Shaping Rewards for Combined Reinforcement and Imitation Learning using Generative Models

Yuchen Wu[1], Melissa Mozifian[2], and Florian Shkurti[1]

Abstract— The potential benefits of model-free reinforcement learning to real robotics systems are limited by its uninformed exploration that leads to slow convergence, lack of data-efficiency, and unnecessary interactions with the environment. To address these drawbacks we propose a method that combines reinforcement and imitation learning by shaping the reward function with a state-and-action-dependent potential that is trained from demonstration data. We show that this accelerates policy learning by specifying high-value areas of the state and action space that are worth exploring first. Unlike the majority of existing methods that assume optimal demonstrations and incorporate the demonstration data as hard constraints on policy optimization, we instead incorporate demonstration data as *advice* in the form of a reward shaping potential trained as a generative model of states and actions. In particular, we use normalizing flows and Generative Adversarial Networks to represent these potentials. We show that, unlike existing approaches that incorporate demonstrations as hard constraints, our approach is unbiased even in the case of sub-optimal and noisy demonstrations. We present an extensive range of simulations, as well as experiments on the Franka Emika 7DOF arm, to demonstrate the practicality of our method.

I. INTRODUCTION

Model-free reinforcement learning has been making significant progress in complex sensorimotor control problems, particularly when optimizing end-to-end vision-based policies [1]. The lack of need for a dynamics model has nevertheless incurred a significant cost in the form of long training times, large number of interactions with the environment, and uninformed exploration. These drawbacks often make model-free reinforcement learning impractical and unsafe to apply to real robotic systems.

We propose a method that combines reinforcement learning (RL) with demonstrations and imitation learning (IL) in order to address these issues and accelerate the policy optimization process. Our method improves upon, and provides an alternative to, existing methods that combine RL with demonstrations, by gracefully handling the case of sub-optimal and noisy demonstrations. We do this by shaping the reward function to incorporate user demonstrations in the form of *advice* [2] that biases the optimization process
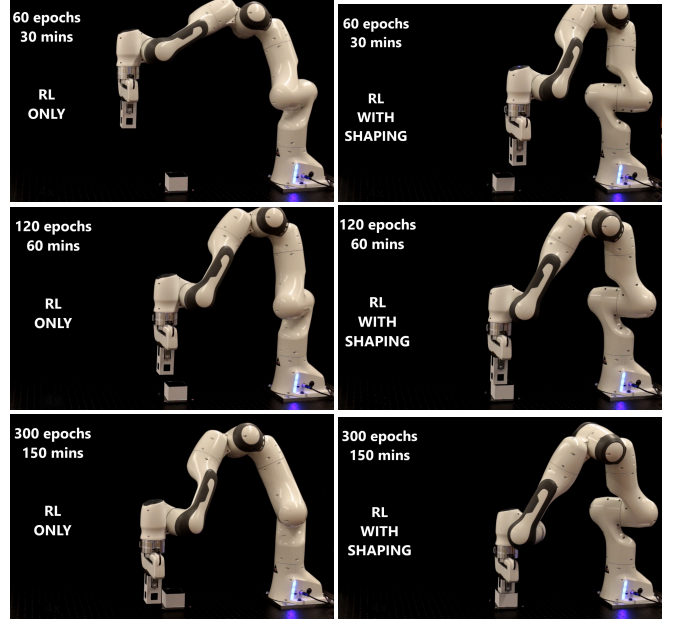
Fig. 1. Snapshots of the trained peg in hole policy after 30, 60, and 150 minutes on the Franka Emika Panda arm. On the first column, the policy has been trained using reinforcement learning. On the right column, the policy has been trained through reinforcement learning and reward shaping, such that the shaping potential is a generative model that describes the demonstration data. Code and videos are available at: http://www.cs.toronto.edu/~florian/rl_with_shaping

towards areas of the state-action space that the demonstrator deems high-value, without biasing the learned policy away from the optimal one. Our shaping potential is learned from a small number of demonstrations using either normalizing flows [3] or Generative Adversarial Networks [4], [5].

The vast majority of existing works that combine RL with demonstrations [6], [7], [8] implicitly assume optimality of demonstrations, or lack of bias in the off-policy data. If the demonstration dataset is $\mathcal{D} = \{(s_i, a_i), i = 1...N\}$ these methods typically solve a variant of the following problem:

$$\max_{\theta} \mathcal{V}^{\pi_\theta}(s_0) \text{ subject to } \pi_\theta(s_i) = a_i \ \forall i \qquad (1)$$

where $\mathcal{V}^{\pi_\theta}(s_0) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0\right]$ is the value function corresponding to the policy π_θ and the fixed starting state s_0. This problem ends up being converted to one that instead has a soft regularization term for the demonstrations:

$$\max_{\theta} \mathcal{V}^{\pi_\theta}(s_0) - \lambda \sum_{(s_i, a_i) \in \mathcal{D}} (\pi_\theta(s_i) - a_i)^2 \qquad (2)$$

There are a number of drawbacks to these formulations:

(a) They assume optimal demonstrations, which is often not a realistic assumption, particularly when sub-optimal

actions occur at states near which optimal trajectories need to pass through. In another instance of this same issue, a number of recent works, for example [6], [7], include the demonstrations permanently in the replay buffer for off-policy RL methods, which again assumes optimality of the demonstrations. Sub-optimality could be a result of the demonstrator not optimizing the same underlying reward as the RL problem, or not being an expert. Both of these possibilities are unaccounted for by the formulations in Eqns. 1 and 2 and can bias the learned policy away from the optimal policy.

