Predictive Routing Replay for MoE-based LLM Reinforcement Learning

Daize Dong ♣ **Junlin Chen** ♣ **Haolong Jia** ♣ **Jiawei Wu** ♣ **Jiang Liu** ♦ **Jialian Wu** ♦ **Zhengzhong Liu** ★ **Zicheng Liu** ♦ **Emad Barsoum** ♦ **Dimitris N. Metaxas** ♣ **Hongyi Wang** ♣

Abstract

Mixture of Experts (MoE) Large Language Models (LLMs) achieve strong performance at scale. However, reinforcement learning (RL) on MoE-based LLMs often suffers from training instability. A root cause is *router drift*, i.e., expert activations can change drastically across model updates and differ between disaggregated rollout and training phases, causing large rollout–training mismatch and unstable importance sampling weights in PPO-style RL algorithms. Routing replay mitigates this issue by freezing the replay route within each reasoning trajectory, but it ignores how the router evolves under off-policy updates and thus causes *router staleness*. To address this limitation, we propose **Predictive Routing Replay (PR^2)**, which augments each router with a lightweight evolution predictor that learns to anticipate short-horizon router evolution. During the rollout phase, we use the predictive routing distribution to apply top-k routing, enabling gradients to reach experts that are likely to become active after updates. During the training phase, we replay the resulting predicted route to retain consistency for stable importance estimation. Theoretical analysis and experiments support that PR^2 reduces routing-induced mismatch, improves RL stability, and yields stronger performance across various reasoning benchmarks.

1. Introduction

Large language models (LLMs) have demonstrated strong reasoning capabilities when scaled through increased model parameters and training compute (OpenAI, 2024; DeepSeek-AI, 2026; Gemini Team, 2025). Mixture of Experts (MoE) models have recently emerged as a promising LLM architecture, enabling training compute to scale sublinearly with model size (Jiang et al., 2024; Zhu et al., 2024; Liu et al., 2024a; Yang et al., 2025; Agarwal et al., 2025; Kimi Team et al., 2025; GLM Team, 2026; DeepSeek-AI, 2026). This efficiency is achieved by activating a subset of expert networks within the large model, where the activated parameters typically account for less than 10% of the total model parameters (Liu et al., 2024a; Yang et al., 2025; GLM Team, 2026). This conditional computation paradigm has enabled models with trillions of parameters with manageable inference costs, and has become a core design choice in large-scale LLMs.

Beyond pretraining, reinforcement learning (RL) has emerged as a central mechanism for optimizing pre-trained LLMs to improve reasoning and agentic capabilities, as well as alignment with human preferences (Ouyang et al., 2022; Rafailov et al., 2023; Shao et al., 2024; Liu et al., 2025b; Gao et al., 2025b). Algorithms such as Proximal Policy Optimization (PPO) (Schulman et al., 2017) and its variants (Shao et al., 2024; Guo et al., 2025; Yu et al., 2025) are now widely used to improve reasoning, long-horizon decision making, and tool use tasks (Chen et al., 2022; Yao et al., 2023). However, applying RL to MoE-based LLMs introduces a unique and under-explored set of stability challenges that do not arise in dense models (Yao et al., 2025).

A key difficulty for conducting stable RL on MoE-based LLMs stems from the presence of learned routers that dynamically assign tokens to experts (Yao et al., 2025; Zheng et al., 2025a; Kim et al., 2025). When policy updates reuse trajectories generated by a stale snapshot, MoE models expose this off-policy gap as *router drift*, where the same token may be routed to different experts by the old snapshot and the current training policy. Such routing changes alter the computation path behind PPO-style importance ratios and can amplify their variance, thereby destabilizing policy optimization (Schulman et al., 2017; Shao et al., 2024; Yu et al., 2025).

To mitigate routing mismatch, routing replay records expert routes before training updates and reuses them during gradient evaluation; the old-snapshot variant records routes

♣ Rutgers University, ♦ AMD, ★ MBZUAI. Correspondence to: Daize Dong <daize.dong@rutgers.edu>, Hongyi Wang <hw689@cs.rutgers.edu>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

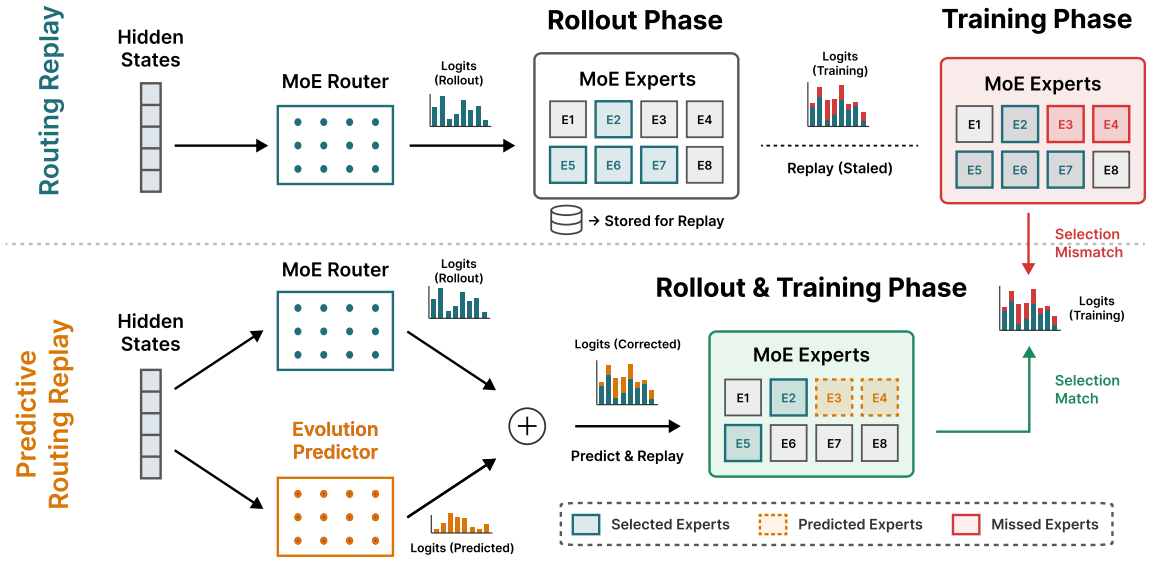


Figure 1. **Overview of Predictive Routing Replay (PR^2).** Routing replay stabilizes MoE RL by fixing routes, but cached routes become stale after a few off-policy steps. PR^2 adds an evolution predictor before top-k selection, caches the predicted expert indices during rollout, and replays them during training to preserve route consistency while tracking short-horizon router evolution.

from $\pi_{\theta_{\text{old}}}$ (Zheng et al., 2024). While effective at restoring routing consistency, routing replay introduces its own limitations. By fixing the cached routes, it prevents gradients from reaching experts that would become active after subsequent policy updates. Over time, this induces *router staleness*, as newly activated experts under the updated routing policy are excluded from gradient updates. Theoretically, this yields stale gradient estimation under replayed routes, where gradients are evaluated under a fixed replay route whose distribution deviates from the evolving current training policy.

In this work, we revisit routing replay from a principled perspective. We formalize routing replay as replacing the current routing distribution with a degenerate replay measure and derive a bound on router staleness, showing that replay staleness controls route-induced gradient deviation in fixed-route PPO gradients. This view reveals:

> *Stable importance estimation favors frozen routing, while effective learning requires routing distributions that track policy evolution.*

Motivated by this perspective, we propose **Predictive Routing Replay (PR^2)**, which augments routing replay with a route prediction scheme (Figure 1). PR^2 augments the router in each MoE layer with a lightweight *evolution predictor* that anticipates *router drift*. During the rollout phase, the predictor outputs a learned logit bias to adjust the routing distribution. We then perform top-k routing under the biased logits to obtain a predicted expert index, which is cached alongside trajectories for subsequent updates. During the training phase, we replay the cached top-k expert indices while disabling the predictor, keeping the replay route fixed to stabilize PPO-style importance ratios yet allowing gradient flow to experts that are likely to become active after policy updates. We train the evolution predictor with a KL divergence objective motivated by this bound on router staleness, along with an efficient training scheme and a dedicated learning-rate multiplier for fast adaptation.

PR^2 can be integrated into existing MoE RL frameworks, e.g., VeRL (Sheng et al., 2025), Slime (Zhu et al., 2025) and TRL (von Werra et al., 2020), without modifying the policy optimization objective. Empirically, PR^2 substantially reduces routing mismatch, stabilizes RL training, and improves performance over strong baselines (Guo et al., 2025; Yu et al., 2025; Zheng et al., 2025b) across reasoning tasks with negligible computation overhead. Notably, on AIME24 (Zhang & Math-AI, 2025), GRPO with PR^2 achieves 40.31% accuracy on Qwen3-30B-A3B-Base (Yang et al., 2025), improving over routing replay and GSPO by 12.29% points and 9.38% points, respectively.

Our main contributions are summarized below.

- We formalize *router staleness* as a key source of degradation in routing replay for MoE-based LLM reinforcement learning, and derive a divergence-based bound showing that replay staleness controls route-induced gradient deviation in fixed-route PPO gradients.
- We propose **Predictive Routing Replay (PR^2)**, which predicts short-horizon router evolution using evolution predictors trained with a KL objective motivated by the above bound, enabling rollout-time routing to anticipate future expert activation.
- We demonstrate PR^2 significantly reduces routing mis-

match, improves RL stability, and yields stronger performance across several reasoning tasks, establishing it as an effective alternative.

2. Related Work

Mixture of Experts. The Mixture of Experts (MoE) layer scales model capacity through conditional computation, where a learned router selects a subset of expert networks per token to reduce FLOPs while retaining a large number of parameters (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022; Zhu et al., 2024; Jiang et al., 2024; Yang et al., 2025; Kimi Team et al., 2025). A large body of work studies the routing strategy (Lewis et al., 2021; Huang et al., 2024), load balancing (Shazeer et al., 2017; Fedus et al., 2022; Liu et al., 2024a), and gradient estimation (Kool et al., 2021; Liu et al., 2023; 2024b) in MoE training. Analysis also highlights that routing decisions can be fragile (Dai et al., 2022; Zoph et al., 2022), which leads to unstable gradient flow and optimization during training (Kim et al., 2025). These findings motivate treating router evolution as an explicit modeling target when systems separate rollout from training (Yao et al., 2025).

