SAMPLE EFFICIENT IMITATION LEARNING FOR CONTINUOUS CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

The goal of imitation learning (IL) is to enable a learner to imitate expert behavior given expert demonstrations. Recently, generative adversarial imitation learning (GAIL) has shown significant progress on IL for complex continuous tasks. However, GAIL and its extensions require a large number of environment interactions during training. In real-world environments, the more an IL method requires the learner to interact with the environment for better imitation, the more training time it requires, and the more damage to the environments and the learner it causes. We believe that IL algorithms could be more applicable to real-world environments if the number of interactions could be reduced. In this paper, we propose a model-free, off-policy IL algorithm for continuous control. Experimental results show that our algorithm achieves competitive results with GAIL while significantly reducing the environment interactions. The reduction is made possible by mainly two proposed methods – Q-learning without IRL process, and the policy function representation in the same way as the generator of the conditional generative adversarial networks.

1 INTRODUCTION

Recent advances in reinforcement learning (RL) have achieved super-human performance on several domains (Mnih et al., 2015; Silver et al., 2016; Mnih et al., 2016; Lillicrap et al., 2015). However, on most of such domains with the success of RL, the design of reward, that explains what agent's behavior is favorable, is obvious for humans. Conversely, on domains where it is unclear how to design the reward, agents trained by RL algorithms often obtain poor policies and behave worse than what we expected them to do. Imitation learning (IL) comes in such cases. The goal of IL is to enable the learner to imitate expert behavior given the expert demonstrations without the reward signal. We are interested in IL because we desire an algorithm that can be applied in real-world environments where it is often hard to design the reward. In addition, since it is generally hard to model a variety of real-world environments with an algorithm, and the state-action pairs in a vast majority of real-world applications such as robotics control can be naturally represented in continuous spaces, we focus on model-free IL for continuous control.

A wide variety of IL methods have been proposed in last few decades. The simplest IL method among those is behavioral cloning (BC) (Pomerleau, 1991) which learns an expert policy in a supervised fashion. As opposed to IL methods utilizing RL algorithms, BC does not require environment interactions during training. BC could be the first IL option when enough demonstration is available. However, when just a limited number of demonstrations are available, BC often fails to imitate the expert behavior because of the problem which is referred to *compounding error* (Ross & Bagnell, 2010) – the inaccuracies compound over time and can lead the learner to encounter unseen states in the expert demonstrations. Since it is often hard to obtain a large number of demonstrations in real-world environments, BC is often not be the best choice for real-world IL scenarios.

Another widely used approach, which overcomes the compounding error problem, is Inverse Reinforcement Learning (IRL) (Russell, 1998; Ng & Russell, 2000; Abbeel & Ng, 2004; Ziebart et al., 2008). Recently, Ho & Ermon (2016) have proposed generative adversarial imitation learning (GAIL) which is based on prior IRL works. Since GAIL has achieved state-of-the-art performance on a variety of continuous control tasks, the adversarial IL (AIL) framework has become a popular choice for IL (Baram et al., 2017; Hausman et al., 2017; Li et al., 2017). It is known that AIL

methods are more sample efficient than BC in terms of the expert demonstration. However, as pointed out by Ho & Ermon (2016), the existing AIL methods have sample complexity in terms of the environment interaction. That is, even if enough demonstration was given by the expert before training the learner, they require a large number of state-action pairs obtained through the interaction between the learner and the environment[1]. The sample complexity keeps existing AIL from being employed to real-world applications because of two reasons. First, the more an AIL method requires the interactions, the more training time it requires. Second, even if the expert safely demonstrated, the learner may have policies that damage the environments and the learner itself during training. Hence, the more it performs the interactions, the more it raises the possibility of getting damaged. For the real-world applications, we desire algorithms that can reduce the number of interactions while keeping the imitation capability satisfied as well as the existing AIL methods.

In this paper, we propose a model-free, off-policy IL algorithm for continuous control. Whereas the existing AIL methods adopt on-policy RL methods which have fundamentally sample complexity in terms of the environment interactions, our algorithm is off-policy which is commonly known as the promising approach to improve the complexity in RL literature. Experimental results show that our algorithm enables the learner to imitate the expert behavior as well as GAIL while significantly reducing the environment interactions. The reduction is made possible by mainly two methods – Q-learning without the reward learning process, and the policy function representation in the same way as the generator of the conditional generative adversarial networks.

2 Background

2.1 Preliminaries

We consider a Markov Decision Process (MDP) which is defined as a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, R, d_0, \gamma\}$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of possible actions agents can take, $\mathcal{T} : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to [0, 1]$ is a transition probability, $R : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ is a reward function, $d_0 : \mathcal{S} \to [0, 1]$ is a distribution over initial states, and $\gamma \in [0, 1)$ is a discount factor. The agent's behavior is defined by a stochastic policy $\pi : \mathcal{S}\times\mathcal{A} \to [0, 1]$ and Π denotes a set of the stochastic policies. We denote $\mathcal{S}_E \subset \mathcal{S}$ and $\mathcal{A}_E \subset \mathcal{A}$ as sets of states and actions observed in the expert demonstration, and $\mathcal{S}_\pi \subset \mathcal{S}$ and $\mathcal{A}_\pi \subset \mathcal{A}$ as sets of those observed in rollouts following a policy π. We will use $\pi_E, \pi_\theta, \beta \in \Pi$ to refer to the expert policy, the learner policy parameterized by θ, and a behavior policy, respectively. Given a policy π, performance measure of π is defined as $\mathcal{J}(\pi, R) = \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|d_0, \mathcal{T}, \pi\big]$ where $s_t \in \mathcal{S}$ is a state that the agent received at discrete time-step t, and $a_t \in \mathcal{A}$ is an action taken by the agent after receiving s_t. The performance measure indicates expectation of the discounted return $\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$ when the agent follows the policy π in the MDP. Using discounted state visitation distribution denoted by $\rho_\pi(s) = \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_t = s|d_0, \mathcal{T}, \pi)$ where $\mathbb{P}$ is a probability that the agent receives the state s at time-step t, the performance measure can be rewritten as $\mathcal{J}(\pi, R) = \mathbb{E}_{s\sim\rho_\pi, a\sim\pi}\big[R(s, a)\big]$. The state-action value function commonly called as Q-function for the agent following π is defined as $Q_\pi(s_t, a_t) = \mathbb{E}\big[\sum_{t'=t}^{\infty} \gamma^{t'-t} R(s_{t'}, a_{t'})|\mathcal{T}, \pi\big]$, and $Q_{\pi,\nu}$ denotes its approximator parameterized by ν.