(b) A number of recent papers, for example [9], address (a) by reducing the effect of the demonstrations over time, by replacing λ with a decreasing sequence λ_t such that $\lim_{t \to \infty} \lambda_t = 0$. While this addresses the issue of sub-optimality and eventually forgets the demonstrations, it introduces another design parameter, namely the speed at which the demonstrations will be forgotten.

(c) The formulations in Eqns. 1 and 2 cannot gracefully handle multi-modal action distributions at a given state. If the dataset includes (s_i, a_i) and (s_i, a_j) then the policy is forced to compromise by selecting the average action, which might be neither desirable nor safe. Multi-modal policies avoid this issue, but deterministic or unimodal policies do not.

Our method addresses the issues above and brings the following advantages:

- It does not make any assumptions about optimality of the demonstrations, and it does not allow the demonstrations to introduce bias to the learned policy.
- It does not require a forgetting schedule for the demonstrations.
- It can handle multi-modal demonstrations gracefully.

We demonstrate these properties via an extensive range of simulations as well as via real robot experiments on the Franka Emika 7DOF compliant arm.

II. RELATED WORK

There is a large number of methods that combine reinforcement learning with demonstrations, many of them in the realm of discrete MDPs and game playing. In the last decades there have been significant results in trying to combine reinforcement and imitation learning for robotics applications and continuous dynamics. Although an exhaustive list is too large to include, we mention a few representative papers.

RL + Shaping: Our work builds upon Wiewiora et al. [2], who showed that a state-action potential biases the Q-function of the original MDP, by the exact amount of the shaping potential. They introduce the notion of *advice* for an RL agent. The class of shaping potentials they considered, however, was limited to discrete action and state spaces, and was not applicable to robotics, or high-dimensional systems. Our work addresses this setting by using shaping potentials that are directly trained from demonstration data via generative models. Also related is the seminal work of Ng et al. [10], that introduced the notion of reward shaping and the conditions under which policy invariance holds when rewards are modified. It is worth mentioning that aside

from state-action potentials, there are other possible shaping formulations that are time-dependent [11].

Our work extends Brys, Harutunyan et al. [12], who incorporate demonstrations via shaping potentials that, unlike the ones we propose here, are not suited for handling images or other high-dimensional objects. In contrast, we propose to use shaping potentials that are in the form of deep generative models, such as normalizing flows and Generative Adversarial Networks, which can represent high-dimensional random variables.

RL + Demonstrations: Among the papers that combine reinforcement learning with demonstration data, we note Deep Q-Learning from Demonstrations [6] and DDPG from Demonstrations [7], both of which implicitly assume optimality of the demonstrations data, unlike our work herein. In fact, [7] assumes that the demonstration data are in the form (s, a, r, s'), which severely constrains the type of interactions that the demonstrator can have with the robot. Having access to the reward, in addition to states and actions, is problematic in scenarios where the robot is not aware of the task that the demonstrator is executing. It is also very restrictive, in the sense that it does not easily allow extensions, where only the states are given but not the actions. We therefore avoid assuming imitation data in that form and opt for tuples (s, a) as the demonstration dataset. The notion of *advice* assumes other forms, for example high-level Linear Temporal Logic formulas that guide the (discrete) RL process, as was done in [13]. Other papers in this category include [8], [9]. The latter is one of the few papers that impose a schedule for forgetting the demonstrator. Optimization eventually focuses on the RL objective, but only after the policy has been initialized with behavioral cloning. SQIL [14] incorporates demonstrations in the replay buffer and assigns a reward of +1 to them. AC-Teach [15] handle the case of suboptimal demonstrations using an ensemble of demonstrators and Bayesian actor-critic methods. Nair et al. [16] provide another way of not assuming optimal demonstrations, called *Q-Filtering*, whereby they only keep the terms of the behavioral cloning loss for which the demonstrated action has higher Q-value than the action returned by the policy.

In [17] the case of failed demonstrations is considered, enabling the robot to learn from both successful and unsuccessful demonstrations. TAMER [18] and other interactive RL approaches assume a continuous user feedback mechanism that supplements the RL reward. The required feedback stream is a significant burden on the user.

Finally, it is worth mentioning a representative body of work in off-policy reinforcement learning [19], [20], which aim to constrain the effect of Q overestimation away from the collected data. Importance sampling is a commonly used way to do this, as long as one has access to the behavioral policy or can approximate it [21], [22], [23], which may lead to issues of lack of scalability in high dimensions.

III. METHODOLOGY

A. State-Action, Potential-Based Reward Shaping

Given a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma)$, reward shaping, as introduced in the seminal work by Ng et al [10] refers to modifying the (often sparse) reward function in order to solve another MDP $\widetilde{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \widetilde{r}, \gamma)$ such that:

$$\widetilde{r}_t = r(s_t, a_t, s_{t+1}) + \gamma \Phi(s_{t+1}) - \Phi(s_t) \tag{3}$$

The function Φ is called a *shaping potential*, and it is meant to make sparse reward functions more dense by providing more reward signal for the recursive computation of the state-action value function $Q(s, a)$. Ng et al. showed that the optimal value functions between the original and modified MDPs satisfies the following equation:

$$\widetilde{Q}^*(s, a) + \Phi(s) = Q^*(s, a) \tag{4}$$

Every optimal policy corresponding to these state-action value functions will satisfy $\pi^*(s) = \mathrm{argmax}_a Q^*(s, a) = \mathrm{argmax}_a \widetilde{Q}^*(s, a) = \widetilde{\pi}^*(s)$. In other words, every optimal policy for $\mathcal{M}$ will be optimal for $\widetilde{\mathcal{M}}$ and vice versa, so the optimal behavior is not affected by the shaping function, even though the value function is.

