VIMPO: VALUE-IMPLICIT POLICY OPTIMIZATION FOR LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards has become a central tool for improving the reasoning ability of large language models, but current methods face a trade-off between simplicity and credit assignment. Group-relative methods such as GRPO avoid training a critic, but typically assign a trajectory-level advantage to every token. Actor-critic methods provide denser learning signals, but require a learned value function with its own training instability. We introduce VIMPO, a critic-free policy optimization method that derives a policy-implied value function from the optimality conditions of KL-regularized reinforcement learning. For autoregressive generation, the resulting value recurrence can be written in terms of policy-reference log-ratios and anchored by the terminal condition that no future reward remains at the end of a trajectory. This gives a simple value loss that incorporates outcome-level verifiable rewards without training a critic. The value loss directly updates the policy using outcome supervision, while an auxiliary PPO-style actor reuses a token-dependent signal derived from the learned policy-reference log-ratios. In the reported mathematical RLVR runs, VIMPO achieves higher scores than GRPO on MATH-500, AIME 2024, AIME 2025, and OlympiadBench, including under the tested reward-flipping condition. Our ablations show that the actor speeds up language-model training. Beyond language modeling, the value loss alone learns effective control policies on a terminal-reward variant of Walker2d, achieving higher mean displacement than PPO and GRPO. Together, these results highlight outcome-supervised policy regression and the reuse of policy-implied advantages as complementary components of critic-free policy optimization.

1 INTRODUCTION

Reinforcement learning has emerged as the central post-training paradigm for large language models (LLMs), particularly for tasks demanding multi-step reasoning such as mathematical problem solving and code generation (Ouyang et al., 2022; DeepSeek-AI et al., 2025; Jaech et al., 2024). Existing RL post-training methods for LLMs largely divide into actor-critic and critic-free approaches. Actor-critic methods such as PPO (Schulman et al., 2017) and VAPO (Yue et al., 2025) retain a learned value function, enabling dense token-level supervision but making training sensitive to critic quality and critic-policy co-adaptation (Yuan et al., 2025). In contrast, group-relative methods such as GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025) eliminate the critic and estimate advantages from groups of rollouts for the same prompt. This simplifies training by replacing the learned state-dependent baseline with a sample-based group baseline, at the cost of broadcasting the same advantage across all tokens. The result is a practical tension between the stability and simplicity of critic-free methods and the dense credit assignment provided by actor-critic methods.

This tension is especially important for long-form reasoning. Group-relative methods are attractive because they avoid training a value network that must remain calibrated to a continually evolving policy. Yet this simplicity comes with a coarse credit assignment signal. A single trajectory-level advantage is typically broadcast across all tokens, treating a completion as an indivisible unit rather than separating pivotal reasoning steps from routine connective tokens. Recent work attempts to recover token-level granularity *post hoc*. FIPO (Ma et al., 2026) re-weights the GRPO advantage using a future-KL factor, while attention-based credit assignment methods (Li et al., 2025) identify "anchor" tokens heuristically. These remedies are useful, but they remain add-ons to objectives that were not designed for fine-grained credit assignment.

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.

We investigate whether policy-derived signals can provide token-dependent updates without a separate critic. We model autoregressive generation as a deterministic-transition MDP and analyze its KL-regularized policy optimization objective. Under the stated local optimality condition, this yields an identity representing the corresponding value function through policy log-likelihood ratios against a frozen reference model. Instead of learning a separate critic, the resulting method estimates the initial value from the same Monte Carlo rollout groups used in Reinforcement Learning with Verifiable Rewards (RLVR) and anchors the recurrence with the terminal condition that the value at the end of a trajectory is zero. The terminal condition closes the Bellman recurrence, and the Monte Carlo estimate provides the trajectory-level reward target. This motivates a critic-free regression objective that encourages the trajectory-level policy-implied prediction to match centered outcome rewards.

The same identity motivates a policy-implied one-step advantage estimate for a PPO-style actor update. The resulting algorithm, **VIMPO** (**V**alue-**IM**plicit **P**olicy **O**ptimization), combines outcome-supervised policy regression with a token-dependent actor signal motivated by the Bellman identity as shown in Figure 1. VIMPO therefore occupies a middle ground between group-relative methods and actor-critic methods. It keeps RLVR-style group training while providing token-dependent updates without a separate critic. The value loss directly updates the policy using outcome supervision, and the auxiliary actor reuses the learned policy-reference log-ratios. In the reported runs, VIMPO improves accuracy per training step and achieves higher scores than GRPO under the tested reward-flipping condition.

VIMPO is motivated by critic-free credit assignment, but our experiments also reveal that the value loss can train effective policies on its own. We also study a terminal-reward variant of Walker2d, a continuous-control locomotion task, using MLP policies trained from scratch. This setting allows repeated runs and matched ablations to study how outcome regression trains the policy. To examine how the value loss improves the policy, we decompose its gradient into a return-weighted update and a correction based on the current prediction. Matched ablations on Walker2d then test the contribution of this correction.

Our contributions are fourfold.

- We derive a closed-form representation of the optimal value function for a KL-regularized policy on the deterministic-transition MDP induced by autoregressive generation. The representation is expressed through policy log-ratios against a frozen reference model, together with a Monte Carlo estimate of the initial value (Section 3.2).
- We use the terminal boundary condition $V^*(\boldsymbol{s}_T) = 0$ to obtain a critic-free value optimization objective. The resulting objective directly trains the policy without a separate critic. We decompose its gradient into a return-dependent component and a prediction-dependent correction, providing a basis for component analysis (Section 3.3; Appendix D).
- We derive a log-ratio representation of the one-step advantage under the stated optimality condition and use it to define a policy-implied actor signal during training. This estimate can be integrated into PPO-style updates with GAE-style aggregation without a separate critic. (Section 3.4).
- We evaluate VIMPO on mathematical reasoning benchmarks and use terminal-reward Walker2d for controlled component comparisons. VIMPO outperforms GRPO in the language-model experiments and both GRPO and PPO on Walker2d. Our ablations further demonstrate the effectiveness of outcome regression for policy learning and the additional benefit of the implied actor in language-model training (Section 4).

2 Related Work

PPO (Schulman et al., 2017) and GAE (Schulman et al., 2016) use learned values for credit assignment; VAPO (Yue et al., 2025) adapts this approach to long-form LLM reasoning. GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025) avoid a separate critic but derive advantages from trajectory outcomes. FIPO (Ma et al., 2026) and attention-based methods (Li et al., 2025) refine these signals through token weighting. VIMPO instead derives its actor signal from a policy-implied value recurrence.

Policy-reference log-ratios also underlie DPO (Rafailov et al., 2023), IPO (Gheshlaghi Azar et al., 2024), and TDPO (Zeng et al., 2024), which study preference optimization. VIMPO applies this parameterization to online RL with rollout rewards. Its Bellman-based construction is related to Path

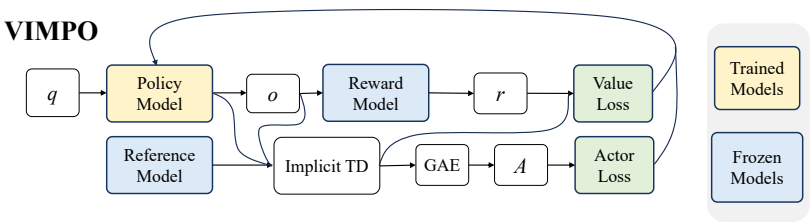


Figure 1: Overview of VIMPO. Given a prompt q, the policy generates a completion o, scored to obtain an outcome reward r. VIMPO uses this reward to train the policy-implied value loss, while the policy and frozen reference model define a token-level TD signal used to form the actor advantage. Both losses update the policy, with the value loss providing outcome supervision and the auxiliary actor reusing a policy-derived signal without introducing a separate critic.

Consistency Learning (Nachum et al., 2017) and soft Q-learning (Haarnoja et al., 2017), while using the standard expected-return value and an explicit reference-policy log-ratio. Appendix I provides a fuller comparison.

3 Methodology

We derive VIMPO in four steps. Section 3.1 formalizes autoregressive generation as a deterministic-transition MDP and states the KL-regularized policy optimization problem. Section 3.2 derives a value recurrence from the KL-regularized optimality condition and the Bellman equation. Section 3.3 converts this recurrence into a critic-free value optimization objective using a terminal boundary condition. Section 3.4 uses the resulting policy-implied value function to construct a critic-free PPO-style actor update.

3.1 Setup: autoregressive generation as a deterministic MDP

For a prompt $\boldsymbol{x} \in \mathcal{V}^{\ell_x}$, an autoregressive policy π generates a completion $a_0, a_1, \ldots, a_{T-1}$, where each $a_t \in \mathcal{V}$. We define the token-prefix state as

$$\boldsymbol{s}_t = (\boldsymbol{x}, a_0, \ldots, a_{t-1}) \in \mathcal{V}^{\ell_x + t}, \qquad t \in [0, T],$$

with $\boldsymbol{s}_0 = \boldsymbol{x}$ and the terminal state $\boldsymbol{s}_T$. Given the next token a_t, the transition is deterministic:

$$P(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, a_t) = 1.$$

For a fixed policy π, we use the standard finite-horizon expected-return action-value function:

$$Q^\pi(\boldsymbol{s}_t, a_t) = \mathbb{E}_\pi \left[\sum_{k=t}^{T-1} \gamma^{k-t} r(\boldsymbol{s}_k, a_k) \,\middle|\, \boldsymbol{s}_t, a_t \right],$$

and the corresponding value function:

$$V^\pi(\boldsymbol{s}_t) = \mathbb{E}_{a \sim \pi(\cdot|\boldsymbol{s}_t)} \left[Q^\pi(\boldsymbol{s}_t, a) \right].$$

Our value function is the standard expected-return value used in actor-critic RLHF, rather than the entropy-regularized soft value used in Path Consistency Learning and soft Q-learning (Nachum et al., 2017; Haarnoja et al., 2017). As a result, we can control how KL enters the value and actor objectives, and we avoid requiring estimates of future KL contributions when constructing the advantage. This separation is consistent with recent RLVR practice, where reducing or removing explicit KL penalties can improve reasoning performance in some settings (Yu et al., 2025).

