EASYPPO: STABILIZING THE CRITIC IS KEY

Anonymous authors
Paper under double-blind review

ABSTRACT

A key strength of Proximal Policy Optimization (PPO) is its learned critic, which uses historical trajectories collected during reinforcement learning to estimate expected returns and reduce policy-gradient variance. However, we find that the critic is also a major source of instability in reinforcement learning for large language models (LLMs). We identify two critic failure modes that destabilize PPO. First, filtering truncated rollouts from both actor and critic shifts the policy objective to reward conditioned on completion, allowing truncation to increase even as conditional reward improves. Second, heterogeneous return noise can cause high-variance prompts to dominate critic updates in finite batches. We introduce **EasyPPO** to address these failures. Actor-only overlong filtering trains the critic on returns from both completed and truncated rollouts. Noise-normalized critic regression weights each prompt's critic loss by the inverse standard deviation of its sampled returns, balancing noise contributions across prompts. Moderately smaller critic mini-batches confine outlier influence to fewer rollouts during gradient clipping. Across continuous-reward coding on FrontierCS, binary-reward mathematical reasoning on AIME24, and multi-turn search on Search-R1, EasyPPO remains stable throughout the full training horizon and consistently outperforms vanilla PPO, VAPO, and HL-Gauss PPO. Its best validation scores show relative gains of 14.89%, 2.28%, and 9.47% over PPO, respectively.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has become a key post-training paradigm for improving reasoning in Large Language Models (LLMs) (Guo et al., 2025; Jaech et al., 2024). Among RL algorithms, Proximal Policy Optimization (PPO) is particularly appealing for long-horizon reasoning because its actor–critic framework learns token-level return estimates from historical rollouts, enabling fine-grained credit assignment and lower-variance policy updates (Schulman et al., 2017). These benefits may become more important as post-training expands to tasks such as automated research (He et al., 2026; Lyu et al., 2026; Mang et al., 2025; Xu et al., 2026; Zhu et al., 2026), where rollouts grow longer and require more computation on evaluation.

The critic can also become a source of instability in PPO. Even with rollout batches of 512 responses in our experiments, we observe two failure modes in critic learning that heavily destabilize training.

Policy Objective Shift from Overlong Rollout Filtering. Truncated rollouts are a known source of reward noise in LLM reinforcement learning (Yu et al., 2025). Their rewards reflect incomplete responses, which can destabilize policy updates. Overlong filtering mitigates this noise by excluding truncated rollouts from policy updates (Yu et al., 2025). For example, during the 24K stage of Math RL, Nemotron-Cascade (Wang et al., 2025) excludes overlong rollouts to avoid noisy penalties on unfinished reasoning. In PPO, however, extending this filtering to critic updates can adversely change what the critic learns. For a prompt s, the critic $V_\phi(s)$, parameterized by ϕ, should predict the mean return $\mathbb{E}[R \mid s]$, where R is the sampled rollout return. Training it only on non-truncated rollouts instead makes it predict the mean return among completed responses. We show that, in the accurate-critic limit, filtering both networks shifts the policy objective to reward conditioned on completion, $\mathbb{E}[R \mid s, \text{not truncated}]$. This conditional reward can improve even as truncation becomes more frequent. We observe a growing fraction of unfinished rollouts when filtering both actor and critic updates. The critic should therefore retain these rollouts despite the noise in their returns.

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.

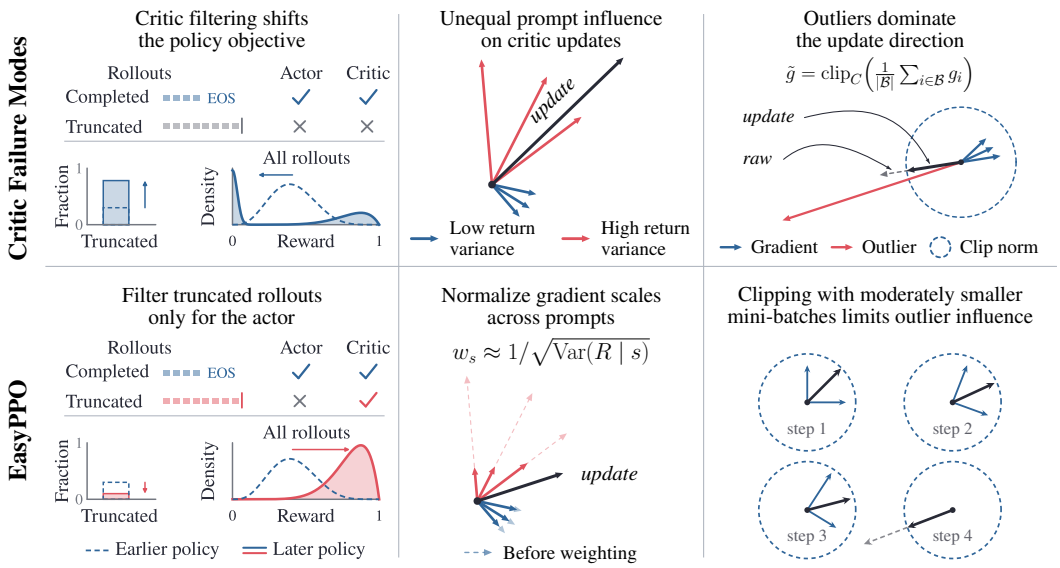

Figure 1: **EasyPPO addresses two sources of critic instability with three simple modifications. Left:** Filtering truncated rollouts from both actor and critic updates can increase truncation and lower overall reward; EasyPPO retains these rollouts for critic updates only. **Middle:** Return variability differs across prompts, causing unequal critic gradient scales; EasyPPO weights prompts inversely by their standard deviation. **Right:** Smaller critic mini-batches limit outlier influence through gradient clipping.

Heterogeneous Return Noise in Critic Updates. Retaining all rollouts for critic learning raises a second question: *how should the critic handle return noise that varies across prompts within the same batch?* These differences arise not only from truncation, but also because the actor policy produces consistent returns on some prompts and highly variable returns on others. They are especially pronounced in continuous-reward tasks, where some prompts permit only small gains while others admit widely varying levels of improvement. The critic aims to predict the mean return, minimizing $(V_\phi(s) - \mathbb{E}[R \mid s])^2$, but learns from sampled losses $(V_\phi(s) - R)^2$. Our analysis shows that return noise leaves the optimal prediction unchanged, yet contributes to the expected squared gradient magnitude alongside prediction error. As predictions improve, this noise can dominate, so prompts with more variable returns can disproportionately influence finite-batch updates.

Together, these two failures suggest that stable PPO should balance overlong filtering and critic learning while controlling how heterogeneous return noise shapes finite-batch updates. We introduce **EasyPPO**, which addresses these failures through three simple yet effective modifications to PPO. Figure 1 connects the two critic failure modes to the three changes in EasyPPO. First, we apply *actor-only overlong filtering*, retaining all rollouts for critic learning. Second, *noise-normalized critic regression* weights each prompt inversely by the empirical return standard deviation of its rollout group, to focus critic updates on prediction error normalized by return variability. Third, although larger critic mini-batches better average out return noise, we found that smaller ones can be preferable under gradient clipping because they confine the remaining outliers' influence to fewer rollouts; we therefore choose a moderate size to balance these effects.

Our experiments across continuous-reward coding on FrontierCS (He et al., 2026; Mang et al., 2025), binary-reward mathematical reasoning on AIME24, and multi-turn search on Search-R1 (Jin et al., 2025) show that EasyPPO remains stable throughout 200–300 training updates, substantially improving training stability over vanilla PPO and two recent PPO variants, VAPO (Yue et al., 2025) and HL-Gauss PPO (Zhou et al., 2026). This stability translates into consistently stronger task performance, without the late-stage collapse frequently observed in the baselines. EasyPPO's best validation scores show relative gains of 14.89%, 2.28%, and 9.47% over PPO, respectively; it ranks first and is the only compared method stable across all three tasks. Ablation studies further show that each component improves training stability and that combining all three yields the strongest gains. A stable critic is therefore key to reliable PPO training for LLMs.

2 PRELIMINARIES

We study PPO for reinforcement learning with verifiable rewards. Given a prompt x, the policy π_θ generates a response $y_{1:T}$ and receives a scalar reward $R = R(x, y_{1:T})$ at the end of the rollout. At token t, the state is the response prefix $s_t = (x, y_{<t})$ and the action is $a_t = y_t$. PPO learns a critic $V_\phi(s_t)$ to estimate the expected return from each token state and uses these value estimates to construct token-level advantages with GAE (Schulman et al., 2017). In our terminal-reward setting, the return from any prefix is the final rollout reward R when $\gamma = 1$, so $V^{\pi_\theta}(s_t) = \mathbb{E}[R \mid s_t]$.

For simplicity, we treat a response τ as one action from prompt state s, with final reward R and baseline $V_\phi(s)$. We omit PPO clipping and KL regularization, yielding the actor objective

$$\mathcal{L}_{\text{actor}}(\theta) = -\mathbb{E}_\tau[\log \pi_\theta(\tau \mid s)\, \text{sg}(R - V_\phi(s))]\,, \tag{1}$$

where $\mathbb{E}_\tau$ averages over sampled responses and $\text{sg}(\cdot)$ stops gradients through the advantage. The critic is trained against the final reward using the standard MSE objective

$$\mathcal{L}_V(\phi) = \frac{1}{2}\mathbb{E}_\tau\Big[\big(V_\phi(s) - R\big)^2\Big]\,. \tag{2}$$

Our implementation retains token-level GAE, the full clipped PPO objective, and KL regularization. We consider a fully synchronized setting in which each rollout batch is sampled from the latest policy snapshot.

3 OVERLONG FILTERING IN PPO

Overlong filtering offers a straightforward way to reduce reward noise in Group Relative Policy Optimization (GRPO) (Shao et al., 2024) by excluding truncated rollouts from policy updates (Wang et al., 2025; Yu et al., 2025). In PPO, however, filtering critic updates changes the learned value baseline and hence the advantages supplied to the actor. Figure 2 compares three filtering strategies on FrontierCS (Mang et al., 2025), using Qwen3.5-9B (Qwen Team, 2026) trained on 200 problems generated by FrontierSmith (He et al., 2026). All runs use a supervised fine-tuning (SFT) checkpoint, a 30-step critic warm-up, batches of 512 rollouts, and a 32,768-token response limit.

Without filtering, PPO exhibits repeated reward collapses. Filtering both actor and critic updates improves reward among non-truncated rollouts, but the truncation ratio approaches 1 and overall reward remains low. Among the three strategies, actor-only filtering is the most stable and achieves the highest overall score. Joint filtering also exhibits rising truncation and deteriorating overall reward in the AIME setting (Appendix C.5).

Why Joint Filtering Shifts the Policy Objective. For a fixed prompt s, let C denote the event that the rollout is not truncated. Excluding truncated rollouts restricts the critic loss in Equation 2 to completed responses, changing its pointwise population target from $\mathbb{E}[R \mid s]$ to $\mathbb{E}[R \mid s, C]$.

