

Adversarial Active Exploration for Inverse Dynamics Model Learning

Zhang-Wei Hong Tsu-Jui Fu Tzu-Yun Shann Yi-Hsiang Chang
Chun-Yi Lee

{williamd4112, rayfu1996ozig, arielshann, shawn420, cylee}@gapp.nthu.edu.tw
Elsa Lab, Department of Computer Science
National Tsing Hua University, Hsinchu, Taiwan

Abstract: We present an adversarial active exploration for inverse dynamics model learning, a simple yet effective learning scheme that incentivizes exploration in an environment without any human intervention. Our framework consists of a deep reinforcement learning (DRL) agent and an inverse dynamics model contesting with each other. The former collects training samples for the latter, with an objective to maximize the error of the latter. The latter is trained with samples collected by the former, and generates rewards for the former when it fails to predict the actual action taken by the former. In such a competitive setting, the DRL agent learns to generate samples that the inverse dynamics model fails to predict correctly, while the inverse dynamics model learns to adapt to the challenging samples. We further propose a reward structure that ensures the DRL agent to collect only moderately hard samples but not overly hard ones that prevent the inverse model from predicting effectively. We evaluate the effectiveness of our method on several robotic arm and hand manipulation tasks against multiple baseline models. Experimental results show that our method is comparable to those directly trained with expert demonstrations, and superior to the other baselines even without any human priors.

1 Introduction

Over the past decade, inverse dynamics models have shown considerable successes in robotic control [1, 2, 3, 4] and even in humanoid robots [5]. The main objective of an inverse dynamics model is to predict the control action (e.g., torque) between two states. With such a model, it becomes practically feasible to carry out complex control behaviors through inferring a series of control actions given a pre-defined trajectory of states. An inverse dynamics model is usually trained from streaming of data (i.e., states and actions) collected at online execution time, as it is extremely difficult to analytically specify the inverse dynamics model for a complicated robot. Traditionally, a branch of research directions attempt to employ gaussian process (GP) to approximate the inverse dynamics model. However, the computational complexity of such methods is intractable in high-dimensional state space, thus prohibiting their further usage from more sophisticated robotic control applications.

In the light of the above constraint, researchers in recent years turn into developing inverse dynamics models based on deep neural networks (DNNs) in the hope to effectively cope with high-dimensional state space. Rueckert et al. [3] trains an inverse dynamics model using recurrent neural networks (RNNs), and demonstrates better prediction accuracy and computational efficiency than GP-based methods. Pathak et al. [6], Nair et al. [7], and Agrawal et al. [8] further show that DNNs are able to learn an image-based inverse dynamics model for robotic arm manipulation tasks. In spite of their promising results on challenging tasks, however, this line of works demand tremendous amount of data for training DNNs, posing a considerable challenge on data-efficiency.

As data efficiency is crucial for robot learning in practical applications, an efficient data acquisition methodology is critically essential for inverse dynamics model learning. Training data for an inverse dynamics model typically come from interactions with an environment. Previous approaches, however, usually perform such interactions with the environments in inefficient manners. For instance, Agrawal et al. [8] and Nair et al. [7] employ an agent to take random actions in an environment to collect training data for their inverse dynamics models. Random actions, nevertheless, are less likely to result in effective exploration behaviors, and may thus lead to a lack of comprehensiveness in the data samples collected by the agent. Nair et al. [7] attempts to deal with the above problem by adding human bias in its data sampling process, which is specifically tailored to certain pre-defined robot configurations. Despite their promising results, tailored sampling ranges require significant amount

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

of human effort. A more general exploration strategy called curiosity-driven exploration was later proposed in [9]. While this approach is effective in exploring novel states in an environment, their exploration behavior is solely driven by the prediction errors of the forward dynamics models. Thus, the approach is unsuitable and not specifically tailored for inverse dynamics model learning.

In order to deal with the above issues, in this paper we propose a straightforward and efficient active data acquisition method, called **adversarial active exploration**, which motivates exploration of an environment in a self-supervised manner (i.e., without human intervention). Inspired by Pinto et al. [10], Shioya et al. [11], Sukhbaatar et al. [12], we implement the proposed method by jointly training a deep reinforcement learning (DRL) agent and an inverse dynamics model competing with each other. The former explores the environment to collect training data for the latter, and receives rewards from the latter if the data samples are considered difficult. The latter is trained with the data collected by the former, and only generates rewards when it fails to predict the true actions performed by the former. In such an adversarial setting, the DRL agent is rewarded only for the failure of the inverse dynamics model. Therefore, the DRL agent learns to sample hard examples to maximize the chances to fail the inverse dynamics model. On the other hand, the inverse dynamics model learns to be robust to the hard examples collected by the DRL agent by minimizing the probability of failures. As a result, as the inverse dynamics model becomes stronger, the DRL agent is also incentivized to search for harder examples to obtain rewards. Overly hard examples, however, may lead to biased exploration and cause instability of the learning process. In order to stabilize the learning curve of the inverse dynamics model, we further propose a reward structure such that the DRL agent are encouraged to explore moderately hard examples for the inverse dynamics model, but refraining from too difficult ones for the latter to learn. The self-regulating feedback structure between the DRL agent and the inverse dynamics model enables them to automatically construct a curriculum for exploration.

We perform extensive experiments to validate our method on multiple robotic arm and hand manipulation task environments simulated by the MuJoCo physics engine [13]. These environments are intentionally selected by us for evaluating the performance of inverse dynamics model, as each of them allows only a very limited set of chained actions to transition the robotic arms and hands to target states. We examine the effectiveness of our method by comparing it against a number of data collection schemes. The experimental results show that our method is more effective and data-efficient than the other schemes, as well as in environments with high-dimensional action spaces. We also demonstrate that in most of the cases the performance of the inverse dynamics model trained by our method is comparable to that directly trained with expert trajectories. The above observations suggest that our method is superior to the other data collection schemes even in the absence of human priors. To justify each of our design decisions, we provide a comprehensive set of ablative analysis and discuss their implications. The contributions of this work are summarized as the following:

- We introduce an adversarial active exploration method, which leverages a DRL agent to actively collect informative data samples for training an inverse dynamics model effectively.
- We employ a competitive scheme for the DRL agent and the inverse dynamics model, enabling them to automatically construct a curriculum for efficient data acquisition.
- We introduce a reward structure for the proposed scheme to stabilize the training process.
- We demonstrate the effectiveness of the proposed method and compare it with a number of baseline data acquisition approaches for multiple robotic arm and hand manipulation tasks.
- We validate that our method is generalizable to tasks with high-dimensional action spaces.

