

Accelerating Self-Imitation Learning from Demonstrations via Policy Constraints and Q-Ensemble

ABSTRACT

Deep reinforcement learning (DRL) provides a new way to generate robot control policy. However, the process of training control policy requires lengthy exploration, resulting in a low sample efficiency of reinforcement learning (RL) in real-world tasks. Both imitation learning (IL) and learning from demonstrations (LfD) improve the training process by using expert demonstrations, but imperfect expert demonstrations can mislead policy improvement. Offline to Online reinforcement learning requires a lot of offline data to initialize the policy, and distribution shift can easily lead to performance degradation during online fine-tuning. To solve the above problems, we propose a learning from demonstrations method named A-SILfD, which treats expert demonstrations as the agent’s successful experiences and uses experiences to constrain policy improvement. Furthermore, we prevent performance degradation due to large estimation errors in the Q-function by the ensemble Q-functions. Our experiments show that A-SILfD can significantly improve sample efficiency using a small number of different quality expert demonstrations. In four Mujoco continuous control tasks, A-SILfD can significantly outperform baseline methods after 150,000 steps of online training and is not misled by imperfect expert demonstrations during training.

KEYWORDS

Deep Reinforcement Learning, Learning from Demonstrations, Self-Imitation Learning, Sample Efficiency

ACM Reference Format:

. 2023. Accelerating Self-Imitation Learning from Demonstrations via Policy Constraints and Q-Ensemble. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 INTRODUCTION

Deep reinforcement learning (DRL) for sequential decision problems has shown significant advantages, with excellent performance in robot control [3, 23]. However, building a control policy based on DRL requires a fine-grained design of the training process in order to explore the environment. The DRL agents must struggle with high sample complexity for a long time, hindering the development of DRL in robot control. Therefore, researchers have proposed two approaches for building a DRL control policy. On the one hand, some methods train the control policy in a simulation environment with unlimited data sources and perform the sim-to-real transfer [53]. On the other hand, many methods use offline data collected from previous training to guide exploration and improve sample efficiency.

There are several methods to use expert demonstrations. In imitation learning (IL), the agents do not need environmental rewards but learn policy by imitating the teacher’s behavior [7, 15, 22, 26, 41]. Therefore, IL requires high-quality expert demonstrations. In practice, expert demonstrations may be generated by human or simple control methods, and the quality of expert demonstrations varies. IL performs poorly in imperfect expert demonstrations. Learning from demonstrations (LfD) also uses expert demonstrations, combining the advantages of IL and RL to improve the sample efficiency of online training [40]. Existing LfD methods enhance RL by either putting expert demonstrations into a replay buffer for value estimation or using expert demonstrations to pre-train the policy [21, 47]. Unfortunately, the utilization of expert demonstration is inefficient. Other LfD methods are based on on-policy RL algorithms that add a demonstration-guided term to guide policy improvement [10, 24, 42]. Offline RL uses a large number of expert demonstrations to learn the policy without interacting with the environment [12, 14, 29, 30]. Offline to Online RL combines offline training and online fine-tuning, enabling policy to achieve better performance [28, 32, 35, 54]. However, both types of methods require a large amount of offline data to train the policy, which cannot be effective under limited expert demonstrations. In addition, Offline to Online RL methods faces severe Q-function estimation errors due to distribution shift.

Noting the limitations of existing methods, we propose a LfD method, **Accelerating Self-Imitation Learning from Demonstrations (A-SILfD)**. It is based on the Actor-Critic update framework and more effectively uses a small number of expert demonstrations to improve sample efficiency.

The main contributions of this paper are as follows.

- **Adaptive adjustment of experience replay buffer:** We add expert demonstrations to the experience replay buffer and replace old data with better experiences during training to avoid misleading policy improvement through imperfect demonstrations.
- **Full use of experience data:** We use policy constraints to make the agent imitate agent’s successful experience and realize the full use of data in the experience replay buffer.
- **Ensemble Q-functions:** By introducing randomness through ensemble learning, our method smoothes the policy improvement process and avoids the performance degradation caused by distribution shift.

We conducted experiments in four Mujoco continuous control tasks using expert demonstrations of different quality (expert demonstrations, mixed expert demonstrations, and imperfect expert demonstrations). The same demonstrations were used for all methods in the experiments. Our experimental results show that our methods are not affected by the quality of the expert demonstrations. Furthermore, our method outperforms all baseline methods regarding sample efficiency and final performance for different

Table 1: Comparison of different methods.

Method	Environmental Reward	Trajectory Reward	Number of Trajectories	Offline training	Trajectory Evaluation	Policy Constraints	Sample Efficiency	Problem
IL	×	×	Small	×	✓	×	Low	Misleading
LfD	✓	✓	Small	Not True	✓	✓	Medium	Misleading and Utilization
Offline to Online RL	✓	✓	Large	✓	×	✓	High	Distribution Shift

quality of expert demonstrations. Moreover, we performed an ablation experiment to analyze the impact of each component on the algorithm.

2 RELATED WORK

2.1 Imitation Learning

IL aims to train a policy to mimic an expert policy as closely as possible and does not use environmental rewards. The simplest IL method is behavioral cloning (BC) [6], which uses a supervised learning approach to fit expert demonstrations. GAIL [22] is a classical adversarial-based IL method that uses discriminators to classify expert and sampled data. Besides, AIRL [11], IC-GAIL [51], WGAIL [48] and F-IRL [36] are also adversarial-based IL methods. DAC [26] extended IL to the Actor-Critic methods. OPOLO [55] emphasizes imitating only expert states. Self-Adaptive Imitation Learning (SAIL) [56] uses sub-optimal expert demonstrations to adaptively adjust the teacher buffer to bring the policy close to and beyond the expert’s policy. IL is limited by the expert demonstrations’ quality and requires additional sample data to learn the discriminator, resulting in a less efficient sample for the method.

Self-imitation learning (SIL) [38] is different from imitation learning. SIL is a method that uses the agent’s successful experience to encourage the agent to explore and improve the sample efficiency. Our work draws on the idea of SIL.

2.2 Learning from Demonstrations

Unlike IL, the idea of LfD is to guide online learning with the help of expert demonstrations, which require environmental rewards. DDPGfD [47] first uses expert demonstrations and training policy, followed by online fine-tuning. DQfD [21] trains the Q-network with expert demonstrations to achieve an accurate Temporal-Difference (TD) error at the beginning of the interaction between the agent and the environment. POfD [24] uses the idea of GAIL to reshape a reward function by minimizing the occupancy measure of agents and experts. LOGO [42] uses demonstrations to guide exploration during the policy guidance phase, but it is based on the on-policy method—TRPO [43], resulting in a less efficient sample. LfD can learn effective policy faster by using fewer expert demonstrations in combination with online learning, but the pre-trained policy may also perform poorly in online learning. Our method allows for more efficient use of imperfect expert demonstrations.

