ROUTER-CENTRIC ADAPTATION FOR MOE LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing parameter-efficient fine-tuning (PEFT) methods for Mixture-of-Experts (MoE) LLMs primarily introduce trainable parameters into experts, yet ignore the router. In this paper, we find that, under top-k expert selection, models adapted with these methods largely inherit their token-to-expert assignment patterns from pre-training. Based on our empirical observations, we argue that adapting routing preferences is crucial for downstream adaptation. We therefore propose Router-Centric Adaptation (RoCA), a PEFT method that accounts for the routing characteristics of MoE LLMs.[1] Its core strategy is to explore potentially task-relevant experts to help shape task-adaptive routing preferences and improve expert utilization. It further uses routing uncertainty to adaptively inject domain knowledge. Experiments across three recent MoE backbones (Qwen3-30B-A3B, DeepSeek-V2-Lite, and OLMoE-1B-7B) and five domains show that RoCA achieves the highest average downstream performance while using 5–10$\times$ fewer trainable parameters than the second-best method, demonstrating its efficiency and effectiveness. RoCA also improves general capabilities after fine-tuning. We hope our work provides a new perspective on PEFT for MoE LLMs.

1 INTRODUCTION

Mixture-of-Experts (MoE) architectures activate only a small subset of experts per token, substantially increasing model capacity without proportional computational overhead during inference (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022). This design has been adopted by a growing range of MoE LLMs (Yang et al., 2025; DeepSeek-AI et al., 2024; Muennighoff et al., 2025). Adapting these models to downstream tasks is increasingly important, but their massive parameter counts make full fine-tuning prohibitively expensive, highlighting the importance of parameter-efficient fine-tuning (PEFT).

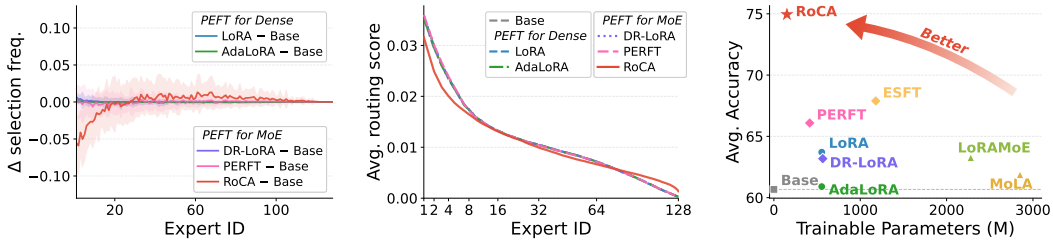
PEFT has been widely studied for dense models (Hu et al., 2022; Zhang et al., 2023), whereas methods explicitly tailored to MoE architectures remain scarce (Deng et al., 2026; Liu et al., 2026). Whether directly borrowed from dense models or designed specifically for MoE, these methods primarily introduce trainable parameters into experts.

Beyond the experts, the router is a critical component of MoE models: it determines token-to-expert assignment. Yet these methods focus on adapting experts and ignore the router. To examine the resulting routing behavior, we compare expert selection frequencies in Qwen3-30B-A3B before and after fine-tuning on mathematical data. We measure each expert's selection frequency as the fraction of answer tokens routed to it on MATH-500 (Lightman et al., 2024). As shown in Figure 1a, not surprisingly, the evaluated PEFT methods produce mean expert selection frequency residuals close to zero. This suggests that they largely continue to rely on token-to-expert assignment inherited from pre-training. However, data from a specific downstream task can differ substantially from the pre-training corpus in both knowledge content and input formats, highlighting the need to reconsider routing preferences during downstream adaptation.

Besides, inherited routing preferences may also hinder diverse expert utilization. Figure 1b shows that, in the math domain, a small group of experts receives much higher average routing scores than the remaining experts, forming a long-tailed distribution (He et al., 2026). Figure 1b also shows that average routing scores decline much more sharply from ranks 1 to 8 than from ranks 8 to 16, suggesting that experts just below the top-k boundary receive comparable routing scores and may

[1] Code is available at `https://anonymous.4open.science/r/RoCA-E77E/`.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Selection Frequency Residual (b) Expert Routing Scores (c) Efficiency and Effectiveness

Figure 1: Routing behavior and adaptation performance of Qwen3-30B-A3B. (a) Expert selection frequency residual on MATH-500, where selection frequency is the fraction of answer tokens for which an expert is selected, averaged across MoE layers. (b) Expert routing scores on MATH-500, obtained by applying softmax to the router logits and averaged over answer tokens and across MoE layers. (c) Average accuracy across downstream tasks versus the number of trainable parameters.

also be relevant to the downstream task. However, under top-k selection, these potentially relevant experts have limited opportunities to contribute because they are less favored by the routing preferences inherited from pre-training. Consequently, this limited routing diversity leaves the model's parameter capacity underutilized, potentially constraining its performance on downstream tasks.

Based on these observations, we propose **Ro**uter-**C**entric **A**daptation (**RoCA**), a PEFT method for MoE LLMs that promotes task-adaptive, diverse routing. RoCA introduces lightweight, trainable modulation of the frozen router and a trainable domain expert, enabling routing adaptation and additional domain knowledge acquisition. Within this framework, its core strategy, *candidate expert exploration*, randomly samples a subset of experts from a candidate pool near the top-k boundary during training. This allows potentially task-relevant experts to contribute to downstream computation and provide feedback for learning routing preferences, supporting more diverse routing. Observing that routing uncertainty varies across tokens, we further introduce *dynamic knowledge injection*, which uses this signal to adaptively control the domain expert's contribution to each token. Figures 1a and 1b preview the resulting changes in expert selection and routing diversity. Figure 1c further highlights RoCA's parameter efficiency and effectiveness on downstream tasks. Overall, our contributions are threefold:

- We analyze routing behavior during MoE adaptation, finding that the evaluated PEFT methods largely retain pre-trained expert preferences and concentrated routing scores. We also identify variation in routing uncertainty across tokens.
- We propose RoCA with lightweight routing modulation and a trainable domain expert. Its core strategy, candidate expert exploration, promotes task-adaptive, diverse routing, while dynamic knowledge injection controls token-level domain-expert contributions.
- Across three MoE backbones and five domains, RoCA achieves the highest average downstream performance with 5–10$\times$ fewer trainable parameters than the second-best method, while also improving average general benchmark scores over the pre-trained models.

