	26.59	26.59	66.53
NSFT-G16-th0.3	5.45%	30.21	39.29	16.48	27.27	27.27	65.99
NSFT-G16-th0.4	8.57%	33.33	45.09	12.18	15.00	28.54	66.31
NSFT-G32-th0.2	3.09%	30.21	39.55	8.03	18.00	24.11	66.40
NSFT-G8-th0.2	2.98%	57.29	46.85	16.95	18.00	33.77	66.04
NSFT-G4-th0.2	2.86%	48.96	50.88	16.34	18.00	34.44	66.18
NSFT-G2-th0.2	2.66%	43.75	45.59	18.36	16.00	32.28	66.04
NSFT-G1-th0.2	2.40%	43.75	41.31	16.98	16.00	29.83	64.98

Table 4: Detailed TableQA results on Ling-mini-2.0. We compare Full FT, LoRA with different ranks and target modules, ESFT under different selection thresholds, and NSFT under different thresholds and group sizes. The table reports performance on four TableQA subtasks, the PoT average, and the average performance on general benchmarks. Bold numbers indicate the best result within each column.

limited gains without a much larger trainable budget. A likely reason is that Ling-mini itself adopts a finer expert partitioning than OLMoE, using 256 experts instead of 64. This architectural design indicates that Ling-mini distributes its computation across more fine-grained expert units. Accordingly, adaptation should also be performed at a finer granularity: updating entire selected experts may still be too coarse, while sub-expert-level updates can better match the model's internal organization.

This highlights the need to align adaptation granularity with model granularity. NSFT further decomposes experts into structured sub-expert groups and updates only the most responsive ones, providing a finer update unit that better matches Ling-mini's architecture. This explains why NSFT achieves stronger TableQA performance with fewer trainable parameters and suggests that fine-grained MoE backbones may benefit more from sub-expert-level adaptation.

C Robustness across Selection Thresholds

Discussion. Tables 5 and 6 report detailed results under different selection thresholds. NSFT consistently outperforms ESFT across both MMedC and SciRIFF, showing that sub-expert-level selection is more robust than expert-level sparse tuning under different sparsity budgets. Performance generally improves as τ increases, since more task-relevant units are retained. These results suggest that the optimal granularity depends on the target domain, but the advantage of fine-grained sub-expert selection remains consistent.

Method	$\tau = 0.1$	$\tau = 0.2$	$\tau = 0.3$	$\tau = 0.4$	$\tau = 0.5$
ESFT-OLMoE	42.90	44.31	46.16	47.34	48.24
NSFT-OLMoE	**43.32**	**45.46**	**46.81**	**47.76**	**48.78**

Table 5: Performance comparison under different selection thresholds on MMedC. We compare ESFT and NSFT variants under different selection thresholds τ. Best results under each threshold are bolded. The NSFT results correspond to NSFT-SG, which uses learning-rate scaling and static gradient scaling.

Method	$\tau = 0.1$	$\tau = 0.2$	$\tau = 0.3$	$\tau = 0.4$	$\tau = 0.5$
ESFT-OLMoE	36.44 / 35.62	40.15 / 38.08	42.67 / 40.21	42.55 / 39.59	44.13 / 41.13
NSFT-OLMoE	**37.84 / 36.97**	**43.27 / 39.99**	**43.67 / 40.74**	**44.36 / 41.06**	**44.67 / 41.40**

Table 6: Performance comparison under different selection thresholds on SciRIFF under 4k / 8k settings. Each cell reports results under 4k and 8k training lengths, respectively. Best results under each threshold and setting are bolded. The NSFT results correspond to NSFT-SG, which uses learning-rate scaling and static gradient scaling.

D Group Size Sensitivity across MoE Backbones

The optimal group size is backbone-dependent. Although our main experiments on OLMoE suggest that a moderate group size such as $G = 16$ provides a strong balance between flexibility and stability, the results on Ling-mini-2.0 show that the optimal group size can shift across MoE backbones. As shown in Table 7, when applying NSFT to TableQA on Ling-mini-2.0, the best PoT performance is achieved at $G = 4$, reaching 34.44. This differs from the trend observed on OLMoE, where larger groups such as $G = 16$ are often more favorable. This indicates that group size should not be treated as a universal constant, but rather as a backbone- and task-dependent hyperparameter.

