
UPCYCLED AND MERGED MOE REWARD MODEL FOR MITIGATING REWARD HACKING

Lingling Fu

Guangxi University

2313593009@st.gxu.edu.cn

December 2, 2025

ABSTRACT

Reward models play a critical role in Reinforcement Learning from Human Feedback (RLHF) by assessing the consistency between generated outputs and human preferences. However, conventional reward models are prone to reward hacking or over-optimization, where the policy exploits shortcut patterns to obtain high reward scores that do not reflect true human preference. Although Mixture-of-Experts (MoE)-based reward models can enhance discriminative capability, they typically introduce substantial computational overhead. To address these challenges, we propose an upcycle and merge MoE reward modeling approach. We first upcycle a dense reward model into a MoE architecture, where a shared expert captures general knowledge, while normal experts specialize in instruction-specific patterns. We then apply routing-weight normalization and merge experts back into a dense model through a learnable weight-averaging mechanism, preserving performance gains while significantly reducing inference cost. Experimental results demonstrate that our method effectively mitigates reward hacking across various model scales. Our work highlights the potential of upcycle and merge MoE structures for improving both robustness and efficiency of RLHF reward models.

1 Introduction

With the rapid development of artificial intelligence, large language models (LLMs) have achieved significant breakthroughs in solving various complex question-answering tasks, including code generation and reasoning. A key research focus is ensuring that LLM outputs align with human values across diverse tasks. Since LLMs are trained on large-scale datasets collected from multiple sources, they may produce biased or harmful outputs that conflict with human values. Therefore, aligning LLM outputs with human preferences is crucial for safety. Alignment methods mainly include supervised fine-tuning (SFT) and reinforcement learning from human feedback (RLHF). SFT fine-tunes pre-trained models using high-quality human-labeled data, which is direct and effective, quickly aligning model outputs with human preferences. RLHF first trains LLMs to produce instruction-following outputs via SFT, then trains a reward model on user preference data, and finally fine-tunes the SFT model using reinforcement learning (e.g., PPO). RLHF can capture more nuanced human preferences and iteratively improve model behavior online. Compared to SFT, RLHF can handle more complex human value alignment challenges. Despite RLHF's success in aligning LLMs with human preferences, its core reward models still suffer from reliability issues. Reward hacking often arises from overfitting to certain patterns during training, learning only shortcut features or overfitting to local input regions. While such outputs may receive high reward scores, they do not reflect true human preferences. During reinforcement learning, the policy may diverge from the training distribution of the reward model, leading to distortion of the reward evaluation and resulting in hacking or over-optimization of the reward [1]. To mitigate these issues, previous studies have proposed ensemble-based reward models and reward model structure optimization. The former improves robustness by aggregating multiple model judgments but significantly increases computational and storage costs. The latter leverages Mixture-of-Experts (MoE) to enhance reward model expressivity[2, 3], but conventional MoE introduces many additional parameters, increasing inference cost. MoE's expert structure is essentially a multi-modal learning framework, with different experts capturing distinct aspects of preferences, such as harmfulness, helpfulness, safety, fluency, or responsiveness. Reward hacking in MoE-based reward models is often caused by individual experts overfitting specific preference

dimensions, leading to anomalous scoring patterns. In this work, inspired by [4] we propose an upcycled and merged MoE reward model to alleviate reward over-optimization while reducing the computational overhead of standard MoE reward models. Specifically, we first upgrade a dense reward model to a conventional MoE structure and fine-tune it on human preference datasets, extracting rich knowledge from MoE feed-forward (FFN) layers. The routing weights of the MoE experts are then normalized. Finally, the MoE model is merged back into a dense model, retaining the benefits of MoE experts while maintaining a model size similar to the original. This merged MoE reward model can suppress high-variance responses from “hacked” experts, dilute the influence of spurious-feature experts, and effectively mitigate reward hacking during RLHF. Our contributions can be summarized as follows:

- We introduce an upcycled and merged MoE reward model to mitigate reward over-optimization, reducing computational resources compared to conventional MoE-based reward models during large-scale RLHF training.
- Through a series of experiments, we show that the number of experts affects the mitigation of reward over-optimization; generally, more experts yield better mitigation.
- We validate our approach across multiple model sizes and types on datasets such as AlpacaFarm. Evaluations using Best-of-N and PPO demonstrate that our method effectively reduces reward hacking compared to the original models.