2.2 Adversarial Imitation Learning

We briefly describe objectives of RL, IRL, and AIL below. We refer the readers to Ho & Ermon (2016) for details. First, the goal of RL is to find an optimal policy that maximizes the performance measure. Given the reward function R, the objective of RL with parameterized stochastic policies $\pi_\theta : \mathcal{S}\times\mathcal{A} \to [0, 1]$ is defined as follows:

$$\mathrm{RL}(R) = \arg\max_\theta \mathcal{J}(\pi_\theta, R) \tag{1}$$

Next, the goal of IRL is to find a reward function based on an assumption that the discounted returns earned by the expert behavior are greater than or equal to those earned by any non-experts behavior. Technically, the objective of IRL is to find a reward functions $R_\omega : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ parameterized by ω that satisfies $\mathcal{J}(\pi_E, R_\omega) \geq \mathcal{J}(\pi, R_\omega)$ where π denotes the non-expert policy. The existing AIL

[1]Throughout this paper, we refer to "number of interactions" as the number of state-action pairs obtained through interaction between the learner and the environment during training the learner.

methods adopt max-margin IRL (Abbeel & Ng, 2004) of which objective can be defined as follows:

$$\mathrm{IRL}(\pi_E) = \arg\max_\omega \mathcal{J}(\pi_E, R_\omega) - \mathcal{J}(\pi, R_\omega) \tag{2}$$

Then, the objective of AIL can be defined as a composition of the objectives (1) and (2) as follows:

$$\mathrm{AIL}(\pi_E) = \arg\min_\theta \arg\max_\omega \mathcal{J}(\pi_E, R_\omega) - \mathcal{J}(\pi_\theta, R_\omega) \tag{3}$$

3 ALGORITHM

In this section, we present our off-policy IL algorithm. Our algorithm is based on off-policy actor-critic (Off-PAC) algorithm (Degris et al., 2012) which is composed of two optimization processes: Q-learning and RL. Unlike the existing AIL methods, our algorithm trains Q-function approximator without IRL process, while aiming to satisfy Bellman equation with optimal reward functions for IRL objective we propose. In the following subsections, we describe the proposed IRL objective, Q-learning without IRL process, and RL with the Q-function approximator.

3.1 INVERSE REINFORCEMENT LEARNING

We define the parameterized reward function as $R_\omega(s,a) = \log r_\omega(s,a)$, with a function $r_\omega : \mathcal{S}\times\mathcal{A} \to [0,1]$ parameterized by ω. $r_\omega(s,a)$ represents a probability that the state-action pairs (s,a) belong to $\mathcal{S}_E \times \mathcal{A}_E$, and thus $R_\omega(s,a)$ represents its log probability. In other words, $r_\omega(s,a)$ explains how likely the expert executes the action a at the state s. With this reward, we can also define a Bernoulli distribution $p_\omega : \Pi\times\mathcal{S}\times\mathcal{A} \to [0,1]$ such that $p_\omega(\pi_E|s,a) = r_\omega(s,a)$ for the expert policy π_E and $p_\omega(\pi|s,a) = 1 - r_\omega(s,a)$ for any other policies $\pi \in \Pi \setminus \{\pi_E\}$ which include π_θ and β. A nice property of this definition of the reward is that the discounted return for a trajectory $\{(s_0,a_0),(s_1,a_1),...\}$ can be written as a log joint (discounted) probability of $p_\omega(\pi_E|s_t,a_t)$:

$$\sum_{t=0}^{\infty} \gamma^t R_\omega(s_t,a_t) = \log \prod_{t=0}^{\infty} r_\omega^{\gamma^t}(s_t,a_t) = \log \prod_{t=0}^{\infty} p_\omega^{\gamma^t}(\pi_E|s_t,a_t) \tag{4}$$

Here, we assume Markov property in terms of p_ω such that $p_\omega(\pi_E|s_t,a_t)$ for $t \geq 1$ is independent of $p_\omega(\pi_E|s_{t-u},a_{t-u})$ for $u \in \{1,...,t\}$. Under this assumption, the return naturally represents how likely the trajectory is the ones the expert demonstrated. The discount factor γ plays a role to make sure the return is finite as in standard RL.

