How Does Corruption Affect Preference-Based Reinforcement Learning?

Andrew Freeman
University of Alberta
alfreem1@ualberta.ca

Minh Pham
University of Alberta
minh.pham@ualberta.ca

Euijin Baek
University of Alberta
euijin1@ualberta.ca

Abstract

Preference-based Reinforcement Learning aims to train an agent to learn an optimal policy based on human-provided preferences rather than relying on numeric reward signals. While this approach can help circumvent problems in the regular reward engineering process, human teachers can make mistakes due to inherent irrationality, which introduce inconsistency and inaccuracies to the preference data. We investigate how these errors in preference data affect policy performance across discrete and continuous action environments. Contrary to initial expectations, certain tasks show surprising robustness, even when the data has a high error rate. We explore why these results occur and highlight the interplay between our underlying algorithm and certain environment characteristics. Our contributions involve carefully interpreting results, providing understandable explanations for surprising outcomes, and aligning future research questions accordingly. Code is available here.

1 Introduction and Motivation

Designing effective reward functions is a longstanding problem in Reinforcement Learning (RL), especially for complex real-world tasks, such as autonomous driving and robotic manipulation [1, 2]. Many real-world environments lack clearly defined rewards, and even when they exist, they are often complex and highly specific to the environment [2]. Thus, rewards must be designed. However, poorly designed or suboptimal reward signals can result in unintended behaviors, and hinder an agent's ability to learn effectively, if it can learn at all [1, 3]. Designing rewards by hand is challenging, as it requires expertise in both the task domain and in RL systems [2].

Preference-based reinforcement learning (PbRL) offers a potential solution by using human knowledge to guide learning in tasks that are inherently human-centric [3, 4]. Instead of relying on a manually specified reward function that takes into account all possible state-action transitions, PbRL allows agents to optimize their policy based on pairwise comparisons of trajectory segments. For example, for a set of trajectories, having a human teacher indicate which trajectories are 'better' than others. This feedback can then be used by an agent to guide its future behavior toward trajectories more aligned with the human's preferences [3, 4]. This method has been successfully applied to tasks like fine-tuning large, pretrained models like GPT-4, but can also be used to train reinforcement learning agents from scratch [5, 6, 3, 7].

However, the use of PbRL comes with its own challenges. Human feedback is inherently inconsistent and prone to errors [4, 8, 9]. For example, humans may make mistakes, misunderstand task goals, misinterpret information, change their mind, or change their opinions based on their mood or circumstances. When applied to preferences generated by humans, this inconsistency is known as **preference error** or **preference corruption**, and introduces uncertainty into an agent's learning process [10]. Continuous action environments pose an additional challenge. These environments often involve large observation and action spaces, where individual preferences provide less relative coverage of the space than in discrete environments. Preference error in a continuous action space

could therefore mislead learning across a larger region of the environment compared to discrete spaces, where actions and observations are more limited and well-defined. To better understand these challenges, we focus on the following research question:

How does preference corruption influence the performance of PbRL methods in discrete versus continuous action environments?

It is from this foundation that we hypothesized the following:

1. Preference-based learning will generally perform worse as the level of corruption increases, demonstrating a lack of robustness.
2. Preference-based learning will not be as robust to increasing levels of corruption in continuous compared to discrete action environments.

By addressing this research question, we aim to investigate the robustness of PbRL algorithms under varying levels of corruption and identify factors that contribute to the robustness of PbRL methods.

2 Algorithm Explanation

Our approach builds on a PbRL method where an agent learns a reward model $\hat{r}$ that encodes pairwise preferences over trajectory segments [3]. Rather than requiring an explicitly designed reward function, we rely on pairwise comparisons between trajectory segments to infer a suitable reward signal for policy optimization.

Synthetic Preference Generation and Corruption: To simulate errors in human preferences, we generate synthetic preferences using a known ground-truth reward function r. Given two trajectory segments σ^1 and σ^2 following π initialized with random policy, we label σ^1 as preferred ($\sigma^1 \succ \sigma^2$) if:

$$\sum_t r(s_t^1, a_t^1) > \sum_t r(s_t^2, a_t^2),$$

where (s_t, a_t) is a state-action pair. We corrupt each label by flipping with a probability of an error rate ε. Higher ε indicates a greater chance of corruption.