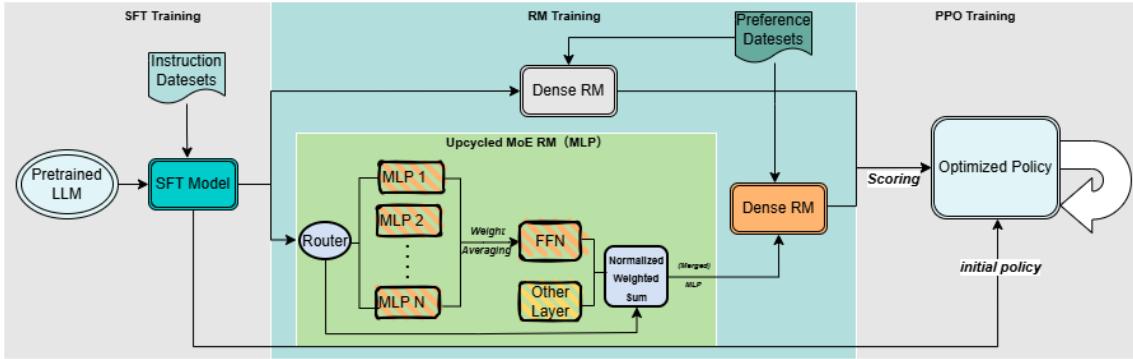


Figure 1: The RLHF training process using our method compared to the normal method

2 Background

2.1 SFT

In the RLHF training process, supervised fine-tuning (SFT) is an essential step. First, high-quality human-annotated training data are collected and used to fine-tune the pre-trained model in a supervised manner. This ensures that the SFT model can generate outputs that meet the expected text quality and format, providing high-quality positive samples for training the reward model.

2.2 Reward modeling

The reward model aims to learn human preferences from pairwise annotated data. For a given question, the responses are ranked so that those that align better with human expectations receive higher scores, while the least preferred responses receive lower scores. During reinforcement learning of large language models, the reward model assigns a reward value to each generated response. This reward reflects the quality of the model’s output and serves as a signal to guide the model toward generating responses that better match human preferences. Typically, the reward model is based on the Bradley-Terry model [5], with the objective of minimizing the following loss function:

$$\mathcal{L}(R) = -\mathbb{E}_{(x, y_w, y_l) \in D} [\log \sigma(R(x, y_w) - R(x, y_l))] \quad (1)$$

Where $D = \{x, y_w, y_l\}$ refers to a batch of pairwise human preference data, $R(x, y_w)$ refers to the reward score assigned to the selected response y_w given the prompt x , and $R(x, y_l)$ refers to the reward score for the rejected response y_l given the same prompt. $\sigma(\cdot)$ denotes the sigmoid function. By minimizing this loss function, the reward model assigns higher scores to responses that better align with human preferences.

2.3 BoN

Best-of-N (BoN) is a simple, popular, and effective preference optimization method[6]. Given a prompt x for a reference model, BoN generates N response samples Y_{ref} from the reference model, and then the reward model selects the sample with the highest reward score $R(x, y)$. As N increases, the reliability of the reward score improves, but the computational cost also increases,

$$y_{BoN}(x) = \arg \max_{y \in Y_{ref}} R(x, y) \quad (2)$$

2.4 PPO

Proximal Policy Optimization (PPO)[7] is an online reinforcement learning method based on policy gradient optimization, which maximizes a given reward function through repeated small incremental updates of the policy. PPO has become a standard RL algorithm for reinforcement learning tasks based on human feedback.

$$R^{\text{PPO}}(q, a) = R(q, a) - \beta \log \left(\frac{\pi^{\text{PPO}}(a | q)}{\pi^{\text{init}}(a | q)} \right) \quad (3)$$

where π^{PPO} refers to the policy model to be optimized, and π^{init} refers to the reference model fine-tuned via SFT. The degree of policy optimization is measured by the KL divergence between the policy model and the reference model.

