AUTO-ENCODING ADVERSARIAL IMITATION 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. Adversarial Imitation Learning (AIL) sheds light on automatic policy acquisition without access to the reward signal from the environment. In this work, we propose Auto-Encoding Adversarial Imitation Learning (AEAIL), a robust and scalable AIL framework. To induce expert policies from demonstrations, AEAIL utilizes the reconstruction error of an auto-encoder as a reward signal, which provides more information for optimizing policies than the prior discriminator-based ones. Subsequently, we use the derived objective functions to train the auto-encoder and the agent policy. Experiments show that our AEAIL performs superior compared to state-of-the-art methods in the MuJoCo environments. More importantly, AEAIL shows much better robustness when the expert demonstrations are noisy. Specifically, our method achieves 11% and 50.7% relative improvement overall compared to the best baseline GAIL and PWIL on clean and noisy expert data, respectively. Video results, open-source code and dataset are available in supplementary materials.

1 INTRODUCTION

Reinforcement learning (RL) provides a powerful framework for automated decision-making. However, RL still requires significantly engineered reward functions for good practical performance. Imitation learning offers the instruments to learn policies directly from the demonstrations, without an explicit reward function. It 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 induce the expert policies from expert demonstrations without access to the reward signal from the environment. We divide these methods into two broad categories: Behavioral Cloning (BC) and Inverse Reinforcement Learning (IRL). Among IRL, adversarial imitation learning (AIL) induces expert policies by minimizing the distribution distance between expert samples and agent policy rollouts. Prior AIL methods model the reward function as a discriminator to learn the mapping from the state-action pair to a scalar value, i.e., reward (Ho & Ermon, 2016; Zhang et al., 2020; Fu et al., 2017). However, the discriminator in the AIL framework would easily find the differences between expert samples and agent-generated ones, even though some differences are minor. Therefore, the discriminator-based reward function would yield a sparse reward signal to the agent. Consequently, how to make AIL robust and efficient to use is still subject to research.

Our AEAIL is an instance of AIL by formulating the reward function as an auto-encoder. Since auto-encoder reconstruct the full state, unlike traditional discriminator based AIL, our method will not overfit to the minor differences between expert samples and generated samples. In many cases, our reward signal provides richer feedback to the policy training process. Thus, our new method achieves better performance on a wide range of tasks.

Our contributions are three-fold:

- We propose the Auto-Encoding Adversarial Imitation Learning (AEAIL) architecture. It models the reward function as the reconstruction error of an auto-encoder. It focuses on the full-scale differences between the expert and generated samples, and less susceptible to the discriminator attending to minor differences.
- Experiments show that our proposed AEAIL achieves the best overall performance on many environments compared to other state-of-the-art baselines.
- Empirically, we justify that our AEAIL works under a wide range of distribution divergences and auto-encoders. And the major contributing factor of our method is the encoding-decoding process rather than the specific divergence or auto-encoder.

2 Related work

2.1 Imitation learning

Adversarial Imitation Learning (AIL) AIL methods such as GAIL, DAC, f-GAIL, EAIRL, FAIRL (Ho & Ermon, 2016; Kostrikov et al., 2019; Zhang et al., 2020; Qureshi et al., 2019; Ghasemipour et al., 2020) formulates the learned reward function as a discriminator that learns to differentiate expert transitions from non-expert ones. Among these methods, GAIL (Ho & Ermon, 2016) considers the Jensen-Shannon divergence. DAC (Kostrikov et al., 2019) extends GAIL to the off-policy setting and significantly improves the sample efficiency of adversarial imitation learning. Furthermore, f-divergence is utilized in f-GAIL (Zhang et al., 2020), which is considered improving sample-efficiency. Recently, FAIRL utilizes the forward KL divergence (Ghasemipour et al., 2020) and achieves great performance, but it is still not robust and efficient enough. These methods rely heavily on a carefully tuned discriminator-based reward function. It might focus on the minor differences between the expert and the generated samples and thus gives a sparse reward signal to the agent. In comparison, our auto-encoder-based reward function learns the full-scale differences between the expert and generated samples.

Inverse Reinforcement Learning (IRL) IRL learns a reward function from the expert demonstrations, which assumes the expert performs optimal behaviors. One category of IRL uses a hand-crafted similarity function to estimate the rewards (Boularias et al., 2011; Klein et al., 2013; Piot et al., 2016). The idea of these methods is still inducing the expert policy by minimizing the distance between the state action distributions of the expert and sampled trajectories. Primal Wasserstein Imitation Learning (PWIL) (Dadashi et al., 2021) is one such algorithm, which utilizes the upper bound of Wasserstein distance's primal form as the optimization objective. The advantage of these methods is that they are relatively more robust and less demonstration dependent than AIL methods. However, the performance of these methods heavily depends on the similarity measurement, and therefore, it varies greatly on different tasks. Compared to PWIL, our method achieves superior performance via automatically acquiring a similarity measurement. Random Expert Distillation (RED) (Wang et al., 2019) uses random network distillation to compute the rewards for imitation learning. The error between the predictor and the random target network could guide the policy to mimic the expert behaviors. Meanwhile, auto-encoder could be used in place of the random network distillation. In comparison, RED uses a fixed auto-encoder while our AEAIL utilizes the AE in an adversarial manner. ValueDice (Kostrikov et al., 2020) utilizes a critic based reward function which is optimized during training the policy and critic. However, the performance of ValueDice is sensitive to the expert data used. IQ-Learn (Garg et al., 2021) is a dynamics-aware IL method which avoids adversarial training by learning a single Q-function, implicitly representing both the reward and the policy.

2.2 Imitation Learning with Imperfection

The robustness of adversarial imitation learning is questionable with imperfection in observations (Stadie et al., 2017; Berseth & Pal, 2020), actions, transition models (Gangwani & Peng, 2020; Christiano et al., 2016), expert demonstrations (Brown et al., 2019; Shiarlis et al., 2016; Jing et al., 2020) and their combination (Kim et al., 2020). Prior robust imitation learning methods require the demonstrations to be annotated with confidence scores (Wu et al., 2019; Brown et al., 2019; Grollman & Billard, 2012). However, these annotations are rather expensive. Compared to them, our auto-encoder-based reward function won't easily overfit to the noisy feature. Therefore, our AEAIL

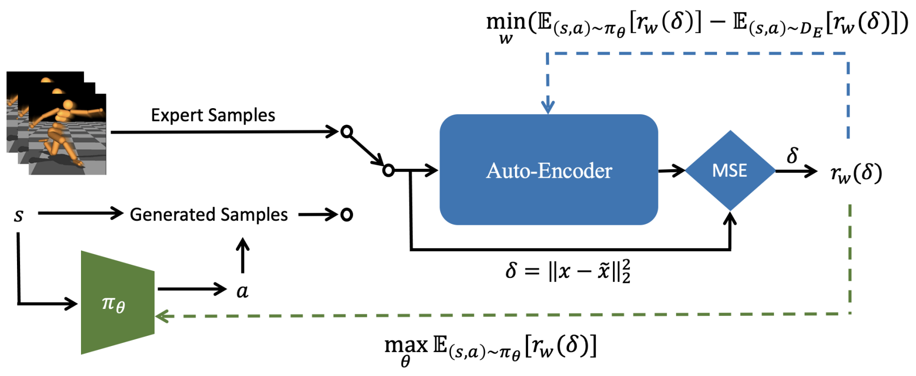
$\min_w(\mathbb{E}_{(s,a)\sim\pi_\theta}[r_w(\delta)] - \mathbb{E}_{(s,a)\sim D_E}[r_w(\delta)])$
Expert Samples
Auto-Encoder
MSE
δ
$r_w(\delta)$
s
Generated Samples
$\delta = \|x - \tilde{x}\|_2^2$
π_θ
a
$\max_\theta \mathbb{E}_{(s,a)\sim\pi_\theta}[r_w(\delta)]$

Figure 1: Training framework of our AEAIL. The auto-encoder computes the reconstruction error for these two mini-batches of data examples and optimizes the derived objectives. Therefore, a surrogate reward function $r_w(\delta)$ provides the signal to the agent. The agent can induce the expert policies by iteratively training the auto-encoder and the agent policy.

is relatively straightforward and robust to the noisy expert data and does not require any annotated data.