We analyze the idealized limit $V_\phi(s) = \mathbb{E}[R \mid s, C]$. Let $P_\theta(C \mid s) = \sum_{\tau \in C} \pi_\theta(\tau \mid s) > 0$ be the completion probability. Differentiating the conditional expected reward via the quotient rule gives

$$\begin{aligned}\nabla_\theta \mathbb{E}[R \mid s, C] &= \nabla_\theta \frac{\sum_{\tau \in C} \pi_\theta(\tau \mid s) R}{P_\theta(C \mid s)} \\ &= \frac{\sum_{\tau \in C} \nabla_\theta \pi_\theta(\tau \mid s) R}{P_\theta(C \mid s)} - \underbrace{\frac{\sum_{\tau \in C} \pi_\theta(\tau \mid s) R}{P_\theta(C \mid s)}}_{V_\phi(s) = \mathbb{E}[R \mid s, C]} \frac{\nabla_\theta P_\theta(C \mid s)}{P_\theta(C \mid s)}.\end{aligned} \tag{3}$$

Substituting $\nabla_\theta P_\theta(C \mid s) = \sum_{\tau \in C} \nabla_\theta \pi_\theta(\tau \mid s)$ and applying the log-derivative identity gives

$$\begin{aligned}\nabla_\theta \mathbb{E}[R \mid s, C] &= \frac{\sum_{\tau \in C} \nabla_\theta \pi_\theta(\tau \mid s)\big(R - V_\phi(s)\big)}{P_\theta(C \mid s)} \\ &= \sum_{\tau \in C} \frac{\pi_\theta(\tau \mid s)}{P_\theta(C \mid s)}\big(R - V_\phi(s)\big)\nabla_\theta \log \pi_\theta(\tau \mid s) = \underbrace{\mathbb{E}[(R - V_\phi(s))\nabla_\theta \log \pi_\theta(\tau \mid s) \mid s, C]}_{\textbf{Expected policy gradient of actor–critic filtering}}.\end{aligned} \tag{4}$$

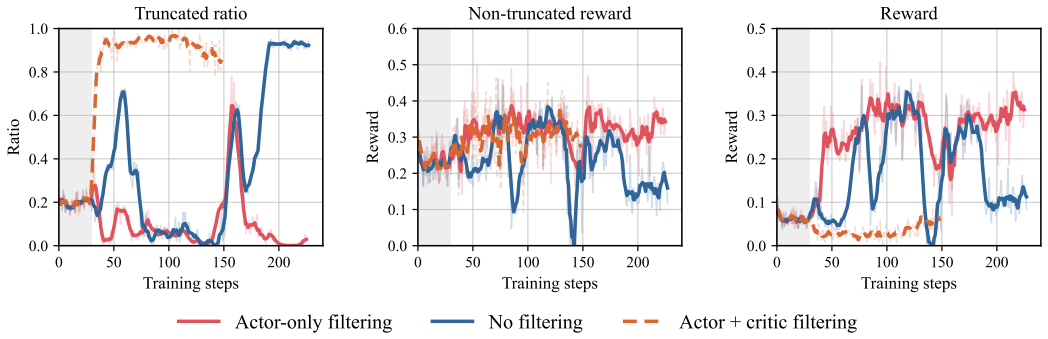

Figure 2: **Actor-only overlong filtering improves stability but does not eliminate instability.** Rollout statistics during PPO training of Qwen3.5-9B in the FrontierCS setting, using 200 problems generated by FrontierSmith, rollout batches of 512, a 30-step critic warm-up (gray), and a 32,768-token response limit. **Left:** fraction of truncated rollouts. **Middle:** mean reward among non-truncated rollouts. **Right:** mean reward over all rollouts. *Actor + critic filtering* improves reward among completed rollouts while the truncation ratio approaches 1, leaving overall reward low. *No filtering* exhibits repeated reward collapses. *Actor-only filtering* achieves higher reward and lower truncation overall, but the spike and reward drop near step 160 reveal residual instability.

Equation 4 shows that joint filtering optimizes reward conditioned on completion, without directly encouraging the policy to avoid truncation. This matches Figure 2: reward among completed rollouts rises while truncation becomes more frequent and overall reward remains low. At the token level, filtering similarly changes each critic target from $\mathbb{E}[R \mid s_t]$ to $\mathbb{E}[R \mid s_t, C]$, but the response-level policy-gradient identity need not hold (Appendix B.2).

Actor-Only Overlong Filtering. We therefore apply overlong filtering only to actor updates and retain all rollouts for critic training. As in GRPO, excluding truncated rollouts still biases the actor update. Retaining all rollouts preserves the critic target $\mathbb{E}[R \mid s]$, restoring a completion-probability term in the idealized filtered policy gradient. When completed rollouts have higher expected returns than truncated ones, this term encourages completion. However, the remaining truncation spike and reward drop in Figure 2 show that retaining all rollouts alone is not sufficient for stability. We next examine how heterogeneous return noise affects critic updates. We defer a full analysis of actor-only filtering to Appendix B.2.

4 Stabilizing the Critic

Following Section 3, we retain all rollouts for critic training. However, the actor's sampled returns can vary for the same prompt, and retaining truncated rollouts can increase this *return noise*. The noise level varies across prompts in the same batch, reflecting differences in truncation rates and in return variability among completed rollouts. This heterogeneity can be pronounced in continuous-score tasks with different reward scales. For example, in GPU kernel optimization, GEMM solutions may achieve only around $1\times$ speedup over an optimized baseline (Xing et al., 2026), whereas specialized operators can exceed $100\times$ over their respective library baselines (Yang et al., 2026).

Noise-Normalized Critic Regression. At a prefix s, consider a linear value head $V_\phi(s) = W^\top h_\phi(s)$ with critic features $h_\phi(s)$. For a sampled return R, the MSE loss $\mathcal{L}_V(\phi) = \frac{1}{2}(V_\phi(s) - R)^2$ has gradient $\nabla_W \mathcal{L}_V(\phi) = (V_\phi(s) - R)h_\phi(s)$. Since $R - \mathbb{E}[R \mid s]$ has zero conditional mean, the gradient second moment decomposes as

$$\mathbb{E}\Big[\|\nabla_W \mathcal{L}_V(\phi)\|_2^2 \,\Big|\, s\Big] = \|h_\phi(s)\|_2^2 \left[\underbrace{\big(V_\phi(s) - \mathbb{E}[R \mid s]\big)^2}_{\textbf{Prediction error}} + \underbrace{\mathrm{Var}(R \mid s)}_{\textbf{Return noise}}\right]. \tag{5}$$

As prediction error decreases, return noise can dominate this second moment, giving prompts with greater return variability disproportionate influence in finite batches.

This decomposition suggests dividing the critic loss at each prefix s by its own conditional return standard deviation. With $w(s) = 1/\sqrt{\mathrm{Var}(R \mid s)}$ for positive conditional variance, Equation 5 gives

$$\mathbb{E}\Big[\|w(s)\nabla_W \mathcal{L}_V(\phi)\|_2^2 \,\Big|\, s\Big] = \|h_\phi(s)\|_2^2 \left[\underbrace{\left(\frac{V_\phi(s) - \mathbb{E}[R \mid s]}{\sqrt{\mathrm{Var}(R \mid s)}}\right)^2}_{\textbf{Normalized prediction error}} + \underbrace{\frac{\mathrm{Var}(R \mid s)}{\mathrm{Var}(R \mid s)}}_{\textbf{Constant noise term}}\right]. \tag{6}$$

Here, positive state weighting preserves the pointwise population optimum $\mathbb{E}[R \mid s]$. Apart from the feature norm, the gradient second moment depends only on normalized prediction error. Fixing the noise term at one removes differences in return-noise scale as a source of imbalance across prompts.

In practice, estimating return variance separately at each prefix is costly, so we use the prompt's return standard deviation across its token states. With bounded critic-feature norms, this yields a common upper bound on the noise contribution to the gradient second moment, averaged over prefixes. We estimate $\hat{\sigma}(s)$ from rollout groups including truncated responses, and normalize weights to mean one over the prompt batch $\mathcal{B}$:

$$\begin{aligned} w(s) &= \frac{|\mathcal{B}|/\max\{\hat{\sigma}(s), \varepsilon\}}{\sum_{s' \in \mathcal{B}} 1/\max\{\hat{\sigma}(s'), \varepsilon\}}, \qquad \varepsilon > 0, \\ \mathcal{L}_V^{\text{weighted}}(\phi) &= \frac{1}{2\sum_i T_i} \sum\nolimits_i w(s_i) \sum\nolimits_{t=1}^{T_i} \big(V_\phi(s_{i,t}) - R_i\big)^2. \end{aligned} \tag{7}$$

Here, response i has prompt s_i, token states $s_{i,t}$, and T_i valid tokens; the loss follows widely used token-level averaging (Sheng et al., 2025; Yue et al., 2025). For discrete rewards with range Δ and group size n, we propose the floor $\varepsilon = \Delta/(2\sqrt{n})$, where groups with identical returns receive about twice the weight of groups containing one maximum reward and $n - 1$ minimum rewards.

For simplicity, we use the same rollout group to estimate weights and train the critic, which can introduce bias but works well in practice (Section 5). When fewer rollouts per prompt are desired, alternatives such as offline profiling with online updates could provide variance estimates; we leave these alternatives to future work. Appendix B.3 details these approximations and the variance floor, including the continuous-reward setting.

To test whether prompts with noisier returns contribute larger critic gradients, we study Qwen3.5-9B trained on FrontierSmith with actor-only filtering and noise-normalized critic regression. We use the initialization and rollout settings of Figure 2. At training steps 30, 60, and 90, we analyze 512 responses per checkpoint (16 prompts, 32 responses each). Offline, we compute critic loss gradients for predicting each response's sampled return, sum them over its tokens, and divide by the batch's total token count. Figure 3 compares these gradient norms before clipping, with and without noise normalization, grouped by the empirical prompt return standard deviation.

Without normalization, prompts with more variable returns tend to contribute larger gradients, and this pattern is stronger at later checkpoints. This is consistent with Equation 5: as the critic learns to predict the mean return, prediction error decreases, and return noise plays a larger role in the gradient second moment. Noise normalization largely removes this dependence on return variability, making gradient contributions more balanced across prompts. We also observe a similar overall trend in the AIME setting. Note that some gradient outliers still remain, so we next adjust the critic mini-batch size to limit their impact through clipping. Appendix C.4 provides the AIME results, scatter plots for both tasks at each checkpoint, and details of the gradient calculation.

Critic Mini-Batch Updates. The group estimates above can understate a prompt's return variability and give it excessive weight, leaving gradient outliers. Even with exact weights, stochastic return noise remains. Let B and m denote the numbers of rollouts in a critic batch and each mini-batch, respectively. For each of the $\frac{B}{m}$ mini-batches, we compute the gradient g of the mini-batch estimate of Equation 7 and clip its norm to at most $c > 0$ via $g \leftarrow g \cdot \min\{1, c/\|g\|_2\}$. We take one optimizer step with this clipped gradient, then recompute the next mini-batch gradient at the updated critic parameters.