Reward Model Training: A neural network $\hat{r}$ predicts a scalar score for each state-action pair. For a pair of trajectory segments σ^1, σ^2, we estimate the probability that σ^1 is preferred:

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

We train $\hat{r}$ by minimizing the cross-entropy loss over a dataset $\mathcal{D}$ of corrupted trajectory comparisons $(\sigma^1, \sigma^2, \mu)$, where $\mu(1) = 1$ if σ^1 is preferred and $\mu(2) = 1 - \mu(1)$:

$$\text{loss}(\hat{r}) = -\sum_{(\sigma^1, \sigma^2, \mu) \in \mathcal{D}} \Big[\mu(1) \log \hat{P}[\sigma^1 \succ \sigma^2] \; + \; \mu(2) \log \hat{P}[\sigma^2 \succ \sigma^1]\Big].$$

Policy Optimization: After the reward model $\hat{r}$ is trained, we replace the environment's ground-truth rewards with $\hat{r}$. The policy π is then optimized using standard RL, treating $\hat{r}(s, a)$ as the surrogate reward. This updated policy is used to collect the trajectories in the next iteration for a better exploration of state and action space. Algorithm 1 describes the detailed process and Figure 2 in Appendix A illustrates the algorithm in detail.

3 Experiment

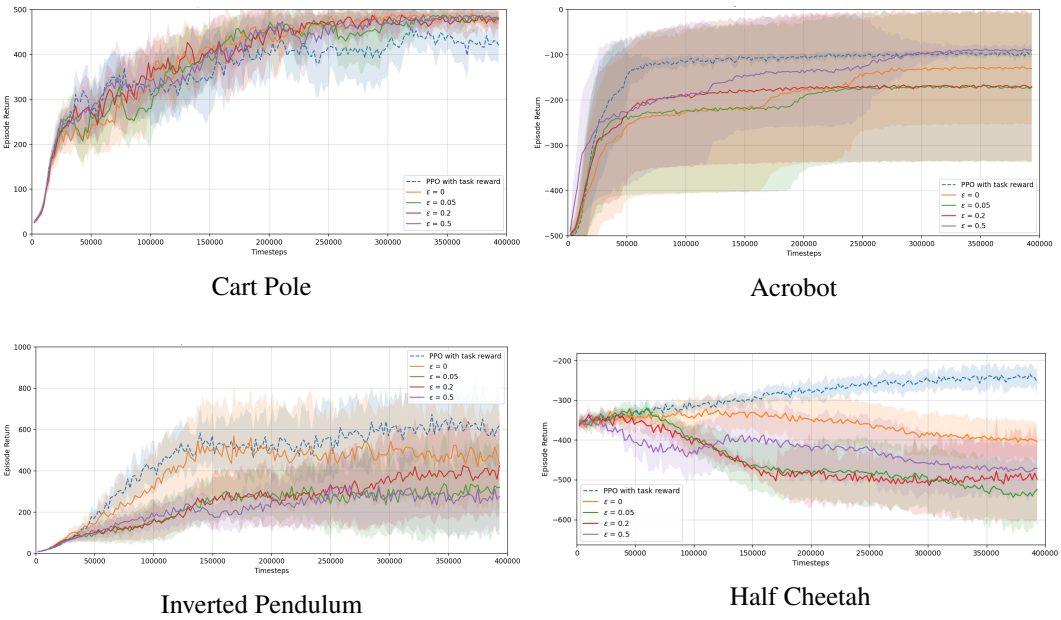
We evaluate the algorithm in two discrete (CartPole-v1, Acrobot-v1) and two continuous (InvertedPendulum-v4, HalfCheetah-v4) action environments. We chose these environments because the baseline (PPO on task reward) and the no-error case perform relatively well here. We generate synthetic preferences by collecting 100 pairs of trajectories in discrete tasks and 200 pairs