Smaller groups provide broader expert coverage on Ling-mini-2.0. Table 7 reveals a clear relationship between group size, expert coverage, and task performance. As G decreases from 32 to 1, the number of involved experts increases substantially, from 230 to 1024. This suggests that smaller groups allow NSFT to select sub-experts from a much broader set of experts, thereby covering more diverse computation paths. For Ling-mini-2.0 on TableQA, this broader coverage appears to be important, likely because TableQA requires heterogeneous capabilities such as instruction understanding, fact checking, numerical reasoning, and data analysis. Therefore, a smaller group size can better capture the distributed expert patterns needed by this task.

Connection to expert-level tuning. It is also worth noting that when the group size becomes equal to the expert intermediate dimension, i.e., $G = 1024$ in Ling-mini-2.0, each expert contains only one group. In this extreme case, sub-expert selection degenerates into expert-level selection, which is essentially equivalent to ESFT. As shown in Table 7, this setting involves 173 experts and achieves a TableQA PoT score of 17.68, much lower than the best sub-expert setting with $G = 4$. This result further confirms that expert-level granularity is too coarse: updating an entire expert introduces substantial redundant parameters, while decomposing experts into finer-grained sub-experts enables more precise and effective adaptation.

These results provide a practical guideline for choosing the group size. When task-relevant computation is distributed across many experts, smaller groups such as $G = 4$ or $G = 8$ offer broader expert coverage and greater selection flexibility. When useful activations are concentrated within fewer experts, moderately larger groups such as $G = 16$ can be more stable and efficient. We therefore treat G as a lightweight hyperparameter and select it with a small validation sweep, using expert coverage and intra-expert activation ratio as diagnostic signals to avoid both overly fragmented and overly coarse updates.

Group Size	1	2	4	8	16	32	1024
Number of involved experts a	**1024**	730	526	376	288	230	173
Average activated ratio within selected experts b	12.09%	18.85%	28.09%	40.91%	54.69%	69.46%	**100.0%**
TableQA PoT	29.83	32.28	**34.44**	33.77	26.59	24.11	17.68

Table 7: Relationship between group size, activated expert coverage, and TableQA PoT performance on Ling-mini-2.0.

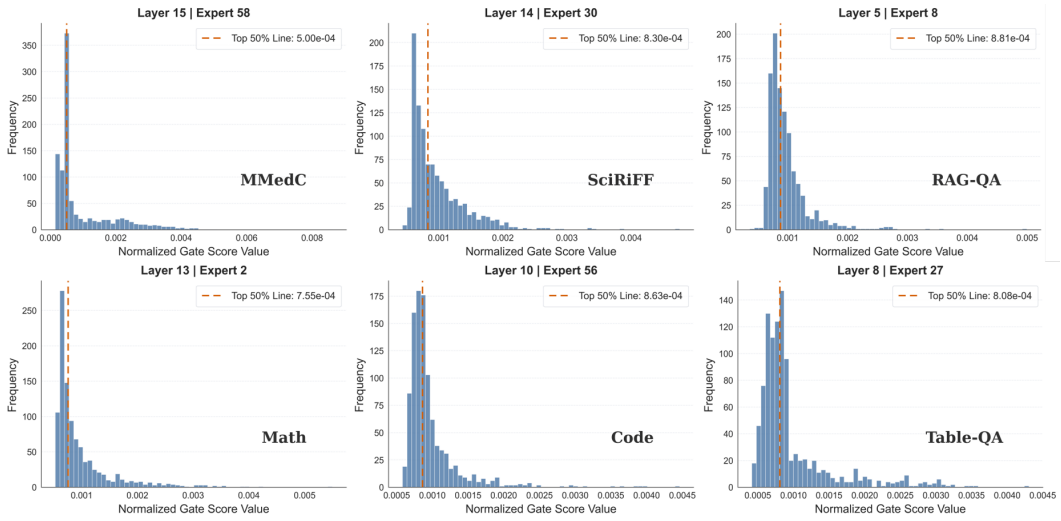

Figure 5: Intra-expert activation sparsity across domains. Each subplot shows the normalized channel-wise gate-score distribution of a representative OLMoE expert. The dashed orange line marks the top-50% cumulative activation threshold. The strong concentration near zero across domains indicates that only a small subset of channels is highly activated inside selected experts.