The IRL objective (2) can be said to aim at assigning $r_\omega = 1$ for state-action pairs $(s,a) \in \mathcal{S}_E \times \mathcal{A}_E$ and $r_\omega = 0$ for $(s,a) \in \mathcal{S}_\pi \times \mathcal{A}_\pi$, while using the same definition of the reward $R_\omega(s,a) = \log r_\omega(s,a)$. However, following this fashion easily leads to make the return earned by the non-expert policy $-\infty$, since $\log r_\omega(s,a) = -\infty$ if $r_\omega(s,a) = 0$ and thus $\log \prod_{t=0}^{\infty} r_\omega^{\gamma^t}(s_t,a_t) = -\infty$ for $(s,a) \in \mathcal{S}_\pi \times \mathcal{A}_\pi$. The existing AIL methods seem to mitigate this problem by trust region optimization for parameterized value function approximator (Schulman et al., 2015b), and it works somehow. However, we think this problem should be got rid of in fundamental ways. We propose a different approach to evaluate state-action pairs $(s,a) \in \mathcal{S}_\pi \times \mathcal{A}_\pi$. Intuitively, the learner does not know how the expert behaves in the states $s \in \mathcal{S} \setminus \mathcal{S}_E$ — that is, it is uncertain which actions the expert executes in the states the expert unvisited. We thereby define a new IRL objective as follows:

$$\arg\max_\omega \mathbb{E}_{s\sim\rho_{\pi_\mathrm{E}},a\sim\pi_\mathrm{E}}[p_\omega(\pi_E|s,a)] + \mathbb{E}_{s\sim\rho_\pi,a\sim\pi}[H(p_\omega(\cdot|s,a))] \tag{5}$$

where H denotes entropy of Bernoulli distribution such that:

$$H(p_\omega(\cdot|s,a)) = -p_\omega(\pi_E|s,a)\log p_\omega(\pi_E|s,a) - p_\omega(\pi|s,a)\log p_\omega(\pi|s,a) \tag{6}$$

Our IRL objective is to assign $p_\omega(\pi_E|s,a) = p_\omega(\pi|s,a) = 0.5$ for $(s,a) \in \mathcal{S}_\pi \times \mathcal{A}_\pi$. This uncertainty $p_\omega(\pi_E|s,a) = 0.5$ explicitly makes the return earned by the non-expert policy finite. On the other hand, the objective is to assign $r_\omega = 1$ for $(s,a) \in \mathcal{S}_E \times \mathcal{A}_E$, as do the existing AIL methods. Note that, even though our IRL objective (5) does not aim at discriminating between $(s,a) \in \mathcal{S}_E \times \mathcal{A}_E$ and $(s,a) \in \mathcal{S}_\pi \times \mathcal{A}_\pi$ as do the existing AIL methods, the optimal solution satisfies the assumption of IRL : $\mathcal{J}(\pi_E, R_\omega) \geq \mathcal{J}(\pi, R_\omega)$.

3.2 Q-LEARNING

As we see in Equation (4), the discounted return can be represented as a log joint probability. Therefore, Q-function Q_{π_θ} following the learner policy π_θ also can be represented as a log probability. We introduce a function $q_{\pi_\theta,\nu} : \mathcal{S}\times\mathcal{A} \to [0,1]$ parameterized by ν to represent parameterized Q-function approximator $Q_{\pi_\theta,\nu}$ as follows:

$$Q_{\pi_\theta,\nu}(s_t,a_t) = \log q_{\pi_\theta,\nu}(s_t,a_t) = \mathbb{E}\Big[\log \prod_{u=t}^{\infty} p_\omega^{\gamma^{u-t}}(\pi_E|s_u,a_u)|\mathcal{T},\pi_\theta\Big] \tag{7}$$

The optimal Q-function following a policy π satisfies the the Bellman equation $Q_\pi(s_t,a_t) = R(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim\mathcal{T},a_{t+1}\sim\pi}\big[Q_\pi(s_{t+1},a_{t+1})\big]$. Substituting π_θ for π, $\log r_\omega(s_t,a_t)$ for $R(s_t,a_t)$, and $\log q_{\pi_\theta,\nu}(s_t,a_t)$ for $Q_\pi(s_t,a_t)$, the Bellman equation for the learner policy π_θ can be written as follows:

$$\log q_{\pi_\theta,\nu}(s_t,a_t) = \mathbb{E}_{s_{t+1}\sim\mathcal{T},a_{t+1}\sim\pi_\theta}\big[\log r_\omega(s_t,a_t) q^\gamma_{\pi_\theta,\nu}(s_{t+1},a_{t+1})\big] \tag{8}$$

A popular approach in RL literature to obtain the optimal Q-function approximator is to minimize temporal difference (TD) with mean squared error (MSE) $(R_\omega(s_t,a_t)+\gamma Q_{\pi_\theta,\nu}(s_{t+1},a_{t+1}) - Q_{\pi_\theta,\nu}(s_t,a_t))^2$. However, since our definition of R_ω and $Q_{\pi_\theta,\nu}$ have the range $(-\infty,0]$, it is hard to minimize TD because of the large variance of the MSE. We propose a different approach to minimize the TD error without such a large variance by introducing additional Bernoulli distributions $P_\nu : \Pi\times\mathcal{S}\times\mathcal{A} :\to [0,1]$ and $P_{\omega\nu\gamma} : \Pi\times\mathcal{S}\times\mathcal{A}\times\mathcal{S}\times\mathcal{A} :\to [0,1]$ as follows:

$$P_\nu(\pi|s_t,a_t) = \begin{cases} q_\nu(s,a) & \text{if } \pi=\pi_E \\ 1-P_\nu(\pi_E|s_t,a_t) & \text{otherwise} \end{cases} \tag{9}$$

$$P_{\omega\nu\gamma}(\pi|s_t,a_t,s_{t+1},a_{t+1}) = \begin{cases} r_\nu(s_t,a_t)q^\gamma_\nu(s_{t+1},a_{t+1}) & \text{if } \pi=\pi_E \\ 1-P_{\omega\nu\gamma}(\pi_E|s_t,a_t,s_{t+1},a_{t+1}) & \text{otherwise} \end{cases} \tag{10}$$

