RECONCILING ROUTING AND LEARNING FOR MIXTURE-OF-EXPERTS EXPANSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Expert expansion adds task-specific capacity to mixture-of-experts (MoE) models without increasing inference-time computation. However, experts receive task gradients only from tokens assigned by pretrained routing, which may exclude critical downstream learning signals and ultimately limit task performance. In this paper, we propose RECONEXP, a proactive expansion approach that addresses this misalignment in both expert initialization and training. Specifically, RECONEXP uses downstream gradients to rank pretrained experts and initialize the new expert from the highest-ranked expert. RECONEXP also distills pretrained expert outputs to expand learning beyond the inherited routing region. Together, these designs make the new expert responsive to downstream learning rather than solely to pretrained routing. Experiments show that RECONEXP improves the accuracy on BFCL over the strongest expansion by 5.3 points and reaches 97% of the Full-Finetune score on WebShop, while updating only 1.45% of the model parameters.

1 INTRODUCTION

Mixture-of-Experts (MoE) models scale parameter capacity through sparse conditional computation, activating only a small subset of experts for each token (Shazeer et al., 2017; Lepikhin et al., 2021; Fedus et al., 2022). By decoupling model capacity from per-token computation, this design has become increasingly common in large language models (DeepSeek-AI, 2024; Muennighoff et al., 2024). Beyond efficient scaling, the expert structure also provides a natural interface for adaptation: new experts can acquire downstream capabilities while the pretrained model remains frozen and the inference-time computation remains unchanged (Zhong et al., 2024; Wu et al., 2026; Wang et al., 2025). Expert expansion therefore offers a structural alternative to updating the full model.

Adding expert capacity (Zhong et al., 2024; Wu et al., 2026; Wang et al., 2025; Zhang et al., 2025; Wang et al., 2026), however, does not ensure that downstream optimization can use it effectively. Sparse routing gates ordinary expert updates: an expert receives task gradients only from tokens routed through it (Shazeer et al., 2017; Fedus et al., 2022). This restriction is particularly important in pretrained MoEs, whose routers have already developed highly non-uniform routing patterns (Muennighoff et al., 2024). In our measurements, downstream task states span 75.2% of a fixed partition of the pretrained expert-input space, yet the most frequently activated expert receives task gradients from only 18.0% of these regions, and activation frequency does not reliably identify experts most relevant to downstream improvement. We refer to this mismatch between pretrained routing and downstream learning as *routing–learning misalignment*.

This misalignment affects expansion at two stages. Under expert–router inheritance, donor selection determines the routing region in which the new expert begins training, yet pretrained usage need not identify the region containing the most informative downstream gradients. Even with a suitable initialization, subsequent learning remains restricted to states admitted by the inherited routing region. Effective expansion therefore requires both a task-relevant initialization and training support beyond the inherited routing region.

To this end, we propose RECONEXP, a proactive expansion approach that makes both stages responsive to downstream learning rather than solely to pretrained routing. *Learning-Guided Initialization* (LGI) combines downstream gradient signals with pretrained routing responses to rank pretrained experts, then copies the highest-ranked expert together with its router row. *Support-Expanding Distillation* (SED) distills outputs from frozen pretrained experts as off-route supervision, allowing

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.

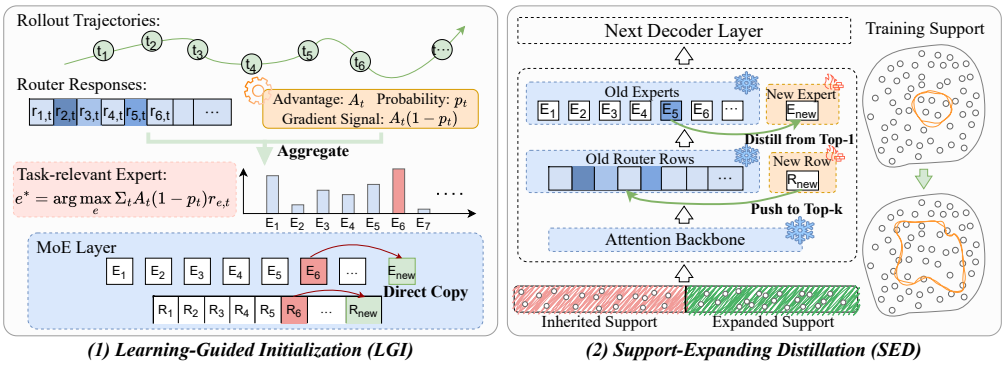


Figure 1: Overview of RECONEXP. **Left:** Learning-Guided Initialization combines downstream gradient signals with pretrained routing responses to rank donor experts and jointly copies the selected expert–router pair to provide a task-relevant initialization. **Right:** Support-Expanding Distillation distills pretrained expert outputs as off-route supervision and applies routing pressure to expand inherited training support. The pretrained MoE remains frozen.

the new expert to learn beyond its inherited routing region, and progressively introduces routing pressure as the new expert becomes competent on these off-route states.

Experiments on BFCL (Patil et al., 2025) and WebShop (Yao et al., 2022) support this mechanism. LGI selects donor experts better aligned with downstream learning than activation-frequency selection, while SED increases effective gradient support from 0.180 to 0.714 without a corresponding increase in routing traffic. Their interaction further indicates that support expansion is effective after a task-relevant initialization. These mechanism-level gains translate to downstream performance: relative to the matched LEGO baseline, RECONEXP improves BFCL overall accuracy by 5.3 points and reaches 97% of the Full-Finetune score on WebShop while updating only 1.45% of the model parameters.

Our main contributions are: (1) We identify *routing–learning misalignment* as a limitation of expert expansion: pretrained routing can misalign both the new expert's initialization and subsequent training support with downstream learning. (2) We introduce RECONEXP, combining LGI for downstream-gradient-guided initialization with SED for expanding learning beyond the inherited routing region. (3) Mechanism analyses, ablations, and end-task evaluations show that task-relevant initialization and broader training support jointly improve expansion without requiring increased routing traffic.

2 PRELIMINARIES

Sparse Mixture-of-Experts. We consider decoder-only transformers in which sparsely routed Mixture-of-Experts (MoE) layers replace the feed-forward sublayers (Shazeer et al., 2017; Fedus et al., 2022). At routed layer ℓ, a router matrix W_ℓ scores token state $h_{\ell,t}$,

$$z_{\ell,t} = W_\ell h_{\ell,t}, \qquad p_{\ell,t} = \mathrm{softmax}(z_{\ell,t}), \tag{1}$$

and activates only the k experts with the highest router probabilities. Let $\mathcal{T}_{\ell,t} = \mathrm{TopK}(p_{\ell,t})$ and let $\widetilde{p}_{\ell,t,e}$ denote the router probability renormalized over this selected set. The MoE output is

$$y_{\ell,t} = \sum_{e \in \mathcal{T}_{\ell,t}} \widetilde{p}_{\ell,t,e} E_{\ell,e}(h_{\ell,t}), \tag{2}$$

where $E_{\ell,e}$ is expert e at layer ℓ. Sparse routing therefore couples the model's computation graph to the router-selected expert set.

Task-facing expert expansion. We append one task-facing expert $n = N + 1$ and one router row to every routed layer, following the structural expansion setting of Zhong et al. (2024). All pretrained experts and router rows, attention modules, embeddings, and output layers remain frozen; only $\theta_{\ell,n}$ and $w_{\ell,n}$ are trainable. The active-expert budget k remains unchanged, requiring the new expert to compete under the pretrained sparse-routing rule. This setting isolates how pretrained routing constrains the downstream optimization of the new expert.

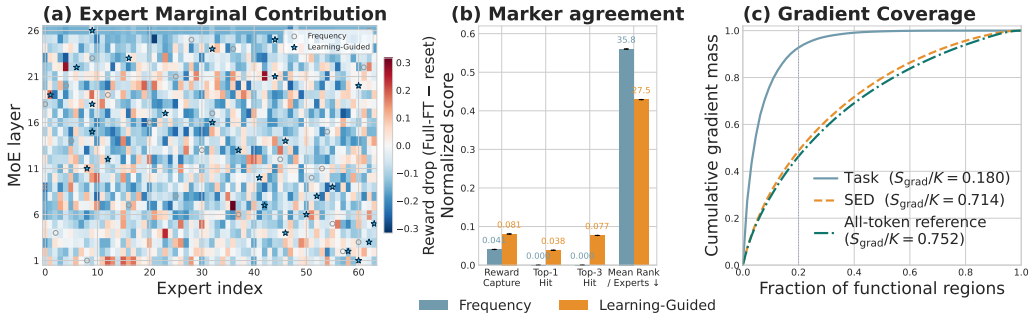


Figure 2: Diagnostics of routing–learning misalignment. **(a)** Full-Finetune rollback reward contributions vary substantially across layer–expert paths; circles and stars mark the donors selected by Frequency and LGI, respectively. **(b)** LGI improves RewardCapture from 0.041 to 0.081, achieves nonzero Top-1 and Top-3 hit rates, and lowers the mean rank from 35.8 to 27.5. **(c)** Under the fixed functional partition, ordinary task gradients are concentrated on far fewer regions ($\bar{S}^{\text{eff}}_{\text{grad}} = 0.180$) than SED gradients ($0.714$), motivating explicit expansion of inherited training support.

3 Motivation: Routing–Learning Misalignment in Expansion

Sparse routing restricts the new expert's task gradients to tokens assigned to it by pretrained routing. Let $g_{\ell,t} = \partial \mathcal{L}_{\text{task}} / \partial y_{\ell,t}$. Away from Top-$k$ boundaries,

$$\nabla_{\theta_{\ell,n}} \mathcal{L}_{task} = \mathbb{I}[n \in \mathcal{T}_{\ell,t}]\, \widetilde{p}_{\ell,t,n}\, J_{\theta_{\ell,n}} E_n(h_{\ell,t})^\top g_{\ell,t}. \tag{3}$$

Equation 3 shows that ordinary task gradients reach the new expert only on states admitted by the current routing region. Under expert–router inheritance, donor selection shapes the routing region inherited at initialization, while subsequent optimization remains limited to routed states. We therefore examine how pretrained routing affects expert initialization and subsequent training support.

Expert initialization under pretrained routing. The new router row must compete within an established, highly non-uniform routing geometry. Across $7{,}226{,}258$ token–layer states, the pretrained 64-expert router assigns a mean top-ranked probability of 0.1185, far above the uniform baseline of $1/64 = 0.0156$, with scores decaying by roughly two orders of magnitude across expert ranks (Figure 3). A randomly initialized router row is therefore unlikely to receive sufficient routing traffic and, consequently, sufficient task-gradient signal under Equation 3. Copying a pretrained expert together with its router row avoids this routing disadvantage, but makes the selection critical.

