Intrinsic Reward Driven Imitation Learning via Generative Model

Xingrui Yu [1] **Yueming Lyu** [1] **Ivor W. Tsang** [1]

Abstract

Imitation learning in a high-dimensional environment is challenging. Most inverse reinforcement learning (IRL) methods fail to outperform the demonstrator in such a high-dimensional environment, e.g., Atari domain. To address this challenge, we propose a novel reward learning module to generate intrinsic reward signals via a generative model. Our generative method can perform better forward state transition and backward action encoding, which improves the module's dynamics modeling ability in the environment. Thus, our module provides the imitation agent both the intrinsic intention of the demonstrator and a better exploration ability, which is critical for the agent to outperform the demonstrator. Empirical results show that our method outperforms state-of-the-art IRL methods on multiple Atari games, even with one-life demonstration. Remarkably, our method achieves performance that is up to 5 times the performance of the demonstration.

1. Introduction

Imitation Learning (IL) offers an approach to train an agent to mimic the demonstration of an expert. Behavioral cloning (BC) is probably the simplest form of imitation learning (Pomerleau, 1991). The promise of this method is to train a policy to predict the demonstrator's actions from the states using supervised learning. However, despite its simplicity, behavioral cloning suffers from a compounding error problem if the data distribution diverges too much from the training set (Ross et al., 2011). In other words, an initial minor error can result in severe deviation from the demonstrator's behavior. On the other hand, inverse reinforcement learning (IRL) (Abbeel & Ng, 2004; Ng & Russell, 2000) aims at recovering a reward function from the demonstration, and then execute reinforcement learning (RL) on that reward function. However, even with many demonstrations, most state-of-the-art inverse reinforcement learning methods fail to outperform the demonstrator in high-dimensional environments, e.g., Atari domain.

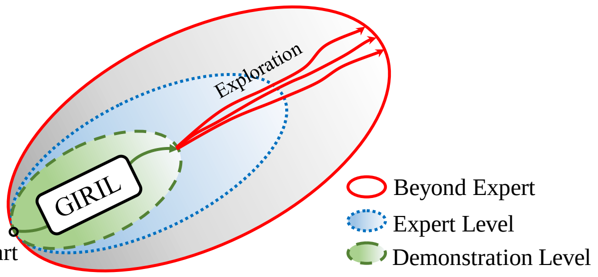


Figure 1. Generative intrinsic reward driven imitation learning (GIRIL) seeks a reward function to achieve three imitation goals. 1) Match the basic demonstration-level performance. 2) Reach the expert-level performance. and 3) Exceed expert-level performance. GIRIL performs beyond the expert by generating a family of intrinsic rewards for sampling-based self-supervised exploration.

In our quest to find a solution to this curse of dimensionality, the primary goal to train an agent to reach the expert-level performance with limited demonstration data. This goal is very challenging because, in our context, limited demonstration data means that the agent can only learn from the states and actions recorded from gameplay up until the demonstrator, i.e. the player, loses their first life. We call this a "one-life demonstration". With currently-available approaches, it is unrealistic to expect an agent to even develop sufficient skills to match a player's basic demonstration-level performance. Our ultimate goal for imitation learning is to build an agent that yields better-than-expert imitation performance from only a one-life demonstration. Figure 1 illustrates the three steps towards achieving this goal.

Most existing IRL methods fail to reach the first goal in Figure 1, i.e. the demonstration-level performance. This is because IRL methods seek a reward function that justifies demonstrations only. Given extremely limited state-action data in a one-life demonstration, the recovered reward can be biased. Executing RL on such reward usually results in an agent performing worse than the demonstration. To address this problem, we propose *Generative Intrinsic Reward driven Imitation Learning* (GIRIL), which seeks a family

[1]Australian Artificial Intelligence Institute, University of Technology Sydney. Correspondence to: Xingrui Yu <Xingrui.Yu@student.uts.edu.au>, Ivor W. Tsang <Ivor.Tsang@uts.edu.au>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

of intrinsic reward functions that enables the agent to do sampling-based self-supervised exploration in the environment. This is critical to better-than-expert performance.

GIRIL operates by reward inference and policy optimization, and includes a novel generative intrinsic reward learning (GIRL) module based on a generative model. We chose variational autoencoder (VAE) (Kingma & Welling, 2013) as our model base. It operates by modeling the forward dynamics as a conditional *decoder* and the backward action encoding as a conditional *encoder*. The *decoder* learns to generate diverse future states from the action conditioned on the current state. Accordingly, the *encoder* learns to encode the future state back to the action latent variable (conditioned on the current state). In this way, our generative model performs better forward state transition and backward action encoding, which improves its dynamics modeling ability in the environment. Our model generates a family of intrinsic rewards, which enables the agent to do sampling-based self-supervised exploration in the environment, which is the key to better-than-expert performance.

Within our chosen domain of Atari games, we first generated a one-life demonstration for each of six games and then trained our reward learning module on the corresponding demonstration data. Finally, we optimize the policy on the intrinsic reward that is provided by the learned reward module. Empirical results show that our method, GIRIL, outperforms several state-of-the-art IRL methods on multiple Atari games. Moreover, GIRIL produced agents that exceeded the expert performance for all six games and the one-life demonstration performance by up to 5 times. The implementation will be available online[1].

2. Problem Definition and Related Work

Problem definition The problem is formulated as a Markove Decision Process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ is the environment transition distribution, $r : \mathcal{S} \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor (Puterman, 2014). The expected discounted return of the policy π is given by

$$\eta(\pi) = \mathbb{E}_\tau \left[\sum_{t=0} \gamma^t r_t \right],$$

where $\tau = (s_0, a_0, \cdots, a_{T-1}, s_T)$ denotes the trajectory, $s_0 \sim \mathbb{P}_0(s_0)$, $a_t \sim \pi(a_t|s_t)$, and $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$.

