Positive-Unlabeled Reward Learning

Danfei Xu
Stanford University
danfei@cs.stanford.edu

Misha Denil
Deepmind
mdenil@google.com

Abstract: Learning reward functions from data is a promising path towards achieving scalable Reinforcement Learning (RL) for robotics. However, a major challenge in training agents from learned reward models is that the agent can learn to exploit errors in the reward model to achieve high reward behaviors that do not correspond to the intended task. These reward delusions can lead to unintended and even dangerous behaviors. On the other hand, adversarial imitation learning frameworks [1] tend to suffer the opposite problem, where the discriminator learns to trivially distinguish agent and expert behavior, resulting in reward models that produce low reward signal regardless of the input state. In this paper, we connect these two classes of reward learning methods to positive-unlabeled (PU) learning, and show that by applying a large-scale PU learning algorithm to the reward learning problem, we can address both the reward under- and over-estimation problems simultaneously. Our approach drastically improves both GAIL and supervised reward learning, without any additional assumptions.

1 Introduction

While Reinforcement Learning (RL) has shown itself to be a powerful tool for automating control and decision making, hand-specifying reward requires significant engineering effort, especially in real-world settings. Recent works have made promising progress in training reward models from human annotations, such as ratings [2] and behavior preferences [3, 4]. However, in practice, these annotations are expensive to curate and thus often cover only a fraction of the state space. As a result, the learned reward models may make large errors on unannotated states, and policy learning algorithms tend to exploit these delusions to achieve high pseudo-reward via unintended behaviors [5]. Practical solutions often require a human to provide annotations iteratively, in lockstep with the policy training loop [6, 4], resulting in a even more laborious process.

On the other hand, works in Inverse Reinforcement Learning (IRL) aim to learn reward models directly from expert behaviors [7, 8], but scaling these methods to high-dimensional state spaces remains a challenge. Recently, Ho and Ermon [1] introduced Generative Adversarial Imitation Learning (GAIL), which directly recovers expert behaviors via a proxy reward[1] derived from a discriminator that is trained to distinguish between expert demonstrations and the behaviors of an imitating policy. Ho and Ermon [1], and many follow-up works, show that GAIL can learn complex behaviors. However, a major limitation of GAIL-like methods is that the learned reward functions may *overfit* to trivially distinguish between the expert and the agent via features that are irrelevant to the intended behaviors [10, 11, 12, 13].

In this paper, we develop a unified reward learning algorithm that addresses both the reward delusion problem that arises with supervised reward learning and the overfitting problem encountered with GAIL. We begin by illustrating how a binary (positive-negative) classification problem arises in both settings. This classification problem corresponds to *support set estimation* for supervised reward learning, and the standard discriminator learning formulation for GAIL.

We then show how both issues above can be addressed by reformulating the corresponding (PN) classification problems as PU classification [14, 15, 16]. The resulting discriminators can be trained

[1]A proxy reward because GAIL will in general not recover an underlying environment reward for the task at hand. See Ho and Ermon [1] and Fu et al. [9] for more discussions.

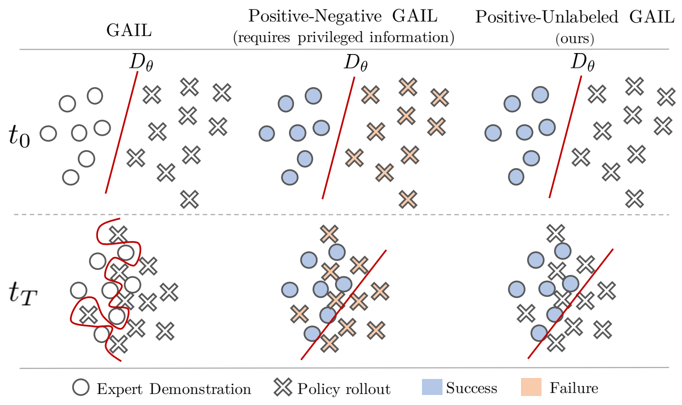

Figure 1: (Best viewed in color) Illustration of the overfitting problem of GAIL [1], along with our solution. Early in training (t_0) agents generally fail to complete the task, and the discriminator can trivially separate agent and expert experience. Later in training when the agent becomes competent(t_T), the GAIL discriminator often overfits to the demonstrations by focusing on irrelevant features, preventing the agent from improving. If we could detect success and failure for policy rollouts, then we could train the discriminator to distinguish between them directly (middle column), but this information is generally not available. Our solution (right column) uses PU learning to train a success vs. failure classifier using positive (expert) and unlabeled (agent) data, without requiring success and failure annotations for policy rollouts.

reliably by applying a recent large scale PU learning algorithm [17]. We call this approach Positive-Unlabeled Reward Learning (PURL).

We empirically evaluate PURL on a standard benchmark task and two challenging robotic manipulation tasks with pixel inputs and continuous control space. We show that PURL is able to learn robust reward functions with limited reward supervisions, and outperform a robust GAIL baseline by a large margin. In addition, we test the robustness of PURL by comparing against GAIL and supervised reward learning when there is a domain gap between training and testing environments.

2 Related Works

In this paper, we focus on two classes of reward learning methods: learning from expert demonstration by adversarial imitation learning [1] and learning from supervised reward signals [2].

Generative Adversarial Imitation Learning (GAIL) [1] has accrued a wide range of variants including Li et al. [18], Fu et al. [9], and Baram et al. [19]. The central idea is to train a discriminator model to assign higher reward values to expert demonstrations than the imitating policy through a binary classifcation objective. However, a key limitation of GAIL is that the discriminator may overfit to features of the observations that are irrelevant to the intended behaviors [13]. This issue is especially pronounced in the cases of high-dimensional observations, where artifacts such as lighting can be used to trivially distinguish the data sources.

A few works have attempted to address the overfitting problem of GAIL by regularizing the discriminator [10, 11, 12, 20]. Peng et al. [10] introduce a variational information bottlebeck to hide information from the discriminator. Blondé and Kalousis [11] and Reed et al. [12] propose to limit the capacity of the discriminator model. Żołna et al. [13] regularize the discriminator directly accounting for spurious features. In this paper, we point out a more fundamental issue to the discriminator objective, namely that agent behavior should not be treated as negative data but rather unlabeled data. Bahdanau et al. [20] identify a similar overfitting problem in an adversarial reward learning setting, but they handle it by heuristically filtering out states that get assigned with high reward by the discriminator. In this paper, we introduce a principled method under the positive-unlabeled (PU) learning formulation to address the overfitting discriminator problem.

We also consider the case of learning a reward function from explicit supervision. There is a large body of works on learning reward functions from supervisions such as human ratings or preferences [2, 4, 21]. However, because the reward supervision often only covers a small part of the state space, RL algorithms may exploit the errors in the reward models to achieve high pseudo-reward from unintended behaviors [22]. We directly address this problem by training a discriminator to identify the reliable support set of the reward function. Our approach can be applied to deep network-based reward models that take raw pixels as inputs.