A natural heuristic is frequency (Zhong et al., 2024),

$$F_{\ell,e} = \frac{1}{T} \sum_t \mathbb{I}[e \in \mathcal{T}_{\ell,t}], \tag{4}$$

$F_{\ell,e}$ measures where the pretrained model already allocates computation, rather than where downstream optimization requires changes. We probe this mismatch with an independent Full-Finetune rollback diagnostic. For each layer–expert path, we restore the expert and its router row to their pretrained values while keeping other Full-Finetune parameters fixed, and measure the reward drop

$$\Delta R_{\ell,e} = R(M^{\text{FT}}; D) - R(M^{\text{FT}}_{\ell,e \leftarrow \text{PT}}; D), \tag{5}$$

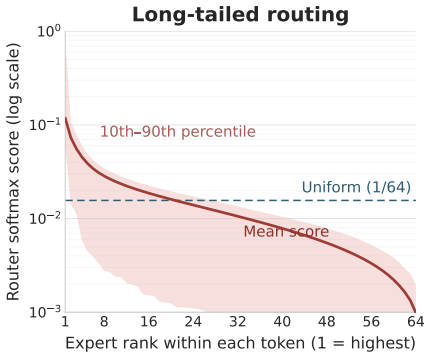


Figure 3: **Pretrained routing is strongly long-tailed.** Router scores decay by two orders of magnitude from head to tail, leaving a non-uniform routing geometry.

Figure 2(a–b) shows that frequently activated experts are not necessarily those with the largest rollback-based reward contributions. Frequency captures only 0.041 of the normalized positive rollback reward and yields no Top-1 or Top-3 matches, indicating that pretrained usage frequency provides a weak proxy for downstream learning relevance. This mismatch motivates donor selection that accounts explicitly for downstream learning signals rather than routing frequency alone.

Because rollback contribution is measured within the fine-tuned model and reflects interactions with the surrounding backbone and experts, we use it only as a comparative diagnostic rather than an absolute measure of expert quality. Appendix B provides the construction and evaluation details.

Training support under pretrained routing. A task-relevant initialization determines where the new expert starts, but not how broadly it can learn. Under sparse training, task gradients reach the new expert only on routed states, potentially concentrating its training support. We quantify this concentration by partitioning pretrained expert-input states at each layer into K fixed clusters, where $c_\ell(t) \in \{1, \dots, K\}$ denotes the cluster of token t. We use these fixed clusters as a proxy for functional regions. For each region c, we define the fraction of task-gradient mass it receives as

$$q_{\ell,c} = \frac{\sum_{t:c_\ell(t)=c} w^{\text{grad}}_{\ell,t}}{\sum_t w^{\text{grad}}_{\ell,t}}, \qquad \bar{S}^{\text{eff}}_\ell = \frac{1}{K} \exp\left(-\sum_{c=1}^{K} q_{\ell,c} \log q_{\ell,c} \right), \tag{6}$$

where $w^{\text{grad}}_{\ell,t} = \|\partial \mathcal{L}_{\text{task}} / \partial E_n(h_{\ell,t})\|_2$ is the task-gradient mass of token t. The normalized effective support $\bar{S}^{\text{eff}}_\ell$ measures how broadly task gradients are distributed across the K functional regions. During frequency-based expansion, only 20% of the functional regions account for 93.0% of the task-gradient mass, with $\bar{S}^{\text{eff}} = 0.180$ (Figure 2(c)). Appendix B details the fixed-partition probe.

Together, these observations expose the routing–learning misalignment underlying expert expansion: pretrained routing shapes the region inherited at initialization and restricts subsequent task gradients to routed states. As a result, critical downstream learning signals can remain inaccessible even after new capacity is added. Effective expansion therefore requires both a task-relevant initialization and training support beyond the inherited routing region.

4 METHOD

In this section, we propose RECONEXP, which combines Learning-Guided Initialization (LGI) and Support-Expanding Distillation (SED) to address the two stages jointly. LGI selects a donor expert–router pair using downstream gradient signals, while SED extends learning beyond the inherited routing region through off-route distillation from pretrained experts. Figure 1 summarizes the procedure, and Appendix C.3 provides the training pseudocode.

4.1 LEARNING-GUIDED INITIALIZATION (LGI)

Copying an expert–router pair provides an established routing region, but the donor selection determines which downstream learning signals are initially accessible to the new expert. Activation frequency captures where the pretrained router allocates computation, but not where critical downstream gradients concentrate. We therefore rank donor experts by the alignment between their pretrained routing responses and downstream gradient signals.

Downstream gradient signal. For each valid response token j, let s_j denote the logit of the sampled or supervised token. We define its downstream gradient signal as

$$g_j = -\frac{\partial \mathcal{L}_{\text{task}}}{\partial s_j}. \tag{7}$$

We use the token-logit gradient because it provides an objective-consistent signal at the same positions where routing responses are observed, without introducing expert-specific Jacobians into donor selection. The sign of g_j indicates whether the task objective reinforces or suppresses the token, while $|g_j|$ reflects the local update strength. Appendix C.1 gives objective-specific forms of this signal and discusses its relation to the full optimization gradient.

Projecting downstream gradients onto pretrained routing. Let $r^{(\ell)}_{j,e} \geq 0$ denote the frozen pretrained routing response of expert e to token j at layer ℓ. We combine this response with the downstream gradient signal to measure how strongly each expert's pretrained routing region overlaps with tokens emphasized by the downstream update:

$$S^{(\ell)}_e = \frac{\sum_{j \in \mathcal{J}} g_j \, r^{(\ell)}_{j,e}}{\sum_{j \in \mathcal{J}} |g_j| + \epsilon}, \qquad e^d_\ell = \arg\max_e S^{(\ell)}_e. \tag{8}$$

A high score indicates stronger alignment between pretrained routing responses and downstream gradient signals. We select the highest-scoring expert e_ℓ^d as the donor for layer ℓ. Unlike activation frequency, which captures only routing availability, LGI also accounts for where critical downstream gradients concentrate and whether those gradients favor or suppress the routed behavior. Appendix C.1 formalizes this distinction and shows that the resulting score provides a routing-level proxy for the downstream learning signals initially accessible to the new expert.

Expert–router pair inheritance. We jointly copy the selected expert and router row,

$$\theta_{\ell,n} \leftarrow \theta_{\ell,e_\ell^d}, \qquad w_{\ell,n} \leftarrow w_{\ell,e_\ell^d}. \tag{9}$$

Copying the expert preserves a pretrained function compatible with the frozen backbone, while copying its router row lets the new expert inherit the donor's routing region. Since the donor is selected for its alignment with downstream gradient signals, this initialization places the new expert where informative task gradients are more likely to be accessible from the start.

4.2 Support-Expanding Distillation (SED)

LGI provides a task-relevant initialization, but subsequent learning remains limited by the training support inherited from pretrained routing. Ordinary task gradients reach the new expert only on routed states, leaving learning signals outside this region inaccessible. SED expands this support by distilling outputs from pretrained experts as off-route supervision, then progressively introducing routing pressure as the new expert's relative competence improves.

Off-route distillation. For token t at layer ℓ, let $h_{\ell,t}$ denote the hidden state entering the MoE sublayer. We evaluate three experts on this same state: the frozen pretrained top-1 expert $e_{\ell,t}^\star$, the donor e_ℓ^d, and the new expert n, yielding

$$z_{\ell,t}^\star = E_{e_{\ell,t}^\star}(h_{\ell,t}), \quad z_{\ell,t}^d = E_{e_\ell^d}(h_{\ell,t}), \quad z_{\ell,t}^n = E_n(h_{\ell,t}), \tag{10}$$

where $e_{\ell,t}^\star$ is determined by the frozen pretrained router. To compare the new expert with the donor against the same pretrained reference, we define the normalized discrepancies

$$d_{\ell,t}^n = \frac{\|z_{\ell,t}^n - z_{\ell,t}^\star\|_2^2}{\|z_{\ell,t}^\star\|_2^2 + \epsilon}, \qquad d_{\ell,t}^d = \frac{\|z_{\ell,t}^d - z_{\ell,t}^\star\|_2^2}{\|z_{\ell,t}^\star\|_2^2 + \epsilon}. \tag{11}$$

We then define the relative competence of the new expert as

$$\alpha_{\ell,t} = \mathrm{sg}\left[\mathrm{clip}\left(\frac{d_{\ell,t}^d - d_{\ell,t}^n}{d_{\ell,t}^d + \epsilon}, 0, 1\right)\right], \tag{12}$$

which measures how much of the donor-to-reference discrepancy the new expert has closed. We use this competence estimate to weight the distillation objective,

$$\mathcal{L}_{\ell,t}^{\mathrm{dist}} = (1 - \alpha_{\ell,t}) d_{\ell,t}^n. \tag{13}$$

Unlike the task loss, this objective can be evaluated when $n \notin \mathrm{TopK}(p_{\ell,t})$, providing off-route supervision beyond the inherited routing region. When the frozen top-1 reference differs from the donor, the new expert is exposed to a complementary pretrained function; when they coincide, $d_{\ell,t}^d \approx 0$, and the same objective instead preserves the inherited donor behavior. As relative competence increases, the factor $(1 - \alpha_{\ell,t})$ progressively reduces this distillation signal. Appendix C.2 further discusses the boundary cases and scope.

Competence-aware routing pressure. As the new expert becomes competent on these states through off-route supervision, SED progressively introduces routing pressure to make it accessible under sparse routing. Let $p_{\ell,t}^n$ be the new-expert probability and $\bar{p}_{\ell,t}^{\mathrm{old}}$ be the mean probability of the frozen pretrained experts selected by Top-k. We define

$$\Delta_{\ell,t} = \mathrm{ReLU}\left(\frac{\bar{p}_{\ell,t}^{\mathrm{old}} - p_{\ell,t}^n}{\bar{p}_{\ell,t}^{\mathrm{old}} + \epsilon}\right), \qquad \mathcal{L}_{\ell,t}^{\mathrm{route}} = \alpha_{\ell,t} \Delta_{\ell,t}^2. \tag{14}$$

Routing pressure increases with $\alpha_{\ell,t}$ and vanishes once the new expert reaches the probability scale of the locally selected pretrained experts. SED therefore encourages the new expert to become sufficiently competitive for routing, rather than maximizing its routing frequency. This expands training support while avoiding unnecessary disruption to the pretrained routing geometry.

The complete objective is

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_{\text{SED}}\, \mathbb{E}_{\ell,t}\left[\mathcal{L}_{\ell,t}^{\text{dist}} + \mathcal{L}_{\ell,t}^{\text{route}}\right]. \tag{15}$$

Together, the two SED terms provide a competence-aware mechanism for expanding inherited training support. The distillation term $\mathcal{L}^{\text{dist}}$ provides off-route supervision beyond the inherited routing region, while $\mathcal{L}^{\text{route}}$ progressively increases routing access as the new expert becomes competent on these states. These states can then contribute complete task gradients under sparse routing. Importantly, SED does not specify the task-specific behavior to be learned: its auxiliary supervision is derived from frozen pretrained experts and serves to broaden where learning can occur. The downstream objective $\mathcal{L}_{\text{task}}$ remains responsible for determining which behaviors should be reinforced or suppressed, while SED expands the set of states on which those task-specific learning signals can reach the new expert.

