A STUDY OF CAUSAL CONFUSION IN PREFERENCE-BASED REWARD LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning policies via preference-based reward learning is an increasingly popular method for customizing agent behavior, but has been shown anecdotally to be prone to spurious correlations and reward hacking behaviors. While much prior work focuses on causal confusion in reinforcement learning and behavioral cloning, we aim to study it in the context of reward learning. We demonstrate that preference-based reward learning often fails to produce causal rewards. In particular, we perform a series of sensitivity and ablation analyses on three benchmark domains where rewards learned from preferences achieve minimal test error but fail to generalize to out-of-distribution states—resulting in poor policy performance when optimized. We find that the presence of non-causal distractor features, noise in the stated preferences, and partial state observability can all exacerbate causal confusion. We also identify a set of methods with which to interpret causally confused learned rewards: we observe that optimizing causally confused rewards drives the policy off the reward's training distribution, resulting in high predicted (learned) rewards but low true rewards. These findings illuminate the susceptibility of reward learning to causal confusion, especially in high-dimensional environments—failure to consider even one of many factors (data coverage, state definition, etc.) can quickly result in unexpected, undesirable behavior.

1 INTRODUCTION

Preference-based reward learning (Wirth et al., 2017; Sadigh et al., 2017; Christiano et al., 2017; Brown et al., 2020b) is a well-studied technique for learning from pairwise preferences or rankings and holds the potential of enabling AI systems to learn specifications for tasks without requiring a human to write down an explicit reward function, and to adapt the AI system's behavior to individual preferences and needs. However, recent anecdotal evidence suggests that these methods are prone to learning rewards that pick up on spurious correlations in the data and miss the true underlying causal structure, especially when learning from limited numbers of preferences (Christiano et al., 2017; Ibarz et al., 2018; Javed et al., 2021). While the effects of reward misspecification have recently been studied in the context of reinforcement learning agents that optimize a proxy reward function (Pan et al., 2022), and the effects of causal confusion have been emphasized in behavioral cloning approaches that directly mimic an expert (De Haan et al., 2019; Zhang et al., 2020; Swamy et al., 2022), there has not been a systematic study of causal confusion when learning reward functions.

As an example of the type of causal confusion we study in this paper, consider the assistive feeding task in Fig. 2b. Note that all successful robot executions will move the spoon toward the mouth in the area in front of the patient's face—few, if any, executions demonstrate behavior behind the patient's head (the exact likelihood depending on how trajectories for preference queries are generated). In practice, we find that a preference-based reward learning approach will often pick up on the signed difference between the spoon and the mouth rather than the absolute value of the distance. The two correlate on the training data, but using the former instead of the latter leads to the robot thinking that moving behind the patient's head is even better than feeding the person!

In this work, we contribute the first systematic study of causal confusion as it occurs in preference-based reward learning. First, we demonstrate the failure of preference-based reward learning to

produce causal rewards that lead to desirable behavior on three benchmark robotic tasks, even when given large amounts of preference data—in these settings, the learned reward has high test accuracy but leads to poor policies when optimized. We then study the effect on causal confusion of several factors related to reward model specification and training: the presence of non-causal distractor features, reward model capacity, noise in the stated preferences, and partial state observability. For each of these, we perform analysis of what errors the learned reward has, how it compares to the ground truth reward, and the amount of distribution shift it induces during policy optimization. Some effects are predictable (e.g. removing distractors helps), some are perhaps surprising (e.g. increasing the amount of data when preferences are noisy can hurt when the data has diversity issues to begin with), and they are not always correlated to changes in the validation accuracy.

Another of our contributions is to point to the importance of data coverage and perhaps interactive techniques that iteratively or actively query for feedback to help learners disambiguate causal features from correlates. But they also caution that there are many other aspects that can go wrong or make the problem much more challenging, from the very way we choose to define state, i.e. what to include and what *not* to include in the input to the learned model, to how we manage the relationship between noisy data quantity and diversity.

1.1 Related Work

Reward Hacking and gaming behaviors are known to commonly occur in reinforcement learning (Ng et al., 1999; Krakovna et al., 2020) and reward learning (Christiano et al., 2017; Ibarz et al., 2018; He & Dragan, 2021). However, these behaviors are often only mentioned anecdotally (Krakovna et al., 2020). Recently, Pan et al. (2022) proposed to systematically analyze reward misspecification in RL by creating a set of domains where the agent optimizes a proxy reward function and studying when this leads to incorrect behavior. By contrast, we study cases where the reward function *must be learned* from human preferences.

Preference Learning is a popular method for training AI systems when a reward function is unavailable (Wirth et al., 2017; Christiano et al., 2017; Sadigh et al., 2017; Ibarz et al., 2018; Biyik & Sadigh, 2018; Brown et al., 2019; 2020a). Preferences are often easier to provide than demonstrations or raw numerical values (Sidney, 1957; Kendall, 1948; Wirth et al., 2017) since they do not require expert proficiency or fine-grained feedback. However, optimizing a reward function that has been trained using preferences can sometimes lead to unintended behaviors. Annecdotal evidence of this has been documented for Atari games (Christiano et al., 2017; Ibarz et al., 2018; Brown et al., 2020a), simple pointmass navigation tasks (Javed et al., 2021), and language model fine-tuning (Stiennon et al., 2020); however, there has been no systematic study of causal confusion when learning reward functions. In this paper, we provide the first concrete and systematic study of causal confusion across three robot control benchmarks when learning and optimizing reward functions from preference labels over trajectories.

Causal Confusion in Imitation Learning has previously been studied in the context of behavioral cloning (Pomerleau, 1988; Torabi et al., 2018), where a policy is trained to directly map from states to actions using demonstrated state-action pairs. Prior work has shown that behavioral cloning approaches can suffer causal confusion due to "causal misidentification," where giving imitation learning policies more information leads to worse performance (De Haan et al., 2019) due to temporally correlated noise in expert actions (Swamy et al., 2022). Similarly, we find strong evidence of causal misidentification when expert noise is present. Zhang et al. (2020) use causal diagrams to investigates causal confusion for simplified imitation learning tasks with discrete actions and small numbers of states when the features available to the demonstrator are different from those of the imitator. We also study what happens when the observation space of the imitator changes; however, we focus on imitation via preference learning, consider tasks with continuous states and actions, and evaluate causal confusion when using non-linear reward function approximation.

Goal Misgeneralization has been studied by Langosco et al. (2022) and Shah et al. (2022)—here, the RL setting has a known goal and "features of the environment are correlated and predictive of the reward on the training distribution but not OOD." This setting is similar to ours in that there is misidentification of the state features that are causal with respect to the reward (e.g. going to the end of a maze rather than collecting the coin that is the true reward). However, in contrast to our

work, during training the reward signal is always present and always the ground-truth reward. We show that when learning a reward function the learned reward can be misidentified, leading to poor RL performance. By contrast, Langosco et al. (2022) and Shah et al. (2022) show that even if the learned reward is perfect, the RL optimization can still fail due to spurious correlations during policy optimization.

1.2 Background: Reward Learning from Preferences

We model the environment as a finite horizon MDP (Puterman, 2014), with state space $\mathcal{S}$, action space $\mathcal{A}$, horizon T, and reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Note that we assume that the reward function is unobserved and must be learned. The reward function is learned from preferences over trajectories where, using the popular Bradley-Terry model (Bradley & Terry, 1952), the probability a trajectory τ_B is preferred over another trajectory τ_A is given by

$$P(\tau_A \prec \tau_B) = \frac{\exp(r(\tau_B))}{\exp(r(\tau_A)) + \exp(r(\tau_B))}, \tag{1}$$

where $r(\tau) = \sum_{(s,a)\in\tau} r(s,a)$ and where a trajectory τ is defined to be a sequence of state-action pairs: $\tau = (s_0, a_0, \ldots, s_T, a_T)$.

To learn a reward function from preferences, we assume access to a set of pair-wise preference labels $\mathcal{P}$ over trajectories $\tau_1, \ldots, \tau_N$, where $(i,j) \in \mathcal{P}$ implies that $\tau_i \prec \tau_j$. We then optimize a reward function $r_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ parameterized by θ that maximizes the likelihood:

$$\mathcal{L}(\theta) = \prod_{(i,j)\in\mathcal{P}} \frac{\exp(r_\theta(\tau_j))}{\exp(r_\theta(\tau_i)) + \exp(r_\theta(\tau_j))}. \tag{2}$$

We describe how we implement this in practice in Alg. 1.

2 Causal Reward Misidentification

Fig. 1 displays the causal structure of preference-based reward learning. In preference-based reward learning, pairwise preferences (in the form of a binary label) are given based on an underlying reward function $R(x)$—usually that of the human user or teacher (Fig. 1a). Within the state space, there are features x_t^i that are causal and influence $R(x)$, as well as other features z_t^i that are nuisance variables and have no bearing on $R(x)$. Note that z_t^i may very well exhibit correlations with x_t^i, potentially due to their sharing of the same causal parent or there being biases during data collection. Given the pairs of trajectories (specified by state information (x, z)) and binary preference labels, the goal of preference-based reward learning is to learn a reward function $\hat{R}(x, z)$ that best matches the stated preferences. In Figure 1b, there are state features u which are causal with respect to the true reward, but are unobserved by the learning agent. In both cases, unobserved human bias and noise, denote by η, also affects the preferences labels.

The learned reward $\hat{R}$ may be able to achieve low and even near-perfect performance on a held-out test set by making use of 1) nuisance variables that correlate with causal variables or 2) faulty/incomplete correlations between causal variables and R that happen to hold true for the training data distribution. Optimizing a misidentified reward function leads to distribution shift when performing reinforcement learning on the learned reward values $\hat{R}([x, z])$, resulting in behaviors with low performance under the true reward function R. We define this behavior of learning a reward that achieves low test error but results in poor performance (under the true reward function) when the learned reward function is optimized via RL as *causal reward misidentification*.

3 Experimental Setup

We now introduce the environments and analysis methods that we use in our paper.