2.3 Offline RL and Offline to Online RL

Offline RL uses a large amount of pre-collected offline data to train the policy without online interaction. CQL [30] adds a regularization term to the update of the Q-function so that the Q-function

can conservatively estimate a lower bound on the Q-value. TD3-BC [12] avoids the distribution shift problem by adding additional BC loss in the policy improvement.

Offline to Online RL emphasizes offline training and online fine-tuning. However, a severe bootstrapping error is triggered due to the distribution shift during the online training, which destroys a good policy for Offline RL training. AWAC [35] solves the distribution shift by adding regular constraints to the policy. Balanced Replay [32] sets different priorities for online and offline data. IQL [27] uses the SARSA style to reconstruct the value function. Offline to Online RL is ineffective when there are few expert demonstrations. Moreover, online training can easily cause the offline policy to collapse. In contrast, our method does not have offline training, directly uses a small number of expert demonstrations to guide online training, and is less affected by distribution shift.

Table 1 compares the differences between the three kinds. The main problems can be summarized as imperfect demonstrations misleading policy improvement, low utilization of demonstrations, and performance degradation due to distribution shift. In this work, we attempt to fill the aforementioned research gaps.

3 LEVERAGE THE CHALLENGES OF EXPERT DEMONSTRATIONS

In this section, we have compiled the challenges that can be faced using expert demonstrations.

3.1 Problem Setting

In real-world robotic tasks, a small number of trajectories can be collected by human experts or simple controllers (e.g., PID controllers), which we call expert demonstrations, defined as $\zeta = \{\zeta_1, \zeta_2, \dots, \zeta_n\}$, where $\zeta_i = (s_t, a_t, r_t, s_{t+1})_{t=1}^T$ and $i = 1 \sim n$. The quality of these trajectories can be judged using the reward function. Due to the difficulty of collecting data in the real world, our goal is to fully use a small number of trajectories ζ , combined with online training to improve the sample efficiency. Eventually, the agent outperforms the imperfect sampling policy and approaches the optimal policy π^* . We mainly consider off-policy RL algorithms, which can train the agent with online and offline data. In particular, the Actor-Critic method can make full use of off-policy data. The two easiest ways to use expert demonstrations are as follows. (1) Use BC to pre-train the policy and fine-tune the policy with online training. (2) Populate the offline data directly into the replay buffer and train the policy. However, these methods have the following disadvantages: (1) They do not take full advantage of the demonstrations; (2) The demonstrations may be imperfect, leading to negative guidance during online training [35].

3.2 Imperfect Expert Demonstrations

As shown in Figure 1, the left side represents the best trajectory, while two imperfect expert trajectories appear on the right. Suppose the policy is initialized directly using imperfect expert demonstrations. Then, when using this policy for decision-making, the policy directs the agent to reach the *goal* along the sub-optimal route.

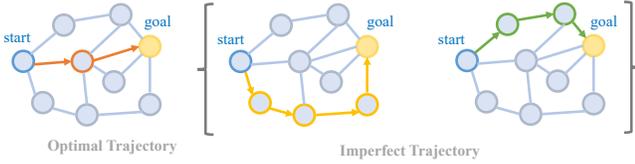


Figure 1: Schematic diagram of optimal and imperfect trajectories.

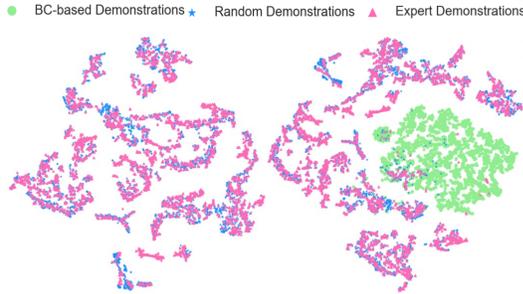


Figure 2: The distribution of different data in the state space. We use t-SNE to downscale the states of Hopper-v2 to the two-dimensional plane.

In the continuous state space, the influence of imperfect expert demonstrations remains. The blue area in Figure 2 shows the state distribution of the expert demonstrations, and the pink area shows the state distribution of the data sampled using the BC policy. The demonstrations collected by the pre-trained policy will overlap with the expert demonstrations in the state space. As a result, imperfect demonstrations will be used to update the policy. This negative guidance may move the agent away from the state space where the optimal trajectories are located, thus affecting the agent’s exploration.

3.3 Distribution Shift and Bootstrapping Error Accumulation

If we want to explore the unknown region, we need to increase the randomness of the policy. As shown on the right side of Figure 2, the blue area is the state of the expert demonstrations, while the green area is the state of the sampled data of the policy with more randomness. While random data may start with the same state as the expert demonstrations, actions with greater randomness may result in subsequent states that deviate significantly from the expert demonstrations. Due to the distribution shift, the data covers different state-action regions. Therefore, the Q-function cannot provide accurate value estimates for such out-of-distribution (OOD) samples. Severe bootstrapping errors can destroy the good initial policy obtained by pre-training.

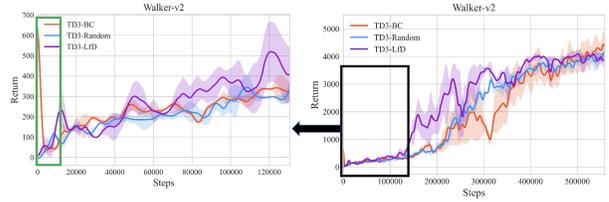


Figure 3: Experiments on Walker2d-v2. TD3-Random denotes the TD3 algorithm trained from scratch. TD3-BC denotes the TD3 algorithm using a BC initialization policy. TD3-LfD denotes the TD3 algorithm based on the LfD idea.

Let us analyze what the errors associated with the Q-function are. Let the actual Q-value for the current k iterations be $Q^\pi(s, a)$ and the estimated Q-value be $\hat{Q}_k(s, a)$, and the estimation error can be expressed as $\Gamma_k = |Q_k(s, a) - Q^\pi(s, a)|$.

The current Bellman error is $\delta_k(s, a) = |Q_k(s, a) - \hat{Q}_{k-1}(s, a)|$, where $\hat{Q}_{k-1}(s, a) = r(s, a) + \gamma \mathbb{E}_{T(s'|s, a)} [\max_{a'} \hat{Q}_{k-1}(s', a')]$.

We can obtain

$$\Gamma_k(s, a) \leq \delta_k(s, a) + \gamma \max_{a'} \mathbb{E}_{T(s'|s, a)} [\Gamma_{k-1}(s, a)] \quad (1)$$

We want $\delta_k(s, a)$ to be as large as possible because $\delta_k(s, a)$ affects the update of the Q-function, which helps to fit the out-of-distribution state-action pairs and reduces the estimation error. However, in practice, out-of-distribution samples lead to a large estimation error Γ_k , accumulating in each iteration. Thus it can lead to the training policy that deviates significantly from the optimal policy.