2.3 Auto-Encoding based GANs

Auto-encoders have been successfully applied to improve the training stability and modes capturing in GANs. We classify auto-encoding based GANs into three categories: (1) utilizing an auto-encoder as the discriminator such as energy-based GANs (Zhao et al., 2016) and boundary-equilibrium GANs (Berthelot et al., 2017); (2) using a denoising auto-encoder to derive an auxiliary loss for the generator (Warde-Farley & Bengio, 2017); (3) combining variational auto-encoder and GANs to generate both vivid and diverse samples by balancing the objective of reconstructing the training data and confusing the discriminator (Larsen et al., 2016). Our method AEAIL takes inspirations from EBGAN (Zhao et al., 2016), utilizing the auto-encoder in the reward function.

3 Background

Inverse Reinforcement Learning (IRL) IRL extracts the policy towards mimicking expert behaviors with the help of a learned reward function. On the other hand, AIL directly induces the expert policies by minimizing the distribution distance between expert demonstrations and agent-generated samples.

Adversarial Imitation Learning (AIL) Assume that we are given a set of expert demonstrations, AIL involves a discriminator which tries to distinguish the expert and policy generated samples and a policy that is optimized to confuse the discriminator accordingly. Ideally, the rollouts from the agent will be indistinguishable from the expert demonstrations in the end. The discriminator cannot classify the expert samples and policy-generated ones any more at the end of the training process. Consequently, this kind of discriminator-based reward function can be seen as a pseudo-reward signal, which is only used for optimizing policies during training.

Wasserstein Divergence One popular distribution divergence for measuring the distance between policy samples and expert samples is the Wasserstein divergence (Arjovsky et al., 2017):

$$d(p, q) = \sup_{|\phi|_L \leq 1} \mathbb{E}_{x\sim p}[\phi(x)] - \mathbb{E}_{y\sim q}[\phi(y)], \tag{1}$$

where p and q are two probability distributions, ϕ is some function that is 1-Lipschitz continuous. Equation 1 measures the distance between these two distributions. Intuitively, we can view it as the minimum cost of turning the piles from distribution p into the holes in distribution q.

4 METHODOLOGY

4.1 OVERVIEW

The reward function in AIL methods is a discriminator, which attempts to assign high values to the regions near the expert demonstrations and assign low values to the other areas. However, the discriminator would easily overfit to the minor differences between expert data and policy-generated samples. Consider the CartPole balancing task, the goal of this task is to keep the pole balancing via moving the cart left or right. The state of the environment is a feature consisting of position, angle, angle's rate, and cart velocity while the action is moving left or right. Here, we assume that all the expert states' velocity is 2, for example. When the generated states' velocity is 1.9, and other dimensions of state-action pair are the same as the expert's, the discriminator of GAIL would still give a low reward on these generated state-action samples. However, the policy rollouts may perform very well towards mimicking the expert's behaviors. In other words, the reward function in GAIL on this example overfits to the minor differences between the expert and the sampled data and leads the policy to miss the underlying goal of the expert.

In this paper, we propose an auto-encoder-based reward function for adversarial imitation learning. It utilizes the reconstruction error of the state action pairs to compute the reward function. The reconstruction error based reward signal force the discriminator to consider all information in the state-action tuple, rather than focusing on the minor differences. Therefore, this reward signal wouldn't lead to overconfidence in distinguishing the expert and the generated samples. Recall the CartPole example, the mean square error between states' velocity 1.9 and 2 is very small. And it could still feed a good reward to the agent under this situation. Therefore, intuitively our proposed reward signal can better focus on the full-scale differences between the expert and generated state action pairs, yielding more informative rewards and thus improving the policy learning.

4.2 OUR METHOD

Our approach is to minimize the distance between the state action distribution of the policy π_θ and that of the expert demonstrations D_E.

The objective formulation we used in our method is Ho et al. (2016):

$$d(\pi_E, \pi_\theta) = \sup_{|r_w|_L \leq K} \mathbb{E}_{\pi_E}[r_w(s,a)] - \mathbb{E}_{\pi_\theta}[r_w(s,a)], \tag{2}$$

where the reward function network's parameters are denoted as w and the policy network's parameters are represented as θ. Minimizing this distance will induce the expert policy from expert demonstrations. Therefore, the optimization of the policy π_θ and the reward function $r_w(s,a)$ forms a bi-level optimization problem, which can be formally defined as:

$$\min_{\pi_\theta} \max_{r_w} \left(\mathbb{E}_{(s,a)\sim D_E}[r_w(s,a)] - \mathbb{E}_{(s,a)\sim \pi_\theta}[r_w(s,a)] \right). \tag{3}$$

Optimizing Equation 3 leads to an adversarial formulation for imitation learning. The outer level minimization with respect to the policy leads to a learned policy that is close to the expert. The inner level maximization recovers a reward function that attributes higher values to regions close to the expert data and penalizes all other areas.

In our method, we use an auto-encoder based surrogate pseudo-reward function instead, which is defined as:

$$r_w(s,a) = 1/(1 + \text{AE}_w(s,a)), \tag{4}$$

where AE is the reconstruction error of an auto-encoder:

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

Here, x represents the state-action pairs. Equation 5 is the mean square error between the sampled state-action pairs and the reconstructed samples. This form of the reward signal uses the reconstruction error of an auto-encoder to score the state-action pairs in the trajectories. Equation 4 is a monotonically decreasing function over the reconstruction error of the auto-encoder. We give high rewards to the state-action pairs with low reconstruction errors of the auto-encoding process and vice versa. Section

4.1 motivates that this form of reward signal focuses more on the full-scale differences between the expert and generated samples, and it won't easily overfit to the noise of the expert data.

Training the auto-encoder is an adversarial process considering the objective 3, which is minimizing the reconstruction error for the expert samples while maximizing this error for generated samples. When combining Equation 3 and Equation 4, we can obtain the training objective for the auto-encoder as:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{(s,a)\sim\pi_\theta}[r_w(s,a)] - \mathbb{E}_{(s,a)\sim D_E}[r_w(s,a)] \\ &= \mathbb{E}_{(s,a)\sim\pi_\theta}[1/(1+\text{AE}_w(s,a))] - \mathbb{E}_{(s,a)\sim D_E}[1/(1+\text{AE}_w(s,a))].\end{aligned} \tag{6}$$

With this adversarial objective, the auto-encoder learns to maximize the full-scale differences between the expert and the generated samples. As a result, it gives the agent a denser reward signal. Furthermore, the agent can also be more robust when facing noisy expert data due to the robust auto-encoding objective.

Figure 1 depicts the architecture for AEAIL. The auto-encoder-based reward function takes either expert or the generated state-action pairs and estimates the reconstruction error based rewards accordingly. We iteratively optimize the auto-encoder and the agent policy under this adversarial training paradigm. Algorithm 1 in Appendix A.1 shows the pseudo-code for training AEAIL.