2 Related Work

MoE LLMs. Mixture-of-Experts (MoE) routes each token to a sparse subset of experts (Shazeer et al., 2017), scaling model capacity without proportional increases in per-token computation (Lepikhin et al., 2021; Fedus et al., 2022). Models such as Mixtral (Jiang et al., 2024), DeepSeek-V2 (DeepSeek-AI et al., 2024), Qwen3 (Yang et al., 2025), and OLMoE (Muennighoff et al., 2025) demonstrate the effectiveness of this design. The introduction of the router also brings the load balancing problem: a few experts may receive most tokens while others remain underutilized throughout pre-training. Auxiliary losses encourage uniform expert utilization (Shazeer et al., 2017; Fedus et al., 2022). DeepSeek-V2 further uses fine-grained expert segmentation and shared experts to promote specialization and reduce knowledge redundancy. These efforts show that the router has been a central focus during pre-training. However, its potential during downstream adaptation remains largely overlooked, motivating our work.

Parameter-Efficient Fine-Tuning for LLMs. Parameter-efficient fine-tuning (PEFT) reduces adaptation costs by training only a small number of parameters. LoRA (Hu et al., 2022) is the most widely adopted approach, injecting trainable low-rank matrices into frozen weight matrices based on the hypothesis that weight updates have low intrinsic rank. AdaLoRA (Zhang et al., 2023) adaptively allocates rank budgets, while LoRAMoE (Dou et al., 2024) and MoLA (Gao et al., 2025) extend LoRA by integrating MoE mechanisms. However, these methods target dense models, whereas MoE's conditional computation calls for specialized PEFT designs. PERFT (Liu et al., 2026) adds independently routed adapter experts parallel to the base MoE block. ESFT (Wang et al., 2024) fine-tunes task-relevant experts selected using pre-computed activation statistics. DR-LoRA (Deng et al., 2026) adds per-expert LoRA adapters and grows their ranks according to expert importance. These methods primarily introduce trainable parameters into experts, but it is the router that determines token-to-expert assignment. Our work focuses on routing characteristics in downstream adaptation.

3 Router-Centric Adaptation (RoCA)

In this paper, we introduce **Ro**uter-**C**entric **A**daptation (**RoCA**), a PEFT method tailored for MoE LLMs that enables routing adaptation and domain knowledge acquisition through lightweight modulation of the frozen router and a trainable domain expert. RoCA employs *candidate expert exploration* to adapt expert selection by exploring candidates near the top-k boundary, and *dynamic knowledge injection* to regulate domain expert contributions based on routing entropy.

3.1 Preliminaries

Consider an MoE layer with N experts $\{E_1, \ldots, E_N\}$ and a router $W_r \in \mathbb{R}^{N\times d}$. Given a token representation $x \in \mathbb{R}^d$, routing probabilities and top-k selections are:

$$p(x) = \mathrm{softmax}(W_r x), \qquad \mathcal{S}_0(x) = \mathrm{TopK}(p(x), k). \tag{1}$$

Here, TopK returns expert indices, and the routed output is $y_0 = \sum_{i\in\mathcal{S}_0(x)} w_i E_i(x)$, with $w_i = p_i / \sum_{j\in\mathcal{S}_0(x)} p_j$ when renormalized and $w_i = p_i$ otherwise. Layer indices are omitted for clarity.

3.2 Framework

RoCA builds on two trainable components: lightweight modulation of the frozen router and a domain-specific expert. The former allows routing scores to adapt to downstream supervision, while the latter learns additional domain knowledge. Figure 2 illustrates the framework.

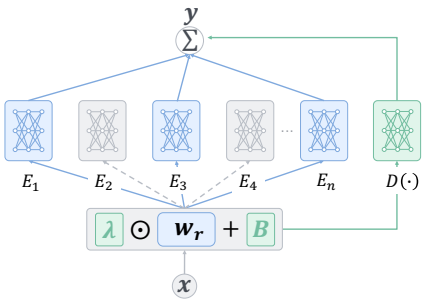


Figure 2: Framework of RoCA.

Scale-and-Shift modulation. Directly fine-tuning the router weights risks catastrophic forgetting of pre-trained routing knowledge. Instead, we freeze W_r and modulate its output through per-expert scaling and bias parameters:

$$z'(x) = \lambda \odot (W_r x) + b, \tag{2}$$

$$p'(x) = \mathrm{softmax}(z'(x)). \tag{3}$$

The vectors $\lambda, b \in \mathbb{R}^N$ are initialized to $\mathbf{1}$ and $\mathbf{0}$, respectively. This preserves the original routing scores at initialization and adds only $2N$ trainable parameters per layer.

Domain-specific expert. To learn knowledge not fully captured by the pre-trained experts, we introduce a trainable expert $D(\cdot)$ at each MoE layer:

$$D(x) = \frac{\gamma}{r} W_{\mathrm{up}}\, \mathrm{SiLU}(W_{\mathrm{down}} x), \tag{4}$$

where $W_{\mathrm{down}} \in \mathbb{R}^{r\times d}$ and $W_{\mathrm{up}} \in \mathbb{R}^{d\times r}$. Different from the linear adapters commonly used for weight adaptation in dense models, we incorporate the SiLU activation function into the domain-specific expert to introduce nonlinear transformations, enabling it to capture more complex domain-specific patterns. The intermediate dimension r is set to match that of the original pre-trained experts, and γ/r is an output scaling factor (Hu et al., 2022). We use Kaiming initialization for W_{down} and zero initialization for W_{up}, so the domain expert initially contributes zero output.

3.3 Candidate Expert Exploration

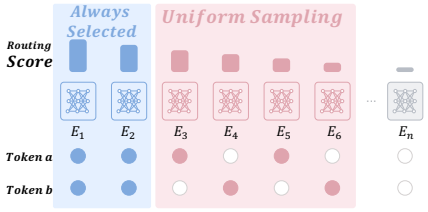


Figure 3: Candidate expert exploration.

Based on the former observations, routing preferences inherited from pre-training should be adapted for downstream tasks, as data from specific downstream tasks may exhibit different knowledge distributions and input characteristics. Meanwhile, experts near the top-k boundary may also be useful for downstream tasks, as they receive comparable routing scores but are rarely selected under the original routing preferences. We therefore introduce candidate expert exploration, which retains the highest-ranked experts and randomly selects candidates near this boundary, allowing them to participate in computation and help shape domain-specific routing preferences during adaptation.

For each token, we rank experts by the modulated probabilities $p'(x)$ and divide the k activation slots into k_1 deterministic slots and $k_2 = k - k_1$ exploration slots. We retain the top-k_1 experts and form a candidate pool from the next $c\,k_2$ experts. Formally, let $\pi_1, \ldots, \pi_N$ be expert indices sorted by decreasing $p'_i(x)$, and define

$$\mathcal{S}_{\text{keep}}(x) = \{\pi_1, \ldots, \pi_{k_1}\}, \qquad \mathcal{C}(x) = \{\pi_{k_1+1}, \ldots, \pi_{\min(N, k_1+ck_2)}\}. \tag{5}$$

We uniformly sample k_2 experts from $\mathcal{C}(x)$:

$$\mathcal{S}(x) = \mathcal{S}_{\text{keep}}(x) \cup \text{Sample}(\mathcal{C}(x), k_2). \tag{6}$$

The multiplier $c \geq 1$ controls the exploration range. The pool is constructed for each token from the current routing scores, and the total number of activated experts remains k. Figure 3 illustrates this selection strategy. At inference, we return to standard top-k selection.