Smaller mini-batches confine the joint rescaling caused by outliers to fewer rollouts, but average out less return noise. At fixed B, an idealized analysis gives an $O(m)$ bound on a single outlier's

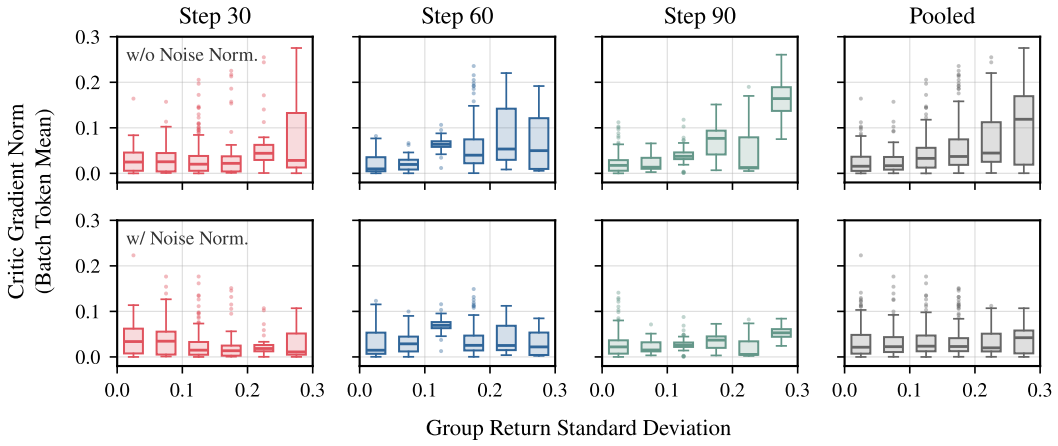

Figure 3: **Noise normalization balances critic gradient contributions across prompts.** Offline Qwen3.5-9B PPO diagnostics in the FrontierCS setting, using the data and sampling settings of Figure 2. **Columns:** steps 30, 60, 90, and their pooled responses. **Top:** without noise normalization. **Bottom:** with noise normalization. Boxes summarize per-response gradient norms before clipping within each return-standard-deviation interval. Consistent with Equation 5, prompts with more variable returns contribute larger gradients; noise normalization markedly reduces this dependence.

influence on the average clipped gradient, versus $O(1/m)$ gradient variance per mini-batch. We defer the assumptions and proof to Appendix B.4. In practice, we use $\frac{B}{m} = 4$ critic mini-batches per rollout batch. When changing the mini-batch size, the critic learning rate should be adjusted accordingly (Li et al., 2026b).

5 Experiments

Experimental Setup. For continuous-score coding, we train on a separate set of 200 problems generated by FrontierSmith (He et al., 2026) and validate on the algorithmic track of FrontierCS (Mang et al., 2025). Their diversity lets us study cross-prompt return heterogeneity, complementing TailRL's focus on high-reward exploration (Ramasubramanian et al., 2026). To test stability on single-turn and multi-turn binary-reward tasks, we also train on DAPO-Math-17K (Yu et al., 2025) with AIME24 validation, and on the Search-R1 mixture with its seven validation datasets (Jin et al., 2025).

We implement all methods in verl (Sheng et al., 2025), using its vanilla PPO (Schulman et al., 2017) as the baseline. We also compare with PPO + actor-only filtering, HL-Gauss PPO (Zhou et al., 2026) for its alternative critic objective, and VAPO (Yue et al., 2025) for its value-learning and advantage-estimation improvements. We omit VAPO's auxiliary positive-example language-modeling loss to focus on PPO updates. EasyPPO uses actor-only overlong filtering; PPO, HL-Gauss PPO, and VAPO use no filtering, following their original recipes.

We use Qwen3.5-9B-Base for AIME and Search-R1, and Qwen3.5-9B for FrontierCS (Qwen Team, 2026). For FrontierCS, we initialize from SFT on 347 nonzero-score trajectories generated by DeepSeek-V3.1 on the same 200 training problems. All methods share a 30-step critic warmup, rollout batches of 512 responses on FrontierCS and AIME and 1024 on Search-R1, and group sizes of 32 on FrontierCS and 16 on AIME and Search-R1. Training is strictly on-policy, with actor updates starting after critic warmup.

Training metrics are task score on FrontierSmith, reward on DAPO-Math-17K, and accuracy on Search-R1. Validation averages over five responses per FrontierCS problem and 32 per AIME24 problem; Search-R1 averages greedy accuracies equally across seven datasets and their own metrics. Training retains the original DAPO-Math-17K rewards in $\{-1, +1\}$. For plotting only, we map these rewards to $(r + 1)/2$ and divide FrontierCS scores by 100. Figure 4 shows one run per method and

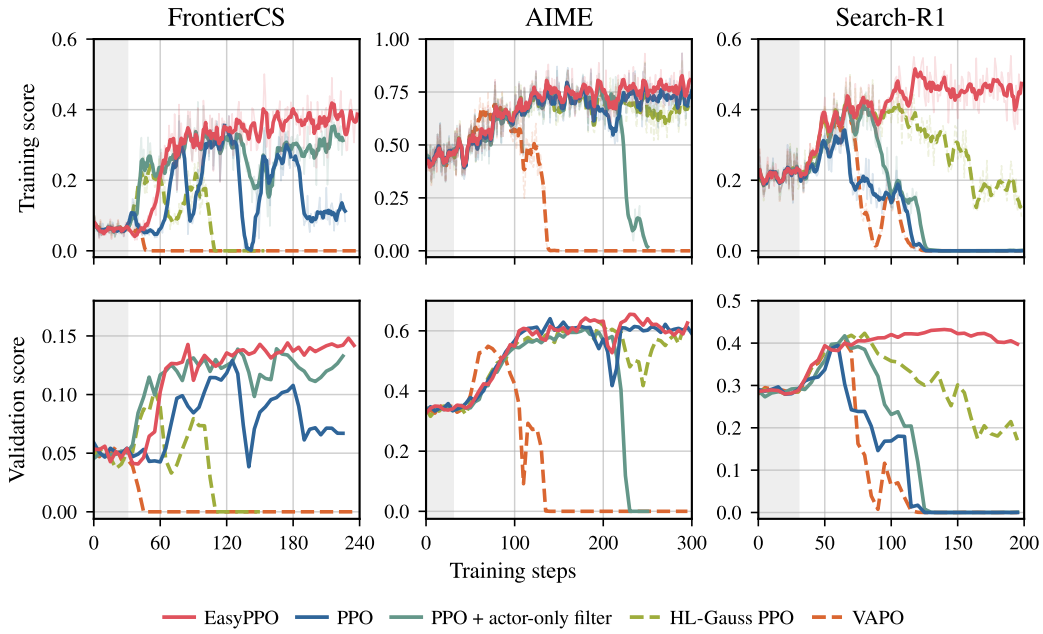

Figure 4: **EasyPPO sustains learning across tasks. Left:** FrontierCS. **Middle:** AIME. **Right:** Search-R1. **Top:** training scores. **Bottom:** validation scores. Gray marks critic warmup.

task. Training curves use a five-point centered moving average with faint raw values; validation is unsmoothed. Full configurations are provided in Appendix A.

EasyPPO Stabilizes Critic Learning and Policy Training. Across the evaluated runs and training horizons in Figure 4, EasyPPO remains stable and achieves the best validation performance among the compared methods on all three tasks. Comparing each method's best validation checkpoint on a $0 - 100$ scale, EasyPPO improves over PPO by 1.92, 1.46, and 3.74 points on FrontierCS, AIME24, and Search-R1, respectively; gains over the second-best method on each task are 0.91, 1.46, and 0.89 points (Table H). The stability benefit is particularly pronounced on continuous-score coding, consistent with our analysis of critic instability under heterogeneous returns. Every baseline experiences performance collapse in at least one setting, despite achieving substantial scores earlier in training. With the EasyPPO recipe, we observe no training collapse in any of the evaluated settings.

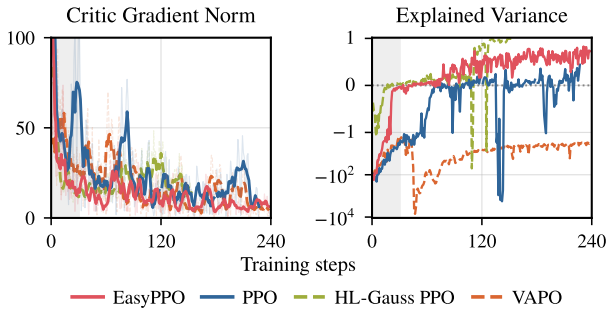

Figure 5: **EasyPPO stabilizes critic learning.** Critic diagnostics on FrontierCS. **Left:** critic gradient norm before clipping. **Right:** explained variance of critic value predictions. Gray marks critic warmup.

PPO with actor-only filtering achieves higher FrontierCS scores than vanilla PPO, but its scores still fluctuate substantially. It also collapses on AIME. EasyPPO remains stable with the same filtering rule, showing the benefit of our critic modifications.

To examine the critic behavior behind these gains, Figure 5 shows critic gradient norms before clipping and explained variance on FrontierCS. After warmup, EasyPPO's critic gradient norm declines and its explained variance follows a steady upward trend with relatively small fluctuations, whereas

the baselines exhibit sharp swings in explained variance. HL-Gauss approaches an explained variance of 1 only after its actor collapses. In contrast, EasyPPO's improving critic predictions accompany sustained reward gains, supporting our motivation for stabilizing critic learning to improve PPO training. We defer AIME and Search-R1 critic diagnostics to Appendix C.2.

Sensitivity to Random Seeds. To test seed sensitivity, we compare three runs each of EasyPPO and PPO + actor-only filtering, the strongest baseline on FrontierCS. Due to training cost, we restrict this study to these two methods on FrontierCS. None of the three EasyPPO runs collapses, whereas two of the three actor-only-filtering runs do. EasyPPO also sustains its validation gains with a narrow min–max band across runs (Figure 6). Further details are provided in Appendix C.3.

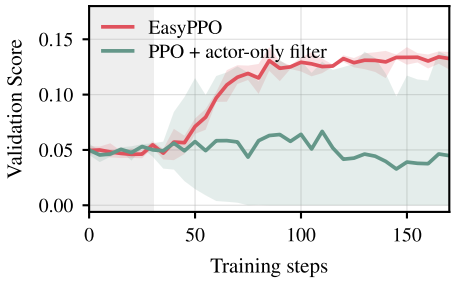

Figure 6: **Stability across critic initializations.**

Noise Normalization Improves Stability Across Critic Mini-Batch Sizes. We ablate noise-normalized critic regression on FrontierCS. Within each mini-batch configuration, all other training settings remain unchanged. We use a fixed critic learning rate of 2×10^{-6} for this ablation.

Figure 7 shows that noise normalization stabilizes training with both mini-batch configurations. Without normalization, the performance drop is recoverable with one mini-batch but becomes a sustained collapse when the same rollout batch is split into four smaller mini-batches. With normalization, both configurations preserve their gains in training and validation scores. This contrast is consistent with our analysis: smaller mini-batches average out less return noise. These results suggest that noise normalization enables stable training with smaller critic mini-batches, allowing finer-grained clipping to limit outlier influence as analyzed in Section 4.

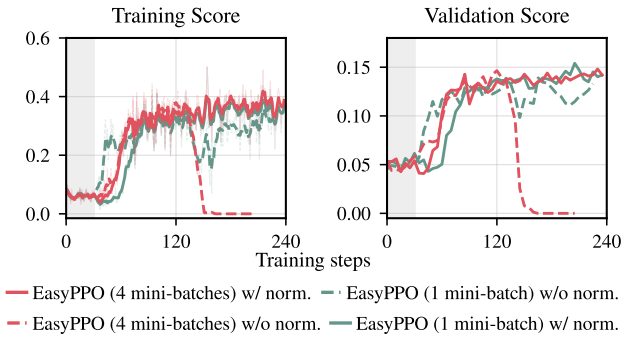

Figure 7: **Noise normalization stabilizes training with various critic mini-batch configurations.** FrontierCS ablation. **Left:** training score. **Right:** validation score.

