

Adversarial Training of Reward Models

Alexander Bukharin^{1,2}, Haifeng Qian¹, Shengyang Sun¹,
Adithya Renduchintala¹, Soumye Singhal¹, Zhilin Wang¹,
Oleksii Kuchaiev¹, Olivier Delalleau¹, Tuo Zhao²

¹NVIDIA ²Georgia Institute of Technology

Abstract

Reward modeling has emerged as a promising approach for the scalable alignment of language models. However, contemporary reward models (RMs) often lack robustness, awarding high rewards to low-quality, out-of-distribution (OOD) samples. This can lead to reward hacking, where policies exploit unintended shortcuts to maximize rewards, undermining alignment. To address this challenge, we introduce Adv-RM, a novel adversarial training framework that automatically identifies adversarial examples — responses that receive high rewards from the target RM but are OOD and of low quality. By leveraging reinforcement learning, Adv-RM trains a policy to generate adversarial examples that reliably expose vulnerabilities in large state-of-the-art reward models such as Nemotron 340B RM. Incorporating these adversarial examples into the reward training process improves the robustness of RMs, mitigating reward hacking and enhancing downstream performance in RLHF. We demonstrate that Adv-RM significantly outperforms conventional RM training, increasing stability and enabling more effective RLHF training in both synthetic and real-data settings. We will open-source all code and data.

1 Introduction

As large language models (LLMs) become increasingly capable, ensuring their alignment with human values remains a fundamental challenge (Amodei et al., 2016; Askell et al., 2021). One of the most promising approaches for scalable alignment of LLMs is reward modeling, in which a reward model is trained to predict human feedback (Christiano et al., 2017; Leike et al., 2018). These reward models (RMs) are then used as a proxy for human feedback in a variety of places throughout the LLM lifecycle, including data selection (Chen et al., 2023), Reinforcement Learning from Human Feedback (RLHF, Ouyang et al. 2022), and inference time moderation (Inan et al., 2023). This approach has been very successful, resulting in powerful language models such as GPT-4 and Llama 3 (Achiam et al., 2023; Dubey et al., 2024).

Despite their widespread use in LLM alignment, RMs are imperfect proxies for human judgment (Hendrycks et al., 2021; Gao et al., 2023). In particular, they exhibit poor out-of-distribution (OOD) generalization, often failing to reliably assess prompt-response pairs that diverge from their training distribution (Eisenstein et al., 2023). OOD generalization is challenging because RMs learn from a limited set of human-annotated examples, which do not cover the full diversity of possible model behaviors, leading to systematic failures when encountering novel model responses. These failures lead to reward hacking—where models trained on RMs optimize reward scores through unintended shortcuts, reducing their actual alignment with human values. Such reward hacking can result in unexpected behaviors and difficult-to-detect misalignment, an issue that becomes even more pernicious as model capabilities increase and evaluation becomes more challenging.

Building robust RMs to mitigate reward hacking presents two unsolved challenges. The first is measuring a RM’s robustness. Unlike image and speech models, where adversarial attacks can serve as robustness metrics (Carlini & Wagner, 2017), the discrete nature of language

makes adversarial attacks on language models challenging and their robustness harder to define (Wang et al., 2021). Reward models introduce another layer of complexity, as their accuracy is measured on preference pairs, requiring adversarial preference pairs rather than a single adversarial input. So far the only known reliable way to quantify robustness of a RM is to measure how long it takes for reward hacking to occur in RLHF. This in itself can be challenging: we can rely on human evaluation to tell us if the policy training has significantly degraded, but this is costly and small misalignments can easily go undetected.

The second problem is how to improve the robustness of a RM. Prior works show that reward hacking often occurs because the RM gives high reward scores to model responses outside the reward model training distribution (Casper et al., 2023; Eisenstein et al., 2023). Giving a high reward to OOD responses is bad, as they deviate significantly from the expert responses in the RLHF dataset and are therefore likely to be low quality (Ho & Ermon, 2016). To avoid rewarding such responses, contemporary works have tried a variety of approaches, including regularizing training so that the policy model remains close to its initialization (Jaques et al., 2019; Stiennon et al., 2020), penalizing out-of-distribution (OOD) model responses using uncertainty estimates (Eisenstein et al., 2023; Coste et al., 2023), and regularizing reward learning (Ibarz et al., 2018; Liu et al., 2024b). However, we find that uncertainty estimates are not reliable indicators of response quality for in-distribution responses, which may limit the effectiveness of such regularization.

In this work we propose a new adversarial training framework, Adv-RM, to address both problems. Adv-RM automatically discovers adversarial responses that the targeted RM gives a high reward to while being OOD and low quality. We find these examples by training an adversarial policy using reinforcement learning to generate responses that maximize discrepancy across different RMs, while receiving high rewards from the target RM. This approach reliably finds adversarial examples for state-of-the-art (SOTA) RMs, including Skywork-Reward-Gemma-2-27B (Liu et al., 2024a), Llama-3.1-Nemotron-70B-Reward (Wang et al., 2024), and Nemotron-4-340B-Reward (Adler et al., 2024), with over 80% attack success rates. Interestingly, we find that these popular RMs give high reward scores to a variety of low-quality responses, such as responses that have no punctuation or responses with random text in the middle of a response.

Once we find adversarial samples, Adv-RM incorporates them into the RM training procedure. Since the success rate of our attack method is high, we can include adversarial samples in the training data set without additional human annotation cost by simply using an in-distribution response as the chosen response and the adversarial response as the rejected response. We conduct extensive evaluation of these adversarially trained RMs through adversarial attacks, downstream RLHF training in the synthetic setting of Gao et al. (2023), and RLHF training on real datasets (Wang et al., 2024). We found that Adv-RM reward models not only are more robust against attacks but also mitigate reward hacking. With these models, RLHF can proceed for three times as many steps as runs with conventional RMs without exhibiting reward hacking, resulting in more performant aligned models.