The selected experts remain frozen, but their outputs contribute to the task loss and thereby inform updates to the routing modulation parameters. Scale-and-Shift and candidate exploration thus work together: the former allows expert preferences to change, while the latter exposes learning to potentially useful expert choices beyond deterministic top-k.

3.4 Dynamic Knowledge Injection

Beyond expert selection, routing uncertainty can also reveal token-level variation in the need for additional domain knowledge, as tokens with higher uncertainty may benefit more from domain-expert contributions. For example, for Qwen3-30B-A3B, Figure 4 shows higher routing entropy for the LaTeX closing delimiter `)$` than for `the` within the same math sample. This delimiter marks the end of a mathematical expression, where deciding what comes next may require more domain knowledge than routine language continuation. We therefore introduce *dynamic knowledge injection*, using routing uncertainty as a proxy for token-level knowledge demand to strengthen domain-expert contributions at high-uncertainty positions and reduce them at low-uncertainty ones. Before expert selection, we compute Shannon entropy (Shannon, 1948) over the modulated routing distribution across all experts:

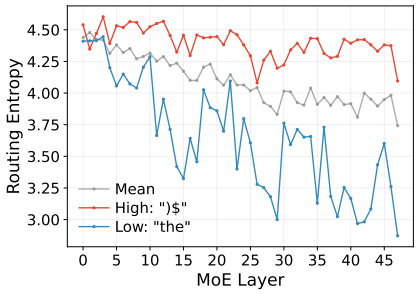


Figure 4: An example of token-level routing entropy (Equation 7).

$$H(x) = -\sum_{i=1}^{N} p'_i(x) \log p'_i(x). \tag{7}$$

The injection weight is then:

$$\alpha(x) = \text{sigmoid}\left(m\frac{H(x) - \mu}{\sigma}\right), \tag{8}$$

where μ and σ are the mean and standard deviation of routing entropy, computed separately for each domain and layer on a subset of the training data before fine-tuning. These statistics remain fixed, and $m > 0$ controls the steepness of the gate.

Entropy above μ yields a larger α, increasing domain knowledge injection when expert selection is more uncertain; entropy below μ yields a smaller α, reducing injection when the router already has a clear expert preference. This design allows the domain expert to focus on tokens where the original pre-trained experts provide less decisive routing signals, while avoiding unnecessary interference with confident expert decisions. Figure 5 illustrates this adaptive weighting. The gate remains active during both training and inference, dynamically adjusting the domain expert's contribution for each token.

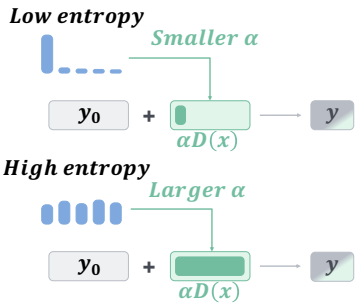


Figure 5: Dynamic knowledge injection guided by routing entropy.

3.5 Overview

Combining the above components, RoCA produces the final MoE output as:

$$y = \sum_{i \in \mathcal{S}_{\text{keep}}(x)} w_i' E_i(x) + \underbrace{\sum_{j \in \text{Sample}(\mathcal{C}(x), k_2)} w_j' E_j(x)}_{\text{candidate expert exploration}} + \underbrace{\alpha(x) D(x)}_{\text{dynamic knowledge injection}}, \tag{9}$$

where w_i' is derived from p_i' in the same way as w_i from p_i in Section 3.1. During fine-tuning, all pre-trained parameters remain frozen. Only λ, b, W_{down}, and W_{up} are optimized with the downstream task loss, adding $2N + 2dr$ trainable parameters per adapted layer. As shown in Equation 9, the first term preserves the most relevant knowledge, the second term, as the core of RoCA, explores candidate experts near the top-k boundary to adapt routing preferences and improve capacity utilization, and the third term adaptively injects domain knowledge guided by routing uncertainty.

4 Experiments

4.1 Experimental Settings

Models. We evaluate on three MoE backbones of different scales: Qwen3-30B-A3B (Yang et al., 2025), DeepSeek-V2-Lite (DeepSeek-AI et al., 2024), and OLMoE-1B-7B (Muennighoff et al., 2025), providing a comprehensive evaluation across different model sizes, expert counts, and routing configurations. Further architectural details are provided in Appendix A. **Datasets and evaluation.** We conduct independent fine-tuning on five diverse domains, each with 7.5K training samples sampled from the corresponding dataset and paired with a standardized evaluation benchmark: Mathematical Reasoning (MATH (Hendrycks et al., 2021d), evaluated on MATH-500), Code Generation (APPS (Hendrycks et al., 2021a) and Code Instructions 18k Alpaca (Iamtarun, 2023), evaluated on HumanEval (Chen et al., 2021)), Medical QA (MedQA (Jin et al., 2020)), Ethical Reasoning (Ethics (Hendrycks et al., 2021b)), and Legal Understanding (LEDGAR (Tuggener et al., 2020; Chalkidis et al., 2022)). For the latter three domains, we evaluate on their respective test splits. For general capability evaluation, we additionally evaluate on MMLU (Hendrycks et al., 2021c;b), ARC-Challenge and ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), and WinoGrande (Sakaguchi et al., 2019). Training data and evaluation details are provided in Appendices B and C, respectively.

Baselines and implementation. We compare against seven PEFT baselines, including PEFT methods for dense models (LoRA (Hu et al., 2022), AdaLoRA (Zhang et al., 2023), LoRAMoE (Dou et al., 2024), MoLA (Gao et al., 2025)) and PEFT methods for MoE models (ESFT (Wang et al., 2024), DR-LoRA (Deng et al., 2026), PERFT (Liu et al., 2026)). All methods are trained with the same data, learning rate (2×10^{-5}), 3 epochs, and linear learning rate schedule with 3% warmup. For RoCA, we set the domain expert dimension r equal to the native MoE intermediate size and $k_1 = 3$ across all backbones. The candidate pool multiplier c is set to 4 for Qwen3-30B-A3B, 3 for DeepSeek-V2-Lite, and 2 for OLMoE-1B-7B. The entropy statistics μ and σ are pre-computed on 10% of each domain's training set.

Table 1: Main results on three MoE backbones fine-tuned separately in each of five domains. Bold and underlined scores denote the best and second-best results. Green × gives the ratio of ESFT's parameter-range midpoint to RoCA's parameter count, with ESFT the strongest baseline on average; green + shows gains over the corresponding base model.