The Trade-off in Critic Mini-Batch Size. We next vary the number of critic mini-batches per rollout batch, $K = B/m$, on FrontierCS, fixing $B = 512$ and retaining actor-only filtering and noise normalization. We scale $\eta \propto 1/\sqrt{K}$ from the default at $K = 4$ to account for update frequency (Li et al., 2026b); clipping granularity and optimizer dynamics still vary together, but the observed trade-off is consistent with our fixed-parameter analysis.

Figure 8 shows five-point-smoothed EV (left) and the first step with raw EV ≥ 0 (right), including warmup. First-crossing steps decrease approximately linearly with $\log K$, whereas $K = 4$ and 8 achieve higher EV later in training. This suggests finer-grained outlier control helps early, while noise averaging matters more as prediction error decreases (Equation 5). Together, this analysis and the observed trade-off motivate our default choice of $K = 4$.

6 Related Work

Policy Optimization for LLM Post-Training. Reinforcement learning from human feedback (RLHF) established PPO as a standard actor–critic method for LLM post-training (Ouyang et al., 2022; Schulman et al., 2017), while RLVR replaces learned reward models with task-specific verifiers (Guo et al., 2025; Shao et al., 2024).

Critic-free methods estimate advantages from groups of responses to the same prompt (Ahmadian et al., 2024; He et al., 2025; Hu et al., 2025a; Shao et al., 2024), commonly filtering truncated rollouts from the update (Wang et al., 2025; Yu et al., 2025).

Actor–critic methods keep a critic for token-level credit assignment, and recent work improves its training through value pretraining and decoupled GAE (Yuan et al., 2025; Yue et al., 2025), categorical value prediction (Zhou et al., 2026), or hybrid designs (Pan et al., 2026a; Qi et al., 2026). Concurrent work supervises the critic sparsely to counter value flattening inside a response (Li et al., 2026a), takes multiple critic updates per rollout batch (Hu et al., 2025b), excludes length-penalty rewards from critic training (Pan et al., 2026b), or reweights actor baselines by length (Cognition Team, 2026). Each addresses one piece of critic training; EasyPPO targets the critic's exposure to prompt-level return noise, showing that filtering both actor and critic shifts the policy objective, normalizing critic-gradient noise across prompts, and bounding each mini-batch's influence.

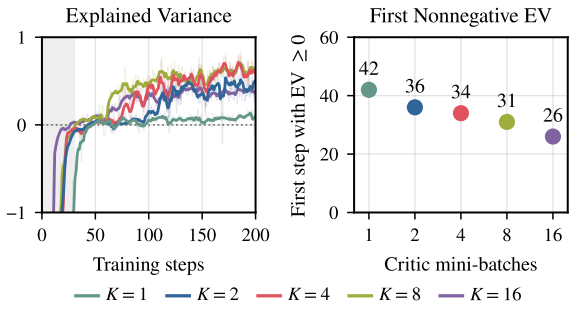

Figure 8: **Explained variance across critic mini-batch sizes.** FrontierCS. **Left:** smoothed explained variance; gray marks critic warmup. **Right:** first training step with $\mathrm{EV} \geq 0$ versus critic mini-batch count, measured from unsmoothed values.

Continuous Verifiable Rewards. While mathematical and coding RLVR often use binary correctness rewards, open-ended optimization admits continuously varying solution quality. Recent benchmarks provide verifiable graded scores for algorithm design, ML research, and GPU kernel optimization (He et al., 2026; Kong et al., 2026; Lyu et al., 2026; Mang et al., 2025; Xing et al., 2026; Zhu et al., 2026). These benchmarks also provide training data: Evolution Fine-Tuning uses evolutionary search trajectories from FrontierCS for supervised fine-tuning (Lee et al., 2026). For RL with continuous rewards, TailRL (Ramasubramanian et al., 2026) targets upper-tail outcomes through reward-threshold exceedance probabilities based on GRPO. These settings make differences in return scale and variability across prompts particularly visible, motivating EasyPPO's focus on critic stability under heterogeneous returns.

Variance-Weighted Critic Regression. Variance-aware regression has been studied in both RL and supervised learning. In RL, variance-weighted critic objectives improve statistical efficiency in linear MDPs (Kitamura et al., 2023; Zhou et al., 2021), while IV-RL weights TD errors using estimated target variance (Mai et al., 2022) and PopArt normalizes target scales across tasks (Hessel et al., 2019; Van Hasselt et al., 2016). Related work on heteroscedastic regression similarly adjusts each example's contribution according to its target uncertainty (Kendall & Gal, 2017; Seitzer et al., 2022). Group sampling in LLM RL provides an empirical estimate of prompt-level return variance, which EasyPPO uses for noise-normalized critic regression without an auxiliary variance model. Dr. GRPO finds that standard-deviation normalization of actor advantages can over-weight low-variance prompts (Liu et al., 2025); EasyPPO instead applies this weighting to the critic loss.

7 Conclusion

We identify overlong-rollout handling and heterogeneous return noise as two sources of critic instability in PPO. Our analysis shows how filtering both actor and critic changes the policy objective, and how return noise can dominate critic-gradient second moments. These findings motivate EasyPPO: actor-only filtering, noise-normalized critic regression, and moderately smaller critic mini-batches with gradient clipping, while retaining the standard PPO actor update. Across continuous-score coding, mathematical reasoning, and multi-turn search, the resulting stability and performance gains show that stabilizing the critic is key to reliable PPO training for LLMs.

AI USE STATEMENT

Some of our experiments use synthetic open-ended coding problems from FrontierSmith (He et al., 2026), whose construction pipeline uses LLMs to generate problem formulations, test cases, and scoring verifiers.

We used generative AI tools to assist with developing theoretical analyses, formulating mathematical claims, writing proofs, designing experiments, implementing methods, interpreting results, creating and editing software code, creating and modifying scientific figures, and editing the manuscript to improve readability. The authors reviewed and revised the AI-assisted content and take full responsibility for the final content of this work.

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, 2024.

Cognition Team. Introducing swe-2: Pushing the pareto frontier. `https://cognition.com/blog/swe-2`, September 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Bingxiang He, Zekai Qu, Zeyuan Liu, Yinghao Chen, Yuxin Zuo, Cheng Qian, Kaiyan Zhang, Weize Chen, Chaojun Xiao, Ganqu Cui, et al. Justrl: Scaling a 1.5 b llm with a simple rl recipe. *arXiv preprint arXiv:2512.16649*, 2025.

Runyuan He, Qiuyang Mang, Shang Zhou, Kaiyuan Liu, Hanchen Li, Huanzhi Mao, Qizheng Zhang, Zerui Li, Bo Peng, Lufeng Cheng, et al. Frontiersmith: Synthesizing open-ended coding problems at scale. *arXiv preprint arXiv:2605.14445*, 2026.

Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado Van Hasselt. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3796–3803, 2019.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. Reinforce++: Stabilizing critic-free policy optimization with global advantage normalization. *arXiv preprint arXiv:2501.03262*, 2025a.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *Advances in Neural Information Processing Systems*, 38:162239–162262, 2025b.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

Toshinori Kitamura, Tadashi Kozuno, Yunhao Tang, Nino Vieillard, Michal Valko, Wenhao Yang, Jincheng Mei, Pierre Ménard, Mohammad Gheshlaghi Azar, Rémi Munos, et al. Regularization and variance-weighted regression achieves minimax optimality in linear mdps: Theory and practice. In *International Conference on Machine Learning*, pp. 17135–17175. PMLR, 2023.

Minwei Kong, Chonghe Jiang, Ao Qu, Wenbin Ouyang, Zhaoming Zeng, Xiaotong Guo, Zhekai Li, Junyi Li, Yi Fan, Xinshou Zheng, et al. Frontieror: Benchmarking llms' capacity for efficient algorithm design in large-scale optimization. *arXiv preprint arXiv:2605.25246*, 2026.

Young-Jun Lee, Seungone Kim, Minki Kang, Alistair Cheong Liang Chuen, Zerui Chen, Seungho Han, Taehee Jung, and Dongyeop Kang. Evolution fine-tuning: Learning to discover across 371 optimization tasks. *arXiv preprint arXiv:2606.29082*, 2026.

Yizhuo Li, Jianhao Yan, Yun Luo, Zhi Wang, Futing Wang, Rong-Xi Tan, Kanghui Tian, Ganqu Cui, Ning Ding, Peilin Zhao, Yafu Li, and Yu Cheng. Rethinking critic learning in PPO: Understanding and mitigating value flattening. *arXiv preprint arXiv:2609.18708*, 2026a. URL https://arxiv.org/abs/2609.18708.

Ziniu Li, Jinbo Wang, Guanhua Huang, Feiyuan Zhang, Pengbo Li, and Alex Chen. When do larger batches help scale llm reinforcement learning? *arXiv preprint arXiv:2608.29296*, 2026b.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Bohan Lyu, Yucheng Yang, Siqiao Huang, Jiaru Zhang, Qixin Xu, Xinghan Li, Xinyang Han, Yicheng Zhang, Huaqing Zhang, Runhan Huang, et al. Mls-bench: A holistic and rigorous assessment of ai systems on building better ai. *arXiv preprint arXiv:2605.08678*, 2026.

Vincent Mai, Kaustubh Mani, and Liam Paull. Sample efficient deep reinforcement learning via uncertainty estimation. *arXiv preprint arXiv:2201.01666*, 2022.

Qiuyang Mang, Wenhao Chai, Zhifei Li, Huanzhi Mao, Shang Zhou, Alexander Du, Hanchen Li, Shu Liu, Edwin Chen, Yichuan Wang, et al. Frontiercs: Evolving challenges for evolving intelligence. *arXiv preprint arXiv:2512.15699*, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Chengjun Pan, Shichun Liu, Jiahang Lin, Dingwei Zhu, Jiazheng Zhang, Shihan Dou, Songyang Gao, Zhenhua Han, Binghai Wang, Rui Zheng, et al. Evpo: Explained variance policy optimization for adaptive critic utilization in llm post-training. *arXiv preprint arXiv:2604.19485*, 2026a.

Haoxuan Pan et al. Justrl-ii: Scaling small llms to 128k reasoning with a critic. https://panhaoxuan.notion.site/justrl-ii-scaling-small-llms-to-128k-reasoning-with-a-critic, 2026b. Chinese version: https://panhaoxuan.notion.site/justrl-ii-small-llms-to-128k-reasoning-with-a-critic-cn.

Penghui Qi, Xiangxin Zhou, and Wee Sun Lee. Best practice critic optimization. *arXiv preprint arXiv:2608.23566*, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Shrinivas Ramasubramanian, Daman Arora, Fahim Tajwar, Guanning Zeng, Qingyang Wu, Zhongzhu Zhou, Chenfeng Xu, Haiwen Feng, Yuda Song, Aarti Singh, et al. Tail-likelihood reinforcement learning. *arXiv preprint arXiv:2609.02987*, 2026.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Maximilian Seitzer, Arash Tavakoli, Dimitrije Antic, and Georg Martius. On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks. *arXiv preprint arXiv:2203.09168*, 2022.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Hado P Van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude. *Advances in neural information processing systems*, 29, 2016.