At optimality, the KL-regularized policy satisfies the local optimality condition:

$$\pi^*(\cdot \mid \boldsymbol{s}_t) = \arg\max_\pi \; \mathbb{E}_{a \sim \pi(\cdot|\boldsymbol{s}_t)} \left[Q^*(\boldsymbol{s}_t, a) \right] - \beta \, \mathrm{KL}\left(\pi(\cdot \mid \boldsymbol{s}_t) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid \boldsymbol{s}_t) \right), \tag{1}$$

where Q^* is the optimal action-value function associated with the same fixed point. Solving the local maximization gives:

$$\pi^*(a \mid \boldsymbol{s}_t) = \frac{\pi_{\mathrm{ref}}(a \mid \boldsymbol{s}_t)}{Z(\boldsymbol{s}_t)} \exp\left(\frac{1}{\beta} Q^*(\boldsymbol{s}_t, a) \right). \tag{2}$$

3.2 FROM KL-REGULARIZED OPTIMALITY TO VALUE RECURRENCE

The deterministic transition collapses the Bellman equation to:

$$Q^*(\boldsymbol{s}_t, a_t) = r(\boldsymbol{s}_t, a_t) + \gamma V^*(\boldsymbol{s}_{t+1}). \tag{3}$$

Combining (3) with (2), and eliminating $\ln Z$ via the identity $\ln Z(\boldsymbol{s}_t) = \frac{1}{\beta} V^*(\boldsymbol{s}_t) - \mathrm{KL}^*(\boldsymbol{s}_t)$, where $\mathrm{KL}^*(\boldsymbol{s}_t) := \mathrm{KL}(\pi^*(\cdot \mid \boldsymbol{s}_t) \| \pi_{\mathrm{ref}}(\cdot \mid \boldsymbol{s}_t))$, yields the central identity of this paper:

$$\beta \ln \frac{\pi^*(a_t \mid \boldsymbol{s}_t)}{\pi_{\mathrm{ref}}(a_t \mid \boldsymbol{s}_t)} \;=\; r(\boldsymbol{s}_t, a_t) + \gamma V^*(\boldsymbol{s}_{t+1}) - V^*(\boldsymbol{s}_t) \;+\; \beta\, \mathrm{KL}^*(\boldsymbol{s}_t). \tag{4}$$

Equation (4) is the backbone of VIMPO. It states that, under optimality, the per-token log-ratio against the reference policy *is* a Bellman residual plus a KL correction. The following three observations immediately follow:

- **Closed-form value.** Rearranging (4) as $\gamma V^*(\boldsymbol{s}_{t+1}) - V^*(\boldsymbol{s}_t) = \beta \ln(\pi^*/\pi_{\mathrm{ref}}) - \beta\, \mathrm{KL}^* - r$ defines a one-step recurrence for V^*. Given any anchor value $V^*(\boldsymbol{s}_0)$, we can solve this recurrence forward without ever evaluating a learned critic.
- **Zero-mean signal.** The bracket $\beta \ln(\pi^*/\pi_{\mathrm{ref}}) - \beta\, \mathrm{KL}^*$ has expectation zero under $a_t \sim \pi^*(\cdot \mid \boldsymbol{s}_t)$, since KL^* is exactly the expectation of the log-ratio. This centering identity also holds for any policy under matching on-policy sampling with exact KL.
- **Closed-form one-step advantage.** The Bellman-residual form of (4) gives a closed-form one-step TD advantage $A_t^{\mathrm{TD}} = r + \gamma V^* - V^* = \beta \ln(\pi^*/\pi_{\mathrm{ref}}) - \beta\, \mathrm{KL}^*$, which we exploit in Section 3.4 for PPO integration.

3.3 TRAINING THE POLICY-IMPLIED VALUE

Let $V_t := V^*(\boldsymbol{s}_t)$ and $B_t := \beta \ln(\pi^*(a_t \mid \boldsymbol{s}_t)/\pi_{\mathrm{ref}}(a_t \mid \boldsymbol{s}_t)) - \beta\, \mathrm{KL}^*(\boldsymbol{s}_t) - r(\boldsymbol{s}_t, a_t)$. Equation (4) is equivalent to $B_t = \gamma V_{t+1} - V_t$, a first-order linear recurrence. Solving forward from V_0 yields

$$V^*(\boldsymbol{s}_t) = \gamma^{-t} V^*(\boldsymbol{s}_0) + \sum_{k=0}^{t-1} \gamma^{k-t} B_k. \tag{5}$$

Constructing the implied value. Equation (5) characterizes the optimal value function. To turn this into a training objective, we follow the DPO (Rafailov et al., 2023) strategy of substituting the trainable policy π for π^* in the closed-form expression. This defines the *policy-implied value*:

$$V_\pi(\boldsymbol{s}_t) := \gamma^{-t} \widehat{V}_0 + \sum_{k=0}^{t-1} \gamma^{k-t} \left[\beta \ln \frac{\pi(a_k \mid \boldsymbol{s}_k)}{\pi_{\mathrm{ref}}(a_k \mid \boldsymbol{s}_k)} - \beta\, \mathrm{KL}_\pi(\boldsymbol{s}_k) - r(\boldsymbol{s}_k, a_k) \right], \tag{6}$$

where $\widehat{V}_0$ is a Monte Carlo estimate of the initial value and $\mathrm{KL}_\pi(\boldsymbol{s}_k) := \mathrm{KL}(\pi(\cdot \mid \boldsymbol{s}_k) \| \pi_{\mathrm{ref}}(\cdot \mid \boldsymbol{s}_k))$.

The terminal anchor. The true expected future return at termination is zero. We therefore use zero as the supervision target for the policy-implied terminal value $V_\pi(\boldsymbol{s}_T)$, which need not satisfy this boundary condition during training. We train π to reduce this terminal residual:

$$\mathcal{L}_V(\pi) := \tfrac{1}{2}\, V_\pi(\boldsymbol{s}_T)^2. \tag{7}$$

Specialization to outcome-only rewards. For the common LLM RL setting where $r(\boldsymbol{s}_k, a_k) = 0$ for $k < T-1$ and $r(\boldsymbol{s}_{T-1}, a_{T-1}) = R_{\mathrm{final}}$, with $\gamma = 1$ and $\widehat{V}_0$ estimated by the group-mean return $\overline{R}_{\mathrm{final}}$, equation (7) unfolds into

$$\mathcal{L}_V(\pi) = \frac{1}{2} \left[\sum_{k=0}^{T-1} \left(\beta \ln \frac{\pi(a_k \mid \boldsymbol{s}_k)}{\pi_{\mathrm{ref}}(a_k \mid \boldsymbol{s}_k)} - \beta\, \mathrm{sg}[\mathrm{KL}_\pi(\boldsymbol{s}_k)] \right) - \left(R_{\mathrm{final}} - \overline{R}_{\mathrm{final}} \right) \right]^2 \tag{8}$$

where $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator. This is the operational form of the value-loss component of VIMPO. The target is centered by the group-mean reward, matching the normalization used in group-relative RLVR methods. We stop gradients through the KL term so that it acts only

as a centering baseline. Otherwise, the value loss could be reduced by directly manipulating the full-distribution KL term rather than by aligning sampled-token log-ratios with centered rollout rewards. Intuitively, the loss encourages the cumulative policy-reference log-ratio along a trajectory to track the centered final reward, tying the policy-implied value function to rollout outcomes without introducing a learned critic.

How outcome regression updates the policy. Let $y = R_{\text{final}} - \overline{R}_{\text{final}}$, let p_θ denote the summed log-ratio prediction inside Equation 8, and define $S_\theta = \sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)$. With the target and KL terms detached, the per-trajectory regression gradient decomposes as

$$\nabla_\theta \mathcal{L}_V = \beta(p_\theta - y) S_\theta = \underbrace{-\beta y S_\theta}_{\text{return-dependent component}} + \underbrace{\beta p_\theta S_\theta}_{\text{prediction-dependent correction}}. \tag{9}$$

The return-dependent term reinforces above-group outcomes, while the correction adjusts the update using the current prediction. Removing the correction defines our return-only ablation. Appendix D derives the decomposition and its minibatch form.

