

Self-Imitation Learning

Junhyuk Oh^{*1} Yijie Guo^{*1} Satinder Singh¹ Honglak Lee^{2,1}

Abstract

This paper proposes *Self-Imitation Learning* (SIL), a simple off-policy actor-critic algorithm that learns to reproduce the agent’s past *good* decisions. This algorithm is designed to verify our hypothesis that exploiting past good experiences can indirectly drive deep exploration. Our empirical results show that SIL significantly improves advantage actor-critic (A2C) on several hard exploration Atari games and is competitive to the state-of-the-art count-based exploration methods. We also show that SIL improves proximal policy optimization (PPO) on MuJoCo tasks.

1. Introduction

The trade-off between exploration and exploitation is one of the fundamental challenges in reinforcement learning (RL). The agent needs to *exploit* what it already knows in order to maximize reward. But, the agent also needs to *explore* new behaviors in order to find a potentially better policy. The resulting performance of an RL agent emerges from this interaction between exploration and exploitation.

This paper studies how exploiting the agent’s past experiences improves learning in RL. More specifically, we hypothesize that learning to reproduce past good experiences can indirectly lead to deeper exploration depending on the domain. A simple example of how this can occur can be seen through our results on an example Atari game, Montezuma’s Revenge (see Figure 1). In this domain, the first and more proximal source of reward is obtained by picking up the key. Obtaining the key is a precondition of the second and more distal source of reward (i.e., opening the door with the key). Many existing methods occasionally generate experiences that pick up the key and obtain the first reward, but fail to exploit these experiences often enough to learn how to open the door by exploring after picking up

^{*}Equal contribution ¹University of Michigan ²Google Brain. Correspondence to: Junhyuk Oh <junhyuk@umich.edu>, Yijie Guo <guoyijie@umich.edu>.

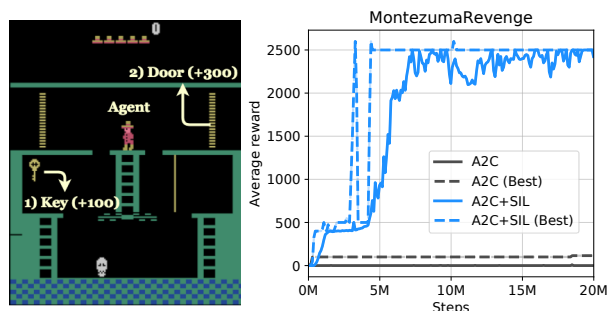


Figure 1. Learning curves on Montezuma’s Revenge. (Left) The agent needs to pick up the key in order to open the door. Picking up the key gives a small reward. (Right) The baseline (A2C) often picks up the key as shown by the best episode reward in 100K steps (A2C (Best)), but it fails to consistently reproduce such an experience. In contrast, self-imitation learning (A2C+SIL) quickly learns to pick up the key as soon as the agent experiences it, which leads to the next source of reward (door).

the key. Thus, they end up with a poor policy (see A2C in Figure 1). On the other hand, by exploiting the experiences that pick up the key, the agent is able to explore onwards from the state where it has the key to successfully learn how to open the door (see A2C+SIL in Figure 1). Of course, this sort of exploitation can also hurt performance in problems where there are proximal distractor rewards and repeated exploitation of such rewards does not help in learning about more distal and higher rewards; in other words, these two aspects may both be present. In this paper we will empirically investigate many different domains to see how exploiting past experiences can be beneficial for learning agents.

The main contributions of this paper are as follows: (1) To study how exploiting past good experiences affects learning, we propose a *Self-Imitation Learning* (SIL) algorithm which learns to imitate the agent’s own past good decisions. In brief, the SIL algorithm stores experiences in a replay buffer, learns to imitate state-action pairs in the replay buffer only when the return in the past episode is greater than the agent’s value estimate. (2) We provide a theoretical justification of the SIL objective by showing that the SIL objective is derived from the lower bound of the optimal Q-function. (3) The SIL algorithm is very simple to implement and can be applied to any actor-critic architecture. (4) We demonstrate that SIL combined with advantage actor-critic (A2C) is competitive to the state-of-the-art count-based exploration

actor-critic methods (e.g., Reactor-PixelCNN (Ostrovski et al., 2017)) on several hard exploration Atari games (Bellemare et al., 2013); SIL also improves the overall performance of A2C across 49 Atari games. Finally, SIL improves the performance of proximal policy optimization (PPO) on MuJoCo continuous control tasks (Brockman et al., 2016; Todorov et al., 2012), demonstrating that SIL may be generally applicable to any actor-critic architecture.

2. Related Work

Exploration There has been a long history of work on improving exploration in RL, including recent work that can scale up to large state spaces (Stadie et al., 2015; Osband et al., 2016; Bellemare et al., 2016; Ostrovski et al., 2017). Many existing methods use some notion of curiosity or uncertainty as a signal for exploration (Schmidhuber, 1991; Strehl & Littman, 2008). In contrast, this paper focuses on exploiting past good experiences for better exploration. Though the role of exploitation for exploration has been discussed (Thrun, 1992), prior work has mostly considered exploiting what the agent has learned, whereas we consider exploiting what the agent has experienced, but has not yet learned.

Episodic control Episodic control (Lengyel & Dayan, 2008) can be viewed as an extreme way of exploiting past experiences in the sense that the agent repeats the same actions that gave the best outcome in the past. MFEC (Blundell et al., 2016) and NEC (Pritzel et al., 2017) scaled up this idea to complex domains. However, these methods are slow during test-time because the agent needs to retrieve relevant states for each step and may generalize poorly as the resulting policy is non-parametric.

Experience replay Experience replay (Lin, 1992) is a natural way of exploiting past experiences for parametric policies. Prioritized experience replay (Moore & Atkeson, 1992; Schaul et al., 2016) proposed an efficient way of learning from past experiences by prioritizing them based on temporal-difference error. Our self-imitation learning also prioritizes experiences based on the full episode rewards. Optimality tightening (He et al., 2017) introduced an objective based on the lower/upper bound of the optimal Q-function, which is similar to a part of our theoretical result. These recent advances in experience replay have focused on value-based methods such as Q-learning, and are not easily applicable to actor-critic architectures.

Experience replay for actor-critic In fact, actor-critic framework (Sutton et al., 1999; Konda & Tsitsiklis, 2000) can also utilize experience replay. Many existing methods are based on off-policy policy evaluation (Precup et al., 2001; 2000), which involves importance sampling. For example, ACER (Wang et al., 2017) and Reactor (Gruslys et al., 2018) use Retrace (Munos et al., 2016) to evaluate

the learner from the behavior policy. Due to importance sampling, this approach may not benefit much from the past experience if the policy in the past is very different from the current policy. Although DPG (Silver et al., 2014; Lillicrap et al., 2016) performs experience replay without importance sampling, it is limited to continuous control. Our self-imitation learning objective does not involve importance sampling and is applicable to both discrete and continuous control.