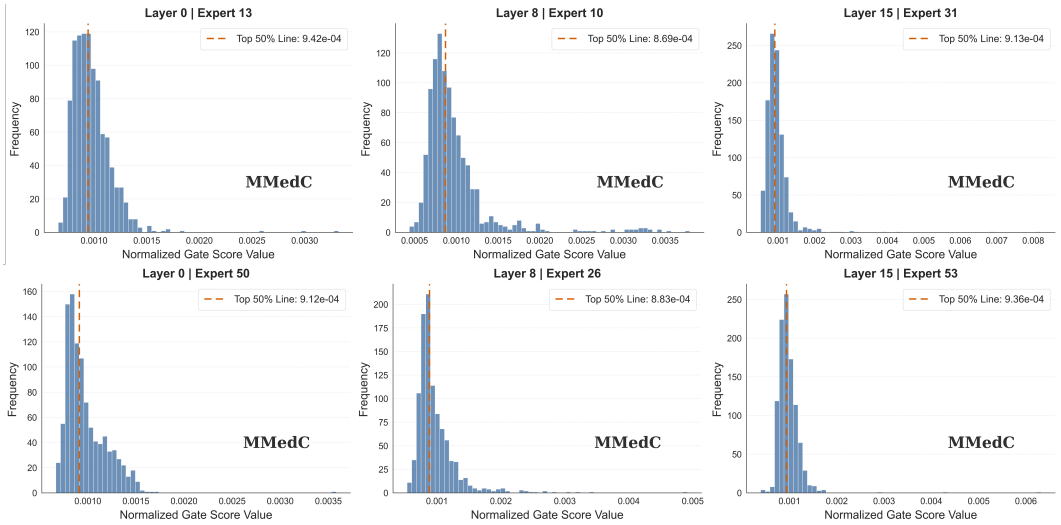

Figure 6: Channel-wise gate-score distributions for randomly sampled experts from shallow, middle, and deep layers on MMedC, with two experts sampled from each depth range. The dashed orange line marks the threshold covering the top 50% cumulative activation mass.

Method	Chinese	English	French	Japanese	Spanish	Russian	W-AVG	UW-AVG	General Avg.
NSFT-SG	46.82	49.73	41.80	25.13	42.74	59.77	45.46	44.33	45.43
NSFT-DSG_iter1	47.23	50.04	**45.34**	23.62	**42.78**	59.38	**45.89**	44.73	**45.75**
NSFT-DSG_iter5	**48.25**	**48.23**	43.73	**26.63**	42.16	**60.16**	45.81	**44.86**	45.51

Table 8: Static vs. dynamic gradient scaling on MMedC multilingual subtasks using OLMoE. **DSG** denotes dynamic gradient scaling, and **DSG-EMA** applies EMA smoothing. The table reports language-wise results, weighted/unweighted averages, and general-benchmark average.

E Additional Analysis for Intra-expert Sparsity in ESFT

E.1 Intra-expert activation sparsity across domains

Figure 5 visualizes the channel-wise gate-score distributions of representative OLMoE experts from different layers and domains. Across all datasets, the distributions are highly concentrated near zero, while only a small fraction of channels receive relatively large activation scores. The dashed orange line marks the threshold covering the top 50% cumulative activation mass, and it consistently lies in a low-value region. This indicates that even within activated or selected experts, the effective activation mass is dominated by a small subset of internal channels. Such a pattern is observed across diverse domains, including medical, scientific, retrieval, mathematical, code, and table reasoning tasks, suggesting that intra-expert sparsity is a general property rather than a domain-specific artifact. These findings further support our motivation for sub-expert selection: updating an entire selected expert, as in expert-level sparse tuning, inevitably includes many weakly activated channels, whereas fine-grained sub-expert selection can focus the trainable budget on the most responsive internal components.

E.2 Consistent sparsity across experts

To further verify that intra-expert sparsity is not limited to a few special cases, Figure 6 visualizes the distributions of normalized gate scores for randomly sampled experts from shallow, middle, and deep layers. Specifically, we randomly select two experts from each of the shallow, middle, and deep layers. Despite differences in layer depth and expert identity, all sampled experts exhibit a highly similar pattern: most channels are concentrated in the near-zero region, while only a small fraction of channels form a long tail with relatively larger activations. The dashed line marking the threshold that covers the top 50% cumulative activation mass also lies in a very small-value region across all cases.