3.4 PPO INTEGRATION VIA THE CLOSED-FORM ADVANTAGE

The same identity also induces a one-step advantage estimator. For a trainable policy π_θ, we define the policy-implied TD advantage

$$\widehat{A}_t^{\text{TD}} := \beta \log \frac{\pi_\theta(a_t \mid \boldsymbol{s}_t)}{\pi_{\text{ref}}(a_t \mid \boldsymbol{s}_t)} - \beta \, \mathrm{KL}_{\pi_\theta}(\boldsymbol{s}_t). \tag{10}$$

This quantity corresponds to the Bellman residual $r(\boldsymbol{s}_t, a_t) + \gamma V_{\pi_\theta}(\boldsymbol{s}_{t+1}) - V_{\pi_\theta}(\boldsymbol{s}_t)$ when the policy-implied value satisfies the recurrence in Equation 6. Motivated by this identity, we use $\widehat{A}_t^{\text{TD}}$ as a policy-implied advantage estimate for the PPO-style actor update.

The one-step estimates can be accumulated in the same form as GAE:

$$\widehat{A}_t^{\lambda} = \sum_{\ell=0}^{T-t-1} (\gamma\lambda)^\ell \widehat{A}_{t+\ell}^{\text{TD}}. \tag{11}$$

As is common in PPO implementations, before applying the PPO surrogate we normalize the detached advantages over valid response tokens in each optimization step:

$$\widetilde{A}_t^{\lambda} = \frac{\widehat{A}_t^{\lambda} - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}} + \epsilon}, \tag{12}$$

where $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}$ are computed over valid response tokens in the minibatch. We then use $\widetilde{A}_t^{\lambda}$ in the PPO actor loss $\mathcal{L}_A$, with no gradient propagated through the advantage normalization.

The external reward enters VIMPO through the value-loss term, not directly through the actor advantage in Equation 10. The value loss directly updates the policy using outcome supervision and encourages its cumulative log-ratio signal to predict centered rewards. The auxiliary actor reuses the resulting policy-derived signal for additional policy updates.

Formally, let $\mathcal{L}_A$ denote the negative clipped PPO surrogate, and $c_A \geq 0$ control the actor update strength. The overall VIMPO training objective is then:

$$\mathcal{L}_{\text{VIMPO}}(\pi) = \mathcal{L}_V(\pi) + c_A \, \mathcal{L}_A(\pi). \tag{13}$$

In implementation, each training step samples groups of rollouts for prompts in the batch, computes the centered final reward, evaluates the policy-reference log-ratio terms along each trajectory, forms the value loss in Equation 8, and applies the PPO actor loss in Equation 13. Appendix Algorithm 1 summarizes the resulting training step.

4 EXPERIMENTS

We evaluate VIMPO against GRPO in an RLVR setting for mathematical reasoning. We first train both methods on a math-reasoning dataset for an extended number of steps and compare their validation performance and learning dynamics. We then stress-test both methods under noisy rewards to evaluate robustness to imperfect reward signals. Finally, we conduct ablations over key hyperparameters and monitor additional training metrics to better understand the behavior of VIMPO. We complement these LLM experiments with terminal-reward Walker2d, using repeated runs and matched component comparisons to examine outcome regression and its prediction-dependent correction.

4.1 EXPERIMENT SETUP

LLM experiments. We train Qwen3-4B-Base (Yang et al., 2025) on Guru Math (Cheng et al., 2025) using verl (Sheng et al., 2024), and evaluate on MATH-500, AIME 2024, AIME 2025, and OlympiadBench. VIMPO and GRPO share optimizer settings; VIMPO uses $\beta = 5 \times 10^{-4}$ and $c_A = 5 \times 10^{-3}$, selected by coarse tuning and held fixed. The primary GRPO baseline uses token-level loss aggregation without a KL reward penalty; naive GRPO uses sequence-mean-token-mean aggregation. Neither includes additional DAPO-style sampling or filtering. For the noisy-reward test, we independently flip binary training rewards with probability 25% and evaluate with the clean verifier. Appendix J gives the benchmark citations, evaluation pipelines, and baseline details; Appendix A.1 lists shared hyperparameters.

Continuous control. We adapt Gymnasium's Walker2d-v5 environment (Towers et al., 2024), simulated with MuJoCo (Todorov et al., 2012), to use terminal forward displacement as its sole reward, retaining termination on a fall and a 1,000-step horizon. All methods use Gaussian MLP policies trained from scratch and observe the physical state, elapsed time, and accumulated displacement. We compare full and value-only VIMPO with GRPO, PPO, and matched component controls, training for 16.384M transitions across ten seeds and evaluating each final policy on 256 sampled episodes. Section 4.5 presents the results, whereas Appendix E details the configurations and evaluation procedure.

4.2 MAIN COMPARISON

We first compare VIMPO and the two GRPO baselines under clean golden rewards. Figure 2 reports training accuracy, response length, MATH-500 accuracy, and AIME 2024 avg@32 accuracy during RL fine-tuning. All RL methods improve substantially over the base model, but VIMPO shows stronger learning dynamics than both GRPO baselines in the later stage of training and achieves the highest validation accuracy on both MATH-500 and AIME 2024.

Table 1 gives the final clean-reward evaluation results. All RL methods improve substantially over Qwen3-4B-Base, confirming that the training setup provides a strong RLVR signal. The token-level GRPO baseline outperforms naive GRPO on the overall average, consistent with prior findings that token-level loss aggregation is a stronger implementation choice for GRPO (Yu et al., 2025). VIMPO further improves over both GRPO variants, achieving the best average performance and the highest score on each benchmark. The improvement over GRPO is especially pronounced on AIME 2025, where VIMPO improves from 17.6 to 20.8, suggesting that the policy-implied value loss and actor update are most helpful on harder competition-style reasoning evaluations.

These results suggest that the policy-implied value loss and critic-free actor update provide a useful learning signal beyond the group-relative baseline. The response-length curve in Figure 2 also indicates that VIMPO's gains are not simply explained by monotonically increasing completion length. VIMPO explores longer responses early in training, but its response length later decreases while validation accuracy remains higher than GRPO.

4.3 ROBUSTNESS TO NOISY REWARDS

Appendix Figure 11 shows the noisy-reward stress test described in Section 4.1. We plot the first 200 training steps under 25% reward flipping, with clean-reward runs shown as faint references.

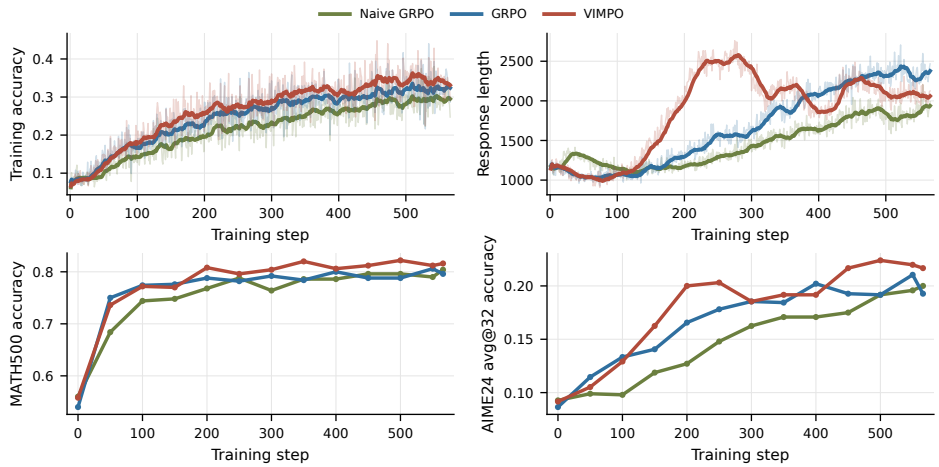


Figure 2: Main comparison among naive GRPO, GRPO and VIMPO under clean verifier rewards. We report training accuracy, response length, MATH-500 accuracy, and AIME 2024 avg@32 accuracy over training. Faint curves show raw logged values, while solid curves show a 15-step moving average for visualization.

Table 1: Final evaluation accuracy. Clean rows are evaluated after full training, while noisy rows are evaluated after 200 training steps with 25% reward flipping. Avg. is the unweighted average over the four benchmark columns.

Method	**MATH-500 avg@1**	**AIME24 avg@32**	**AIME25 avg@32**	**OlympiadBench avg@1**	**Avg.**
Qwen3-4B-Base	54.0	8.6	3.6	18.0	21.1
Naive GRPO	80.4	20.0	14.6	31.3	36.6
GRPO	79.6	19.3	17.6	33.2	37.4
VIMPO	**81.6**	**21.7**	**20.8**	**34.0**	**39.5**
GRPO noisy	75.2	13.0	11.7	28.9	32.2
VIMPO noisy	**78.2**	**18.3**	**14.9**	**32.3**	**35.9**

The lower two rows of Table 1 report results with 25% reward flipping. VIMPO outperforms GRPO on all four benchmarks, retaining its advantage despite the corrupted training rewards.

Appendix Figure 11 shows the corresponding learning curves, with clean-reward runs included for reference and all accuracy metrics measured using the clean verifier. Under noisy rewards, VIMPO learns faster and maintains higher AIME 2024 validation accuracy than GRPO. One possible explanation is that VIMPO incorporates reward information through the value loss and reuses it through the policy-implied actor advantage. This reliance on a learned signal may make actor updates less sensitive to individual reward errors.

4.4 Ablations

We ablate the two VIMPO-specific coefficients, the value coefficient β and the actor coefficient c_A. Figure 3 shows training accuracy and VO KL over the first 200 training steps for three representative settings. The value-only variant with $c_A = 0$ improves training accuracy throughout training. Adding the PPO-style actor update with $c_A = 0.005$ speeds up learning and achieves the highest training accuracy among the tested settings.