Using P_ν and $P_{\omega\nu\gamma}$, the TD error can be rewritten as follows:

$$\begin{aligned} L(\omega,\nu,\theta) &= \mathbb{E}_{s_{t+1}\sim\mathcal{T},a_{t+1}\sim\pi_\theta}\big[\log P_{\omega\nu\gamma}(\pi_E|s_t,a_t,s_{t+1},a_{t+1})\big] - \log P_\nu(\pi_E|s_t,a_t) \\ &\le \log \frac{\mathbb{E}_{s_{t+1}\sim\mathcal{T},a_{t+1}\sim\pi_\theta}\big[P_{\omega\nu\gamma}(\pi_E|s_t,a_t,s_{t+1},a_{t+1})\big]}{P_\nu(\pi_E|s_t,a_t)} \end{aligned} \tag{11}$$

We use Jensen's inequality with the concave property of logarithm in Equation (11). Now we see that TD error $L(\omega,\nu,\theta)$ is bounded by the log likelihood ratio between the two Bernoulli distributions P_ν and $P_{\omega\nu\gamma}$, and $L(\omega,\nu,\theta) = 0$ if $P_\nu(\pi_E|s_t,a_t) = \mathbb{E}_{s_{t+1}\sim\mathcal{T},a_{t+1}\sim\pi_\theta}\big[P_{\omega\nu\gamma}(\pi_E|s_t,a_t,s_{t+1},a_{t+1})\big]$. In the end, the problem minimizing the TD error turns out to be the one matching the two Bernoulli distributions. A natural way to measure the difference between two probability distributions is divergence. We choose JS divergence to measure the difference because we empirically found it works better, and thereby the objective of Q-learning can be written as follows:

$$\arg\min_\nu \mathbb{E}\Big[D_{JS}\Big(P_\nu(\cdot|s_t,a_t) \parallel \mathbb{E}_{s_{t+1}\sim\mathcal{T},a_{t+1}\sim\pi_\theta}\big[P_{\omega\nu\gamma}(\cdot|s_t,a_t,s_{t+1},a_{t+1})\big]\Big)\Big] \tag{12}$$

where D_{JS} denotes JS divergence between two Bernoulli distributions. Note that the range of MSE $(R_\omega(s_t,a_t)+\gamma Q_{\pi_\theta,\nu}(s_{t+1},a_{t+1}) - Q_{\pi_\theta,\nu}(s_t,a_t))^2$ could be $[0,\infty)$, whereas the range of JS divergence is $[0,\log 2]$. Hence, even though R_ω and $Q_{\pi_\theta,\nu}$ have the range $(-\infty,0]$, Q-learning using the objective (12) is expected to train $Q_{\pi_\theta,\nu}$ more stable than ones adopting the MSE objective.

3.3 Q-LEARNING WITHOUT INVERSE REINFORCEMENT LEARNING

The existing AIL methods alternate three optimization processes — IRL, value estimation with learned reward functions, and RL using the estimated value. In this way, when the reward function got changed in IRL process, the value is re-estimated to follow the new reward functions, and then the policy tries to maximize the new value functions in RL process. In general, as the number of functions to be optimized increased, it is more complicated to optimize the functions. As it turns out, it makes the training progress slower.

If the optimal reward function $R_{\omega^*} = \log r_{\omega^*}$ for the objective (5) can be obtained, the Bellman equation (8) can be rewritten as follows:

$$\log r_{\omega^*}(s_t, a_t) = \log q_{\pi_\theta,\nu}(s_t, a_t) - \mathbb{E}_{s_{t+1}\sim\mathcal{T},a_{t+1}\sim\pi_\theta}\left[\log q^{\gamma}_{\pi_\theta,\nu}(s_{t+1}, a_{t+1})\right] \tag{13}$$

Recall that IRL objective in our algorithm (5) aims at assigning $r_{\omega^*}(s_t, a_t) = 1$ for $(s_t, a_t) \in \mathcal{S}_E \times \mathcal{A}_E$ and $r_{\omega^*}(s_t, a_t) = 0.5$ for $(s_t, a_t) \in \mathcal{S}_\pi \times \mathcal{A}_\pi$ where $\pi \in \Pi \setminus \{\pi_E\}$. It indicates that the $q_{\pi_\theta,\nu}(s_t, a_t)$ in Equation (13) is supposed to satisfy the following objective to be optimal:

$$\begin{aligned} \arg\min_\nu \mathbb{E}_{s_t\sim\rho_{\pi_\mathrm{E}},a_t\sim\pi_\mathrm{E}}&\left[\log q_{\pi_\theta,\nu}(s_t, a_t) - \mathbb{E}_{s_{t+1}\sim\mathcal{T},a_{t+1}\sim\pi_\theta}\left[\log q^{\gamma}_{\pi_\theta,\nu}(s_{t+1}, a_{t+1})\right]\right] \\ + \mathbb{E}_{s_t\sim\rho_\pi,a_t\sim\pi}&\left[\log q_{\pi_\theta,\nu}(s_t, a_t) - \mathbb{E}_{s_{t+1}\sim\mathcal{T},a_{t+1}\sim\pi_\theta}\left[\log \left\{q^{\gamma}_{\pi_\theta,\nu}(s_{t+1}, a_{t+1})/2\right\}\right]\right] \end{aligned} \tag{14}$$

