Reinforcement Learning from Imperfect Demonstrations

Yang Gao* [1] Huazhe(Harry) Xu* [1] Ji Lin [2] Fisher Yu [1] Sergey Levine [1] Trevor Darrell [1]

Abstract

Robust real-world learning should benefit from both demonstrations and interactions with the environment. Current approaches to learning from demonstration and reward perform supervised learning on expert demonstration data and use reinforcement learning to further improve performance based on the reward received from the environment. These tasks have divergent losses which are difficult to jointly optimize and such methods can be very sensitive to noisy demonstrations. We propose a unified reinforcement learning algorithm, Normalized Actor-Critic (NAC), that effectively normalizes the Q-function, reducing the Q-values of actions unseen in the demonstration data. NAC learns an initial policy network from demonstrations and refines the policy in the environment, surpassing the demonstrator's performance. Crucially, both learning from demonstration and interactive refinement use the same objective, unlike prior approaches that combine distinct supervised and reinforcement losses. This makes NAC robust to suboptimal demonstration data, since the method is not forced to mimic all of the examples in the dataset. We show that our unified reinforcement learning algorithm can learn robustly and outperform existing baselines when evaluated on several realistic driving games.

1. Introduction

Deep reinforcement learning (RL) has achieved significant success on many complex sequential decision-making problems. However, RL algorithms usually require a large amount of interactions with an environment to reach good performance (Kakade et al., 2003); initial performance may be nearly random, clearly suboptimal, and often rather dangerous in real-world settings such as autonomous driving. Learning from demonstration is a well-known alternative, but typically does not leverage reward, and presumes relatively small-scale noise-free demonstrations. We develop a new robust algorithm that can learn value and policy functions from state, action and reward (s, a, r) signals that either come from imperfect demonstration data or the environment.

Recent efforts toward policy learning which does not suffer from a suboptimal initial performance generally leverage an initial phase of supervised learning and/or auxiliary task learning. Several previous efforts have shown that demonstrations can speed up RL by mimicking expert data with a temporal difference regularizer (Hester et al., 2017) or via gradient-free optimization (Ebrahimi et al., 2017), yet these methods presume near-optimal demonstrations. (Jaderberg et al., 2016) and (Shelhamer et al., 2016) obtained better initialization via auxiliary task losses (e.g., predicting environment dynamics) in a self-supervised manner; policy performance is still initially random with these approaches.

A simple combination of several distinct losses can learn from demonstrations; however, it is more appealing to have a single principled loss that is applicable to learning both from the demonstration and from the environment. Our approach, Normalized Actor-Critic (NAC), uses a unified loss function to process both off-line demonstration data and on-line experience based on the underlying maximum entropy reinforcement learning framework (Toussaint, 2009; Haarnoja et al., 2017b; Schulman et al., 2017). Our approach enables robust learning from corrupted (or even partially adversarial) demonstrations that contains (s, a, r), because no assumption on the optimality of the data is required. A normalized formulation of the soft Q-learning gradient enables the NAC method, which can also be regarded as a variant of the policy gradient.

We evaluate our approach in a toy Minecraft Game, as well as two realistic 3D simulated environments, Torcs and Grand Theft Auto V (GTA V), with either discrete states and tabular Q functions, or raw image input and Q functions approximated by neural networks. Our experimental results outperform previous approaches on driving tasks with only a modest amount of demonstrations while tolerating significant noise in the demonstrations, as our method utilizes

*Equal contribution [1]Department of Electrical Engineering and Computer Science, UC Berkeley, CA, USA [2]Department of Electrical Engineering, Tsinghua University, Beijing, China. Correspondence to: Yang Gao <yg@eecs.berkeley.edu>, Huazhe Xu <huazhe_xu@eecs.berkeley.edu>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

rewards rather than simply imitates demonstrated behaviors.

We summarize the contributions in this paper as follows.

- We propose the NAC method for learning from demonstrations and discover its practical advantages on a variety of environments.
- To the best of our knowledge, we are the first to propose a unified objective, capable of learning from both demonstrations and environments, that outperforms methods including the ones with an explicit supervised imitation loss.
- Unlike other methods that utilize supervised learning to learn from demonstrations, our pure reinforcement learning method is robust to noisy demonstrations.

2. Preliminaries

In this section, we will briefly review the reinforcement learning techniques that our method is built on, including maximum entropy reinforcement learning and the soft Q-learning.

2.1. Maximum Entropy Reinforcement Learning

The reinforcement learning problem we consider is defined by a Markov decision process(MDP) (Thie, 1983). Specifically, the MDP is characterized by a tuple $<\mathbb{S}$,$\mathbb{A}$,R,T,$\gamma >$, where $\mathbb{S}$ is the set of states, $\mathbb{A}$ is the set of actions, $R(s,a)$ is the reward function, $T(s,a,s') = P(s'|s,a)$ is the transition function and γ is the reward discount factor. An agent interacts with the environment by taking an action at a given state, receiving the reward, and transiting to the next state.

In the standard reinforcement learning setting (Sutton & Barto, 1998), the goal of an agent is to learn a policy π_{std}, such that an agent maximizes the future discounted reward:

$$\pi_{std} = \operatorname*{argmax}_{\pi} \sum_t \gamma^t \mathop{\mathbb{E}}_{s_t, a_t \sim \pi} [R_t] \tag{1}$$

Maximum entropy policy learning (Ziebart, 2010; Haarnoja et al., 2017b) uses an entropy augmented reward. The optimal policy will not only optimize for discounted future rewards, but also maximize the discounted future entropy of the action distribution:

$$\pi_{ent} = \operatorname*{argmax}_{\pi} \sum_t \gamma^t \mathop{\mathbb{E}}_{s_t, a_t \sim \pi} [R_t + \alpha H(\pi(\cdot|s_t))] \tag{2}$$

where α is a weighting term to balance the importance of the entropy. Unlike previous attempts that only adds the entropy term at a single time step, maximum entropy policy learning maximizes the discounted future entropy over the whole trajectory. Maximum entropy reinforcement learning has many benefits, such as better exploration in multi-modal problems and connections between Q-learning and the actor-critic method (Haarnoja et al., 2017b; Schulman et al., 2017).