Connection between policy gradient and Q-learning

The recent studies on the relationship between policy gradient and Q-learning have shown that entropy-regularized policy gradient and Q-learning are closely related or even equivalent depending on assumptions (Nachum et al., 2017; O’Donoghue et al., 2017; Schulman et al., 2017a; Haarnoja et al., 2017). Our application of self-imitation learning to actor-critic (A2C+SIL) can be viewed as an instance of PGQL (O’Donoghue et al., 2017) in that we perform Q-learning on top of actor-critic architecture (see Section 4). Unlike Q-learning in PGQL, however, we use the proposed lower bound Q-learning to exploit good experiences.

Learning from imperfect demonstrations A few studies have attempted to learn from imperfect demonstrations, such as DQfD (Hester et al., 2018), Q-filter (Nair et al., 2017), and normalized actor-critic (Xu et al., 2018). Our self-imitation learning has a similar flavor in that the agent learns from imperfect demonstrations. However, we treat the agent’s own experiences as demonstrations without using expert demonstrations. Although a similar idea has been discussed for program synthesis (Liang et al., 2016; Abo-lafia et al., 2018), this prior work used classification loss without justification. On the other hand, we propose a new objective, provide a theoretical justification, and systematically investigate how it drives exploration in RL.

3. Self-Imitation Learning

The goal of self-imitation learning (SIL) is to imitate the agent’s past good experiences in the actor-critic framework. To this end, we propose to store past episodes with cumulative rewards in a replay buffer: $\mathcal{D} = \{(s_t, a_t, R_t)\}$, where s_t, a_t are a state and an action at time-step t , and $R_t = \sum_{k=t}^{\infty} \gamma^{k-t} r_k$ is the discounted sum of rewards with a discount factor γ . To exploit only good state-action pairs in the replay buffer, we propose the following off-policy actor-critic loss:

$$\mathcal{L}^{sil} = \mathbb{E}_{s,a,R \in \mathcal{D}} [\mathcal{L}_{policy}^{sil} + \beta^{sil} \mathcal{L}_{value}^{sil}] \quad (1)$$

$$\mathcal{L}_{policy}^{sil} = -\log \pi_{\theta}(a|s) (R - V_{\theta}(s))_+ \quad (2)$$

$$\mathcal{L}_{value}^{sil} = \frac{1}{2} \|(R - V_{\theta}(s))_+\|^2 \quad (3)$$

where $(\cdot)_+ = \max(\cdot, 0)$, and $\pi_{\theta}, V_{\theta}(s)$ are the policy (i.e., actor) and the value function parameterized by θ . $\beta^{sil} \in \mathbb{R}^+$

Algorithm 1 Actor-Critic with Self-Imitation Learning

```

Initialize parameter  $\theta$ 
Initialize replay buffer  $\mathcal{D} \leftarrow \emptyset$ 
Initialize episode buffer  $\mathcal{E} \leftarrow \emptyset$ 
for each iteration do
    # Collect on-policy samples
    for each step do
        Execute an action  $s_t, a_t, r_t, s_{t+1} \sim \pi_\theta(a_t|s_t)$ 
        Store transition  $\mathcal{E} \leftarrow \mathcal{E} \cup \{(s_t, a_t, r_t)\}$ 
    end for
    if  $s_{t+1}$  is terminal then
        # Update replay buffer
        Compute returns  $R_t = \sum_k \gamma^{k-t} r_k$  for all  $t$  in  $\mathcal{E}$ 
         $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, R_t)\}$  for all  $t$  in  $\mathcal{E}$ 
        Clear episode buffer  $\mathcal{E} \leftarrow \emptyset$ 
    end if
    # Perform actor-critic using on-policy samples
     $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}^{a2c}$  (Eq. 4)
    # Perform self-imitation learning
    for  $m = 1$  to  $M$  do
        Sample a mini-batch  $\{(s, a, R)\}$  from  $\mathcal{D}$ 
         $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}^{sil}$  (Eq. 1)
    end for
end for
    
```

is a hyperparameter for the value loss.

Note that $\mathcal{L}_{policy}^{sil}$ can be viewed as policy gradient using the value $V_\theta(s)$ as the state-dependent baseline except that we use the off-policy Monte-Carlo return (R) instead of on-policy return. $\mathcal{L}_{policy}^{sil}$ can also be interpreted as cross entropy loss (i.e., classification loss for discrete action) with sample weights proportional to the gap between the return and the agent’s value estimate ($R - V_\theta$). If the return in the past is greater than the agent’s value estimate ($R > V_\theta$), the agent learns to choose the action chosen in the past in the given state. Otherwise ($R \leq V_\theta$), and such a state-action pair is not used to update the parameter due to the $(\cdot)_+$ operator. This encourages the agent to imitate its own decisions in the past only when such decisions resulted in larger returns than expected. $\mathcal{L}_{value}^{sil}$ updates the value estimate towards the off-policy return R .

Prioritized Replay The proposed self-imitation learning objective \mathcal{L}^{sil} is based on our theoretical result discussed in Section 4. In theory, the replay buffer (\mathcal{D}) can be any trajectories from any policies. However, only *good* state-action pairs that satisfy $R > V_\theta$ can contribute to the gradient during self-imitation learning (Eq. 1). Therefore, in order to get many state-action pairs that satisfy $R > V_\theta$, we propose to use the prioritized experience replay (Schaul et al., 2016). More specifically, we sample transitions from the replay buffer using the clipped advantage $(R - V_\theta(s))_+$ as priority

(i.e., sampling probability is proportional to $(R - V_\theta(s))_+$). This naturally increases the proportion of *valid* samples that satisfy the constraint $(R - V_\theta(s))_+$ in SIL objective and thus contribute to the gradient.

Advantage Actor-Critic with SIL (A2C+SIL) Our self-imitation learning can be combined with any actor-critic method. In this paper, we focus on the combination of advantage actor-critic (A2C) (Mnih et al., 2016) and self-imitation learning (A2C+SIL), as described in Algorithm 1. The objective of A2C (\mathcal{L}^{a2c}) is given by (Mnih et al., 2016):

$$\mathcal{L}^{a2c} = \mathbb{E}_{s, a \sim \pi_\theta} [\mathcal{L}_{policy}^{a2c} + \beta^{a2c} \mathcal{L}_{value}^{a2c}] \quad (4)$$

$$\mathcal{L}_{policy}^{a2c} = -\log \pi_\theta(a_t|s_t)(V_t^n - V_\theta(s_t)) - \alpha \mathcal{H}_t^{\pi_\theta} \quad (5)$$

$$\mathcal{L}_{value}^{a2c} = \frac{1}{2} \|V_\theta(s_t) - V_t^n\|^2 \quad (6)$$

where $\mathcal{H}_t^\pi = -\sum_a \pi(a|s_t) \log \pi(a|s_t)$ denotes the entropy in simplified notation, and α is a weight for entropy regularization. $V_t^n = \sum_{d=0}^{n-1} \gamma^d r_{t+d} + \gamma^n V_\theta(s_{t+n})$ is the n -step bootstrapped value.