Boxin Wang, Chankyu Lee, Nayeon Lee, Sheng-Chieh Lin, Wenliang Dai, Yang Chen, Yangyi Chen, Zhuolin Yang, Zihan Liu, Mohammad Shoeybi, et al. Nemotron-cascade: Scaling cascaded reinforcement learning for general-purpose reasoning models. *arXiv preprint arXiv:2512.13607*, 2025.

Shanli Xing, Yiyan Zhai, Alexander Jiang, Yixin Dong, Yong Wu, Zihao Ye, Charlie F Ruan, Yingyi Huang, Yineng Zhang, Liangsheng Yin, et al. Flashinfer-bench: Building the virtuous cycle for ai-driven llm systems. *Proceedings of Machine Learning and Systems*, 8:2016–2064, 2026.

Zhangchen Xu, Junda Chen, Yue Huang, Dongfu Jiang, Jiefeng Chen, Hang Hua, Zijian Wu, Zheyuan Liu, Zexue He, Lichi Li, et al. Autolab: Can frontier models solve long-horizon auto research and engineering tasks? *arXiv preprint arXiv:2606.05080*, 2026.

Shuo Yang, Haocheng Xi, Yilong Zhao, Qiuyang Mang, Zhe Wang, Shanlin Sun, Kurt Keutzer, Joseph E. Gonzalez, Song Han, Chenfeng Xu, and Ion Stoica. Flashlib: Bringing flash magic to classical machine learning operators, 2026. URL `https://flashml-org.github.io/`.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2025.

Yufeng Yuan, Yu Yue, Ruofei Zhu, Tiantian Fan, and Lin Yan. What's behind ppo's collapse in long-cot? value optimization holds the secret. *arXiv preprint arXiv:2503.01491*, 2025.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.

Dongruo Zhou, Quanquan Gu, and Csaba Szepesvari. Nearly minimax optimal reinforcement learning for linear mixture markov decision processes. In *Conference on Learning Theory*, pp. 4532–4576. PMLR, 2021.

Zhijian Zhou, Long Li, Xuan Zhang, Zongkai Liu, Yulei Qin, Ke Li, Xing Sun, Xiaoyu Tan, Chao Qu, and Yuan Qi. Start classifying: Categorical critics for llm reinforcement learning. *arXiv preprint arXiv:2608.02181*, 2026.

Deyao Zhu, Xin Zhou, Shengling Qin, Xuekai Zhu, Hangliang Ding, Shu Zhong, Zixin Wen, Zhonglin Xie, Chenhui Gou, Linxuan Ren, et al. Edgebench: Unveiling scaling laws of learning from real-world environments. *arXiv preprint arXiv:2607.05155*, 2026.

APPENDIX: CONTENTS

A Training Configurations and Implementation Details 13
A.1 Data and FrontierCS Initialization 13
A.2 Training and Evaluation Settings 13
A.3 Critic and Baseline Configurations 13
A.4 Diagnostic and Ablation Configurations 15
B Theoretical Analyses 16
B.1 Effect of Return Heterogeneity on Critic Updates 16
B.2 Actor-only Overlong Filtering 17
B.3 Noise-Normalized Critic Regression 18
B.4 Critic Mini-Batch Clipping 19
C Additional Experimental Results 20
C.1 Best Validation Scores 20
C.2 Critic Diagnostics on AIME and Search-R1 20
C.3 Different Critic Initializations on FrontierCS 21
C.4 Offline Return-Variability and Gradient Diagnostics 21
C.5 Overlong Filtering on AIME 22
C.6 Search-R1 Results on Individual Validation Sets 24

A TRAINING CONFIGURATIONS AND IMPLEMENTATION DETAILS

This section provides the configurations for Section 5 and the filtering study in Section 3.

A.1 DATA AND FRONTIERCS INITIALIZATION

For our FrontierCS experiments, we use a distilled Qwen3.5-9B (Qwen Team, 2026) model as the initial policy. SFT and RL use a separate FrontierSmith training set; validation uses the algorithmic track of FrontierCS (Mang et al., 2025). We use DeepSeek-V3.1 to sample three responses for each of the 200 problems generated by FrontierSmith (He et al., 2026), yielding 600 responses in total. Each response is truncated to at most 32,768 tokens, with length measured using the Qwen3.5-9B tokenizer. After removing 253 responses with a score of zero, we retain 347 responses covering 179 distinct problems. We perform supervised fine-tuning (SFT) of Qwen3.5-9B on these retained responses for a single epoch to obtain the distilled model.

A.2 TRAINING AND EVALUATION SETTINGS

AIME and Search-R1 start from Qwen3.5-9B-Base; FrontierCS uses the SFT initialization above. Table A summarizes the task settings for Figure 4. Batch sizes count responses; the number of prompts per batch is listed separately. Table B lists the PPO and optimizer settings; method-specific critic settings follow in Appendix A.3. Learning rates are constant after any optimizer warmup.

Critic warmup is reported in rollout steps; the optimizer learning-rate warmup is listed separately.

The actor KL loss uses the `low_var_kl` estimator.

A.3 CRITIC AND BASELINE CONFIGURATIONS

Table C summarizes the critic configurations in Figure 4. EasyPPO uses $K = 4$ critic mini-batches per rollout batch, each with $m = B/K$ responses: 128 on FrontierCS and AIME, and 256 on

Table A: Training and evaluation configurations for the main comparisons. Learning rates are the configured target values.

Hyperparameter	FrontierCS	AIME	Search-R1
Prompts per rollout batch	16	32	64
Samples per prompt	32	16	16
Rollout batch size B	512	512	1024
Maximum prompt length	8192	2048	4096
Maximum response length	32768	8192	4096
Rollout temperature	1.0	1.0	1.0
Rollout top-p	1.0	1.0	1.0
Actor learning rate	10^{-6}	10^{-6}	10^{-6}
Default critic learning rate	2×10^{-6}	2×10^{-6}	2×10^{-6}
Actor updates per rollout batch	1	1	1
Critic epochs per rollout batch	1	1	1
Actor/critic LR warmup (optimizer steps)	0	20	0
Critic warmup steps	30	30	30
Validation responses per problem	5	32	1
Validation decoding	Sampling	Sampling	Greedy
Validation temperature	1.0	1.0	0.0
Validation top-p	1.0	0.7	1.0

Table B: PPO and optimizer settings. The Search-R1 upper clipping parameter is 0.28 for VAPO and 0.2 for the other methods.

Hyperparameter	FrontierCS	AIME	Search-R1
Discount factor γ	1	1	1
GAE parameter λ	1	1	1
PPO lower clipping parameter	0.2	0.2	0.2
PPO upper clipping parameter	0.2	0.28	0.2 / 0.28
Dual-clip parameter	3	3	3
MSE value-clipping parameter	0.2	0.2	0.2
Actor / critic gradient clipping	1 / 1	1 / 1	1 / 1
Actor / critic PPO epochs	1 / 1	1 / 1	1 / 1
Actor entropy coefficient	0	0	0
Actor KL-loss coefficient	10^{-3}	10^{-3}	10^{-3}
KL penalty in reward	Off	Off	Off
Actor / critic loss aggregation	Token mean	Token mean	Token mean
Optimizer	AdamW	AdamW	AdamW
AdamW (β_1, β_2)	$(0.9, 0.999)$	$(0.9, 0.999)$	$(0.9, 0.999)$
Weight decay	0.01	0.01	0.01
AdamW numerical epsilon	1e-8	1e-8	1e-8

Search-R1. Table D lists the EasyPPO-specific settings. All methods retain truncated responses for critic training. VAPO retains its value-learning and advantage-estimation recipe, while its auxiliary positive-example language-modeling loss is disabled as described in Section 5.

Table E gives the task-specific categorical-critic settings used by our HL-Gauss PPO baseline (Zhou et al., 2026).

Table C: Critic objectives, overlong filtering, and update counts in the main comparisons. K is the number of critic mini-batches per rollout batch.

Method	Critic objective	Overlong filtering	K
PPO	MSE	None	1
PPO + actor-only filtering	MSE	Actor only	1
HL-Gauss PPO	HL-Gauss	None	1
VAPO	MSE	None	1
EasyPPO	Noise-normalized MSE	Actor only	4

Table D: EasyPPO-specific configurations. Shared PPO settings appear in Table B.

Hyperparameter	FrontierCS	AIME	Search-R1
Critic mini-batches K	4	4	4
Responses per critic mini-batch m	128	128	256
Critic learning rate	2×10^{-6}	2×10^{-6}	2×10^{-6}
Overlong filtering	Actor only	Actor only	Actor only
Return-STD floor ε	0.075	0.25	0.125

Table E: HL-Gauss critic configurations used in our experiments.

Hyperparameter	FrontierCS	AIME	Search-R1
Number of bins	101	101	101
Support range	$[-0.1, 1.1]$	$[-1.1, 1.1]$	$[-0.1, 1.1]$
Smoothing bandwidth	0.009	0.0165	0.009

A.4 Diagnostic and Ablation Configurations

Overlong filtering. For Figure 2, we use Qwen3.5-9B (Qwen Team, 2026) and train on 200 problems generated by FrontierSmith (He et al., 2026). We use the distilled initialization described in Appendix A.1. We warm up the critic for 30 steps before updating the policy with PPO, using rollout batches of 512. The maximum response length is 32,768 tokens. The three runs compare no overlong filtering, filtering both actor and critic updates, and filtering actor updates only. The figure reports training-rollout statistics; the algorithmic track of FrontierCS (Mang et al., 2025) is used for validation.

Return-variability diagnostics. Table F summarizes the shared settings and differences between Figures 2 and 3. Figure 3 uses checkpoints from an actor-only-filtered, noise-normalized training run, and compares the same offline response gradients before and after weighting. Its diagnostic return-STD floor is $\varepsilon = 0.075$; the corresponding AIME diagnostic uses 0.5. The diagnostic uses terminal-return MSE gradients before clipping, without taking an optimizer step; its weighting and token denominator are detailed in Appendix C.4.

Noise normalization and critic mini-batches. Table G lists the configurations plotted in Figures 7 and 8. All use FrontierCS with $B = 512$, actor-only overlong filtering, and the initialization in Appendix A.1. Figure 7 holds the critic learning rate fixed to isolate normalization within each mini-batch configuration. Figure 8 instead scales it as $\eta \propto 1/\sqrt{K}$ from the default $K = 4$ setting.

Table F: FrontierCS settings for the filtering study and offline gradient diagnostic. Both use the distilled initialization from Appendix A.1, 512 responses per rollout batch, group size 32, a 32,768-token response limit, and 30 critic warmup steps.

Setting	Figure 2	Figure 3
Actor learning rate	10^{-6}	10^{-6}
Critic learning rate	2×10^{-6}	2×10^{-6}
Critic mini-batches per rollout batch	1	4
Noise normalization during training	Off	On
Overlong filtering during training	None / both / actor only	Actor only
Diagnostic checkpoints	—	30, 60, 90
Responses analyzed per checkpoint	—	512
Diagnostic return-STD floor	—	0.075

Table G: Configurations used in the FrontierCS ablation figures. $m = 512/K$ counts responses per critic mini-batch. Only plotted configurations are listed.