Inverse reinforcement learning was proposed as a way to find a reward function that could explain observed behavior (Ng & Russell, 2000). With such a reward function, an optimal policy can be learned via reinforcement learning (Sutton et al., 1998) techniques. In the maximum entropy variant of inverse reinforcement learning, the aim is to find a reward function that makes the demonstrations appear near-optimal on the principle of maximum entropy (Ziebart et al., 2008; 2010; Boularias et al., 2011; Finn et al., 2016). However, these learning methods still seek a reward function that justifies the demonstration data only. And, since the demonstration data obviously does not contain information on how to be better than itself, achieving better-than-expert performance with these methods is difficult.

Generative Adversarial Imitation Learning (GAIL) (Ho & Ermon, 2016) treats imitation learning problem as a distribution matching based generative model, which extends IRL by integrating adversarial training technique (Goodfellow et al., 2014). However, this also means that GAIL inherits some problems from adversarial training along with its benefits, such as instability in the training process. GAIL performs well in low-dimensional application, e.g., MuJoCo. However, it does not scale well to high-dimensional scenarios, such as Atari games (Brown et al., 2019a). Variational adversarial imitation learning (VAIL) (Peng et al., 2019) improves GAIL by compressing the information via variational information bottleneck. We have included both these methods as comparisons to our GIRIL in our experiments.

Schroecker et al. (2019) proposed to match the predecessor state-action distributions modeled by the masked autoregressive flows (MAFs) (Papamakarios et al., 2017). Although they have demonstrated the advantages of their approach against GAIL and BC with robot experiments, their approach still requires multiple demonstrations to reach good performance levels and high-dimensional game environments were not included in their evaluations.

Despite their generative ability, GAIL and VAIL just match the state-action pairs from the demonstrations only. GIRIL also uses a generative model, but it does not depend on distribution matching. Rather, it simply improves the modeling of both forward dynamics and backward action encoding of MDP in the environment. Moreover, our method utilizes the generative model based reward learning to generate a family of intrinsic rewards for better exploration in the environment, which is critical for better-than-expert imitation.

Reward learning based on curiosity Beyond the aforementioned methods, reward learning is another important component of reinforcement learning research. For example, intrinsic curiosity module (ICM) is a state-of-the-art exploration algorithm for reward learning (Pathak et al., 2017; Burda et al., 2019). ICM transforms high dimensional states into a visual feature space and imposes cross-entropy and Euclidean loss to learn the feature with a self-supervised inverse dynamics model. The prediction error in the feature space becomes the intrinsic reward function for exploration. Although ICM has a tendency toward over-fitting, we believe it has potential as a good reward learning module and

[1]https://github.com/xingruiyu/GIRIL

so have incorporated it to perform that function in the experiments. Accordingly, we have treated the resulting algorithm *curiosity-driven imitation learning* (CDIL) as related baseline in our experiments.

Reward learning for video games Most imitation learning methods have only been evaluated on low-dimensional environments, and do not scale well to high-dimensional tasks such as video games (e.g., Atari) (Ho & Ermon, 2016; Finn et al., 2016; Fu et al., 2018; Qureshi et al., 2019). Tucker et al. (2018) showed that state-of-the-art IRL methods are unsuccessful on the Atari environments. Hester et al. (2018) proposed deep Q-learning from demonstrations (DQfD), utilizing demonstrations to accelerate the policy learning in reinforcement learning. Since DQfD still requires the ground-true reward for policy learning, it cannot be considered as a pure imitation learning algorithm. Ibarz et al. (2018) proposed to learn to play Atari games by combining DQfD and active preference learning (Christiano et al., 2017). However, it often performs worse than the demonstrator even with thousands of active queries from an oracle. Brown et al. (2019a) learns to extrapolate beyond the suboptimal demonstrations from observations via IRL. However, their method relies on multiple demonstrations with additional ranking information. Our method outperforms the expert and state-of-the-art IRL methods on multiple Atari games, requiring only a one-life demonstration for each game. Moreover, our method does not require any ground-truth rewards, queries or ranking information.

3. Imitation Learning via Generative Model

Most of existing IRL methods do not scale to high-dimensional space, and IL from a one-life demonstration is even more challenging. Our solution to address this challenge, *Generative Intrinsic Reward driven Imitation Learning* (GIRIL), seeks a reward function that will incentivize the agent to outperform the performance of the expert via sampling-based self-supervised exploration. Our motivation is that the expert player's intentions can be distilled even from extremely limited demonstrations, like the amount of data collected prior to the player losing a single life in an Atari game. And when a notion of the player's intentions is combined with an incentive for further exploration in the environment, the agent should be compelled to meet and exceed each of the three goals: par basic demonstration performance, par expert performance and, ultimately, better-than-expert performance. GIRIL works through two main mechanisms: intrinsic reward inference and policy optimization[2].

Intrinsic Rewards Since hand-engineered extrinsic rewards are infeasible in complex environments, our solution is to leverage intrinsic rewards that enable the agent to explore actions that reduce the uncertainty in predicting the consequence of the states. Note, in ICM (Pathak et al., 2017), a network-based regression is used to fit the demonstration data, it is likely to overfit to the limited state-action data drawn from a one-life demonstration. And ICM only produces deterministic rewards. These two problems limit its exploration ability. Empirically, CDIL can only outperform the basic one-life demonstration with the guide of limited exploration ability. Therefore, it is imperative to call for intrinsic reward inference with more powerful exploration ability to achieve better-than-expert performance.