Reinforcement Learning in LLMs. Reinforcement learning (RL) is widely used to align LLMs for improved reasoning and instruction following, with PPO-style algorithms and their variants serving as widely used optimization methods (Ouyang et al., 2022; Shao et al., 2024; Guo et al., 2025; Yu et al., 2025; Zhao et al., 2026). Large-scale LLM RL is often implemented in disaggregated pipelines, where rollouts and training are lagged, thereby introducing off-policy effects (Mnih et al., 2016; Espeholt et al., 2018). For MoE-based LLMs, this mismatch is exacerbated by *router drift*, where the same token can be assigned to different experts during rollout and training forward passes. This routing mismatch can amplify importance-ratio variance and destabilize policy optimization, harming RL performance (Yao et al., 2025).

Stabilizing RL in MoE. To overcome *router drift*, routing replay caches expert indices and reuses them during training to ensure routing consistency; the old-snapshot variant (Zheng et al., 2024) records routes from the old policy snapshot. Another variant, rollout routing replay (Ma et al., 2025), targets the rollout-engine setting. Beyond routing replay, GSPO (Zheng et al., 2025b) defines the importance ratio based on sequence likelihood and performs sequence-level clipping, avoiding token-level importance discrepancies induced by *router drift*. Other methods improve stability by modifying the clipping behavior (Gao et al., 2025a) or directly adjusting importance ratios to reduce sensitivity to off-policy errors (Zhang et al., 2025). In contrast, our work uses a staleness-control view of routing replay to predict short-horizon router evolution.

3. Preliminaries

We study off-policy reinforcement learning of an autoregressive MoE-based LLM. Following the standard splitting of rollout and training stages in PPO-style optimization, we let an old policy snapshot generate trajectories that are then reused to update the current training policy.

3.1. Off-Policy Reinforcement Learning

Let θ denote the current training parameters and θ_{old} the stale parameters associated with the rollout batch. Given a prompt x, the old policy snapshot $\pi_{\theta_{\text{old}}}$ samples a completion $y = (y_1, \ldots, y_T)$ one token at a time. At step t, the next token y_t is generated conditioned on the prefix $(x, y_{<t})$, and a reward $R(x, y)$ is assigned after generation. We use π_θ to denote the current training policy. This gives the off-policy objective

$$J(\theta) = \mathbb{E}_{x,\, y\sim\pi_{\theta_{\text{old}}}} \left[w_\theta(x, y)\, R(x, y) \right].$$

PPO-style methods (Ouyang et al., 2022; Shao et al., 2024; Guo et al., 2025) typically factorize importance weights w_θ into token-level importance ratios $r_t(\theta)$.

$$w_\theta(x, y) = \prod_{t=1}^{T} r_t(\theta), \quad r_t(\theta) = \frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\theta_{\text{old}}}(y_t \mid x, y_{<t})}. \quad (1)$$

We focus on PPO-style policy optimization, where training is driven by importance ratios in Eq. (1). We summarize the detailed objectives in Appendix A.

3.2. Reinforcement Learning on Mixture of Experts

We consider an MoE-based LLM with L MoE layers and N experts per layer. At token t and layer l, the router produces logits and routing probabilities over experts, and selects a top-k expert index $\mathcal{I}_t^{(l)} \subseteq \{1, \ldots, N\}$. We denote the layer-wise route at step t as

$$\mathcal{R}_t := \{\mathcal{I}_t^{(1)}, \ldots, \mathcal{I}_t^{(L)}\}.$$

Let $\rho^{\pi}_{\theta,t}(\mathcal{R}_t \mid x, y_{<t})$ denote the current route distribution induced by the routers of π_θ at token t (possibly degenerate under deterministic top-k selection). Conditioned on a route, the token distribution is $\pi_\theta(y_t \mid x, y_{<t}, \mathcal{R}_t)$, and the marginal token distribution satisfies

$$\pi_\theta(y_t \mid x, y_{<t}) = \mathbb{E}_{\mathcal{R}_t\sim\rho^{\pi}_{\theta,t}(\cdot\mid x, y_{<t})} \big[\pi_\theta(y_t \mid x, y_{<t}, \mathcal{R}_t)\big]. \quad (2)$$

In an autoregressive MoE model, computing $\pi_\theta(y_t \mid x, y_{<t}, \mathcal{R}_t)$ also depends on the key–value caches of all previous positions, which were produced under the routes

selected at those steps. Eq. (2) therefore conditions implicitly on the entire sequence of routes $\mathcal{R}_{\le t}$ used along the prefix, and we suppress this conditioning for notational brevity.

Router Drift. Router drift acts as a form of the stale policy gap specific to MoE. The same prefix $(x, y_{<t})$ may induce different routes under the old snapshot and current training policy. Let $\mathcal{R}^{\pi}_{\theta_{\text{old}},t}$ denote the route induced by $\pi_{\theta_{\text{old}}}$ and let $\mathcal{R}^{\pi}_{\theta,t}$ denote the route induced by π_θ, with corresponding routing distributions $\rho^{\pi}_{\theta_{\text{old}},t}(\cdot \mid x, y_{<t})$ and $\rho^{\pi}_{\theta,t}(\cdot \mid x, y_{<t})$, respectively. Router drift refers to the discrepancy $\mathcal{R}^{\pi}_{\theta_{\text{old}},t} \neq \mathcal{R}^{\pi}_{\theta,t}$, which yields inconsistent expert activation for tokens and can amplify the variance of importance ratios in PPO-style optimization (Zheng et al., 2025a; Yao et al., 2025).

Routing Replay. Routing replay fixes a cached replay route $\tilde{\mathcal{R}}_t$ during training to enforce routing consistency, denoted as:

$$\text{replay forward at step } t \;=\; \pi_\theta(y_t \mid x, y_{<t}, \tilde{\mathcal{R}}_t), \qquad (3)$$

where $\tilde{\mathcal{R}}_t$ is a cached replay route. Routing replay sets $\tilde{\mathcal{R}}_t = \mathcal{R}^{\pi}_{\theta_{\text{old}},t}$ by replaying the route induced by the old snapshot (Zheng et al., 2024). This method stabilizes importance estimation by fixing the route during gradient evaluation, but may prevent gradients from reaching experts that would become active under subsequent router updates.

3.3. Router Staleness

The stability induced by routing replay comes at the cost of staleness. As the training router evolves, a cached route may drift away from the route preferred by the current training policy. Let ρ^{rep}_t denote the route distribution induced by a replay scheme. For deterministic replay, ρ^{rep}_t is a point mass at the cached route. The TV distance is therefore binary on raw deterministic routes, and we use it here as a discrete-route summary. Appendix B.2 introduces a soft layer-wise categorical relaxation on which the same divergence is differentiable and is used by the PR^2 predictive loss. We define token-level router staleness as

$$\mathcal{S}_t(\rho^{\text{rep}}_t) = D_{\text{TV}}\big(\rho^{\pi}_{\theta,t}(\cdot \mid x, y_{<t}), \rho^{\text{rep}}_t(\cdot \mid x, y_{<t})\big)\,.$$

Let $g_t(P_t, \theta)$ denote the fixed-route PPO gradient kernel averaged under a route distribution P_t. Appendix B.1 shows that, for any bounded fixed-route gradient kernel, replay staleness controls the route-induced gradient deviation.

$$\big\|g_t(\rho^{\pi}_{\theta,t}, \theta) - g_t(\rho^{\text{rep}}_t, \theta)\big\| \le 2M_t\, \mathcal{S}_t(\rho^{\text{rep}}_t).$$

Thus, frozen replay is stable only when cached and current routes remain close.

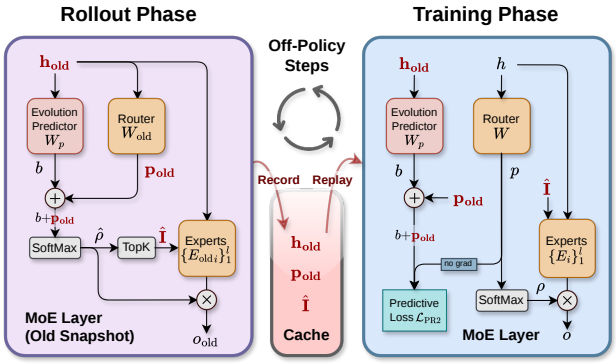


Figure 2. **Detailed PR^2 workflow.** *Left: rollout phase* $\pi_{\theta_{old}}$. PR^2 adds a learned logit bias before top-k selection and caches the predicted expert indices $\hat{\mathcal{I}}$ and route-recording features $h_{\text{old}}, p_{\text{old}}$. *Right: training phase* π_θ. Cached expert indices are replayed for route consistency. The evolution predictor is updated using cached features and the current routing distribution ρ via the predictive loss $\mathcal{L}_{\text{PR}^2}$.

4. PR^2: Predictive Routing Replay

The design principle of PR^2 is to keep deterministic replay consistency while predicting the expert index likely to become active after short-horizon policy updates. PR^2 keeps the fixed-route pattern of routing replay but replaces the cached route with a predicted one. For clarity, we describe PR^2 relative to routing replay. The same prediction mechanism can be inserted before any deterministic replay route. For each token, PR^2 constructs predicted expert indices from route-recording features of the old snapshot $\pi_{\theta_{\text{old}}}$ and trains a lightweight evolution predictor against routing distributions observed later on the same cached batch. Figure 2 summarizes our core idea.

4.1. Predictive Replay Route

We first define how PR^2 predicts the replay route and how that route is reused during training. The subscript "old" denotes route-recording quantities under the old policy snapshot $\pi_{\theta_{\text{old}}}$.