Figure 3 shows the problem caused by the distribution shift. The right panel shows the reward curve for the evaluation phase of the training process, and the left panel captures the first 120,000 steps of the reward curve. If the BC initialization policy is used, it leads to a large estimation error Γ_k of the Q-function. The estimation error accumulates in the early stage of training, thus distorting the pre-trained policy. As a result, the agent performance may increase degradation rapidly (as in the green box on the left side of Figure 3). Although the idea of LfD can be used for pre-training, the estimation error Γ_k of the Q-function in the unknown region will also be larger, resulting in the pre-training policy not achieving the expected results.

4 METHOD

In this section, we describe A-SILfD. Figure 4 illustrates the training process of A-SILfD. First, Section 4.2 describes how the expert demonstrations are used and how the experience replay buffer is adaptively adjusted. Then, in Section 4.3, we introduce our policy constraints method. Finally, Section 4.4 describes how to train Ensemble Q-functions and policy.

4.1 Preliminaries

RL is typically modeled with Markovian Decision Processes (MDP) and is defined as a five-tuple $M = \langle \mathcal{S}, \mathcal{A}, T, r, \gamma \rangle$, where \mathcal{S}, \mathcal{A} represent state and action spaces; $T(s'|s, a)$ is the state transfer probability; $r(s, a)$ represents the reward function; $\gamma \in [0, 1]$ is the discount

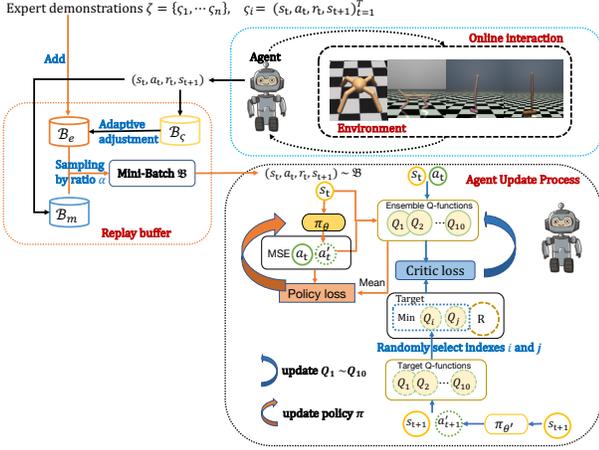


Figure 4: The training flow of A-SILfD.

factor [2] indicating the influence of future reward at the current time.

The goal of RL is to obtain a policy π that maximizes the cumulative (discounted) reward:

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | a_t = \pi(\cdot | s_t) \right]$$

The Actor-Critic methods learn both the value function and the policy function. Twin Delayed Deep Deterministic policy gradient algorithm (TD3) [13] is an off-policy Actor-Critic method that learns Q-function $Q_{\phi}(s, a)$ parameterized by ϕ and a deterministic policy π_{θ} modeled as a feedforward network, parameterized by θ . TD3 alternates between critic and actor updates by minimizing the following objectives respectively:

$$\begin{aligned} \mathcal{L}^{critic}(\phi) &= \mathbb{E}_{(s, a, s' \in \mathcal{B})} [Q_{\phi}(s, a) - r - \gamma Q_{\phi'}(s', \pi_{\theta}(s'))] \\ \mathcal{L}^{actor}(\theta) &= \mathbb{E}_{s \in \mathcal{B}, a \sim \pi_{\theta}} [-Q_{\phi}(s, a)] \end{aligned}$$

where \mathcal{B} is the replay buffer and ϕ' is the target parameters.

4.2 Self-Imitation Learning from Demonstrations

We propose A-SILfD, which can use expert demonstrations and the agent's successful experiences. Our method differs from existing LfD methods in that we consider the expert demonstration as the agent's successful experience as a reference for the agent to learn.

Our method has two replay buffers. First, we store expert demonstrations and successful experiences into the experience replay buffer \mathcal{B}_e . We store transitions sampled by the agent into the sample replay buffer \mathcal{B}_m .

For the deterministic policy, if the cumulative reward of the trajectory generated by the policy π_i is high, it is closer to the optimal policy π^* . As the policy is updated, trajectories with high cumulative reward should be fully used as successful experiences. In addition, we use the cumulative reward of the whole trajectory as the basis for judging the quality of the trajectory, which can effectively avoid the influence of locally optimal actions. Locally

optimal actions may lead to a high reward for a particular step, but a high cumulative reward means a high quality of the whole trajectory.

4.2.1 Adjustment of experience replay buffer \mathcal{B}_e . We record the cumulative reward list $r_{sum} = \{r_{\zeta_1}, r_{\zeta_2}, \dots, r_{\zeta_n}\}$ of the trajectories in \mathcal{B}_e and derive the minimum cumulative reward $r_{min}(\zeta)$. During the training process, we store the transitions of each episode in the trajectory replay buffer \mathcal{B}_{ζ} . At the end of each episode, we calculate the cumulative reward $r_e(\zeta)$ for transitions in \mathcal{B}_{ζ} . If $r_e(\zeta) > r_{min}(\zeta)$, we add all transitions in \mathcal{B}_{ζ} to \mathcal{B}_e , update the cumulative reward list r_{sum} and recalculate $r_{min}(\zeta)$. By tuning \mathcal{B}_e , we ensure that the quality of trajectories in \mathcal{B}_e increases gradually as the policy improvement, avoiding misleading policy improvement by imperfect experience.

4.2.2 Usage of \mathcal{B}_e and \mathcal{B}_m . Since \mathcal{B}_m stores all transitions sampled by online interactions, and \mathcal{B}_e stores the agent's successful experiences, some successful experiences will exist in \mathcal{B}_m and \mathcal{B}_e . Since the size of \mathcal{B}_e is much smaller than that of \mathcal{B}_m , when sampling both proportionally, the successful experiences will have a higher probability of being sampled as training data, ensuring the full use of experiences.

4.3 Policy Constraints

The goal of RL is to maximize the expected cumulative (discounted) rewards $\eta(\pi_{\theta}) = \mathbb{E}_{\pi_{\theta}} [\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$ along the entire decision process under the current policy π_{θ} .

We have an experience replay buffer \mathcal{B}_e and a sample replay buffer \mathcal{B}_m . We use $d_{\pi}(s, a)$ to denote the state-action distribution under the policy π . Moreover, we denote the state-action distribution of the samples in \mathcal{B}_e and \mathcal{B}_m by $d_e(s, a)$ and $d_m(s, a)$.

