Parameter-Efficient Routed Fine-Tuning: Mixture-of-Experts Demands Mixture of Adaptation Modules

Yilun Liu[1,2,3,†] **Yunpu Ma**[2,3,†] **Yuetian Lu**[2,3,4]
Shuo Chen[2,3,5,6] **Zifeng Ding**[2,7] **Volker Tresp**[2,3,5,†]
[1]Technical University of Munich [2]Ludwig Maximilian University of Munich
[3]Munich Center for Machine Learning [4]Technical University of Darmstadt
[5]Konrad Zuse School of Excellence in Reliable AI
[6]Siemens AG [7]University of Cambridge
[†]*Equal contribution.* yilun.liu@tum.de cognitive.yunpu@gmail.com

Abstract

Mixture-of-Experts (MoE) benefits from a dynamic routing mechanism among their specialized experts, which existing Parameter- Efficient Fine-Tuning (PEFT) strategies often fail to leverage. This motivates us to investigate whether adaptation modules themselves should incorporate routing mechanisms to align with MoE's multi-expert architecture. We analyze dynamics of core components when applying PEFT to MoE language models, and examine how different routing strategies affect adaptation effectiveness. Extensive experiments adapting OLMoE-1B-7B and Mixtral-8×7B on various commonsense and math reasoning tasks validate the performance and efficiency of our routed approach. We identify optimal configurations for different scenarios and provide empirical analyses with practical insights to facilitate better PEFT and MoE applications. [1]

1 Introduction

As modern transformer-based large language models (LLMs) continue to scale (Vaswani et al., 2017), Mixture-of-Experts (MoE) has emerged as a promising approach (Shazeer et al., 2017), powering series of frontier models (Jiang et al., 2024; Qwen, 2024; DeepSeek-AI, 2025). Fine-tuning these sparse yet massive models poses unique challenges, that direct full fine-tuning is not only expensive but ignores the routed dynamics and sparsity of experts, negating their computational advantages (Wang et al., 2024). Existing Parameter-Efficient Fine-Tuning (PEFT) strategies like LoRA (Low-Rank Adaptation) have been widely studied on dense LLMs (Houlsby et al., 2019; Hu et al., 2022; He et al., 2022). Yet directly adapting MoE LLMs with PEFT is not an ideal solution, since current practice often treats MoE as dense and only addresses MoE-irrelevant modules.

These observations motivate us to investigate the designs for PEFT modules that consider the underlying routing mechanisms of MoE. Recent studies have explored *MoE-inspired* PEFT modules targeting dense backbones (Zadouri et al., 2023; Li et al., 2024; Hao et al., 2024), which inspired us to propose that a mixture of PEFT modules should be similarly required for adapting MoE LLMs.

To verify this, we start by analyzing the dynamics between *key memory vectors* (Geva et al., 2021) in experts and *expert vectors* in routers. In §2.1 and Figure 1, We demonstrate that properly routed PEFT experts can unlock a much more expressive adaptation space while maintaining MoE's efficiency and flexibility.

Guided by these insights, we introduce a framework to explore meaningful design choices for integrating PEFT modules into MoE LLMs in §2.2 and Figure 2. We define (i) *functional* strategies, including the architecture, multiplicity, routing among PEFT experts; and (ii) *compositional* strategies, specifying how PEFT modules interact with the original MoE module. Within this framework, we further propose **P**arameter-**E**fficient **R**outed **F**ine-**T**uning (**PERFT**) and three ablated variants (PERFT-E/D/S) in §2.3 and Figure 3. These strategies allow us to systematically verify if MoE actually demands a mixture of adaptation modules.

We evaluate our proposed strategies on OLMoE-1B-7B (Muennighoff et al., 2024) and Mixtral-8×7B (Jiang et al., 2024) across 14 commonsense and arithmetic reasoning tasks. PERFT yields up to 17.9% average improvement over MoE-agnostic baselines with equivalent number of activated parameters, showing that mixture of adaptation modules can indeed achieve better results on MoE LLMs. We also systematically explore the opti-

Work partially conducted while Yuetian Lu was at the Technical University of Munich.

[1]Code for PERFT and experiments available at https://github.com/liuyilun2000/PERFT.

Findings of the Association for Computational Linguistics: EACL 2026, pages 4439–4457
March 24-29, 2026 ©2026 Association for Computational Linguistics

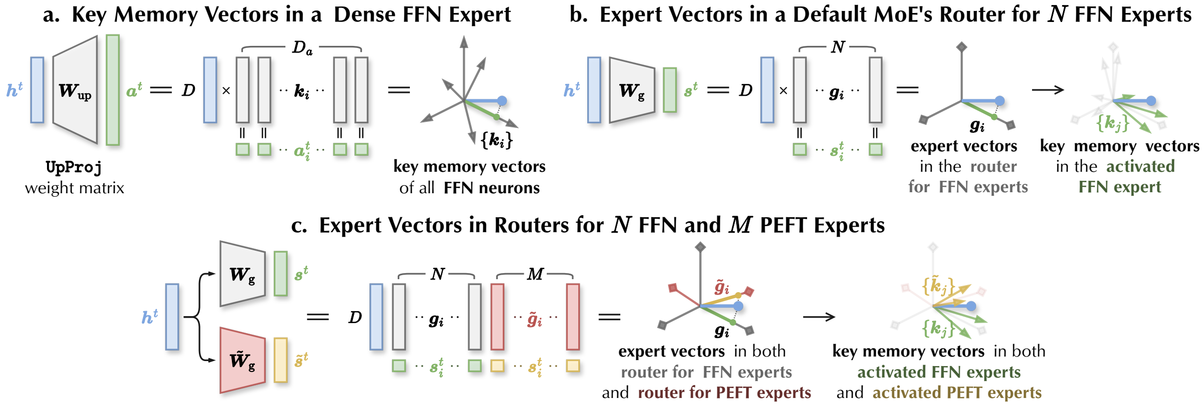


Figure 1: **Dynamics between key memory vectors in experts and expert vectors in routers. a.** Dense Feed-Forward Network (FFN) projects hidden state $\boldsymbol{h}^t \in \mathbb{R}^D$ onto D_a key memory vectors $\boldsymbol{k}_i \in \mathbb{R}^D$ in weight matrix $\boldsymbol{W}_{\text{up}}$, yielding activation scores $\boldsymbol{a}^t \in \mathbb{R}^{D_a}$. **b.** Router for N FFN experts projects $\boldsymbol{h}^t$ onto N expert vectors $\boldsymbol{g}_i \in \mathbb{R}^D$ in router weight matrix $\boldsymbol{W}_g$, yielding token-to-expert affinity scores $\boldsymbol{s}^t \in \mathbb{R}^N$. Each $\boldsymbol{g}_i$ symbolizes a characteristic $\boldsymbol{h}^t$ pattern with the activation of corresponding expert's $\boldsymbol{k}_j$. **c.** Routers for both FFN and PEFT experts introduce interesting dynamics among their expert vectors, resulting a flexible space for fine-tuning.

mal scaling, sparsity, and routing configurations, and empirically analyzed our findings with insights that generalize across settings and may facilitate better PEFT and MoE applications.

The primary contributions of this paper are:

1. **Dynamics** between experts and routers when applying PEFT to MoE LLMs;
2. **Framework & Strategies** for systematic exploration of PEFT design choices;
3. **Evidence & Guidelines** for the gains of routed adaptation strategies on MoE LLMs.

2 Methodology

We start from investigating how the core components of MoE and PEFT modules interact, which creates new opportunities for designing PEFT on MoE LLMs.

2.1 The Dynamics

For a transformer with L layers, each with attention and a Feed-Forward Network (FFN), given token embeddings $\boldsymbol{x}_0^{1:T} \in \mathbb{R}^{T\times D}$, layer l computes:[2]

$$\boldsymbol{h}_l^{1:T} = \text{SelfAttn}_l\big(\boldsymbol{x}_{l-1}^{1:T}\big) + \boldsymbol{x}_{l-1}^{1:T}, \tag{1}$$

$$\boldsymbol{x}_l^{t} = \text{FFN}_l\big(\boldsymbol{h}_l^{t}\big) + \boldsymbol{h}_l^{t}. \tag{2}$$

Key Memory Vectors. A standard FFN takes form as $\sigma(\boldsymbol{h}\boldsymbol{W}_{\text{up}})\boldsymbol{W}_{\text{down}}$[3], where $\sigma(\cdot)$ represents the activation. Following the key-value memory perspective of Geva et al. (2021), each column $\boldsymbol{k}_i \in \mathbb{R}^D$ in $\boldsymbol{W}_{\text{up}}$ serves as a key memory vector that fires on certain input patterns. Projecting $\boldsymbol{h}^t \in \mathbb{R}^D$ onto these keys yields activation scores $\boldsymbol{a}^t \in \mathbb{R}^{D_a}$ (Figure 1a). These key vectors function as specialized $\boldsymbol{h}^t$ pattern detectors, with their activations determining the subsequent output of the value memory vectors for each token.

Expert Vectors. Scaling up transformers brings redundancy in FFN, with most tokens trigger only a few keys (Elhage et al., 2021). MoE groups key memory vectors into N sparse experts E_i. A router $G(\cdot)$ picks the top-K experts per token:

$$\text{FFN}\big(\boldsymbol{h}^t\big) = \sum_{i=1}^{N} G_i(\boldsymbol{h}^t)\, E_i(\boldsymbol{h}^t), \tag{3}$$

$$G_i(\boldsymbol{h}^t) = \text{TopK}\Big(\text{Softmax}\big(\boldsymbol{h}^t\boldsymbol{W}_g\big)\Big)_i. \tag{4}$$

The router learns its weight matrix $\boldsymbol{W}_g \in \mathbb{R}^{D\times N}$ that can be interpreted as a set of N individual D-dimensional expert vectors $\boldsymbol{g}_i$, each responding to a characteristic hidden state $\boldsymbol{h}_i$ that should activate the corresponding expert E_i (and their key memory vectors) (Zhou et al., 2022), as illustrated in Figure 1b. During training, G dynamically learns which $\boldsymbol{g}_i$ and $\boldsymbol{k}_i$ should better fire together.

PEFT for MoE. A PEFT block $\Delta(\boldsymbol{h}) = \text{UpProj}\,(\text{Act}\,(\text{DownProj}(\boldsymbol{h})))$ mirrors the FFN structure but is much smaller (He et al., 2022), with $\text{Act}(\cdot)$ as non-linear $\sigma(\cdot)$ or identity function in LoRA. Its down-projection contains new keys $\tilde{\boldsymbol{k}}_j$ that respond to task-specific patterns.

When integrating PEFT into MoE, we can choose between several intuitive approaches. A straightforward but limited one is *MoE-agnostic*

[2]LayerNorms and dropout are omitted for clarity.
[3]For alternative FFN structures, see Appendix A.2.

adaptation of individual matrices, which fails to leverage any of the rich dynamics[4] described above. We focuses on the other approach that introduces PEFT module(s) in parallel with FFN experts[5]. This brings additional configurations with intriguing dynamics. A single parallel PEFT module acts as a shared expert that is always active (Dai et al., 2024). Alternatively, we can attach M PEFT experts with their own router $\tilde{G}(\cdot)$ (Figure 1c). The two routers, $\boldsymbol{g}_i$ and $\tilde{\boldsymbol{g}}_j$, can interact so that $\tilde{\boldsymbol{k}}_j$ can either refine existing subspaces or explore new ones. This interaction can substantially enlarge the adaptation space while keeping the backbone frozen.

2.2 The Framework

Based on our insights in §2.1, we examine how PEFT designs can integrate with MoE. As illustrated in Figure 2, we introduce a framework focusing on two key design dimensions: how the adaptation modules operate, and how they interact with MoE's existing expert routing mechanisms.

2.2.1 Functional Strategies

Architecture inside PEFT Experts. Each PEFT expert uses the bottleneck layout in Eq.2.1: $\mathrm{DownProj}(\cdot) : \mathbb{R}^D \mapsto \mathbb{R}^{D_B}$ and $\mathrm{UpProj}(\cdot) : \mathbb{R}^{D_B} \mapsto \mathbb{R}^D$. The bottleneck D_B linearly sets the trainable-parameter budget, like the rank r in LoRA (Hu et al., 2022). It controls the capacity for adaptation and the effectiveness of learning (Hu et al., 2022).