Route Prediction During Rollout. For each MoE layer l, let $h^{(l)}_{\text{old},t}$ be the router input at token t under $\pi_{\theta_{\text{old}}}$, and let

$$p^{(l)}_{\text{old},t} = h^{(l)}_{\text{old},t} W^{(l)}_{\text{old}}$$

be the corresponding router logits. Since the replay route must be fixed when data are recorded, only these features are available at rollout time. PR^2 therefore introduces a lightweight evolution predictor $W^{(l)}_p$, initialized at $\mathbf{0}$, which outputs an additive logit bias

$$b^{(l)}_t = h^{(l)}_{\text{old},t} W^{(l)}_p.$$

Algorithm 1 Predictive Routing Replay (PR^2)

Route Prediction During Rollout on old snapshot θ_{old}.
for each token t and MoE layer l **do**
  Compute $p^{(l)}_{\text{old},t} \leftarrow h^{(l)}_{\text{old},t} W^{(l)}_{\text{old}}$.
  Compute $b^{(l)}_t \leftarrow h^{(l)}_{\text{old},t} W^{(l)}_p$.
  Set $\hat{\rho}^{(l)}_t \leftarrow \text{Softmax}(p^{(l)}_{\text{old},t} + b^{(l)}_t)$.
  Select $\hat{\mathcal{I}}^{(l)}_t \leftarrow \text{TopK}(\hat{\rho}^{(l)}_t, k)$ and dispatch to experts.
  Cache $\hat{\mathcal{I}}^{(l)}_t$ and $(h^{(l)}_{\text{old},t}, p^{(l)}_{\text{old},t})$.
end for
Replay During Training on training model θ.
for each inner RL update **do**
  for each token t and layer l **do**
    Compute $\rho^{(l)}_t \leftarrow \text{Softmax}(p^{(l)}_t)$.
    Replay $\hat{\mathcal{I}}^{(l)}_t$ as the MoE expert index.
  end for
  Calculate $\mathcal{L}_{\text{Pr}^2}$ and update $\{W^{(l)}_p\}_{l=1}^{L}$.
  Calculate the base RL loss and update θ.
end for

The corrected logits define the predictive routing distribution

$$\hat{\rho}^{(l)}_t = \text{Softmax}\left(p^{(l)}_{\text{old},t} + b^{(l)}_t\right). \tag{4}$$

The predicted expert index $\hat{\mathcal{I}}^{(l)}_t$ is chosen by top-k selection, forming the predicted route $\hat{\mathcal{R}}_t$:

$$\begin{aligned} \hat{\mathcal{I}}^{(l)}_t &= \text{TopK}\left(\hat{\rho}^{(l)}_t, k\right), \\ \hat{\mathcal{R}}_t &= \left\{\hat{\mathcal{I}}^{(l)}_t\right\}_{l=1}^{L}. \end{aligned} \tag{5}$$

Using this predicted expert index, the MoE output during route recording is

$$o^{(l)}_{\text{old},t} = \sum_{j \in \hat{\mathcal{I}}^{(l)}_t} \hat{\rho}^{(l)}_{t,j} E^{(l)}_{\text{old},j}\left(h^{(l)}_{\text{old},t}\right). \tag{6}$$

The same predicted route is then cached for replay, i.e., $\tilde{\mathcal{R}}_t = \hat{\mathcal{R}}_t$ in Eq. (3). Thus, PR^2 keeps the fixed replay pattern of routing replay, but replaces the expert index from old snapshot with a predicted expert index. Weighting with $\hat{\rho}^{(l)}_t$ is intentional: the predictor is trained to match the current router distribution $\rho^{(l)}_t$, and $\hat{\rho}^{(l)}_t$ approximates the weights that π_θ would assign on the same index.

Replay During Training. For each MoE layer l, during training, the current policy π_θ computes router logits $p^{(l)}_t$ and the corresponding routing distribution

$$\rho^{(l)}_t = \text{Softmax}\left(p^{(l)}_t\right).$$

Similar to routing replay, PR^2 does not run fresh top-k selection. Instead, it reuses the cached expert index $\hat{\mathcal{I}}^{(l)}_t$ and restricts MoE computation to that set. The layer output is

$$o^{(l)}_t = \sum_{j \in \hat{\mathcal{I}}^{(l)}_t} \rho^{(l)}_{t,j} E^{(l)}_j\left(h^{(l)}_t\right). \tag{7}$$

This design freezes only the indices of selected experts during MoE computation, while expert outputs and policy gradients are always evaluated with the current parameters θ. The predictor is bypassed in the training pass and supervised separately by the predictive loss. An analogous PR^3 variant is described in Appendix C.

4.2. Evolution Predictor Training

The evolution predictor is trained on cached route-recording features to match the routing preference of the current training router on the same rollout batch.

Predictive Loss. Given a cached token t and layer l, we reconstruct the predictive routing distribution from the route-recording features using Eq. (4), and compare it with the current routing distribution $\rho^{(l)}_t$:

$$\mathcal{L}_{\text{Pr}^2} = \sum_{l=1}^{L} \mathbb{E}_t\left[D_{\text{KL}}\left(\left\langle \rho^{(l)}_t \right\rangle \middle\| \hat{\rho}^{(l)}_t\right)\right], \tag{8}$$

where $\langle\cdot\rangle$ denotes the stop-gradient operator. Gradients update only the predictor parameters $\{W^{(l)}_p\}_{l=1}^{L}$. The current router acts as a teacher but is not changed by this auxiliary loss. We use Eq. (8) as the default PR^2 predictive loss and examine a delta-matching alternative in Appendix G. The cached index for the current batch remains fixed during training, and the updated predictor takes effect at the next route-recording phase.

Predictive Loss as a Staleness Surrogate. The staleness view in Section 3.3 suggests choosing cached indices that remain close to routes preferred by the current router. The predictive loss bounds the route-induced gradient deviation:

$$\mathbb{E}_t\left[\left\| g_t(P^{\rho}_t, \theta) - g_t(P^{\hat{\rho}}_t, \theta) \right\|\right] \le M\sqrt{2\,\mathcal{L}_{\text{Pr}^2}},$$

where P^{ρ}_t and $P^{\hat{\rho}}_t$ are the current and predictive soft route distributions. Appendix B.2 further shows that the same KL objective controls the current-router mass regret of the predicted hard top-k indices.

Off-Policy Training and Learning-Rate Scaling. The PR^2 predictive loss is evaluated after the first inner update, when the current router has moved away from the route-recording snapshot and provides a nontrivial target. The evolution predictors use a dedicated learning-rate multiplier α, allowing them to track router evolution on the timescale of RL updates without changing the base PPO objective. We provide the per-token rollout and training pseudocode in Algorithm 1; the full version is deferred to Appendix D.

Policy	Method	AIME24 (Avg@32)	AIME25 (Avg@32)	AMC23 (Avg@16)	HMMT25 (Avg@16)	Average
Off-2	GRPO	31.04	24.68	74.06	9.16	34.74
	GSPO	30.42	23.54	77.18	11.88	35.76
	GRPO + R^2	35.73	25.73	77.50	11.25	37.55
	GRPO + PR^2	**47.71**	**32.81**	**87.50**	**19.17**	**46.80**
Off-4	GRPO	25.42	17.71	70.93	8.33	30.60
	GSPO	32.50	22.08	79.38	12.08	36.51
	GRPO + R^2	28.13	21.25	73.59	10.00	33.24
	GRPO + PR^2	**47.40**	**31.67**	**85.47**	**20.42**	**46.24**
Off-8	GRPO	25.00	15.93	72.81	7.50	30.31
	GSPO	30.93	22.18	72.03	7.50	33.16
	GRPO + R^2	28.02	21.77	76.88	8.30	33.74
	GRPO + PR^2	**40.31**	**28.54**	**83.13**	**15.63**	**41.90**

Table 1. **Downstream reasoning accuracy on Qwen3-30B-A3B-Base.** R^2 refers to routing replay for simplicity. Bold values mark the best accuracy in each metric column within an off-policy strength, and shaded rows mark PR^2.

5. Experiments

We organize our experiments around three empirical questions: whether PR^2 improves downstream reasoning under repeated rollout reuse, whether it stabilizes PPO-style optimization, and whether it keeps replay routes closer to the evolving router. We first describe the shared experimental setup, and then evaluate these questions through downstream accuracy, training dynamics, and routing analysis.

5.1. Implementation Details

Models and Training Data. We evaluate PR^2in three MoE-based LLM settings. Qwen3-30B-A3B-Base (Yang et al., 2025) is trained on DAPO-17K (Yu et al., 2025), Moonlight-16B-A3B (Liu et al., 2025a) is trained on GSM8K (Cobbe et al., 2021), and OLMoE-1B-7B (Muennighoff et al., 2024) is trained on the RLVR-GSM dataset[1] following its original implementation. All methods are implemented in the VeRL (Sheng et al., 2025) framework.

Baselines and Off-Policy Strength. We implement PR^2on top of GRPO (Shao et al., 2024), and compare it with GSPO (Zheng et al., 2025b) and routing replay (Zheng et al., 2024) under the same settings. Rollout routing replay (Ma et al., 2025) is compared with PR^3in Appendix F. We denote each off-policy setting as off-κ, where $\kappa = \frac{\mathcal{B}_{\text{global}}}{\mathcal{B}_{\text{update}}}$ is the ratio between the rollout batch size and the training update batch size. A larger κ means that each rollout batch is reused for more training updates, corresponding to stronger off-policy reuse.

[1] https://huggingface.co/datasets/allenai/RLVR-GSM

Evaluation Benchmarks. For the main Qwen3-30B-A3B-Base setting, we evaluate off-2, off-4, and off-8 performance on AIME24 and AIME25 (Zhang & Math-AI, 2025), AMC23 (Mathematical Association of America, 2026), and HMMT25 (Balunović et al., 2025). For Moonlight-16B-A3B and OLMoE-1B-7B, we report GSM8K and MATH500 (Hendrycks et al., 2021) results in the cross-model evaluation. Appendix E provides detailed implementation settings.

5.2. Downstream Reasoning Accuracy

Main Comparison. Table 1 reports the main comparison on Qwen3-30B-A3B-Base under off-2, off-4, and off-8. We compare $PR^2$with GRPO, GSPO, and routing replay. PR^2 achieves the best average accuracy in all three settings, reaching 46.80% under off-2, 46.24% under off-4, and 41.90% under off-8. Compared with routing replay, PR^2 improves average accuracy by 9.25% points, 13.00% points, and 8.16% points, respectively. Compared with GSPO, the corresponding gains are 11.04% points, 9.73% points, and 8.74% points. The gains remain substantial as rollout reuse becomes stronger, showing that PR^2 is not limited to mild off-policy settings. This trend is consistent with the router-staleness view. Routing replay preserves route consistency, but its cached indices cannot adapt to the routes preferred by the evolving router after repeated updates. In contrast, PR^2 moves the replay indices toward the router's short-horizon evolution while retaining deterministic replay during training.