4.3 Theoretical properties

Our AEAIL also shares a similar theoretical framework with GAIL, where the auto-encoder based reward function can be seen as a generalized discriminator. Guan et al. (2021) investigates the theoretical properties of GAIL. We make the same set of assumptions for our auto-encoder based reward function as these for the discriminator in GAIL (Guan et al., 2021):

(i) the auto-encoder based reward function has bounded gradient, i.e., $\exists C_r \in \mathbb{R}$, such that

$$\|\nabla_w r_w\|_{\infty,2} := \sqrt{\sum_i \|\frac{\partial r_w}{\partial w_i}\|_\infty^2} \leq C_r$$

(ii) the auto-encoder based reward function has L-Lipschitz continuous gradient, i.e., $\exists L_r \in \mathbb{R}$, such that $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}, \forall w_1, w_2 \in \mathcal{W}$, then:

$$\|\nabla_w r_{w_1}(s,a) - \nabla_w r_{w_2}(s,a)\|_2 \leq L_r \|w_1 - w_2\|_2$$

(iii) the distribution divergence

$$d_{\mathcal{F}}(P_r, P_\theta) = \sup_{r \in \mathcal{R}} \mathbb{E}_{x\sim P_r}[r(x)] - \mathbb{E}_{x\sim P_\theta}[r(x)] + \phi(r(x))$$

is properly regularized to be strongly concave with respect to the reward function. Here, $\phi(r(x))$ is the regularizer. We prove that the reward function in our AEAIL is K-Lipschitz with clipping the weights of the auto-encoder. (Proof see in Appendix A.2).

These conditions are assumptions for theoretical convenience since they don't always hold for a multi-layer neural network-based discriminator in GAIL or our proposed auto-encoder. When these assumptions hold, our AEAIL shares the same theoretical properties as GAIL: it converges globally.

5 Experiments

Environments: we conduct the experiments on six locomotion tasks with varying dynamics and difficulties: Swimmer-v2 (Schulman et al., 2015), Hopper-v2 (Levine & Koltun, 2013), Walker2d-v2 (Schulman et al., 2015), HalfCheetah-v2 (Heess et al., 2015), Ant-v2 (Schulman et al., 2016), and Humanoid-v2 (Tassa et al., 2012).

Baselines: the baselines we choose for comparison includes: GAIL (Ho & Ermon, 2016), FAIRL (Ghasemipour et al., 2020), PWIL (Dadashi et al., 2021) and DAC (Kostrikov et al., 2019).

Implementation: experimental details, including the implementation and more results, are shown in Appendix A.3 to A.5.

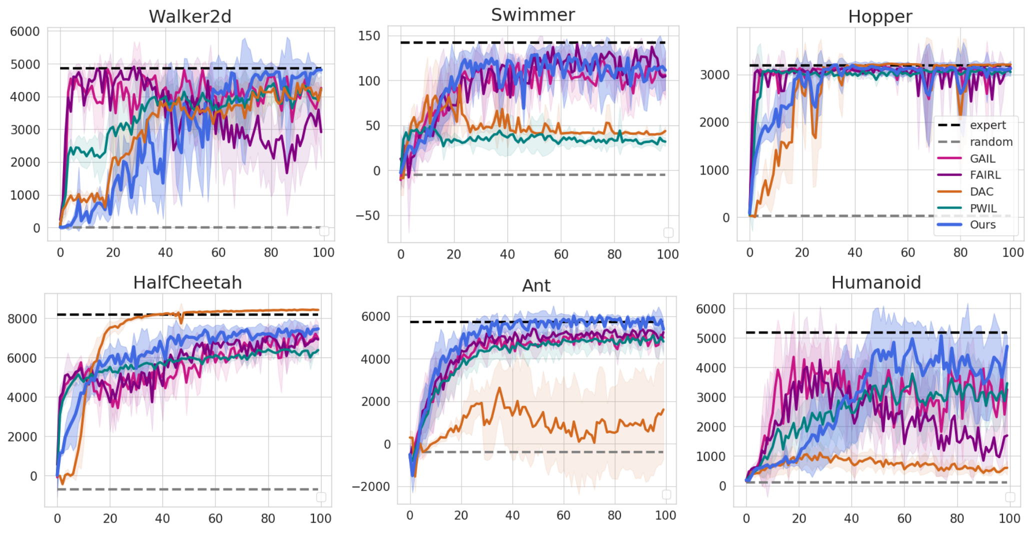

Figure 2: Mean and standard deviation return of the deterministic evaluation policy over 10 rollouts and 5 seeds, reported every 100k timesteps, which are learning from non-noisy expert demonstrations.

Table 1: Learned policy performance for different imitation learning algorithms on clean expert data.

Task	Walker2d	Hopper	Swimmer	HalfCheetah	Ant	Humanoid
Expert	4865.1	3194.9	142.2	8193.6	5730.9	5187.9
Random	-0.6 ± 0.3	26 ± 4	-5 ± 14	-694 ± 108	-401 ± 233	110 ± 4
GAIL	4259 ± 369	3152 ± 102	106 ± 27	6964 ± 807	4998 ± 686	3257 ± 1393
FAIRL	2912 ± 825	3117 ± 75	105 ± 21	6917 ± 267	5247 ± 259	1695 ± 1098
PWIL	4226 ± 501	3051 ± 51	32 ± 5	6376 ± 229	4808 ± 645	3466 ± 736
DAC	4533 ± 248	$\mathbf{3208 \pm 16}$	44 ± 5	$\mathbf{8420 \pm 29}$	1610 ± 2328	602 ± 232
Ours	$\mathbf{4810 \pm 111}$	3125 ± 159	$\mathbf{111 \pm 22}$	7458 ± 191	$\mathbf{5394 \pm 451}$	$\mathbf{4717 \pm 346}$

5.1 Learning from expert demonstrations

Imitation with Clean Expert To compare the performance of our proposed AEAIL and other baselines, we run the experiments on the clean expert dataset. We used the ground truth average return for evaluating different learned policies. Higher rewards indicate better mimicking expert behaviors. Figure 2 depicts the training curves of ground truth rewards for different imitation learning algorithms, and Table 1 shows the final learned policy performance on MuJoCo tasks with ground truth episode return.

Results show that PWIL performs well on Walker2d, Hopper, HalfCheetah, and Humanoid but still not good enough. GAIL and FAIRL are comparable on Hopper, Swimmer, HalfCheetah, while GAIL is somehow better than FAIRL on other tasks. DAC achieves great performance on Walker2d, Hopper and HalfCheetah while it performs poorly on other tasks.

Our method's overall scaled reward is about 0.921, whereas the best baseline is 0.83 for GAIL. There is an about 11% relative improvement. Our method outperforms other state-of-the-art baselines on all locomotion tasks except for Hopper and HalfCheetah, where our approach doesn't outperform DAC. Here we would like to point out that our AEAIL has already

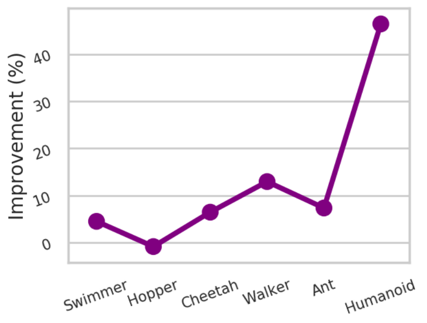

Figure 3: Relative improvement of AEAIL compared to GAIL. Environments are sorted by the number of state-action dimensions.

