AUTO-ENCODING INVERSE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) provides a powerful framework for decision-making, but its application in practice often requires a carefully designed reward function. Inverse Reinforcement Learning (IRL) has shed light on automatic reward acquisition, but it is still difficult to apply IRL to solve real-world tasks. In this work, we propose Auto-Encoding Inverse Reinforcement Learning (AEIRL), a robust and scalable IRL framework, which belongs to the adversarial imitation learning class. To recover reward functions from expert demonstrations, AEIRL utilizes the reconstruction error as the learning signal, which provides more information for optimizing policies, compared to the binary logistic loss. Subsequently, we use the derived safe objective functions to train the reward function and the RL agent. In the experiments, we show that AEIRL performs superior in comparison with state-of-the-art methods in the MuJoCo environments. More importantly, in the more realistic settings, AEIRL shows much better robustness compared to other methods, when the expert demonstrations are noisy.

1 INTRODUCTION

Reinforcement learning (RL) provides a powerful framework for automating decision making. However, RL still requires significantly engineered reward functions for good practical performance. To make RL more applicable in the real-world, it is important to learn a reward function from expert demonstrations. Imitation learning offers the instruments to learning policies directly from the data, without an explicit reward function. Imitation learning enables the agents to learn to solve tasks from expert demonstrations, such as helicopter control (Abbeel et al., 2006; 2007; Ng et al., 2004; Coates et al., 2008; Abbeel et al., 2008a; 2010), robot navigation (Ratliff et al., 2006; Abbeel et al., 2008b; Ziebart et al., 2008; 2010), and building controls (Barrett & Linder, 2015).

The goal of imitation learning is to extract the expert policies from the expert demonstrations without the access to the reward signal from the environment. The algorithms in this field can be divided into two broad categories: behavioral cloning (BC) and Inverse Reinforcement Learning (IRL). BC formulates the learning task as a supervised learning problem, which learns a mapping from the states to the actions, using the expert trajectories. However, BC methods suffer from the problem of compounding errors, i.e., covariate shift, which only learn the actions of the expert but not reason about what the expert attempts to achieve. In the contrast, IRL directly infers the underlying reward function from the data, and then teaches the RL agent to learn a policy to achieve the highest accumulated reward. Empirical results show that IRL methods are more efficient than BC methods in multi-step decision making tasks. However, how to make IRL stable and efficient to use is still subject to research.

Current IRL methods, such as adversarial imitation learning approaches, model the reward function as a neural network to learn the mapping from the state-action pair to a scalar value, i.e., the reward (Ho & Ermon, 2016; Fu et al., 2017; Ghasemipour et al., 2020). However, this formulation focuses on the state-action features rather than the distribution of state-action visitations. We propose to formulate the reward function as the reconstruction error of an auto-encoder, which learns to reconstruct the expert demonstrations. In the adversarial training framework, we design several loss functions and the corresponding reward functions for stable optimizing the auto-encoder and the policy. The derived formulations ensure the reward signal is informative and bounded, which are efficient and safe for optimization. Additionally, The encoding-decoding process empowers the

capability to denoise and make the agent robust to the noisy expert, in the more realistic settings. In the experiments, we show that the proposed method AEIRL achieves the best overall performance on both clean and noisy settings.

Our contributions are two-folds:

- We propose the Auto-Encoding Inverse Reinforcement Learning (AEIRL) architecture, which models the reward function as a surrogate function using the reconstruction error of the state-action features. The reconstruction error provides more informative learning signals, compared to the binary logistic loss.
- The experimental results on the MoJoCo tasks show that AEIRL outperforms state-of-the-art imitation learning methods. Furthermore, the empirical results show that the learning processes of our methods are more stable in general.

2 Related Work and Background

IRL infers the reward function using the expert demonstrations, which are assumed to be the observations of optimal behaviors (Ng & Russell, 2000). In the past two decades, the related works can be categorized into two directions. The first direction is to enhance the representation ability of the reward functions, such as FIRL (Levine & Popvić, 2010), non-parametric IRL (Levine & Popvić, 2010), GP-IRL (Jin et al., 2015), and a class of methods that formulates the reward functions using deep neural networks, including GCL, GAIL, AIRL, f-GAIL (Finn et al., 2016b; Ho & Ermon, 2016; Fu et al., 2017; Zhang et al., 2020). The second direction is to extend the previous methods to the more complex settings, such as the high dimensional state space environments (Cai et al., 2021), dynamic environments (Finn et al., 2016b), and continuous control environments (Ho & Ermon, 2016).

In general, IRL is formulated as a bi-level optimization process meaning that iteratively training the reward function and optimizing the policy. A series of works show that Energy-Based Models (EBMs) play an important role in IRL. The basic assumption of EBMs is that the expert trajectories lie in a low energy space. Thus, the probability distribution of these trajectories can be modeled as a Boltzmann distribution:

$$p(\tau) = \frac{1}{Z}\exp(r_\theta(\tau)), \tag{1}$$

where the energy is given by the negative reward function $-r_\theta$, and Z is the partition function normalizing the probability. Maximum entropy IRL (MaxEnt IRL) (Ziebart et al., 2008; 2010) maximizes the following log-likelihood:

$$\mathcal{L} = \log(\prod_{\tau\in D} p(\tau)) = \sum_{\tau\in D}\log(\frac{1}{Z}\exp(r_\theta(\tau))) = \sum_{\tau\in D} r_\theta(\tau)) - \log Z \tag{2}$$

However, estimating the partition function Z is computationally expensive. MatEnt IRL is applicable in a finite MDP. Guided Cost Learning (GCL) (Finn et al., 2016b) uses a mixture of policy distributions to estimate the importance weights of the demonstrations, in order to adapt the method to more complex environments. Finn et al. (2016a) address the connection between GCL and GAIL (Ho & Ermon, 2016) i.e., the equivalent discriminators and the generator loss functionals, and further propose GAN-GCL. In practice, using full trajectories in the reward function optimization (GAN-GCL) could result high variance estimates. Adversarial Inverse Reinforcement Learning (AIRL) (Fu et al., 2017) addresses the connection between the reward function $r_\theta(s,a)$ and the discriminator $D(s,a)$ of adversarial imitation learning. Mathematically:

$$D_\theta(s,a) = \frac{\exp\left(r_\theta(s,a)\right)}{\exp\left(r_\theta(s,a)\right) + \pi(s,a)} \tag{3}$$

where $\pi(s,a)$ is the current policy we need to optimize. Minimizing the JS divergence between the occupancy measure of expert policy and the current policy leads to the loss functions of the original GAN method (Goodfellow et al., 2014). Although AIRL has limited improvements upon GAIL, AIRL can learn disentangled rewards and transfer better among different environments. Following the direction of AIRL, EAIRL (Qureshi et al., 2019) and FAIRL (Ghasemipour et al., 2020) focus

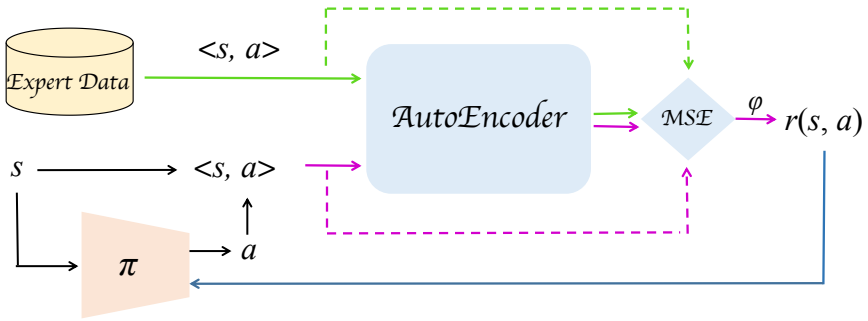


Figure 1: Adversarial training framework of autoencoding inverse reinforcement learning. The green line represents the expert data examples, and the purple line represents the rollout pairs. The autoencoder calculates the reconstruction error for these two minibatch of data examples and optimize itself. The surrogate reward function would feed back to update the policy.

