Self-Imitation Learning from Demonstrations

Anonymous Author(s)
Affiliation
Address
email

Abstract

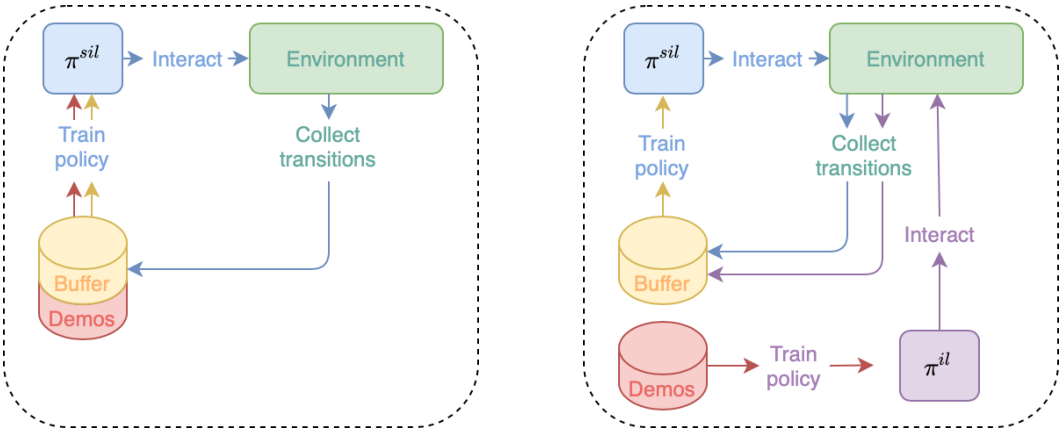
Despite the numerous breakthroughs achieved with Reinforcement Learning (RL), solving environments with sparse rewards remains a challenging task that requires sophisticated exploration. Learning from Demonstrations (LfD) remedies this issue by guiding agent's exploration towards states experienced by an expert. Naturally, the benefits of this approach hinge on the quality of demonstrations, which are rarely optimal in realistic scenarios. Modern LfD algorithms lack provable robustness to suboptimal demonstrations and introduce additional hyperparameters to control the influence of demonstrations. To address these issues, we extend Self-Imitation Learning (SIL), a recent RL algorithm that exploits agent's past good experience, to the LfD setup by initializing its replay buffer with demonstrations. We denote our algorithm as SIL from Demonstrations (SILfD). Our theoretical analysis highlights that SILfD is safe to apply to demonstrations of any degree of suboptimality and automatically adjusts the influence of demonstrations throughout the training. Our empirical investigation shows the superiority of SIL over existing LfD algorithms in settings of suboptimal demonstrations and sparse rewards.

1 Introduction

Deep Reinforcement Learning (RL) algorithms have recently achieved multiple breakthroughs in solving games [18, 20, 3, 14, 4], hard visuomotor [16] and manipulation [10] tasks, but some of these algorithms additionally rely on incorporating information from human demonstrations [34, 38]. Finding the optimal solution requires tremendous amount of environment interactions, which makes RL algorithms costly and dependent on sophisticated exploration techniques. This is especially true for environments with sparse rewards where encountering a positive reward requires a long and precise sequence of actions. An alternative approach is to additionally leverage a set of expert demonstrations, which has shown to help with exploration difficulties and provide magnitudes of improvement in learning speed and performance. This approach is known in the literature as Learning from Demonstrations (LfD) [1, 30].

LfD algorithms can be attributed to one of the three categories based on the technique to incorporate demonstrations. The first technique is to treat demonstrations as additional learning references by placing them in the experience replay buffer [11, 37, 9, 21, 25]. The second technique is to optimize a mixture of reinforcement and imitation objectives, either by mixing rewards [15, 40, 42, 5, 13] or by introducing an auxiliary imitation loss term [11, 28, 21]. The third technique is to initialize agent's parameters with supervised [34, 28, 32] or imitation [7] pretraining. Despite the impressive results on a variety of problems, modern algorithms typically assume access to high-quality demonstrations and, as our experiments confirm, degrade when demonstrations are suboptimal. Furthermore, these algorithms have to rely on additional hyperparameters to properly balance between learning from agent and expert experience, such as priority bonus of demonstrations in the replay buffer [11, 37, 25] and coefficients of auxiliary losses [11, 37, 21, 32] or rewards [28, 15, 40, 13].

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) SILfD – for demos with rewards

(b) SILfIL – for demos without rewards

Figure 1: Schematic architectures of the proposed algorithms. The agent is represented by the policy π^{sil} trained with Self-Imitation Learning. a) SILfD. The replay buffer with agent experience is concatenated with a set of expert demonstrations. The agent learns from both the replay buffer and the demonstrations. b) SILfIL. A secondary Imitation Learning policy π^{il} is trained to mimic expert based on a set of demonstrations. The agent learns from the replay buffer that contains experience of both policies π^{sil} and π^{il} interacting with the environment.

Self-Imitation Learning (SIL) [22] is a recent RL algorithm that imitates its past positive experience while ignoring negative experience. SIL has shown to fit particularly well in environments with sparse rewards where it can mimic complex behaviour required to reach the reward. Still, encountering the reward in the first place can be problematic, especially when using naive exploration. In this paper we argue that SIL can greatly benefit from expert demonstrations by alleviating the need to encounter positive experience and propose Self-Imitation Learning from Demonstrations (SILfD).