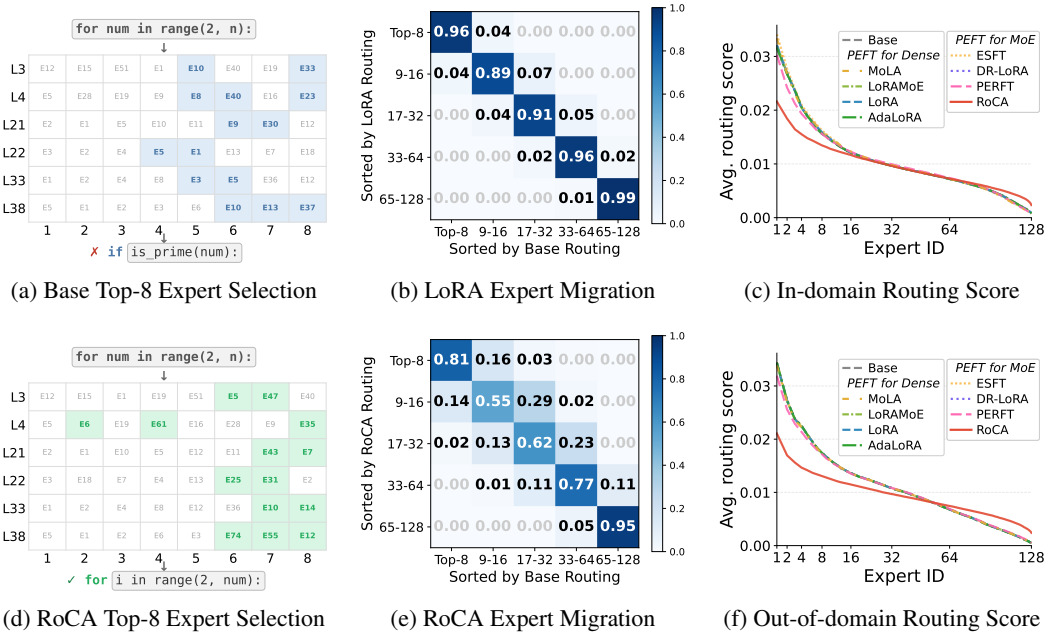
Method	Trainable Params	MATH-500	Human Eval	MedQA	Ethics	LEDGAR	Avg.
			Qwen3-30B-A3B				
Base model	—	62.0	55.48	77.14	69.16	39.55	60.67
PEFT for Dense Models							
MoLA	2.85B	65.0	56.71	76.51	71.59	39.55	61.87
LoRAMoE	2.28B	<u>67.4</u>	56.61	77.30	75.13	39.88	63.26
LoRA	554M	65.2	59.76	77.53	75.83	40.24	63.71
AdaLoRA	554M	63.2	54.88	77.14	69.77	39.52	60.90
PEFT for MoE Models							
ESFT	0.74B–1.62B	66.6	67.07	<u>78.71</u>	81.41	<u>45.64</u>	<u>67.89</u>
DR-LoRA	566M	64.2	59.14	77.14	75.39	40.04	63.18
PERFT	415M	62.2	<u>70.73</u>	74.63	<u>82.01</u>	40.88	66.09
RoCA	151M×7.81	**69.4**+7.4	**73.17**+17.69	**80.91**+3.77	**83.17**+14.01	**68.12**+28.57	**74.95**+14.28
			DeepSeek-V2-Lite				
Base model	—	17.2	28.66	41.87	53.62	32.41	34.75
PEFT for Dense Models							
MoLA	920M	15.8	28.05	49.65	58.63	33.83	37.19
LoRAMoE	759M	15.4	28.05	48.47	50.66	33.32	35.18
LoRA	188M	15.6	28.66	49.65	65.88	33.94	38.75
AdaLoRA	188M	16.8	28.66	41.24	49.63	33.12	33.89
PEFT for MoE Models							
ESFT	1.15B–1.45B	**18.8**	29.27	<u>52.08</u>	<u>78.29</u>	<u>36.70</u>	<u>43.03</u>
DR-LoRA	188M	16.4	27.44	47.21	53.12	33.21	35.48
PERFT	113M	15.0	**30.49**	45.48	67.82	33.02	38.36
RoCA	150M×8.67	<u>18.0</u>+0.8	**30.49**+1.83	**52.24**+10.37	**79.83**+26.21	**43.15**+10.74	**44.74**+9.99
			OLMoE-1B-7B				
Base model	—	3.2	13.41	39.91	58.86	35.39	30.15
PEFT for Dense Models							
MoLA	519M	4.8	13.41	46.27	70.21	36.91	34.32
LoRAMoE	415M	4.0	14.02	45.25	69.81	37.09	34.03
LoRA	101M	<u>5.6</u>	14.63	<u>47.68</u>	73.52	38.12	35.91
AdaLoRA	101M	3.4	14.02	40.93	57.78	36.63	30.55
PEFT for MoE Models							
ESFT	264M–604M	5.4	**17.68**	47.13	<u>77.00</u>	<u>49.57</u>	<u>39.36</u>
DR-LoRA	103M	4.8	14.02	44.46	70.41	37.56	34.25
PERFT	69M	5.0	15.24	39.12	74.58	38.16	34.42
RoCA	67M×6.48	**6.0**+2.8	**17.68**+4.27	**47.76**+7.85	**78.38**+19.52	**64.46**+29.07	**42.86**+12.71

4.2 Main Results

Strong performance with fewer parameters. As shown in Table 1, RoCA achieves the highest average performance across all three MoE backbones. RoCA shows promising gains on reasoning-related tasks. For example, on Qwen3-30B-A3B, it raises HumanEval performance from 55.48 to 73.17, outperforming the second-best method by 2.44 points. Methods designed for dense models can require substantially more trainable parameters without corresponding performance gains. For example, on Qwen3-30B-A3B, MoLA (2.85B) and LoRAMoE (2.28B) use approximately 19× and 15× as many trainable parameters as RoCA (151M), respectively, yet both achieve lower average scores. This suggests that simply scaling up trainable parameters does not guarantee performance gains on MoE models. One possible explanation is that sparse expert activation limits the utilization of added trainable parameters during adaptation. Among PEFT methods for MoE models, ESFT consistently ranks second in average performance across all three backbones, while RoCA achieves

Table 2: General capability evaluation on Qwen3-30B-A3B. Results show mean ± standard deviation across each method's five domain-specific checkpoints evaluated in Table 1. Bold and underlined scores denote the best and second-best results, respectively.