on more meaningful reward functions. Energy-Based Imitation Learning (EBIL) (Liu et al., 2021) realizes IRL by two stages. First, estimate and use the expert energy as the surrogate reward function (Saremi et al., 2018). Secondly, utilize the reward to learn the policy. This method does not need to train the policy in an inner loop of optimizing the reward function. Thus, it has a good convergence rate. Strictly batch imitation learning (EDM) (Jarrett et al., 2020) proposes offline imitation learning with energy-based distribution matching. It doesn't need any interactions with the environment, but it also suffers from the problem of compounding errors. Energy Based GAN model (EBGAN) (Zhao et al., 2016) takes the inspiration from EBMs, which formulates the discriminator as an auto-encoder and achieves good image generation performance. The generator tries to minimize the energy, i.e., the reconstruction error of data examples. The energy function is optimized via a margin loss. However, EBGAN is hard to train due to the unbounded energy value. Another category of IRL is using a fixed similarity function to calculate the rewards. To the best of our knowledge, the most powerful method in this category is Primal Wasserstein Imitation Learning (PWIL) (Dadashi et al., 2021), which utilizes the upper bound of its primal form as the optimization objective. our method AEIRL takes inspirations from EBGAN and utilizes an auto-encoder as the reward function and derives the efficient and safe reconstruction error-based objective functions. Compared to the state-of-the-art methods, our method AEIRL achieves superior performance.

Furthermore, the robustness of adversarial imitation learning methods is questionable, even with a little noise in the expert demonstrations. Recently, learning from imperfect or noisy demonstrations become a popular topic in imitation learning (Sasaki & Yamashina, 2021). Previous methods require the demonstrations to be annotated with confidence scores (Wu et al., 2019). However, these annotations are rather expensive. In comparison to this, our method AEIRL is relatively robust to noisy expert demonstrations and does not require annotated data.

3 Auto-Encoding Inverse Reinforcement Learning

3.1 Our Method

Generally, our approach is to minimize the distance between the state action distribution of the policy we seek to train π and the state action distribution of the expert demonstrations D_E. We aim to optimize the $Earth\text{-}Mover$ distance i.e., Wasserstein-1 distance between two distributions P_r and P_g:

$$\mathcal{W}(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(s,y)\in\gamma}[\|x - y\|], \tag{4}$$

where $\Pi(P_r, P_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively P_r and P_g. Intuitively, $\gamma(x, y)$ indicates how much mass must be transported from x to y in order to transform the distributions P_r into the distribution P_g. Therefore, this distance is the cost of the optimal transport plan.

In our work, we propose that the reward function is structured with an auto-encoder which is defined as:

$$AE(x) = \|Dec \circ Enc(x) - x\|_2^2 \tag{5}$$

where x represents the state-action pairs.

To train the auto-encoder adversarially with the reinforcement learning step, we use the Wasserstein-1 distance in Equation 4:

$$\mathcal{L} = \mathbb{E}_{(s,a)\sim\pi}[1/(1 + AE(s,a))] - \mathbb{E}_{(s,a)\sim\pi_E}[1/(1 + AE(s,a))], \tag{6}$$

where AE is the reconstruction error of the auto-encoder as defined in Equation 5.

The loss function can be quickly minimized with just updating the last or last several layers of the auto-encoder while using unbound mean square error directly. So here, we use the reciprocal form to train the auto-encoder rather than the MSE directly which may leads to unstable training of the auto-encoder.

Figure 1 depicts the architecture for AEIRL. The reinforcement learning component is trained to roll out trajectories, which are the sequences of state-action pairs. The auto-encoder takes either expert or generated state-action pairs, and estimates the reconstruction error $AE(x)$ accordingly.

The output of the auto-encoder goes through an objective functional in order to shape the reward function which is defined as:

$$r(s,a) = \frac{1}{1 + AE(s,a)} \tag{7}$$

Considered that when the auto-encoder maps the samples to a latent space (might be a linear embedding with some noises) and reconstructs to the sample space, the noises can be removed by this reconstruction period. So our method can achieve great performance even with some noisy expert.

Algorithm 1 Auto-Encoding Inverse Reinforcement Learning (AEIRL)

Require: Initial parameters of policy, auto-encoder θ_0, w_0; Expert trajectories D_E.
Ensure:

1: **for** $i = 0$ to N **do**
2: Sample state-action pairs $(s_i, a_i) \sim \pi_{\theta_i}$ and $(s_E, a_E) \sim D_E$ with same batch size.
3: Update w_i to w_{i+1} by decreasing with the gradient:

$$\mathbb{E}_{(s_i,a_i)}[\nabla_{w_i} 1/(1 + AE_{w_i}(s_i,a_i))] - \mathbb{E}_{(s_E,a_E)}[\nabla_{w_i} 1/(1 + AE_{w_i}(s,a))]$$

4: Take a policy step from θ_i to θ_{i+1}, using the TRPO update rule with the reward function $1/(1 + AE_{w_i}(s,a))$, and the objective function for TRPO is:

$$\mathbb{E}_{(s,a)}[-1/(1 + AE_{w_i}(s,a))] - \lambda\mathcal{H}(\pi_{\theta_i}).$$

5: **end for**

Algorithm 1 depicts the procedure for training AEIRL. The first step is to sample the state-action pairs from expert demonstrations D_E and the trajectories rolled out by current policy with the same batch size. Then we update the auto-encoder by decreasing with the gradient with loss function 9. The Third step is to update the policy assuming that the reward function is equation 7 which leads to a adversarial training paradigm. We runs these steps with N iterations until the policy converges.

3.2 Alternative Surrogate Reward Signals and Auto-Encoders

As mentioned in section 3.1, we design the reward function to avoid auto-encoder "cheating" problem with bounded loss functions. In detail, this problem is that once the loss function to train the auto-encoder is unbounded in adversarial training, the loss may be easily minimized with just updating the last layer or part of the nets.

So we can use some other distribution distances to build a bounded reward signals without loss of generality. Here we propose another two reward signal formulations while following Jensen–Shannon distance to build the adversarial training paradigm.

One reward formulation is:

$$r(s,a) = -log(1 - exp(-AE(s,a))), \tag{8}$$

where the corresponding loss function for updating the auto-encoder is:

$$\mathcal{L} = \mathbb{E}_{(s,a)\sim D_E}[AE(s,a)] - \mathbb{E}_{(s,a)\sim\pi}[log(1 - exp(-AE(s,a)))] \tag{9}$$

Another reward formulation is:

$$r(s,a) = log(1 - \text{sigmoid}(-AE(s,a))), \tag{10}$$

where the corresponding loss function for updating the auto-encoder is:

$$\mathcal{L} = \mathbb{E}_{(s,a)\sim D_E}[-log(\text{sigmoid}(-AE(s,a))] + \mathbb{E}_{(s,a)\sim\pi}[log(1 - \text{sigmoid}(-AE(s,a)))] \tag{11}$$

This two forms of reward signals achieve comparable performance on almost all the tasks shown in Appendix 8. While the ordinary auto-encoder is used in our paper, variational auto-encoder could also be an alternative with just restrict the latent space distribution to a fixed Gaussian.