Environments for Preference Learning. We identify a set of robot learning benchmarks that exhibit causal reward misidentification when learning reward functions from preferences. In ***Reacher*** (Brockman et al., 2016) (Fig. 2a), the goal is to move an end effector to a desired goal

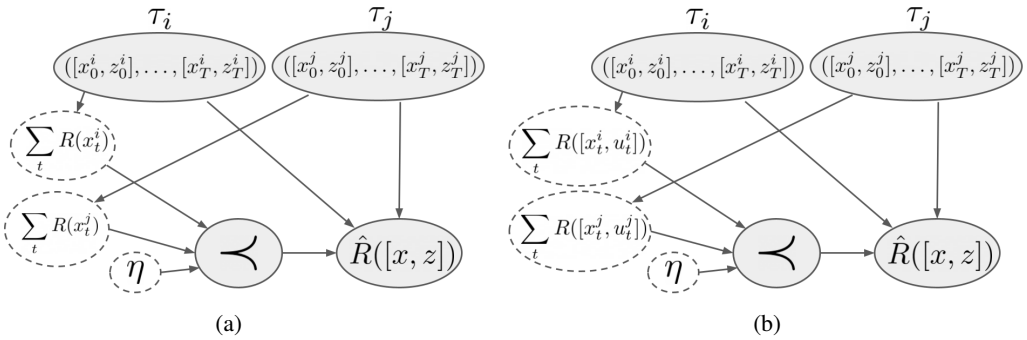


Figure 1: **Causal Structure of Preference-Based Reward Learning.** The reward function $\hat{R}$ is learned from pairwise preference labels over trajectory pairs (τ_i, τ_j). Unobserved variables are denoted by dashed lines. Unobservable user noise η affects the preference node. In (a) the true reward is not effected by nuisance variables z. In (b) the true reward is also based on unobserved state features u.

location. In ***Feeding*** (Erickson et al., 2020) (Fig. 2b), the goal is to feed the human using a spoon carrying pieces of food. Finally, in ***Itch Scratching*** (Erickson et al., 2020) (Fig. 2c), the goal is to repeatedly scratch a desired itch location on the human's arm.

True Rewards and Preference Data Generation. Each domain has a predefined "true" reward function r (outlined in Appx. A.5). This enables us to create synthetic demonstrations and preference labels by adding increasing amounts of noise to an expert policy trained on r (details in Appx. A.2). As shown by Brown et al. (2020b), adding this type of disturbance will result in monotonically decreasing performance in expectation and produce a diverse dataset for preference learning. Gleave et al. (2020) also uses a similar way of switching between an expert policy and a random policy, which they find produces the best coverage distribution (over a purely random or purely expert policy). We do compare this method of diverse trajectory generation against that proposed by Brown et al. (2019) and find no significant difference in performance; see Appx. A.11 for details.

Note that while we use the ground-truth reward function for obtaining preference labels, we assume no access to this reward function during policy learning, but rather seek to learn a policy that obeys a users preferences by first learning a model r_θ of the true reward function from preference labels $\mathcal{P}$, and then run RL on the learned reward function to produce a policy. We can then evaluate the learned policy on the true reward function r. To facilitate reproducibility and encourage future research on causal reward learning, we open-source our code and training datasets. This combination of domains and training data forms the first set of benchmarks for studying causal confusion when learning reward functions.

Evaluating Learned Reward Functions. In order to establish that a learned reward is causally confused, we first check for ***low test error*** and establish that the learned reward performs well on unseen in-distribution test data (thereby ruling out model selection and training failures like not having enough data, not using high enough capacity, not regularizing enough, etc.). We then show that the learned reward fails in two ways: 1) it leads to a policy that, when optimized, has ***poor performance with respect to the true reward*** – we call this policy "PREF", and 2) this poor performance is not due to RL failures, i.e. the learned reward actually does prefer its (poorly-performing) optimized policy, "PREF", over the optimal policy with respect to the true reward, which we call "GT": the learned reward ***prefers PREF over GT***. Finally, we analyze the learned reward qualitatively via gradient saliency maps and quantitatively via the Equivalent-Policy Invariant Comparison pseudometric (Gleave et al., 2020), then elucidate the effects of the reward error by quantifying the distribution shift induced by policy optimization of the learned reward.

Saliency maps are one of the few methods that allow one to interpret learned reward functions in an isolated, relatively lightweight manner. Following Michaud et al. (2020), we use raw gradient saliencies, or $\frac{\partial R}{\partial(s,a)}$—the gradient with respect to each element of the input. We extend upon this by examining gradient saliencies *per timestep* along with feature *spread maps*—maps of each input feature's variation (standard deviation, variance, range) over the course of a trajectory.

Table 1: **Empirical evidence of causal confusion.** We compare policies optimized with a reward learned from preferences (PREF) against policies optimized with the true reward (GT). State features on which preferences are based are fully-observable in all three tasks. S, M, and L correspond to training dataset sizes of 780, 7140, and 52326 unique pairwise preferences, respectively. Both PREF and GT are optimized with 1M RL iterations and averaged over 3 seeds. Despite high pairwise classification accuracy, the policy performance achieved by PREF under the true reward is very low compared with GT, irrespective of data size. However, *the reward learned from preferences consistently prefers PREF over GT.* This, combined with the low success rates of PREF compared to GT, suggests that preference-based reward learning fails to learn a good reward, even as the amount of data increases, for all of these tasks.

	PREF. LEARNING ACC.			RL POLICY PERFORMANCE		
DOMAIN	TRAIN	VAL	TEST	LEARNED (PREF/GT)	TRUE (PREF/GT)	SUCCESS (PREF/GT)
REACHER (S)	0.955	0.913	0.939	-1.097 / -6.002	-13.331 / -5.560	0.040 / 0.827
REACHER (M)	0.957	0.949	0.962	-12.002 / -14.936	-11.890 / -5.560	0.053 / 0.827
REACHER (L)	0.954	0.956	**0.966**	**44.988** / 3.395	-42.716 / **-5.560**	0.100 / **0.827**
FEEDING (S)	0.976	0.902	0.891	90.671 / 13.206	-153.012 / 128.933	0.057 / 0.990
FEEDING (M)	0.979	0.968	0.960	106.415 / 68.835	-45.427 / 128.933	0.437 / 0.990
FEEDING (L)	0.987	0.976	**0.976**	**277.152** / 124.016	-27.432 / **128.933**	0.603 / **0.990**
ITCHING (S)	0.974	0.908	0.869	18.757 / 10.337	-56.591 / 248.397	0.000 / 0.970
ITCHING (M)	0.967	0.924	0.918	17.871 / 12.685	-68.024 / 248.397	0.003 / 0.970
ITCHING (L)	0.954	0.933	**0.928**	**16.588** / 10.282	-47.190 / **248.397**	0.013 / **0.970**

EPIC, or Equivalent-Policy Invariant Comparison, is a pseudometric proposed by Gleave et al. (2020) that quantifies the difference between two reward functions on a given coverage distribution and proposes to be predictive of policy performance without the need for policy optimization. We apply the EPIC pseudometric to compare the distance between various learned rewards and the ground truth reward on the coverage distributions seen during reward learning and those seen during policy training.

***Kullback-Leibler divergence*:** We use a discriminator trained to minimize the cross-entropy loss on states from two different distributions following the approach proposed by Huszár (2017) and Laidlaw & Dragan (2021). Specifically, we approximate the KL divergence between the distribution of state-action pairs seen during reward learning and those seen during RL on the learned reward. We use this to measure the amount of distribution shift from the reward learning distribution induced by optimizing (potentially causally confused) learned rewards during policy training. Details on how this is done are in Appx. A.6.

4 EVIDENCE OF CAUSAL CONFUSION

Before varying different factors that affect the performance of the learned reward, we start with a generous setting where we provide large amounts of data to the learner in each environment, and we make sure to add features to the default input space for each environment such that all necessary information needed to infer the ground truth reward, TRUE, is available. We produce the preference training data as detailed in the previous section: again, this ensures that there are comparisons that involve optimal behavior relative to slightly suboptimal one, as well as highly suboptimal, in principle ensuring good variety. For hyperparameter tuning and other training details, see Appx. A.1. Table 1 details the results.

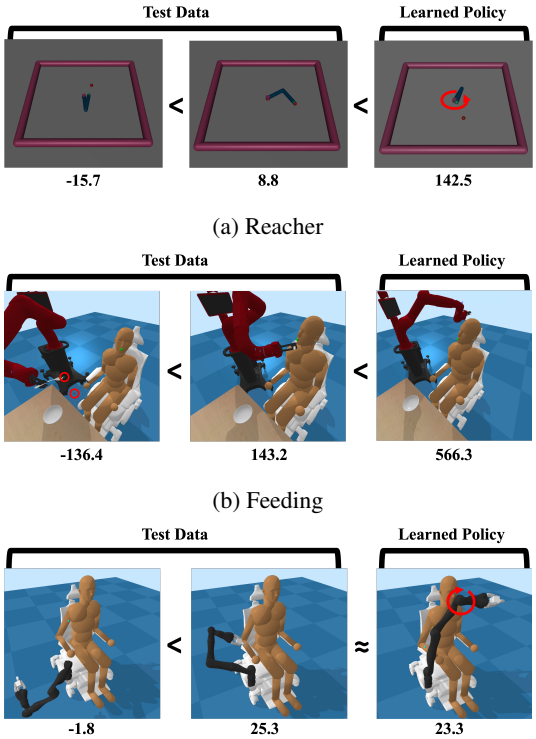
First, note that, especially with the largest data setting (L), the learned reward achieves high test accuracy, comparable to the training accuracy (columns 2-4 in the table). This is a sanity check that the model learned does not overfit, and that the model capacity is high enough and data plentiful enough to get decent performance. Note that in later sections, we observe that even models that reach over 99% test accuracy sometimes end up failing to produce good polices.

Second, when optimizing the learned reward to obtain the PREF policy, the optimization works in the sense that the learned reward does prefer PREF to the GT policy (obtained by optimizing the

true reward). This shows that the poor performance is not due to an issue in RL exploration. We see this in column 5 of the table.

Unfortunately, when looking at the actual performance of the PREF policy in columns 6 and 7, it is disastrous: not only does it have poor TRUE reward (compared to the GT policy), but it leads to poor success rates. Overall, the learned reward incentivizes poor behavior, despite reasonable test data accuracy.

In the Reacher environment, we observe that the policy optimized using the learned reward chooses to simply spin very fast rather than reaching for the target. We note, however, that the learned reward correctly classifies the leftmost image in Fig. 2a (where the agent just folds its arm) as being worse than the middle image (where the agent successfully reaches the target), and does so with 96.6% accuracy on such diverse pairs of comparisons. One would think that this would apply to the behavior in the rightmost image, where the agent folds its arm as well (and subsequently spins), but the learned reward turns out to strongly prefer the rightmost case. As will be explored in Sec. 5.1, this is due to the slight positive gradient for the angular velocity feature in the learned reward (Fig. 4a), which stems from a slight correlation between angular velocity and reward (Fig. 8a). Similarly, with the Feeding environment, the learned reward learns to minimize the *signed* difference between the spoon and the mouth rather than the *absolute value* of the difference because the majority of trajectories approach the mouth from in front (Fig. 8b). As a result, the learned reward (Fig. 9b) correctly identifies spilling food (left) as being worse than feeding (middle), but goes further and incentivizes bringing the spoon 'into' and even *behind* the head in an attempt to minimize the signed difference. In Itch Scratching, the learned reward correctly identifies flailing (left) as being worse than actual scratching (middle). However, as seen in Fig. 9c, the Itch Scratching agent (spuriously) learns a higher weight on two components of the action (corresponding to two robot arm joints), which results in the agent turning the last section of the arm in a circle while trying to keep the end effector close to the itch target—another type of flailing! It also falls into the same trap as the Feeding agent in minimizing the signed rather than the absolute difference (Fig. 8c). Fig. 2 summarizes this behavior. Appx. A.3 provides plots of the aforementioned features' spurious correlations with the true reward and gradient saliency maps of the learned rewards. Next, we analyze why these failures occur.