Multiplicity of PEFT Experts. More experts create multiple copies Δ_i, increasing adaptation diversity. Studies on dense models show that adapter count strongly affects performance (Zadouri et al., 2023; Liu et al., 2023a; Dou et al., 2023; Li et al., 2024), and the optimum varies by task, model, and layer (Gao et al., 2024).

Routing among PEFT Experts. The third is whether to add a separate router $\tilde{G}(\cdot)$. Prior MoE-style PEFT targets dense LLMs (Hao et al., 2024; Gao et al., 2024; Wu et al., 2024); our design leverages MoE-specific dynamics (§2.1). Token-wise routing over M PEFT experts mirrors Eq.4:

$$\Delta(\boldsymbol{h}^t) = \sum_{i=1}^{M} \left(\tilde{G}_i \left(\boldsymbol{h}^t\right) \Delta_i(\boldsymbol{h}^t) \right). \quad (5)$$

[4]As Figure 1c, and discussed in §2.2.2 & Appendix A.

[5]As MoE experts run in parallel and prior work shows parallel PEFT works the best (He et al., 2022; Hu et al., 2023; Luo et al., 2024; Hao et al., 2024), we only consider parallel composition of PEFT modules in this study.

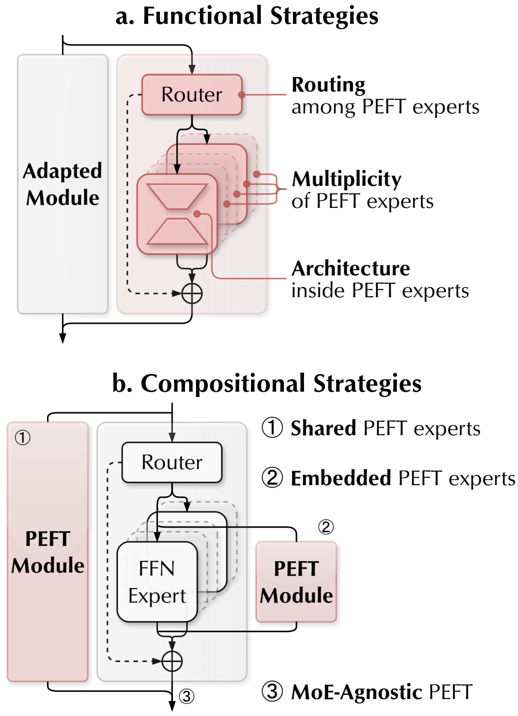


Figure 2: **The framework of how PEFT designs can integrate with an MoE module. a.** Functional strategies specify the internal implementation of the PEFT module introduced. **b.** Compositional strategies describe the PEFT module's interaction with the original MoE mechanism.

2.2.2 Compositional Strategies

Shared PEFT Experts. A single PEFT block can act as a shared expert that runs in parallel with the MoE layer. With input $\boldsymbol{h}^{1:T}$, we have:

$$\begin{aligned} \boldsymbol{x}^{1:T} = & \sum_{i=1}^{N} G_i\left(\boldsymbol{h}^{1:T}\right) E_i\left(\boldsymbol{h}^{1:T}\right) \\ & + \Delta\left(\boldsymbol{h}^{1:T}\right) + \boldsymbol{h}^{1:T}. \end{aligned} \quad (6)$$

The PEFT block sees the same input and adds its output to the residual stream alongside the MoE result. Like shared FFN experts, this block captures common adaptations for all routed experts and can raise parameter efficiency.

Embedded PEFT Experts. Here, each PEFT expert pairs with one FFN expert and receives the same token-wise input from the MoE router:

$$\boldsymbol{x}^t = \sum_{i=1}^{N} G_i(\boldsymbol{h}^t)\left(E_i(\boldsymbol{h}^t) + \Delta_i(\boldsymbol{h}^t)\right) + \boldsymbol{h}^t, \quad (7)$$

where both outputs are weighted by G_i and then added to the residual.

MoE-Agnostic PEFT. MoE-agnostic PEFT treats the model as dense and ignores routing mechanisms. We keep it as a baseline to compare the gains of our MoE-aware designs.

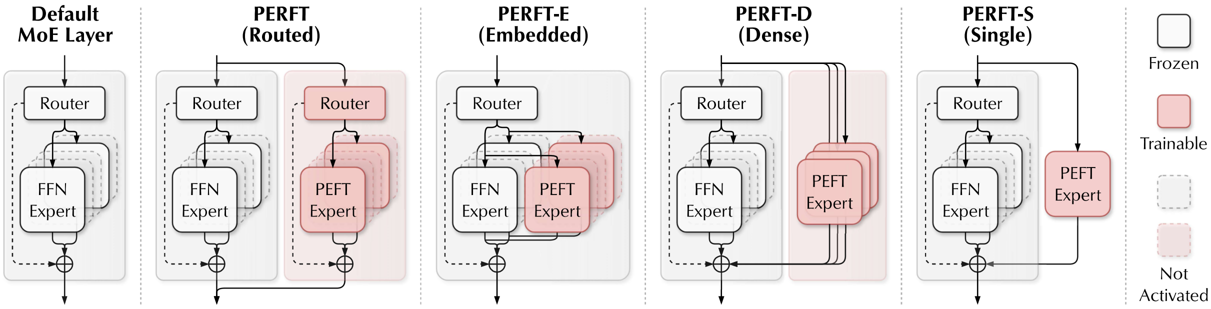


Figure 3: **Illustration of PERFT and its ablated variants.** PERFT holds an independent routing among the introduced PEFT experts. PERFT-E embeds PEFT experts within the original MoE module and directly utilizes its routing patterns. PERFT-D and PERFT-S simply work as independent shared expert(s) alongside the MoE module.

2.3 The Strategies

Within our framework of all meaningful design choices, we implement Parameter-Efficient Routed Fine-Tuning (PERFT), a PEFT strategy tailored for MoE models (Figure 3), whose parallel block owns an independent router:

$$\boldsymbol{x}^{1:T} = \sum_{i=1}^{N} G_i(\boldsymbol{h}^{1:T})\, E_i(\boldsymbol{h}^{1:T}) + \sum_{j=1}^{M} \tilde{G}_j(\boldsymbol{h}^{1:T})\, \Delta_j(\boldsymbol{h}^{1:T}) + \boldsymbol{h}^{1:T}. \tag{8}$$

The new $\tilde{G}(\cdot) : \mathbb{R}^D \mapsto \mathbb{R}^M$ introduces vectors $\tilde{\boldsymbol{g}}_j$ that interact with $\boldsymbol{g}_i$ and enable flexible adaptation, as demonstrated in §2.2.1 and Figure 1c.

If M equals N, we can also reuse the pretrained G and yield the variant **PERFT-E (Embedded)**:

$$\begin{aligned}\boldsymbol{x}^{1:T} &= \sum_{i=1}^{N} G_i(\boldsymbol{h}^{1:T})\, E_i(\boldsymbol{h}^{1:T}) \\ &\quad + \sum_{i=1}^{N} G_i(\boldsymbol{h}^{1:T})\, \Delta_i(\boldsymbol{h}^{1:T}) + \boldsymbol{h}^{1:T} \\ &= \sum_{i=1}^{N} G_i(\boldsymbol{h}^{1:T}) (E_i + \Delta_i)(\boldsymbol{h}^{1:T}) + \boldsymbol{h}^{1:T}.\end{aligned} \tag{9}$$

Our experiments show that reusing G helps when data are too scarce to train a fresh router.

Dropping the routing mechanism and sharing all PEFT experts gives **PERFT-D (Dense):**

$$\boldsymbol{x}^{1:T} = \sum_{i=1}^{N} G_i(\boldsymbol{h}^{1:T})\, E_i(\boldsymbol{h}^{1:T}) + \sum_{j=1}^{M} \Delta_j(\boldsymbol{h}^{1:T}) + \boldsymbol{h}^{1:T}. \tag{10}$$

And further collapsing the M blocks into one yields **PERFT-S (Single):**

$$\boldsymbol{x}^{1:T} = \sum_{i=1}^{N} G_i(\boldsymbol{h}^{1:T})\, E_i(\boldsymbol{h}^{1:T}) + \Delta_0(\boldsymbol{h}^{1:T}) + \boldsymbol{h}^{1:T}. \tag{11}$$

Together, with PERFT and its variants, we can systematically experiment the design choices in our framework and verify if MoE demands a mixture of adaptation modules as expected.

3 Experiments and Analyses

3.1 Experiment Setup

Datasets. We follow the benchmark suite proposed by Hu et al. (2023). It contains 8 commonsense-reasoning datasets and 6 arithmetic-reasoning datasets. We utilize their amalgamated training sets Commonsense170K and Math50K to fine-tune models respectively for each domain. Evaluations are conducted correspondingly across all individual benchmark test sets.

LLM Backbones. We use two open-source MoE LLMs as backbones: OLMoE-1B-7B (Muennighoff et al., 2024) and Mixtral-8×7B (Jiang et al., 2024), selected among publicly available MoE models based on their outstanding performance in the 1B and 10B activated parameter ranges.

Baselines. Applying LoRA to attention matrices $\boldsymbol{W}_q$ and $\boldsymbol{W}_v$ is the most popular PEFT setting under a tight parameter budget (Hu et al., 2022). We therefore adopt it as our primary baseline for all scales and tasks. For the smaller OLMoE-1B-7B, we additionally LoRA-tune the router matrix $\boldsymbol{W}_g$ (results in Table 6, Appendix D).

Additional training details and design choices are provided in Appendix A.

LLM	Arch.	Strategy	#Act.	%Act.	BoolQ	PIQA	SIQA	HellaS	WinoG	ARC-e	ARC-c	OBQA	MultiA	GSM8K	AddSub	AQuA	SingleEq	SVAMP	Avg.*	%Err.↓
OLMoE 1B-7B (Top8/64)	LoRA_2	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	0.26M	0.020	62.02	71.11	59.77	28.48	50.36	70.37	48.89	48.00	20.00	8.72	43.04	20.47	52.95	29.40	43.42	–
	LoRA_8	PERFT(Top1/1)	0.29M	0.023	63.43	77.53	70.68	42.13	66.14	77.10	59.30	66.20	17.00	6.22	34.18	17.32	39.17	30.20	51.59	14.44
	LoRA_4	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	0.52M	0.041	60.40	73.61	62.90	32.08	50.20	74.12	52.65	51.20	21.83	8.11	40.51	20.47	50.79	28.80	45.45	–
	LoRA_8	PERFT(Top2/2)	0.59M	0.046	65.26	78.18	72.31	42.11	71.82	77.90	60.49	67.80	23.33	7.35	51.65	18.50	52.76	33.50	53.26	14.31
	LoRA_8	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	1.05M	0.082	63.76	74.86	65.30	37.01	50.83	76.81	55.46	56.40	17.33	8.57	44.05	24.02	50.59	30.90	48.58	–
	LoRA_{16}	PERFT(Top2/2)	1.11M	0.087	66.18	77.97	72.52	43.99	70.64	78.24	60.75	69.80	26.50	8.49	52.15	20.87	56.69	32.30	54.30	11.12
	LoRA_{16}	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	2.10M	0.164	64.95	76.88	69.60	39.27	53.35	78.07	57.34	63.40	18.83	9.02	46.58	24.02	50.59	29.20	50.52	–
	LoRA_{32}	PERFT(Top2/2)	2.16M	0.169	65.81	79.38	73.59	49.42	71.59	77.78	61.18	71.80	23.67	9.25	44.81	21.65	53.35	35.20	56.46	12.02
	LoRA_4	PERFT-E(Top8/64)	2.10M	0.164	64.80	79.49	74.36	58.39	72.69	75.00	58.45	72.20	26.67	6.44	46.58	22.05	53.94	32.10	59.35	17.85
	LoRA_{32}	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	4.19M	0.327	66.79	78.56	70.93	41.63	58.41	79.38	60.41	65.00	19.17	8.79	43.54	23.23	51.97	28.20	52.31	–
	LoRA_{16}	PERFT(Top4/4)	4.33M	0.337	65.44	79.43	73.08	48.35	71.19	77.48	59.98	73.40	21.50	7.43	45.06	20.87	59.84	30.30	55.71	7.12
	LoRA_8	PERFT-E(Top8/64)	4.19M	0.327	65.81	78.84	73.85	58.84	71.51	74.41	56.06	69.20	28.33	7.81	43.80	21.26	57.28	32.60	59.43	14.93
Mixtral 13B-47B (Top2/8)	LoRA_8	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	3.41M	0.026	73.49	90.04	81.17	89.67	82.16	93.56	83.87	86.20	60.00	50.87	90.13	28.74	89.37	69.20	82.78	–
	LoRA_8	PERFT (Top2/2)	4.46M	0.035	74.68	89.77	81.47	94.33	86.27	92.05	81.48	89.80	82.83	55.80	87.59	29.92	89.76	68.30	85.43	15.39
	LoRA_8	PERFT (Top2/8)	5.24M	0.046	73.76	89.12	81.63	94.51	85.16	91.75	80.89	88.60	79.00	54.06	87.34	29.13	88.98	70.30	85.10	13.46

Table 1: **Evaluation results of baseline and PERFT variants on 8 commonsense and 6 arithmetic reasoning benchmarks.** "Arch." denotes the architecture inside PEFT modules. "#Act." and "%Act." represent the number of activated trainable parameters and their ratio to the total activated parameters. "(TopK/N)" refers to activating K experts among the total number of N experts. Performance is averaged with weight adjusted by benchmark testset sizes. Improvement is measured by reduced error ratio. Dataset names are partially abbreviated, including BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), Social IQa (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), Easy Set and Challenge Set of ARC (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018); MultiArith (Roy and Roth, 2015), GSM8K (Cobbe et al., 2021), AddSub (Hosseini et al., 2014), AQuA (Ling et al., 2017), SingleEq (Koncel-Kedziorski et al., 2015), and SVAMP (Patel et al., 2021).