The ablation also shows that this gain comes with a larger policy shift. The main VIMPO setting reaches a much higher VO KL than the value-only variant, indicating that the actor update enables faster movement away from the reference policy. In contrast, the high-β, high-c_A setting keeps VO KL nearly flat after the early stage, but its accuracy remains close to the value-only baseline. This

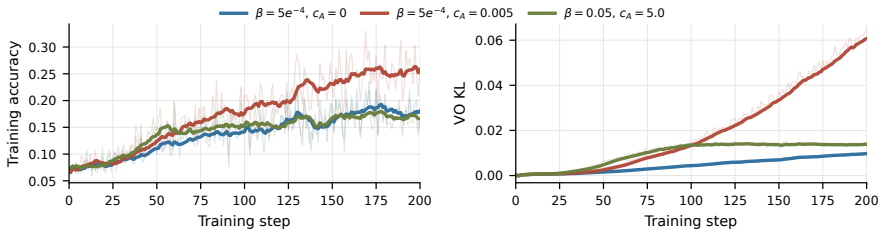


Figure 3: Ablation over VIMPO coefficients. We compare a value-only variant with $c_A = 0$, the main VIMPO setting, and a high-β, high-c_A setting. The actor update improves training accuracy but also increases VO KL, while the high-β, high-c_A setting keeps KL controlled at the cost of slower learning.

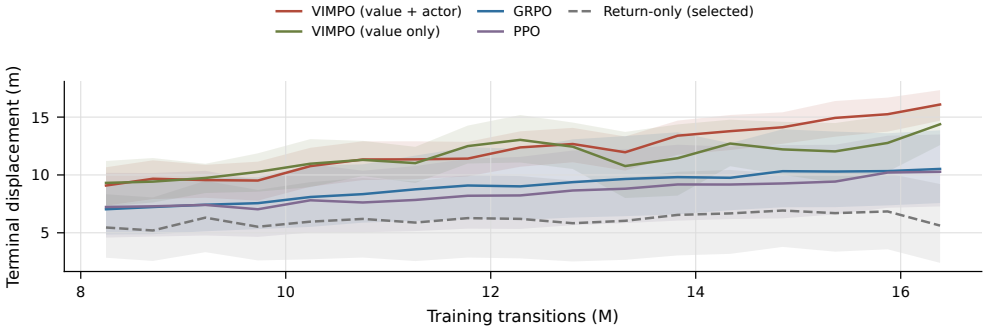


Figure 4: Walker2d learning with sampled actions over the common 8.2M–16.4M continuation window. Lines show means across ten training seeds. Shaded bands are pointwise descriptive 95% seed-bootstrap intervals. Curve evaluations use 32 episodes per seed. Each method uses its separately selected optimizer configuration (Appendix E).

suggests that overly strong value scaling and actor weighting can over-constrain the effective update and reduce the benefit of the actor component.

Interestingly, the high-β, high-c_A setting improves quickly in the first 50 steps before plateauing. This suggests that stronger early regularization or actor weighting may help initial optimization, but can become limiting later in training. A natural extension is to adapt or anneal β and c_A over training, using larger values early for stabilization and smaller values later to permit further policy improvement.

Overall, the ablation supports the design of VIMPO as a combination of the value loss and actor update. The value loss alone provides a usable learning signal, while adding the actor improves training accuracy per step in the reported configuration. At the same time, the coefficients must be chosen carefully, since stronger policy updates can affect response-length dynamics, which we report in Appendix B.2.

4.5 Continuous control with terminal rewards

We use continuous control to study VIMPO in a setting that supports repeated runs and controlled component comparisons. This lets us examine the contribution of outcome regression and its prediction-dependent correction to policy learning from initialization. The task and training protocol are described in Section 4.1.

Figure 4 shows continued improvement for both VIMPO variants over the displayed training window. In the final 256-episode evaluation, VIMPO with the actor reaches a mean displacement of 16.18 m, compared with 10.56 m for GRPO and 10.40 m for PPO. Value-only VIMPO also learns an effective policy, reaching 14.64 m, versus 5.63 m for separately selected return-only training. Under matched optimizer settings, retaining the prediction-dependent correction in Equation 9 improves

displacement by 14.79 m. Seed-level endpoint results and sustained-progress rates are shown in Appendix Figure 9. Thus, beyond providing a policy-implied actor signal, the regression objective itself contributes substantially to policy learning in this setting.

5 Discussion and Limitations

Outcome regression directly trains the policy from rewards: value-only VIMPO learns effective Walker2d policies, while the auxiliary actor accelerates our LLM training. The current implementation uses a frozen reference and fixed coefficients, and its regression updates do not account explicitly for rollout reuse through behavior-policy importance weighting or clipping. Exact KL computation also requires full policy and reference distributions. Broader evaluation across model scales, tasks, and LLM training seeds would extend the present evidence. Appendix H discusses these directions in detail.

6 Conclusion

We introduced VIMPO, a critic-free policy optimization method for RLVR that derives value and actor signals from the optimality conditions of KL-regularized reinforcement learning. By modeling autoregressive generation as a deterministic-transition process, VIMPO expresses the value recurrence in terms of policy-reference log-ratios and uses a terminal boundary condition to train a policy-implied value function without a learned critic. The value loss directly trains the policy from outcome rewards, and the auxiliary PPO-style actor reuses a policy-implied advantage estimate.

In the reported mathematical reasoning experiments, VIMPO improves over GRPO, including under the tested reward-flipping condition. These results motivate further study of outcome-supervised policy regression and the reuse of policy-derived learning signals. The Walker2d extension further shows that outcome regression can train effective continuous-control policies and that its prediction-dependent correction improves performance over return-only training. Future work should study how to account for rollout reuse in regression updates, and how adaptive coefficients, reference updates, and approximate KL computation affect performance.

AI use statement

Generative AI tools were used to enhance the clarity and presentation of the manuscript and to support the development and refinement of visualizations. The tools were used solely in an assistive capacity. The authors retained full control over all final decisions and assume complete responsibility for the paper's content, claims, and results.

References

Zhoujun Cheng, Shibo Hao, Tianyang Liu, Fan Zhou, Yutao Xie, Feng Yao, Yuexin Bian, Yonghao Zhuang, Nilabjo Dey, Yuheng Zha, Yi Gu, Kun Zhou, Yuqi Wang, Yuan Li, Richard Fan, Jianshu She, Chengqian Gao, Abulhair Saparov, Haonan Li, Taylor W. Killian, Mikhail Yurochkin, Zhengzhong Liu, Eric P. Xing, and Zhiting Hu. Revisiting reinforcement learning for llm reasoning from a cross-domain perspective, 2025. URL https://arxiv.org/abs/2506.14965.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Rémi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from

human preferences. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 4447–4455. PMLR, 2024.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1352–1361. PMLR, 2017.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Hugging Face. Lighteval: A lightweight toolkit for evaluating large language models. `https://github.com/huggingface/lighteval`, 2025. Software package.

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

Yang Li, Zhichen Dong, Yuhan Sun, Weixun Wang, Shaopan Xiong, Yijia Luo, Jiashun Liu, Han Lu, Jiamang Wang, Wenbo Su, Bo Zheng, and Junchi Yan. Attention illuminates llm reasoning: The preplan-and-anchor rhythm enables fine-grained policy optimization. *arXiv preprint arXiv:2510.13554*, 2025.

Chiyu Ma, Shuo Yang, Kexin Huang, Jinda Lu, Haoming Meng, Shangshang Wang, Bolin Ding, Soroush Vosoughi, Guoyin Wang, and Jingren Zhou. Fipo: Eliciting deep reasoning with future-kl influenced policy optimization. *arXiv preprint arXiv:2603.19835*, 2026.

Mathematical Association of America. American invitational mathematics examination 2024. `https://maa.org/maa-invitational-competitions/`, 2024. MAA Invitational Competitions.

Mathematical Association of America. American invitational mathematics examination 2025. `https://maa.org/maa-invitational-competitions/`, 2025. MAA Invitational Competitions.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 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. *arXiv preprint arXiv:2409.19256*, 2024.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 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, Xiangpeng Wei, Gaohong Liu, Juncai Liu, Lingjun Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Chi Zhang, Mofan Zhang, Wang Zhang, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.

Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. Token-level direct preference optimization. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 2024.

A Experimental Details

A.1 Shared hyperparameters

Table 2 lists the hyperparameters shared by GRPO and VIMPO in the main experiments. Method-specific VIMPO coefficients are reported in Section 4.1.

Table 2: Shared experimental hyperparameters for GRPO and VIMPO.