Wiewiora et al. [24] showed that the shaping potential did not need to depend only on states, it could also depend on actions. The modified reward then becomes:

$$\widetilde{r}_t = r(s_t, a_t, s_{t+1}) + \gamma \Phi(s_{t+1}, a_{t+1}) - \Phi(s_t, a_t) \tag{5}$$

which gives rise to the following state-action value function:

$$\widetilde{Q}^*(s, a) + \Phi(s, a) = Q^*(s, a) \tag{6}$$

In this case of state-action shaping potentials, there are no guarantees about the preservation of the optimal policy of the original MDP to the modified MDP. In fact, the optimal policy of the original MDP is

$$\pi^*(s) = \mathrm{argmax}_a \left[\widetilde{Q}^*(s, a) + \Phi(s, a) \right] \tag{7}$$

while the optimal policy of the modified MDP is $\widetilde{\pi}^*(s) = \mathrm{argmax}_a \left[\widetilde{Q}^*(s, a) \right]$, which is in general different.

Wiewiora et al. [24] demonstrated potential functions for discrete state and action spaces, that were constrained to low-dimensional discrete planning problems, which are not applicable to robotics. Our paper analyzes the case where the state and action space is high-dimensional and continuous, and the shaping potential is trained via generative models, in order to support many types of demonstration data and improve the convergence properties of imitation-guided model-free RL. As long as we are able to optimally solve Eqn. 7 and $\widetilde{Q}^*(s, a)$ is well estimated, the learned policy incorporates *advice* $\Phi(s, a)$, without imposing the demonstrations as hard constraints, and without introducing bias compared to the optimal policy.

B. Potentials Based On Normalizing Flows

One of the types of state-action shaping potentials that we consider herein is a trained function $\Phi_{\psi,c}(s, a) = c \log p_\psi(s, a)$ on demonstration data $\mathcal{D} = \{(s_i, a_i), i = 1...N\}$. One class of generative models that have emerged in the last few years, that is able to directly optimize this log-density objective on a given dataset are *normalizing flows*.

The main idea behind this class of models is that we can use the change-of-variables formula for probabilistic distributions to transform a normal distribution (a distribution that is easy to sample) to an arbitrary distribution (from which it is difficult to sample). Given a random variable $z_0 \in \mathbb{R}^d$, such that $z_0 \sim p_0(z_0) = \mathcal{N}(0, I_d)$, and an invertible, smooth function $f : \mathbb{R}^d \to \mathbb{R}^d$ with $z_1 = f(z_0)$, the change of variables formula for distributions is:

$$p_1(z_1) = p_0(z_0) \left| \det \left(J_f(z_0)^{-1} \right) \right| \tag{8}$$

$$= p_0(z_0) \left| \det \left(J_f(z_0) \right) \right|^{-1} \tag{9}$$

Rezende and Mohamed [3] chained multiple of these bijective transformations to create a normalizing flow:

$$z_0 \sim p_0(z_0) \tag{10}$$

$$z_K = f_K \circ f_{K-1} \circ ... \circ f_1(z_0) \tag{11}$$

$$p_K(z_K) = p_0(z_0) \prod_{k=1}^{K} \left| \det \left(J_{f_k}(z_{k-1}) \right) \right|^{-1} \tag{12}$$

where $\circ$ denotes function composition. The vast majority of the recent literature on normalizing flows concerns itself with different ways to parameterize bijective functions $f_{\psi_i}(z)$ in a way that chaining multiple of them results in an expressive enough output distribution. We follow Papamakarios et al [25] and we use the same bijective transformation as Masked Autoregressive Flow (MAF):

$$
\begin{aligned}
z_k^{(1)} &= \mu_{w_{k_1}} + \exp(\alpha_{v_{k_1}}) z_{k-1}^{(1)} \\
z_k^{(i)} &= \mu_{w_{k_i}}(z_k^{(1:i-1)}) + \exp(\alpha_{v_{k_i}}(z_k^{(1:i-1)})) z_{k-1}^{(i)}
\end{aligned} \tag{13}
$$

Here, the superscript $i \leq d$ indexes the dimensions of the random variable $z_k \in \mathbb{R}^d$, and makes the ith entry of the output variable depend only on entries $1...i$ of the input variable. This preserves the triangular structure of the Jacobian matrix, so the determinant remains easy to compute. The parameters of the transform $f_{\psi_k}(z_{k-1})$ described in Eqn. 13 are $\psi_k = (w_{k_1}, v_{k_1}, ..., w_{k_d}, v_{k_d})$. The exponential term for the scaling factor is meant to ensure the positivity of standard deviation.