Thus, r_{ω^*} can be obtained by the objective (13) as long as the optimal solution for Equation (14) can be obtained. In addition, the optimal solution for objective (14) satisfies the Bellman equation with r_{ω^*}. We optimize $q_{\pi_\theta,\nu}(s_t, a_t)$ to be optimal for objective (14) by minimizing the upper bound in the same way of objective (12) as follows:

$$\begin{aligned} \arg\min_\nu \mathbb{E}_{s_t\sim\rho_{\pi_\mathrm{E}},a_t\sim\pi_\mathrm{E}}&\left[D_{JS}\Big(P_\nu(\cdot|s_t, a_t) \parallel \mathbb{E}_{s_{t+1}\sim\mathcal{T},a_{t+1}\sim\pi_\theta}\left[P^{\gamma}_{\nu}(\cdot|s_{t+1}, a_{t+1})\right]\Big)\right] \\ + \mathbb{E}_{s_t\sim\rho_\pi,a_t\sim\pi}&\left[D_{JS}\Big(P_\nu(\cdot|s_t, a_t) \parallel \mathbb{E}_{s_{t+1}\sim\mathcal{T},a_{t+1}\sim\pi_\theta}\left[P^{\gamma}_{\nu}(\cdot|s_{t+1}, a_{t+1})/2\right]\Big)\right] \end{aligned} \tag{15}$$

We use P^{γ}_{ν} instead of $P_{\omega\nu\gamma}$ in objective (15) unlike the objective (12). Thus, we omit IRL process from the three processes that the existing AIL methods require, while aiming at obtaining r_{ω^*} via Q-learning with the objective (15). Reducing functions to be optimized makes the optimization simpler and thus more stable in general. Hence, our off-policy IL algorithm is expected to be more stable than naive off-policy algorithms which have the three optimization process. Note that, the second term in objective (15) samples the state-action pairs according to non-expert policy $\pi \in \Pi \setminus \{\pi_E\}$. Since our algorithm is off-policy, we adopt a behavior policy β as π. We employ a mixture of the past learner policies as β, and the sampling is performed using a replay buffer (Mnih et al., 2015). Thus, our algorithm benefits from the off-policy data while stabilizing the optimization.

3.4 Reinforcement Learning

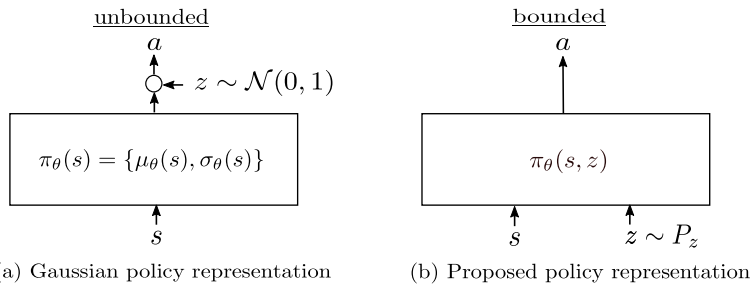
As we mentioned above, we follow Off-PAC algorithm in RL process. In our IL algorithm, the learner with a policy π_θ and the Q-function approximator $Q_{\pi_\theta,\nu}$ trained to satisfy the objective (15) can be interpreted as the actor and the critic, respectively. The objective of Off-PAC to train the learner can be described as follows:

$$\arg\max_\theta \mathbb{E}_{s\sim\rho_\beta,a\sim\pi_\theta}\left[\, Q_{\pi_\theta,\nu}(s, a)\right] \tag{16}$$

To update the learner policy, we need to derive the policy gradients (PG) (Sutton et al., 2000). Degris et al. (2012) proposed the PG as $\mathbb{E}_{s\sim\rho_\beta,a\sim\pi_\theta}\left[Q_{\pi_\theta,\nu}(s, a)\nabla_\theta \log \pi_\theta(a|s)\right]$. However, because $Q_{\pi_\theta,\nu}(s_t, a_t) = \log q_{\pi_\theta,\nu}(s_t, a_t)$ takes $(-\infty, 0]$, using PG with this form always punish (or ignore) the learner's actions. Instead, we follow stochastic value gradient (SVG) algorithms (Heess et al., 2015) to derive the PG directly using Jacobian of $Q_{\pi_\theta,\nu}$. We make use of "re-parameterization trick" to sample actions from the learner policy $a = \pi_\theta(s, z)$ with random variables $z \sim P_z$ generated by a distribution P_z. Thereby, the PG can be presented as follows:

$$\mathbb{E}_{s\sim\rho_\beta,z\sim P_z}\left[\nabla_a Q_{\pi_\theta,\nu}(s, a)|_{a=\pi_\theta(s,z)}\nabla_\theta \pi_\theta(s, z)\right] \tag{17}$$

Prior RL and IL works such as Heess et al. (2015) and Baram et al. (2017) which adopt th re-parameterization trick assume that the learner policy follows Gaussian distributions. On this assumption, the learner's Gaussian policy can be represented as $\pi_\theta(\cdot|s) = \mathcal{N}(\mu_\theta(s), {\sigma^2}_\theta(s))$ with two deterministic functions, $\mu_\theta : \mathcal{S} \to \mathcal{A}$ and $\sigma_\theta : \mathcal{S} \to \mathcal{A}$, and thereby the actions can be sampled as $a = \mu_\theta(s) + z\sigma_\theta(s)$ with $P_z = \mathcal{N}(0, 1)$. However, we empirically found that training the learner with this policy representation is slow or unstable. Recently, Chou & Scherer. (2017) shows that the