5 Experiments

We evaluate ReconExp by testing the mechanism underlying routing–learning misalignment and its downstream consequences. Specifically, we ask whether (1) effective adaptation depends on training support rather than increased routing traffic alone, (2) LGI provides a task-relevant initialization, (3) SED expands inherited training support without merely increasing routing frequency, and (4) support expansion is most effective after task-relevant initialization. We then examine how these mechanisms translate to end-task adaptation, pretrained capability retention, schema robustness, and training cost.

5.1 Experimental Setup

We use DeepSeek-V2-Lite-Chat (DeepSeek-AI, 2024) and adapt it separately on ToolRL-4K (Qian et al., 2025) and WebShop (Yao et al., 2022). We evaluate end-task adaptation on BFCL (Patil et al., 2025) and WebShop, and use ToolRL-4K for controlled mechanism analyses and ablations. Pretrained capability retention is evaluated on MMLU (Hendrycks et al., 2021), PIQA (Bisk et al., 2020), ARC-C (Clark et al., 2018), and WinoGrande (Sakaguchi et al., 2020). Dataset splits and evaluation protocols are detailed in Appendix D, with prompts, parsers, and reward definitions provided in Appendices E and F.

We compare ReconExp with the Base Model, Full-Finetune, LEGO (Wu et al., 2026), and Expert-LoRA, a modified MoE-Sieve (Manzoni, 2026) baseline that applies LoRA only to the top-ranked expert. LEGO serves as the matched structural-expansion baseline: both methods append one expert and one router row at each routed layer under the same active-expert budget. Detailed baseline construction is provided in Appendix H.2. Diagnostic protocols, implementation details, and hyperparameters appear in Appendices B, H.1, and H.3, respectively.

5.2 End-Task Adaptation

Table 1 compares end-task adaptation across different trainable-parameter budgets, with Full-FT as the unconstrained adaptation reference, LEGO as the strongest matched structural-expansion baseline, and Expert-LoRA as a lower-capacity expert-side baseline. On BFCL, ReconExp reaches 49.25% overall accuracy, improving over LEGO (43.99%) and Expert-LoRA (41.33%) while narrowing the gap to Full-FT (54.57%). The gains over LEGO are also consistent on Non-Live AST and execution accuracy. The lower Live Accuracy and Irrelevance results are also observed under Full-FT, which may partly reflect limited coverage of these behaviors in ToolRL-4K rather than a limitation specific to expert expansion.

On WebShop, ReconExp achieves a score of 0.4627, compared with 0.4359 for LEGO, 0.4410 for Expert-LoRA, and 0.4754 for Full-FT. This corresponds to 97% of the Full-FT score while updating only the appended experts and router rows. Its strict success rate reaches 9.4%, exceeding LEGO (7.8%), Expert-LoRA (8.2%), and Full-FT (9.0%). Overall, ReconExp consistently improves over parameter-efficient baselines while approaching unconstrained adaptation performance. Additional

Table 1: End-task adaptation on BFCL and WebShop. Full-FT serves as the unconstrained adaptation reference, LEGO as the matched structural-expansion baseline, and Expert-LoRA as a lower-capacity expert-side baseline. RECONEXP and LEGO update only the appended experts and router rows. Higher values are better for all metrics.

Method	Berkeley Function Calling Leaderboard						WebShop	
	Overall acc	Non-Live AST	Non-Live Exec	Live Acc	Relevance	Irrelevance	Score	SR
Base	40.23%	38.46%	33.13%	42.04%	70.59%	60.60%	0.1547	3.8%
Full-FT	54.57%	59.50%	55.66%	40.36%	88.24%	40.26%	0.4754	9.0%
Expert-LoRA	41.33%	45.92%	33.38%	43.11%	76.47%	54.56%	0.4410	8.2%
LEGO	43.99%	50.19%	45.62%	35.16%	58.82%	33.50%	0.4359	7.8%
RECONEXP	49.25%	53.21%	49.18%	37.91%	76.47%	33.05%	0.4627	9.4%

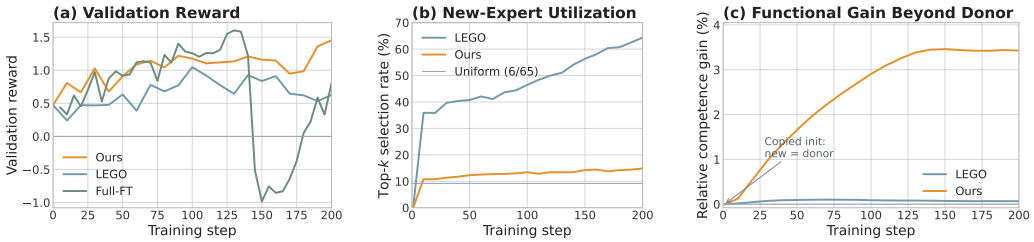


Figure 4: Training dynamics on ToolRL-4K. **(a)** Validation reward during training. **(b)** Top-k selection rate of the new expert, with the uniform $6/65$ reference. **(c)** Functional competence gain across fixed functional regions, relative to the copied donor initialization.

results with RLOO (Ahmadian et al., 2024) in Appendix C.4 show a similar overall trend, indicating that the gains are not specific to GRPO.

5.3 TRAINING DYNAMICS: TRAINING SUPPORT BEYOND ROUTING TRAFFIC

Figure 4 contrasts routing frequency with the new expert's actual learning progress. RECONEXP improves steadily in validation reward, while LEGO converges to a lower level and Full-Finetune exhibits a pronounced late-stage collapse. Despite this, LEGO increases the new expert's Top-k selection rate to approximately 64%, whereas RECONEXP remains near 15%, only moderately above the uniform $6/65$ reference. More routing therefore does not translate directly into better adaptation: RECONEXP achieves a competence gain of approximately 3.4% over the copied donor, while LEGO remains close to its initialization despite substantially higher routing traffic. These dynamics suggest that routing frequency alone is not a reliable proxy for effective training support; what matters is whether downstream learning signals reach states that support task-relevant improvement. Additional training curves for reward, response length, and policy entropy are provided in Appendix G.

5.4 MECHANISM: PRETRAINED ROUTING RESTRICTS TRAINING SUPPORT

LGI provides a task-relevant initialization. We evaluate whether LGI better aligns donor initialization with downstream learning under the same independent Full-FT rollback diagnostic. As shown in Figure 2(a,b), LGI increases RewardCapture from 0.041 to 0.081, achieves Top-1 and Top-3 hit rates of 0.038 and 0.077, and reduces the mean donor rank from 35.8 to 27.5. It also selects a positive-contribution donor in $8/26$ layers, compared with $3/26$ for frequency. Together, these results show that incorporating downstream gradient signals yields expert initialization better aligned with downstream learning than those obtained from pretrained usage frequency alone.

SED expands inherited training support without increasing routing traffic. Figure 2(c) shows that SED increases effective gradient support from 0.180 to 0.714 under the same fixed functional partition. Figure 5 further tests whether this broader support can be explained by increased routing traffic. At step 200, SED and No-SED have nearly identical new-expert Top-k selection rates (14.9% versus 14.5%), while SED achieves substantially broader functional-region coverage (40.8% versus

Table 2: 2×2 factorial ablation of LGI and SED. Removing either component degrades the full method. SED is beneficial when paired with LGI but not when LGI is absent, consistent with the intended ordering of task-relevant initialization followed by expansion of inherited training support.

Variant	ToolRL-4K Val	WebShop Score	WebShop SR
ReconExp (Ours)	**1.454**	**0.4627**	**9.4%**
− w/o LGI	0.951	0.4267	7.0%
− w/o SED	1.206	0.4476	8.2%
− w/o LGI and SED	1.047	0.4359	7.8%

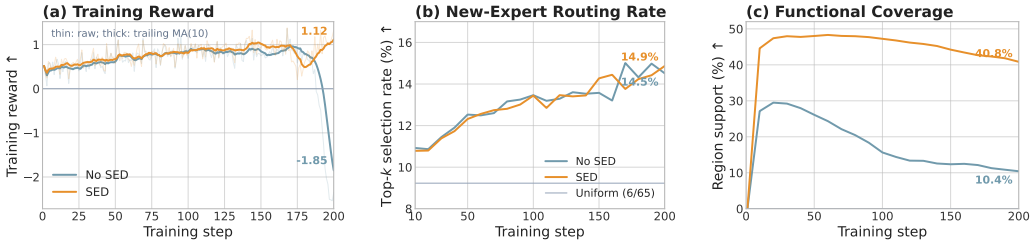


Figure 5: Effect of Support-Expanding Distillation (SED) on training support. The paired runs differ only in whether SED is used. **(a)** Training reward. **(b)** Top-k selection rate of the new expert. **(c)** Functional-region coverage measures the breadth of competence gains across regions.

10.4%). The two runs also exhibit markedly different reward trajectories. With routing frequency nearly matched, the broader coverage cannot be attributed to simply routing more tokens through the new expert. Instead, SED enables learning across a broader set of functional regions beyond the support inherited at initialization. These results provide direct mechanism evidence that SED expands inherited training support rather than merely increasing routing traffic.

5.5 Component Interaction: Support Expansion Depends on Initialization

The 2×2 factorial ablation in Table 2 examines whether the benefit of SED depends on task-relevant initialization. With LGI, adding SED improves ToolRL validation reward from 1.206 to 1.454 and WebShop score from 0.4476 to 0.4627, with success rate increasing from 8.2% to 9.4%. Without LGI, the same intervention instead reduces these metrics, including ToolRL reward from 1.047 to 0.951 and WebShop score from 0.4359 to 0.4267. This interaction suggests that expanding training support is most effective after the new expert is initialized in a routing region aligned with downstream learning. SED is not entirely independent of this initialization: its relative-competence signal is calibrated against the donor output, so the donor defines both the starting function and the reference point for when distillation is reduced and routing pressure is introduced. When the donor is weakly aligned with downstream learning, this competence signal may be calibrated around a less useful starting point, making broader off-route supervision less effective. LGI therefore benefits SED not only by providing a task-relevant starting point, but also by anchoring its competence signal to a donor better aligned with downstream learning.

5.6 Retention and Out-of-Domain Transfer

Table 3 evaluates pretrained capability retention and out-of-domain transfer. Across MMLU, PIQA, and ARC-C, all methods remain close to the Base model, while WinoGrande shows a clearer contrast: Full-FT drops from 54.30% to 52.72%, whereas ReconExp reaches 55.33% with the shared backbone and pretrained experts frozen. However, LEGO falls to 49.88% under the same frozen-backbone setting, indicating that behavioral interference can still arise from changes in the expert–router pathway. After ToolRL adaptation, all parameter-efficient methods improve over the Base model on WebShop, but Expert-LoRA shows weaker transfer, reaching only 0.1922 compared with 0.2605 for ReconExp. ReconExp further achieves a success rate of 5.8%, below Full-FT in score but above LEGO in strict success rate. Overall, these results suggest that ReconExp can preserve pretrained capabilities while transferring more effectively than Expert-LoRA in this setting, although freezing pretrained parameters alone does not guarantee behavioral preservation.