Method	MMLU	ARC-C	ARC-E	Hella Swag	Wino Grande	Avg.
Base model	78.59	57.51	79.50	81.43	72.77	73.96
PEFT for Dense Models						
MoLA	78.72±0.08	58.17±0.82	80.66±0.70	81.38±0.07	72.63±0.33	74.31±0.40
LoRAMoE	78.78±0.17	57.97±0.83	81.01±0.73	81.37±0.08	72.60±0.39	74.35±0.44
LoRA	78.78±0.14	58.16±0.77	80.70±0.46	81.44±0.08	72.64±0.63	74.34±0.42
AdaLoRA	78.66±0.09	57.34±0.31	79.84±0.24	81.37±0.06	72.55±0.26	73.95±0.19
PEFT for MoE Models						
ESFT	<u>78.85</u>±0.25	57.32±0.95	79.92±0.54	81.43±0.05	<u>73.10</u>±0.23	74.13±0.40
DR-LoRA	78.75±0.11	57.63±0.67	80.08±0.69	81.40±0.07	72.94±0.36	74.16±0.38
PERFT	78.39±0.25	<u>60.03</u>±1.29	<u>82.30</u>±1.55	**81.99**±0.32	71.71±1.46	<u>74.88</u>±0.98
RoCA	**79.33**±0.43	**62.03**±1.87	**83.12**±1.35	<u>81.54</u>±0.28	**73.29**±0.51	**75.86**±0.89



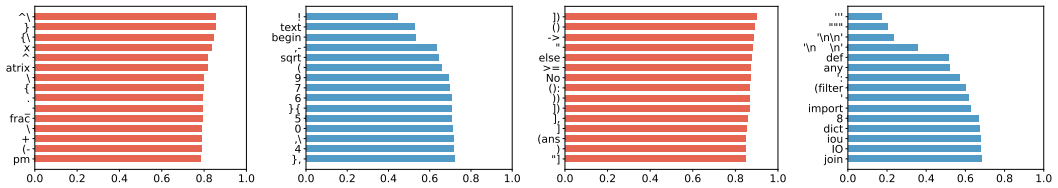
(a) Base Top-8 Expert Selection (b) LoRA Expert Migration (c) In-domain Routing Score

(d) RoCA Top-8 Expert Selection (e) RoCA Expert Migration (f) Out-of-domain Routing Score

Figure 6: Candidate expert exploration analysis on Qwen3-30B-A3B. (a, d) Top-8 expert selections at six MoE layers (two early, two middle, two late) after `for num in range(2, n):`. Gray/blue/green indicate shared/base-only/RoCA-only experts. (b, e) Expert migration on HumanEval answer tokens. Experts are grouped by routing score. For each group after fine-tuning (y-axis), the cells show the fractions of its experts that belonged to each group in the base model (x-axis). (c, f) Mean routing scores on domain-specific and general benchmarks, respectively.

higher scores with substantially fewer trainable parameters. On Qwen3-30B-A3B, RoCA uses only 151M parameters compared with ESFT's 0.74B–1.62B, a roughly 5–11× reduction. These results demonstrate the parameter efficiency and effectiveness of RoCA. Results under mixed-domain training are provided in Appendix D.

General capability is even enhanced. A common concern with domain-specific fine-tuning is the degradation of general-purpose capabilities. To investigate this, we select Qwen3-30B-A3B, the strongest backbone among the three, and evaluate each method's five domain-specific checkpoints on general benchmarks. We report the mean and standard deviation across these checkpoints. As shown in Table 2, most methods achieve average general performance comparable to or slightly



(a) Math High-α (b) Math Low-α (c) Code High-α (d) Code Low-α

Figure 7: The 15 tokens with the highest (red) and lowest (blue) mean injection weights in Math and Code on Qwen3-30B-A3B. The x-axes show mean α values.

above the base model. One possible explanation is that the relatively small amount of data used in each fine-tuning run does not substantially undermine the general capabilities acquired during pre-training. RoCA achieves the highest average (75.86), gaining 1.90 points over the base model, with the largest improvements on ARC-Challenge (+4.52) and ARC-Easy (+3.62), both requiring reasoning with scientific knowledge. This is consistent with the reasoning gains in Table 1 and suggests that the benefits of more diverse expert utilization extend to general capabilities. Additional general capability results on DeepSeek-V2-Lite are provided in Appendix E.

4.3 Candidate Expert Exploration: Analysis

Helping shape task-adaptive routing. Figure 1a shows that RoCA changes expert selection frequencies on Math. Here, we examine a code-generation case in Figures 6a and 6d: the model is asked to return the prime numbers below a given non-negative integer. The base model and RoCA generate the same code up to `for num in range(2, n):`, after which their next-token predictions diverge. The base model generates `if` and later calls an undefined `is_prime()` function, producing an incorrect solution. At this branching point, RoCA selects several green-highlighted experts in place of the blue-highlighted experts selected only by the base model. RoCA then generates `for`, checks divisibility using a loop, and solves the task correctly. The full problem statement and responses from the base model and RoCA are provided in Appendix F.

This case illustrates how changes in expert selection accompany a successful task-specific generation. To further characterize these routing changes, we sort experts by routing score and track their movement between groups across all answer tokens and MoE layers on HumanEval (Figures 6b and 6e). Approximately 19% of RoCA's top-8 expert selections originate outside the base model's top-8 group (16% from its 9–16 group and 3% from lower-ranked groups), compared with only 4% for LoRA. RoCA also exhibits substantial expert migration beyond the top-8 group, as reflected by the substantial proportions immediately to the right of the diagonal. In contrast, LoRA's values are concentrated along the diagonal, indicating that it largely preserves the base model's token-to-expert assignment. Together, the case study and dataset-level analysis suggest that RoCA helps shape task-adaptive routing preferences.

Improved utilization of model capacity. Beyond helping shape task-adaptive routing, candidate expert exploration also promotes broader expert utilization. In the base model, just eight experts account for 35.1% of all selections (Appendix G). The evaluated PEFT baselines also exhibit concentrated routing score distributions (Figures 6c and 6f). In contrast, RoCA produces a notably flatter routing score distribution than the baselines, with probability mass spread more evenly across experts. This pattern appears on both in-domain and general benchmarks, suggesting that RoCA makes use of a broader range of expert knowledge rather than relying on a few dominant experts. The persistence of this pattern on general benchmarks may also help explain the general capability improvements observed in Table 2.

Token-level injection analysis. Figure 7 shows the tokens with the highest and lowest mean α values on Math and Code. High-α tokens tend to be structural elements where the model must decide what follows: in Math, mathematical operators such as `^`, `frac`, and `+` are followed by a single value or a more complex expression; in Code, tokens associated with conditional logic, such as `else` and `>=`, require the model to complete a branch or comparison. Closing delimiters in both domains also receive high α, as they mark the end of one expression and the model must decide what expression comes next. These positions require stronger domain knowledge to resolve what follows. In contrast, low-α tokens, such as specific numerical values in Math and language

Table 3: Component ablation on Qwen3-30B-A3B. The framework combines learnable routing modulation with a trainable domain expert; candidate expert exploration and dynamic knowledge injection are added individually or jointly (RoCA). Bold scores denote the best results.