We expect the $d_{\pi}(s, a)$ and $d_e(s, a)$ to be close as the policy is learned. In addition, we need to avoid over-exploration of the action. Over-exploration actions can cause the next state to be shifted far from the known region. So, we represent the policy improvement as the following constrained optimization problem.

$$\begin{aligned} \min_{\theta} J(\theta) = -\eta(\pi_{\theta}) \quad s.t. \quad & \mathbb{D}[d_{\pi_{\theta}}(s, a) | d_e(s, a)] < \kappa \\ & \text{and} \quad \mathbb{D}[d_{\pi_{\theta}}(s, a) | d_m(s, a)] < \kappa \end{aligned} \quad (2)$$

\mathbb{D} denotes a measure of distribution distance. $\mathbb{D}[d_{\pi}(s, a) | d_e(s, a)] < \kappa$ encourages the state-action distribution of the current policy π to match the agent's successful experience state-action distribution. $\mathbb{D}[d_{\pi}(s, a) | d_m(s, a)] < \kappa$ seems counterintuitive at first glance, but we aim to avoid exploring too much.

In practice, we avoid explicitly dealing with the inequality constraint by solving the dual problem of Equation 2 to solve for Equation 2. Namely, we consider the Lagrangian

$$\mathcal{L}(\theta, \lambda) = -\eta(\pi_{\theta}) + \lambda(\mathcal{G}(\pi_{\theta}) - 2\kappa) \quad (3)$$

where $\mathcal{G}(\pi_{\theta}) = \mathbb{D}[d_{\pi_{\theta}}(s, a) | d_e(s, a)] + \mathbb{D}[d_{\pi_{\theta}}(s, a) | d_m(s, a)]$ and solve the problem $\min_{\theta} \max_{\lambda > 0} \mathcal{L}(\theta, \lambda)$ by alternately optimizing θ and λ .

Optimizing λ with fixed θ is equivalent to

$$\min_{\lambda > 0} \lambda(2\kappa - \mathcal{G}(\pi_{\theta})) \quad (4)$$

and optimizing θ with fixed λ boils down to solving

$$\min_{\theta} -\eta (\pi_{\theta}) + \lambda \mathcal{G}(\pi_{\theta}) \quad (5)$$

To solve the above problem, we need to adjust to both θ and λ .

In the optimization Equation 4 process, the constraints requires $\mathcal{G}(\pi_{\theta}) < 2\kappa$. Therefore, if $\mathcal{G}(\pi_{\theta}) > 2\kappa$, the λ must be increased. When $\mathcal{G}(\pi_{\theta}) < 2\kappa$, the λ can be made as small as possible. All $\lambda > 0$ encourage $\mathcal{G}(\pi_{\theta})$ to decrease gradually. From another point of view, $\mathcal{G}(\pi_{\theta}) > 2\kappa$ indicates a constraint violation and requires an increase in the lambda to impose a larger penalty.

In the optimization Equation 5 process, as the policy continues to converge, the state-action distributions of $d_{\pi_{\theta}}(s, a)$, $d_e(s, a)$ and $d_m(s, a)$ keep approaching, $\mathcal{G}(\pi_{\theta})$ gradually decreases.

To facilitate training, we can fix λ as a hyper-parameter and turn this problem into a function related only to θ . The optimization problem becomes

$$\theta \leftarrow \arg \min_{\theta} \mathcal{L}(\theta, \lambda) = \arg \min_{\theta} J(\theta) + \lambda \mathcal{G}(\pi_{\theta}) \quad (6)$$

Since it is difficult to estimate the state-action distribution of each small batch sample during training and impossible to estimate $\mathcal{G}(\pi_{\theta})$ accurately, we simplify Equation 6. After considering the implications of $\mathcal{G}(\pi_{\theta})$, we take the mean square error between the predicted action $a' = \pi_{\theta}(s)$ and the actual action a as an approximate estimate of $\mathcal{G}(\pi_{\theta})$. Since the data of each mini-batch is sampled from two replay buffers, \mathcal{B}_e and \mathcal{B}_m , this approximate estimation can quickly reach the approximation of $\mathcal{G}(\pi_{\theta})$. The final optimization objective becomes

$$\theta \leftarrow \arg \min_{\theta} [J(\theta) + \lambda \mathbb{E}_{\pi_{\theta}} [(a - \pi_{\theta}(s))^2]] \quad (7)$$

We can expand or contract the constraints region by decreasing or increasing the λ . In the early stage of training, we want the policy π to mimic the agent's successful experience as much as possible, so we use a larger λ . As training continues, we want the agent to explore the unknown state space gradually, so we gradually decrease the λ , expanding the constraints region.

4.4 Random Selection of Ensemble Q-Functions

It is well known that the Actor-Critic method has been suffering from the over-estimation problem [13, 18]. At the same time, the distribution shift leads to excessive errors in the estimation of the Q-function, and the combination of the two causes further degradation in the performance of the Q-function. To more effectively mitigate the Q-function estimation error due to distribution shift, we use the idea of the REDQ (Chen et al.) [8] to train the ensemble Q-functions.

Our method is updated in the following way. The estimation of the Q-function follows the Bellman Equation and is updated using the Temporal-Difference method. We still use the target network to avoid the over-estimation problem.

Let the parameters of the ensemble Q-functions be $\{\phi_1, \phi_2, \dots, \phi_N\}$ and the parameters of the target network be $\{\phi'_1, \phi'_2, \dots, \phi'_N\}$. The ensemble Q-functions estimates for (s', a') are expressed as

$$Q_{\phi}(s', a') = \min_{i \in \mathcal{M}} Q_{\phi'_i}(s', a') \quad (8)$$

where \mathcal{M} denotes a subset of ensemble Q-functions, with the number of elements in the subset $M < N$.

Algorithm 1 A-SILfD

Input:

Initialize experience replay buffer \mathcal{B}_e , sample replay buffer \mathcal{B}_m ;
Initialize ensemble Q-functions parameters ϕ_1, \dots, ϕ_N ;
To achieve the above goals, we choose the mean θ ;
Initialize target parameters $\theta' \leftarrow \theta$ and $\phi'_i \leftarrow \phi_i (i := 1 \sim N)$;
Initialize batch size b , experience ratio $\alpha = 0.25$;
1: Add the expert demonstrations to \mathcal{B}_e , record the cumulative reward $\{r_{\zeta_1}, r_{\zeta_2}, \dots, r_{\zeta_n}\}$ of the trajectory in \mathcal{B}_m and get the minimum cumulative reward $r_{min}(\zeta)$;
2: **for** $i = 0$ to $n_episodes$ **do**
3: Initialize trajectory replay buffer R_{ζ} ;
4: **for** $t = 0$ to n_steps **do**
5: Act $a_t = \pi_{\theta}(s_t) + \varepsilon$, $\varepsilon \sim clip(\mathcal{N}(0, \sigma), -c, c)$;
6: Observe next state s_{t+1} and reward r_t
7: Add $(s_t, a_t, r_t, s_{t+1}, done)$ to R_B ;
8: Add $(s_t, a_t, r_t, s_{t+1}, done)$ to R_{ζ} ;
9: **if** $len(\mathcal{B}_m) > b$ **then**
10: $\{s_i, a_i, r_i, \dots\}_{i=1}^{\alpha \times b} \sim \mathcal{B}_e$, $\{s_i, a_i, r_i, \dots\}_{i=1}^{(1-\alpha) \times b} \sim \mathcal{B}_m$
11: Update parameters ϕ_1, \dots, ϕ_N using the Equation 9;
12: Update parameters θ using the Equation 11;
13: $\theta' \leftarrow \tau \times \theta + (1 - \tau) \times \theta'$;
14: $\phi'_i \leftarrow \tau \times \phi_i + (1 - \tau) \times \phi'_i$;
15: **end if**
16: **end for**
17: Calculate the cumulative reward $r_e(\zeta)$ in R_{ζ} ;
18: **if** $r_e(\zeta) > r_{min}(\zeta)$ **then**
19: $\mathcal{B}_e \leftarrow \mathcal{B}_e \cup R_{\zeta}$
20: Update the cumulative reward $\{r_{\zeta_1}, r_{\zeta_2}, \dots, r_{\zeta_n}\}$ of the trajectory in \mathcal{B}_m and get the minimum value $r_{min}(\zeta)$;
21: **end if**
22: **end for**
23: **return** policy π ;

Then the loss of the i-th Q-function is

$$\mathcal{L}_i^{critic} = \sum_{(s, a, r, s') \in \mathfrak{B}} (Q_{\phi_i}(s, a) - r - \gamma Q_{\phi}(s', a'))^2 \quad (9)$$

where \mathfrak{B} is the transitions of the currently sampled mini-batch and $a' = \pi_{\theta}(s') + \varepsilon$, $\varepsilon \sim clip(\mathcal{N}(0, \sigma), -c, c)$.

We add constraints to the policy improvement to make more effective use of the agent's successful experience in \mathcal{B}_e and avoid exploring farther unknown regions.

The following describes how the policy is updated. Estimation of the Q-value of the current state-action pair (s, a) by ensemble Q-functions.

$$Q_{\pi_{\theta}}(s, \hat{a}) = \frac{1}{N} \sum_{i=1}^N Q_{\phi_i}(s, \hat{a}) \quad (10)$$

where $\hat{a} = \pi_{\theta}(s)$.

The loss of the policy can be expressed as

$$\mathcal{L}^{policy} = \sum_{(s, a, r, s') \in \mathfrak{B}} -Q_{\pi_{\theta}}(s, \hat{a}) + \lambda(a - \hat{a})^2 \quad (11)$$

An et al. use the minimum of N Q-functions as the estimate of Q-value, which can achieve a pessimistic estimation of Q-value and reduce the estimation error. With the gradual increase of N ,

the difference between the gradients of Q-functions gradually increases, which is the fundamental reason for the improved effect of Q-Ensemble [1]. However, more than a pessimistic estimation of values in the online training phase can lead to a lack of exploration. Thus, as shown in equation eq15, the TD error is derived by randomly sampling a subset of Q-functions from the ensemble Q-functions and computing the minimum of the subset estimates. The trade-off between pessimistic estimation and exploration is made by introducing randomness.

In the policy improvement phase, ensemble Q-functions estimates' average value is used as the current estimated value. This is because the average value represents the intermediate level of the estimation error of the current state-action pair, which can effectively reduce the estimation error.

5 EXPERIMENT EVALUATION

We conducted extensive experiments aimed at answering the following questions.

- **Sample Efficiency:** Does A-SILfD have a higher sample efficiency than other methods (see Figure 5)?
- **Imperfect Expert Demonstrations:** Can A-SILfD perform better despite imperfect expert demonstrations (see Figure 5)?
- Is A-SILfD more advantageous than other Ensemble Q-functions based methods (see Table 3)?
- What is the impact of each part of A-SILfD on the overall performance (see Figure 6 and Figure 7)?

5.1 Setup

In our experiments, we chose four classic Mujoco control tasks: Ant-v2, HalfCheetah-v2, Hopper-v2 and Walker2d-v2, which can be used as representatives of robot control tasks.

For each task, our expert demonstrations come from the implementation provided by the OPOLO¹, and the imperfect expert demonstrations come from the implementation provided by the SAIL². All tasks were performed on three random seeds (10, 20, 30) using four trajectories with no more than 1000 steps.

Our method uses a single hidden layer feedforward network as the actor-network and critic-network (Q-function) on all tasks, with 256 neurons. Based on the ensemble learning, we have ten critic networks with the same structure and random initialization, respectively.

All experimental results are evaluated in the corresponding intensive reward environment provided by OpenAI Gym for the final performance. The horizontal coordinates of all figures are the number of steps that have interacted with the environment during training, and the vertical coordinates are the cumulative reward of the current evaluation. The figures allow us to see the sample efficiency of the different algorithms.

5.2 Sample Efficiency

Our baseline methods are shown in Table 2. All methods are off-policy, based on the TD3 [13] or the Soft Actor-Critic (SAC) algorithm [18], with high data utilization. In particular, REDQ does not

require expert demonstrations and is trained from scratch, achieving state-of-the-art results compared to similar methods.

Due to the limited availability of the LfD methods code, we cannot access all source codes. Moreover, many LfD methods are based on the idea of on-policy, which has low sample efficiency. So we implement REDQ-LfD based on the idea of LfD in combination with REDQ methods and compare them in Section 5.3.

Table 2: Comparison of baseline methods.

Algorithm	Types	Basic algorithm	Year
REDQ [8]	Scratch	TD3	2021
AWAC [35]	Offline to Online RL	SAC	2020
IQL [28]	Offline to Online RL	SAC	2022
SAIL [56]	Imitation learning	TD3	2022
OPOLO [55]	Imitation learning	TD3	2020
DAC [26]	Imitation learning	TD3	2019

REDQ-TD3 is only compared to A-SILfD alone because it does not use expert demonstrations, reflecting the higher sample efficiency that A-SILfD can achieve with a small number of expert demonstrations. Figure 5 shows the performance of all methods on Mujoco's four tasks, in which we experimented with expert demonstrations, mixed expert demonstrations, and imperfect expert demonstrations, respectively. In all experiments, our method was not affected by the quality of the expert demonstrations and had a higher sample efficiency.

5.2.1 Expert demonstrations (Figure 5, first row). This part of the experiment uses four expert trajectories. Our method outperforms the baseline methods in all four tasks. AWAC and IQL have a higher initial cumulative reward due to the initial pre-training phase. Affected by the distribution shift, AWAC suffered from a sudden performance degradation at the beginning. The strict policy constraints and the small amount of offline data lead to poor results of the IQL. At the same time, A-SILfD can avoid performance degradation at the initial moment due to the distribution shift. SAIL applies to the case of imperfect expert demonstrations, which perform poorly in this case. Although DAC and OPOLO do not use environmental rewards, the sample efficiency is lower, and the training process is volatile.