(a) Reacher

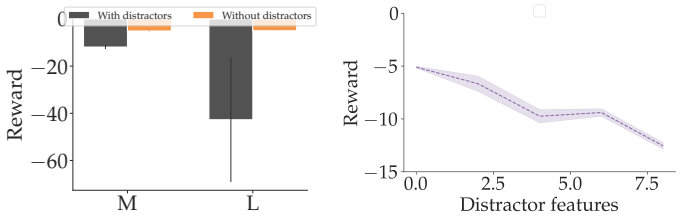
(b) Feeding

(c) Itch Scratching

Figure 2: **Poor behaviors resulting from learned rewards** (rightmost column), despite correct classification of bad vs. good behaviors on the test data (left two columns). The predicted trajectory reward (and resulting preferences) produced by the learned reward function is displayed under each image; each image corresponds to a trajectory.

5 Factors that Lead to Causal Confusion

In the following sections, we examine various factors of the preference-based reward learning problem setup and their effects on causal reward misidentification. The motivation for exploring each is as follows:



(a) RL Performance (b) Removing distractors

Dataset Size	Distractors With	Distractors Without
M	**53.731**	6.762
L	**46.032**	7.366

(c) KL Divergence

Figure 3: **Distractor Features.** Removing distractor features dramatically increases the performance of the policy learned through preference-learning. Fig. 3a compares performance with distractors present in the learned reward with performance without distractors present in the learned reward across two dataset sizes. Fig. 3b is a sensitivity analysis on the number of distractor features done on the M dataset size case from Fig. 3a. In Fig. 3c, we see that distractor features result in a much larger distribution shift (as measured by the KL divergence between state-action pair distributions at reward learning and RL).

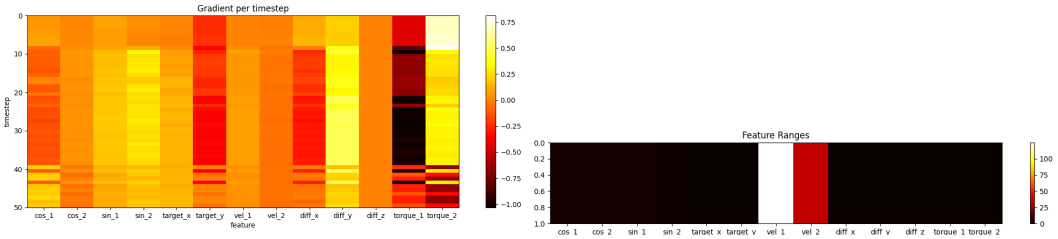
- **Distractor Features** draws directly from the findings in Sec. 4, where we find that certain non-causal features may be spuriously correlated with the reward.
- **Model Capacity** is inspired by Pan et al. (2022)'s findings on the effects of increased agent capabilities on reward hacking.
- **Noise in Stated Preferences** is inspired by the fact that preference data collected from real humans is often rife with different types of bias and noise.
- **Partial Observability of Causal Features** is also due to the fact that it may not always be possible to fully-observe all causal features in the real world.
- **Complex Causal Features** is a topic that may become increasingly relevant in higher-dimensional, complex robotic tasks—unlike toy environments, where certain non-causal features can be ignored entirely, features here may be causal in certain contexts and non-causal in others.

As we will see, these factors (which may often be overlooked) can contribute significantly to the causal confusion and eventual success or failure of the reward model. We find that causal confusion in the learned reward induces a distribution shift that can make the learned reward appear deceptively close to the true reward during reward learning but terribly misspecified during policy optimization.

5.1 Distractor Features

One of the main reasons for learning a wrong (causally confused) reward is the presence of too many features in the input that spuriously correlate with the causal features. To study this effect, we incrementally remove such 'distractor' features and test the impact this has on the learned reward. For example, for Reacher, we know that joint angles and angular velocities are not causal to the ground truth reward, which is based purely on distance between end effector and target and the norm of the action (a control penalty) (see Appx. A.4 for a complete list of causal and distractor features). As we see in Fig. 3, removing all distractors drastically improves performance, and, indeed, the more such features are left in the input, the worse the performance gets in Reacher, almost linearly. See Appx. A.7 for results in other tasks. Granted, removing distractors does also slightly improve the validation and test performance in this case—indicating that there is some signal in the training data to help discern the spuriousness of certain features. Nonetheless, over the several experiments in this paper, we find that the validation error is not strongly correlated to performance—in some cases, increasing the validation accuracy results in a large loss in both true reward and success rate.

Fig. 2a depicts the policy of the learned reward with distractors—the Reacher robot learns to simply fold its arm and spin. Looking at the gradient saliency maps (Fig. 4) illuminates why this is the case. Firstly, in Fig. 4a, we observe that the learned reward is misaligned with respect to 'diff_y', the feature corresponding to the difference between end effector position and target position along the y-axis; specifically, we note that the learned reward actually rewards an increase in 'diff_y', rather than



(a) Gradients - With distractors

(b) Feature Ranges - With distractors

Figure 4: **Saliency Maps: Distractor Features.** Fig 4a is a gradient saliency map of the (causally confused) learned reward. Fig. 4b is a heatmap displaying the ranges of each feature over the course of a trajectory produced by the learned reward. The large range of the 'vel_1' feature and its corresponding positive gradient are the reason why the trajectory does well under the learned reward.

a decrease. Next, we notice that the reward doesn't penalize action evenly. It penalizes 'torque_1' (the one responsible for the spinning), but rewards 'torque_2' across nearly all timesteps. It should instead be largely a negative penalty, as seen in Fig. 11a. Lastly and perhaps most importantly, we observe in Fig. 4b that 'vel_1' and 'vel_2' have very large feature ranges, corresponding to the large variations in angular velocity achieved by the Reacher robot's spinning behavior. Looking back to the gradient for each of these velocity features in Fig. 4a, we observe that the reward has a slight positive gradient with respect to each. By spinning fast, the Reacher robot is thus able to achieve much higher (learned) reward than performing the proper reaching action.

The KL divergence (Figure 3c) between the distribution of observation-action pairs seen during reward learning and those seen during policy optimization provides further insights—in incentivizing the Reacher robot to spin fast, it leads the RL optimization toward states that were not seen during reward learning (specifically, states where the robot is spinning very fast).

5.2 Model Capacity

In Appx. A.8, we study the effect of model capacity on the learned reward. We find that, despite careful tuning of hyperparameters with each model and dataset size, increasing the capacity of the reward model does not necessarily result in an increase in subsequent policy performance.

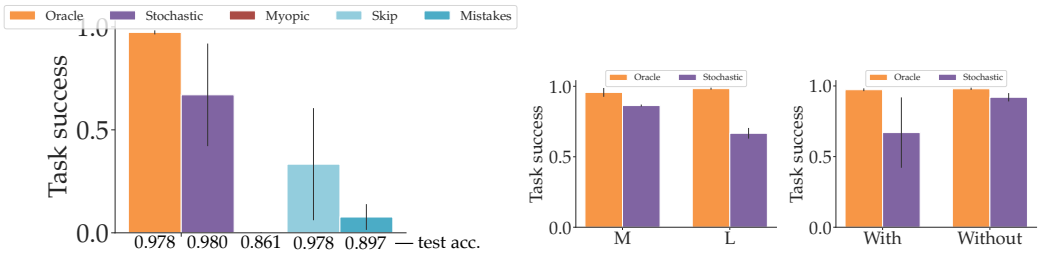
5.3 Noise in Stated Preferences

Because user noise is often inevitable, we explore the effect of various types of user noise on causal reward misidentification. Following Lee et al. (2021), we modify Eq. 1 to include a user rationality constant β and a myopic discount factor $\gamma \in (0, 1]$:

$$P(\tau_A \prec \tau_B) = \frac{\exp(\beta \sum_{t=1}^{H} \gamma^{H-t} r(s_t^B, a_t^B))}{\exp(\beta \sum_{t=1}^{H} \gamma^{H-t} r(s_t^A, a_t^A)) + \exp(\beta \sum_{t=1}^{H} \gamma^{H-t} r(s_t^B, a_t^B))}. \tag{3}$$

Using Eq. 3, we explore four types of user noise (varied independently of each other): STOCHASTIC, where the user is rational with $\beta = 1$; MYOPIC, where earlier rewards are discounted with a $\gamma = 0.99$; SKIP, where the teacher/preference-giver skips a pair of trajectories if both of have rewards below a certain threshold; and MISTAKES, where the preference label is randomly flipped with probability $\epsilon = 0.1$. ORACLE has $\beta = \infty$. Results are displayed in Fig. 5.

We find that noise in the stated preferences exacerbates causal reward misidentification—test accuracy stays high while policy performance plunges (Fig. 5a). Notably, two instances of user noise, Stochastic and Skip, have test accuracies greater than or equal to the Oracle (despite far worse performance). In addition, increasing the amount of preference data when noise is present actually has a negative effect, as seen in Fig. 5b. To elaborate: for a fixed β and a fixed range of trajectory qualities (say, from r_{min} to r_{max}), increasing the number of trajectories increases the number of *similar* trajectories (the 'density' of trajectories in reward-space), which in turn increases the likelihood of drawing a pair of similar trajectories and mislabeling. Further, there is a compounding effect with distractor features, where both noise and distractors result in a large loss in performance



(a) Feeding, User Noise (b) Reacher, Data Size (c) Feeding, Distractors

Figure 5: **Noise in Stated Preferences.** Noise in the stated preferences result in policies that perform worse than those trained from perfect preferences. Notably, however, increasing the amount of data *worsens* the performance. In addition, the presence of distractor features has a compounding effect on the error from noisy preferences.

Table 2: **EPIC Distances, Noise in Stated Preferences.** Distances from the ground truth reward are displayed. Causally confused rewards (Stochastic) are closer to the ground truth reward on the reward learning distribution (RewL) but further from it on the RL distribution (RL) when compared to the properly learned rewards (Oracle). EPIC distances are computed on the rewards learned with distractors in the Feeding environment in Fig. 5c and averaged over three seeds.