Generative Intrinsic Reward Learning **(GIRL)** To empower the generalization of intrinsic rewards on unseen state-action pairs, our novel reward learning module is based on conditional VAE (Sohn et al., 2015). As illustrated in Figure 2, the module is composed of several neural networks, including recognition network $q_\phi(z|s_t, s_{t+1})$, a generative network $p_\theta(s_{t+1}|z, s_t)$, and prior network $p_\theta(z|s_t)$. We refer to the recognition network (i.e. the probabilistic *encoder*) as a backward action encoding model, and the generative network (i.e. the probabilistic *decoder*) as a forward dynamics model. Maximizing the following objective to optimize the module:

$$\begin{aligned}\mathcal{L}(s_t, s_{t+1}; \theta, \phi) = \mathbb{E}_{q_\phi(z|s_t,s_{t+1})}[\log p_\theta(s_{t+1}|z, s_t)] \\ - \mathrm{KL}(q_\phi(z|s_t, s_{t+1})\|p_\theta(z|s_t)) \\ - \alpha\mathrm{KL}(q_\phi(\hat{a}_t|s_t, s_{t+1})\|\pi_E(a_t|s_t))]\end{aligned} \tag{1}$$

where z is the latent variable, $\pi_E(a_t|s_t)$ is the expert policy distribution, $\hat{a}_t = \mathrm{Softmax}(z)$ is the transformed latent variable, α is a positive scaling weight.

The first two terms of the right-hand side (RHS) in Eq. (1) denote the evidence lower bound (ELBO) of the conditional VAE (Sohn et al., 2015). These two terms are critical for our reward learning module to perform better forward state transition and backward action encoding. In other words, the *decoder* performs forward state transition by taking the action $\tilde{a}_t$ and output the reconstruction $\hat{s}_{t+1}$. On the other hand, the *encoder* performs backward action encoding by taking in the states s_t and s_{t+1} and producing the action latent variable z. Additionally, we integrated the third term of the RHS in Eq. (1) to further boost the backward action encoding. The third term minimizes the KL divergence between the expert policy distribution $\pi_E(a_t|s_t)$ and the action encoding distribution $q_\phi(\hat{a}_t|s_t, s_{t+1})$, where $\hat{a}_t = \mathrm{Softmax}(z)$ is transformed from the latent variable z. In this way, we combine a backward action encoding model and a forward dynamics model into a single generative model.

Note that the full objective in Eq. (1) is still a variational lower bound of the marginal likelihood $\log(p_\theta(s_{t+1}|s_t))$, which is reasonable to maximize as an objective of our reward learning module. Typically, we have $p_\theta(s_{t+1}|z, s_t) \propto$

[2]Policy can be optimized with any policy gradient method.

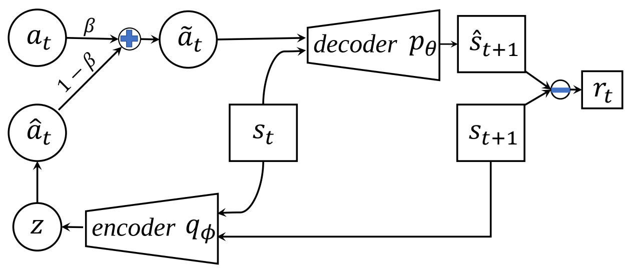


Figure 2. Illustration of the intrinsic reward inference procedure of the proposed GIRIL model.

$\exp(-\lambda\|\hat{s}_{t+1} - s_{t+1}\|_2^2)$, where $\hat{s}_{t+1} = decoder(\tilde{a}_t, s_t)$ is the reconstruction of s_{t+1}. By optimizing the objective, we improve the forward state transition and backward action encoding. Therefore, our reward learning module can better model the dynamics in the environment. During training, we use the latent variable z as the intermediate action $\tilde{a}_t$.

Unlike GAIL and VAIL which fit state-action pairs only, the forward state transition in our GIRL captures the dynamics of the MDP and generates next states; while its backward action encoding use states s_t and s_{t+1} to accurately infer expert's action information. These two parts together can generate new state-action pairs for self-supervised exploration, which is critical for improving the generalization of intrinsic rewards of GIRL on unseen state-action pairs in the environment, resulting in more effective exploration ability.

Sampling-based reward for self-supervised exploration After training, we infer intrinsic reward with the trained reward module following the inference procedure in Figure 2. Each time we first sample a latent variable z from the learned *encoder* by

$$z = encoder(s_t, s_{t+1})$$

and transform it into an action encoding $\hat{a}_t = \text{Softmax}(z)$. The Softmax transformation used here is continuous approximation of the discrete variable (i.e. action) for better model training (Jang et al., 2017). We then achieve an intermediate action data $\tilde{a}_t$ by calculating a weighted sum of the true action a_t and the action encoding $\hat{a}_t$, i.e. $\tilde{a}_t = \beta * a_t + (1-\beta) * \hat{a}_t$, where $\beta \in (0, 1]$ is a positive weight. We use the learned *decoder* to generate the reconstruction $\hat{s}_{t+1}$ from the state s_t and the intermediate action $\tilde{a}_t$. The intrinsic reward is calculated as the reconstruction error between $\hat{s}_{t+1}$ and s_{t+1}:

$$r_t = \lambda\|\hat{s}_{t+1} - s_{t+1}\|_2^2 \tag{2}$$

where $\|\cdot\|_2$ denotes the L2 norm, λ is a positive scaling weight, $\hat{s}_{t+1} = decoder(\beta * a_t + (1-\beta) * \text{Softmax}(z), s_t)$.

Policy Optimization Algorithm 1 summarizes GIRIL's full training procedure. The process begins by training a reward learning module for E epochs (steps 3-6). In each training

Algorithm 1 Generative Intrinsic Reward driven Imitation Learning (GIRIL)

```
1: Input: Expert demonstration $\mathcal{D} = \{(s_i, a_i, s_{i+1})\}_{i=1}^N$.
2: Initialize policy $\pi$, encoder $q_\phi$ and decoder $p_\theta$.
3: for $e = 1, \cdots, E$ do
4:     Sample a batch of demonstration $\tilde{\mathcal{D}} \sim \mathcal{D}$.
5:     Train $q_\phi$ and $p_\theta$ to maximize the objective (1) on $\tilde{\mathcal{D}}$.
6: end for
7: for $i = 1, \cdots$, MAXITER do
8:     Update policy via any policy gradient method, e.g.,
       PPO on the intrinsic reward inferred by Eq. (2).
9: end for
10: Output: Policy $\pi$.
```

Table 1. Demonstration lengths in the Atari environment.