2.2. Soft Value Functions

Since the maximum entropy RL paradigm augments the reward with an entropy term, the definition of the value functions naturally changes to

$$Q_\pi(s,a) = R_0 + \mathop{\mathbb{E}}_{(s_t,a_t)\sim\pi} \sum_{t=1}^{\infty} \gamma^t (R_t + \alpha H(\pi(\cdot|s_t))) \tag{3}$$

$$V_\pi(s) = \mathop{\mathbb{E}}_{(s_t,a_t)\sim\pi} \sum_{t=0}^{\infty} \gamma^t (R_t + \alpha H(\pi(\cdot|s_t))) \tag{4}$$

where π is some policy that value functions evaluate on. Given the state-action value function $Q^*(s,a)$ of the optimal policy, (Ziebart, 2010) shows that the optimal state value function and the optimal policy could be expressed as:

$$V^*(s) = \alpha \log \sum_a \exp(Q^*(s,a)/\alpha) \tag{5}$$

$$\pi^*(a|s) = \exp((Q^*(s,a) - V^*(s))/\alpha) \tag{6}$$

2.3. Soft Q-Learning and Policy Gradient

With the entropy augmented reward, one can derive the soft versions of Q-learning (Haarnoja et al., 2017a) and policy gradient. The soft Q-learning gradient is given by

$$\nabla_\theta Q_\theta(s,a)(Q_\theta(s,a) - \hat{Q}(s,a)) \tag{7}$$

where $\hat{Q}(s,a)$ is a bootstrapped Q-value estimate obtained by $R(s,a) + \gamma V_Q(s')$. Here, $R(s,a)$ is the reward received from the environment, V_Q is computed from $Q_\theta(s,a)$ with Equation (5). We can also derive a policy gradient, which includes the gradient of form:

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(\hat{Q}_\pi - b(s_t)) + \alpha \nabla_\theta H(\pi_\theta(\cdot|s_t))\right] \tag{8}$$

where $b(s_t)$ is some arbitrary baseline (Schulman et al., 2017).

3. Robust Learning from Demonstration and Reward

Given a set of demonstrations that contains (s,a,r,s') and the corresponding environment, an agent should perform appropriate actions when it starts the interaction, and continue to improve. Although a number of off-policy RL algorithms could in principle be used to learn directly from off-policy

Algorithm 1 Normalized Actor-Critic for Learning from Demonstration

```
**$\theta$: parameters for the rapid Q network, $\theta'$: parameters for the target Q network, $\mathcal{D}$: demonstrations collected by human or a trained policy network, $T$: target network update frequency, $\mathcal{M}$: replay buffer, $k$: number of steps to train on the demonstrations**
for step t ∈ $\{1, 2, ...\}$ do
    if t $\leq k$ then
        Sample a mini-batch of transitions from $\mathcal{D}$
    else
        Start from s, sample $a$ from $\pi$, execute $a$, observe ($s'$,$r$) and store $(s, a, r, s')$ in $\mathcal{M}$
        Sample a mini-batch of transitions from $\mathcal{M}$
    end if
    Update $\theta$ with gradient: $\nabla_\theta J_{PG} + \nabla_\theta J_V$
    if t mod T = 0 then
        $\theta' \leftarrow \theta$
    end if
end for
```

demonstration data, standard methods can suffer from extremely poor performance when trained entirely on demonstration data. This can happen when the demonstration set is a strongly biased sample of the environment transitions, which violates the assumptions of many off-policy RL methods. Although they are closely related, off-policy learning and learning from demonstrations are different problems. In section 5, we show that Q-learning completely fails on the demonstration data. The intuition behind this problem is that if the Q-function is trained only on good data, it has no way to understand why the action taken is appropriate: it will assign a high Q-value, but will not necessarily assign a low Q-value to other alternative actions.

The framework of soft optimality provides us with a natural mechanism to mitigate this problem by *normalizing* the Q-function over the actions. Our approach, Normalized Actor-Critic (NAC), utilizes soft policy gradient formulations described in Section 2.3 to obtain a Q-function gradient that reduces the Q-values of actions that were not observed along the demonstrations. In other words, without data to indicate otherwise, NAC will opt to follow the demonstrations. This method is a well-defined RL algorithm without any auxiliary supervised loss and hence, it is able to learn without bias in the face of low-quality demonstration data. We will first describe our algorithm, and then discuss why it performs well when trained on the demonstrations.

3.1. Normalized Actor-Critic for Learning from Demonstration

We propose a unified learning from demonstration approach, which applies the normalized actor-critic updates to both off policy demonstrations and in-environment transitions. The NAC method is derived from the soft policy gradient objective with a Q function parametrization. Specifically, we take gradient steps to maximize the future reward objective (Equation (2)), and parametrize π and V in terms of Q (Equation (6) and (5)). As derived in the appendix, the updates for the actor and critic are:

$$\nabla_\theta J_{PG} = \mathop{\mathbb{E}}_{s,a\sim\pi_Q}\left[(\nabla_\theta Q(s,a) - \nabla_\theta V_Q(s))(Q(s,a) - \hat{Q})\right] \tag{9}$$

$$\nabla_\theta J_V = \mathop{\mathbb{E}}_{s}\left[\nabla_\theta \frac{1}{2}(V_Q(s) - \hat{V}(s))^2\right] \tag{10}$$

where V_Q and π_Q are deterministic functions of Q:

$$V_Q(s) = \alpha \log \sum_a \exp(Q(s,a)/\alpha) \tag{11}$$

$$\pi_Q(a|s) = \exp((Q(s,a) - V_Q(s))/\alpha) \tag{12}$$

$\hat{Q}(s,a), \hat{V}(s)$ are obtained by:

$$\hat{Q}(s,a) = R(s,a) + \gamma V_Q(s') \tag{13}$$

$$\hat{V}(s) = \mathop{\mathbb{E}}_{a\sim\pi_Q}[R(s,a) + \gamma V_Q(s')] + \alpha H(\pi_Q(\cdot|s)) \tag{14}$$

The difference is the $\nabla_\theta V(s)$ term comparing NAC's actor update term (Equation (9)) with the soft Q update (Equation (7)). We emphasize the normalization effect of this term: it avoids pushing up the Q-values of actions that are not demonstrated. The mechanism is explained in Section 3.2.