The remainder of this paper is organized as follows. Section 2 discusses the related works. Section 3 introduces background material necessary for understanding the contents of this paper. Section 4 describes the proposed adversarial active exploration method in detail. Section 5 reports the experimental results, and provides an in-depth ablative analysis of our method. Section 6 concludes.

2 Related Works

This section reviews the related works of the areas: (i) active learning and (ii) exploration in DRL.

Active learning [14] is an efficient way to query human for ground truth data labels when a large amount of unlabeled data are accessible while human labelling is expensive. Traditional works search for samples most uncertain for the model [15, 16] or data that has the highest disagreement among an ensemble of models [17] to query. A recent work [18] further trains an DRL [19] agent to select informative training samples for a model, where unlabeled data are sequentially presented to the

agent. The method proposed in this paper differs from the above techniques in that it actively searches and collects data samples from an environment, rather than discriminate samples from a static dataset.

Exploration in DRL refers to the process of searching for a better control policy to solve tasks via interactions with the environment. Apart from the naive random exploration approaches [20, 21], a plethora of advanced exploration techniques [9, 22, 23, 24, 25, 26] has been proposed in the past few years to enhance the performance of DRL policies. The purpose of these works differs from ours in that our method aims to collect beneficial samples for learning an inverse dynamics model rather than a policy. In addition, some of these approaches are unable to be applied to our problem setting due to the absence of the goal space (i.e., representation of the task objectives) assumption in inverse dynamics model learning [24, 25, 26]. A curiosity-driven DRL exploration technique which maximizes the prediction errors of the forward dynamics model [9] has recently been adapted to inverse dynamics model learning [6]. This approach encourages a DRL agent to visit novel states that a forward dynamics model is unable to accurately predict. However, novel states for a forward model do not necessarily reflect the novel states required for training an inverse dynamics model. A comparison of our method and a data acquisition approach based on [9] is presented in Section 5.

3 Background

3.1 Deep Reinforcement Learning

DRL methods train an agent to interact with an environment \mathcal{E} . At each timestep t , the agent receives an state $x_t \in \mathcal{X}$, where \mathcal{X} is the state space of \mathcal{E} . It then takes an action a_t from the action space \mathcal{A} based on its current policy π , receives a reward $r_t = r(x_t, a_t)$ (where r is a reward function encoded with specific task objectives), and transitions to the next state x' . The policy $\pi : \mathcal{X} \times \mathcal{A} \mapsto [0, 1]$ is a conditional probability distribution parameterized by a deep neural network with parameters θ . The goal of the agent is to learn a π that maximizes the discounted sum of rewards $G_t = \sum_{\tau=t}^T \gamma^{\tau-t} r(x_\tau, a_\tau)$, where t is the current timestep, $\gamma \in (0, 1]$ the discount factor, and T the horizon. Policy gradient methods [20, 27] are a class of DRL techniques that directly optimize θ in the direction of $\nabla_{\theta} \mathbb{E}_{a_t \sim \pi} [\log \pi(a_t | x_t, \theta) G_t]$. In this paper, we employ a more recent family of policy gradient methods, called proximal policy optimization (PPO) [28]. PPO is superior to the vanilla policy gradient methods [27] in terms of performance and sample efficiency.

3.2 Inverse Dynamics Model for Robotic Control

An inverse dynamics model I takes as input a pair of states (x, x') , and predicts the action \hat{a} required to reach the next state x' from the current state x . In this paper, we focus on parametric inverse dynamics models which can be formally express as: $\hat{a} = I(x, x' | \theta_I)$, where (x, x') can come from sensor measurements or a robot’s states in the configuration space, and θ_I represents the trainable parameters of I . During the training phase, θ_I is iteratively updated to minimize the loss function L_I . The optimal θ_I is represented as: $\theta_I^* = \arg \min_{\theta_I} L_I(a, \hat{a} | \theta_I) \forall x, x' \in \mathcal{X}, a \in \mathcal{A}$, where a is the ground truth action. Once a sufficiently acceptable θ_I^* is obtained, it can then be used to derive the control action sequence $\{\hat{a}_0, \hat{a}_1, \dots, \hat{a}_{T-1}\}$ in a closed-loop manner, and therefore realize an arbitrarily given trajectory $\{\hat{x}_1, \dots, \hat{x}_T\}$. The derivation procedure is formulated as: $\hat{a}_t = I(x_t, \hat{x}_{t+1} | \theta_I) \forall t$, where x_t and \hat{x}_t denote the actual and the desired states, respectively.

4 Methodology

In this section, we first describe the proposed adversarial active exploration. Then, we explain the training methodology in detail. Finally, we discuss a technique for stabilizing the training process.

4.1 Adversarial Active Exploration

Fig. 1 shows a framework that illustrates the proposed adversarial active exploration, which includes a DRL agent P and an inverse dynamics model I . Assume that $\Phi_{\pi} : \{x_0, a_0, x_1, a_1, \dots, x_T\}$ is the sequence of states and actions generated by P as it explores \mathcal{E} using a policy π . At each timestep t , P collects a 3-tuple training sample (x_t, a_t, x_{t+1}) for I , while I predicts an action \hat{a}_t and generates a reward r_t for P . I is modified to recurrently encode the information of its past states by including an additional hidden vector h_t , as suggested in [3, 6]. The inverse dynamics model I is thus given by:

$$\begin{aligned} \hat{a}_t &= I(x_t, x_{t+1} | h_t, \theta_I) \\ h_t &= f(h_{t-1}, x_t), \end{aligned} \tag{1}$$

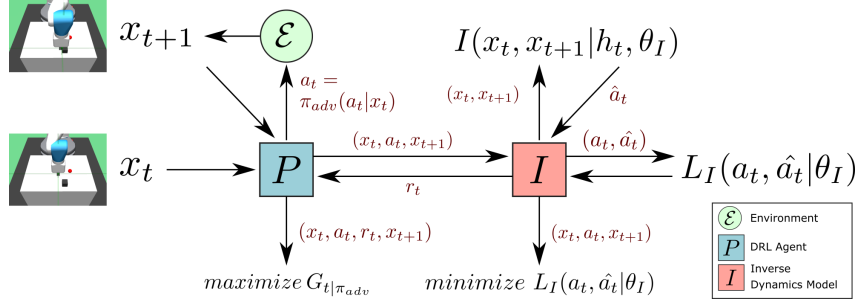


Figure 1: Framework of adversarial active exploration.