The idea behind SILfD is to initialize the experience replay buffer with demonstrations that are preserved in the buffer throughout the training. While this idea is not new and is also used in algorithms like DQfD [11] and DDPGfD [37], we argue that the ability of SIL to prioritize positive experience while ignoring negative experience makes it a natural choice to learn from demonstrations. On the one hand, prioritization mechanism ensures that demonstrations are not under-exploited when they are more useful than agent experience. Moreover, being able to focus on the most useful experience makes it safe to learn from suboptimal demonstrations with varying quality of behaviour. On the other hand, SILfD can forgo learning from demonstrations once they become obsolete. As a result, SILfD dynamically balances between learning from agent and expert experience based on its current usefulness. Our theoretical analysis supports these claims. Additionally, we propose SILfIL, an extension of SILfD to the case where rewards are not observed in demonstrations, which can be especially relevant when demonstrations are collected by a human expert.

To test the effectiveness of SILfD, we compare it with the existing LfD algorithms in several environments: a toy hard-exploration environment Chain [35]; four DeepMind Control Suite tasks [36] with continuous actions and sparsified rewards; and Pommerman environment [29] with procedural map generation. Experiments show that both SILfD and SILfIL outperform the existing LfD algorithms, especially when demonstrations are highly suboptimal.

2 Background

2.1 Reinforcement Learning

We consider the standard Markov Decision Process $\langle S, A, r, T, \gamma \rangle$, where

- S denotes the space of states s, A denotes the space of actions a,
- $r : S \times A \to \mathbb{R}$ denotes the reward function,
- $T : S \times A \to \Delta(S)$ denotes the transition function, where Δ denotes probability distribution,

- $\gamma \in (0, 1)$ denotes the discount factor.
- Further, $R_t = \sum_{n=t}^{\infty} \gamma^{n-t} r_n$ denotes return, where subscripts t and n denote time steps,
- $\pi_\theta : S \to \Delta(A)$ denotes policy parameterized by θ,
- $V_\phi(s) = \mathbb{E}_{\pi_\theta}[R_t | s_t = s]$, $Q_\phi(s, a) = \mathbb{E}_{\pi_\theta}[R_t | s_t = s, a_t = a]$, and $A_\phi(s, a) = Q_\phi(s, a) - V_\phi(s)$ denote value, Q-value, and advantage functions parameterized by ϕ.

Advantage-Actor-Critic (A2C) [19] is one of the most prevalent RL frameworks where Actor chooses actions in the environment by predicting policy in a given state while Critic evaluates the state to aid Actor's learning. Proximal Policy Optimization (PPO) [33] is based on A2C framework and focuses on staying within a trust region during updates of policy parameters.

2.2 Imitation Learning

The purpose of Imitation Learning (IL) is to train a policy that mimics expert behaviour, the samples from which are stored in a buffer of demonstrations $\mathcal{D} = (s, a)$. Behavioral Cloning (BC) is a classic IL algorithm that trains a policy to predict the demonstrated action for a given state in a supervised manner without environment interactions [27]. Generative Adversarial Imitation Learning (GAIL) is a modern algorithm that trains two adversarial models: discriminator and generator [12]. Discriminator is a binary classifier that distinguishes generated and expert transitions, whereas generator interprets discriminator's predictions as rewards and tries to confuse it.

2.3 Learning from Demonstrations

Unlike IL, Learning from Demonstrations (LfD) assumes both the reward signal r and a buffer of demonstrations $\mathcal{D} = (s, a, r)$ to be available. Typically, LfD algorithms use demonstrations to increase sample efficiency and aid exploration in environments with sparse rewards. Hester et al. [11] propose DQfD which extends DQN to the LfD setup by initializing the replay buffer with demonstrations, pretraining Q-network offline, mixing 1-step and n-step losses, and regularizing the network with an auxiliary supervised loss. As an analogue of DQfD for continuous control, Vecerik et al. [37] propose DDPGfD by applying similar modifications to the critic of DDPG. As an alternative approach, POfD [15] enforces occupancy measure matching between the agent and the expert by shaping the reward with the predictions of a GAIL-like discriminator. Similar to POfD ideas are employed in [40, 42]. Finally, supervised pretraining from demonstrations with BC is routinely used to assist solving complex tasks, e.g. Go [34] and Minecraft [32].

2.4 Self-Imitation Learning

SIL [22] aims to reproduce agent's past good decisions based on the experience stored in a replay buffer $\mathcal{B} = (s, a, r)$. The algorithm alternates between the standard on-policy update of A2C or PPO and the off-policy update that minimizes A2C loss with clipped advantages:

$$L^{\text{sil}}_{\text{policy}} = -\mathbb{E}_{\mathcal{B}}[\log \pi_\theta A^+_\phi(s, a)] - \alpha \mathcal{H}(\pi_\theta) \quad \text{(1a)} \qquad L^{\text{sil}}_{\text{value}} = \mathbb{E}_{\mathcal{B}}[A^+_\phi(s, a)]^2 \quad \text{(1b)}$$

where $(\cdot)^+ = max(\cdot, 0)$ ensures that only good transitions are considered for updates, advantage is estimated as $A(s, a) = R - V(s)$, $\mathcal{H}(\pi_\theta) = \mathbb{E}_{\pi_\theta}[-\log \pi_\theta(a|s)]$ denotes the entropy of the policy, and $\alpha \geq 0$.