The expectations in Eq. (9) and Eq. (10) are taken with respect to π_Q. In the demonstration set, we only have transition samples from the behavioral policy $\mu(a|s)$. To have a proper policy gradient algorithm, we can employ importance sampling to correct the mismatch. To be specific, when estimating $\mathbb{E}_{(s,a)\sim\pi_Q}[f(s,a)]$, we estimate $\mathbb{E}_{(s,a)\sim\mu}[f(s,a)\beta]$, where $\beta = \min\left\{\frac{\pi_Q(a|s)}{\mu(a|s)}, c\right\}$ and c is some constant that prevents the importance ratio from being too large. Although the importance weights are needed to formalize our method as a proper policy gradient algorithm, we find in our empirical evaluation that the inclusion of these weights consistently reduces the performance of our method. We found that omitting the weights results in better final performance even when training entirely on demonstration data. For this reason, our final algorithm does not use importance sampling.

We summarize the proposed method in Algorithm 1. Our method uses samples from the demonstrations and the replay buffer, rather than restricting the samples to be on policy as in standard actor-critic methods. Similar to DQN, we utilize a target network to compute $\hat{Q}(s,a)$ and $\hat{V}(s)$, which stabilizes the training process.

3.2. Analysis of the Method

We provide an intuitive analysis in this section to explain why our method can learn from demonstrations while other reinforcement learning methods, such as Q-learning cannot. The states and actions in the demonstrations generally have higher Q-values than other states. Q-learning will push up $Q(s, a)$ values in a sampled state s. However, if the values for the bad actions are not observed, the Q-function has no way of knowing whether the *action* itself is good, or whether all actions in that *state* are good, so the demonstrated action will not necessarily have a higher Q-value than other actions in the demonstrated state.

Comparing the actor update (Eq. (9)) of our method with the soft Q-learning (Eq. (7)) update, our method includes an extra term in the gradient: $-\nabla_\theta V_Q(s)$. This term falls out naturally when we derive the update from a policy gradient algorithm, rather than a Q-learning algorithm. Intuitively, this term will decrease $V_Q(s)$ when increasing $Q(s, a)$ and vice versa, since $\nabla_\theta Q(s, a)$ and $-\nabla_\theta V_Q(s)$ have different signs. Because $V_Q(s) = \alpha \log \sum_a \exp(Q(s, a)/\alpha)$, decreasing $V_Q(s)$ will prevent $Q(s, a)$ from increasing for the actions that are not in the demonstrations. That is why the aforementioned normalization effect emerges with the extra $\nabla_\theta V_Q(s)$ term.

Besides having the normalizing behaviors, NAC is also less sensitive to noisy demonstrations. Since NAC is an RL algorithm, it is naturally resistant to poor behaviors. One could also see this with similar analysis as above. When there is a negative reward in the demonstrations, $Q(s, a)$ tends to decrease and $V_Q(s)$ tends to increase, hence having the normalizing behavior in a reverse direction.

Moreover, NAC provides a single and principled approach to learn from both demonstrations and environments. It avoids the use of imitation learning. Therefore, besides its natural robustness to imperfect demonstrations, it is a more unified approach comparing with other methods.

Figure 1. Sample frames from Torcs (upper) and GTA (lower).

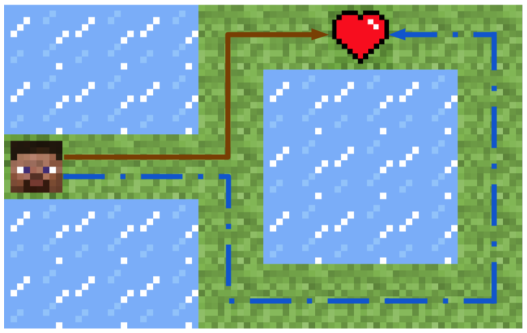


Figure 2. The Toy Minecraft environment. We aim to learn a policy that moves the agent to the goal. Two possible paths are shown, the shorter optimal one (solid, brown) and the longer sub-optimal one (dashed, blue). See text for details about the environment (Sec. 5.1) and comparison between NAC and DQfD on this environment (Sec. 5.3).

4. Related Work

4.1. Maximum entropy Reinforcement Learning

Maximum entropy reinforcement learning has been explored in a number of prior works (Todorov, 2008; Toussaint, 2009; Ziebart et al., 2008), including several recent works that extend it into a deep reinforcement learning setting (Nachum et al., 2017; Haarnoja et al., 2017b; Schulman et al., 2017). However, most of those works do not deal with the learning from demonstration settings. (Haarnoja et al., 2017b) and (Schulman et al., 2017) propose maximum entropy RL methods to learn from environments. PCL (Nachum et al., 2017) is the only prior work that studies the learning from demonstration task with the maximum entropy RL framework. Unlike their method where the loss is derived from an objective similar to Bellman error, our method is derived from policy gradient. Instead of minimizing Bellman errors, policy gradient directly optimizes future accumulated reward. As shown in Section 5, our method has large performance advantage compared with PCL, due to the different objective function.

Our method not only admits a unified objective on both demonstrations and environments but also performs better than alternative methods, such as PCL (Nachum et al., 2017) and DQfD (Hester et al., 2017). To the best of our knowledge, our proposed method is the first unified method across demonstrations and environments that outperforms methods including the ones with explicit supervised imitation loss such as DQfD.

4.2. Learning from Demonstration

Most of the prior learning from demonstration efforts assume the demonstrations are perfect, *i.e.* the ultimate goal is to copy the behaviors from the demonstrations. Imitation learning is one of such approaches, examples including (Xu et al., 2016; Bojarski et al., 2016). Extensions such as DAGGER (Ross et al., 2011) are proposed to have the expert in the loop, which further improves the agent's performance. Recently, (Ho & Ermon, 2016; Wang et al., 2017; Ziebart et al., 2008) explore an adversarial paradigm for the behavior cloning method. Another popular paradigm is Inverse Reinforcement Learning (IRL) (Ng et al., 2000; Abbeel & Ng, 2004; Ziebart et al., 2008). IRL learns a reward model which explains the demonstrations as optimal behavior.

Instead of assuming that the demonstrations are perfect, our pure RL method allows imperfect demonstrations. Our method *learns* which part of the demonstrations is good and which part is bad, unlike the methods that simply imitate the demonstrated behaviors. We follow the Reinforcement Learning with Expert Demonstrations (RLED) framework (Chemali & Lazaric, 2015; Kim et al., 2013; Piot et al., 2014), where both rewards and actions are available in the demonstrations. The extra reward in the demonstrations allows our method to be aware of poor behaviors in the demonstrations. DQfD (Hester et al., 2017) is a recent method that also uses rewards in the demonstrations. It combines an imitation hinge loss with the Q-learning loss in order to learn from demonstrations and transfer to environments smoothly. Due to the use of the imitation loss, DQfD is more sensitive to noisy demonstrations, as we show in the experiment section.