where $f(\cdot)$ denotes the recurrent function. Similar to [3, 6], in this work we approximate θ_I in an offline batch training manner. θ_I is iteratively updated to minimize L_I , formulated as the following:

$$\underset{\theta_I}{\text{minimize}} \mathbb{E}_{(x_t, a_t, x_{t+1}) \sim U(Z_I)} [L_I(a_t, \hat{a}_t | \theta_I)], \quad (2)$$

where $U(Z_I)$ is an uniform distribution over the buffer Z_I storing the 3-tuple data samples collected by P . We employ $\beta \|a_t - \hat{a}_t\|^2$ (where β is a scaling factor) as the loss function L_I , since we only consider continuous control domains in this paper. The loss function can be replaced with a cross-entropy loss for discrete control tasks. We directly use the loss function L_I as the reward r_t for P , where r_t can be expressed as the following:

$$r_t = r(x_t, a_t, x_{t+1}, h_t) = L_I(a_t, \hat{a}_t | \theta_I) = \beta \|a_t - I(x_t, x_{t+1} | h_t, \theta_I)\|^2. \quad (3)$$

Our method targets at improving both the quality and efficiency of the data collection process performed by P as well as the performance of I via collecting difficult and non-trivial training samples. Therefore, the goal of the proposed framework is twofold. First, P has to learn an adversarial policy π_{adv} such that its accumulated discounted rewards $G_t | \pi_{adv} = \sum_{\tau=t}^T \gamma^{\tau-t} r(x_\tau, a_\tau, x_{\tau+1}, h_t)$ is maximized. Second, I requires to learn an optimal θ_I such that Eq. (3) is minimized. Minimizing L_I (i.e., r_t) leads to a decreased $G_t | \pi_{adv}$, forcing P to enhance π_{adv} to explore more difficult samples to increase $G_t | \pi_{adv}$. This implies that P is motivated to concentrate on discovering I 's weak points in the state space, instead of randomly or even repeatedly collecting ineffective training samples for I . Training I with hard samples not only accelerates its learning progress, but also helps to boost its performance. First of all, hard samples that provide higher prediction errors in Eq. (3) are unfamiliar to I , offering a direction for P to explore in the environment. Such a directed exploration scheme enables our adversarial active exploration to discover unfamiliar data more rapidly than random exploration. Moreover, those hard samples that require more gradient steps to learn tend to result in larger errors of I than samples which can be learned in just a few steps, which in turn help to expedite the learning progress of I . Data samples causing negligible errors of I do not provide the same benefits, as they are already familiar to I . Our method is able to discover those hard samples in an environment and focus on learning them, which is similar to what boosting algorithms [29] do.

Comparing to active learning and curiosity-driven exploration, our adversarial active exploration demonstrates several strengths on inverse dynamics model learning. First of all, active learning [14] does not have a control policy to acquire the informative data in an environment, while our method casts the data acquisition process for inverse dynamics models as a decision process and solves it using DRL. Second, the curiosity-driven exploration approach [6] collects samples not directly tailored for I 's learning, while our method gathers samples that directly influence I 's learning progress. Moreover, the incentives for exploration in curiosity-based approaches disappear rapidly when the forward dynamics model quickly learns to perform correct predictions, which may further undermine the training progress of their inverse dynamics models. On the contrary, our method continuously explores the environment until I converges. A more detailed comparison is presented in Section 5.2.

4.2 Training Methodology

We describe the training methodology of our adversarial active exploration method by a pseudocode presented in Algorithm 1. Assume that P 's policy π_{adv} is parameterized by a set of trainable parameters θ_P , and is represented as $\pi_{adv}(a_t | x_t, \theta_P)$. We create two buffers Z_P and Z_I for storing the training samples of P and I , respectively. In the beginning, Z_P , Z_I , \mathcal{E} , θ_P , θ_I , π_{adv} , as well as a timestep cumulative counter c are initialized. A number of hyperparameters are set to appropriate values, including the number of iterations N_{iter} , the number of episodes $N_{episode}$, the horizon T , as well as the update period \mathcal{T}_P of θ_P . At each timestep t , P perceives the current state x_t from \mathcal{E} , takes an action a_t according to $\pi_{adv}(a_t | x_t, \theta_P)$, and receives the next state x_{t+1} and a termination indicator ξ (lines 9-11). ξ is set to 1 only when t equals T , otherwise it is set to 0. We then store

Algorithm 1 Adversarial Active Exploration

```
1: Initialize  $Z_P, Z_I, \mathcal{E}$ , and model parameters  $\theta_P$  &  $\theta_I$ 
2: Initialize  $\pi_{adv}(a_t|x_t, \theta_P)$ 
3: Initialize the timestep cumulative counter  $c = 0$ 
4: Set  $N_{iter}, N_{episode}, T$ , and  $\mathcal{T}_P$ 
5: for iteration  $i = 1$  to  $N_{iter}$  do
6:   for episode  $e = 1$  to  $N_{episode}$  do
7:     for timestep  $t = 0$  to  $T$  do
8:        $P$  perceives  $x_t$  from  $\mathcal{E}$ , and predicts an action  $a_t$  according to  $\pi_{adv}(a_t|x_t, \theta_P)$ 
9:        $x_{t+1} = \mathcal{E}(x_t, a_t)$ 
10:       $\xi = \mathbb{1}[t == T]$ 
11:      Store  $(x_t, a_t, x_{t+1}, \xi)$  in  $Z_P$ 
12:      Store  $(x_t, a_t, x_{t+1})$  in  $Z_I$ 
13:      if  $(c \bmod \mathcal{T}_P) == 0$  then
14:        Initialize an empty batch  $B$ 
15:        Initialize a recurrent state  $h_t$ 
16:        for  $(x_t, a_t, x_{t+1}, \xi)$  in  $Z_P$  do
17:          Evaluate  $\hat{a}_t = I(x_t, x_{t+1}|h_t, \theta_I)$  (calculated from Eq. (1))
18:          Evaluate  $r(x_t, a_t, x_{t+1}, h_t) = L_I(a_t, \hat{a}_t|\theta_I)$  (calculated from Eq. (3))
19:          Store  $(x_t, a_t, x_{t+1}, r_t)$  in  $B$ 
20:        Update  $\theta_P$  with the gradient calculated from the samples of  $B$ 
21:        Reset  $Z_P$ 
22:         $c = c + 1$ 
23:      Update  $\theta_I$  with the gradient calculated from the samples of  $Z_I$  (according to Eq. (2))
24: end
```

(x_t, a_t, x_{t+1}, ξ) and (x_t, a_t, x_{t+1}) in Z_P and Z_I , respectively. We update θ_P every \mathcal{T}_P timesteps using the samples stored in Z_P , as shown in (lines 13-21). At the end of each episode, we update θ_I with samples drawn from Z_I according to the loss function L_I defined in Eqs. (2) and (3) (line 23).