4 Theoretical Analysis

The reward function in our method can be represented as $r(x, \tilde{x}_w)$, where w means the parameters for the auto-encoder and x represents the state action pair, $\tilde{x}_w$ is the reconstructed pair. Let π_θ denote an agent's stochastic policy with parameter θ. And $\rho_\pi(x)$ denotes the probability density of visiting some state action pairs x, which is also known as occupancy measure (Syed et al., 2008). The optimization of the policy π_θ and recovering period of the reward function $r(x, \tilde{x}_w)$ forms a bi-level optimization problem, which can be formally defined as:

$$\begin{aligned} &\min_w \int_X r(x,\tilde{x}_w)\rho_{\pi_\theta}(x) - r(x,\tilde{x}_w)\rho_{\pi_E}(x)dx \\ &\quad \text{s.t.} \quad w \in W \\ &\max_\theta \int_X r(x,\tilde{x}_w)\rho_{\pi_\theta}(x)dx + \lambda\mathcal{H}(\pi(\cdot|s_t)) \\ &\quad\quad \text{s.t.} \quad \theta \in \Theta, \end{aligned} \tag{12}$$

where W and Θ denote the parameter space of the auto-encoder and the policy respectively, $\mathcal{H}$ denote the entropy of policy. Obviously, the lower bound of the inner optimization problem is:

$$\max_\theta \int_X r(x,\tilde{x}_w)\rho_{\pi_\theta}(x)dx, \tag{13}$$

where we can optimize this lower bound instead of the original one empirically. In this section, we want to show some analysis on the question whether the reward function could converge or not theoretically.

Assumption 1. *There is no region where $\rho_{\pi^*}(x) = 0$ where π^* is the induced optimal policy.*

Theorem 1 (Main Results)**.** *Suppose Assumption 1 holds, the reward function of our method could converge to a constant η in [0, 1] once the game for bi-level optimization problem reaches a Nash equilibrium.*

Proof. See Appendix A.1. □

5 Experiments

We conduct the experiments on six locomotion tasks of varying dynamics and difficulty: Swimmer-v2 (Schulman et al., 2015a), Hopper-v2 (Levine & Koltun, 2013), Walker2d-v2 (Schulman et al., 2015a), HalfCheetah-v2 (Heess et al., 2015), Ant-v2 (Schulman et al., 2016), Humanoid-v2 (Tassa et al., 2012). The goal for all these tasks is to move forward as quickly as possible. These tasks are more challenging than the basic tasks due to high degrees of freedom. In addition, a great amount of exploration is needed to learn to move forward without getting stuck at local optima.

The baselines we choose for comparison includes: behavior cloning (BC) (Dhariwal et al., 2017), GAIL (Ho & Ermon, 2016), Wasserstein GAIL (WGAIL) (Arjovsky et al., 2017), and PWIL (Dadashi et al., 2021). For fair comparison, we use TRPO (Schulman et al., 2015b) as the policy search method for all the imitation learning algorithms which is implemented by OpenAI Baselines (Dhariwal et al., 2017). The expert demonstrations are sampled with PPO (Schulman et al., 2017) for 5 tasks except for Humanoid task. And the expert data for Humanoid is sampled with SAC (Haarnoja et al., 2018) to get higher expert performance.

To our knowledge, PWIL achieves state-of-the-art performance, which outperforms Discriminator-Actor-Critic (Kostrikov et al., 2019) on almost all MuJoCo tasks (Dadashi et al., 2021), especially on Humanoid that claimed to be best performance so far.

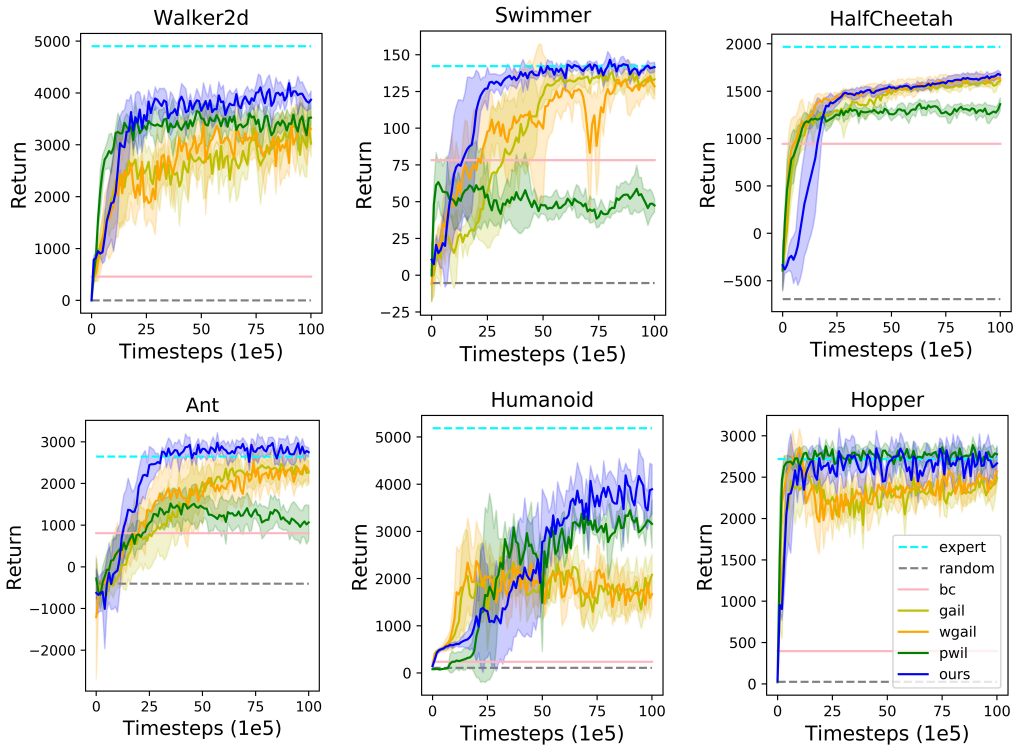


Figure 2: Mean and standard deviation return of the evaluation policy over 10 rollouts and 5 seeds learning from non-noisy expert demonstrations, reported every 100k timesteps. The return is in term of the environment's original reward.

We used all the algorithms to train policies of the same neural network architecture for all tasks: two hidden layers of 64 units each, with tanh nonlinearities in between. All networks were always initialized randomly at the start of each trial. For each task, we optimize BC with 100k iterations and gave GAIL, WGAIL, PWIL and our AEIRL exactly the same amount of environment interactions 10M for training. We ran all algorithms 5 times over different random seeds in all environments. Detailed experiments including implementation settings and more results on toy experiments and MuJoCo tasks can be seen in Appendix.

Note that the reward function tags used in experiments for our methods are shown in Table 4.

Table 1: Reward functions for different variants of our AEIRL

Method	Reward Function
Ours & Ours-VAE	$1/(1+AE(s,a))$
Ours-v1	$-log(1-exp(-AE(s,a)))$
Ours-v2	$log(1-\text{sigmoid}(-AE(s,a)))$

5.1 LEARNING FROM EXPERT DEMONSTRATIONS

Figure 2 depicts the results which are the training curve of original rewards for different imitation learning algorithms. The training curve shows the mean and standard deviation return of the evaluation policy over 10 rollouts and 5 seeds learning from non-noisy expert data. Table 2 shows the final learned policy performance on MuJoCo tasks with raw episode return.

Table 2: Learned policy performance for different imitation learning algorithms on non-noisy expert data, evaluated by using the mode of sampling from the policy distributions. The best top 3 results are marked as **red**, **blue** and **green**.

