On The Fragility of Learned Reward Functions

Lev McKinney[*]
University of Toronto
lev.mckinney@mail.utoronto.ca

Yawen Duan[*]
University of Cambridge
yd338@cam.ac.uk

David Krueger
University of Cambridge
david.scott.krueger@gmail.com

Adam Gleave
University of California, Berkeley
gleave@berkeley.edu

Abstract

Reward functions are notoriously difficult to specify, especially for tasks with complex goals. Reward learning approaches attempt to infer reward functions from human feedback and preferences. Prior works on reward learning have mainly focused on the performance of policies trained alongside the reward function. This practice, however, may fail to detect learned rewards that are not capable of training new policies from scratch and thus do not capture the intended behavior. Our work focuses on demonstrating and studying the causes of these *relearning* failures in the domain of preference-based reward learning. We demonstrate with experiments in tabular and continuous control environments that the severity of relearning failures can be sensitive to changes in reward model design and the trajectory dataset composition. Based on our findings, we emphasize the need for more retraining-based evaluations in the literature.

1 Introduction

Reward functions for most real-world tasks are difficult or impossible to specify procedurally. Specifically, hand-designed reward functions frequently misspecify the task [19]. The field of reward learning attempts to overcome this challenge by designing algorithms to infer reward functions from data. These learned reward functions aim to succinctly represent the desired behaviors [23], drastically reduce the amount of human feedback required to learn a task [8] and allow practitioners to generalize these behaviors to new environments [10].

One of the most promising approaches is to learn reward functions from binary human preferences over trajectory segments where these segments are collected online using a *sampler agent* trained to optimize the learned reward [8]. This form of preference-based reward learning is already being used to train large language models to summarize [43] and become more helpful and harmless [3].

Prior work has typically focused on the performance of the sampler agent [20, 8]. Unfortunately, the sampler agent performing the correct behavior does not guarantee that a robust reward function has been uncovered. In particular, when using reinforcement learning to train a randomly initialized *relearner agent* on the learned reward, the reward may fail to induce the correct behavior despite the sampler agent behaving well [15]. If we only require a policy that works passably well in the exact training environment, this may not be an issue because we can use the sampler agent and throw away the learned reward. We argue, however, that such a method cannot be accurately described as learning a *reward* function. At most, it is a preference-based policy learning technique, using reward functions to give a helpful inductive bias during training. Moreover, it is desirable for many applications to

[1] Equal contribution. Work done during internship at Center for Human-Compatible AI, UC Berkeley.

ML Safety Workshop, 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

truly uncover a reward function. For example, we might wish to train a new policy using a learned reward function with a more powerful R.L. algorithm or different agent architecture than was used during the initial reward learning process.

Past work has preformed preliminary investigations into the robustness of learned rewards in toy environments [29], in Atari [15] and for fine tuning language models [3, 33]. A more complete survey is included in Appendix A. However, these investigations are typically only reported short sections of their respective papers.

Inspired by this work, our paper empirically examines the relearner performance for learned reward functions. Since we have access to a ground truth reward in our synthetic experiments, we define poor relearner performance as achieving relatively low ground truth returns. Our results show that relearing can produce very different policies than the sampler, frequently achieving low ground-truth returns. Thus, we argue that current preference-based reward learning methods may produce reward functions that are not reliable as signals for policy relearning.

Our paper makes three key contributions: We demonstrate that state-of-the-art reward learning algorithms can produce reward models that fail to train new agents from scratch in tabular and continuous control settings; we show that the severity can increase as the trajectory dataset concentrates on high reward regions; finally, as an example of how how these relearning failures can be sensitive to changes in reward model design, we demonstrate that reward ensembles can effect relearning failures.

2 Background

Deep RL from Human Preferences We follow the framework of learning a preference model $\hat{r}_\phi$ from trajectory segment comparisons. Our method is the closest to deep reinforcement learning from human preferences [8]. It consists of four phases iterated: trajectory collection, preference elicitation, reward inference and policy optimization. During **trajectory collection**, the current policy, initially a random policy, samples rollouts from the environment collecting trajectory segments $\sigma_i = (s_0, a_0, s_1, a_1, \cdots, s_n)$ without reward labels and stores them in B. In phase two, the algorithm, **elicits preferences** $y \in \{\succ, \prec, \equiv\}$ for randomly selected pairs of segments $(\sigma_1, \sigma_2) \in \mathcal{B}$ from a labeler — human or synthetic [1]. The preferences are then stored in a preference dataset $\mathcal{D}$. The algorithm assumes these preferences have been been sampled from the Bradly-Terry model [4],

$$P(\sigma_1 \succ \sigma_2) = \frac{\exp\left(\sum_{s,a,s' \in \sigma_1} r(s,a,s')\right)}{\exp\left(\sum_{s,a,s' \in \sigma_1} r(s,a,s')\right) + \exp\left(\sum_{s,a,s' \in \sigma_2} r(s,a,s')\right)}. \tag{1}$$