4.3 Stabilization Technique

Although adversarial active exploration is effective in collecting hard samples, it requires additional adjustments if P becomes too strong such that the collected samples are too difficult for I to learn. Overly difficult samples lead to a large magnitudes in gradients derived from L_I , which in turn cause a performance drop in I and instability in its learning process. We analyze this phenomenon in greater detail in Section 5.4. To tackle the issue, we propose a training technique that reshapes r_t as follows:

$$r_t := -|r_t - \delta|, \quad (4)$$

where δ is a pre-defined threshold value. This technique poses a restriction on the range of r_t , driving P to gather moderate samples instead of overly hard ones. Note that the value of δ affects the learning speed and the final performance. We plot the impact of δ on the learning curve of I in Section 5.4. We further provide an example in our supplementary material to visualize the effect of this technique.

5 Experimental Results

In this section, we present experimental results for a series of robotic control tasks, and validate that (i) our method is effective for common robotic arm control and in-hand manipulation tasks; (ii) our method is effective in environments with high-dimensional action spaces; (iii) our method is more data efficient than the baseline methods. We first introduce our experimental setup. Then, we report our experimental results on a number of robotic arm and hand manipulation tasks. Finally, we present a comprehensive set of ablative analysis to validate the effectiveness for each of our design decisions.

5.1 Experimental Setup

We first describe the environments and tasks. Next, we explain the evaluation procedure and the method for collecting expert demonstrations. We then walk through the baseline approaches used in our comparisons. Please note that for all of our experiments, we train our DRL agent using PPO [28].

5.1.1 Environments and Tasks

We evaluate our method on a number of robotic arm and hand manipulation tasks via OpenAI gym [30] environments simulated by the MuJoCo [13] physics engine. We use the Fetch and Shadow Dexterous Hand [31] for the arm and hand manipulation tasks, respectively. For the arm manipulation tasks, which include *FetchReach*, *FetchPush*, *FetchPickAndPlace*, and *FetchSlide*, I takes as inputs the positions and velocities of a gripper and a target object. It then infers the gripper’s action in 3-dimensional space to manipulate it. For the hand manipulation task *HandReach*, the inverse dynamics model takes as inputs the positions and velocities of the fingers of a robotic hand, and determines the

velocities of the joints to achieve the goal. The detailed description of the above tasks is specified in [31]. For the detailed configurations of these tasks, please refer to our supplementary material.

5.1.2 Evaluation Procedure

The primary objective of our experiments is to demonstrate the efficiency of the proposed adversarial active exploration in collecting training data (in a self-supervised manner) for inverse dynamics models. We compare our method against a number of data collection methods (referred to as "baselines" or "baseline methods") described in Section 5.1.4. As different baseline methods employ different data collection strategies, the learning curves of their inverse dynamics models also vary for different cases. For a fair comparison, the model architecture of the inverse dynamics model and the amount of training data are fixed for all cases. All of the experimental results are evaluated and averaged over 20 trials, corresponding to 20 different random initial seeds. In each trial, we train an inverse dynamics model by the training data collected by a single data collection method. We periodically evaluate the inverse dynamics model when every 10K training samples are collected. At the beginning of each episode of evaluation, the inverse dynamics model receives a sequence of states $\{\hat{x}_1, \hat{x}_2, \dots, \hat{x}_T\}$ from a successful expert demonstration. At each timestep t , the inverse dynamics model infers an action \hat{a}_t from an expert state \hat{x}_{t+1} and its current state x_t by Eq. (1). The evaluation is performed by averaging the success rates of reaching \hat{x}_T over 500 episodes. The configuration of the inverse dynamics model and the hyperparameters are summarized in the supplementary material.

5.1.3 Expert Demonstrations for Evaluation

For each task mentioned in Section 5.1.1, we first randomly configure task-relevant settings (e.g., goal position, initial state, etc.). The configuration details of these tasks are specified in the codebase of OpenAI gym¹. We then collect demonstrations from non-trivial and successful episodes performed by a pre-trained expert agent [32]. Please note that the collected demonstrations only contain sequences of states. The implementation details of the expert agent and the methodology for filtering out trivial episodes are presented and discussed in detail in our supplementary material.

5.1.4 Baseline Data Collection Methods

We compare our proposed methodology with the following four baseline methods in our experiments.

- *Random*: This method collects training samples by random action sampling. We consider it to be an important baseline method because of its simplicity and prevalence in a number of research works on inverse dynamics models learning [6, 7, 8].
- *Demo*: This method trains the inverse dynamics model directly with expert demonstrations. Since expert demonstrations comprise only successful trajectories, this method serves as a baseline with human priors similar to [7].
- *Curiosity*: This method incentivizes a DRL agent to collect samples that lead to large errors of its forward dynamics model [6, 9]. Unlike the original implementation, we replace its DRL algorithm with PPO. This is also an important baseline due to its effectiveness in [6].
- *Noise* [33]: In this method, noise is injected to the parameter space of a DRL agent to encourage exploration [33]. We include this baseline as a reference to validate if a standard DRL exploration strategy can provide positive impact on inverse dynamics model learning.