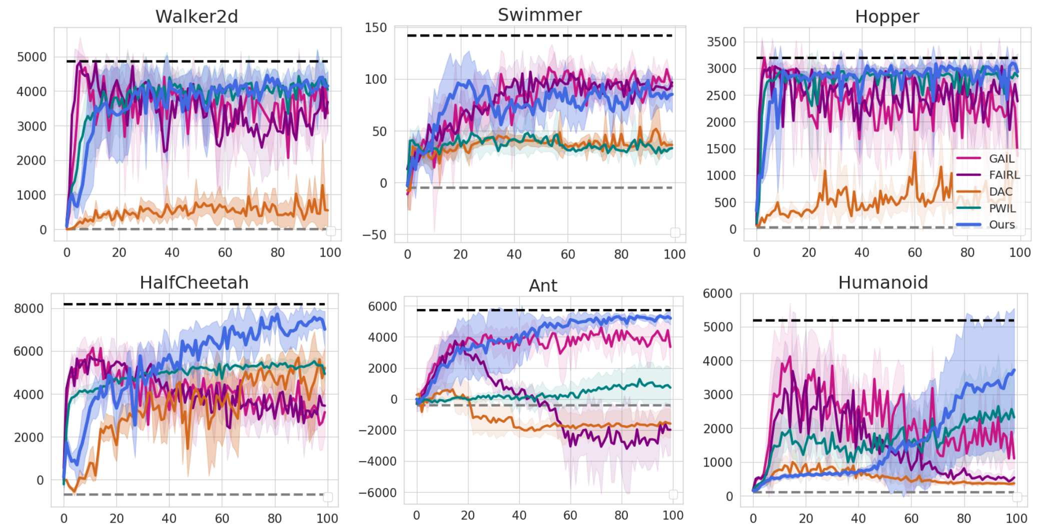

Figure 4: Mean and standard deviation return of the deterministic evaluation policy over 10 rollouts and 5 seeds learning from noisy expert demonstrations, reported every 100k timesteps.

Table 2: Learned policy performance for different imitation learning algorithms on noisy expert data.

Task	Walker2d	Hopper	Swimmer	HalfCheetah	Ant	Humanoid
GAIL	3358 ± 623	1361 ± 725	93 ± 5	3148 ± 1096	3353 ± 1370	1110 ± 296
FAIRL	3686 ± 772	2483 ± 899	$\mathbf{97 \pm 6}$	3451 ± 588	-1987 ± 1633	544 ± 241
PWIL	4043 ± 559	2854 ± 48	33 ± 8	4933 ± 343	729 ± 1274	2321 ± 1038
DAC	529 ± 138	357 ± 171	36 ± 3	2019 ± 900	-1586 ± 922	374 ± 66
Ours	$\mathbf{4149 \pm 423}$	$\mathbf{2935 \pm 110}$	85 ± 17	$\mathbf{7017 \pm 844}$	$\mathbf{5204 \pm 421}$	$\mathbf{3721 \pm 1831}$

achieved 97.8% of the expert performance on Hopper while 91.7% on HalfCheetah, which is very close to completely solving the tasks.

Relative Improvements of AEAIL Figure 3 shows the relative improvement of overall scaled rewards for our AEAIL compared to GAIL on clean expert data. The abscissa axis represents different tasks with the growth of task dimension, from Swimmer (dim=10) to Humanoid (dim=393). With the increasing task dimension, the relative improvement of our AEAIL shows an increasing trend. It indicates that our method has much more benefits when applied into high dimensional environments. We hypothesize that the discriminator in GAIL would focus much more on the minor differences between the expert and agent samples with the growth of task dimension. It is harder for GAIL to scale in higher dimensional tasks. On the other hand, our encoding-decoding process provides more information to the policy learning. Consequently, AEAIL scales relatively better than GAIL in higher dimensional tasks.

5.2 Learning from noisy expert demonstrations

Imitation with Noisy Expert To show the robustness of our proposed AEAIL, we further conduct experiments on noisy expert data. We add a Gaussian noise distribution $(0, 0.3)$ to the expert data for Walker2d, Hopper, Swimmer and HalfCheetah. Since Ant and Humanoid are much more sensitive to noise, we add $(0, 0.03)$ Gaussian noise to these two tasks.

Table 2 shows the final learned policy performances on benchmarks. Figure 4 shows the training curves on noisy expert setting. These results show that our method outperforms other state-of-the-art algorithms on all tasks except for Swimmer, on which FAIRL wins. Our AEAIL offers an excellent capability in learning from noisy expert data on these tasks. The overall scaled rewards for our AEAIL is 0.813, whereas the best baseline is 0.539 for PWIL on the noisy expert setting. There is an about 50.7% relative improvement. Other baselines including GAIL, FAIRL and especially DAC, are very sensitive to the noisy expert.

Video Performance We also record a video[1] for the performance of different methods on both clean and noisy data. It shows that only our AEAIL can move forward as quickly as the expert from noisy expert data, while other baselines suffer from various types of failures.

5.3 Empirical Analysis

A denser reward signal is beneficial for policy search (Ng et al., 1999; Hu et al., 2020). Section 4.1 motivates that our auto-encoder-based reward function focused on the full-scale differences between the expert and generated samples. Our method intuitively provides the agent with a denser reward signal than discriminator-based ones. In this subsection, we will verify this empirically.

Question 1. *Is the auto-encoder-based reward signal denser than the discriminator-based one?*

To answer Question 1, we define the denseness as the amount of reward information for the same batch from agent rollouts. Higher amounts of reward information for the same batch would be beneficial for policy training since it contains much more learning signals. And we define it as a denser reward signal. Specifically, we compute the differential entropy for each reward batch as:

$$H(x) = -\int_{\mathcal{X}} f(x)\log f(x)dx \tag{7}$$

where f is the probability density function whose support is a set $\mathcal{X}$. For computation convenience, we discrete the rewards into 100 sections to compute the probabilities. Therefore, we can get the entropy of each reward batch from agent rollouts.

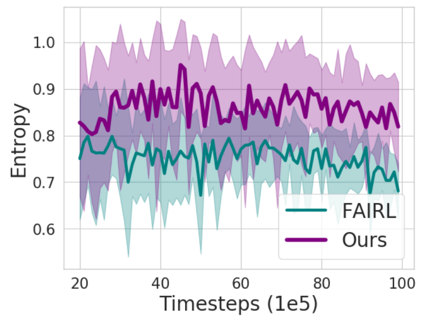

Figure 5: Entropy of learned rewards for state-action pairs in each batch for training the agent-policy overall.

Figure 5 shows the averaged differential entropy curve of the learned pseudo-rewards. This shows the generated samples' entropy in each batch during policy optimization, reported every 100 steps. Our AEAIL gets a much higher entropy during the training process. This means the reward batch contains much more information than the baseline. It also shows that our auto-encoder-based reward signal is indeed denser than the discriminator-based one.

5.4 Ablation Study

Question 2. *What is the major contributing factor of our AEAIL? Could it be the specific W-distance metric?*

To analyze the major contributing factor of our AEAIL, we conduct an ablation study that replaces the distance to other distribution divergences. Comparable performances indicate that the major contributing factor is the encoding-decoding process rather than the specific distribution divergence. To justify this hypothesis, we formulate another form of surrogate reward function based on Jensen-Shannon divergence (Goodfellow et al., 2014). Details see in Appendix A.4.

