REWARD SHAPING TO MITIGATE REWARD HACKING IN RLHF

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning from human feedback (RLHF) is widely used to align large language models (LLMs) with human preferences. However, RLHF remains vulnerable to *reward hacking*, whereby a policy exploits imperfections in the reward function instead of learning the intended behavior, thereby undermining alignment. Although reward shaping can stabilize RLHF training and partially mitigate reward hacking, shaping methods and their underlying design principles have not been systematically investigated. To address this gap, we conduct a comprehensive study of prevalent reward-shaping techniques. Our analysis identifies two key design principles: (1) the reinforcement-learning reward should be bounded, and (2) it should grow rapidly at first and then gradually saturate. Motivated by these principles, we propose Preference as Reward (PAR), a novel method that uses the latent preferences encoded in the reward model as the reinforcement-learning signal. We further show that PAR possesses two variance-reduction properties that stabilize RLHF training and substantially widen the practical window for early stopping. Our evaluation consists of two parts. First, we compare PAR with several other reward-shaping strategies using Proximal Policy Optimization (PPO) as the reinforcement-learning algorithm and Gemma2-2B as the base model. Second, we compare PAR with the vanilla baseline (i.e., unshaped reward) across four base models and four reinforcement-learning algorithms. In the first set of experiments, PAR consistently outperforms other reward-shaping methods and also reflects high data efficiency and robustness. The second set of experiments shows that PAR is particularly effective for actor-critic RL algorithms when value estimates become unstable and demonstrates its effectiveness across different base models. The code will be released after the review process.

1 INTRODUCTION

Reinforcement learning from human feedback (RLHF) is widely used to align large language models (LLMs) with human intentions and improve their capabilities (Ouyang et al., 2022; OpenAI, 2024; Bai et al., 2022; Guo et al., 2024). However, RLHF remains vulnerable to *reward hacking*, whereby a policy exploits weaknesses in the reward model to obtain high rewards without genuinely improving alignment or response quality (Amodei et al., 2016; Gao et al., 2023; Singhal et al., 2023). This may lead to undesirable behaviors such as repetitive or unnecessarily verbose responses.

Proximal Policy Optimization (PPO) (Schulman et al., 2017) is widely adopted in RLHF (Ouyang et al., 2022), yet it is particularly susceptible to reward hacking (Gao et al., 2023). Existing mitigation strategies often reshape proxy rewards through clipping or rescaling (Dai et al., 2024; Wang et al., 2024b). Nevertheless, reward-shaping methods have not been systematically compared, and their underlying design principles remain unclear.

To address this gap, we systematically study reward shaping for RLHF. Our analysis suggests that excessively large rewards can become decoupled from true response quality and destabilize critic learning, motivating our first design principle: *(1) the RL reward should be bounded*. We further find that optimization is more stable in low-reward regions, leading to our second principle: *(2) the RL reward should grow rapidly at first and then gradually saturate*. Guided by these principles, we introduce *Preference as Reward* (PAR), a reward-shaping method illustrated in Figure 1. PAR applies a sigmoid function to the centered proxy reward: $r_{\text{RL}} = \text{sigmoid}(r - r_{\text{ref}})$, where r and r_{ref}

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.

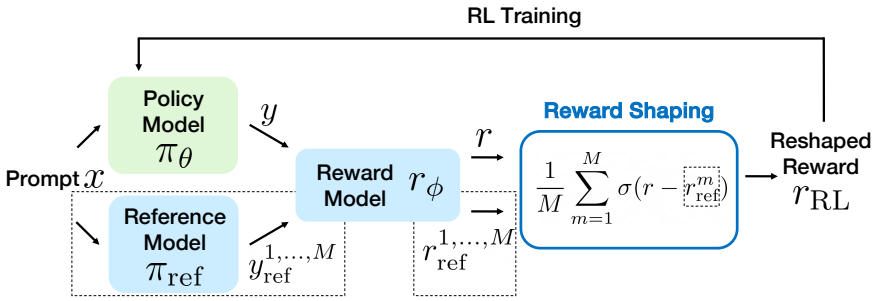


Figure 1: RLHF training with reward shaping. The reward model assigns proxy rewards to policy responses, which are then optionally transformed using reference rewards (dashed box) before policy optimization. PAR applies a sigmoid transformation to the centered proxy reward.

denote the rewards of the policy and reference responses, respectively. Because the policy is typically initialized from the reference model, the centered reward is initially close to zero, where the sigmoid has its steepest slope and thus facilitates rapid learning. As training progresses, the sigmoid gradually saturates, limiting extreme rewards and improving stability.

PAR is also closely related to the Bradley–Terry preference model (Bradley & Terry, 1952). When exponentiated proxy rewards are viewed as latent preference strengths, analogous to Elo ratings (Elo, 1978), r_{RL} represents the reward model's preference for the policy response over the reference response. PAR therefore converts absolute proxy scores into bounded and interpretable preference signals. We further show that PAR reduces the variances of both accumulated returns and policy-gradient estimates, two factors that are critical to stable RLHF training. Although reward shaping cannot eliminate reward hacking under sufficiently long training, PAR delays its onset and widens the practical window for early stopping.

Our experiment consists of two parts. First, we compare PAR with several other reward-shaping methods using Gemma2-2B (Gemma Team, 2024), UltraFeedback Binarized (Cui et al., 2023), and PPO. PAR achieves the strongest performance on both the in-domain test set and out-of-domain benchmarks like AlpacaEval 2.0 (Li et al., 2023) and MT-Bench (Zheng et al., 2023a). It also requires only a single reference reward to achieve high performance and remains robust to reward hacking after two training epochs. Second, we evaluate PAR across multiple base models and optimization algorithms on HH-RLHF (Bai et al., 2022). PAR consistently outperforms the unshaped-reward baseline under PPO and Advantage Actor-Critic (A2C) algorithms, mitigating reward hacking and stabilizing critic learning. Its effect is smaller under Group Relative Policy Optimization (GRPO), supporting our hypothesis that critic stability is an important source of PAR's benefits. We also find that Direct Preference Optimization (DPO) is already highly effective on preference datasets.

Our main contributions are as follows:

- We identify two principles for effective reward shaping: boundedness and rapid initial growth followed by gradual saturation.
- We introduce PAR, connect it to the preferences encoded by the reward model, and establish its variance-reduction properties.
- We empirically verify that PAR mitigates reward hacking, stabilizes training, and generalizes across models, datasets, and optimization algorithms.

2 Related Work

Characterizing reward hacking. Reward hacking occurs when optimization exploits discrepancies between a proxy reward and the intended objective (Amodei et al., 2016). In a synthetic setting, Gao et al. (2023) show that gold reward can decline as proxy-reward optimization intensifies, while Singhal et al. (2023) identify response length as a major driver of apparent RLHF improvements. KL regularization constrains policy drift but provides no guarantee under heavy-tailed reward misspecification (Kwa et al., 2024). RLHF can also make incorrect responses more convincing (Wen et al.,

2025), and outcome-only supervision can reward invalid reasoning that reaches correct answers (Yuan et al., 2026). These findings motivate interventions throughout the RLHF pipeline.

Robust reward modeling. Reward-model ensembles mitigate overoptimization, although correlated errors may persist (Coste et al., 2024; Eisenstein et al., 2024); WARM improves efficiency by averaging reward models in weight space (Rame et al., 2024). Other approaches include information-bottleneck reward modeling (Miao et al., 2024), hidden-state regularization (Yang et al., 2024), and causal data augmentation to separate contextual preferences from spurious artifacts (Liu et al., 2025). Length-specific defenses use disentangled reward heads (Chen et al., 2024), context-adaptive length modeling (Bu et al., 2025), post-hoc calibration (Huang et al., 2025), or nonlinear bias fitting (Zhao et al., 2026). For reasoning tasks, rubric-based rewards evaluate intermediate steps rather than only final answers (Yuan et al., 2026).

Policy-side mitigation. Constrained RLHF restricts reward optimization to regions where reward models remain reliable (Moskovitz et al., 2024). AdvPO incorporates lightweight reward uncertainty into a distributionally robust objective (Zhang et al., 2024), while BSPO regularizes out-of-distribution value estimates (Dai et al., 2025). EPPO penalizes excessive final-layer energy loss (Miao et al., 2025), and PF-PPO filters samples with unreliable rewards (Zhang et al., 2025). These policy-side methods are complementary to post-hoc reward transformation.

Reward transformation. Reward transformation modifies the optimization signal without retraining the reward model. Wang et al. (2024b) derive a log-sigmoid-centered transformation of Bradley–Terry rewards, while Shen et al. (2024) construct contrastive rewards relative to baseline responses. Related inference-time work regularizes best-of-N selection using minimum Bayes risk (Jinnai et al., 2024). PAR instead uses the Bradley–Terry preference probability itself as the RL reward relative to a reference response, yielding a bounded signal in $[0, 1]$ that can be combined with reward-model and policy-side defenses.

3 METHOD

3.1 DESIGN PRINCIPLES

As detailed in Section 1, we restate our two design principles here: (1) RL reward should be bounded, (2) RL reward should grow rapidly at first and then gradually saturate. To elucidate the rationale behind these principles, we examine the PPO policy and critic loss functions for prompt x and response y (notation detailed in Table 3):

$$\mathcal{L}_{\text{policy}}(\theta) = -\mathbb{E}_{x\sim\mathcal{D},y\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\min\left(\frac{\pi_\theta(y_t|x,y_{<t})}{\pi_{\theta_{\text{old}}}(y_t|x,y_{<t})}\cdot A_t, \text{clip}\left(\frac{\pi_\theta(y_t|x,y_{<t})}{\pi_{\theta_{\text{old}}}(y_t|x,y_{<t})}, 1-\epsilon, 1+\epsilon\right)\cdot A_t\right)\right],$$

$$\mathcal{L}_{\text{critic}}(\alpha) = \mathbb{E}_{x\sim\mathcal{D},y\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[||V_\alpha(x,y_{<t}) - G_t||_2^2\right].$$

For the policy loss, $A_t = \sum_{l=t}^{T}(\gamma\lambda)^{l-t}\delta_l$ represents the generalized advantage estimation (GAE) at token t, where $\delta_t = r_t + \gamma V_{\alpha_{\text{old}}}(s_{t+1}) - V_{\alpha_{\text{old}}}(s_t)$ is the temporal difference (TD) error. π_θ denotes the current policy model, and $\pi_{\theta_{\text{old}}}$ refers to the policy model from the previous iteration. $V_{\alpha_{\text{old}}}$ is the critic's value function from the previous iteration. For the critic loss, $G_t = \sum_{l=t}^{T}\gamma^{l-t}r_l$ represents the return, defined as the discounted sum of per-token rewards. The per-token reward at position t, denoted as r_t, is defined as: $r_t = \begin{cases} r_{\text{RL}} - \eta\log\frac{\pi_\theta(y_t|x,y_{<t})}{\pi_{\text{ref}}(y_t|x,y_{<t})} & \text{if } t = T \\ -\eta\log\frac{\pi_\theta(y_t|x,y_{<t})}{\pi_{\text{ref}}(y_t|x,y_{<t})} & \text{if } t < T \end{cases}$.

The first principle, advocating for bounded RL rewards, is crucial for stabilizing critic training. Excessively large rewards can hinder the critic model's ability to accurately learn the value function, as illustrated in Figure 2. We hypothesize that this issue arises from the nature of the regression loss used in the critic model. Specifically, large RL rewards r_{RL} lead to large variance of returns G_t (see Theorem 3.1), making the critic loss $\mathcal{L}_{\text{critic}}(\alpha)$ more challenging to optimize. Furthermore, this effect propagates to the excessive advantage estimate A_t, rendering it unstable and leading to overly aggressive policy updates. The second principle aims to encourage the model to learn more aggressively in the low-reward region, which we find to be relatively safer to optimize. In contrast,

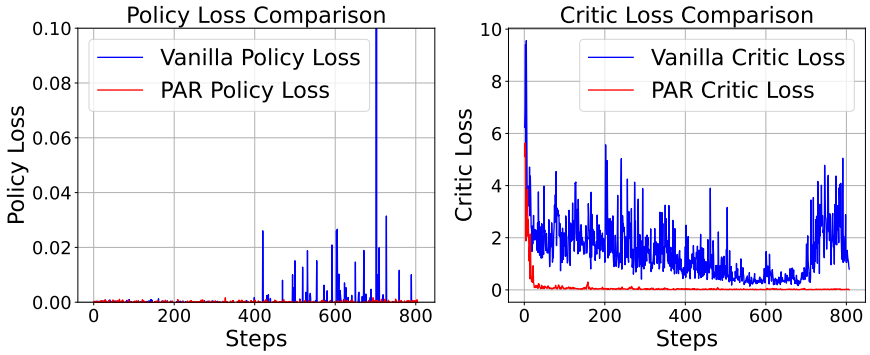


Figure 2: Loss curves from PPO training show that PAR exhibits greater stability, particularly in critic loss, compared to Vanilla training. This stability is attributed to PAR's bounded RL reward.

the high-reward region is more susceptible to reward hacking. Therefore, we design the RL reward to converge gradually in later stages of training.

We investigate several candidate functions that satisfy the two design principles, with a primary focus on sigmoid-like functions. This choice is motivated by the theoretical result that the sigmoid function minimizes the variance of the policy gradient among unbiased shaping functions under logistic preference noise (see Theorem 3.2). The corresponding curves for these candidate functions are illustrated in Figure 4b.