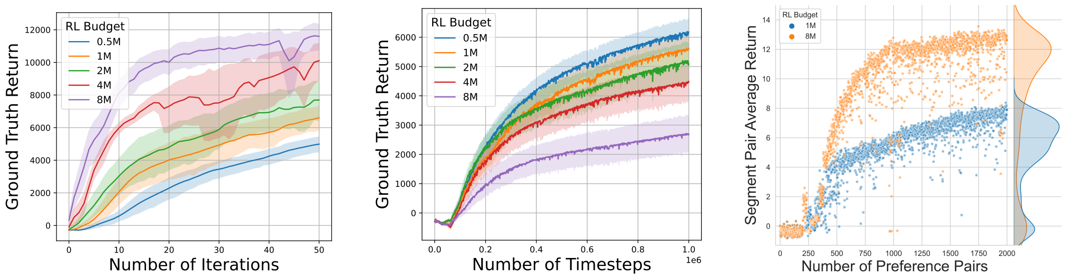
a widely used approximate model for human data in the preference based reward learning literature [8, 15, 20]. In the third phase **reward inference**, the reward $\hat{r}_\phi$ is fit by using Adam [17] to minimize the negative log likelihood of $\hat{r}_\phi$ under $\mathcal{D}$. The fourth and final phase of each iteration consists of **policy optimization**. In this stage, we can apply existing deep reinforcement learning algorithms to improve our policies expected return under the learned reward, and the process repeats.

3 Training and Evaluation Procedure

Reward Learning We train reward models with **synthetic data** that is sampled from the Bradley-Terry model of Eq. 1 with r set to the ground truth reward. In the tabular setting, we train the sampler policy using soft Q-Learning [13] and the learned reward networks simply take the current state as a one-hot vector for input. In the continuous control setting, we use soft actor-critic (SAC) [14] from Stable-Baselines3 [27] and the learned reward networks receive the observation, action and next observation as input. See Appendix B for further details.

Reward Ensembles Christiano et al. and Lee et al. use reward ensembles to estimate the uncertainty of the learned reward [8, 20]. We explore how these ensembles may have an another benefit, reducing the variance of off-distribution transitions. As in prior works, we train each ensemble member on bootstrapped datasets, normalize their outputs separately and use their mean as the reward.

[1] We follow Ibarz et al. in selecting preference pairs to query uniformly at random [15]



(a) *Reward learning* curves (b) *Relearning* curves (c) Preference datasets

Figure 1: Anti-correlated sampeler and relearner ground truth returns in `HalfCheetah`. (a) x-axis represents the number of iterations of each run. See section 2 RL budget is the total number of RL timesteps available to the sampler. (b) x-axis represents the number of timesteps during *relearning*. In plots (a-b), for each RL budget setting, we performed ten runs of reward learning, and for each of these, we ran five relearning evaluations for a total of 50 relearning runs. Solid lines and shaded lines represent the mean and 90% confidence respectively. (c) Scatterplot of average ground truth reward of each segment pair in the example preference datasets with 1M and 8M RL budgets.

Relearning We freeze the learned reward and train a new, randomly initialized, *relearner* policy to evaluate our reward functions. We evaluate this policy under the ground truth reward. This is similar to the method employed by Ibarz et al. [15, Section 3.2] to study reward hacking. In the continuous control settings this consist of training a new agent from the learned reward function using the same R.L. algorithm as the sampler, then evaluating it under the ground truth reward. In the tabular setting, we simply solve for the soft-optimal policy ($\alpha = 0.1$) [13] under the learned reward function.

4 Experiments

First, we investigate the occurrence of relearning failures in the continuous control domain. We use `HalfCheetah` environment as our test bed since it has been used in past works on preference-based reward learning [8]. Here we find that increasing the number of training timesteps the sampler takes between sampling trajectories for labeling increases the severity of relearning failures. Next, we focus on the effects of reward model design and observe that reward ensembles may reduce reward hacking in tabular environments by reducing the variance of off-distribution transitions. To demonstrate this failure mode we use the stay inside environment which consists of two rooms separated by a wall with a small doorway. The agent receives reward for staying in the inside room see Figure 2b.