Game	Demonstration Length		# Lives
	One-life	Full-episode	available
Space Invaders	697	750	3
Beam Rider	1,875	4,587	3
Breakout	1,577	2,301	5
Q*bert	787	1,881	4
Seaquest	562	2,252	4
Kung Fu Master	1,167	3,421	4

epoch, we sample a mini-batch demonstration data $\tilde{\mathcal{D}}$ with a mini-batch size of B and maximize the objective in Eq. (1). Then in steps 7-9, we update the policy π via any policy gradient method, e.g., PPO (Schulman et al., 2017), so as to optimize the policy π with the intrinsic reward r_t inferred by Eq. (2).

A family of reward function for exploration Our method generates a family of reward functions instead of a fixed one. After training, we achieve a family of reward functions $r(s_t, a_t, \boldsymbol{z})$ with $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$ where $\mathcal{N}$ denotes the Gaussian distribution. The mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}$ are the output of the encoding network, which is adaptively computed according to the state of environment. The reward inference procedure is shown in Figure 2. This provides more flexibility than a fixed reward function for policy learning. With the family of reward functions, the agent can perform more diverse self-supervised exploration in the environment, which is critical for the agent achieving better-than-expert performance from the limited data in a one-life demonstration.

4. Experiments and Results

4.1. Atari

We evaluate our proposed GIRIL on one-life demonstration data for six Atari games within OpenAI Gym (Brockman et al., 2016). The games and demonstration details are provided in Table 1.

Table 2. Architectures of *encoder*, *decoder* and policy network for Atari games.

encoder	*decoder*	policy network
$4 \times 84 \times 84$ States and Next States	One-hot Actions and $4 \times 84 \times 84$ States	$4 \times 84 \times 84$ States
	dense # Actions $\rightarrow$ 64, LeakyReLU	
concatenate States and Next States	dense 64 $\rightarrow$ 1024, LeakyReLU	
3×3 conv, 32 LeakyReLU	3×3 deconv, 64 LeakyReLU	8×8 conv, 32, stride 4, ReLU
3×3 conv, 32 LeakyReLU	3×3 deconv, 64 LeakyReLU	4×4 conv, 64, stride 2, ReLU
3×3 conv, 64 LeakyReLU	3×3 deconv, 32 LeakyReLU	3×3 conv, 32, stride 1, ReLU
3×3 conv, 64 LeakyReLU	3×3 deconv, 32 LeakyReLU	dense $32 \times 7 \times 7 \rightarrow 512$
dense 1024 $\rightarrow \boldsymbol{\mu}$, dense 1024 $\rightarrow \boldsymbol{\sigma}$	concatenate with States	Categorical Distribution
reparameterization $\rightarrow$ # Actions	3×3 conv, 32 LeakyReLU	
Latent Variable	$4 \times 84 \times 84$ Predicted Next States	Actions

As mentioned, a one-life demonstration only contains the states and actions performed by a expert player until they die for the first time in a game. In contrast, one full-episode demonstration contains states and actions after the expert player losing all available lives in a game. Therefore, the one-life demonstration data is (much) more limited than an one full-episode demonstration. We have defined the performance tries as: basic one-life demonstration-level - gameplay up to one life lost ("one-life"), expert-level - gameplay up to all-lives lost ("one full-episode"), and beyond expert - "better-than-expert" performance. Our ultimate goal is to train an imitation agent that can achieve better-than-expert performance from the demonstration data recorded up to losing their first life.

4.1.1. Demonstrations

To generate one-life demonstrations, we trained a Proximal Policy Optimization (PPO) (Schulman et al., 2017) agent with the ground-truth reward for 10 million simulation steps. We used the PPO implementation with the default hyperparameters in the repository (Kostrikov, 2018). As Table 1 shows, the one-life demonstrations are all much shorter than the full-episode demonstrations, which makes for extremely limited training data.

