Finetuning PPO with RLHF

Luke Chin A Foeng
14603373

Abstract

This paper combines Proximal Policy Optimization (PPO) [15] with Reinforcement Learning with Human Feedback (RLHF) through the use of both the Bradley-Terry model [1] for binary ranking, and a k-wise generalization for tertiary and quaternary ranking. The primary goal of this paper is to explore the benefits that RLHF provides when used to finetune a PPO model within the Car Racing Gymnasium environment [2], while also exploring the differences between binary, tertiary and quaternary trajectory ranking. Ultimately, the combination of RLHF and PPO was able to improve over the original PPO model, yielding large, consistent increases in the score obtained within a trajectory. This improvement can be attributed to the addition of human trajectory classification involved in the RLHF component of the combined model. Additionally, the extension of the performed (expectedly) worse than their binary counterpart. This yields results in line with previous literature, but not seen in the combined setting of PPO and RLHF.

1 Introduction

Reinforcement Learning with Human Feedback (RLHF) has emerged as a promising approach to improve the performance and safety of reinforcement learning (RL) models in recent years. This methodology has been employed in a number of Large Language Models' (LLM) training processes, particularly those employed in [12]. This process is normally employed as a finetuning step after a model has already been pre-trained. Proximal Policy Optimization (PPO) models have been used in a number of applications, and have been able to learn policies in a manner that is similar to trust region policy optimization (TRPO), but is much more computationally efficient due to its quick convergence and sample efficiency. This paper aims to extend the results of a PPO model implemented on the OpenAI Gym environment Car-Racing-V2 [2][10] through the addition of an RLHF regime, ultimately exploring the effects of the simulated human labeling, as well as the impact that multi-trajectory ranking would have on said results. In the end, the experimentation resulted in the confirmation of the hypothesis that RLHF can be used in this scenario as a post-training regime, improving over the results of the vanilla PPO model. The code for this paper can be found in the link provided in the footnote. [1]

2 Background & Related Work

In this section, Proximal Policy Optimization and Reinforcement Learning with Human Feedback will be discussed to better frame the experiments conducted within this paper.

2.1 PPO

Proximal Policy Optimization (PPO) was first presented by Schulman et al. [15], and built off of Trust Region Policy Optimization, another paper authored by Schulman et al. [14]. These policy

[1]https://github.com/lukegtc/HitLML_PPO_RLHF

gradient methods aim to learn some policy $\pi(a|s,\theta)$, where a denotes the selected action based on the policy parameters θ and the state s by maximizing the expectation of the reward function, denoted as $J(\theta) = \mathbb{E}_{\pi}\left[r(\tau)\right]$, where τ is some trajectory.
TRPO makes use of the surrogate advantage equation, which calculates the difference between two reward values, one derives from the original policy parameters, and the other derived from the original policy parameters shifted by some amount$\Delta\theta$. The equation for this difference is shown in Equation 1, where A^{π_θ} is the advantage for the policy π_θ, which finds the relative value of the state-action pair [14].

$$J(\pi_{\theta+\Delta\theta}) - J(\pi_\theta) \approx \mathbb{E}_{s\ \rho_{\pi_{theta}}}\left[\frac{\pi_{\theta+\Delta\theta}(a|s)}{\pi_\theta(a|s)}A^{\pi_{theta}}(s,a)\right] \tag{1}$$

There are some issues with this methodology, mainly that it is computationally complex, and struggles when the model it is used in conjunction with has dropout or parameter sharing [7]. Proximal Policy Optimization aimed to solve these issues in a less complex manner than the TRPO method.
PPO, and particularly the clipped variant of PPO, makes use of the loss function shown in Equation 2.

$$\mathcal{L}_{\pi_\theta}(\pi_{\theta_{old}}) = \mathbb{E}_{\tau\ \pi_\theta}\left[\sum_{t=0}^{T}[min(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}A_t^{\pi_{\theta_{old}}}, clip(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon)A_t^{\pi_{\theta_{old}}})]\right] \tag{2}$$

This cost function is the clipped form of the equation Equation 3, which is the conservative policy iteration (CPI) cost function [8]. When this equation is left un-clipped, it tends to induce large update steps. Since the new cost function is bounded within the range $[1-\epsilon, 1+\epsilon]$, it no longer has the option to travel out of this interval. The minimum is then taken between the original CPI cost function and the new clipped cost function so as to lower bound the CPI cost function.

$$\mathcal{L}(\theta) = \mathbb{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}A_t\right] \tag{3}$$

This methodology improves over the previous TRPO methods computationally because of this clipping method, allowing for smooth updates.