Algorithm 1 Preference-Based RL with Corruption

1: Initialize policy π with random policy and reward network $\hat{r}$.
2: **for** $d = 1$ to D **do**
3: Collect a set of pairs of trajectory segments $\{\sigma\}$ using π.
4: Compute ground-truth returns using r and derive preferences μ.
5: Introduce corruption by flipping each preference label with a probability of ε
6: Update $\hat{r}$ by minimizing cross-entropy loss with the corrupted μ.
7: Update π using PPO with $\hat{r}$ as the reward signal.
8: **Output:** Trained policy π and reward network $\hat{r}$.

in continuous tasks; this hyperparameter was tuned to optimize the reward model accuracy, and we believe it does not affect the robustness shown in the result. To assess the relationship between the agent's performance and the error rate ε in preference data, we run the experiment with $\varepsilon \in \{0\%, 5\%, 20\%, 50\%\}$, which represents none/little/medium/very high error rate. We then train a reward network on the resulting preference dataset. The reward network used for both types of environments has the same size, but the learning rate is tuned. A PPO agent then optimizes the policy on predicted rewards instead of the ground-truth rewards. We chose to use a PPO agent because it has been shown to have great performance in both discrete and continuous action environments [11]. Our agent is adapted from the CleanRL framework implementation [12], with modifications in the following hyperparameters after tuning: the entropy coefficient is bumped up to 0.02 (for both discrete and continuous action environments) to accommodate errors in the data, and the learning rate is decreased to 3e-5/1e-5 (for discrete/continuous action space) to stabilize learning against corrupted preference data. We ran 10 seeds per setting. We evaluated the agent at every PPO training iteration in an independent environment with ground-truth rewards. Results are reported as average episodic returns with the shaded regions indicating the standard deviation across seeds.

4 Results and Discussion



Cart Pole | Acrobot

Inverted Pendulum | Half Cheetah

Figure 1: PPO on preference data with different error rates in different environments

Results We show the large image versions of Figure 1 in Appendix B. In CartPole, the preference-based method has robust performance, with seemingly no decay in performance even at a high error rate. In Acrobot, while the method shows some signs of robustness, it has a higher variance. In both Inverted Pendulum and Half Cheetah, results show clear signs of a lack of robustness. An increase in

error rate doesn't always correspond to worse performance (such as in Acrobot, Inverted Pendulum, and Half Cheetah, where the agent performs worse with $\varepsilon = 0.05$ than with higher ε)—this refutes our hypothesis 1.

Discussion In discrete tasks, the policy remains robust even at the 50% data error rate. In more complex continuous tasks, corruption severely hinders PPO-based learning. Surprising results, such as success under a high error rate, stem from the inherent simplicities of certain environments.

In Cart Pole's true reward environment, the agent receives a +1 reward at each timestep the pole remains upright. Our method employs a sigmoid function to scale predicted rewards to the range of the ground-truth rewards (0 to 1). The results show that, regardless of data errors, the agent achieves high returns. The predicted rewards always being positive provide a reliable "stay alive" signal that aligns with the ground-truth reward, and the sigmoid scaling prevents individual data errors from significantly misleading the agent. In short, the agent shows strong performance and robustness due to the interaction between our underlying algorithm and the environment's inherent structure.

In Acrobot's true reward environment, the agent receives a -1 reward at each timestep the arm has not reached a bar. To scale the predicted rewards to the range of the ground-truth rewards (-1 to 0), our method now applies a negative sigmoid function to the predictions. The results show that, in most cases, the agent achieves high returns. However, certain random seeds (Figure 7 in Appendix C) result in the agent failing to learn, which lowers the average return and increases variance. Compared to the Cart Pole agent, which starts in a healthy state and aims to maintain it, the Acrobot agent begins in an undesired state and strives to reach a desired state. This means that the Acrobot agent may not get to sufficiently explore that desired state, which hinders learning. Overall, our method performs robustly in Acrobot, with exceptions arising due to the characteristics of the environment compared to Cart Pole.