Cross-Model Evaluation. Table 2 extends the comparison to Moonlight-16B-A3B and OLMoE-1B-7B under matched off-policy strengths. On Moonlight-16B-A3B, GRPO + PR^2 obtains the best average accuracy in all three settings, improving over routing replay by 1.19% points,

Policy	Method	Moonlight-16B-A3B GSM8K (Avg@1)	Moonlight-16B-A3B MATH500 (Pass@4)	Moonlight-16B-A3B Average	OLMoE-1B-7B GSM8K (Avg@1)	OLMoE-1B-7B MATH500 (Avg@4)	OLMoE-1B-7B Average
Off-2	GRPO	81.80	41.60	61.70	**73.76**	20.90	47.33
	GSPO	81.50	51.00	66.25	73.16	20.95	47.06
	GRPO + R^2	82.79	52.60	67.69	72.56	20.80	46.68
	GRPO + PR^2	**83.17**	**54.60**	**68.88**	73.47	**21.45**	**47.46**
Off-4	GRPO	62.09	36.40	49.25	72.48	20.15	46.32
	GSPO	62.85	37.80	50.33	**73.09**	22.05	47.57
	GRPO + R^2	73.69	49.20	61.45	72.71	21.85	47.28
	GRPO + PR^2	**74.07**	**51.20**	**62.64**	72.78	**23.00**	**47.89**
Off-8	GRPO	58.07	39.60	48.84	70.66	20.80	45.73
	GSPO	62.78	38.80	50.79	73.39	20.75	47.07
	GRPO + R^2	59.29	40.80	50.04	**73.92**	20.90	47.41
	GRPO + PR^2	**63.76**	**43.20**	**53.48**	72.71	**22.50**	**47.60**

Table 2. **Downstream reasoning accuracy on Moonlight-16B-A3B and OLMoE-1B-7B.** Columns are grouped by model. R^2 refers to routing replay for simplicity. Bold values mark the best accuracy in each metric column within an off-policy strength, and shaded rows mark PR^2.

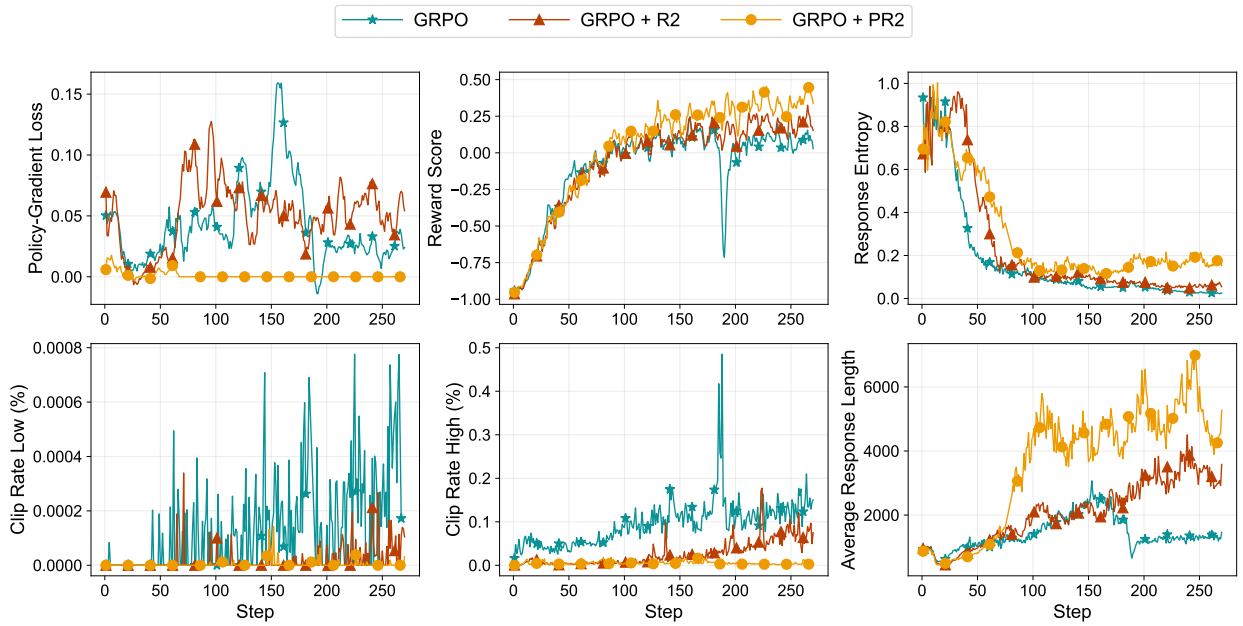
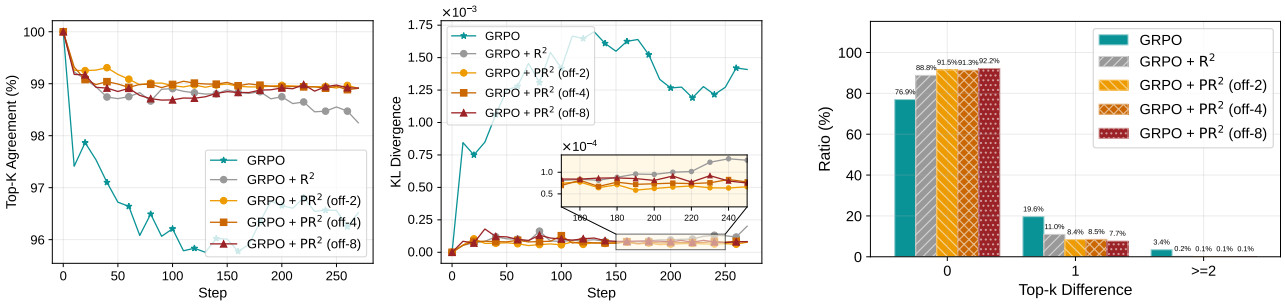


Figure 3. **Training dynamics on Qwen3-30B-A3B-Base under off-2.** We compare GRPO, routing replay, and PR^2 on policy-gradient loss, reward, response entropy, clipping rates, and response length. Smoother curves and lower clipping volatility indicate more stable updates.

1.19% points, and 3.44% points under off-2, off-4, and off-8, respectively. The largest gain appears under the strongest rollout reuse, again matching the router-staleness view that frozen replay routes drift farther from the current router as κ increases. On OLMoE-1B-7B, the absolute gains are smaller, but GRPO + PR^2 still achieves the best average accuracy across off-2, off-4, and off-8. These results suggest that predictive routing replay is not specific to a single MoE backbone or dataset.

5.3. Training Stability

Training Dynamics. Figure 3 compares off-2 training dynamics on Qwen3-30B-A3B-Base for GRPO, routing replay, and PR^2. We use these curves to examine whether repeated rollout reuse only affects final accuracy or also changes the optimization trajectory. Vanilla GRPO exhibits policy-gradient loss spikes and abrupt reward drops under rollout reuse. Routing replay reduces the most severe spikes, but still shows late-stage oscillations in clipping rates and entropy. In contrast, PR^2 keeps clipping rates substantially



(a) Top-k agreement and KL divergence curves.

(b) Routing deviation distribution.

Figure 4. **Routing prediction behavior during training. (a)** Top-k agreement between cached and current routing indices, together with route KL divergence curves. Higher agreement and lower KL indicate better staleness control by PR^2. **(b)** Routing deviation distribution at the final step. More mass at 0 difference and less mass at ≥ 1 differences indicate better route tracking. Baselines are from off-2 runs unless otherwise specified.

lower and yields smoother reward growth. The response-entropy and length curves show a similar pattern. PR^2 avoids abrupt entropy collapse while allowing response length to increase steadily, suggesting that the policy can improve reasoning behavior without entering unstable high-clipping updates. This supports the interpretation that predictive routing replay stabilizes the optimization process.

Optimization Behavior. The curves in Figure 3 illustrate how routing mismatch affects PPO-style optimization. In off-policy MoE RL, stale routing can produce extreme token-level importance ratios, which appear as elevated clipping rates and can further lead to loss spikes and reward regressions. Routing replay mitigates the most severe instability by enforcing replay consistency, but its cached route can still become stale as repeated updates move the router. PR^2 improves the training trajectory itself by replaying predicted routes that better match short-horizon router evolution.

5.4. Routing Prediction Analysis

The downstream comparisons in Tables 1 and 2 motivate the route-level analysis below.

Top-k Agreement and Route KL. Figure 4a reports top-k agreement and route KL on Qwen3-30B-A3B-Base. These two measurements are directly tied to the behavior of the evolution predictor. We compute agreement as $|\hat{\mathcal{I}}_t^{(l)} \cap \mathcal{I}_t^{(l)}| \,/\, k$ and average over tokens and layers. Vanilla GRPO shows a clear drop in top-k agreement and a rapid increase in route KL, indicating severe router drift during training. Routing replay reduces this drift but still becomes stale in the later stage. In contrast, PR^2 maintains higher agreement and lower KL across off-2, off-4, and off-8, suggesting that the predicted replay routes better track short-horizon router evolution under repeated rollout reuse.

Deviation Tails. Average agreement can hide rare but severe route flips. Figure 4b therefore reports the deviation count $k - |\hat{\mathcal{I}}_t^{(l)} \cap \mathcal{I}_t^{(l)}|$ on Qwen3-30B-A3B-Base. Compared with GRPO, PR^2 shifts substantially more mass to zero deviation. The zero-deviation ratio increases from 76.9% for GRPO and 88.8% for routing replay to 91.5%, 91.3%, and 92.2% under off-2, off-4, and off-8, respectively. Correspondingly, the 1-slot mismatch ratio decreases from 19.6% and 11.0% to 8.4%, 8.5%, and 7.7%, while the fraction of cases with ≥ 2 mismatched slots is reduced to only 0.1%. These results show that PR^2 not only improves average route tracking, but also suppresses large expert mismatches that can destabilize fixed-route updates.

6. Conclusion

This work identifies router drift as a source of instability in off-policy RL for MoE-based LLMs and introduces **Predictive Routing Replay (PR^2)**, which predicts short-horizon expert indices from cached route-recording features while preserving deterministic replay. Across theory and experiments, anticipating router evolution improves reasoning accuracy, PPO-style stability, and route tracking across MoE backbones and off-policy strengths.

LLM Usage

Large language models were used only for prose polishing, LaTeX cleanup, and formatting inspection while preparing the final manuscript. The authors wrote and revised the technical content, verified all numerical claims against experiment records, and remain responsible for the final paper.

Acknowledgments

The authors thank the AMD University Program, the open-source RL/MoE community, and the workshop reviewers for their support and constructive feedback.

References

Agarwal, S., Ahmad, L., Ai, J., Altman, S., Applebaum, A., Arbus, E., Arora, R. K., Bai, Y., Baker, B., Bao, H., et al. GPT-oss-120b and GPT-oss-20b model card. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2508.10925.

Balunović, M., Dekoninck, J., Petrov, I., Jovanović, N., and Vechev, M. MathArena: Evaluating LLMs on uncontaminated math competitions. MathArena, 2025. URL https://matharena.ai/.

Chen, W., Ma, X., Wang, X., and Cohen, W. W. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint (arXiv)*, 2022. URL https://arxiv.org/abs/2211.12588.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint (arXiv)*, 2021. URL https://arxiv.org/abs/2110.14168.