To sum up, A2C+SIL performs both on-policy A2C update (\mathcal{L}^{a2c}) and self-imitation learning from the replay buffer M times (\mathcal{L}^{sil}) to exploit past good experiences. A2C+SIL is relatively simple to implement as it does not involve importance sampling.

4. Theoretical Justification

In this section, we justify the following claim.

Claim. *The self-imitation learning objective (\mathcal{L}^{sil} in Eq. 1) can be viewed as an implementation of lower-bound-soft-Q-learning (Section 4.2) under the entropy-regularized RL framework.*

To show the above claim, we first introduce the entropy-regularized RL (Haarnoja et al., 2017) in Section 4.1. Section 4.2 introduces *lower-bound-soft-Q-learning*, an off-policy Q-learning algorithm, which learns the optimal action-value function from good state-action pairs. Section 4.3 proves the above claim by showing the equivalence between self-imitation learning and lower-bound-soft-Q-learning. Section 4.4 further discusses the relationship between A2C and self-imitation learning.

4.1. Entropy-Regularized Reinforcement Learning

The goal of entropy-regularized RL is to learn a stochastic policy which maximizes the entropy of the policy as well as the γ -discounted sum of rewards (Haarnoja et al., 2017; Ziebart et al., 2008):

$$\pi^* = \operatorname{argmax}_\pi \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t (r_t + \alpha \mathcal{H}_t^\pi) \right] \quad (7)$$

where $\mathcal{H}_t^\pi = -\log \pi(a_t|s_t)$ is the entropy of the policy π , and $\alpha \geq 0$ represents the weight of entropy bonus. Intu-

itively, in the entropy-regularized RL, a policy that has a high entropy is preferred (i.e., diverse actions chosen given the same state).

The optimal *soft* Q-function and the optimal *soft* value function are defined as:

$$Q^*(s_t, a_t) = \mathbb{E}_{\pi^*} \left[r_t + \sum_{k=t+1}^{\infty} \gamma^{k-t} (r_k + \alpha \mathcal{H}_k^{\pi^*}) \right] \quad (8)$$

$$V^*(s_t) = \alpha \log \sum_a \exp(Q^*(s_t, a)/\alpha). \quad (9)$$

It is shown that the optimal policy π^* has the following form (see Ziebart (2010); Haarnoja et al. (2017) for the proof):

$$\pi^*(a|s) = \exp((Q^*(s, a) - V^*(s))/\alpha). \quad (10)$$

This result provides the relationship among the optimal Q-function, the optimal policy, and the optimal value function, which will be useful in Section 4.3.

4.2. Lower Bound Soft Q-Learning

Lower bound of optimal soft Q-value Let π^* be an optimal policy in entropy-regularized RL (Eq. 7). It is straightforward that the expected return of any behavior policy μ can serve as a lower bound of the optimal soft Q-value as follows:

$$Q^*(s_t, a_t) = \mathbb{E}_{\pi^*} \left[r_t + \sum_{k=t+1}^{\infty} \gamma^{k-t} (r_k + \alpha \mathcal{H}_k^{\pi^*}) \right] \quad (11)$$

$$\geq \mathbb{E}_{\mu} \left[r_t + \sum_{k=t+1}^{\infty} \gamma^{k-t} (r_k + \alpha \mathcal{H}_k^{\mu}) \right], \quad (12)$$

because the entropy-regularized return of the optimal policy is always greater or equal to that of any other policies.

Lower bound soft Q-learning Suppose that we have full episode trajectories from a behavior policy μ , which consists of state-action-return triples: (s_t, a_t, R_t) where $R_t = r_t + \sum_{k=t+1}^{\infty} \gamma^{k-t} (r_k + \alpha \mathcal{H}_k^{\mu})$ is the entropy-regularized return. We propose *lower bound soft Q-learning* which updates $Q_{\theta}(s, a)$ parameterized by θ towards the optimal soft Q-value as follows (t is omitted for brevity):

$$\mathcal{L}^{lb} = \mathbb{E}_{s,a,R \sim \mu} \left[\frac{1}{2} \|(R - Q_{\theta}(s, a))_+\|^2 \right], \quad (13)$$

where $(\cdot)_+ = \max(\cdot, 0)$. Intuitively, we update the Q-value only when the return is greater than the Q-value estimate ($R > Q_{\theta}(s, a)$). This is justified by the fact that the lower bound (Eq. 12) implies that the estimated Q-value is lower than the optimal soft Q-value: $Q^*(s, a) \geq R > Q_{\theta}(s, a)$ when the environment is deterministic. Otherwise ($R \leq Q_{\theta}(s, a)$), such state-action pairs do not provide any useful

information about the optimal soft Q-value, so they are not used for training. We call this lower-bound-soft-Q-learning as it updates Q-values towards the lower bounds of the optimal Q-values observed from the behavior policy.

4.3. Connection between SIL and Lower Bound Soft Q-Learning

In this section, we derive an equivalent form of lower-bound-soft-Q-learning (Eq. 13) for the actor-critic architecture and show a connection to self-imitation learning objective.

Suppose that we have a parameterized soft Q-function Q_{θ} . According to the form of optimal soft value function and optimal policy in the entropy-regularized RL (Eq. 9-10), it is natural to consider the following forms of a value function V_{θ} and a policy π_{θ} :

$$V_{\theta}(s) = \alpha \log \sum_a \exp(Q_{\theta}(s, a)/\alpha) \quad (14)$$

$$\pi_{\theta}(a|s) = \exp((Q_{\theta}(s, a) - V_{\theta}(s))/\alpha). \quad (15)$$

