DEEP Q LEARNING FROM DYNAMIC DEMONSTRATION WITH BEHAVIORAL CLONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Although Deep Reinforcement Learning (DRL) has proven its capability to learn optimal policies by directly interacting with simulation environments, scaling up a DRL model is difficult due to exploding computational complexity compared with a supervised learning model. This study proposes a novel approach integrating deep Q learning from dynamic demonstrations with a behavioral cloning model (DQfDD-BC), which includes a supervised learning technique of instructing a DRL model to enhance its performance. Specifically, the DQfDD-BC model leverages historical demonstrations to pre-train a supervised BC model and to consistently update it by using the generated dynamic demonstrations. Then the DQfDD-BC model manages the sample complexity by exploiting both the historical and generated demonstrations. An 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 adapts to different imperfection levels of demonstrations, and meanwhile, significantly accelerates the learning processes. As illustrated in an ablation study, the dynamic demonstration and expert loss mechanisms with the use of a BC model contribute to improving the learning convergence performance compared with the origin DQfD model.

1 INTRODUCTION

Deep reinforcement learning (DRL) methods have made great progress (Mnih et al., 2013; 2015; Silver et al., 2017) when applied in several rule-based applications such as the Go game (Silver et al., 2016). However, due to the diversity and uncertainty of complex systems, the establishment of a simulation environment is difficult consistent with the real-world system. Therefore, DRL algorithms usually fail in a direct application to many real-world scenarios. Meanwhile, a DRL model may produce an action that is sampled from a stochastic policy when exploring a state-action space. However, random actions are not allowed in many real-world circumstances. For example, in autonomous driving experiments (Kiran et al., 2020), a stochastic policy may contribute to road accidents in real-world scenarios. Therefore, fitting to complex situations becomes one of the most urgent tasks for applying a DRL model for complicated decision-making tasks.

It is noted that human experts have great advantages in learning efficiency and decision-making performance (Tsividis et al., 2017). Incorporating expert knowledge is a potential solution to enhance the adaptability of DRL models for complex tasks (Hester et al., 2018; Matas et al., 2018). Nevertheless, the knowledge and experience of expert are difficult to be modeled and described directly. One solution, attracting more and more attention, is to indirectly learn expert strategies by learning their decision trajectories, also known as demonstrations (Schaal, 1997; Behbahani et al., 2019; Ravichandar et al., 2020). Particularly, deep Q learning from demonstrations (DQfD) is a typical algorithm that succeeded in combining DRL with demonstrations (Hester et al., 2018), which combines the temporal difference (TD) error in the traditional DQN algorithm with supervised expert loss, by constructing a hybrid loss function. Through a specially designed large margin supervised loss function (Piot et al., 2014a;b), the DQfD method can guide and assist an agent to learn the expert's knowledge by constantly steering the agent learning strategies closer to those represented by the demonstration.

However, the DQfD model suffers from three major issues: (1) In the DQfD learning process, the trajectory data in historical demonstration dataset is the only source for contributing loss values, which does not include the self-generated trajectories of the learned 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 too limited to cover sufficient state-action space during the agent training process, especially when it is difficult to collect demonstrations in real-world applications. Also, with more newly-generated transitions in the experience replay buffer, historical demonstrations would make smaller contributions to the policy improvement as their sampling probability becomes lower. (3) The DQfD algorithm ignores the imperfection of historical demonstrations, which commonly exists in real-world applications. When a trained model learns from an imperfect demonstration, the continuous learning from the demonstration is also detrimental to the improvement of the policy.

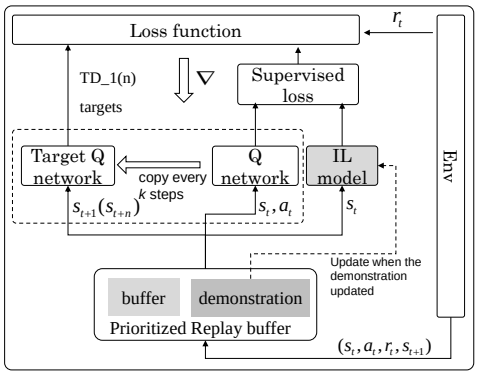


Figure 1: The technique framework of the proposed DQfDD-BC method where IL model represents an imitation learning model.