Experiment	K	m	Noise normalization	Critic learning rate
Figure 7	1	512	On / off	2×10^{-6}
Figure 7	4	128	On / off	2×10^{-6}
Figure 8	1	512	On	4×10^{-6}
Figure 8	2	256	On	$2\sqrt{2} \times 10^{-6}$
Figure 8	4	128	On	2×10^{-6}
Figure 8	8	64	On	$\sqrt{2} \times 10^{-6}$
Figure 8	16	32	On	1×10^{-6}

B Theoretical Analyses

B.1 Effect of Return Heterogeneity on Critic Updates

This derivation supports the gradient second-moment decomposition in Equation 5.

We provide a simple qualitative analysis to illustrate how heterogeneous return noise affects critic optimization.

Let the critic be parameterized as

$$V_\phi(s) = W^\top h_\phi(s), \tag{8}$$

where $h_\phi(s) \in \mathbb{R}^d$ denotes the hidden representation produced by the critic backbone and $W \in \mathbb{R}^d$ is the linear value head. Given a sampled rollout return R, the critic minimizes the squared value loss

$$\ell = \frac{1}{2} \left(V_\phi(s) - R\right)^2 . \tag{9}$$

The gradient with respect to the value head is

$$\begin{aligned} \nabla_W \ell &= \left(V_\phi(s) - R\right) \nabla_W V_\phi(s) \\ &= \left(V_\phi(s) - R\right) h_\phi(s). \end{aligned} \tag{10}$$

For a fixed prompt s, define the conditional mean and variance of the rollout return as

$$\mu_s \triangleq \mathbb{E}[R \mid s], \qquad \sigma_s^2 \triangleq \mathrm{Var}(R \mid s). \tag{11}$$

We first consider the expected value-head gradient. Taking expectation over rollout returns conditioned on s gives

$$\begin{aligned}\mathbb{E}\left[\nabla_W \ell \mid s\right] &= \mathbb{E}\left[\left(V_\phi(s) - R\right) h_\phi(s) \mid s\right] \\ &= \left(V_\phi(s) - \mu_s\right) h_\phi(s).\end{aligned} \tag{12}$$

Hence, the expected gradient vanishes when the critic accurately predicts the expected return, $V_\phi(s) = \mu_s$. However, individual sampled returns may still induce nonzero stochastic updates.

To see this, consider the squared norm of the value-head gradient:

$$\begin{aligned}\|\nabla_W \ell\|_2^2 &= \|\left(V_\phi(s) - R\right) h_\phi(s)\|_2^2 \\ &= \left(V_\phi(s) - R\right)^2 \|h_\phi(s)\|_2^2 .\end{aligned} \tag{13}$$

Taking the conditional expectation yields

$$\mathbb{E}\left[\|\nabla_W \ell\|_2^2 \mid s\right] = \|h_\phi(s)\|_2^2 \, \mathbb{E}\left[\left(V_\phi(s) - R\right)^2 \mid s\right]. \tag{14}$$

We can decompose the remaining term by adding and subtracting the conditional mean μ_s:

$$V_\phi(s) - R = \left(V_\phi(s) - \mu_s\right) + \left(\mu_s - R\right). \tag{15}$$

Therefore,

$$\begin{aligned}\mathbb{E}\big[\left(V_\phi(s) - R\right)^2 \mid s\big] &= \left(V_\phi(s) - \mu_s\right)^2 + 2\left(V_\phi(s) - \mu_s\right)\mathbb{E}[\mu_s - R \mid s] \\ &\quad + \mathbb{E}\big[\left(R - \mu_s\right)^2 \mid s\big].\end{aligned} \tag{16}$$

By definition,

$$\mathbb{E}\left[\mu_s - R \mid s\right] = \mu_s - \mathbb{E}[R \mid s] = 0, \tag{17}$$

while

$$\mathbb{E}\left[\left(R - \mu_s\right)^2 \mid s\right] = \sigma_s^2. \tag{18}$$

Thus,

$$\mathbb{E}\left[\left(V_\phi(s) - R\right)^2 \mid s\right] = \left(V_\phi(s) - \mu_s\right)^2 + \sigma_s^2. \tag{19}$$

Substituting Equation 19 into Equation 14, we obtain

$$\boxed{\mathbb{E}\left[\|\nabla_W \ell\|_2^2 \mid s\right] = \|h_\phi(s)\|_2^2 \left[\underbrace{\left(V_\phi(s) - \mathbb{E}[R \mid s]\right)^2}_{\text{prediction error}} + \underbrace{\mathrm{Var}(R \mid s)}_{\text{return variability}}\right].} \tag{20}$$

B.2 Actor-only Overlong Filtering

This analysis supports Section 3.

We analyze a fixed prompt and omit it from the notation. Let C denote the event that a trajectory is not truncated. We consider the on-policy gradient without PPO clipping, which characterizes the first-order update around the behavior policy.

Conditional objective under double-sided filtering. If the critic discards truncated trajectories, the population minimizer of its squared loss is

$$V_{\text{filt}} = \mathbb{E}_{\pi_\theta}[R(\tau) \mid C].$$

The conditional expected reward is

$$\mathbb{E}_{\pi_\theta}[R(\tau) \mid C] = \frac{\sum_{\tau \in C} \pi_\theta(\tau) R(\tau)}{P_{\pi_\theta}(C)}.$$

Differentiating the numerator and denominator gives

$$\nabla_\theta \mathbb{E}_{\pi_\theta}[R(\tau) \mid C] = \mathbb{E}_{\pi_\theta}[R(\tau)\nabla_\theta \log \pi_\theta(\tau) \mid C] - V_{\text{filt}}\, \nabla_\theta \log P_{\pi_\theta}(C). \tag{21}$$

The score-function identity also gives

$$\nabla_\theta \log P_{\pi_\theta}(C) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(\tau) \mid C]\,.$$

Therefore,

$$\mathbb{E}_{\pi_\theta}\big[\big(R(\tau) - V_{\text{filt}}\big)\nabla_\theta \log \pi_\theta(\tau) \mid C\big] = \nabla_\theta \mathbb{E}_{\pi_\theta}[R(\tau) \mid C]. \tag{22}$$

The conditional critic supplies the normalization correction in this gradient. Double-sided filtering therefore optimizes expected reward among trajectories that already avoid truncation.

Why the full-distribution critic helps. Actor-only filtering instead uses the full-distribution critic

$$V_{\text{full}} = \mathbb{E}_{\pi_\theta}[R(\tau)].$$

Its filtered actor update decomposes as

$$\begin{aligned}&\mathbb{E}_{\pi_\theta}\big[\big(R(\tau) - V_{\text{full}}\big)\nabla_\theta \log \pi_\theta(\tau) \mid C\big] \\ &\quad = \nabla_\theta \mathbb{E}_{\pi_\theta}[R(\tau) \mid C] + \big(V_{\text{filt}} - V_{\text{full}}\big)\nabla_\theta \log P_{\pi_\theta}(C).\end{aligned} \tag{23}$$

When non-truncated trajectories have higher expected reward than the full rollout distribution, the second term directly favors a higher probability of avoiding truncation. Truncated trajectories do not enter the actor loss, but their returns still affect this signal through the critic baseline.

Selection bias of actor filtering. Actor filtering differs from the unfiltered policy gradient because it discards truncated trajectories. For a fixed baseline b, assume the norm of each trajectory-level gradient contribution is bounded by Γ. Then

$$\Big\| \begin{aligned}&\mathbb{E}_{\pi_\theta}[(R(\tau) - b)\nabla_\theta \log \pi_\theta(\tau) \mid C] \\ &- \mathbb{E}_{\pi_\theta}[(R(\tau) - b)\nabla_\theta \log \pi_\theta(\tau)]\end{aligned} \Big\|_2 \le 2\Gamma P_{\pi_\theta}(C^{\text{c}}). \tag{24}$$

The discrepancy therefore vanishes as the probability of truncation approaches zero.

Token-level conditional baselines. With Monte Carlo return targets, filtering critic updates changes the squared-loss minimizer at each token state s_t:

$$\arg\min_v \mathbb{E}\big[(R - v)^2 \mid s_t, C\big] = \mathbb{E}[R \mid s_t, C].$$

For an accurate filtered critic, the expected advantage of token a_t is therefore

$$\mathbb{E}[R - V_\phi(s_t) \mid s_t, a_t, C] = \mathbb{E}[R \mid s_t, a_t, C] - \mathbb{E}[R \mid s_t, C].$$

Each token thus compares its return with the mean among non-truncated continuations from its own state. These advantages weight the token gradients $\nabla_\theta \log \pi_\theta(a_t \mid s_t)$, which are aggregated to update the shared actor parameters. Actor-only filtering instead preserves the critic target $\mathbb{E}[R \mid s_t]$, so truncated returns still affect each token's advantage through its baseline.

B.3 Noise-Normalized Critic Regression

This analysis justifies the prompt-level variance approximation in Section 4.

We show that the prompt-level return variance $\mathrm{Var}(R \mid s)$ upper-bounds the average prefix-level conditional return variance across token states s_t generated from prompt s.

Law of Total Variance Across Token Prefixes. Fix an initial prompt state s and a generation step t. Let $s_t = (s, y_{<t})$ denote the random token prefix generated by π_θ at step t, and let R denote the terminal return of the sampled rollout. The true state-value function at step t is the conditional expectation $V^{\pi_\theta}(s_t) = \mathbb{E}[R \mid s_t]$. Conditioning on the initial prompt s, the law of total variance decomposes the prompt-level return variance $\sigma_s^2 = \mathrm{Var}(R \mid s)$ into

$$\sigma_s^2 = \mathrm{Var}(R \mid s) = \underbrace{\mathbb{E}_{s_t|s}[\mathrm{Var}(R \mid s_t)]}_{\text{Expected prefix-level return variance}} + \underbrace{\mathrm{Var}_{s_t|s}[V^{\pi_\theta}(s_t)]}_{\text{Variance of true state values across prefixes}}. \tag{25}$$

Because $\mathrm{Var}_{s_t|s}[V^{\pi_\theta}(s_t)] \ge 0$, the expected prefix-level return variance satisfies the upper bound

$$\mathbb{E}_{s_t|s}[\mathrm{Var}(R \mid s_t)] \le \sigma_s^2. \tag{26}$$

Bound on the Prefix Value-Gradient Second Moment. For the token-level squared value loss $\mathcal{L}_V(\phi) = \frac{1}{2}(V_\phi(s_t) - R)^2$ with stochastic gradient $\mathbf{g}_t = (V_\phi(s_t) - R)\nabla_\phi V_\phi(s_t)$, the conditional expectation of the squared residual at state s_t decomposes as

$$\mathbb{E}\Big[\big(V_\phi(s_t) - R\big)^2 \,\Big|\, s_t\Big] = \big(V_\phi(s_t) - V^{\pi_\theta}(s_t)\big)^2 + \mathrm{Var}(R \mid s_t). \tag{27}$$

Assuming the critic Jacobian norm is locally bounded by $\|\nabla_\phi V_\phi(s_t)\|_2 \le J$ across prefixes of prompt s, taking the expectation over $s_t \mid s$ and applying Inequality 26 yields

$$\mathbb{E}\big[\|\mathbf{g}_t\|_2^2 \,\big|\, s\big] \le J^2 \mathbb{E}_{s_t|s}\Big[\big(V_\phi(s_t) - V^{\pi_\theta}(s_t)\big)^2\Big] + J^2\sigma_s^2. \tag{28}$$

The noise contribution to the prefix-averaged gradient second moment is bounded by $J^2\sigma_s^2$. Multiplying the per-prompt critic loss by $w(s)$ scales $\mathbf{g}_t$ by $w(s)$ and changes the return-noise term in the bound in Equation 28 to $w(s)^2J^2\sigma_s^2$. For positive prompt variance, selecting $w(s) \propto \mathrm{Var}(R \mid s)^{-1/2}$ renders the noise bound $w(s)^2J^2\sigma_s^2 \propto J^2$ invariant to the prompt-level return standard deviation.