These results confirm that strong intra-expert sparsity is a stable and widespread phenomenon, rather than an artifact of a particular layer or expert. This consistency further motivates our sub-expert selection strategy, which allocates updates to the most responsive internal channel groups instead of updating the full expert uniformly.

F Additional Results on Static and Dynamic Gradient Scaling

In the main text, we report the averaged results over three datasets to summarize the overall effect of static and dynamic gradient scaling. Here, we provide detailed results on each dataset for a more fine-grained comparison.

Tables 8 and 9 provide additional comparisons between static gradient scaling and dynamic gradient scaling on MMedC and SciRiFF. Overall, dynamic scaling generally improves in-domain adaptation over the static variant, but the best update frequency varies across tasks. On MMedC, updating the scaling mask every training step achieves the best weighted average and general-benchmark average, while updating every five steps gives the best unweighted average. On SciRiFF, the five-step update performs better, achieving the highest weighted and unweighted validation averages under both 4096 and 8192 settings. These results suggest that dynamic scaling can better track the evolving importance of selected sub-experts, while a less frequent update schedule can be beneficial when online statistics are noisy.

Method	**BioASQ** (64)		**BioRED** (100)		**Evidence_Inference** (250)		**SciERC** (50)		**W-AVG**		**UW-AVG**		**General Avg.**
	4096	8192	4096	8192	4096	8192	4096	8192	4096	8192	4096	8192	
NSFT-SG	62.17	43.41	72.73	71.88	26.63	25.61	43.35	43.75	43.27	39.99	51.22	46.16	**45.55**
NSFT-DSG_iter1	63.06	42.73	72.43	72.21	25.36	25.25	**43.86**	**44.22**	42.70	39.83	51.18	46.10	45.05
NSFT-DSG_iter5	**63.52**	**44.55**	**73.50**	**73.54**	**28.46**	28.68	41.23	42.70	**44.38**	**42.05**	**51.68**	**47.37**	45.46

Table 9: Static vs. dynamic gradient scaling on SciRiFF subtasks using OLMoE.

G Low-Budget and Budget-Matched Comparisons

Lower-budget regime. Table 10 extends the comparison between ESFT and NSFT to selection thresholds below the default $\tau = 0.2$. At $\tau = 0.05$, the coarse selection unit of ESFT causes nearly every layer to retain only one complete expert, which represents a practical lower bound on its parameter budget. In contrast, NSFT can continue reducing the budget by selecting groups within experts. NSFT consistently obtains higher MMedC performance while updating fewer parameters across all three thresholds, demonstrating a stronger accuracy–parameter trade-off in the low-budget regime.

	ESFT		NSFT		Δ
Threshold	Train.	MMedC Avg.	Train.	MMedC Avg.	NSFT$-$ESFT
$\tau = 0.20$	5.64%	44.31	4.87%	**45.74**	+1.43
$\tau = 0.10$	3.09%	42.90	2.08%	**43.32**	+0.42
$\tau = 0.05$	1.55%	41.35	0.97%	**42.30**	+0.95

Table 10: Low-budget comparison between ESFT and NSFT on MMedC. Train. denotes the percentage of trainable parameters.

Closely matched parameter budgets. Because ESFT selects complete experts whereas NSFT selects groups within experts, their trainable-parameter ratios cannot generally be matched exactly using the same threshold. We therefore adjust the NSFT threshold to $\tau = 0.22$ to obtain budgets close to those of ESFT at $\tau = 0.20$. As shown in Table 11, NSFT remains stronger on both SciRIFF and MMedC under closely matched budgets. This result rules out a larger trainable-parameter budget as the source of NSFT's improvement.

		SciRIFF		MMedC	
Method	Threshold	Train.	4k / 8k Avg.	Train.	Avg.
ESFT	$\tau = 0.20$	7.46%	40.15 / 38.08	5.64%	44.31
NSFT	$\tau = 0.20$	6.57%	40.48 / 38.88	4.87%	44.93
NSFT	$\tau = 0.22$	7.55%	**41.88 / 39.63**	5.54%	**45.50**

Table 11: Comparison under closely matched trainable-parameter budgets. The NSFT threshold is adjusted to match the ESFT budget as closely as permitted by their different selection granularities.