To address the above issues, we propose a novel deep Q learning from dynamic demonstration with behavioral cloning (DQfDD-BC) method. Fig. 1 illustrates the structure of the proposed approach. DQfDD-BC shares the same basic components of the DDQN algorithm (Hasselt et al., 2016), including a target Q network and an experience replay buffer, with an additional module of imitation learning (IL) (Hussein et al., 2017) in the replay buffer. Besides, the loss value is calculated considering both the output of the trained Q network and the IL model. Two main contributions are summarized as follows:

1. An IL model, named behavioral cloning (BC) (Torabi et al., 2018; Bühler et al., 2020), is proposed to generate expert loss to utilize all the transitions in the experience replay buffer. It firstly attempts to extract the experts' behaviors from the transitions in demonstrations and allows the agent to provide a reasonable action when facing newly generated states. Meanwhile, the agent's actions are compared with those generated by the BC model by computing a self-designed expert loss function. The inclusion of the BC model allows the knowledge in the demonstrations to be sufficiently utilized in the training process, also with the capability of training the model in the environment states where experts have not ever encountered.
2. An automatic update mechanism is proposed to adaptively enhance the BC model. In particular, new transitions are generated by the trained agents to fine-tune the BC model if the model achieves a relative high performance score. Such mechanism tries to include more high-quality transition samples, and avoids potential adverse impacts caused by imperfect demonstrations.

In this study, we evaluate the proposed DQfDD-BC method on several gym environments (Brockman et al., 2016). For comparison purposes, the DDQN and DQfD methods are used as the baselines (Hasselt et al., 2016; Hester et al., 2018). The experiments clearly demonstrate that DQfDD-BC surpasses all the baselines in terms of convergence speed as well as decision-making performances. The ablation experiments also show that both the proposed expert loss function together with the BC model and dynamic demonstration contribute significantly to the performance superior of the DQfDD-BC algorithm.

2 Related works

Learning from demonstration (LfD) is a class of decision-making methods by learning from demonstration (Schaal, 1997; Behbahani et al., 2019; Ravichandar et al., 2020) and imitation learning

(Hussein et al., 2017; Brown et al., 2020) is a subset of LfD. As one type of imitation learning, the behavioral cloning (Torabi et al., 2018) method has received extensive attention (Zhang et al., 2018; Bühler et al., 2020) due to its great performance. It 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 demonstration by minimizing various supervised losses. Behavioral cloning is an end-to-end learning method. In a study by NVIDIA, the model obtained surprisingly results by minimizing the mean squared error between the steering command output by the network and the command of a human driver (Bojarski et al., 2016). *Chauffeurnet* further enhances the performance of imitation learning in autopilot by synthesizing the worst scenarios (Bansal et al., 2018).

DAGGER (Ross et al., 2011) is proposed to address the shortcoming of limited state-action space covered by demonstration in traditional imitation learning. The goal of performance improvement is achieved by adding new samples continuously during the learning process. However, the DAGGER method requires an expert, who is always available, to assist in labeling the data, which reduces the practicality of the method. Deeply AggreVaTeD (Sun et al., 2017) enables the DAGGER to cope with continuous action spaces by using deep neural networks but the determination of DAGGER was also preserved.

Another typical class of imitation learning algorithms is generative adversarial imitation learning (GAIL) (Ho & Ermon, 2016; Wang et al., 2019). GAIL does not model the map from state to action directly, and it learns from the demonstrations by introducing the generator and discriminator. The generator is used to generate state-action pairs and learn a policy, while the discriminator is used to distinguish whether a state-action pair is from the expert or the learned policy. The information contained in the demonstration is indirectly combined with the policy improvement through the process of adversarial learning. GAIL shows good performance on the high-dimensional continuous-control problem (Kang et al., 2018; Song et al., 2018). Policy optimization with demonstrations (POfD) (Kang et al., 2018) utilizes the demonstration by an adversarial learning method and has yielded impressive results in 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 system for evaluating the performance of the policy, reshaping the reward function through the demonstration is an effective approach to improve DRL performance (Brys et al., 2015; Suay et al., 2016). By transforming the form of the reward function, the difficulty of agent training has been reduced. Model-Based DRL (MBRL) has shown great power in some difficult tasks (Kaiser et al., 2019) and demonstration is also leveraged to deal with the MBRL problems (Lambert et al., 2019; Thananjeyan et al., 2020).