Jensen-Shannon divergence (Goodfellow et al., 2014) is defined as:

$$\text{JS}(\text{P}_r, \text{P}_g) = \mathbb{E}_{x\sim \text{P}_r}[\log(\frac{\text{P}_r(x)}{\frac{1}{2}(\text{P}_r(x)+\text{P}_g(x))})] + \mathbb{E}_{x\sim \text{P}_g}[\log(\frac{\text{P}_g(x)}{\frac{1}{2}(\text{P}_r(x)+\text{P}_g(x))})] \tag{8}$$

where P_r and P_g represent the expert data distribution and the agent-generated sample distribution accordingly.

Our JS-based variant induces the expert policy by minimizing the distribution distance between expert and generated samples. Specifically, we define that:

$$\text{P}_r(x)/(\text{P}_r(x)+\text{P}_g(x)) = \exp(-\text{AE}(x)) \tag{9}$$

[1]Video results, code and data attached in supplementary materials.

where AE(x) is the same with Equation 5 in our original AEAIL. Minimizing Jensen-Shannon divergence assigns low mean square errors to the expert samples and high mean square errors to other regions.

Table 3 shows that our JS-based variant achieves 10.5% and 44.9% relative improvement compared to the best baseline GAIL and PWIL on clean and noisy expert data, respectively. Similar to the original AEAIL, our JS-based variant also greatly improves the imitation performance on these benchmarks. The relative improvements are comparable between the two distance metrics. It indicates that the major contributing factor of our AEAIL is the encoding-decoding process. This also shows that AEAIL works not limited to a specific distance metric.

Table 3: Relative improvements for different variants of our AEAIL compared to the best baseline GAIL and PWIL on clean and noisy data, respectively.

Improvements	Ours	Ours-JS	Ours-VAE
Clean Data	11.0%	10.5%	7.6%
Noisy Data	50.7%	44.9%	42.1%

Question 3. *Is AEAIL limited to the specific type of auto-encoders? How about utilizing variational auto-encoders?*

To prove that our AEAIL is not sensitive to a specific type of auto-encoders, we conduct experiments by replacing the vanilla auto-encoder with the variational auto-encoder. If variational auto-encoder leads to similar performance on the benchmarks, it illustrates that AEAIL works under a wide range of different auto-encoders. We run the VAE-based variant on MuJoCo tasks under the same setting as the AE-based one. Details see in Appendix A.5.

Table 3 shows our VAE-based variant gets 7.6% and 42.1% relative improvement compared to the best baseline GAIL and PWIL on clean and noisy expert data, respectively. This means that our VAE-based variant improves the imitation performance considerably compared to other baselines. It justifies that our AEAIL is flexible with different auto-encoders.

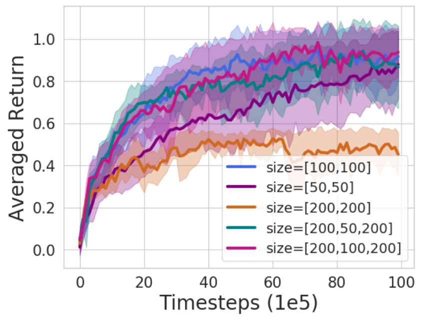

Figure 6: Training curves with averaged scaled rewards overall for varying hidden sizes of the AE.

Question 4. *How our AEAIL performs with varying hidden size of the AE?*

To justify that our AEAIL is not sensitive to different sizes of the auto-encoder, we conduct ablations with different hidden sizes of the auto-encoder. If different sizes of the AE lead to similar imitation performance, it proofs that our AEAIL is not sensitive to the hidden size of the auto-encoder.

Figure 6 shows the averaged overall scaled rewards of AEAIL with different hidden sizes of the auto-encoder. They achieve comparable results for most of the choices. However, when the size of AE is too large, the imitation performance of our AEAIL would decline. It depicts that the hidden size of auto-encoder cannot be too large. Otherwise, the AEAIL is not sensitive to the size of the auto-encoder.

6 Conclusions

This paper presents a straightforward and robust adversarial imitation learning method based on auto-encoding (AEAIL). We utilize the reconstruction error of an auto-encoder as the surrogate pseudo-reward function for reinforcement learning. The advantage is that the auto-encoder based reward function focused on the full-scale differences between the expert and generated samples, which provides a denser reward signal to the agent. As a result, it enables the agents to learn better. Experimental results show that our methods achieve strong competitive performances on both clean and noisy expert data. In the future, we want to further investigate our approach in more realistic scenarios, such as autonomous driving and robotics.

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.

Glen Berseth and Christopher Pal. Visual imitation with reinforcement learning using recurrent siamese networks. 2020. URL `https://openreview.net/forum?id=BJgdOh4Ywr`.

David Berthelot, Thomas Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 182–189. JMLR Workshop and Conference Proceedings, 2011.

Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning (ICML)*, pp. 783–792. PMLR, 2019.

Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. 2016.

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.

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

Tanmay Gangwani and Jian Peng. State-only imitation with transition dynamics mismatch. In *International Conference on Learning Representations (ICLR)*, 2020.

Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, Matthieu Geist, and Stefano Ermon. Iq-learn: Inverse soft-q learning for imitation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Seyed Ghasemipour, Kamyar Seyed, 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 Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

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.

Daniel H Grollman and Aude G Billard. Robot learning from failed demonstrations. *International Journal of Social Robotics*, 4(4):331–342, 2012.

Ziwei Guan, Tengyu Xu, and Yingbin Liang. When will generative adversarial imitation learning algorithms attain global convergence. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1117–1125. PMLR, 2021.

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.

Jonathan Ho, Jayesh Gupta, and Stefano Ermon. Model-free imitation learning with policy optimization. In *International Conference on Machine Learning*, pp. 2760–2769. PMLR, 2016.

Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Mingxuan Jing, Xiaojian Ma, Wenbing Huang, Fuchun Sun, Chao Yang, Bin Fang, and Huaping Liu. Reinforcement learning from imperfect demonstrations under soft expert guidance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5109–5116, 2020.

Kuno Kim, Yihong Gu, Jiaming Song, Shengjia Zhao, and Stefano Ermon. Domain adaptive imitation learning. In *International Conference on Machine Learning (ICML)*, pp. 5286–5295. PMLR, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

Edouard Klein, Bilal Piot, Matthieu Geist, and Olivier Pietquin. A cascaded supervised learning approach to inverse reinforcement learning. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 1–16. Springer, 2013.

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.

Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Hyg-JC4FDr`.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International Conference on Machine Learning (ICML)*, 2016.

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

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning (ICML)*, 1999.

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.

Bilal Piot, Matthieu Geist, and Olivier Pietquin. Bridging the gap between imitation learning and inverse reinforcement learning. *IEEE transactions on neural networks and learning systems*, 28 (8):1814–1826, 2016.

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.

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, 2015.

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.

Kyriacos Shiarlis, Joao Messias, and Shimon Whiteson. Inverse reinforcement learning from failure. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*, pp. 1060–1068, 2016.

Bradly C. Stadie, Pieter Abbeel, and Ilya Sutskever. Third-person imitation learning. In *International Conference on Learning Representations (ICLR)*, 2017.

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.

Ruohan Wang, Carlo Ciliberto, Pierluigi Vito Amadori, and Yiannis Demiris. Random expert distillation: Imitation learning via expert policy support estimation. In *International Conference on Machine Learning*, pp. 6536–6544. PMLR, 2019.

David Warde-Farley and Yoshua Bengio. Improving generative adversarial networks with denoising feature matching. In *International Conference on Learning Representations (ICLR)*, 2017.