Table 3: Held-out evaluation after ToolRL-4K adaptation. MMLU, PIQA, ARC-C, and WinoGrande probe general-capability retention, while WebShop is evaluated as an out-of-domain agentic task for this adaptation run. Higher values are better for all metrics.

Method	MMLU	PIQA	ARC-C	WinoGrande	WebShop Score	WebShop SR
Base	52.75%	80.96%	70.99%	54.30%	0.1547	3.8%
Full-FT	52.87%	81.01%	69.97%	52.72%	0.2825	6.2%
Expert-LoRA	52.95%	80.90%	70.31%	53.75%	0.1922	3.6%
LEGO	53.04%	81.12%	70.90%	49.88%	0.2704	4.4%
RECONEXP	52.63%	81.39%	69.97%	55.33%	0.2605	5.8%

Table 4: WebShop action-schema switch. "Before" and "After" report scores before and after the controlled schema change.

Method	Before	After	Drop (%)
Full-FT	0.475	0.264	44.4
Expert-LoRA	0.441	0.207	53.1
LEGO	0.436	0.159	63.5
RECONEXP	0.463	0.288	37.8

Table 5: Training cost comparison across methods. SED incurs only a modest increase while maintaining a comparable token-normalized computational cost.

Method	Params	Relative Cost	GPU-h / Mtoken
Full-FT	15.71 B	1.00×	1.77
Expert-LoRA	5.75 M	0.95×	1.65
LEGO	228.38 M	0.97×	1.92
RECONEXP	228.38 M	1.05×	1.82

5.7 SCHEMA ROBUSTNESS AND TRAINING COST

The controlled WebShop schema switch from `<webshop>` to `[tool_call]` probes sensitivity to the interface used during adaptation. RECONEXP retains 62.2% of its pre-switch score, compared with 55.6% for Full-FT. This smaller degradation may partly reflect the broader functional support induced by SED, together with the fact that the pretrained backbone and expert pool remain frozen and available under sparse routing. While this single schema switch does not establish general robustness, it suggests less dependence on the exact training interface.

Table 5 further shows that off-route distillation incurs only a modest additional training cost. RECONEXP updates 228.38M parameters, 1.45% as many trainable parameters as Full-FT, while its measured relative training cost is $1.05\times$. This comparison uses the same micro-batch size across methods for consistency, so it does not exploit the additional memory headroom available when optimizing fewer parameters. The remaining cost is dominated by rollout generation and forward computation through the frozen pretrained model, which are largely unchanged across adaptation strategies. Thus, RECONEXP substantially reduces the trainable parameter budget while retaining comparable system-level training cost under the matched configuration.

6 CONCLUSION

We study routing–learning misalignment in expert expansion: adding task-specific capacity is insufficient when pretrained routing constrains both expert initialization and the task gradients available during subsequent training. RECONEXP addresses this mismatch through proactive expansion. Learning-Guided Initialization (LGI) uses downstream gradients to rank pretrained experts and initialize the new expert from the highest-ranked one, while Support-Expanding Distillation (SED) distills pretrained expert outputs to expand learning beyond the inherited routing region.

Experiments support this mechanism. LGI provides an initialization better aligned with downstream learning, while SED broadens the regions receiving learning signals without a corresponding increase in routing traffic. Their interaction further indicates that expanding learning beyond the inherited routing region is most effective after a task-relevant initialization. Accordingly, RECONEXP consistently improves over the strongest matched expert-expansion baseline and reaches 97% of the Full-Finetune score on WebShop while updating only 1.45% as many parameters. More broadly, these results suggest that effective expert expansion depends not only on adding capacity, but also on making the new expert responsive to downstream learning rather than solely to pretrained routing.

AI USE STATEMENT

We used generative AI tools to assist with conceptual development, mathematical formulation and proof writing, hypothesis refinement, experimental feedback, implementation, dataset cleaning and reformatting, result interpretation, language editing, and literature discovery. We did not use generative AI to provide critical ingredients for mathematical proofs or to design the research methodology. Synthetic-data generation, translation, and qualitative/thematic data analysis were not applicable. The authors reviewed and verified all AI-assisted content, including technical claims, mathematical formulations, experimental results, interpretations, and citations, and take responsibility for the final paper.

ETHICS STATEMENT

This study complies with the ICLR Code of Ethics. Our experiments use existing benchmarks and do not involve human subjects or the collection of new personally identifiable or sensitive data.

REPRODUCIBILITY STATEMENT

All datasets are publicly available (Section 5.1); Appendix C.3 and Appendices D–F and H document the data, prompts/parsers, rewards, training procedure, implementation, evaluation protocols, and hyperparameters. Code will be released upon acceptance.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.662.

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

Wuyang Chen, Yanqi Zhou, Nan Du, Yanping Huang, James Laudon, Zhifeng Chen, and Claire Cui. Lifelong language pretraining with distribution-specialized experts. In *International Conference on Machine Learning*, pp. 5383–5395, 2023.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

DeepSeek-AI. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

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.

Jinghan He, Haiyun Guo, Kuan Zhu, Zihan Zhao, Ming Tang, and Jinqiao Wang. SEEKR: Selective attention-guided knowledge retention for continual learning of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3254–3266, 2024.

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*, 2021.

ZhiYan Hou, Haiyun Guo, Haokai Ma, Yandu Sun, Yonghui Yang, and Jinqiao Wang. PASs-MoE: Mitigating misaligned co-drift among router and experts via pathway activation subspaces for continual learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31959–31972, 2026.

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.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. Sparse upcycling: Training mixture-of-experts from dense checkpoints. In *International Conference on Learning Representations*, 2023.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

Andrea Manzoni. MoE-Sieve: Routing-guided LoRA for efficient MoE fine-tuning. *arXiv preprint arXiv:2603.24044*, 2026.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. OLMoE: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024.

Taishi Nakamura, Takuya Akiba, Kazuki Fujii, Yusuke Oda, Rio Yokota, and Jun Suzuki. Drop-upcycling: Training sparse mixture of experts with partial re-initialization. In *International Conference on Learning Representations*, 2025.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Yujia Qin, Jiajie Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. ELLE: Efficient lifelong pre-training for emerging data. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2789–2810, 2022.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

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

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.

Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Rozière, Jacob Kahn, Daniel Li, Wen-tau Yih, Jason Weston, and Xian Li. Branch-Train-MiX: Mixing expert LLMs into a mixture-of-experts LLM. *arXiv preprint arXiv:2403.07816*, 2024.

Ruizhe Wang, Yucheng Ding, Xiao Liu, Yaoxiang Wang, Peng Cheng, Baining Guo, Zhengjun Zha, and Yeyun Gong. Beyond sunk costs: Boosting LLM pre-training efficiency via orthogonal growth of mixture-of-experts. In *International Conference on Machine Learning*, 2026.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, 2023.

Zhilin Wang, Yafu Li, Xiaoye Qu, and Yu Cheng. SEE: Continual fine-tuning with sequential ensemble of experts. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 7418–7432, 2025.

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

Jiawei Wu, Junlin Chen, Daize Dong, Haolong Jia, Souvik Kundu, Muhao Chen, and Hongyi Wang. LEGO: Expert expansion for continual learning of mixture-of-experts language models. In *SCALE Workshop at the International Conference on Machine Learning*, 2026.

Zhihe Yang, Xufang Luo, Zilong Wang, Dongqi Han, Zhiyuan He, Dongsheng Li, and Yunjian Xu. Do not let low-probability tokens over-dominate in RL for LLMs. In *International Conference on Learning Representations*, 2026.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Xue Zhang, Yunlong Liang, Fandong Meng, Songming Zhang, Yufeng Chen, Jinan Xu, and Jie Zhou. Less, but better: Efficient multilingual expansion for LLMs via layer-wise mixture-of-experts. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17948–17963, 2025.

Haotian Zhao, Songlin Zhou, Yuxin Zhang, Stephen S.-T. Yau, Wenyu Zhang, Lun Tian, Tianshu Zhu, Yifeng Huang, Yucheng Zeng, Jingnan Gu, Daxiang Dong, and Jianmin Wu. AEM: Adaptive entropy modulation for multi-turn agentic reinforcement learning. *arXiv preprint arXiv:2605.00425*, 2026.

Shanshan Zhong, Shanghua Gao, Zhongzhan Huang, Wushao Wen, Marinka Zitnik, and Pan Zhou. MoExtend: Tuning new experts for modality and task extension. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pp. 494–505, 2024.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. LLaMA-MoE: Building mixture-of-experts from LLaMA with continual pre-training. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15913–15923, 2024.

A Related Work

Continual learning and modular growth. Continual learning preserves useful behavior through regularization, replay, parameter isolation, or architectural growth (Kirkpatrick et al., 2017; He et al., 2024; Wang et al., 2023; Rusu et al., 2016; Qin et al., 2022). MoE architectures are well suited to growth because routing can preserve frozen computation while directing selected inputs to new modules. Lifelong-MoE, SEE, and LayerMoE add or allocate experts for new distributions or tasks (Chen et al., 2023; Wang et al., 2025; Zhang et al., 2025); recent MoE–LoRA work also examines router–expert co-drift (Hou et al., 2026). We focus on reconciling pretrained routing with downstream learning through task-relevant expert initialization and expansion of inherited training support, rather than on capacity allocation itself.

Parameter-efficient adaptation of pretrained MoEs. LoRA adapts frozen weights through trainable low-rank updates (Hu et al., 2022). In pretrained MoEs, ESFT selects experts according to their relevance to the downstream task and directly updates only the selected expert parameters (Wang et al., 2024). MoE-Sieve instead uses pretrained routing statistics to identify frequently activated experts and attaches LoRA modules to the selected experts, while also adapting attention, router, and shared-expert components (Manzoni, 2026). These methods exploit pretrained expert specialization to reduce the adaptation budget while retaining the pretrained routing structure. Our work instead studies expert expansion, focusing on how pretrained routing constrains the initialization and training support of newly added experts.

Expert construction and expansion. Sparse Upcycling, LLaMA-MoE, Branch-Train-MiX, and Drop-Upcycling construct or diversify expert pools from dense or separately trained components (Komatsuzaki et al., 2023; Zhu et al., 2024; Sukhbaatar et al., 2024; Nakamura et al., 2025). MoExtend provides an earlier routing-aware approach to expert expansion based on expert responses to target samples (Zhong et al., 2024). LEGO further incorporates routing into both initialization and subsequent training: it initializes new experts by round-robin replication of pretrained experts together with their corresponding router entries, then applies target-side routing pressure to steer new-domain tokens toward the added capacity and anchor-side stabilization to preserve pretrained behavior (Wu et al., 2026). Our method instead makes both stages responsive to downstream learning: it combines downstream gradient signals with pretrained routing responses to select a task-relevant donor expert–router pair, then expands the new expert's inherited training support through off-route distillation. Beyond Sunk Costs expands a converged MoE more globally by duplicating the expert pool and increasing active capacity (Wang et al., 2026); we add task-facing expert capacity while retaining the original top-k budget.