We summarize our contributions as follows:

- We develop an RL-based approach that successfully generates adversarial samples for SOTA RMs. To the best of our knowledge, this is the first work to do so without human domain knowledge, providing a new means of evaluating RM robustness.
- We use adversarial samples to augment RM training. Through extensive experiments, we find that such adversarial training results in reward models that are more robust to hacking and have better downstream performance in RLHF.

2 Related Work

Mitigating Reward Hacking. There have been several works that attempt to mitigate reward hacking. Some of the earliest works used reward ensembles, and found that different ensembling approaches can slow (but not prevent) reward hacking (Coste et al., 2023; Eisenstein et al., 2023; Zhai et al., 2023; Ramé et al., 2024; Yan et al., 2024). In our experiments we compare with two simple reward ensemble aggregation approaches, MEAN and

MEAN_MINUS_STD (Eisenstein et al., 2023). Chen et al. (2024) propose a new architecture that disentangles length from human preferences. Recently Liu et al. (2024b) proposes RR which augments the RM training data by comparing responses from a prompt with random responses from other prompts, an approach we find to be effective.

Two works discuss adversarial training of verifiers. Amodei et al. (2016) presents the idea of adversarial reward models, a training framework in which adversarial training is used to find samples where RMs and human annotators disagree. However, they do not implement this idea, and relying on human annotators in their suggested way would be expensive and difficult. Kirchner et al. (2024) propose adversarial training of a verifier model but it is limited to tasks with rule-based verifiers like GSM8K and is not applicable to general reward modeling. Concurrently with our work, Wu et al. (2025) use manually defined input transformations to increase the difficulty of RM benchmarks.

Adversarial Attacks on LLMs. There is a wide literature discussing adversarial attacks on LLMs (Zhang et al., 2020). Typically these works attempt to greatly alter the model output while only slightly changing the model input (we only black-box attacks for simplicity (Guo et al., 2019)). We compare Adv-RM with two such approaches: Textfooler (Jin et al., 2020), which replaces words with synonyms such that the model output is greatly changed and StyleAdv (Qi et al., 2021), which changes the style of the text to greatly change the model output. We observe that such approaches usually do not successfully attack RMs as (1) these attacks were not designed with the paired nature of RM training data in mind and (2) these attacks were designed to attack weaker models such as BERT (Devlin et al., 2019).

3 Methodology

3.1 Background

RLHF. We follow the standard RLHF setup of Stiennon et al. (2020); Ouyang et al. (2022). We consider a setting where we have a dataset of N human preference annotations, $D = \{x_i, y_{w,i}, y_{\ell,i}\}_{i=1}^N$. Here x_i is the i th prompt, $y_{w,i}$ is the response preferred by human annotators, and $y_{\ell,i}$ is the rejected response. With this dataset, conventional RLHF trains a Bradley-Terry (Bradley & Terry, 1952) reward model $R_\theta(x, y)$ by minimizing the loss

$$\mathcal{L}(\theta) = -\mathbb{E}_{x,y_w,y_\ell \sim D} [\log(\sigma(R_\theta(x, y_w) - R_\theta(x, y_\ell)))] \quad (1)$$

Next, RLHF trains a policy π_ϕ to maximize the expected reward $R_\theta(x, y)$ while remaining close to an initial policy π_{SFT} . This is done by maximizing the following KL-regularized objective function over a prompt distribution P ,

$$\mathbb{E}_{(x,y) \sim P, y \sim \pi_\phi(x)} [R_\theta(x, y) - \beta \text{D}_{\text{KL}}(\pi_\phi(\cdot|x) || \pi_{\text{SFT}}(\cdot|x))]$$

using a reinforcement learning algorithm such as PPO, RLOO, or GRPO (Schulman et al., 2017; Ahmadian et al., 2024; Kool et al., 2019; Shao et al., 2024).

Out-of-Distribution Detection. Detecting OOD data is a long studied problem. Here we consider a simple approach for OOD data detection in RLHF: Ensemble disagreement (Lakshminarayanan et al., 2017). Concretely, we use the disagreement of an ensemble of reward models trained on different data splits to estimate the uncertainty of different prompt-response pairs. Such an approach has been shown to effectively capture model uncertainty and detect out-of-distribution data in a wide variety of domains, including continuous control (Brantley et al., 2019), computer vision (Vyas et al., 2018), and natural language processing (Yuan et al., 2023). Most recently, Eisenstein et al. (2023) showed that reward model disagreement is an effective way to detect OOD responses in RLHF. Here, we denote reward model disagreement as $U_{\theta_1, \dots, \theta_K}(x, y) = \text{STD}(R_{\theta_1}(x, y), \dots, R_{\theta_K}(x, y))$, where $R_{\theta_1}, \dots, R_{\theta_K}$ are K different reward models trained with different data splits or random seeds. We remark that when $K = 2$, $U_{\theta_1, \theta_2}(x, y)$ is proportional to $|R_{\theta_1}(x, y) - R_{\theta_2}(x, y)|$. We normalize all RMs to have mean 0 and standard deviation 1.