Our work builds on the recent progress of positive-unlabeled (PU) learning [16, 23, 17], where the task is to train a classifier from positively-labeled data and unlabeled data. Most existing works in PU learning require certain loss functions, linear models, and/or special optimizers [16, 23, 24]. Recently, Kiryo et al. [17] proposed a large-scale PU learning algorithm that can be applied to complex decision functions such as deep networks and can be optimized using common stochastic optimizers [25]. We frame reward learning problems discussed above as positive-unlabeled classification problems, and adapt the empirical risk estimator introduced in Kiryo et al. [17] to train reward functions from either expert demonstrations or reward supervisions, combined with unlabeled agent experiences.

3 Background

Generative Adversarial Imitation Learning (GAIL) Ho and Ermon [1] pose the Inverse Reinforcement Learning (IRL) problem as a two-player zero-sum game within the Generative Adversarial Networks (GAN) framework [26]: A discriminator D_θ is trained to distinguish between the behaviors of an agent policy π, and an expert policy π_e, while the agent policy is trained to generate behaviors that maximally resemble the expert. The game reaches equilibrium when the discriminator cannot distinguish the agent behaviors from the expert behaviors. We take a *state-only* formulation of GAIL that is known to improve the robustness of the algorithm [9, 27] and also does not require expert action information. The training objective for the discriminator is given by maximizing[2]

$$L_D = E_\pi[\log(1 - D_\theta(s))] + E_{\pi_e}[\log(D_\theta(s))] \ , \tag{1}$$

where $D_\theta : \mathcal{S} \to (0, 1)$. GAIL alternates between training the discriminator D_θ using Equation 1, and the policy π using the learned reward function $r_\theta(s) = -\log(1 - D_\theta(s))$.

Semi-supervised reward learning In addition to learning reward functions from expert demonstrations, we also consider the settings where we have access to a pool of experiences annotated with reward signals. Possible forms of annotations include human rating or ranking of states or trajectories [4, 28, 2]. Without loss of generality, we assume we have access to a set of reward-annotated environment states $\mathcal{D}_s = \{(s_i, r_i)\}_{i=1}^{N_s}$. The goal of *supervised* reward learning is to find a reward function r_ϕ that minimizes a supervised regression loss on $\mathcal{D}_s$. However, as will be discussed in Section 4, and shown empirically in Section 5, a major challenge in using r_ϕ for RL is that policy learning algorithms tend to exploit errors in r_ϕ to achieve high pseudo-reward [12]. In this paper, we address this challenge by making use of *unlabeled* states $\mathcal{D}_u = \{(s_i)\}_{i=1}^{N_u}$. An ideal source of such unlabeled states is the replay buffer filled with agent experiences. We refer to the new setup as the *semi-supervised* reward learning problem.

Positive-unlabeled learning Our reward learning algorithm builds on a long line of works in positive-unlabeled (PU) learning, which tackles the problem of learning classifiers from positive data $\mathcal{D}_p$, and unlabeled data $\mathcal{D}_u$. Following Kiryo et al. [17], let (X, Y) be the input and output of a binary classification problem, where $X \in \mathbb{R}^d$ and $Y \in \{0, 1\}$. We consider the two-sample problem setting, where the $\mathcal{D}_p \sim P(X, Y = 1)$ data and the $\mathcal{D}_u \sim P(X)$ data are drawn independently. Let $g : \mathbb{R}^d \to \mathbb{R}$ be the decision function, and $l : \mathbb{R} \times \{0, 1\} \to \mathbb{R}$ be the loss function. Using the labeled risk operator and its empirical counterpart,

$$R_g^y(\mathcal{D}) = \mathbb{E}_\mathcal{D}[l(g(x), y)] \ , \ \hat{R}_g^y(\mathcal{D}) = \hat{\mathbb{E}}_\mathcal{D}[l(g(x), y)] = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} l(g(x), y) \ , \tag{2}$$

we can write the risk of g as the sum of the true positive and true negative risks

$$R_g^{pn}(\mathcal{D}_p, \mathcal{D}_n) = \eta R_g^1(\mathcal{D}_p) + (1 - \eta) R_g^0(\mathcal{D}_n) \ . \tag{3}$$

where $\eta = P(Y = 1)$ is the *positive class prior*, which is assumed to be known.

[2]Note that we label expert data as 1 and agent data as 0, which is inverted from Ho and Ermon [1].

The key insight of PU learning is that for an appropriately chosen loss function, the true negative risk $R_g^0(\mathcal{D}_n)$ can be expressed in terms of positive and unlabeled data as [16, 23]

$$(1-\eta)R_g^0(\mathcal{D}_n) = R_g^0(\mathcal{D}_u) - \eta R_g^0(\mathcal{D}_p) \ . \tag{4}$$

By substituting the above identity into Equation 3, the PU risk of decision function g is

$$R_g^{pu}(\mathcal{D}_p, \mathcal{D}_u) = \eta R_g^1(\mathcal{D}_p) - \eta R_g^0(\mathcal{D}_p) + R_g^0(\mathcal{D}_u) \ , \tag{5}$$

which does not depend on the negative data. The empirical PU risk,

$$\hat{R}_g^{pu}(\mathcal{D}_p, \mathcal{D}_u) = \eta \hat{R}_g^1(\mathcal{D}_p) - \eta \hat{R}_g^0(\mathcal{D}_p) + \hat{R}_g^0(\mathcal{D}_u) \ , \tag{6}$$

is both an unbiased and consistent estimator of the true PU risk [16], and this estimator also enjoys other desirable properties such as upper-bounded risk [29]. However, these bounds become extremely loose when the decision function g becomes too complex, e.g. if g is a deep neural network, leading to issues with overfitting [17].

Kiryo et al. [17] subsequently propose a *non-negative* empirical estimator for Equation 5, for which the estimation error can be bounded even when g is complex, by enforcing the constraint $\hat{R}_g^0(\mathcal{D}_u) - \eta \hat{R}_g^0(\mathcal{D}_p) \geq 0$. In practice, this non-negativity constraint is relaxed with a slack variable $\beta \geq 0$ to account for mini-batch stochastic optimization. The non-negative PU risk estimator of Kiryo et al. [17] is given by

$$\tilde{R}_g^{pu}(\mathcal{D}_p, \mathcal{D}_u) = \eta \hat{R}_g^1(\mathcal{D}_p) + \max(-\beta, \hat{R}_g^0(\mathcal{D}_u) - \eta \hat{R}_g^0(\mathcal{D}_p)) \ . \tag{7}$$

In Section 4, we reduce both adversarial imitation learning and semi-supervised reward learning to PU learning problems, and show that we can use this non-negative risk estimator to improve both reward learning methods.