REWL / RL	STOCHASTIC	ORACLE
GROUND TRUTH	**0.136 / 0.164**	**0.165 / 0.130**

(Fig. 5c). Importantly, the poor performance of the learned policies is not well predicted by the validation accuracies, with models achieving 0.995 resulting in poorer reward and success rate than models trained on less data with lower validation accuracy of 0.985. Additional results are located in Appx. A.9.

To examine these results further, we compare the learned rewards directly with the ground truth reward (without performing policy optimization) with the EPIC pseudometric in Table 2. Interestingly, the EPIC distances between the learned rewards and the ground truth reward vary depending on the choice of coverage distribution. With the distribution of observation-action pairs seen during reward learning (generated by randomly switching between an expert and random policy), the reward trained with labeling errors appears to be closer (in EPIC distance) to the ground truth than the reward trained without labeling errors. However, these results are flipped when we instead evaluate the pseudometric on the distribution of observation-action pairs seen during policy optimization. This suggests that the causally confused reward model more closely 'mimics' the ground truth reward *on the reward learning distribution* (hence the lower EPIC distance) but fails to generalize to the reinforcement learning distribution.

Similarly, we again observe (see Tab. 8 in the Appendix) that the KL divergence between reward learning and reinforcement learning state distributions is greater when the learned reward is misspecified. This indicates that the learned rewards are causally confused: they have low test error but incentivize RL to deviate from the optimal behavior encountered during reward learning.

5.4 PARTIAL OBSERVABILITY OF CAUSAL FEATURES

Partial observability over aspects of the state on which the user's preferences are based is nearly inevitable in the real world. It is important to note that RL with the ground truth reward is able to learn proper feeding behavior *in spite of this lack of critical state information*. However, as displayed in Fig. 6, reward learning is only successful when the model is able to observe all the features that influence the true reward.

Examining the gradient saliency plots (Fig. 15) in the Appendix, we see that without having access to all the features that play into the true reward, the reward model is much more likely to become causally confused—it may over-weight the causal features that are available or pick up on correlations with non-causal variables that are totally spurious. With Feeding, we observe both cases: the

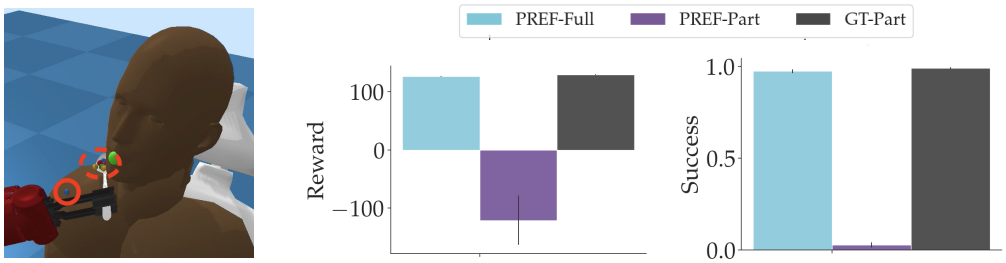


Figure 6: **Effects of partial state observability.** Minimizing spoon-mouth distance without observability on food particles leads to spilling food (left). Adding observation features to make the true reward fully observable results in high performance, whereas using the default environment features (which make the ground truth reward partially observable) results in low performance.

reward model learns stronger weights on one of the joint angles and two components of the action, all of which have no bearing on the true reward. Simultaneously, the reward also learns a greater weight on the second component of the feature corresponding to the vector distance between the end effector and the mouth, which is causal in the ground truth reward. (Note that a similar weighting also occurs in the learned reward that has access to all features of the true reward.) This results in the behavior depicted in Fig. 6—the robot manipulator is able to successfully maneuver the spoon close to the patient's mouth (by observing the distance feature), but does so without ensuring that the food particles themselves stay on the spoon and end up in the patient's mouth.

EPIC distances are displayed in Table 10 in the Appendix. Unlike with previous factors, EPIC distance on the reward learning distribution is predictive of eventual policy performance. The KL divergences (Table 3) again reveal that the causally confused reward (this time, due to lack of information in the observation) leads to a greater distribution shift during RL training.

5.5 Complex Causal Features

To explore how results may differ as we increase the complexity of the task, we evaluate causal reward misidentification on a more complex task of Itch Scratching, which is challenging as it requires a robot to (1) maneuver around the human body in Euclidean 3-space $\mathbb{E}^3$ to reach itch locations that are randomly generated along the 3D surface of the human arm, and (2) to perform a scratching motion, rather than staying still, at the itch target. We find that even after increasing the amount of training data for reward learning (Fig. 7a), increasing the reward model capacity (Fig. 7b), removing distractor features that are not causally related to the ground truth reward, having perfectly stated preferences, and ensuring full observability over all the features that are involved in the ground truth reward, performing preference-based reward learning still fails to produce a policy that successfully scratches the itch location on the patient. However, as shown in Table 1, a standard model-free RL algorithm trained using the ground truth reward is able to solve such a task.

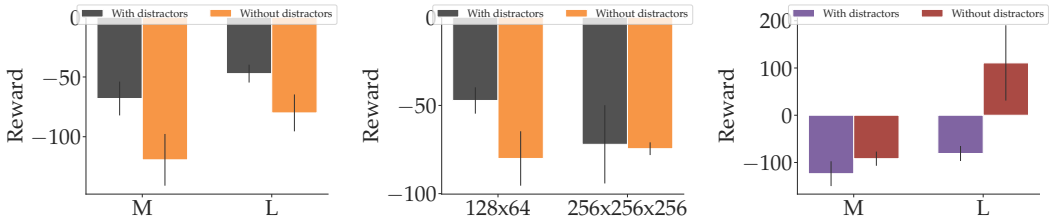
Table 3: **KL Divergence: Partial Observability.** The reward with partial observability on causal features induces a greater distribution shift than the reward with full observability.

	Full	Partial
Feeding	4.732	**45.731**

The ground truth reward (over which preferences are given) in this task rewards the robot for performing a 'scratching' motion. This scratching motion entails not only making contact with the target using the end-effector (being within a certain radius of the target coordinates), but also requires that the target contact position be greater than a certain δ away from the previous target contact position and that the exerted force be no more than a specified F_{max}. Although we provide the reward model all the necessary information to infer this scratching motion (including information about the state at the previous timestep; see Appx. A.4 for details), we find that the reward model is not able to learn the scratching motion. As seen in Fig. 7c, once we explicitly include an indicator variable, 'scratched', denoting whether the robot has successfully performed the aforementioned scratching motion, performance drastically increases. We suspect that the reward model's tendency to pick

Table 4: **KL Divergence: Complex Causal Features.** Learning a reward without the 'scratched' indicator feature causes the learned reward to pick up on spurious correlations involving other features, leading to a greater amount of distribution shift (KL divergence) during policy optimization.

WITH 'SCRATCHED'	WITHOUT 'SCRATCHED'
8.570	**15.553**



(a) Increasing dataset size (b) Increasing model capacity (c) With 'scratched' indicator

Figure 7: **Feature Complexity.** Learning the 'scratched' feature is difficult, despite increasing dataset size and model capacity. Performance significantly improves when given access to a high-level 'scratched' feature.

up on spurious correlations that occur consistently over the course of the trajectory involving just a few variables prevents it from learning the true causal relation that involves many variables, each of which are causal only in a particular context. This is supported by Fig 9c, which shows gradients staying relatively constant for each feature rather than varying with time and context. As a result, the learned reward without 'scratched' leads to a greater amount of distribution shift (Table 4). Future work in reward learning would do well to explore how to learn these more complex, multi-feature causal relationships.

6 CONCLUSION

Our work provides an analysis into the generalization of learned reward functions. We identify three tasks where a standard way to generate diverse preference data often results in poor learned reward models. These models have good validation and test accuracy—in some of our settings even 99.5%, seeming to distinguish the basics of task success vs. failure. However, optimizing them pushes the policy outside of the training distribution, where the model falsely believes the reward is highest—we demonstrate this via saliency maps, the EPIC distance, and KL divergence and also qualitatively by examining the resulting behavior (e.g., going behind a patient's head when feeding them). This out-of-distribution effect results in policies that the learned model assesses to have high reward, whereas in reality they have poor true reward and low success rates. Digging into the challenges of reward learning, we find that it is easy for the models to pick up on non-causal features, as some issues go away when we eliminate these non-causal features from the input. Furthermore, noisy preference data aggravates poor generalization, which is not fixed by simply getting more noisy data from the same distribution—neither is test error predictive of performance, which points to causal confusion as the culprit. And when not all causal features are observable, the learned model will struggle even though there exists a high-performing reward defined only on what is observable.

Based on our results, we have identified several promising directions for aleviating causal reward misidentification. First, our results show that causally confused rewards result in greater distribution shifts during policy optimization. As such, future work should investigate methods for penalizing excess exploration beyond the training data distribution. In Appx. A.12, we examine the effect of adding a direct penalty on the KL divergence into the cost function during RL. We find that, unfortunately, the RL agent is again able to hack the reward—this time, the performance and causal reward misidentification are even worse because of the additional degree of freedom in the reward function afforded by the discriminator penalty. However, we hypothesize that there may be other less obvious ways to successfully incorporate such a penalty. Next, our results demonstrate that spu-

rious ("nuisance") features can significantly increase the chance of causal reward misidentification. Thus, in cases where we can query for human knowledge on exactly which features are spurious or causal, we can use this feedback to learn which features are non-causal or remove non-causal features entirely. Further, our results indicate that high-level features such as the 'scratch' feature are difficult for neural networks to learn, even when all the necessary low-level information is available. Future work should examine incorporating methods for learning high-level features. One promising approach would be to explicitly learn the causal feature(s) by querying for feature traces (Bobu et al., 2021). Finally, we recognize that active and iterative methods for data acquisition is also a widely popular solution—as such, we provide a preliminary exploration in Appx. A.13. Using other metrics to compare learned rewards (such as DARD, by Wulfe et al. (2022)) may also prove fruitful.

Our work cautions that reward learning is brittle—natural choices for acquiring data, deciding on the amount of data, or defining the input space can lead to models that seem very close to the true reward, but lead to spectacular failures when optimized. And even though active and iterative methods for data acquisition may alleviate these issues, the problem of learning the best reward model when some of the causal features are unobservable remains wide open.

REFERENCES

Erdem Biyik and Dorsa Sadigh. Batch active preference-based learning of reward functions. In *Conference on robot learning*, pp. 519–528. PMLR, 2018.

Andreea Bobu, Marius Wiggert, Claire Tomlin, and Anca D Dragan. Feature expansive reward learning: Rethinking human input. In *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 216–224, 2021.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

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

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*, pp. 783–792. PMLR, 2019.

Daniel Brown, Russell Coleman, Ravi Srinivasan, and Scott Niekum. Safe imitation learning via fast bayesian reward inference from preferences. In *International Conference on Machine Learning*. PMLR, 2020a.

Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on robot learning*, pp. 330–359. PMLR, 2020b.

Paul F Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *NIPS*, 2017.

Pim De Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Zackory Erickson, Vamsee Gangaram, Ariel Kapusta, C Karen Liu, and Charles C Kemp. Assistive gym: A physics simulation framework for assistive robotics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10169–10176. IEEE, 2020.

Adam Gleave, Michael D Dennis, Shane Legg, Stuart Russell, and Jan Leike. Quantifying differences in reward functions. In *International Conference on Learning Representations*, 2020.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Jerry Zhi-Yang He and Anca D Dragan. Assisted robust reward design. *arXiv preprint arXiv:2111.09884*, 2021.

Ferenc Huszár. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. *arXiv preprint arXiv:1811.06521*, 2018.

Zaynah Javed, Daniel S Brown, Satvik Sharma, Jerry Zhu, Ashwin Balakrishna, Marek Petrik, Anca Dragan, and Ken Goldberg. Policy gradient bayesian robust optimization for imitation learning. In *International Conference on Machine Learning*, pp. 4785–4796. PMLR, 2021.

Maurice George Kendall. Rank correlation methods. 1948.

Victoria Krakovna, Jonathan Uesato, Vladimir Mikulik, Matthew Rahtz, Tom Everitt, Ramana Kumar, Zac Kenton, Jan Leike, and Shane Legg. Specification gaming: the flip side of ai ingenuity. *DeepMind Blog*, 2020.

Cassidy Laidlaw and Anca Dragan. The boltzmann policy distribution: Accounting for systematic suboptimality in human models. In *International Conference on Learning Representations*, 2021.

Lauro Langosco Di Langosco, Jack Koch, Lee D Sharkey, Jacob Pfau, and David Krueger. Goal misgeneralization in deep reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 12004–12019. PMLR, 17–23 Jul 2022.

Kimin Lee, Laura Smith, Anca Dragan, and Pieter Abbeel. B-pref: Benchmarking preference-based reinforcement learning. *arXiv preprint arXiv:2111.03026*, 2021.

Eric J Michaud, Adam Gleave, and Stuart Russell. Understanding learned reward functions. *arXiv preprint arXiv:2012.05862*, 2020.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287, 1999.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. *arXiv preprint arXiv:2201.03544*, 2022.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems*, 2017.

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

Rohin Shah, Vikrant Varma, Ramana Kumar, Mary Phuong, Victoria Krakovna, Jonathan Uesato, and Zac Kenton. Goal misgeneralization: Why correct specifications aren't enough for correct goals. *arXiv preprint arXiv:2210.01790*, 2022.

Siegel Sidney. Nonparametric statistics for the behavioral sciences. *The Journal of Nervous and Mental Disease*, 125(3):497, 1957.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Gokul Swamy, Sanjiban Choudhury, J Andrew Bagnell, and Zhiwei Steven Wu. Causal imitation learning under temporally correlated noise. *arXiv preprint arXiv:2202.01312*, 2022.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

Christian Wirth, Riad Akrour, Gerhard Neumann, Johannes Fürnkranz, et al. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136):1–46, 2017.

Blake Wulfe, Logan Michael Ellis, Jean Mercat, Rowan Thomas McAllister, and Adrien Gaidon. Dynamics-aware comparison of learned reward functions. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=CALFyKVs87`.

Junzhe Zhang, Daniel Kumor, and Elias Bareinboim. Causal imitation learning with unobserved confounders. *Advances in neural information processing systems*, 33:12263–12274, 2020.

A APPENDIX

A.1 PREFERENCE LEARNING TRAINING DETAILS

To learn a reward function from preferences, we assume access to a set of pair-wise preference labels $\mathcal{P}$ over trajectories $\tau_1, \ldots, \tau_N$, where $(i, j) \in \mathcal{P}$ implies that $\tau_i \prec \tau_j$. We then optimize a reward function $r_\theta : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ parameterized by θ that maximizes the likelihood:

$$\mathcal{L}(\theta) = \prod_{(i,j)\in\mathcal{P}} \frac{\exp(r_\theta(\tau_j))}{\exp(r_\theta(\tau_i)) + \exp(r_\theta(\tau_j))}. \tag{4}$$

This likelihood function is differentiable, allowing us to leverage non-linear function approximation to learn the reward function from trajectory preferences. In practice, we use the Adam optimizer in PyTorch to learn the reward function, r_θ, and then use PPO Schulman et al. (2017) or SAC Haarnoja et al. (2018) for policy optimization given r_θ.

We perform preference learning on three dataset sizes (given in terms of unique pairwise comparisons): SMALL (780), MEDIUM (7140), and LARGE (52326). Our test set is composed of 1770 unique pairwise preferences drawn from a disjoint set of 60 trajectories. See Appx. A.2 for dataset generation details. Hyperparameters—learning rate and weight decay—are tuned coarsely using the MEDIUM dataset size due to runtime limits and cost of computation. The tuned hyperparameters (best performance on a held-out validation set) for each environment are as follows: Reacher: weightdecay=0.0001, lr=0.01, Feeding: weightdecay=0.00001, lr=0.001, Itch Scratching: weightdecay=0.001, lr=0.001.

Using the dataset of preferences (SMALL, MEDIUM, or LARGE) obtained offline, we train a neural network reward function approximator with two hidden layers (128 units and 64 units, respectively) and Leaky ReLU activations, after which we perform 1,000,000 timesteps of reinforcement learning with PPO Schulman et al. (2017) (for Feeding and Itch Scratching) and SAC Haarnoja et al. (2018) (for Reacher) using the learned reward function in place of the ground-truth reward function. We optimize the reward function approximator using Adam with weight decay and early-stopping on the validation loss (with a patience of 10 epochs). The full preference-based reward learning procedure we use is detailed in Alg. 1. The hyperparameters for the PPO and SAC agents are as follows (if not specified, then hyperparameters are set to RLLib's default values):

PPO:

- training batch size = 19200
- number of SGD iterations = 50
- SGD minibatch size = 128
- lambda = 0.95
- dimensions of fcnet hidden layers = [100, 100]

SAC:

- timesteps_per_iteration = 400
- learning_starts = 1000
- Q_model fcnet_hiddens = [100, 100]
- policy_model fcnet_hiddens = [100, 100]
- train_batch_size = 4096
- actor_learning_rate = 3e-3
- critic_learning_rate = 3e-3
- entropy_learning_rate = 3e-3

Algorithm 1 Preference-Based Reward Learning

Input: Training dataset of pairwise preferences d_{train}

```
R̂ ← initialize network
for epoch ∈ (0, 100) do
    for traj_i, traj_j, label ∈ d_train do
        r̂_i ← R̂(traj_i)
        r̂_j ← R̂(traj_j)
        loss ← CrossEntropyLoss(r̂_i, r̂_j, label)
        loss.backward()
    end for
    loss_val ← calculate cross-entropy loss on validation set
    if loss_val doesn't decrease for 10 epochs then
        break
    end if
end for
π̂ ← Run RL (PPO or SAC) using R̂ as reward for 1M iterations
```

Output: $\hat{R}, \hat{\pi}$

A.2 Synthetic Preference Generation

To enhance scalability and reproducibility, we automatically generate a large amount of synthetic trajectory preferences. This was done using an expert RL policy trained using the ground-truth reward function provided with each of environment. We then generate a large number of diverse trajectories by adding ϵ-greedy noise during policy rollouts, where ϵ is the probability that the policy takes an action uniformly at random from its action space. Thus, $\epsilon = 0$ corresponds to the fully trained RL policy and $\epsilon = 1$ corresponds to a uniformly random policy. As noted by Brown et al. (2020b), adding this type of disturbance will result in monotonically decreasing performance in expectation.

To generate pairwise preferences over trajectories, we select all pairs of trajectories from a set of 40, 120, and 324 total trajectories (for the SMALL, MEDIUM, and LARGE dataset sizes, respectively) generated with ϵ-greedy rollouts for $\epsilon \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. We use held-out sets of trajectories for validation and testing. We then use the ground-truth reward functions provided by each environment to provide ground-truth preference labels.

A.3 Evidence of Causal Confusion

Fig. 8 provides plots of select features' spurious correlations with the true reward. Fig. 9 shows gradient saliency maps of the learned rewards that elucidate how the learned reward may improperly 'weight' a feature.

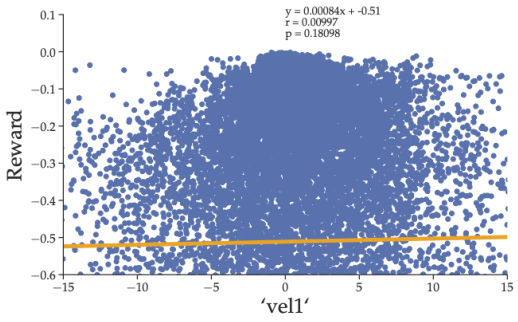
A.4 Input Features for Environments

Features that are distractors are labeled with a *D*. Features that are causal are labeled with a *C*. Reacher:

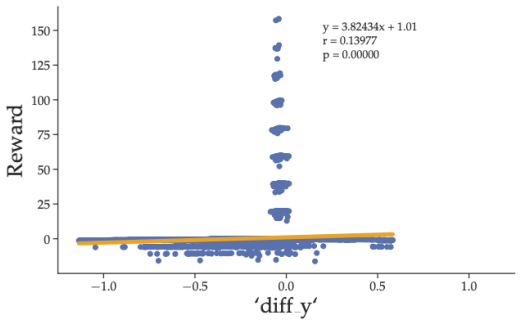
- D - Cos of angle of first and second arm (2)
- D - Sin of angle of first and second arm (2)
- D - Coordinates of target (2)
- D - Angular velocity of first and second arm (2)
- C - Position_fingertip - position_target (3)
- C - Action — Torque applied at first and second hinge (2)

Feeding:

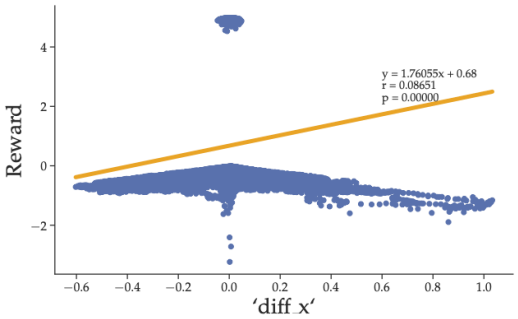
- D - Spoon_pos_real (3)



(a) Reacher, 'vel_1' (feature 6)

