Neurocomputing 467 (2022) 300–309

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

SADRL: Merging human experience with machine intelligence via supervised assisted deep reinforcement learning

Xiaoshuang Li [a,b], Xiao Wang [a,c], Xinhu Zheng [d], Junchen Jin [e], Yanhao Huang [f], Jun Jason Zhang [a,g], Fei-Yue Wang [a,c,*]

[a] *The State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China*
[b] *The School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China*
[c] *The Parallel Intelligence Research Center, Qingdao Academy of Intelligent Industries, Qingdao 266109, China*
[d] *The Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, USA*
[e] *PCITECH, PCI Intelligent Building, No.2 Xincen Fourth Road, Tianhe District, Guangzhou 510653, China*
[f] *The State Key Laboratory of Power Grid Safety and Energy Conservation, China Electric Power Research Institute, Beijing 100192, China*
[g] *The School of Electrical Engineering and Automation, Wuhan University, Wuhan 430072, China*

ARTICLE INFO

Article history:
Received 11 March 2021
Revised 6 August 2021
Accepted 25 September 2021
Available online 4 October 2021
Communicated by Zidong Wang

Keywords:
Deep reinforcement learning
Behavioral cloning
Dynamic demonstration
Double DQN

ABSTRACT

Deep Reinforcement Learning (DRL) has proven its capability to learn optimal policies in decision-making problems by directly interacting with environments. Meanwhile, supervised learning methods also show great capability of learning from data. However, how to combine DRL with supervised learning and leverage additional knowledge and data to assist the DRL agent remains difficult. This study proposes a novel Supervised Assisted Deep Reinforcement Learning (SADRL) framework integrating deep Q-learning from dynamic demonstrations with a behavioral cloning model (DQfDD-BC). Specifically, the proposed DQfDD-BC method leverages historical demonstrations to pre-train a behavioral cloning model and consistently update it by learning the dynamically updated demonstrations. A supervised expert loss function is designed to compare actions generated by the DRL model with those obtained from the BC model to provide advantageous guidance for policy improvements. Experimental results in several OpenAI Gym environments show that the proposed approach accelerates the learning processes, and meanwhile, adapts to different performance levels of demonstrations. As illustrated in an ablation study, the dynamic demonstration and expert loss mechanisms using a BC model contribute to improving the learning convergence performance compared with the baseline models. We believe that SADRL provides an elegant framework and the proposed method can promote the integration of human experience and machine intelligence.

1. Introduction

Deep reinforcement learning (DRL) has made great milestones in several applications such as the Go [1,2], Atari games [3] and robot skill acquisition [4] in recent years. However, DRL-based approaches, such as Deep Q-learning Network (DQN), usually fail in many real-world scenarios [5–7]. The gap between the real-world and the simulation environment prevents the wide application of the DRL methods. Due to the diversity and uncertainty of complex systems and real-world situations, it is a big challenge to establish a simulation environment for implementing DRL models, which is consistent with the real-world systems. Technically, the DRL agent can not obtain sufficient experience to improve the strategies without a reliable and informative simulation environment. The inability to fully interact with the real world restrains the agents from learning some valuable policies that can be directly applied to real-world scenarios. During the learning process, the DRL model may choose a random action sampled from a random policy [8], which is essential for the agent to explore the entire state-action space. However, the random actions are restricted in many real-world circumstances because of the possibility of causing severe damage to the system. For instance, in autonomous driving scenarios [9,10], a random policy may lead to traffic congestion or even road accidents. These prohibited dangerous actions make it impossible for the agent to accumulate the subsequent states or learn how to avoid potentially serious consequences. To ensure the reliability of the agent, shifting from limited

* Corresponding author at: The State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China.
E-mail address: feiyue.wang@ia.ac.cn (F.-Y. Wang).

https://doi.org/10.1016/j.neucom.2021.09.064
0925-2312/© 2021 Elsevier B.V. All rights reserved.

simulation environments to mimic real-world ones becomes one of the most urgent tasks for applying DRL models for complicated decision-making tasks.

Human experts may have significant advantages in learning efficiency and decision-making performance [11] in complex real-world circumstances. Recent studies show that incorporating human experience is a potential solution to enhance the adaptability of DRL models for complex tasks [12,13]. Although it is difficult to model and learn that expert knowledge, many supervised learning methods based on deep neural networks show powerful capabilities of feature extraction and modeling of various data, including expert decisions extracted from trajectory data and human experience, which are also known as demonstrations [14–16]. To be specific, deep Q-learning from demonstrations (DQfD) is a typical algorithm that succeeded in integrating DRL with demonstrations [12], which combines the temporal difference (TD) error in the traditional Double DQN (DDQN) algorithm [17] with supervised expert loss by constructing a hybrid loss function. Through a specially designed large margin supervised loss function [18,19], the DQfD method can guide and assist an agent in learning the expert's knowledge by constantly steering the agent learning policies closer to those represented by the demonstration.

However, the DQfD model suffers from three major issues: (1) In the learning process, the transition trajectory in the historical demonstration dataset is the single data source for contributing expert loss, which does not include the self-generated transitions of the trained agents. As a result, the DQfD algorithm merely relies on TD errors to improve the policy but the demonstration is idle when the self-generated transitions are sampled from the experience replay buffer, which reduces the efficiency of utilizing demonstrations. (2) According to the learning mechanism, static demonstrations are not enough to cover sufficient state-action space during the agent training process, especially when it is difficult or expensive to collect demonstrations in real-world applications. Also, with a constant stream of self-generated transition samples added into the experience replay buffer, historical demonstrations would make smaller and smaller contributions to the policy improvement as their sampling probability gradually decreases. (3) The DQfD algorithm requires the learned policy to approximate the demonstration but ignores the negative influence of imperfect demonstrations, which existing commonly in real-world applications. Instead of providing appropriate guidance like a perfect demonstration, the imperfect demonstration is detrimental to the learned policies when the demonstration represents a policy inferior to the learned ones.

In this paper, a novel supervised assisted deep reinforcement learning (SADRL) framework is proposed. In addition, a specific method deep Q-learning from dynamic demonstrations with a behavioral cloning model (DQfDD-BC) is developed. In general, the main contributions are summarized as follows:

1 A hybrid framework SADRL is proposed to merge human experience and machine intelligence and the supervised learning method is used to assist DRL.
2 A specific method DQfDD-BC that assists the DRL by leveraging behavioral cloning (BC) [20,21]. Experimental results on OpenAI Gym show that the proposed method has a better performance compared with the baseline models.
3 An automatic update mechanism that adaptively enhances the demonstration and the BC model, which can include more high-quality transition samples to improve the demonstration and mitigate potential pernicious influences caused by imperfect demonstration.

The remainder of this paper is organized as follows: In Section 2, the related work in imitation learning methods and some studies which leveraged the demonstration to improve the DRL methods are discussed. The problem definition is presented in Section 3 and the details of the method proposed in this paper are demonstrated in Section 4. In Section 5, some experiments and analyses are conducted to validate the proposed method. Conclusions are summarized in Section 6, with some future research directions.

2. Related works

Generally, demonstration comes from the human or other kinds of "experts" who can provide valuable information [22] for different hard problems, such as robot control [14,23], self-driving [24], etc [25,26]. Imitation learning [27,28] methods can exploit the demonstration to mimic expert actions and behavioral cloning [20], as a kind of imitation learning, has received extensive attention [21,29,30] due to its significant advantages, such as fast learning speed, simplicity, and effective utilization of demonstration, etc. The BC method constructs mappings from states to actions (or distributions of actions) in a supervised manner to model the policies represented by the demonstrations and achieves the goal of learning from a demonstration by minimizing various supervised losses. In a study by NVIDIA, the model obtained impressive results by minimizing the mean squared error between the steering command output by the network and the command of a human driver [24]. *Chauffeurnet* further enhances the performance of imitation learning in autopilot by synthesizing the worst scenarios [31].