Task	Walker	Hopper	Swimmer	HalfCheetah	Ant	Humanoid
Random	-0.6 ± 0.3	25.5 ± 4.2	-5.3 ± 13.9	-694 ± 108	-401.4 ± 233.4	110 ± 4.1
Expert	4904.2	2719.8	142.2	1969.2	2646.1	5187.9
BC	459.7 ± 374.8	396.5 ± 435.7	78.3 ± 39.6	945.2 ± 160.7	807.8 ± 1843.9	237.8 ± 621.4
GAIL	3023.1 ± 254.4	2487.3 ± 80.3	133.2 ± 10.71	610.2 ± 25.3	2262.6 ± 280.7	2078.9 ± 433.8
WGAIL	3308.5 ± 361.4	2570.1 ± 122.1	133.4 ± 5.7	1658.4 ± 42.7	2356.5 ± 299.2	1669.6 ± 296.3
PWIL	3511.4 ± 130.3	**2809.9 ± 47.2**	47.9 ± 4.8	1372.5 ± 71.3	1067.9 ± 462.3	**3157.2 ± 507.8**
Ours	**3897.8 ± 177.8**	**2706 ± 219.7**	**141.5 ± 3.7**	**1674.1 ± 33.6**	**2715.7 ± 93.1**	**3892.3 ± 522.9**
Ours-v1	**4055.9 ± 304.0**	2580.3 ± 186.4	**146.1 ± 2.6**	**1660.6 ± 26.3**	**2930.4 ± 236.8**	122 ± 17
Ours-v2	**3589.1 ± 427.4**	**2770.5 ± 66.8**	**143.1 ± 3**	**1670.6 ± 20.4**	**3062.9 ± 282**	**2167 ± 1212.1**
Ours-VAE	$\pm$	$\pm$	$\pm$	$\pm$	$\pm$	$\pm$

Obviously, our proposed AEIRL method achieves expert performane on these tasks. AEIRL outperforms other SoTA methods on almost all the locomotion tasks except for Hopper where PWIL achieves best performance on this task. PWIL achieves higher performance compared with GAIL and WGAIL on Walker2d and Hopper tasks but performs even worse than BC on Swimmer. GAIL and WGAIL get similar results while WGAIL could be a little better than GAIL on these tasks. Table 2 shows the best top 3 results marked with different colors. Three different loss functions for our methods are applied and all achieves strong results and get best performance compared with other methods except for on Hopper.

5.2 LEARNING FROM NOISY DEMONSTRATIONS

Table 3: Learned policy performance for different imitation learning algorithms on noisy expert data, evaluated by using the mode of sampling from the policy distributions. The best top 3 results are marked as **red**, **blue** and **green**.

Task	Walker	Hopper	Swimmer	HalfCheetah	Ant	Humanoid
BC	728.2 ± 835.3	195.8 ± 15.7	35.6 ± 2.3	898.1 ± 827.9	468.6 ± 495.9	186.7 ± 5243.6
GAIL	1254.1 ± 231.8	1478.0 ± 405.9	**106.4 ± 19.7**	862.3 ± 94.2	**-836.5 ± 965.0**	1085.0 ± 622.8
WGAIL	940.9 ± 383.8	2570.1 ± 122.1	95.9 ± 15.4	696.4 ± 211.7	-2719.1 ± 1396.2	1479.1 ± 539.8
PWIL	3052.4 ± 346.6	**2588.9 ± 159.7**	57.9 ± 8.1	1372.5 ± 71.3	-1886.2 ± 1591.9	**3218.9 ± 281.4**
Ours	**3554.6 ± 275.6**	**1891.8 ± 197.0**	105.0 ± 2.7	**1501.6 ± 92.4**	**527.2 ± 584.1**	**3690.2 ± 374.8**
Ours-v1	**3364.5 ± 377.3**	1638.3 ± 308.5	**107.8 ± 9.6**	**1444.3 ± 16.8**	-1207.6 ± 168.2	123.2 ± 23.1
Ours-v2	**3561.8 ± 345.7**	**1671.7 ± 126.7**	**105.7 ± 4.4**	**1429.2 ± 40.1**	**-934.4 ± 441.6**	**2521.1 ± 860.9**
Ours-VAE	$\pm$	$\pm$	$\pm$	$\pm$	$\pm$	$\pm$

Figure 3 depicts the original rewards for different methods on noisy expert data with Gaussian noise $(0, 0.3)$ on every task except for Humanoid which with $(0, 0.01)$ Gaussian noise. Table 3 shows the final learned policy performance on MuJoCo tasks with raw episode return.

These results show that our method outperforms other state-of-the-art algorithms almost on all tasks except for Hopper which PWIL wins. And for the top 3 results, This definitely shows that our AEIRL can effectively denosie the expert data with the reconstruction period.

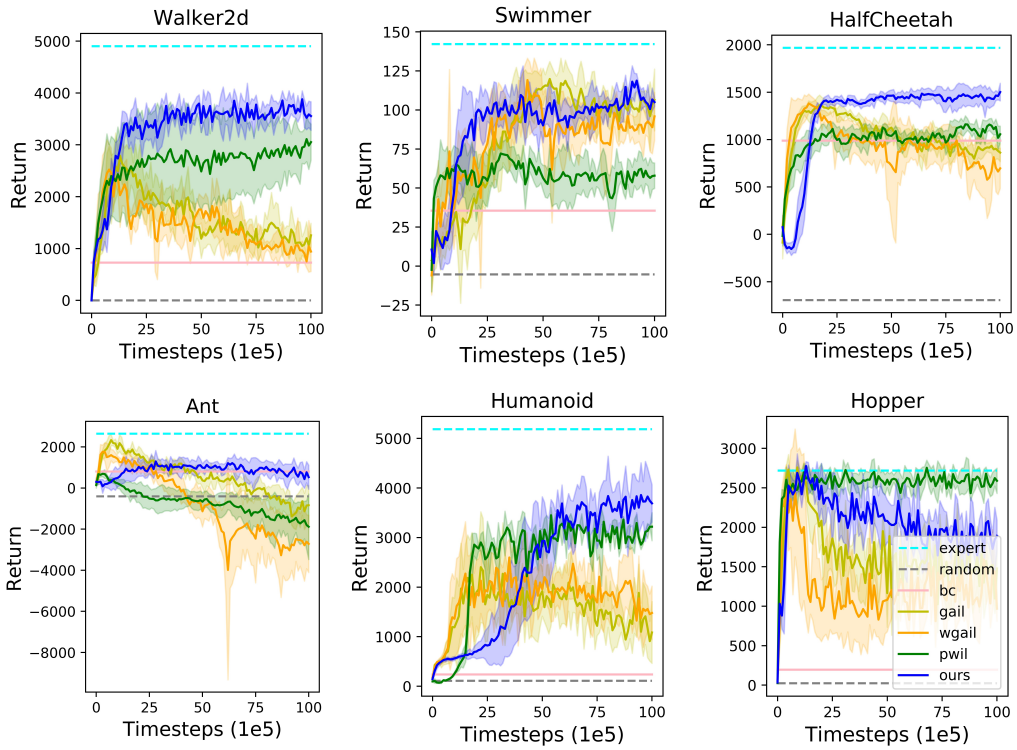


Figure 3: Mean and standard deviation return of the evaluation policy over 10 rollouts and 5 seeds learning from noisy expert demonstrations with Gaussian noise $(0, 0.3)$ on every dimension of state action pairs for Walker, Hopper, Swimmer, HalfCheetah, and Ant, with Gaussian noise $(0, 0.01)$ for Humanoid, reported every 100k timesteps. The return is in term of the environment's original reward.