2.5 Mixture-of-Experts

Feed-forward network (FFN) layers are an essential component of Transformer-based models, storing a rich amount of knowledge. Typically, a Mixture-of-Experts (MoE) network consists of multiple FFNs, where each FFN represents an expert, and a gating network controls the allocation of tokens to specific experts. MoE can effectively scale up model capacity while increasing computational cost only sublinearly[8]. For each input, only a small subset of experts is activated, allowing an MoE expert model to increase the total number of parameters with minimal computation, thereby enhancing model capability [9, 10]. MoE expert models have been shown to outperform dense models in instruction tuning tasks[11]. The output of an MoE model for a given input can be computed as:

$$y = \sum_{i=1}^n G(x)_i \cdot E_i(x) \quad (4)$$

where x refers to the input and y refers to the output. n refers to the number of feed-forward networks (FFNs), i.e., the number of experts, and i refers to the index of the i -th expert. $G(x)$ refers to the normalized routing weight assigned by the gating network to the selected experts, and $E_i(x)$ refers to the output of the i -th expert given the input x .

2.6 Reward Hacking

Since reward models are trained on human-annotated pairwise preference data, their coverage is inherently limited. As a result, when operating outside the distribution of the training data, reward models are prone to reward hacking. This phenomenon tends to occur more frequently in smaller reward models. During RL training, when a small reward model is used to score the responses generated by the policy model, we often observe that in the later stages of training, the reward model assigns high scores even to poor responses that deviate significantly from the correct answer. In addition to qualitatively inspecting the generated responses, reward hacking can also be detected by evaluating the policy model using a larger “gold” reward model and comparing the score trajectories between the two reward models. Typically, the scores from both the proxy reward model and the gold reward model increase at a similar rate during the early phase of training. After reaching a certain peak, however, the score from the gold reward model begins to decline, whereas the score from the proxy reward model continues to rise. The divergence point indicates the onset of reward hacking. We present this trend in detail in the experimental results. Gao et al.[1] found that human preference patterns are highly complex, and larger reward models generally demonstrate higher accuracy and are less susceptible to reward over-optimization. However, simply scaling up reward models is not always a practical or sustainable solution. Previous works have attempted to mitigate reward over-optimization by improving reward model design, such as reward model ensembles and structural optimization of reward models. The former aggregates scores from multiple smaller reward models, improving robustness and generalization but requiring substantial computational resources for training. The latter introduces architectures such as Mixture-of-Experts to enhance flexibility and expressiveness, thereby reducing bias induced by reward over-optimization, though it still incurs considerable training and inference cost.

3 Methods

3.1 Shared-Expert Enhanced Upcycling

To mitigate the issue of insufficient training data for each expert in traditional sparse upcycling, a shared expert E_s is added to each MoE layer and is enforced to be activated for all tokens. The remaining $K - 1$ activated experts are selected by the router from the set of normal experts. In this manner, the shared expert captures general knowledge across instructions, while the normal experts learn instruction-specific patterns. For the t -th token, the output of the MoE layer at the ℓ -th layer can be formulated as:

$$h_t^{(\ell)} = \sum_{i=1}^N g_{i,t}^{(\ell)} \text{FFN}_i^{(\ell)}(u_t^{(\ell)}) + u_t^{(\ell)} \quad (5)$$

The routing weight includes one shared expert weight and other normal expert weights:

$$g_{s,t}^{(\ell)} = 1 - s_t^{\max}(\cdot) \quad (6)$$

$$g_{i,t}^{(\ell)} = \text{Softmax}(s_{i,t}) \cdot s_t^{\max}, \quad i \neq s \quad (7)$$

where $s_{i,t}$ refers to the affinity score between the i -th expert and the t -th token, and s_t^{\max} refers to the maximum affinity score among all normal experts excluding the shared expert. The constraint $\sum_i g_{i,t} = 1$ ensures that the output scale of the upcycled MoE layer remains consistent with that of the original FFN layer.

3.2 Learnable Merging with Shared Expert Rate

The MoE layers are merged into the FFN layer via weighted averaging, while the other layers are replicated. The merged MoE structure is then converted into a dense model, eliminating inference overhead while preserving performance gains. The experts are merged using weight averaging, and a shared-expert rate λ is introduced to fix the merging weight of the shared expert:

$$W_\lambda^{(\ell)} = \lambda W_s^{(\ell)} + \sum_{i=2}^N \alpha_i^{(\ell)} W_i^{(\ell)} \quad (8)$$

constrain the normal expert coefficient,

$$\sum_{i=2}^N \alpha_i^{(\ell)} = 1 - \lambda \quad (9)$$

This approach ensures that while maintaining the consistent contributions of shared experts, other experts can also learn specific knowledge.