3.2 Learning to Attack

The goal of our adversarial attack is, given a prompt, to find responses that receive high rewards from the target reward model and yet are OOD. Given a set of reward models $R_{\theta_1}, \dots, R_{\theta_K}$, one reward model we seek to attack R_{θ_1} , and set of prompts D , we seek to train a policy π_{adv} to satisfy the constrained optimization problem:

$$\max_{\pi_{\text{adv}}} \mathbb{E}_{x \sim D, y \sim \pi_{\text{adv}}(x)} [U_{\theta_1, \dots, \theta_k}(x, y)] \quad \text{s.t.} \quad \mathbb{E}_{x \sim D, y \sim \pi_{\text{adv}}(x)} [R_{\theta_1}(x, y) > T(x)]. \quad (2)$$

Here $T(x)$ is a threshold dependent on the prompt. In our experiments we use two RMs (i.e. $K = 2$) and set $T(x)$ to be the average reward achieved by π_{SFT} on the prompt x according to $R_{\theta_1}(\cdot)$. Optimizing (2) means finding OOD samples that the target RM $R_{\theta_1}(\cdot)$ predicts to be better than those generated by π_{SFT} . Although this approach would indeed find highly OOD responses that get high reward, it is possible that by optimizing (2) we would find samples that all reward models predict will receive a high score, and the reward models just disagree on how high the score is. In order to encourage the found samples to be OOD and low quality, we use the formulation

$$\max_{\pi_{\text{adv}}} \mathbb{E}_{x \sim D, y \sim \pi_{\text{adv}}(x)} [R_{\theta_1}(x, y) - \lambda R_{\theta_2}(x, y)] \quad \text{s.t.} \quad \mathbb{E}_{x \sim D, y \sim \pi_{\text{adv}}(x)} [R_{\theta_1}(x, y) > T(x)]. \quad (3)$$

Here λ is a hyperparameters, with λ typically greater than 1. This formulation encourages finding responses with high uncertainty and a very low score from $R_{\theta_2}(x, y)$, increasing the likelihood of low quality samples. We overload our notation, and denote $R_{\theta_1}(x, y) - \lambda R_{\theta_2}(x, y)$ as $U_{\theta_1, \theta_2}(x, y)$.

Unlike previous adversarial attacks that rely on perturbations, the key innovation of our work is training an adversarial policy π_{adv} to generate adversarial examples. Specifically, we optimize (3) using reinforcement learning and train an adversarial policy π_{adv} to maximize the following reward:

$$R_{\text{adv}}(x, y) = \begin{cases} R_{\theta_1}(x, y) - \lambda R_{\theta_2}(x, y) & \text{if } R_{\theta_1}(x, y) > T(x) \\ R_{\theta_1}(x, y) - C & \text{otherwise} \end{cases}.$$

Here C is a constant that ensures the second case always gives a lower reward than the first case. This reward encourages the policy to maximize RM uncertainty and find OOD samples, while ensuring they receive a high reward from $R_{\theta_1}(x, y)$ (the attack target). We maximize this reward using RLOO (Ahmadian et al., 2024) over the RLHF prompt dataset, starting from π_{SFT} . The reinforcement learning training procedure leaves us with a set of prompt-response pairs $\{(x_1, y_1^{\text{adv}}), \dots, (x_n, y_n^{\text{adv}})\}$ generated by the adversarial policy throughout training.

Data Filtering. We filter the collected prompt-response pairs using $R_{\theta_1}(x, y)$ and $U_{\theta_1, \theta_2}(x, y)$ to only select samples where the policy was able to successfully attack $R_{\theta_1}(x, y)$. Concretely, we select the adversarial dataset D_{adv} as

$$D_{\text{adv}} = \{(x, y_{\text{adv}}) \mid R_{\theta_1}(x, y_{\text{adv}}) > T(x) \text{ and } Z(U_{\theta_1, \theta_2}(x, y_{\text{adv}})) > 1.96\}.$$

Here, $Z(U_{\theta_1, \theta_2}(x, y))$ is the z-score of the uncertainty $U_{\theta_1, \theta_2}(x, y)$, computed over responses sampled from π_{SFT} . A value of $Z(U_{\theta_1, \theta_2}(x, y)) > 1.96$ indicates that the reward uncertainty of (x, y_{adv}) is significantly higher than the uncertainty of responses within the distribution of π_{SFT} . This ensures each sample in D_{adv} receives a high reward score yet is OOD.

3.3 Preliminary Analysis: Uncertainty as a Quality Signal

Using a synthetic RLHF setup (Gao et al., 2023), we investigate how and when reward models uncertainty (i.e. $U_{\theta_1, \theta_2}(\cdot)$) correlates with response quality. Concretely, we train two reward models ($R_{\theta_1}(\cdot)$ and $R_{\theta_2}(\cdot)$) based on different random seeds on a preference dataset created by a gold reward model, and then measure $U_{\theta_1, \theta_2}(\cdot)$ versus the gold score which measures quality. We investigate this relationship on two datasets: one dataset where the responses are samples from the SFT model, and one sampled from the Adv-RM model.

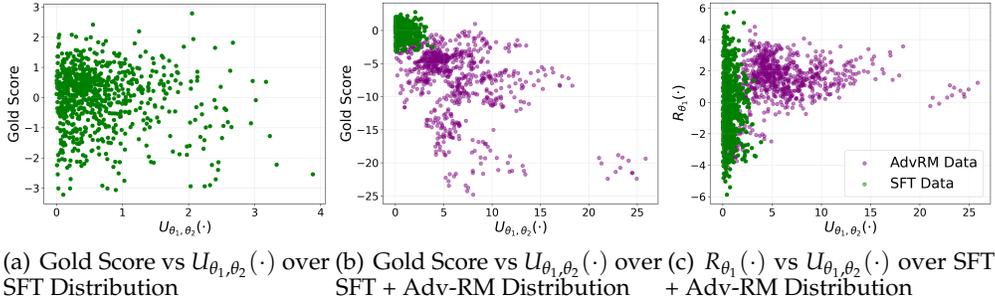


Figure 1: (a) and (b) show $U_{\theta_1, \theta_2}(\cdot)$ versus gold score. (c) shows that R_{θ_1} is similar for the Adv-RM and SFT data, which helps isolate the relationship between $U_{\theta_1, \theta_2}(\cdot)$ and gold score.

As we can see from Figure 1(a), uncertainty $U_{\theta_1, \theta_2}(\cdot)$ is not very correlated with the gold reward (pearson’s correlation -0.05) when we sample from π_{SFT} . This indicates that directly penalizing ensemble uncertainty during RLHF may add some noise to training. On the other hand, when we use Adv-RM to generate OOD datapoints with a higher uncertainty level (1(b)), we see that uncertainty is more correlated with gold score (pearson’s correlation -0.70). Together, these plots suggest that ensemble uncertainty is a useful indicator of response quality when uncertainty is high (and can therefore be used to find adversarial examples), but it provides little signal for distinguishing quality among responses with low uncertainty (implying directly regularizing RM uncertainty during policy training is not ideal).