From these definitions, Q_{θ} can be written as:

$$Q_{\theta}(s, a) = V_{\theta}(s) + \alpha \log \pi_{\theta}(a|s). \quad (16)$$

For convenience, let us define the following:

$$\hat{R} = R - \alpha \log \pi_{\theta}(a|s) \quad (17)$$

$$\Delta = R - Q_{\theta}(s, a) = \hat{R} - V_{\theta}(s). \quad (18)$$

By plugging Eq. 16 into Eq. 13, we can derive the gradient estimator of lower-bound-soft-Q-learning for the actor-critic architecture as follows:

$$\nabla_{\theta} \mathbb{E}_{s,a,R \sim \mu} \left[\frac{1}{2} \|(R - Q_{\theta}(s, a))_+\|^2 \right] \quad (19)$$

$$= \mathbb{E} [-\nabla_{\theta} Q_{\theta}(s, a) \Delta_+] \quad (20)$$

$$= \mathbb{E} [-\nabla_{\theta} (\alpha \log \pi_{\theta}(a|s) + V_{\theta}(s)) \Delta_+] \quad (21)$$

$$= \mathbb{E} [-\alpha \nabla_{\theta} \log \pi_{\theta}(a|s) \Delta_+ - \nabla_{\theta} V_{\theta}(s) \Delta_+] \quad (22)$$

$$= \mathbb{E} [\alpha \nabla_{\theta} \mathcal{L}_{policy}^{lb} - \nabla_{\theta} V_{\theta}(s) \Delta_+] \quad (23)$$

$$= \mathbb{E} [\alpha \nabla_{\theta} \mathcal{L}_{policy}^{lb} - \nabla_{\theta} V_{\theta}(s) (R - Q_{\theta}(s, a))_+] \quad (24)$$

$$= \mathbb{E} [\alpha \nabla_{\theta} \mathcal{L}_{policy}^{lb} - \nabla_{\theta} V_{\theta}(s) (\hat{R} - V_{\theta}(s))_+] \quad (25)$$

$$= \mathbb{E} \left[\alpha \nabla_{\theta} \mathcal{L}_{policy}^{lb} + \nabla_{\theta} \frac{1}{2} \left\| (\hat{R} - V_{\theta}(s))_+ \right\|^2 \right] \quad (26)$$

$$= \mathbb{E} [\alpha \nabla_{\theta} \mathcal{L}_{policy}^{lb} + \nabla_{\theta} \mathcal{L}_{value}^{lb}]. \quad (27)$$

Each loss term in Eq. 27 is given by:

$$\mathcal{L}_{policy}^{lb} = -\log \pi_{\theta}(a|s) \left(\hat{R} - V_{\theta}(s) \right)_+ \quad (28)$$

$$\mathcal{L}_{value}^{lb} = \frac{1}{2} \left\| (\hat{R} - V_{\theta}(s))_+ \right\|^2. \quad (29)$$

Thus, $\mathcal{L}_{policy}^{lb} = \mathcal{L}_{policy}^{sil}$ and $\mathcal{L}_{value}^{lb} = \mathcal{L}_{value}^{sil}$ as $\alpha \rightarrow 0$ (see Eq. 2-3). This shows that the proposed self-imitation

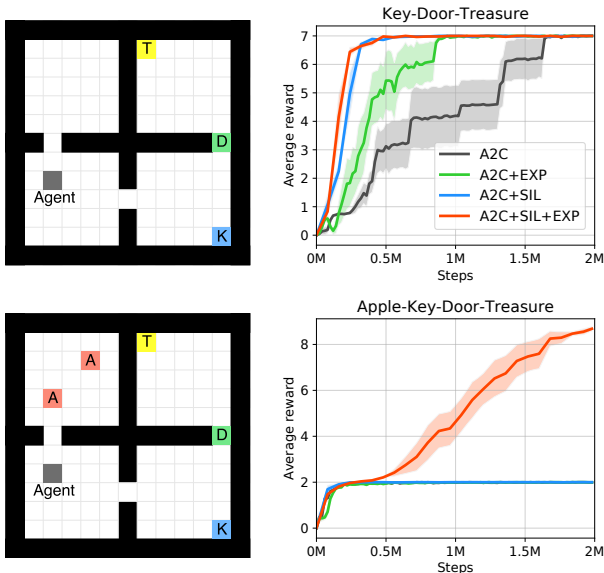


Figure 2. Key-Door-Treasure domain. The agent should pick up the key (K) in order to open the door (D) and collect the treasure (T) to maximize the reward. In the Apple-Key-Door-Treasure domain (bottom), there are two apples (A) that give small rewards (+1). ‘SIL’ and ‘EXP’ represent our self-imitation learning and a count-based exploration method respectively.

learning objective \mathcal{L}^{sil} (Eq. 1) can be viewed as a form of lower-bound-soft-Q-learning (Eq. 13), but without explicitly optimizing for entropy bonus reward as $\alpha \rightarrow 0$. Since the lower-bound-soft-Q-learning directly updates the Q-value towards the lower bound of the optimal Q-value, self-imitation learning can be viewed as an algorithm that updates the policy (π_θ) and the value (V_θ) directly towards the optimal policy and the optimal value respectively.

4.4. Relationship between A2C and SIL

Intuitively, A2C updates the policy in the direction of increasing the expected return of the learner policy and enforces consistency between the value and the policy from on-policy trajectories. On the other hand, SIL updates each of them directly towards optimal policies and values respectively from off-policy trajectories. In fact, Nachum et al. (2017); Haarnoja et al. (2017); Schulman et al. (2017a) have recently shown that entropy-regularized A2C can be viewed as n -step online soft Q-learning (or path consistency learning). Therefore, both A2C and SIL objectives are designed to learn the optimal soft Q-function in the entropy-regularized RL framework. Thus, we claim that both objectives can be complementary to each other in that they share the same optimal solution as discussed in PGQL (O’Donoghue et al., 2017).

5. Experiment

The experiments are designed to answer the following:

- Is self-imitation learning useful for exploration?
- Is self-imitation learning complementary to count-based exploration methods?
- Does self-imitation learning improve the overall performance across a variety of tasks?
- When does self-imitation learning help and when does it not?
- Can other off-policy actor-critic methods also exploit good experiences (e.g., ACER (Wang et al., 2017))?
- Is self-imitation learning useful for continuous control and compatible with other learning algorithms like PPO (Schulman et al., 2017b)?

5.1. Implementation Details

For Atari experiments, we used a 3-layer convolutional neural network used in DQN (Mnih et al., 2015) with last 4 stacked frames as input. We performed 4 self-imitation learning updates per on-policy actor-critic update ($M = 4$ in Algorithm 1). Instead of treating losing a life as episode termination as typically done in the previous work, we terminated episodes when the game ends, as it is the true definition of episode termination. For MuJoCo experiments, we used an MLP which consists of 2 hidden layers with 64 units as in Schulman et al. (2017b). We performed 10 self-imitation learning updates per each iteration (batch). More details of the network architectures and hyperparameters are described in the Appendix. Our implementation is based on OpenAI’s baseline implementation (Dhariwal et al., 2017).¹

5.2. Key-Door-Treasure Domain

To investigate how self-imitation learning is useful for exploration and whether it is complementary to count-based exploration method, we compared different methods on a grid-world domain, as illustrated in Figure 2. More specifically, we implemented a count-based exploration method (Strehl & Littman, 2008) that gives an exploration bonus reward: $r_{exp} = \beta / \sqrt{N(s)}$, where $N(s)$ is the visit count of state s and β is a hyperparameter. We also implemented a combination with self-imitation learning shown as ‘A2C+SIL+EXP’ in Figure 2.