4.3. Off-policy Learning

It is tempting to apply various off-policy methods to the problem of learning from demonstration, such as policy gradient variants (Gu et al., 2017; 2016; Degris et al., 2012; Wang et al., 2016), Q-learning (Watkins & Dayan, 1992) and Retrace (Munos et al., 2016). However, we emphasize that off-policy learning and learning from demonstration are different problems. For most of the off-policy methods, their convergence relies on the assumption of visiting each (s, a) pair infinitely many times. In the learning from demonstration setting, the samples are highly biased and off-policy method can fail to learn anything from the demonstrations, as we explained the Q-learning case in Section 3.

5. Results

Our experiments address several questions: (1) Can NAC benefit from both demonstrations and rewards? (2) Is NAC robust to ill-behaved demonstrations? (3) Can NAC learn meaningful behaviors with a limited amount of demonstrations? We compare our algorithm with DQfD (Hester et al., 2017), which has been shown to learn efficiently from demonstrations and to preserve performance while acting in an environment. Other methods include supervised behavioral cloning method, Q-learning, soft Q-learning, the version of our method with importance sampling weighting, PCL and Q-learning, soft Q-learning without demonstrations.

5.1. Environments

We evaluate our result in a grid world, the toy Minecraft (Fig. 2), as well as two realistic 3D simulated environments, Torcs and Grand Theft Auto V (GTA V) shown in Figure 1.

Toy Minecraft: The toy Minecraft is a customized grid world environment. As shown in Figure 2, the agent starts from the left and would like to reach the final goal (marked as a heart). The agent can walk on the green grass and go into the blue water ends the episode. The input to the agent is its current (x, y) location. At each step, the agent can move *Up*, *Down*, *Left* or *Right*. It gets a reward of 1 when reaching the goal, 0 otherwise. For more details, please refer to the OpenAI gym Frozen-Lake environment (Brockman et al., 2016).

Torcs: Torcs is an open-source racing game that has been used widely as an experimental environment for driving. The goal of the agent is to drive as fast as possible on the track while avoiding crashes. We use an oval two-lane racing venue in our experiments. The input to the agent is an 84×84 gray scale image. The agent controls the vehicle at 5Hz, and at each step, it chooses from a set of 9 actions which is a Cartesian product between {left, no-op, right} and {up, no-op, down}. We design a dense driving reward function that encourages the car to follow the lane and to avoid collision with obstacles. [1]

GTA: Grand Theft Auto is an action-adventure video game with goals similar in part to the Torcs game, but with a more diverse and realistic surrounding environment, including the presence of other vehicles, buildings, and bridges. The agent observes 84×84 RGB images from the environment. It chooses from 7 possible actions from {left-up, up, right-up, left, no-op, right, down} at 6Hz. We use the same reward function as in Torcs.

5.2. Comparisons

We compare our approach with the following methods:

[1] $reward = (1 - \mathbb{1}_{damage})[(\cos\theta - \sin\theta - lane\_ratio) \times speed] + \mathbb{1}_{damage}[-10]$, where $\mathbb{1}_{damage}$ is an indicator function of whether the vehicle is damaged at the current state. $lane\_ratio$ is the ratio between distance to lane center and lane width. θ is the angle between the vehicle heading direction and the road direction.

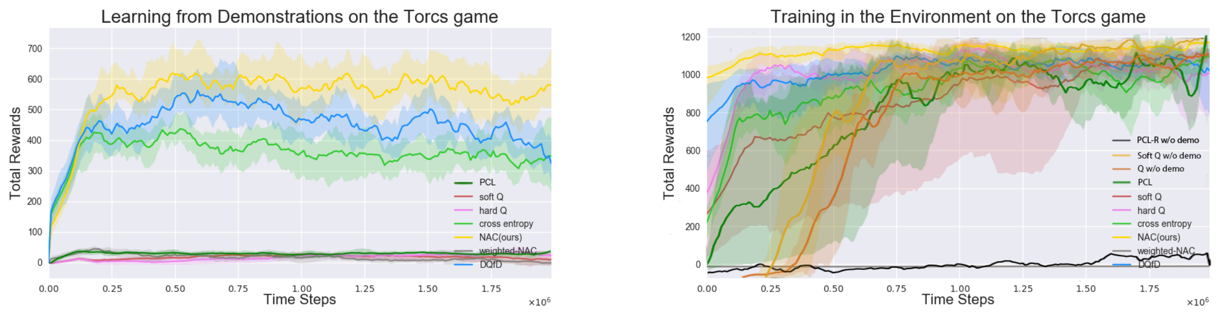


Figure 3. Performances on the Torcs game. The x-axis shows the training iterations. The y-axis shows the average total rewards. Solid lines are average values over 10 random seeds. Shaded regions correspond to one standard deviation. The left figure shows the performance for each agent when they only learn from demonstrations, while the right one shows the performance for each agent when they interact with the environments after learning from demonstrations. Our method consistently outperforms other methods in both cases.

- **DQfD:** the method proposed by (Hester et al., 2017). For the learning from demonstration phase, DQfD combines a hinge loss with a temporal difference (TD) loss. For the finetuning-in-environment phase, DQfD combines a hinge loss on demonstrations and a TD loss on both the demonstrations and the policy-generated data. To alleviate over-fitting issues, we also include weight decay following the original paper.
- **Q-learning:** the classic DQN method (Mnih et al., 2015). We first train DQN with the demonstrations in a replay buffer and then finetune in the environment with regular Q-learning. Similar to DQfD, we use a constant exploration ratio of 0.01 in the finetuning phase to preserve the performance obtained from the demonstrations. We also train from scratch a baseline DQN in the environment, without any demonstration.
- **Soft Q-learning:** similar to the Q-learning method, but with an entropy regularized reward. This is the method proposed by (Haarnoja et al., 2017a; Schulman et al., 2017). We also include the soft Q-learning trained without demonstration, as another baseline.
- **Behavior cloning with Q-learning:** the naive way of combining cross-entropy loss with Q-learning. First we perform behavior cloning with cross-entropy loss on the demonstrations. Then we treat the logit activations prior the softmax layer as an initialization of the Q function and finetune with regular Q-learning in the environment.
- **Normalized actor-critic with importance sampling:** the NAC method with the importance sampling weighting term mentioned in Sec 3.1. The importance weighting term is used to correct the action distribution mismatch between the demonstration and the current policy.
- **Path Consistency Learning (PCL):** the PCL (Nachum et al., 2017) method that minimizes the soft path consistency loss. The method proposed in the original paper (denoted as *PCL-R*) does not utilize a target network. We find that PCL-R does not work when it is trained from scratch in the visually complex environment. We stabilize it by adding a target network (denoted as *PCL*), similar to (Haarnoja et al., 2017a).