3.4 Adversarial Training of Reward Models

Our analysis shows that Adv-RM’s attacks are successful at producing adversarial responses. To increase RM robustness, we incorporate these adversarial responses into training.

Preference Pair Construction. We construct adversarial preference pairs by setting (x, y_{SFT}) as the preferred pair and setting (x, y_{adv}) as the rejected pair. We choose a SFT response y_{SFT} such that $R_{\theta_1}(x, y_{\text{SFT}}) > \mathbb{E}_{y \sim \pi_{\text{SFT}}(x)} [R_{\theta_1}(x, y_{\text{SFT}})]$, to avoid using low quality SFT responses as the preferred response. All of these adversarial preference pairs (x, y_{adv}) are constructed such that the rejected response y_{adv} receives a higher reward score from $R_{\theta_1}(\cdot)$ than the SFT response, yet y_{adv} has very high $U_{\theta_1, \theta_2}(\cdot)$ so it is OOD and likely low quality.

Multiple Rounds of Training. Once we construct the adversarial preference pairs, we add around 1000 of them to the original RLHF preference dataset and train a new RM from scratch using (1). To further improve RM robustness, we conduct several rounds of adversarial training, in each round trying to find vulnerabilities of the new robust RM. Once we find these vulnerabilities, we append them to our current robust RLHF dataset and train a new RM from scratch. We find that after 2 rounds of adversarial training, our attack procedure is no longer successful. At this point we terminate training.

4 Experiments: Adversarial Attacks on Reward Models

4.1 Evaluation

We consider an adversarial attack on a reward model successful if it produces a preference pair $(x, y_1), (x, y_2)$ such that

$$R_{\theta_1}(x, y_1) > R_{\theta_1}(x, y_2) \text{ while } R_{\text{gold}}(x, y_1) < R_{\text{gold}}(x, y_2). \quad (4)$$

The “gold” reward here is meant to capture the reward induced by human values. In synthetic setups we have access to the simulated gold reward model, and in real RLHF settings we can approximate R_{gold} with a LLM (Dubois et al., 2023) or human feedback.

In synthetic settings, we find (4) can sometimes be met by chance if y_1 and y_2 are similar. To mitigate this, we introduce a stricter criterion: the attack is only considered successful if

$$Z_{R_{\theta_1}}(R_{\theta_1}(x, y_1)) > 0 \text{ and } Z_{R_{\text{gold}}}(R_{\text{gold}}(x, y_1)) < -1.96. \quad (5)$$

Intuitively, a successful attack means that $R_{\theta_1}(x, y_1)$ is above the average SFT response for the response x , while $R_{\text{gold}}(x, y_1)$ falls within the bottom 5% of SFT scores. When using LLM judges or humans as R_{gold} , we only consider (4).

4.2 Experimental Setup

Synthetic RLHF setup. Following Gao et al. (2023), we evaluate Adv-RM in a synthetic RLHF setup. We use Llama-3.1-Nemotron-70B-Reward as the gold RM, and create a synthetic training dataset by relabelling Helpsteer-2-Preferences (Wang et al., 2024) with it. We then train proxy RMs based on Llama-3.1-8B-Instruct (Grattafiori et al., 2024). In this setting $R_{\theta_1}(\cdot)$ and $R_{\theta_2}(\cdot)$ are two RMs trained on different random seeds controlling the data order. We evaluate attack success rate using the gold RM. Details can be found in Appendix C. Adv-RM adversarial policies are initialized from Llama-3.1-8B-Instruct.

Real RLHF setup. We attack three top performing RMs on RewardBench (Lambert et al., 2024), Skywork-Reward-Gemma-2-27B, Llama-3.1-Nemotron-70B-Reward, and Nemotron-4-340B-Reward. When attacking Skywork-Reward-Gemma-2-27B and Nemotron-4-340B-Reward we use Llama-3.1-Nemotron-70B-Reward as $R_{\theta_2}(\cdot)$ in OOD detection and when attacking Llama-3.1-Nemotron-70B-Reward we use an in-house RM as $R_{\theta_2}(\cdot)$. We then evaluate the attack success rate using (4) with human judges, Llama 405B, and Deepseek-R1 (Guo et al., 2025). See Appendix G for details. Adv-RM adversarial policies are initialized from Llama-3.1-8B-Instruct.

Method	Success Rate: Train		Success Rate: Test	
	Standard	Strict	Standard	Strict
RM Over-optimization	8.31 (2.44)	2.85 (1.47)	7.59 (2.34)	1.34 (1.02)
RRM	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)
StyleAdv	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)
Textfooler	9.57 (2.60)	0 (0.00)	18.8 (3.45)	0 (0.00)
Adv-RM	99.31 (0.73)	92.91 (2.27)	100 (0.00)	100 (0.00)

Table 1: Comparison of attack success rates over 128 Helpsteer-2 prompts in the synthetic setting, with standard errors in parentheses.

Baselines. Although we are the first to adversarially attack RMs (to the best of our knowledge), we consider a few baselines that try to attack LLM classifiers (Textfooler and StyleAdv), RRM (Liu et al., 2024b), and a reward over-optimization baseline. As Textfooler and StyleAdv only produce a single adversarial response y_1 to a prompt x , we pair their attacks with the original response from the SFT model $y_2 \sim \pi_{\text{SFT}}(x)$ to calculate performance in (4) and (5). For reward over-optimization, we optimize the policy for three times the number of steps as achieves the best downstream performance and use the resulting policy to generate adversarial attacks. We present more details in Appendix D.