Training a normalizing flow is typically done via maximum likelihood estimation, by optimizing the parameters $\psi = (\psi_1, \psi_2, ..., \psi_K)$, so that the log likelihood of the points in the sample dataset is maximized. In our case, we treat $z_K = (s, a)$, since we assume kinesthetic teaching and not high-dimensional image data. The log-likelihood objective

we want to maximize is:

$$\mathcal{L}(\psi, \mathcal{D}) = - \sum_{(s_i, a_i) \in \mathcal{D}} \sum_{k=1}^{K} \log \left| \det \left(J_{f_{\psi_k}}(z_{k-1}) \right) \right| \quad (14)$$

In order to avoid learning density functions $p_K(z_K)$ that exhibit large changes whenever $z_K = (s, a)$ changes slightly, we regularize the Jacobian of the learned density with respect to its input z_K. Our final training cost for the shaping potential based on normalizing flows is:

$$\mathcal{L}_{\text{flow}}(\psi, \mathcal{D}) = \mathcal{L}(\psi, \mathcal{D}) + \eta ||\nabla_{z_K} \log p_K(z_K)||^2 \quad (15)$$

Once the optimal parameters ψ^* are identified from the training process, we use the following shaping potential:

$$\Phi_{\psi^*, c}(s, a) = c \log \left(p_{\psi^*}(s, a) + \epsilon \right) \quad (16)$$

with $z_K = (s, a)$, $c \in \mathbb{R}^+$ a hyperparameter, and ϵ is a small constant to prevent numerical issues and the log probability from going to negative infinity.

Scalability: We note that if we had chosen to make the policy input be high-dimensional, for example image-based, our current model with $z_K = (s, a)$ would be very slow to train due to the cost of evaluating the Jacobian in Eqn. 16 and the autoregressive structure of the flow transform in Eqn. 13. That said, as we will see in the experimental results section, we have used normalizing flow shaping potentials with dimension of s, a being around 30 without any issues.

C. Potentials Based On Generative Adversarial Networks

The second type of state-action shaping potentials that we consider in this paper are functions $\Phi_{\psi, c}(s, a) = c \, D_\psi(s, a)$, trained on demonstration data $\mathcal{D} = \{(s_i, a_i), i = 1...N\}$, where $D_\psi(s, a)$ is the discriminator of a Generative Adversarial Network (GAN) [4]. These models also include a generative model $G_\phi(z) = \widetilde{x}$ that accepts a noise input $z \sim \mathcal{N}(0, I_d)$ and transforms it into a more structured random variable $\widetilde{x} \in \mathbb{R}^d$.

Training the generator and the discriminator is not done via maximum likelihood in this case, but through a minimax optimization problem. Let $p_r(x)$ be the real distribution of the random variable x and $p_\phi(x)$ is the distribution induced by the generator. The end goal of the training process is to optimize the parameters of the generator, so that the distance between the real distribution and the generated distribution is minimized. The discriminator parameters are optimized so that its output is high on real samples and low on (fake) generated samples.

We follow Arjovsky et al. [26] and Gulrajani et al [27] to estimate the Wasserstein-1, or Earth Mover's distance, in order to evaluate the cost of the optimal transport plan between two probability distributions p_r and p_ϕ:

$$W(p_r, p_\phi) = \inf_{\gamma \in \Pi(p_r, p_\phi)} \mathbb{E}_{(x,y) \sim \gamma(x,y)} [||x - y||] \quad (17)$$

where $\gamma(x, y)$ indicates how much mass needs to be transported from x to y to transform the distribution p_r to p_ϕ, and $\Pi(p_r, p_\phi)$ is the set of all joint distributions, whose marginals

are p_r and p_ϕ. Given a fixed generator G_ϕ, the intractable definition above is equivalent to the more tractable one:

$$W(p_r, p_\phi) = \sup_{D \in \mathcal{F}} \left[\mathbb{E}_{x \sim p_r} [D(x)] - \mathbb{E}_{\widetilde{x} \sim p_\phi} [D(\widetilde{x})] \right] \quad (18)$$

where $\mathcal{F} = \{D : \mathbb{R}^d \to \mathbb{R} \text{ such that } ||D||_L \le 1\}$ is the set of discriminator functions with Lipschitz constant 1. Sampling from p_ϕ is done by $z \sim \mathcal{N}(0, I_d)$ and $\widetilde{x} = G_\phi(z)$. To impose the Lipschitz constant of 1 on the discriminator we follow WGAN-GP in Gulrajani et al. [27], and impose a soft constraint to its gradient. The approximate Wasserstein distance can be computed this way:

$$L_1(\psi, \phi) = \mathbb{E}_{x \sim p_r} [D_\psi(x)] - \mathbb{E}_{\widetilde{x} \sim p_\phi} [D_\psi(\widetilde{x})] \quad (19)$$

$$L_2(\psi, \phi) = \mathbb{E}_{\widehat{x} \sim p_\phi} \left[(||\nabla_{\widehat{x}} D_\psi(\widehat{x})|| - 1)^2 \right] \quad (20)$$

$$\widetilde{W}(p_r, p_\phi) = \max_{\psi} L_1(\psi, \phi) - \alpha L_2(\psi, \phi) \quad (21)$$

where $\widehat{x} = \epsilon x + (1 - \epsilon)\widetilde{x}$ with $\epsilon \sim U[0, 1], x \sim p_r, \widetilde{x} \sim p_\phi$ is used to enforce the Lipschitz constraint on samples between the real distribution and the generated distribution, since the Lipschitz constant needs to be 1 for every possible input to the discriminator.

The approximate Wasserstein distance in Eqn. 21 corresponds to a fixed generator. For the generator to improve and minimize this distance, we solve the following problem:

$$\psi^*, \phi^* = \underset{\phi}{\mathrm{argmin}} \max_{\psi} L_1(\psi, \phi) - \alpha L_2(\psi, \phi) \quad (22)$$

The shaping potential then becomes:

$$\Phi_{\psi^*, c}(s, a) = c \, D_{\psi^*}(s, a) \quad (23)$$

Scalability: Unlike the potential based on Masked Autoregressive Flows, in this case, training the potential on high-dimensional demonstration data is scalable as GAN training has been demonstrated to produce realistic images of faces at high resolution [28].