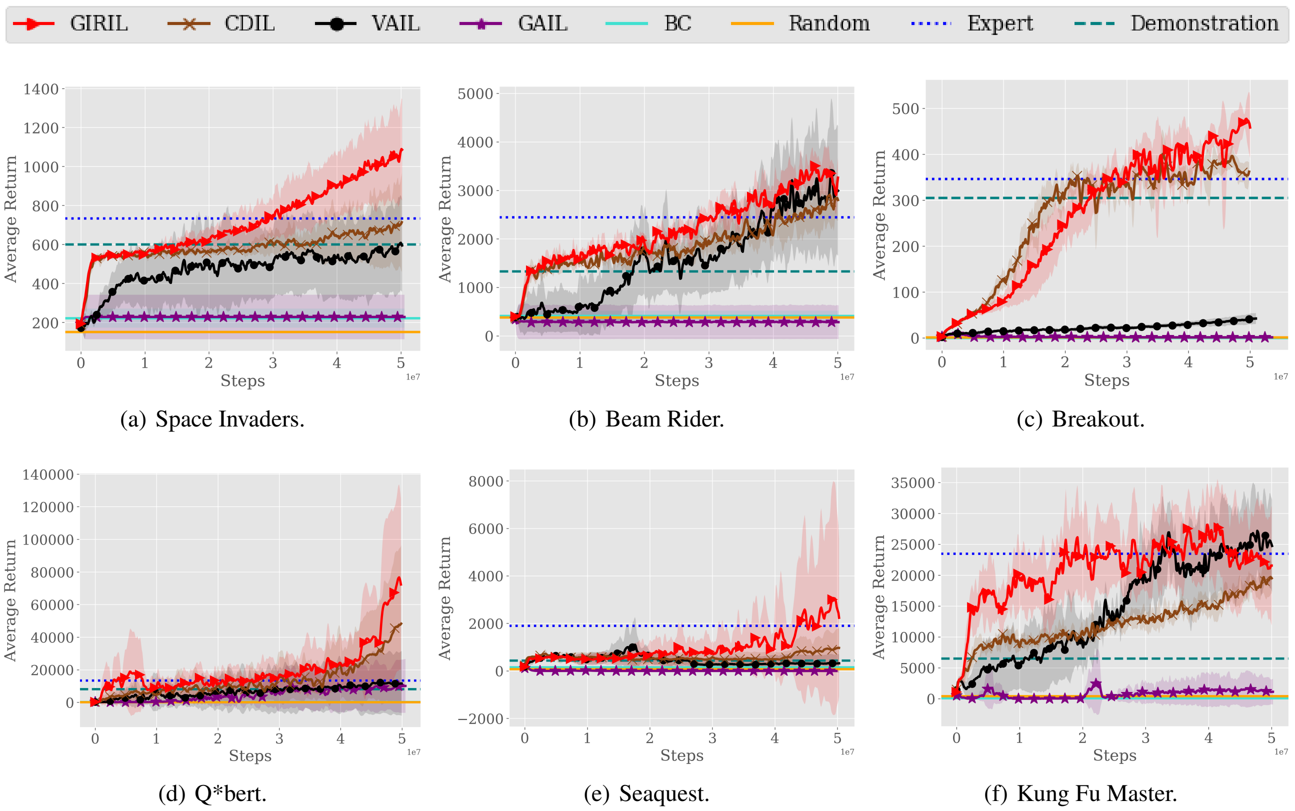
4.1.2. Experimental Setup

Our first step was to train a reward learning module for each game on the one-life demonstration. The proposed reward learning module consists of an *encoder* and a *decoder*. The *encoder* consists of four convolutional layers and one fully-connected layer. Each convolutional layer is followed by a batch normalization layer (BN) (Ioffe & Szegedy, 2015). The *decoder* is nearly an inverse version of *encoder* without the batch normalization layer, except that the *decoder* uses the deconvolutional layer and also includes an additional fully-connected layer at the top of the *decoder* and a convolutional layer at the bottom. For both the *encoder* and the *decoder*, we used the LeakyReLU activation (Maas et al., 2013) with a negative slope of 0.01. Training was conducted with the Adam optimizer (Kingma & Ba, 2015) at a learning rate of 3e-5 and a mini-batch size of 32 for 50,000 epochs. In each training epoch, we sampled a mini-batch of data every four states. We have summarized the detailed architectures of the *encoder* and the *decoder* network in Table 2.

To evaluate the quality of our learned reward, we used the trained reward learning module with a λ of 1.0 to produce rewards, and trained a policy to maximize the inferred reward function via PPO. We normalize s_{t+1} and $\hat{s}_{t+1}$ to [-1,1] before calculating the reward. To further speed up the learning of value function, we performed standardization on the rewards by dividing the intrinsic rewards with a running estimate of the standard deviation of the sum of discounted rewards (Burda et al., 2019). The same discount factor γ of 0.99 was used throughout the paper. We set $\alpha = 100$ for training our reward learning module on Atari games. More experiments have been shown in the Appendix. Additionally, an ablation study in Section A.1 shows the impact of standardization.

We trained the PPO on the learned reward function for 50 million simulation steps to obtain our final policy. The PPO is trained with a learning rate of 2.5e-4, a clipping threshold of 0.1, an entropy coefficient of 0.01, a value function coefficient of 0.5, and a GAE parameter of 0.95 (Schulman et al., 2016). We compared game-play performance by our GIRIL agent against behavioral cloning (BC), and two state-of-the-art inverse reinforcement learning methods, GAIL (Ho & Ermon, 2016) and VAIL (Peng et al., 2019). Additionally, we adopt the intrinsic curiosity module (ICM) (Pathak et al., 2017; Burda et al., 2019) as a reward learning module, and also compare against with the resulting imitation learning algorithm CDIL. We have shown more details about the CDIL algorithm in the Appendix.

For a fair comparison, we used an identical policy network for all methods. The architecture of the policy network is shown in the third column of Table 2. We used the actor-



(a) Space Invaders. (b) Beam Rider. (c) Breakout.

(d) Q*bert. (e) Seaquest. (f) Kung Fu Master.

Figure 3. Average return vs. number of simulation steps on Atari games. The solid lines show the mean performance over five random seeds. The shaded area represents the standard deviation from the mean. The blue dotted line denotes the average return of expert. The area above the blue dotted line means performance beyond the expert.

critic approach in the PPO training for all imitation methods except BC (Kostrikov, 2018). The *discriminator* for both GAIL and VAIL takes in a state (a stack of four frames) and an action (represented as a 2d one-hot vector with a shape of $(|\mathcal{A}| \times 84 \times 84)$, where $|\mathcal{A}|$ is the number of valid discrete actions in each environment) (Brown et al., 2019b). The network architecture of GAIL's *discriminator* D is almost the same as the *encoder* of our method, except that it only outputs a binary classification value, and $-\log(D(s, a))$ is the reward. VAIL was implemented following the repository of Karnewar (2018). The *discriminator* network architecture has an additional convolutional layer (with a kernel size of 4) as the final convolutional layer to encode the latent variable in VAIL. We used the default setting of 0.2 for the information constraint (Karnewar, 2018). PPO with the same hyper-parameters was used to optimize the policy network for all the methods. For both GAIL and VAIL, we trained the *discriminator* using the Adam optimizer with a learning rate of 0.001. The *discriminator* was updated every policy step. The ablation study in Section A.1 will show the effects of different reward functions in GAIL and different information constraints in VAIL.

4.1.3. Results

Figure 3 shows the average performance of the expert, demonstration and imitation learning methods with 5 random seeds. The results reported for all games with our method, with the exception of Seaquest, were obtained by offering the standardized intrinsic reward. The Seaquest results were obtained with the original reward to show that our method also works well without standardization, achieving better performance than other baselines on multiple Atari games. There is more discussion on the influence of standardizing rewards in the ablation study in Section A.1.