The Inverted Pendulum task is more sensitive to precise control adjustments compared to the discrete action environments above. The agent must choose an amount of force, which comes from a continuous space, to apply to the cart to maintain balance. The results show that our method is sensitive to errors in the data. Half Cheetah is a significantly more complex continuous environment—achieving high returns requires fine-tuning a running gait and optimizing multiple joints and muscles. In this environment, our methods prove highly sensitive to data errors. A possible explanation is: in continuous action environments, the preference dataset has low coverage of the whole observation/action space, leading to high sensitivity to error in each individual piece of data. We show the evidence (relationship between reward model accuracy and agent performance) and explain our intuition for this hypothesis in Appendix D.

Overall, the results across the four environments show that our preference-based method has stronger robustness in discrete than in continuous action environments—which supports our hypothesis 2.

5 Future Work and Conclusion

This project lays the foundation for future research to investigate how environment complexity, independent of the action space, can influence space coverage of data and agent performance. Our initial results on the relationship between reward model accuracy and agent performance (Appendix D) highlight a key direction for further exploration. Additionally, our method can be extended to work with additional algorithms, such as SAC, in order to gather more experimental results—we present the motivation for this research direction and the initial results in Appendix E.

Our findings show that preference corruption is in fact not uniform between environments: task complexity and the underlying algorithm being used are important factors that affect performance. These results imply that using human preferences to train agents in simple discrete-action environments may show more robust behavior than in complex continuous-action spaces.

Acknowledgments

This report was prepared through discussions with Professor Michael Bowling and Simone Parisi.

References

[1] Tom Everitt and Marcus Hutter. Reward tampering problems and solutions in reinforcement learning: A causal influence diagram perspective. *CoRR*, abs/1908.04734, 2019.

[2] Jonas Eschmann. *Reward Function Design in Reinforcement Learning*, pages 25–33. Springer, 01 2021.

[3] Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023.

[4] Jie Cheng, Gang Xiong, Xingyuan Dai, Qinghai Miao, Yisheng Lv, and Fei-Yue Wang. Rime: Robust preference-based reinforcement learning with noisy preferences, 2024.

[5] OpenAI. Chatgpt: Optimizing language models for dialogue, 2022. Accessed: 2024-12-14.

[6] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[7] Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. Learning to summarize from human feedback. *CoRR*, abs/2009.01325, 2020.

[8] C. Wirth, R. Akrour, G. Neumann, and J. Fürnkranz. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18, 12 2017.

[9] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A survey of reinforcement learning from human feedback, 2024.

[10] Alexander Bukharin, Ilgee Hong, Haoming Jiang, Zichong Li, Qingru Zhang, Zixuan Zhang, and Tuo Zhao. Robust reinforcement learning from corrupted human feedback, 2024.