5.3. Experiments on Toy Minecraft

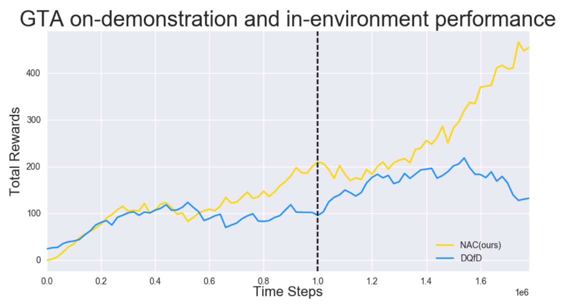
To understand the basic properties of our proposed method, we design the toy Minecraft environment. In this experiment, the state is simply the location of the agent. We use a tabular Q function. With those settings, we hope to reveal some differences between our NAC algorithm and algorithms that incorporate supervised loss.

As shown in Figure 2, there are only two paths to reach the goal. In terms of the discounted reward, the shorter path is more favorable. To make the problem more interesting, we provide the longer suboptimal path as the demonstrations. We found that in the learning from demonstration phase, both DQfD and NAC have learned the suboptimal path since both methods do not have access to the environment and could not possibly figure out the optimal path. When the two methods finetune their policies in the environment, NAC succeeds in finding the optimal path, while DQfD stucks with the suboptimal one. It is because DQfD has the imitation loss, thus preventing it from deviating from the original solution.

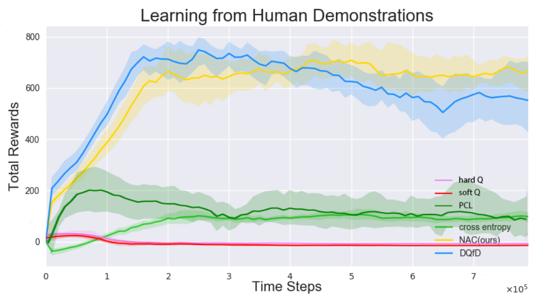
5.4. Comparison to Other Methods

We compare our NAC method with other methods on 300k transitions. The demonstrations are collected by a trained Q-learning expert policy. We execute the policy in the environment to collect demonstrations. To avoid deterministic executions of the expert policy, we sample an action randomly with probability 0.01.

To explicitly compare different methods, we show separate



(a) The on-demonstration and in-environment performance of the NAC and DQfD methods on GTA. The vertical line separates the learning from demonstration phase and finetuning in environment phase. Our method consistently outperforms DQfD in both phases.



(b) Performances on the Torcs game with human demonstrations. DQfD performs well in the beginning, but overfits in the end. The behavior cloning method is much worse than NAC and DQfD. Our NAC method performs best at convergence.

Figure 4. Performance on GTA (left) and performance on Torcs with human demonstrations (right)

figures for performances on the demonstrations and inside the environments. In Fig 3, we show that our method performs better than other methods on demonstrations. When we start finetuning, the performance of our method continues to increase and reaches peak performance faster than other methods. DQfD (Hester et al., 2017) has similar behavior to ours but has lower performance. Behavior cloning learns well on demonstrations, but it has a significant performance drop while interacting with environments. All the methods can ultimately learn by interacting with the environment but only our method and DQfD start from a relatively high performance. Empirically, we found that the importance weighted NAC method does not perform as well as NAC. The reason might be the decrease in the gradient bias is not offset sufficiently by the increase in the gradient variance. Without the demonstration data, Q-learning (Q w/o demo) and soft Q-learning (soft-Q w/o demo) suffer from low performance during the initial interactions with the environment. The original PCL-R method (PCL-R w/o demo) fails to learn even when trained from scratch in the environments. The improved PCL method (PCL) is not able to learn on the demonstrations, but it can learn in the environment.

We also test our method on the challenging GTA environment, where both the visual input and the game logic are more complex. Due to the limit of the environment execution speed, we only compare our method with DQfD. As shown in Fig. 4a, our method outperforms DQfD both on the demonstrations and inside the environment.

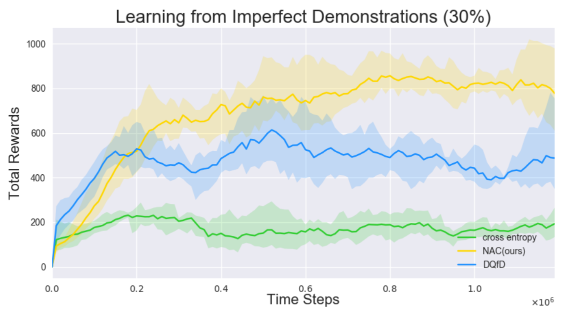
5.5. Learning from Human Demonstrations

For many practical problems, such as autonomous driving, we might have a large number of human demonstrations, but no demonstration available from a trained agent at all. In contrast to a scripted agent, humans usually perform actions diversely, both from multiple individuals (e.g. conservative players will slow down before a U-turn; aggressive players will not) and a single individual (e.g. a player may randomly turn or go straight at an intersection). Many learning from demonstration methods do not study this challenging case, such as (Ho & Ermon, 2016). We study how different methods perform with diverse demonstrations. To collect human demonstrations, we asked 3 non-expert human players to play TORCS for 3 hours each. Human players control the game with the combination of four arrow keys, at 5Hz, the same rate as the trained agent. In total, we collected around 150k transitions. Among them, 4.5k transitions are used as a validation set to monitor the Bellman error. Comparing with data collected from a trained agent, the data is more diverse and the quality of the demonstrations improves naturally when the players get familiar with the game.

In Fig. 4b, we observe that the behavior cloning method performs much worse than NAC and DQfD. DQfD initially is better than our method but later is surpassed by the NAC method quickly, which might be caused by the supervised hinge loss being harmful when demonstrations are suboptimal. Similar to the policy generated demonstrations case, PCL, hard Q-learning and soft Q-learning do not perform well.