Oh et al. [22] theoretically justify SIL by connecting it to lower-bound soft Q-learning that approximates the lower bound of the optimal soft Q-values Q^* by minimizing:

$$L^{\text{lb}} = \mathbb{E}_{\mathcal{B}}[(R^\mu - Q_\phi(s, a))^+]^2 \tag{2}$$

where $R^\mu_t = r_t + \sum_{n=t+1}^{\infty} \gamma^{n-t}(r_n + \alpha \mathcal{H}_n(\mu))$ is the entropy-regularized return. Specifically, the authors show that minimizing L^{lb} is equivalent to minimizing $L^{\text{sil}}_{\text{policy}}$ and $L^{\text{sil}}_{\text{value}}$ when $\alpha \to 0$. To improve training efficiency, SIL uses prioritized replay buffer [31].

3 Self-Imitation Learning from Demonstrations

The previous section described how SIL is able to exploit past good experience. However, encountering such experience can be problematic, especially in environments with sparse rewards. An alternative source of good experience can be a set of expert demonstrations $\mathcal{D}$. As an extension of SIL to LfD setting, we propose SIL from Demonstrations (SILfD) based on one simple modification of the original SIL: the experience replay buffer is initialized with demonstrations $\mathcal{D}$ that are preserved in the buffer throughout the training (Fig. 1a). The policy and value losses are accordingly modified:

$$L^{\text{sil}}_{\text{policy}} = -\mathbb{E}_{\mathcal{B}\cup\mathcal{D}}[\log \pi_\theta A^+_\phi(s,a)] - \alpha\mathcal{H}(\pi_\theta) \tag{3a}$$

$$L^{\text{sil}}_{\text{value}} = \mathbb{E}_{\mathcal{B}\cup\mathcal{D}}[A^+_\phi(s,a)]^2 \tag{3b}$$

3.1 Theoretical Analysis

This section provides insights into the effect of adding demonstrations to the experience replay buffer of SIL. Specifically:

- We establish the conditions under which demonstrations improve the speed of convergence to the optimal policy (Lemma 1)
- We show that completely useless demonstrations are ignored and thus are safe to include in the buffer (Lemma 2)
- We describe how demonstrations are sampled progressively less according to an automated schedule that naturally arises as agent improves (Lemma 3)

Since SIL can be viewed as a form of lower-bound soft Q-learning, we perform our analysis on the latter. Like the original SIL paper, we restrict the analysis to deterministic environments.

Preliminaries. We start by rewriting the lower-bound objective (2). We consider the LfD setup where agent and expert transitions are stored in the same replay buffer with λ being the proportion of expert transitions. In this setup, (2) can be rewritten as:

$$L^{\text{lb}}_d = (1-\lambda)\sum_{s,a}\rho_\pi(s,a)\mathbb{E}_{R\sim\pi}[\Delta^+_\pi]^2 + \lambda\sum_{s,a}\rho_{\pi_E}(s,a)\mathbb{E}_{R\sim\pi_E}[\Delta^+_{\pi_E}]^2 \tag{4}$$

where $\rho(s,a)$ is occupancy measure, $\Delta^+_\nu = \Delta^+_\nu(s,a,R) = (R^\nu - Q_\phi(s,a))^+$ will be referred as clipped advantage, ν denotes an arbitrary policy, and subscript d means that the loss is augmented with demonstrations.

We now rewrite objective (4) for the prioritized sampling employed in SIL. Each transition is sampled from the buffer with probability proportional to the clipped advantage:

$$L^{\text{lb}}_d = \frac{1-\lambda}{Z}\sum_{s,a}\rho_\pi(s,a)\mathbb{E}_{R\sim\pi}sg[\Delta^+_\pi][\Delta^+_\pi]^2 + \frac{\lambda}{Z}\sum_{s,a}\rho_{\pi_E}(s,a)\mathbb{E}_{R\sim\pi_E}sg[\Delta^+_{\pi_E}][\Delta^+_{\pi_E}]^2 \tag{5}$$

where $sg[\cdot]$ is a stopgradient operation that indicates that the gradient is not propagated through the sampling probability, $Z = (1-\lambda)Z_\pi + \lambda Z_{\pi_E}$ is the normalizing constant for the joint buffer, $Z_\nu = \sum_{s,a}\rho_\nu(s,a)\mathbb{E}_{R\sim\nu}sg[\Delta^+_\nu]$. We can further rewrite this by multiplying the first term by $\frac{Z_\pi}{Z_\pi}$, the second term by $\frac{Z_{\pi_E}}{Z_{\pi_E}}$:

$$L^{\text{lb}}_d = (1-\overline{p})\sum_{s,a}\mathbb{E}_{R\sim\pi}p_\pi(s,a,R)[\Delta^+_\pi]^2 + \overline{p}\sum_{s,a}\mathbb{E}_{R\sim\pi_E}p_{\pi_E}(s,a,R)[\Delta^+_{\pi_E}]^2 \tag{6}$$

where $(1-\overline{p}) = \frac{(1-\lambda)Z_\pi}{Z}$ and $\overline{p} = \frac{\lambda Z_{\pi_E}}{Z}$ are the respective marginal probabilities to sample an agent or an expert transition, $p_\nu(s,a,R) = \rho_\nu(s,a)sg[\Delta^+_\nu]/Z_\nu$ is the probability of prioritized sampling of a triple (s,a,R) from the policy ν.