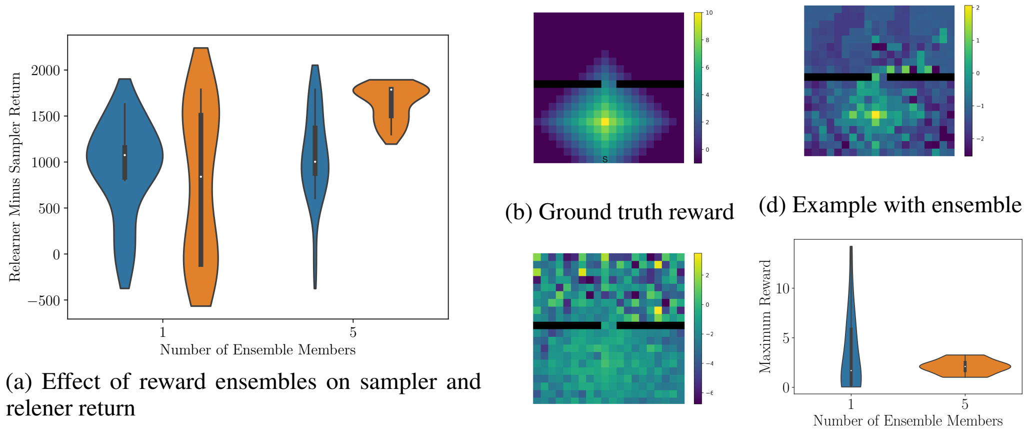
4.1 Preference Trajectory Dataset Imbalance and Relearning Failure

The reward model is a function of the dataset $\mathcal{D}$ used to train it. One of the simplest ways to change the preference dataset is to vary the number of timesteps T spent training the sampler between collecting trajectory fragments. We call the total number of interactions the sampler has with the environment during reward learning the *RL budget*. Note the RL budget does **not** affect the number of comparisons collected.

Figure 1 shows the learning curves of the sampler and relearner experiments in `HalfCheetah`. We find that despite higher RL budget leading to higher sampler returns during reward learning, the relearners' performance has the reverse trend; increasing sampler RL budget actually decreases relearner ground truth return.

We can gain some insight into why this is happening by exploring preference datasets shown in Figure 2c. First let's consider the preference dateset produced by one of the runs with the highest-budget (8M timesteps). We find that the trajectory segments contained in this datset are concentrated in high ground truth reward regions. On the other hand, when we consider the low-budget dataset (1M timesteps), the distribution of trajectory segments provides a better coverage across all the ground truth reward scales within the support. We hypothesize that having an overwhelming proportion of high-reward trajectory segments in the preference dataset — and little preference data on trajectories in the transition from high to low reward — may cause the reward model to effectively over-fit to the high-reward region. This overfitting leads to poor supervision over randomly initialized policies. Overall, we believe this could explain the observed relearning failures.

It's important to note that we did not see a significant increase in relearning failure when increasing the RL budget in the tabular setting. See Appendix E.



(a) Effect of reward ensembles on sampler and relener return

(b) Ground truth reward

(d) Example with ensemble

(c) Example no ensemble

(e) Max reward

Figure 2: Ensembles eliminate relearning failures in the stay inside environment. (b) depicts the ground truth reward in the stay inside environment. (c) shows an example individual learned reward and (d) with a five member ensemble. Finally, (e) shows the distribution of max learned-reward across all states. All sub-figures come from the same run which included 20 seeds.

4.2 Reward Ensembles

Our tabular experiments provide a concrete, interpretable example of how relearning failures can be effected by reward model implementation details. In particular, we focus on reward ensembles and observe that they have drastic effects on relearners but leave the sampler's performance unchanged.

In the stay inside environment, when using a reward ensemble of size five, all relearners preform at least as well as their respective samplers, as can be seen in Figure 2a. However, if we use only a single reward, the relearners behaviour is inconsistent; some relearners do substantially better then their respective samplers, but almost as many do substantially worse, getting near zero return. Thus, while adding an ensemble has a minimal effect on the sampler, it changes the behaviour of the relearners.

To understand why this happens we must consider the off-distribution behaviour of our reward models. In the stay inside environment, the samplers typically stay in the inside half of the environment. Thus there is often insufficient coverage of the outside half of the environment in our trajectory dataset. Thus, the reward off-distribution is largely unconstrained by the data. This means that small changes in the off-distribution behavior of our reward network can become critically important. Reward models based on neural networks produce spurious high rewards off distribution, see Figure 2. When these *reward delusions* are more rewarding than any of the in-distribution transitions, reward hacking can occur and cause relearning failures. Reward ensembles tends to have lower variance off distribution than an individual reward network. Thus, any reward delusions tend to have a lower reward (according to the reward model). This can be directly seen in Figure 2 (c-e). This effect reduces the chance that the optimal policy will be attracted to one of these spurious rewards during relearning, which is what we see in Figure 2a.