5.6. Effects of Imperfect Demonstrations

In the real world, collected demonstrations might be far from optimal. The human demonstrations above have already shown that imperfect demonstrations could have a large effect on performance. To study this phenomenon in a principled manner, we collect a few versions of demonstrations with varying degrees of noise. When collecting the demonstrations with the trained Q agent, we corrupt a certain percentage of the demonstrations by choosing non-optimal actions ($argmin_a Q(s, a)$). The data corruption process is conducted while interacting with the environment; therefore, the error will affect the collection of the follow-



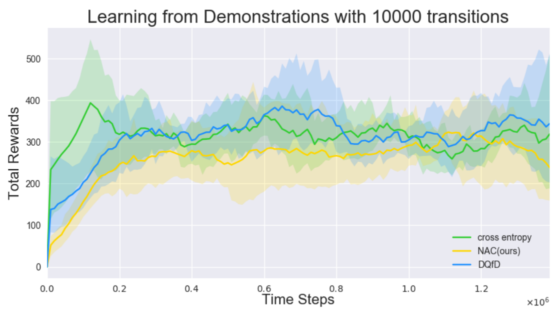


Figure 5. Left: Learning from imperfect data when the imperfectness is 30%. Our NAC method does not clone suboptimal behaviors and thus outperforms DQfD and behavior cloning. Right: Learning from a limit amount of demonstrations. Even with only 30 minutes (10k transitions) of experience, our method could still learn a policy that is comparable with supervised learning methods. More results are available in the appendix, including 50% and 80% imperfect data ablations, as well as 150k and 300k data amount studies.

ing steps. We get 3 sets of $\{30\%, 50\%, 80\%\}$ percentage of imperfect data. In the left of Fig. 5, we show that our method performs well compared with DQfD and behavior cloning methods. Supervised behavior cloning method is heavily influenced by the imperfect demonstrations. DQfD is also heavily affected, but not as severely as the behavior cloning. NAC is robust because it does not imitate the sub-optimal behaviors. The results for 50% and 80% percentage of imperfect data are similar, and they are available in the appendix.

5.7. Effects of Demonstration Size

In this section, we show comparisons among our method and other methods with different amounts of demonstration data. We use a trained agent to collect three sets of demonstrations which include 10k, 150k, and 300k transitions each. In the experiments, we find that our algorithm performs well when the amount of data is large and is comparable to supervised methods even with a limited amount of data. In Fig. 5 (right), we show when there are extremely limited amounts of demonstration data (10k transitions or 30 minutes of experience), our method performs on par with supervised methods. In the appendix, we show the results for 150k and 300k transitions: our method outperforms the baselines by a large margin with 300k transitions. In summary, our method can learn from small amounts of demonstration data and dominates in terms of performance when there is a sufficient amount of data.

5.8. Effects of Reward Choice

In the above experiments, we adopt a natural reward: it maximizes speed along the lane, minimizes speed perpendicular to the lane and penalizes when the agent hits anything. However, very informative rewards are not available under many conditions. In this section, we study whether our method is robust to a less informative reward. We change the reward

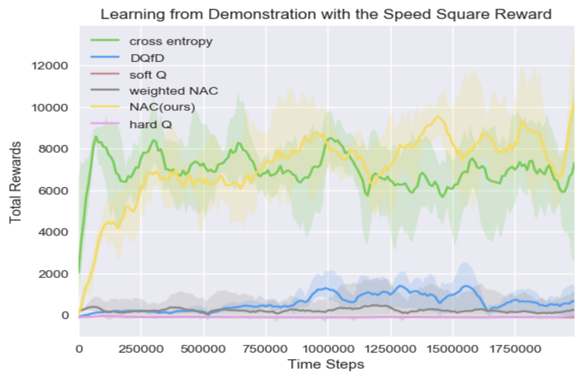


Figure 6. Similar to Figure 3 (left), we compare NAC with other methods when only learning from the demonstrations, except that we use a different reward: $speed^2$. Our method still performs the best.

function to be the square of the speed of an agent, irrespective of the speed's direction. This reward encourages the agent to drive fast, however, it is difficult to work with because the agent has to learn by itself that driving off-road or hitting obstacles reduce its future speed. It is also hard because $speed^2$ has a large numerical range. Figure 6 shows that NAC method still performs the best at convergence, while DQfD suffers from severe performance degeneration.

6. Conclusion

We proposed a Normalized Actor-Critic algorithm for reinforcement learning from demonstrations. Our algorithm provides a unified approach for learning from reward and demonstrations, and is robust to potentially suboptimal demonstration data. An agent can be fine-tuned with rewards after training on demonstrations by simply continuing to perform the same algorithm on on-policy data. Our algorithm preserves and improves the behaviors learned from demonstrations while receiving reward through interaction with an environment.

References

Abbeel, Pieter and Ng, Andrew Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1. ACM, 2004.

Bojarski, Mariusz, Del Testa, Davide, Dworakowski, Daniel, Firner, Bernhard, Flepp, Beat, Goyal, Prasoon, Jackel, Lawrence D, Monfort, Mathew, Muller, Urs, Zhang, Jiakai, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

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

Chemali, Jessica and Lazaric, Alessandro. Direct policy iteration with demonstrations. In *IJCAI*, pp. 3380–3386, 2015.

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

Ebrahimi, Sayna, Rohrbach, Anna, and Darrell, Trevor. Gradient-free policy architecture search and adaptation. In Levine, Sergey, Vanhoucke, Vincent, and Goldberg, Ken (eds.), *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pp. 505–514. PMLR, 13–15 Nov 2017. URL http://proceedings.mlr.press/v78/ebrahimi17a.html.

Gu, Shixiang, Lillicrap, Timothy, Ghahramani, Zoubin, Turner, Richard E, and Levine, Sergey. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*, 2016.

Gu, Shixiang, Lillicrap, Timothy, Ghahramani, Zoubin, Turner, Richard E, Schölkopf, Bernhard, and Levine, Sergey. Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. *arXiv preprint arXiv:1706.00387*, 2017.

Haarnoja, Tuomas, Tang, Haoran, Abbeel, Pieter, and Levine, Sergey. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017a.

Haarnoja, Tuomas, Tang, Haoran, Abbeel, Pieter, and Levine, Sergey. Reinforcement learning with deep energy-based policies. In Precup, Doina and Teh, Yee Whye (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1352–1361, International Convention Centre, Sydney, Australia, 06–11 Aug 2017b. PMLR. URL http://proceedings.mlr.press/v70/haarnoja17a.html.