3.2 Experiment Results

We validate the optimal configurations by exhaustively fine-tuning OLMoE under each configuration. The results are summarized in Figure 4. Table 1 presents a numerical comparison between some best-performing PERFT configurations and MoE-agnostic baselines with equivalent levels of activated trainable parameters. PERFT on average improves by up to 17.2% in commonsense, 12.3% in arithmetic, and 15.5% overall.[6] Appendix D lists full results for each configuration and task.

PERFT outperforms baselines. Our results verified that designing PEFT with considering the underlying MoE mechanisms can indeed achieve better results. Notably, PERFT and its variants yields drastically different performance patterns. PERFT and PERFT-E are the best-performing variants, especially at higher efficiency levels.

PERFT and PERFT-E can benefit from scaling up. Different variants show different scaling performances. PERFT and PERFT-E gain from larger bottleneck sizes D_B within a certain range (shown by bigger markers in Figure 4).

PERFT is more sensitive to overall PEFT expert number rather than activated ratio. Figure 5 isolates the effect of total activated PEFT-expert count and trainable parameter efficiency. When fixing total number, the performance gain from increasing the activated ratio is relatively modest.

Additional results are provided in Appendix D, underlining the effectiveness of balanced expert count, sparsity and computational efficiency in tuning PERFT family.

3.3 Discussion

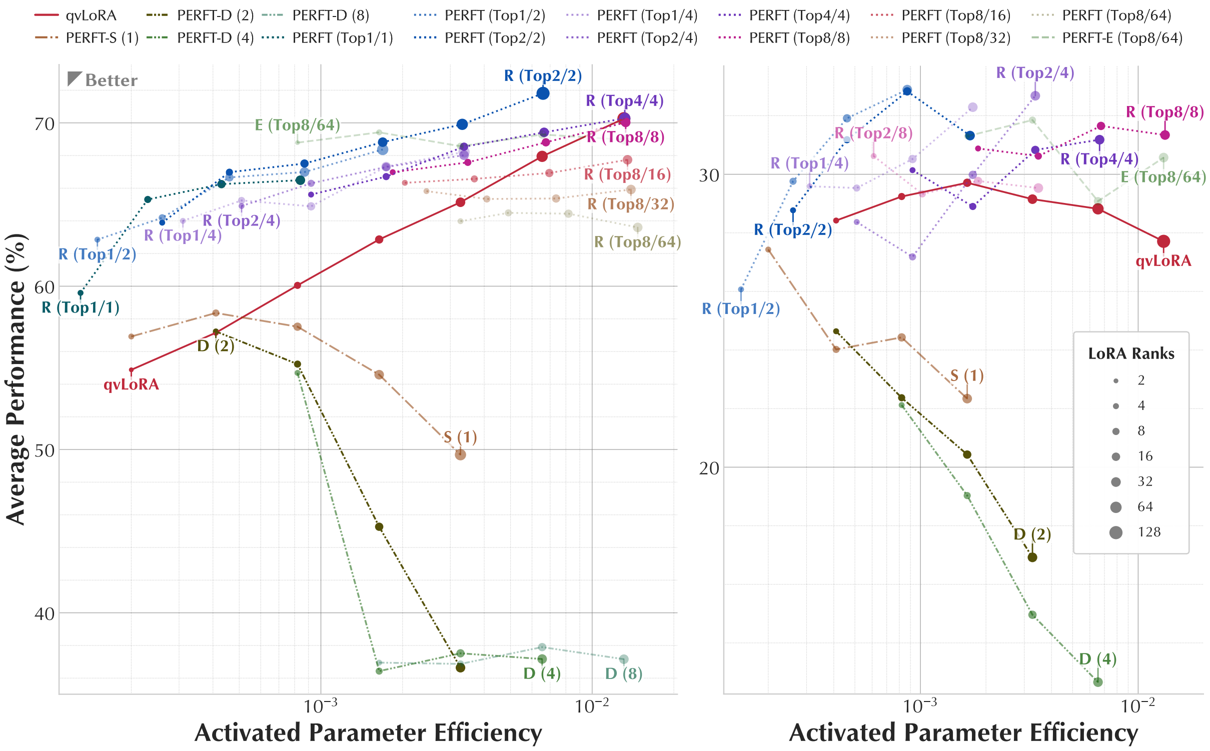
We observe two consistent patterns across all tasks. First, **token-wise routing** among PEFT experts (the PERFT-R family) drives most of the gains and enables extreme parameter efficiency. Second, when the number of PEFT experts is large, **re-using the pretrained MoE router** (PERFT-E) is more stable than training a new router from scratch. Detailed ablations and analyses are provided in Appendix C.

3.3.1 Role of Routing

Across most tasks and budgets, the routed variant PERFT outperforms PERFT-S/D/E, showing that a learnable router is the main driver of PEFT gains. We summarize the advantage in three aspects.

Sparse Activation. Figure 4 shows that PERFT-S/D, which always activate every PEFT block, degrade quickly as the bottleneck widens. This phenomenon stems from inefficient parameter utilization in always-activated shared experts. Section 2.2.1 shows that the bottleneck must balance capacity against learning effectiveness to reach peak performance. PERFT avoids this by activating only

[6] Notice that the reported PERFT-E performs better than PERFT on commonsense reasoning tasks with similar activated trainable parameters, yet this is achieved with much higher total number of trainable parameters, which is intuitive as commonsense reasoning is more knowledge-intensive and benefit from a broader pool of PEFT experts.



(a) Commonsense Reasoning (b) Arithmetic Reasoning

Figure 4: **Performance of OLMoE fine-tuned with baselines and PERFT.** Scores on y-axes are averaged performance across each individual benchmark; Activated Parameter Efficiency on x-axes indicates the ratio of activated trainable parameters to the total activated parameters. "qvLoRA" stands for applying LoRA on attention matrices $\boldsymbol{W}_q$ and $\boldsymbol{W}_v$. Transparency indicates different sparsity levels (ratio of activated PEFT experts).

the few experts whose keys $\tilde{\boldsymbol{k}}_i$ match the token, guided by router vectors $\tilde{\boldsymbol{g}}_i$. Without routing, when the PEFT module's dimensions exceed the intrinsic amount required, the surplus capacity becomes detrimental rather than beneficial.

Weight Distribution. When $\tilde{G}(\cdot)$ is absent, adding more PEFT experts hurts performance: PERFT-D consistently lags behind PERFT-S, and the gap widens as the expert count grows. Even when every PEFT is allowed to fire (TopN/N), PERFT still beats non-routed baselines, confirming that token-wise weights, not mere capacity, lift performance. The router assigns token-wise gating weights, letting the model control how much each expert adapts. This dynamic weighting improves capacity utilization and supports the analysis in §2.2.1. This operates similarly to how Gated Linear Units (GLU) improve FFN layers (Dauphin et al., 2017). Without such a mechanism, the potential benefits of multiple PEFT experts would be counterbalanced by the redundancy across them.

Efficiency. With effective routing, total PEFT capacity module matters more than the number of the activated parameters, enabling highly efficient adaptation. Figure 5 shows that for a fixed total number of PEFT experts, increasing the sparsity by activating fewer PEFT experts does not severely impact performance. Figure 6 supports this result with UMAP projections of $\boldsymbol{k}_i$ and $\boldsymbol{g}_i$ in OLMoE and $\tilde{\boldsymbol{k}}_i$ and $\tilde{\boldsymbol{g}}_i$ in different PERFT variants. Comparing Top2/4 with Top4/4, it confirms that an adequate subset of activated $\tilde{\boldsymbol{k}}_i$ is sufficient to capture the appropriate adaptation space.

3.3.2 Pretrained Routing

The relationship between PERFT-E and PERFT reveals important insights about leveraging pretrained knowledge versus learning new adaptation patterns, as discussed in Section 2.2.2. We notice that the performance between PERFT-E and PERFT can vary in practice, especially when considering scenarios with different activated parameters. Results in Figure 4a show that given the same total number of PEFT experts, PERFT-E outperforms PERFT (Top8/64) across all bottleneck sizes; while many PERFT configurations with fewer experts in turn outperform PERFT-E. Figure 6 illustrates the distinct dynamics between PERFT-E and PERFT. PERFT-E utilizes the frozen $\boldsymbol{g}_i$ in $G(\cdot)$ for FFN experts, while PERFT learns an independent $\tilde{G}(\cdot)$

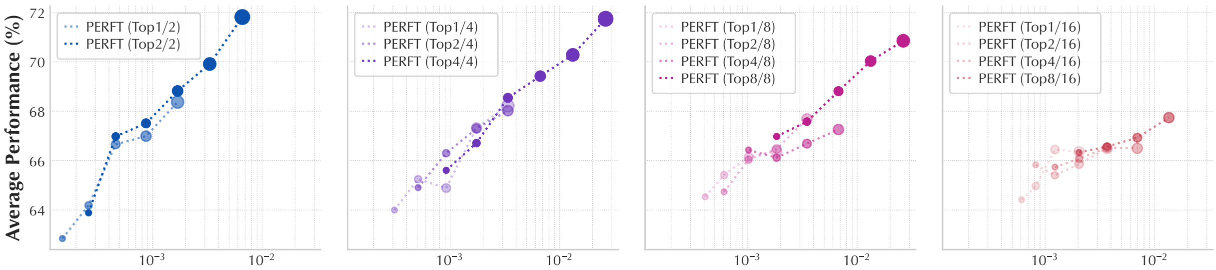


Figure 5: **Performance of PERFT configurations with different total number and activated number of PEFT experts.** Results from OLMoE fine-tuned for commonsense reasoning. x-axes indicate activated parameter efficiency. Transparency represents different sparsity levels. Marker size represents bottleneck size D_B.

from scratch for PEFT experts. These results suggest that when using a larger number of PEFT experts, leveraging the well-pretrained $G(\cdot)$, which already encodes effective patterns for distributing hidden space across FFN experts, would provide more stable and efficient learning for PEFT experts. In contrast, the routed PERFT may expend too much training resources exploring larger subspaces without effectively capturing the optimal distribution patterns for the large number of PEFT experts. This variability highlights the complex trade-off between the flexibility offered by learning new routing mechanisms versus the stability gained from utilizing pretrained components in large-scale models, underscoring the need to consider training configuration- and task-specific factors when choosing between these approaches for large-scale model adaptation.