2.2 RLHF

Reinforcement learning with human feedback (RLHF) has its roots in the Inverse Reinforcement Learning (IRL) and the Bradley-Terry model [1]. The 'inverse' portion of the name Inverse Reinforcement Learning originates from the fact that the reward function is assumed to already be obtained. This reward function is subsequently used to define a policy as well. In IRL, it is also assumed that obtaining the reward function is much easier than obtaining the policy [11]. The algorithmic thought process involved in IRL starts off with the optimization of a reward function's parameters using demonstrations obtained from some human experts. After obtaining this reward function, a policy is fit to said reward function using reinforcement learning. This new policy is then compared to the expert policy, and if the policies vary significantly, this process is repeated. This process, however, is quite computationally expensive, which is why Reinforcement Learning with Human Feedback (RLHF) [7] was devised.
The RLHF methodology takes human demonstrations and learns some reward function from them, and subsequently fine tunes an underlying model. This is done through a binary ranking operation, where both the model and the human annotator are shown some dataset of positive and negative trajectories, where one pair is shown and ranked by both the annotator and the model at a time, and subsequently used to further learn some policy. We can exemplify this by assuming that some policy generates some set of trajectories $\{\tau^1, \tau^2, ..., \tau^n\}$ where this policy has its parameters optimized to obtain some maximum reward. After this, pairs are selected from these trajectories, where one trajectory has a higher reward than the other. From here, the model conducts a softmax function on the rewards obtained, as shown in Equation 4, where we take the softmax over the summation over the rewards of the trajectory pair.

$$\hat{P}\left[\sigma^1 \succ \sigma^2\right] = \frac{exp(\sum_t \hat{r}(s_t^1, a_t^1))}{exp(\sum_t \hat{r}(s_t^1, a_t^1)) + exp(\sum_t \hat{r}(s_t^2, a_t^2))} \tag{4}$$

The human expert preferences are then defined as being over some distribution, where the probability of the annotator selecting trajectory 1 out of the two trajectories in this binary example is $\mu(1)$ and

the probability of said annotator selecting trajectory 2 is $\mu(2)$.
A binary cross-entropy loss function can then be calculated between the human and the model, as was described in the paper introducing RLHF [7], shown in Equation 5.

$$\mathcal{L}_{CE} = -\sum_{(\sigma^1,\sigma^2,\mu)\in\mathcal{D}} \mu(1)log(\hat{P}\left[\sigma^1 \succ \sigma^2\right]) + \mu(2)log(\hat{P}\left[\sigma^2 \succ \sigma^1\right]) \tag{5}$$

For the tertiary and quaternary ranking RLHF formulation, the following equation is formulated, where you simply take the probability of an option being the best (ranked higher than all the other options) [3], and perform a cross entropy on this. This equation is shown in Equation 6. This is a k-wise Bradley-Terry model, where the value $\hat{P}\left[\sigma^1 \succ \sigma^{2,3,\ldots,k}\right]$ is described in Equation 7.

$$\begin{aligned}\mathcal{L}_{CE} = -\sum_{(\sigma^1\ldots\sigma^k,\mu)\in\mathcal{D}} \mu(1)log(\hat{P}\left[\sigma^1 \succ \sigma^{2,3,\ldots,k}\right]) + \mu(2)log(\hat{P}\left[\sigma^2 \succ \sigma^{1,3,\ldots,k}\right]) \\ +\ldots + \mu(k)log(\hat{P}\left[\sigma^k \succ \sigma^{1,2,\ldots,k-1}\right])\end{aligned} \tag{6}$$

$$\hat{P}\left[\sigma^1 \succ \sigma^2, \ldots, \sigma^k\right] = \frac{exp(\sum_t \hat{r}(s_t^1, a_t^1))}{\sum_{i=1}^k exp(\sum_t \hat{r}(s_t^i, a_t^i))} \tag{7}$$

2.3 Previous Similar Implementations

Recently, there has been a growing interest in combining RLHF with PPO, as this is a methodology employed in the final stages of Large Language Model (LLM) training regimes [17]. Notably, this RLHF variation of PPO is a method employed by OpenAI as a finetuning step in their model ChatGPT4 [12]. This can also be seen as a model alignnment step. Additionally, an implementation of a variation of the Plackett-Luce[13][9] model (similar to the k-wise Bradley-Terry model) used in conjunction with PPO was recently implemented in the LLM space [18]. This paper saw an increased efficiency when using a k-wise comparison, as opposed to a binary comparison.