5.2.2 Mixed expert demonstrations (Figure 5, second row). This part of the experiment uses two expert trajectories and two imperfect expert trajectories. A-SILfD is ahead of the baseline method on all tasks except the Hopper-v2 task. The IL methods OPOLO and DAC can be misled by imperfect expert demonstrations, leading to poor performance. SAIL does not distinguish well between different quality demonstrations. AWAC and IQL perform better than Section 5.2.1 because demonstrations of varying quality can occupy a larger region of the state space, resulting in better initial results for the Q-function.

5.2.3 Imperfect Expert demonstrations (Figure 5, third row). This part of the experiment uses four imperfect expert trajectories. A-SILfD effectively uses imperfect demonstrations. The IL methods OPOLO and DAC fail to learn effective control policy on Hopper-v2,

¹<https://github.com/illidanlab/opolo-code>

²<https://github.com/illidanlab/SAIL>

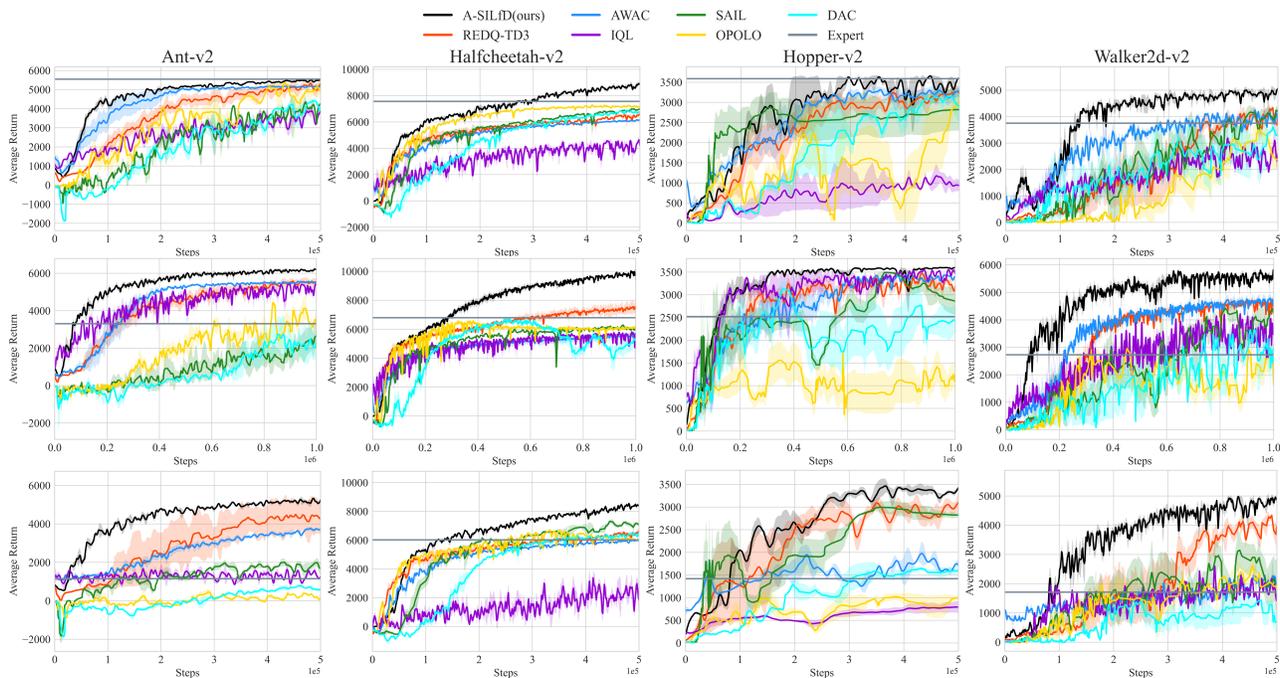


Figure 5: Results for the Mujoco tasks. We conducted experiments on four Mujoco tasks using expert demonstrations (first row), mixed expert and imperfect expert demonstrations (second row), and imperfect expert demonstrations (third row), respectively. The solid lines and shaded regions represent the mean and standard deviation across three runs.

Ant-V2 and Walker-V2 due to the limitation of the quality of the expert demonstrations. While SAIL can outperform imperfect expert demonstrations, it cannot achieve better performance. AWAC and IQL are only close to the performance of A-SILfD on Hopper-v2 and underperform on other tasks. IQL has no significant performance improvement in the online training. This experiment shows that A-SILfD can better use imperfect expert demonstrations to improve sample efficiency and avoid imperfect data misleading policy improvement.

5.3 Comparison of Methods based on Ensembles Q-functions

In this section, we discuss the effect of ensemble Q-functions. Based on the idea of ensemble Q-functions, we implemented the following method to enhance sample efficiency using expert demonstrations.

- **REDQ-LfD:** Due to the limited availability of the LfD methods code, we implemented REDQ-LfD based on the LfD idea. We use the offline data for pre-training before the REDQ-TD3 training and still use the offline data for the subsequent online training.
- **REDQ-BC:** We use BC combined with expert demonstrations to initialize the policy and then train the policy online.

Table 3 shows the results of our experiments. Evaluated Performance indicates the ratio of the cumulative reward to the basic reward at the time of evaluation at 500,000 online interaction steps. Surpass the basic reward indicates the number of interactions required to exceed the basic reward smoothly. On the four Mujoco

Table 3: Comparison of the method based on the ensemble Q-functions.

Benchmark	Ant	HalfCheetah	Hopper	Walker2d
(S, \mathcal{A})	(111,8)	(17,6)	(11,3)	(17,6)
Basic Reward	4600	6000	3500	3400
	Evaluated Performance/Surpass the Basic Reward			
REDQ	1.17/340k	1.13/320k	0.95/No	1.09/410k
REDQ-BC	1.20/380k	1.25/180k	1.01/420k	1.20/260k
REDQ-LfD	1.22/210k	1.46/120k	1.02/240k	1.37/200k
Ours	1.24/180k	1.48/120k	1.05/180k	1.43/130k

tasks, the ideas of BC and LfD combined with Ensemble Q-functions can improve the sample efficiency. However, A-SILfD requires fewer steps and has higher sample efficiency than the other methods. In addition, the A-SILfD algorithm still achieves better performance after 500,000 step iterations. The results show that in addition to the idea of ensemble Q-functions, other parts of A-SILfD can effectively improve the algorithm’s performance.