When do demonstrations help SIL to converge faster? To answer this question, we first notice that because the lower bound Q_ϕ can only increase, the speed of its convergence to Q^* is directly tied into the magnitude of the gradient of the loss function $\nabla_\phi L^{\text{lb}}$. For this reason, we are interested in the conditions under which adding demonstrations to the buffer increases the gradient and thus improves convergence. We formulate such conditions for one step of update in Lemma 1.

Lemma 1.

$$\nabla_\phi L_d^{lb} > \nabla_\phi L^{lb} \quad \textit{if} \quad \frac{\mathbb{E}_{s,a,R\sim\pi_E}[\Delta^+_{\pi_E}]^2}{\mathbb{E}_{s,a,R\sim\pi_E}\Delta^+_{\pi_E}} > \frac{\mathbb{E}_{s,a,R\sim\pi}[\Delta^+_{\pi}]^2}{\mathbb{E}_{s,a,R\sim\pi}\Delta^+_{\pi}}$$

We leave the proof to the Appendix. Lemma 1 can be seen as a formalization of a simple statement that 'demonstrations help if they are more useful than the collected experience', giving precise definitions to 'help' and 'more useful'. In particular, the condition in Lemma 1 can be read as follows: expected clipped advantage, weighted by the sampling probability proportional to the clipped advantage, is higher for the expert than the agent. This condition might hold due to expert outperforming agent in states visited by both or due to expert visiting more rewarding states.

Completely useless demonstrations are ignored. Lemma 1 establishes the condition under which demonstrations help. However, we are still interested in the case of suboptimal demonstrations where this condition is not satisfied. A particularly interesting special case is the case of completely useless demonstrations, which we define below:

Definition 1. Let demonstrations be called completely useless if:

$$\forall\{s,a|\rho_{\pi_E}(s,a)>0\}, R\sim\pi_E : R^{\pi_E} \leq Q_\phi(s,a)$$

Since the definition depends on the current estimate of Q-value, the demonstrations can be completely useless at the very beginning of training (if a random agent outperforms expert) or become completely useless as a result of Q_ϕ becoming tighter during training. Completely useless demonstrations can potentially be harmful to LfD algorithms by guiding exploration towards regions of low returns. Fortunately, SILfD is robust to such demonstrations as they do not affect the loss function and thus are safe to include in the replay buffer. We formulate this property in the following lemma:

Lemma 2. *If demonstrations are completely useless,* $\nabla_\phi L_d^{lb} = \nabla_\phi L^{lb}$.

We leave the formal proof to the Appendix.

Automated scheduling. We now discuss how SILfD adheres to an automated schedule that arises from agent's improvement and results in a gradual decrease of the influence of demonstrations. Suppose that lower bound Q_ϕ got tighter by Q_{diff} as a result of an update. Suppose that the agent's policy has also improved from π to π_{new} and that the buffer got filled with experience from π_{new} instead of π. As a result, the expected clipped advantage of the collected experience has changed by $d(\pi,\pi_{\text{new}}) = \mathbb{E}_{s,a,R\sim\pi_{\text{new}}}(\Delta_{\pi_{\text{new}}} - Q_{\text{diff}})^+ - \mathbb{E}_{s,a,R\sim\pi}\Delta^+_\pi$. In the next lemma we connect $d(\pi,\pi_{\text{new}})$ to the change of the sampling probability of demonstrations.

Lemma 3. *Suppose that $Q_\phi(s,a)$ increases by $Q_{diff}(s,a)$ and the agent's policy changes from π to π_{new}. Then, $\overline{p}$ decreases if the following condition is satisfied:*

$$d(\pi,\pi_{new}) > \left(\frac{\mathbb{E}_{s,a,R\sim\pi_E}(\Delta_{\pi_E} - Q_{diff})^+}{\mathbb{E}_{s,a,R\sim\pi_E}\Delta^+_{\pi_E}} - 1\right)\mathbb{E}_{s,a,R\sim\pi}\Delta^+_\pi$$

The formal proof of Lemma 3 is reported in the Appendix. Notice that the right-hand side of the condition in Lemma 3 is negative since $Q_{\text{diff}} > 0$. Consequently, as long as the agent improves proportionally to the lower bound estimate (so that $d(\pi,\pi_{\text{new}}) \geq 0$), the condition in Lemma 3 is satisfied and the probability to sample from demonstrations $\overline{p}$ decreases. This property ensures that even if demonstrations hurt convergence at some point of training (i.e. Lemma 1 is not satisfied) as agent improves and starts collecting experience more useful than demonstrations, SILfD eventually learns to ignore such demonstrations.

Together, the three lemmas provided in this subsection characterize valuable properties of SILfD: it benefits from useful demonstrations, ignores completely useless demonstrations, and automatically schedules learning from demonstrations. In the next subsection, we compare SILfD with the most prominent existing LfD algorithms, none of which exhibits the same set of properties.