3 Method(s)

In this paper, RLHF is employed in addition to PPO in order to further optimize the trajectory of a race car around a racing tracking, in the OpenAI Gymnasium environment. This gym environment consists of a state-space of 96×96 RGB pixels, and a continuous action space that allows for steering, gas and braking, where the steering exists on a range $[-1, 1]$ with -1 being a $90°$ turn to the left and 1 being a $90°$ turn to the right. The gas spans $[0, 1]$ and the brake spans $[0, 1]$ as well. The action space also has a "die" state and a "done" state. A reward of -0.1 is given out for every frame generated. The actor receives $\frac{1000}{N}$ points for N track tiles visited. In addition to this, the "die" state obtains a reward of 100 so as it encourages the car to visit as many tiles as possible, even if it ends up travelling off the screen. There is also a -0.05 penalty received for all green pixels, so as to encourage the car to minimize time spent off the track.
The PPO training loop functions similarly to what is described above. With the current policy, the model selects an action, from which the environment will select a new state. This is done for a set number of trajectories of length N. The reward for this new trajectory is then added to memory, along with the state and action sets that produced this reward. These trajectories then generated until a done or die state is achieved, after which the cycle continues, but with trajectories being generated starting from the last state generated in the previous cycle.
Once the memory of the model is filled with a set number of trajectories, the model training process begins. This training process makes use of the clipped PPO loss function described in subsection 2.1. The RLHF loss is also included within this section through a simple summation of the loss values.

The RLHF portion of this model is implemented by first taking all the generated trajectories and pairing them with on another, creating a set of pairs containing all possible trajectory combinations, and obtaining the softmax of each pair of reward summations. Once these values are obtained, they are passed through a cross entropy loss function, then subsequently added to the original PPO loss function, to penalize or reward the entropy within the model.
The agent probabilities are found by simply applying a softmax function to the pair of reward summations, yielding the probabilities of either trajectory occurring, as described in section 2. Within

the human probability function, a bit of gaussian noise is added to the incoming rewards, as a human would not have access to the rewards generated by a certain trajectory. This gaussian noise is meant to simulate the change in medium from the reward function, to a set of images, and finally to a short clip meant for a human annotator to interpret. Additionally, the same process of having a 10% chance of generating a random distribution is also included, as was first implemented within the original RLHF paper. The training process is shown in Figure 1.