4 Positive-Unlabeled Reward Learning

Our goal is to learn reward functions entirely from data for RL. We consider two problem settings. In the first, we learn reward functions from expert demonstrations in an adversarial imitation framework [1]. The challenge is that the discriminator may trivially distinguish between agent and expert behavior and that the resulting reward function produces low reward regardless of the input. The second setting is to learn reward functions from a fixed dataset of reward-annotated experiences and a pool of unlabeled agent experiences. The challenge here is that the RL algorithms tend to exploit errors in the reward functions to achieve high pseudo-reward. In this section, we reduce both problems to positive-unlabeled (PU) learning problems and propose an unified reward learning algorithm based on large-scale PU learning [17].

4.1 Adversarial Imitation Learning as PU Learning

As explained in Section 3, the objective of the discriminator model in GAIL is to distinguish between policy rollouts and expert demonstrations. Here, we argue that this objective contributes to the discriminator overfitting problem, and we provide a new discriminator formulation that addresses the problem. The following discussion is also illustrated in Figure 1.

To see how the discriminator objective leads to overfitting, we first observe that the GAIL objective (Equation 1) is a *positive-negative* (PN) classification objective, where the positive data is generated by the expert policy π_e and the negative data is generated by the agent policy π.

Early in the learning process when the behavior of π is substantially different than that of π_e, distinguishing between the state visitation distributions of the two policies provides an informative signal for how the agent behavior can be modified to more reliably complete the task at hand. However, as π improves the agent behavior becomes increasingly similar to that of the expert, and it will begin to successfully complete the task. To continue distinguishing between the two data sources, the discriminator is forced to focus on features of the trajectories that are not relevant to task success.

If we could label trajectories as success (positive) or failure (negative) then we could avoid this overfitting problem by training the discriminator to distinguish between success and failure directly. Early in training this scheme would behave in the same way as GAIL, with the expert trajectories providing an initial set of positive examples, and the untrained agent producing examples of failures; however, the overfitting dynamics would be altered, effectively by co-opting successful agent trajectories to serve as additional expert demonstrations rather than as examples of behaviors to be avoided.

In practice we do have access to labels for the expert trajectories (in GAIL the expert trajectories are successful by definition), but we do not have a way of providing labels for agent trajectories. In GAIL the agent trajectories are treated as examples of failure; however, we are not forced to do so. Instead we can formulate the discriminator objective as PU learning problem, which allows us to train the discriminator to distinguish success and failure while treating the agent trajectories as an unlabeled mixture of positive and negative outcomes.

Formally, denote the expert demonstrations by $\mathcal{D}_{\pi_e}$, and the running stream of agent behavior data (i.e., the replay buffer) by $\mathcal{D}_{\pi}$. Based on Equation 5, we have an empirical risk estimate for the discriminator D_θ under the non-negative PU learning setup as

$$\tilde{R}^{pu}_{D_\theta}(\mathcal{D}_{\pi_e}, \mathcal{D}_\pi) = \eta \hat{R}^1_{D_\theta}(\mathcal{D}_{\pi_e}) + \max(-\beta, \hat{R}^0_{D_\theta}(\mathcal{D}_\pi) - \eta \hat{R}^0_{D_\theta}(\mathcal{D}_{\pi_e})) \ . \tag{8}$$

Switching to a more familiar notation, and assuming the standard logistic loss for the the discriminator, the above nn-PUGAIL objective becomes maximization of

$$\tilde{L}^{PU}_{D_\theta} = \hat{\mathbb{E}}_{\pi_e}[\log(D_\theta(s))] + \min(\frac{\beta}{\eta}, \frac{1}{\eta}\hat{\mathbb{E}}_{\pi}[\log(1 - D_\theta(s))] - \hat{\mathbb{E}}_{\pi_e}[\log(1 - D_\theta(s))]) \tag{9}$$

where $\hat{\mathbb{E}}$ denotes an empirical expectation. This should be compared with the ordinary GAIL objective in Equation 1.

4.2 Semi-supervised Reward Learning as PU Learning

We turn to the semi-supervised reward learning setup, where the problem is to learn reward functions from a set of reward-annotated experiences $\mathcal{D}_s = \{(s_i, r_i)\}_{i=1}^{N_s}$, and unlabeled agent experiences $\mathcal{D}_u = \{s_i\}_{i=1}^{N_u}$. As discussed previously, the main challenge in this setting is that policy learning tends to exploit errors in the learned reward model to achieve high pseudo-reward. To address the challenge, we consider modeling whether a reward prediction $r = r_\phi(s)$ is reliable by training a discriminator that can suppress the predicted reward if it is deemed unreliable. We define the discriminator D_θ on the input space of the supervised reward function. In other words, D_θ is a discriminator of which *states* result in reliable reward prediction, i.e., $D_\theta : S \to \{0, 1\}$, and if $r_\phi(s) \geq 0$ we can write the discriminator-augmented reward function as

$$\hat{r}_{\phi,\theta}(s) = [D_\theta(s) > 0.5] r_\phi(s) \ . \tag{10}$$

where $[\cdot]$ denotes an indicator function for the predicate it contains.

Following a similar line of reasoning as the previous section, we would like the discriminator to distinguish between states where r_ϕ is reliable (positive) and states where it is not (negative). As before, we have access to labels for the positive data, which in this case correspond to the states in the annotated data $\mathcal{D}_s$, for which we can verify that r_ϕ is accurate. Agent data again provides an unlabeled mixture of positive and negative since the agent is free to explore the state space and may wander far from the support of the reward model training set, and we do not have a way assess the reliability of unannotated states directly.

With positive data drawn from $\mathcal{D}_s$ and unlabeled data drawn from the agent's own experience $\mathcal{D}_\pi$ we can write a PU objective for $D_\theta(s)$ that corresponds exactly to Equation 8, except that the expert data $\mathcal{D}_{\pi_e}$ is replaced with the annotated data $\mathcal{D}_s$,

$$\tilde{R}^{pu}_{D_\theta}(\mathcal{D}_s, \mathcal{D}_\pi) = \eta \hat{R}^1_{D_\theta}(\mathcal{D}_s) + \max(-\beta, \hat{R}^0_{D_\theta}(\mathcal{D}_\pi) - \eta \hat{R}^0_{D_\theta}(\mathcal{D}_s)) \ . \tag{11}$$

We separately train the reward function $r_\phi(s)$ using supervised reward learning, and the discriminator D_θ using Equation 11, and combine them using Equation 10.

4.3 A Unified Reward Learning Algorithm

Having reduced both the adversarial imitation learning problem and the semi-supervised reward learning problem to PU learning, we present a unified algorithm for Positive Unlabeled Reward Learning (PURL) in Algorithm 1, which is a variant of the nnPU algorithm introduced in [17].