3.2 Preference as Reward

After careful consideration and empirical evaluation, we recommend using the sigmoid function applied to centered rewards as the reward shaping method. The sigmoid function is bounded, has the steepest slope at the initial point (zero), and converges gradually to its upper bound of one. This property makes it particularly suitable for stabilizing the RL training process. Furthermore, our analysis reveals that this shaping approach is intrinsically linked to the hidden preferences encoded within the reward model. Given a reward model r_ϕ, the hidden preference between two responses y and y' to a prompt x can be expressed as: $\mathcal{P}_\phi(y \succ y'|x) = \sigma(r_\phi(x, y) - r_\phi(x, y'))$. This formulation shows that applying the sigmoid function to centered rewards corresponds precisely to the preference score of the policy response over the reference response. Consequently, we term this method **Preference As Reward (PAR)**. To enhance stability, we use M reference rewards: $r_{\text{RL}} = \frac{1}{M}\sum_{m=1}^{M} \sigma(r - r_{\text{ref}}^m) = \frac{1}{M}\sum_{m=1}^{M} \mathcal{P}_\phi(y \succ y_{\text{ref}}^m)$. The pseudo-code for the reward shaping procedure under PAR is presented in Algorithm 12. The overall reward shaping pipeline is illustrated separately in Figure 1.

3.3 Theoretical Analysis

We further establish that our proposed PAR method satisfies two key theoretical properties: (1) The return variance is upper bounded, and (2) Policy gradient variance is minimized among unbiased shaping functions under logistic preference noise. A rigorous derivation of these guarantees is provided in Appendix F. These theoretical foundations are essential for ensuring the robustness and reliability of PAR in practical applications.

Theorem 3.1 (Bounded rewards reduce return variance)**.** *Let $\gamma \in [0, 1)$ and define the discounted return $G_t = \sum_{l=t}^{T} \gamma^{l-t} r_l$, where the per-step reward satisfies $|r_l| < 1$, then Var$[G_t] \le \frac{1}{(1-\gamma)^2}$.*

Theorem 3.2 (Sigmoid is the minimum-variance unbiased shaping under logistic preference noise)**.** *Sample a prompt $x \in \mathcal{D}, y \sim \pi_\theta(\cdot \mid x), y_{\text{ref}} \sim \pi_{\text{ref}}(\cdot|x)$, let $z(x, y, y_{\text{ref}}) = r_\phi(x, y) - r_\phi(x, y_{\text{ref}})$.*

Define the REINFORCE estimator $\tilde{g} = \nabla_\theta \log \pi_\theta(y \mid x)\, \tilde{r}$. where $\tilde{r}$ may use any additional randomness but must satisfy the pointwise unbiasedness constraint $\mathbb{E}[\tilde{r} \mid x, y, y_{\text{ref}}] = \sigma(z(x, y, y_{\text{ref}}))$. Then the unique minimum-variance estimator in this class is $g_\sigma := \nabla_\theta \log \pi_\theta(y \mid x)\, \sigma(z(x, y, y_{\text{ref}}))$, and for every estimator $\tilde{g}$, Var$(\tilde{g}) \ge$ Var(g_σ).

Theorem 3.1 addresses the stability of critic model training, while Theorem 3.2 focuses on stabilizing policy model training. Together, these variance reduction properties help mitigate the reward hacking problem by enabling a prolonged time window during which the policy model maintains high performance.

4 Comparison of Reward-Hacking Mitigation Methods

This first set of experiments evaluates the two proposed reward-shaping principles, compares PAR with existing reward-hacking mitigation methods, and examines the data efficiency and robustness of PAR.

4.1 Experimental Setting

Datasets and models. We use UltraFeedback Binarized (Cui et al., 2023) as the training dataset, Gemma2-2B (Gemma Team, 2024) as the base model, and Proximal Policy Optimization (PPO) as the RL algorithm.

Mitigation baselines. We compare PAR with seven reward-hacking mitigation methods:

- **WARM** (Rame et al., 2024) averages the weights of multiple reward models and uses the resulting model to provide rewards during RL training.
- **ODIN** (Chen et al., 2024) jointly trains length-correlated and length-decorrelated reward heads. Only the length-decorrelated quality head is used during RL training.
- **Reg** (Dai et al., 2024) adds an ℓ_2 regularization term to the reward-model training objective: $\mathcal{L}_{\text{reward}} = \mathbb{E}_{(x,y_w,y_l)\sim D}\left[-\log\sigma(r_\phi(x,y_w) - r_\phi(x,y_l)) + \beta\|r_\phi(x,y_w)\|_2^2 + \beta\|r_\phi(x,y_l)\|_2^2\right]$.
- **Meanstd** standardizes the reward using its running mean and standard deviation: $r_{\text{RL}} = \frac{r-\mu}{s}$, where μ and s are the running mean and standard deviation, respectively.
- **Clip** clips the reward: $r_{\text{RL}} = \text{clip}(r, \mu - s, \mu + s)$.
- **Minmax** rescales the reward using its running minimum and maximum: $r_{\text{RL}} = \frac{r-r_{\min}}{r_{\max}-r_{\min}}$, where $r_{\min}$ and $r_{\max}$ are the running minimum and maximum, respectively.
- **LSC** (Wang et al., 2024b) applies the log-sigmoid-centered transformation $r_{\text{RL}} = \log\sigma\left(r - r_{\text{ref}}^{(0.85)}\right)$, where $r_{\text{ref}}^{(0.85)}$ is the 85th percentile of the Gaussian distribution fitted using the mean and variance of the reference rewards.

Evaluation metrics. We monitor two test-set metrics during training: the *proxy reward* and the *win rate*. The win rate measures how often the policy response is preferred to the SFT response by DeepSeek-V3 (DeepSeek-AI, 2024). The horizontal axis denotes the cumulative number of training examples. For AlpacaEval 2.0 (Li et al., 2023), we report the length-controlled (LC) win rate, raw win rate, and average response length. Both win rates are computed against the SFT model using DeepSeek-V3 as the judge. For MT-Bench (Zheng et al., 2023a), DeepSeek-V3 evaluates the first-turn, second-turn, and overall performance.

Training details. After removing noisy examples from UltraFeedback Binarized, we train the SFT model on the chosen responses for two epochs and the reward model—the base model with a linear reward head—for one epoch, both at a learning rate of 5×10^{-6}. The policy is initialized from the SFT model and trained for one epoch at 3×10^{-7}, and the critic from the reward model at 5×10^{-6}. All stages use a linear schedule with warm-up over the first 0.1 epoch. We evaluate every 0.1 epoch, yielding ten checkpoints per mitigation method. Full details are in Appendix B.1.

4.2 Principle 1: Bounded Rewards

To evaluate the principle that *the RL reward should be bounded*, we vary the KL-penalty coefficient and impose different upper bounds on the reward during PPO training. As shown in Figure 3, restricting excessive reward optimization substantially mitigates reward hacking. Increasing the

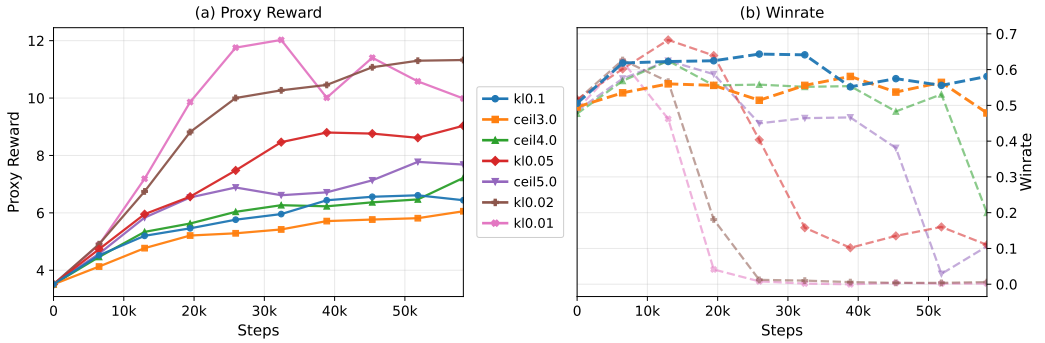
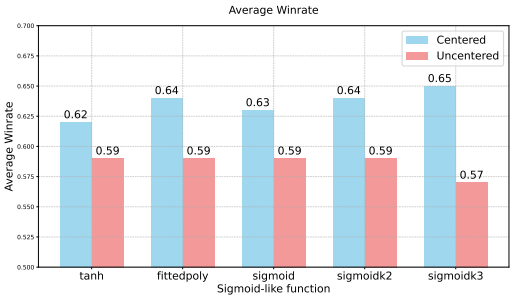
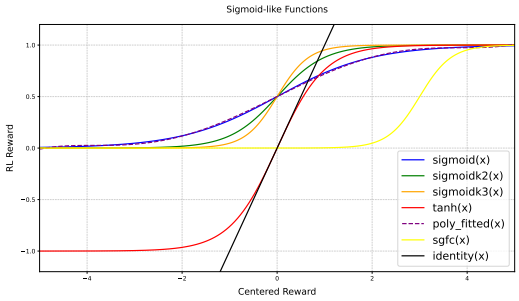


Figure 3: PPO training curves over two epochs. `ceil5.0` denotes $r_{\text{RL}} = \min(r, 5.0)$, and `kl0.1` denotes a KL-penalty coefficient of $\beta = 0.1$. Restricting excessive reward optimization mitigates reward hacking and supports the use of bounded RL rewards.



(a) Comparison of sigmoid-like shaping functions. `tanh(centered)` denotes $r_{\text{RL}} = \frac{1}{M}\sum_{m=1}^{M} \tanh(r - r_{\text{ref}}^{m})$, `tanh(uncentered)` denotes $r_{\text{RL}} = \tanh(r)$, and `sigmoid(centered)` denotes PAR. Centered variants achieve higher win rates than their uncentered counterparts.



(b) Sigmoid-like functions used in the comparison: $\sigma_k(x) = 1/(1 + e^{-kx})$ with $k \in \{2, 3\}$, a fifth-order polynomial approximation (`poly_fitted`), and a shifted sigmoid (`SgFc`$= \sigma_3(x - 3)$).

Figure 4: Validation of the rapid-growth and gradual-saturation principle using centered and uncentered reward transformations.

KL coefficient from 0.01 to 0.1 lowers the proxy reward while improving the win rate. Imposing a smaller reward ceiling produces a similar effect, providing direct evidence that limiting reward magnitude improves training stability.

4.3 Principle 2: Rapid Initial Growth and Gradual Saturation

We next evaluate the principle that *the RL reward should grow rapidly at first and then gradually saturate*. We compare several sigmoid-like functions and their centered and uncentered variants, as shown in Figure 4a. Centered transformations consistently achieve higher win rates than their uncentered counterparts. Because the policy is initialized from the reference model, centered rewards are concentrated near zero at the beginning of training, where sigmoid-like functions have their largest gradients. This facilitates rapid early learning. By contrast, uncentered rewards may lie in regions with weaker or poorly controlled gradients. The right-shifted Slow-Grow-Fast-Converge (SgFc) function further illustrates this effect: its small initial gradient yields lower early win rates, while its abrupt transition and saturation are followed by reward hacking later in training (Figure 5). These results support the need for rapid initial growth followed by gradual saturation.

4.4 Comparison of Mitigation Methods

Test-set performance. We compare PAR with methods that either reshape the reward or modify the reward model. Figure 5 shows that vanilla PPO, which directly optimizes the unshaped proxy

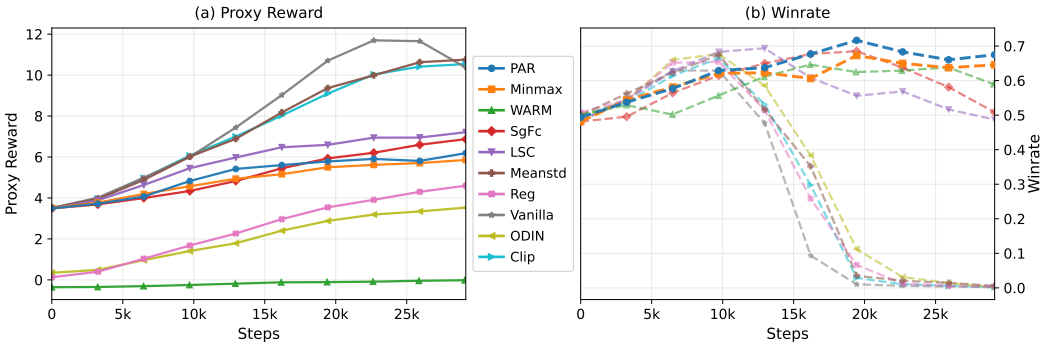


Figure 5: (Zoom in to view clearly) PPO training curves for different reward hacking mitigation methods using Gemma2-2B and UltraFeedback-Binarized. Solid lines denote proxy reward, and dashed lines denote test-set win rate against the SFT model. Vanilla PPO exhibits substantial reward hacking. In this setting, WARM, Minmax, and PAR mitigate the decline in win rate, with PAR achieving the highest win rate at the end of training.