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 Pseudo code

Algorithm 1 Auto-Encoding Adversarial Imitation Learning (AEAIL)

```
Input: Initial parameters of policy, auto-encoder $\theta_0$, $w_0$; Expert trajectories $D_E$.
repeat
    Sample state-action pairs $(s_i, a_i) \sim \pi_{\theta_i}$ and $(s_E, a_E) \sim D_E$ with same batch size.
    Update $w_i$ to $w_{i+1}$ by decreasing with the gradient:
        $\mathbb{E}_{(s_i,a_i)}[\nabla_{w_i} 1/(1 + \text{AE}_{w_i}(s_i, a_i))] - \mathbb{E}_{(s_E,a_E)}[\nabla_{w_i} 1/(1 + \text{AE}_{w_i}(s, a))]$
    Take a policy step from $\theta_i$ to $\theta_{i+1}$, using the TRPO update rule with the reward function $1/(1 + \text{AE}_{w_i}(s, a))$, and the objective function for TRPO is:
        $\mathbb{E}_{(s,a)}[-1/(1 + \text{AE}_{w_i}(s, a))]$.
until $\pi_\theta$ converges
```

Algorithm 1 depicts the pseudo-code for training AEAIL. The first step is to sample the state-action pairs from expert demonstrations D_E and the trajectories sampled by the current policy with the same batch size. Then we update the auto-encoder by decreasing with the gradient with loss function Eq. 6. Finally, we update the policy assuming that the reward function is Eq. 4 which leads to an adversarial training paradigm. We repeat these steps until the approach converges.

A.2 Proof on that AEAIL is minimizing the W-distance

In this section, we provide a proof that clipping the weights of the auto-encoder can make the reward function K-Lipschitz. Therefore, we can obtain that our AEAIL actually minimize W-distance.

Theorem 1. *(AE Error is $2m(K+1)$-Lipschitz) Provided that $f(x)$ is K-Lipschitz, then the reconstruction error of our auto-encoder $(x - f(x))^2$ is 2m(K+1)-Lipschitz everywhere.*

Proof. If $f(x)$ is K-Lipschitz, then according to the definition of K-Lipschitz. There exists a constant K, for all real x_1 and x_2, such that:

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$$

where $f(x)$ is a real-valued function.

Then, we can obtain:

$$|(x_1 - f(x_1) - (x_2 - f(x_2))| = |(x_1 - x_2) - (f(x_1) - f(x_2))| \leq (K + 1)|x_1 - x_2|$$

So there exist $K_1 = K + 1$, for all real x_1 and x_2, such that

$$|(x_1 - f(x_1) - (x_2 - f(x_2))| \leq K_1|x_1 - x_2|$$

We can obtain that $(x - f(x))$ is K_1-Lipschitz.

Let $g(x) = x - f(x)$, we can obtain that $g(x)$ is K_1-Lipschitz, then there exists a constant K, for all real x_1 and x_2, such that:

$$|g(x_1) - g(x_2)| \leq K_1|x_1 - x_2| \tag{10}$$

In our AEAIL, x is the input feature which is bounded while $f(x)$ is also bounded since the weights of the auto-encoder is bounded via weight clipping. So $g(x)$ is bounded.

We can obtain:

$$|g^2(x_1) - g^2(x_2)| = |g(x_1) - g(x_2)| \cdot |g(x_1) + g(x_2)| \leq K_1|x_1 - x_2| \cdot |g(x_1) + g(x_2)|$$

We assume that $g(x)$ is bounded by a constant m, then:

$$|g^2(x_1) - g^2(x_2)| \leq 2mK_1|x_1 - x_2|$$

So there exist $K_2 = 2mK_1$, such that for all real x_1 and x_2:

$$|g^2(x_1) - g^2(x_2)| \leq K_2|x_1 - x_2| = 2m(K+1)|x_1 - x_2|$$

We can obtain that the reconstruction error $g^2(x) = (x - f(x))^2$ is also $2m(K+1)$-Lipschitz.

□

Corollary 1. *(Our AEAIL is minimizing Wasserstein distance) the divergence for AEAIL:*

$$d(\pi_E, \pi_\theta) = \sup_{|r_w|_L \leq K} \mathbb{E}_{\pi_E}[r_w(s,a)] - \mathbb{E}_{\pi_\theta}[r_w(s,a)], \tag{11}$$

is Wasserstein distance.

Proof. Arjovsky et al. (2017) shows that given $\mathcal{F}$ is a set of functions from $\mathcal{X}$ to $\mathbb{R}$, we can define

$$d_{\mathcal{F}}(P_r, P_\theta) = \sup_{f \in \mathcal{F}} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_\theta}[f(x)]$$

as an integral probability metric with respect to the function class $\mathcal{F}$. When $\mathcal{F}$ is K-Lipschitz functions, we can obtain $d_{\mathcal{F}}(P_r, P_\theta)$ is the Wasserstein distance.

For our AEAIL, our reward function is:

$$r(s,a) = 1/(1 + \|x - \text{Dec} \circ \text{Enc}(x)\|^2)$$

So the reward function is K-Lipschitz since we clip the weight of the auto-encoder. This will yield the same behaviour. Therefore, our AEAIL actually optimizes the W-distance.

□

A.3 Implementation

Data, code, and video see in supplementary materials. This section will introduce the implementation details, including tasks, basic settings for all algorithms, baseline implementations, network architecture for our approach.

A.3.1 Tasks

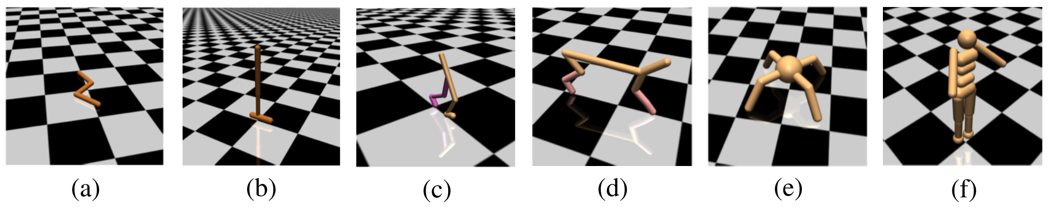

Figure 7: Illustrations for locomotion tasks we used in our experiments: (a) Swimmer; (b) Hopper; (c) Walker; (d) HalfCheetah; (e) Ant; (f) Humanoid.

The goal for all these tasks is to move forward as quickly as possible. These tasks are more challenging than basic ones 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 and falling over, during the initial stage of learning, when the robot cannot move forward for a sufficient distance without falling, apparent local optima exist, including staying at the origin or moving forward slowly. Figure 7 depicts locomotion tasks' environments.

A.3.2 Resources and License

We use 8 GTX 3080 GPU and 40 CPU to run the experiments. Our algorithm needs ~ 120 hours for training on 6 tasks with 5 random seeds. We use an open source MuJoCo license to run the locomotion tasks. The license can be found here: https://mujoco.org.

A.3.3 Basic settings

We sample the expert trajectories with SAC on Walker2d, Hopper, Swimmer, HalfCheetah, and Ant with 50 trajectories while 320 trajectories on Humanoid.

We normalize the state features for each task and use the normalized features as the input for the auto-encoder. We also use the raw data input to compute the final output of the mean square error.

We train these algorithms directly on Walker2d, Hopper, and Swimmer, 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 five random seeds and using ten rollouts to estimate the ground truth rewards to plot the training curves.