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

[12] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

Appendix / supplemental material

A Method

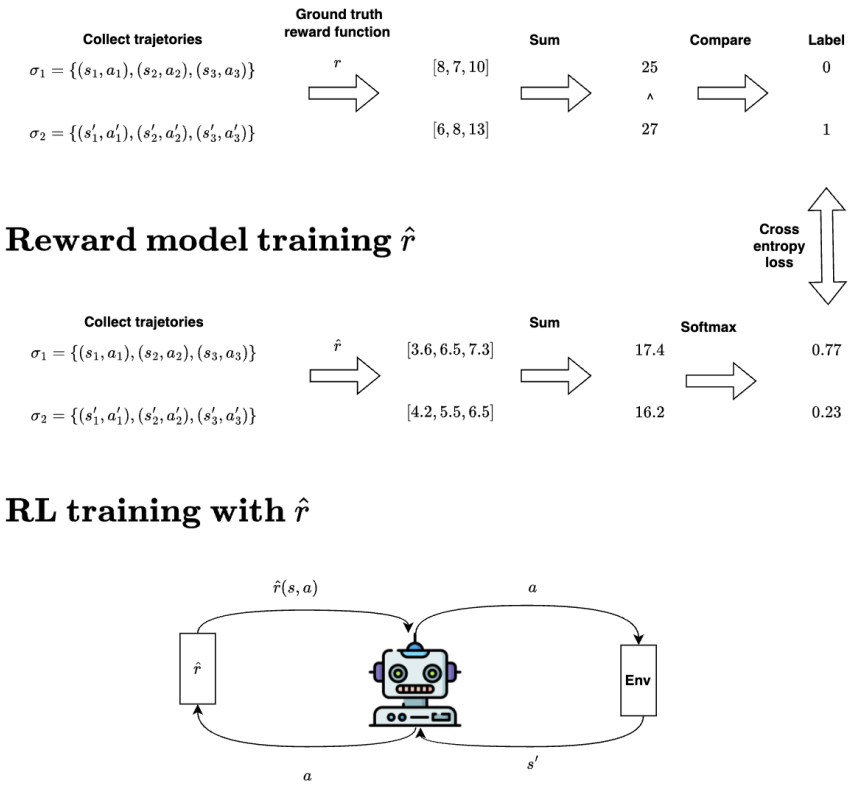


Figure 2: Illustration of the preference-based RL pipeline: (Top) Trajectory data is collected and assigned labels based on a known ground-truth reward function r. We compare cumulative returns of pairs of trajectory segments to create synthetic preferences, introducing corruption by flipping a certain fraction of labels. (Middle) The reward model $\hat{r}$ is trained via cross-entropy loss to predict the probability that one trajectory is preferred over another. (Bottom) Once $\hat{r}$ is learned, it is used as a surrogate reward signal to train a policy π using standard RL algorithms such as PPO.