4.3 Main Results: Adversarial Attacks

Synthetic Setting. Table 1 compares Adv-RM to other baselines. For Adv-RM we generate 4 attacks per evaluation prompt and select the attack with highest $U_{\theta_1, \theta_2}(\cdot)$. Adv-RM consistently finds adversarial samples, indicating its efficacy. In contrast, StyleAdv and TextFooler have low success rates despite generating 50 attacks per prompt and filtering by reward difference. These methods slightly perturb response semantics, rarely causing conflicts between the trained and gold RMs. RM over-optimization can occasionally produce successful attacks; however, we find that the over-optimized policy often generates high-quality responses with only minor quirks (e.g., excessive formatting). These responses are

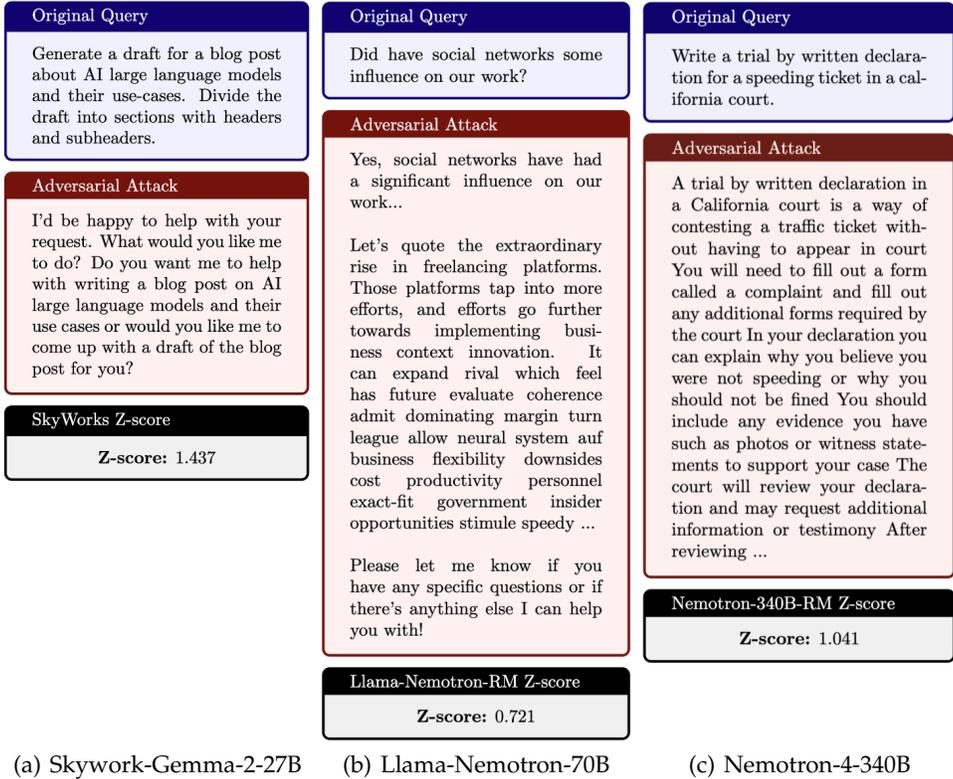


Figure 2: Adversarial examples generated by Adv-RM for top RewardBench models. The Z-score is computed by normalizing the reward score by the average reward achieved by Llama-3.1-8b-Instruct for that prompt.

	Human Evaluation	Deepseek R1	Llama 405B
Skyworks Gemma 27B			
TextFooler	19.61 (5.60)	56 (4.41)	2 (1.24)
RM Over-Optimization	20 (5.66)	32 (3.99)	14 (3.00)
Adv-RM	100 (0.00)	99 (0.73)	99 (0.73)
Llama-Nemotron 70B			
TextFooler	13.2 (4.86)	27 (3.85)	1 (0.88)
RM Over-Optimization	23.64 (6.07)	7 (2.23)	3 (1.50)
Adv-RM	83.01 (5.13)	71 (4.12)	24 (3.73)
Nemotron 340B			
TextFooler	17.86 (5.22)	38 (4.33)	2 (1.24)
RM Over-Optimization	17.65 (5.20)	23 (3.73)	19 (3.52)
Adv-RM	78.85 (5.76)	76 (4.11)	82 (3.42)

Table 2: Adv-RM attack effectiveness across different real RMs and judges. Values in parentheses represent standard errors.

not necessarily of lower quality than those produced by π_{SFT} , making them less adversarial in nature.

Real Reward Models. The main results when attacking Skyworks-Gemma 27B reward, Llama-Nemotron 70B reward, and Nemotron-4 340B reward can be found in Table 2. Note we only compare Adv-RM to the strongest baselines in Table 1. As we can see from Table 2, Adv-RM is able to successfully attack all three state-of-the-art reward models, generating adversarial pairs that the target RM predicts as correct while the judge (humans or models)

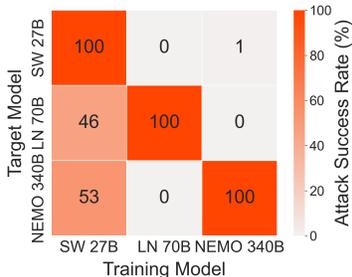


Figure 3: Attack transferability.

Method	Success Rate
Adv-RM	92.91 (2.27)
- Filtering	73.27 (3.94)
- Unequal RM Weights	16.41 (3.13)
- Reward Threshold	0.0 (0.0)

Table 3: Adv-RM attack ablation in the synthetic setup.

predicts as incorrect. We remark that using LLM as a judge may significantly underestimate the success rate of Adv-RM as they may also be fooled by adversarial examples.

We provide examples of these attacks in Figure 2. Interestingly, Adv-RM uncovers distinct vulnerabilities in each reward model. For Skyworks-Gemma 27B, responses that simply repeat the prompt can achieve high scores. For Llama-Nemotron 70B, nonsensical gibberish in the middle of long text can be rewarded. And for Nemotron 340B, responses that lack punctuation can receive unexpectedly high scores.