DAgger [32] is proposed to cope with the covariate shift problem of BC method. Experts are required by the DAgger method to respond to self-generated transitions so that totally new and valuable transitions can be added to the demonstration. These new transitions can expand the state-action space covered by the demonstration. Thus the goal of performance improvement can be achieved during the learning process. However, the DAgger method requires an always-available expert to assist in labeling the data, which reduces the practicality of the method. Deeply AggreVaTeD [33] enables the DAgger to handle continuous action spaces by using deep neural networks but the weakness of DAgger was preserved.

Another typical class of imitation learning algorithms is generative adversarial imitation learning (GAIL) [34–36]. Instead of mapping from state to action directly, GAIL learns from the demonstrations by introducing the generator and discriminator. The generator is used to learn the policy contained in the demonstration and generate a state-action pair, while the discriminator is trained to distinguish whether the state-action pair is from the expert or the learned policy. The goal of policy improvement is achieved through the adversarial learning process. GAIL shows good performance on the high-dimensional continuous control problem [37,38]. Policy optimization with demonstrations (POfD) [37] utilizes the demonstration by an adversarial learning method and has made efforts on sparse and dense reward environments.

Recently, demonstrations have been leveraged to tutor DRL to achieve better performance. Since the DRL reward is immediate feedback from the environment for evaluating the performance of the policy, reshaping the reward function through the demonstration is an effective approach to improve DRL performance [39–41]. By transforming the form of the reward function, the difficulty of agent training has been reduced. Model-Based DRL (MBRL) has shown the capability in handling some difficult tasks [42] and demonstration is also leveraged to deal with the MBRL problems [43,44].

All of the previously mentioned approaches yielded impressive results but most of them utilized demonstrations through reward shaping or imitating the demonstration. An alternative approach is to improve the policy directly from demonstrations. The samples in the experience replay buffer and demonstrations are both extracted by DRL agents [12,45,5]. These approaches put the demonstration into the experience replay buffer and sample from the hybrid experience replay buffer. The DQfD algorithm enhances the DDQN [17] algorithm by keeping the demonstration in the replay buffer at all times. A designed, supervised expert loss is generated when the transitions in the demonstration are sampled to update the parameters of the neural networks. The self-imitation learning (SIL) method chooses different loss functions and adds new trajectories to the demonstration. It updates the same neural network twice with the A2C loss function and the SIL loss function [45]. The DDPGfD algorithm [46] is similar to the DQfD and they both assist in enhancing the original algorithms through a hybrid loss function with an expert loss. The difference between these two methods is that the DDPGfD algorithm is based on the DDPG algorithm [47], which is designed to solve problems with continuous action space and the DQfD is more suitable for the discrete action space. However, both DQfD and DDPGfD suffer from the problem of under-exploiting demonstration data [37].

The proposed DQfDD-BC method is designed to make up for the shortcomings of the above methods. The supervised learning model and expert loss can fully extract the demonstration and mimic the expert policy. The constant renewal of demonstration is beneficial to overcome the negative effects of static demonstration. To our best known, this is the first time the imitation learning model has been introduced to the DRL and the combination of these two function modules results in some new and comprehensive progress.

3. Preliminaries

3.1. Q-function and DQN

Standard Deep Q-Networks (DQN) is designed to deal with the standard Markov Decision Process (MDP) [48], which is defined by a tuple $\langle \mathscr{S}, \mathscr{A}, \mathscr{P}, r, \gamma \rangle$ where $\mathscr{S}, \mathscr{A}, \mathscr{P}, r, \gamma$ represent the state space, action space, state transition distribution, reward function and discount factor, respectively. The transition distribution $\mathscr{P}(s'|s,a)$ describes the process of taking action a at state s, s' is the next state and reward function $r(s,a)$ gives the feedback. A policy $\pi(a|s)$ specifies how the agent responds to various states. The goal of the agent is to find the policy π which maps states to actions that maximize the expected discounted total reward $\mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right]$ over the agent's lifetime. $Q^\pi(s,a)$ is an estimate of the expected future reward that can be obtained from (s,a) by the agent with policy π. The optimal value function $Q^*(s,a)$ provides maximal action values in all states and is determined by solving the Bellman equation, shown as Eq. (1).

$$Q^*(s,a) = \mathbb{E}\left[r(s,a) + \gamma\sum_{s'} P(s'|s,a)\max_{a'} Q^*(s',a')\right] \tag{1}$$

Deep Q learning is one of the most famous deep reinforcement learning methods. It leverages deep neural networks to approximate the value function and outputs the action value $Q(s,a)$ for each state-action pair (s,a). Transitions are sampled uniformly from the experience replay buffer to update the parameters of the neural networks by minimizing the loss function, denoted by Eq. (2). DQN uses two neural networks to improve the policy learned by the agent. The current network, θ, is used to calculate the value function and the target network, represented by θ', copy the parameters of the current network every k steps to stable the training process.

$$\mathscr{L}_Q(Q) = \left(r(s,a) + \gamma\max_a Q(s_{t+1},a;\theta') - Q(s,a;\theta)\right)^2 \tag{2}$$

3.2. DDQN and DQfD

To avoid the issue of over-estimation for the action value in DQN, double DQN (DDQN) uses the current Q network to choose the best action by $a_{t+1}^{\max} = \operatorname{argmax}_a Q(s_{t+1},a;\theta)$ instead of the optimal action value Q^*. Similarly, as in DQN, the target Q network is utilized to calculate the TD target, but it uses the best action instead of searching for the optimal action value. Thus, the loss function of DDQN can be written as Eq. (3).

$$\mathscr{L}_{DQ}(Q) = \left(r(s,a) + \gamma Q\left(s_{t+1},a_{t+1}^{\max};\theta'\right) - Q(s,a;\theta)\right)^2 \tag{3}$$

Based on DDQN, the DQfD algorithm proposes a hybrid loss function with supervised loss. In DQfD, the complete loss function is denoted by Eq. (4) which consists of four parts: the double Q learning TD loss, n-step double Q learning TD loss, expert loss and L2 regularization loss, denoted by $\mathscr{L}_{DQ}(Q), \mathscr{L}_n(Q), \mathscr{L}_E(Q)$ and $\mathscr{L}_{L2}(Q)$, respectively. The parameters $\lambda_1, \lambda_2, \lambda_3$ adjusts the weights of different losses.

$$\mathscr{L}(Q) = \mathscr{L}_{DQ}(Q) + \lambda_1\mathscr{L}_n(Q) + \lambda_2\mathscr{L}_E(Q) + \lambda_3\mathscr{L}_{L2}(Q) \tag{4}$$

The expert loss $\mathscr{L}_E(Q)$ is considered as the most significant loss among the losses in DQfD, expressed in Eq. (5). It allows the model to satisfy the Bellman equation and also enables the agent to learn from the demonstrations directly. The large margin supervised loss $\ell(a_E,a)$ is set to 0 when $a = a_E$ where a_E denotes the expert action in the demonstration, and a positive constant otherwise. This loss forces the action values of the other actions to be at least a margin lower than the value of the demonstrator's action [12]. Note that, when the sampled transitions are not in the demonstration, $\lambda_2 = 0$ means that the expert loss $\mathscr{L}_E(Q)$ is non-functional.

$$\mathscr{L}_E(Q) = \max_{a\in A}[Q(s,a) + \ell(a_E,a)] - Q(s,a_E) \tag{5}$$

In this paper, the proposed DQfDD-BC method substitutes the large margin supervised loss with a cross-entropy loss and the expert loss is always available for the complete loss function during the self-learning process via imitation learning model (see details in Section 4.3).