D. Combining Reinforcement and Imitation Learning via Shaping

We now show how to integrate the learned shaping potentials in a model-free reinforcement learning method. We use Twin Delayed Deterministic Policy Gradient (TD3) [29] since it is one of the best performing model-free RL methods at the time of writing. That said, our potential-based shaping method is agnostic to the RL method used.

TD3 is an actor-critic method that maintains two critic networks for the Q function and one actor network for the deterministic policy. The use of the double-Q networks helps by reducing overestimation bias in the Q-function, which leads to sub-optimality in the learned policy. TD3 also updates the policy network less frequently than the Q value network to minimize error in Q value estimation. This is done by updating the policy after every d updates of the value network. In addition, TD3 smooths the Q value estimation by adding a small amount of noise ϵ to the target policy.

Algorithm 1 TD3 with Demonstrations via Reward Shaping

Offline pre-training

1: Collect demonstrations $\mathcal{D} = \{(s_i, a_i), i = 1...N\}$
2: Train shaping potential $\Phi_{\psi^*, c}(s, a)$ from Eqn. 16 or 23

3: Given MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma)$
4: Consider MDP $\widetilde{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \widetilde{r}, \gamma)$ from Eqn. 5 with
   $\widetilde{r}_t = r(s_t, a_t, s_{t+1}) + \gamma\Phi_{\psi^*, c}(s_{t+1}, a_{t+1}) - \Phi_{\psi^*, c}(s_t, a_t)$

TD3 ([29]) training with shaping

5: Initialize two critic networks for $\widetilde{M}$: $\widetilde{Q}_{\theta_1}, \widetilde{Q}_{\theta_2}$
6: Initialize actor network π_ϕ
7: Initialize target networks $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2, \phi' \leftarrow \phi$
8: Initialize replay buffer $\mathcal{B}$ to empty
9: **while** not converged **do**
10:    **for** episode $e = 1...E$ **do**
11:       **for** step $t = 1...T$ **do**
12:          Apply action $a = \pi_\phi(s) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$
13:          Observe reward r and new state s' from $\mathcal{M}$
14:          Store transition tuple (s, a, r, s') in $\mathcal{B}$
15:    **for** batch $b = 1...B$ **do**
16:       Sample mini-batch $\mathcal{B}_b$ of (s, a, r, s') from $\mathcal{B}$
17:       Sample mini-batch $\mathcal{D}_b$ of (s_d, a_d) from $\mathcal{D}$
18:       $a' \leftarrow \pi_{\phi'}(s') + \epsilon, \epsilon \sim \text{clip}(\mathcal{N}(0, \sigma'), -\delta, \delta)$
19:       Target value
         $y = \widetilde{r} + \gamma\min\{\widetilde{Q}_{\theta_1'}(s', a'), \widetilde{Q}_{\theta_2'}(s', a')\}$
20:       Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} \sum(y - \widetilde{Q}_{\theta_i}(s, a))^2$
21:       **if** $b \bmod d$ **then**
22:          Update policy
23:          $\phi \leftarrow \text{argmax}_\phi \sum_{s \in \mathcal{B}_b \cup \mathcal{D}_b}[\widetilde{Q}_{\theta_1}(s, \pi_\phi(s)) + \Phi_{\psi^*, c}(s, \pi_\phi(s))]$
24:    Update target networks
25:       $\theta_i' \leftarrow \tau\theta_i + (1 - \tau)\theta_i'$
26:       $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$

IV. EVALUATION

We evaluate our method both in simulation and on a real robot. Our aim is to clarify the following questions:

A) Does our method exceed the performance of (a) behavioral cloning and (b) pure RL?
B) Is our method robust to random seeds?
C) Is our method robust to sub-optimal demonstrations? In particular, does it do better than RL with behavioral cloning, as formulated in Eqn. 2?
D) Is our method practical on a real robot?

We answer all these questions in the affirmative and we analyse our experimental results below.

A. Robustness to Random Seeds

The issue of robustness of policies learned via reinforcement is intricately linked to the choice of random seeds, which determine the sequence of pseudorandom number generation that will drive the exploration process, as well as the random dynamics of the environment. Henderson
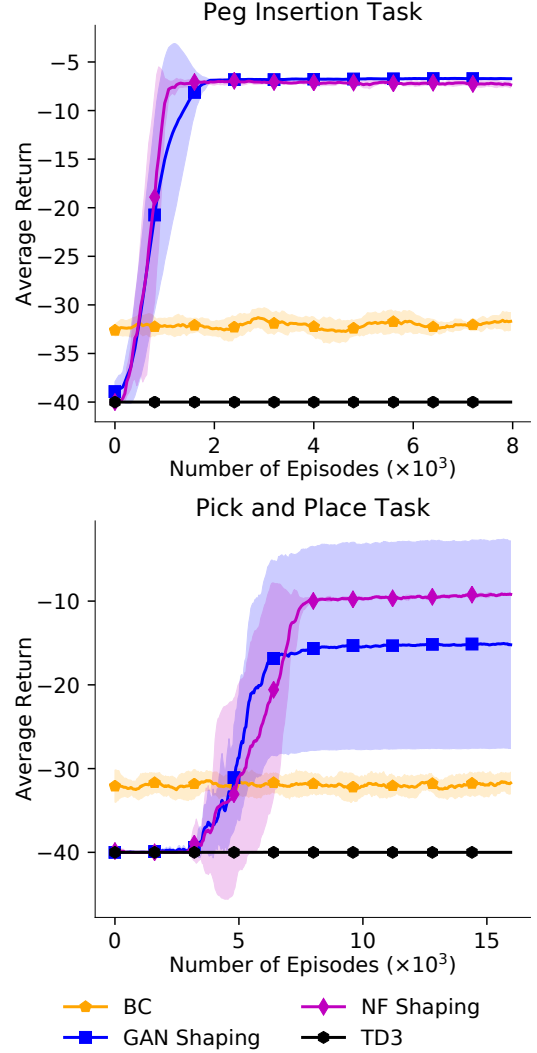