Weight-decay control. The main experiments follow the recommended settings of each baseline: ESFT uses weight decay 0.1, whereas NSFT uses weight decay 0 to avoid decay-induced updates to masked parameters. As a control, setting the ESFT weight decay to 0 yields an MMedC average of 44.01, compared with 44.31 under its original weight decay of 0.1. The result is slightly lower rather than higher, indicating that the advantage of NSFT is not caused by the different weight-decay settings.

H Alternative Sub-Expert Grouping Strategies

Our default method forms structured groups from consecutive channels. This choice does not assume that individual channels have explicit human-interpretable semantics; rather, it provides a simple and

low-cost way to construct stable update units in the learned parameter layout. To test whether the result depends on this choice, we compare it with three alternatives while keeping the remaining training settings fixed: (1) random grouping, which randomly assigns channels to groups of size 16; (2) activation-clustering grouping, which clusters channels according to the cosine similarity of their normalized up-projection responses over tokens; and (3) individual-channel selection ($G = 1$), which performs top-K non-contiguous selection without structured grouping.

Table 12 shows that random and activation-clustering groups mitigate the general-capability degradation of individual-channel selection, but remain weaker than consecutive grouping. The default strategy therefore provides the best overall balance between in-domain adaptation, general capability, simplicity, and grouping overhead.

Grouping Strategy	MMedC W-Avg.	General Avg.
Random grouping ($G = 16$)	44.91	43.71
Activation-clustering grouping ($G = 16$)	44.04	43.77
Individual channels ($G = 1$)	44.65	38.84
Consecutive grouping ($G = 16$, default)	**44.93**	**44.41**

Table 12: Comparison of alternative sub-expert grouping strategies on MMedC. W-Avg. denotes the weighted average over the multilingual subtasks.

I Contributions of Selection and Adaptive Optimization

To isolate the effect of fine-grained selection, we begin with NSFT-G16-Select, which uses only sub-expert selection at $\tau = 0.2$ and removes all adaptive optimization strategies. We then progressively add static and dynamic gradient scaling. The static variant includes learning-rate scaling and a fixed gradient-scaling mask, whereas the dynamic variant periodically refreshes the mask using updated activation statistics.

As shown in Table 13, selection alone already outperforms ESFT on both MMedC and SciRIFF-8k, confirming that the primary gain comes from refining the update unit from experts to sub-experts. Static and dynamic scaling then provide further, complementary improvements, with dynamic scaling giving the largest gain on SciRIFF-8k.

Method	MMedC Avg.	SciRIFF-8k Avg.
ESFT	44.31	38.08
NSFT-G16-Select	44.93	38.74
+ Static Gradient Scaling	45.46	39.99
+ Dynamic Gradient Scaling	**45.81**	**42.05**

Table 13: Step-by-step ablation separating sub-expert selection from adaptive optimization.

J Effect of the Entropy-Adaptive Exponent

The entropy-adaptive exponent γ smooths group-wise gradient scaling: when group energies are relatively uniform, γ approaches 1 and preserves their contrast; when the distribution is concentrated, γ moves toward 0.5 and suppresses extreme scaling coefficients. We evaluate its effect using the $G = 32$ model with static gradient scaling on MMedC. Table 14 shows that replacing the adaptive exponent with a fixed exponent of 1.0 reduces the MMedC average from 45.06 to 44.87. Although the gain is modest, it consistently supports adaptive smoothing over the raw energy-ratio scaling rule.

K Training Efficiency and Dynamic-Scaling Overhead

Table 15 reports wall-clock training time, peak GPU memory, and MMedC performance. Wall-clock time is measured using eight NVIDIA A100 GPUs, while peak per-GPU memory is measured using four A100 GPUs under the same MMedC setting. NSFT-selection has a training time close to

Method	Exponent in Gradient Scaling	MMedC Avg.
NSFT w/o adaptive γ	1.0	44.87
NSFT w/ adaptive γ	Entropy-adaptive	**45.06**

Table 14: Ablation of the entropy-adaptive exponent using $G = 32$ and static gradient scaling on MMedC.

ESFT and already achieves higher accuracy, showing that sub-expert selection itself introduces little additional time overhead. All NSFT variants also retain memory consumption close to ESFT and substantially below Full FT.