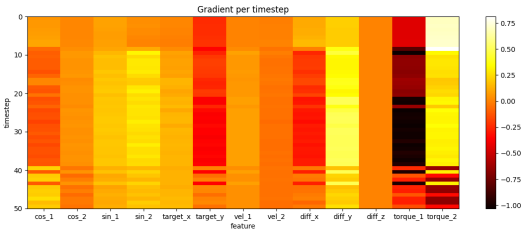


(b) Feeding, 'diff_y' (feature 8)

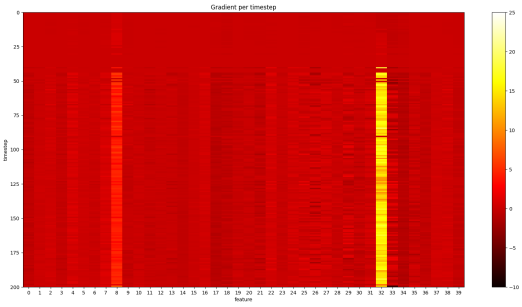


(c) Itch Scratching, 'diff_x' (feature 7)

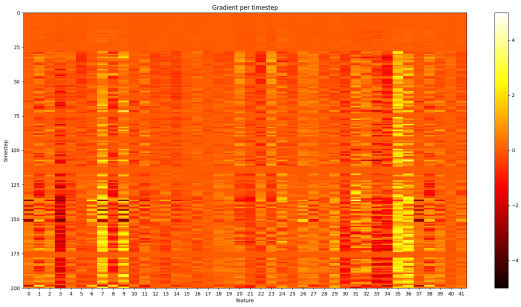
Figure 8: **Correlations of Spuriously 'Weighted' Features with Reward.**



(a) Reacher



(b) Feeding



(c) Itch Scratching

Figure 9: **Gradient Saliency Maps for Fig 2.**

- D - Spoon_orient_real (4)
- C - spoon_pos_real - target_pos_real (3)
- D - Robot_joint_angles (7)
- D - Head_pos_real (3)
- D - Head_orient_real (4)
- C - Spoon_force_on_human (1)
- C - Action (7)
- C - Foods_in_mouth (1)
- C - Foods_on_floor (1)
- C - Foods_hit_human (1)
- C - Sum_food_mouth_velocities (1)
- C - Prev_spoon_pos_real (3)
- C - Robot_force_on_human (1)

Itch Scratching:

- C - Tool_pos_real (3)
- D - Tool_orient_real (4)
- C - Tool_pos_real - target_pos_real (3)
- D - Target_pos_real (3)
- D - Robot_joint_angles (7)
- D - Shoulder_pos_real (3)
- D - Elbow_pos_real (3)
- D - Wrist_pos_real (3)
- C - Tool_force (1)
- C - Action (7)
- C - Prev_tool_pos_real (3)
- C - Robot_force_on_human (1)
- C - Prev_tool_force (1)

A.5 Ground Truth Reward Functions for Environments

We outline the ground truth reward functions for each environment below. We also refer the reader to the publicly available code repositories for each environment, which describe the rewards in more detail.

Reacher (`https://github.com/openai/gym`):

$$R(x) = -||position_{fingertip} - position_{target}||_2 + -||action||_2^2$$

Feeding (`https://github.com/Healthcare-Robotics/assistive-gym`):

$$\begin{aligned} r_{distance} &= -||pos_{target} - pos_{spoon}||_2 \\ r_{action} &= -||action||_2 \\ r_{food} &= f(Foods_in_mouth, Foods_on_floor) \\ preferences_score &= g(Foods_hit_human, Sum_food_mouth_velocities, \\ &\quad Spoon_pos_real, Prev_spoon_pos_real, Robot_force_on_human) \\ R(x) &= W_{distance} * r_{distance} + W_{action} * r_{action} + W_{food} * r_{food} + preferences_score \end{aligned}$$

Table 5: **KL Divergence: Distractor Features.**

ENV	WITH	WITHOUT
FEEDING	4.732	**6.985**
ITCH SCRATCHING	**18.749**	8.570

Itch Scratching (`https://github.com/Healthcare-Robotics/assistive-gym`):

$$
\begin{aligned}
r_{distance} &= -||pos_{target} - pos_{spoon}||_2 \\
r_{action} &= -||action||_2 \\
r_{scratch} &= f(Tool_pos_real, Target_pos_real, Prev_tool_pos_real, \\
&\quad Tool_force, Prev_tool_force) \\
preferences_score &= g(Spoon_pos_real, Prev_spoon_pos_real, \\
&\quad Robot_force_on_human, Tool_force, Target_pos_real) \\
R(x) &= W_{distance} * r_{distance} + W_{action} * r_{action} + W_{scratch} * r_{food} + preferences_score
\end{aligned}
$$

A.6 EVALUATING LEARNED REWARD FUNCTIONS

Saliency maps: We produce saliency maps of the learned reward as follows: we forward propagate a single rollout from the learned reward's policy through the reward network. Then, with the reward model's weights fixed, we backpropagate the output from the forward pass through the network and into the input (the policy rollout) to obtain the gradient with respect to each feature in the observation-action pair at each timestep.

Kullback-Leibler divergence: For each learned reward, we sample 50 trajectories from the reward's training data and from the resulting policy (taking care to label each trajectory's origin distribution). From these 50x2 trajectories, we create the training and validation splits and then flatten each group of trajectories into a dataset of observation-action pairs. We train a discriminator model (hidden dimensions of 128x128x128) to distinguish between observation-action pairs seen during reward learning and those during RL by minimizing the binary cross-entropy loss. We tune hyperparameters (learning rate, weight decay) on the validation loss and accuracy.

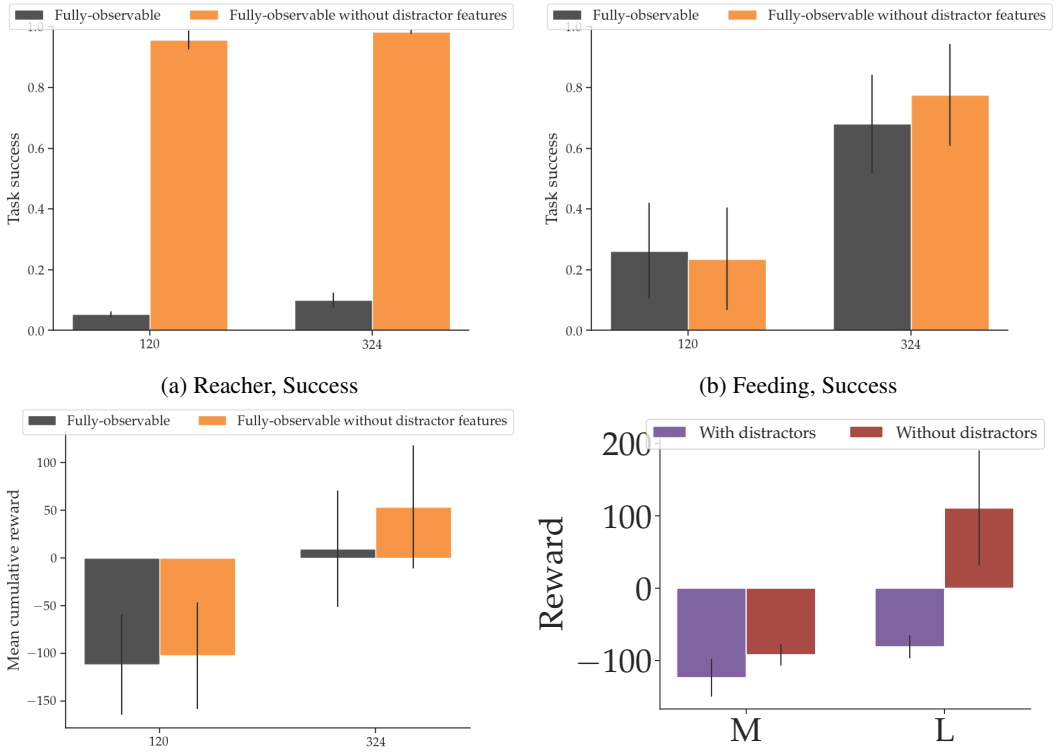
With this trained discriminator model, we calculate $D_{KL}(p||q)$ by taking the discriminator's negative mean return/logit of all reward learning observation-action pairs, where p is the reward learning distribution and q is the policy optimization distribution. Similarly, we calculate $D_{KL}(q||p)$ by taking the mean return of all RL observation-action pairs. Since the KL divergence is not symmetric, we report $D_{KL}(p||q) + D_{KL}(q||p)$, or the symmetric KL divergence. For a proof on why a discriminator can be used to approximate the KL divergence, we refer the reader to Appendix B.2 of Laidlaw & Dragan (2021).

A.7 DISTRACTOR FEATURES

Fig. 10 shows the additional results for removing distractors. Fig. 11 shows the saliency map for the Reacher learned reward without distractors. Table 5 shows additional KL divergences for the Feeding and Itch Scratching environments. We note here that the reason removing non-causal distractor features appears to not benefit for Feeding is that the main spurious correlation (discussed in Section 4) involves one of the *causal* features—namely, the difference between spoon position and target position. Thus, removing purely non-causal features fails to address this issue in Feeding.

A.8 MODEL CAPACITY

We study the effect of model capacity on the learned reward. We find that, despite careful tuning of hyperparameters with each model and dataset size, increasing the capacity of the reward model does not necessarily result in an increase in subsequent policy performance. In fact, as seen in Fig. 12, only the Feeding task trained with the LARGE dataset size (52326 preferences) benefits from steadily increasing reward model capacity. Indeed, although increasing the capacity for the