In the first domain (Key-Door-Treasure), the chance of picking up the key followed by opening the door and obtaining the treasure is low due to the sequential dependency between them. We found that the baseline A2C tends to get stuck at a sub-optimal policy that only opens the door for a long time. A2C+EXP learns faster than A2C because exploration bonus encourages the agent to collect the treasure more often. Interestingly, A2C+SIL and A2C+SIL+EXP learn most quickly. We observed that once the agent opens

¹The code is available on <https://github.com/junhyukoh/self-imitation-learning>.

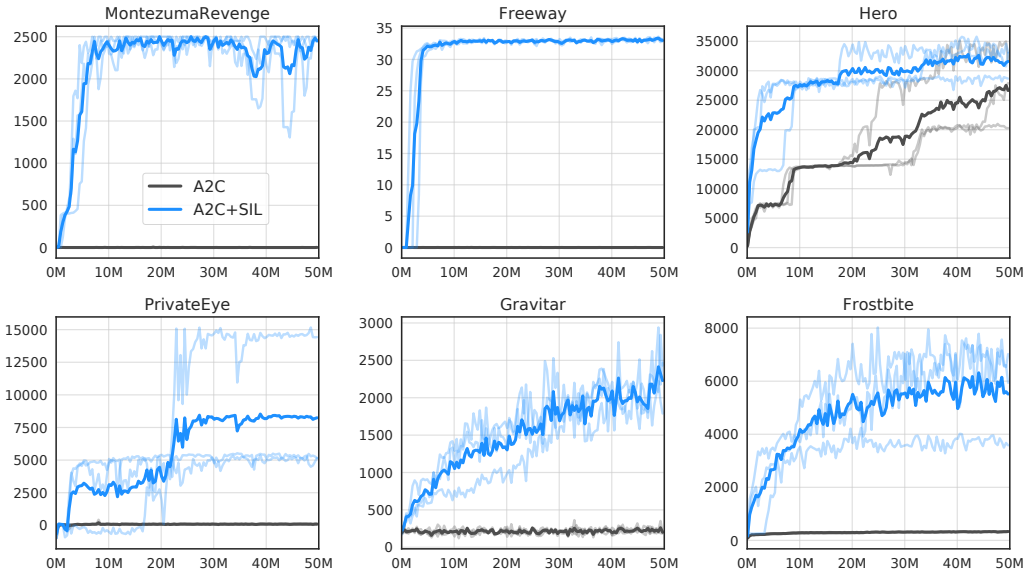


Figure 3. Learning curves on hard exploration Atari games. X-axis and y-axis represent steps and average reward respectively.

the door with the key by chance, our SIL helps exploit such good experiences and quickly learns to open the door with the key. This increases the chance of getting the next reward (i.e., treasure) and helps learn the optimal policy. This is an example showing that self-imitation learning can drive deep exploration.

In the second domain (Apple-Key-Door-Treasure), collecting apples near the agent’s initial location makes it even more challenging for the agent to learn the optimal policy, which collects all of the objects within the time limit (50 steps). In this domain, many agents learned a sub-optimal policy that only collects two apples as shown in Figure 2. On the other hand, only A2C+SIL+EXP consistently learned the optimal policy because count-based exploration increases the chance of collecting the treasure, while self-imitation learning can quickly exploit such a good experience as soon as the agent collects it. This result shows that self-imitation learning and count-based exploration methods can be complementary to each other. This result also suggests that while exploration is important for increasing the chance/frequency of getting a reward, it is also important to exploit such rare experiences to learn a policy to consistently achieve it especially when the reward is sparse.

5.3. Hard Exploration Atari Games

We investigated how useful our self-imitation learning is for several hard exploration Atari games on which recent advanced exploration methods mainly focused. Figure 3 shows that A2C with our self-imitation learning (A2C+SIL) outperforms A2C on six hard exploration games. A2C failed to learn a better-than-random policy, except for Hero, whereas our method learned better policies and achieved human-level performances on Hero and Freeway. We observed that

Table 1. Comparison to count-based exploration actor-critic agents on hard exploration Atari games. A3C+ and Reactor+ correspond to A3C-CTS (Bellemare et al., 2016) and Reactor-PixelCNN respectively (Ostrovski et al., 2017). SimHash represents TRPO-AE-SimHash (Tang et al., 2017). †Numbers are taken from plots.

	A2C+SIL	A3C+	REACTOR+ [†]	SIMHASH
MONTEZUMA	2500	273	100	75
FREEWAY	34	30	32	33
HERO	33069	15210	28000	N/A
PRIVATEEYE	8684	99	200	N/A
GRAVITAR	2722	239	1600	482
FROSTBITE	6439	352	4800	5214
VENTURE	0	0	1400	445

even a random exploration occasionally leads to a positive reward on these games, and self-imitation learning helps exploit such an experience to learn a good policy from it. This can drive deep exploration when the improved policy gets closer to the next source of reward. This result supports our claim that exploiting past experiences can often help exploration.

We further compared our method against the state-of-the-art count-based exploration actor-critic agents (A3C-CTS (Bellemare et al., 2016), Reactor-PixelCNN (Ostrovski et al., 2017), and SimHash (Tang et al., 2017)). These methods learn a density model of the observation or a hash function and use it to compute pseudo visit count, which is used to compute an exploration bonus reward. Even though our method does not have an explicit exploration bonus that encourages exploration, we were curious how well our self-imitation learning approach performs compared to these exploration approaches.

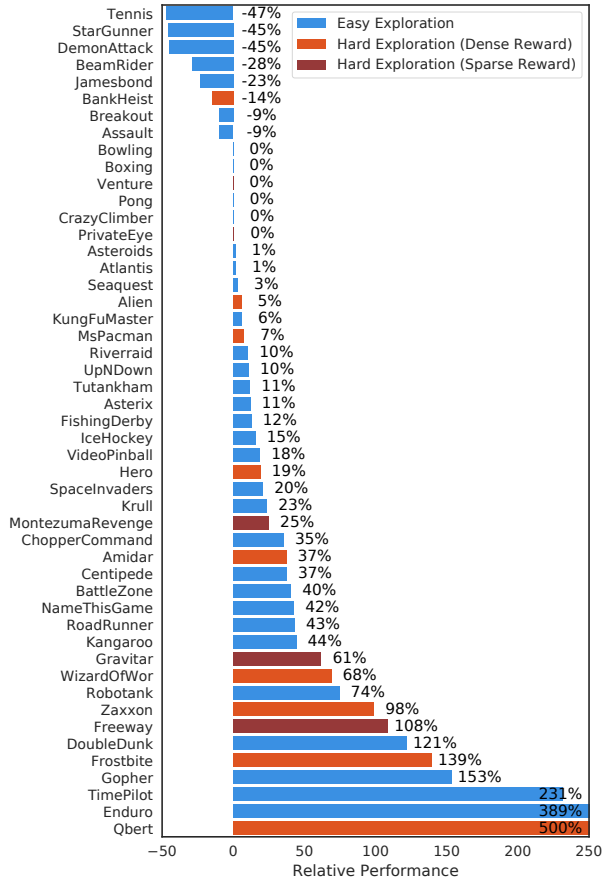


Figure 4. Relative performance of A2C+SIL over A2C.