4 Related Work

4.1 Mixture-of-Experts

MoE was originally introduced as a viable solution to the computational challenges of scaling up and improving specialization (Jacobs et al., 1991; Jordan and Jacobs, 1994; Eigen et al., 2013; Shazeer et al., 2017). With the rise of transformers, researchers observed that FFNs hold the largest share of parameters and capture substantial knowledge (Geva et al., 2021; Dai et al., 2022). This capacity is linked to sparsely represented features in their activations (Dalvi et al., 2019; Durrani et al., 2020; Gurnee et al., 2023). MoE leverages this sparsity by activating only a subset of experts for each input, which improves resource utilization (Liu et al., 2023b). The idea has led to several successful MoE LLMs (Lepikhin et al., 2020; Du et al., 2022; Fedus et al., 2022; Zoph et al., 2022a; Jiang et al., 2024; Dai et al., 2024; Qwen, 2024; Grok, 2024; DeepSeek-AI, 2025). Recent studies explore *shared experts*, modules that run in parallel with routed FFN experts and remain active for every token. This design captures common knowledge and can improve parameter efficiency (Gou et al., 2023; Dai et al., 2024; Qwen, 2024).

4.2 Parameter-Efficient Fine-tuning

Classical full fine-tuning approaches have become increasingly expensive as transformers scale (Devlin et al., 2019; Qiu et al., 2020). Recent work introduce diverse PEFT methods offering comparable performance with significantly reduced computational demands. He et al. (2022) present a unified view for PEFT, where any PEFT method can be viewed as a combination of several design dimensions. This perspective has inspired many hybrid designs. They also show that parallel PEFT modules outperform sequential ones and that modifying FFN is more effective than modifying attention. Later studies confirm these findings (Hu et al., 2023; Zhang et al., 2023; Dettmers et al., 2024; Hao et al., 2024).

Recent success of MoE has sparked MoE-structured PEFT methods. Some insert mixtures of LoRA experts into the attention layers (Liu et al., 2023a; Luo et al., 2024). Others place them next to dense FFNs (Zadouri et al., 2023; Dou et al., 2023; Page-Caccia et al., 2024; Chen et al., 2024; Hao et al., 2024; Li et al., 2024; Wu et al., 2024; Gao et al., 2024). All these studies primarily focus on adapting dense models, which motivates us to investigate designing PEFT modules considering the underlying routing mechanisms of MoE. Recently, Wang et al. (2024) propose expert-specialized fine-tuning as an alternative approach to PEFT, which selectively fine-tunes the most relevant experts for downstream tasks and comes closest to this research gap, although no PEFT techniques are involved and the experts weights are modified. In our exploration of whether MoE LLMs requires

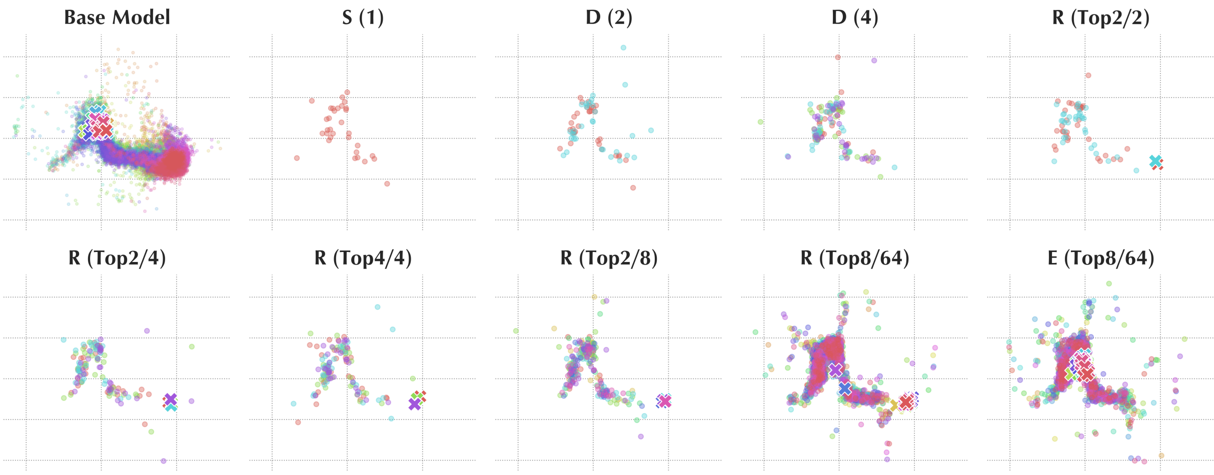


Figure 6: **Visualization of key memory vectors and expert vectors in OLMoE and PERFT fine-tuned for commonsense reasoning.** Results show projections of vectors with $D_B = 32$ from layer 8 of OLMoE. Each subplot corresponds to a different configuration: "Base Model" showing vectors of FFN experts and router in the original MoE layer; "S", "D", "R" and "E" referring to vectors in the PEFT experts and router (if any) of the corresponding PERFT variants. Markers ● represent key memory vectors in FFN or PEFT experts, and ✖ expert vectors in routers for either FFN experts (in Base Model and PERFT-E) or PEFT experts (in PERFT). All vectors are projected using the same PCA and UMAP trained on FFN experts' key memory vectors. Different colors distinguish vectors associated with different indices.

mixture of adaptation modules, we directly consider introducing PEFT modules for MoE LLMs, offering more flexible and efficient solutions while preserving the original weights untouched.

5 Conclusion

This study addresses the gap in efficiently adapting MoE LLMs to downstream tasks. We investigate the dynamics of core components when performing PEFT for MoE. Building on these insights, we introduce a unified framework with a comprehensive set of design dimensions. We further propose a flexible family of PEFT strategies tailored for MoE modules. Extensive experiments on OLMoE and Mixtral, covering commonsense and arithmetic reasoning, show that our methods outperform MoE-agnostic baselines in both effectiveness and scalability. We identify the optimal configuration for each design dimension and analyze the results. These observations provide practical guidance for future PEFT and MoE applications.

Limitations

Model scale and hardware assumptions. Due to computational resource constraints, all experiments in this study are performed on *OLMoE-1B-7B* and *Mixtral-8×7B*, i.e., MoE backbones with *activated* parameter counts in the 1B–10B range (see §3.1). It remains unclear whether the efficiency–quality trade-offs of the PERFT family generalize to substantially larger or smaller models, or to environments with limited hardware resources such as edge GPUs or CPUs. As efficiency metrics, we report the number of activated trainable parameters and their ratio to total activated parameters (activated parameter efficiency), since these directly correspond to FLOPs and memory usage, which can be precisely derived from the chosen sparsity pattern. We do not report other system-specific efficiency measures, such as inference latency, throughput, or energy costs, as these can vary significantly across hardware and device configurations, and in most cases these measures are approximately proportional to the reported parameter-based metrics.

Mergeability and inference cost. In PERFT, a key difference from conventional PEFT methods such as LoRA is that adapters cannot be merged into the backbone weights for static inference because routing is input-dependent, which raises concerns about increased inference overhead. However, the additional cost remains modest and controllable. The per-token latency and memory overheads from routing and extra adapters is marginal compared to total MoE computation cost, since only a small (<1%) bottleneck is added and the number of activated PEFT experts per token is fixed as Top-K. Meanwhile, PERFT-E allows eliminating the routing overhead entirely by reusing the pretrained

MoE router. Overall, the ability to adapt in a MoE-aware manner brings consistent efficiency gains at matched activated parameter efficiency.

Hyperparameter search cost. Determining the optimal bottleneck size D_B, number of PEFT experts M, and routing sparsity K/N required a comprehensive grid search (§3.2, Figures 4b, 5). Once selected, these configurations generalized well across all evaluated tasks. However, replicating such a search may be impractical for practitioners with limited computational resources. Developing adaptive or automated hyperparameter tuning strategies could address this challenge.

Task coverage. Our evaluation focuses on 14 widely used English benchmarks: 8 commonsense reasoning and 6 arithmetic reasoning datasets (Tables 1, 6 - 10). This scope allows us to test design predictions from our formalization without confounds from task-specific prompting/decoding, in order to isolate how routing and adapter multiplicity interact in MoE under controlled settings. The gains may not directly transfer to other domains such as language generation, code synthesis, safety, multilingual, low-resource, or scenarios under distribution shift with adversarial or noisy inputs.

Bias and societal risk. While our experiments utilize 2 open-source MoE LLMs and 14 widely adopted English academic benchmarks commonly used in a broad body of peer-reviewed research, they are nonetheless limited in scope and may not capture the full range of biases present in real-world applications. Both the underlying MoE backbone LLMs and the fine-tuning datasets could reflect demographic or geographic biases originating from large-scale web corpora. We did not undertake a dedicated bias or robustness audit (e.g., evaluating accuracy by gender, ethnicity, or language variety), nor did we assess privacy leakage or data memorization in this study, and we stress that the absence of such analyses means that potential fairness, privacy, or safety issues may persist. It is incumbent upon downstream users and deployers to conduct appropriate task-specific fairness, privacy, and safety evaluations before considering any real-world deployment. We expressly disclaim responsibility for any unintended consequences arising from the use or deployment of models fine-tuned by our methods.

These limitations highlight promising directions for extending the current study and for responsibly deploying PEFT techniques on sparse MoE LLMs.

Acknowledgments

The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center of the Friedrich-Alexander-Universität Erlangen-Nürnberg (NHR@FAU) under the NHR project b207dd, and by the Karlsruhe Institute of Technology National High Performance Computing Center (NHR@KIT) under the NHR projects 22189 and 22560. The funding for NHR@FAU is provided by federal and Bavarian state authorities, and the hardware is partially funded by the German Research Foundation (DFG) – 440719683. The HoreKa supercomputer at NHR@KIT is funded by the Ministry of Science, Research and the Arts Baden-Württemberg and by the Federal Ministry of Education and Research of Germany. This work is also partially supported by funding from Munich Center for Machine Learning (MCML).

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. 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 7319–7328.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Shaoxiang Chen, Zequn Jie, and Lin Ma. 2024. Llava-mole: Sparse mixture of lora experts for mitigating data conflicts in instruction finetuning mllms. *arXiv preprint arXiv:2401.16160*.

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

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias

Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, and 1 others. 2024. Deepseek-moe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proc. of ACL*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James R. Glass. 2019. What is one grain of sand in the desert? analyzing individual neurons in deep NLP models. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6309–6317. AAAI Press.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL-HLT*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

S Dou, E Zhou, Y Liu, S Gao, J Zhao, W Shen, Y Zhou, Z Xi, X Wang, X Fan, and 1 others. 2023. Loramoe: Alleviate world knowledge forgetting in large language models via moe-style plugin. *arXiv preprint arXiv:2312.09979*.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, and 1 others. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR.

Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and Yonatan Belinkov. 2020. Analyzing individual neurons in pre-trained language models. In *Proc. of EMNLP*, pages 4865–4880, Online. Association for Computational Linguistics.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. 2013. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, and 1 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and VS Subrahmanian. 2024. Higher layers need more lora experts. *arXiv preprint arXiv:2402.08562*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proc. of EMNLP*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yunhao Gou, Zhili Liu, Kai Chen, Lanqing Hong, Hang Xu, Aoxue Li, Dit-Yan Yeung, James T Kwok, and Yu Zhang. 2023. Mixture of cluster-conditional lora experts for vision-language instruction tuning. *arXiv preprint arXiv:2312.12379*.

Grok. 2024. Open release of grok-1.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing. *ArXiv preprint*, abs/2305.01610.

Jitai Hao, WeiWei Sun, Xin Xin, Qi Meng, Zhumin Chen, Pengjie Ren, and Zhaochun Ren. 2024. Meft: Memory-efficient fine-tuning through sparse adapter. *arXiv preprint arXiv:2406.04984*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *Proc. of ICLR*. OpenReview.net.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019.

Parameter-efficient transfer learning for NLP. In *Proc. of ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *Proc. of ICLR*. OpenReview.net.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*.

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

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, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Michael I Jordan and Robert A Jacobs. 1994. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhiyuan Cheng, Lei Duan, Jie Zuo, Cal Yang, and Mingjie Tang. 2024. Mixlora: Enhancing large language models fine-tuning with lora based mixture of experts. *arXiv preprint arXiv:2404.15159*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. 2023a. Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications. *arXiv preprint arXiv:2310.18339*.