B Diagnostic Definitions and Protocols

Rollback construction. The rollback diagnostic resets an expert's gate, up, and down projections and its corresponding router row to pretrained values, evaluates ToolRL validation examples greedily with $n = 1$, and then restores the tensors. For a selector with donor d_ℓ, positive RewardCapture is defined as

$$\mathrm{RC}(d) = \frac{\sum_\ell [\Delta R_{\ell,d_\ell}]_+}{\sum_\ell \max_e [\Delta R_{\ell,e}]_+}. \tag{16}$$

The Top-j hit rate is the fraction of layers in which the selected donor ranks among the j largest rollback values.

Gradient-source probe. We use the fixed K-means partition defined in Section 3 to measure how broadly different gradient sources reach the new expert. For ordinary task gradients, we use the unclipped probe loss

$$\mathcal{L}_{\mathrm{probe}} = -\frac{1}{T} \sum_{t \in \mathcal{V}} A_t \log \pi_\theta(y_t \mid x, y_{<t}), \tag{17}$$

where T is the number of valid probe tokens. At each new-expert output, we record the gradient vector induced by $\mathcal{L}_{\mathrm{probe}}$ and define its mass as

$$w^{\mathrm{grad}}_{\ell,t} = \left\| \frac{\partial \mathcal{L}_{\mathrm{probe}}}{\partial E_n(h_{\ell,t})} \right\|_2. \tag{18}$$

We aggregate this mass within each fixed functional region and compute the normalized effective support using Equation 6.

For the SED comparison in Figure 2(c), we set $K = 64$ and repeat the same procedure using gradients induced by the SED objective at the corresponding new-expert outputs on ToolRL. The probe therefore compares where task and SED gradients are available under an identical functional partition. It measures local gradient availability before optimizer preconditioning, rather than parameter-update magnitude.

Functional competence gain and coverage. We evaluate the relative-competence measure in Equation 12 over the fixed functional partition used in Section 3. For each region center $\mu_{\ell,c}$, let $y^{\star}_{\ell,c}$, $y^{d}_{\ell,c}$, and $y^{n}_{\ell,c}$ denote the outputs of the frozen pretrained Top-1 expert, the donor, and the new expert, respectively. We compute their normalized discrepancies as in Equation 11, with $\epsilon = 10^{-8}$, and let $a_{\ell,c} \in [0, 1]$ denote the corresponding relative competence from Equation 12. We report the mean functional competence gain as

$$G_{\text{comp}} = \frac{1}{LK} \sum_{\ell=1}^{L} \sum_{c=1}^{K} a_{\ell,c}. \tag{19}$$

To distinguish gains concentrated in a small number of regions from those distributed broadly across the functional space, we further use $a_{\ell,c}$ as non-negative region weights in the normalized effective-support measure of Equation 6:

$$C_{\text{comp}} = \frac{1}{L} \sum_{\ell=1}^{L} \bar{S}^{\text{eff}}_{\ell}(a). \tag{20}$$

Thus, G_{comp} measures the average magnitude of functional improvement over the donor, whereas C_{comp} measures how broadly that improvement is distributed across functional regions. Both metrics are reported as percentages.

C Additional Method Derivations

C.1 Learning-Guided Initialization (LGI): Donor Selection

This section develops the optimization interpretation of Equation 8. We first formulate donor selection in terms of the downstream objective and then instantiate the criterion for policy optimization. Finally, we relate the score to activation frequency and explain why it serves as a proxy for the new expert's initial training support.

Objective-consistent donor scoring. For each valid response token j, let s_j denote the logit of the sampled or supervised token. Rather than introducing a separate token-weighting rule for donor selection, we derive the signal directly from the complete task objective:

$$g_j = -\frac{\partial \mathcal{L}_{\text{task}}}{\partial s_j}. \tag{21}$$

Because g_j is evaluated at the same token positions as the pretrained routing responses, the two signals can be aligned directly. This definition also preserves the weighting induced by $\mathcal{L}_{\text{task}}$, including masking, sequence aggregation, importance weighting, and clipping when applicable.

Given the frozen pretrained routing response $r^{(\ell)}_{j,e}$ of expert e, we define

$$S^{(\ell)}_{e} = \frac{\sum_j g_j r^{(\ell)}_{j,e}}{\sum_j |g_j| + \epsilon}. \tag{22}$$

The score is high when an expert responds strongly at token positions emphasized in the direction favored by the downstream objective.

This criterion intentionally operates at the token-logit level rather than estimating the full parameter gradient of each candidate expert. A parameter-level score would additionally depend on routing

weights and expert-specific Jacobians, making donor selection sensitive to the local parameterization of each expert. In contrast, the token-logit gradient provides a common optimization signal that can be compared across pretrained routing regions. The trade-off is that $S_e^{(\ell)}$ captures local alignment with the downstream objective, rather than predicting the exact parameter update or final task performance. We therefore use it as a routing-level proxy for where downstream learning signals are initially accessible to the new expert.

Policy-gradient specialization. The token-level signal admits a simple interpretation under policy optimization. For the local policy term

$$\mathcal{L}_{g,i,t} = -A_{g,i} \log p_{g,i,t}, \tag{23}$$

let ℓ_{g,i,t,y_t} denote the logit of the sampled token. Since

$$\frac{\partial \log p_{g,i,t}}{\partial \ell_{g,i,t,y_t}} = 1 - p_{g,i,t}, \tag{24}$$

the corresponding downstream gradient signal is

$$g_{g,i,t} = -\frac{\partial \mathcal{L}_{g,i,t}}{\partial \ell_{g,i,t,y_t}} = A_{g,i}(1 - p_{g,i,t}). \tag{25}$$

Here, $A_{g,i}$ determines whether the sampled trajectory is reinforced or suppressed, while $1 - p_{g,i,t}$ reflects the local sensitivity of the sampled-token probability. Thus, positive-advantage trajectories contribute positive evidence to donor selection, whereas negative-advantage trajectories contribute in the opposite direction.

Equation 25 is the unclipped local form used for interpretation. In practice, $g_{g,i,t}$ is obtained from the complete task objective as in Equation 21, so importance ratios, clipping, and other objective weights are already reflected in the gradient.

For approximately centered GRPO advantages,

$$\frac{1}{|g|} \sum_{i \in g} A_{g,i} \approx 0, \tag{26}$$

Equation 22 becomes contrastive within each prompt group: a donor receives a high score when its pretrained routing responses are stronger on trajectories favored by the policy update and weaker on trajectories that are suppressed. This explains why LGI captures downstream update structure beyond activation frequency alone.

Normalization in score aggregation. Raw trajectory-level scores can differ substantially in scale because sampled tokens may have different amounts of headroom for increasing their probabilities. Without normalization, trajectories with systematically larger headroom can dominate the aggregated score even when their relative advantage signals are comparable. We therefore normalize each trajectory by its mean sampled-token headroom before aggregation, so that expert ranking is driven by relative downstream learning demand rather than trajectory-specific score scale.

For trajectory (g, i), define its mean sampled-token headroom

$$\mu_{g,i} = \frac{1}{T_{g,i}} \sum_t (1 - p_{g,i,t}). \tag{27}$$

Ignoring the numerical stabilizer ϵ, the local policy-gradient signal can then be factorized as

$$\frac{A_{g,i}}{T_{g,i}}(1 - p_{g,i,t}) = A_{g,i} \mu_{g,i} \alpha_{g,i,t}, \qquad \alpha_{g,i,t} = \frac{1 - p_{g,i,t}}{\sum_u (1 - p_{g,i,u}) + \epsilon}. \tag{28}$$

Here, $\mu_{g,i}$ introduces a trajectory-level confidence scale, whereas $\alpha_{g,i,t}$ preserves the relative emphasis on lower-probability tokens within the trajectory. We retain the latter while removing the former, so that trajectory-level credit is determined by advantage rather than average policy confidence. This separation is consistent with recent observations that probability-dependent gradient magnitudes can systematically bias RL updates toward low-probability tokens, and that policy uncertainty can exhibit meaningful response-level structure (Yang et al., 2026; Zhao et al., 2026).

Trajectory-level expert exposure is aggregated within each prompt group using signed advantages and normalized by the total absolute advantage,

$$S_{g,e}^{(\ell)} = \frac{\sum_{i \in g} A_{g,i} R_{g,i,e}^{(\ell)}}{\sum_{i \in g} |A_{g,i}| + \epsilon}. \tag{29}$$

The sign of $A_{g,i}$ preserves the distinction between relatively successful and unsuccessful trajectories, while the denominator removes the overall scale of the group's advantages. Donor selection therefore depends primarily on routing–outcome alignment within a prompt group, rather than on the group's reward variance or advantage magnitude. Under GRPO, this normalization also largely cancels the group-level standard-deviation scaling of advantages.

Finally, the normalized group scores are averaged uniformly,

$$S_e^{(\ell)} = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} S_{g,e}^{(\ell)}. \tag{30}$$

Uniform averaging gives each prompt group equal influence, preventing groups with larger advantage scales or greater reward dispersion from dominating the ranking. Together, the three normalization steps remove response-length, advantage-scale, and prompt-scale effects. The resulting donor score primarily reflects whether an expert consistently responds to tokens with strong downstream gradient signals in relatively better trajectories.

From activation frequency to task-relevant initialization. To clarify how LGI differs from frequency-based selection, define the average pretrained routing response

$$F_e^{(\ell)} = \mathbb{E}_j[r_{j,e}^{(\ell)}]. \tag{31}$$

$F_e^{(\ell)}$ measures how broadly expert e is exposed under the pretrained routing geometry. It characterizes routing availability, but does not account for each state's relevance to the downstream update.

Let $w_j = |g_j|$. Weighting the same routing response by downstream gradient magnitude gives the unsigned gradient–routing score

$$C_e^{(\ell)} = \frac{\mathbb{E}_j[w_j r_{j,e}^{(\ell)}]}{\mathbb{E}_j[w_j]}. \tag{32}$$

Using

$$\mathbb{E}[w r_e] = \mathbb{E}[w]\mathbb{E}[r_e] + \mathrm{Cov}(w, r_e), \tag{33}$$

we obtain

$$C_e^{(\ell)} = F_e^{(\ell)} + \frac{\mathrm{Cov}\left(|g_j|, r_{j,e}^{(\ell)}\right)}{\mathbb{E}_j[|g_j|]}. \tag{34}$$

The second term captures information unavailable from frequency alone. If routing responses and downstream gradient magnitudes are independent, the covariance vanishes and the unsigned score $C_e^{(\ell)}$ reduces to $F_e^{(\ell)}$. The signed LGI score additionally depends on directional consistency. This criteria differ when downstream gradient signals are distributed non-uniformly across the pretrained expert space.