4 Related Work

4.1 RLHF

Reinforcement Learning from Human Feedback (RLHF) is a crucial technique for aligning the behavior of large language models with human values and preferences. The core idea involves collecting human preference data to train a reward model (RM), and leveraging reinforcement learning—typically Proximal Policy Optimization (PPO), to guide the model toward producing outputs that better match human expectations[6, 12]. Existing research has demonstrated that RLHF achieves remarkable performance across a variety of tasks, including general question answering[13], mathematical reasoning[14], code generation[15] and safety alignment[16]. By incorporating human feedback signals, RLHF improves both the coherence and naturalness of model responses, while also reducing the generation of unsafe or irrelevant content. However, since reward models are trained from limited data, they can suffer from reward hacking, a phenomenon that is more pronounced in smaller reward models. To analyze this effect, a larger reward model is typically used as a “gold standard” surrogate for human evaluation[1].

4.2 Upcycled MoE Model

Training large deep neural network models is generally very costly. In recent years, an increasing number of researchers have attempted to decouple model size from computational cost by adopting sparse activation models. However, training

sparse models from scratch still requires substantial computational resources. Many researchers have employed upcycled MoE models to address these challenges. Komatsuzaki et al. [17] proposed initializing sparse activation MoE models from dense checkpoints to reduce training costs, ultimately achieving performance at only 50% of the cost of dense pre-trained models. Ding et al.[4] observed that conventional sparse upcycled MoE models are insufficient for effective instruction tuning, and introduced a shared-expert mechanism combined with a novel routing-weight normalization strategy to enhance instruction-tuning capability. Xue et al. [18] proposed extracting integrated knowledge from sparse expert teacher models and then distilling this knowledge into a dense student model.

4.3 Mitigating Reward Hacking

Previous research has focused on mitigating reward hacking through approaches such as optimizing model training methods[19], reward model ensembles [13, 20], and structural optimization of reward models [21]. Qin et al. [19] proposed a reward-confidence-based RM training method aimed at reducing scoring errors caused by ambiguity or noise in the training labels. Yang et al. [21] improved out-of-distribution (OOD) generalization by regularizing hidden states and integrating a text-generation loss on the model head, enabling the RM to preserve generative capabilities while learning reward values conditioned on the same hidden representation. Most existing approaches mitigate reward over-optimization via reward model ensembles. Coste et al. [20] first introduced reward ensemble methods to combat over-optimization by training multiple reward models and comparing three aggregation strategies: Mean Optimization, Worst-Case Optimization, and Uncertainty-Weighted Optimization. Ahmed et al. [13] modified the conventional ensemble approach by sharing a common backbone with multiple linear heads instead of training separate reward models, achieving comparable mitigation effects while reducing training time. Zhang et al. [22] proposed ensemble strategies based on linear-layer aggregation and LoRA-based aggregation, both using mean optimization, and demonstrated superior performance for the LoRA-based ensemble. Ramé et al. [23] used weight-averaging of reward models pretrained under different conditions to improve OOD robustness; however, it requires substantial computational resources. Eisenstein et al. [24] argued that simple ensembling is insufficient to fully eliminate reward hacking, emphasizing the importance of diversity among reward models. More recently, Quan et al. [2] introduced MoE architecture into reward modeling for the first time, proposing a two-level MoE-based RM in which a sparse outer MoE routes samples to dense inner experts, and rewards are merged at the MLP layer. Another line of work [3] evaluated responses across multiple reward dimensions, using a MoE gating network to assign weights based on prompt semantics, and combining them into a final scalar reward. Although these ensemble-based reward models can effectively score responses during RL, they incur considerable computational cost, typically requiring multiple GPUs for parallel inference. In contrast, our approach trains a reward model that can run efficiently on a single 80GB GPU, significantly reducing compute requirements while effectively mitigating reward hacking during PPO.

5 Experimental Setup

5.1 Datasets

We conduct our experiments using the AlpacaFarm dataset [25] to evaluate the effectiveness of our method in mitigating reward hacking. AlpacaFarm is a high-quality dataset containing both open-ended and closed-form questions, and has been widely used in prior RLHF training and evaluation studies[20, 26, 27]. We use the alpaca_instructions subset, selecting 10k samples for training and 2k samples for evaluating the SFT model. For training and evaluating reward models, we adopt the alpaca_human_preference pairwise preference dataset. During PPO training, we use 20k unlabeled samples for optimization and 2k samples for evaluation. In our Best-of-N experiments, we use the alpaca_human_evaluation dataset. Additionally, we compare the scoring accuracy of different reward models on Anthropic’s Helpful & Harmless (HH) dataset [28] and the WebGPT dataset [29]. For both HH and WebGPT, we select 10k samples from the training split for model training and 1k samples from the test set for evaluating accuracy.