What we can clearly see from Figure 3 is that performance of the one-life demonstration is much lower than that of the expert. IL from such little data in a one-life demonstration is challenging. However, as shown in Figure 1, our first goal is to train an agent to outperform the player's performance up to the point of losing one life. This is a start, but it is still a long way from building an agent that can ultimately succeed in the game. Therefore, the second goal is to equal the player's performance across all lives. And, since we always want a student to do better than their master, the third goal is for the agent to outperform the expert player.

Table 3. Average return of GIRIL, CDIL, BC and state-of-the-arts IRL algorithms, GAIL (Ho & Ermon, 2016) and VAIL (Peng et al., 2019), with one-life demonstration data on six Atari games. The results shown are the mean performance over 5 random seeds with better-than-expert performance in bold.

Game	Expert	Demonstration	Imitation Learning Algorithms					Random
	Average	Average	GIRIL	CDIL	VAIL	GAIL	BC	Average
Space Invaders	734.1	600.0	**992.9**	668.9	549.4	228.0	186.2	151.7
Beam Rider	2,447.7	1,332.0	**3,202.3**	**2,556.9**	**2,864.1**	285.5	474.7	379.4
Breakout	346.4	305.0	**426.9**	**369.2**	36.1	1.3	0.9	1.3
Q*bert	13,441.5	8,150.0	**42,705.7**	**30,070.8**	10,862.3	8,737.4	298.4	159.7
Seaquest	1,898.8	440.0	**2,022.4**	897.7	312.9	0.0	155.2	75.5
Kung Fu Master	23,488.5	6,500.0	**23,543.6**	17,291.6	**24,615.9**	1,324.5	44.9	413.7

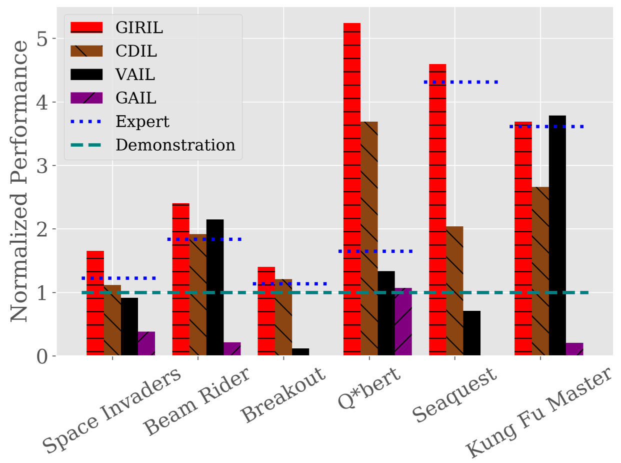


Figure 4. Performance improvement of GIRIL on six Atari games. The results are averages over 5 random seeds and reported by normalizing the one-life demonstration performance to 1.

Better-than-expert Imitation Figure 3 shows that BC and random agent are hopelessly far from achieving the first goal. GAIL only manages to exceed the one-life demonstration for one game, Q*bert. VAIL did better than GAIL, achieving the first goal with three games and the second goal with two. CDIL managed to exceed one-life performance in all six games and even up to goal three, better-than-expert performance, on three games, while our GIRIL accomplished all three goals on all six games and often with a performance much higher than the expert player. A detailed quantitative comparison can be found in Table 3.

The full comparison of imitation performance is shown in Figure 4. To make the comparison more clear, we report the performance by normalizing the demonstration performance to 1. GIRIL's performance excels the expert performance on all six games, and often beating the one-life demonstration by a large margin, for example, 2 times better on Beam Rider, 3 times better on Kung Fu Master, and 4 times better on Seaquest. More impressively, our GIRIL exceeds the one-life demonstration performance on Q*bert by more than 5 times and the expert performance by more than 3 times.

Overall, CDIL came in second our GIRIL. It outdid the one-life demonstration on all six games, but only the expert performance on three, Beam Rider, Breakout and Q*bert. It is promising to seed that both GIRIL and CDIL performed better than the two current state-of-the-arts, GAIL and VAIL. GAIL only stepped out of the lackluster performance with Q*bert. However, VAIL beat the one-life performance on Beam Rider, Q*bert and Kung Fu Master, and the expert performance on Beam Rider and Kung Fu Master. It is clear that, overall, VAIL performed better than GAIL in every game. We attribute VAIL's improvements to the use of variational information bottleneck in the discriminator (Tishby & Zaslavsky, 2015; Alemi et al., 2017).

Comparison with generative model based imitation learning Although GAIL and VAIL are based on generative models just like our GIRIL, they do not have a mechanism for modeling the dynamics of the environment. Distribution matching is done by brute-force and, because one-life demonstration data can be extremely limited, direct distribution matching may result in an over-fitting problem. This is probably the most limiting factor over GAIL and VAIL's performance. Our GIRIL uses a generative model to better perform forward state transition and backward action encoding, which improves the modeling of the dynamics of MDP in environment. Moreover, our method generates a family of intrinsic reward via the sampling-based reward inference. This enables the agent to do self-supervised exploration in the environment. As shown in Figure 3, GIRIL's performance improves at a sharp rate to ultimately outperform the expert player. The difference between matching demonstration performance and delivering better-than-expert performance comes as a result of the enhanced ability to explore, which is also a benefit of our generative model. A final observation is that GIRIL was often more sample-efficient than the other baselines.

Comparison with curiosity-based reward learning The ICM reward learning module is also able to provide the imitation agent with an improved ability to explore. However, the limited demonstration data will give this model a tendency to overfit. Even so, CDIL exceeded the expert

Table 4. Parameter Analysis of the GIRIL with different β on Atari games. Best performance in each row is in bold.