	Method	AlpacaEval 2.0			MT-Bench		
		LC Win Rate (%) ↑	Win Rate (%) ↑	Length ↓	T1 ↑	T2 ↑	Overall ↑
	SFT	50.000	50.000	**899**	5.150	3.975	4.563
PPO training	Vanilla	0.100	0.370	2008	2.150	1.700	1.925
	WARM	60.670	63.170	1073	5.525	3.938	4.731
	ODIN	0.000	0.000	3672	1.375	1.338	1.356
	Reg	0.000	0.000	1868	1.513	1.388	1.450
	Meanstd	0.030	0.120	3183	1.713	1.300	1.506
	Clip	0.000	0.000	3096	1.288	1.225	1.256
	Minmax	66.980	70.930	1159	5.750	4.013	4.881
	LSC	47.560	53.790	1556	5.538	4.100	4.819
	PAR	**70.810**	**75.370**	1207	**5.813**	**4.313**	**5.063**

Table 1: Benchmark results for checkpoints selected after one epoch of PPO training; the SFT model is trained for two epochs, and the winrate is calculated against the SFT model. PAR achieves the highest AlpacaEval 2.0 win rates and MT-Bench scores among the shaping methods.

reward, exhibits severe reward hacking: the proxy reward continues to increase while the win rate declines. In this setting, ODIN, Reg, Meanstd, Clip, and LSC do not prevent this divergence. WARM, Minmax, and PAR provide varying degrees of mitigation over one epoch, with PAR achieving the highest final win rate.

Benchmark performance. To evaluate generalization beyond the training distribution, we select the checkpoint obtained after one epoch for each reward hacking mitigation method and evaluate it on AlpacaEval 2.0 and MT-Bench. As shown in Table 1, the benchmark results broadly agree with the test-set performance in Figure 5. Vanilla PPO suffers severe performance degradation. PAR and Minmax perform strongest overall, while WARM also yields substantial improvements over vanilla PPO. PAR achieves the highest LC and raw win rates on AlpacaEval 2.0 and the highest first-turn, second-turn, and overall scores on MT-Bench.

4.5 Abalation Study

We also do some ablation studies about the data efficiency, robustness, and calibration. The result shows that PAR is highly data-efficient, requires only one reference reward to achieve good performance (i.e., m=1), and is robust against prolonged training to 2 epochs. Besides, the winrate and the latent preference of the reward model calibrate well before reward hacking. The detailed result is shown at Appendix D.

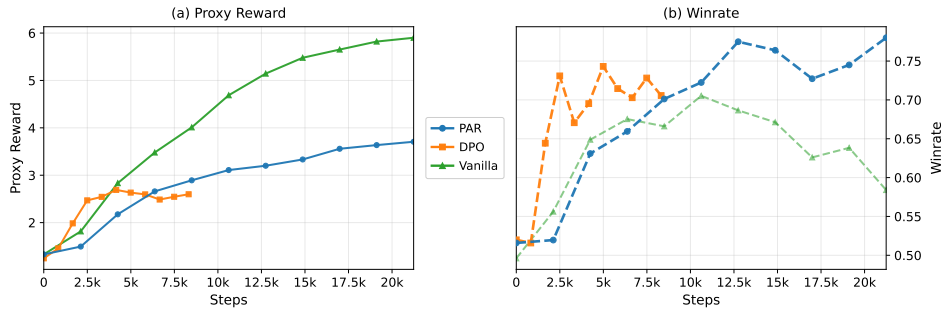


Figure 6: PPO training curves on the HH-RLHF dataset with Gemma2-2B as the base model.

5 Validation of PAR across Base Models and Optimization Algorithms

The preceding experiments focus on a single base model and RL algorithm. In this section, we extend our evaluation to multiple base models and optimization algorithms.

5.1 Experimental Setting

We use the HH-RLHF dataset (Bai et al., 2022) and evaluate four base models: Gemm2-2B, Gemma2-9B (Gemma Team, 2024), Qwen3-4B-Base (Team, 2025) and Llama3.1-8B (Grattafiori et al., 2024). We consider four optimization algorithms: PPO, Advantage Actor-Critic (A2C), Group Relative Policy Optimization (GRPO) (Shao et al., 2024), and Direct Preference Optimization (DPO) (Rafailov et al., 2023). To limit computational cost, we only compare PAR with the vanilla baseline, which directly uses the unshaped proxy reward. Because DPO optimizes offline preference pairs without an explicit reward signal, reward shaping is not applicable; we therefore include DPO as an additional reference method. We use the same test-set proxy-reward and win-rate metrics as in the preceding section, but do not conduct additional benchmark evaluations. GPT-5-nano (OpenAI, 2025) serves as the judge for computing win rates. We follow the training configuration described in the preceding section, with learning rates adjusted for each base model and optimization algorithm. Complete hyperparameter settings are provided in Appendix B.2.

5.2 Results

Figures 6, 7, 12, and 13 compare PPO training across the four base models. For Gemma2-2B, vanilla PPO exhibits reward hacking after approximately 10k steps: its proxy reward continues increasing, while its win rate gradually decreases, whereas PAR maintains a steadily improving win rate and outperforms DPO. Gemma2-9B shows a more severe vanilla failure, with reward hacking from 1.5k steps to 2.5k steps and the proxy reward collapsing after approximately 2.6k steps while the win rate falls to zero; PAR remains stable throughout training. A similar pattern occurs for Llama3.1-8B, where vanilla PPO reward hacking is followed by a sharp proxy-reward and win-rate collapse over approximately 2.6k–4.2k steps, while PAR preserves a win rate above 0.6. In contrast, Qwen3-4B-Base shows no evident reward collapse: both PAR and vanilla improve gradually. Overall, PAR substantially improves PPO robustness on the models susceptible to reward hacking and collapse, while retaining competitive performance on the more stable Qwen3-4B-Base. We also find that the DPO is already a simple and effective algorithm for preference datasets. More results on A2C and GRPO are shown in Appendix D.

6 Human Evaluation

To reduce reliance on a single LLM judge and introduce greater diversity in automated evaluation, we use DeepSeek-V3 for the Part-1 experiments and GPT-5-nano for the Part-2 experiments, and additionally conduct a small-scale human evaluation to assess the reliability of the LLM-as-a-judge results. For human evaluation, we focus on PPO training with PAR on HH-RLHF using Gemma2-2B

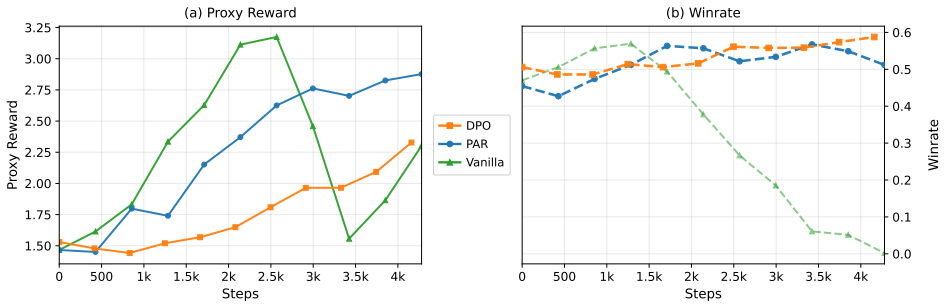


Figure 7: PPO training curves on the HH-RLHF dataset with Gemma2-9B as the base model.

Table 2: Human evaluation of PPO with PAR on the HH-RLHF dataset, using Gemma2-2B as the base model. Ties count as half a win when calculating win rates and as a separate category when calculating agreement metrics.

	Policy	**vs. other annotator**		**vs. GPT-5-nano**	
Rater	**win rate**	**Agr.**	**Cohen's κ**	**Agr.**	**Cohen's κ**
Annotator 1	70.0%	78.0%	0.527	75.5%	0.451
Annotator 2	71.5%	78.0%	0.527	75.0%	0.451
GPT-5-nano	68.5%	–	–	–	–

and construct an evaluation pool of 200 examples sampled from the test-set comparisons across all training checkpoints. Each example consists of a prompt, a policy response, a reference response, and GPT-5-nano's pairwise judgment indicating whether the policy or reference response is preferred. Two authors independently annotate all 200 examples. As reported in Table 2, the two annotators agree on 78.0% of the examples, with a Cohen's κ of 0.527. Their policy win rates are 70.0% and 71.5%, respectively, closely matching the GPT-5-nano win rate of 68.5%. The annotators agree with GPT-5-nano on 75.5% and 75.0% of the examples, with Cohen's κ values of 0.451 for both annotators.

7 Limitations

Although PAR effectively mitigates reward hacking and stabilizes RLHF training, it does not substantially improve peak performance, as measured by the win rate of the best checkpoint. Moreover, the dynamics of reward shaping—including the optimal rate of initial growth and the appropriate pace of saturation—remain insufficiently understood. Also, the datasets are all preference dialogue datasets where the simple and effective DPO already performs very well; a further study on the reasoning datasets remains as future research. And on the Qwen3-4B-Base model, the effectiveness is not obvious, indicating that PAR can work on some base models, not all.

8 Conclusion

We identify two principles for designing effective reward-shaping methods: the RL reward should be bounded, and it should exhibit rapid initial growth followed by gradual saturation. Guided by these principles, we introduce *Preference as Reward* (PAR) and establish two variance-reduction properties that contribute to stable RLHF training. Under the setting of the first set of experiments, across a broad range of reward-hacking mitigation methods, PAR achieves the strongest performance after one training epoch and maintains a high win rate after two epochs, demonstrating robustness under prolonged optimization. PAR is also highly data-efficient, requiring only a single reference reward to achieve strong performance. Under the setting of the second set of experiments, PAR demonstrates effectiveness across different base models and Critic-based RL algorithms.

AI USE STATEMENT

In this work, we used generative AI tools for [assist in the writing of proofs, implement methods]. We have not used generative AI tools for [help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, interpret results], and [Generate synthetic data sets, clean and reformat dataset, support qualitative and thematic data analysis] are not applicable to this work.

Additionally, we used generative AI tools for [create or modify scientific figures or images, create or edit software code, draft parts of a research paper, summarize or analyse existing literature, edit a research paper to improve readability, identify relevant literature, format references]. We have reviewed all AI-assisted work. We inspect the proofs carefully; LLM-generated code was verified and tested for correctness by 2 authors; we check all the references manually to avoid LLM hallucination.

We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work investigates reward shaping as a practical approach to mitigating reward hacking, a persistent challenge that can undermine the reliability of RLHF. Although PAR does not eliminate reward hacking under prolonged optimization, it delays its onset and widens the practical window for early stopping. We expect PAR and the design principles identified in this work to benefit future research on reward shaping, reward-model robustness, and stable policy optimization by providing a simple method and a systematic foundation for developing improved mitigation strategies. PAR may also help practitioners train RLHF models more reliably and efficiently. Nevertheless, reward shaping should complement, rather than replace, robust reward modeling, careful evaluation, human oversight, and application-specific safety measures.

REPRODUCIBILITY STATEMENT

All the details that are needed to reproduce the result are shown in Section 4 and Appendix B. The code will be released after the review process.

REFERENCES

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

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety, 2016. URL https://arxiv.org/abs/1606.06565.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022. URL https://arxiv.org/abs/2204.05862.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. doi: 10.2307/2334029. URL https://doi.org/10.2307/2334029.

Yuyan Bu, Liangyu Huo, Yi Jing, and Qing Yang. Beyond excess and deficiency: Adaptive length bias mitigation in reward models for RLHF. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3091–3098. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.169. URL https://aclanthology.org/2025.findings-naacl.169/.

Lichang Chen, Chen Zhu, Jiuhai Chen, Davit Soselia, Tianyi Zhou, Tom Goldstein, Heng Huang, Mohammad Shoeybi, and Bryan Catanzaro. ODIN: Disentangled reward mitigates hacking in RLHF. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 7935–7952. PMLR, 2024. URL https://proceedings.mlr.press/v235/chen24bn.html.

Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dcjtMYkpXx.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. UltraFeedback: Boosting language models with high-quality feedback, 2023. URL https://arxiv.org/abs/2310.01377.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. Safe RLHF: Safe reinforcement learning from human feedback. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/dd1577afd396928ed64216f3f1fd5556-Abstract-Conference.html.

Juntao Dai, Taiye Chen, Yaodong Yang, Qian Zheng, and Gang Pan. Mitigating reward over-optimization in RLHF via behavior-supported regularization. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1646e34971facbcda3727d1dc28ab635-Abstract-Conference.html.

DeepSeek-AI. DeepSeek-V3 technical report, 2024. URL https://arxiv.org/abs/2412.19437.

Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D'Amour, D. J. Dvijotham, Adam Fisch, Katherine Heller, Stephen R. Pfohl, Deepak Ramachandran, Peter Shaw, and Jonathan Berant. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=5u1GpUkKtG.

Arpad E. Elo. The rating of chessplayers, past and present, 1978. URL https://api.semanticscholar.org/CorpusID:142610973.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/5c48ff18e0a47baaf81d8b8ea51eec92-Abstract.html.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10835–10866. PMLR, 2023. URL https://proceedings.mlr.press/v202/gao23h.html.

Gemma Team. Gemma: Open models based on gemini research and technology, 2024. URL https://arxiv.org/abs/2403.08295.

Aaron Grattafiori et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. Deepseek-coder: When the large language model meets programming – the rise of code intelligence, 2024. URL https://arxiv.org/abs/2401.14196.

Zeyu Huang, Zihan Qiu, Zili Wang, Edoardo M. Ponti, and Ivan Titov. Post-hoc reward calibration: A case study on length bias. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5d50c76fdf75c24ece568fc84a7125fb-Abstract-Conference.html.