Dai, D., Dong, L., Ma, S., Zheng, B., Sui, Z., Chang, B., and Wei, F. StableMoE: Stable routing strategy for mixture of experts. *arXiv preprint (arXiv)*, 2022. URL https://arxiv.org/abs/2204.08396.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. Technical report, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash/blob/main/DeepSeek_V4.pdf.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *International Conference on Machine Learning (ICML)*, 2018. URL https://proceedings.mlr.press/v80/espeholt18a.html.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research (JMLR)*, 2022. URL https://www.jmlr.org/papers/v23/21-0998.html.

Gao, C., Zheng, C., Chen, X.-H., Dang, K., Liu, S., Yu, B., Yang, A., Bai, S., Zhou, J., and Lin, J. Soft adaptive policy optimization. *arXiv preprint (arXiv)*, 2025a. URL https://arxiv.org/abs/2511.20347.

Gao, J., Fu, W., Xie, M., Xu, S., He, C., Mei, Z., Zhu, B., and Wu, Y. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous RL. *arXiv preprint (arXiv)*, 2025b. URL https://arxiv.org/abs/2508.07976.

Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2507.06261.

GLM Team. GLM-5: From vibe coding to agentic engineering. *arXiv preprint (arXiv)*, 2026. URL https://arxiv.org/abs/2602.15763.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2501.12948.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint (arXiv)*, 2021. URL https://arxiv.org/abs/2103.03874.

Huang, Q., An, Z., Zhuang, N., Tao, M., Zhang, C., Jin, Y., Xu, K., Chen, L., Huang, S., and Feng, Y. Harder task needs more experts: Dynamic routing in MoE models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL https://aclanthology.org/2024.acl-long.696/.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., Casas, D. d. l., Hanna, E. B., Bressand, F., et al. Mixtral of experts. *arXiv preprint (arXiv)*, 2024. URL https://arxiv.org/abs/2401.04088.

Kim, J., Song, M., Shin, S., and Son, S. Defending MoE LLMs against harmful fine-tuning via safety routing alignment. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2509.22745.

Kimi Team, Bai, Y., Bao, Y., Chen, G., Chen, J., Chen, N., Chen, R., Chen, Y., Chen, Y., Chen, Y., et al. Kimi K2: Open agentic intelligence. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2507.20534.

Kool, W., Maddison, C. J., and Mnih, A. Unbiased gradient estimation with balanced assignments for mixtures of experts. In *Neural Information Processing Systems Workshop (NeurIPS Workshop)*, 2021. URL https://openreview.net/forum?id=Hvfva7l1tcj.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. GShard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint (arXiv)*, 2020. URL https://arxiv.org/abs/2006.16668.

Lewis, M., Bhosale, S., Dettmers, T., Goyal, N., and Zettlemoyer, L. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning (ICML)*, 2021. URL https://proceedings.mlr.press/v139/lewis21a.html.

Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al. DeepSeek-V3 technical report. *arXiv preprint (arXiv)*, 2024a. URL https://arxiv.org/abs/2412.19437.

Liu, J., Su, J., Yao, X., Jiang, Z., Lai, G., Du, Y., Qin, Y., Xu, W., Lu, E., Yan, J., Chen, Y., Zheng, H., Liu, Y., Liu, S., Yin, B., He, W., Zhu, H., Wang, Y., Wang, J., Dong, M., Zhang, Z., Kang, Y., Zhang, H., Xu, X., Zhang, Y., Wu, Y., Zhou, X., and Yang, Z. Muon is scalable for LLM training. *arXiv preprint (arXiv)*, 2025a. URL https://arxiv.org/abs/2502.16982.

Liu, L., Gao, J., and Chen, W. Sparse backpropagation for MoE training. *arXiv preprint (arXiv)*, 2023. URL https://arxiv.org/abs/2310.00811.

Liu, L., Kim, Y. J., Wang, S., Liang, C., Shen, Y., Cheng, H., Liu, X., Tanaka, M., Wu, X., Hu, W., Chaudhary, V., Lin, Z., Zhang, C., Xue, J., Awadalla, H., Gao, J., and Chen, W. GRIN: Gradient-informed MoE. *arXiv preprint (arXiv)*, 2024b. URL https://arxiv.org/abs/2409.12136.

Liu, Z., Tang, L., Jin, L., Li, H., Ranjan, N., Fan, D., Rohatgi, S., Fan, R., Pangarkar, O., Wang, H., et al. K2-V2: A 360-open, reasoning-enhanced LLM. *arXiv preprint (arXiv)*, 2025b. URL https://arxiv.org/abs/2512.06201.

Ma, W., Zhang, H., Zhao, L., Song, Y., Wang, Y., Sui, Z., and Luo, F. Stabilizing MoE reinforcement learning by aligning training and inference routers. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2510.11370.

Mathematical Association of America. AMC student programs. Mathematical Association of America (MAA), 2026. URL https://maa.org/student-programs/amc/.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2016. URL https://proceedings.mlr.press/v48/mniha16.html.

Muennighoff, N., Soldaini, L., Groeneveld, D., Lo, K., Morrison, J., Min, S., Shi, W., Walsh, P., Tafjord, O., Lambert, N., et al. OLMoE: Open mixture-of-experts language models. *arXiv preprint (arXiv)*, 2024. URL https://arxiv.org/abs/2409.02060.

OpenAI. GPT-4 technical report. *arXiv preprint (arXiv)*, 2024. URL https://arxiv.org/abs/2303.08774.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint (arXiv)*, 2017. URL https://arxiv.org/abs/1707.06347.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint (arXiv)*, 2024. URL https://arxiv.org/abs/2402.03300.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint (arXiv)*, 2017. URL https://arxiv.org/abs/1701.06538.

Sheng, G., Zhang, C., Ye, Z., Wu, X., Zhang, W., Zhang, R., Peng, Y., Lin, H., and Wu, C. HybridFlow: A flexible and efficient RLHF framework. In *European Conference on Computer Systems (EuroSys)*, 2025. URL https://arxiv.org/abs/2409.19256.

von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., Huang, S., Rasul, K., and Gallouédec, Q. TRL: Transformers Reinforcement Learning. GitHub repository, 2020. URL https://github.com/huggingface/trl.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2505.09388.

Yao, F., Liu, L., Zhang, D., Dong, C., Shang, J., and Gao, J. Your efficient RL framework secretly brings you off-policy RL training. Feng Yao's Notion, 2025. URL https://fengyao.notion.site/off-policy-rl.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, L., et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2503.14476.

Zhang, D., Wu, X., Huang, S., Hao, Y., Dong, L., Chi, Z., Sui, Z., and Wei, F. Towards stable and effective reinforcement learning for mixture-of-experts. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2510.23027.

Zhang, Y. and Math-AI, T. American invitational mathematics examination (AIME). Mathematical Association of America (MAA), 2025. URL https://maa.org/maa-invitational-competitions/.

Zhao, Y., Liu, Y., Liu, J., Chen, J., Wu, X., Hao, Y., Lv, T., Huang, S., Cui, L., Ye, Q., Wan, F., and Wei, F. Geometric-mean policy optimization. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=nCEs0tSwc2.

Zheng, C., Dang, K., Yu, B., Li, M., Jiang, H., Lin, J., Liu, Y., Lin, H., Wu, C., Hu, F., et al. Stabilizing reinforcement learning with LLMs: Formulation and practices. *arXiv preprint (arXiv)*, 2025a. URL https://arxiv.org/abs/2512.01374.

Zheng, C., Liu, S., Li, M., Chen, X.-H., Yu, B., Gao, C., Dang, K., Liu, Y., Men, R., Yang, A., et al. Group sequence policy optimization. *arXiv preprint (arXiv)*, 2025b. URL https://arxiv.org/abs/2507.18071.

Zheng, L., Yin, L., Xie, Z., Sun, C. L., Huang, J., Yu, C. H., Cao, S., Kozyrakis, C., Stoica, I., Gonzalez, J. E., et al. SGLang: Efficient execution of structured language model programs. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/724be4472168f31ba1c9ac630f15dec8-Abstract-Conference.html.

Zhu, T., Qu, X., Dong, D., Ruan, J., Tong, J., He, C., and Cheng, Y. LLaMA-MoE: Building mixture-of-experts from LLaMA with continual pre-training. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024. URL https://aclanthology.org/2024.emnlp-main.890/.

Zhu, Z., Xie, C., Lv, X., and slime Contributors. slime: An LLM post-training framework for RL scaling. GitHub repository, 2025. URL https://github.com/THUDM/slime.

Zoph, B., Bello, I., Kumar, S., Du, N., Huang, Y., Dean, J., Shazeer, N., and Fedus, W. ST-MoE: Designing stable and transferable sparse expert models. *arXiv preprint (arXiv)*, 2022. URL https://arxiv.org/abs/2202.08906.

A. RL Objectives and Notation

A.1. Notation and Setup

We consider an autoregressive policy π_θ generating completion $y = (y_1, \ldots, y_T)$ for prompt x. At token t, the policy conditions on the prefix $(x, y_{<t})$ and predicts the next token y_t. We use $\pi_{\theta_{\text{old}}}$ for the old policy snapshot that produced the rollout trajectories and π_θ for the current training policy. We use $\text{Clip}(u, 1-\epsilon, 1+\epsilon)$ for PPO clipping (Schulman et al., 2017; Yu et al., 2025).

A.2. Proximal Policy Optimization (PPO)

PPO optimizes a clipped surrogate objective with a token-level ratio

$$r_t(\theta) = \frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\theta_{\text{old}}}(y_t \mid x, y_{<t})}. \tag{9}$$

Given advantage A_t,

$$\mathcal{L}_{\text{PPO}}(\theta) = \mathbb{E}\left[\sum_{t=1}^{T} \min\Big(r_t(\theta)\, A_t,\ \text{Clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\, A_t\Big)\right]. \tag{10}$$

A KL regularizer to reference policy π_{ref} is often added.

$$\mathcal{L}_{\text{PPO+KL}}(\theta) = \mathcal{L}_{\text{PPO}}(\theta) - \beta\, \mathbb{E}\left[\sum_{t=1}^{T} D_{\text{KL}}\big(\pi_\theta(\cdot \mid x, y_{<t}) \,\big\|\, \pi_{\text{ref}}(\cdot \mid x, y_{<t})\big)\right]. \tag{11}$$

A.3. Group Relative Policy Optimization (GRPO)

GRPO uses group-relative advantages without value-function training (Shao et al., 2024; Guo et al., 2025). Given a group of G rollouts $\{y^{(g)}\}_{g=1}^{G}$ for prompt x, GRPO defines

$$A_t^{(g)} = \frac{R(x, y^{(g)}) - \frac{1}{G}\sum_{g'=1}^{G} R(x, y^{(g')})}{\text{Std}\big(\{R(x, y^{(g')})\}_{g'=1}^{G}\big) + \epsilon_A}. \tag{12}$$

The GRPO objective follows the PPO clipping form with these group-relative advantages.