	Expert	Demonstration	GIRIL with different β.				
Game	Average	Average	1.0	0.999	0.99	0.95	0.9
Space Invaders	734.1	600.0	992.9	**1,110.9**	997.3	1,032.1	1,042.6
Beam Rider	2,447.7	1,332.0	3,202.3	3,351.4	3,276.5	**3,402.6**	3,145.0
Breakout	346.4	305.0	**426.9**	397.5	419.3	416.0	361.5
Q*bert	13,441.5	8,150.0	**42,705.7**	25,104.9	29,618.8	23,532.2	38,296.1
Seaquest	1,898.8	440.0	**2,022.4**	526.4	443.3	433.4	355.6
Kung Fu Master	23,488.5	6,500.0	23,543.6	16,521.4	**23,847.7**	20,388.5	19,968.6

performance on three of the games: Beam Rider, Breakout and Q*bert. Granted, for Beam Rider and Breakout, the improvements were only negligible. However, GIRIL's use of generative model to improve dynamics modeling clearly demonstrates the performance improvements to be gained from better exploration ability.

4.1.4. Ablation study of our method with different β.

When $\beta = 1$, we construct a basic version of our reward using the decoder only. When blending actions with different β values, we construct a complete version of our reward that uses both encoder and decoder. Table 4 reports results of our methods, GIRIL, on the six Atari games with different β. The action sampling from the encoder enforces the agent with additional exploration. As a result, it potentially further improves the imitation performance versus $\beta = 1$, eg., improving 1.29% on Kung Fu Master with a β of 0.99, improving 6.25% on Beam Rider with a β of 0.95 and improving 11.88% on Space Invaders with a β of 0.999. Overall, our method generates a family of reward functions, and enables the imitation agent to achieve better-than-expert performance on multiple Atari games. The full learning curves of our method with different β have been shown in the Appendix.

4.2. Continuous control tasks

Except the above evaluation on the Atari games with high-dimensional state space and discrete action space, we also evaluated our method on continuous control tasks where the state space is low-dimensional and the action space is continuous. The continuous control tasks were from Pybullet environment (Coumans & Bai, 2016–2019).

4.2.1. Demonstrations

To generate demonstrations, we trained a PPO agent with the ground-truth reward for 5 million simulation steps. We used PPO implementation in the repository (Kostrikov, 2018) with the default hyper-parameters for continuous control tasks. In each task, we used one demonstration with a fixed length of 1,000 for evaluation.

4.2.2. Experimental Setup

Our first step was also to train a reward learning module for each continuous control task on the one demonstration. To build our reward learning module for continuous tasks, we used two-layer feed forward neural networks with tanh activation function as the model bases of the *encoder* and *decoder*. Two addition hidden layers were added to the model base in *encoder* to output $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, respectively. The dimension of latent variable z is set to the action dimension for each task. Additionally, we used a two-layer feed forward neural network with tanh activation function as policy architecture. The number of hidden unit is set to 100 for all tasks. To extend our method on continuous control tasks, we made minor modification on the training objective. In Atari games, we used KL divergence to measuring the distance between the expert policy distribution and the action encoding distribution in Eq. (1). In continuous control tasks, we instead directly treated the latent variable z as the action encoding and used mean squared error to measure the distance between the action encoding and the true action in the demonstration. We set the scaling weight α in Eq. (1) to 1.0 for all tasks. Training was conducted with the Adam optimizer (Kingma & Ba, 2015) at a learning rate of 3e-5 and a mini-batch size of 32 for 50,000 epochs. In each epoch, we sampled a mini-batch of data every 20 states.

To evaluate our learned reward, we used the trained reward learning module with a λ of 1.0 to produce rewards, and trained a policy to maximize the inferred reward function via PPO. States s_{t+1} and $\hat{s}_{t+1}$ were also normalized to [-1,1] before calculating rewards using Eq. (6). We trained the PPO on the learned reward function for 10 million simulation steps to obtain our final policy. The PPO is trained with a learning rate of 3e-4, a clipping threshold of 0.1, an entropy coefficient of 0.0, a value function coefficient of 0.5, and a GAE parameter of 0.95 (Schulman et al., 2016).

For a fair comparison, we used a two-layer feed forward neural network with tanh activation function as the feature extractor in ICM, and the *discriminator* in GAIL and VAIL. The number of hidden layer was also set to 100. The reward function of GAIL and VAIL was chosen according to the original papers (Ho & Ermon, 2016; Peng et al., 2019). The

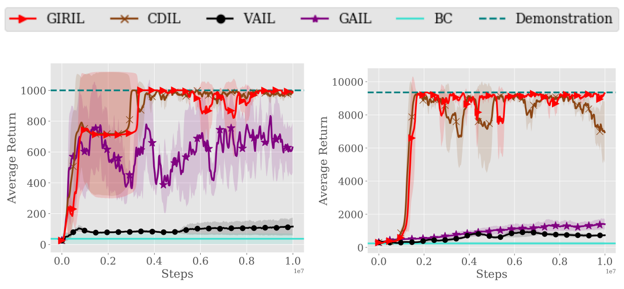
Table 5. Average return of GIRIL, CDIL, BC and state-of-the-arts inverse reinforcement learning algorithms GAIL (Ho & Ermon, 2016) and VAIL (Peng et al., 2019) with one demonstration data on continuous control tasks. The results shown are the mean performance over 3 random seeds with best imitation performance in bold.

	Demonstration	Imitation Learning Algorithms				
Task	Average	GIRIL	CDIL	VAIL	GAIL	BC
InvertedPendulum	1,000.0	**990.2**	979.7	113.6	612.6	36.0
InvertedDoublePendulum	9,355.1	**9,164.9**	7,114.7	725.2	1,409.0	241.6

information constraint I_c in VAIL was set to 0.5 for all tasks. In our experiments with continuous control tasks, we use mean squared error as the discrepancy measure in the objective of inverse dynamics of ICM (in Eq. (3)). To enable fast training, we trained all the imitation methods with 16 parallel processes.