5 Limitations and Discussion

Our experiments have a few important limitations. First, they are limited to simple ground truth reward functions and environments. For example, in `Half-Cheetah-v3` [5, 11], the reward function is essentially a linear in the observation, action and next observation. While these relearning failures also appear in more complex tasks [15, Section 3.2], it is unclear if it is precisely the same phenomenon that causes them. The design decisions that seem to improve retraining performance in small-scale experiments, in our case, reward ensembles and less sampler training, may not be the same as those that address the problem at a larger scale. We leave such explorations to future work.

Overall, we have demonstrated that evaluations of relearning performance can differ substantially from the results of simply evaluating the sampler agent trained alongside the reward model. We hope to see future works include relearning evaluation as they appear to hold fruitful insights into the quality of the learned reward functions.

Author Contributions

Lev McKinney designed and implemented the tabular experiments and wrote the relevant parts of the method and experiments sections. In addition, he wrote the introduction, discussion and related works sections of the paper/appendix. Yawen Duan designed and ran the initial experiments that displayed reward model relearning failure on continuous control environments, and wrote relevant sections of the paper. David Krueger provided ideas, guidance and general feedback on experiment design and analysis. Adam Gleave provided initial ideas of the project, provided high-level and detailed feedback on experiments and analysis.

Acknowledgments and Disclosure of Funding

This paper was completed as part of an internship at the Center for Human-Compatible Artificial Intelligence. Funding for this internships was provided by the Berkeley Existential Risk Initiative.

References

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In C. E. Brodley, editor, *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004. doi: 10.1145/1015330.1015430. URL `https://doi.org/10.1145/1015330.1015430`.

[2] R. Akrour, M. Schoenauer, and M. Sebag. Preference-based policy learning. In D. Gunopulos, T. Hofmann, D. Malerba, and M. Vazirgiannis, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011. Proceedings, Part I*, volume 6911 of *Lecture Notes in Computer Science*, pages 12–27. Springer, 2011. doi: 10.1007/978-3-642-23780-5_11. URL `https://doi.org/10.1007/978-3-642-23780-5_11`.

[3] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, N. Joseph, S. Kadavath, J. Kernion, T. Conerly, S. El-Showk, N. Elhage, Z. Hatfield-Dodds, D. Hernandez, T. Hume, S. Johnston, S. Kravec, L. Lovitt, N. Nanda, C. Olsson, D. Amodei, T. Brown, J. Clark, S. McCandlish, C. Olah, B. Mann, and J. Kaplan. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback, Apr. 2022.

[4] R. A. Bradley and M. E. Terry. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika*, 39(3/4):324–345, 1952. ISSN 0006-3444. doi: 10.2307/2334029.

[5] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

[6] D. S. Brown, W. Goo, P. Nagarajan, and S. Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 783–792. PMLR, 2019. URL `http://proceedings.mlr.press/v97/brown19a.html`.

[7] S. Cabi, S. Gómez Colmenarejo, A. Novikov, K. Konyushova, S. Reed, R. Jeong, K. Zolna, Y. Aytar, D. Budden, M. Vecerik, O. Sushkov, D. Barker, J. Scholz, M. Denil, N. de Freitas, and Z. Wang. Scaling data-driven robotics with reward sketching and batch reinforcement learning. In *Robotics: Science and Systems XVI*. Robotics: Science and Systems Foundation, July 2020. ISBN 978-0-9923747-6-1. doi: 10.15607/RSS.2020.XVI.076.

[8] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural*

Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 4299–4307, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html.

[9] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In M. Balcan and K. Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 49–58. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/finn16.html.

[10] J. Fu, K. Luo, and S. Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *ICLR*, 2018.

[11] A. Gleave, P. Freire, S. Wang, and S. Toyer. seals: Suite of environments for algorithms that learn specifications. https://github.com/HumanCompatibleAI/seals, 2020.

[12] A. Gleave, M. D. Dennis, S. Legg, S. Russell, and J. Leike. Quantifying differences in reward functions. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=LwEQnp6CYev.

[13] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1352–1361. PMLR, 2017. URL http://proceedings.mlr.press/v70/haarnoja17a.html.

[14] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft actor-critic algorithms and applications. Technical report, 2018.

[15] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei. Reward learning from human preferences and demonstrations in Atari. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[16] H. J. Jeon, S. Milli, and A. D. Dragan. Reward-rational (implicit) choice: A unifying formalism for reward learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/2f10c1578a0706e06b6d7db6f0b4a6af-Abstract.html.

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[18] R. Kiryo, G. Niu, M. C. du Plessis, and M. Sugiyama. Positive-unlabeled learning with non-negative risk estimator. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1675–1685, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/7cce53cf90577442771720a370c3c723-Abstract.html.