Yuu Jinnai, Tetsuro Morimura, Kaito Ariu, and Kenshi Abe. Regularized best-of-n sampling to mitigate reward hacking for language model alignment. In *ICML 2024 Workshop on Models of Human Feedback for AI Alignment*, 2024. URL https://openreview.net/forum?id=ewRlZPAReR.

Thomas Kwa, Drake Thomas, and Adrià Garriga-Alonso. Catastrophic goodhart: Regularizing RLHF with KL divergence does not mitigate heavy-tailed reward misspecification. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0466. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/1a8189929f3d7bd6183718f42c3f4309-Abstract-Conference.html.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. AlpacaEval: An automatic evaluator of instruction-following models. GitHub repository, 2023. URL https://github.com/tatsu-lab/alpaca_eval.

Tianqi Liu, Wei Xiong, Jie Ren, Lichang Chen, Junru Wu, Rishabh Joshi, Yang Gao, Jiaming Shen, Zhen Qin, Tianhe Yu, Daniel Sohn, Anastasiia Makarova, Jeremiah Liu, Yuan Liu, Bilal Piot, Abe Ittycheriah, Aviral Kumar, and Mohammad Saleh. RRM: Robust reward model training mitigates reward hacking. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=88AS5MQnmC.

Yuchun Miao, Sen Zhang, Liang Ding, Rong Bao, Lefei Zhang, and Dacheng Tao. InfoRM: Mitigating reward hacking in RLHF via information-theoretic reward modeling. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/f25d75fc760aec0a6174f9f5d9da59b8-Abstract-Conference.html.

Yuchun Miao, Sen Zhang, Liang Ding, Yuqi Zhang, Lefei Zhang, and Dacheng Tao. The energy loss phenomenon in RLHF: A new perspective on mitigating reward hacking. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 44076–44105. PMLR, 2025. URL https://proceedings.mlr.press/v267/miao25c.html.

Ted Moskovitz, Aaditya K. Singh, D. J. Strouse, Tuomas Sandholm, Ruslan Salakhutdinov, Anca D. Dragan, and Stephen McAleer. Confronting reward model overoptimization with constrained RLHF. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=gkfUvn0fLU.

OpenAI. GPT-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

OpenAI. GPT-5 system card, August 2025. URL https://openai.com/index/gpt-5-system-card/.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll 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. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Andrew Patterson, Victor Liao, and Martha White. Robust losses for learning value functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):6157–6167, 2023. doi: 10.1109/TPAMI.2022.3213503. URL https://doi.org/10.1109/TPAMI.2022.3213503.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

Alexandre Rame, Nino Vieillard, Léonard Hussenot, Robert Dadashi, Geoffrey Cideron, Olivier Bachem, and Johan Ferret. WARM: On the benefits of weight averaged reward models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 42048–42073. PMLR, 2024. URL https://proceedings.mlr.press/v235/rame24a.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

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, 2024. URL https://arxiv.org/abs/2402.03300.

Wei Shen, Xiaoying Zhang, Yuanshun Yao, Rui Zheng, Hongyi Guo, and Yang Liu. Improving reinforcement learning from human feedback using contrastive rewards, 2024. URL https://arxiv.org/abs/2403.07708.

Prasann Singhal, Tanya Goyal, Jiacheng Xu, and Greg Durrett. A long way to go: Investigating length correlations in RLHF, 2023. URL https://arxiv.org/abs/2310.03716.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9440–9450. Association for Computational Linguistics, 2024a. doi: 10.18653/v1/2024.acl-long.511. URL https://aclanthology.org/2024.acl-long.511/.

Zihao Wang, Chirag Nagpal, Jonathan Berant, Jacob Eisenstein, Alex D'Amour, Oluwasanmi Koyejo, and Victor Veitch. Transforming and combining rewards for aligning large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 51161–51176. PMLR, 2024b. URL https://proceedings.mlr.press/v235/wang24ay.html.

Jiaxin Wen, Ruiqi Zhong, Akbir Khan, Ethan Perez, Jacob Steinhardt, Minlie Huang, Samuel R. Bowman, He He, and Shi Feng. Language models learn to mislead humans via RLHF. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/b9a5a60573637f329b04d1beda4cd404-Abstract-Conference.html.

Rui Yang, Ruomeng Ding, Yong Lin, Huan Zhang, and Tong Zhang. Regularizing hidden states enables learning generalizable reward models for LLMs. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/71f7154547c748c8041505521ca433ab-Abstract-Conference.html.

Youliang Yuan, Qiuyang Mang, Jingbang Chen, Hong Wan, Xiaoyuan Liu, Junjielong Xu, Jen-tse Huang, Wenxuan Wang, Wenxiang Jiao, and Pinjia He. Curing miracle steps in LLM mathematical reasoning with rubric rewards. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18556–18577. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.844. URL https://aclanthology.org/2026.acl-long.844/.

Chuheng Zhang, Wei Shen, Li Zhao, Xuyun Zhang, Xiaolong Xu, Wanchun Dou, and Jiang Bian. Policy filtration for RLHF to mitigate noise in reward models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76023–76046. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhang25bq.html.

Xiaoying Zhang, Jean-François Ton, Wei Shen, Hongning Wang, and Yang Liu. Mitigating reward overoptimization via lightweight uncertainty estimation. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-2596. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/94bbcb744bbada8808fda05b9d9290d6-Abstract-Conference.html.

Kangwen Zhao, Jianfeng Cai, Jinhua Zhu, Ruopei Sun, Dongyun Xue, Wengang Zhou, Li Li, and Houqiang Li. Bias fitting to mitigate length bias of reward model in RLHF. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2912–2927. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.133. URL https://aclanthology.org/2026.acl-long.133/.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623, 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, et al. Secrets of RLHF in large language models part i: PPO, 2023b. URL https://arxiv.org/abs/2307.04964.

Symbol	Meaning
$\mathcal{D}$	Dataset
$x, y_w, y_l \sim \mathcal{D}$	Prompt, chosen response, rejected response in Dataset
π_θ	Policy model
π_{ref}	Reference model, also the SFT model
r_ϕ	Reward model
V_α	Critic model
$y \sim \pi_\theta(.\|x)$	The response generated by policy model for prompt x
$y_{\text{ref}} \sim \pi_{\text{ref}}(.\|x)$	Reference response, the response generated by reference model
$r = r_\phi(x, y)$	Proxy reward, the reward given directly by reward model
$r_{\text{ref}} = r_\phi(x, y_{\text{ref}})$	Reference reward, the proxy reward for reference response
$\mathcal{P}_\phi(y \succ y_{\text{ref}}\|x) = \text{sigmoid}(r - r_{\text{ref}})$	The hidden preference of reward model r_ϕ
$r_{\text{centered}} = r - r_{\text{ref}}$	Centered reward, the proxy reward subtracted by reference reward.
$r_{\text{RL}} = f(r_{\text{centered}})$	RL reward, the reward for RL training
$s_t = [x, y_{1,\ldots,t}]$	The state at position t
$a_t = y_{t+1}$	The Action taken at position t
$\hat{A}_t = \sum_{l=t}^{T} (\gamma\lambda)^{l-t}\delta_l$	The generalized advantage estimation (GAE)
$\delta_t = r_t + \gamma V_{\alpha_{\text{old}}}(s_{t+1}) - V_{\alpha_{\text{old}}}(s_t)$	The temporal difference (TD) error
$G_t = \sum_{l=t}^{T} \gamma^{l-t} r_l$	The return
$r_t = \begin{cases} r_{\text{RL}} - \eta \log \frac{\pi_\theta(y_t\|x, y_{<t})}{\pi_{\text{ref}}(y_t\|x, y_{<t})} & \text{if } t = T \\ -\eta \log \frac{\pi_\theta(y_t\|x, y_{<t})}{\pi_{\text{ref}}(y_t\|x, y_{<t})} & \text{if } t < T \end{cases}$	The per token reward

Table 3: Summary of notations.

A Notations

The definitions of the notations used in this paper are summarized in Table 3.

B Training Details

B.1 Part-1 Experiments

Dataset preprocessing. We conduct the Part-1 experiments on the UltraFeedback Binarized (Cui et al., 2023) dataset. To reduce noise and control sequence length, we retain only examples for which the prompt, chosen response, and rejected response each contain fewer than 512 tokens. We additionally require the annotated score of the chosen response to exceed that of the rejected response and remove examples in which either response contains the substring `"confidence"`. We deduplicate prompts across the training and test splits and restrict the test set to 256 examples. After preprocessing, the training set contains approximately 33,000 examples.

Compute and optimization schedule. All Part-1 experiments are conducted on eight NVIDIA A800 GPUs with 80 GB of memory each. We use Gemma2-2B (Gemma Team, 2024) as the base model. Unless otherwise specified, all training stages use linear learning-rate warm-up, increasing the learning rate from zero to its maximum value over the first 0.1 epoch.

SFT model. The supervised fine-tuned (SFT) model is initialized from the base model and trained on the chosen responses for two epochs with a learning rate of 5×10^{-6}. Gradients are clipped to a maximum norm of 10.

Reward model. The reward model is initialized from the base model by replacing its output head with a linear scalar head applied to the final hidden representation. It is trained for one epoch with a learning rate of 5×10^{-6} and achieves approximately 70% pairwise accuracy on the test set. Gradients are clipped to a maximum norm of 5.

For ODIN (Chen et al., 2024), we train two linear heads to predict length-related and quality-related rewards using the original ODIN objective. Only the quality head is used during RL training. For WARM (Rame et al., 2024), we train five reward models with learning rates of 3×10^{-6}, 4×10^{-6},

5×10^{-6}, 6×10^{-6}, and 7×10^{-6}, using a different random seed for each model, and average their weights. For Reg (Dai et al., 2024), we use the original regularized reward-model objective with a regularization coefficient of 0.005.

Policy and critic models. The policy is initialized from the SFT model and optimized on the training prompts for one epoch using PPO with a learning rate of 3×10^{-7}. The critic is initialized from the reward model and trained jointly with the policy for one epoch using a learning rate of 5×10^{-6}. Gradients for both models are clipped to a maximum norm of 5.

RL hyperparameters. Responses are sampled from the policy with a temperature of 0.9, top-k sampling with $k = 50$, top-p sampling with $p = 0.9$, and a length penalty of 2. The KL-penalty coefficient is 0.005, and PAR uses ten reference rewards per prompt by default. For PPO, we use a buffer size of 4, a clipping parameter of $\epsilon = 0.2$, a generalized advantage estimation parameter of $\lambda = 0.95$, and a discount factor of $\gamma = 1.0$. For GRPO, we use $\epsilon = 0.2$, a buffer size of 4, and a group size of 5.

B.2 Part-2 Experiments

Dataset preprocessing. We conduct the Part-2 experiments on the `helpful-base` subset of HH-RLHF (Bai et al., 2022). We apply the same sequence-length constraints as in Part-1, retaining examples whose prompts, chosen responses, and rejected responses each contain fewer than 512 tokens. We deduplicate prompts across the training and test splits and restrict the test set to 256 examples. After preprocessing, the training set contains approximately 43,000 examples.

Models and compute. We evaluate Gemma2-2B, Gemma2-9B (Gemma Team, 2024), and Llama3.1-8B (Grattafiori et al., 2024), Qwen3-4B-Base (Team, 2025). All Part-2 experiments are conducted on four NVIDIA H200 GPUs with 140 GB of memory each. We use the same learning-rate warm-up schedule as in Part-1.

SFT and reward models. The SFT and reward models are trained using the same objectives, initialization strategies, learning rates, gradient-clipping thresholds, and training durations as in Part-1. Although DPO does not require a reward model during training, we train the reward model for consistent proxy-reward evaluation across algorithms.

Policy optimization. For PPO and A2C, the policy and critic follow the initialization and optimization settings described for Part-1. GRPO and DPO do not use a separately trained critic. We adjust the learning rate and training duration for each combination of base model and optimization algorithm, as reported in Table 4.

C Evaluation

C.1 Test-Set Win Rate

We use DeepSeek-V3 and GPT-5-nano as pairwise judges to compare responses generated by the SFT and policy models on the test set. The complete system prompt and user-input format are provided in Figure 8.

To mitigate positional bias (Wang et al., 2024a), each response pair is evaluated twice with the presentation order reversed. Given responses A and B, the judge first evaluates them in the order A–B and then in the order B–A. In each evaluation, the preferred response receives a score of 1, the non-preferred response receives 0, and each response receives 0.5 in the case of a tie. We then aggregate the scores across the two orderings. The response with the higher aggregate score is considered the winner; if the aggregate scores are equal, each response receives one-half of a win. The policy win rate is computed by averaging its win count across all test examples.

Table 4: Learning rates and training durations for each base-model and optimization-algorithm combination on HH-RLHF.