5.2 Performance Comparison in Robotic Arm Manipulation Tasks

We compare the performance of the proposed method and the baselines on the robotic arm manipulation tasks described in Section 5.1.1. We discuss the experimental results and plot them in Fig. 2. All of the results are obtained by following the procedure described in Section 5.1.2. The shaded regions in Fig. 2 represent the confidence intervals. Fig. 2 plots the learning curves for all of the methods. In all of the tasks, our method yields superior or comparable performance to the baselines except for *Demo*, which is trained directly with expert demonstrations (i.e. human priors). In *FetchReach*, it can be seen that every method achieves a success rate of 1.0. This implies that it does not require a sophisticated exploration strategy to learn an inverse dynamics model in an environment where the dynamics is relatively simple. It should be noted that although all methods reach the same final success rate, ours learns significantly faster than *Demo*. In contrast, in *FetchPush*, our method is comparable to *Demo*, and demonstrates superior performance to the other baselines. Our method also learns drastically faster than all the other baselines, which confirms that the proposed strategy does improve the performance and efficiency of inverse dynamics model learning. Our method is

¹<https://github.com/openai/gym>

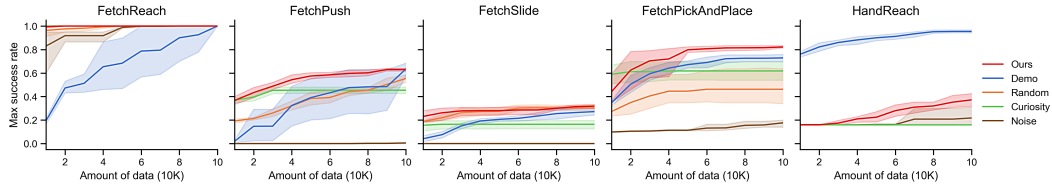


Figure 2: Performance comparison of robotic arm and hand tasks.

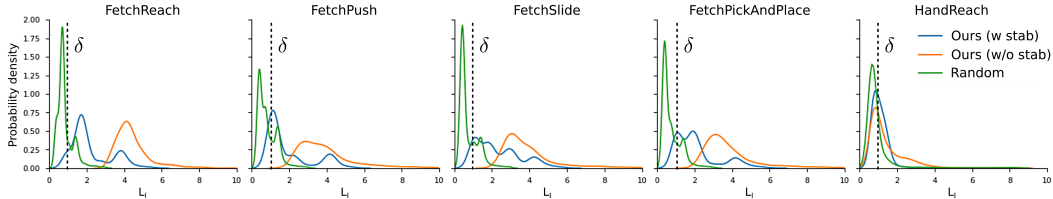


Figure 3: PDFs of L_I for the first 2K training batches.

particularly effective in tasks that require complex control. In *FetchPickAndPlace*, for example, our method surpasses all the other baselines. However, all methods including *Demo* fail to learn a successful inverse dynamics model in *FetchSlide*, which suggests that it is difficult to train an inverse dynamics model when the outcome of an action is partially affected by the unobservable factors of an environment. In *FetchSlide*, the movement of the object on the slippery surface is affected by both friction and the force exerted by the gripper. It is worth noting that *Curiosity* loses to *Random* in *FetchPush* and *FetchSlide*, and *Noise* performs even worse than these two methods in all of the tasks. We therefore conclude that *Curiosity* and the parameter space noise strategy cannot be directly applied to inverse dynamics model learning. Moreover, Fig. 2 shows that the performance of *Curiosity* saturates rapidly, indicating that *Curiosity* is unable to continuously explore the environment until I converges. In addition to the quantitative results presented above, we further discuss the empirical results qualitatively. Please refer our supplementary material for a detailed description of the results.

5.3 Performance Comparison in Robotic Hand Manipulation Task

Fig. 2 plots the learning curves for each of the methods considered. Please note that *Curiosity*, *Noise* and our method are pre-trained with 30K samples collected by random exploration, as we observe that these methods on their own suffer from large errors in an early stage during training, which prevents them from learning at all. After the first 30K samples, they are trained with data collected by their exploration strategy instead. From the results in Fig. 2, it can be seen that *Demo* easily stands out from the other methods as the best-performing model, surpassing them all by a considerable extent. Although our method is not as impressive as *Demo*, it significantly outperforms all of the other baseline methods, achieving a success rate of 0.4 while the others are still stuck at around 0.2.

The reason that the inverse dynamics models trained by the autonomous data-collection strategies discussed in this paper (including ours and the other baselines) are not comparable to the *Demo* baseline in the *HandReach* task is primarily due to the high-dimensional action space. It is observed that the data collected by the autonomous data-collection strategies only cover a very limited range of the state space in the *HandReach* environment. Therefore, the inverse dynamics models trained with these data only learn to predict trivial poses, leading to the poor success rates presented in Fig. 2.

5.4 Ablative Analysis

In this section, we provide a set of ablative analysis. We examine the effectiveness of our method by an investigation of the training loss distribution, the stabilization technique, and the influence of δ . Please note that the value of δ is set to 1.5 by default, as described in our supplementary material.

Training loss distribution. Fig. 3 plots the probability density function (PDF) of L_I (derived from Eq. (2)) by kernel density estimation (KDE) for the first 2K training batches during the training phase. The vertical axis corresponds to the probability density, while the horizontal axis represents the scale of L_I . The curves *Ours (w stab)* and *Ours (w/o stab)* represent the cases where the stabilization technique described in Section 4.3 is employed or not, respectively. We additionally plot the curve *Random* in Fig. 3 to highlight the effectiveness of our method. It can be observed that both *Ours (w stab)* and *Ours (w/o stab)* concentrate on notably higher loss values than *Random*. This observation implies that adversarial active exploration does explore hard samples for inverse dynamics model.

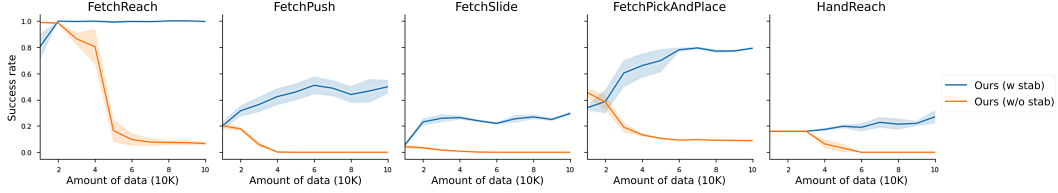


Figure 4: Learning curves w/ and w/o the stabilization technique.

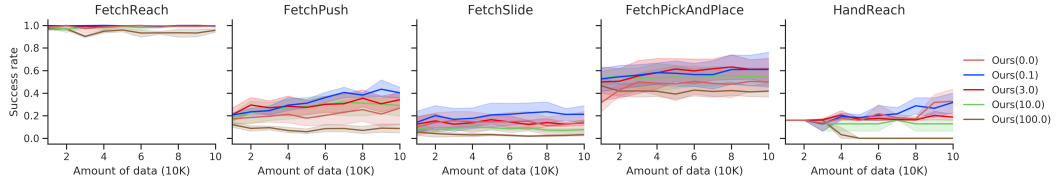


Figure 5: Performance comparison for different values of δ .