4. Methodology

In this section, the overall structure of SADRL and two major modules of the proposed method are introduced, which are utilizing dynamic demonstration and obtaining a better supervised expert loss via behavioral cloning model. Last but not least, the training process and the pseudo-code are presented.

4.1. Framework of SADRL

The fusion of both supervised learning and reinforcement learning is an important advantage of SADRL. Supervised learning methods are good at extracting knowledge from vast amounts of data, which facilitates the utilization of supervised learning to learn a decision policy from the expert experience and offline operation records. However, supervised learning is limited by the data itself and cannot discover new knowledge. On the contrary, DRL realizes the full potential of machine intelligence and the DRL agent constantly learns new knowledge from the environment. However, the learning speed of DRL is relatively slow and it is expensive to obtain a perfect result. The SADRL intends to mitigate their disadvantages while retaining their strengths and realizes a new learning paradigm of human–machine integration.

Fig. 1 illustrates the technical architecture of the proposed DQfDD-BC approach under the SADRL framework. DQfDD-BC shares the same basic components of the DDQN algorithm [17], including two Q networks and experience relay buffer where alterable demonstrations are added for further usage. An additional BC module is introduced to utilize the demonstration in the replay buffer. In addition, the loss value is calculated considering both the output of the trained Q network and the BC model.

Firstly, the DQfDD-BC method attempts to extract the experts' policies from the initial demonstration and allows the agent to provide reasonable actions when facing self-generated states. During the self-learning process, BC model is introduced to generate expert loss to utilize all transitions in the experience replay buffer. The agent's actions are compared with those generated by the BC model through a supervised expert loss function. The inclusion of the BC model allows the knowledge in the demonstrations to be sufficiently utilized in the training process and enables the model to cope with the states which experts have not ever encountered. The supervised learning process and self-learning process can promote each other. The supervised model provides a baseline to guide the learning process and the self-learning process keeps improving and generates new demonstration samples to reduce the limitation of the BC model and suboptimal samples. In particular, new transitions are added to the demonstration if the model achieves a relatively high-performance episode score. Hence, the BC model can be improved after the update of the demonstration. There are two main differences between the proposed DQfDD-BC method and previous methods that also combine DRL with supervised learning. Firstly, existing methods that combined DRL and supervised learning make use of supervised learning techniques from the perspective of data. Specifically, only a supervised loss function is designed to utilize the demonstration. And the supervised loss is not available when the self-generated transitions are sampled. In the proposed DQfDD-BC method, we construct a BC model to imitate the demonstration so that the policy behind the demonstration can be used all the time once the pre-training process is completed. When a new state appears, the previous method is not able to make use of the demonstration to facilitate the learning process, while our method can provide a demonstration-like decision policy based on the pre-trained BC model and generate supervised loss to guide the self-learning process. Secondly, the demonstration in previous methods is static and it is dynamically updated in the DQfDD-BC. The demonstration is of great significance for the supervised learning process and dynamic data can effectively expand the original dataset to avoid overfitting and improve the generalization ability of the model.

4.2. Dynamic demonstrations

As the transition quality of historical demonstrations is usually much higher than that associated with random policies, DQfD chooses to learn its initial knowledge from preset demonstrations instead of random interaction trajectories. One of the major improvements of the proposed DQfDD-BC method is adding the newly generated interactions into the demonstrations. These newly generated transitions, who have a higher performance score for the same task, are collected during the training process. Particularly, after the early stage of the training process, the transition samples are automatically generated and inserted into the demonstration dataset when the final cumulative reward of each episode reaches a new high score. After the performance of the model has been improved effectively, the new and better transition samples are added to the demonstration. Thus, the quality of the demonstration is continuously improved while attempting to cover the full state-action space. Such a mechanism can continuously improve the diversity and quantity of the demonstration and continuously generate a positive effect on the imitation learning model performance.

The concept of the proposed approach is close to the DAgger algorithm, especially for improving the decision-making capability by adding new transitions to the demonstration and utilizing the added data to optimize the model parameters. However, unlike DAgger, the proposed method does not rely on manual labeling and automatically determines whether new transitions need to be added to the demonstration, which can significantly reduce the computational cost of the training process and improves the applicability of the method.

4.3. Expert loss with supervised BC model

The DQfD method leverages a binary supervised large margin loss function $\ell(a_E, a)$ to compare the generated actions with those in demonstrations under the same environment states. However, the large margin loss function is not able to distinguish the difference of various actions and the binary loss values are prone to cause a volatile gradient which all lead to the instability of the training process. Instead of utilizing the demonstration directly, a BC model is used to extract useful information from the demonstration.

In specific, the proposed DQfDD-BC method contains a deep neural network-based BC model and maintains two different experience replay buffers: D^{replay} and D^{demo}. In particular, D^{replay} refers to the common experience replay buffer in a DRL model and D^{demo} consists of both historical and self-generated demonstrations. The BC model is first pre-trained with demonstrations D^{demo} to obtain its initial decision-making ability. Prioritized experience replay mechanism [49] is also applied to both of the two replay buffers to improve the sample efficiency. The performance of the DRL agent is improved continuously during the learning process. Hence, the newly generated transition are given the highest priority to ensure that transitions with higher performance are extracted timely.

The DQfDD-BC model takes full advantage of the demonstration by generating expert losses for all self-generated transitions instead of directly using historical demonstrations. And it is designed for discrete action spaces while the additional BC model is considered to solve a multi-label classification problem. Hence, instead of the binary supervised large margin loss function, the cross-entropy loss function, $H(output, target)$, is selected to evaluate the difference between different actions. In the pre-train process, the BC model is trained by minimizing the cross-entropy loss between a_E and $\pi_{bc}(s_t)$ (Eq. (6)) to learn the demonstration and construct the BC model.

$$\ell_{BC} = H(\pi_{bc}(s_t), a_E) \tag{6}$$

where a_E refers to the expert action in the demonstration and π_{bc} represents the learned policy of the BC model. Once the pre-training of the BC model is completed, π_{bc} can always provide relatively reasonable expert actions for each state.

$$\ell(a, \pi_{bc}(s)) = H(a, \pi_{bc}(s)) \tag{7}$$

In the self-learning stage, the cross-entropy loss (Eq. (7)) evaluates the difference between a and $\pi_{bc}(s)$, where a is the action provided by the Q network and $\pi_{bc}(s)$ represents the action generated by the latest state of BC model. The supervised loss $\ell(a, \pi_{bc}(s))$ can be constructed all the time during the training process and it is zero when the action of the Q network output is the same as the output of the BC model and otherwise a positive number. Consequently, the supervised expert loss function of DQfDD-BC is shown as Eq. (8), which is different from DQfD and the complete loss function is the same as Eq. (4).

$$\mathscr{L}_E(Q) = \max_{a \in A}[Q(s,a) + \ell(a, \pi_{bc}(s))] - Q(s, a_E) \tag{8}$$