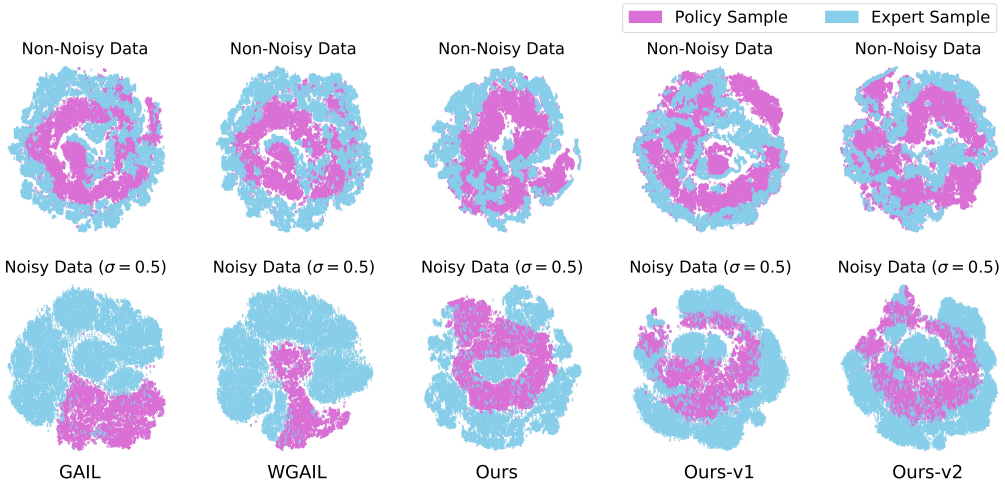


Figure 4: t-SNE visulization of latent representations of the reward function on locomotion environment Walker2d for different methods. Top row: non-noisy expert data, bottom row: noisy expert data with Gaussian noise $(0, 0.5)$ on every dimension of state-action pairs.

5.3 More Informative Reward Function and More Stable Training Procedure

We use t-SNE visulization to show the latent representations of the auto-encoder for our methods and the discriminator for baselines. The auto-encoder or the discriminator is using two hidden layers of 100 units each, with tanh nonlinearities in between, and the first layer is using normalization and the last layer is using sigmoid for GAIL and WGAIL. So the latent representations is actually the output of the second layer.

Figure 4 depicts that the latent representations are more indistinguishable between policy samples and expert samples for our methods than GAIL and WGAIL. And under the noisy expert, the latent representations of the samples becomes two clusters while our methods still learns the expert sample distributions.

We train PPO on Walker2d to sample 50 different levels of trajectories which can be seen as the trajectories in the whole trajectory space.

Figure 5 shows that the final recovered rewards of the state action pairs for different levels of the trajectories. And the curve depicts that our method is more informative than GAIL and WGAIL with higher degree of distinction for normalized learned rewards.

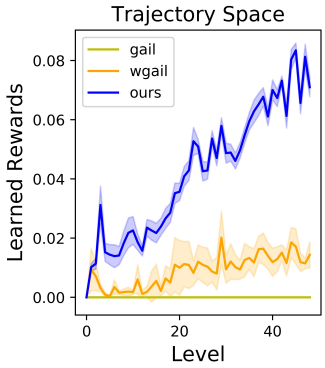


Figure 5: More Informative Reward Function, normalized learned rewards for different levels in the trajectory space on Walker2d.

To show the training stability of our AEIRL, we record the norm of weights for policy network and the variance of policy gradients. Consider that the relative variance of policy gradients shows the training stability for different methods. Figure 6 shows that our method get lower relative variance of gradients which means that the network updating can be smoother and more stable.

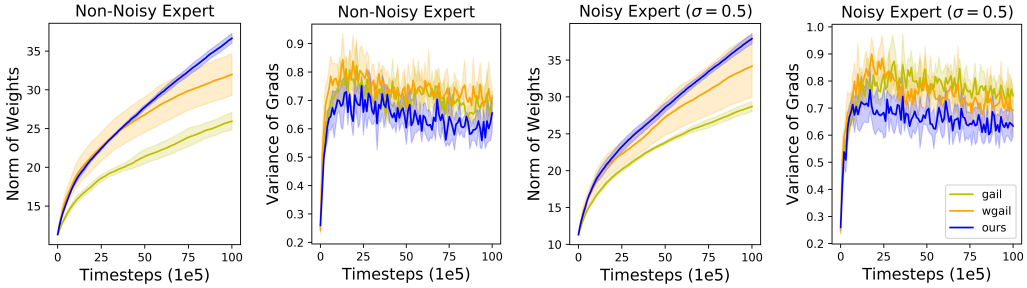


Figure 6: More Stable Training Procedure, norm of weights for the policy network and corresponding variance of policy gradients on Walker2d for non-noisy and noisy expert data, reported every 100k timesteps.

6 Conclusions

In our paper, we present a simplified and robust adversarial imitation learning method based on AutoEncoding (AEIRL). We structured the reward function with the reconstruction error of an auto-encoder. The reward signal is more robust and informative compared with existing methods. And further more, AEIRL could help denoise the expert demonstrations with the reconstruction period of training the reward function. Experiment results show that our proposed methods achieve strong competitive performance on locomotion tasks even from imperfect (noisy) demonstrations. We hope that our work can further be applied into more realistic scenarios such as autonomous driving systems and so on.

7 REPRODUCIBILITY STATEMENT

Code and data are attached in supplementary materials for reproducing the results, more details on experimental implementations are shown in Appendix A.2.

REFERENCES

Pieter Abbeel, Varun Ganapathi, and Andrew Y. Ng. A learning vehicular dynamics, with application to modeling helicopters. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2006.

Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2007.

Pieter Abbeel, Adam Coates, Timothy Hunter, and Andrew Y. Ng. Autonomous autorotation of an rc helicopter. In *International Symposium on Robotics*, 2008a.

Pieter Abbeel, Dmitri Dolov, Andrew Y. Ng, and Sebastian Thrun. Apprenticeship learning for motion planning with application to parking lot navigation. In *IEEE/RSj International Conference on Intelligent Robots and Systems*, 2008b.

Pieter Abbeel, Adam Coates, and Andrew Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 2010.

Martin Arjovsky, Soumith Chintala, and Leon Bottou. Wasserstein gan. In *International Conference on Machine Learning (ICML)*, 2017.

Enda Barrett and Stephen Linder. Autonomous hva control, a reinforcement learning approach. *Machine Learning and Knowledge Discovery in Databases*, 2015.

Xin-Qiang Cai, Yao-Xiang Ding, Yuan Jiang, and Zhi-Hua Zhou. Imitation learning from pixel-level demonstrations by hashreward. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2021.

Adam Coates, Pieter Abbeel, and Andrew Y. Ng. Learning for control from multiple demonstrations. In *International Conference on Machine Learning (ICML)*, 2008.

Robert Dadashi, Leonard Hussenot, Matthieu Geist, and Olivier Pietquin. Primal wasserstein imitation learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines, 2017.

Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *Deep Reinforcement Learning Workshop on NeurIPS*, 2016a.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning (ICML)*, pp. 49–58, 2016b.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2017.

Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning (CoRL)*, pp. 1259–1277. PMLR, 2020.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, pp. 1861–1870. PMLR, 2018.

Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2944–2952, 2015.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Daniel Jarrett, Ioana Bica, and Mihaela van der Schaar. Strictly batch imitation learning by energy-based distribution matching. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Ming Jin, Andreas Damianou, Pieter Abbeel, and Costas Spanos. Inverse reinforcement learning via deep gaussian process. In *International Conference on Machine Learning (ICML)*, 2015.

Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning (ICML)*, pp. 1–9, 2013.

Sergey Levine and Zoran Popvić. Feature construction for inverse reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

Minghuan Liu, Tairan He, Minkai Xu, and Weinan Zhang. Energy-based imitation learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2021.

Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2000.

Andrew Y. Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Inverted autonomous helicopter flight via reinforcement learning. In *International Symposium on Experimental Robotics*, 2004.