5.4 Ablation Experiment

5.4.1 The Importance of Policy Constraints. As shown in Figure 6, the red curve indicates the A-SILfD’s performance without policy

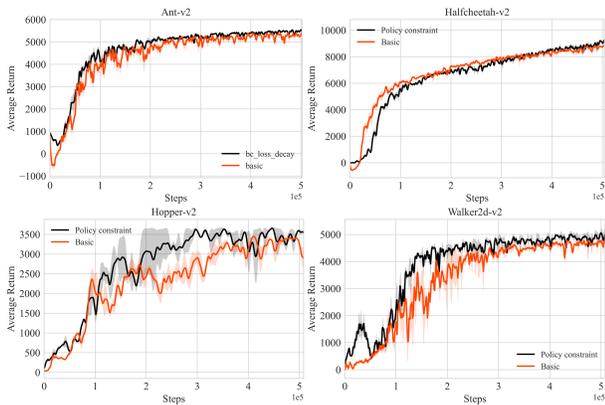


Figure 6: The importance of policy constraints. Basic means no policy constraints is added.

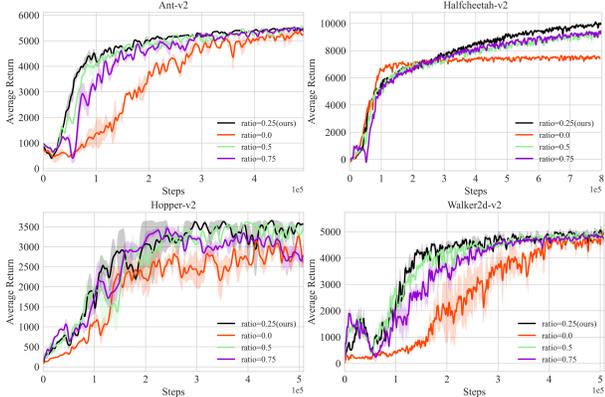


Figure 7: Comparison of different $ratio$ on the four Mujoco tasks.

constraints, and the black curve indicates the A-SILfD’s performance after adding policy constraints. The training data of each mini-batch contains the agent’s successful experience and other sampled data. Therefore, the results show that the policy constraint term has the following two effects. (1) Make the policy try to mimic the agent’s successful experience so that the algorithm performance reaches the expert’s performance more quickly. (2) Constrain the exploration region of the policy to make the algorithm representation more stable. The experimental results show that the policy constraints can effectively improve the performance of A-SILfD.

5.4.2 Effect of Sampling Ratio. When sampling the training data, the data in the sample replay buffer and the experience replay buffer are sampled proportionally. $ratio$ indicates the percentage of data from the experience replay buffer for each mini-batch. We consider four settings. $ratio \in \{0.25, 0.5, 0.75\}$ means that each mini-batch samples a different proportion of experience data, while $ratio = 0.0$ means that the data in the experience replay buffer is not used. As shown in Figure 7, we can see that $ratio = 0.0$ is ineffective, showing the importance of expert demonstration and the agent’s successful experience. The $ratio = 0.75$ means that each mini-batch

uses more of the agent’s successful experience, and the training data lacks exploration, resulting in poor performance. $ratio = 0.25$ considers both the imitation of the agent’s successful experience and environmental exploration, and we finally choose $ratio = 0.25$.

6 CONCLUSION

In this paper, we discuss how to effectively use fewer expert demonstrations to improve the sample efficiency of DRL. Considering that the expert demonstrations may be small and imperfect, the existing methods cannot use them effectively. Therefore, we propose an Actor-Critic framework-based LfD method named A-SILfD. A-SILfD stores expert demonstrations as the agent’s successful experience in the experience replay buffer. During the training process, A-SILfD evaluates the quality of the trajectory and dynamically adjusts the data in the experience replay buffer. Our experimental results in the Mujoco control task show that.

- Our method can improve sample efficiency using different quality expert demonstrations.
- Through effective self-imitation learning, we can fully use the agent’s successful experience to learn the policy so that the policy can eventually achieve higher performance.
- By ensemble Q-functions, our method makes the policy training process smoother and avoids the performance degradation caused by distribution shift.

Although our method could significantly speed up the training process using fewer expert demonstrations, it encountered some limitations. First, there is a challenge in designing the weights for the policy constraints; we set the initial weight to 1 and decay it with the training process. However, the weights must be adjusted separately for different environments to achieve higher performance. Second, although the experimental results surface that the ensemble Q-functions can reduce the estimation error due to distribution shift, we do not provide detailed proof of this point. Finally, since our method relies on the reward function to evaluate the trajectory quality and uses the TD error learning Q-function, it may not work well in a sparse reward setting.