5.2 Models

To evaluate the effectiveness of our method across different model configurations, we experiment with models of varying sizes and architectures. Specifically, we train our optimized MoE reward model on three base models—Qwen2.5-0.5B, TinyLlama-1.1B, and Pythia-1.4B. We use Llama-3-8B as the gold-standard reward model, whose outputs serve as a proxy for human preference judgments.

5.3 Experimental Results

5.3.1 PPO

In our PPO experiments, we set the number of training steps to 3000, use a fixed random seed of 22, and adopt a learning rate of 1e-5. For the upcycle and merge MoE reward model, we evaluate reward models with 2, 4 and 6 experts, respectively, and compare their performance throughout the PPO training process, as shown in Figure 2.

For the ensemble reward model setting, following the findings of [20], we train four reward models with different random seeds, as the ensemble of four models yields the best performance. We compare three ensemble Mean Optimization, Worst-Case Optimization, and Uncertainty-Weighted Optimization during PPO training.

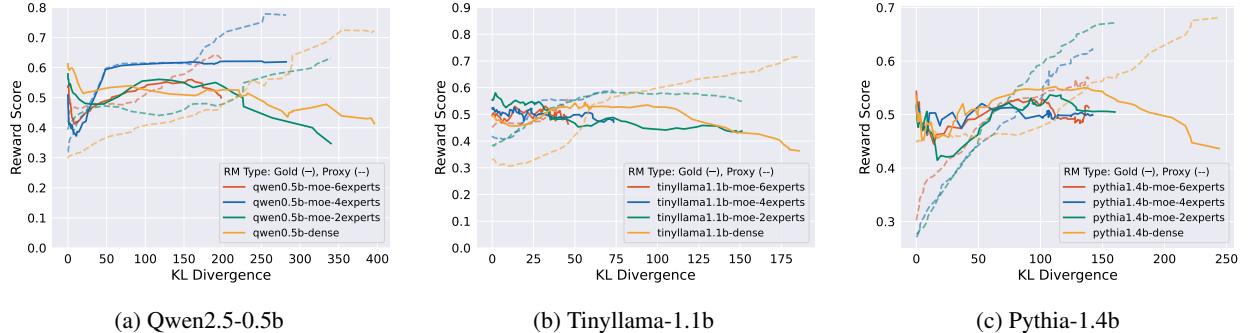


Figure 2: Comparison results for all three models

5.3.2 Accuracy

We also evaluated the accuracy performance of different models on the Anthropic and WebGPT datasets, as shown in Table 1. And we also compared the accuracy of different reward model types based on the Tinyllama-1.1b model, as shown in Table 2.

Models	Anthropic		WebGPT
	Harmful	Helpful	
Qwen-0.5b			
Qwen-0.5b (dense)	<u>45.6</u>	<u>44.7</u>	<u>57.2</u>
Qwen-0.5b (2-experts)	48.9	50.5	58.5
Qwen-0.5b (4-experts)	51.1	50.1	58.7
Qwen-0.5b (6-experts)	53.9	52.3	58.4
TinyLlama-1.1b			
TinyLlama-1.1b (dense)	54.2	53.4	58.2
TinyLlama-1.1b (2-experts)	56.4	54.1	58.9
TinyLlama-1.1b (4-experts)	58.0	55.2	58.9
TinyLlama-1.1b (6-experts)	57.5	57.7	60.8
Pythia-1.4b			
Pythia-1.4b (dense)	52.6	52.2	<u>56.5</u>
Pythia-1.4b (2-experts)	54.6	53.0	57.3
Pythia-1.4b (4-experts)	54.2	53.4	57.1
Pythia-1.4b (6-experts)	55.1	53.9	57.7

Table 1: The bold values refer to the highest scores, while the underlined values refer to the lowest scores.