(a) Gaussian policy representation (b) Proposed policy representation

Figure 1: Functional representations of (a) Gaussian policy and (b) proposed policy.

policy with Beta distributions, which generate bounded actions, provides faster convergence than Gaussian policy. According to the work of Chou & Scherer. (2017), we assume the reason for the slowness of the training with Gaussian policy is that it easily produces "redundant" actions which take values out of the range of $\mathcal{A}_E$. In common IL settings, we are able to obtain the bound the learner is supposed to follow by observing the expert's actions during demonstration. Thus, utilizing the bound of the expert's action as prior knowledge for representing the learner policy is expected to make training faster. A choice of policy representation bounding the action ranges is to adopt the Beta policy following Chou & Scherer. (2017). However, the punishment for the learner's actions will always occur since they adopt PG as $\mathbb{E}_{s\sim\rho_{\pi_\theta}, a\sim\pi_\theta}\left[Q_{\pi_\theta,\nu}(s,a)\nabla_\theta \log \pi_\theta(a|s)\right]$. Instead, we propose another way to represent the stochastic policy with bounded actions. Recall that the aim of IL is to imitate the expert behavior. It can be summarized that the IL attempts to obtain generative models the expert has over $\mathcal{A}$ conditioned on states in $\mathcal{S}$. We see that the aim itself is equivalent to that of conditional generative adversarial networks (cGANs) (Mirza & Osindero, 2014). In addition, the generator of cGANs can generate stochastic outputs of which range is bounded. In the end, we adopt the form of the conditional generator to represent the stochastic learner policy $\pi_\theta(s, z)$. The difference between Gaussian policy and proposed policy representations with neural networks is described in Figure 1. Note that, unlike cGANs, Q-function in objective (16) does not play a role of the conditional discriminator. However, the experimental results detailed in Section 5 show that our algorithm successfully imitates the expert behavior, which means that the algorithm can obtain the conditional generative model which we refer to the expert policy.

3.5 Off-policy Imitation Learning Algorithm

Algorithm.1 shows overview of our algorithm. As we mentioned in subsection 3.3, we use a replay buffer $\mathcal{B}_\beta$ to perform sampling $s_t \sim \rho_\pi$, $a_t \sim \pi$ and $s_{t+1} \sim \mathcal{T}$. The buffer $\mathcal{B}_\beta$ is a finite cache and stores the (s_t, a_t, s_{t+1}) triplets in a first-in-first-out manner while the learner interacts with the environment. As do off-policy RL methods such as Mnih et al. (2015) and Lillicrap et al. (2015), we use the target Q-function, of which parameters are updated to softly track ν , to optimize $Q_{\pi_\theta,\nu}$. We update $Q_{\pi_\theta,\nu}$ and π_θ at the end of each episode rather than following each step of interaction.

4 Related Work

In recent years, the connection between generative adversarial networks (GAN) (Goodfellow et al., 2014) and IL has been pointed out (Ho & Ermon, 2016; Finn et al., 2016a). Ho & Ermon (2016) show that IRL is a dual problem of RL which can be deemed as a problem to match the learner's occupancy measure (Syed et al., 2008) to that of the expert, and found a choice of regularizer for the cost function yields an objective which is analogous to that of GAN. After that, their algorithm, namely GAIL, has become a popular choice for IL and some extensions of GAIL have been proposed (Baram et al., 2017; Hausman et al., 2017; Li et al., 2017). However, those extensions have never addressed reducing the number of interactions during training. There has been a few attempts that try to improve the sample complexity in IL literatures, such as Guided Cost Learning (GCL) (Finn et al., 2016b). However, those methods do not have worse imitation capability in comparison with GAIL, as reported by Fu & Levine (2017). As detailed in section 5, our algorithm have the capability as well as GAIL while improving the sample complexity. In this paper, we address the problem by

Algorithm 1 Overview of our IL algorithm

Initialize parameters ν and θ.
Fulfill a buffer $\mathcal{B}_{\pi_E}$ by the expert demonstrations and initialize the replay buffer $\mathcal{B}_\beta \leftarrow \emptyset$.
for episode = 1, M **do**
  Initialize time-step $t = 0$ and receive initial state s_0
  while not terminate condition **do**
    Execute an action $a_t = \pi_\theta(s_t, z)$ with $z \sim P_z$ and observe new state s_{t+1}
    Store a state-action pair (s_t, a_t, s_{t+1}) in $\mathcal{B}_\beta$.
    $t = t + 1$
  end while
  for t' = 1, t **do**
    Sample mini-batches of triplets $(s_t^i, a_t^i, s_{t+1}^i)$ and $(s_t^j, a_t^j, s_{t+1}^j)$ from $\mathcal{B}_{\pi_E}$ and $\mathcal{B}_\beta$, respectively.
    Update ν using the sampled gradients of (15) w.r.t ν using $(s_t^i, a_t^i, s_{t+1}^i)$ and $(s_t^j, a_t^j, s_{t+1}^j)$.
    Sample a mini-batch of triplets $(s_t^k, a_t^k, s_{t+1}^k)$ from $\mathcal{B}_\beta$.
    Update θ using the sampled policy gradients (17) using $(s_t^k, a_t^k, s_{t+1}^k)$.
  end for
end for

introducing the off-policy method, which does not adopt "adversarial" training scheme but has the stochastic policy representation corresponding to the generator in cGANs.

There is another line of IL works where the learner can ask the expert which actions should be taken during training, such as DAgger (Ross & Bagnell, 2011), SEARN (Daumé & Marcu, 2009), SMILe (Ross & Bagnell, 2010), and AggreVaTe (Ross & Bagnell, 2014). As opposed to those methods, we do not suppose the learner can query the expert during training.