Hester, Todd, Vecerik, Matej, Pietquin, Olivier, Lanctot, Marc, Schaul, Tom, Piot, Bilal, Sendonaris, Andrew, Dulac-Arnold, Gabriel, Osband, Ian, Agapiou, John, et al. Learning from demonstrations for real world reinforcement learning. *arXiv preprint arXiv:1704.03732*, 2017.

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

Jaderberg, Max, Mnih, Volodymyr, Czarnecki, Wojciech Marian, Schaul, Tom, Leibo, Joel Z, Silver, David, and Kavukcuoglu, Koray. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Kakade, Sham Machandranath et al. *On the sample complexity of reinforcement learning*. PhD thesis, University of London London, England, 2003.

Kim, Beomjoon, massoud Farahmand, Amir, Pineau, Joelle, and Precup, Doina. Learning from limited demonstrations. In *Advances in Neural Information Processing Systems*, pp. 2859–2867, 2013.

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

Munos, Rémi, Stepleton, Tom, Harutyunyan, Anna, and Bellemare, Marc. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1054–1062, 2016.

Nachum, Ofir, Norouzi, Mohammad, Xu, Kelvin, and Schuurmans, Dale. Bridging the gap between value and policy based reinforcement learning. *arXiv preprint arXiv:1702.08892*, 2017.

Ng, Andrew Y, Russell, Stuart J, et al. Algorithms for inverse reinforcement learning. In *Icml*, pp. 663–670, 2000.

Piot, Bilal, Geist, Matthieu, and Pietquin, Olivier. Boosted bellman residual minimization handling expert demonstrations. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 549–564. Springer, 2014.

Ross, Stéphane, Gordon, Geoffrey J, and Bagnell, Drew. 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.

Schulman, John, Abbeel, Pieter, and Chen, Xi. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017.

Shelhamer, Evan, Mahmoudieh, Parsa, Argus, Max, and Darrell, Trevor. Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307*, 2016.

Sutton, Richard S and Barto, Andrew G. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Thie, Paul R. *Markov decision processes*. Comap, Incorporated, 1983.

Todorov, Emanuel. General duality between optimal control and estimation. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pp. 4286–4292. IEEE, 2008.

Toussaint, Marc. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, pp. 1049–1056. ACM, 2009.

Wang, Ziyu, Bapst, Victor, Heess, Nicolas, Mnih, Volodymyr, Munos, Remi, Kavukcuoglu, Koray, and de Freitas, Nando. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

Wang, Ziyu, Merel, Josh, Reed, Scott, Wayne, Greg, de Freitas, Nando, and Heess, Nicolas. Robust imitation of diverse behaviors. *arXiv preprint arXiv:1707.02747*, 2017.

Watkins, Christopher JCH and Dayan, Peter. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Xu, Huazhe, Gao, Yang, Yu, Fisher, and Darrell, Trevor. End-to-end learning of driving models from large-scale video datasets. *arXiv preprint arXiv:1612.01079*, 2016.

Ziebart, Brian D. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

Ziebart, Brian D, Maas, Andrew L, Bagnell, J Andrew, and Dey, Anind K. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

7. Appendix

7.1. Normalized Actor-Critic with Q Parametrization

Usually the actor-critic method parametrizes $\pi(s, a)$ and $V(s)$ with a neural network that has two heads. In this section, we explore an alternative parametrization: Q-parametrization. Instead of outputting π and V directly, the neural network computes $Q(s, a)$. We parametrize π and V based on Q by specifying a fixed mathematical transform:

$$V_Q(s) = \alpha \log \sum_a \exp(Q(s,a)/\alpha) \tag{15}$$

$$\pi_Q(a|s) = \exp((Q(s,a) - V_Q(s))/\alpha) \tag{16}$$

Note that the Q-parametrization we propose here can be seen as a specific design of the network architecture. Instead of allowing the net to output arbitrary $\pi(s, a)$ and $V(s)$ values, we restrict the network to only output $\pi(s, a)$ and $V(s)$ pairs that satisfy the above relationship. This extra restriction will not harm the network's ability to learn since the above relationship has to be satisfied at the optimal solution(Schulman et al., 2017; Haarnoja et al., 2017b; Nachum et al., 2017).

Based on the Q-parametrization, we can derive the update of the actor. Note that we assume the behavioral policy is π_Q, and we sample one step out of a trajectory, thus dropping the subscript t. The goal is to maximize expected future reward, thus taking gradient of it we get:

$$\begin{aligned}
&\nabla \mathbb{E}_{s,a\sim\pi_Q}\left[R(s,a)\right] \\
=&\mathbb{E}_{s,a\sim\pi_Q}\left[R(s,a)\nabla \log_\theta p(a,s|\pi_Q)\right] \\
\approx&\mathbb{E}_{s,a\sim\pi_Q}\left[R(s,a)\nabla_\theta \log \pi_Q(a|s)\right]
\end{aligned} \tag{17}$$

where the last step ignores the state distribution, thus an approximation. By adding some baseline functions, it turns to the following format, where $\hat{Q}(s,a) = R(s,a) + \gamma V_Q(s')$:

$$\begin{aligned}
&\mathbb{E}_{s,a}\left[\nabla_\theta \log \pi_Q(a|s)(\hat{Q}(s,a) - b(s))\right] \\
=&\mathbb{E}_s\left[\sum_a \pi_Q(a|s)\nabla_\theta \log \pi_Q(a|s)(\hat{Q}(s,a) - b(s))\right]
\end{aligned} \tag{18}$$

As in previous work, an entropy-regularized policy gradient simply add the gradient of the entropy of the current policy with a tunable parameter α in order to encourage exploration. The entropy term is:

$$\begin{aligned}
&\mathbb{E}_s\left[\alpha\nabla_\theta H(\pi_Q(\cdot|s))\right] \\
=&\mathbb{E}_s\left[\alpha\nabla_\theta \sum_a -\pi_Q(a|s)\log \pi_Q(a|s)\right] \\
=&\mathbb{E}_s\left[\alpha \sum_a -\nabla_\theta \pi_Q(a|s)\log \pi_Q(a|s)\right. \\
&\qquad \left. -\pi_Q(a|s)\nabla_\theta \log \pi_Q(a|s)\right] \\
=&\mathbb{E}_s\left[\alpha \sum_a -\nabla_\theta \pi_Q(a|s)\log \pi_Q(a|s)\right. \\
&\qquad \left. -\pi_Q(a|s)\frac{1}{\pi_Q(a|s)}\nabla_\theta \pi_Q(a|s)\right] \\
=&\mathbb{E}_s\left[\alpha \sum_a -\nabla_\theta \pi_Q(a|s)\log \pi_Q(a|s)\right] \\
=&\mathbb{E}_s\left[\alpha \sum_a -\pi_Q(a|s)\nabla_\theta \log \pi_Q(a|s)\log \pi_Q(a|s)\right]
\end{aligned} \tag{19}$$