Setting	MATH-500	Human Eval	MedQA	Ethics	LEDGAR	Avg.
Base model (no fine-tuning)	62.0	55.48	77.14	69.16	39.55	60.67
Framework	63.8	70.73	79.34	82.57	59.34	71.16
+ candidate expert exploration	68.2	**73.17**	80.20	82.89	**69.62**	74.82
+ dynamic knowledge injection	65.0	**73.17**	80.44	82.82	60.82	72.45
+ both (RoCA)	**69.4**	**73.17**	**80.91**	**83.17**	68.12	**74.95**

Table 4: Effect of candidate pool multiplier c on Qwen3-30B-A3B. Larger c expands the exploration pool; shading marks the default setting, and bold scores indicate the best results.

Pool multiplier c	MATH-500	Human Eval	MedQA	Ethics	LEDGAR	Avg.
$c = 2$	67.2	71.34	80.28	82.70	59.00	72.10
$c = 3$	68.6	72.56	79.89	82.93	63.16	73.43
$c = 4$ (default)	**69.4**	**73.17**	**80.91**	**83.17**	**68.12**	**74.95**
$c = 5$	69.2	70.12	79.81	82.42	62.47	72.80

keywords or built-in types (`def`, `import`, `dict`) in Code, are largely determined by syntax or carry concrete semantic content, thus requiring less domain-specific knowledge. This shows that the entropy gate automatically calibrates injection intensity to match the actual knowledge demand at each token. Beyond these token-level patterns, Appendix H further examines how injection intensity varies across domains and MoE layers.

4.4 Ablation Results

Component ablation. We evaluate candidate expert exploration and dynamic knowledge injection both individually and jointly on top of the base framework. As shown in Table 3, candidate expert exploration is the primary source of improvement, raising the average score from 71.16 to 74.82 (+3.66). This substantial gain highlights the importance of exploring potentially relevant experts beyond deterministic top-k routing. Dynamic knowledge injection provides a modest yet consistent benefit, improving the average score to 72.45 (+1.29), confirming the effectiveness of adaptively injecting domain knowledge according to demand. Combining both components yields the best average performance of 74.95, outperforming the base framework by 3.79 points.

4.5 Dynamic Knowledge Injection: Analysis

Effect of candidate pool size. We vary the candidate pool multiplier c on Qwen3-30B-A3B, as shown in Table 4. Increasing c from 2 to 4 steadily improves performance (72.10 $\rightarrow$ 74.95), as a larger candidate pool exposes the model to more diverse experts during training, enabling it to discover more task-relevant experts. However, increasing c to 5 leads to a decline (72.80), as an overly large pool introduces less relevant experts that may undermine effective routing adaptation.

5 Discussion

In this work, we introduce RoCA, a PEFT method tailored to the routing characteristics of MoE LLMs. RoCA combines lightweight, learnable routing modulation with a trainable domain expert. Its core strategy, candidate expert exploration, explores experts near the top-k boundary to help shape task-adaptive routing preferences and improve expert utilization. RoCA further uses routing uncertainty to adaptively inject domain knowledge. Across three MoE backbones and five domains, RoCA achieves the highest average downstream performance while using 5–10$\times$ fewer trainable parameters than the second-best method, and also improves general capabilities after fine-tuning. These results underscore the importance of adapting routing preferences for effective and efficient fine-tuning of MoE LLMs.

AI USE STATEMENT

In this work, we used generative AI tools to polish the language and presentation of the manuscript. We have reviewed all AI-assisted work and carefully revised the manuscript to ensure accuracy and fidelity to the intended meaning. We independently formulated the research questions, designed the methodology, conducted the experiments, and drew the scientific conclusions. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel Katz, and Nikolaos Aletras. LexGLUE: A benchmark dataset for legal language understanding in English. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4310–4330, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.297. URL https://aclanthology.org/2022.acl-long.297/.

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

Daixuan Cheng, Shaohan Huang, and Furu Wei. Adapting large language models via reading comprehension. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 48624–48652, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/d51cd79a85833b022841f7a2383b32d3-Paper-Conference.pdf.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

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

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

Guanzhi Deng, Bo Li, Ronghao Chen, Xiujin Liu, Zhuo Han, Huacan Wang, Lijie Wen, and Linqi Song. Dr-lora: Dynamic rank lora for fine-tuning mixture-of-experts models, 2026. URL https://arxiv.org/abs/2601.04823.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1932–1945, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.106. URL https://aclanthology.org/2024.acl-long.106/.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL http://jmlr.org/papers/v23/21-0998.html.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Ruibo Liu, Baochen Sun, Yawen Zhang, Daiyi Peng, Xiaoyuan Guo, and Vs Subrahmanian. MoLA: MoE LoRA with layer-wise expert allocation. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5112–5127, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.284. URL https://aclanthology.org/2025.findings-naacl.284/.

Haoze He, Xingyuan Ding, Xuan Jiang, Xinkai Zou, Alex Cheng, Yibo Zhao, Juncheng B Li, and Heather Miller. Preserving long-tailed expert information in mixture-of-experts tuning. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=CpYgIGSrZu.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with apps. In J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021a. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/c24cd76e1ce41366a4bbe8a49b02a028-Paper-round2.pdf.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. Aligning {ai} with shared human values. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=dNy_RKzJacY.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021c. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021d. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/be83ab3ecd0db773eb2dc1b0a17836a1-Paper-round2.pdf.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Iamtarun. Python code instructions 18k alpaca. Hugging Face Datasets, 2023. URL https://huggingface.co/datasets/iamtarun/python_code_instructions_18k_alpaca. Accessed: 2026-09-14.

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

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams, 2020. URL https://arxiv.org/abs/2009.13081.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. {GS}hard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qrwe7XHTmYb.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Yilun Liu, Yunpu Ma, Yuetian Lu, Shuo Chen, Zifeng Ding, and Volker Tresp. Parameter-efficient routed fine-tuning: Mixture-of-experts demands mixture of adaptation modules. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 4439–4457, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-386-9. doi: 10.18653/v1/2026.findings-eacl.232. URL https://aclanthology.org/2026.findings-eacl.232/.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Evan Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. OLMoe: Open mixture-of-experts language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xXTkbTBmqq.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Don Tuggener, Pius von Däniken, Thomas Peetz, and Mark Cieliebak. LEDGAR: A large-scale multi-label corpus for text classification of legal provisions in contracts. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis (eds.), *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 1235–1241, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.155/.

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 Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 784–801, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.46. URL https://aclanthology.org/2024.emnlp-main.46/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lq62uWRJjiY.

A MoE Backbone Architectures

Table 5: Architecture of the three MoE backbones. Active parameters are reported per token. Experts and Top-K refer to routed experts per MoE layer; shared experts are listed separately. Expert Dim. denotes the expert intermediate dimension r. A dash indicates no shared experts.