[19] V. Krakovna, J. Uesato, V. Mikulik, M. Rahtz, T. Everitt, R. Kumar, Z. Kenton, J. Leike, and S. Legg. Specification gaming: the flip side of ai ingenuity, Apr 2020. URL https://deepmindsafetyresearch.medium.com/specification-gaming-the-flip-side-of-ai-ingenuity-c85bdb0deeb4.

[20] K. Lee, L. M. Smith, and P. Abbeel. PEBBLE: feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6152–6163. PMLR, 2021. URL http://proceedings.mlr.press/v139/lee21i.html.

[21] K. Lee, L. M. Smith, A. D. Dragan, and P. Abbeel. B-pref: Benchmarking preference-based reinforcement learning. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021.

URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/d82c8d1619ad8176d665453cfb2e55f0-Abstract-round1.html.

[22] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, X. Jiang, K. Cobbe, T. Eloundou, G. Krueger, K. Button, M. Knight, B. Chess, and J. Schulman. WebGPT: Browser-assisted question-answering with human feedback, June 2022.

[23] A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 663–670. Morgan Kaufmann, 2000.

[24] T. Ni, H. Sikchi, Y. Wang, T. Gupta, L. Lee, and B. Eysenbach. F-IRL: Inverse Reinforcement Learning via State Marginal Matching, Dec. 2020.

[25] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback, Mar. 2022.

[26] A. Pan, K. Bhatia, and J. Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=JYtwGwIL7ye.

[27] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22 (268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

[28] D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In M. M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2586–2591, 2007. URL http://ijcai.org/Proceedings/07/Papers/416.pdf.

[29] S. Reddy, A. D. Dragan, S. Levine, S. Legg, and J. Leike. Learning human objectives by evaluating hypothetical behavior, 2020. URL http://proceedings.mlr.press/v119/reddy20a.html.

[30] D. Sadigh, A. D. Dragan, S. Sastry, and S. A. Seshia. Active preference-based learning of reward functions. In N. M. Amato, S. S. Srinivasa, N. Ayanian, and S. Kuindersma, editors, *Robotics: Science and Systems XIII, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 12-16, 2017*, 2017. doi: 10.15607/RSS.2017.XIII.053. URL http://www.roboticsproceedings.org/rss13/p53.html.

[31] J. Skalse, M. Farrugia-Roberts, S. Russell, A. Abate, and A. Gleave. Invariance in policy optimisation and partial identifiability in reward learning, 2022.

[32] J. Skalse, N. H. R. Howe, D. Krasheninnikov, and D. Krueger. Defining and characterizing reward hacking. In *NeurIPS*, 2022.

[33] N. Stiennon, L. Ouyang, J. Wu, D. M. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. Christiano. Learning to summarize from human feedback, Feb. 2022.

[34] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109. URL https://doi.org/10.1109/IROS.2012.6386109.

[35] R. Wang, C. Ciliberto, P. V. Amadori, and Y. Demiris. Random expert distillation: Imitation learning via expert policy support estimation. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6536–6544. PMLR, 2019. URL http://proceedings.mlr.press/v97/wang19d.html.

[36] S. Wang, S. Toyer, A. Gleave, and S. Emmons. The `imitation` library for imitation learning and inverse reinforcement learning. `https://github.com/HumanCompatibleAI/imitation`, 2020.

[37] A. Wilson, A. Fern, and P. Tadepalli. A Bayesian approach for policy learning from trajectory preference queries. In *NIPS*, 2012.

[38] D. Xu and M. Denil. Positive-unlabeled reward learning. In J. Kober, F. Ramos, and C. J. Tomlin, editors, *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155 of *Proceedings of Machine Learning Research*, pages 205–219. PMLR, 2020. URL `https://proceedings.mlr.press/v155/xu21c.html`.

[39] S. Zhuang and D. Hadfield-Menell. Consequences of misaligned AI. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/b607ba543ad05417b8507ee86c54fcb7-Abstract.html`.

[40] B. D. Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, CMU, 2010.

[41] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In D. Fox and C. P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 1433–1438. AAAI Press, 2008. URL `http://www.aaai.org/Library/AAAI/2008/aaai08-227.php`.

[42] B. D. Ziebart, J. A. Bagnell, and A. K. Dey. Modeling interaction via the principle of maximum causal entropy. In J. Fürnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 1255–1262. Omnipress, 2010. URL `https://icml.cc/Conferences/2010/papers/28.pdf`.

[43] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. F. Christiano, and G. Irving. Fine-tuning language models from human preferences. *CoRR*, abs/1909.08593, 2019. URL `http://arxiv.org/abs/1909.08593`.

Appendices

A Related Work

Preference-based reward learning Our primary focus is on methods that learn from *preference comparisons* between two trajectories [2, 37, 30, 8]. Preference comparison is one of the most scalable reward learning methods, successfully applied to fine-tune large transformer language models [25, 33, 22, 3] to enhance their performance at certain tasks. Note that trajectory comparison methods contain more information about the reward than demonstrations, so they tend to produce better results when available [31]. However, note that these methods may still fare poorly when the human preference feedback does not match their model of human rationality [21].