Ahmed H. Qureshi, Byron Boots, and Michael C. Yip. Adversarial imitation via variational inverse reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. Maximum margin planning. In *International Conference on Machine Learning (ICML)*, 2006.

Saeed Saremi, Arash Mehrjou, Bernhard Schölkopf, and Aapo Hyvärinen. Deep energy estimator networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Fumihiro Sasaki and Ryota Yamashina. Behavioral cloning from noisy demonstrations. In *International Conference on Learning Representations (ICLR)*, 2021.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning (ICML)*, pp. 1889–1897, 2015a.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, pp. 1889–1897. PMLR, 2015b.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

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

Umar Syed, Michael Bowling, and Robert E. Schapire. Apprenticeship learning using linear programming. In *International Conference on Machine Learning (ICML)*, 2008.

Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913. IEEE, 2012.

Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. Imitation learning from imperfect demonstration. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 6818–6827, 2019.

Xin Zhang, Yanhua Li, Ziming Zhang, and Zhi-Li Zhang. f-gail: Learning f-divergence for generative adversarial imitation learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.

Junbo Zhao, Machael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *International Conference on Learning Representations (ICLR)*, 2016.

Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2008.

Brian D. Ziebart, J. Andrew Bagnell, and Anind K. Dey. Modeling interaction via the principle of maximum causal entropy. In *International Conference on Machine Learning (ICML)*, 2010.

A APPENDIX

A.1 PROOF FOR THEOREM 1

Proof. The bi-level optimization problem for our method is discussed before:

$$\begin{aligned} &\min_{w} \int_X r(x, \tilde{x}_w)\rho_{\pi_\theta}(x) - r(x, \tilde{x}_w)\rho_{\pi_E}(x)dx \\ &\quad \text{s.t.} \quad w \in W \\ &\max_{\theta} \int_X r(x, \tilde{x}_w)\rho_{\pi_\theta}(x)dx + \lambda\mathcal{H}(\pi(\cdot|s_t)) \\ &\quad\quad \text{s.t.} \quad \theta \in \Theta, \end{aligned} \tag{14}$$

where W and Θ denote the parameter space of the auto-encoder and the policy respectively, $\mathcal{H}$ denote the entropy of policy.

We define quantity $V(\pi, r)$ as:

$$V(\pi, r) = \int_X r(x, \tilde{x}_w)\rho_{\pi_\theta}(x) - r(x, \tilde{x}_w)\rho_{\pi_E}(x)dx, \tag{15}$$

and quantity $Q(\pi, r)$ as:

$$Q(\pi, r) = \int_X r(x, \tilde{x}_w)\rho_{\pi_\theta}(x)dx + \lambda\mathcal{H}(\pi(\cdot|s_t)), \tag{16}$$

and quantity $U(\pi, r)$ as:

$$U(\pi, r) = \int_X r(x, \tilde{x}_w)\rho_{\pi_\theta}(x)dx \tag{17}$$

where quantity U is the lower bound of the quantity Q. In our experiments and proofs, we optimize U as the reinforcement learning objective.

Solving the bi-level optimization problem 14 is actually finding a Nash equilibrium between the two players in this game, policy π and reward function r.

A Nash equilibrium of system 14 is a pair (π^*, r^*) which satisfies:

$$\begin{aligned} V(\pi^*, r^*) &\leq V(\pi^*, r) \qquad \forall r \\ U(\pi^*, r^*) &\geq U(\pi, r^*) \qquad \forall \pi \end{aligned} \tag{18}$$

Due to that $r(x, \tilde{x}_w) = 1/(1 + \|x - \tilde{x}_w\|_2^2)$, the reward is bounded in $[0, 1]$.

Now we prove that the reward function r would converge to a constant by proof of contradiction once the system reaches a Nash equilibrium. We assume that r is not constant, then there exists a constant C and a set $\mathcal{S}$ of non-zero measure such that $\forall x \in \mathcal{S}, r \leq C$ and $\forall x \notin \mathcal{S}, r > C$.

Suppose Assumption 1 holds, then we have $\rho_{\pi^*}(x) > 0$ on $\mathcal{S}$. We can build a policy whose occupancy measure is ρ_0 such that $\rho_{\pi_0}(x) < \rho_{\pi^*}(x)$ over $\mathcal{S}$. Then we compute

$$U(\pi^*, r^*) - U(\pi_0, r^*) = \int_X (\rho_{\pi^*}(x) - \rho_{\pi_0}(x)) \cdot r^*(x, \tilde{x}_w)dx \tag{19}$$

Considered that $\rho_{\pi^*}(x)$ and $\rho_{\pi_0}(x)$ are probability density functions, so we can hold that:

$$\int_X (\rho_{\pi^*}(x) - \rho_{\pi_0}(x)) \cdot Cdx = C \cdot \int_X \rho_{\pi^*}(x)dx - C \cdot \int_X \rho_{\pi_0}(x)dx = C \cdot 1 - C \cdot 1 = 0 \tag{20}$$

So by equation 19 and equation 20, we have:

$$U(\pi^*, r^*) - U(\pi_0, r^*) = \int_X (\rho_{\pi^*}(x) - \rho_{\pi_0}(x))(r^*(x, \tilde{x}_w) - C)dx \tag{21}$$

So we can obtain that:

$$U(\pi^*, r^*) - U(\pi_0, r^*) = \int_{\mathcal{S}} (\rho_{\pi^*}(x) - \rho_{\pi_0}(x))(r^*(x, \tilde{x}_w) - C)dx + \int_{X \setminus \mathcal{S}} (\rho_{\pi^*}(x) - \rho_{\pi_0}(x))(r^*(x, \tilde{x}_w) - C)dx \tag{22}$$

Here we know the quantity $\rho_{\pi^*}(x) - \rho_{\pi_0}(x) \geq 0$ and $r^*(x, \tilde{x}_w) - C \leq 0$ for $x \in \mathcal{S}$, and the quantity $\rho_{\pi^*}(x) - \rho_{\pi_0}(x) < 0$ and $r^*(x, \tilde{x}_w) - C > 0$ for $x \in X \setminus \mathcal{S}$.

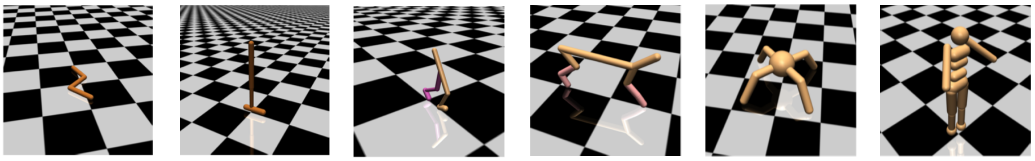
So for equation 22, i.e., $U(\pi^*, r^*) - U(\pi_0, r^*) < 0$, which violates equation 18.

So the reward function of our method could converge to a constant η in [0, 1] when the game for bi-level optimization problem reaches a Nash equilibrium.

This completes the proof. □

A.2 Implementation

A.2.1 Tasks



(a) (b) (c) (d) (e) (f)

Figure 7: Illustrations for locomotion tasks we used in our experiments: (a) Swimmer; (b) Hopper; (c) Walker; (d) Half-Cheetah; (e) Ant; (f) Simple Humanoid.

The goal for all these tasks is to move forward as quickly as possible. These tasks are more challenging than the basic tasks due to high degrees of freedom. In addition, a great amount of exploration is needed to learn to move forward without getting stuck at local optima. Since we penalize for excessive controls as well as falling over, during the initial stage of learning, when the robot is not yet able to move forward for a sufficient distance without falling, apparent local optima exist including staying at the origin or diving forward slowly. Figure 7 depicts locomotion tasks' environments.