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

We clip the auto-encoder's parameters into $[-0.1, 0.1]$ to make the AE L-Lipschitz continuous.

A.3.4 Baselines

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

GAIL (Ho & Ermon, 2016) borrows the GAN (Goodfellow et al., 2014) architecture to realize the IRL process, and it is applicable in high dimensional dynamic environments. We implement GAIL with open-source code, OpenAI baselines (Dhariwal et al., 2017), with best tunned hyperparameters. We choose TRPO as the reinforcement learning algorithm for a fair comparison.

FAIRL (Ghasemipour et al., 2020) utilizes the forward KL divergence as the objective, which shows competitive performances on MuJoCo tasks. FAIRL is implemented based on GAIL with corresponding objective function as in (Ghasemipour et al., 2020). We also choose TRPO as the reinforcement learning algorithm for a fair comparison.

PWIL (Dadashi et al., 2021) is an imitation learning method based on an offline similarity reward function. It is pretty robust and easy to fine-tune the hyper-parameters of its reward function (only two hyper-parameters). PWIL is implemented based on open source code (Dadashi et al., 2021) with TRPO as the reinforcement learning method for a fair comparison.

DAC (Kostrikov et al., 2019) is discriminator-actor-critic algorithm which introduces absorbing states' rewards to improve the imitation performance. DAC is implemented via its open source code.

A.3.5 Architecture of our methods

Policy Net: two hidden layers with 64 units, with tanh nonlinearities in between, which is the same with baselines. The learning rate we used for updating the policy is 0.01.

Auto-Encoder Net: (both vanilla auto-encoder and variational auto-encoder) the number of layers is the same with the architecture of discriminator in GAIL, i.e., four layers. The two hidden layers are with 100 units, with tanh nonlinearities in between, the final output layer is an identity layer. We normalize the state feature after the input layer, and we use the raw state action input features to compute the mean square error. The learning rate for the auto-encoder is 3e-4.

A.4 Ablation study: different distribution divergences

We conduct an ablation study with Jensen-Shannon divergence. We run the experiments with a new form of reward function based on Jensen–Shannon divergence (Goodfellow et al., 2014) for comparison.

Jensen-Shannon divergence is defined as:

$$\text{JS}(\text{P}_r, \text{P}_g) = \mathbb{E}_{x\sim \text{P}_r}[\log(\frac{\text{P}_r(x)}{\frac{1}{2}(\text{P}_r(x) + \text{P}_g(x))})] + \mathbb{E}_{x\sim \text{P}_g}[\log(\frac{\text{P}_g(x)}{\frac{1}{2}(\text{P}_r(x) + \text{P}_g(x))})], \tag{12}$$

where P_r and P_g represents the expert data distribution and the generated data distribution. The adversarial training procedure of AEAIL will match the state-action distribution of expert and generated samples.

In this form of reward function, we consider the exponential of negative reconstruction error as: $\text{P}_r(x)/(\text{P}_r(x) + \text{P}_g(x))$. The derived Jensen-Shannon divergence for updating the auto-encoder based reward function is:

$$\mathcal{L} = -\,\mathbb{E}_{(s,a)\sim D_E}[\log(\exp(-\text{AE}_w(s,a)))] - \mathbb{E}_{(s,a)\sim\pi}[\log(1 - \exp(-\text{AE}_w(s,a)))] \tag{13}$$

$$= \mathbb{E}_{(s,a)\sim D_E}[\text{AE}_w(s,a)] - \mathbb{E}_{(s,a)\sim\pi}[\log(1 - \exp(-\text{AE}_w(s,a)))] \tag{14}$$

Here, w represents the parameters of the auto-encoder network. Its formulation of surrogate reward function is:

$$r_w(s,a) = -\log(1 - \exp(-\text{AE}_w(s,a))). \tag{15}$$

Minimizing the Jensen-Shannon divergence between state-action distribution of the expert and generated trajectories assigns low mean square errors to the expert samples and high mean square errors to other regions. And intuitively, we give high rewards to the regions with low reconstruction errors and vice versa.

Table 4: Learned final policy performance for our AEAIL with W-distance, AE based variant (Ours), JS-distance (Ours-JS), and VAE based variant (Ours-VAE). Top three rows: non-noisy expert; Bottom three rows: noisy expert.

Task	Walker2d	Hopper	Swimmer	HalfCheetah	Ant	Humanoid
Ours	4810 ± 111	3125 ± 159	111 ± 22	7458 ± 191	5394 ± 451	4717 ± 346
Ours-JS	4589 ± 498↓	3245 ± 65↑	123 ± 3↑	7901 ± 226↑	5873 ± 512↑	3564 ± 432↓
Ours-VAE	4813 ± 345↓	3198 ± 142↑	106 ± 6↓	8042 ± 208↑	4901 ± 687↓	3981 ± 494↓
Ours	4149 ± 423	2935 ± 110	85 ± 17	7017 ± 844	5204 ± 421	3721 ± 1831
Ours-JS	4298 ± 338↑	2698 ± 128↓	81 ± 4↓	6768 ± 687↓	5381 ± 512↑	3121 ± 1254↓
Ours-VAE	4354 ± 397↑	3018 ± 105↑	78 ± 7↓	6782 ± 781↓	4292 ± 418↓	3082 ± 1291↓

The results in Table 4 show that W-distance based method is comparable with JS-distance based method on almost all tasks. Our JS based variant achieves 10.5% and 44.9% relative improvement compared to the best baseline GAIL and PWIL on clean and noisy expert data, respectively. It is similar to the W-distance based variant. This indicates that the encoding-decoding process is the major contributing factor in our AEAIL which improves the imitation performance rather than the specific distribution divergence.

A.5 Ablation study: utilizing other auto-encoders

We conduct experiments with variational auto-encoder based reward function as an ablation. The variational auto-encoder (Kingma & Welling, 2014) is a directed model that uses learned approximate inference. The critical insight behind training variational auto-encoder is to maximize the variational lower bound $\mathcal{L}(q)$ of the log-likelihood for the training examples (Goodfellow et al., 2016):

$$\mathcal{L}(q) = \mathbb{E}_{z\sim q(z|x)} \log p_{\text{model}}(z,x) + \mathcal{H}(q(z|x)) \tag{16}$$

$$= \mathbb{E}_{z\sim q(z|x)} \log p_{\text{model}}(x|z) - D_{\text{KL}}(q(z|x) \| p_{\text{model}}(z)) \tag{17}$$

$$\leq \log p_{\text{model}}(x) \tag{18}$$

where z is the latent representation for data points x. And p_{model} is chosen to be a fixed Gaussian distribution $\mathcal{N}(0, I)$. Training the variational auto-encoder is still a adversarial process. We notice