Interestingly, Table 1 shows that A2C with our self-imitation learning (A2C+SIL) achieves better results on 6 out of 7 hard exploration games without any technique that explicitly encourages exploration. This result suggests that it is important to exploit past good experiences as well as efficiently explore the environment to drive deep exploration.

On the other hand, we found that A2C+SIL never receives a positive reward on Venture during training. This makes it impossible for our method to learn a good policy because there is no good experience to exploit, whereas one of the count-based exploration methods (Reactor-PixelCNN) achieves a better performance, because the agent is encouraged to explore different states even in the absence of reward signal from the environment. This result suggests that an advanced exploration method is essential in such environments where a random exploration never generates a good experience within a reasonable amount of time. Combining self-imitation learning with state-of-the-art exploration methods would be an interesting future work.

5.4. Overall Performance on Atari Games

To see how useful self-imitation learning is across various types of environments, we evaluated our self-imitation learn-

Table 2. Performance of agents on 49 Atari games after 50M steps (200M frames) of training. ‘ACPER’ represents A2C with prioritized replay using ACER objective. ‘Median’ shows median of human-normalized scores. ‘>Human’ shows the number of games where the agent outperforms human experts.

AGENT	MEDIAN	>HUMAN
A2C	96.1%	23
ACPER	46.8%	18
A2C+SIL	138.7%	29

ing method on 49 Atari games. It turns out that our method (A2C+SIL) significantly outperforms A2C in terms of median human-normalized score as shown in Table 2. Figure 4 shows the relative performance of A2C+SIL compared to A2C using the measure proposed by Wang et al. (2016). It is shown that our method (A2C+SIL) improves A2C on 35 out of 49 games in total and 11 out of 14 hard exploration games defined by Bellemare et al. (2016). It is also shown that A2C+SIL performs significantly better on many easy exploration games such as Time Pilot as well as hard exploration games. We observed that there is a certain learning stage where the agent suddenly achieves a high score by chance on such games, and our self-imitation learning exploits such experiences as soon as the agent experiences them.

On the other hand, we observed that our method often learns faster at the early stage of learning, but sometimes gets stuck at a sub-optimal policy on a few games, such as James Bond and Star Gunner. This suggests that excessive exploitation at the early stage of learning can hurt the performance. We found that reducing the number of SIL updates per iteration or using a small weight for the SIL objective in a later learning stage indeed resolves this issue and even improve the performance on such games, though the reported numbers are based on the single best hyperparameter. Thus, automatically controlling the degree of self-imitation learning would be an interesting future work.

5.5. Effect of Lower Bound Soft Q-Learning

A natural question is whether existing off-policy actor-critic methods can also benefit from past good experiences by exploiting them. To answer this question, we trained ACPER (A2C with prioritized experience replay) which performs off-policy actor-critic update proposed by ACER (Wang et al., 2017) by using the same prioritized experience replay as ours, which uses $(R - V_\theta)_+$ as sampling priority. ACPER can also be viewed as the original ACER with our proposed prioritized experience replay.

Table 2 shows that ACPER performs much worse than our A2C+SIL and is even worse than A2C. We observed that ACPER also benefits from good episodes on a few hard exploration games (e.g., Freeway) but was very unstable on many other games.

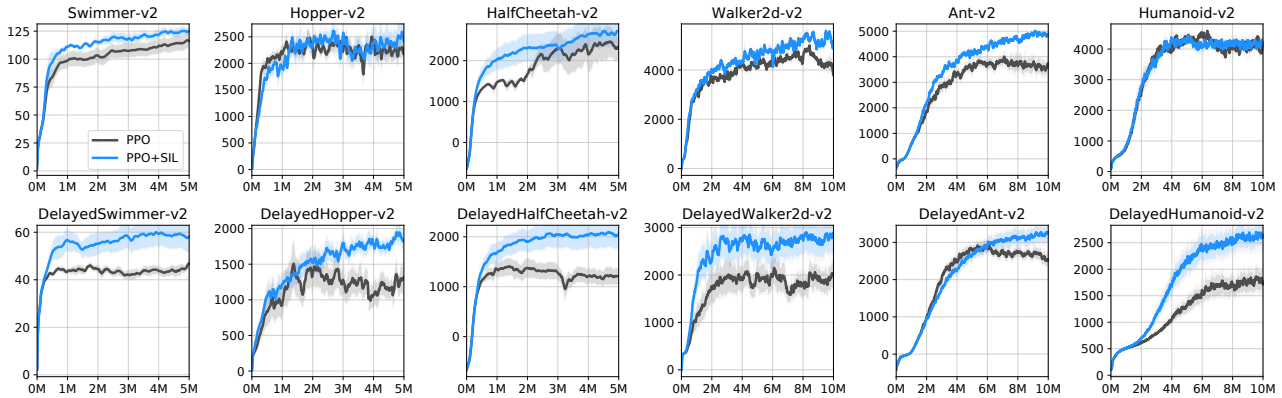


Figure 5. Performance on OpenAI Gym MuJoCo tasks (top row) and delayed-reward versions of them (bottom row). The learning curves are averaged over 10 random seeds.

We conjecture that this is due to the fact that the ACER objective has an importance weight term ($\pi(a|s)/\mu(a|s)$). This approach may not benefit much from the good experiences in the past if the current policy deviates too much from the decisions made in the past. On the other hand, the proposed self-imitation learning objective (Eq. 1) does not have an importance weight and can learn from any behavior, as long as the behavior policy performs better than the learner. This is because our gradient estimator can be interpreted as lower-bound-soft-Q-learning, which updates the parameter directly towards the optimal Q-value regardless of the similarity between the behavior policy and the learner as discussed in Section 4.2. This result shows that our self-imitation learning objective is suitable for exploiting past good experiences.

5.6. Performance on MuJoCo

This section investigates whether self-imitation learning is beneficial for continuous control tasks and whether it can be applied to other types of policy optimization algorithms, such as proximal policy optimization (PPO) (Schulman et al., 2017b). Note that unlike A2C, PPO does not have a strong theoretical connection to our SIL objective. However, we claim that they can still be complementary to each other in that both PPO and SIL try to update the policy and the value towards the optimal policy and value. To empirically verify this, we implemented PPO+SIL, which updates the parameter using both the PPO algorithm and our SIL algorithm and evaluated it on 6 MuJoCo tasks in OpenAI Gym (Brockman et al., 2016).