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{g=1}^{G}\sum_{t=1}^{T} \min\Big(r_t^{(g)}(\theta)\, A_t^{(g)},\ \text{Clip}(r_t^{(g)}(\theta), 1-\epsilon, 1+\epsilon)\, A_t^{(g)}\Big)\right], \tag{13}$$

where $r_t^{(g)}(\theta) = \pi_\theta(y_t^{(g)} \mid x, y_{<t}^{(g)}) / \pi_{\theta_{\text{old}}}(y_t^{(g)} \mid x, y_{<t}^{(g)})$ is the token-level importance ratio for the g-th group completion, and the leading $1/G$ keeps the loss magnitude scale-invariant to group size.

In our implementation we follow the Clip-Higher variant of DAPO (Yu et al., 2025), which replaces the symmetric ϵ in the clip with an asymmetric pair $(\epsilon_{\text{low}}, \epsilon_{\text{high}})$:

$$\mathcal{L}_{\text{GRPO}}^{\text{DAPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{g=1}^{G}\sum_{t=1}^{T} \min\Big(r_t^{(g)}(\theta)\, A_t^{(g)},\ \text{Clip}(r_t^{(g)}(\theta), 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}})\, A_t^{(g)}\Big)\right]. \tag{14}$$

B. Theoretical Details

B.1. Routing Replay Gradient Deviation Bound

We restate the TV-based gradient-deviation bound used in Section 4. Let $\psi_t(\mathcal{R}_t, \theta)$ denote the token-level PPO gradient contribution evaluated under fixed route $\mathcal{R}_t$. For any route distribution P_t, define

$$g_t(P_t, \theta) := \mathbb{E}_{\mathcal{R}_t \sim P_t(\cdot \mid x, y_{<t})}[\psi_t(\mathcal{R}_t, \theta)]. \tag{15}$$

For any two route distributions P_t and Q_t, assume the fixed-route gradient kernel is locally bounded on the parameter region visited during training, with $M_t \geq \|\psi_t(\mathcal{R}_t, \theta)\|$ for all $\mathcal{R}_t$. The variational characterization of total variation gives

$$\|g_t(P_t, \theta) - g_t(Q_t, \theta)\| \leq 2M_t\, D_{\text{TV}}(P_t, Q_t). \tag{16}$$

Here and below, common conditioning on $(x, y_{<t})$ is suppressed inside divergences. Eq. (16) links router staleness and route-induced gradient deviation without depending on the specific PPO clipping form.

B.2. Soft PR^2 Surrogate and Top-k Bridge

We analyze the PR^2 predictive loss through a layer-wise relaxation. Instead of treating a layer route as a top-k set, the relaxed route draws one categorical expert variable $Z_t^{(l)} \in \{1, \dots, N\}$ from $\rho_t^{(l)}$ or $\hat{\rho}_t^{(l)}$. Let $Z_t = (Z_t^{(1)}, \dots, Z_t^{(L)})$.

$$P_t^{\rho}(Z_t) = \prod_{l=1}^{L} \rho_t^{(l)}(Z_t^{(l)}), \qquad P_t^{\hat{\rho}}(Z_t) = \prod_{l=1}^{L} \hat{\rho}_t^{(l)}(Z_t^{(l)}). \tag{17}$$

This relaxation is not the hard top-k replay distribution. It is the soft distributional object optimized by Eq. (8), where the current routing distribution is evaluated under the same replay-conditioned forward pass used for training. In this way, the bound characterizes staleness within the fixed-route training computation rather than the fully free-running MoE computation. The analysis uses the layer-summed definition in Eq. (8). If one reports a layer-averaged variant, the corresponding bound carries an additional factor L inside the square root.

Proposition B.1 (Predictive loss controls soft route-gradient deviation)**.** *Under the local boundedness assumption $\|\psi_t(Z_t, \theta)\| \le M$ for all tokens t and relaxed routes Z_t, and under the categorical-route relaxation in Eq. (17),*

$$\mathbb{E}_t\Big[\Big\|g_t(P_t^{\rho}, \theta) - g_t(P_t^{\hat{\rho}}, \theta)\Big\|\Big] \le M\sqrt{2\,\mathcal{L}_{\mathrm{Pr}^2}}. \tag{18}$$

Proof. Let P_t^{ρ} and $P_t^{\hat{\rho}}$ be the categorical-route distributions in Eq. (17), and write $\Delta g_t = g_t(P_t^{\rho}, \theta) - g_t(P_t^{\hat{\rho}}, \theta)$. Applying Eq. (16) gives $\|\Delta g_t\| \le 2M_t\, D_{\mathrm{TV}}(P_t^{\rho}, P_t^{\hat{\rho}})$, and Pinsker's inequality yields $D_{\mathrm{TV}}(P_t^{\rho}, P_t^{\hat{\rho}}) \le \sqrt{\frac{1}{2} D_{\mathrm{KL}}(P_t^{\rho} \| P_t^{\hat{\rho}})}$, so

$$\|\Delta g_t\| \le 2M_t \sqrt{\tfrac{1}{2}\, D_{\mathrm{KL}}\Big(P_t^{\rho} \| P_t^{\hat{\rho}}\Big)}. \tag{19}$$

Under the layer-wise categorical relaxation, $D_{\mathrm{KL}}(P_t^{\rho} \| P_t^{\hat{\rho}}) = \sum_{l=1}^{L} D_{\mathrm{KL}}(\rho_t^{(l)} \| \hat{\rho}_t^{(l)})$, hence

$$\|\Delta g_t\| \le 2M_t \sqrt{\tfrac{1}{2} \sum_{l=1}^{L} D_{\mathrm{KL}}\Big(\rho_t^{(l)} \| \hat{\rho}_t^{(l)}\Big)}. \tag{20}$$

Taking expectation over t, using $M_t \le M$, and applying Jensen's inequality to the $\sqrt{\cdot}$, we obtain $\mathbb{E}_t[\|\Delta g_t\|] \le M\sqrt{2 \sum_{l=1}^{L} \mathbb{E}_t[D_{\mathrm{KL}}(\rho_t^{(l)} \| \hat{\rho}_t^{(l)})]} = M\sqrt{2\,\mathcal{L}_{\mathrm{Pr}^2}}$, where the last equality uses Eq. (8). □

Remark on k-Slot Relaxations. The categorical relaxation above samples one expert variable per layer because it is the soft object directly matched by Eq. (8). A closer k-slot relaxation could sample $(Z_{t,1}^{(l)}, \dots, Z_{t,k}^{(l)})$ from $\rho_t^{(l)}$. The same TV and Pinsker argument would introduce the corresponding slot factor, while Lemma B.2 gives a deterministic support-level (activated expert indices) guarantee for the hard top-k expert support used by PR^2.

Lemma B.2 (Predictive KL controls top-k support regret)**.** *Consider a single layer and write $p = \rho_t^{(l)}$ and $\hat{p} = \hat{\rho}_t^{(l)}$. Let $S^\star = \mathrm{TopK}(p, k)$ and $\hat{S} = \mathrm{TopK}(\hat{p}, k)$, with $p(S) = \sum_{j \in S} p_j$. Then*

$$p(S^\star) - p(\hat{S}) \le \sqrt{2\, D_{\mathrm{KL}}(p \| \hat{p})}.$$

Consequently, if the current router has top-k margin $\Delta_k(p) = p_{(k)} - p_{(k+1)} > 0$ and $D_{\mathrm{KL}}(p \| \hat{p}) < \Delta_k(p)^2/2$, then $\mathrm{TopK}(\hat{p}, k) = \mathrm{TopK}(p, k)$.

Proof of Lemma B.2. Let $S^\star = \mathrm{TopK}(p, k)$ and $\hat{S} = \mathrm{TopK}(\hat{p}, k)$. Since $\hat{S}$ maximizes $\hat{p}(S)$ over all k-element supports, we have $\hat{p}(\hat{S}) \ge \hat{p}(S^\star)$. Decomposing $p(S^\star) - p(\hat{S})$ into three telescoping terms,

$$\begin{aligned} p(S^\star) - p(\hat{S}) &= \big(p(S^\star) - \hat{p}(S^\star)\big) + \big(\hat{p}(S^\star) - \hat{p}(\hat{S})\big) + \big(\hat{p}(\hat{S}) - p(\hat{S})\big) \\ &\le D_{\mathrm{TV}}(p, \hat{p}) + 0 + D_{\mathrm{TV}}(p, \hat{p}) \\ &\le \sqrt{2\, D_{\mathrm{KL}}(p \| \hat{p})}, \end{aligned}$$

where the second line uses $\hat{p}(S^\star) - \hat{p}(\hat{S}) \le 0$ together with the variational bound $|p(S) - \hat{p}(S)| \le D_{\mathrm{TV}}(p, \hat{p})$ for any subset S, and the third line uses Pinsker's inequality.

For the margin statement, the variational definition of total variation gives the elementwise bound

$$\|p - \hat{p}\|_\infty \le D_{\mathrm{TV}}(p, \hat{p}),$$

obtained by taking the singleton $A = \{j\}$ in $D_{\mathrm{TV}}(p, \hat{p}) = \max_A |p(A) - \hat{p}(A)|$, and Pinsker's inequality further yields

$$D_{\mathrm{TV}}(p, \hat{p}) \le \sqrt{\tfrac{1}{2} D_{\mathrm{KL}}(p \| \hat{p})}.$$

Hence whenever $\Delta_k(p) = p_{(k)} - p_{(k+1)} > 0$ and $D_{\mathrm{KL}}(p\|\hat{p}) < \Delta_k(p)^2/2$, we obtain $\|p - \hat{p}\|_\infty < \Delta_k(p)/2$. Every top-$k$ expert under p therefore remains above every non-top-k expert under p when scored by $\hat{p}$, so $\mathrm{TopK}(\hat{p}, k) = \mathrm{TopK}(p, k)$. □

C. Rollout-Engine Behavior Policy

The main text in Section 3 treats the trajectory source as a single old policy snapshot $\pi_{\theta_{\text{old}}}$. In disaggregated training pipelines, rollouts are actually produced by a separate rollout engine whose effective behavior policy $\mu_{\theta_{\text{old}}}$ can differ from the training-engine snapshot $\pi_{\theta_{\text{old}}}$ at the implementation level. Following (Zheng et al., 2025a), the token-level importance ratio admits the decomposition

$$r_t(\theta) = \underbrace{\frac{\pi_{\theta_{\text{old}}}(y_t \mid x, y_{<t})}{\mu_{\theta_{\text{old}}}(y_t \mid x, y_{<t})}}_{\text{system discrepancy}} \cdot \underbrace{\frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\theta_{\text{old}}}(y_t \mid x, y_{<t})}}_{\text{policy staleness}}, \tag{21}$$

which factorizes the rollout/training-engine gap from update-induced staleness. Each RL step then involves three MoE forward passes: a rollout pass under $\mu_{\theta_{\text{old}}}$ in the rollout engine that generates trajectories, a log-prob pass under $\pi_{\theta_{\text{old}}}$ in the training engine that evaluates the snapshot likelihoods used as the importance-ratio denominator, and a gradient pass under π_θ in the training engine that evaluates and backpropagates the surrogate loss. Each pass can pick different expert indices, and replay schemes resolve this by reusing cached indices across the latter two training-engine passes.