Zeyu Liu, Tim Dettmers, Xi Lin, Veselin Stoyanov, and Xian Li. 2023b. Towards a unified view of sparse feed-forward network in pretraining large language model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15038–15061.

Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. 2024. Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models. *arXiv preprint arXiv:2402.12851*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, and 1 others. 2024. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*.

Lucas Page-Caccia, Edoardo Maria Ponti, Zhan Su, Matheus Pereira, Nicolas Le Roux, and Alessandro Sordoni. 2024. Multi-head adapter routing for cross-task generalization. *Advances in Neural Information Processing Systems*, 36.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China technological sciences*, 63(10):1872–1897.

Qwen. 2024. Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Zihan Wang, Deli Chen, Damai Dai, Runxin Xu, Zhuoshu Li, and Y Wu. 2024. Let the expert stick to his last: Expert-specialized fine-tuning for sparse architectural large language models. *arXiv preprint arXiv:2407.01906*.

Xun Wu, Shaohan Huang, and Furu Wei. 2024. Mixture of lora experts. *arXiv preprint arXiv:2404.13628*.

Greg Yang and Edward J Hu. 2020. Feature learning in infinite-width neural networks. *arXiv preprint arXiv:2011.14522*.

Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. 2023. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. *arXiv preprint arXiv:2309.05444*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*. Openreview.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, and 1 others. 2022. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022a. Designing effective sparse expert models. *arXiv preprint arXiv:2202.08906*, 2(3):17.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022b. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*.

Hyperparameters	OLMoE-1B-7B	Mixtral-8×7B
Training precision	BFloat16	
Dropout	0.05	
Optimizer	AdamW	
LR	1e-5	2e-5
LR scheduler	Linear	
Batch size	16	
Warmup steps	100	
Epochs	3	
Auxiliary loss coef.	0.01	0.02

Table 2: **Hyperparameter configurations for OLMoE-1B-7B and Mixtral-8×7B.**

A Additional Experiment Setup and Discussions

A.1 Training Configurations.

Hardware. For each experiment we trained OLMoE-1B-7B on a single NVIDIA A100 GPU. Mixtral-8×7B was trained on our 4×NVIDIA H100 GPUs connected with NV-link. Both models are evaluated on NVIDIA A100 GPUs.

Hyperparameters. We display the hyperparameter configurations used in fine-tuning and evaluating OLMoE-1B-7B and Mixtral-8×7B in Table 2. We use the LoRA settings recommended by Hu et al. (2023) and keep all other hyperparameters at their model-default values.

Loss Functions. In our experiments, we maintain consistency with the original training process of each LLM by incorporating their respective auxiliary losses alongside the cross-entropy loss for token outputs. All evaluated models include a load-balancing loss, which encourages an equal token distribution among experts (Shazeer et al., 2017). OLMoE-1B-7B additionally incorporates a router z-loss to penalize large routing logits and stabilize training (Zoph et al., 2022b). To ensure a fair comparison, we keep all auxiliary losses active during fine-tuning for baseline and all PERFT variants. For PERFT, we extend this approach with the load balancing loss for the PEFT expert router as well for a similar balanced distribution of tokens among PEFT experts. Detailed hyperparameters and resource configurations for our experiments are provided in Table 2.

A.2 Gated Linear Unit

Modern transformers often adopt the Gated Linear Unit (GLU), which adds an element-wise multiplicative gate after activation (Dauphin et al., 2017; Shazeer, 2020). Formally: $\mathrm{FFN}_{\mathrm{GLU}}(\boldsymbol{h}) = [\sigma(\boldsymbol{h}\boldsymbol{W}_{\mathrm{gate}}) \otimes (\boldsymbol{h}\boldsymbol{W}_{\mathrm{up}})]\boldsymbol{W}_{\mathrm{down}}$. We focus on the

Hypothesis	Mean Δ	95% CI	p-value
H1: Weighted aggregate	5.77%	[3.67%, 6.84%]	<0.001
H2: Unweighted aggregate	3.90%	[2.79%, 5.03%]	<0.001
H3a: OLMoE-1B-7B	6.71%	[4.12%, 8.07%]	<0.001
H3b: Mixtral-8×7B	2.48%	[0.60%, 3.75%]	0.013

Table 3: **Aggregate hypothesis statistics.** H1 tests whether PERFT and variants generally improve over LoRA baselines using dataset-size-weighted aggregation (primary hypothesis). H2 verifies robustness using unweighted aggregation. H3a and H3b test model-specific improvements. All tests use one-sided permutation testing with bootstrap 95% confidence intervals. Each hypothesis pools multiple configurations to test the general efficacy of routing rather than individual configuration performance.

matrix $\boldsymbol{W}_{\text{gate}}$ since it directly processes h and its output passes through $\sigma(\cdot)$, which controls key-memory activation. The same argument applies to both vanilla FFN and GLU variants.

B Statistical Significance Analysis

To test whether routed PEFT modules improve over MoE-agnostic baselines, we formulated four pre-specified hypotheses pooling configurations across 14 benchmarks. This aggregate framework evaluates the general efficacy of routing while avoiding multiple comparison penalties. For each comparison, we computed weighted mean improvements across benchmarks (weights proportional to test set sizes), then tested whether the mean significantly exceeds zero using one-sided permutation testing with bootstrap 95% confidence intervals.

For the major results presented in Table 1, we tested four hypotheses: **H1** (primary): PERFT and variants improve over LoRA baselines using weighted aggregation across all 9 configurations (7 on OLMoE-1B-7B, 2 on Mixtral-8×7B). **H2** (sensitivity): improvement using unweighted aggregation to verify robustness. **H3a** and **H3b**: model-specific improvements on OLMoE-1B-7B and Mixtral-8×7B respectively.

Table 3 presents the overall results for all hypotheses. Our primary hypothesis (H1) confirms significant improvement over LoRA baselines ($p < 0.001$), with the sensitivity analysis (H2) yielding consistent results. Model-specific tests reveal stronger improvements on OLMoE-1B-7B (H3a: $p < 0.001$) compared to Mixtral-8×7B (H3b: $p = 0.013$). All four hypotheses achieve statistical significance ($\alpha = 0.05$), with H1, H2, and H3a reaching $p < 0.001$. The effect is robust across diverse configurations, evaluation domains, and analytical choices. Table 4 provides detailed per-comparison statistics for reference.

C Additional Analyses for Design Configurations

C.1 Architecture inside PEFT Experts

LoRA Versus Parallel Adapters. We centre our study on LoRA adapters because they are simple yet effective. Output scaling with α also reduces the need to retune hyperparameters when the bottleneck size changes (Yang and Hu, 2020; Hu et al., 2022). Motivated by results on dense models (He et al., 2022; Hu et al., 2023), we also analyze *parallel adapters* (Houlsby et al., 2019; He et al., 2022), which add an activation after the bottleneck.

Table 5 compares the commonsense reasoning performance of LoRA and Parallel Adapters (PA) as PEFT experts in OLMoE-1B-7B with several well-performing PERFT configurations. As we can see, under equivalent activated trainable parameter levels, the average performance difference between LoRA and PA is only marginal. Interestingly, on specific tasks, certain architectures consistently outperform others. For instance, parallel adapters generally perform better on BoolQ, PIQA, and ARC, while LoRA excels in SIQA and OBQA. These task-specific gaps may reflect differences in required knowledge or data distribution. A deeper investigation into these task-specific variations is beyond the scope of this study. Given the similar average performance, we opted to focus on LoRA for our experiments due to its simpler structure without the additional activation function.

It is also viable to consider copying the original FFN structure as PEFT experts. We have opted not to investigate this option further in our current study based on two reasons. First, copying the full FFN violates the spirit of PEFT because it effectively upsizes the model to a version with more experts. Second, recent advancements have introduced more complex implementations that go beyond the simple $\sigma(\boldsymbol{h}\boldsymbol{W}_{\text{up}})\boldsymbol{W}_{\text{down}}$ pattern how FFN was initially designed as. GLU has become widely adopted in modern transformers including OLMoE-1B-7B and Mixtral-8×7B. The increased complexity of GLU, with its three matrices, presents challenges for a fair controlled comparison under the same parameter budget. Given these considerations, we focus on experimenting within our current scope.

Bottleneck Sizes. We experiment with different

PERFT Variant		Baseline		Abs. Δ	%Err.↓	95% CI (Rel)	Perm. p
OLMoE-1B-7B							
$LoRA_8$	PERFT(Top1/1)	$LoRA_2$	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	8.17%	14.44%	[4.24%, 19.14%]	0.0112
$LoRA_8$	PERFT(Top2/2)	$LoRA_4$	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	7.80%	14.31%	[8.91%, 19.85%]	<0.001
$LoRA_{16}$	PERFT(Top2/2)	$LoRA_8$	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	5.72%	11.12%	[6.67%, 16.96%]	<0.001
$LoRA_{32}$	PERFT(Top2/2)	$LoRA_{16}$	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	5.95%	12.02%	[3.95%, 16.28%]	<0.001
$LoRA_4$	PERFT-E(Top8/64)	$LoRA_{16}$	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	8.83%	17.85%	[1.04%, 26.76%]	0.0319
$LoRA_{32}$	PERFT(Top2/4)	$LoRA_{32}$	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	3.40%	7.12%	[-0.12%, 11.02%]	0.0651
$LoRA_8$	PERFT-E(Top8/64)	$LoRA_{32}$	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	7.12%	14.93%	[-2.42%, 24.08%]	0.1347
Mixtral-8×7B							
$LoRA_8$	PERFT(Top2/2)	$LoRA_8$	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	2.65%	15.39%	[0.67%, 29.40%]	0.0498
$LoRA_8$	PERFT(Top2/8)	$LoRA_8$	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	2.32%	13.46%	[-1.81%, 28.96%]	0.1062

Table 4: **Individual comparison statistics (exploratory).** These individual tests are presented for descriptive purposes; the primary statistical results stay significant as presented in Table 3.

bottleneck sizes ranging from 2 to 128. Here we provide a detailed empirical analysis about the inefficient parameter utilization when always-activated shared experts are employed without an effective routing mechanism. Such cases reveal a mismatch between task dimensionality and adapter capacity. When the bottleneck is too wide, the extra dimensions add little signal and can even hurt performance. Large, randomly-initialized bottlenecks in PERFT-S or PERFT-D inject noise into otherwise unused subspaces and may corrupt pretrained representations. If the residual stream is viewed as limited bandwidth between modules (Elhage et al., 2021), then only a small subspace should carry task-specific adaptation when most weights stay frozen. Any over-parameterized adaptation can unnecessarily disrupt normal functioning on the residual stream's bandwidths, potentially destabilizing the original gradient flow in the transformer and leading to unstable training or sub-optimal solutions (Aghajanyan et al., 2021). Simultaneously, in the PEFT context with limited adaptation information compared to model pretraining, an excessively large parameter space without gating control can easily result in over-fitting on fine-tuning data, which is exacerbated by the sparse nature of the MoE module we are adapting. As the MoE module hosts multiple different patterns on various combinations of activated FFN experts that dynamically interact with each other on the residual stream, the always-activated PERFT-S and PERFT-D variants may learn unnecessary adaptations during the training process, further aggravating the disrupted functionality and over-fitting problems.

It is also worth noting that since FFN tends to learn task-specific textual patterns (Geva et al., 2021) and attention learns more about positional interactions (Elhage et al., 2021), the nature of different components to which PEFT is introduced also contributes to different phenomena. For the baseline LoRA operating on attention matrices, individual attention heads are already operating on relatively smaller subspaces and can easily write outputs to disjoint subspaces without interaction. Because each attention head operates in a low-rank subspace, its read/write patterns are relatively fixed. Consequently, additional parameters introduced by scaling the bottleneck of attention LoRA may not interfere with information from other components as severely as adapting the MoE FFN module.

C.2 Multiplicity of PEFT Experts

We vary the total number of PEFT experts from 1 to 64 and the number of activated experts from 1 to 8. This grid lets us study how expert count and activation ratio affect performance. We denote K out of M routed PEFT experts activated per token as "(TopK/M)", and N shared PEFT experts without routing as "(N)".