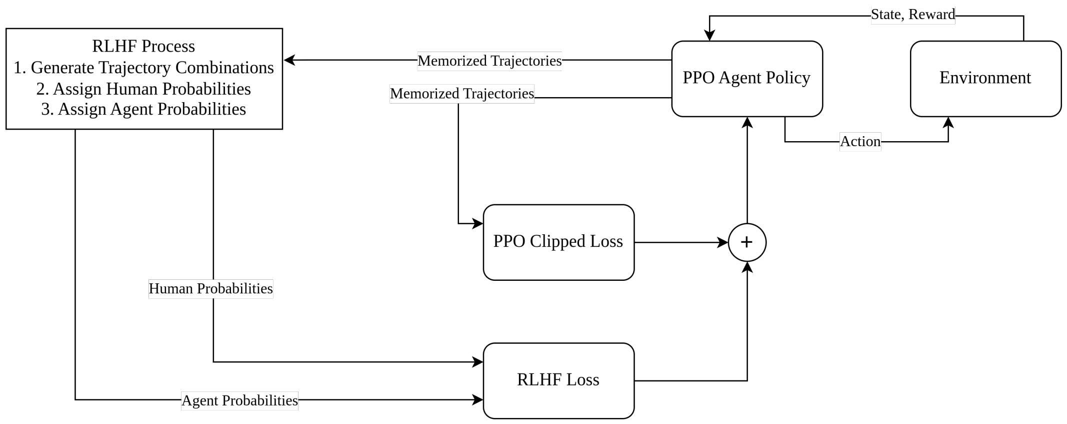


Figure 1: A Depiction of the Model Training Process

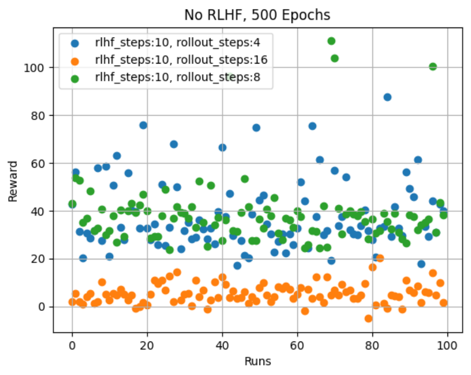
4 Experimental Results

The experiments conducted were mainly organized into two separate categories. The first category includes those only making use of the PPO framework of the original model. This was then used as a baseline for the experiments conducted through the use of the combination of PPO and RLHF. For the first set of experiments, the effects of the RLHF implementation were assessed, where results were generated so as to compare not only the effect of RLHF on PPO, but also the variations in parameters. In addition to this first set of experiments, a second set of experiments were also conducted, comparing the vanilla binary ranking methodology of RLHF to tertiary and quaternary ranking methodologies.

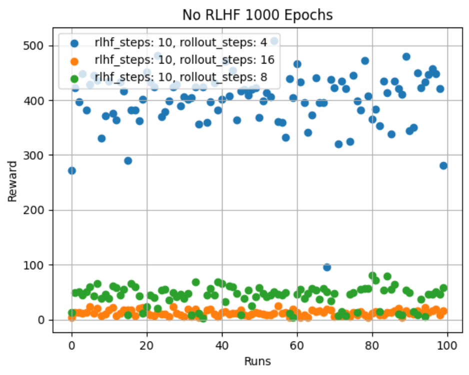
4.1 PPO+RLHF

The PPO framework had several parameters that were adjusted over the course of the trials. These included the rollout steps, number of states, and number of epochs. The rollout steps parameter denotes the number of frames the model generates during each state. The frames are then subsequently passed into the agent model, to then be used to select an action. The number of states defines the number of state-action pairs the model must iterate through during each epoch. Throughout the vanilla model baseline experiments, the parameters that were modified included the rollout steps and the number of epochs. For all experiments, the number of states generated during the training process remained fixed at 500.

When RLHF was employed, the parameters altered also included the number of RLHF steps and the number of rollouts. The number of RLHF steps remained fixed at 10 steps for all experiments. Only the number of rollout steps were altered in these experiments.
For the initial baselines, only the number of rollout steps was altered for two rounds of experiments trained with 500 and 1000 epochs. The results for these two experiments can be seen in Figure 2a and Figure 2b, respectively, where each point on the plots represents a test run. These test runs were conducted on randomly initialized tracks, over the course of 500 time steps each. The reward achieved per run is the total reward across the entire run. Note that out of the configurations, the 1000 epoch, 4 rollout step configuration had the best performance when no RLHF was employed, as well as when RLHF was employed. This is due to an increased granularity in measurement when it comes to the PPO model, as fewer frames are now required for an action to be selected, resulting in higher rewards during the testing phase. Although this yields the best results, the use of



(a) 500 Epochs without RLHF



(b) 1000 Epochs without RLHF

only 4 rollout steps would be too few frames for a human annotator to discern between different trajectories, as the frame rate for these trajectories is set at 10 frames per second, for one second [6]. Therefore, only the rollout step size of 8 and 16 will be used in the tertiary and quaternary ranking RLHF experiments, as these are close enough to the minimum threshold. Note that the number of frames selected for each rollout is a power of two. This is a requirement for efficient processing of the trajectories by the PPO model. From these results, we see that the 500 epoch variant saw little to no improvement after the implementation of the RLHF algorithm. The 1000 epoch variant of this experiment clearly displays a substantial improvement over the vanilla PPO implementation, where all three rollout step variants saw significant improvements. The results are further specified in Table 1, which displays the mean and standard deviation of each experiment group.