Hyperparameter	**Value**
Base model	Qwen3-4B-Base
Training data	Guru Math subset, 54.4K examples
Max prompt length	2048
Max response length	4096
Responses per prompt	8
Prompt batch size	96
Generated responses per batch	768
Rollout temperature	1.0
Rollout top-p	1.0
Validation temperature	1.0
Validation top-p	0.7
Mini-batch size	96
Actor learning rate	1×10^{-6}
Warmup steps	10
Weight decay	0.1
Gradient clipping	1.0

A.2 GRPO baseline loss aggregation

For the GRPO baselines, we keep the advantage estimator group-relative and consider two loss aggregations. The main GRPO baseline uses token-level loss aggregation following the DAPO training recipe (Yu et al., 2025). We also include a naive GRPO baseline using the original sequence-mean-token-mean aggregation (Shao et al., 2024). Both are different from the VIMPO advantage in Equation 10. For a prompt $\boldsymbol{x}$, we sample a group of completions $\{\tau^{(i)}\}_{i=1}^{G}$ from $\pi_{\theta_{\text{old}}}$, where $\tau^{(i)} = (a_0^{(i)}, \ldots, a_{T_i-1}^{(i)})$. Let $R^{(i)}$ be the final reward for completion $\tau^{(i)}$. The group-normalized GRPO advantage is

$$\widehat{A}_i^{\text{GRPO}} = \frac{R^{(i)} - \overline{R}}{\text{std}\big(\{R^{(j)}\}_{j=1}^{G}\big) + \epsilon}, \qquad \overline{R} = \frac{1}{G}\sum_{j=1}^{G} R^{(j)}. \tag{14}$$

The same scalar $\widehat{A}_i^{\text{GRPO}}$ is assigned to every valid token in completion $\tau^{(i)}$.

For each token, define the PPO probability ratio and clipped token objective as

$$r_t^{(i)}(\theta) = \frac{\pi_\theta(a_t^{(i)} \mid \boldsymbol{s}_t^{(i)})}{\pi_{\theta_{\text{old}}}(a_t^{(i)} \mid \boldsymbol{s}_t^{(i)})}, \tag{15}$$

$$\ell_t^{(i)}(\theta) = \min\left(r_t^{(i)}(\theta)\widehat{A}_i^{\text{GRPO}},\ \text{clip}\Big(r_t^{(i)}(\theta), 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}}\Big)\,\widehat{A}_i^{\text{GRPO}} \right). \tag{16}$$

The token-mean clipped GRPO objective used for the main GRPO baseline is

$$\mathcal{J}_{\text{GRPO}} = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\mathbb{E}_{\{\tau^{(i)}\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|\boldsymbol{x})}\left[\frac{1}{\sum_{i=1}^{G} T_i}\sum_{i=1}^{G}\sum_{t=0}^{T_i-1}\ell_t^{(i)}(\theta)\right]. \tag{17}$$

For the naive GRPO baseline, we use the sequence-mean-token-mean aggregation:

$$\mathcal{J}_{\text{GRPO-seq}} = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\mathbb{E}_{\{\tau^{(i)}\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|\boldsymbol{x})}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=0}^{T_i-1}\ell_t^{(i)}(\theta)\right]. \tag{18}$$

In implementation, both objectives are minimized by negating the corresponding surrogate objective. The main GRPO baseline and VIMPO both use token-level loss aggregation, but they differ in the advantage signal. GRPO broadcasts the group-relative trajectory advantage $\widehat{A}_i^{\mathrm{GRPO}}$ to all tokens in a completion, while VIMPO uses the policy-implied token-level advantage in Equation 10. The naive GRPO baseline uses the same group-relative advantage as GRPO, but averages the token loss within each completion before averaging across completions.

B Additional Experimental Results

B.1 Training entropy

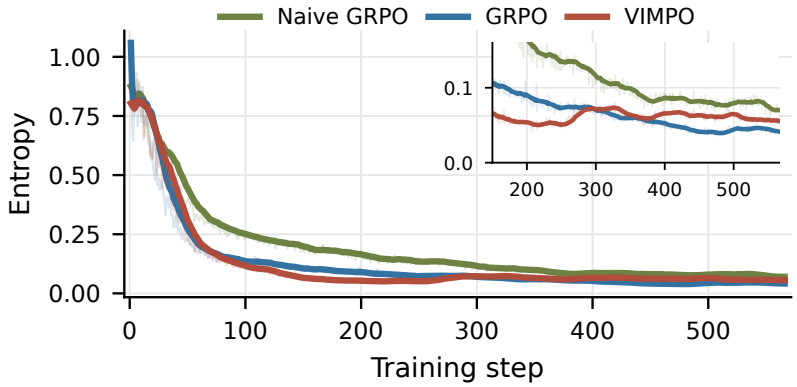


Figure 5: Training entropy of naive GRPO, GRPO, and VIMPO under clean verifier rewards. Faint curves show raw logged values, while solid curves show a moving average. Both GRPO baselines exhibit a mostly monotonic decrease in entropy, whereas VIMPO shows small fluctuations after entering the low-entropy regime.

Figure 5 reports the average token-level policy entropy during training. For a generated trajectory $\tau = (a_0, \ldots, a_{T-1})$, we compute

$$\mathcal{H}(\pi_\theta) = \frac{1}{T} \sum_{t=0}^{T-1} \left[- \sum_{a_t \in \mathcal{V}} \pi_\theta(a_t \mid \boldsymbol{s}_t) \log \pi_\theta(a_t \mid \boldsymbol{s}_t) \right], \tag{19}$$

and average this quantity over sampled completions.

The two GRPO baselines show a largely monotonic entropy decrease throughout training. Naive GRPO maintains higher entropy than token-level GRPO for much of training, but both baselines continue moving toward lower-entropy policies. VIMPO also rapidly enters a low-entropy regime, but its entropy does not decrease as monotonically and shows small recoveries in the later stage of training. This is consistent with the response-length dynamics in Figure 2, where VIMPO changes its generation behavior during training rather than simply increasing confidence monotonically.

These results provide an additional diagnostic difference between VIMPO and group-relative baselines. The monotonic entropy decrease observed for both GRPO variants is consistent with the entropy-collapse behavior discussed by Cui et al. (2025), and may be connected to the uniform token-level advantage assignment used by group-relative methods. In contrast, VIMPO uses a policy-implied token-level advantage, which may allow more local adjustment of the policy distribution after entering the low-entropy regime. We view Figure 5 as suggestive evidence that VIMPO is less tightly coupled to monotonic entropy collapse, while leaving a controlled study of this mechanism to future work.

B.2 Response length in the coefficient ablation

Figure 6 reports the response-length dynamics for the coefficient ablation in Section 4.4. The main VIMPO setting, which uses both the value loss and actor update, learns fastest in training accuracy

but also produces longer responses by step 200. This supports the observation that VIMPO-specific coefficients affect not only optimization speed and policy movement, but also generation-length dynamics.

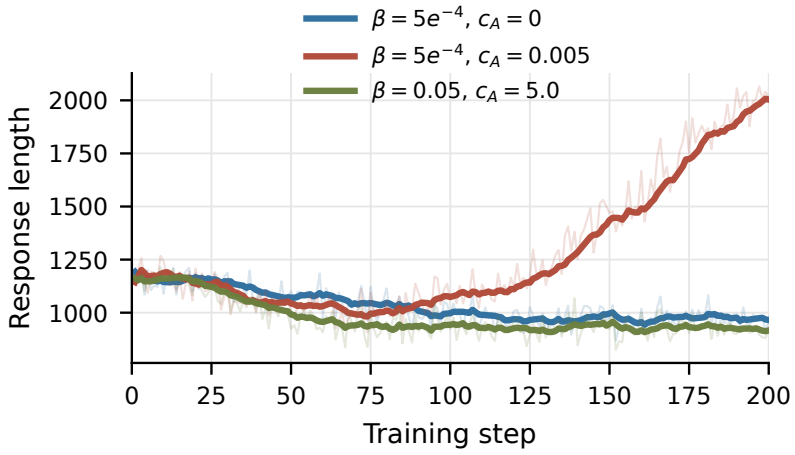


Figure 6: Response-length dynamics for the VIMPO coefficient ablation. The main VIMPO setting improves training accuracy fastest, but also produces longer responses by step 200.

C Qualitative Token-Level Signal Visualization

We provide a qualitative case study illustrating the token-level signal induced by VIMPO on a MATH-500 combinatorics problem. Both completions are sampled from a Qwen3-4B-Base model trained with VIMPO. For the same prompt, the model produces one correct response and one incorrect response, allowing us to compare the resulting token-level signals on two closely related reasoning trajectories.

Case study: Mr. Potato Head counting problem

Question. Tom got a Mr. Potato Head for his birthday. It came with 3 hairstyles, 2 sets of eyebrows, 1 pair of googly eyes, 2 sets of ears, and 2 sets of lips, a pair of regular shoes, and a bonus pair of clown shoes. If a complete Mr. Potato Head personality includes eyebrows, eyes, ears, lips, shoes and optionally hair, how many different wacky personalities can Tom come up with? Note that Mr. Potato Head can be bald. You cannot mix and match components within a pair or set.

Correct reasoning branch. The correct completion treats each pair or set as an indivisible component choice. Thus the valid counts are

$$\text{eyebrows} = 2, \quad \text{eyes} = 1, \quad \text{ears} = 2, \quad \text{lips} = 2, \quad \text{shoes} = 2, \quad \text{hair} = 4,$$

where the four hair options are the three hairstyles plus bald. This gives

$$2 \times 1 \times 2 \times 2 \times 2 \times 4 = 64.$$

Incorrect reasoning branch. The incorrect completion treats "both sets" as an additional valid option for some paired components. This changes the counts for eyebrows, ears, and lips from 2 to 3, producing

$$3 \times 1 \times 3 \times 3 \times 2 \times 4 = 216.$$

Figure 7 compares the VIMPO actor signal with a Monte Carlo rollout-based diagnostic on one correct and one incorrect completion for this problem. The VIMPO curve corresponds to the policy-implied GAE advantage defined in Equation 11. The Monte Carlo diagnostic is computed at 10 selected prefixes by sampling 32 continuations from the current policy at each prefix, then measuring the change in empirical correctness across adjacent prefixes to obtain a temporal-difference proxy.