Other reward learning approaches Many other methods have been developed to learn reward functions from human data [16]. One of the most popular is *Inverse reinforcement learning* (IRL) [23] methods that infer a reward function from demonstrations [1, 28, 41, 40, 42, 9, 10]. T-REX [6] is a hybrid approach, learning from a *ranked* set of demonstrations. An alternative approach learns from "sketches" of cumulative reward over an episodeCabi et al. [7].

Reward hacking Pan et al. provides the first systematic empirical study of *reward hacking*: RL agents exploiting misspecified reward functions [26]. Notably, they find that increasing agent capabilities, such as by increasing the RL policy's model size, can sometimes lead to *worse* performance on the ground truth reward, while performance on the misspecified *proxy* reward increases. In contrast to our work, Pan et al. only study reward hacking in hand-designed rewards designed to illustrate the phenomenon, whereas we investigate this phenomenon in learned rewards.

Reward hacking has also been studied from a theoretical perspective. Under the framework of general principle agent problems Zhuang and Hadfield-Menell examines the case where the agent's utility function can only account for a limited subspace of the set of attributes that make the true utility [39]. The authors proceed to show that, within their model, an optimal state under this proxy utility can have arbitrarily low ground truth utility, assuming the attributes that make up the reward exhibit a condition analogous to decreasing marginal utility and increasing opportunity cost. Skalse et al. instead propose a formal definition of reward hacking [32]. Concretely, they define a proxy reward as being *hackable* relative to a true reward if there exists a policy π' favored over policy π in the proxy reward, but where π' is actively worse than π under the true reward.Note that this definition is extremely demanding—indeed, if the policy is unconstrained, then a proxy reward is unhackable only if it induces an identical policy ordering to the ground truth. These works provide incite into why we might expect reward hacking to arise and provide an ideal to strive towards, unhackable rewards. In our paper, however, we focus on more concrete cases of relearning failures and practically attainable measures, such as relearner performance being close to sampler performance.

The most closely related work is by Ibarz et al., which evaluates their learned reward functions by freezing them and training a new policy, analogous to our *relearner* evaluations [15, Section 3.2]. However, this study was only a small, half-page section of their paper, and they did not examine factors that may increase or decrease the chances/severity of relearning failures.

Another important related work is by Reddy et al., who observes that rewards can fail to generalize due to a lack of informative trajectories in their training data [29]. They attempt to ameliorate this by querying humans on diverse hypothetical trajectories generated from a model. However, their method requires a world model and primarily focuses on taking advantage of this model to improve reward quality. In addition, they assume the user provides feedback through quantitative reward labels, whereas we focus on the more realistic and widely used preference comparison setting.

Finally, past work has found that language models trained on a preference-based reward model can learn to exploit their reward model [33, Section 4.3]. In a similar vein, Bai et al. found that the ability of their reward model to correctly predict human preferences over a pair of inputs degraded as those inputs where perceived as more rewarding by the model [3, Section 4.2]. However, none of these works have offered much analysis of what leads to reward hacking or in general relearning failures, beyond training against the learned reward for to long [33, Section 4.3] or a lack of data from off distribution [29].

Robustness of learned rewards from demonstrations There has also been work to improve the robustness of rewards learned from expert demonstrations i.e. inverse reinforcement learning (IRL). Xu and Denil [38] propose addressing the reward generalization problem via *positive-unlabeled* learning [18], with human feedback (e.g. demonstrations) being "positive" data and rollouts from the agent being unlabeled data. They demonstrate this approach improves robustness in a semi-supervised RL and imitation learning setting. In addition, past IRL methods have learned a reward with no sampler removing the problem of relearning failures entirely. For example Wang et al. [35] borrows methods from random network distillation to directly estimate the expert distribution with only expert data. In addition, multiple works have explored relearning and transfer in this domain. Fu et al. [10] propose an IRL method to learn state-only reward functions disentangled from transition dynamics via a state-only reward and preform some experiments on retraining new environments. Ni et al. [10] derive an analytic gradient estimator for an arbitrary f-divergence between expert and on policy distributions with respect to the reward functions parameters. They preform relearning evaluations. In these evaluations, their method appears to be able to recover the expert policy well. In contrast to these IRL methods, our work focuses on the more scalable preference-based reward learning setting.

B Training Details and Hyperparameters

B.1 Reinforcement learning algorithms

In the tabular setting we train the sampler policy using soft Q-Learning [13]. We use soft actor-critic (SAC) [14] implementations of Stable-Baselines3 [27] in the locomotion control tasks.