Algorithm 1 POLICY LEARNING WITH PURL

```
Inputs: Hyperparameters $\beta \geq 0$, $\eta \in [0, 1]$, policy $\pi$, replay buffer $\mathcal{D}_\pi$, positive data $\mathcal{D}_p$
for i = 1, 2, 3, ... do
    Sample minibatch $\mathbf{s}^p \sim \mathcal{D}_p$; $\mathbf{s}_t^\pi, \mathbf{a}_t^\pi, \mathbf{s}_{t+1}^\pi \sim D_\pi$
    if $\hat{R}^0_{D_\theta}(\mathbf{s}_t^\pi) - \eta \hat{R}^0_{D_\theta}(\mathbf{s}^p) \geq -\beta$ then
        Update $D_\theta$ using $\nabla_\theta \tilde{R}^{pu}_{D_\theta}(\mathbf{s}^p, \mathbf{s}_t^\pi)$
    else
        Update $D_\theta$ using $\nabla_\theta(\eta \hat{R}^0_{D_\theta}(\mathbf{s}^p) - \hat{R}^0_{D_\theta}(\mathbf{s}_t^\pi))$
    end if
    $\mathbf{r}_t \leftarrow \begin{cases} [D_\theta(\mathbf{s}_{t+1}) > 0.5] r_\phi(\mathbf{s}_{t+1}) & \text{if semi-supervised RL (Sec. 4.2)} \\ -\log(1 - D_\theta(\mathbf{s_{t+1}})) & \text{if adversarial imitation (Sec. 4.1)} \end{cases}$
    Update the policy $\pi$ with $\mathbf{s}_t^\pi, \mathbf{a}_t^\pi, \mathbf{r}_t, \mathbf{s}_{t+1}^\pi$ using D4PG [30].
end for
```

For policy training we use D4PG [30], which has been shown by [12] to be an effective policy learning method for GAIL. Several authors have noted that the data efficiency of GAIL is improved by using off-policy actor-critic learners [31, 11]. Using GAIL with an off-policy learning method strictly speaking requires importance sampling correction [32], but we did not find this to be necessary.

Reward learning in Algorithm 1 is agnostic to the policy learning component, and could even be combined with on-policy RL learning; however, it is convenient to combine PURL with off-policy RL, since PURL requires a replay buffer of agent experience which can be shared with the RL component.

Remark on η. We note that unlike the standard PU learning setup, the positive class prior η in our setting changes as policy learning progresses and the distribution of states in the replay buffer evolves. Specifically, η should increase as the policy improves. There have been many works that study the data distribution shift in the PU learning community [33, 34]. However, further developing and adapting these techniques to a large-scale PU learning algorithm embedded within an inverse reinforcement learning loop is beyond the scope of this work, and we consider it as an exciting future direction. We treat η as a fixed hyperparameter in this paper.

5 Experiments

We conduct experiments with both a standard benchmark task (*walker*) from the DM Control Suite [35] and two much more challenging table-top manipulation tasks (*lifting* and *stacking*) with pixel observation input. The manipulation environment consists of a Kinova Jaco arm, and four objects randomly initialized in a tray located in front of the arm. The *lifting* task is to control the arm to grasp and lift up the cyan banana-shaped object. The *stacking* task is to pick up the red cube and stack it on top of the blue cube. More experiment details are included in Appendix.

Expert demonstrations used for adversarial imitation are generated by an policy trained using D4PG [30]. We collect $N = 50$ expert trajectories for *walker* and *lifting*, and $N = 200$ for *stacking*.

Reward supervision. For supervised and semi-supervised reward learning, we obtain reward supervisions from ground truth reward in all three tasks. We collect training data by first training an expert policy and then collect trajectories from checkpoints of the policy at different stages of learning: from the initial random exploration to the final convergence. We collect $N = 50$ reward-annotated trajectories for *walker* and *lifting*, and $N = 200$ for *stacking* to reflect the difficulties of the tasks.

Data augmentation. We apply the data augmentation of Żołna et al. [13] to the inputs of all the discriminator models evaluated in the experiments. We find that data augmentation is in general effective in regularizing the discriminators and is necessary for the baseline GAIL agent to work on the more challenging manipulation tasks.

5.1 Evaluation on Adversarial Imitation Learning

To show the advantage of our PU and non-negative PU formulation of GAIL, we compare the following methods in an adversarial imitation learning setup: **GAIL (no-reg)**: The original GAIL implementation [1], which is to contrast our strong *GAIL* baseline with heavy regularization and data

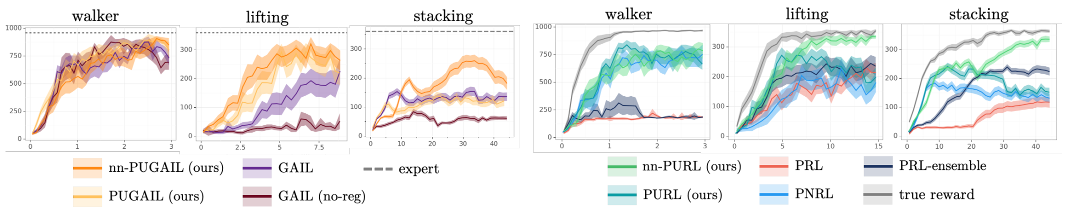

Figure 2: Results of adversarial imitation learning (left) and supervised reward learning (right) on the three evaluation tasks. Each curve is the mean of 5 trials with confidence interval of 95%, with episodic return in y-axis and million environment steps in x-axis.

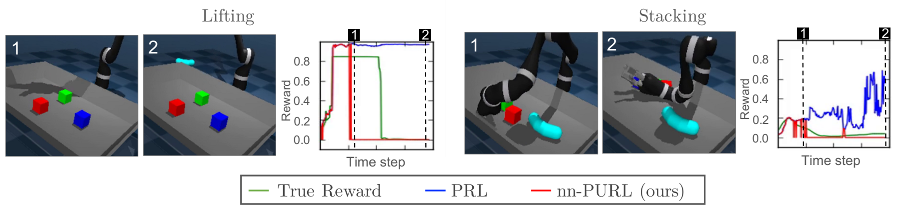

Figure 3: Qualitative result of preventing reward hacking. RL algorithm exploits various errors in the learned reward model such as moving out of the camera view (lifting) and occluding camera view (stacking). We show that supervised reward learning method (blue curve) suffers from reward hacking while our method (red curve) learns to suppress unreliable reward predictions.

augmentation. **GAIL**: We apply extensive data augmentation techniques such as image flipping and random cropping to the discriminator input. We show that heavy regularization is crucial for GAIL to work on the challenging manipulation domain. **PUGAIL**: A PU formulation of GAIL introduced in Section 4.1. **nn-PUGAIL**: Our final non-negative PU formulation of GAIL introduced in Section 4.1.