Model	Total Params	Active Params	Layers	MoE Layers	Experts	Shared Experts	Top-K	Expert Dim. r
Qwen3-30B-A3B-Base	30.5B	3.3B	48	48	128	—	8	768
DeepSeek-V2-Lite	15.7B	2.4B	27	26	64	2	6	1408
OLMoE-1B-7B-0924	6.9B	1.3B	16	16	64	—	8	1024

Table 5 summarizes the architectures and parameter counts of Qwen3-30B-A3B-Base (Yang et al., 2025), DeepSeek-V2-Lite (DeepSeek-AI et al., 2024), and OLMoE-1B-7B-0924 (Muennighoff et al., 2025) used in our experiments. For each backbone, RoCA's added domain expert uses the same intermediate dimension r as its native routed experts: 768 for Qwen3, 1408 for DeepSeek-V2-Lite, and 1024 for OLMoE.

B Details of Training Data

We conduct independent fine-tuning on five domains, each with 7,500 training examples:

- **MATH** (Hendrycks et al., 2021d): A 12.5K-scale dataset of challenging competition mathematics problems spanning seven subsets (algebra, counting and probability, geometry, intermediate algebra, number theory, prealgebra, and precalculus), with each problem accompanied by a step-by-step solution and a difficulty level from 1 to 5. We proportionally sample 7,500 training examples from its training split.
- **APPS** (Hendrycks et al., 2021a) + **Code Instructions 18k Alpaca** (Iamtarun, 2023): APPS contains coding problems of varying difficulty with natural language problem descriptions and Python solutions. Code Instructions 18k Alpaca provides 18.6K instruction-response pairs for Python code generation in Alpaca format. We use the entire APPS training split (5,000 examples) and randomly sample 2,500 examples from Code Instructions 18k Alpaca to form the 7,500 code domain training set.
- **MedQA** (Jin et al., 2020): A free-form multiple-choice open-domain question answering dataset collected from professional medical board exams. It covers three languages (English, simplified Chinese, and traditional Chinese) with 12,723, 34,251, and 14,123 questions, respectively. We use the USMLE 4-option English subset (10.2K training examples) and randomly sample 7,500 examples from its training split.
- **Ethics** (Hendrycks et al., 2021b): An ethics benchmark for evaluating language models on moral judgment across five dimensions: commonsense morality, deontology, justice, utilitarianism, and virtue ethics. We proportionally sample 7,500 training examples from the training splits of these five subsets.
- **LEDGAR** (Tuggener et al., 2020; Chalkidis et al., 2022): A contract provision classification dataset where each provision paragraph is extracted from contracts filed with the U.S. Securities and Exchange Commission (SEC) and labeled with one of 100 provision types. We use the version provided in LexGLUE (Chalkidis et al., 2022) and randomly sample 7,500 examples from its training split (60K examples). Each example is prepended with the prompt "`The following is a provision from a legal contract. What is the main topic of this provision?`".

C Evaluation Details

C.1 In-Domain Benchmarks

- **MATH-500** (Lightman et al., 2024): A curated subset of 500 problems from the MATH test set. We use 4-shot chain-of-thought prompting and report accuracy using the

`math-verify` metric, which checks mathematical equivalence between the model's answer and the ground truth.

- **HumanEval** (Chen et al., 2021): A code generation benchmark consisting of 164 handcrafted Python programming problems. We use 0-shot prompting and report pass@1 with greedy decoding (temperature = 0).
- **MedQA** (Jin et al., 2020): We evaluate on the USMLE 4-option English test split using 0-shot prompting and report accuracy based on exact match of the selected option.
- **Ethics** (Hendrycks et al., 2021b): We evaluate on the test splits of all five subsets (commonsense morality, deontology, justice, utilitarianism, and virtue ethics) using 0-shot prompting and report the average accuracy across subsets.
- **LEDGAR** (Tuggener et al., 2020; Chalkidis et al., 2022): We follow Cheng et al. (2024), which converts the original 100-class classification into a four-choice format. We use 0-shot prompting and report the average accuracy across all 100 topics.

C.2 General Benchmarks

For general capability evaluation, we use the following benchmarks with 0-shot prompting unless otherwise noted. All benchmarks report accuracy.

- **MMLU** (Hendrycks et al., 2021c): A massive multitask language understanding benchmark covering 57 subjects. We use 5-shot prompting and report accuracy.
- **ARC-Challenge and ARC-Easy** (Clark et al., 2018): Science question answering benchmarks. ARC-Challenge contains the more difficult questions that require reasoning. We use 0-shot prompting and report accuracy.
- **HellaSwag** (Zellers et al., 2019): A commonsense reasoning benchmark for sentence completion. We use 0-shot prompting and report accuracy.
- **WinoGrande** (Sakaguchi et al., 2019): A large-scale commonsense reasoning benchmark for pronoun resolution. We use 0-shot prompting and report accuracy.

D Mixed-Domain Training

Table 6: Mixed-domain training on Qwen3-30B-A3B and OLMoE-1B-7B. Bold and underlined scores denote the best and second-best results within each backbone, respectively.

Method	Trainable Params	MATH-500	Human Eval	MedQA	Ethics	LEDGAR	Avg.
			Qwen3-30B-A3B				
Base model	—	62.0	55.48	77.14	69.16	39.55	60.67
LoRA	554M	<u>66.2</u>	61.59	77.61	77.36	40.87	64.73
DR-LoRA	566M	<u>66.2</u>	60.98	76.83	73.57	40.34	63.58
PERFT	415M	62.6	<u>67.68</u>	<u>77.69</u>	<u>82.18</u>	<u>41.17</u>	<u>66.26</u>
RoCA	151M	**67.4**	**76.83**	**80.52**	**82.34**	**64.67**	**74.35**
			OLMoE-1B-7B				
Base model	—	3.2	13.41	39.91	58.86	35.39	30.15
LoRA	101M	4.6	<u>14.02</u>	<u>43.52</u>	<u>70.91</u>	<u>38.81</u>	<u>34.37</u>
DR-LoRA	103M	**6.6**	<u>14.02</u>	42.26	65.91	37.54	33.27
PERFT	69M	<u>6.0</u>	<u>14.02</u>	37.39	69.93	34.78	32.42
RoCA	67M	5.4	**15.85**	**46.03**	**75.78**	**56.25**	**39.86**

The main results in Table 1 are obtained by training separately on each domain. To evaluate whether RoCA remains effective when all domains are combined, we train each method on the merged 37.5K samples (5 domains $\times$ 7.5K) on Qwen3-30B-A3B and OLMoE-1B-7B. As shown in Table 6,

RoCA achieves the highest average score of 74.35 on Qwen3-30B-A3B, outperforming the second-best method by 8.09 points. Notably, RoCA's HumanEval score on Qwen3-30B-A3B (76.83) even exceeds its single-domain counterpart (73.17 in Table 1), indicating that RoCA shows promising potential to benefit from richer training data by exploring more diverse expert combinations.