5.3.3 BoN

In addition to using PPO to examine whether the reward model mitigates reward hacking, we also evaluate the effectiveness of combining the reward model with the commonly used Best-of-N (BoN) technique in RLHF. In the BoN

Models	Anthropic		WebGPT
	Harmful	Helpful	
TinyLlama-1.1b (dense)	<u>54.2</u>	53.4	58.2
TinyLlama-1.1b (2-experts)	56.4	54.1	58.9
TinyLlama-1.1b (4-experts)	58.0	55.2	58.9
TinyLlama-1.1b (6-experts)	57.5	57.7	60.8
UnMerged MoE RM (2-experts)	56.8	53.9	<u>58.1</u>
UnMerged MoE RM (4-experts)	58.2	55.8	59.1
UnMerged MoE RM (6-experts)	57.4	56.2	60.9
Mean Optimization (4-RMs)	59.0	57.2	60.8
Worst-Case Optimization (4-RMs)	60.4	55.9	60.6
Uncertainty-Weighted Optimization (4-RMs)	60.1	58.5	59.6

Table 2: The bold values refer to the highest scores, while the underlined values refer to the lowest scores.

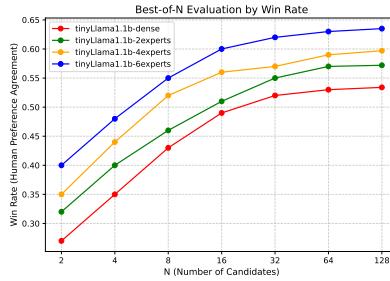


Figure 3: the BoN result of Tinyllama-1.1b

setting, the SFT model generates N candidate responses for each prompt, and the reward model selects the sample with the highest predicted reward. We conduct BoN experiments with $N=\{2,4,8,16,32,64,128\}$.

5.4 Discussion

Through PPO and BoN experiments, we demonstrate that reward models modified using our method can effectively mitigate reward hacking.

In the PPO experiments, reward models with more MoE experts perform better during the RL phase, exhibiting lower KL divergence, and the proxy score and gold score curves are closer, indicating smaller deviations between the policy model and the ground-truth reward. In contrast, MoE models with fewer experts show larger deviations between proxy and gold scores and higher KL values. Observations indicate that when KL exceeds 150, the model tends to generate meaningless responses. The Qwen2.5-0.5B model, being relatively small, consistently reaches higher KL values, and its outputs beyond $KL = 150$ contain many nonsensical answers. Among the three base models, TinyLlama-1.1B achieves the best performance, reflecting its inherent advantage. For PPO training with the three types of ensemble reward models, Uncertainty-Weighted Optimization performs the worst, while Mean Optimization performs the best. The upcycle and merge MoE reward models with 6 and 4 experts outperform the three ensemble reward models, indicating that our method produces reward models that more effectively mitigate reward hacking during PPO.

In the BoN experiments, we observe that the model’s win rate generally increases with larger values of N , gradually leveling off as N becomes sufficiently large. For the TinyLlama-1.1B model, the win rate grows rapidly before $N=16$ and then begins to plateau. We also find that reward models with more experts achieve higher final win-rate scores. In particular, MoE-based reward models obtain higher win rates than the original dense model, indicating that the reward models trained with our method produce reward signals that better align with human preferences in the BoN setting.

In the industry, reward model performance is commonly evaluated by accuracy on a validation set, measuring how well the model aligns with human-annotated preference data. Some studies have noted that accuracy alone does not fully reflect the true effectiveness of reward models [30]. As shown in Table 1, models with higher reward accuracy do not necessarily perform better during PPO training. For example, the Qwen2.5-0.5B model has relatively high reward accuracy but exhibits large KL deviations during PPO. In contrast, reward models modified using our upcycle and merge MoE approach achieve higher accuracy scores than the original models, while also demonstrating improved

performance in both PPO and BoN experiments. This further validates that our upcycle and merge MoE reward models effectively mitigate reward hacking.

Compared to the original dense reward models, MoE reward models optimized with our method achieve superior performance in both PPO and BoN experiments, confirming the effectiveness of our approach in alleviating reward hacking.

6 Conclusions

In this work, we adopt an upcycle and merge MoE approach to modify reward model training, addressing the high resource consumption of traditional ensemble-based and conventional MoE reward models during training and inference. Experimental results show that this modified reward model can effectively mitigate reward hacking in RLHF.