The result in Figure 5 shows that our self-imitation learning improves PPO on Swimmer, Walker2d, and Ant tasks. Unlike Atari games, the reward structure in this benchmark is smooth and dense in that the agent always receives a reasonable amount of reward according to its continuous progress. We conjecture that the agent has a relatively low chance to occasionally perform well and learn much faster

by exploiting such an experience in this type of domain. Nevertheless, the overall improvement suggests that self-imitation learning can be generally applicable to actor-critic architectures and a variety of tasks.

To verify our conjecture, we further conducted experiments by delaying reward the agent gets from the environment. More specifically, the modified tasks give an accumulated reward after every 20 steps (or when the episode terminates). This makes it more difficult to learn a good policy because the agent does not receive a reward signal for every step. The result is shown in the bottom row in Figure 5. Not surprisingly, we observed that both PPO and PPO+SIL perform worse on the delayed-reward tasks than themselves on the standard OpenAI Gym tasks. However, it is clearly shown that the gap between PPO+SIL and PPO is larger on delayed-reward tasks compared to standard tasks. Unlike the standard OpenAI Gym tasks where reward is well-designed and dense, we conjecture that the chance of achieving high overall rewards is much low in the delayed-reward tasks. Thus, the agent can benefit more from self-imitation learning because self-imitation learning captures such rare experiences and learn from them.

6. Conclusion

In this paper, we proposed self-imitation learning, which learns to reproduce the agent’s past good experiences, and showed that self-imitation learning is very helpful on hard exploration tasks as well as a variety of other tasks including continuous control tasks. We also showed that a proper level of exploitation of past experiences during learning can drive deep exploration, and that self-imitation learning and exploration methods can be complementary. Our results suggest that there can be a certain learning stage where exploitation is more important than exploration or vice versa. Thus, we believe that developing methods for balancing between exploration and exploitation in terms of collecting and learning from experiences is an important future research direction.

Acknowledgement

This work was supported by NSF grant IIS-1526059. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the views of the sponsor.

References

- Abolafia, D. A., Norouzi, M., and Le, Q. V. Neural program synthesis with priority queue training. *arXiv preprint arXiv:1801.03526*, 2018.
- Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. In *NIPS*, 2016.
- Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.
- Blundell, C., Uria, B., Pritzel, A., Li, Y., Ruderman, A., Leibo, J. Z., Rae, J., Wierstra, D., and Hassabis, D. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016.
- Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. OpenAI Baselines. <https://github.com/openai/baselines>, 2017.
- Gruslys, A., Azar, M. G., Bellemare, M. G., and Munos, R. The reactor: A sample-efficient actor-critic architecture. In *ICLR*, 2018.
- Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *ICML*, 2017.
- He, F. S., Liu, Y., Schwing, A. G., and Peng, J. Learning to play in a day: Faster deep reinforcement learning by optimality tightening. In *ICLR*, 2017.
- Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Sendonaris, A., Dulac-Arnold, G., Osband, I., Agapiou, J., et al. Deep q-learning from demonstrations. In *AAAI*, 2018.
- Konda, V. R. and Tsitsiklis, J. N. Actor-critic algorithms. In *NIPS*, 2000.
- Lengyel, M. and Dayan, P. Hippocampal contributions to control: the third way. In *NIPS*, 2008.
- Liang, C., Berant, J., Le, Q., Forbus, K. D., and Lao, N. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *ACL*, 2016.
- Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *ICLR*, 2016.
- Lin, L. J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321, 1992.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.
- Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.
- Moore, A. W. and Atkeson, C. G. Memory-based reinforcement learning: Efficient computation with prioritized sweeping. In *NIPS*, 1992.
- Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. G. Safe and efficient off-policy reinforcement learning. In *NIPS*, 2016.
- Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Bridging the gap between value and policy based reinforcement learning. In *NIPS*, 2017.
- Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Overcoming exploration in reinforcement learning with demonstrations. *arXiv preprint arXiv:1709.10089*, 2017.
- O’Donoghue, B., Munos, R., Kavukcuoglu, K., and Mnih, V. Combining policy gradient and q-learning. In *ICLR*, 2017.
- Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. In *NIPS*, 2016.
- Ostrovski, G., Bellemare, M. G., Oord, A. v. d., and Munos, R. Count-based exploration with neural density models. In *ICML*, 2017.
- Precup, D., Sutton, R. S., and Singh, S. P. Eligibility traces for off-policy policy evaluation. In *ICML*, 2000.
- Precup, D., Sutton, R. S., and Dasgupta, S. Off-policy temporal difference learning with function approximation. In *ICML*, 2001.

- Pritzel, A., Urias, B., Srinivasan, S., Puigdomènech, A., Vinyals, O., Hassabis, D., Wierstra, D., and Blundell, C. Neural episodic control. In *ICML*, 2017.
- Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. In *ICLR*, 2016.
- Schmidhuber, J. Adaptive confidence and adaptive curiosity. In *Institut für Informatik, Technische Universität München*, 1991.
- Schulman, J., Chen, X., and Abbeel, P. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017a.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017b.
- Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *ICML*, 2014.
- Stadie, B. C., Levine, S., and Abbeel, P. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.
- Strehl, A. L. and Littman, M. L. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.
- Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, 1999.
- Tang, H., Houthoofd, R., Foote, D., Stooke, A., Chen, O. X., Duan, Y., Schulman, J., DeTurck, F., and Abbeel, P. #exploration: A study of count-based exploration for deep reinforcement learning. In *NIPS*, 2017.
- Thrun, S. B. The role of exploration in learning control. *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, 1992.
- Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. *IROS*, 2012.
- Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., and de Freitas, N. Dueling network architectures for deep reinforcement learning. In *ICML*, 2016.
- Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and de Freitas, N. Sample efficient actor-critic with experience replay. In *ICML*, 2017.
- Xu, H., Gao, Y., Lin, J., Yu, F., Levine, S., and Darrell, T. Reinforcement learning from imperfect demonstrations. In *ICML*, 2018.
- Ziebart, B. D. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, 2010.
- Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.

A. Hyperparameters

Table 3. A2C+SIL hyperparameters on Atari games.

Hyperparameters	Value
Architecture	Conv(32-8x8-4) -Conv(64-4x4-2) -Conv(64-3x3-1) -FC(512)
Learning rate	0.0007
Number of environments	16
Number of steps per iteration	5
Entropy regularization (α)	0.01
SIL update per iteration (M)	4
SIL batch size	512
SIL loss weight	1
SIL value loss weight (β^{sil})	0.01
Replay buffer size	10^5
Exponent for prioritization	0.6
Bias correction for prioritized replay	0.1 for hard exploration experiment (Section 5.3) 0.4 for overall evaluation (Section 5.4)

Table 4. PPO+SIL hyperparameters on MuJoCo.