4.2 PPO+RLHF Tertiary and Quaternary Ranking

The second experiment set conducted involved altering the number of options the modeled human annotator had available. This ultimately led to the results presented in Figure 4.

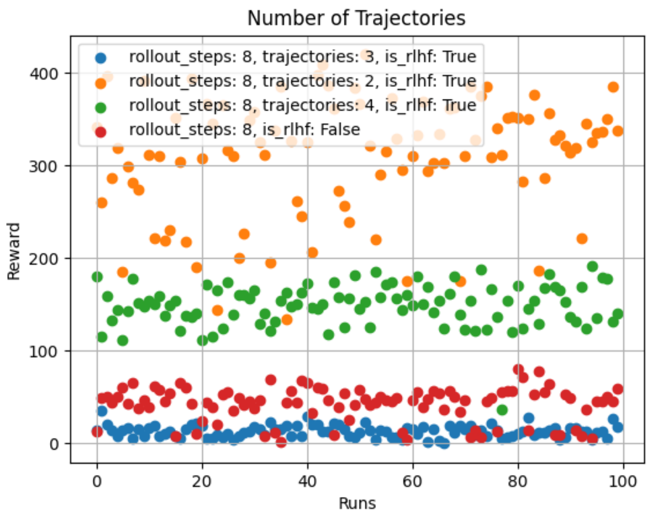


Figure 4: Results of tertiary and quaternary ranking RLHF variant.

We can see that performance was reduced after introducing more than two trajectories to the annotator. This can be attributed to the increase in complexity of the task, similar to the reduction in accuracy a human annotator would attribute.

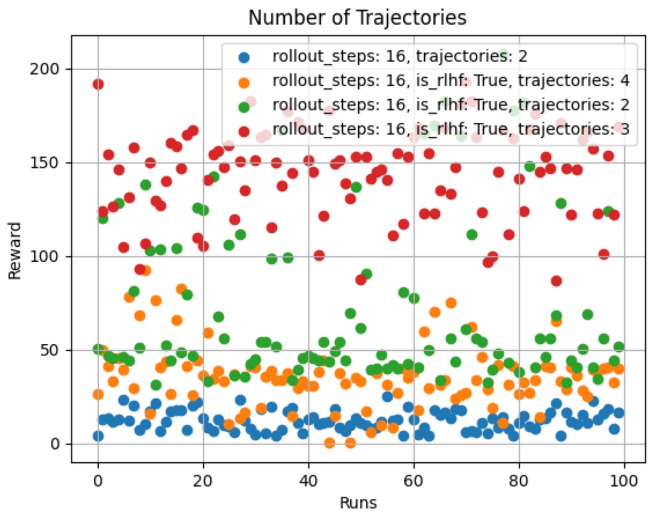


Figure 5: Results of tertiary and quaternary ranking RLHF variant.

However, when extending the trajectory from 8 rollout steps to 16, we see an increase in the reward of the tertiary variation, but not the quaternary variant. This peculiar result could be attributed to an increase in noise in the results between the two variants, but this requires an additional level of investigation in future extensions of this work. The increase in trajectories still improves over the original vanilla PPO model, just as the bianry classification RLHF+PPO model did. [5]

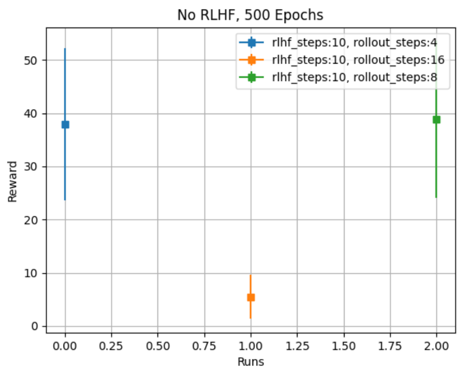
5 Conclusions, Limitation, & Future Work