B PPO in PbRL

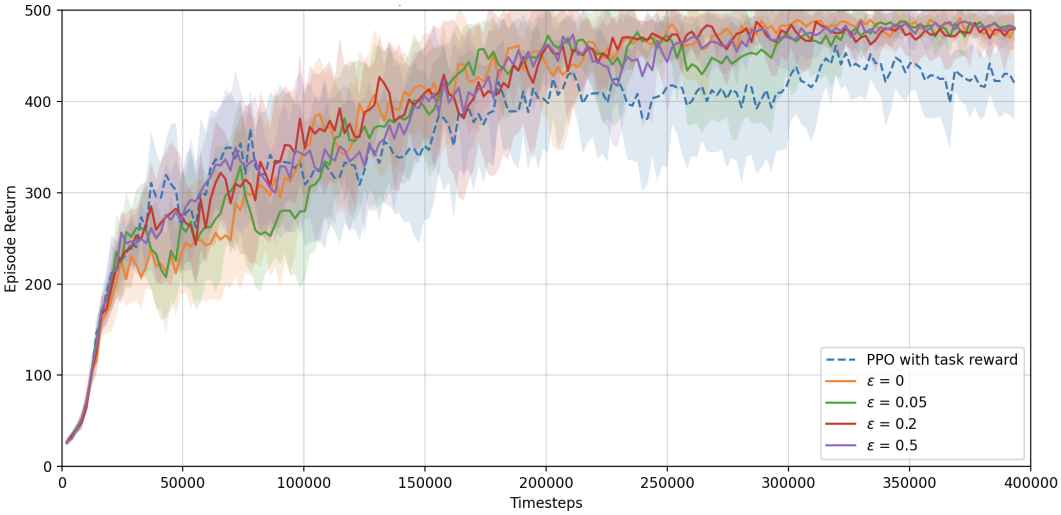


Figure 3: PPO on preference data with different error rates in Cart Pole

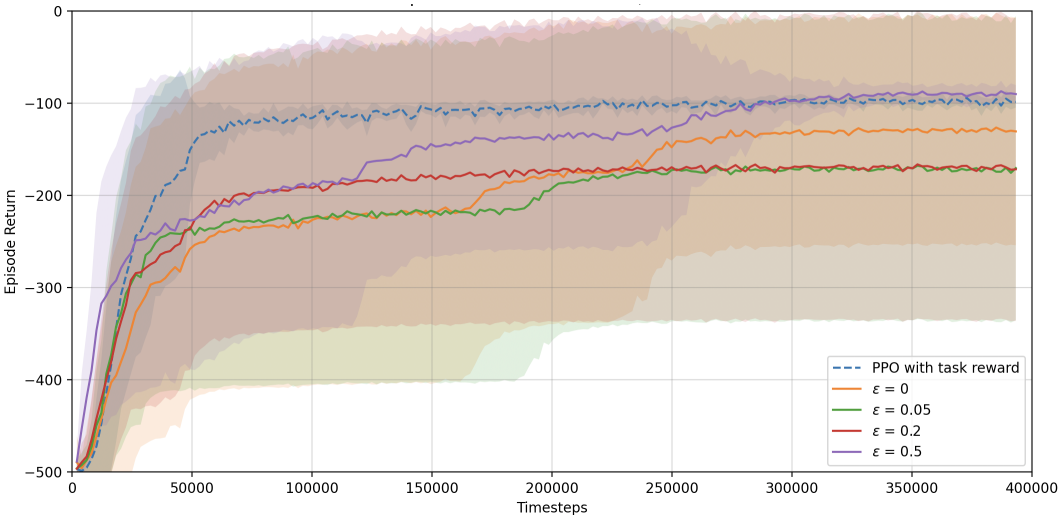


Figure 4: PPO on preference data with different error rates in Acrobot

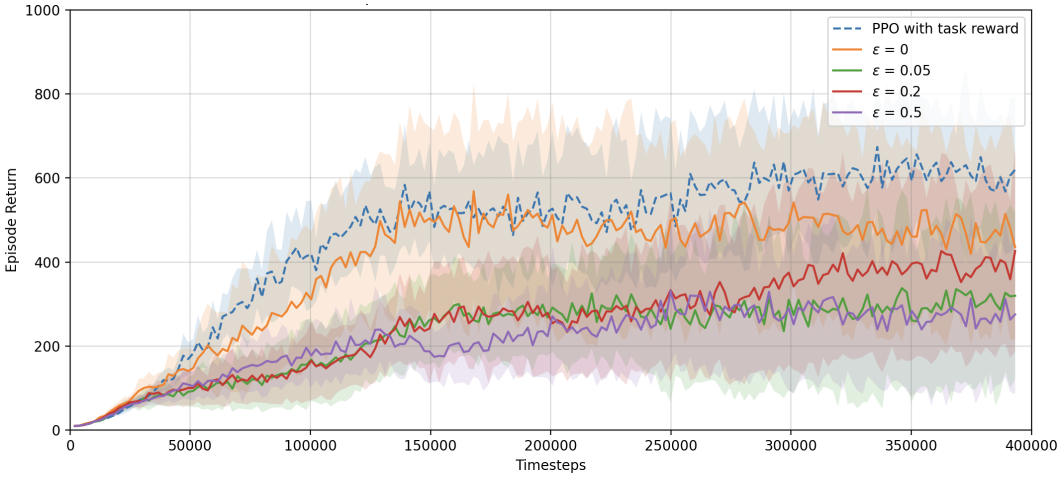


Figure 5: PPO on preference data with different error rates in Inverted Pendulum

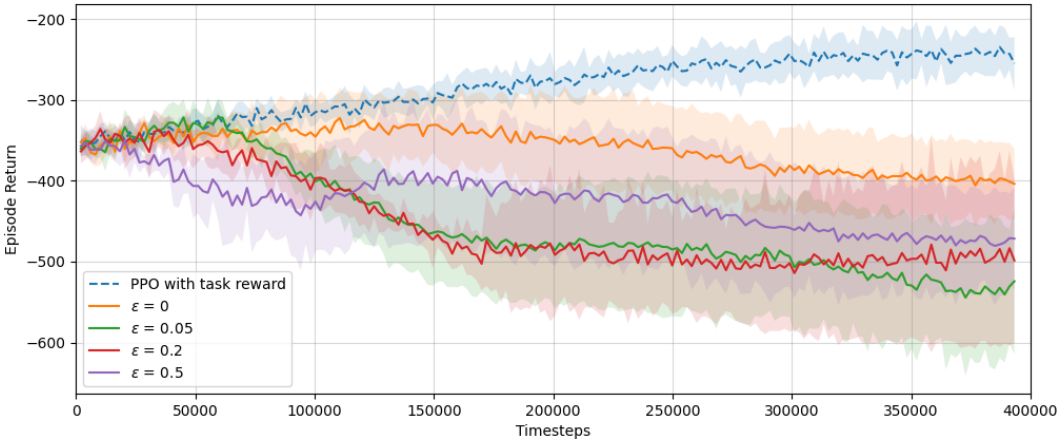


Figure 6: PPO on preference data with different error rates in Half Cheetah

C Acrobot Seed Failure

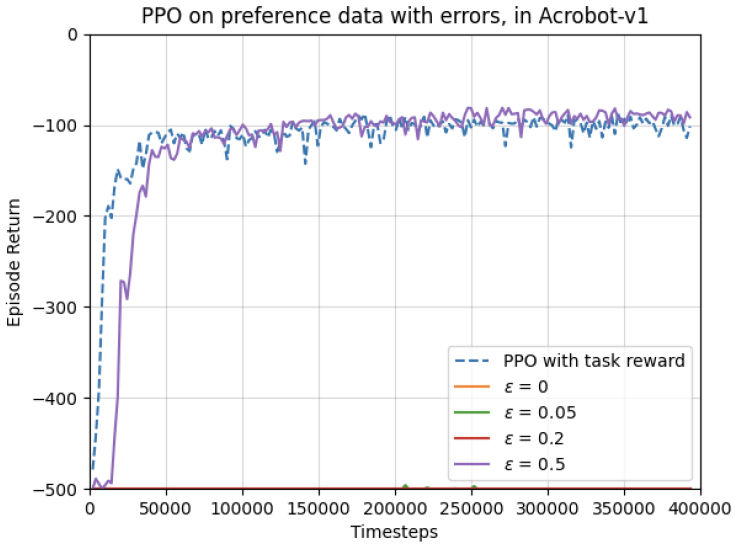


Figure 7: Seed where Acrobot agent fails to learn at all

D Reward Model Accuracy

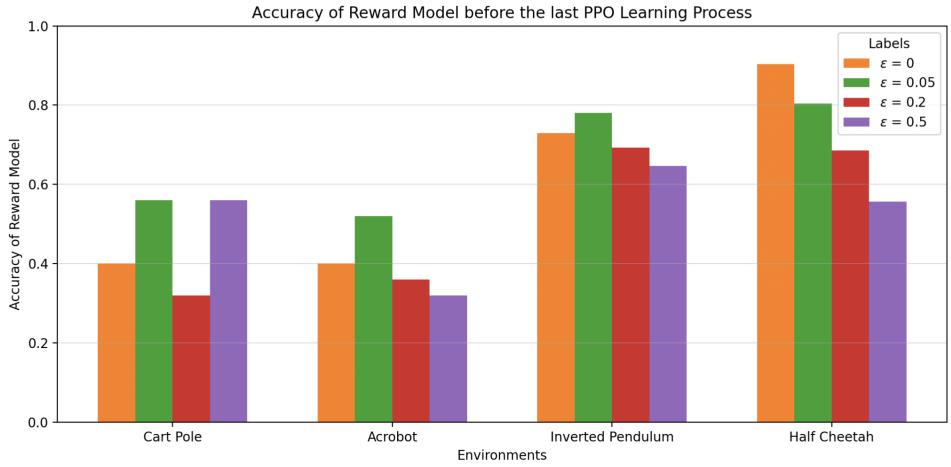


Figure 8: The accuracy of the reward model in the last data collection iteration across different environments and error rates

We compare the reward model's inference accuracy in discrete and continuous environments (Figure 8). The chart suggests that in discrete-action tasks, the observation and action space are small, so the data can well cover it. Thus, the model can realize there are conflicting data in the preference dataset, which is reflected in the low accuracy. Because the accuracy is low even with corruption, the model tends to underfit to the corrupted reward, reducing the impact of data corruption. On the other hand, in continuous-action tasks, the collected data has low coverage of the large observation and action space. Thus, the model can fail to realize there are conflicting data, hence the high accuracy. Consequently, the model tends to overfit to the corrupted rewards, which significantly affects performance.

E Soft Actor Critic in PbRL

While there are merits in this experiment for using the same PPO-based agent as the reference point across the two types of environments, it is worth noting that the baseline performance of the employed algorithm (PPO) in continuous-action environments is suboptimal (which could also be due to hyperparameters that we didn't tune), which may have led to the lack of robustness shown in the results. This raises an important question that can extend our current research: How do different algorithms perform across different environments? To explore this, we provide some initial results where we substitute PPO with SAC. Another way to explore this as a potential future work is to control the hyperparameters for PPO.