Algorithm	Base Model	Policy Learning Rate	Critic Learning Rate	Epochs
PPO	Gemma2-2B	6×10^{-7}	6×10^{-6}	0.5
	Gemma2-9B	3×10^{-7}	3×10^{-6}	0.1
	Llama3.1-8B	2×10^{-7}	2×10^{-6}	0.1
	Qwen3-4B-Base	2×10^{-7}	2×10^{-6}	0.1
GRPO	Gemma2-2B	2×10^{-6}	-	0.2
	Gemma2-9B	3×10^{-7}	-	0.1
	Llama3.1-8B	2×10^{-6}	-	0.1
	Qwen3-4B-Base	4×10^{-6}	-	0.1
A2C	Gemma2-2B	3×10^{-7}	3×10^{-6}	0.5
	Gemma2-9B	1.5×10^{-7}	1.5×10^{-6}	0.1
	Llama3.1-8B	1×10^{-7}	1×10^{-6}	0.1
	Qwen3-4B-Base	1×10^{-7}	1×10^{-6}	0.1
DPO	Gemma2-2B	2×10^{-6}	-	0.2
	Gemma2-9B	3×10^{-7}	-	0.1
	Llama3.1-8B	2×10^{-6}	-	0.1
	Qwen3-4B-Base	4×10^{-6}	-	0.1

C.2 Benchmark Evaluation

We additionally evaluate the models on AlpacaEval 2.0 and MT-Bench, using DeepSeek-V3 as the judge. The reported metrics are defined below.

AlpacaEval 2.0.

- **LC Win Rate**: The length-controlled win rate estimates how often the policy response is preferred to the baseline response while adjusting for the influence of response length.
- **Win Rate**: The raw win rate measures the proportion of pairwise comparisons in which the policy response is preferred to the baseline response.
- **Length**: The average response length, as computed by the evaluation pipeline, provides an indication of model verbosity.

MT-Bench.

- **T1**: The first-turn score evaluates the relevance, coherence, and informativeness of the model's initial response on a 0–10 scale.
- **T2**: The second-turn score evaluates the model's ability to preserve conversational context and provide a consistent, high-quality follow-up response on the same scale.
- **Overall**: The overall score is the average of the first- and second-turn scores.

```
Please act as an impartial evaluator to assess the quality of two
    responses from different AI assistants to an incomplete dialogue
    between a user (<|user|>) and an AI assistant (<|assistant|>). The
    dialogue will be missing the last turn, and both Assistant-A (<
    Assistant-A response>) and Assistant-B (<Assistant-B response>) are
    expected to complete it. Focus your evaluation on the following five
    aspects:
1. Clarity and Relevance: Responses should be concise, directly
    addressing the question. They should use clear, natural language and
    remain on-topic.
2. Accuracy and Honesty: Responses must provide factual, truthful
    information. Disclose limitations or uncertainties when necessary.
3. Ethics and Appropriateness: Ensure the responses are free from harmful
    , offensive, or discriminatory content.
4. Engagement and Depth: Responses should be engaging, educational, and
    sufficiently detailed to comprehensively address the user question.
5. Structure and Creativity: Responses should be logically organized and
    show originality or adaptability when necessary.

Note: The quality of the responses should not be judged solely by their
    length. Both brevity and detail are important depending on the
    context of the question.
You will be given an incomplete dialogue (<question>) with the last turn
    left blank. Assistant-A (<Assistant-A response>) and Assistant-B (<
    Assistant-B response>) have each provided a response to complete the
    dialogue. Your task is to evaluate each response based on the five
    criteria above and provide a comparison.

Evaluation Format:
Assistant-A Response:
(Evaluate the quality of Assistant-A response based on the five aspects
    mentioned above.)
Assistant-B Response:
(Evaluate the quality of Assistant-B response based on the five aspects
    mentioned above.)
Comparison and Analysis:
Compare and contrast the responses from Assistant-A and Assistant-B to
    determine which one is more effective overall. Justify your reasoning
     clearly and concisely.

At the end, output the comparison result for both responses in the
    following format:
Better: X (X is A, B, or N, representing A is better, B is better, or
    both are of equal quality)
```

Listing 1: System Prompt For Winrate Evaluation on Test Set

```
<question>:
{user_question}
<Assistant-A response>:
{policy_response}
<Assistant-B response>:
{sft_response}
```

Listing 2: User Input Template For Winrate Evaluation on Test Set

Figure 8: System prompt for win-rate evaluation

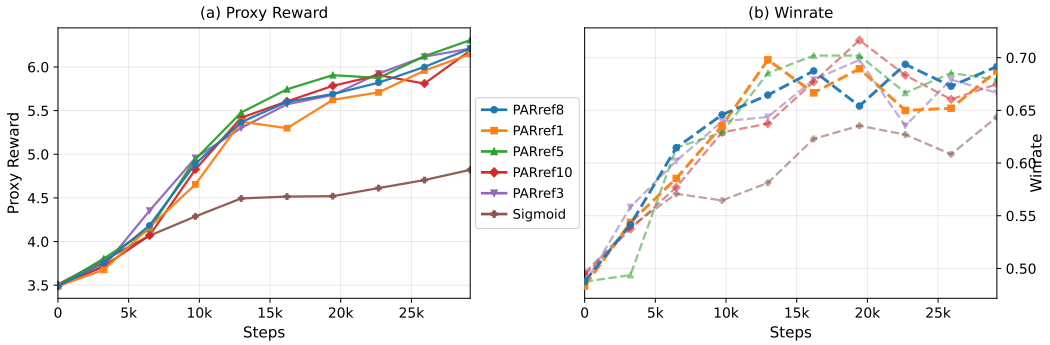


Figure 9: PPO training curves with different numbers of reference rewards. For example, `PAR5` denotes $r_{\mathrm{RL}} = \frac{1}{5}\sum_{m=1}^{5} \sigma(r - r_{\mathrm{ref}}^{m})$. A single reference reward achieves a win rate comparable to those obtained with additional references.

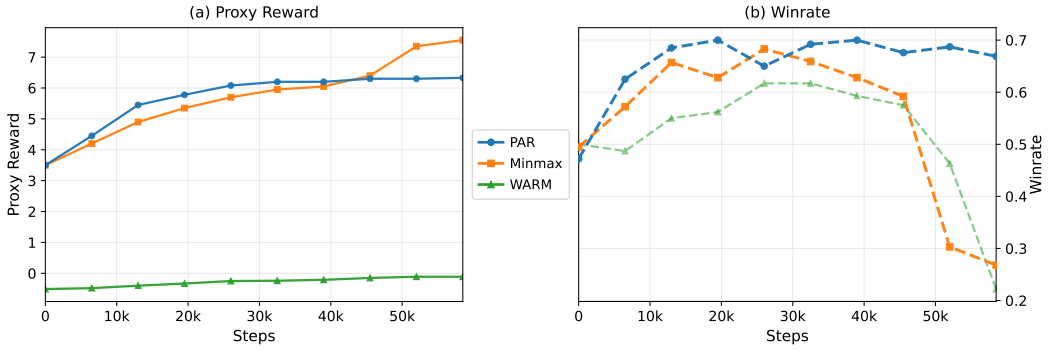


Figure 10: PPO training curves over two epochs for PAR, Minmax, and WARM. PAR maintains a higher and more stable win rate under prolonged optimization.

D Additional Results

This section presents additional results for the Part-1 ablation study and the Part-2 training curves across base models and optimization algorithms.

D.1 Data Efficiency and Robustness

Number of reference rewards. By default, PAR uses ten reference rewards per prompt. To determine whether fewer references are sufficient, we vary this number from one to ten. Figure 9 shows that all PAR variants follow similar proxy-reward and win-rate trajectories, indicating that a single reference reward is sufficient for effective training. In contrast, the uncentered sigmoid baseline, $r_{\mathrm{RL}} = \sigma(r)$, performs substantially worse than PAR with one reference reward. Thus, reference centering is important, but it can be estimated from only one reference response.

Robustness under extended training. We next evaluate robustness under prolonged reward optimization. We select the three methods with the highest AlpacaEval 2.0 win rates—PAR, Minmax, and WARM—and extend PPO training from one to two epochs. We define robustness operationally as the ability to maintain a high win rate as proxy-reward optimization continues. As shown in Figure 10, the win rates of Minmax and WARM decline during the second epoch, indicating renewed reward overoptimization. In contrast, PAR maintains a comparatively high and stable win rate throughout training. These results suggest that PAR delays the onset of reward hacking and provides a wider practical window for early stopping.

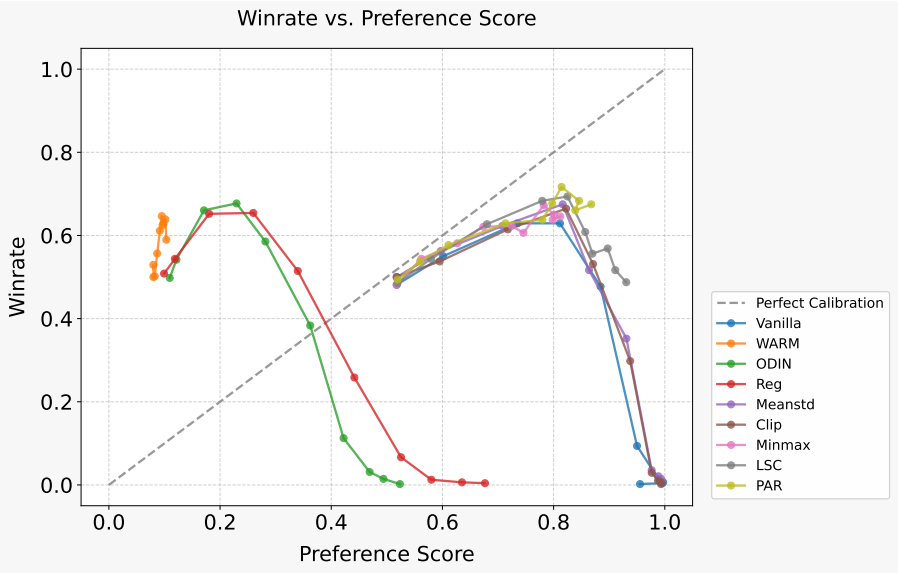


Figure 11: Relationship between latent preference score and the win rate.

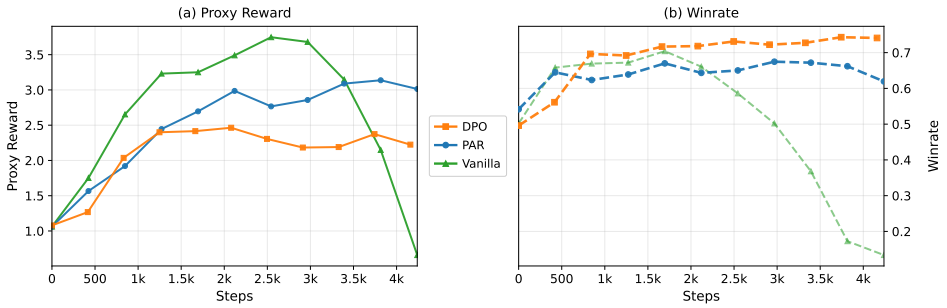


Figure 12: PPO training curves on the HH-RLHF dataset with Llama3.1-8B as the base model.

D.2 Calibration of Preference Scores

We further investigate whether the reward model's latent preference score remains predictive of externally evaluated response quality. As shown in Figure 11, preference scores initially track the win rates judged by DeepSeek-V3 across reward-shaping methods. However, this relationship weakens once the preference score exceeds approximately 0.8, suggesting that very high reward-model scores are poorly calibrated indicators of actual response quality. By bounding the transformed reward, PAR limits optimization within this poorly calibrated high-score regime and maintains a closer correspondence between preference score and observed win rate. In contrast, methods that directly modify the reward model exhibit weaker calibration overall. This result provides additional evidence that controlling extreme reward values improves the reliability of the RL training signal.

D.3 Results on PPO

Figure 12 shows that the vanilla PPO using unshaped reward experiences both reward hacking (From 1.6K steps to 2.5K steps) and reward collapse (From 2.5K steps to 4K steps) problems, while PAR shaping training steadily on the Llama3.1-8B model. Figure 13 shows that PAR shaping seems not too effective; we hypothesize that this is due to the Qwen3-4B-Base model undergoing R1-like training before, so its thinking and reasoning ability is better than previous models like Gemma2 and LLama3.1, and not sensitive to the reward shaping strategies.

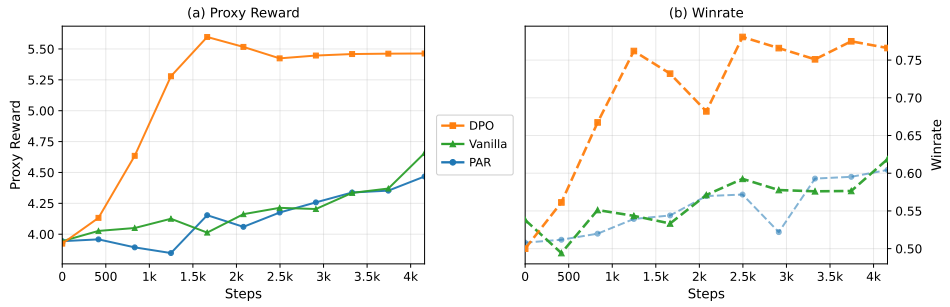


Figure 13: PPO training curves on the HH-RLHF dataset with Qwen3-4B-Base as the base model.