Ultimately, this paper provided an interesting set of experiments combining PPO and the RLHF Bradley-Terry, as well as k-wise Bradley-Terry models in the Gym Car Racing environment. It was found that the implementation of the RLHF Bradley-Terry model did improve the clipped Proximal Policy Optimization trajectories, since improvements were seen across all variations of the trajectories in the initial set of 1000 epoch experiments, while also improving the results for most 500 epoch experiments. In addition to these first experiment set results, the second experiment set yielded some peculiar findings. Namely, improvements over the baseline clipped PPO model only occurred in the binary, tertiary and quaternary trajectory ranking experiments only when the trajectories were short enough, as the 8 rollout step variants all performed better than the baseline, while this was only the case for some of the 16 rollout step variants. This was attributed to the number of epochs spent training the variants. It would be an interesting extension to see how exactly an increase in the number of epochs would impact model variants with more trajectory rollout steps, as there is an obvious correlation with the increase in performance of trajectory sets with more steps and the number of epochs spent training on these experiments. This becomes even more apparent when viewing the renderings of the tests, where agents trained with longer trajectories simply moved slower than those that were trained with shorter trajectories. Some setup improvements could be implemented to expand upon the findings of this report. Namely, extending the number of epochs, or creating a more accurate human model through the use of more sophisticated modeling methods. Additionally, while a cross entropy approach was used to train the RLHF portion, a different entropy-based loss function could also be employed, making use of more descriptive matching of the targeted behavior and generated trajectories.

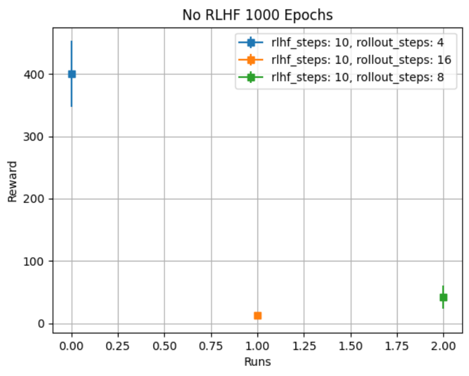
While conducting further research on the topic of clipped PPO models, an adaptive clipped PPO model was found. This may be another interesting extension to this work, as it would be interesting to see the impact of the sample efficiency benefits that such a model provides to this report's RLHF implementation [16]. A weak preference structure has also been previously been implemented in a MuJoCo games setting [4], where a modeled or real human annotator is provided pairs of trajectory segments, which they must assign to one of five categories, preferred, not preferred, equivalent, weakly preferred, weakly not preferred. Such a ranking methodology may provide more context to the PPO model than simply using a best-trajectory selection methodology as was employed in this report.

Appendices

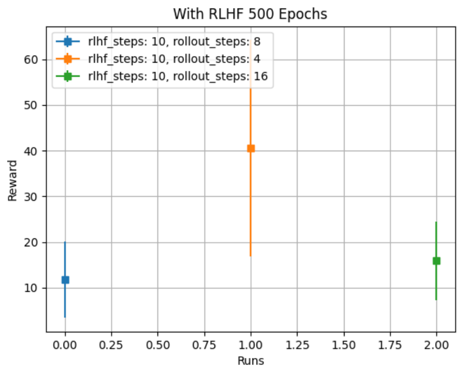
A RLHF Experiment Means and Errors



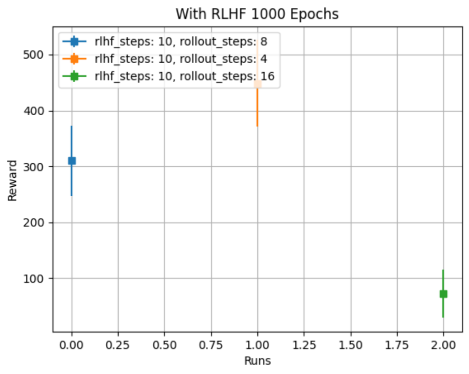
(a) 500 Epochs without RLHF



(b) 1000 Epochs without RLHF



(a) 500 Epochs with RLHF



(b) 1000 Epochs with RLHF

	Epochs	Rollout Steps	Mean	Standard Deviation
No RLHF	500	4	37.90	14.24
		8	38.81	14.74
		16	5.47	4.16
	1000	4	400.10	52.66
		8	42.14	18.73
		16	12.24	5.21
RLHF	500	4	40.51	23.69
		8	11.73	8.34
		16	15.90	8.58
	1000	4	448.90	77.13
		8	310.11	63.19
		16	72.07	42.92