Fig. 2. GAN and Normalizing Flow (NF) shaping and baseline results for *peg insertion* and *pick and place* tasks on the Fetch environment adopted from OpenAI Gym. The initial position of the gripper is selected randomly but at a certain distance away from the hole, and demonstrations are near-optimal. In both cases, both RL + shaping methods converge to the optimal policy. TD3 fail to converge due to insufficient exploration, i.e. it never finds the goal state. Behavioral Cloning (BC) only succeeds when the arm is initialized to certain states. The empirical mean has been computed from 5 random seeds, and the error bars represent 1σ standard deviation.

et al [30] showed that many recent deep RL methods are extremely sensitive to the selection of random seeds.

We evaluated our method on complex manipulation tasks of *pick and place* and *peg insertion* in simulation.

(1) *peg insertion*: the end effector of the robot arm is initialized at a random location that is at a certain distance away from the hole, holding the peg. The location and orientation of the hole is fixed. A reward of 0 is given when more than half of the peg is inserted in the hole and -1 otherwise.

(2) *pick and place*: the object is placed at a random location that is at a certain distance away from both the end effector of the arm and the goal. The initial pose of the robot arm and the goal location are fixed. A reward of 0 is given

when the object is within a small threshold around the goal location and -1 otherwise.

For both environments, the episode length is set to 40, and the environment dos not terminate early.

Fig. 2 shows our method and baseline results for *peg insertion* and *pick and place* tasks. We consider two baselines for all experiments, namely Behavioral Cloning (BC), which is pure supervised learning, and pure model-free RL (TD3) without demonstrations. All empirical results are presented with empirical mean and a single standard deviation across 5 random seeds. The demonstration data for Fig. 2 are near optimal with no additional noise having been added.

Fig. 2 shows that while the two RL with shaping methods converge to goal, the Behavioral Cloning and pure RL method fail to explore sufficiently to find the goal area. The GAN shaping method converged to $4/5$ seeds, so the lower mean and the higher variance is due to that.

B. Robustness to Sub-Optimal Demonstrations

To illustrate the sensitivity of TD3+BC to noise, we simplified the *peg insertion* task by fixing the initial pose of the robot arm and limiting the state space to a 2D plane as shown in Figure 3. We provided sub-optimal demonstration data that encourages the agent to lift the peg to a high location and then perform the insertion, shown as red arrows. In addition, we also included demonstration data that pushes the learned policy away from the optimal trajectory, shown as green arrows. More crucially, these sub-optimal actions are given in areas of the state space where the optimal trajectory passes through, so the imitation objective is directly clashing with the RL objective.

The effect of these sub-optimal demonstrations is shown in Fig. 4. In particular, we see that the RL + shaping methods converge to the optimal policy, regardless of whether the demonstration data is near-optimal or not. On the other hand, the RL + Behavioral Cloning (with constraints, such as in Eqn. 2), is sensitive to the relative weight of the RL vs the imitation objective. When the role of the RL objective gets reduced by $\lambda = 0.0001$ compared to the imitation objective, the learned policy does not manage to find the optimal solution, while for other such settings (e.g. $\lambda = 0.01$) it does. This sensitivity to the relative weighting of the RL and imitation objective is undesirable, as it will affect any demonstrator forgetting schedule that adjusts this weight over time. GAIL [5], one of the leading imitation learning methods, is unable to solve the task in the presence of imperfect demonstrations.

C. Real Robot Experiments

For real robot experiments, we use a Franka Emika Panda 7DOF arm to perform the *peg insertion* task similar to the experiments in simulation as discussed in section A. The learned policy controls the end effector of the robot in Cartesian velocity control mode. In order to encourage faster convergence, we fixed the initial pose of the arm and modified the reward structure such that a reward of 0 is given when the peg is inside the hole, -0.5 is given when the
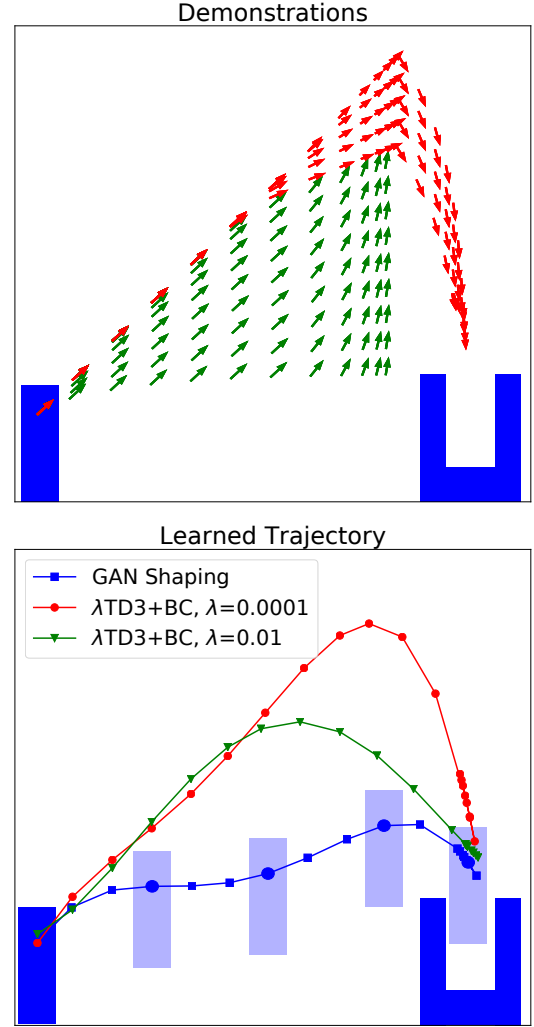