Attack Transferability. We show the transferability of attacks in Figure 3. We find that there is some weak transfer between Adv-RM attacks on the Skyworks 27B model, but no transfer between the different Nemotron RMs.

Ablation. Our ablation study in Table 3 shows that all components of Adv-RM are helpful.

5 Experiments: Downstream Policy Performance

Experimental Setup. For the synthetic setting, we use the RMs trained with Adv-RM in Section 4 to optimize Llama-3.1-8B-Instruct models using RLOO. We measure performance using the Gold Reward. For the real RLHF setting, we train Llama-3.1-8B-Instruct RMs on the Helpsteer-2-Preferences dataset (Wang et al., 2024), and use them to optimize Llama-3.1-8B-Instruct models. We measure performance on a held out set of 128 prompts using different judge models (Llama 405B and Deepseek R1), as well as Llama-Nemotron 70B RM.

Baselines. We consider several baselines, including conventional RLHF (baseline), the mean of an ensemble of RMs (Ens (Mean)), the mean minus the uncertainty of an ensemble of RMs (Ens (STD), also called UWO in Coste et al. (2023)), and RRM (Liu et al., 2024b). Details can be found in Appendix E.

Results: Synthetic Setting. The training curve, best reward achieved, and final reward achieved for Adv-RM policy training and other baselines can be found in Figure 4. The policies trained with Adv-RM achieves a higher gold reward with minimal reward hacking, exhibits little length hacking, and performs better or as well as all of the baselines. We find that Adv-RM only results in a small improvement in best-case policy performance, and mostly improves robustness against reward hacking. Overall, these results show that including adversarial samples in RM training can improve downstream performance.

Results: Real RLHF Setting. The judge scores across training can be found in Figure 5. Adv-RM results in the most stable training out of any of the baselines, improving over the initial model for 2-3 times more steps than RLHF. We find that RRM is quite a strong baseline, but that Adv-RM’s training is still much more stable (note Adv-RM and RRM are complementary approaches that can be used together). This indicates the utility of training with adversarial examples, rather than the randomly sampled responses RRM uses. Furthermore, we show the RewardBench scores achieved by Adv-RM in Table 4, where we find Adv-RM (83.99) gets a slightly higher score compared to conventional training (83.29).

Robustness Against Attacks. In Figure 6(a) we plot the attack success rate of Adv-RM on policies trained with multiple rounds of Adv-RM training. As we can see, the models

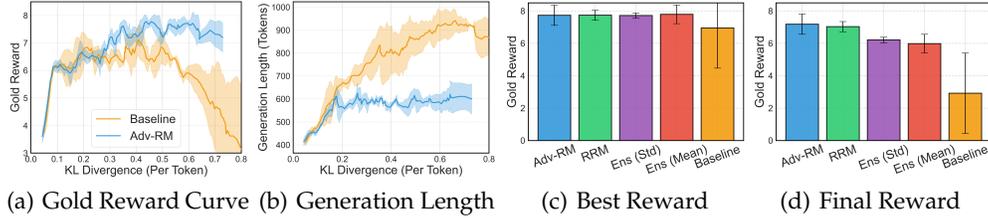


Figure 4: Downstream policy results in the synthetic setup. Error bars represent \pm one standard deviation over three random seeds.

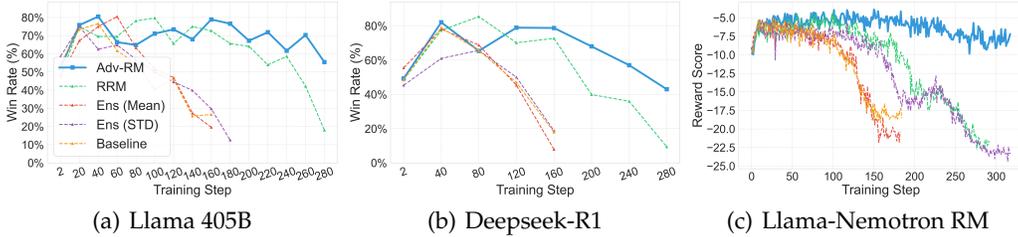


Figure 5: Downstream policy results with different judge models.

trained with Adv-RM become increasingly robust to such attacks, indicating that Adv-RM vulnerabilities can be mitigated with adversarial training.

Ablation. In Figure 6(b), we show the final downstream policy performance of policies trained with different amounts of Adv-RM data. We find that around 1000 samples results in the best downstream performance, and that more samples actually hurts performance. We hypothesize this is due to the fact that Adv-RM data is not very diverse, so adding too much such data can result in overfitting. In Figure 6(c), we show the performance with multiple rounds of Adv-RM training. In general we find two rounds leads to the best performance, and that after two rounds, there is minimal gain.

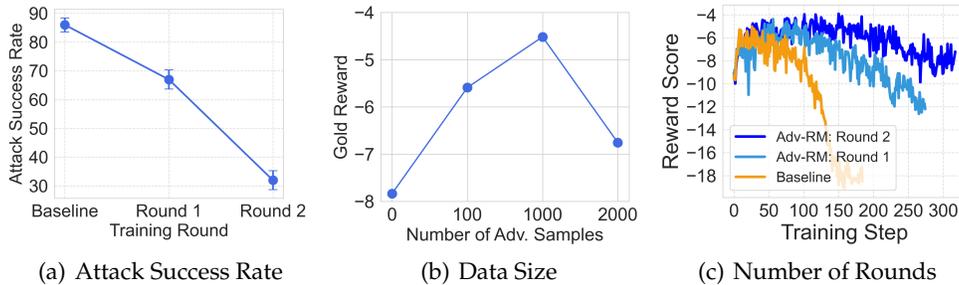


Figure 6: Robustness of Adv-RM models and ablation study. In (a) and (c) we consider the real RLHF setting, where in (b) it is the synthetic setting. For (a) Deepseek-R1 is the judge, and in (b) and (c) Llama-Nemotron RM is.