The main additional cost of full NSFT comes from collecting updated activation statistics and refreshing the scaling mask. Removing dynamic updates reduces training time from 35m 29s to 20m 38s without increasing memory usage. Dynamic scaling should therefore be viewed as an optional performance enhancement: users may choose selection-only, static-scaling, or full dynamic-scaling NSFT according to their compute budget. More frequent updates track rapidly changing sub-expert importance more closely, whereas less frequent updates can be more stable when the activation statistics are noisy or heterogeneous.

Method	MMedC Avg.	Time (8×A100)	Peak Memory (4×A100)
Full FT	**52.12**	25m 24s	67 GB
ESFT	44.31	17m 19s	24 GB
NSFT-selection	44.93	18m 30s	26 GB
NSFT-static	45.46	20m 38s	26 GB
NSFT	45.81	35m 29s	26 GB

Table 15: Training cost and performance on MMedC. Wall-clock time and peak per-GPU memory are measured with eight and four A100 GPUs, respectively.

L Limitations

Although NSFT achieves strong results across representative MoE backbones and diverse downstream tasks, several aspects deserve further exploration. First, we evaluate NSFT on two competitive MoE models, OLMoE-7B and Ling-mini-2.0-16B, which already cover different scales and architectural characteristics. The consistent improvements across these models suggest that sub-expert-level adaptation is broadly effective. Future work may further examine NSFT on larger and more heterogeneous MoE LLMs to better characterize its scaling behavior.

Second, NSFT uses the group size G and selection threshold τ to control the granularity and budget of sub-expert selection. Our experiments show stable gains across multiple choices of these hyperparameters, and the default setting works well in practice. Still, for new domains or substantially different MoE architectures, a lightweight validation sweep may help identify the best efficiency–performance trade-off.

M Broader Impacts

This work improves the parameter efficiency of adapting MoE LLMs to downstream domains, which can reduce computational cost, memory usage, and energy consumption compared with full-parameter fine-tuning. This may make domain-specific model adaptation more accessible to users with limited hardware resources.

However, efficient adaptation may also lower the barrier to customizing powerful models for harmful or misleading purposes. Moreover, domain-specific fine-tuning can amplify biases, errors, or unsafe behaviors present in the training data. We therefore recommend careful data curation, safety evaluation, and deployment monitoring, especially for high-stakes applications.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the scope of the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in Section L,

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper presents formal theoretical theorems or proofs. We provide mathematical definitions of sub-expert partitioning, activation-based selection, and masked optimization.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe the models, datasets, baselines, selection thresholds, group sizes, trainable parameter ratios, optimization settings, and evaluation benchmarks in Section 4.1 and Appendix A. These details are sufficient to reproduce the main comparisons between full fine-tuning, LoRA, ESFT, and the proposed method.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: At submission time, we submitted the code in the form of a compressed file in the supplementary materials.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify the training datasets, evaluation benchmarks, baselines, optimizer, learning rate, batch size, training steps, selection threshold, group size, and decoding/evaluation settings in Section 4.1 and Appendix A.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Due to the high computational cost of fine-tuning and evaluating multiple large MoE language models across several domains, we do not report multi-seed error bars for all experiments. Instead, we provide consistent comparisons across multiple datasets, benchmarks, model families, and ablation settings to support the main empirical conclusions.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the GPU type, number of GPUs, memory requirements, training time, and approximate total compute used for the main experiments in Appendix A.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses publicly available models, datasets, and evaluation benchmarks, and does not involve deception, privacy-sensitive data collection, or human-subject experiments.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss broader impacts in Section M. The proposed method may reduce the computational cost of adapting large MoE models, but it may also lower the barrier to adapting powerful language models for harmful or unreliable downstream uses if deployed without appropriate safeguards.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a new pretrained language model, image generator, or scraped dataset with high misuse risk. Our contribution is a fine-tuning method, and experiments are conducted on existing models and datasets under their respective licenses and usage terms.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite all existing models, datasets, and codebases used in the paper, including the base MoE models, domain-specific datasets, general evaluation benchmarks, and baseline implementations.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce a new dataset or pretrained model asset.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

Justification: The paper does not involve crowdsourcing, user studies, or research with human subjects. All experiments are conducted using existing datasets and automatic evaluation protocols.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human-subject research, crowdsourcing, or collection of data from participants. Therefore, IRB approval or equivalent review is not applicable.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were used only for language polishing and editing assistance. They were not used to develop the core idea, design the algorithm, implement the method, conduct experiments, analyze results, or produce any original scientific or engineering contributions of this work.