C.1. Rollout Routing Replay

Rollout routing replay (Ma et al., 2025) caches the rollout-engine route $\mathcal{I}^{(l)}_{\text{old},t}$ produced by $\mu_{\theta_{\text{old}}}$ at each token t and layer l during the rollout pass, and replays it in *both* training-engine passes. The log-prob pass under $\pi_{\theta_{\text{old}}}$ and the gradient pass under π_θ each substitute $\mathcal{I}^{(l)}_{\text{old},t}$ for their own top-k selection, so the two forward passes share the same expert indices. Reusing the indices is what makes the snapshot ratio $\pi_\theta(y_t \mid x, y_{<t})/\pi_{\theta_{\text{old}}}(y_t \mid x, y_{<t})$ in the importance weight well-defined despite the rollout-training engine gap, at the cost of freezing the indices to $\mathcal{I}^{(l)}_{\text{old},t}$ throughout.

C.2. PR^3: Predictive Rollout Routing Replay

PR^3 inserts the PR^2 prediction mechanism before rollout routing replay's frozen indices. The route-recording phase runs under $\mu_{\theta_{\text{old}}}$ in the rollout engine. We reuse the PR^2 conventions of Section 4, with $h^{(l)}_{\text{old},t}$, $p^{(l)}_{\text{old},t} = h^{(l)}_{\text{old},t} W^{(l)}_{\text{old}}$, $E^{(l)}_{\text{old},j}$ now denoting rollout-engine quantities. The predictive distribution and predicted top-k indices follow Eqs. (4)–(5),

$$\hat{\rho}^{(l)}_t = \mathrm{Softmax}\Big(p^{(l)}_{\text{old},t} + h^{(l)}_{\text{old},t} W^{(l)}_p\Big), \qquad \hat{\mathcal{I}}^{(l)}_t = \mathrm{TopK}\Big(\hat{\rho}^{(l)}_t,\, k\Big), \tag{22}$$

and the route-recording output reuses the same form as Eq. (6),

$$o^{(l)}_{\text{old},t} = \sum_{j \in \hat{\mathcal{I}}^{(l)}_t} \hat{\rho}^{(l)}_{t,j}\, E^{(l)}_{\text{old},j}\Big(h^{(l)}_{\text{old},t}\Big), \tag{23}$$

which drives autoregressive generation. We cache $\hat{\mathcal{I}}^{(l)}_t$ together with the route-recording features $(h^{(l)}_{\text{old},t},\, p^{(l)}_{\text{old},t})$. Both training-engine passes then replay $\hat{\mathcal{I}}^{(l)}_t$ as the MoE expert indices. The log-prob pass under $\pi_{\theta_{\text{old}}}$ uses the cached indices with the snapshot's own routing weights and expert parameters, and the gradient pass under π_θ follows Eq. (7) with the cached indices. The predictor is bypassed in both replay passes and is supervised by the KL objective in Eq. (8), evaluated on the cached rollout-engine features against the routing distribution observed under π_θ on the same batch. PR^3 thus stands to rollout routing replay exactly as PR^2 stands to routing replay. The predicted indices replace the frozen old-snapshot indices, and are replayed across training-engine forward passes. The PR^3 versus rollout routing replay comparison on Qwen3-30B-A3B-Base is reported in Table 4.

D. Predictive Routing Replay Pseudocode

Algorithms 2 and 3 give the per-token, per-layer route-recording and replay steps used in Section 4. The per-layer predictive losses are aggregated as in Eq. (8).

Algorithm 2 PR^2 route recording under the rollout policy $\pi_{\theta_{\text{old}}}$.

Require: Token position τ, layer index l, hidden feature h_{old}, router weights $W_{\text{old}}^{(l)}$, predictor $W_p^{(l)}$, experts $\{E_{\text{old},j}^{(l)}\}_{j=1}^{N}$, index cache $\mathcal{P}$, and feature cache $\mathcal{F}$.

Ensure: MoE layer output o_{old}.

1: Form old-snapshot logits and predictor bias $p_{\text{old}} \leftarrow h_{\text{old}} W_{\text{old}}^{(l)}$, $b \leftarrow h_{\text{old}} W_p^{(l)}$.
2: Construct the predictive distribution and predicted top-k indices $\hat{\rho} \leftarrow \mathrm{Softmax}(p_{\text{old}} + b)$, $\hat{\mathcal{I}} \leftarrow \mathrm{TopK}(\hat{\rho}, k)$.
3: Cache the predicted indices $\mathcal{P}[(\tau, l)] \leftarrow \hat{\mathcal{I}}$, and store route-recording features $\mathcal{F}[(\tau, l)] \leftarrow (h_{\text{old}}, p_{\text{old}})$ when (τ, l) is retained by the fixed-length feature cache.
4: Compute the layer output on the predicted indices $o_{\text{old}} \leftarrow \sum_{j \in \hat{\mathcal{I}}} \hat{\rho}_j \, E_{\text{old},j}^{(l)}(h_{\text{old}})$.

Algorithm 3 PR^2 replay and predictor update under the training policy π_θ.

Require: Mini-step index $i \in \{1, \ldots, \kappa\}$, token position τ, layer index l, hidden feature h, router weights $W^{(l)}$, predictor $W_p^{(l)}$, experts $\{E_j^{(l)}\}_{j=1}^{N}$, index cache $\mathcal{P}$, feature cache $\mathcal{F}$.

Ensure: MoE layer output o, per-layer predictive loss $\ell_{\text{Pr}^2}^{(l)}$.

1: Compute current routing weights and load the cached indices $p \leftarrow hW^{(l)}$, $\rho \leftarrow \mathrm{Softmax}(p)$, $\hat{\mathcal{I}} \leftarrow \mathcal{P}[(\tau, l)]$.
2: Compute the layer output with current experts on the cached indices $o \leftarrow \sum_{j \in \hat{\mathcal{I}}} \rho_j \, E_j^{(l)}(h)$.
3: **if** $i = 1 \vee (\tau, l) \notin \mathcal{F}$ **then**
4: Skip the predictive loss on the on-policy first mini-step or when no cached feature is available $\ell_{\text{Pr}^2}^{(l)} \leftarrow 0$.
5: **else**
6: Reconstruct the predictive distribution from cached features $(h_{\text{old}}, p_{\text{old}}) \leftarrow \mathcal{F}[(\tau, l)]$, $\hat{\rho} \leftarrow \mathrm{Softmax}(p_{\text{old}} + h_{\text{old}} W_p^{(l)})$.
7: Compute the per-layer predictive KL loss $\ell_{\text{Pr}^2}^{(l)} \leftarrow D_{\text{KL}}\Big(\langle \rho \rangle \,\Big\|\, \hat{\rho}\Big)$.
8: **end if**

E. Implementation Details

E.1. Model and Hyperparameters

All three runs share a common off-policy schedule: per model we fix a rollout batch size $\mathcal{B}_{\text{global}}$ and let the per-update batch size $\mathcal{B}_{\text{update}}$ take three values to realize off-2, off-4, and off-8. Unless noted otherwise, the asymmetric clip ratios are $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.28$, the SGLang oversampling ratio is 0.1, and predictors are zero-initialized. Following the trade-off between the training update mini-steps and learning rate described by (Ma et al., 2025), we use a smaller per-update learning rate under stronger off-policy reuse. Each rollout batch supports κ inner mini-step updates under off-κ, so a smaller per-update learning rate keeps the cumulative parameter drift over a rollout comparable across off-policy strengths.

Qwen3-30B-A3B-Base. Rollout uses $\mathcal{B}_{\text{global}} = 64$ with 8 responses per prompt and a maximum generation length of 16K tokens. We sweep $\mathcal{B}_{\text{update}} \in \{32, 16, 8\}$ for off-2, off-4, and off-8 with learning rates $\{2,\ 1.5,\ 1\} \times 10^{-6}$ and predictor learning-rate multipliers $\{10^4,\ 10^3,\ 10^3\}$. Each run uses 32 NVIDIA H200 GPUs for about 72 hours.

Moonlight-16B-A3B. Rollout uses $\mathcal{B}_{\text{global}} = 32$ with 16 responses per prompt and a maximum generation length of 1K tokens. We sweep $\mathcal{B}_{\text{update}} \in \{16, 8, 4\}$ for off-2, off-4, and off-8 with learning rates $\{5,\ 3.7,\ 2.5\} \times 10^{-7}$ and predictor learning-rate multipliers $\{10^2,\ 5 \times 10^1,\ 5 \times 10^1\}$. Each run uses 32 NVIDIA H200 GPUs for about 24 hours.

OLMoE-1B-7B. Rollout uses $\mathcal{B}_{\text{global}} = 32$ with 8 responses per prompt and a maximum generation length of 1K tokens. We sweep $\mathcal{B}_{\text{update}} \in \{16, 8, 4\}$ for off-2, off-4, and off-8 with learning rates $\{5,\ 3.7,\ 2.5\} \times 10^{-7}$ and a fixed predictor learning-rate multiplier 10^3. The clip ratios are $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.1$. Each run uses 8 NVIDIA RTX PRO 6000 GPUs for about 4 hours.

Baseline Settings. GRPO follows DAPO Clip-Higher without KL regularization. GSPO uses $\epsilon_{\text{low}} = 3 \times 10^{-4}$, $\epsilon_{\text{high}} = 4 \times 10^{-4}$, and KL coefficient 10^{-3}. Routing replay records routes from the old policy snapshot. Rollout routing replay shares PR^2's data, optimizer, off-policy schedule, and validations but replays routes produced by the rollout-engine behavior policy.