5 EXPERIMENTS

In our experiments, we aim to answer the following four questions:

Q1. Can our algorithm enable the learner to imitate the expert behavior?
Q2. Is our algorithm more sample efficient than BC in terms of the expert demonstration?
Q3. Is our algorithm more efficient than GAIL in terms of the training time?
Q4. Does our algorithm actually obtain stochastic policies using the input noise variables z?

5.1 SETUP

To answer the questions above, we use five physics-based control tasks that are simulated with MuJoCo physics simulator (Todorov et al., 2012). See Appendix A for the description of each task. In this experiment, we compare the performance of our algorithm, a variant of our algorithm which has a deterministic policy with fixed input noises, BC, and GAIL[2]. The implementation details can be found in Appendix B. We train agents on each task by TRPO (Schulman et al., 2015a) using the rewards defined in the OpenAI Gym (Brockman et al., 2016), then we use the resulting agents with a stochastic policy as the experts for the IL algorithms. We store (s_t, a_t, s_{t+1}) triplets during the expert demonstration, then the triplets are used as training samples in the IL algorithms. In order to study the sample efficiency of the IL algorithms, we arrange two setups. The first is *sparse sampling setup*, where we sample a part of the trajectory. Each part contains 100 (s_t, a_t, s_{t+1}) triplets randomly sampled from a trajectory which contains 1000 triplets. Then we perform the IL algorithms using datasets that consist of a different number of the parts. Another setup is *dense sampling setup*, where we sample full (s_t, a_t, s_{t+1}) triplets in each trajectory, then train the learner using datasets that consist of a different number of the trajectories. If an IL algorithm succeeds to imitate the expert behavior in the dense sampling setup whereas it fails in the sparse sampling setup, we evaluate the algorithm as sample inefficient in terms of the expert demonstration. The performance of the experts and the learners are measured by cumulative reward they earned in a trajectory. We run three experiments on each task, and measure the performance during training.

[2]We also conducted the same experiments to evaluate MGAIL(Baram et al., 2017) with an online available code provided by the authors. However, we never achieved the same performance as reported in the paper on all tasks except for Hopper-v1 even if we followed the author's advice.

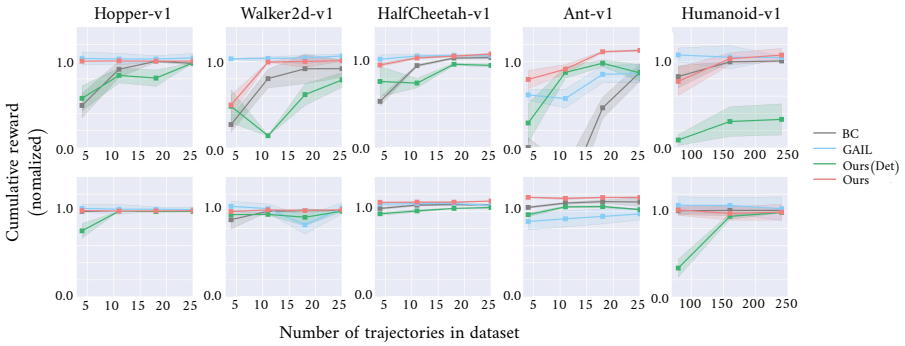


Figure 2: The cumulative reward (normalized) vs. the number of trajectories in a dataset. The results in sparse and dense sampling setup are depicted on top and bottom row, respectively.

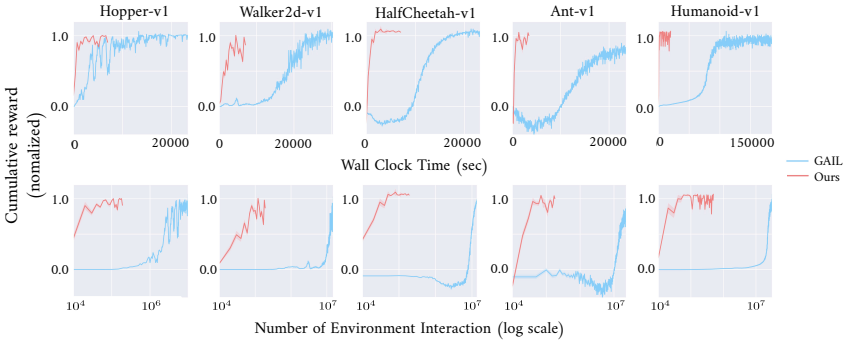


Figure 3: The cumulative reward (normalized) vs. training time (top row) and the number of environment interactions (bottom row).

5.2 Results

Figure 2 shows the experimental results in both sparse and dense sampling setup. In comparison with GAIL, our algorithm marks worse performance on Walker2d-v1 and Humanoid-v1 with the datasets of the smallest size in sparse sampling setup, better performance on Ant-v1 in both setups, and competitive performance on the other tasks in both setups. Overall, we conclude that our algorithm is competitive with GAIL with regards to performance. That is, our algorithm enables the learner to imitate the expert behavior as well as GAIL. BC imitates the expert behavior on all tasks in the dense sampling setup. However, BC often fails to imitate the expert behavior in the sparse sampling setup, while our algorithm achieves better performance than BC does all over tasks. It shows that our algorithm is more sample efficient than BC in terms of the expert demonstration. The deterministic variant of our algorithm denoted by Ours(Det) marks worse performance than our algorithm all over tasks and setups. It indicates that input noise variables z in our algorithm make a role to obtain stochastic policies.