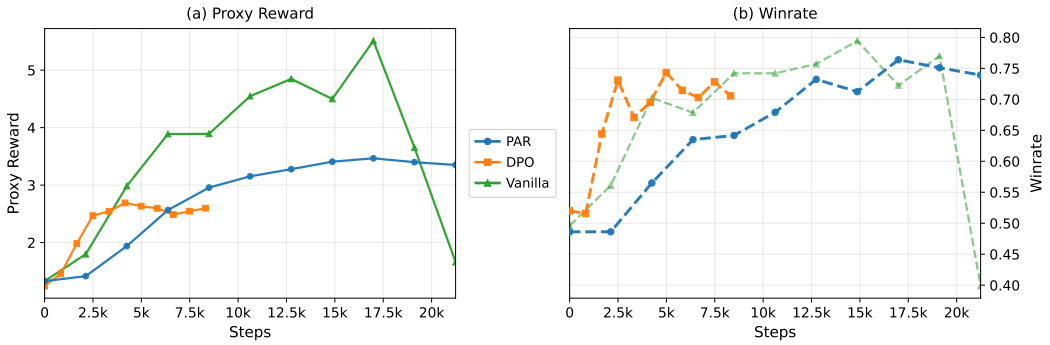


Figure 14: A2C training curves on the HH-RLHF dataset with Gemma2-2B as the base model.

D.4 Results on A2C and GRPO

Gemma2-2B. As shown in Figures 14 and 15, vanilla A2C exhibits the clearest reward-hacking and collapse behavior. Between approximately 15k and 17k steps, its proxy reward increases from about 4.5 to 5.5 while the win rate decreases, indicating reward hacking. This is followed by a severe collapse over 17k–21k steps, with the proxy reward falling to about 1.7 and the win rate eventually dropping to 0.40. PAR prevents this failure and maintains a win rate near 0.74. Under GRPO, both PAR and vanilla remain stable, with only mild late-stage reward–win-rate divergence, while DPO achieves competitive but slightly lower final performance.

Gemma2-9B. Figures 16 and 17 show that vanilla A2C increases the proxy reward to approximately 3.1 by 3k steps without a corresponding win-rate improvement, suggesting reward hacking. A catastrophic collapse then occurs at approximately 3.4k–3.8k steps, where the vanilla win rate falls from around 0.5 to zero. PAR avoids this collapse, although its rising proxy reward is accompanied

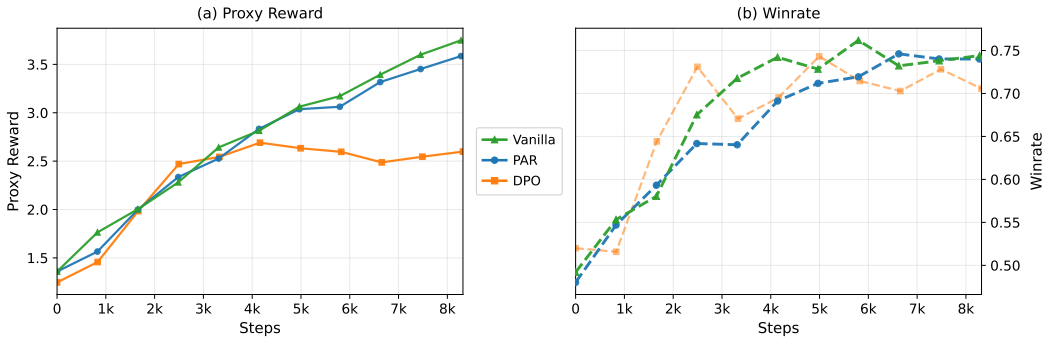


Figure 15: GRPO training curves on the HH-RLHF dataset with Gemma2-2B as the base model.

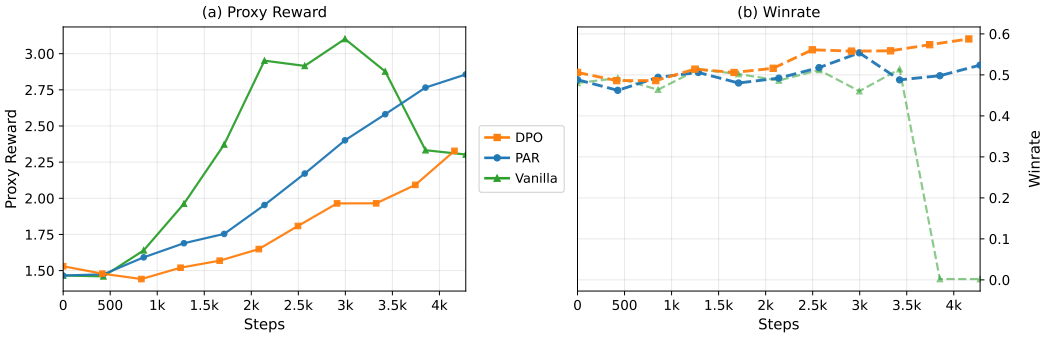


Figure 16: A2C training curves on the HH-RLHF dataset with Gemma2-9B as the base model.

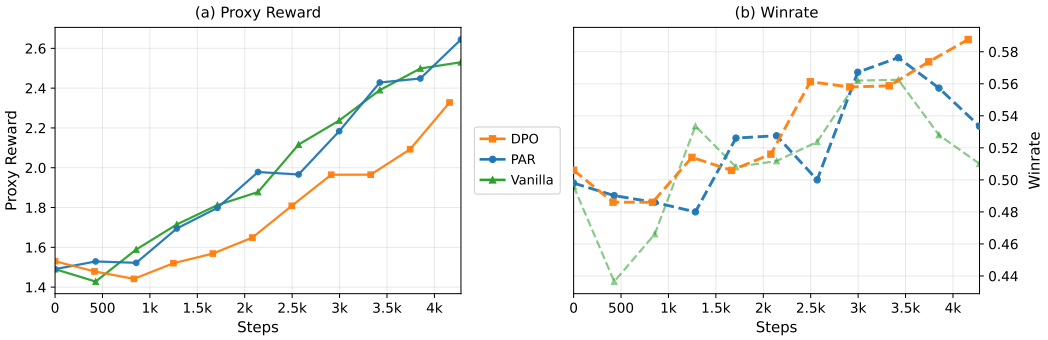


Figure 17: GRPO training curves on the HH-RLHF dataset with Gemma2-9B as the base model.

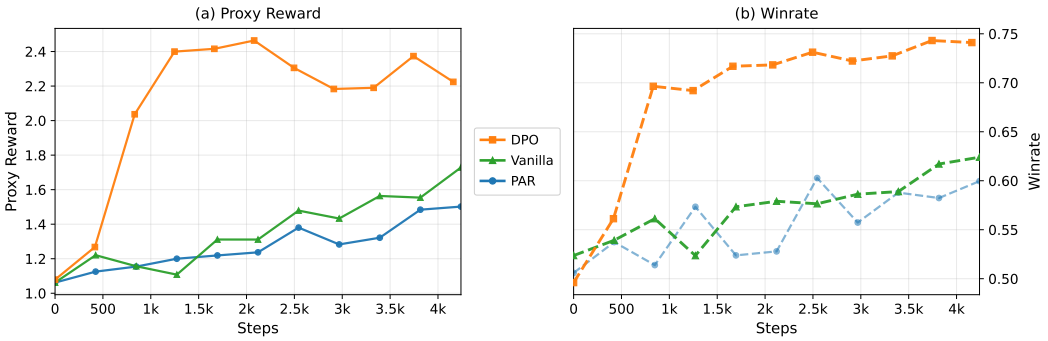


Figure 18: A2C training curves on the HH-RLHF dataset with Llama3.1-8B as the base model.

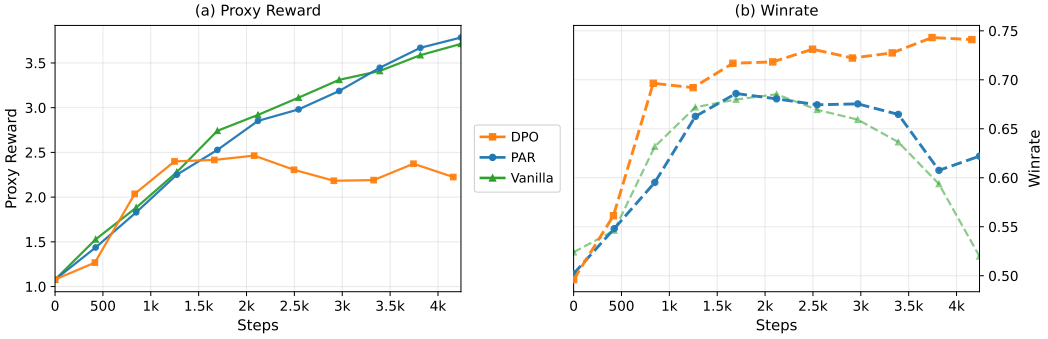


Figure 19: GRPO training curves on the HH-RLHF dataset with Llama-3.1-8B as the base model.

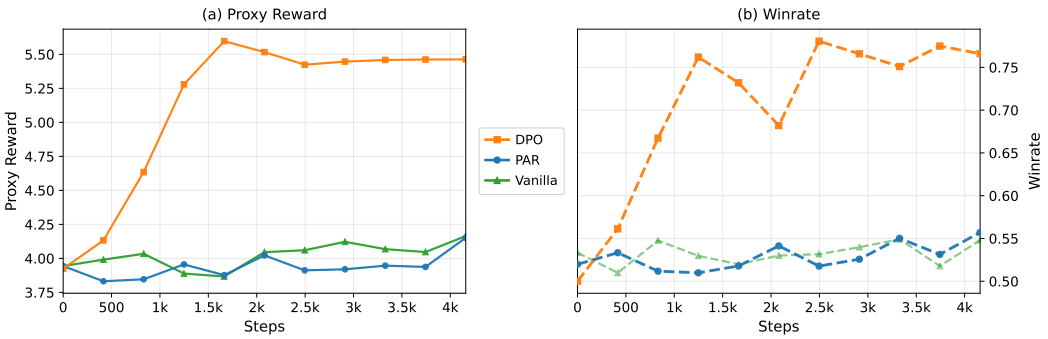


Figure 20: A2C training curves on the HH-RLHF dataset with Qwen3-4B-Base as the base model.

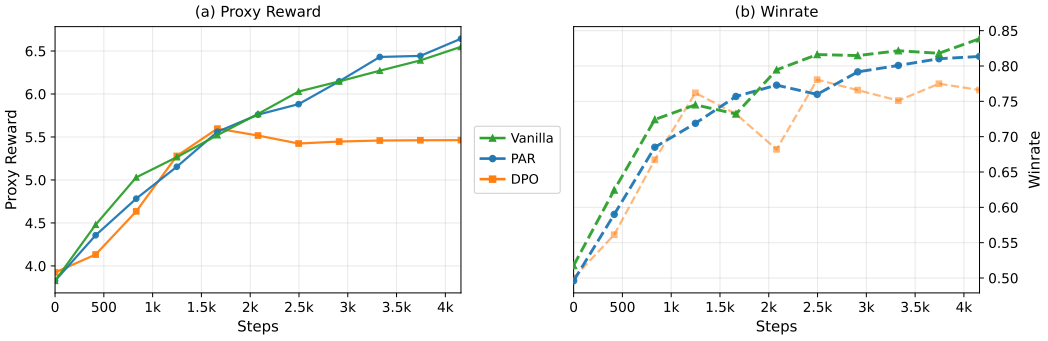


Figure 21: GRPO training curves on the HH-RLHF dataset with Qwen3-4B-Base as the base model.

by only a modest improvement in win rate. GRPO is more stable than A2C, but PAR and vanilla still show mild late-stage hacking around 3.4k–4.2k, when their rewards increase while their win rates decline. DPO provides the best terminal win rate and the strongest reward–performance alignment for this model.

Llama3.1-8B. For Llama3.1-8B, Figure 18 shows no clear reward hacking or collapse under A2C: PAR and vanilla remain stable, although both underperform DPO in win rate. In contrast, Figure 19 reveals sustained reward hacking under GRPO. From approximately 1.7k to 4.2k steps, the vanilla proxy reward continues to increase while its win rate falls from about 0.68 to 0.52. PAR exhibits the same divergence but reduces its severity, ending with a win rate near 0.62. DPO achieves a lower proxy reward but a substantially higher win rate of approximately 0.74, demonstrating better alignment with external performance.

Qwen3-4B-Base. Figures 20 and 21 show no sustained reward hacking or reward collapse for Qwen3-4B-Base. Under A2C, PAR and vanilla remain stable but improve little, whereas DPO rapidly achieves a higher proxy reward and a win rate of approximately 0.77. Under GRPO, both PAR and vanilla improve consistently in proxy reward and win rate, reaching terminal win rates of approximately 0.81 and 0.84, respectively. Only brief local divergences are observed, and neither develops into a collapse. Thus, GRPO is particularly stable for this model, while vanilla slightly outperforms PAR in final win rate.

E Comparison of Some Reward Shaping Techniques

Previous work has explored various approaches to reward transformation in RLHF. Shen et al. (2024) proposed contrastive rewards that share similarities with our method. Their approach employs a reference model to generate multiple baseline responses (y_{ref}^m), computing rewards as:

$$r_{\text{RL}} = \frac{1}{M}\sum_{m=1}^{M}\left[r(x,y) - r(x,y_{\text{ref}}^m)\right].$$

This method primarily focuses on enhancing PPO through contrastive learning against reference responses.

Ahmadian et al. (2024) revisited the REINFORCE algorithm for RLHF, adopting a response-centric approach analogous to GRPO. Their formulation centers on rewards using multiple sampled responses:

$$g = \frac{1}{M}\sum_{i=1}^{M}\left[r(x,y_i) - \frac{1}{M-1}\sum_{j\neq i} r(x,y_j)\right]\nabla \log \pi_\theta(y_i|x),$$

where $y_1,\ldots,y_M \overset{\text{i.i.d.}}{\sim} \pi_\theta(\cdot|x)$. This work emphasizes algorithmic simplicity and stability by replacing PPO with REINFORCE.