Algorithm 1: DQfDD-BC: Deep Q-learning from Dynamic Demonstration with Behavioral Cloning

```
1  Initialization: $D^{replay}$ and $D^{demo}$: both initialized
     with the incipient demonstration data set, $\theta$:
     weights for initial Q network, $\theta'$: weights for the
     target network, $\tau$: frequency at which to update the
     target network, $E$: maximum number of episodes,
     $M$: maximum number of newly generated
     trajectories, $k$: number of pre-training gradient
     updates;
2  Train the BC model with $D^{demo}$ using the
     behavioral cloning cross-entropy loss $\ell_{BC}$;
3  for step = 1 to k do
4      Sample a mini-batch of transitions from $D^{replay}$
         with prioritization;
5      Calculate expert loss $\mathcal{L}_E(Q)$ using the BC
         model;
6      Calculate loss $\mathcal{L}(Q)$ using target net;
7      Perform a gradient descent step to update $\theta$;
8      every $\tau$ steps update the target network $\theta' = \theta$;
9  for episode = 1 to E do
10     while not done do
11         Sample action $a$ from $\epsilon - greedy$ policy;
12         Play action $a$ and observe $(s', r, done)$;
13         Store $(s, a, r, s', done)$ into $D^{replay}$;
14         Sample a mini-batch of transitions from
             $D^{replay}$ with prioritization;
15         Calculate expert loss $\mathcal{L}_E(Q)$ using the BC
             model;
16         Calculate $\mathcal{L}(Q)$ using target net;
17         Perform a gradient descent step to update $\theta$;
18         every $\tau$ steps update the target network
             $\theta' = \theta$;
19         $s = s'$;
20     if get the best episode score then
21         for i = 1 to $min(episode//2, M)$ do
22             Agent interacts with the environment
                 until done;
23             Store the trajectory into $D^{demo}$;
24         Fine-tune the BC model with new $D^{demo}$;
```

The goal of introducing a different expert loss function is to smooth the output of the total loss function and stabilize the training process. More importantly, the BC model is utilized to provide reasonable actions and generate supervised losses for all self-generated transitions in the experience replay buffer. Compared with the DQfD method, the sample efficiency is significantly improved.

4.4. Training process

The pseudo-code of the proposed DQfDD-BC method is sketched in Algorithm 1 and consists of three main stages: BC model pre-training, agent pre-training, and joint model self-learning. Firstly, the BC model is pre-trained with Eq. (6) to gain the initial decision-making ability, and then the Q network is also pre-trained in a supervised manner. During the self-learning process, the trained BC model is leveraged to generate expert actions and construct the expert loss function. Combined with TD errors, the self-learning ability is retained and the L2 regularization loss on the network weights is able to prevent over-fitting. After each episode, the best episode score is obtained from the sum of step rewards in each episode compared to the previous record. The policy is evaluated via the episode score and the dynamic demonstration method is applied to D^{demo} to update the demonstration. To reduce the interference of the lucky episode, the demonstration will not be updated in the early stage of the self-learning process. The update is initiated only after the self-learning process has passed at least 10 episodes. Fine-tuning of the BC model is conducted after D^{demo} is updated, thus enabling the BC model to constantly improve the supervised strategy and providing a better target for the agent.

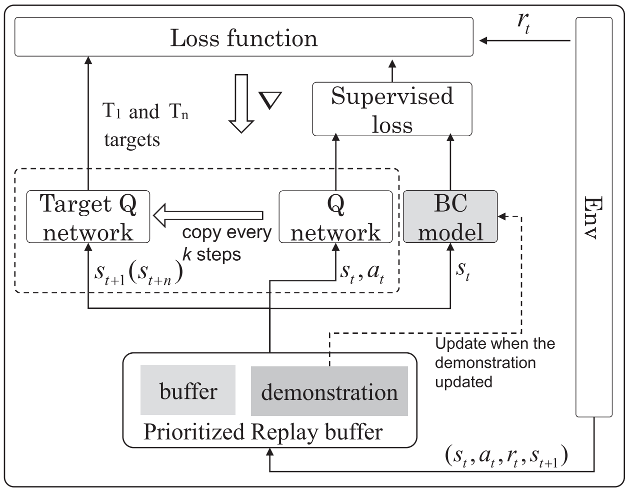


Fig. 1. The technique architecture of the proposed DQfDD-BC method where BC model represents the behavioral cloning model.

5. Experiments and results

5.1. Overview

Experiments configuration The proposed DQfDD-BC method is validated in the classic OpenAI Gym environments: CartPole-v0, CartPole-v1, Acrobot-v1 and LunarLander-v2 [50]. The original reward functions provided by the gym environments are used without giving any additional rewards or penalties during the learning process. At the end of each episode, the rewards for all steps in the episode are summed up to form an episode score. Since the DQfD method is based on DDQN, both DQN [3], DDQN [17], Duel DQN [51] and DQfD [12] algorithms are chosen as the benchmarks to compare with the proposed method.

Two different experimental conditions are considered: perfect demonstrations and imperfect demonstrations. All demonstrations are obtained from the interaction trajectories of pre-trained DDQN models to represent the human experience. The major difference between perfect and imperfect demonstrations is as follows: The perfect demonstrations are obtained from state transitions data of a well-trained DDQN agent, while the imperfect demonstrations are derived from the model during the training process. In all demonstrations, the initial amount of transitions is around 10K before the training and the number will increase when new transitions are added into the demonstrations. The average episode scores of perfect demonstration collected from CartPole-v0, CartPole-v1, Acrobot-v1, LunarLander-v2 are 200, 500, −84, 246. We omit the imperfect demonstration of CartPole-v0 in Table 1, and further discuss the reason in experimental results. The scores of imperfect demonstrations in CartPole-v1, Acrobot-v1, LunarLander-v2 are 357, −238, 14, respectively.

Hyperparameters There are numerous hyperparameters in the experiments and different hyperparameters have a significant effect on the results. In all experiments, the n in $n-step$ return is 10, the batch_size = 64, the size of experience replay buffer is 1000000, the learning rate of supervised model is 0.001, the learning rate of Q networks is 0.00001, and the discount factor $\gamma = 0.95$ in the CartPole environments (0.99 for other environments). Different random seeds have different performances and we choose $[100, 200, 300, \ldots, 1000]$ as the random seed list. In the $\epsilon-greedy$ policy, ϵ linearly decreases from a maximum value of 0.99 to a minimum value of 0.001 and remains at the minimum value. In the CartPole-v1 environment, the descent is maintained for 10000 interaction steps and in other environments 5000. In the prioritized experience replay buffers, new transitions are given the highest probability of being sampled and are later updated based on the absolute values of TD-1 errors. For each transition, the sampling weight is denoted as $w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^{\beta}$, where N is the size of the replay buffer, $P(i)$ is the probability of being sampled and parameter β controls the amount of importance sampling and it is annealed linearly from 0.4 to 1.

The parameters $\lambda_1, \lambda_2, \lambda_3$ determine how the weights of the different losses are assigned and they are important for the hybrid loss function. In all experiments, $\lambda_1 = \lambda_2 = 0.5, \lambda_3 = 0.001$. According to our experimental experience, λ_1 and λ_2 are more appropriate between 0.1–0.7. Too large will easily lead to a decline in the model's self-learning ability, while too small will result in a weaker impact of expert loss on the hybrid loss function and slower learning speed. λ_3 is leveraged to avoid overfitting but too large will restrict the model's expressive ability significantly, causing difficulty in model convergence. $1e-3$ is an appropriate magnitude and the L2 regularization is provided by the optimizer in the PyTorch. The large margin in DQfD is fine-tuned and set to be 1. In case of a margin set too large, the model tends to rely on the supervised learning process heavily and lose its ability to learn independently; conversely, a margin that is too small will cause the advantages of supervised learning unavailable.