This relation can also be expressed through the gradient-enrichment factor

$$Q_e^{(\ell)} = \frac{\mathbb{E}_j[|g_j| r_{j,e}^{(\ell)}]}{\mathbb{E}_j[|g_j|]\mathbb{E}_j[r_{j,e}^{(\ell)}] + \epsilon}. \tag{35}$$

Ignoring ϵ,

$$C_e^{(\ell)} = F_e^{(\ell)} Q_e^{(\ell)}. \tag{36}$$

Here, F_e captures routing availability, while Q_e measures the enrichment of downstream gradient magnitude within that routing region. A frequently activated expert need not be a useful donor if most states routed to it require little downstream correction. Conversely, a less frequently activated expert can be preferable when its routing region disproportionately covers states with strong downstream gradient signals.

Signed objectives introduce one additional component:

$$D_e^{(\ell)} = \frac{\mathbb{E}_j[g_j r_{j,e}^{(\ell)}]}{\mathbb{E}_j[|g_j| r_{j,e}^{(\ell)}] + \epsilon}. \tag{37}$$

Combining the three terms gives

$$S_e^{(\ell)} \approx F_e^{(\ell)} Q_e^{(\ell)} D_e^{(\ell)}. \tag{38}$$

The factors represent routing availability, gradient enrichment, and directional consistency, respectively. Frequency-based selection retains only F_e. Policy optimization additionally uses D_e to distinguish experts exposed to similar amounts of downstream gradient signal but different mixtures of reinforcing and suppressing trajectories.

Why donor selection predicts initial training support. The preceding factorization characterizes where downstream gradient signals lie in the pretrained routing geometry. We next relate this statistical criterion to the trainability of the new expert. At routed layer ℓ, the ordinary task gradient of new expert n is

$$\nabla_{\theta_{\ell,n}} \mathcal{L}_{task} = \sum_j \mathbb{I}[n \in \mathcal{T}_{\ell,j}] \widetilde{p}_{\ell,j,n} J_{\theta_{\ell,n}} E_{\ell,n} (h_{\ell,j})^\top \delta_{\ell,j}, \tag{39}$$

where

$$\delta_{\ell,j} = \frac{\partial \mathcal{L}_{task}}{\partial y_{\ell,j}}. \tag{40}$$

A state can update the new expert only if it carries a downstream gradient signal and is routed through that expert. The inherited routing region therefore determines which signals are available during early training.

Learning-Guided Initialization (LGI) initializes the new expert and its router row jointly:

$$(\theta_{\ell,n}^{(0)}, w_{\ell,n}^{(0)}) = (\theta_{\ell,e}, w_{\ell,e}). \tag{41}$$

Copying the router row places the new expert near the donor's pretrained routing region, subject to the changes in softmax normalization and Top-k competition introduced by duplication. Consequently, $r_{j,e}^{(\ell)}$ indicates where the new expert is likely to receive ordinary task gradients at initialization.

Under the mild assumption that local Jacobian scales do not vary arbitrarily across copied donors,

$$\sum_j |g_j| r_{j,e}^{(\ell)} \tag{42}$$

provides a routing-level proxy for the amount of downstream gradient signal initially accessible to the new expert. For signed objectives, the signed score further accounts for cancellation between opposing update directions.

The selected donor therefore has a specific interpretation: *it is the pretrained expert–router pair whose routing responses overlap most strongly with downstream gradient signals at initialization*. The criterion neither assumes that the donor is globally optimal for the downstream task nor directly predicts final task performance. Its purpose is to provide a task-relevant initialization within a region of the pretrained routing geometry where downstream learning already supplies informative updates.

C.2 Support-Expanding Distillation (SED): Boundary Cases and Scope

Equation 12 handles two cases without an additional rule. If $e_{\ell,t}^\star \neq e_\ell^d$, the frozen reference is a pretrained expert other than the donor and exposes a complementary functional region. If $e_{\ell,t}^\star = e_\ell^d$, then $d_{\ell,t}^d \approx 0$, and no complementary function remains to be acquired; the auxiliary term instead stabilizes inherited donor behavior. The factor $(1-\alpha)$ makes exposure strongest before the new expert acquires the reference function and reduces it as the discrepancy closes. Conversely, α gates the routing-admission term, so routing pressure begins only as relative competence improves and stops at the local competitive scale.

SED addresses feasibility rather than downstream value: $e^{\star}_{\ell,t}$ provides a local reference from the frozen router, and α measures functional progress relative to the donor, not task reward. Task-specific specialization remains governed by $\mathcal{L}_{\text{task}}$. SED therefore expands inherited training support by distilling pretrained expert outputs as off-route supervision, while retaining the downstream objective as the criterion for useful computation.

C.3 Training Pseudocode

Algorithm 1 describes the ReconExp pipeline.

Algorithm 1 ReconExp with GRPO

Require: Pretrained MoE π_0 with N experts; group size G
Ensure: Expanded policy π

Let $n = N + 1$, Append $(E_{\ell,n}, w_{\ell,n})$; freeze pretrained parameters
for $k = 1, \ldots, K$ **do**
  $\tau_{g,i} \sim \pi_{\theta_k}, \quad A_{g,i} \leftarrow \mathrm{GRPO}(r_{g,1:G})$
  Construct $\mathcal{L}_{\mathrm{GRPO}}$ from sampled trajectories
  if $k = 1$ **then**
    $g_{g,i,t} \leftarrow -\dfrac{\partial \mathcal{L}_{\mathrm{GRPO}}}{\partial s_{g,i,t}}$ for valid response tokens
    $S_e^{(\ell)} \leftarrow \dfrac{\sum_{g,i,t} g_{g,i,t}\, r^{(\ell)}_{g,i,t,e}}{\sum_{g,i,t} |g_{g,i,t}| + \epsilon}$
    $e^d_\ell \leftarrow \arg\max_{e \le N} S_e^{(\ell)}$
    $(E_{\ell,n}, w_{\ell,n}) \leftarrow (E_{\ell,e^d_\ell}, w_{\ell,e^d_\ell})$
  end if
  for (ℓ, t) : valid-token mask $m_t = 1$ **do**
    $e^{\star}_{\ell,t} \leftarrow \arg\max_{e \le N} p^0_{\ell,t,e}$
    $z^{\star} \leftarrow \mathrm{sg}[E_{e^{\star}_{\ell,t}}(h_{\ell,t})], \quad z^d \leftarrow \mathrm{sg}[E_{e^d_\ell}(h_{\ell,t})], \quad z^n \leftarrow E_n(h_{\ell,t})$
    $d^n \leftarrow \dfrac{\|z^n - z^{\star}\|_2^2}{\|z^{\star}\|_2^2 + \epsilon}, \quad d^d \leftarrow \dfrac{\|z^d - z^{\star}\|_2^2}{\|z^{\star}\|_2^2 + \epsilon}$
    $\alpha \leftarrow \mathrm{sg}\left[\mathrm{clip}\left(\dfrac{d^d - d^n}{d^d + \epsilon}, 0, 1\right)\right]$
    $\Delta \leftarrow \mathrm{ReLU}\left(\dfrac{\bar{p}^{\mathrm{old}}_{\ell,t} - p^n_{\ell,t}}{\bar{p}^{\mathrm{old}}_{\ell,t} + \epsilon}\right)$
    $\mathcal{L}^{\mathrm{SED}}_{\ell,t} \leftarrow (1 - \alpha) d^n + \alpha \Delta^2$
  end for
  $\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{GRPO}} + \lambda_{\mathrm{SED}}\, \mathbb{E}_{\ell,t}[\mathcal{L}^{\mathrm{SED}}_{\ell,t}]$
  $\theta_{k+1} \leftarrow \mathrm{OptStep}(\theta_k, \nabla_\theta \mathcal{L})$
end for
return $\pi_{\theta_{K+1}}$

C.4 Results with RLOO

Table 6 evaluates whether the benefits of ReconExp persist under a different policy optimization algorithm. RLOO (Ahmadian et al., 2024) is a REINFORCE-style method that estimates the advantage of each sampled response using the rewards of the other responses to the same prompt as a leave-one-out baseline. With RLOO, ReconExp achieves the highest BFCL overall accuracy among the compared methods, while its Non-Live AST accuracy remains below Full-FT. On WebShop, ReconExp achieves a score of 0.4768 and a success rate of 10.4%, both higher than the compared baselines. These results suggest that the effectiveness of routing–learning reconciliation is not specific to the GRPO training setup.

Table 6: Further results with the RLOO algorithm. Higher values are better for all metrics.

Method	Berkeley Function Calling Leaderboard						WebShop	
	Overall acc	Non-Live AST	Non-Live Exec	Live Acc	Relevance	Irrelevance	Score	SR
Base	40.23%	38.46%	33.13%	42.04%	70.59%	60.60%	0.1547	3.8%
Full-FT	40.32%	60.73%	51.20%	29.69%	100.00%	8.82%	0.4616	9.4%
LEGO	40.33%	52.75%	47.93%	33.51%	82.35%	24.78%	0.4409	8.6%
Expert-LoRA	39.47%	53.40%	41.50%	34.18%	88.24%	23.34%	0.4397	8.2%
ReconExp	40.58%	56.62%	47.11%	33.16%	88.24%	17.84%	0.4768	10.4%

C.5 Training Stability

To assess robustness to RL stochasticity, we fix the donor choices for each method and repeat downstream training with three matched random seeds. As shown in Figure 6, ReconExp achieves a mean score $\pm$ std of 1.344 ± 0.103, compared with 1.036 ± 0.080 for LEGO. Moreover, the variation across seeds within each initialization strategy is substantially smaller than the gap between the two strategies. These results indicate that the performance gap is robust to RL stochasticity and is unlikely to arise from a favorable training run.

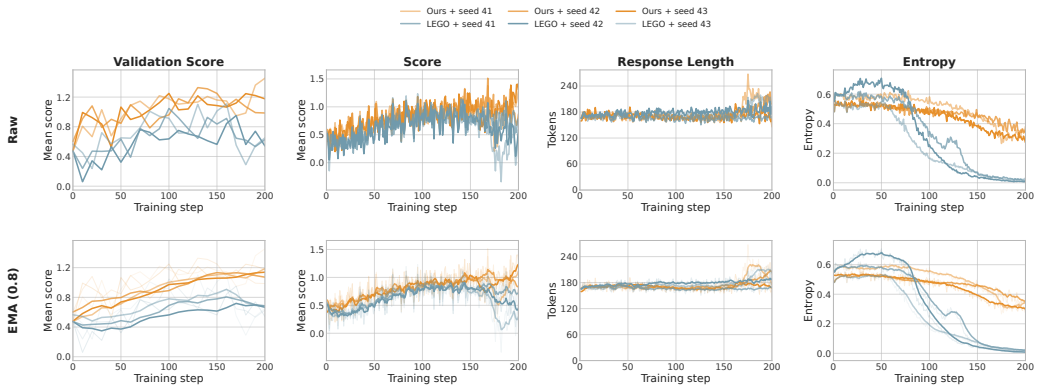


Figure 6: Multi-seed runs on ToolRL.