Table 1: Mean and Standard Deviation of each experiment

B RLHF Variants Experiment Means and Errors

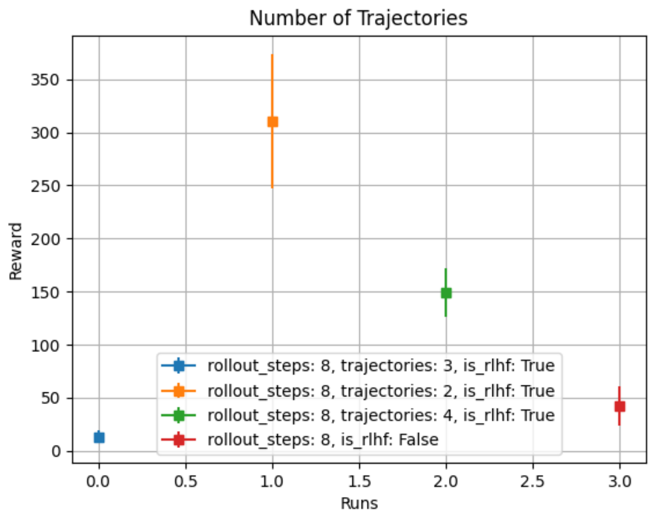


Figure 8: The mean and error values attained when adjusting the number of samples the model and classifier each had to classify for 8 step trajectories.

	Epochs	Number of Trajectories	Mean	Standard Deviation
No RLHF	1000	2	42.14	18.73
RLHF	1000	2	310.11	63.19
		3	13.31	6.30
		4	149.21	22.54

Table 2: Mean and Standard Deviation of each trajectory set experiment for 8 rollout steps

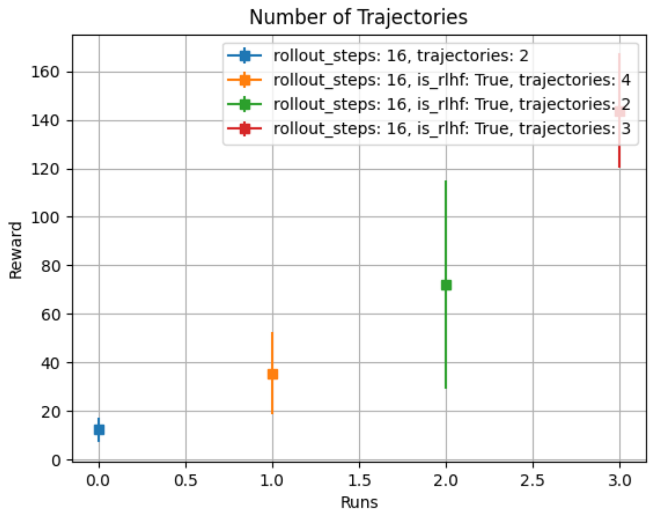


Figure 9: The mean and error values attained when adjusting the number of samples the model and classifier each had to classify for 16 step trajectories.

	Epochs	Number of Trajectories	Mean	Standard Deviation
No RLHF	1000	2	12.24	5.21
RLHF	1000	2	72.07	42.92
		3	143.67	23.61
		4	35.57	16.95

Table 3: Mean and Standard Deviation of each trajectory set experiment for 16 rollout steps

References

[1] 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.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[3] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR, 2019.

[4] Zehong Cao, KaiChiu Wong, and Chin-Teng Lin. Weak human preference supervision for deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 32(12):5369–5378, 2021.

[5] 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.

[6] Jessie YC Chen and Jennifer E Thropp. Review of low frame rate effects on human performance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 37(6):1063–1076, 2007.

[7] 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.

[8] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 267–274, 2002.

[9] R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2012.

[10] Xiaoteng Ma. Xtma/pytorch_car_caring: Reinforcement learning for gym carracing-v0 with pytorch, 2018.

[11] Eric Nalisnick. Machine learning under human guidance, 2023.

[12] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[13] Robin L Plackett. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202, 1975.

[14] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

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

[16] Yuhui Wang, Hao He, and Xiaoyang Tan. Truly proximal policy optimization. In *Uncertainty in Artificial Intelligence*, pages 113–122. PMLR, 2020.

[17] 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. *arXiv preprint arXiv:2307.04964*, 2023.

[18] Banghua Zhu, Jiantao Jiao, and Michael I Jordan. Principled reinforcement learning with human feedback from pairwise or k-wise comparisons. *arXiv preprint arXiv:2301.11270*, 2023.