3.2 Comparison with existing algorithms

DQfD [11] and DDPGfD [37] utilize demonstrations by placing them into the replay buffer alongside with the agent experience. As Kang et al. [15] show, DQfD and DDPGfD employ guided exploration based on occupancy measure matching between the agent and the expert. This approach exhibits several issues. First, demonstrations can be under-exploited by being diluted with agent experience. In attempts to mitigate this problem, DQfD and DDPGfD specifically give a priority bonus to demonstrations, and DQfD additionally introduces a supervised loss term as a regularizer. SILfD does not rely on such heuristics and enjoys a provable improvement of convergence speed regardless of proportions λ (Lemma 1). Second, suboptimal demonstrations can be over-exploited. In case when useful and completely useless demonstrations are mixed, such demonstrations can potentially hurt convergence of DQfD and DDPGfD by guiding exploration towards regions of low returns, whereas SILfD ignores completely useless demonstrations (Lemma 2) and is able to only learn from useful experience. Our experiments in Chain environment also support this claim. Third, dynamic adjustment of the influence of demonstrations requires hand-tuned schedules of hyperparameters. Unless expert behaves optimally, agent's exploration becomes a better source of experience at some point of training. DQfD and DDPGfD might be unable to forgo learning from obsolete demonstrations if the priority bonus is set too high, whereas scheduling it requires tuning additional hyperparameters. As we show in Lemma 3, SILfD schedules learning from demonstrations automatically.

Like DQfD and DDPGfD, POfD [15] employs guided exploration based on occupancy measure matching and is therefore susceptible to the same issues. POfD is trained on a mixture of environmental and imitation rewards. To avoid under- or over-dependence on demonstrations, the mixture coefficient needs to be carefully tuned and potentially scheduled. Furthermore, the benefits of exploration via POfD are tied in to a crucial assumption that π_E is more advantageous than π in expectation. This assumption is not satisfied when useful demonstrations are mixed with a sufficient number of completely useless demonstrations, which SILfD is robust to.

3.3 Demonstrations without rewards

One drawback of SILfD is the assumption that rewards are observed in expert demonstrations. This limits applicability of SILfD in the most interesting cases where demonstrations are collected by a human expert, possibly optimizing a different reward or acting in a different environment [41, 8, 32, 26]. To mitigate this drawback, we modify SILfD when rewards are unavailable in demonstrations. Instead of directly filling the replay buffer with demonstrations, we train an IL algorithm like BC to mimic demonstrations and initialize the replay buffer with its experience (Fig. 1b). As an incidental benefit, IL algorithm may produce imperfect imitations that serve as more diverse learning references than the original demonstrations. We denote this modification as SILfIL. Another alternative we test is supervised pretraining.

4 Experimental Procedure

4.1 Environments

We have designed the experiments with the following desiderata in mind. First, SILfD should be tested in both discrete and continuous environments. To this end, we have chosen Pommerman and Chain as discrete environments and four DeepMind Control Suite tasks as continuous environments. Second, environments should have sparse rewards since in this setting demonstrations are the most helpful. While rewards in Chain and Pommerman are already sparse, we have additionally sparsified the rewards in DMC. Third, demonstrations should be imperfect to test both robustness of algorithms and their ability to surpass the expert. To this end, we vary the proportion of suboptimal demonstrations mixed with one optimal demonstration in Chain and use a suboptimal expert in DMC. While expert always wins in Pommerman, procedural map generation prevents agent from copying the expert.

Chain [35] is a simple but popular exploration benchmark [23, 24]. An illustration is provided in Appendix. The environment represents a square grid where the agent starts at the upper-left corner, its goal is to reach the bottom-right corner, and its actions are to move diagonally either to the lower-left cell or to the lower right cell. In our modification, the agent is penalized for moving right but receives a significant positive reward that exceeds any achievable sum of penalties by 100 when reaching the bottom-right corner. Additionally, if the agent is located on the left edge of the grid and steps left, it

simply moves one cell down and does not receive a penalty. Without specific exploration strategies or relying on demonstrations, standard RL algorithms cannot find the positive reward and converge to the policy that avoids penalties by always moving left. We record one optimal demonstration where the expert only moves right and dilute it with $n \in [0, 99]$ adversarial demonstrations where the expert only moves left. We fix the size of the map at 40x40.

Pommerman [29] is a challenging multi-agent environment with discrete control and high-dimensional observations. We adapt the single-agent regime proposed in [2] where the agent needs to defeat a single random opponent. Similarly to [2], we use the champion solution of FFA 2018 competition [39] to gather 300 demonstrations. However, we do not modify the original environment used in the competition, including its procedural generation. This amplifies the difficulty for the algorithms, as they have to imitate the expert in unseen states. Furthermore, the only reward signal the agent receives is +1 for defeating the opponent or -1 for being eliminated.

DeepMind Control Suite [36] is a set of popular benchmarks with continuous control. For our experiments, we select Cartpole Swing Up, which is a classic task where the agent needs to balance an unactuated pole by moving a cart, and three locomotive tasks: Cheetah Run, Walker Run and Hopper Hop, where the agent is supposed to control and to move forward a specific robot. In order to make exploration more challenging, we sparsify the rewards in all environments, the details of which are reported in the Appendix. We collect 25 suboptimal demonstrations with similar score in each environment. The details about the experts are also reported in the Appendix.

4.2 Algorithms

For technical details such as hyperparameters and how they were tuned, please refer to Appendix.