A.2.2 Basic Settings

On Walker2d, Hopper, and Swimmer, we train these algorithms directly, while on HalfCheetah, Ant and Humanoid we use BC to pre-train the policy with 10k iterations. We test all the algorithms on these six locomotion tasks with environment seed $0, 1, 2, 3, 4$, and using 10 rollouts to estimates the original reward value to plot the training curves.

The maximum length for sampled trajectories is 1024. The discounted factor is 0.995. Each iteration we update 3 times of policies and 1 time of reward function (discriminator or auto-encoder). While updating TRPO, we update the critic 5 times with learning rate 2e-4 every iteration.

A.2.3 Baselines

We use open sourced code, OpenAI baselines (Dhariwal et al., 2017), to get the performance of BC and the training iterations is 100k for all tasks.

We implement GAIL using open sourced code, OpenAI baselines (Dhariwal et al., 2017), with best tunned hyperparameters. We choose TRPO as the reinforcement learning algorithm.

WGAIL is implemented based on GAIL with new loss function and reward signals as in (Arjovsky et al., 2017). We choose TRPO as the reinforcement learning algorithm.

PWIL is implemented based on open sourced code (Dadashi et al., 2021) with TRPO as the reinforcement learning step for fair comparison (original PWIL code uses SAC as the RL algorithm).

A.2.4 OUR METHODS

Policy Net: the same with baselines, two hidden layers with 64 units, with tanh nonlinearities in between. The learning rate to update the policy is 0.01.

Auto-Encoder Net and VAE Net: the same number of layers with the architecture of discriminator in GAIL and WGAIL, 4 layers. The two hideen layers are with 100 units, with tanh nonlinearities in between, the final output layer is identity. And we normalize the state feature after the input layer and we use the raw state action features to compute the mean square error. Learning rate for the auto-encoder is 3e-4.

We doesn't add the entropy term to the reinforcement learning step but already get great performance. So fine-tuning the hyper-parameters might have better performance on these tasks.

A.2.5 TOY EXPERIMENTS

We design a 1D walking task as toy experiments. The state space is discrete with 40 states and the action is 0 walking left one step or 1 walking right one step. The initial state is sampled by uniform distribution between state 10 to 29. The maximum length of trajectory is 10.

We set the expert policy is always walking right and sample 100 trajectories. Every algorithms will be trained 3 times with random initialization of the policy, reward function and critic network. The timesteps for training is set to 100k.

The reward function is structured with 4 dense layers with relu nonlinerities in between. The policy net is structured with 4 dense layers with 20 units with tanh nonlinerities in between. We use PPO to train the policy as the reinforcement learning step.

A.3 ABLATION STUDIES

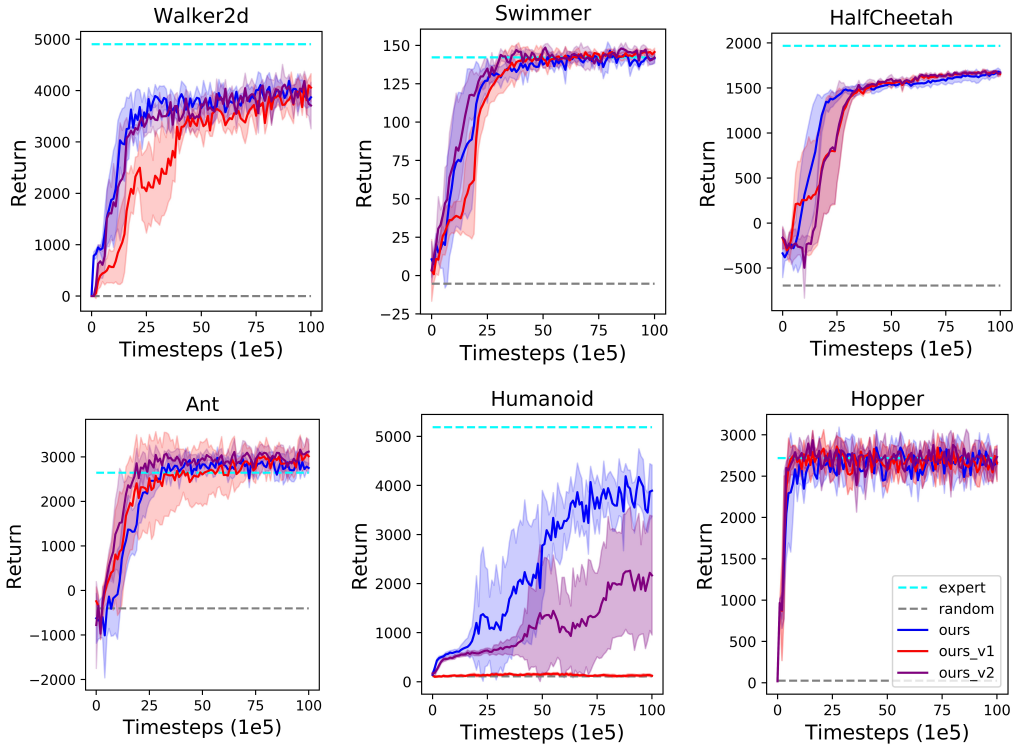


Figure 8: Mean and standard deviation return of the evaluation policy over 10 rollouts and 5 seeds learning from non-noisy expert demonstrations, reported every 100k timesteps. The return is in term of the environment's original reward.

Figure 8 depicts the comparison for different variants of our methods with different loss functions on non-noisy expert demonstrations. Clearly, Our methods with different loss function and reward signal formulation have comparable results on Walker2d, Swimmer, HalfCheetah, Hopper, and Ant. But on Humanoid, the Wasserstein-1 distance achieves better performance than JS distance based loss functions.

A.4 AUTO-ENCODER CONVERGENCE

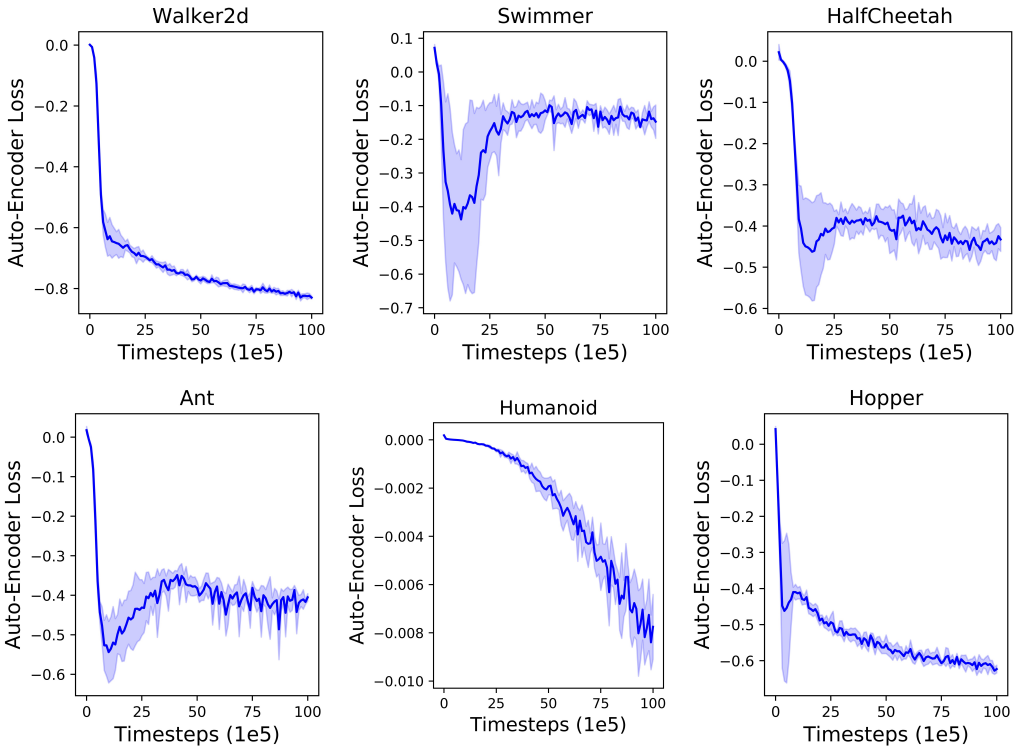


Figure 9: Mean and standard deviation loss of training the auto-encoder for our method AEIRL on non-noisy expert demonstrations.