For discrete rewards with range Δ and group size $n \ge 2$, consider a reference group containing one maximum reward and $n-1$ minimum rewards (or vice versa). Its empirical variance is $\hat{\sigma}^2 = \Delta^2(1/n)(1-1/n)$, using variance divisor n. We choose the STD floor for groups with identical returns to be approximately half this reference STD:

$$\frac{\hat{\sigma}}{2} = \frac{\Delta\sqrt{n-1}}{2n} = \frac{\Delta}{2\sqrt{n}}\sqrt{1-\frac{1}{n}} \approx \frac{\Delta}{2\sqrt{n}} = \varepsilon.$$

Under inverse-STD weighting, their weight relative to the reference group is therefore $\hat{\sigma}/\varepsilon = 2\sqrt{1-1/n} \approx 2$. For the continuous-reward FrontierCS setting, we retain the floor $\varepsilon = 0.075$ used in our initial experiments, given the cost of rerunning the experimental suite.

B.4 Critic Mini-Batch Clipping

This analysis supports the outlier–noise trade-off in Section 4 and the experiment in Figure 8.

We prove the scaling trends in Section 4 using an idealized comparison at fixed critic parameters and regression weights. Let $z_1, \dots, z_B$ be independent rollout-gradient contributions with common mean μ and finite covariance Σ. Partition them into $M = B/m$ disjoint mini-batches $\mathcal{I}_k$ of size m. For a fixed clipping threshold $c > 0$, define $C_c(g) = g/\max\{1, \|g\|_2/c\}$ and

$$g_k = \frac{1}{m}\sum_{i\in\mathcal{I}_k} z_i, \qquad \bar{g}_c = \frac{1}{M}\sum_{k=1}^{M} C_c(g_k). \tag{29}$$

The normalized average $\bar{g}_c$ isolates the effect of clipping granularity at a fixed rollout budget.

Single-outlier influence. Replace one rollout gradient by an arbitrary vector. Only its mini-batch gradient g_j changes, to g_j', giving a new average $\bar{g}_c'$. Since $\|C_c(g)\|_2 \le c$ for every g, the triangle inequality gives

$$\begin{aligned}\|\bar{g}_c' - \bar{g}_c\|_2 &= \frac{1}{M}\|C_c(g_j') - C_c(g_j)\|_2 \\ &\le \frac{\|C_c(g_j')\|_2 + \|C_c(g_j)\|_2}{M} \le \frac{2c}{M} = \frac{2cm}{B}.\end{aligned} \tag{30}$$

At fixed B and c, this yields the $O(m)$ outlier-sensitivity bound stated in the main text. This deterministic bound does not require independence.

Mini-batch gradient noise. For the original, uncontaminated gradients, independence gives

$$\begin{aligned}\mathrm{Cov}(g_k) &= \frac{1}{m^2}\sum_{i\in\mathcal{I}_k}\mathrm{Cov}(z_i) = \frac{\Sigma}{m}, \\ \mathbb{E}\|g_k - \mu\|_2^2 &= \mathrm{tr}(\mathrm{Cov}(g_k)) = \frac{\mathrm{tr}(\Sigma)}{m}.\end{aligned} \tag{31}$$

Table H: Best validation scores. Scores use a 0–100 scale; bold and underlining mark the best and second best per task.

Method	FrontierCS	AIME24	Search-R1
PPO	12.90	<u>64.06</u>	39.48
PPO + actor-only filter	<u>13.91</u>	60.73	41.74
HL-Gauss PPO	10.43	63.13	<u>42.33</u>
VAPO	5.58	54.90	40.36
EasyPPO	**14.82**	**65.52**	**43.22**

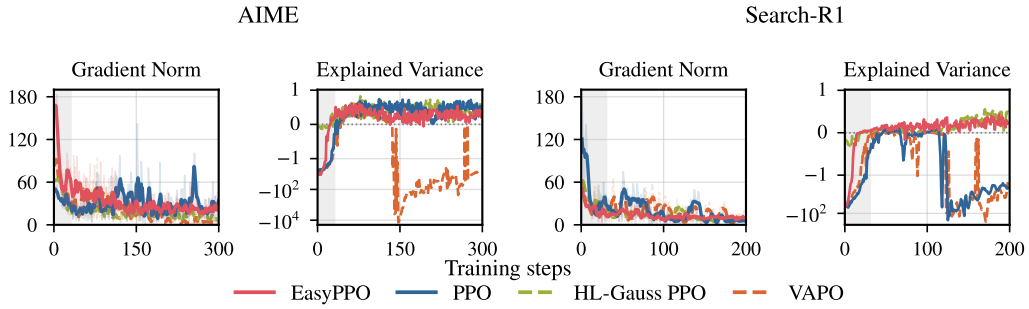

Figure A: Critic diagnostics on AIME and Search-R1. First two panels: AIME. **Last two:** Search-R1. Each pair shows critic gradient norm before clipping and explained variance of value predictions; the latter uses symmetric-log axes. Gray marks critic warmup.

Thus, smaller mini-batches present larger stochastic fluctuations to each clipping operation. Without clipping, the average over all mini-batches has covariance Σ/B, independent of m. The $O(1/m)$ trend therefore concerns the noise entering each clipping operation, rather than the variance of the full-batch average.

These results explain the competing effects of mini-batch size in a fixed-parameter model. Sequential Adam steps recompute gradients and update optimizer state, so Equation 30 does not bound the final parameter change. The variance identity assumes independent rollout contributions; correlated rollouts or weights estimated jointly from a group may change this scaling.

C Additional Experimental Results

C.1 Best Validation Scores

Table H reports each method's best unsmoothed validation score from the runs and training horizons in Figure 4, including critic warmup. Search-R1 uses the mean over its seven validation datasets at a single checkpoint. Score gains are computed before rounding, with the second-best method selected separately for each task.

C.2 Critic Diagnostics on AIME and Search-R1

These results extend Figure 5 in Section 5.

Figure A extends the critic diagnostics to the AIME and Search-R1 runs in Figure 4. Gradient norms are measured before clipping and shown with a five-point centered moving average over faint raw values; explained variance is unsmoothed. On both tasks, EasyPPO avoids the large negative excursions in explained variance seen in VAPO on AIME and in PPO and VAPO on Search-R1. The behavior remains task-dependent: HL-Gauss retains positive explained variance during its Search-R1 performance decline, so these diagnostics should be interpreted alongside the task scores.



Figure B: EasyPPO maintains stable learning across critic initializations. FrontierCS. **Left:** training score. **Middle:** validation mean and min–max. **Right:** explained variance. Gray marks critic warmup.

C.3 Different Critic Initializations on FrontierCS

The runs in Figure 6 use three different critic initializations per method, with matched core training settings; execution configurations, including GPU count, vary across runs. Validation is unsmoothed, and every mean and min–max interval uses all three runs over the shared steps 0–170.

Figure B provides additional EasyPPO diagnostics over the available horizons. All three runs maintain training gains. Run 3 ends at step 172; this figure's validation mean and min–max summarize three runs through step 170 and two from step 175 onward, without extrapolation.

C.4 Offline Return-Variability and Gradient Diagnostics

These diagnostics extend Figure 3 in Section 4. We analyze 512 rollouts from each of the checkpoints at steps 30, 60, and 90 in the FrontierCS and AIME training settings, giving 1,536 responses per setting. Each FrontierCS batch contains 16 prompts with 32 responses each; each AIME batch contains 32 prompts with 16 responses each. The latter are training rollouts, not AIME24 validation responses. The step labels are PPO global training iterations (rollout batches), not individual critic optimizer steps. The FrontierCS source run shares the model initialization, training data, 512-rollout batches, group size 32, 30-step critic warmup, and 32,768-token response limit of Figure 2. It uses noise-normalized critic learning with four critic mini-batch updates per rollout batch, whereas the actor-only filtering reference in Figure 2 uses one critic mini-batch without noise normalization. Using reconstructed response text, we compute the full-critic gradient of terminal-return MSE before clipping, without loss weighting or an optimizer step. These offline diagnostics isolate the return-regression objective; they do not replay the training-time GAE targets or optimizer updates.

For a response with T_i valid tokens, we sum its token-loss gradients and divide by $N = \sum_{j=1}^{512} T_j$, the total valid-token count across all 512 responses in the same rollout batch. The plotted quantity is the norm of this response contribution, $\|N^{-1} \sum_{t=1}^{T_i} \nabla_\phi \frac{1}{2}(V_\phi(s_t) - R_i)^2\|_2$. It is neither a projection onto the batch gradient nor a fraction of its norm: individual gradient vectors can cancel. Four FrontierCS responses at step 30 have no valid tokens and contribute zero; we retain them in the plots and group return statistics.

The horizontal axis is each prompt group's empirical return standard deviation, using population normalization. For the reweighted panels, we multiply each response gradient by $1/\max\{\hat{\sigma}(s), \varepsilon\}$, normalized to mean one across prompt groups within that checkpoint. This retains a comparable overall weight scale for the diagnostic; the batch-token denominator is unchanged. We use $\varepsilon = 0.075$ on FrontierCS and 0.5 in the AIME setting, whose returns are in $[0, 1]$ and $\{-1, +1\}$, respectively. The appendix scatter plots retain all responses without jitter or trimming; each line is an ordinary least-squares fit with an intercept. These fits summarize associations, rather than treating responses from the same prompt as independent evidence of causality. The box plots in Figure 3 summarize these same FrontierCS responses separately by checkpoint within six fixed return-STD intervals of width 0.05 spanning $[0, 0.30]$. The intervals are right-closed, with zero included in the

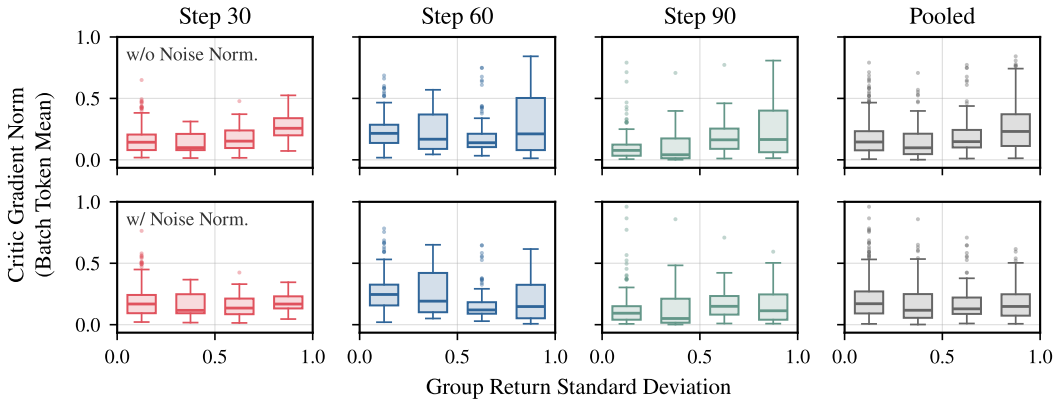

Figure C: Noise normalization reduces overall gradient imbalance in the AIME setting. Offline diagnostics with 512 responses per checkpoint (32 prompts, 16 responses each). **Columns:** steps 30, 60, 90, and their pooled responses. **Top:** without noise normalization. **Bottom:** with noise normalization. Boxes summarize per-response critic-gradient norms before clipping, grouped by prompt return standard deviation.

first, and contain 17, 9, 7, 7, 5, and 3 checkpoint-specific prompt groups, respectively. The first three columns correspond to steps 30, 60, and 90; the fourth pools all 1,536 responses. The top and bottom rows show gradients before and after noise normalization, respectively, using identical bins and axis limits. Boxes show medians and interquartile ranges; whiskers extend to the outermost observations within 1.5 interquartile ranges of the box, and all points beyond them are shown.