SILfD and SILfIL Our algorithms are implemented according to Section 3. In SILfD, the replay buffer is initialized with demonstrations. In SILfIL, the buffer is instead initialized with experience generated by a pretrained IL algorithm, in our case, BC.

SIL Since vanilla SIL [22] does not leverage demonstrations, comparing it with our algorithms highlights the benefits of demonstrations in hard-exploration environments.

BC This is a classic IL approach based on supervised learning that predicts a demonstrated actions in a given state [27]. Since BC ignores rewards, it is unlikely to outperform the expert.

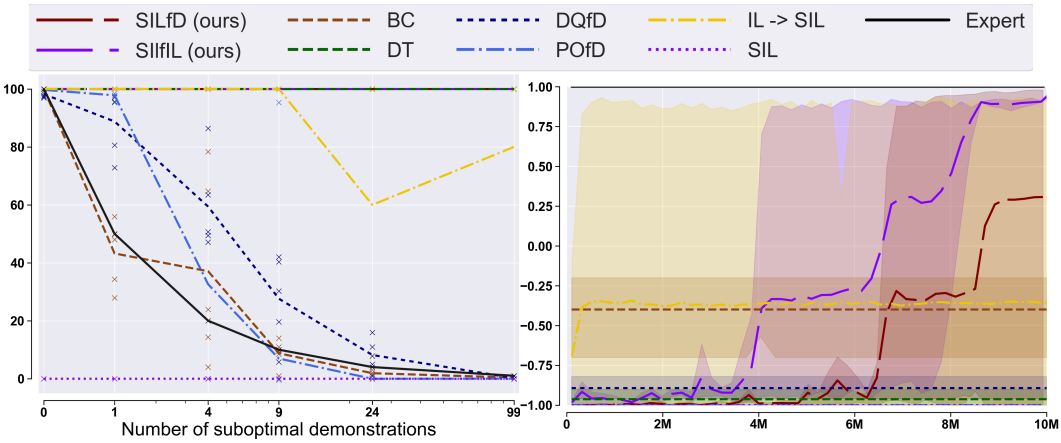
IL $\rightarrow$ SIL A standard sequential LfD approach is to first pretrain a policy with an IL algorithm and then fine-tune it with a RL algorithm [34, 28, 32]. We choose BC and SIL as IL and RL algorithms, respectively. This way, we can directly compare utilizing demonstrations in SIL through the buffer and through pretraining (note that both could be applied simultaneously). This baseline is very similar to SILfIL as both utilize pretrained BC to train SIL.

POfD A modern LfD approach employed in POfD [15] and several other studies [40, 42] is to optimize a mixture of environmental and imitation rewards: $r = r_{env} + \lambda_1 r_{im}, \lambda_1 \in [0, \infty]$. Similarly to GAIL, the imitation reward is obtained using predictions of a discriminator trained to distinguish agent and expert experience. As the base policy optimization algorithm, we use PPO.

DQfD and DDPGfD Similarly to SILfD, DQfD [11] and DDPGfD [37] store demonstrations in the replay buffer. Our implementations of DQN and DDPG are based on RLlib framework [17]. These algorithms are more sample efficient than other baselines and, according to our preliminary experiments, can overfit if trained for too long, so we train them less and show only the final performance rather than the learning curves on charts.

Decision Transformer Decision Transformer (DT) is a recent application of transformers to offline RL based on a framework that casts RL as conditional sequence modelling [6]. This model treats a projection of a past state-action pair and a desired return as a token. A casually masked sequence of such tokens representing past trajectory is passed through several attention layers and a linear decoder to predict an action that achieves a desired return in a given state. DT performs at least comparably with offline TD-based algorithms and BC and is able to extrapolate to returns beyond those provided during training. We use the authors' implementation of DT.[1] During evaluation, we set the desirable return as the maximal return achieved in demonstrations and the average final return achieved by SILfD, and report the higher score of the two.

[1] https://github.com/kzl/decision-transformer



(a) Results in Chain

(b) Results in Pommerman

Figure 2: Results in discrete environments. Y-axis measures average performance per 100 episodes. (a) In Chain, each experiment is repeated five times. The average final performance is illustrated with lines, while crosses denote individual experiments. X-axis measures number of suboptimal demonstrations mixed with one optimal demonstration. (b) In Pommerman, each experiment is repeated three times. X-axis measures millions of transitions and the lines represent learning curves.