Validation of the stabilization technique. We validate the proposed stabilization technique in terms of the PDF of L_I and the learning curve of the inverse dynamics model, and plot the results in Figs. 3 and 4, respectively. From Fig. 3, it can be observed that the modes of *Ours (w stab)* are lower than those of *Ours (w/o stab)* in most cases, implying that the stabilization technique indeed motivates the DRL agents to favor those moderately hard samples. We also observe that for each of the five cases, the mode of *Ours (w stab)* is close to the value of δ (plotted in a dotted line), indicating that our reward structure presented in Eq. (4) does help to regulate L_I (and thus r_t) to be around δ . To further demonstrate the effectiveness of the stabilization technique, we compare the learning curves of *Ours (w stab)* and *Ours (w/o stab)* in Fig. 4. It is observed that for the initial 10K samples of the five cases, the success rates of *Ours (w/o stab)* are comparable to those of *Ours (w stab)*. However, their performance degrade drastically during the rest of the training phase. This observation confirms that the stabilization technique does contribute significantly to our adversarial active exploration.

Although most of the DRL works suggest that the rewards should be re-scaled or clipped within a range (e.g., from -1 to 1), the unbounded rewards do not introduce any issues during the training process of our experiments. The empirical rationale is that the rewards received by the DRL agent are regulated by Eq. (7) to be around δ , as described in Section 5.4 and depicted in Fig. 3. Without the stabilization technique, however, the learning curves of the inverse dynamics model degrade drastically (as illustrated in Fig. 2).

Influence of δ . Fig. 5 compares the learning curves of the inverse dynamics model for different values of δ . For instance, *Ours(0.1)* corresponds to $\delta = 0.1$. It is observed that for most of the tasks, the success rates drop when δ is set to an overly high or low value (e.g., 100.0 or 0.0), suggesting that a moderate value of δ is necessary for the stabilization technique. The value of δ can be adjusted dynamically by the adaptive scaling technique presented in [33], which is left as our future direction.

6 Conclusion

In this paper, we presented an adversarial active exploration, which consists of a DRL agent and an inverse dynamics model competing with each other for efficient data collection. The former is encouraged to actively collect difficult training data for the latter, such that the training efficiency of the latter is significantly enhanced. Experimental results demonstrated that our method substantially improved the data collection efficiency in multiple robotic arm and hand manipulation tasks, and boosted the performance of the inverse dynamics models. The comparisons with the forward dynamics errors driven approach [6] validated that our method is more effective for training inverse dynamics models. We further showed that our method is generalizable to environments with high-dimensional action spaces. Finally, we provided a set of ablative analysis to justify each of our design decisions.

Acknowledgment

This work was supported by the Ministry of Science and Technology (MOST) in Taiwan under grant nos. MOST 108-2636-E-007-010 (Young Scholar Fellowship Program) and MOST 108-2634-F-007-002. Z.-W. Hong and C.-Y. Lee acknowledge the financial support from MediaTek Inc., Taiwan. The authors would also like to acknowledge the donation of the Titan XP GPUs from NVIDIA Corporation and the Titan V GPUs from NVIDIA AI Technology Center used in this research work.

References

- [1] D. Nguyen-Tuong and J. Peters. Using model knowledge for learning inverse dynamics. In *Proc. Int. Conf. Robotics and Automation (ICRA)*, pages 2677–2682, May 2010.
- [2] F. Meier, D. Kappler, N. Ratliff, and S. Schaal. Towards robust online inverse dynamics learning. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pages 4034–4039, Oct. 2016.
- [3] E. Rueckert, M. Nakatenus, S. Tosatto, and J. Peters. Learning inverse dynamics models in $o(n)$ time with lstm networks. In *Proc. IEEE-RAS Int. Conf. Humanoid Robotics (Humanoids)*, pages 811–816, Nov. 2017.
- [4] D. Nguyen-Tuong and J. Peters. Model learning for robot control: A survey. *Cognitive processing*, 12(4):319–340, Nov. 2011.
- [5] O. Kanoun, J.-P. Laumond, and E. Yoshida. Planning foot placements for a humanoid robot: A problem of inverse kinematics. *The Int. J. Robotics Research*, 30(4):476–485, Jun. 2011.
- [6] D. Pathak, P. Mahmoudieh, M. Luo, P. Agrawal, D. Chen, F. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell. Zero-shot visual imitation. In *Proc. Int. Conf. Learning Representations (ICLR)*, Apr.-May 2018.
- [7] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *Proc. Int. Conf. Robotics and Automation (ICRA)*, pages pp. 2146-2153, May-Jun. 2017.
- [8] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pages pp. 5074-5082, Dec. 2016.
- [9] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proc. Int. Conf. Machine Learning (ICML)*, Aug. 2017.
- [10] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta. Robust adversarial reinforcement learning. In *Proc. Int. Conf. Machine Learning (ICML)*, Aug. 2017.
- [11] H. Shioya, Y. Iwasawa, and Y. Matsuo. Extending robust adversarial reinforcement learning considering adaptation and diversity. In *Proc. Int. Conf. Learning Representations (ICLR) Workshop*, Apr.-May 2018.
- [12] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *Proc. Int. Conf. Learning Representations (ICLR)*, Apr.-May 2018.
- [13] E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control view. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pages 5026-5033, Oct. 2012.
- [14] B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.
- [15] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proc. ACM Int. Conf. Research and Development in Information Retrieval (SIGIR)*, pages 3–12. Springer, Jul. 1994.
- [16] Y. Yang, Z. Ma, F. Nie, X. Chang, and A. G. Hauptmann. Multi-class active learning by uncertainty sampling with diversity maximization. *Int. J. of Computer Vision*, 113(2):113–127, Jun. 2015.
- [17] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proc. Annual Wksp. Computational Learning Theory*, pages 287–294, Jul. 1992.
- [18] M. Fang, Y. Li, and T. Cohn. Learning how to active learn: A deep reinforcement learning approach. *arXiv:1708.02383*, Aug. 2017.