References

- [1] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR, 2024.
- [2] Shanghaoran Quan. Dmoerm: Recipes of mixture-of-experts for effective reward modeling. *arXiv preprint arXiv:2403.01197*, 2024.
- [3] Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. *arXiv preprint arXiv:2406.12845*, 2024.
- [4] Yifeng Ding, Jiawei Liu, Yuxiang Wei, Terry Yue Zhuo, and Lingming Zhang. Xft: Unlocking the power of code instruction tuning by simply merging upcycled mixture-of-experts. *arXiv preprint arXiv:2404.15247*, 2024.
- [5] Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39:324, 1952.
- [6] 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.
- [7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [8] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.
- [9] Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Moefication: Transformer feed-forward layers are mixtures of experts. *arXiv preprint arXiv:2110.01786*, 2021.
- [10] Jiamin Li, Qiang Su, Yitao Yang, Yimin Jiang, Cong Wang, and Hong Xu. Adaptive gating in mixture-of-experts based language models. *arXiv preprint arXiv:2310.07188*, 2023.
- [11] Sheng Shen, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, and Xinyun Chen. Mixture-of-experts meets instruction tuning: A winning combination for large language models. *arXiv preprint arXiv:2305.14705*, 2023.
- [12] 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.
- [13] Ahmed M Ahmed, Rafael Rafailov, Stepan Sharkov, Xuechen Li, and Sanmi Koyejo. Scalable ensembling for mitigating reward overoptimisation. *arXiv preprint arXiv:2406.01013*, 2024.
- [14] Huayu Chen, Kaiwen Zheng, Qinsheng Zhang, Ganqu Cui, Yin Cui, Haotian Ye, Tsung-Yi Lin, Ming-Yu Liu, Jun Zhu, and Haoxiang Wang. Bridging supervised learning and reinforcement learning in math reasoning. *arXiv preprint arXiv:2505.18116*, 2025.
- [15] Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Quentin Carbonneau, Taco Cohen, and Gabriel Synnaeve. Rlef: Grounding code llms in execution feedback with reinforcement learning. *arXiv preprint arXiv:2410.02089*, 2024.
- [16] 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. *arXiv preprint arXiv:2310.12773*, 2023.

- [17] Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. Sparse upcycling: Training mixture-of-experts from dense checkpoints. *arXiv preprint arXiv:2212.05055*, 2022.
- [18] Fuzhao Xue, Xiaoxin He, Xiaozhe Ren, Yuxuan Lou, and Yang You. One student knows all experts know: From sparse to dense. *arXiv preprint arXiv:2201.10890*, 2022.
- [19] Bowen Qin, Duanyu Feng, and Xi Yang. Towards understanding the influence of reward margin on preference model performance. *arXiv preprint arXiv:2404.04932*, 2024.
- [20] Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. *arXiv preprint arXiv:2310.02743*, 2023.
- [21] Rui Yang, Ruomeng Ding, Yong Lin, Huan Zhang, and Tong Zhang. Regularizing hidden states enables learning generalizable reward model for llms. *Advances in Neural Information Processing Systems*, 37:62279–62309, 2024.
- [22] Shun Zhang, Zhenfang Chen, Sunli Chen, Yikang Shen, Zhiqing Sun, and Chuang Gan. Improving reinforcement learning from human feedback with efficient reward model ensemble, 2024. URL <https://arxiv.org/abs/2401.16635>, 2024.
- [23] 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.
- [24] Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D’Amour, DJ Dvijotham, Adam Fisch, Katherine Heller, Stephen Pfahl, and Deepak Ramachandran. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking. *arXiv preprint arXiv:2312.09244*, 2023.
- [25] 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.
- [26] Yuchun Miao, Sen Zhang, Liang Ding, Rong Bao, Lefei Zhang, and Dacheng Tao. Inform: Mitigating reward hacking in rlhf via information-theoretic reward modeling. *Advances in Neural Information Processing Systems*, 37:134387–134429, 2024.
- [27] Hao Lang, Fei Huang, and Yongbin Li. Fine-tuning language models with reward learning on policy. *arXiv preprint arXiv:2403.19279*, 2024.
- [28] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, and Tom Henighan. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.
- [29] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, and William Saunders. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.
- [30] Xueru Wen, Jie Lou, Yaojie Lu, Hongyu Lin, Xing Yu, Xinyu Lu, Ben He, Xianpei Han, Debing Zhang, and Le Sun. Rethinking reward model evaluation: Are we barking up the wrong tree? *arXiv preprint arXiv:2410.05584*, 2024.