Our observations reveal that naively scaling up the number of experts without a routing mechanism leads to severe performance degradation. Consistently, PERFT-D underperforms PERFT-S, with performance declining as the number of PEFT experts increases. Figure 6 visualizes this effect through UMAP projections of key memory vectors and expert vectors. In an ideal adaptation scenario, PEFT expert key vectors that may activate simultaneously should be distributed evenly within subspaces formed by task-relevant FFN experts' key vectors, maximizing hidden space utilization. However, PERFT-D variants exhibit tightly clustered key vectors from different experts (shown with different colors), indicating functional redundancy and inefficient use of model capacity.

Without routing to specialize different experts

Arch.	Strategy	# Act.	% Act.	BoolQ	PIQA	SIQA	HellaS	WinoG	ARC-e	ARC-c	OBQA	Avg.
LoRA$_4$	PERFT (Top1/1)	0.16M	0.013	62.48	75.73	68.17	25.16	51.07	76.81	55.72	61.60	59.59
PA$_4$	PERFT (Top1/1)	0.16M	0.013	63.09	76.50	64.94	31.23	52.72	77.02	56.31	55.40	59.65
LoRA$_8$	PERFT (Top1/1)	0.29M	0.023	63.43	77.53	70.68	42.13	66.14	77.10	59.30	66.20	65.31
PA$_8$	PERFT (Top1/1)	0.29M	0.023	65.63	78.94	68.68	40.46	53.75	79.25	56.14	61.20	63.01
LoRA$_{16}$	PERFT (Top1/1)	0.56M	0.043	64.98	78.56	72.52	41.99	67.25	77.82	58.70	68.20	66.25
PA$_{16}$	PERFT (Top1/1)	0.56M	0.043	66.61	78.56	71.34	41.26	59.75	78.87	59.30	66.20	65.24
LoRA$_{32}$	PERFT (Top1/1)	1.08M	0.084	66.36	78.84	72.36	42.83	63.38	78.62	58.36	71.20	66.49
PA$_{32}$	PERFT (Top1/1)	1.08M	0.084	66.61	79.54	72.62	42.36	66.46	79.29	62.03	67.40	67.04
LoRA$_4$	PERFT (Top2/2)	0.33M	0.026	64.86	76.71	69.60	40.89	62.43	77.23	55.80	63.60	63.89
PA$_4$	PERFT (Top2/2)	0.33M	0.026	65.44	77.48	69.40	41.14	51.54	78.83	57.94	63.20	63.12
LoRA$_8$	PERFT (Top2/2)	0.59M	0.046	65.26	78.18	72.31	42.11	71.82	77.90	60.49	67.80	66.98
PA$_8$	PERFT (Top2/2)	0.59M	0.046	67.31	80.03	71.14	41.70	61.80	78.58	58.87	66.60	65.75
LoRA$_{16}$	PERFT (Top2/2)	1.11M	0.087	66.18	77.97	72.52	43.99	70.64	78.24	60.75	69.80	67.51
PA$_{16}$	PERFT (Top2/2)	1.11M	0.087	66.76	79.38	72.47	43.52	69.85	80.85	61.26	71.00	68.14
LoRA$_{32}$	PERFT (Top2/2)	2.16M	0.169	65.81	79.38	73.59	49.42	71.59	77.78	61.18	71.80	68.82
PA$_{32}$	PERFT (Top2/2)	2.16M	0.169	67.61	80.96	73.18	45.57	70.64	80.68	61.18	72.00	68.98
LoRA$_4$	PERFT (Top2/4)	0.66M	0.051	63.98	75.68	69.29	40.26	65.75	77.36	59.56	67.40	64.91
PA$_4$	PERFT (Top2/4)	0.66M	0.051	65.93	77.75	69.96	40.81	61.09	79.17	58.28	65.80	64.85
LoRA$_8$	PERFT (Top2/4)	1.18M	0.092	65.02	77.86	71.90	41.61	68.75	77.31	59.13	68.80	66.30
PA$_8$	PERFT (Top2/4)	1.18M	0.092	64.40	78.07	71.24	41.80	70.17	79.76	61.09	67.80	66.79
LoRA$_{16}$	PERFT (Top2/4)	2.23M	0.174	64.07	76.61	73.59	42.10	71.90	78.32	60.58	71.20	67.30
PA$_{16}$	PERFT (Top2/4)	2.23M	0.174	65.99	79.92	72.62	43.14	61.64	80.09	60.58	69.20	66.65
LoRA$_{32}$	PERFT (Top2/4)	4.33M	0.337	66.30	77.75	75.44	45.88	71.43	76.18	60.58	70.60	68.02
PA$_{32}$	PERFT (Top2/4)	4.33M	0.337	66.70	79.33	73.18	42.57	70.40	81.10	62.20	70.60	68.26

Table 5: **Commonsense reasoning performance of OLMoE with PERFT using LoRA and Parallel Adapter (PA).** "Arch." denotes the architecture inside PEFT modules. "# Act." and "% Act." represent the number of activated trainable parameters and their ratio to the total activated parameters. "(TopK/N)" refers to activating K experts among the total number of N experts. Dataset names are partially abbreviated, including BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), Social IQa (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), Easy Set and Challenge Set of ARC (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018).

on different input patterns, all PEFT experts receive identical gradients for each token, causing their parameters to converge toward similar solutions. This redundancy means that adding more experts provides diminishing returns—the effective capacity grows sublinearly with expert count while computational cost grows linearly. The clustering observed in Figure 6 directly reflects this: key memory vectors from different PERFT-D experts occupy nearly identical regions of the representation space, confirming they have learned redundant transformations.

C.3 Routing among PEFT Experts

We investigate both learned routing (PERFT) and embedded routing using the pretrained MoE router (PERFT-E). We also include non-routed variants (PERFT-D/S) for comparison. This allows us to systematically study the impact of parameter efficiency on performance across PERFT variants.

Comparing PERFT to PERFT-S and PERFT-D in Figure 4, we observe that even when all experts are activated (TopN/N), PERFT still improves performance significantly by introducing learnable token-wise gating weights that dynamically assign importance to each expert's output. This mechanism parallels how Gated Linear Units (GLU) improve FFN layers in transformers (Shazeer, 2020), since the gating provides input-dependent modulation that improves capacity utilization. Figure 6 shows that gating weights lead to more balanced vector distribution and more effective utilization of hidden space. The router learns to specialize different PEFT experts on different input patterns, breaking the symmetry that causes redundancy in PERFT-D. Without such a mechanism, the potential benefits of increased expert count are counterbalanced by capacity redundancy.

For a fixed total number of PEFT experts, increasing sparsity by activating fewer experts does not severely degrade performance, as shown in Figure 5. This is also supported by Figure 6, which suggests that an adequate subset of activated expert vectors is sufficient to capture the distribution of the adaptation space. Key memory vectors from different PEFT experts that appear clustered can be utilized by a sparser router to ensure they are not activated simultaneously, maintaining performance while reducing computation. This finding indicates that total PEFT module capacity may be more critical than activated capacity for determining performance.

D Additional Results

D.1 OLMoE-1B-7B for Commonsense Reasoning

Arch.	Strategy	# Act.	% Act.	BoolQ	PIQA	SIQA	HellaS	WinoG	ARC-e	ARC-c	OBQA	Avg.
Base	(pretrained)	—	—	42.42	52.61	16.53	21.27	28.10	13.13	13.99	6.80	24.36
Base	(instruct)	—	—	59.94	62.68	12.03	22.27	5.84	15.15	17.15	8.00	25.38
LoRA$_2$	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	0.26M	0.020	62.02	71.11	59.77	28.48	50.36	70.37	48.89	48.00	54.88
LoRA$_4$	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	0.52M	0.041	60.40	73.61	62.90	32.08	50.20	74.12	52.65	51.20	57.15
LoRA$_8$	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	1.05M	0.082	63.76	74.86	65.30	37.01	50.83	76.81	55.46	56.40	60.05
LoRA$_{16}$	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	2.10M	0.164	64.95	76.88	69.60	39.27	53.35	78.07	57.34	63.40	62.86
LoRA$_{32}$	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	4.19M	0.327	66.79	78.56	70.93	41.63	58.41	79.38	60.41	65.00	65.14
LoRA$_{64}$	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	8.39M	0.654	67.13	80.30	73.34	44.28	65.90	80.72	61.95	70.00	67.95
LoRA$_{128}$	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	16.8M	1.309	68.32	82.64	74.16	45.71	72.45	81.36	63.82	73.60	70.26
LoRA$_4$	$\boldsymbol{W}_g$@Gate	0.14M	0.011	62.14	59.79	39.66	25.94	51.62	42.63	36.52	29.00	43.41
LoRA$_8$	$\boldsymbol{W}_g$@Gate	0.27M	0.021	59.11	66.49	47.59	27.37	51.70	52.06	42.06	33.20	47.45
LoRA$_{16}$	$\boldsymbol{W}_g$@Gate	0.54M	0.042	62.05	64.04	47.85	28.08	49.33	57.37	43.17	34.40	48.29
LoRA$_{32}$	$\boldsymbol{W}_g$@Gate	1.08M	0.084	59.24	60.07	43.19	26.62	49.09	41.50	32.34	31.60	42.96
LoRA$_4$	PERFT-S (1)	0.26M	0.020	63.82	72.31	63.87	25.45	50.12	73.91	49.49	56.40	56.92
LoRA$_8$	PERFT-S (1)	0.52M	0.041	63.52	73.56	66.33	25.45	51.93	72.60	52.47	61.00	58.36
LoRA$_{16}$	PERFT-S (1)	1.05M	0.082	63.49	71.71	65.71	25.11	51.22	71.13	50.60	61.20	57.52
LoRA$_{32}$	PERFT-S (1)	2.10M	0.164	62.08	68.28	64.69	25.37	52.17	64.73	44.54	54.80	54.58
LoRA$_{64}$	PERFT-S (1)	4.19M	0.327	61.59	63.76	59.11	24.48	54.06	53.75	36.86	43.80	49.68
LoRA$_4$	PERFT-D (2)	0.52M	0.041	62.14	71.87	66.53	25.41	51.07	72.60	50.43	57.80	57.23
LoRA$_8$	PERFT-D (2)	1.05M	0.082	62.87	71.44	63.41	25.47	51.70	65.28	46.84	54.80	55.23
LoRA$_{16}$	PERFT-D (2)	2.10M	0.164	62.14	59.68	46.98	25.51	49.25	45.96	33.45	39.20	45.27
LoRA$_{32}$	PERFT-D (2)	4.19M	0.327	62.17	48.20	32.86	25.38	48.86	24.87	25.17	25.60	36.64
LoRA$_4$	PERFT-D (4)	1.05M	0.082	62.87	69.37	61.98	24.93	50.91	65.78	46.08	55.60	54.69
LoRA$_8$	PERFT-D (4)	2.10M	0.164	62.17	49.29	33.06	24.57	49.57	25.46	25.09	22.20	36.43
LoRA$_{16}$	PERFT-D (4)	4.19M	0.327	62.17	50.60	33.21	24.67	48.78	26.01	24.74	30.00	37.52
LoRA$_{32}$	PERFT-D (4)	8.39M	0.654	62.17	52.18	33.47	25.02	50.51	25.80	22.18	26.00	37.17
LoRA$_4$	PERFT-D (8)	2.10M	0.164	62.11	48.86	35.11	24.57	48.22	25.51	23.38	27.80	36.94
LoRA$_8$	PERFT-D (8)	4.19M	0.327	62.17	49.13	33.27	25.37	49.41	25.00	24.23	26.40	36.87
LoRA$_{16}$	PERFT-D (8)	8.39M	0.654	62.17	52.01	33.47	24.91	53.20	25.29	26.96	25.20	37.90
LoRA$_{32}$	PERFT-D (8)	16.8M	1.309	62.17	50.92	33.88	24.58	49.64	24.16	26.71	25.20	37.16
LoRA$_4$	PERFT (Top1/1)	0.16M	0.013	62.48	75.73	68.17	25.16	51.07	76.81	55.72	61.60	59.59
LoRA$_8$	PERFT (Top1/1)	0.29M	0.023	63.43	77.53	70.68	42.13	66.14	77.10	59.30	66.20	65.31
LoRA$_{16}$	PERFT (Top1/1)	5.57M	0.043	64.98	78.56	72.52	41.99	67.25	77.82	58.70	68.20	66.25
LoRA$_{32}$	PERFT (Top1/1)	1.08M	0.084	66.36	78.84	72.36	42.83	63.38	78.62	58.36	71.20	66.49
LoRA$_4$	PERFT (Top1/2)	0.20M	0.015	63.67	77.04	69.09	39.92	58.09	76.81	55.80	62.40	62.85
LoRA$_8$	PERFT (Top1/2)	0.33M	0.026	63.98	78.13	70.93	41.00	58.88	78.11	56.66	65.80	64.19
LoRA$_{16}$	PERFT (Top1/2)	0.59M	0.046	65.14	76.93	72.42	41.39	70.64	78.03	59.56	69.20	66.66
LoRA$_{32}$	PERFT (Top1/2)	1.11M	0.087	65.60	78.18	73.13	43.47	69.61	77.40	58.53	70.00	66.99
LoRA$_{64}$	PERFT (Top1/2)	2.16M	0.169	66.09	77.97	73.75	46.36	72.61	78.79	62.20	69.20	68.37
LoRA$_4$	PERFT (Top2/2)	0.33M	0.026	64.86	76.71	69.60	40.89	62.43	77.23	55.80	63.60	63.89
LoRA$_8$	PERFT (Top2/2)	0.59M	0.046	65.26	78.18	72.31	42.11	71.82	77.90	60.49	67.80	66.99
LoRA$_{16}$	PERFT (Top2/2)	1.11M	0.087	66.18	77.97	72.52	43.99	70.64	78.24	60.75	69.80	67.51
LoRA$_{32}$	PERFT (Top2/2)	2.16M	0.169	65.81	79.38	73.59	49.42	71.59	77.78	61.18	71.80	68.82
LoRA$_{64}$	PERFT (Top2/2)	4.26M	0.332	65.96	79.87	72.82	53.93	73.40	78.91	62.20	72.20	69.91
LoRA$_{128}$	PERFT (Top2/2)	8.45M	0.659	67.09	80.09	74.67	68.44	70.32	79.55	60.49	73.80	71.81
LoRA$_4$	PERFT (Top1/4)	0.39M	0.031	63.94	76.88	69.91	39.14	60.54	78.49	57.68	65.40	64.00
LoRA$_8$	PERFT (Top1/4)	0.66M	0.051	64.34	77.75	71.75	40.30	67.01	77.06	58.96	64.80	65.25
LoRA$_{16}$	PERFT (Top1/4)	1.18M	0.092	64.46	77.04	71.29	41.83	62.51	77.57	59.39	65.00	64.89
LoRA$_{32}$	PERFT (Top1/4)	2.23M	0.174	66.21	78.51	71.49	43.87	69.61	77.69	61.01	70.20	67.32
LoRA$_{64}$	PERFT (Top1/4)	4.33	0.337	65.32	79.60	73.49	45.33	71.11	77.69	62.20	71.00	68.22
LoRA$_4$	PERFT (Top2/4)	0.66M	0.051	63.98	75.68	69.29	40.26	65.75	77.36	59.56	67.40	64.91
LoRA$_8$	PERFT (Top2/4)	1.18M	0.092	65.02	77.86	71.90	41.61	68.75	77.31	59.13	68.80	66.30
LoRA$_{16}$	PERFT (Top2/4)	2.23M	0.174	64.07	76.61	73.59	42.10	71.90	78.32	60.58	71.20	67.30
LoRA$_{32}$	PERFT (Top2/4)	4.33M	0.337	66.30	77.75	75.44	45.88	71.43	76.18	60.58	70.60	68.02
LoRA$_4$	PERFT (Top4/4)	1.18M	0.092	64.25	75.84	71.03	41.40	69.22	77.65	57.08	68.40	65.61
LoRA$_8$	PERFT (Top4/4)	2.23M	0.174	65.14	77.64	72.98	42.67	72.45	76.98	59.39	66.40	66.71
LoRA$_{16}$	PERFT (Top4/4)	4.33M	0.337	65.44	79.43	73.08	48.35	71.19	77.48	59.98	73.40	68.55
LoRA$_{32}$	PERFT (Top4/4)	8.52M	0.665	66.70	79.49	73.75	55.95	71.43	77.53	60.07	70.40	69.41
LoRA$_{64}$	PERFT (Top4/4)	16.9M	1.319	66.02	79.71	75.49	59.29	73.32	76.64	59.90	71.80	70.27
LoRA$_{128}$	PERFT (Top4/4)	33.7M	2.628	65.99	78.94	75.13	67.21	73.72	78.24	59.90	74.80	71.74