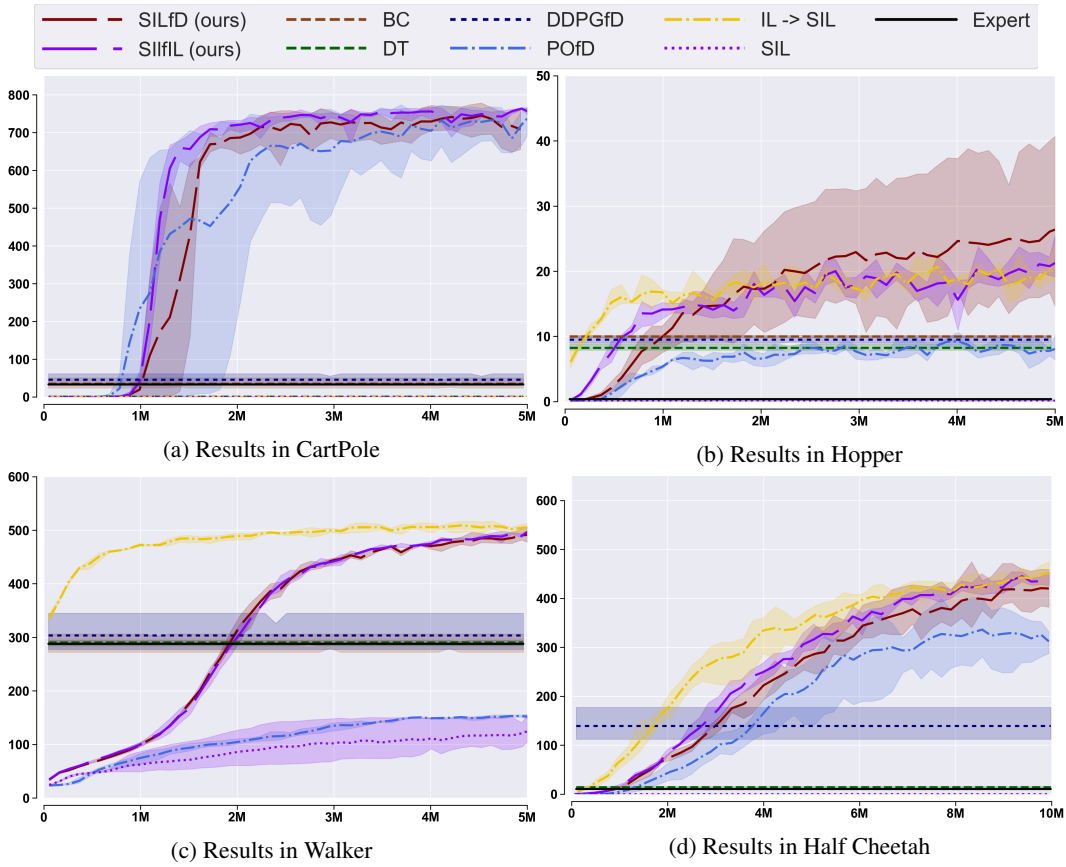
5 Experimental Results

Chain The results for the Chain environment are presented in Figure 2a. In these experiments, we put emphasis on the influence of suboptimal demonstrations and intentionally dilute one optimal expert trajectory with completely useless episodes. We find that only three of the tested algorithms can consistently solve all the tasks: both our algorithms and DT.[2] For SILfD and SILfIL, this result complements the theoretical robustness to completely useless demonstrations (Lemma 2). Furthermore, since in SILfIL the buffer is initialized with demonstrations collected by BC, the performance of BC is crucial. On the plot, we can see that the average return of BC is close to 0 on the hardest task, so most of the generated experience is useless, but even a rare successful episode is sufficient for SIL. Similarly, the sequential baseline can also successfully leverage pretrained BC to train SIL but is not as stable on harder tasks. By itself, SIL is unable to encounter the positive reward. Regarding DT, its flawless performance on Chain is expected since it is designed to be able to distinguish episodes with different returns. The performance of other algorithms gradually falls off on harder tasks. BC is expected to approximate expert performance, which it does. However, POfD and DQfD perform only slightly better than BC, despite optimizing (at least partly) for environmental reward. This supports our discussion in Section 3.2 about these algorithms having difficulties with controlling influence and discerning usefulness of demonstrations.

Pommerman The results for the Pommerman environment are presented in Figure 2b. Due to randomized map initialization, Pommerman presents a challenge for all algorithms, but only our algorithms reach the performance of the optimal expert. Furthermore, while SILfD learns a winning policy on only two seeds out of three, our SILfIL solves the task on all seeds and does it faster than SILfD. This might be due to having access to a more diverse experience as we provide SILfIL with more demonstrations, generated by BC. The sequential baseline, an alternative way to leverage demonstrations with SIL, performs significantly less consistently: while it manages to learn a winning policy almost instantly on one of the seeds, it completely fails on the other two seeds. BC performs relatively well and outperforms all other baselines, including DT. While both algorithms experience a mismatch between training and inference due to random maps, BC makes decisions based only on a given state, whereas DT processes the whole past trajectory up to a given state. This might make BC more robust to such mismatches. Finally, POfD and DQfD fail on the task almost completely.

DMC The results for the DMC environments are presented in Figure 3. In all DMC environments, our SILfD and SILfIL perform nearly indistinguishably and almost always outperform the baselines.

[2]Three horizontal lines at the 100 return might be hard to distinguish visually on the top of the plot 2a



(a) Results in CartPole

(b) Results in Hopper

(c) Results in Walker

(d) Results in Half Cheetah

Figure 3: Results in DMC. Each experiment is repeated 3 times.

The only difference between the two is that SILfD performs slightly better on average in Hopper. The sequential baseline behaves similarly in Hopper and Half Cheetah, which is not surprising given that it is also based on SIL. While it fails to provide SIL with useful experience in CartPole, it excels in sample efficiency in Walker. All algorithms that are not based on SIL perform worse. POfD, while performing well in CartPole, slightly falls off in Half Cheetah, significantly falls off in Hopper, and fails to reach the expert performance in Walker. DDPGfD generally performs even worse. While consistently outperforming the expert, DDPGfD never reaches the level of performance of SIL-based algorithms. Once again, the results point towards the inability of these algorithms to efficiently leverage suboptimal demonstrations. As anticipated, BC replicates the performance of the expert (with the exception of Hopper where it performs better). Unlike the experiments in Chain, DT fails to outperform the expert and performs similarly to BC. Apparently, DT is having difficulties with extrapolating to returns beyond the expert performance due to being a purely offline algorithm.