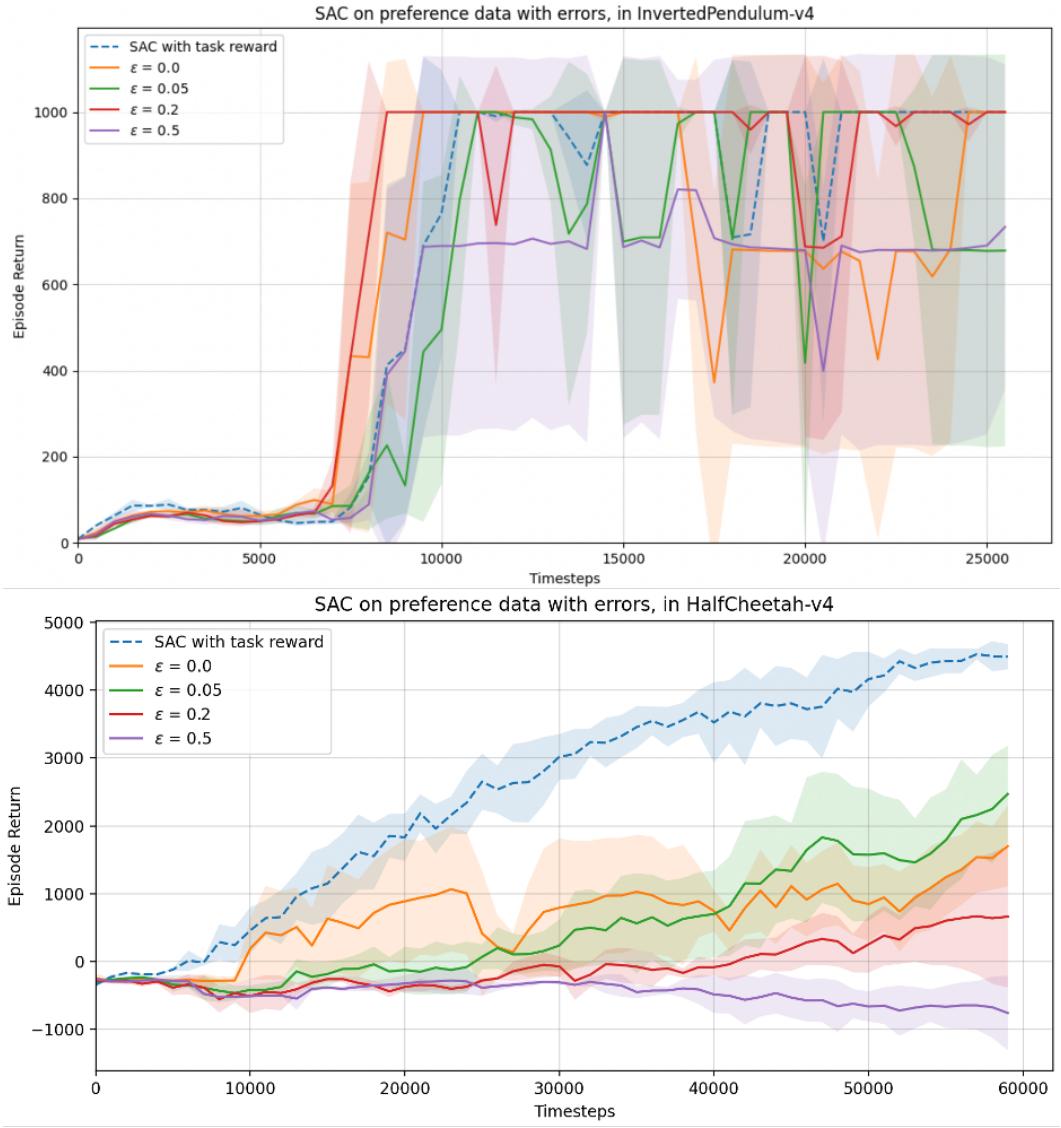


Figure 9: SAC on preference data with different error rates in Inverted Pendulum and Half Cheetah