E General Capacity Evaluation on DeepSeek-V2-Lite

Table 7: General capability evaluation on DeepSeek-V2-Lite. Results show mean ± standard deviation across each method's five domain-specific checkpoints evaluated in Table 1. Bold and underlined scores denote the best and second-best results, respectively.

Method	MMLU	ARC-C	ARC-E	Hella Swag	Wino Grande	Avg.
Base model	55.48	46.67	74.41	77.90	70.56	65.01
PEFT for Dense Models						
MoLA	55.63±0.26	47.18±0.89	74.87±1.35	77.76±0.20	70.92±0.33	65.27±0.61
LoRAMoE	55.52±0.17	47.49±1.10	74.78±1.56	77.77±0.17	70.66±0.37	65.24±0.67
LoRA	<u>55.77</u>±0.44	48.09±1.05	<u>75.52</u>±1.25	77.93±0.10	70.78±0.28	65.62±0.63
AdaLoRA	55.39±0.13	46.62±0.43	74.70±0.42	77.79±0.04	70.78±0.18	65.06±0.24
PEFT for MoE Models						
ESFT	**55.87**±0.60	<u>48.14</u>±1.79	75.43±2.05	78.05±0.06	70.78±0.23	<u>65.65</u>±0.95
DR-LoRA	55.57±0.18	47.01±0.82	74.86±1.25	77.78±0.08	70.39±0.48	65.12±0.56
PERFT	55.38±0.21	47.12±0.87	75.06±1.25	**78.84**±0.28	<u>71.03</u>±0.24	65.49±0.57
RoCA	54.60±1.33	**50.22**±1.17	**75.58**±1.72	<u>78.18</u>±0.59	**71.24**±1.20	**65.96**±1.20

Table 7 reports general capability evaluation results on DeepSeek-V2-Lite, complementing the Qwen3-30B-A3B results in the main text (Table 2). RoCA again achieves the highest average score, consistent with the findings on Qwen3-30B-A3B.

F Case Study: Code Generation

We present the full code generation case discussed in Section 4.2 (Figures 6a and 6d). The task is HumanEval/96. Both the base model and RoCA generate the same prefix `for num in range(2, n):`, but diverge afterward.

Prompt.

```
def count_up_to(n):
    """Implement a function that takes an non-negative
    integer and returns an array of the first n
    integers that are prime numbers and less than n.
    for example:
    count_up_to(5) => [2,3]
    count_up_to(11) => [2,3,5,7]
    count_up_to(0) => []
    count_up_to(20) => [2,3,5,7,11,13,17,19]
    count_up_to(1) => []
    count_up_to(18) => [2,3,5,7,11,13,17]
    """
```

Base model output (pass@1 = 0).

```
def count_up_to(n):
    primes = []
    for num in range(2, n):
        if is_prime(num):
```

```
            primes.append(num)
    return primes
```

The base model calls an undefined function `is_prime()`, resulting in a runtime error.

RoCA output (pass@1 = 1).

```
def count_up_to(n):
    primes = []
    for num in range(2, n):
        for i in range(2, num):
            if (num % i) == 0:
                break
        else:
            primes.append(num)
    return primes
```

RoCA implements a correct trial-division primality check using a `for-else` construct, passing all test cases.

G Expert Utilization in the Pre-trained Model

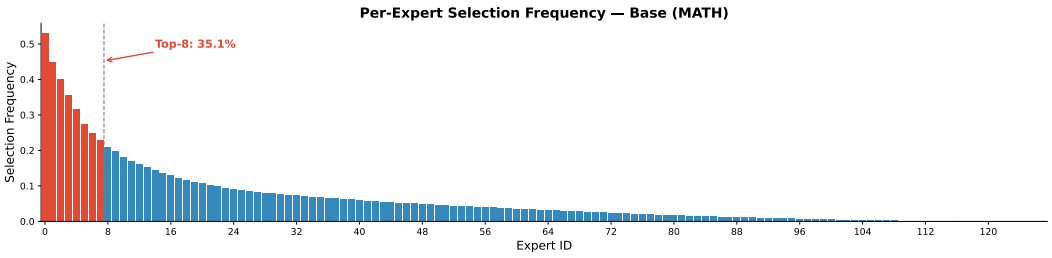


Figure 8: Expert selection frequencies of pre-trained Qwen3-30B-A3B on MATH-500.

Figure 8 shows expert selection frequencies for the pre-trained Qwen3-30B-A3B on MATH-500, sorted in descending order. Red bars mark the eight most frequently selected experts, and blue bars show the remaining experts. Selection frequency is the fraction of evaluated tokens routed to an expert. Since each token selects eight experts, the frequencies sum to 8 rather than 1. The eight most frequently selected experts account for 35.1% of all selections, computed as their summed frequencies divided by 8. This concentration provides context for the analysis of model capacity utilization in the main text.

H DOMAIN-ADAPTIVE INJECTION INTENSITY

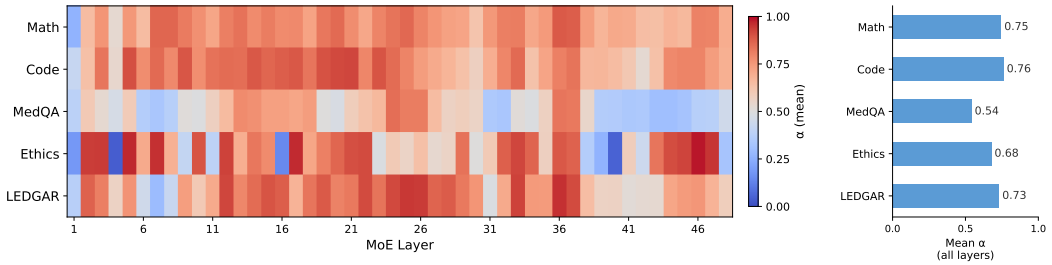


Figure 9: Per-layer entropy-gated α on Qwen3-30B-A3B across five domains. Left: heatmap of α per MoE layer (blue=low, red=high). Right: mean α averaged over all layers. MedQA receives the lowest α, consistent with its smallest performance gain over the base model.

We visualize the per-layer entropy-gated α values across all five domains on Qwen3-30B-A3B in Figure 9. The heatmap reveals that α varies substantially across domains: Math, Code, Ethics, and LEDGAR exhibit high α values, indicating that the domain expert contributes strongly on these tasks. In contrast, MedQA shows notably lower α (mean ≈ 0.54), meaning the entropy gate suppresses the domain expert's contribution on this domain. This aligns with the observation that MedQA is also the domain where all methods show the smallest improvement over the base model on Qwen3-30B-A3B in Table 1, suggesting that the base model's pre-trained routing is already relatively well-suited for medical QA and less additional knowledge injection is needed. The entropy gate automatically captures this, reducing intervention where it is least necessary.