Hyperparameters	Value
Architecture	FC(64)-FC(64)
Learning rate	Best chosen from {0.0003, 0.0001, 0.00005, 0.00003}
Horizon	2048
Number of epochs	10
Minibatch size	64
Discount factor (γ)	0.99
GAE parameter (λ)	0.95
Entropy regularization (α)	0
SIL update per batch	10
SIL batch size	512
SIL loss weight	0.1
SIL value loss weight (β)	Best chosen from {0.01, 0.05}
Replay buffer size	50000
Exponent for prioritization	Best chosen from {0.6, 1.0}
Bias correction for prioritized replay	0.1

B. Performance on Atari Games

Table 5. Performances on 49 Atari games with 30 random no-op after 50M steps of training (200M frames).

	A2C	ACPER	A2C+SIL
Alien	1859.2	390.2	2242.2
Amidar	739.9	424.8	1362.0
Assault	1981.4	818.2	1812.0
Asterix	16083.3	3533.1	17984.2
Asteroids	2056.0	1780.1	2259.4
Atlantis	3032444.2	58012.5	3084781.7
BankHeist	1333.7	1203.2	1137.8
BattleZone	10683.3	15025.0	25075.0
BeamRider	3931.7	2602.4	2366.2
Bowling	31.2	59.3	31.1
Boxing	99.7	100.0	99.6
Breakout	501.6	118.5	452.0
Centipede	3857.8	7790.1	7559.5
ChopperCommand	3464.2	1307.5	6710.0
CrazyClimber	129715.8	19918.8	130185.8
DemonAttack	18331.4	4777.5	10140.5
DoubleDunk	-0.5	-9.8	21.5
Enduro	0.0	3113.3	1205.1
FishingDerby	39.1	59.8	55.8
Freeway	0.0	31.4	32.2
Frostbite	339.5	2342.5	6289.8
Gopher	9358.5	3919.5	23304.2
Gravitar	329.2	627.5	1874.2
Hero	28008.1	13299.1	33156.7
IceHockey	-4.3	0.0	-2.4
Jamesbond	399.2	598.1	310.8
Kangaroo	1563.3	5875.0	2888.3
Krull	8883.9	11323.2	10614.6
KungFuMaster	32507.5	20485.0	34449.2
MontezumaRevenge	5.8	0.0	1100.0
MsPacman	2843.4	1016.0	4025.1
NameThisGame	11174.2	2888.0	14958.2
Pong	20.8	20.9	20.9
PrivateEye	210.8	100.0	661.2
Qbert	17605.2	657.2	104975.6
Riverraid	13036.0	2224.5	14306.1
RoadRunner	39874.2	8925.0	57071.7
Robotank	3.2	7.7	10.5
Seaquest	1795.2	804.5	2456.5
SpaceInvaders	2466.1	729.5	2951.7
StarGunner	57371.7	1107.5	31309.2
Tennis	-10.3	-17.0	-17.3
TimePilot	5346.7	3952.5	10811.7
Tutankham	305.6	270.7	340.5
UpNDown	48131.8	9562.5	53314.6
Venture	0.0	0.0	0.0
VideoPinball	391241.6	21797.7	461522.4
WizardOfWor	4196.7	1550.0	7088.3
Zaxxon	124.2	4278.8	9164.2

Self-Imitation Learning

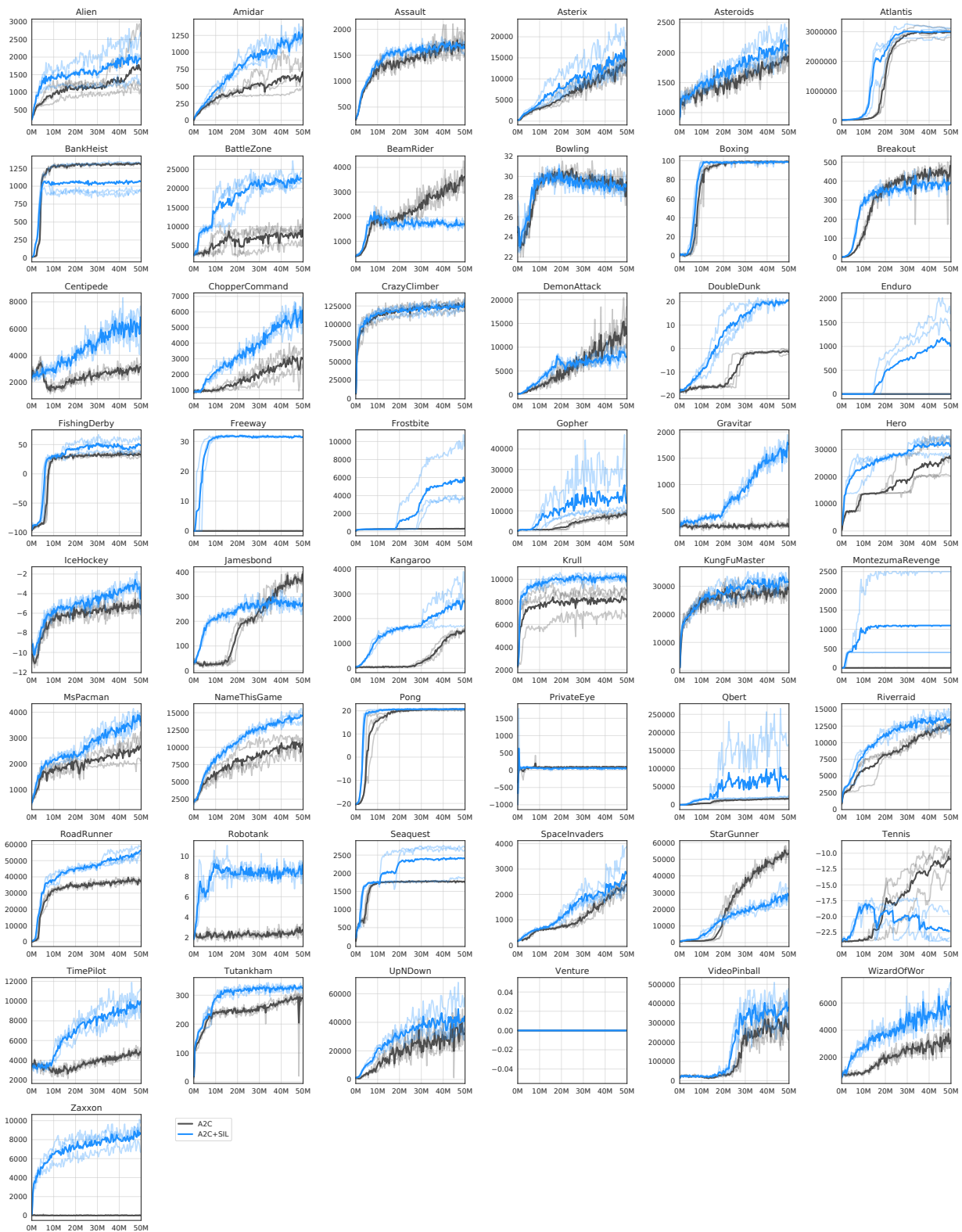


Figure 6. Learning curves on 49 Atari games.