D Datasets and Environments

The experiments use two distinct data regimes. ToolRL-4K is a **static, single-turn** tool-calling dataset: the model samples a completion for a prompt, and a rule-based scorer compares it with the stored reference. WebShop is a **stateful, multi-turn** environment: the model samples an action, receives an observation from the local WebShop server, and continues until a purchase, termination, or budget limit. The two regimes share the GRPO outer loop but use different rollout and reward implementations.

ToolRL-4K. The preparation script uses the official ToolRL release (Qian et al., 2025). Each record contains an instruction, an input, and an expected output. A seeded NumPy shuffle (seed 31415) assigns 98% of the 4,000 records to training and 2% to validation. ToolRL rollouts use the configured single-turn stop strings and do not interact with an external environment.

WebShop. The official preparation script loads the 12,087 human goals from WebShop (Yao et al., 2022) and verifies session IDs before writing the split: sessions 0–499 form the test set (500), 500–1499 the development set (1,000), and 1500–12086 the training set (10,587). The native WebShop prompt constructor builds the prompt, and the session ID is passed to the environment server through `extra_info`. The action grammar supports two actions, `search[query]` and `click[button-or-product-id]`, enclosed in opening and closing `webshop` tags. Training limits prompts to 1,024 tokens, responses or complete trajectories to 20,480 tokens, actions to 1,024

Table 7: Evaluation protocols. Counts denote evaluation-set denominators, not training-set sizes.

Benchmark	Data and denominator	Decoding and scoring
ToolRL validation	79 post-filter examples	Greedy $n = 1$; scalar is the strict ToolRL format-plus-correctness reward; used for checkpoint selection
BFCL	2,251 question-function-answer pairs from BFCL-V2	Checkpoint-native OpenAI-compatible gateway; 61,440-token prompt cap, 4,096-token response cap, temperature 0.001; official harness aggregate and submetrics
MMLU	14,042 test questions	5-shot independent completion; 512-token cap; answer-letter accuracy
PIQA	1,838 validation questions	Zero-shot candidate likelihood; raw log-likelihood and character-normalized likelihood, with the latter primary; no chat template or truncation
ARC-Challenge	1,172 test questions	25-shot independent completion; 512-token cap; answer-letter accuracy
WinoGrande	1,267 validation questions	5-shot independent completion; 512-token cap; answer-letter accuracy
WebShop	500 official test sessions	Native `agent_loop`, greedy $n = 1$, temperature 0, 1,024 prompt, 20,480 trajectory, 1,536 observation, 1,024 action tokens, 15 executed actions; report score, strict SR

tokens, and observations to 1,536 tokens, with at most 15 environment actions. Observations are appended to the context but masked from the language-model loss by default.

Evaluation-only data. BFCL (Patil et al., 2025) is evaluated using its pinned external harness. MMLU (Hendrycks et al., 2021), PIQA (Bisk et al., 2020), ARC-Challenge (Clark et al., 2018), and WinoGrande (Sakaguchi et al., 2020) are also used only for evaluation. Their scoring protocols, summarized in Table 7, are separate from the GRPO reward managers.

Capability interpretation. ToolRL-4K (Qian et al., 2025) tests whether the model can follow a strict tool-call protocol, select the correct function, and produce correctly structured parameter values. Its partial score also measures graded recovery from incorrect names or parameters.

WebShop (Yao et al., 2022) tests grounded, long-horizon web navigation, including search formulation, product identification, option selection, and terminal purchase under changing observations.

BFCL (Patil et al., 2025) tests API-oriented function selection and argument construction across simple, parallel, multiple-function, live, irrelevance, and multi-turn cases.

MMLU (Hendrycks et al., 2021) measures broad multi-task academic knowledge and the retention of instruction-following ability.

PIQA (Bisk et al., 2020) tests physical commonsense and affordance reasoning through pairwise candidate-likelihood comparisons.

ARC-Challenge (Clark et al., 2018) tests science-question reasoning.

WinoGrande (Sakaguchi et al., 2020) tests commonsense coreference resolution in context.

Checkpoint selection. ToolRL checkpoints are selected using the converted 2% validation file. WebShop checkpoints are selected on the 1,000-session development split, with sessions 0–499 reserved for final testing.

E Prompts, Parsers, and Trajectory Examples

Prompt templates. ToolRL records retain the source instruction as the system message and the source input as the user message. The stored expected output is passed separately to the reward manager as ground truth and is not appended to the prompt. The relevant record schema is:

```
system:     <source instruction>
user:       <source input>
ground_truth: <source output, stored outside the prompt>
```

WebShop records use the local system prompt and instruction template. The instructional portion of the user message follows this exact structure:

Table 8: ToolRL and WebShop action grammars and parser boundaries.

Spec	Stop tokens	Accepted action and terminal rules
`toolrl`	closing `tool_call` and `response` tags	the `tool_call` wrapper must contain a JSON object (or line-delimited JSON objects in the reward parser); the `response` wrapper is a terminal answer branch
`webshop`	closing `webshop` tag by spec; the launcher also adds a closing `answer` tag	the `webshop` wrapper must match `search[...]` or `click[...]`; a terminal answer tag ends the agent turn without an environment action

```
You are shopping in WebShop.

Instruction:
{instruction}

Interact with the environment by emitting exactly one action at a
time:
<webshop>search[keywords]</webshop>
<webshop>click[visible button, option, or product id]</webshop>

Every response must contain exactly two blocks and no other text:
first a brief
reflection inside <think> and </think>, then exactly one complete
WebShop action.
```

The full template also prohibits invented observations and instructs the agent to finish by clicking "Buy Now". The three syntax demonstrations illustrate grammar and do not provide additional training targets. The agent loop inserts tool observations between turns and masks them from the loss by default.

The single-turn retention evaluator uses a separate set of prompts. MMLU, ARC-Challenge, and WinoGrande use a helpful/harmless system message, standard dataset-specific few-shot demonstrations, and a final instruction to place only the answer letter inside the answer tag. PIQA uses zero-shot scoring of the two continuations following `Question: {goal}\nAnswer:`.

Dataset token specifications and parsers. The implementation resolves the dataset token specification before constructing the agent loop. The two specifications used in the experiments are summarized below:

The parser removes think blocks before locating an action, requires the selected stop token to be present, and returns the last matching action block. During WebShop training, `forward_invalid_actions=True` forwards malformed generations to the tool server, preserving invalid-action metadata and format failures. EOS alone does not execute an action because `parse_action_on_eos=False`. Stop strings remain in the response so that the reward manager and evaluator can inspect the complete wrapper.

Trajectory record format. The native WebShop evaluator writes one JSON object per episode, recording the task index, session index, final score, strict-success flag, terminal-done flag, complete response, tool-interaction metadata, and trajectory stop reason. A compact successful trajectory has the following structure:

```
{
 "session_index": 42,
 "response": [
  "<think>Search for the requested item.</think>",
  "<webshop>search[white bluetooth speakers]</webshop>",
  "<think>The matching product is visible.</think>",
  "<webshop>click[B000000001]</webshop>",
  "<think>The selected product satisfies the constraints.</think>",
  "<webshop>click[Buy Now]</webshop>"
 ],
 "score": 1.0, "succ": true, "done": true,
 "stop_reason": "tool_signaled_done"
}
```

Invalid or overlong trajectories retain the same fields but may contain, for example, `score=0`, `succ=false`, `done=false`, and a stop reason such as `max_response_length_reached`. Invalid actions and observations remain in `tool_interact_info`. These fields distinguish positive partial WebShop scores, purchase completion, and strict success, which are separate metrics.

For ToolRL, the minimal reward examples are:

```
exact:    <think>...</think>
          <tool_call>
          {"name":"weather","parameters":{"city":"Paris"}}
          </tool_call>
          -> format=1, correctness=3, total=4

malformed: <think>...</think> <tool_call> not-json </tool_call>
          -> format=0; parse failure gives correctness=-3
```

Figure 7 and Figure 8 provide qualitative comparisons on WebShop and BFCL, respectively.

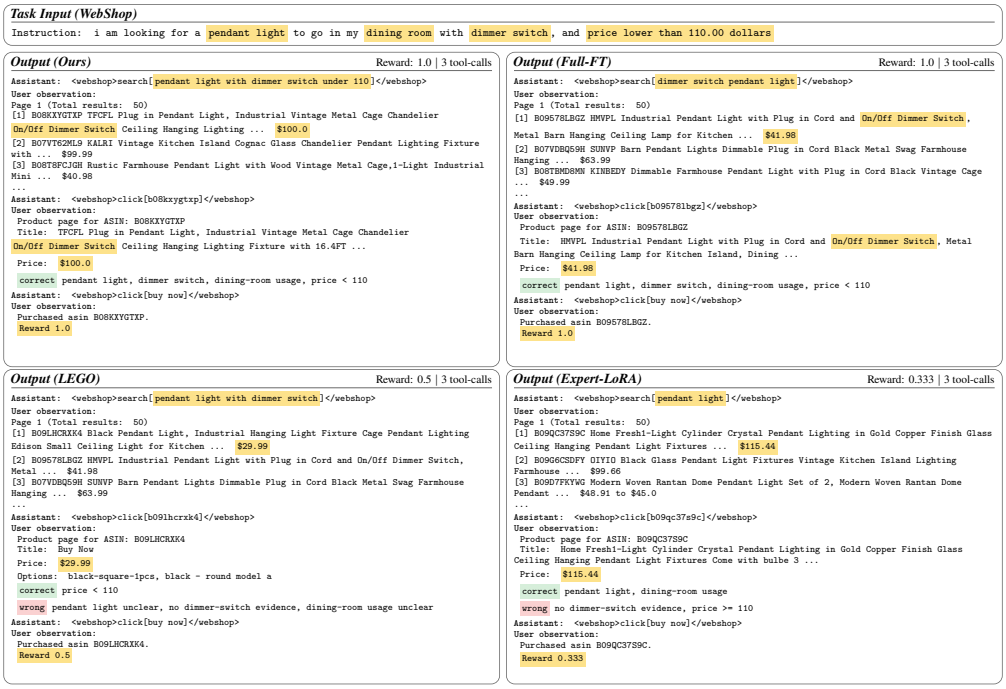


Figure 7: Qualitative comparison on WebShop.

F Reward Definition

In veRL, the GRPO reward managers initialize a token-level reward tensor to zero and assign one scalar to the final valid response position. Summing this tensor yields the sequence score. Intermediate tool observations therefore do not themselves produce dense training rewards in the current implementation.

F.1 ToolRL reward

Let F denote strict output-format validity and let C denote tool-call correctness. The format term is binary,

$$F(y, y^*) = \mathbf{1}\{y \text{ matches the reference branch and delimiters exactly}\}, \tag{43}$$

where the reference determines the branch: response-only, tool-call-only, both, or a think-only response. The implementation also requires exactly one pair of each expected outer tag. For tool calls, it requires line-delimited JSON objects with a string `name` and dictionary `parameters`.