Both algorithms are off-policy and use a replay buffer, which ensures their high sample efficiency compared to on-policy RL algorithms. Note that the learned reward function $\hat{r}_\phi$ changes during training, so we relabel the transitions in the replay buffer after each iteration, similar to PEBBLE [20]. The main difference between our algorithm and PEBBLE is that we omit the unsupervised pre-training stage used in PEBBLE. We used the implementations from Imitation Learning Baseline Implementations [36] to perform the experiments.

B.2 Continuous Control Experiments

In the tabular setting, all reward networks only take the current state as a one-hot vector. They consist of a multi-layer perceptron with two hidden layers of size 256 and ReLU activations, similar to those used in PEBBLE [20].

Training details For reward learning experiments, we used the implementations of Preference Comparisons Algorithm from Imitation Learning Baseline Implementations [36] with a full list of hyperparameters in Table 1. For the RL component, we used soft actor-critic (SAC) [14] implementations from Stable-Baselines3 [27] in the locomotion control tasks with a list of hyperparameters in Table 2. For retraining evaluations, we use the same hyperparameters for SAC to train new agents against the frozen learned reward models.

Reward model The reward model consists of a single multi-layer perceptrons with two hidden layers of size 256 and LeakyReLU activations with slope 0.01. The input of the model consists of the state, action and next state vectors, and the input vector is normalized by running normalization. The output the the reward model is normalized by by exponential moving average. During relearning experiments, we directly use the raw reward output from the reward network while being normalzed by a VecNormalize layer in Stable-Baselines3 (`https://stable-baselines3.readthedocs.io/en/master/guide/vec_envs.html#vecenv`).

Reward normalization We compute a normalized version of the learned reward using an Exponential Moving Average to normalize the reward to mean zero and unit standard deviation. This normalized reward was then used for policy optimization. Note that normalizing the reward does not change the optimal policy, which is invariant to positive affine transformations. However, it does simplify the optimization problem. In particular, a normalized reward is a more stable objective for the critic to learn over time. Additionally, RL hyperparameters can depend on the reward scale (for example, learning rate should be set inversely proportional to reward scale) – normalizing the learned reward therefore allows us to use a consistent set of hyperparameters.

Hyperparameter	Value
Segment Length	50
Total Comparisons	2000
Number of Iteration	50
Reward Training Epochs	5
Query Schedule	constant

Table 1: Reward learning hyperparameters for continuous control experiments

Hyperparameter	Value
Learning Rate	0.0003
Batch Size	256
Discount	0.99
Learning Starts from	10000

Table 2: SAC hyperparameters for continuous control experiments

B.3 Tabular Experiments

Similarly to the continuous control experiments we use Imitation's implementation of preference comparison [36]. However, we use a tabular soft-q learning algorithm with a replay buffer [13] with reward relabling [20] to solve the environments. The reward network again uses a similar MLP architecture to the continuous control setting with a sightly smaller hidden size of 32. Finally, we normalize the reward functions before ensembling them using a simple running norm over sampled transitions which is frozen during retraining. Hyperparamaters can be found in Table 3.

Hyper Parameter	Value
Sampler Soft-Q Learning	
discount	0.99
learning rate	5e-2
replay buffer capacity	∞
temperature	0.1
samples from buffer per env sample	10
initial soft-q value	200
Reward Learning	
trajectory fragment length	30
total comparison budget	2,500
RL budget	500,000
frac. of comparisons from inital random trajs	0.1
select fragments for comparison	randomly
epochs of training per iteration	1
number of iterations	100
query schedule	constant
reward learning rate	1e-3
Reward Network	
reward network hidden layers	[32, 32]
activation function	ReLu
output normilization	Running Norm

Table 3: Tabular Experiment Hyperparamerers

Tabular Relearning When relearning we solve for the soft-optimal policy under the learned reward function with temperature 0.1 and discount factor 0.99.

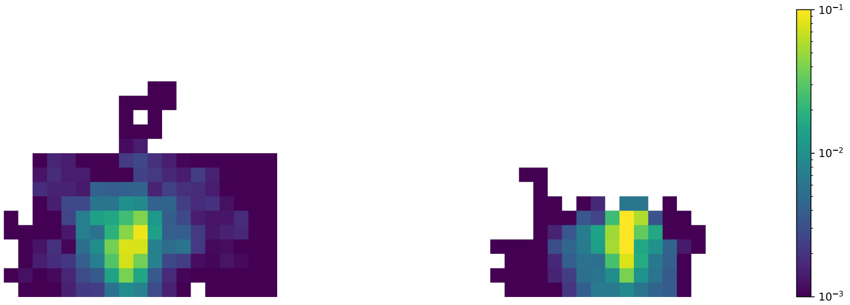
C Environments

Locomotion Control Task We ran reward learning and relearning on a MuJoCo locomotion task [34] – `HalfCheetah` environment from the *seals* benchmark suite [11], a modification of `HalfCheetah-v3` in the *gym* environment suite which adds the x-coordinate of the robot's center of mass (COM) to the first dimension of the observation space. The ground-truth reward function of the `HalfCheetah` environment is a linear combination of the x-velocity of the robot's COM and a control cost dependent on the L_2 norm of the action vector. Consequently, the reward function in *seals* `HalfCheetah` is a function of the observations, which is not strictly true in the original *gym* [5] environment, avoiding a potential confounder.