Model architecture Since the state of experiment environments are vectors, fully-connected layers are appropriate for the experiments and they are leveraged to construct the Q network and the BC model as well. In the BC model, the hidden unit number of all two fully-connected layers are $[150, 64]$ and *LeakyRelu* is selected as the activation function. The Q network which has $[150, 120]$ hidden units in every hidden layer and the activation function is *Relu*.

Termination condition Within 1000 episodes (it is 500 in CartPole-v0 and v1), the average episode score of last 30 consecutive episodes is greater than certain scores (CartPole-v0: 190, CartPole-v1: 490, Acrobot-v1: −100, LunarLander-v2: 200), then end the current execution. The models are considered as successfully converged if the termination conditions are satisfied. Each method is repeated 10 times with different random seeds.

5.2. Perfect demonstration results

In terms of convergence speed and learning capability with perfect demonstrations, the proposed DQfDD-BC model achieved better results than DQfD model, far better than DQN, DDQN and Duel DQN. Fig. 2 shows the episode rewards of various methods in three gym environments. All learning curves are averaged over 10 random seeds and the shades in the figures are designed to show the experimental results of multiple trials. The solid line in the figure is the average of the results of multiple experiments (denoted as μ), while the range of the shaded area is determined by the standard deviation (demoted as σ) of the results of multiple experiments. The upper and lower boundaries of the shaded area are $\mu + \sigma$ and $\mu - \sigma$, respectively. In CartPole-v0, the DQfDD-BC approach has a slightly faster convergence speed than DQfD during the early learning stage. Although some runs of DQfDD-BC and DQfD failed to satisfy the termination condition, the Table 1 shows that the DQfDD-BC has a smaller average stop episode and it also gained a small advantage in the final performance score. In CartPole-v1 and LunarLander-v2 environments, the DQfDD-BC shows obvious faster-learning speed and fewer episodes are needed for DQfDD-BC to get a higher score than DQfD.

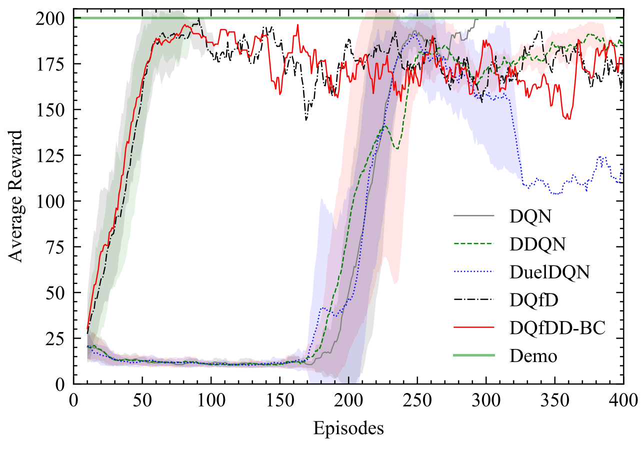
Before the experiment was forced to stop in CartPole-v1, both DQfD and DQfDD-BC can obtain a full score, which proves that the demonstration improves the learning ability of DRL agent, while the DQN, DDQN and Duel DQN failed to trigger the termination condition due to lack of demonstration. The DQfDD-BC method retains the advantages of the DQfD algorithm and further improves the supervised loss function so that all self-generated data can obtain supervised losses, which helps to improve the learning speed.

5.3. Imperfect demonstration results

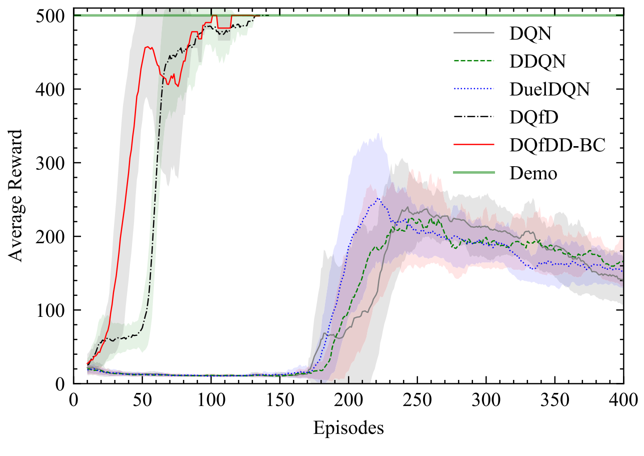
The adaptability of the DQfDD-BC method is validated with imperfect demonstrations. Since perfect demonstrations are difficult to define and acquire in many scenarios, it is more relevant and better to train the agent with imperfect demonstrations than

Table 1
Performance comparison in OpenAI Gym environments. (lower episode is better and higher score is better).

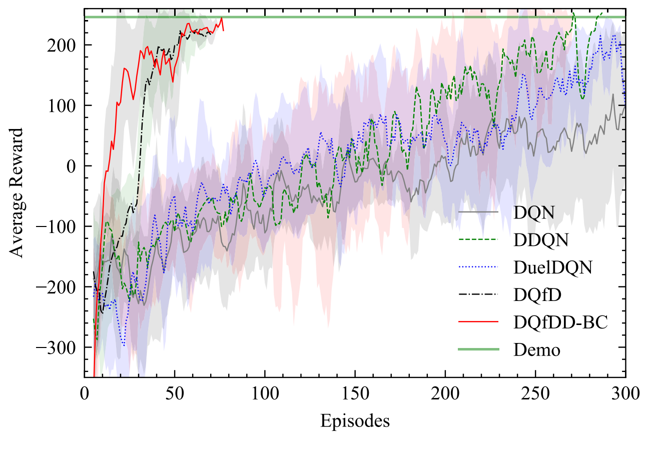
Env	DQN	DDQN	DuelDQN	Perfect Demonstration		Imperfect demonstration	
				DQfD	DQfDD-BC	DQfD	DQfDD-BC
	Average stop episode						
CartPole-v0	254 ± 24	310 ± 85	325 ± 91	162 ± 170	**161** ± 170	–	–
CartPole-v1	500 ± 0	500 ± 0	500 ± 0	121 ± 19	**91** ± 27	500 ± 0	**411** ± 179
Acrobot-v1	500 ± 0	198 ± 123	425 ± 76	51 ± 4	**49** ± 5	261 ± 136	**143** ± 11
LunarLander-v2	443 ± 69	263 ± 21	289 ± 28	65 ± 3	**65** ± 9	402 ± 192	**119** ± 17
Env	Average training score for last 10 episodes						
CartPole-v0	199 ± 0.4	193 ± 0.6	190 ± 4.6	190 ± 10.7	**194** ± 9.7	–	–
CartPole-v1	155 ± 26.8	133 ± 12.1	166 ± 11.9	498 ± 3.9	**500** ± 0.3	348 ± 25.1	**429** ± 46.0
Acrobot-v1	−161 ± 17.8	−97 ± 5.3	−112 ± 3.1	−95 ± 4.0	**−91** ± 2.5	−96 ± 5.3	**−95** ± 7.8
LunarLander-v2	206 ± 12.4	231 ± 6.5	186 ± 19.2	219 ± 8.7	**223** ± 6.1	**234** ± 4.4	211 ± 20.4