Figure 9 depicts the training curve of loss function for the auto-encoder which shows the convergence of the auto-encoder on non-noisy expert demonstrations.

A.5 MORE EXPERIMENTS ON NOISY EXPERT DATA

Figure 10 shows that the training curve for different methods on more noisy expert data. Clearly, our methods get higher performance than other baselines on almost all tasks except for Hopper which is not better than PWIL. With the greater noise, we can see that our methods can effectively denoise the expert data while GAIL and WGAIL cannot. Additionally, PWIL also has some capability to denoise the expert data with a fixed reward function.

Figure 11 shows that t-SNE visualization results for the latent space of reward functions on Walker2d and non-noisy and different noisy expert data.

The indistinguishable policy samples and expert samples means better reward recovering. Clearly, when the noise becomes greater, GAIL and WGAIL fail to learn the reward distribution and two sets of samples become separate clusters. However, our methods still recover the reward function efficiently except for strong noisy expert.

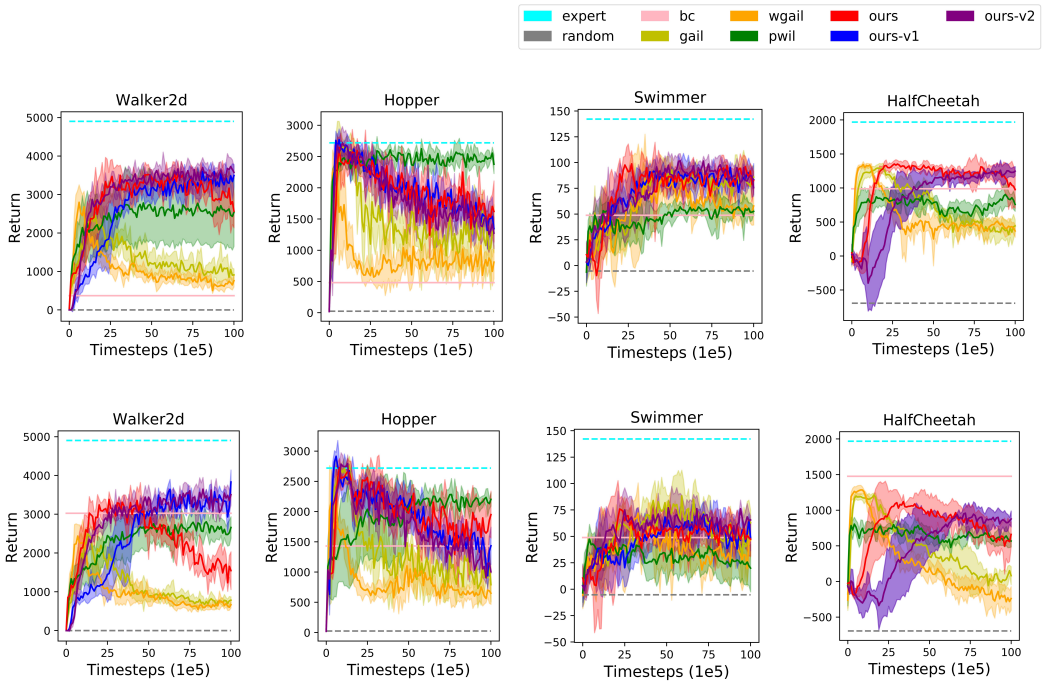


Figure 10: Mean and standard deviation return of the evaluation policy over 10 rollouts and 5 seeds learning from noisy expert demonstrations on every dimension of state action pairs, reported every 100k timesteps. The return is in term of the environment's original reward. Top line: Gaussian noise $(0, 0.5)$, bottom line: Gaussian noise $(0, 0.8)$

A.6 One Shot Imitation Learning on Non-Noisy Expert Data

A.7 Toy Experiments

Note that the reward function tags used in experiments for our methods are shown in Table 4.

Table 4: Reward functions for different methods

Method	Reward Function
GAIL	$-log(1 - D(s, a))$
WGAIL	$D(s, a)$
AIRL	$log(D(s, a)) - log(1 - D(s, a))$
Ours-v1	$-log(1 - exp(-AE(s, a)))$
Ours-v2	$1/(1 + AE(s, a))$
Ours-v3	$log(1 - \text{sigmoid}(-AE(s, a)))$

Figure 12 shows the KL divergence between the learned policy and the expert policy. Our methods learn the policies quite close to the expert with the KL divergence close to 0.

Figure 13 shows the heatmaps for recovered reward functions for the entire state action space. The reward function of our methods are more informative than other baselines as shown in the figure.

Figure 11: t-SNE visulization of latent representations of the reward function on locomotion environment Walker2d on non-noisy expert data and different noisy expert data for different methods.

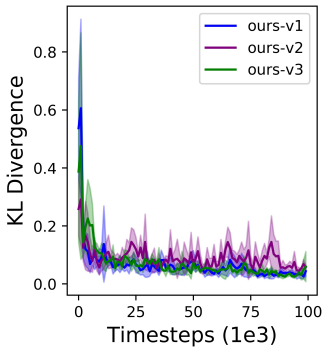


Figure 12: Training curve of KL divergence between the learned policy and the ground truth policy for different reward function optimization loss functions for our method AEIRL on toy experiments, reported every 1k timesteps.

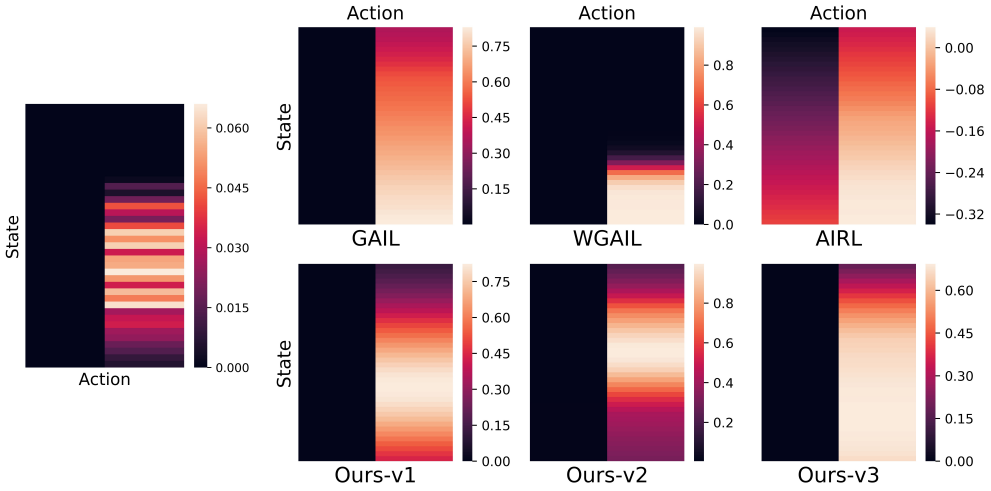


Figure 13: Heatmaps for the recovered reward function with different adversarial imitation learning methods on toy experiments. Left: data distribution of expert demonstrations, Rights: recovered reward functions.