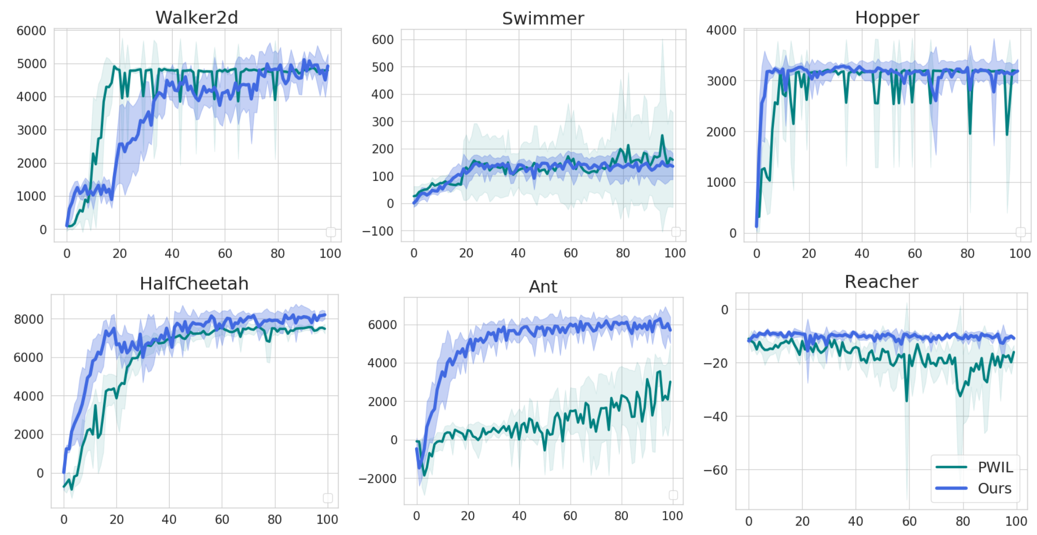

Figure 8: Mean and standard deviation return of the deterministic evaluation policy over 10 rollouts and 5 seeds learning from expert demonstrations, reported every 100k timesteps.

that the first term in Eq. 17 is the reconstruction log-likelihood found in traditional auto-encoders. The objective function for training the variational auto-encoder is:

$$\begin{aligned}\mathcal{L} =&\mathbb{E}_{(s,a)\sim D_E}[r_w(s,a) + D_{\text{KL}}(p(z|(s,a)), p_{\text{model}}(z))] \\ &- \mathbb{E}_{(s,a)\sim \pi_\theta}[r_w(s,a) + D_{\text{KL}}(p(z|(s,a)), p_{\text{model}}(z))]\end{aligned} \tag{19}$$

where the choice for $r_w(s,a)$ is the same with our AE based one, which is Eq. 4 in Section 4.2. When minimizing this objective, the variational auto-encoder based reward function can still assign high values near the expert demonstrations and low values to other regions. On the other hand, optimizing this objective is restricting the latent distribution for the expert to a fixed Gaussian distribution $p_{\text{model}}(z)$ and maximizing the KL distance between the generated samples and the same fixed Gaussian distribution.

Table 4 shows the results for our AE based variant and VAE based variant on both clean and noisy expert data. Our VAE based variant gets 7.6% and 42.1% relative improvement compared to the best baseline GAIL and PWIL, respectively. Comprehensively, our AEAIL works well with the VAE formulation.

A.6 More Experiments built upon DAC

AIL methods formulate the reward function as a discriminator. It can easily overfit to the minor difference between expert and agent generated samples. On the other hand, our AEAIL is a general modification for AIL methods. It views the reward function as an auto-encoder. Under this formulation, our AEAIL can learn the full-scale difference while providing a denser reward function for agent training. So our AEAIL can be built on different variants of AIL methods.

DAC (Kostrikov et al., 2019) is a good variant of GAIL as it uses absorbing states to solve the reward bias problem and also off-policy RL agent training to improve the sample efficiency. In this section, we provide more experiment results built upon DAC with off-policy RL agent. The baseline we choose is PWIL with D4PG. Our AEAIL code is adapted from DAC (Kostrikov et al., 2019). The source code for PWIL is from original paper (Dadashi et al., 2021).

Figure 8 show the training curve of our AEAIL and PWIL. The imitation performance on Reacher shows that our AEAIL also works for negative survival bias tasks. Our AEAIL is better than PWIL on all tasks except for Swimmer. It shows that our AEAIL is general to other variant of AIL methods and it also achieves strong performance.

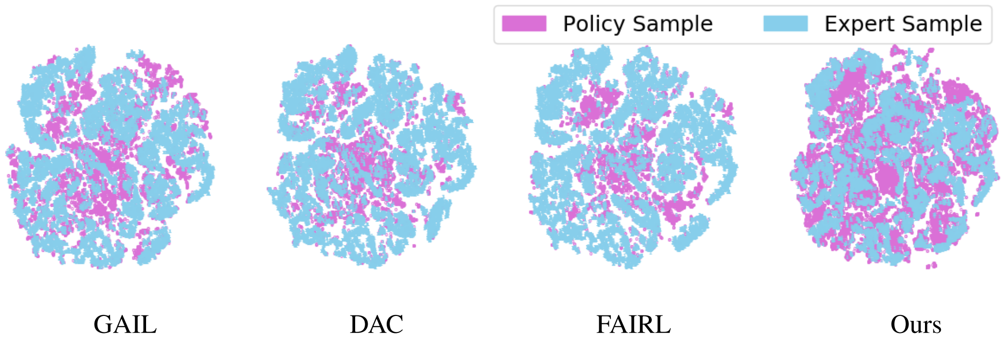

Figure 10: t-SNE visualization for the latent space of discriminator or our auto-encoder on noisy expert data on Walker2d. t-SNE's hyper-parameter perplexity is 30.0.

A.7 Our Auto-encoder is Expressive for Different Rollout Samples

Section 4.1 motivates that our auto-encoder based reward function focuses on the full scale differences between the expert and generated samples. Therefore, it is expressive for different rollout samples. To justify this hypothesis, we illustrate the generated rewards for different samples. Distinct rewards for different trajectories means the reward signal is expressive.

We collect 50 levels of trajectories with performance from random to expert. This can be realized by saving trajectories every 100 iterations when training a PPO. Since the trajectories with different performances have different lengths, we normalize the generated rewards with the length of trajectories. We also scale the generated rewards considering the expert samples are 1 and random policy samples are 0.

Figure 9 shows that the final learned scaled and normalized rewards for different samples with our AEAIL's reward function on Walker2d. Note that learned rewards under our formulation are readily distinct at different levels. Our steep curve indicates that the auto-encoder derived reward function is expressive indeed.

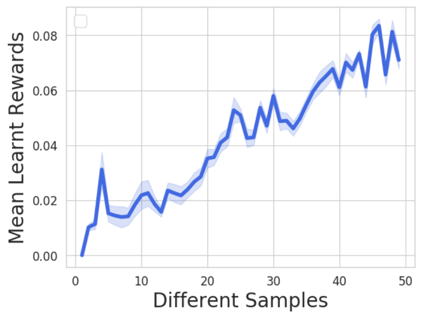

Figure 9: Scaled normalized rewards for different rollout samples on Walker2d.

A.8 t-SNE visualization for the reward function

AIL can be seen as inducing the expert policy through confusing the discriminator. Ideally, the final policy generated samples and expert samples would demonstrate identical distribution. We visualize the latent representations of the discriminator (in AIL methods) and the auto-encoder (in our method) on both the expert and the policy-generated samples with t-SNE. Here, the policy-generated samples are sampled from the same expert policy in order to illustrate the relation between discriminator-based and auto-encoder based reward function.

Ideally, the expert and final-policy-generated samples will be indistinguishable (greatly overlapping) in this latent space. Specifically, we visualize the output of the discriminator's first hidden (middle) layer for the AIL methods. For our AEAIL, we visualize the output of the auto-encoder's first hidden (middle) layer.

However, since the discriminator focus on the minor differences between the expert and generated samples, the embeddings won't fully overlapped. Figure 10 shows that our AEAIL greatly overlaps the two sets of embeddings compared to other discriminator based formulations. It reflects that our auto-encoder can't distinguish the expert and generated samples any more in the latent space. It means our auto-encoder can learn the full-scale difference between expert and generated samples.