Fig. 3. Illustration of our method's robustness to noisy demonstration data. The top figure shows the provided demonstration data to all 3 methods: TD3 with GAN shaping and λTD3 + BC for $\lambda = 0.0001$ and $\lambda = 0.01$, which refers to the relative weight of the RL objective compared to the Behavioral Cloning (BC) objective. In this dataset, sub-optimality in the demonstration data is introduced by exaggerating the lift of the peg. Crucially, the suboptimal demonstrations are in an area of the state space where the optimal trajectory needs to pass through, so the two objectives will clash. The green curve shows the performance of the policy trained with one choice of λ which in turns puts more emphasis on the demonstration data, which leads to convergence to a sub-optimal policy. With careful tuning of λ, TD3 + BC can achieve better performance, whereas with GAN Shaping, the policy performs optimally.

agent is holding the peg above the hole, and -1 otherwise. During training, the number of episode steps is set to 100 and episodes do not terminate early.

The success rates of our method and the baselines on the *peg insertion* task on the real robot arm are presented in Fig. 5, where we compare pure RL and RL with GAN shaping. The failure of TD3 to discover the goal area inside the peg holder is not surprising given the long horizon and sparse rewards involved in the task. To generate demonstration data, a near-optimal predefined trajectory was used.

Fig. 5 shows the average return from 5 episodes. Since the episode length is set to 100, and the agent receives -1
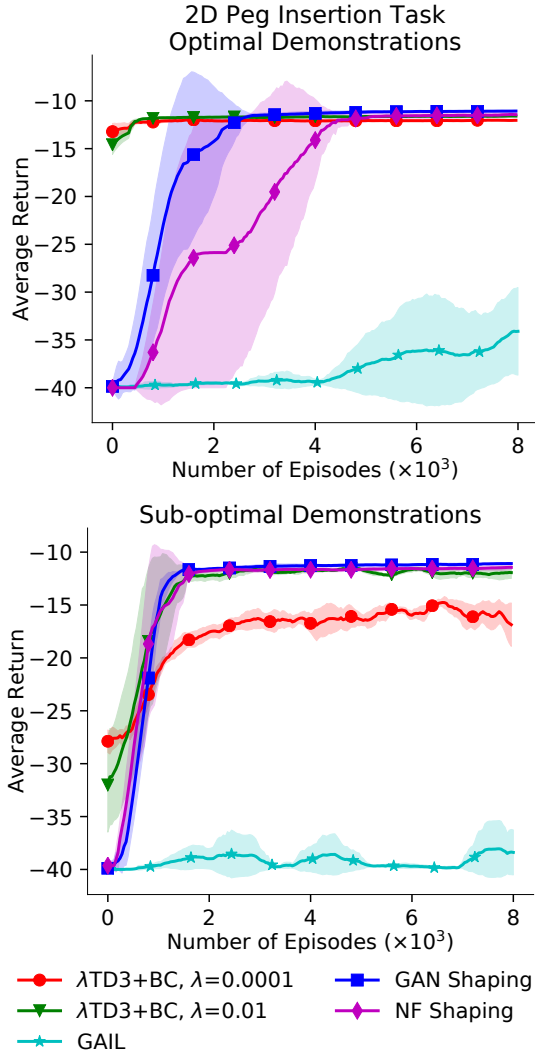
Fig. 4. Comparison of our method that involve TD3 with shaping vs λTD3 + BC with various λ weights, in the case of *optimal demonstrations* (top) and *sub-optimal demonstrations* (bottom). The latter are shown in Fig. 3. The hyperparameter λ refers to the relative weighting of the RL objective vs the behavioral cloning objective. These results show that TD3+BC is sensitive to this hyperparameter. For example, λ = 0.0001 does not find the optimal policy, whereas the other methods do. Both shaping methods outperform TD3 + BC and require minimal tuning and no relative weighting of objectives. GAIL, an imitation learning method, is not able to solve the task, neither with optimal nor with sub-optimal demonstrations.

when the peg is not above or in the hole, an average reward of −100 means the robot received no reward throughout the entire episode. We can see that with our method, RL with GAN Shaping, the robot is able to collect rewards in 20 steps. Note that the agent does not have access to this cumulative, dense reward during training. This dense return is used here for evaluation purposes only.