As shown in Figure 2, all methods achieve near-expert performance in the standard benchmark task *walker*. In the more challenging manipulation task *lifting*, we note that the vanilla GAIL implementation *GAIL (no-reg)* experiences severe overfitting from the beginning, which prevents the policy from learning. This agrees with our hypothesis that the overfitting issue is more pronounced in complex domain with larger state space. Our *nn-PUGAIL* not only outperforms *GAIL* at peak performance and achieve near-expert performance but also learns much faster than *GAIL*. We observe similar trends in the most challenging *stacking* task, where the discriminator of *GAIL* starts to overfit at 10 million environment steps. Although our *nn-PUGAIL* does not achieve the optimal performance, it outperforms *GAIL* by a wide margin at the peak performance. See Appendix for additional experiment on the effect of positive class prior η.

Remark on asymptotic performance. Although our *nn-PUGAIL* compares favorably to the baselines in peak performances, it does not entirely eliminate overfitting, and its performances tend to plataeu and decline eventually in the manipulation tasks. Our hypothesis is that while the original nn-PU work [17] shows that the non-negative constraint with the correct class prior effectively eliminates overfitting when learning from a static dataset, our setup is a lot more complex, with many components interacting and learning from a constantly changing dataset. As noted in the method section, selecting η adaptively is still an open challenge and will be the focus of our future work

5.2 Evaluation on Supervised Reward Learning

We compare the following supervised reward learning methods. **PRL**: A vanilla supervised reward learning baseline. **PRL-ensemble**: A strong supervised reward learning baseline, where the reward prediction is taken as the min of the predictions from a set of supervised reward models trained from different initialization. This method is shown to effectively alleviate the reward hacking problem by Vecerik et al. [28]. **PNRL**: A supervised reward model combined with a positive-negative discriminator. **PURL**: Supervised reward model with a positive-unlabeled discriminator. **nn-PURL**: Our final model, where the discriminator is trained with a non-negative PU formulation (Section 4.2).

Figure 2 (right side) shows the results. In the *walker* task, all discriminator-augmented reward learning methods are able to achieve competitive performance. In contrast, *PRL* and *PRL-ensemble*

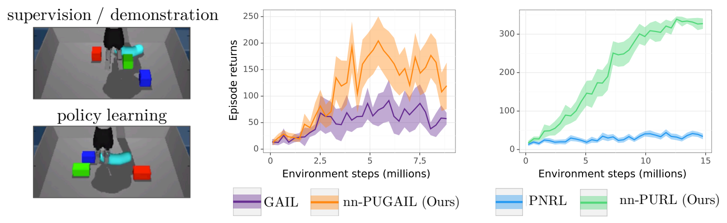

Figure 4: Learning with domain gaps in the *lifting* task. **Left:** We create domain gaps between the environment for collecting demonstrations and reward supervisions (top, distractors are cubes) and the environment for policy learning (bottom, distractors are rectangles). The differences are subtle to the human eye, but extremely salient to the discriminators. **Right:** Results on adversarial imitation learning and supervised reward learning with domain gaps.

both suffer from reward delusions. The performance of *PRL* plateaus soon after the beginning of the training. While *PRL-ensemble* outperforms *PRL* initially thanks to the ensemble strategy, the policy soon deteriorates because none of the ensemble models can make reliable predictions.

In the more challenging *lifting* tasks, all baseline methods but our *nn-PURL* have similar performances, whereas policies trained with *nn-PURL* are able to achieve comparable performance to policies trained with ground truth reward. We observe that the performance of *PNRL* gradually decreases after the initial improvement. This is due to that the discriminator with a positive-negative objective overfits to the training positive samples. This effect is more pronounced in the most challenging *stacking* task, where both *PNRL* and *nn-PURL* have similar progress at the beginning of the learning, but soon *PNRL* plateaus and starts to deteriorate. In contrast, our *nn-PURL* is able to converge to optimal performance. See Appendix for experiment on the effect of positive class prior η in this setting.

Reward delusions. We provide some qualitative examples of reward delusions and illustrate that our *nn-PURL* addresses the problem. In Figure 7, we visualize the behaviors of agents trained with the baseline supervised reward learning method (*PRL*) and compare the reward predictions from *PRL* and *nn-PURL*. In *lifting*, the agent learns to exploit the error where *PRL* predicts high reward even after the agent has dropped the object (Frame 1) because the gripper is out of the camera view. *nn-PURL* learns to suppress reward as soon as the gripper moves out of the camera view. In *stacking*, the agent exploits the reward model error by occluding all objects from the camera. Our *nn-PURL* is able to detect the out-of-distribution state starting from Frame 1. See Appendix for more examples.

5.3 Learning with Domain Gaps

Finally, we test the limit of our method's ability to mitigate the discriminator overfitting problem by introducing *domain gaps* between the environment for generating expert demonstrations and reward supervision and the environment for policy learning. As shown on the left side of Figure 4, we vary the shape of the distractor objects in a *lifting* task such that the discriminator should be able to trivially distinguish the training data from the policy rollout data. The results are shown on the right side of Figure 4. We find that our nnPU-based methods outperform their PN-learning counterparts in both the supervised reward learning and the adversarial reward learning settings. In particular, *nn-PURL* is able to maintain the near-optimal performance despite the domain gap.

6 Conclusion

We presented PURL, a framework for reward learning that allows us to cast both (1) GAIL discriminator training and (2) support set estimation in supervised reward learning as a positive-unlabeled learning problem. In GAIL in particular, this formulation changes the semantics of the discriminator from distinguishing between expert and agent to distinguishing between success and failure of the demonstrated task. By applying a recent large scale PU learning algorithm to the PURL objective for discriminator training, we obtained large improvements in performance of agents, without introducing any additional assumptions. We empirically demonstrated how PURL addresses both the overfitting problem of GAIL, and the underfitting problem of supervised reward learning. We also showed how PURL allows us to train reward models under domain gap between reward model and policy learning.

Acknowledgments

We would like to thank the scientific python community for developing the core set of tools that enabled this work, including Tensorflow [36], Numpy [37], Pandas [38], and Matplotlib [39].

References

[1] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[2] S. Cabi, S. G. Colmenarejo, A. Novikov, K. Konyushkova, S. Reed, R. Jeong, K. Żołna, 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. *Robotics: Science and Systems*, 2020.

[3] A. Wilson, A. Fern, and P. Tadepalli. A bayesian approach for policy learning from trajectory preference queries. In *Advances in Neural Information Processing Systems*, pages 1133–1141, 2012.

[4] 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*, pages 8011–8023, 2018.

[5] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[6] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307, 2017.

[7] A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, volume 1, page 2, 2000.

[8] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum Entropy Inverse Reinforcement Learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 2008.