6 Conclusion

In this paper, we present SILfD, a novel algorithm that incorporates demonstrations into Self-Imitation Learning. We characterize the ability of SILfD to leverage useful demonstrations, ignore completely useless demonstrations, and dynamically adjust influence of demonstrations, providing both intuition and theoretical justification to superiority of SILfD. Our experiments complement the theoretical results by showing that SILfD can learn from suboptimal demonstrations better than existing algorithms and reliably surpass the expert. This conclusion holds for different kinds of suboptimality: when useful and useless demonstrations are mixed, when the expert completes the task successfully but not ideally, and when the expert acts in a limited subset of possible environment instances. Additionally, we propose SILfIL, a modification of SILfD that relaxes the requirement to observe rewards in demonstrations, and find that it performs as well as the original algorithm.

References

[1] C. G. Atkeson and S. Schaal. Robot learning from demonstration. In *ICML*, volume 97, pages 12–20. Citeseer, 1997.

[2] P. Barde, J. Roy, W. Jeon, J. Pineau, C. Pal, and D. Nowrouzezahrai. Adversarial soft advantage fitting: Imitation learning without policy optimization. *arXiv preprint arXiv:2006.13258*, 2020.

[3] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[4] N. Brown, A. Bakhtin, A. Lerer, and Q. Gong. Combining deep reinforcement learning and search for imperfect-information games. *arXiv preprint arXiv:2007.13544*, 2020.

[5] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé. Reinforcement learning from demonstration through shaping. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[6] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

[7] C.-A. Cheng, X. Yan, N. Wagener, and B. Boots. Fast policy learning through imitation and reinforcement. *arXiv preprint arXiv:1805.10413*, 2018.

[8] N. Chentanez, M. Müller, M. Macklin, V. Makoviychuk, and S. Jeschke. Physics-based motion capture imitation with deep reinforcement learning. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*, pages 1–10, 2018.

[9] Y. Gao, H. Xu, J. Lin, F. Yu, S. Levine, and T. Darrell. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

[10] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[11] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[12] J. Ho and S. Ermon. Generative adversarial imitation learning. *arXiv preprint arXiv:1606.03476*, 2016.

[13] L. Hussenot, R. Dadashi, M. Geist, and O. Pietquin. Show me the way: Intrinsic motivation from demonstrations. *arXiv preprint arXiv:2006.12917*, 2020.

[14] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

[15] B. Kang, Z. Jie, and J. Feng. Policy optimization with demonstrations. In *International Conference on Machine Learning*, pages 2469–2478. PMLR, 2018.

[16] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[17] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica. Rllib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pages 3053–3062. PMLR, 2018.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[19] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[20] M. Moravčík, M. Schmid, N. Burch, V. Lisỳ, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[21] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[22] J. Oh, Y. Guo, S. Singh, and H. Lee. Self-imitation learning. In *International Conference on Machine Learning*, pages 3878–3887. PMLR, 2018.

[23] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. *arXiv preprint arXiv:1602.04621*, 2016.

[24] I. Osband, J. Aslanides, and A. Cassirer. Randomized prior functions for deep reinforcement learning. *arXiv preprint arXiv:1806.03335*, 2018.

[25] T. L. Paine, C. Gulcehre, B. Shahriari, M. Denil, M. Hoffman, H. Soyer, R. Tanburn, S. Kapturowski, N. Rabinowitz, D. Williams, et al. Making efficient use of demonstrations to solve hard exploration problems. *arXiv preprint arXiv:1909.01387*, 2019.

[26] T. Pearce and J. Zhu. Counter-strike deathmatch with large-scale behavioural cloning. *arXiv preprint arXiv:2104.04258*, 2021.

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

[28] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[29] C. Resnick, W. Eldridge, D. Ha, D. Britz, J. Foerster, J. Togelius, K. Cho, and J. Bruna. Pommerman: A multi-agent playground. *arXiv preprint arXiv:1809.07124*, 2018.

[30] S. Schaal et al. Learning from demonstration. *Advances in neural information processing systems*, pages 1040–1046, 1997.

[31] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay, 2016.

[32] C. Scheller, Y. Schraner, and M. Vogel. Sample efficient reinforcement learning through learning from demonstrations in minecraft. In *NeurIPS 2019 Competition and Demonstration Track*, pages 67–76. PMLR, 2020.

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

[34] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[35] M. Strens. A bayesian framework for reinforcement learning. In *ICML*, volume 2000, pages 943–950, 2000.

[36] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[37] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[38] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[39] H. Zhou, Y. Gong, L. Mugrai, A. Khalifa, A. Nealen, and J. Togelius. A hybrid search agent in pommerman. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*, pages 1–4, 2018.

[40] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.

[41] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[42] K. Zolna, N. Rostamzadeh, Y. Bengio, S. Ahn, and P. O. Pinheiro. Reinforced imitation in heterogeneous action space. *arXiv preprint arXiv:1904.03438*, 2019.