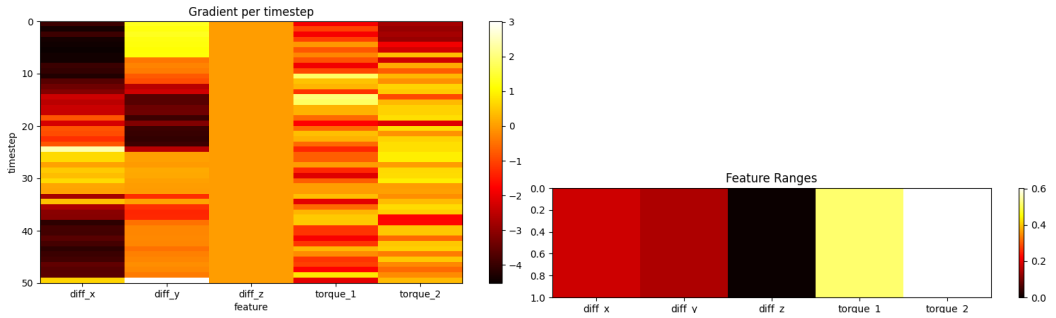
(a) Reacher, Success

(b) Feeding, Success

(c) Feeding, Reward

(d) Itch Scratching, Reward

Figure 10: **Distractor Features.**



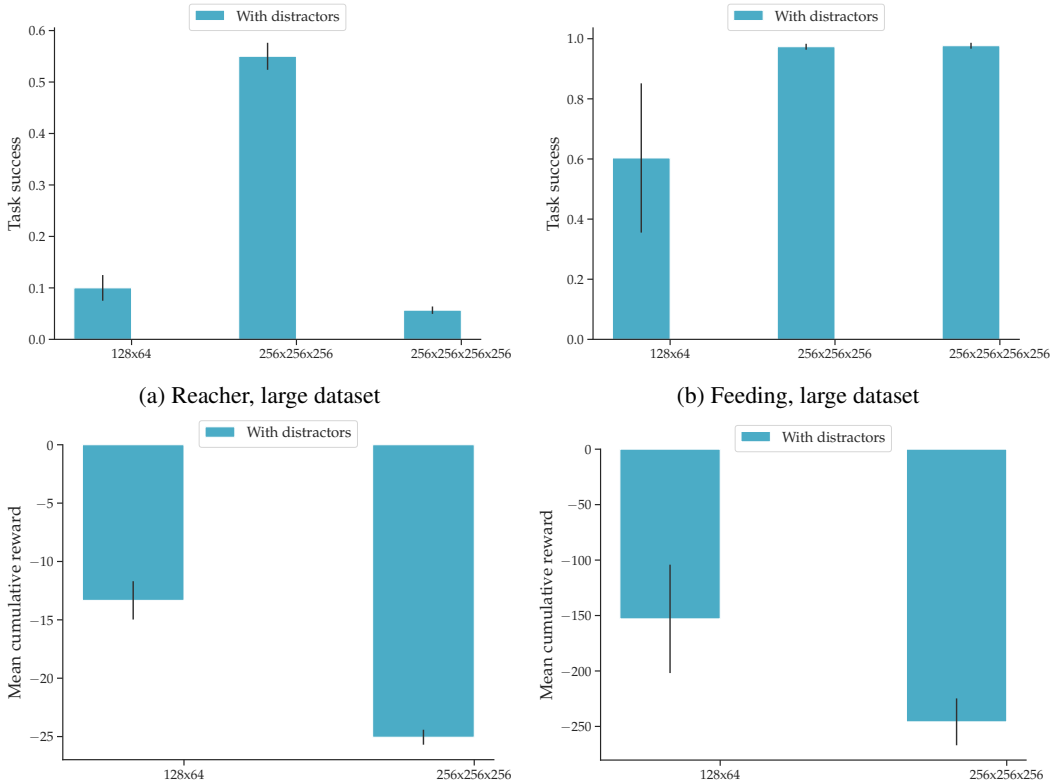
(a) Gradients - W/out distractors

(b) Feature Ranges - W/out distractors

Figure 11: **Saliency Maps: Without Distractor Features.**

Reacher task appears to initially increase performance for the large dataset case (Fig. 12a), performance drops back down to below the performance of the smallest model size when the model size is further increased. Further, increasing model size seems to decrease the performance on the small datasets. This is not surprising, as benefiting from larger capacity tends to require increasing the amount of data, and validation-set accuracy tends to agree with learned reward performance. Interestingly though, the EPIC distance paints the picture that performance improves in distribution but suffers out-of-distribution.

To examine these results further, we compare the learned rewards directly with the ground truth reward (without performing policy optimization) with the EPIC pseudometric in Table 6. Interestingly, the EPIC distances between the learned rewards and the ground truth reward vary widely depending on the choice of coverage distribution. With the distribution of states seen during reward learning (generated by randomly switching between an expert and random policy), the larger (256x256x256)



(a) Reacher, large dataset

(b) Feeding, large dataset

(c) Reacher, small dataset

(d) Feeding, small dataset

Figure 12: **Model Capacity.** Increasing model capacity increases performance when we use the large dataset ($\binom{324}{2} = 52326$ preferences), to a certain extent. However, increasing capacity decreases performance on the small dataset ($\binom{40}{2} = 780$ preferences).

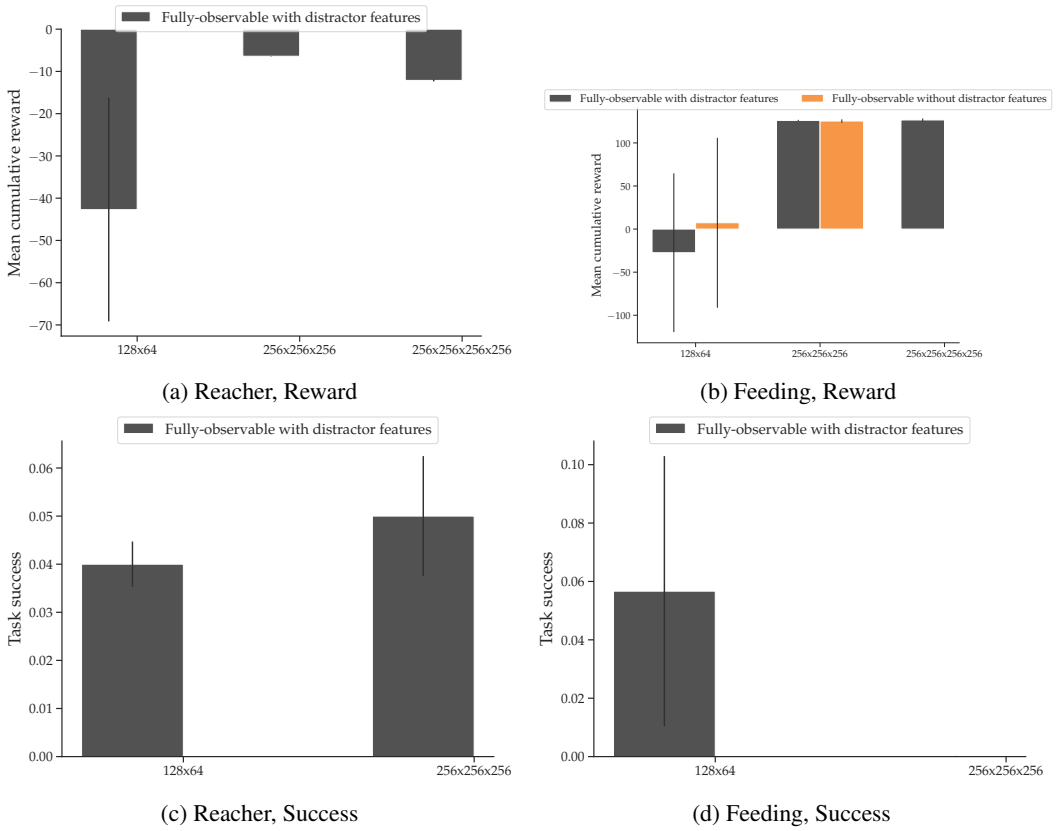
Table 6: **EPIC Distances, Model Capacity.** The reward model with larger capacity (hidden layer dimensions of 256x256x256) appears closer to the ground truth than the reward model with smaller capacity (hidden layer dimensions of 128x64) *on the distribution of states seen during reward learning*, but is much further from the ground truth *on the distribution of states seen during reinforcement learning*.

REWL / RL	256x256x256	128x64
GROUND TRUTH	0.210 / 0.707	0.233 / 0.598

model appears closer to the ground truth reward than the smaller (128x64) one. However, these results are flipped when we instead evaluate the pseudometric on the distribution of states seen during policy optimization. This suggests that increasing the capacity of the reward model allows it to more closely 'mimic' the ground truth reward *on the reward learning distribution* (hence the lower EPIC distance) but worsens its ability to generalize to the reinforcement learning distribution. We see a similar effect with the divergence metric in Tab. 7. .

A.9 NOISE IN STATED PREFERENCES

Fig. 14 shows additional results. Table 8 shows the KL divergences for each model configuration in Fig. 5. Table 9 displays complete results of EPIC distances.



(a) Reacher, Reward

(b) Feeding, Reward

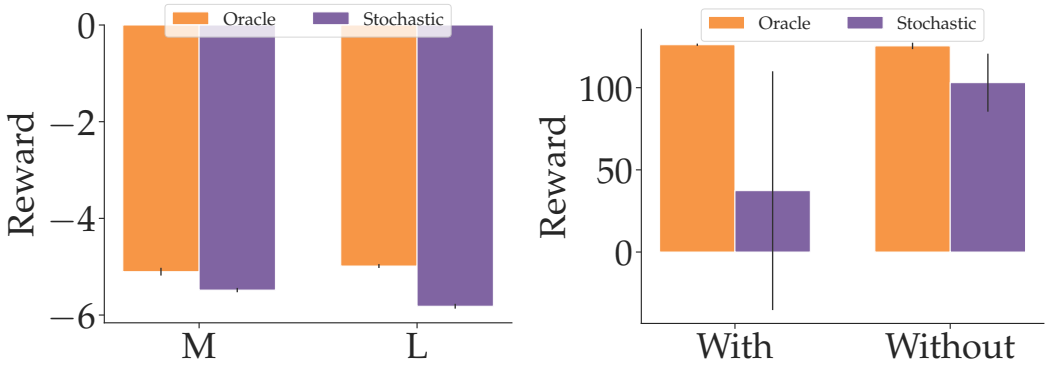
(c) Reacher, Success

(d) Feeding, Success

Figure 13: **Model Capacity.**

Table 7: **KL Divergence Approximations, Model Capacity.** Increasing model capacity may result in a much larger distribution shift (as measured by the KL Divergence).

	128x64	256x256x256	128x64	256x256x256	256x256x256x256
REACHER	64.165	57.724	46.031	22.187	88.927
FEEDING	25.352	32.836	16.900	4.732	9.909



(a) Reacher, Reward

(b) Feeding, Reward

Figure 14: **Noise in Stated Preferences.**

Table 8: **KL Divergence Approximations, Noise in Stated Preferences.**

	ORACLE	STOCHASTIC
REACHER, MEDIUM DATASET	6.762	8.856
REACHER, LARGE DATASET	7.365	6.739
FEEDING, WITH DISTRACTORS	4.732	13.188
FEEDING, WITHOUT DISTRACTORS	6.985	8.046

Table 9: **EPIC Distances, Noise in Stated Preferences.** Causally confused rewards (here, as a result of NOISY data) appear 'closer' to the ground truth reward than the proper learned rewards (here, with PERFECT data) on the reward learning distribution (RewL), but further on the RL distribution. EPIC distances are computed on the rewards learned with distractors in the Feeding environment in Fig. 5c and averaged over three seeds.