V. CONCLUSION

This paper addressed the problem of combining reinforcement learning with sub-optimal demonstrations. We made use of results from reward shaping and state-action potentials in order to model the demonstration data as *advice*, and not as a set of *constraints*, which is the most popular method currently in practice. We modeled the demonstration data as deep generative models, based on normalizing flows or
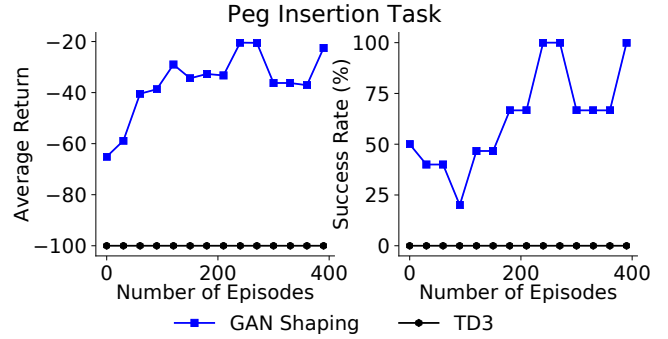

Fig. 5. Comparison of TD3 with TD3 with shaping from a GAN potential, trained from demonstrations on a *peg insertion* task. Our method finds a good policy in about 200 episodes. The performance reduction after that is due to the RL objective trying to optimize the trajectory of the peg so as to minimize time to arrive to the bottom of the hole. As it tries to optimize and straighten the trajectory, the peg starts hitting the holder more frequently, which delays the learning. To address this the reward can be modified to penalize contact with the holder.

Generative Adversarial Networks, and we showed that when used RL with generative model shaping is typically more robust than RL with behavioral cloning constraints, even in the presence of sub-optimal data. We showed that our method is practical on a real robot arm, in addition to validating our method in simulation.

ACKNOWLEDGMENT

REFERENCES

[1] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016.

[2] E. Wiewiora, G. Cottrell, and C. Elkan, "Principled methods for advising reinforcement learning agents," in *International Conference on Machine Learning*, 2003, pp. 792–799.

[3] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International Conference on Machine Learning*, vol. 37. PMLR, Jul 2015, pp. 1530–1538.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NeurIPS 27*, 2014, pp. 2672–2680.

[5] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *NeurIPS*, 2016, pp. 4565–4573.

[6] T. Hester, M. Vecerík, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, G. Dulac-Arnold, J. Agapiou, J. Z. Leibo, and A. Gruslys, "Deep q-learning from demonstrations," in *AAAI*, 2018, pp. 3223–3230.

[7] M. Vecerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. M. O. Heess, T. Rothörl, T. Lampe, and M. A. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *ArXiv*, vol. abs/1707.08817, 2017.

[8] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramar, R. Hadsell, N. de Freitas, and N. Heess, "Reinforcement and imitation learning for diverse visuomotor skills," in *Robotics: Science and Systems*, June 2018.

[9] A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *CoRR*, vol. abs/1709.10087, 2017.

[10] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, vol. 99, 1999, pp. 278–287.

[11] S. Devlin and D. Kudenko, "Dynamic potential-based reward shaping," in *11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, ser. AAMAS '12, 2012, pp. 433–440.

[12] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé, "Reinforcement learning from demonstration through shaping," in *International Conference on Artificial Intelligence*, ser. IJCAI'15, 2015, pp. 3352–3358.

[13] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith, "Using advice in model-based reinforcement learning," in *Reinforcement Learning and Decision Making (RLDM)*, 2017.

[14] S. Reddy, A. D. Dragan, and S. Levine, "SQIL: imitation learning via regularized behavioral cloning," *CoRR*, vol. abs/1905.11108, 2019. [Online]. Available: http://arxiv.org/abs/1905.11108

[15] A. Kurenkov, A. Mandlekar, R. Martin-Martin, S. Savarese, and A. Garg, "Ac-teach: A bayesian actor-critic method for policy learning with an ensemble of suboptimal teachers," 2019.

[16] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," *CoRR*, vol. abs/1709.10089, 2017.

[17] D. H. Grollman and A. G. Billard, "Robot learning from failed demonstrations," *International Journal of Social Robotics*, vol. 4, no. 4, pp. 331–342, Nov 2012.

[18] W. B. Knox and P. Stone, "Combining manual feedback with subsequent MDP reward signals for reinforcement learning," in *Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2010.

[19] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 97. Long Beach, California, USA: PMLR, Jun 2019, pp. 2052–2062.

[20] A. Kumar, J. Fu, G. Tucker, and S. Levine, "Stabilizing off-policy q-learning via bootstrapping error reduction," *CoRR*, vol. abs/1906.00949, 2019. [Online]. Available: http://arxiv.org/abs/1906.00949

[21] D. Precup, R. S. Sutton, and S. Dasgupta, "Off-policy temporal difference learning with function approximation," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01, 2001, pp. 417–424.

[22] R. Munos, T. Stepleton, A. Harutyunyan, and M. G. Bellemare, "Safe and efficient off-policy reinforcement learning," *CoRR*, vol. abs/1606.02647, 2016. [Online]. Available: http://arxiv.org/abs/1606.02647

[23] N. Jiang and L. Li, "Doubly robust off-policy evaluation for reinforcement learning," *CoRR*, vol. abs/1511.03722, 2015. [Online]. Available: http://arxiv.org/abs/1511.03722

[24] E. Wiewiora, "Potential-based shaping and q-value initialization are equivalent," *Journal of Artificial Intelligence Research*, vol. 19, pp. 205–208, 2003.

[25] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked autoregressive flow for density estimation," in *NeurIPS*, 2017, pp. 2338–2347.

[26] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, Aug 2017, pp. 214–223.

[27] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," in *Neural Information Processing Systems*, USA, 2017, pp. 5769–5779.

[28] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *International Conference on Learning Representations*, 2018.

[29] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *ICML*, 2018.

[30] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *AAAI*, 2018.