Figure 3 shows the performance plot curves over validation rollouts during training in the sparse sampling setup. The curves on the top row in Figure 3 show that our algorithm trains the learner more efficiently than GAIL does in terms of training time. In addition, the curves on the bottom row in Figure 3 show that our algorithm trains the learner much more efficiently than GAIL in terms of the environment interaction.

6 Conclusion

In this paper, we proposed a model-free, off-policy IL algorithm for continuous control. Experimental results showed that our algorithm achieves competitive results with GAIL while significantly reducing the environment interactions.

REFERENCES

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

Nir Baram, Oron Anschel, Itai Caspi, and Shie Mannor. End-to-end differentiable adversarial imitation learning. In *International Conference on Machine Learning*, pp. 390–399, 2017.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Daniel Maturana Chou, Po-Wei and Sebastian Scherer. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In *International Conference on Machine Learning*, pp. 834–843, 2017.

John Langford Daumé, Hal and Daniel Marcu. Search-based structured prediction. In *Machine learning*, pp. 297–325, 2009.

Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016a.

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

Katie Luo Fu, Justin and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

Ian 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*, pp. 2672–2680, 2014.

Karol Hausman, Yevgen Chebotar, Stefan Schaal, Gaurav Sukhatme, and Joseph Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. *arXiv preprint arXiv:1705.10479*, 2017.

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

G Hinton, N Srivastava, and K Swersky. Rmsprop: Divide the gradient by a running average of its recent magnitude. 2012.

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

Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. pp. 3815–3825, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning*, volume 30, pp. 3, 2013.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937, 2016.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, pp. 807–814, 2010.

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

Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. volume 3, pp. 88–97. MIT Press, 1991.

Gordon G. Ross, S. and D. Bagnell. 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.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 661–668, 2010.

Stephane Ross and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *arXiv preprint arXiv:1406.5979*, 2014.

Stuart Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 101–103, 1998.

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

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pp. 1057–1063, 2000.

Umar Syed, Michael Bowling, and Robert E Schapire. Apprenticeship learning using linear programming. In *International Conference on Machine Learning*, pp. 1032–1039, 2008.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033, 2012.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Association for the Advancement of Artificial Intelligence*, pp. 1433–1438, 2008.

A DETAILED DESCRIPTION OF EXPERIMENT

Table 1 summarizes the description of each task, an agent's performance with random policy, and the performance of the experts.

Table 1: Description of each task, an agent's performance with random policy, and the performance of the experts. $\mathbf{dim}(\mathcal{S})$ and $\mathbf{dim}(\mathcal{A})$ denote dimensionality of state and action spaces respectively.

Task	$\mathbf{dim}(\mathcal{S})$	$\mathbf{dim}(\mathcal{A})$	Random Policy	Expert's Performance
HalfCheetah-v1	17	6	-282.43 $\pm$ 79.53	4130.22 $\pm$ 75.51
Hopper-v1	11	3	14.47 $\pm$ 7.96	3778.05 $\pm$ 3.34
Walker2d-v1	17	6	0.57 $\pm$ 4.59	5510.67 $\pm$ 74.44
Ant-v1	111	8	-69.68 $\pm$ 111.10	4812.93 $\pm$ 122.26
Humanoid-v1	376	17	122.87 $\pm$ 35.11	10395.51 $\pm$ 205.81

B IMPLEMENTATION DETAILS

We implement our algorithm using two neural networks with two hidden layers. Each network represents π_θ and q_ν. For convenience, we call those networks for π_θ and q_ω as policy network (PN) and Q-network (QN), respectively. PN has 100 hidden units in each hidden layer, and its final output is followed by hyperbolic tangent nonlinearity to bound its action range. RN has 500 hidden units in each hidden layer and a single output is followed by sigmoid nonlinearity to bound the output between [0,1]. Hidden layers in PN except for the final outputs are followed by rectified nonlinearity (Nair & Hinton, 2010) and those in QN are followed by leaky rectified nonlinearity (Maas et al., 2013). The parameters in all layers are initialized by Xavier initialization (Glorot & Bengio, 2010). The input of PN is given by concatenated vector representations for the state s and noise z. The noise vector, of which dimensionality corresponds to that of the state vector, generated by zero-mean normal distribution so that $z \sim P_z = \mathcal{N}(0, 1)$. The input of QN is given by concatenated vector representations for the state s and action a. We employ RMSProp (Hinton et al., 2012) for with a decay rate 0.99 and epsilon 10^{-8} . The learning rates are initially set to 10^{-3} for QN and 10^{-4} for PN, respectively. The target QN with parameters ν' are updated so that $\nu' = 10^{-3} * \nu + (1 - 10^{-3}) * \nu'$ at each update of ν. We empirically found that adding a regularizer $D_{JS}\Big(P_\nu(\cdot|s_t, a_t) \parallel P_{\nu'}(\cdot|s_t, a_t)\Big)$ in the objective (15) stabilizes the training. We linearly decrease the learning rates as the training proceeds. We set mini-batch size of (s_t, a_t, s_{t+1}) triplets 64, the replay buffer size $|\mathcal{B}_\beta| = 15000$, and the discount factor $\gamma = 0.85$. We sample 64 noise vectors for calculating empirical expectation $\mathbb{E}_{z\sim P_z}$ of PG (17). We use publicly available code (`https://github.com/openai/imitation`) for the implementation of GAIL and BC. Note that, the number of hidden units in PN is the same as that of networks for GAIL. All experiments are run on a PC with a 3.30 GHz Intel Core i7-5820k Processor, a GeForce GTX Titan GPU, and 32GB of RAM.