(a) CartPole-v0

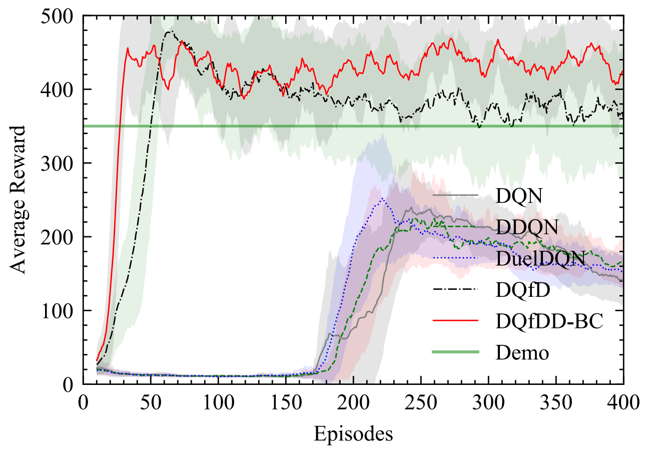


(b) CartPole-v1

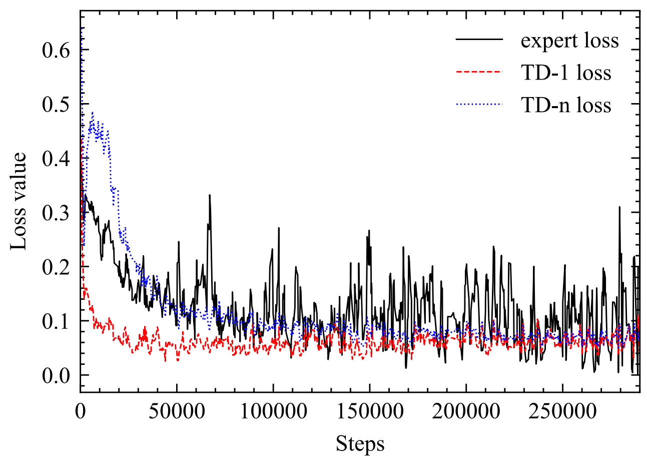


(c) LunarLander-v2

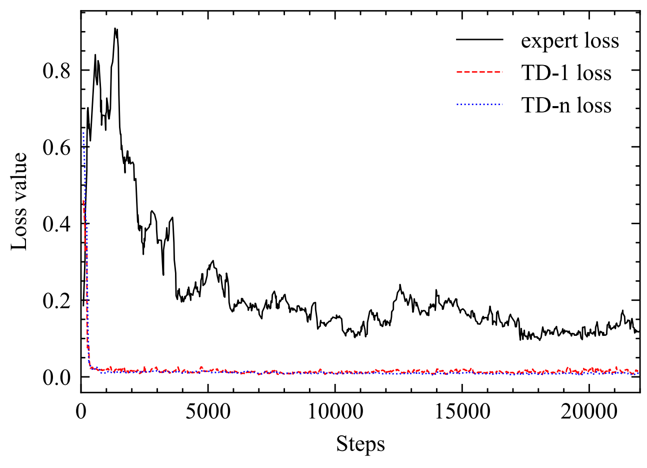
Fig. 2. The training episode scores of different methods with perfect demonstrations in different environments. The horizontal green line is the average episode scores of the perfect demonstrations.



(a) Average training scores



(b) Losses of DQfD



(c) Losses of DQfDD-BC

Fig. 3. Experimental result with an imperfect demonstration. (a) The training score of different methods in the CartPole-v1 environment. The horizontal green line is the average score of the demonstrations. (b) and (c) show the loss function value during the training process.

the random policy. Some off-line optimization methods can also be used to generated imperfect demonstrations. Fig. 3 (a) shows the episode scores of different method with imperfect demonstrations in the CartPole-v1 environment. The results show that the DQfDD-BC model outperforms DQfD by a significant margin in terms of convergence speed. By keeping the demonstrations updated, the proposed DQfDD-BC method allows the newly updated demonstration to avoid the negative influence brought by the imperfect demonstration and eventually converge to a better performance level.

The proposed DQfDD-BC method can adapt to imperfect demonstrations well and obtain a higher average final performance than DQfD, which shows great dependence on the demonstration and ultimately converges to the vicinity of the demonstration; therefore, imperfect demonstrations degrade the performance of the DQfD method. To figure out the reason of DQfDD-BC's outperforming DQfD, the variation of each component of the complete loss function is examined. Fig. 3 (b) shows that the TD losses (TD-1, TD-n) of DQfD have been stabilized within a small fluctuation range in the later stages of self-learning while the expert loss fluctuated in a wide range. This is partially due to the action output by the learned policy can be quite different from those sampled from the imperfect demonstrations. The agent can obtain a better policy through the TD losses and provides an appropriate action. However, the supervised loss draws the learned policy near the policy represented by the imperfect demonstration and inapposite actions are provided. Additionally, the fixed imperfect demonstration is not immune to its own negative effects, resulting in the inability to converge to the highest score. The expert loss of DQfDD-BC is more steady as shown in Fig. 3 (c) and the agent achieves a better final performance. The dynamic demonstration has the ability to reduce the disadvantage of imperfect demonstration and expand the state-action space covered by the demonstration.

Table 1 shows the comparison between DQN, DDQN, Duel DQN, DQfD and DQfDD-BC in different gym environments. The average stop episode means the average stop episode number among different runs and roughly represents the learning speed and the average final score illustrates the final performance of these methods. In all environments, the proposed DQfDD-BC method is successfully converged in the smallest number of episodes and obtains the best final performance in almost all environments, which demonstrates the significant advantages of DQfDD-BC in learning speed and performance. Need to mention that the termination condition of CartPole-v0 is much easier to be satisfied compared to the CartPole-v1. In some lucky episodes, the termination condition can be triggered in CartPole-v0, which is almost impossible in CartPole-v1. The imperfect demonstration of CartPole-v0 is hard to be distinguished from the random policy and trained policy. Therefore we omit the test with an imperfect demonstration of CartPole-v0. The quality of imperfect demonstration of CartPole-v1 is more reliable compared to CartPole-v0.

5.4. Ablations

To confirm the importance of both dynamic demonstrations and expert loss with BC model in the proposed DQfDD-BC method, the method without dynamic demonstration or BC model are tested in the CartPole-v1 environment. Fig. 4 shows the effect of integrating dynamic demonstrations and BC model to the original DQfD model. Clearly, when the dynamic demonstration is added to the DQfD algorithm (new model named DQfDD), the DQfDD model presents a significant improvement in training speed during the early stage. The DQfDD can also converge to the highest score and ended the training process in a short time, while the DQfD needs some more time. This indicates that by adding new samples to the demonstrations, a large number of high-performance transitions can improve the generalization ability and training stability of the model.

In the case of only BC model triggered (new model named DQfD-BC), the training speed of the DQfD-BC model is significantly improved compared with DQfD. As shown in Fig. 4, the average training score of the DQfD algorithm reaches only about 80% of the convergence state, while the DQfD-BC model finishes the training process. The DQfD-BC has a slightly lower learning speed than the DQfDD-BC method during the learning process but the DQfD-BC ends the learning process earlier. By introducing a supervised model, the self-generated transitions can be utilized to generate an expert loss. Therefore higher sample efficiency is achieved and the learning speed is improved significantly. The proposed DQfDD-BC method, which combines the benefits from dynamic demonstrations and the expert loss generated from BC model, can improve the learning speed simultaneously and retain excellent performance with an imperfect demonstration.

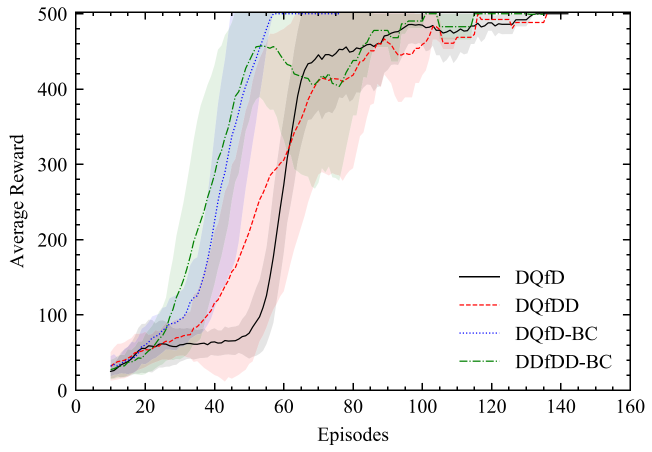


Fig. 4. Ablation experimental results. The dynamic demonstration (DQfDD) and expert loss with a behavioral cloning model (DQfD-BC) are added to the original DQfD method.