Since this diagnostic is available only at a small number of prefixes, we expand it into a token-level step function for visual comparison. The curves are shown on signed normalized scales.

The Monte Carlo estimates suggest that continuation correctness decreases around the overcounting branch in the incorrect completion. The VIMPO advantage provides a complementary view of the trajectory, showing how the signal varies from token to token without requiring additional rollouts from intermediate prefixes.

Figure 8 shows a token-level view of selected critical spans from the same example. Each displayed cell corresponds to a model token, with whitespace and formatting-only tokens omitted for readability. The colors show the min–max normalized VIMPO GAE signal over the displayed tokens. This highlights local variation in the signal across the displayed tokens.

Together, these views illustrate the token-level structure of the policy-implied advantage alongside a rollout-based diagnostic of continuation correctness.

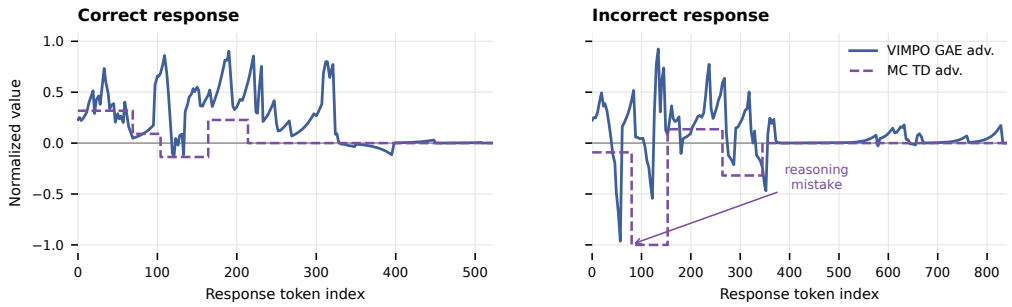


Figure 7: Token-aligned comparison between the VIMPO GAE actor signal and a Monte Carlo diagnostic based on differences in continuation correctness between selected prefixes. The curves are shown on signed normalized scales.

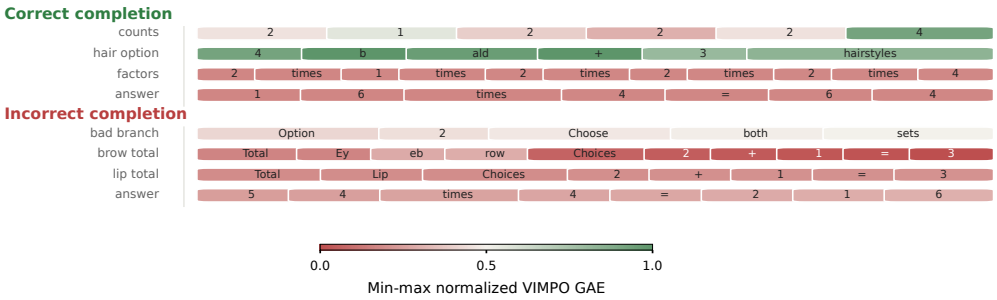


Figure 8: Token-level view of selected critical spans from the same combinatorics case using the VIMPO GAE signal. Each cell corresponds to one model token, with whitespace and formatting-only tokens omitted for readability. Colors show the relative strength of the VIMPO GAE signal after min–max normalization.

D Decomposition of the Outcome-Regression Gradient

Consider a stored trajectory $\tau = (s_t, a_t)_{t=0}^{T-1}$ with centered outcome $y = R_{\text{final}} - \overline{R}_{\text{final}}$. During an optimization step, the sampled states, actions, and outcome target are held fixed. Write the trajectory prediction in Equation 8 as

$$p_\theta(\tau) = \beta \sum_{t=0}^{T-1} \left[\log \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\text{ref}}(a_t \mid s_t)} - \text{sg}[\text{KL}(\pi_\theta \| \pi_{\text{ref}})(s_t)] \right]. \tag{20}$$

The reference is fixed and the KL term is detached, so the implemented derivative is

$$\nabla_\theta p_\theta(\tau) = \beta \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) = \beta S_\theta(\tau). \tag{21}$$

Applying the chain rule to $\mathcal{L}_V(\tau) = \frac{1}{2}(p_\theta(\tau) - y)^2$ gives

$$\begin{aligned} \nabla_\theta \mathcal{L}_V(\tau) &= (p_\theta(\tau) - y)\nabla_\theta p_\theta(\tau) \\ &= -\beta y S_\theta(\tau) + \beta p_\theta(\tau) S_\theta(\tau). \end{aligned} \tag{22}$$

Thus the regression update combines a return-weighted policy gradient with a prediction-dependent correction. The KL is recomputed when the policy changes, but contributes no derivative within this update because of the stop-gradient operation.

For a minibatch of B trajectories, let m_{it} mark valid transitions, let p_i be the corresponding masked trajectory prediction, and let $S_i = \sum_t m_{it} \nabla_\theta \log \pi_\theta(a_{it} \mid s_{it})$. Averaging the trajectory losses yields

$$\nabla_\theta \mathcal{L}_V^{\mathcal{B}} = -\frac{\beta}{B} \sum_{i=1}^{B} y_i S_i + \frac{\beta}{B} \sum_{i=1}^{B} p_i S_i. \tag{23}$$

The return-only ablation removes the second term. It can be implemented by minimizing

$$\mathcal{L}_{\text{return}} = -\frac{\beta}{B} \sum_{i=1}^{B} y_i \sum_t m_{it} \log \pi_\theta(a_{it} \mid s_{it}), \tag{24}$$

with detached y_i. Under matched configurations, this ablation retains the same outcome targets, trajectory aggregation, and optimizer settings as value-only VIMPO. It differs from the GRPO baseline, which standardizes group outcomes and uses a clipped behavior-policy likelihood-ratio surrogate with transition-level aggregation. The comparison therefore isolates the prediction-dependent correction within the regression update.

E WALKER2D: SETUP AND COMPONENT ANALYSIS

E.1 TASK, IMPLEMENTATION, AND CONFIGURATIONS

We use Walker2d-v5 with an eight-second horizon (1,000 transitions at 0.008 s each). The sole training reward is signed terminal displacement, $Y = x_{\text{end}} - x_0$. Native forward, alive, and control rewards are excluded. Both unhealthy termination and the planned horizon are terminal boundaries with no value bootstrap, and $\gamma = 1$. Each observation contains the 17 native physical features, displacement from reset, and elapsed fraction. A fixed random-policy calibration normalizes the observations. Displacement is scaled by ten and elapsed fraction is unscaled. Normalized inputs are clipped to $[-10, 10]$.

The policy is a two-layer, 256-unit tanh MLP with a diagonal Gaussian latent action and a tanh action transform. Its learned, state-independent log standard deviation starts at -0.5 and is bounded to $[-5, 2]$. PPO uses a separate critic of the same width and depth. All methods share paired actor initializations, Adam with $\epsilon = 10^{-5}$, gradient norm cap 0.5, and no entropy bonus. VIMPO uses exact Gaussian KL against the fixed initial policy. Clipped actor updates use the rollout behavior policy and clip ratio 0.2. The regression update is the original loss in Equation 8. Likelihood ratios are computed for latent actions, with the common transform Jacobian cancelling.

Each rollout contains eight reset states and eight independently sampled episodes per state. Complete trajectories are partitioned into eight minibatches. Regression loss averages over trajectories. Actor and critic losses average over valid transitions, excluding padding. VIMPO's detached suffix advantage uses $\lambda = .99$ and is whitened within each minibatch. GRPO standardizes terminal outcomes within each reset group. PPO computes complete-episode advantages and critic targets before minibatching, with critic coefficient 0.5 multiplying the half-squared error.

The initial optimizer search crossed three learning rates, $\{7.5 \times 10^{-5}, 1.5 \times 10^{-4}, 3 \times 10^{-4}\}$, with four or eight passes, using two tuning seeds and 512,000 transitions. VIMPO-specific coefficients

Table 3: Walker2d optimizer configurations. VIMPO uses $\beta = .005$ and, when enabled, $c_A = .0005$. Passes are over each collected rollout. The matched controls share the value-only configuration.