putting the two terms together and using the energy-based policy formulation (Eq. (16)) :

$$\begin{aligned}
&\mathbb{E}_{s,a}\left[\nabla_\theta \log \pi_Q(a|s)(\hat{Q}(s,a) - b(s)) + \alpha\nabla_\theta H(\pi_Q(\cdot|s))\right] \\
=&\mathbb{E}_s\left[\sum_a \pi_Q(s,a)\nabla_\theta \log \pi_Q(s,a)\right. \\
&\qquad \left.(\hat{Q}(s,a) - b(s) - (Q(s,a) - V_Q(s)))\right]
\end{aligned}$$

If we let the baseline $b(s)$ be $V_Q(s)$, we get the update:

$$\begin{aligned}
&\mathbb{E}_s\left[\sum_a \pi_Q(s,a)\nabla_\theta \log \pi_Q(s,a)(\hat{Q}(s,a) - Q(s,a))\right] \\
=&\frac{1}{\alpha}\mathbb{E}_{s,a}\left[(\nabla_\theta Q(s,a) - \nabla_\theta V_Q(s))(\hat{Q}(s,a) - Q(s,a))\right]
\end{aligned} \tag{20}$$

where $\hat{Q}(s,a)$ could be obtained through bootstrapping by $R(s,a) + \gamma V_Q(s')$. In practice $V_Q(s')$ is computed from a target network. For the critic, the update is:

$$\begin{aligned}
&\mathbb{E}_s\left[\nabla_\theta \frac{1}{2}(V_Q(s) - \hat{V}(s))^2\right] \\
=&\mathbb{E}_s\left[\nabla_\theta V_Q(s)(V_Q(s) - \hat{V}(s))\right]
\end{aligned} \tag{21}$$

where $\hat{V}(s)$ could be similarly obtained by bootstrapping: $\hat{V}(s) = \mathbb{E}_a\left[R(s,a) + \gamma V_Q(s')\right] + \alpha H(\pi_Q(\cdot|s))$.

7.2. Effects of Imperfect Demonstrations

See Figure 8 for more results for imperfect demonstrations when the amount of noise varies.

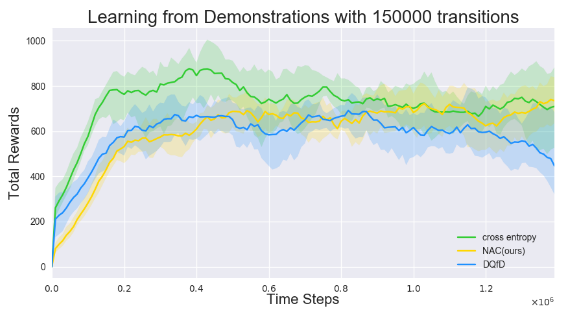
7.3. Effects of Demonstration Amount

See Figure 7 for more results on the effect of the demonstration amount.

7.4. Experiment Details

Network Architecture: We use the same architecture as in (Mnih et al., 2015) to parametrize $Q(s, a)$. With this Q parametrization, we also output $\pi_Q(a|s)$ and $V_Q(s)$ based on Eq. (16) and Eq. (15).

Hyper-parameters: We use a replay buffer with a capacity of 1 million steps and update the target network every 10K steps. Initially, the learning rate is linearly annealed from 1e-4 to 5e-5 for the first $1/10$ of the training process and then it is kept as a constant (5e-5). Gradients are clipped at 10 to reduce training variance. The reward discount factor γ is set to 0.99. We concatenate the 4 most recent frames as the input to the neural network. For the methods with an entropy regularizer, we set α to 0.1, following (Schulman et al., 2017). We truncate the importance sampling weighting factor $\beta = \min\left\{\frac{\pi_Q(a|s)}{\mu(a|s)}, c\right\}$ at 10, i.e., $c = 10$.



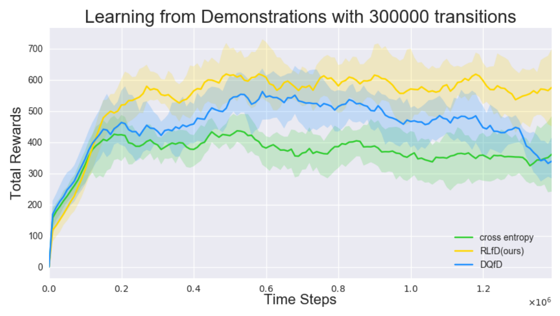
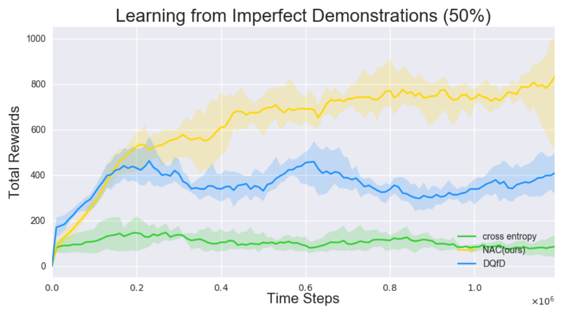


Figure 7. More results when varying the amount of demonstrations. The left and right figures show when there are 150k and 300k transitions respectively. Our NAC method achieves superior performance with a large amount of demonstrations and is comparable to supervise methods with smaller amount of demonstrations.



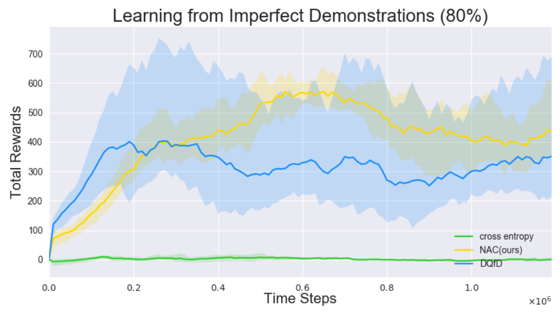


Figure 8. More results when introducing imperfect demonstrations. Left figure shows when there are 50% imperfect actions and the right one shows the case for 80%. Our NAC method is highly robust to noisy demonstrations.