6. Conclusion

In this work, we propose the SADRL framework and develop the DQfDD-BC method to enhance the learning efficiency and performance of DRL. The experimental results demonstrate that regardless of the perfection of historical demonstrations, DQfDD-BC can effectively improve the learning speed and achieve comparable final performance in generic environments. From an ablation experiment, the dynamic demonstration and expert loss with the BC model showed their potential to improve the learning quality.

The proposed model provides a feasible solution to bridge the DRL methods and the decision-making problems in complex real-world environments by learning from the human experience to improve the DRL agent while resisting the interference from the imperfect demonstration. In the future, to fully extract and utilize the human experience and examine the potential of SADRL, real-world settings, such as power grid operation systems, are considered. Considering the properties of recorded human demonstrations, we will explore and improve the ability of the proposed method to handle engineering issues in data sparsity, human diversity, etc.

CRediT authorship contribution statement

Xiaoshuang Li: Conceptualization, Methodology, Software, Investigation, Writing - original draft. **Xiao Wang:** Data curation, Writing - review & editing, Funding acquisition. **Xinhu Zheng:** Writing - review & editing. **Junchen Jin:** Validation, Writing - review & editing. **Yanhao Huang:** Resources, Writing - review & editing, Funding acquisition. **Jun Jason Zhang:** Writing - review & editing, Funding acquisition. **Fei-Yue Wang:** Supervision, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Research supported by the National Key R&D Program of China (2018AAA0101500, 2018AAA0101502). We thank Dr. Peidong Xu, Dr. Xingyuan Dai for their advice and technical supports. We appreciate the help from the implementation of https://github.com/asdiijj/implementation-of-dqfd.

References

[1] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, et al., Mastering the game of Go with deep neural networks and tree search, Nature 529 (7587) (2016) 484.
[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, et al., Mastering the game of Go without human knowledge, Nature 550 (7676) (2017) 354.
[3] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529.
[4] F. Li, Q. Jiang, S. Zhang, M. Wei, R. Song, Robot skill acquisition in assembly process using deep reinforcement learning, Neurocomputing 345 (2019) 92–102.
[5] Y. Xiong, G. Zheng, K. Xu, Z. Li, Learning traffic signal control from demonstrations, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 2289–2292.
[6] T. Hester, P. Stone, Texplore: real-time sample-efficient reinforcement learning for robots, Mach. Learn. 90 (3) (2013) 385–429.
[7] G. Shani, D. Heckerman, R.I. Brafman, C. Boutilier, An mdp-based recommender system, J. Mach. Learn. Res. 6 (9) (2005).
[8] J. Li, X. Shi, J. Li, X. Zhang, J. Wang, Random curiosity-driven exploration in deep reinforcement learning, Neurocomputing 418 (2020) 139–147.
[9] Z. Zhao, Q. Wang, X. Li, Deep reinforcement learning based lane detection and localization, Neurocomputing 413 (2020) 328–338.
[10] B.R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A.A.A. Sallab, S. Yogamani, et al., Deep reinforcement learning for autonomous driving: A survey. arXiv preprint arXiv:200200444 2020..
[11] J. Garrido, W. Yu, A. Soria, Human behavior learning for robot in joint space, Neurocomputing 155 (2015) 22–31.
[12] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, et al., Deep q-learning from demonstrations, in: Association for the Advancement of Artificial Intelligence, 2018.
[13] J. Matas, S. James, A.J. Davison, Sim-to-real reinforcement learning for deformable object manipulation, in: Conference on Robot Learning, 2018, p. 734–743..
[14] S. Schaal, Learning from demonstration. In: Advances in neural information processing systems, 1997, p. 1040–1046..
[15] F. Behbahani, K. Shiarlis, X. Chen, V. Kurin, S. Kasewa, C. Stirbu, et al., Learning from demonstration in the wild, in: 2019 International Conference on Robotics and Automation (ICRA), IEEE, 2019, pp. 775–781.
[16] S. Xu, Y. Ou, J. Duan, X. Wu, W. Feng, M. Liu, Robot trajectory tracking control using learning from demonstration method, Neurocomputing 338 (2019) 249–261.
[17] H.v. Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. 2016, p. 2094–2100..
[18] B. Piot, M. Geist, O. Pietquin, Boosted bellman residual minimization handling expert demonstrations, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2014, pp. 549–564.
[19] B. Piot, M. Geist, O. Pietquin, Boosted and reward-regularized classification for apprenticeship learning, in: Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems. International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 1249–1256.
[20] F. Torabi, G. Warnell, P. Stone, Behavioral cloning from observation. arXiv preprint arXiv:180501954 2018..
[21] A. Bühler, A. Gaidon, A. Cramariuc, R. Ambrus, G. Rosman, W. Burgard, Driving through ghosts: Behavioral cloning with false positives. arXiv preprint arXiv:200812969 2020..
[22] C. Ma, L. Chen, J. Yong, Au r-cnn: Encoding expert prior knowledge into r-cnn for action unit detection, Neurocomputing 355 (2019) 35–47.
[23] H. Ravichandar, A.S. Polydoros, S. Chernova, A. Billard, Recent advances in robot learning from demonstration, Annu. Rev. Control Robotics Autonomous Syst. 3 (2020).
[24] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, et al. End to end learning for self-driving cars. arXiv preprint arXiv:160407316 2016..
[25] X. Li, Z. Guo, X. Dai, Y. Lin, J. Jin, F. Zhu, et al., Deep imitation learning for traffic signal control and operations based on graph convolutional neural networks, in: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2020, pp. 1–6.
[26] J. Duan, Y. Ou, S. Xu, M. Liu, Sequential learning unification controller from human demonstrations for robotic compliant manipulation, Neurocomputing 366 (2019) 35–45.
[27] A. Hussein, M.M. Gaber, E. Elyan, C. Jayne, Imitation learning: a survey of learning methods, ACM Comput. Surveys (CSUR) 50 (2) (2017) 1–35.
[28] D.S. Brown, W. Goo, S. Niekum, Better-than-demonstrator imitation learning via automatically-ranked demonstrations, in: Conference on Robot Learning, 2020, p. 330–359..
[29] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, et al., Deep imitation learning for complex manipulation tasks from virtual reality teleoperation, in: 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 1–8.
[30] X. Li, P. Ye, J. Jin, F. Zhu, F.Y. Wang, Data augmented deep behavioral cloning for urban traffic control operations under a parallel learning framework, IEEE Trans. Intell. Transp. Syst. (2021) 1–10, https://doi.org/10.1109/TITS.2020.3048151.
[31] M. Bansal, A. Krizhevsky, A. Ogale, Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. arXiv preprint arXiv:181203079 2018.
[32] S. Ross, G. Gordon, D. Bagnell, A reduction of imitation learning and structured prediction to no-regret online learning, in: Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011, pp. 627–635.
[33] W. Sun, A. Venkatraman, G.J. Gordon, B. Boots, J.A. Bagnell, Deeply aggravated: Differentiable imitation learning for sequential prediction, in: International Conference on Machine Learning, 2017, pp. 3309–3318.
[34] J. Ho, S. Ermon, Generative adversarial imitation learning, in: Advances in neural information processing systems, 2016, pp. 4565–4573.
[35] B. Wang, E. Adeli, H.k. Chiu, D.A. Huang, J.C. Niebles, Imitation learning for human pose prediction, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, p. 7124–7133..
[36] G. Zuo, K. Chen, J. Lu, X. Huang, Deterministic generative adversarial imitation learning, Neurocomputing 388 (2020) 60–69.
[37] B. Kang, Z. Jie, J. Feng, Policy optimization with demonstrations, in: International Conference on Machine Learning, 2018, p. 2469–2478..
[38] J. Song, H. Ren, D. Sadigh, S. Ermon, Multi-agent generative adversarial imitation learning, in: Advances in neural information processing systems, 2018, pp. 7461–7472.
[39] T. Brys, A. Harutyunyan, H.B. Suay, S. Chernova, M.E. Taylor, A. Nowé, Reinforcement learning from demonstration through shaping, in: Twenty-fourth international joint conference on artificial intelligence, 2015.
[40] H.B. Suay, T. Brys, M.E. Taylor, S. Chernova, Learning from demonstration for shaping through inverse reinforcement learning, in: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, 2016, pp. 429–437.
[41] N. Jiang, S. Jin, C. Zhang, Hierarchical automatic curriculum learning: converting a sparse reward navigation task into dense reward, Neurocomputing 360 (2019) 265–278.
[42] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R.H. Campbell, K. Czechowski, et al., Model-based reinforcement learning for atari. arXiv preprint arXiv:190300374 2019;..
[43] N.O. Lambert, D.S. Drew, J. Yaconelli, S. Levine, R. Calandra, K.S. Pister, Low-level control of a quadrotor with deep model-based reinforcement learning, IEEE Robot. Autom. Lett. 4 (4) (2019) 4224–4230.
[44] B. Thananjeyan, A. Balakrishna, U. Rosolia, F. Li, R. McAllister, J.E. Gonzalez, et al., Safety augmented value estimation from demonstrations (saved): safe deep model-based rl for sparse cost robotic tasks, IEEE Robot. Autom. Lett. 5 (2) (2020) 3612–3619.
[45] J. Oh, Y. Guo, S. Singh, H. Lee, Self-imitation learning, in: International Conference on Machine Learning, 2018, pp. 3878–3887.
[46] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, et al., Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. arXiv preprint arXiv:170708817 2017..
[47] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, et al., Continuous control with deep reinforcement learning. arXiv preprint arXiv:150902971 2015..
[48] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, MIT press, 2018.
[49] T. Schaul, J. Quan, I. Antonoglou, D. Silver, Prioritized experience replay. arXiv preprint arXiv:151105952 2015..
[50] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, et al., Openai gym. arXiv preprint arXiv:160601540 2016..
[51] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, N. Freitas, Dueling Network Architectures for Deep Reinforcement Learning, in: International Conference on Machine Learning. PMLR, 2016, pp. 1995–2003.