6 Conclusion and Limitations

In this paper, we introduce Adv-RM, an adversarial training framework for RMs. Adv-RM can successfully attack state-of-the-art RMs, uncovering surprising failure cases where they assign high scores to low quality responses. Incorporating Adv-RM examples into RM training significantly improves RLHF stability and performance. Although Adv-RM is

successful, it has a few limitations. Adv-RM relies on reward ensembles, meaning adversarial responses may be missed if all RMs incorrectly predict they are high quality, highlighting the need for more robust out-of-distribution detection. Additionally, Adv-RM incurs a substantial computational cost—approximately $3\times$ that of standard RLHF training (see Appendix F)—but we believe this overhead is justified by the resulting gains in robustness.

References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Bo Adler, Niket Agarwal, Ashwath Aithal, Dong H Anh, Pallab Bhattacharya, Annika Brundyn, Jared Casper, Bryan Catanzaro, Sharon Clay, Jonathan Cohen, et al. Nemotron-4 340b technical report. *arXiv preprint arXiv:2406.11704*, 2024.
- 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. *arXiv preprint arXiv:2402.14740*, 2024.
- Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.
- Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.
- 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.
- Kiante Brantley, Wen Sun, and Mikael Henaff. Disagreement-regularized imitation learning. In *International Conference on Learning Representations*, 2019.
- Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. Ieee, 2017.
- Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, 2023.
- Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. Alpapasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*, 2023.
- Lichang Chen, Chen Zhu, Davit Soselia, Jiu-hai Chen, Tianyi Zhou, Tom Goldstein, Heng Huang, Mohammad Shoeybi, and Bryan Catanzaro. Odin: Disentangled reward mitigates hacking in rlhf. *arXiv preprint arXiv:2402.07319*, 2024.
- Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.
- Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. *arXiv preprint arXiv:2310.02743*, 2023.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, and Tatsunori B Hashimoto. AlpacaFarm: A simulation framework for methods that learn from human feedback. *Advances in Neural Information Processing Systems*, 36:30039–30069, 2023.
- Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D’Amour, DJ Dvijotham, Adam Fisch, Katherine Heller, Stephen Pfohl, Deepak Ramachandran, et al. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. *arXiv preprint arXiv:2312.09244*, 2023.
- Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pp. 10835–10866. PMLR, 2023.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In *International conference on machine learning*, pp. 2484–2493. PMLR, 2019.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. Unsolved problems in ml safety. *arXiv preprint arXiv:2109.13916*, 2021.
- Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.
- Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. *Advances in neural information processing systems*, 31, 2018.
- Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.
- Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.
- Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 8018–8025, 2020.
- Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Jan Hendrik Kirchner, Yining Chen, Harri Edwards, Jan Leike, Nat McAleese, and Yuri Burda. Prover-verifier games improve legibility of llm outputs. *arXiv preprint arXiv:2407.13692*, 2024.
- Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! 2019.

- Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.
- Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024.
- Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.
- Chris Yuhao Liu, Liang Zeng, Jiakai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*, 2024a.
- Tianqi Liu, Wei Xiong, Jie Ren, Lichang Chen, Junru Wu, Rishabh Joshi, Yang Gao, Jiaming Shen, Zhen Qin, Tianhe Yu, et al. Rrm: Robust reward model training mitigates reward hacking. *arXiv preprint arXiv:2409.13156*, 2024b.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.
- Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. Mind the style of text! adversarial and backdoor attacks based on text style transfer. *arXiv preprint arXiv:2110.07139*, 2021.
- Alexandre Ramé, Nino Vieillard, Léonard Hussenot, Robert Dadashi, Geoffrey Cideron, Olivier Bachem, and Johan Ferret. Warm: On the benefits of weight averaged reward models. *arXiv preprint arXiv:2401.12187*, 2024.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Gerald Shen, Zhilin Wang, Olivier Delalleau, Jiaqi Zeng, Yi Dong, Daniel Egert, Shengyang Sun, Jimmy Zhang, Sahil Jain, Ali Taghibakhshi, et al. Nemo-aligner: Scalable toolkit for efficient model alignment. *arXiv preprint arXiv:2405.01481*, 2024.
- Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.
- Apoorv Vyas, Nataraj Jammalamadaka, Xia Zhu, Dipankar Das, Bharat Kaul, and Theodore L Willke. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 550–564, 2018.
- Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. Adversarial glue: A multi-task benchmark for robustness evaluation of language models. *arXiv preprint arXiv:2111.02840*, 2021.
- Zhilin Wang, Alexander Bukharin, Olivier Delalleau, Daniel Egert, Gerald Shen, Jiaqi Zeng, Oleksii Kuchaiev, and Yi Dong. Helpsteer2-preference: Complementing ratings with preferences. *arXiv preprint arXiv:2410.01257*, 2024.
- Zhaofeng Wu, Michihiro Yasunaga, Andrew Cohen, Yoon Kim, Asli Celikyilmaz, and Marjan Ghazvininejad. rewordbench: Benchmarking and improving the robustness of reward models with transformed inputs. *arXiv preprint arXiv:2503.11751*, 2025.

Yuzi Yan, Xingzhou Lou, Jialian Li, Yiping Zhang, Jian Xie, Chao Yu, Yu Wang, Dong Yan, and Yuan Shen. Reward-robust rlhf in llms. *arXiv preprint arXiv:2409.15360*, 2024.

Lifan Yuan, Yangyi Chen, Ganqu Cui, Hongcheng Gao, Fangyuan Zou, Xingyi Cheng, Heng Ji, Zhiyuan Liu, and Maosong Sun. Revisiting out-of-distribution robustness in nlp: Benchmarks, analysis, and llms evaluations. *Advances in Neural Information Processing Systems*, 36:58478–58507, 2023.