4.2.3. Results

Table 5 shows the detailed quantitative comparison of the demonstration and imitation methods. The results shown in the table were the mean performance over 3 random seeds. Under a fair comparison, our method GIRIL achieves the best imitation performance in both continuous control tasks, i.e. InvertedPendulum and InvertedDoublePendulum. CDIL also achieves performance that is close to the demonstration, while still slightly worse than our method. A comparison of full learning curves can be found in Figure 5.

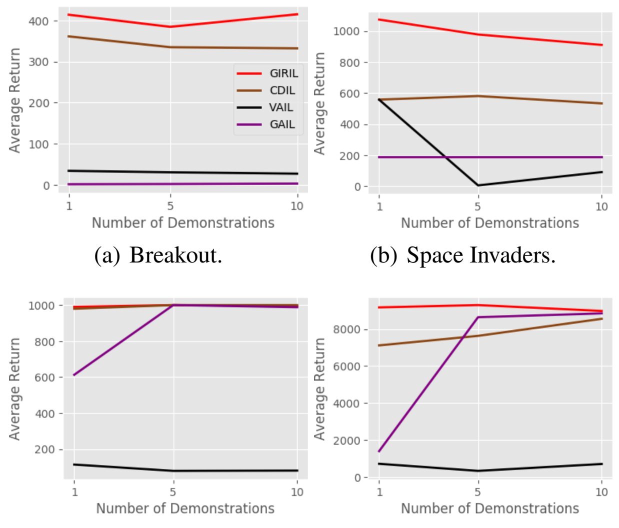


(a) InvertedPendulum. (b) InvertedDoublePendulum.

Figure 5. Average return vs. number of simulation steps on continuous control tasks.

4.3. Full-episode Demonstrations

We also evaluated our method with different number of full-episode demonstrations on both Atari games and continuous control tasks. A comparison of average return versus number of full-episode demonstrations has been shown in Figure 6. The results shows that our method GIRIL achieves the highest performance across different number of full-episode demonstrations on both games and tasks. CDIL usually comes the second best, and GAIL is able to achieve better performance with the increase of the demonstration number in both continuous control tasks. Detailed quantitative results have been shown in the Appendix.



(a) Breakout. (b) Space Invaders.

(c) InvertedPendulum. (d) InvertedDoublePendulum.

Figure 6. Average return vs. number of full-episode demonstrations on Atari games and continuous control tasks.

5. Conclusion

This paper focused on imitation learning from one-life game demonstration in the Atari environment. We propose a novel Generative Intrinsic Reward Learning (GIRL) module based on conditional VAE that combines a backward action encoding and a forward dynamics model into one generative solution. Our generative model can better perform forward state transition and backward action encoding, which improves the modeling of the dynamics of MDP in environment. The better dynamics modeling enables our model to generate more state-action pairs and more accurate rewards. Moreover, our model generates a family of intrinsic rewards, enabling the imitation agent to do sampling-based self-supervised exploration in the environment. Such exploration enables our imitation agent to learn to outperform the expert. Empirical results show that our method outperforms all other baselines including a state-of-the-art curiosity-based reward learning method, two state-of-the-art IRL methods, and behavioral cloning. A comparative analysis of all methods shows the advantages of our imitation learning algorithm across multiple Atari games and continuous control tasks. An interesting topic for future investigation would be to apply our GIRL to a typical, but difficult, exploration task.

Acknowledgements

Ivor W. Tsang was supported by ARC DP180100106 and DP200101328. Xingrui Yu was supported by China Scholarship Council No. 201806450045.

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, pp. 1, 2004.

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. Deep variational information bottleneck. In *International Conference on Learning Representations*, 2017.

Boularias, A., Kober, J., and Peters, J. Relative entropy inverse reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 182–189, 2011.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Brown, D. S., Goo, W., Nagarajan, P., and Niekum, S. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning*, 2019a.

Brown, D. S., Goo, W., and Niekum, S. Ranking-based reward extrapolation without rankings. In *Conference on Robot Learning*, 2019b.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. Large-scale study of curiosity-driven learning. In *International Conference on Learning Representations*, 2019.

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

Coumans, E. and Bai, Y. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019.

Finn, C., Levine, S., and Abbeel, P. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pp. 49–58, 2016.

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

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

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., et al. Deep q-learning from demonstrations. In *AAAI Conference on Artificial Intelligence*, 2018.

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

Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., and Amodei, D. Reward learning from human preferences and demonstrations in atari. In *Advances in Neural Information Processing Systems*, pp. 8011–8023, 2018.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

Karnewar, A. Pytorch implementations of variational discriminator bottleneck, 2018. URL `https://github.com/akanimax/Variational_Discriminator_Bottleneck`.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kostrikov, I. Pytorch implementations of reinforcement learning algorithms, 2018. URL `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail`.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning*, 2013.

Ng, A. Y. and Russell, S. J. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, volume 1, pp. 663–670, 2000.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pp. 2778–2787, 2017.

Peng, X. B., Kanazawa, A., Toyer, S., Abbeel, P., and Levine, S. Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow. In *International Conference on Learning Representations*, 2019.

Pomerleau, D. A. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991.

Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

Qureshi, A. H., Boots, B., and Yip, M. C. Adversarial imitation via variational inverse reinforcement learning. In *International Conference on Learning Representations*, 2019.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 627–635, 2011.

Schroecker, Y., Vecerik, M., and Scholz, J. Generative predecessor models for sample-efficient imitation learning. In *International Conference on Learning Representations*, 2019.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pp. 3483–3491, 2015.

Sutton, R. S., Barto, A. G., et al. *Introduction to reinforcement learning*. MIT press Cambridge, 1998.

Tishby, N. and Zaslavsky, N. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5. IEEE, 2015.

Tucker, A., Gleave, A., and Russell, S. Inverse reinforcement learning for video games. *Workshop on Deep Reinforcement Learning at NeurIPS*, 2018.

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2008.

Ziebart, B. D., Bagnell, J. A., and Dey, A. K. Modeling interaction via the principle of maximum causal entropy. In *International Conference on Machine Learning*, 2010.