Wang et al. (2024b) introduced a log-sigmoid reward transformation for multi-reward integration:

$$r_{\text{RL}} = \text{log-sigmoid}\left[r(x,y) - r(x,y_{\text{ref}}^{.85})\right].$$

Where the $r_{\text{ref}}^{.85}$ denotes the 85th percentile of the normal distribution, built from the mean and variance of the reference rewards. Their method specifically addresses the challenge of effectively combining multiple reward signals during training.

In this work, we present Preference As Reward (PAR), which applies a sigmoid transformation to mitigate reward hacking:

$$r_{\text{RL}} = \frac{1}{M}\sum_{m=1}^{M}\text{sigmoid}\left(r(x,y) - r(x,y_{\text{ref}}^m)\right).$$

Our approach specifically targets reward hacking while maintaining stable policy optimization.

F Theoretical Analysis

We give a principled justification for *Preference As Reward* (PAR) here.

F.1 Bounded Rewards Reduce Return Variance

PAR has the key property $|r_l| < 1, \forall l$ if we choose a sufficiently small hyper-parameter η in Section 3; here r_l is the reward at position l. Let $G_t = \sum_{l=t}^{T}\gamma^{l-t}r_l$ be the return with discount $\gamma \in [0,1)$.

Theorem F.1 (Return Variance Bound)**.** *For* any *trajectory and any* γ, $\text{Var}\big[G_t\big] \leq \frac{1}{(1-\gamma)^2}$.

Proof. Since $r_l \in (-1,1)$ and $\gamma \in [0,1)$, we have

$$|G_t| = \Big|\sum_{k=0}^{T-t}\gamma^k r_{t+k}\Big| \leq \sum_{k=0}^{T-t}\gamma^k|r_{t+k}| \tag{1}$$

$$\leq \sum_{k=0}^{T-t}\gamma^k \leq \sum_{k=0}^{\infty}\gamma^k = \frac{1}{1-\gamma}. \tag{2}$$

According to Popoviciu's inequality, which states that for any random variable X with support inside $[a, b]$, $\text{Var}[X] \leq \frac{1}{4(b-a)^2}$ We have:

$$\text{Var}[G_t] \;\leq\; \frac{(\frac{2}{1-\gamma})^2}{4} \;= \frac{1}{(1-\gamma)^2}.$$

□

Theorem 3.1 explains the smaller and more stable critic loss in Figure 2 also highlights the importance of Design Principle 1 from a theoretical view.

F.2 The Justification of the Sigmoid Function

Probability space and notation. Fix policy parameters θ. A single stochastic gradient sample is generated by the following random process:

(i) draw a prompt $x \sim \mathcal{D}$, and a reference response $y_{\text{ref}} \sim \pi_{\text{ref}}(\cdot|x)$;

(ii) draw a response $y \sim \pi_\theta(\cdot \mid x)$;

Denote

$$z(x, y, y_{\text{ref}}) := r_\phi(x, y) - r_\phi(x, y_{\text{ref}})$$

We allow an alternative shaped reward $\tilde{r}$ that may further depend on additional randomness ω (e.g., sampling noise), i.e., $\tilde{r} = \tilde{r}(x, y, \omega)$. Unless otherwise specified, all expectations and variances, $\mathbb{E}[\cdot]$ and $\text{Var}(\cdot)$, are taken over the joint randomness of

$$x \sim \mathcal{D}, \quad y \sim \pi_\theta(\cdot \mid x), \quad y_{\text{ref}} \sim \pi_{\text{ref}}(\cdot \mid x), \quad \omega$$

With θ treated as fixed. We use $\mathbb{E}[\cdot \mid x, y, y_{\text{ref}}]$ and $\text{Var}(\cdot \mid x, y, y_{\text{ref}})$ to denote conditional expectation/variance where x, y and y_{ref} are fixed and the remaining randomness is over ω only.

Theorem F.2 (Sigmoid is the minimum-variance unbiased shaping under logistic preference noise)**.** *Define the REINFORCE estimator $\tilde{g} = \nabla_\theta \log \pi_\theta(y \mid x)\, \tilde{r}$.*

where $\tilde{r}$ may use any additional randomness but must satisfy the pointwise unbiasedness constraint

$$\mathbb{E}[\tilde{r} \mid x, y, y_{\text{ref}}] = \sigma(z(x, y, y_{\text{ref}})).$$

Then the unique minimum-variance estimator in this class is

$$g_\sigma := \nabla_\theta \log \pi_\theta(y \mid x)\, \sigma(z(x, y, y_{\text{ref}})),$$

and for every estimator $\tilde{g}$,

$$\textit{Var}(\tilde{g}) \;\geq\; \textit{Var}(g_\sigma)\,.$$

Proof. Condition on (x, y, y_{ref}). Then $S := \nabla_\theta \log \pi_\theta(y \mid x)$ is deterministic, while all randomness in $\tilde{g}$ comes from $\tilde{r}$. We apply the law of total variance:

$$\text{Var}(\tilde{g}) = \text{Var}\big(\mathbb{E}[\tilde{g} \mid x, y, y_{\text{ref}}]\big) + \mathbb{E}\big[\text{Var}(\tilde{g} \mid x, y, y_{\text{ref}})\big].$$

By the unbiasedness constraint, $\mathbb{E}[\tilde{r} \mid x, y, y_{\text{ref}}] = \sigma(z(x, y, y_{\text{ref}}))$, hence

$$\begin{aligned}\mathbb{E}[\tilde{g} \mid x, y, y_{\text{ref}}] &= S\, \mathbb{E}[\tilde{r} \mid x, y, y_{\text{ref}}] \\ &= S\, \sigma(z(x, y, y_{\text{ref}})) = \mathbb{E}[g_\sigma \mid x, y, y_{\text{ref}}].\end{aligned}$$

Therefore,

$$\text{Var}\big(\mathbb{E}[\tilde{g} \mid x, y, y_{\text{ref}}]\big) = \text{Var}\big(\mathbb{E}[g_\sigma \mid x, y, y_{\text{ref}}]\big) = \text{Var}(g_\sigma),$$

because g_σ is deterministic given (x, y, y_{ref}) and thus $\text{Var}(g_\sigma \mid x, y, y_{\text{ref}}) = 0$.

Moreover, still conditioning on (x, y, y_{ref}),

$$\text{Var}(\tilde{g} \mid x, y, y_{\text{ref}}) = \text{Var}\big(S\, \tilde{r} \mid x, y, y_{\text{ref}}\big) = (S)^2 \text{Var}(\tilde{r} \mid x, y, y_{\text{ref}}) \;\geq\; 0.$$

Putting these together gives

$$\text{Var}(\tilde{g}) = \text{Var}(g_\sigma) + \mathbb{E}\big[(S)^2 \text{Var}(\tilde{r} \mid x, y, y_{\text{ref}})\big] \;\geq\; \text{Var}(g_\sigma).$$

Equality holds iff $\text{Var}(\tilde{r} \mid x, y, y_{\text{ref}}) = 0$ almost surely, i.e., $\tilde{r}$ is deterministic given (x, y, y_{ref}). Combined with $\mathbb{E}[\tilde{r} \mid x, y, y_{\text{ref}}] = \sigma(z(x, y, y_{\text{ref}}))$, this forces $\tilde{r} = \sigma(z(x, y, y_{\text{ref}}))$ a.s., hence the minimizer is unique. □

Theorem 3.2 justifies our selection of sigmoid-like functions for bounding RL rewards, as the sigmoid's properties minimize policy gradient variance.

G CASE STUDY

We conduct a qualitative analysis of responses generated during vanilla PPO training and identify three recurring reward-hacking patterns: (1) inserting superficially high-reward words or phrases into otherwise incoherent text; (2) producing excessively verbose responses; and (3) generating content that is topically related to the prompt but fails to address the user's actual intent. Representative examples are shown in Figure 22. In the first example, the policy combines technical and aerospace-related terms into a largely meaningless response. In the second, the response discusses Caribbean travel at length but does not provide the requested price comparison with Las Vegas, illustrating both excessive verbosity and failure to follow the user's intent.

```
Prompt:
<|user|> I am going to give a PPT presentation on VLSI in aerospace and
    defence. What I need is an introduction slide, an overview slide, and
     any related content you think should be included. <|assistant|>

Policy model response:
In enchanting architectural mosaics, fuse captivating cosmic threads
    VELCRO-ing silicon's sacred architectural realm with adaptive
    aeronautical nebulae, sculpting celestial gardens for efficient
    holographic compass landmarks. Seamless Integrated Silicon Graphics (
    ISG) ... [response truncated]
```

Listing 3: Gemma2-2B trained on UltraFeedback Binarized

```
Prompt:
<|user|> Can you help compare the price of traveling to the Caribbean
    with the price of a trip to Las Vegas? <|assistant|>

Policy model response:
Sure! Caribbean travel is a wonderful choice for a vacation. Caribbean
    destinations offer some of the most beautiful beaches in the world,
    and the crystal-clear waters of the Caribbean Sea provide amazing
    snorkeling and diving experiences. Caribbean resorts and hotels offer
     amenities such as fine dining, swimming pools, spa services, and
    outdoor activities, including sailing, hiking, golfing, and horseback
     riding ... [response truncated]
```

Listing 4: Gemma2-2B trained on HH-RLHF

Figure 22: Representative examples of reward hacking during vanilla PPO training. The first response exhibits incoherent keyword stuffing, whereas the second is verbose and fails to address the requested price comparison.

H TRAINING ALGORITHMS

We consider four policy-optimization algorithms: PPO, GRPO, A2C, and DPO. The first three optimize rewards assigned to on-policy responses, whereas DPO learns directly from an offline preference dataset. The following algorithms summarize our implementations and the integration of reward shaping.

H.1 PPO TRAINING

Proximal Policy Optimization (PPO) (Schulman et al., 2017) is an online actor–critic algorithm. Given a prompt, the policy generates a response, the reward model assigns a scalar score, and the

Algorithm 1 PPO training

Require: SFT policy π_{sft}, reward model r_ϕ, prompt set $\mathcal{D}$
Ensure: Aligned policy π_θ
Initialize policy $\pi_\theta \leftarrow \pi_{\text{sft}}$
Initialize reference policy $\pi_{\text{ref}} \leftarrow \pi_{\text{sft}}$
Initialize critic $V_\alpha \leftarrow r_\phi$
Initialize replay buffer $\mathcal{B} \leftarrow [\,]$
for $x \in \mathcal{D}$ **do**
  $b_{\text{new}} \leftarrow \textsc{BuildPPOBatch}(x, \pi_\theta, \pi_{\text{ref}}, V_\alpha, r_\phi)$
  $b \leftarrow \textsc{BufferSubstitute}(\mathcal{B}, b_{\text{new}})$
  if $b = \texttt{None}$ **then**
    continue ▷ fill the buffer before the first update
  end if
  $\mathcal{L}_{\text{PPO}}(\theta), \mathcal{L}_V(\alpha) \leftarrow \textsc{ComputePPOLoss}(b, \pi_\theta, V_\alpha)$
  $\theta \leftarrow \theta - \eta_\pi \nabla_\theta \mathcal{L}_{\text{PPO}}(\theta)$
  $\alpha \leftarrow \alpha - \eta_V \nabla_\alpha \mathcal{L}_V(\alpha)$
end for
return π_θ

Algorithm 2 Construction of a PPO batch

Require: Prompt x, policy π_θ, reference policy π_{ref}, critic V_α, reward model r_ϕ
Ensure: PPO batch b
Sample policy response $y \sim \pi_\theta(\cdot \mid x)$
if the shaping method requires reference rewards **then**
  Sample $\{y_{\text{ref}}^m\}_{m=1}^M \sim \pi_{\text{ref}}(\cdot \mid x)$
  $r_{\text{ref}}^m \leftarrow r_\phi(x, y_{\text{ref}}^m)$ for $m = 1, \ldots, M$
end if
$r \leftarrow r_\phi(x, y)$
$r_{\text{RL}} \leftarrow \textsc{RewardReshape}(r, \{r_{\text{ref}}^m\}_{m=1}^M, \text{len}(y), \textsf{mode})$
Tokenize (x, y) into $\{(s_t, a_t)\}_{t=1}^T$
$k_t \leftarrow \log \pi_\theta(a_t \mid s_t) - \log \pi_{\text{ref}}(a_t \mid s_t)$
$r_t \leftarrow -\beta_{\text{KL}} k_t + \mathbb{I}[t = T]\, r_{\text{RL}}$ ▷ terminal reward and per-token KL penalty
$V_t^{\text{old}} \leftarrow V_\alpha(s_t)$
Compute GAE advantages $\hat{A}_t$ and return targets G_t from $\{r_t, V_t^{\text{old}}\}_{t=1}^T$
$b \leftarrow \{\log \pi_\theta(a_t \mid s_t), G_t, \hat{A}_t, V_t^{\text{old}}, s_t, a_t\}_{t=1}^T$
return b

policy and critic are jointly updated to maximize the resulting RL objective. To improve training stability, our implementation uses advantage normalization (Zheng et al., 2023b), clipped value loss (Patterson et al., 2023), a replay buffer (Eysenbach et al., 2019), a per-token KL penalty, and a response-length penalty. Algorithm 1 summarizes the training procedure.