[9] J. Fu, K. Luo, and S. Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

[10] X. B. Peng, A. Kanazawa, S. Toyer, P. Abbeel, and S. Levine. Variational discriminator bottleneck: Improving imitation learning, inverse RL, and GANs by constraining information flow. *arXiv preprint arXiv:1810.00821*, 2018.

[11] L. Blondé and A. Kalousis. Sample-efficient imitation learning via generative adversarial nets. *arXiv preprint arXiv:1809.02064*, 2018.

[12] S. Reed, Y. Aytar, Z. Wang, T. Paine, A. van den Oord, T. Pfaff, S. Gomez, A. Novikov, D. Budden, and O. Vinyals. Visual imitation with a minimal adversary. Technical report, Deepmind, 2018.

[13] K. Żołna, S. Reed, A. Novikov, S. G. Colmenarej, D. Budden, S. Cabi, M. Denil, N. de Freitas, and Z. Wang. Task-Relevant Adversarial Imitation Learning. *arXiv:1910.01077 [cs, stat]*, 2019.

[14] F. Denis. PAC Learning from Positive Statistical Queries. In *Algorithmic Learning Theory*, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. doi:10.1007/3-540-49730-7_9.

[15] C. Elkan and K. Noto. Learning classifiers from only positive and unlabeled data. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 08*, Las Vegas, Nevada, USA, 2008. ACM Press. doi:10.1145/1401890.1401920.

[16] M. C. du Plessis, G. Niu, and M. Sugiyama. Analysis of Learning from Positive and Unlabeled Data. In *Neural Information Processing Systems*, 2014.

[17] R. Kiryo, G. Niu, M. C. du Plessis, and M. Sugiyama. Positive-unlabeled learning with non-negative risk estimator. In *Advances in Neural Information Processing Systems*, pages 1675–1685, 2017.

[18] Y. Li, J. Song, and S. Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In *Advances in Neural Information Processing Systems*, pages 3812–3822, 2017.

[19] N. Baram, O. Anschel, I. Caspi, and S. Mannor. End-to-end differentiable adversarial imitation learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 390–399. JMLR. org, 2017.

[20] D. Bahdanau, F. Hill, J. Leike, E. Hughes, A. Hosseini, P. Kohli, and E. Grefenstette. Learning to understand goal specifications by modelling reward. *ICLR*, 2019.

[21] R. Akrour, M. Schoenauer, and M. Sebag. Preference-based policy learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 12–27. Springer, 2011.

[22] T. Everitt and M. Hutter. Avoiding wireheading with value reinforcement learning. In *International Conference on Artificial General Intelligence*, pages 12–22. Springer, 2016.

[23] M. C. du Plessis, G. Niu, and M. Sugiyama. Convex Formulation for Learning from Positive and Unlabeled Data. In *Proceedings of the International Conference on Machine Learning*, 2015.

[24] G. Patrini, F. Nielsen, R. Nock, and M. Carioni. Loss factorization, weakly supervised learning and label noise robustness. In *International conference on machine learning*, pages 708–717, 2016.

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[27] K. Żołna, N. Rostamzadeh, Y. Bengio, S. Ahn, and P. O. Pinheiro. Reinforced imitation in heterogeneous action space. *arXiv preprint arXiv:1904.03438*, 2019.

[28] M. Vecerik, O. Sushkov, D. Barker, T. Rothörl, T. Hester, and J. Scholz. A practical approach to insertion with variable socket position using deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 754–760. IEEE, 2019.

[29] G. Niu, M. C. du Plessis, T. Sakai, Y. Ma, and M. Sugiyama. Theoretical comparisons of positive-unlabeled learning against positive-negative learning. In *Advances in Neural Information Processing Systems*, pages 1199–1207, 2016.

[30] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. TB, A. Muldal, N. Heess, and T. Lillicrap. Distributed Distributional Deterministic Policy Gradients. *arXiv:1804.08617 [cs, stat]*, 2018.

[31] F. Sasaki, T. Yohira, and A. Kawaguchi. Sample Efficient Imitation Learning for Continuous Control. In *Proceedings of the International Conference on Learning Representations*, 2018.

[32] I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson. Discriminator-Actor-Critic: Addressing Sample Inefficiency and Reward Bias in Adversarial Imitation Learning. *arXiv:1809.02925 [cs, stat]*, 2018.

[33] Y.-G. Hsieh, G. Niu, and M. Sugiyama. Classification from positive, unlabeled and biased negative data. In *International Conference on Machine Learning*, pages 2820–2829, 2019.

[34] T. Sakai and N. Shimizu. Covariate shift adaptation on learning from positive and unlabeled data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4838–4845, 07 2019. doi:10.1609/aaai.v33i01.33014838.

[35] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. Lillicrap, and M. Riedmiller. DeepMind Control Suite. *arXiv:1801.00690 [cs]*, 2018.

[36] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[37] T. Oliphant. *Guide to NumPy*. USA: Trelgol Publishing, 2006.

[38] W. McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56, 2010.

[39] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90, 2007.

[40] 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*, pages 5026–5033. IEEE, 2012.

[41] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014.

[42] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *The International Conference on Learning Representations*, 2015.

[43] M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 449–458. JMLR. org, 2017.

[44] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

7 Appendix

7.1 Additional Experiment for Adversarial Imitation Learning

Effect of the positive class prior η. As aforementioned, the positive class prior η is determined via hyperparameter search. Here we study how different η affect the discriminator behavior of *nn-PUGAIL* and the resulting policies in the *lifting* task. We analyze the discriminator performance with respect to two fixed datasets with known ground truth classes: (1) a set of holdout expert demonstrations (*expert*) and (2) a set of holdout failure trajectories (*failure*). Specifically, we report the average sigmoid (logistic) values predicted by the discriminator for the respective class of the two data sources. An ideal discriminator should classify both datasets correctly, i.e., the sigmoid values should be greater than 0.5 for both datasets. Because the discriminator behavior evolves with the progress of the policy learning, the results are reported at the policy convergence.

Figure 5(a) shows the results. We observe that low positive class prior η causes the discriminator to bias towards predicting all trajectories to be failure. Conversely, high η results in optimistic discriminators where all trajectories are predicted to be successes. At $\eta = 0.5$, the discriminator achieves optimal performance where both datasets are classified correctly. This is reflected in Figure 5(b), where the best policy performance is achieved at $\eta = 0.5$. We note that the optimal η value depends on both the number of expert demonstrations and the task. We defer a principled method for automatically selecting η to future works.

Overfitting discriminator. To better understand the overfitting discriminator problem and the advantage of the PU formulation over PN, we compare the discriminator performance of *GAIL* and *nnPU-GAIL* with a fixed $\eta = 0.5$ over the course of learning a *lifting* task. Specifically, we analyze the probability of states being classified as *success* (*PoS*) of four data sources: (1) the training expert demonstrations, (2) a set of holdout expert demonstrations, (3) a set of holdout failure trajectories, and (4) policy rollouts sampled from the replay buffer. Note that all data sources but (4) are fixed throughout the training.