Yuanzhao Zhai, Han Zhang, Yu Lei, Yue Yu, Kele Xu, Dawei Feng, Bo Ding, and Huaimin Wang. Uncertainty-penalized reinforcement learning from human feedback with diverse reward lora ensembles. *arXiv preprint arXiv:2401.00243*, 2023.

Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41, 2020.

A Appendix

B Reproducibility Statement

We will release all the code and scripts used to reproduce our experiments.

C Adversarial Training Hyperparameters

For both the synthetic and real data setup we use similar training configurations. In particular, we use a constant learning rate of $5e - 7$ with the Adam optimizer (Kingma & Ba, 2014), use a batch size of 64, and generate 4 samples per prompt in RLOO. We fix $\lambda_1 = 1$ and tune $\lambda_2 \in [8, 10, 12]$. We set $C = -25$. For the adversarial policy, we initialize it from llama-3.1-8B-instruct.

For reward model training, we use the following setup: we train for one epoch, and tune the learning rate in $[1e - 6, 5e - 6, 7e - 6, 1e - 5]$. We select intermediate checkpoints throughout training based on their RewardBench scores. We use a batch size of 128 and evaluate performance every 10 steps.

D Adversarial Attack Baselines

Textfooler. For Textfooler, we follow the approach of Jin et al. (2020) and generate 100 adversarial attacks for each prompt-response pair in our dataset by replacing individual words with their synonyms. We then filter these attacks to only select the ones that result in the highest increase in training reward model score.

StyleAdv For StyleAdv, we use Llama-3.3-70b to rewrite text in one of the five following styles: Shakespearean English, Biblical Prose, Journalistic, Stream of Consciousness, and Satirical. This is in line with Qi et al. (2021).

RRM. For RRM, we consider the attack to be the RRM augmentation, where a response from another prompt-response pair is used as the chosen response for the current prompt.

RM over-optimization. For RM over-optimization, we train the policy for three times the number of steps that achieves the best performance. In the synthetic setup this is 150 steps (i.e. best performance is achieved at 50 steps) and in the real-data setup this is 240 steps.

E Robust Policy Optimization Baselines

Ensemble Baselines. We consider two different ensemble baselines: one where we take the mean between reward models to use as the optimization target, and one where we use the mean minus the standard deviation of the reward model as the optimization target. We weigh the standard deviation of the ensemble baseline by a hyperparameter λ that we tune in $[0.1, 0.5, 1.0]$.

RRM. For RRM, we tune the number of augmented datapoints in $[2x, 3x, 5x]$ the original dataset size. We then train reward models on this augmented dataset and conduct policy optimization on the reward model.

F Computational Cost.

We conduct all experiments on Nvidia A100 80G GPUs. Training conventional 8b reward models takes around 4 hours on one 8 GPU node, and conventional policy training takes around 12 hours on 2 8 GPU nodes. Training the Adv-RM adversarial policy takes around 12 hours on on 2 8 GPU nodes. Therefore, conventional RLHF takes 224 A100 hours, while Adv-RM training (with 2 rounds) takes 672 A100 hours. We conduct all experiments in NeMo-Aligner (Shen et al., 2024).

Table 4: Reward Bench Scores

	Overall	Chat	Chat-Hard	Safety	Reasoning
Baseline	0.8329	0.9721	0.6118	0.8541	0.8938
Adv-RM	0.8399	0.9609	0.6404	0.8514	0.9070

G Evaluating Adversarial Attacks

LLM as a Judge. For LLM as a judge, we use Llama 3 405B and Deepseek-R1. To get the preference for each adversarial input pair, we ask the LLM judge to rank them with the following prompt:

```
Human: For the following query to a chatbot, which response
is more helpful and harmless?

Query: <prompt>
Response A: <response A>
Response B: <response B>

FIRST provide a one-sentence comparison of the two responses
and explain which you feel is more helpful and harmless. SEC-
OND, on a new line, state only "A" or "B" to indicate which
response is more helpful. Your response should use the for-
mat:
Comparison: <one-sentence comparison and explanation>
More helpful: <"A" or "B">

Assistant:
```

We make the comparison with both permutations of the preference pair, and say that one response is preferred over the other if it is preferred in both orders. We use a help out dataset of 128 prompts.

Human Evaluation. For human evaluation, we use 4 human annotators, and assign them all preference pairs to annotate. The annotators are not aware of which responses come from different attack methods. We ask them to note if one response is better than the other, or if they are of similar quality. In total, the annotators annotate 50 samples for each method.

H Reward Bench Scores

We present RewardBench scores in Table 4.

I Adversarial Policy Training Curve

We present the training curve for π_{adv} in Figure 7. As shown, π_{adv} successfully optimizes (3) by increasing R_{θ_1} while minimizing R_{θ_2} . This process leads π_{adv} to generate out-of-distribution (OOD) responses that receive low gold reward. Since these responses achieve high R_{θ_1} despite their low gold reward, they qualify as adversarial samples, highlighting the effectiveness of Adv-RM.

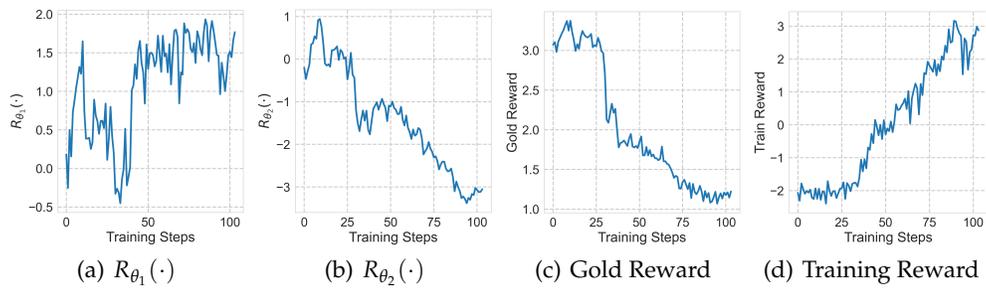


Figure 7: Adversarial policy training curve in the synthetic setting.