Table 6: **(Part 1/2) Evaluation results for OLMoE with baseline methods and PERFT variants on eight commonsense reasoning benchmarks.** "Arch." denotes the architecture inside PEFT modules. "# Act." and "% Act." represent the number of activated trainable parameters and their ratio to the total activated parameters. "(TopK/N)" refers to activating K experts among the total number of N experts. Dataset names are partially abbreviated, including BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), Social IQa (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), Easy Set and Challenge Set of ARC (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018).

Arch.	Strategy	# Act.	% Act.	BoolQ	PIQA	SIQA	HellaS	WinoG	ARC-e	ARC-c	OBQA	Avg.
LoRA$_4$	PERFT (Top1/8)	0.52M	0.041	63.73	75.30	69.91	40.77	66.77	77.69	57.51	64.60	64.54
LoRA$_8$	PERFT (Top1/8)	0.79M	0.061	64.98	77.09	70.78	41.65	66.93	77.78	57.76	66.40	65.42
LoRA$_{16}$	PERFT (Top1/8)	1.31M	0.102	64.89	77.26	70.88	41.95	70.09	77.31	59.39	67.40	66.15
LoRA$_{32}$	PERFT (Top1/8)	2.36M	0.184	64.25	77.58	72.52	42.30	70.64	77.82	58.53	67.40	66.38
LoRA$_4$	PERFT (Top2/8)	0.79M	0.061	64.28	76.99	68.88	40.61	66.85	77.57	57.34	65.40	64.74
LoRA$_8$	PERFT (Top2/8)	1.31M	0.102	63.91	76.88	71.03	43.45	69.69	77.23	58.11	68.00	66.04
LoRA$_{16}$	PERFT (Top2/8)	2.36M	0.184	64.68	77.64	72.36	43.33	71.51	75.97	58.45	67.80	66.47
LoRA$_{32}$	PERFT (Top2/8)	4.46M	0.348	64.40	78.13	74.21	46.80	71.59	76.39	58.79	71.20	67.69
LoRA$_4$	PERFT (Top4/8)	1.31M	0.102	64.74	77.04	71.60	42.82	70.01	77.31	59.73	68.20	66.43
LoRA$_8$	PERFT (Top4/8)	2.36M	0.184	64.86	76.61	73.69	42.10	69.46	76.98	58.02	67.20	66.12
LoRA$_{16}$	PERFT (Top4/8)	4.46M	0.348	65.78	76.33	72.57	45.61	69.53	76.22	58.28	69.20	66.69
LoRA$_{32}$	PERFT (Top4/8)	8.65M	0.675	65.20	77.37	73.64	46.36	72.45	77.02	56.83	69.20	67.26
LoRA$_4$	PERFT (Top8/8)	2.36M	0.184	64.98	77.37	72.77	45.71	70.32	77.15	58.96	68.60	66.98
LoRA$_8$	PERFT (Top8/8)	4.46M	0.348	64.98	78.13	74.21	46.75	69.85	77.19	59.56	70.00	67.58
LoRA$_{16}$	PERFT (Top8/8)	8.65M	0.675	65.93	77.58	74.41	55.14	71.98	76.47	57.59	71.40	68.81
LoRA$_{32}$	PERFT (Top8/8)	17.0M	1.329	65.78	78.07	74.92	58.44	71.82	76.05	61.35	73.80	70.03
LoRA$_{64}$	PERFT (Top8/8)	33.8M	2.638	65.20	80.25	75.13	65.68	73.01	75.67	59.47	72.40	70.85
LoRA$_4$	PERFT (Top1/16)	0.79M	0.061	64.65	75.73	70.83	40.04	63.61	77.06	59.04	64.40	64.42
LoRA$_8$	PERFT (Top1/16)	1.05M	0.082	64.98	76.17	69.60	40.17	67.48	76.30	58.02	67.00	64.97
LoRA$_{16}$	PERFT (Top1/16)	1.57M	0.123	63.79	77.04	73.29	42.39	70.56	76.60	58.96	69.00	66.45
LoRA$_{32}$	PERFT (Top1/16)	2.62M	0.204	64.25	75.79	72.21	43.98	70.24	76.18	59.04	69.20	66.36
LoRA$_4$	PERFT (Top2/16)	1.05M	0.082	63.94	77.31	71.44	41.23	69.22	78.37	58.11	67.00	65.83
LoRA$_8$	PERFT (Top2/16)	1.57M	0.123	62.45	76.12	71.55	41.75	67.80	76.14	59.47	68.00	65.41
LoRA$_{16}$	PERFT (Top2/16)	2.62M	0.204	64.50	76.06	71.03	43.21	69.22	75.59	59.30	68.00	65.86
LoRA$_{32}$	PERFT (Top2/16)	4.72M	0.368	65.35	76.50	72.98	47.08	69.30	74.79	58.19	67.80	66.50
LoRA$_4$	PERFT (Top4/16)	1.57M	0.123	64.37	75.52	72.36	42.12	69.61	76.35	57.59	68.00	65.74
LoRA$_8$	PERFT (Top4/16)	2.62M	0.204	64.92	76.55	72.21	43.09	69.61	75.67	59.30	67.20	66.07
LoRA$_{16}$	PERFT (Top4/16)	4.72M	0.368	65.50	76.50	73.80	43.82	71.43	74.03	57.34	69.80	66.53
LoRA$_{32}$	PERFT (Top4/16)	8.91M	0.695	65.47	77.09	73.64	45.04	69.77	74.49	58.70	67.80	66.50
LoRA$_4$	PERFT (Top8/16)	2.62M	0.204	64.25	76.06	72.31	41.46	71.11	76.81	60.67	68.00	66.33
LoRA$_8$	PERFT (Top8/16)	4.72M	0.368	64.50	77.53	73.34	45.22	71.74	74.92	57.51	67.80	66.57
LoRA$_{16}$	PERFT (Top8/16)	8.91M	0.695	64.53	77.91	73.54	47.24	71.27	75.00	54.78	71.20	66.93
LoRA$_{32}$	PERFT (Top8/16)	17.3M	1.350	65.57	76.82	74.51	53.13	70.01	74.07	57.17	70.60	67.73
LoRA$_4$	PERFT (Top8/32)	3.15M	0.245	63.82	75.52	72.57	41.75	72.30	74.37	57.25	69.00	65.82
LoRA$_8$	PERFT (Top8/32)	5.24M	0.409	63.79	75.35	71.70	43.90	67.88	74.03	58.28	67.80	65.34
LoRA$_{16}$	PERFT (Top8/32)	9.44M	0.736	64.07	75.90	73.39	44.59	72.22	72.31	55.29	65.20	65.37
LoRA$_{32}$	PERFT (Top8/32)	17.8M	1.390	64.71	75.35	73.95	47.17	70.72	72.22	55.46	67.80	65.92
LoRA$_4$	PERFT (Top8/64)	4.19M	0.327	63.55	76.06	70.11	42.16	69.14	72.31	53.67	64.80	63.98
LoRA$_8$	PERFT (Top8/64)	6.29M	0.491	64.53	75.52	72.21	41.79	70.40	71.38	53.92	66.20	64.49
LoRA$_{16}$	PERFT (Top8/64)	10.5M	0.818	64.71	73.61	72.26	42.35	70.88	71.09	54.78	65.80	64.44
LoRA$_{32}$	PERFT (Top8/64)	18.9M	1.472	62.81	74.43	72.31	41.11	69.22	69.49	53.84	65.60	63.60
LoRA$_2$	PERFT-E (Top8/64)	1.05M	0.082	65.54	79.11	73.59	50.06	73.24	77.27	58.70	72.80	68.79
LoRA$_4$	PERFT-E (Top8/64)	2.10M	0.164	64.80	79.49	74.36	58.39	72.69	75.00	58.45	72.20	69.42
LoRA$_8$	PERFT-E (Top8/64)	4.19M	0.327	65.81	78.84	73.85	58.84	71.51	74.41	56.06	69.20	68.56
LoRA$_{16}$	PERFT-E (Top8/64)	8.39M	0.654	65.20	78.24	74.97	64.35	72.30	74.41	55.46	69.40	69.29
LoRA$_{32}$	PERFT-E (Top8/64)	16.8M	1.309	66.51	76.39	74.26	62.55	73.09	72.22	56.14	70.60	68.97
LoRA$_{64}$	PERFT-E (Top8/64)	33.6M	2.617	65.57	77.09	73.80	59.89	73.32	71.72	56.40	68.80	68.32

Table 7: **(Part 2/2) Evaluation results for OLMoE-1B-7B with baseline methods and PERFT variants on eight commonsense reasoning benchmarks.** "Arch." denotes the architecture inside PEFT modules. "# Act." and "% Act." represent the number of activated trainable parameters and their ratio to the total activated parameters. "(TopK/N)" refers to activating K experts among the total number of N experts. Dataset names are partially abbreviated, including BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), Social IQa (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), Easy Set and Challenge Set of ARC (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018).