- [19] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [20] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pages pp. 1057-1063, Dec. 2000.
- [21] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, May 1992.
- [22] G. Ostrovski, M. G. Bellemare, A. Oord, and R. Munos. Count-based exploration with neural density models. In *Proc. Int. Conf. Machine Learning (ICML)*, pages 2721–2730, Aug. 2017.
- [23] B. C. Stadie, S. Levine, and P. Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv:1507.00814*, Nov. 2015.
- [24] A. Laversanne-Finot, A. Péré, and P.-Y. Oudeyer. Curiosity driven exploration of learned disentangled goal spaces. *arXiv:1807.01521*, Nov. 2018.
- [25] C. Colas, P.-Y. Oudeyer, O. Sigaud, P. Fournier, and M. Chetouani. CURIOUS: Intrinsically motivated modular multi-goal reinforcement learning. In *Int. Conf. Machine Learning (ICML)*, pages 1331–1340, Jun. 2019.
- [26] A. Baranes and P.-Y. Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013.
- [27] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages pp. 5-32. Springer, May 1992.
- [28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, Aug. 2017.
- [29] R. E. Schapire. A brief introduction to boosting. In *Proc. Int. Joint Conf. Artificial intelligence (IJCAI)*, volume 99, pages 1401–1406, Aug. 1999.
- [30] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv:1606.01540*, Jun. 2016.
- [31] M. Plappert et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv:1802.09464*, Mar. 2018.
- [32] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pages pp. 5048-5058, Dec. 2017.
- [33] M. Plappert, R. Houthoof, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz. Parameter space noise for exploration. In *Proc. Int. Conf. Learning Representations (ICLR)*, Apr.-May 2018.
- [34] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proc. Int. Conf. Machine Learning (ICML)*, pages 41-48, 2009.
- [35] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997. ISSN 0899-7667. doi:10.1162/neco.1997.9.8.1735. URL <http://dx.doi.org/10.1162/neco.1997.9.8.1735>.

Supplementary Material

A Qualitative Analysis of Robotic Arm Manipulation Tasks

In addition to the quantitative results presented above, we further discuss the empirical results qualitatively. Through visualizing the training progress, we observe that our method initially acts

like *Random*, but later focuses on interacting with the object in *FetchPush*, *FetchSlide*, and *FetchPickAndPlace*. This phenomenon indicates that adversarial active exploration naturally gives rise to a curriculum that improves the learning efficiency, which resembles curriculum learning [34]. Another benefit that comes with the phenomenon is that data collection is biased towards interactions with the object. Therefore, the DRL agent concentrates on collecting interesting samples that has greater significance, rather than trivial ones. For instance, the agent prefers pushing the object to swinging the robotic arm. On the other hand, although *Curiosity* explores the environment very thoroughly in the beginning by stretching the arm into numerous different poses, it quickly overfits to one specific pose. This causes its forward dynamics model to keep maintaining a low error, making it less curious about the surroundings. Finally, we observe that the exploratory behavior of *Noise* does not change as frequently as ours, *Random*, and *Curiosity*. We believe that the method’s success in the original paper [33] is largely due to extrinsic rewards. In the absence of extrinsic rewards, however, the method becomes less effective and unsuitable for data collection, especially in inverse dynamics model learning.

B Proximal Policy Optimization (PPO)

We employ PPO [28] as the RL agent responsible for collecting training samples because of its ease of use and good performance. PPO computes an update at every timestep that minimizes the cost function while ensuring the deviation from the previous policy is relatively small. One of the two main variants of PPO is a clipped surrogate objective expressed as:

$$L^{CLIP}(\theta) = \mathbb{E} \left[\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)} \hat{A}, \text{clip}\left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)}, 1 - \epsilon, 1 + \epsilon\right) \hat{A} \right],$$

where \hat{A} is the advantage estimate, and ϵ a hyperparameter. The clipped probability ratio is used to prevent large changes to the policy between updates. The other variant employs an adaptive penalty on KL divergence, given by:

$$L^{KL PEN}(\theta) = \mathbb{E} \left[\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)} \hat{A} - \beta KL[\pi_{\theta_{old}}(\cdot|s), \pi_{\theta}(\cdot|s)] \right],$$

where β is an adaptive coefficient adjusted according to the observed change in the KL divergence. In this work, we employ the former objective due to its better empirical performance.

C Implementation Details of Inverse Dynamics Model

In the experiments, the inverse dynamics model $I(x_t, x_{t+1}|h_t, \theta_I)$ of all methods employs the same network architecture. We use 3 Fully-Connected (FC) layers with 256 hidden units followed by *tanh* activation units. We then feed the extracted features from the FC layers to Long-Short Term Memory (LSTM) [35].

D Implementation Details of Adversarial Active Exploration

We use the architecture proposed in Schulman et al. [28]. During training, we periodically update the DRL agent with a batch of transitions as described in Algorithm. 1. We split the batch into several mini-batches, and update the RL agent with these mini-batches iteratively. The best hyperparameters are listed in Table. 1 (our method).

E Implementation details of *Curiosity*

Our baseline *Curiosity* is implemented based on the work [6]. The authors in Pathak et al. [6] propose to employ a curiosity-driven RL agent [9] to improve the efficiency of data collection. The curiosity-driven RL agent takes curiosity as intrinsic reward signal, where curiosity is formulated as the error in an agent’s ability to predict the consequence of its own actions. This can be defined as a forward dynamics model:

$$\hat{\phi}(x') = f(\phi(x), a; \theta_F), \tag{5}$$

where $\hat{\phi}(x')$ is the predicted feature encoding at the next timestep, $\phi(x)$ the feature vector at the current timestep, a the action executed at the current timestep, and θ_F the parameters of the forward model f . The network parameters θ_F is optimized by minimizing the loss function L_F :

$$L_F(\phi(x), \hat{\phi}(x')) = \frac{1}{2} \|\hat{\phi}(x') - \phi(x_{t+1})\|_2^2. \tag{6}$$

Hyperparameter	Value
Common	
Batch size for inverse dynamic model update	64
Learning rate of inverse dynamic model	1e-3
Timestep per episode	50
Optimizer for inverse dynamic model	Adam
Our method	
Number of batch for update inverse dynamic model	25
Batch size for RL agent	2050
Mini-batch size for RL agent	50
Number of training iteration (N_{iter})	200
Number of training episode per iteration ($N_{episode}$)	10
Horizon (T) of RL agent	50
Update period of RL agent	2050
Learning rate of RL agent	1e-3
Optimizer for RL agent	Adam
δ of stabilization	1.5
Curiosity	
Number of batch for update inverse dynamic model	500
Batch size for RL agent	2050
Mini-batch size for RL agent	50
Number of training iteration (N_{iter})	10
Number of training episode per iteration ($N_{episode}$)	200
Horizon (T) of RL agent	50
Update period of RL agent	2050
Learning rate of RL agent	1e-3
Optimizer for RL agent	Adam
Noise	
Number of batch for update inverse dynamic model	500
The other hyperparameters	Same as Plappert et al. [33]

Table 1: Hyperparameters settings.