PR^2 Implementation. PR^2 adds an evolution predictor to each router without modifying the PPO objective. Runtime overhead is limited to a single router-side projection during route recording and the predictive-loss evaluation on a bounded feature cache during training. The full predicted top-k indices are always cached for replay. Route-recording features $(h_{\text{old}}, p_{\text{old}})$ used to train the predictor are cached only for a fixed number of tokens, with cache length $T_c = 2\text{K}$ for Qwen3-30B-A3B-Base (whose maximum generation length is $T = 16\text{K}$) and $T_c = T = 1\text{K}$ for Moonlight-16B-A3B and OLMoE-1B-7B. For Qwen3-30B-A3B-Base, the T_c retained positions are drawn by uniform sub-sampling along the rollout, so the predictor sees uniform coverage of the full 16K context rather than only the fixed positions of tokens.

E.2. Resource Details

Model	$(L, d, N, k, s, m, T, T_c)$	Index Cache	Feature Cache	FLOPs	FFN Ratio
Qwen3	$(48, 2048, 128, 8, 0, 768, 16\text{K}, 2\text{K})$	25.2 MB	453 MB	25.2M	0.69%
Moonlight	$(26, 2048, 64, 6, 2, 1408, 1\text{K}, 1\text{K})$	0.64 MB	116 MB	6.82M	0.19%
OLMoE	$(16, 2048, 64, 8, 0, 1024, 1\text{K}, 1\text{K})$	0.52 MB	71.3 MB	4.19M	0.26%

Table 3. PR^2 cache and route-recording forward-compute overhead. Index cache uses $4TLk$ bytes for int32 expert indices, feature cache uses $T_cL(2d + 4N)$ bytes for BF16 hidden features and FP32 router logits, and forward compute reports the extra $2LdN$ FLOPs per token from the predictor projection. Cache values are per response, and FLOPs values are per token.

Predictor Parameters and Compute. Each evolution predictor is a linear map $W_p^{(l)} \in \mathbb{R}^{d\times N}$. The total predictor size is therefore LdN parameters. During route recording, PR^2 adds one predictor projection per MoE layer, or $2LdN$ extra FLOPs per generated token. Table 3 reports the resulting model-specific values. Under the gated-MLP expert form with gate, up, and down projections, the Feed-Forward Network (FFN) ratio compares this route-recording forward overhead with the active expert computation, approximated as $6L(k + s)dm$ FLOPs per token. For Moonlight-16B-A3B, $s = 2$ counts shared experts, while the cached routed indices still use $k = 6$.

Cache Cost. We store replay expert indices as int32, so the index cache adds $4Lk$ bytes per generated token across all MoE layers and scales with the rollout length T. For predictor learning, PR^2 additionally caches route-recording features $(h_{\text{old}}, p_{\text{old}})$ for a fixed feature-cache length T_c, with BF16 hidden features and FP32 router logits, contributing $T_cL(2d + 4N)$ bytes per response. We use $T_c = 2\text{K}$ for Qwen3-30B-A3B-Base, whose maximum generation length is 16K. For Moonlight-16B-A3B and OLMoE-1B-7B the entire generation is already 1K tokens, so no downsampling is applied and $T_c = T = 1\text{K}$. This separation keeps full expert indices cheap while bounding the feature cache under long contexts.

F. Additional Results with PR^3

PR^3 Settings. Table 4 compares PR^3 with rollout routing replay on Qwen3-30B-A3B-Base. Both methods use the same data and hyperparameters. Unlike the main PR^2 experiments, this comparison uses learning-rate multipliers of $\{10^2, 10^1, 10^1\}$ for off-2, off-4, and off-8 runs, respectively.

Policy	Method	AIME24 (Avg@32)	AIME25 (Avg@32)	AMC23 (Avg@16)	HMMT25 (Avg@16)	Average
Off-2	GRPO + R^3	42.36	28.05	84.38	13.75	42.14
	GRPO + PR^3	**44.48**	**29.69**	**87.97**	**16.67**	**44.70**
Off-4	GRPO + R^3	41.22	28.92	**85.21**	15.56	42.73
	GRPO + PR^3	**47.19**	**32.81**	83.28	**20.83**	**46.03**
Off-8	GRPO + R^3	36.18	24.20	79.22	**11.39**	37.75
	GRPO + PR^3	**40.42**	**28.33**	**81.25**	10.00	**40.00**

Table 4. **Additional PR^3 results on Qwen3-30B-A3B-Base.** R^3 denotes rollout routing replay for simplicity. Bold values mark the better accuracy in each metric column within an off-policy strength, and shaded rows mark PR^3. Results are averaged over 3 different seeds.

PR^3 Results. Across all three off-policy settings, PR^3 improves the average competition-math accuracy over R^3 by 2.56%, 3.30%, and 2.25% points under off-2, off-4, and off-8, respectively. The gains are consistent on AIME24 and AIME25 in every setting, and on AMC23 and HMMT25 in most settings. The only exceptions are AMC23 under off-4 and HMMT25 under off-8, where R^3 is higher by 1.93% and 1.39% points. The largest average gain appears under off-4, suggesting that the learned predictive bias in PR^3 is especially useful when replayed rollouts become stale.

G. Additional Experimental Analysis

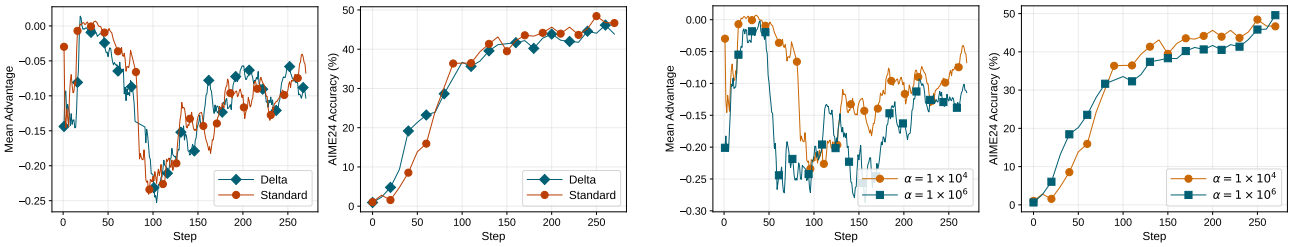
We provide additional ablations and diagnostics to examine whether the predictive component of PR^2 is robust to design choices and whether its route prediction remains reliable across update horizons. All ablations are conducted on Qwen3-30B-A3B-Base.

Predictive Objective Ablation. We compare the default PR^2 predictive loss $\mathcal{L}_{\mathrm{Pr}^2}$ with a delta-matching alternative $\mathcal{L}^{\Delta}_{\mathrm{Pr}^2}$. The delta-matching variant directly supervises the router-logit residual:

$$\mathcal{L}^{\Delta}_{\mathrm{Pr}^2} = \sum_{l=1}^{L} \mathbb{E}_t \left[D_{\mathrm{KL}} \left(\mathrm{Softmax}(\langle \Delta p_t^{(l)} \rangle) \,\middle\|\, \mathrm{Softmax}(b_t^{(l)}) \right) \right],$$

where $\Delta p_t^{(l)} := p_t^{(l)} - p_{\mathrm{old},t}^{(l)}$ and $b_t^{(l)} := h_{\mathrm{old},t}^{(l)} W_p^{(l)}$. As shown in Figure 5a, both objectives produce similar mean-advantage trajectories and comparable AIME24 improvement. The standard KL objective achieves slightly better final accuracy, so we use it in the main experiments.

Learning-Rate Multiplier Ablation. We further sweep the predictor learning-rate multiplier α under off-2. Figure 5b compares $\alpha = 1 \times 10^4$ and $\alpha = 1 \times 10^6$. Both settings yield stable optimization and steadily improve AIME24 accuracy, indicating that PR^2 does not rely on a narrowly tuned predictor learning rate. The larger multiplier learns faster in early training and reaches comparable final accuracy, while $\alpha = 10^4$ gives a slightly smoother mean-advantage trajectory. We therefore use the default multiplier setting in the main experiments.



(a) Predictive loss form. *(b)* Predictor learning-rate multiplier.

Figure 5. **Additional predictive-routing ablations on Qwen3-30B-A3B-Base under off-2.** The panels compare predictive loss forms and predictor learning-rate multipliers, reporting the batch-averaged clipped surrogate integrand and AIME24 accuracy over training steps.

Layer-Wise Top-k Accuracy. PR^2 is designed to predict short-horizon router evolution rather than perform long-range route forecasting. We therefore measure top-k agreement for each MoE layer and each within-batch mini-step. Figure 6 shows that agreement remains consistently high across off-2, off-4, and off-8, with most layers staying above 98%. The curves for different mini-steps largely overlap, suggesting that the prediction remains reliable over multiple reuse updates. The dominant variation comes from layer depth, where later layers show slightly lower agreement, but the values remain high enough for the predictor to provide useful replay supports.

Overall, Figures 5 and 6 show that PR^2 is robust to the predictive objective form and predictor learning-rate multiplier, while maintaining high route-prediction accuracy across layers and repeated rollout reuse.

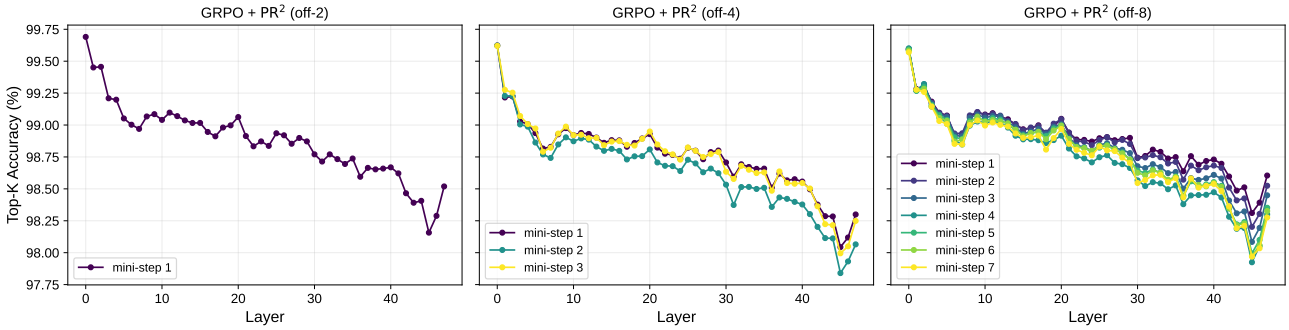


Figure 6. **Layer-wise top-k accuracy across mini-steps on Qwen3-30B-A3B-Base.** The horizontal axis indexes MoE layers, and each curve reports the top-k agreement at a different mini-step.