Xiaoshuang Li received B.S. degree in automation from Wuhan University, Wuhan, China in 2017. He is currently pursuing the Ph.D. degree in control theory and control engineering at the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences. His research interests include intelligent transportation systems, deep learning and deep reinforcement learning.

Xiao Wang received the bachelor's degree in network engineering from the Dalian University of Technology, Dalian, China, in 2011, and the Ph.D. degree in social computing from the University of Chinese Academy of Sciences, Beijing, China, in 2016. She is currently an Associate Professor with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences. She has published more than a dozen SCI/EI articles and translated three technical books (English to Chinese). Her research interests include social transportation, cybermovement organizations, artificial intelligence, and social network analysis. Dr. Wang has served the IEEE Transactions on Intelligent Transportation Systems, the IEEE/CAA Journal of Automatica Sinica, and ACM Transactions on Intelligent Systems and Technology as a Peer Reviewer with a good reputation.

Xinhu Zheng received the B.S. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2011. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Minnesota Twin Cities, Minneapolis, MN, USA. His research interests include social computing, machine learning, and data analytics.

Junchen Jin received the B.Eng. degree in traffic engineering from Beijing Jiaotong University, Beijing, China, and the M.Sc. and Ph.D. degrees in transport science from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2014 and 2018, respectively. He was the Vice Director at the Smart Transportation Research Institute, Enjoyor Co., Ltd., Hangzhou, China. He is also a Post Doctoral Researcher with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing. His research interests include intelligent transport systems, traffic simulation and control, recommender systems, artificial intelligence, deep learning, and reinforcement learning.

Yanhao Huang received his Ph.D. degree in Power System Automation from China Electric Power Research Institute, Beijing, China, in 2015. His research interests include power system simulation, intelligent technologies and electric power big data.

Jun Jason Zhang received the B.E. and M.E. degrees in electrical engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2003 and 2005, respectively, and the Ph.D. degree in electrical engineering from Arizona State University, USA, in 2008. He is currently a Professor with the School of Electrical Engineering and Automation, Wuhan University. He authored/coauthored over 70 peer reviewed publications. His research interests include the areas of sensing theory, signal processing and implementation, time-varying system modeling, and their applications in intelligent power and energy systems. He is the Technical Co-Chair of the 48th North American Power Symposium (NAPS 2016).

Fei-Yue Wang received his Ph.D. degree in computer and systems engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990. He joined The University of Arizona in 1990 and became a Professor and the Director of the Robotics and Automation Laboratory and the Program in Advanced Research for Complex Systems. In 1999, he founded the Intelligent Control and Systems Engineering Center at the Institute of Automation, Chinese Academy of Sciences (CAS), Beijing, China, under the support of the Outstanding Chinese Talents Program from the State Planning Council, and in 2002, was appointed as the Director of the Key Laboratory of Complex Systems and Intelligence Science, CAS. In 2011, he became the State Specially Appointed Expert and the Director of the State Key Laboratory for Management and Control of Complex Systems. His current research focuses on methods and applications for parallel intelligence, social computing, and knowledge automation. He is a fellow of INCOSE, IFAC, ASME, and AAAS. In 2007, he received the National Prize in Natural Sciences of China and became an Outstanding Scientist of ACM for his work in intelligent control and social computing. He received the IEEE ITS Outstanding Application and Research Awards in 2009 and 2011, respectively. In 2014, he received the IEEE SMC Society Norbert Wiener Award. Since 1997, he has been serving as the General or Program Chair of over 30 IEEE, INFORMS, IFAC, ACM, and ASME conferences. He was the President of the IEEE ITS Society from 2005 to 2007, the Chinese Association for Science and Technology, USA, in 2005, the American Zhu Kezhen Education Foundation from 2007 to 2008, the Vice President of the ACM China Council from 2010 to 2011, the Vice President and the Secretary General of the Chinese Association of Automation from 2008–2018. He was the Founding Editor-in-Chief (EiC) of the International Journal of Intelligent Control and Systems from 1995 to 2000, the IEEE ITS Magazine from 2006 to 2007, the IEEE/CAA JOURNAL OF AUTOMATICA SINICA from 2014–2017, and the China's Journal of Command and Control from 2015–2020. He was the EiC of the IEEE Intelligent Systems from 2009 to 2012, the IEEE TRANSACTIONS ON Intelligent Transportation Systems from 2009 to 2016, and is the EiC of the IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS since 2017, and the Founding EiC of China's Journal of Intelligent Science and Technology since 2019. Currently, he is the President of CAA's Supervision Council, IEEE Council on RFID, and Vice President of IEEE Systems, Man, and Cybernetics Society.