Method	Learning rate	Passes	Actor / critic setting
VIMPO (value only)	3×10^{-4}	8	No auxiliary actor
VIMPO (value + actor)	7.5×10^{-5}	8	Implied actor, $\lambda = .99$
GRPO	3×10^{-4}	8	Group-standardized return
PPO	3×10^{-4}	4	GAE $\lambda = 1$
Return-only (selected)	7.5×10^{-5}	4	Return term only
Return-only (matched)	3×10^{-4}	8	Return term only
VIMPO (matched actor)	3×10^{-4}	8	Implied actor, $\lambda = .99$
Value + GRPO actor	3×10^{-4}	8	Replaces implied actor
VIMPO (delayed actor)	3×10^{-4}	8	Actor enabled at 2.048M
PPO ($\lambda = .99$)	3×10^{-4}	4	Original critic setting

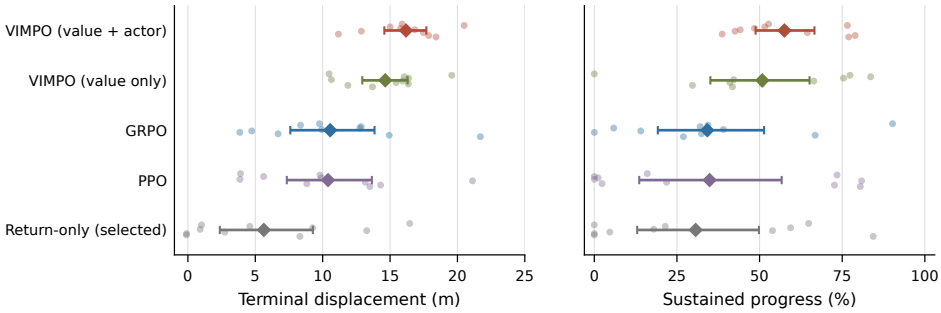


Figure 9: Walker2d with terminal displacement rewards. Faint points are individual training-seed means; diamonds and bars show means and descriptive 95% seed-bootstrap intervals. Sustained progress means completing the horizon upright with forward velocity above 0.25 m/s over the final quarter. PPO uses $\lambda = 1$; return-only uses its separately selected optimizer configuration.

were fixed. A subsequent PPO screen compared the original setting, smaller minibatches, and $\lambda = 1$ over 2.048M transitions on the two tuning seeds. The selected $\lambda = 1$ configuration was then trained from initialization on all ten evaluation seeds. With $\gamma = \lambda = 1$, its critic targets are Monte Carlo terminal returns. The PPO and actor-source follow-ups reuse the existing evaluation seeds. Tuning seeds are distinct.

E.2 Evaluation and learning dynamics

Each run completes the rollout that crosses 16,384,000 valid training transitions, allowing an overshoot of less than 64,000. The endpoint evaluation uses 256 independent reset starts per seed with sampled actions. Sustained progress requires a healthy 1,000-step episode and average forward velocity above 0.25 m/s over its final 250 steps.

Figure 9 reports the seed-level endpoint results. The sampled-action learning curve in Figure 4 and the sampled-/mean-action comparison in Figure 10 use a separate 32-episode panel over the common archived 8.2M–16.4M continuation window.

Training seeds are the replication units. Endpoint intervals describe each method's mean; learning-curve bands are pointwise intervals. Paired comparisons retain their original simultaneous 95% families: 28 historical contrasts, eight PPO follow-up contrasts, four actor-source contrasts, and 12 credit contrasts. These families are adjusted separately. For displacement, selected full VIMPO exceeds GRPO by 5.62 m [1.25, 10.00] in the historical family. Value-only and selected full exceed updated PPO by 4.24 m [0.20, 8.29] and 5.78 m [1.04, 10.52], respectively, in the PPO follow-up family.

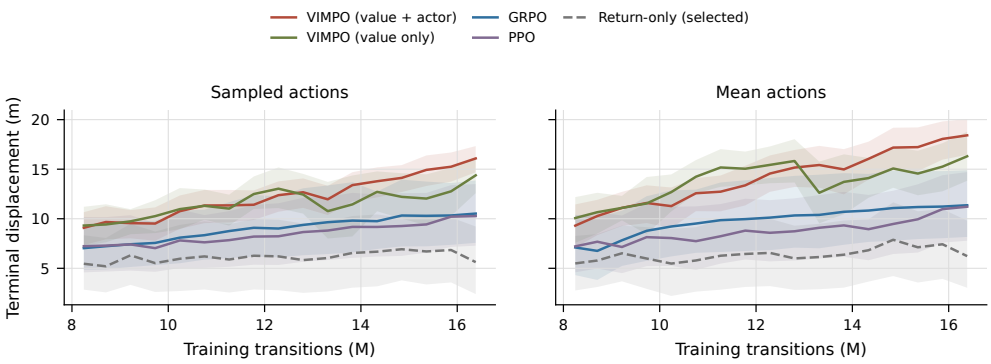


Figure 10: Walker2d learning curves over the common continuation window. Lines are means across ten seeds; shaded bands are pointwise descriptive 95% seed-bootstrap intervals. Each curve evaluation uses 32 episodes per seed. No smoothing is applied.

Table 4: Walker2d endpoints, including matched controls. Entries are means across ten training seeds, each evaluated on 256 sampled episodes.

Method	Displacement (m)	Sustained progress (%)
VIMPO (value + actor)	16.18	57.50
VIMPO (value only)	14.64	50.82
GRPO	10.56	34.18
PPO	10.40	34.88
Return-only (selected)	5.63	30.66
Return-only (matched)	-0.14	0.00
VIMPO (matched actor)	14.62	39.61
Value + GRPO actor	14.46	39.49
VIMPO (delayed actor)	16.40	50.35
PPO ($\lambda = .99$)	8.28	32.85

E.3 Regression and actor contributions

Return-only retains the return-dependent component of Equation 9, with its objective derived in Appendix D. Its matched configuration removes only the prediction-dependent correction; the selected configuration additionally changes learning rate and update passes. Value-only VIMPO improves displacement over these controls by 14.79 m [12.19, 17.39] and 9.01 m [2.72, 15.30], respectively. Matched return-only policies saturate their actions and fall early, whereas the separately selected configuration improves to 5.63 m.

E.4 Same-policy Monte Carlo credit measurements

We measure credit on value-only, selected-full, and matched-full checkpoints at initialization, 8.192M, and 16.384M transitions. Shared initial policies are measured once, giving 70 unique panels across ten seeds. Each panel samples eight groups of eight episodes. The first two episodes in each group supply queries at steps 0, 50, 250, 500, and 750 whenever the trajectory is still active. At each query, we restore the full simulator state and sample 128 paired continuations: one fixes the queried action to estimate Q, while the other samples it to estimate V. Both subsequently follow the checkpoint policy and share future action noise within each pair.

We compare one-step implied TD advantages, whitened implied suffix advantages, and standardized GRPO advantages on the same policy-generated trajectories. Each signal weights the policy score gradients at the queried states; alignment is the cosine with the corresponding Monte Carlo gradient estimate. Table 5 reports final alignment relative to GRPO and its change from 8M to 16M. The intervals do not resolve a consistent ordering of the signals.

Table 5: Walker2d credit alignment: differences in gradient cosine, with simultaneous 95% intervals across the 12 contrasts. Positive values favor the implied signal. "Change" is the change in relative alignment from 8.192M to 16.384M.

Policy	Signal	Final minus GRPO	Change
Value only	One-step	+0.165 [-0.134, +0.464]	+0.240 [-0.111, +0.590]
Value only	Actor	+0.072 [-0.185, +0.328]	+0.195 [-0.130, +0.520]
Selected full	One-step	+0.045 [-0.199, +0.289]	-0.071 [-0.402, +0.260]
Selected full	Actor	+0.095 [-0.139, +0.329]	+0.100 [-0.178, +0.379]
Matched full	One-step	-0.111 [-0.513, +0.291]	-0.330 [-0.815, +0.155]
Matched full	Actor	-0.053 [-0.450, +0.344]	-0.097 [-0.480, +0.287]

Algorithm 1 VIMPO training step

Require: Policy model π_θ, reference model π_{ref}, prompt distribution $\mathcal{D}$, group size G, coefficients β, c_A, learning rate η

for each training step **do**
  Sample prompt $\boldsymbol{x} \sim \mathcal{D}$ and rollouts $\{\tau^{(i)}\}_{i=1}^{G} \sim \pi_\theta(\cdot \mid \boldsymbol{x})$
  Compute final rewards $R^{(i)}$ and group baseline $\overline{R} = \frac{1}{G}\sum_i R^{(i)}$
  for each rollout i and token t **do**
    Compute $\rho_t^{(i)} = \beta \log \frac{\pi_\theta(a_t^{(i)} \mid \boldsymbol{s}_t^{(i)})}{\pi_{\mathrm{ref}}(a_t^{(i)} \mid \boldsymbol{s}_t^{(i)})}$
    Compute $\kappa_t^{(i)} = \beta\, \mathrm{KL}\big(\pi_\theta(\cdot \mid \boldsymbol{s}_t^{(i)}) \| \pi_{\mathrm{ref}}(\cdot \mid \boldsymbol{s}_t^{(i)})\big)$
    Set policy-implied advantage $A_t^{(i)} = \rho_t^{(i)} - \kappa_t^{(i)}$
  end for
  Compute value loss $\mathcal{L}_V = \frac{1}{2G}\sum_i \Big[\sum_t (\rho_t^{(i)} - \mathrm{sg}[\kappa_t^{(i)}]) - (R^{(i)} - \overline{R})\Big]^2$
  Normalize detached actor advantages over valid response tokens: $\widetilde{A}_t^{(i)} = \frac{A_t^{(i)} - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}} + \epsilon}$
  Compute PPO actor loss $\mathcal{L}_A = \text{PPO-Clip}\big(\mathrm{sg}[\{\widetilde{A}_t^{(i)}\}], \pi_\theta, \pi_{\theta_{\mathrm{old}}}\big)$
  Update $\theta \leftarrow \theta - \eta \nabla_\theta (\mathcal{L}_V + c_A \mathcal{L}_A)$
end for

The Monte Carlo gradient's split-half agreement is 0.887 at initialization and 0.066, 0.093, and 0.176 at the final value-only, selected-full, and matched-full checkpoints, respectively. This weak late-training agreement limits the precision of the alignment comparison. Some panels also have nonpositive noise-corrected signal energy, making a complete ten-seed mean normalized error undefined. We therefore report the alignment contrasts together with the precision diagnostic. Queries at later times concern surviving trajectories, and the native velocity observations are clipped; simulator-state restoration is exact. These measurements complement the actor-source training comparison by directly examining the learned signals.