REWL / RL	GROUND TRUTH	STOCHASTIC	ORACLE
GROUND TRUTH	0.000 / 0.000	**0.136 / 0.164**	**0.165 / 0.130**
NOISY	**0.136 / 0.164**	0.000 / 0.000	0.220 / 0.209
PERFECT	**0.165 / 0.130**	0.220 / 0.209	0.000 / 0.000

A.10 PARTIAL OBSERVABILITY OF CAUSAL FEATURES

Fig. 15 displays gradient maps of a reward that partially observes the causal features and a reward that fully observes the causal features. Table 10 displays the EPIC distances.

A.11 DATA GENERATION METHODS

Fig. 16 displays a comparison of our method of generating a diverse dataset of trajectory preferences and the method of using a checkpointed RL policye (proposed by (Brown et al., 2019)).

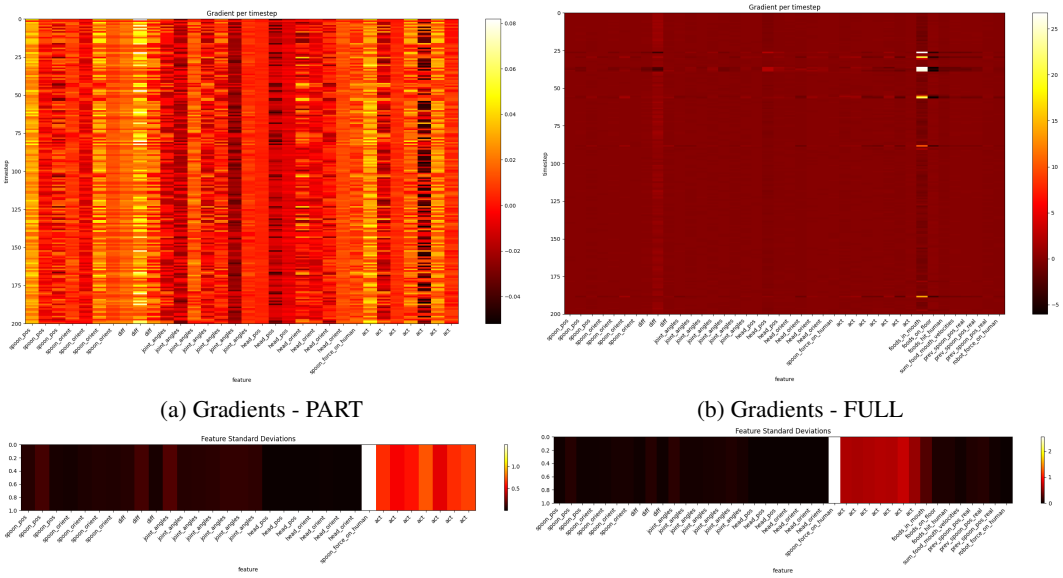
A.12 PENALIZING THE KL DIVERGENCE

Inspired by the results showing that causally confused rewards result in greater distribution shifts during policy optimization, we attempt to address this by adding a KL divergence penalty term to the reward. Specifically, in addition to using the reward learned from preferences, we incorporate a pretrained discriminator (trained to discriminate between state-action pairs from the reward learning distribution and the RL distribution) to estimate the KL divergence, as detailed earlier in Section 3 and Appx. A.6. This KL divergence is scaled by a hyperparameter λ and subtracted from the reward learned from preferences.

We show results for $\lambda = 0, 0.01, 0.1, 1, 10$ in Fig. 17. Using a discriminator that is pretrained offline (the discriminator isn't updated during the RL process) to penalize the KL divergence does not appear to help the performance or alleviate causal reward misidentification in any way. It appears that the RL agent is again able to hack the reward—this time, the performance and causal confusion are even worse because of the additional degree of freedom in the reward function afforded by the discriminator penalty.

Table 10: **EPIC Distances, Partial Observability.**

REWL / RL	GROUND TRUTH	PARTIAL	FULL
GROUND TRUTH	0.000 / 0.000	0.684 / 0.699	0.165 / 0.309
PARTIAL	0.684 / 0.699	0.000 / 0.000	0.685 / 0.697
FULL	0.165 / 0.309	0.685 / 0.697	0.000 / 0.000



(a) Gradients - PART

(b) Gradients - FULL

(c) Feature Standard Deviations - PART

(d) Feature Standard Deviations - FULL

Figure 15: **Saliency Maps, Partial Observability.**

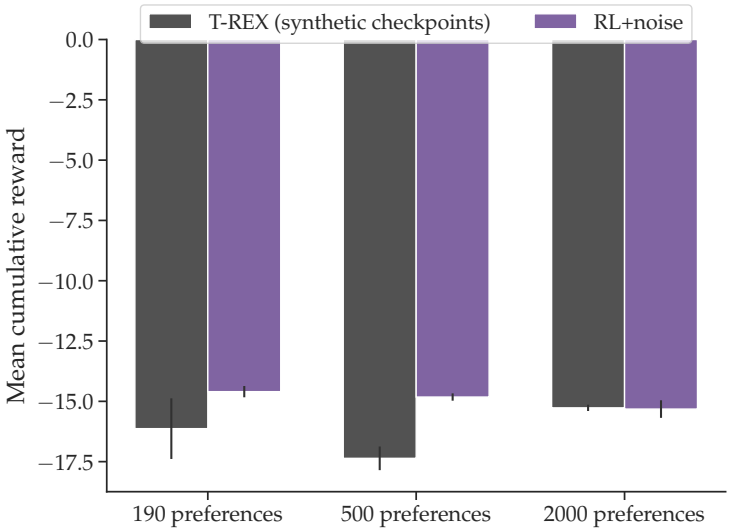
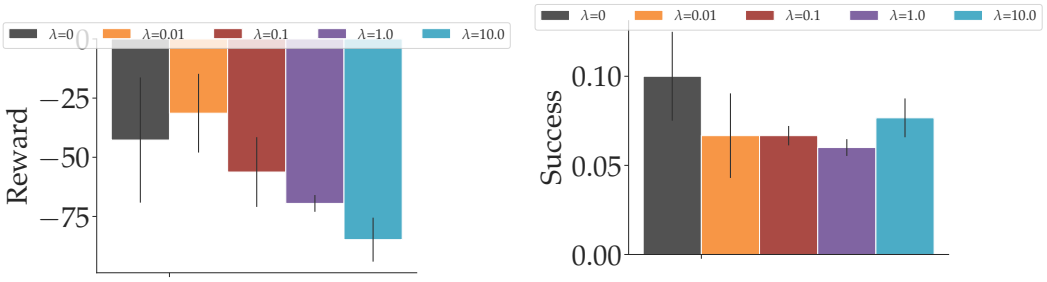


Figure 16: **TREX vs. RL+Noise.** Our method of generating diverse trajectories for preference learning performs on par with, if not better than, the method of using rollouts taken from a check-pointed RL policy, as proposed by (Brown et al., 2019). Displayed are cumulative trajectory rewards from the Reacher environment.



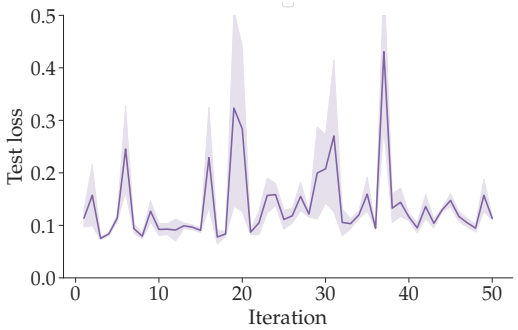
(a) Reacher, Reward

(b) Reacher, Success

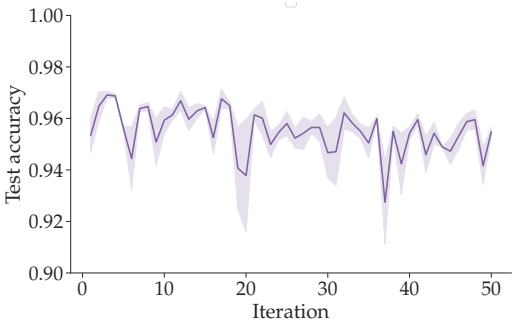
Figure 17: **Penalizing the KL Divergence.**

A.13 Iterative Preference-Based Reward Learning from Online Data

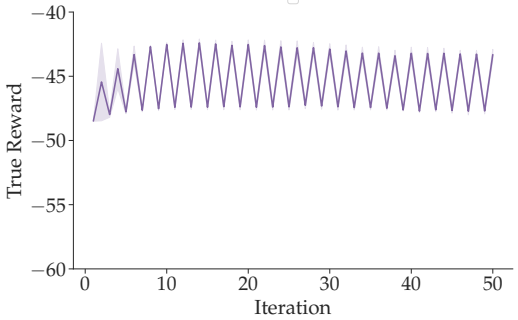
Following Christiano et al. (2017), we verify the occurrence of causal reward misidentification when learning from data that is actively queried online (during the learning process). As before, we use the initial set of offline trajectory and preference data (acquired via Appx. A.2) to train a reward function, which we then use to optimize a policy using RL. However, we then sample 10 trajectories from the most recently optimized policy and generate preferences for all possible pairs involving the sampled trajectories, which we concatenate to our existing training dataset. Using the (now augmented) dataset, we fine-tune the reward function and policy for 10 epochs and 100 iterations, respectively. We repeat this process of taking rollouts from the most recent policy and fine-tuning the existing learned reward and policy for 50 iterations. Results are displayed in Fig. 18. We observe that causal reward misidentification appears to still be present—test error is very low, while resulting policy performance (measured by true reward and success) remains poor.



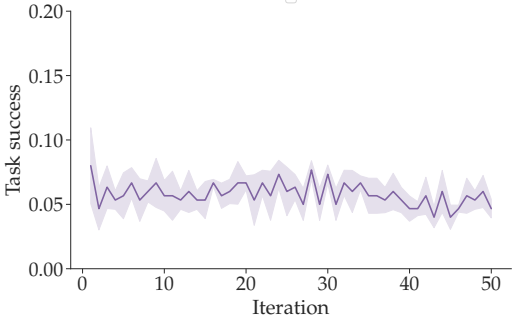
(a) Reacher, Test Loss



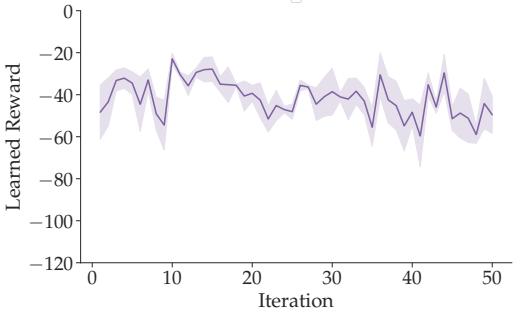
(b) Reacher, Test Accuracy



(c) Reacher, True Reward



(d) Reacher, Success



(e) Reacher, Learned Reward

Figure 18: **Training the reward and policy together iteratively.**