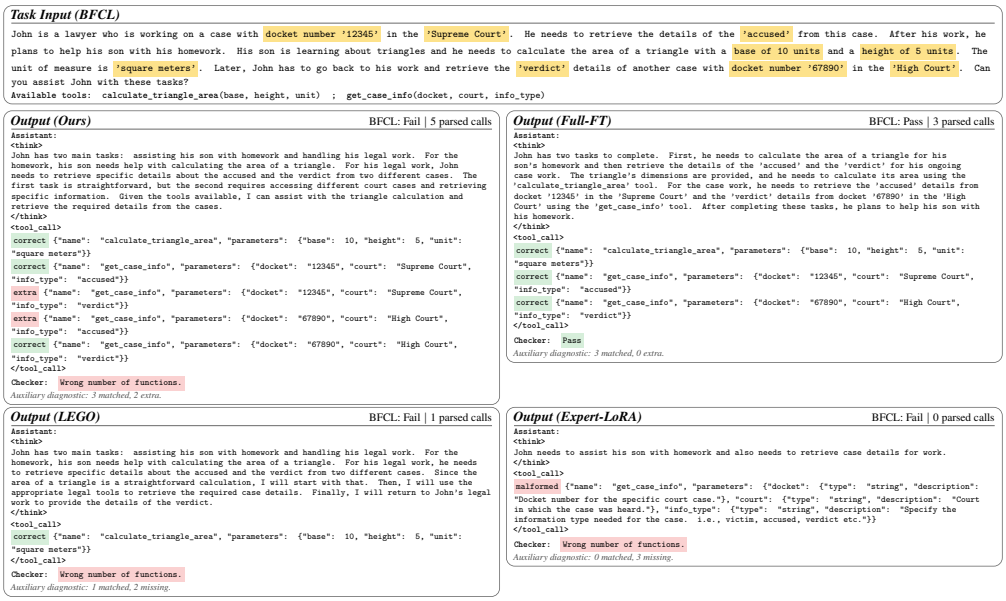


Figure 8: Qualitative comparison on BFCL.

If the reference contains no tool call, $C = 0$. If parsing either side fails, $C = -3$. Exact tool-call equality receives $C = 3$. Otherwise, let J_n be the multiset Jaccard score of expected and predicted tool names, J_p the sum of matched parameter-name Jaccards, V_p the number of exactly matched parameter values, and $M = 1 + \#\text{expected calls} + \#\text{expected parameters}$. The partial score is

$$C = 6\frac{J_n + J_p + V_p}{M} - 3, \qquad R_{\text{ToolRL}} = F + C. \tag{44}$$

The resulting scalar lies in $[-3, 4]$ and additively combines format validity and correctness. The reward manager also reports separate format, parse, name, parameter-name, parameter-value, raw-correctness, and format-failure diagnostics, which do not alter the scalar reward.

F.2 WebShop reward

For a WebShop trajectory, let S be the final non-null environment score and P indicate that the environment reached a terminal purchase state. The dense score serves as the reward signal. The strict-success indicator for an episode is $\mathbf{1}\{P \wedge S = 1\}$; the reported success rate is its mean over the evaluation episodes.

G Additional Training Curves

Figure 9 compares score, response length, and policy entropy during training. On ToolRL, Expert-LoRA exhibits a sharp score collapse accompanied by length explosion and near-zero entropy, while Full-FT and LEGO also show substantial entropy contraction. On WebShop, LEGO and Expert-LoRA suffer clear late-stage score degradation. In contrast, ReconExp maintains more stable task performance on both benchmarks without abnormal response-length growth. These dynamics are consistent with our routing–learning perspective: Learning-Guided Initialization provides a task-relevant starting region, while Support-Expanding Distillation broadens subsequent training support, together yielding more stable adaptation.

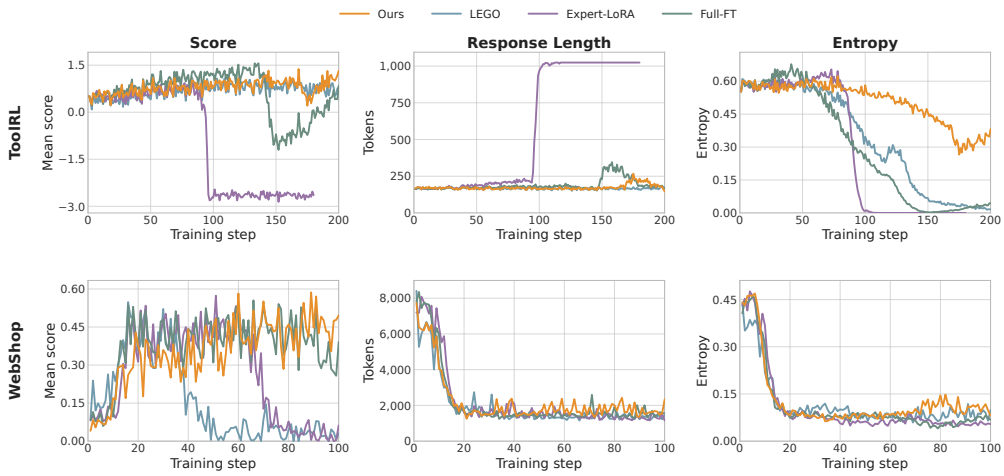


Figure 9: Additional training curves for main experiments.

H Implementation

H.1 Implementation Details

The implementation builds on veRL's FSDP actor and vLLM rollout stack. Expansion appends one expert and one router row at every routed layer. In ReconExp runs, pretrained experts and router rows, attention modules, embeddings, and output layers remain frozen. The new router row competes within the original top-k router; no pretrained expert is replaced, and the new expert is not force-routed.

ToolRL uses synchronous, non-agent rollouts with the ToolRL reward manager. WebShop uses asynchronous agent rollouts, a local environment server, a separate tool gateway, and the native `agent_loop`. The gateway retries tool requests once, records validity and terminal metadata, and truncates returned observations to the configured budget. WebShop forwards malformed actions when `forward_invalid_actions=True`, keeping format failures observable.

Actor and reference computations use FSDP on GPUs. The actor performs one PPO/GRPO epoch per rollout batch. Reward computation may run asynchronously, but the scalar reward is still assigned to the final valid response token before advantage estimation.

LGI uses the first online rollout. The frequency-based selector is a separate baseline that may use a fixed calibration Parquet file. SED detaches its input states and frozen donor/reference outputs, excludes prompt and padding tokens and masked observations, and retains gradients only through the new expert and the routing-admission term. R3 is an independent switch for route capture and replay and remains enabled for stability.

H.2 Baseline Implementation

Full-Finetune. Full-FT starts from the same pretrained checkpoint and updates the entire model without changing its architecture. All attention blocks, pretrained experts, routers, embeddings, and output layers are trainable. The downstream data, rollout protocol, and reward match those of the expansion methods, providing a reference for adaptation without a constraint on the number of trainable parameters.

LEGO expansion. We implement LEGO (Wu et al., 2026) as the structural-expansion baseline, appending one task-facing expert and its router row to each routed layer. Following LEGO, target-side routing pressure encourages downstream tokens to use the added capacity. LEGO also uses anchor-side stabilization to protect pretrained behavior. We adopt LEGO's own New-MoE-only setting, in which all pretrained experts and router rows remain frozen and only the appended expert and router row are optimized. We use the same downstream data and optimization budget as ReconExp.

MoE-Sieve-style Expert-LoRA. Following the routing-guided selection principle of MoE-Sieve (Manzoni, 2026), we profile the pretrained router on downstream samples and rank routed experts by activation frequency independently within each layer. We attach LoRA modules only to the selected high-frequency experts. The original MoE-Sieve implementation additionally trains attention, router, and shared-expert modules; our matched variant restricts trainable parameters to the selected expert adapters to isolate expert-side parameter-efficient adaptation.

H.3 HYPERPARAMETERS

Table 9 defines the reported hyperparameters, and Table 10 lists their values for the ToolRL-4K and WebShop RECONEXP runs. To alleviate early collapse, Full-FT uses a learning rate of 1e-6 with KL loss enabled and $\beta = 0.01$. Expert-LoRA uses a learning rate of 1e-4. Both configurations use GRPO and AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$ and weight decay 0.01.

Table 9: Definitions of the training, rollout, and RECONEXP hyperparameters.

Config	Explanation
`train_batch_size`	Number of prompt groups in one rollout batch
`ppo_mini_batch_size`	Configured actor-update mini-batch before rollout-group expansion and data-parallel sharding
`ppo_micro_batch_size_per_gpu`	Actor-update micro-batch size on each GPU
`log_prob_micro_batch_size_per_gpu`	Log-probability micro-batch size on each GPU
`ulysses_sequence_parallel_size`	Ulysses sequence-parallel degree
`tensor_model_parallel_size`	Tensor-parallel degree for rollout inference
`temperature`	Sampling temperature for training rollouts
`top_p`	Nucleus-sampling threshold for training rollouts
`n_samples_per_prompt`	Number of trajectories sampled per prompt group
`actor_learning_rate`	Learning rate for the new expert parameters
`ppo_epochs`	Actor optimization passes over each rollout batch
`max_epochs`	Maximum number of data epochs
`num_steps`	Explicit cap on optimizer steps
`use_kl_loss`	Whether an actor KL loss is applied
β	Coefficient of the actor KL loss
`max_prompt_length`	Maximum prompt length in tokens
`max_response_length`	Maximum response or complete agent-trajectory length in tokens
`multi_turn_max_turns`	Maximum WebShop environment-action budget
`loss_agg_mode`	Token-to-sequence aggregation used by the actor objective
`router_lr_multiplier`	Router-row learning rate relative to the expert learning rate
λ_{SED}	Coefficient of the complete SED auxiliary loss
`rollout_routing_replay`	Whether rollout-selected routes are replayed during actor forwards (R3)
`training_strategy`	Distributed parameter-sharding strategy

Table 10: Hyperparameters for RECONEXP on ToolRL-4K and WebShop.

Config	ToolRL-4K	WebShop
`train_batch_size`	128	16
`ppo_mini_batch_size`	128	16
`ppo_micro_batch_size_per_gpu`	8	1
`log_prob_micro_batch_size_per_gpu`	16	1
`ulysses_sequence_parallel_size`	1	
`tensor_model_parallel_size`	1	
`temperature`	0.4	1.0
`top_p`	1.0	
`n_samples_per_prompt`	4	8
`actor_learning_rate`	1×10^{-5}	
`ppo_epochs`	1	1
`max_epochs`	8	1
`num_steps`	200	100
`use_kl_loss`	False	
β	0	
`max_prompt_length`	2,048	1,024
`max_response_length`	1,024	20,480
`multi_turn_max_turns`	–	15
`loss_agg_mode`	token-mean	seq-mean-token-mean
`router_lr_multiplier`	0.25	
λ_{SED}	0.01	0.001
`rollout_routing_replay`	True	
`training_strategy`	FSDP	