H.2 GRPO Training

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) avoids training a separate critic by estimating advantages from groups of responses generated for the same prompt. For a group of G shaped rewards, the response-level advantage is

$$\hat{A}_i = \frac{r_{\text{RL}}^i - \text{mean}\Big(\{r_{\text{RL}}^j\}_{j=1}^G\Big)}{\text{std}\Big(\{r_{\text{RL}}^j\}_{j=1}^G\Big) + \varepsilon_A},$$

where ε_A is a small constant for numerical stability. The same response-level advantage is assigned to every token in the response. Algorithm 4 summarizes our GRPO implementation.

Algorithm 3 PPO policy and critic losses

Require: PPO batch b, policy π_θ, critic V_α
Ensure: $\mathcal{L}_{\mathrm{PPO}}(\theta)$ and $\mathcal{L}_V(\alpha)$
1: Extract $\{\log \pi_{\theta_{\mathrm{old}}}(a_t \mid s_t), G_t, \hat{A}_t, V_t^{\mathrm{old}}, s_t, a_t\}$ from b
2: Normalize $\hat{A}_t$ within the batch
3: $\rho_t(\theta) \leftarrow \exp(\log \pi_\theta(a_t \mid s_t) - \log \pi_{\theta_{\mathrm{old}}}(a_t \mid s_t))$
4: $\mathcal{L}_{\mathrm{PPO}}(\theta) \leftarrow -\hat{\mathbb{E}}_t\Big[\min\Big(\rho_t(\theta)\hat{A}_t,\ \mathrm{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_t\Big)\Big]$
5: $\widetilde{V}_t(\alpha) \leftarrow \mathrm{clip}\big(V_\alpha(s_t), V_t^{\mathrm{old}} - \delta, V_t^{\mathrm{old}} + \delta\big)$
6: $\mathcal{L}_V(\alpha) \leftarrow \hat{\mathbb{E}}_t\left[\max\left(\left(V_\alpha(s_t) - G_t\right)^2,\ \left(\widetilde{V}_t(\alpha) - G_t\right)^2\right)\right]$
7: **return** $\mathcal{L}_{\mathrm{PPO}}(\theta), \mathcal{L}_V(\alpha)$

Algorithm 4 GRPO training

Require: SFT policy π_{sft}, reward model r_ϕ, prompt set $\mathcal{D}$
Ensure: Aligned policy π_θ
Initialize policy $\pi_\theta \leftarrow \pi_{\mathrm{sft}}$
Initialize reference policy $\pi_{\mathrm{ref}} \leftarrow \pi_{\mathrm{sft}}$
Initialize replay buffer $\mathcal{B} \leftarrow [\,]$
for $x \in \mathcal{D}$ **do**
  $b_{\mathrm{new}} \leftarrow$ BUILDGRPOBATCH$(x, \pi_\theta, \pi_{\mathrm{ref}}, r_\phi)$
  $b \leftarrow$ BUFFERSUBSTITUTE$(\mathcal{B}, b_{\mathrm{new}})$
  if $b =$ `None` **then**
    continue
  end if
  $\mathcal{L}_{\mathrm{GRPO}}(\theta) \leftarrow$ COMPUTEGRPOLOSS(b, π_θ)
  $\theta \leftarrow \theta - \eta_\pi \nabla_\theta \mathcal{L}_{\mathrm{GRPO}}(\theta)$
end for
return π_θ

H.3 A2C TRAINING

Advantage Actor–Critic (A2C) is an online actor–critic method. In our implementation, A2C uses an unclipped policy-gradient objective and does not use the replay buffer or clipped value loss employed by PPO. Algorithm 7 summarizes the training procedure.

H.4 DPO TRAINING

Direct Preference Optimization (DPO) (Rafailov et al., 2023) is an offline preference-optimization method that does not require an explicit reward model or critic during training. Given a prompt x, a preferred response y_w, and a rejected response y_l, DPO increases the relative likelihood of y_w while regularizing the policy against a fixed reference model.

H.5 SHARED COMPONENTS

Replay buffer. PPO and GRPO use the replay-buffer procedure in Algorithm 11. The buffer is persistent across calls. During warm-up, new batches are stored without performing an update. Once the buffer is full, a previously stored batch is sampled for training and replaced by the current batch.

Reward transformation. Algorithm 12 summarizes the scalar reward transformations used in our experiments. Methods such as WARM, ODIN, and Reg modify the reward model itself and therefore use the vanilla branch after producing their respective proxy rewards.

Algorithm 5 Construction of a GRPO batch

Require: Prompt x, policy π_θ, reference policy π_{ref}, reward model r_ϕ
Ensure: GRPO batch b
Sample $\{y^i\}_{i=1}^{G} \sim \pi_\theta(\cdot \mid x)$
if the shaping method requires reference rewards **then**
    Sample $\{y_{\text{ref}}^m\}_{m=1}^{M} \sim \pi_{\text{ref}}(\cdot \mid x)$
    $r_{\text{ref}}^m \leftarrow r_\phi(x, y_{\text{ref}}^m)$ for $m = 1, \ldots, M$
end if
for $i = 1, \ldots, G$ **do**
    $r^i \leftarrow r_\phi(x, y^i)$
    $r_{\text{RL}}^i \leftarrow \textsc{RewardReshape}(r^i, \{r_{\text{ref}}^m\}_{m=1}^{M}, \text{len}(y^i), \textsf{mode})$
end for
Compute group-relative advantages $\{\hat{A}_i\}_{i=1}^{G}$
for $i = 1, \ldots, G$ **do**
    Tokenize (x, y^i) into $\{(s_{i,t}, a_{i,t})\}_{t=1}^{T_i}$
    Store $\log \pi_{\theta_{\text{old}}}(a_{i,t} \mid s_{i,t}) \leftarrow \log \pi_\theta(a_{i,t} \mid s_{i,t})$
    Store $\log \pi_{\text{ref}}(a_{i,t} \mid s_{i,t})$
end for
$b \leftarrow \{\log \pi_{\theta_{\text{old}}}, \log \pi_{\text{ref}}, \hat{A}_i, s_{i,t}, a_{i,t}\}_{i,t}$
return b

Algorithm 6 GRPO policy loss

Require: GRPO batch b, policy π_θ
Ensure: $\mathcal{L}_{\text{GRPO}}(\theta)$
Extract $\{\log \pi_{\theta_{\text{old}}}, \log \pi_{\text{ref}}, \hat{A}_i, s_{i,t}, a_{i,t}\}$ from b
$\rho_{i,t}(\theta) \leftarrow \exp(\log \pi_\theta(a_{i,t} \mid s_{i,t}) - \log \pi_{\theta_{\text{old}}}(a_{i,t} \mid s_{i,t}))$
$d_{i,t}(\theta) \leftarrow \log \pi_{\text{ref}}(a_{i,t} \mid s_{i,t}) - \log \pi_\theta(a_{i,t} \mid s_{i,t})$
$D_{i,t}(\theta) \leftarrow \exp(d_{i,t}(\theta)) - d_{i,t}(\theta) - 1$ ▷ positive KL estimator
$\mathcal{L}_{\text{GRPO}}(\theta) \leftarrow -\hat{\mathbb{E}}_{i,t}\left[\min\left(\rho_{i,t}(\theta)\hat{A}_i,\ \text{clip}(\rho_{i,t}(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_i\right) - \beta_{\text{KL}} D_{i,t}(\theta)\right]$
return $\mathcal{L}_{\text{GRPO}}(\theta)$

Algorithm 7 A2C training

Require: SFT policy π_{sft}, reward model r_ϕ, prompt set $\mathcal{D}$
Ensure: Aligned policy π_θ
Initialize policy $\pi_\theta \leftarrow \pi_{\text{sft}}$
Initialize reference policy $\pi_{\text{ref}} \leftarrow \pi_{\text{sft}}$
Initialize critic $V_\alpha \leftarrow r_\phi$
for $x \in \mathcal{D}$ **do**
    $b \leftarrow \textsc{BuildA2CBatch}(x, \pi_\theta, \pi_{\text{ref}}, V_\alpha, r_\phi)$
    $\mathcal{L}_{\text{A2C}}(\theta), \mathcal{L}_V(\alpha) \leftarrow \textsc{ComputeA2CLoss}(b, \pi_\theta, V_\alpha)$
    $\theta \leftarrow \theta - \eta_\pi \nabla_\theta \mathcal{L}_{\text{A2C}}(\theta)$
    $\alpha \leftarrow \alpha - \eta_V \nabla_\alpha \mathcal{L}_V(\alpha)$
end for
return π_θ

Algorithm 8 Construction of an A2C batch

Require: Prompt x, policy π_θ, reference policy π_{ref}, critic V_α, reward model r_ϕ
Ensure: A2C batch b

Sample policy response $y \sim \pi_\theta(\cdot \mid x)$
if the shaping method requires reference rewards **then**
    Sample $\{y_{\text{ref}}^m\}_{m=1}^M \sim \pi_{\text{ref}}(\cdot \mid x)$
    $r_{\text{ref}}^m \leftarrow r_\phi(x, y_{\text{ref}}^m)$ for $m = 1, \ldots, M$
end if
$r \leftarrow r_\phi(x, y)$
$r_{\text{RL}} \leftarrow \textsc{RewardReshape}(r, \{r_{\text{ref}}^m\}_{m=1}^M, \text{len}(y), \textsf{mode})$
Tokenize (x, y) into $\{(s_t, a_t)\}_{t=1}^T$
$k_t \leftarrow \log \pi_\theta(a_t \mid s_t) - \log \pi_{\text{ref}}(a_t \mid s_t)$
$r_t \leftarrow -\beta_{\text{KL}} k_t + \mathbb{I}[t = T]\, r_{\text{RL}}$
$V_t \leftarrow V_\alpha(s_t)$
Compute GAE advantages $\hat{A}_t$ and return targets G_t from $\{r_t, V_t\}_{t=1}^T$
$b \leftarrow \{\, G_t, \hat{A}_t, s_t, a_t \,\}_{t=1}^T$
return b

Algorithm 9 A2C policy and critic losses

Require: A2C batch b, policy π_θ, critic V_α
Ensure: $\mathcal{L}_{\text{A2C}}(\theta)$ and $\mathcal{L}_V(\alpha)$

Extract $\{G_t, \hat{A}_t, s_t, a_t\}$ from b
$\mathcal{L}_{\text{A2C}}(\theta) \leftarrow -\hat{\mathbb{E}}_t\Big[\log \pi_\theta(a_t \mid s_t)\hat{A}_t\Big]$
$\mathcal{L}_V(\alpha) \leftarrow \hat{\mathbb{E}}_t\Big[(V_\alpha(s_t) - G_t)^2\Big]$
return $\mathcal{L}_{\text{A2C}}(\theta), \mathcal{L}_V(\alpha)$

Algorithm 10 DPO training

Require: SFT policy π_{sft}, preference dataset $\mathcal{D}$
Ensure: Aligned policy π_θ

Initialize policy $\pi_\theta \leftarrow \pi_{\text{sft}}$
Initialize reference policy $\pi_{\text{ref}} \leftarrow \pi_{\text{sft}}$
for $(x, y_w, y_l) \in \mathcal{D}$ **do**
    $\mathcal{L}_{\text{DPO}}(\theta) \leftarrow -\log \sigma\left[\beta_{\text{DPO}}\left(\log \dfrac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \log \dfrac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]$
    $\theta \leftarrow \theta - \eta_{\text{DPO}} \nabla_\theta \mathcal{L}_{\text{DPO}}(\theta)$
end for
return π_θ

Algorithm 11 Replay-buffer substitution

Require: Persistent buffer $\mathcal{B}$, new batch b_{new}, capacity K
Ensure: Selected batch b or `None`

if $|\mathcal{B}| < K$ **then**
    Append b_{new} to $\mathcal{B}$
    return `None`
else
    Sample b uniformly from $\mathcal{B}$
    Remove b from $\mathcal{B}$
    Append b_{new} to $\mathcal{B}$
    return b
end if

Algorithm 12 Reward transformation

Require: Policy reward r, optional reference rewards $\{r_{\text{ref}}^m\}_{m=1}^M$, response length ℓ, transformation mode mode

Ensure: RL reward r_{RL}

if $\ell > L_{\max}$ **then**
  $r \leftarrow r - \beta_{\text{len}}(\ell - L_{\max})$ ▷ penalize excessive response length
end if
if mode $=$ vanilla **then**
  $r_{\text{RL}} \leftarrow r$
else if mode $=$ meanstd **then**
  $r_{\text{RL}} \leftarrow (r - \mu)/s$ ▷ μ and s are running statistics
else if mode $=$ clip **then**
  $r_{\text{RL}} \leftarrow \text{clip}(r, \mu - s, \mu + s)$
else if mode $=$ minmax **then**
  $r_{\text{RL}} \leftarrow (r - r_{\min})/(r_{\max} - r_{\min})$
else if mode $=$ LSC **then**
  $r_{\text{RL}} \leftarrow \log \sigma\Big(r - r_{\text{ref}}^{(0.85)}\Big)$
else if mode $=$ PAR **then**
  $r_{\text{RL}} \leftarrow \dfrac{1}{M} \displaystyle\sum_{m=1}^{M} \sigma(r - r_{\text{ref}}^m)$
end if
return r_{RL}