Tabular Environment We constructed the **stay inside environment**, which consists of a 20x20 closed grid of cells. The top "outside" and bottom "inside" halves of the environment are separated by a wall with a narrow two cell gap in the middle. The reward for each state is shown in Figure 2 (a), with reward values ranging from +10 to -1.

D Epic Distance as an Evaluation Metric

As an additional evaluation criterion, we consider using EPIC distance [12] to measure the distance between learned reward functions and the ground truth reward. EPIC works by canonicalizing the rewards to be invariant to potential shaping, normalizing them to be invariant to scale, and then computing the L^2 norm of the difference of those functions over a *coverage* distribution of transitions. Here we consider two coverage distributions: uniform and expert distribution. The



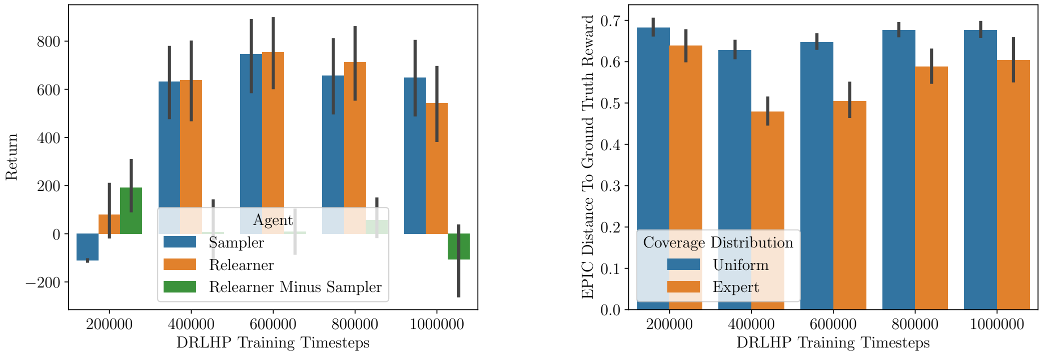
(a) Example without ensemble

(b) Example with ensemble

Figure 3: Example on policy distribution
Examples of the on policy distributions of the samplers in the stay inside environment, marginalized over the entire training run.

uniform distribution is uniform over feasible transitions. The expert distribution is the distribution of a soft-optimal policy with a temperature of 10 to give slightly more coverage.

E Additional Tabular Experiments



(a)

(b)

Figure 4: Increasing the number of time steps of R.L. training does not seem to significantly effect relearning failures.

To study the effects of training the sampler for a more time steps, we first consider a simple environment consisting of a 10x10 grid world. The agent begins in the lower left-hand corner of the environment and gains a ground-truth reward of 10 for reaching the lower right-hand cell, as seen in Figure 5.

The performance of the sampler and relearner initially increases with more training timesteps, with our relearners generalizing well and achieving slightly higher performance than their respective samplers. However, it quickly plateaus even though we *do not* see significant reductions in relearner performance with an increased number of time steps. The EPIC distances of our learned reward functions from the ground truth reward begin to increase after 400,000 timesteps Figure 4 (b).

Increasing the number of total training timesteps used for DRLHP does seem to degrade the quality of the reward function according to EPIC distance. However, it does not appear to hurt relearning performance in the same way in this simple tabular environment.

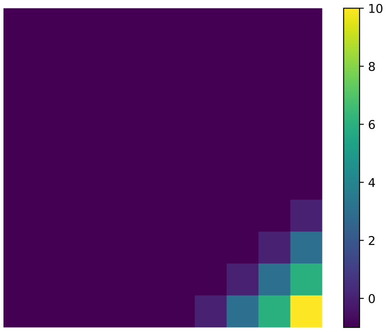


Figure 5: Tiny room environment. The ground-truth reward in the tiny room environment. Note that the reward only depends on the current state.

This is a strikingly different effect than we see in `HalfCheetah`. This may be because in a tabular setting the sampler either finds the optimal policy induced by the learned reward function every iteration, so the sampler and relearner have equal performance, or it insufficiently explores the environment, and reward learning completely fails. This dichotomy leaves little room for the subtle degradation in relearner performance we see in Figure 1.