F Training Algorithm

G Noisy-Reward Learning Curves

H Extended Discussion and Limitations

Outcome regression as policy learning. The value loss in VIMPO is more than a way to construct an actor signal: it directly trains the policy from outcome rewards. On terminal-reward Walker2d, value-only VIMPO learns effective policies from scratch, and matched ablations show that retaining the prediction-dependent correction substantially improves performance over the return-dependent component alone. These results suggest that outcome regression deserves further study as a policy-learning objective in its own right, beyond its role in providing policy-implied advantages.

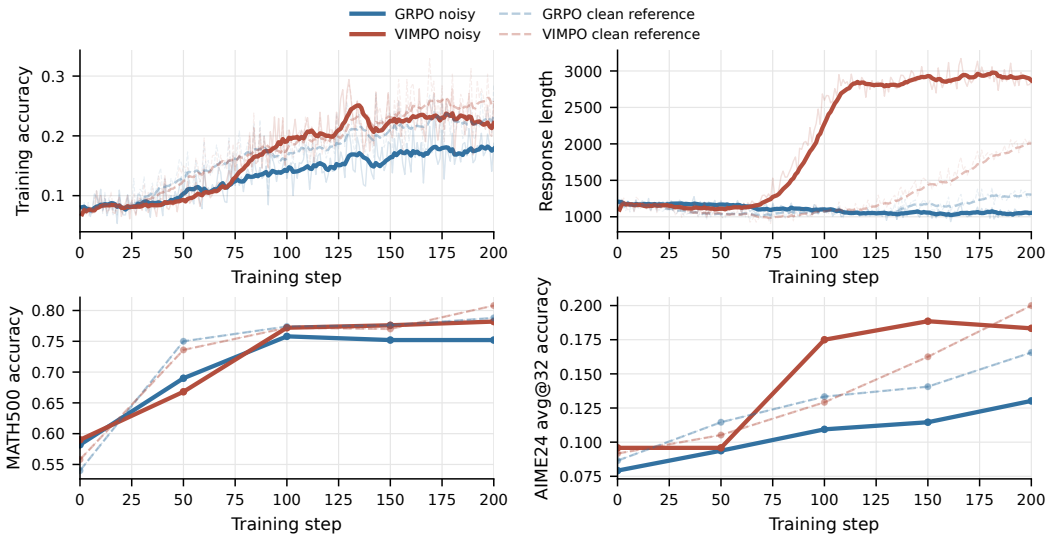


Figure 11: Noisy-reward stress test over the first 200 training steps. Solid curves show runs trained with 25% reward flipping, while faint dashed curves show clean-reward runs for reference. Metrics are evaluated with the clean verifier.

When does the implied actor help? The auxiliary actor improves training accuracy per step in our LLM ablation, whereas matched Walker2d runs achieve similar mean displacement with and without it. Its additional benefit therefore appears to depend on the learning setting. One possibility is that pretraining and greater policy expressiveness make the policy-implied signal more useful for subsequent updates. However, our experiments do not isolate these factors, so this remains a hypothesis rather than an established explanation. Comparisons that vary initialization and model capacity within the same task would help clarify when reusing the policy-implied advantage provides the greatest benefit.

Reference dependence and rollout reuse. Our implementation uses a frozen reference policy and fixed coefficients β and c_A throughout training. The coefficient ablations show that these weights influence learning progress, movement away from the reference, and response length. Adaptive coefficient schedules could accommodate changes in these dynamics over training; whether updating the reference policy offers additional benefits remains to be tested. Rollout reuse presents a separate optimization issue. During repeated updates, the policy changes while the sampled trajectories remain fixed, but the regression loss uses current policy-reference log-ratios without behavior-policy importance weighting or clipping. Understanding the effect of this mismatch could guide the design of importance-weighted or constrained regression updates.

Computational cost and evaluation scope. Although VIMPO removes the separate critic, exact KL computation in the LLM setting still requires access to the full next-token distributions of both the policy and the reference model. Approximate KL estimators could reduce this overhead, but their effect on the centering used by the value objective must be assessed. Our empirical evaluation covers mathematical RLVR with a 4B-scale language model and a terminal-reward variant of Walker2d. The Walker2d study includes ten training seeds and comparisons with PPO and GRPO, while the LLM evaluation uses GRPO variants. Repeated LLM training runs, additional tasks and model scales, and comparisons with tuned actor-critic methods in the LLM setting would help establish how broadly the observed benefits apply.

I Extended Related Work

Actor-critic methods with learned values. PPO (Schulman et al., 2017) remains the standard actor-critic algorithm for policy optimization and is commonly paired with generalized advantage estimation (Schulman et al., 2016). In LLM post-training, actor-critic variants train a value model

alongside the policy to provide dense token-level credit assignment. VAPO (Yue et al., 2025) adapts this paradigm to long-CoT reasoning with value pretraining, decoupled GAE, length-adaptive estimation, and token-level policy-gradient updates. These methods provide fine-grained learning signals but require an additional value model, whose errors can affect policy optimization. VIMPO targets the same credit-assignment role without training a separate critic.

Critic-free and post-hoc credit assignment methods. GRPO (Shao et al., 2024) removes the learned value network by estimating advantages from a group of rollouts sampled for the same prompt. DAPO (Yu et al., 2025) builds on this critic-free paradigm with dynamic sampling, token-level loss aggregation, and decoupled clipping for long-CoT training. These methods avoid critic pretraining and critic-policy co-training, but their advantage signal is usually derived at the trajectory level and shared across tokens. Recent work therefore attempts to refine this coarse signal post hoc. FIPO (Ma et al., 2026) reweights the group-relative advantage using a future-KL factor, while attention-based credit assignment methods (Li et al., 2025) identify important tokens from internal attention statistics. These approaches introduce token-level structure through auxiliary weighting rules. VIMPO instead derives its token-level actor signal from the KL-regularized optimality conditions and a policy-implied value recurrence.

Log-ratio parameterizations and Bellman consistency. DPO (Rafailov et al., 2023) shows that the optimal policy of a KL-regularized preference objective can be expressed through a log-ratio against a reference policy, avoiding explicit reward-model training. IPO (Gheshlaghi Azar et al., 2024) further analyzes preference optimization under related log-ratio parameterizations, and TDPO (Zeng et al., 2024) introduces token-wise log-ratio terms while remaining in the offline pairwise-preference setting. VIMPO also uses policy-reference log-ratios but applies them to online RL with rollout rewards. The Bellman side of our derivation is related to classical consistency-based RL. Path Consistency Learning (Nachum et al., 2017) trains policies and values by enforcing multi-step consistency conditions, while soft Q-learning (Haarnoja et al., 2017) derives policy updates from entropy-regularized Bellman equations. VIMPO shares the idea of deriving learning signals from Bellman-style identities, but uses the standard expected-return value in a KL-regularized LLM post-training objective. This keeps the reference-policy log-ratio explicit and yields a critic-free value and actor update for autoregressive generation.

J Language-Model Evaluation and Baseline Details

LLM experiments. We use the math subset of Guru (Cheng et al., 2025) as the RLVR training set. Guru provides a larger curated collection of verifiable math problems than DAPO-Math (Yu et al., 2025), allowing us to study training behavior over longer runs. Unless otherwise stated, all runs start from Qwen3-4B-Base (Yang et al., 2025) and are implemented in verl (Sheng et al., 2024).

We evaluate on MATH-500 (Hendrycks et al., 2021), AIME 2024 (Mathematical Association of America, 2024), AIME 2025 (Mathematical Association of America, 2025), and OlympiadBench (He et al., 2024). For MATH-500 and AIME 2024, we use the same prompt format and veRL evaluation pipeline used during training. For AIME 2025 and OlympiadBench, we use the LightEval implementation (Hugging Face, 2025). Because these evaluation pipelines differ slightly in prompting and answer extraction, we compare methods only within the same benchmark.

For the language-model experiments, we compare VIMPO with GRPO under the same shared hyperparameters, listed in Appendix A.1. For the main comparison, we use $\beta = 5 \times 10^{-4}$ and actor coefficient $c_A = 5 \times 10^{-3}$ for VIMPO. These values are selected by coarse tuning on training dynamics and then held fixed for the main runs. We do not tune shared optimizer settings, such as the learning rate, separately for each method. This controls the configuration for the reported comparison.

For the GRPO baseline, we use token-level loss aggregation and disable the KL reward penalty, following common RLVR practice and the DAPO training recipe (Yu et al., 2025). We also include a naive GRPO baseline, which uses the original sequence-mean-token-mean GRPO loss (Shao et al., 2024). We do not include additional DAPO-style improvements such as dynamic sampling or filtering. This keeps the comparison focused on the effect of replacing the group-relative objective with the VIMPO value loss and actor update.

For the noisy-reward stress test, we corrupt the training reward by flipping the binary correctness signal independently with probability 25% within each prompt group. The corrupted rewards are used only for optimization. Training accuracy and validation accuracy are always reported using the clean, uncorrupted verifier.