We use the architecture proposed in Schulman et al. [28]. The implementation of ϕ depends on the model architecture of the RL agent. For low-dimensional observation setting, we implement ϕ with the architecture of low-dimensional observation PPO. Note that ϕ does not share parameters with the RL agent in this case. The best hyperparameters settings can be found in Table. 1(Curiosity).

F Implementation Details of *Noise*

We directly apply the same architecture in Plappert et al. [33] without any modification. Please refer to Plappert et al. [33] for more detail.

G Implementation Details of *Demo*

We collect 1000 episodes of expert demonstrations using the procedure defined in Sec. S9 for training *Demo*. Each episodes lasts 50 timesteps. The demonstration data is in the form of a 3-tuple (x_t, a, x_{t+1}) , where x_t is the current observation, a_t the action, and x_{t+1} the next observation. The pseudocode for training *Demo* is shown in Algorithm. S1 below. In each training iteration, we randomly sample 200 episodes, namely 10k transitions (line 4). The sampled data is then used to update the inverse dynamics model (line 5).

H Configuration of Environments

We briefly explain each configuration of the environment below. For detailed description, please refer to Plappert et al. [31].

- *FetchReach*: Control the gripper to reach a goal position in 3D space.
- *FetchPush*: Control the Fetch robot to push the object to a target position.

Algorithm 2 Demo

- 1: Initialize Z_{Demo}, θ_I
 - 2: Set constants N_{iter}
 - 3: **for** iter $i = 1$ to N_{iter} **do**
 - 4: Sample 200 episodes of demonstration from Z_{Demo} as B
 - 5: Update θ_I with the gradient calculated from the samples in B (according to Eq. 6)
 - 6: **end**
-

- *FetchPickAndPlace*: Control the gripper to grasp and lift the object to a goal position. This task requires a more accurate inverse dynamics model.
- *FetchSlide*: Control the robot to slide the object to a goal position. The task requires an even more accurate inverse dynamics model, as the object’s movement on the slippery surface is hard to predict.
- *HandReach*: Control the Shadow Dextrous Hand to reach a goal hand pose. The task is especially challenging due to high-dimensional action spaces.

I Setup of Expert Demonstration

We employ Deep Deterministic Policy Gradient combined with Hindsight Experience Replay (DDPG-HER) [32] as the expert agent. For training and evaluation, we run the expert to collect transitions for 1000 and 500 episodes, respectively. To prevent the inverse dynamics model from succeeding in the task without taking any action, we only collect successful and non-trivial episodes generated by the expert agent. Non-trivial episodes are filtered out based on the following task-specific schemes:

- *FetchReach*: An episode is considered trivial if the distance between the goal position and the initial position is smaller than 0.2.
- *FetchPush*: An episode is determined trivial if the distance between the goal position and the object position is smaller than 0.2.
- *FetchSlide*: An episode is considered trivial if the distance between the goal position and the object position is smaller than 0.1.
- *FetchPickAndPlace*: The episode is considered trivial if the distance between the goal position and the object position is smaller than 0.2.
- *HandReach*: We do not filter out trivial episodes as this task is too difficult for most of the methods.

J Analysis of the number of expert demonstrations

Fig. 6 illustrates the performance of *Demo* with different number of expert demonstrations. *Demo(100)*, *Demo(1,000)*, and *Demo(10,000)* correspond to the *Demo* baselines with 100, 1,000, and 10,000 episodes of demonstrations, respectively. It is observed that their performance are comparable for most of the tasks except *FetchReach*. In *FetchReach*, the performance of *Demo(100)* is significantly worse than the other two cases. A possible explanation is that preparing a sufficiently diverse set of demonstrations in *FetchReach* is relatively difficult with only 100 episodes of demonstrations. A huge performance gap is observed when the number of episodes is increased to 1,000. Consequently, *Demo(1,000)* is selected as our *Demo* baseline for the presentation of the experimental results. Another advantage is that *Demo(1,000)* demands less memory than *Demo(10,000)*.

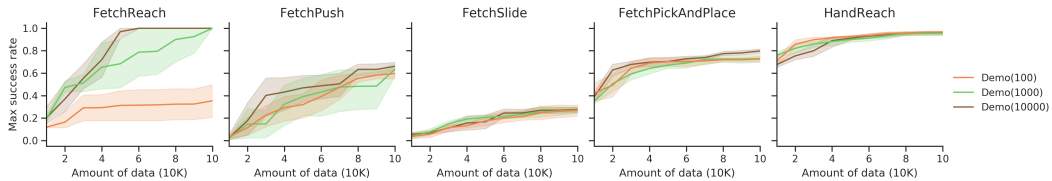


Figure 6: Comparison of different number of expert demonstrations in low-dimensional observation spaces.

K Visualization of stabilization technique

In this section, we visualize the effects of our stabilization technique with a list of rewards r in Fig. 7. The rows of *Before* and *After* represent the rewards before and after reward shaping, respectively. The bar on the right-hand side indicates the scale of the reward. It can be observed in Fig. 8 that after reward shaping, the rewards are transformed to the negative distance to the specified δ (i.e., 2.5 in

this figure). As a result, our stabilization technique is able to encourage the DRL agent to pursue rewards close to δ , where higher rewards can be received.

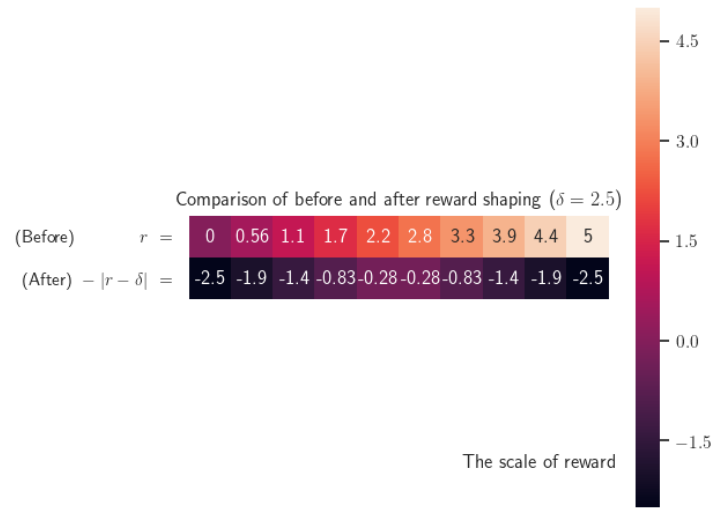


Figure 7: Visualization of the stabilization technique.