REFERENCES

- [1] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. 2021. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems* 34 (2021), 7436–7447.
- [2] Alex M Andrew. 1998. Reinforcement Learning: An Introduction. *Kybernetes* (1998).
- [3] OpenAI: Marcin Andrychowicz and Baker. 2020. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research* 39, 1 (2020), 3–20.
- [4] David A. Anisi. 2003. *Optimal Motion Control of a Ground Vehicle*. Master’s thesis. Royal Institute of Technology (KTH), Stockholm, Sweden.
- [5] Sam Anzaroot and Andrew McCallum. 2013. *UMass Citation Field Extraction Dataset*. University of Massachusetts. Retrieved May 27, 2019 from <http://www.iesl.cs.umass.edu/data/data-umasscitationfield>
- [6] Michael Bain and Claude Sammut. 1995. A Framework for Behavioural Cloning. In *Machine Intelligence 15*. Oxford University Press, 103–129.
- [7] Kianté Brantley, Wen Sun, and Mikael Henaff. 2020. Disagreement-Regularized Imitation Learning. In *ICLR*. OpenReview.net.
- [8] Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. 2021. Randomized Ensembled Double Q-Learning: Learning Fast Without a Model. In *ICLR*. OpenReview.net.
- [9] Kenneth L. Clarkson. 1985. *Algorithms for Closest-Point Problems (Computational Geometry)*. Ph.D. Dissertation. Stanford University, Palo Alto, CA. UMI Order Number: AAT 8506171.
- [10] Gabriel V Cruz Jr, Yunshu Du, and Matthew E Taylor. 2017. Pre-training neural networks with human demonstrations for deep reinforcement learning. *arXiv preprint arXiv:1709.04083* (2017).
- [11] Justin Fu, Katie Luo, and Sergey Levine. 2017. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248* (2017).
- [12] Scott Fujimoto and Shixiang Shane Gu. 2021. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems* 34 (2021), 20132–20145.
- [13] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*. PMLR, 1587–1596.
- [14] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-Policy Deep Reinforcement Learning without Exploration. In *ICML (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 2052–2062.
- [15] Seyed Kamyar Seyed Ghasemipour, Richard S. Zemel, and Shixiang Gu. 2019. A Divergence Minimization Perspective on Imitation Learning Methods. In *CoRL (Proceedings of Machine Learning Research, Vol. 100)*. PMLR, 1259–1277.
- [16] Barbara J. Grosz and Sarit Kraus. 1996. Collaborative Plans for Complex Group Action. *Artificial Intelligence* 86, 2 (1996), 269–357.
- [17] Yijie Guo, Junhyuk Oh, Satinder Singh, and Honglak Lee. 2018. Generative adversarial self-imitation learning. *arXiv preprint arXiv:1812.00950* (2018).
- [18] Tuomas Haaroja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.
- [19] Torben Hagerup, Kurt Mehlhorn, and J. Ian Munro. 1993. Maintaining Discrete Probability Distributions Optimally. In *Proceedings of the 20th International Colloquium on Automata, Languages and Programming (Lecture Notes in Computer Science, Vol. 700)*. Springer-Verlag, Berlin, 253–264.
- [20] David Harel. 1978. *Logics of programs: axiomatizations and descriptive power*. MIT Research Lab Technical Report TR-200. Massachusetts Institute of Technology, Cambridge, MA.
- [21] Todd Hester and Vecerik. 2018. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
- [22] Jonathan Ho and Stefano Ermon. 2016. Generative Adversarial Imitation Learning. In *NIPS*. 4565–4573.
- [23] Yandong Ji, Zhongyu Li, Yinan Sun, Xue Bin Peng, Sergey Levine, Glen Berseth, and Koushil Sreenath. 2022. Hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot. *arXiv preprint arXiv:2208.01160* (2022).
- [24] Bingyi Kang, Zequn Jie, and Jiashi Feng. 2018. Policy optimization with demonstrations. In *International conference on machine learning*. PMLR, 2469–2478.
- [25] Donald E. Knuth. 1997. *The Art of Computer Programming, Vol. 1: Fundamental Algorithms* (3rd ed.). Addison Wesley, Reading, Massachusetts.
- [26] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. 2019. Discriminator-Actor-Critic: Addressing Sample Inefficiency and Reward Bias in Adversarial Imitation Learning. In *ICLR (Poster)*. OpenReview.net.
- [27] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2021. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169* (2021).
- [28] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2022. Offline Reinforcement Learning with Implicit Q-Learning. In *ICLR*. OpenReview.net.
- [29] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. 2019. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. In *NeurIPS*. 11761–11771.
- [30] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 1179–1191.
- [31] Leslie Lamport. 1994. *TEX: A Document Preparation System* (2nd ed.). Addison-Wesley, Reading, MA.
- [32] Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. 2021. Offline-to-Online Reinforcement Learning via Balanced Replay and Pessimistic Q-Ensemble. In *CoRL (Proceedings of Machine Learning Research, Vol. 164)*. PMLR, 1702–1712.
- [33] Yihuan Mao, Chao Wang, Bin Wang, and Chongjie Zhang. 2022. MOORe: Model-based Offline-to-Online Reinforcement Learning. *arXiv preprint arXiv:2201.10070* (2022).
- [34] Volodymyr Mnih and Kavukcuoglu. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.
- [35] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. 2020. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359* (2020).
- [36] Tianwei Ni, Harshit S. Sikchi, Yufei Wang, Tejus Gupta, Lisa Lee, and Ben Eysenbach. 2020. f-IRL: Inverse Reinforcement Learning via State Marginal Matching. In *CoRL (Proceedings of Machine Learning Research, Vol. 155)*. PMLR, 529–551.
- [37] Barack Obama. 2008. A More Perfect Union. Video. Retrieved March 21, 2008 from <http://video.google.com/videoplay?docid=6528042696351994555>
- [38] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. 2018. Self-imitation learning. In *International Conference on Machine Learning*. PMLR, 3878–3887.
- [39] Georgiy Pshikhachev, Dmitry Ivanov, Vladimir Egorov, and Aleksei Shpilman. 2022. Self-Imitation Learning from Demonstrations. *arXiv preprint arXiv:2203.10905* (2022).
- [40] Jorge Ramirez, Wen Yu, and Adolfo Perrusquia. 2022. Model-free reinforcement learning from expert demonstrations: a survey. *Artificial Intelligence Review* 55, 4 (2022), 3213–3241.
- [41] Siddharth Reddy, Anca D. Dragan, and Sergey Levine. 2020. SQIL: Imitation Learning via Reinforcement Learning with Sparse Rewards. In *ICLR*. OpenReview.net.
- [42] Desik Rengarajan, Gargi Vaidya, Akshay Sarvesh, Dileep M. Kalathil, and Srinivas Shakkottai. 2022. Reinforcement Learning with Sparse Rewards using Guidance from Offline Demonstration. In *ICLR*. OpenReview.net.
- [43] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. PMLR, 1889–1897.
- [44] Joseph Scientist. 2009. The fountain of youth. Patent No. 12345, Filed July 1st., 2008, Issued Aug. 9th., 2009.
- [45] David Silver and Huang. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484–489.
- [46] Yunhao Tang. 2020. Self-imitation learning via generalized lower bound q-learning. *Advances in neural information processing systems* 33 (2020), 13964–13975.
- [47] Mel Vecerik and Hester. 2017. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817* (2017).
- [48] Yunke Wang, Chang Xu, Bo Du, and Honglak Lee. 2021. Learning to Weight Imperfect Demonstrations. In *International Conference on Machine Learning*. PMLR, 10961–10970.
- [49] JT-Y Wen and Kenneth Kreutz-Delgado. 1991. The attitude control problem. *IEEE Transactions on Automatic Control* 36, 10 (1991), 1148–1162.
- [50] Michael J. Wooldridge and Nicholas R. Jennings. 1995. Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review* 10, 2 (1995), 115–152.
- [51] Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. 2019. Imitation learning from imperfect demonstration. In *International Conference on Machine Learning*. PMLR, 6818–6827.
- [52] Deheng Ye, Guibin Chen, Peilin Zhao, Fuhao Qiu, Bo Yuan, Wen Zhang, Sheng Chen, Mingfei Sun, Xiaoqian Li, Siqin Li, et al. 2020. Supervised learning achieves human-level performance in moba games: A case study of honor of kings. *IEEE Transactions on Neural Networks and Learning Systems* (2020).
- [53] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. 2020. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 737–744.
- [54] Yi Zhao, Rinu Boney, Alexander Ilin, Juho Kannala, and Joni Pajarinen. 2021. Adaptive Behavior Cloning Regularization for Stable Offline-to-Online Reinforcement Learning. (2021).
- [55] Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. 2020. Off-policy imitation learning from observations. *Advances in Neural Information Processing Systems* 33 (2020), 12402–12413.
- [56] Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. 2022. Self-Adaptive Imitation Learning: Learning Tasks with Delayed Rewards from Sub-Optimal Demonstrations. (2022).