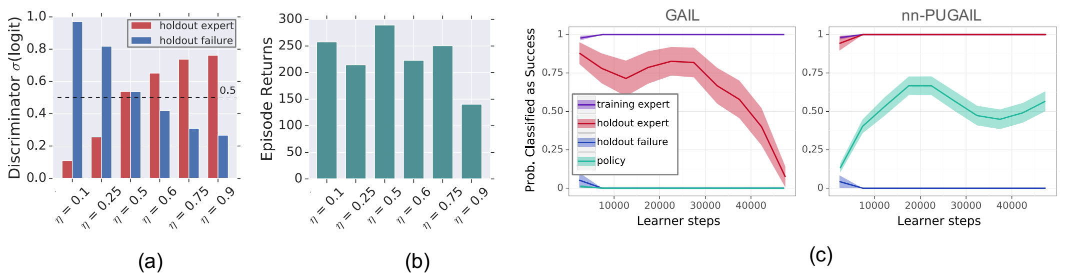

Figure 5: Ablation study of adversarial imitation learning on the *lifting* task: Effect of choosing different positive class prior η values on the (a) discriminator performance and (b) policy performance for *nn-PUGAIL*. (c) compares the discriminator behavior of *GAIL* and *nn-PUGAIL* over the course of the policy learning. Each curve is the mean of 5 trials with confidence interval of 95%.

As shown in Figure 5(c), both methods classify the holdout failure trajectories correctly (near-0% *PoS*). The discriminator of *GAIL* classifies the training demonstrations correctly as *success* throughout the training, while the *PoS* of holdout expert demonstrations decreases as the learning progresses. This indicates that the discriminator of GAIL overfits to the training expert demonstrations. In contrast, the discriminator of *nnPU-GAIL* maintains a 100% *PoS* for both the training and the holdout expert demonstrations. In addition, *nnPU-GAIL* classifies increasingly more policy rollout as *success* as the policy improves, and the *PoS* correctly approximates the positive class prior at the policy convergence.

7.2 Additional Results for Supervised Reward Learning

Effect of the positive class prior η. Here, we study how η affects the performance of *nn-PURL* in the *lifting* task. Figure 6 shows both the episodic reward prediction and the final policy performance. We observe that while *PNRL* and *PRL* have similar performances, the *PNRL* suffers high negative reward error while *PRL* has high positive reward error. This agrees with our hypothesis that (1) learning with only positive data results in reward delusions (high pseudo-reward) and (2) unregularized semi-supervised reward learning leads to overfitting (low pseudo-reward). On the other hand, our *nn-PURL* mitigates the two extremes through the positive class prior η and enables the policy to achieve both the lowest reward prediction errors and the best performance at $\eta = 0.5$. Again, we note that the optimal choice of η is task and data-dependent, and a principled selection method is deferred to future works.

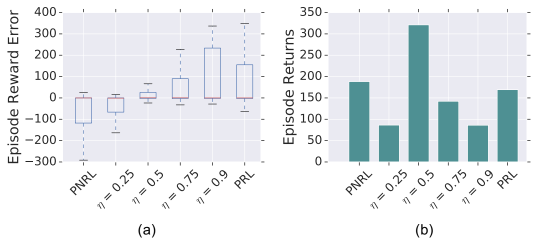

Figure 6: Effect of positive class prior η values on the (a) reward prediction errors and (b) policy performance in *nn-PURL*. Results of *PNRL* and *PRL* are included for reference.

Reward delusion example details. Figure 7 is a more detailed view of the reward delusion examples in the main text. We visualize the behaviors of agents trained with the baseline supervised reward learning method (*PRL*) and compare the reward predictions from *PRL* and *nn-PURL*. In *walker*, *PRL* incorrectly predicts high reward when the agent poses as walking but is in fact not moving, whereas *nn-PURL* predicts zero reward after the agent has stopped moving (Frame 3). In *lifting*, the agent learns to exploit the error where *PRL* predicts high reward even after the agent has dropped the object (Frame 3) because the gripper is out of the camera view. *nn-PURL* learns to suppress reward as soon as the gripper moves out of the camera view (Frame 2). In *stacking*, the agent exploits the reward model error by occluding all objects from the camera (Frame 3 and 4). Our *nn-PURL* is able to detect the out-of-distribution state starting from Frame 2.

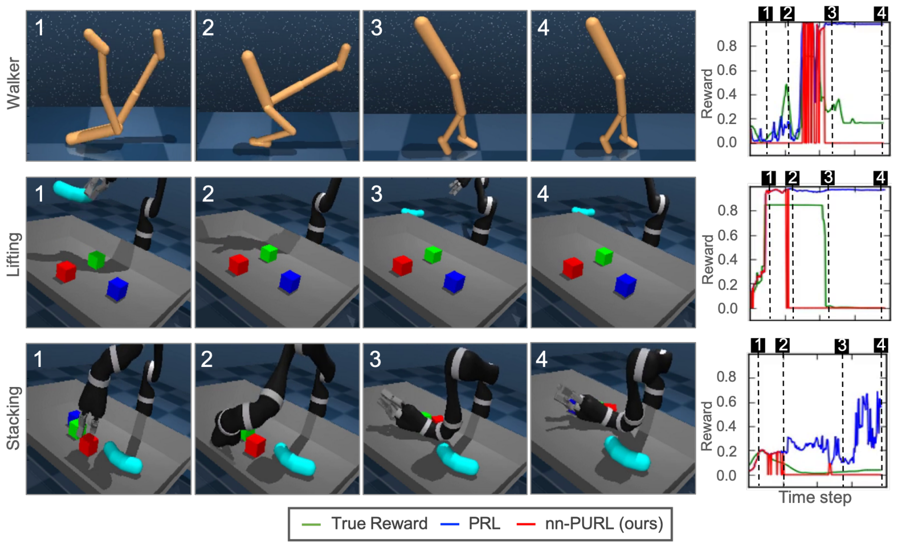

Figure 7: Qualitative result of preventing reward hacking. RL algorithm exploits various errors in the learned reward model such as moving out of the camera view (lifting) and occluding camera view (stacking). We show that supervised reward learning method (blue curve) suffers from reward hacking while our method (red curve) learns to suppress unreliable reward predictions.

7.3 Environment Details and Training Data

The evaluation environments are physically simulated using the Mujoco simulator [40].

Walker The *walker* domain is part of the DeepMind Control Suite [35], where a bipedal agent is tasked to either stand up, walk forward, or run forward. The bipedal agent can only move in a horizontal 2D space (no lateral motion). We used the *walker walk* task in the environment, where the agent is rewarded for moving forward (to the right relative to the camera viewpoint). The task horizon for both the training and evaluation is 1000 time steps.