D.2 OLMoE-1B-7B for Arithmetic Reasoning

Arch.	Strategy	# Act.	% Act.	MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	Avg.
LoRA_{2}	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	0.26M	0.020	20.00	8.72	43.04	20.47	52.95	29.40	29.10
LoRA_{4}	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	0.52M	0.041	21.83	8.11	40.51	20.47	50.79	28.80	28.42
LoRA_{8}	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	1.05M	0.082	17.33	8.57	44.05	24.02	50.59	30.90	29.24
LoRA_{16}	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	2.10M	0.164	18.83	9.02	46.58	24.02	50.59	29.20	29.71
LoRA_{32}	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	4.19M	0.327	19.17	8.79	43.54	23.23	51.97	28.20	29.15
LoRA_{64}	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	8.39M	0.654	17.00	9.10	47.09	22.83	49.80	27.10	28.82
LoRA_{128}	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	16.8M	1.309	15.00	8.11	44.81	22.83	49.02	26.50	27.71
LoRA_{4}	PERFT-S (1)	0.26M	0.020	21.00	5.61	40.00	18.50	50.59	28.90	27.43
LoRA_{8}	PERFT-S (1)	0.52M	0.041	17.00	6.22	34.18	17.32	39.17	30.20	24.02
LoRA_{16}	PERFT-S (1)	1.05M	0.082	14.83	6.29	35.19	21.26	41.73	27.30	24.43
LoRA_{32}	PERFT-S (1)	2.10M	0.164	16.17	4.09	34.68	18.11	37.40	23.60	22.34
LoRA_{4}	PERFT-D (2)	0.52M	0.041	18.67	5.76	37.97	20.08	40.75	24.60	24.64
LoRA_{8}	PERFT-D (2)	1.05M	0.082	15.67	5.46	33.16	18.11	37.40	24.40	22.37
LoRA_{16}	PERFT-D (2)	2.10M	0.164	14.00	4.85	30.13	16.93	34.65	22.00	20.43
LoRA_{32}	PERFT-D (2)	4.19M	0.327	8.17	3.87	29.11	19.29	25.39	15.70	16.92
LoRA_{4}	PERFT-D (4)	1.05M	0.082	14.17	5.08	34.18	22.05	35.43	21.80	22.12
LoRA_{8}	PERFT-D (4)	2.10M	0.164	9.17	3.94	31.65	19.69	29.13	20.60	19.03
LoRA_{16}	PERFT-D (4)	4.19M	0.327	9.33	3.03	21.77	20.87	21.46	13.30	14.96
LoRA_{32}	PERFT-D (4)	8.39M	0.654	4.33	1.97	16.20	21.65	18.90	12.90	12.66
LoRA_{4}	PERFT (Top1/2)	0.20M	0.015	18.83	7.88	41.77	16.93	44.88	26.10	26.07
LoRA_{8}	PERFT (Top1/2)	0.33M	0.026	19.00	7.51	47.09	19.69	53.35	31.90	29.75
LoRA_{16}	PERFT (Top1/2)	0.59M	0.046	21.17	8.79	52.15	19.69	57.68	32.00	31.91
LoRA_{32}	PERFT (Top1/2)	1.11M	0.087	27.17	9.33	50.89	20.87	57.09	32.00	32.89
LoRA_{4}	PERFT (Top2/2)	0.33M	0.026	21.17	8.19	45.82	18.11	49.02	30.30	28.77
LoRA_{8}	PERFT (Top2/2)	0.59M	0.046	23.33	7.35	51.65	18.50	52.76	33.50	31.18
LoRA_{16}	PERFT (Top2/2)	1.11M	0.087	26.50	8.49	52.15	20.87	56.69	32.30	32.83
LoRA_{32}	PERFT (Top2/2)	2.16M	0.169	23.67	9.25	44.81	21.65	53.35	35.20	31.32
LoRA_{4}	PERFT (Top1/4)	0.39M	0.031	18.83	8.87	48.86	21.65	50.20	29.10	29.59
LoRA_{8}	PERFT (Top1/4)	0.66M	0.051	20.83	9.48	44.05	17.32	55.91	29.60	29.53
LoRA_{16}	PERFT (Top1/4)	1.18M	0.092	22.67	7.88	46.84	20.47	51.77	33.50	30.52
LoRA_{32}	PERFT (Top1/4)	2.23M	0.174	25.67	7.35	54.18	19.69	54.72	32.10	32.28
LoRA_{4}	PERFT (Top2/4)	0.66M	0.051	19.33	7.73	45.32	16.93	49.21	31.70	28.37
LoRA_{8}	PERFT (Top2/4)	1.18M	0.092	16.33	6.97	44.30	16.54	48.82	30.10	27.18
LoRA_{16}	PERFT (Top2/4)	2.23M	0.174	20.83	8.34	47.34	18.50	51.18	33.70	29.98
LoRA_{32}	PERFT (Top2/4)	4.33M	0.337	28.00	9.10	49.37	19.29	57.09	33.20	32.67
LoRA_{4}	PERFT (Top4/4)	1.18M	0.092	20.67	7.58	47.85	20.08	53.35	31.30	30.14
LoRA_{8}	PERFT (Top4/4)	2.23M	0.174	25.33	7.73	40.51	20.08	49.02	30.70	28.89
LoRA_{16}	PERFT (Top4/4)	4.33M	0.337	21.50	7.43	45.06	20.87	59.84	30.30	30.83
LoRA_{32}	PERFT (Top4/4)	8.52M	0.665	22.17	8.34	50.38	20.08	55.31	30.80	31.18
LoRA_{4}	PERFT (Top2/8)	0.79M	0.061	21.83	7.88	50.89	21.26	51.97	29.90	30.62
LoRA_{8}	PERFT (Top2/8)	1.31M	0.102	20.00	8.26	47.34	19.29	52.76	28.30	29.33
LoRA_{16}	PERFT (Top2/8)	2.36M	0.184	22.33	8.72	46.08	20.87	50.39	30.20	29.76
LoRA_{32}	PERFT (Top2/8)	4.46M	0.348	22.50	7.43	46.84	18.90	50.59	30.90	29.53
LoRA_{4}	PERFT (Top8/8)	2.36M	0.184	28.33	7.81	47.85	16.93	53.15	31.20	30.88
LoRA_{8}	PERFT (Top8/8)	4.46M	0.348	21.00	8.49	49.37	21.26	51.97	31.60	30.61
LoRA_{16}	PERFT (Top8/8)	8.65M	0.675	28.50	8.04	45.82	20.87	53.74	32.90	31.64
LoRA_{32}	PERFT (Top8/8)	17.0M	1.329	27.67	8.49	45.06	21.26	52.95	32.60	31.34
LoRA_{4}	PERFT-E (Top8/64)	2.10M	0.164	26.67	6.44	46.58	22.05	53.94	32.10	31.30
LoRA_{8}	PERFT-E (Top8/64)	4.19M	0.327	28.33	7.81	43.80	21.26	57.28	32.60	31.85
LoRA_{16}	PERFT-E (Top8/64)	8.39M	0.654	25.17	8.42	43.29	19.29	48.82	29.50	29.08
LoRA_{32}	PERFT-E (Top8/64)	16.8M	1.309	26.17	6.75	44.05	20.87	52.76	32.80	30.56

Table 8: **Evaluation results for OLMoE-1B-7B with baseline methods and PERFT variants on six arithmetic reasoning benchmarks.** "Arch." denotes the architecture inside PEFT modules. "# Act." and "% Act." represent the number of activated trainable parameters and their ratio to the total activated parameters. "(TopK/N)" refers to activating K experts among the total number of N experts. Dataset names are partially abbreviated, including MultiArith (Roy and Roth, 2015), GSM8K (Cobbe et al., 2021), AddSub (Hosseini et al., 2014), AQuA (Ling et al., 2017), SingleEq (Koncel-Kedziorski et al., 2015), and SVAMP (Patel et al., 2021).

D.3 Mixtral-8×7B for Commonsense Reasoning

Arch.	Strategy	# Act.	% Act.	BoolQ	PIQA	SIQA	HellaS	WinoG	ARC-e	ARC-c	OBQA	Avg.
Base	(pretrained)	—	—	51.10	81.12	46.11	47.54	49.88	53.20	52.99	39.20	52.64
Base	(instruct)	—	—	68.87	88.30	68.58	72.06	59.98	89.52	78.50	74.40	75.03
LoRA_8	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	3.41M	0.026	73.49	90.04	81.17	89.67	82.16	93.56	83.87	86.20	85.02
LoRA_{16}	PERFT-S (1)	4.19M	0.033	75.11	90.26	81.63	94.26	84.85	92.85	81.40	87.60	85.99
LoRA_8	PERFT (Top2/2)	4.46M	0.035	74.68	89.77	81.47	94.33	86.27	92.05	81.48	89.80	86.23
LoRA_{16}	PERFT (Top1/4)	4.72M	0.037	72.84	89.12	80.40	92.69	84.37	91.84	82.25	85.80	84.91
LoRA_8	PERFT (Top2/4)	4.72M	0.037	74.71	90.10	79.38	94.18	85.71	92.09	81.31	85.80	85.41
LoRA_8	PERFT (Top2/8)	5.24M	0.041	73.76	89.12	81.63	94.51	85.16	91.67	80.20	87.80	85.48
LoRA_8	PERFT-E (Top2/8)	4.19M	0.033	74.13	90.21	80.81	91.36	86.42	92.21	81.06	88.60	85.60

Table 9: **Evaluation results for Mixtral-8×7B with baseline methods and PERFT variants on eight commonsense reasoning benchmarks.** "Arch." denotes the architecture inside PEFT modules. "# Act." and "% Act." represent the number of activated trainable parameters and their ratio to the total activated parameters. "(TopK/N)" refers to activating K experts among the total number of N experts. Dataset names are partially abbreviated, including BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), Social IQa (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), Easy Set and Challenge Set of ARC (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018).

D.4 Mixtral-8×7B for Arithmetic Reasoning

Arch.	Strategy	# Act.	% Act.	MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	Avg.
LoRA_8	$\boldsymbol{W}_q, \boldsymbol{W}_v$@Attn	3.41M	0.026	60.00	50.87	90.13	28.74	89.37	69.20	64.72
LoRA_8	PERFT (Top2/2)	4.46M	0.035	82.83	55.80	87.59	29.92	89.76	68.30	69.04
LoRA_8	PERFT (Top2/8)	5.24M	0.041	79.00	54.06	87.34	29.13	88.98	70.30	68.13

Table 10: **Evaluation results for Mixtral-8×7B with baseline methods and PERFT variants on six arithmetic reasoning benchmarks.** "Arch." denotes the architecture inside PEFT modules. "# Act." and "% Act." represent the number of activated trainable parameters and their ratio to the total activated parameters. "(TopK/N)" refers to activating K experts among the total number of N experts. Dataset names are partially abbreviated, including MultiArith (Roy and Roth, 2015), GSM8K (Cobbe et al., 2021), AddSub (Hosseini et al., 2014), AQuA (Ling et al., 2017), SingleEq (Koncel-Kedziorski et al., 2015), and SVAMP (Patel et al., 2021).