Figure C shows the same comparison in the AIME training setting. We divide return standard deviations into four equal-width bins over $[0, 1]$, containing 46, 11, 15, and 24 prompt groups across the three checkpoints. Every bin contains responses at each checkpoint. Finer bins would leave gaps because binary rewards allow only a discrete set of empirical return standard deviations. Pooling the checkpoints, gradient norms depend less on return variability after normalization, though the effect varies across checkpoints.

Figure D shows the corresponding per-response scatter plots. The positive gradient–return-variability trend becomes weaker at each checkpoint, while large individual contributions remain. Figure E gives the corresponding binary-reward comparison. Its trends are less uniform: reweighting does not flatten every checkpoint's fit or consistently reduce the largest contribution. Thus, the diagnostic supports reducing return-scale imbalance, not eliminating all sources of gradient variation.

C.5 Overlong Filtering on AIME

Figure F extends the filtering comparison in Section 3 to the AIME setting. The three runs share Qwen3.5-9B-Base initialization, rollout batches of 512, group size 16, an 8192-token response limit, and a 30-step critic warmup. All use one unweighted MSE critic mini-batch with learning rate 2×10^{-6}; only the filtering strategy differs in the recorded training settings. Joint filtering leads to increasing truncation and declining overall training and validation scores. Actor-only filtering initially tracks the gains of no filtering but later collapses, consistent with filtering alone being insufficient for stability.

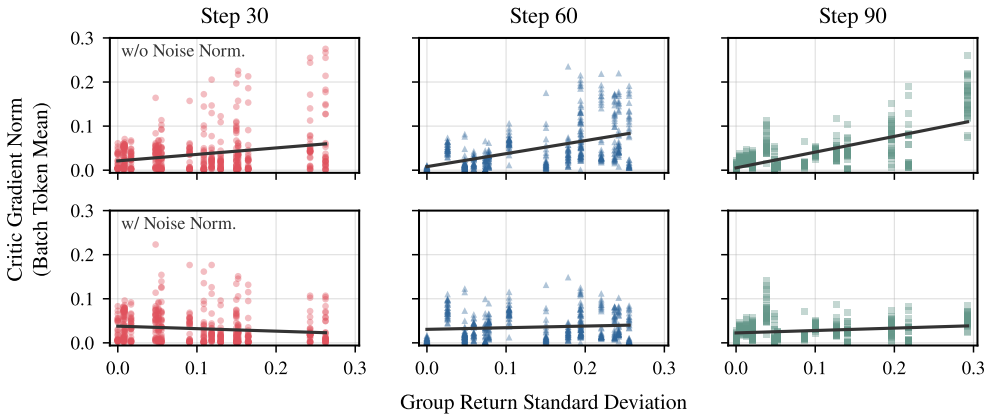

Figure D: The FrontierCS gradient-scale trend weakens after noise normalization at each checkpoint. Columns: steps 30, 60, and 90. **Top/bottom:** the same gradients before/after noise normalization. Each panel shows 512 responses; lines are descriptive linear fits.

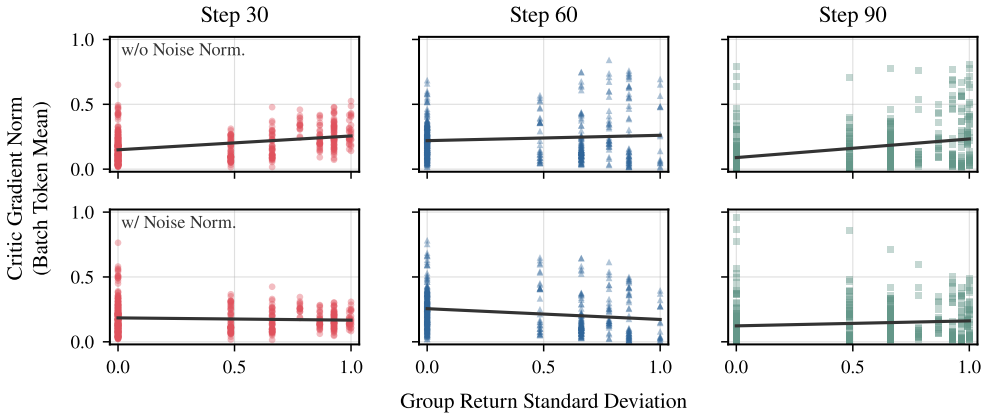

Figure E: Response-level gradient variability remains in the AIME training setting. Columns: steps 30, 60, and 90. **Top/bottom:** the same gradients before/after noise normalization. Each panel shows 512 responses; lines are descriptive linear fits.

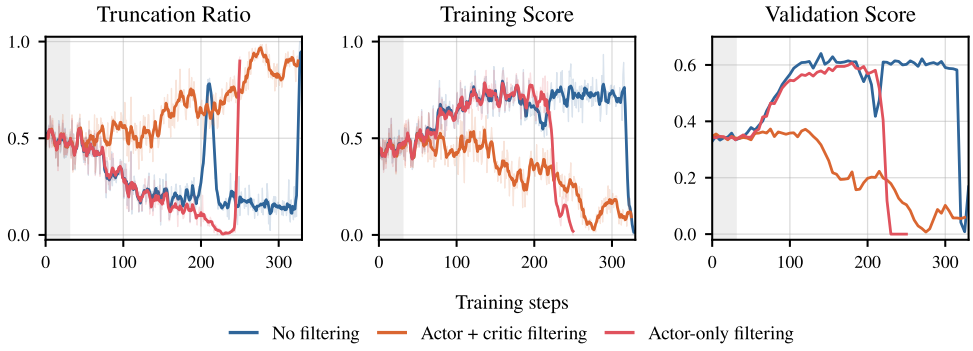

Figure F: Joint filtering exhibits increasing truncation and declining performance on AIME. Left: truncation ratio. **Middle:** overall training score. **Right:** AIME24 validation score. Gray marks critic warmup. Training statistics use a five-point moving average with faint raw values; validation is unsmoothed.

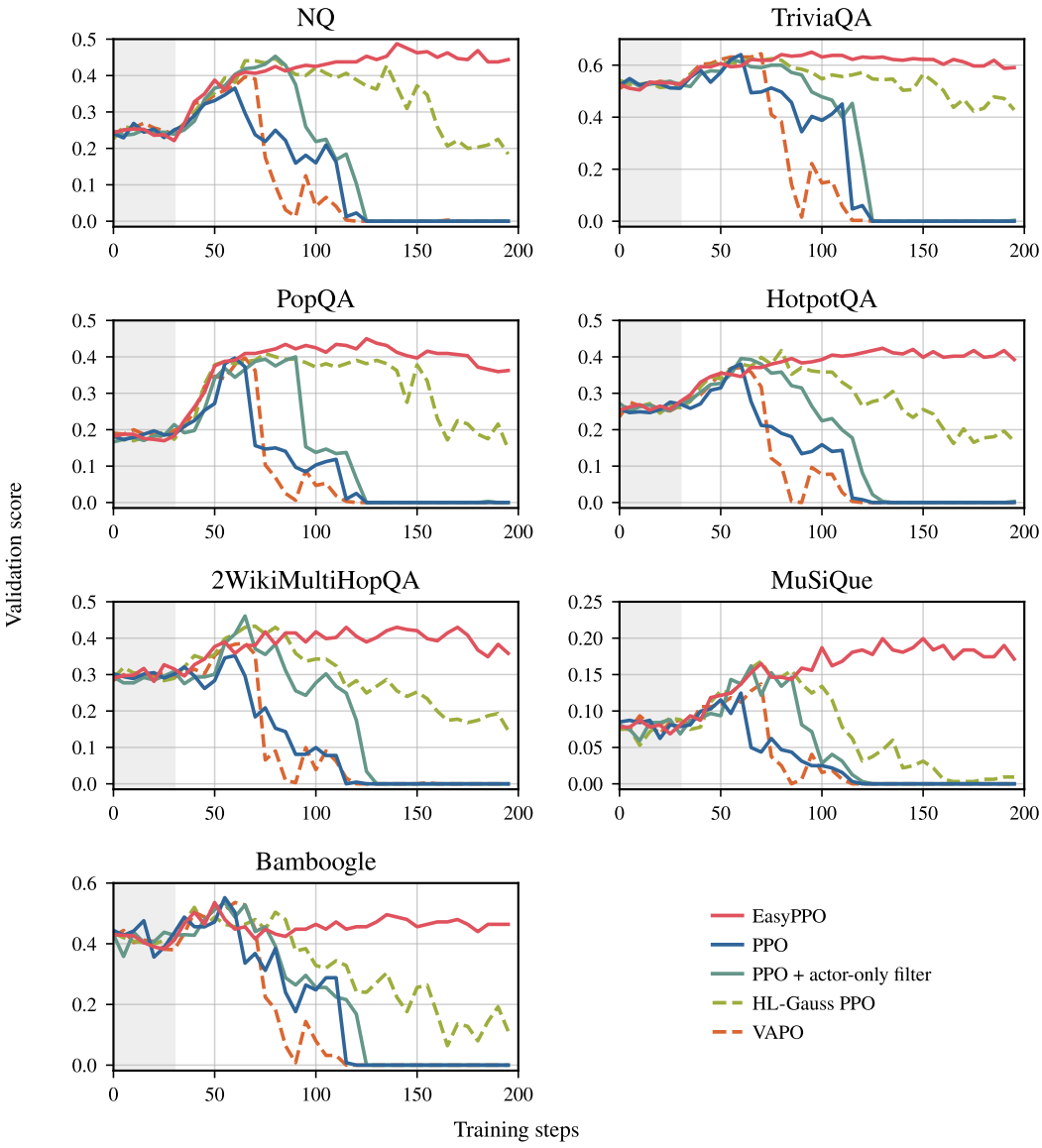

Figure G: EasyPPO sustains performance across all seven Search-R1 validation sets. Each panel shows one dataset from the aggregate comparison in Figure 4. Gray marks the 30-step critic warmup.

C.6 Search-R1 Results on Individual Validation Sets

Figure G breaks down the seven-dataset Search-R1 average in Figure 4 to examine whether the stability advantage holds across individual datasets. We use the same five training runs and report unsmoothed validation accuracies under greedy decoding, evaluated every five training steps.

PPO, VAPO, and PPO + actor-only filtering initially improve but later fall to nearly zero on all seven datasets. HL-Gauss PPO also loses substantial performance, particularly on MuSiQue and Bamboogle. EasyPPO avoids these collapses and achieves the highest final validation score on every dataset.