Robotic Manipulation The manipulation environment consists of a Kinova Jaco arm, a tray set on top of a table, and four objects randomly initialized in the tray. The objects are three 5cm^3 colored blocks and a banana.

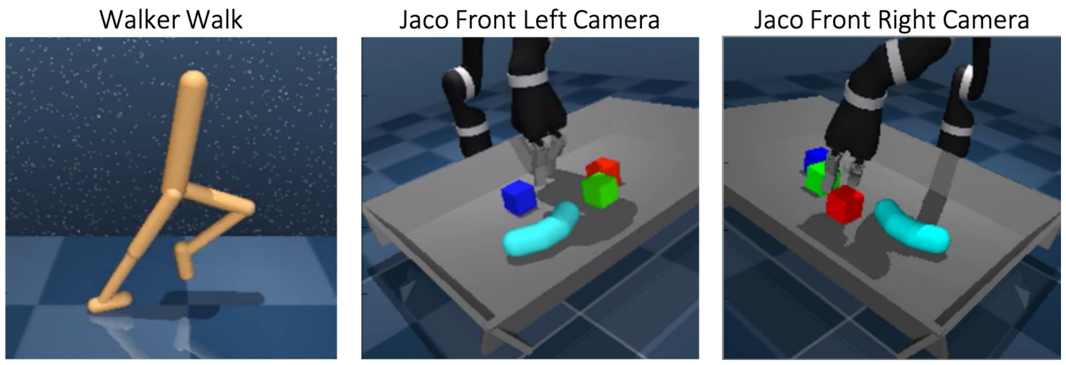

Figure 8: Camera view of the Walker environment and the Jaco environment.

The *lifting* task is to lift the banana. The shaped reward is defined on both the distance between the end-effector and the object and the height of the object above the tray. The *stacking* task is to stack the red block on top of the blue block. The shaped reward is defined on the relative position between the two cubes. Both tasks share the same set of objects, where non-target objects act as distractors. Both tasks have shaped rewards ranging [0, 1] for reward supervision in the supervised reward learning setting.

We use joint velocity control (9DOF) where we control all 6 joints of arm and all 3 joints of the hand. The simulation is run with a numerical time step of 10 milliseconds, integrating 5 steps, to get a control frequency of 20HZ. Both training and evaluation episodes are of 400 time steps long. We use two cameras to capture the scene: a front left camera and a front right camera. The viewpoints are visualized in Figure 8. Both cameras are rendered to RGB images of size 64×64. As described in the main paper, all reward models and discriminator models take only images as input, and the policies take low-dimensional environment state as inputs. For a full list of observations and environment states, see Table 1.

Domain gap Section 5.3 evaluates various methods under visual *domain gaps* between the environment for generating the reward learning training data (i.e., expert demonstrations and reward supervision) and the environment for policy training in a *lifting* task. The domain gaps are created by changing the shapes of the distractor objects from 5cm^3 cuboids in the data generation environment to $4 \times 4 \times 8$cm cuboids in the policy learning environment, as shown in Figure 4.

Table 1: Observation and environment state dimensions in the Jaco environment.

Observation / state name	Dimensions
front left camera	$64 \times 64 \times 3$
front right camera	$64 \times 64 \times 3$
base force and torque sensors	6
arm joints position	6
arm joints velocity	6
wrist force and torque sensors	6
hand finger joints position	3
hand finger joints velocity	3
hand fingertip sensors	3
grip site position	3
pinch site position	3

7.4 Reward Learning Details

Non-negative PU Reward Learning Algorithm. We introduced the unified PU Reward Learning algorithm in Section 4.3. We base the sub-routine for enforcing non-negative constraint on the algorithm first introduced in [17]. Below we present the hyperparameters used in each of the evaluation tasks (Table 2).

Reward learning model architecture Both the supervised reward models and the discriminator models share the same architecture. Table 3 lists model architecture details.

Data augmentation We observed in Section 5 that data augmentation is crucial for regularizing the discriminators. In addition, we empirically found that applying data augmentation to the input of the supervised reward models also improves their performance. We apply the same set of data augmentation operations on the input images for both the discriminator model and the supervised reward model. Table 3 lists the augmentation parameters.

7.5 D4PG Details

We use the Distributed Distributional Deterministic Policy Gradients (D4PG) [30] as our policy learning algorithm framework. D4PG is a distributed off-policy reinforcement learning algorithm designed specifically for continuous control problems. In short, D4PG extends the Q-learning formulation of the Deterministic Policy Gradients [41, 42] to distributional value function [43]. Other features of D4PG include target network for training stability, distributed training [44], and multi-step returns.

Table 4 lists the parameters for D4PG used in our experiments and the network architecture. Again, we note that to focus on comparing the reward learning methods and isolate the effect of the policy learning algorithm, we allow the policy to take environment state as input to achieve faster and more stable policy training. All experiments share the same policy learning setup.

Table 2: Hyperparameters and dataset size for reward learning.

Task	Method	Hyperparameters	Training data size
Walker	nn-PUGAIL	$\beta = 0.0, \eta = 0.25$	50
	nn-PURL	$\beta = 0.0, \eta = 0.5$	50
Lifting	nn-PUGAIL	$\beta = 0.0, \eta = 0.5$	50
	nn-PURL	$\beta = 0.0, \eta = 0.5$	50
Stacking	nn-PUGAIL	$\beta = 0.0, \eta = 0.7$	200
	nn-PURL	$\beta = 0.7, \eta = 0.7$	200

Table 3: Data augmentation specifications and the model architectures of the supervised reward models and the discriminators.

Discriminator / Reward Network	Specifications
Residual Conv Blocks	[2, 2, 2]
Conv Channels	[16, 32, 32]
Conv Kernels Sizes	[(3, 3), (3, 3), (3, 3)]
Pooling	MaxPooling = [(2, 2), (2, 2), (2, 2)]
Activation	ReLU
Supervised Reward Loss	Mean-Squared Error
Optimizer	Adam [25]
Learning Rate (Supervised)	0.0001
Learning Rate (Discriminator)	0.00001
Data Augmentation	**Specifications**
Dropout	ProbKeep=0.5
Random flip	Horizontal
Random crop ratio	0.8
Random satuation range	[0.5, 2.0]
Random hue	max_delta=0.05
Random contrast range	[0.5, 2.0]
Clipped pixel noise	[-16, 16]

Table 4: Details of the D4PG algorithm.

D4PG Parameters	Values
V_{min}	0
V_{max}	100
V_{bins}	51
N step (return)	5
Actor learning rate	0.0001
Critic learning rate	0.0001
Optimizer	Adam [25]
Batch size	256
Discount factor	0.99
Number of actors	16 (*walker*), 128 (*lifting, stacking*)
Network Specifications	**Values**
Actor network	MLP=[300, 200]
Critic network	MLP=[400, 300]
Activation function	ReLU