All of the above methods yielded impressive results but most of them utilized demonstrations through reward shaping or imitating the demonstration. Another idea is to learn demonstration directly and the transitions in the experience replay buffer and demonstrations are both utilized by DRL agents (Hester et al., 2018; Oh et al., 2018; Xiong et al., 2019). These approaches put the demonstration into the experience replay buffer and sample from the hybrid experience replay buffer. The DQfD algorithm enhances the DDQN (Hasselt et al., 2016) 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 (Oh et al., 2018). The DDPGfD algorithm (Vecerik et al., 2017) is similar to the DQfD and they both assist in enhancing the original algorithms through a hybrid loss function with the "expert loss". The difference between the two is that the DDPGfD algorithm is based on the DDPG algorithm (Lillicrap et al., 2015), which is designed to solve continuous action space problems. However, both DQfD and DDPGfD suffer from the problem of under-exploiting demonstration data (Kang et al., 2018) and our DQfDD-BC method has made new and comprehensive progress on solving the problem.

3 PROBLEM STATEMENT

We consider the standard Markov Decision Process (MDP) (Sutton & Barto, 2018), which is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$, where $\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma$ represent the state space, action space, state transition

distribution, reward function and discount factor respectively. The transition distribution $\mathcal{P}(s' \mid s, a)$ describes the process of taking action a at state s and reward function $r(s, a)$ gives the feedback. A policy $\pi(a|s)$ specifies how the agent responses to each state. The goal of the agent is to find the policy π whcih maps states to actions that maximizes 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.

$$J(Q) = J_{DQ}(Q) + \lambda_1 J_n(Q) + \lambda_2 J_E(Q) + \lambda_3 J_{L2}(Q) \tag{1}$$

$$J_E(Q) = \max_{a \in A} \left[Q(s, a) + l(a_E, a)\right] - Q(s, a_E) \tag{2}$$

In the DQfD algorithm, the complete loss (Eq. 1) function contains four parts: $J_{DQ}(Q)$, $J_n(Q)$, $J_E(Q)$ and $J_{L2}(Q)$, which are the double Q learning TD loss, n-step double Q learning TD loss, expert loss and L2 regularization loss, respectively. The λ parameters control the weighting between the losses. The expert loss (Eq. 2) in DQfD is the most significant loss among the losses, allows the model to satisfy the Bellman equation, and also enables the agent to learn from the demonstrations. The large margin supervised loss $l(a_E, a)$ is 0 when $a = a_E$ with a_E being the expert action in the demonstration, and a positive constant otherwise. This loss forces the Q values of the other actions to be at least a margin lower than the value of the demonstrator's action (Hester et al., 2018). It is noted that, when the sampled transitions are not in the demonstration, $\lambda_2 = 0$ and the expert loss $J_E(Q)$ is non-functional.

4 METHODOLOGY

4.1 DYNAMIC DEMONSTRATIONS FOR DQFD

As the transition quality of historical demonstrations is usually much higher than that associated with stochastic policies, a DQfD agent learns its initial knowledge from preset demonstrations. One of the major improvement of our proposed DQfDD-BC approach is adding the newly generated interactions into the demonstrations. Such transitions are collected from a trained model which, on average, can have a higher performance score for the same task. Particularly, during the training process, the transition samples are automatically generated and inserted into the demonstration dataset whenever the agent reaches a new high score. By adding new and better transition samples, the quality of the transition samples is continuously improved while attempting to cover the full state-action space, which can continuously improve the performance of the policy represented by the demonstration, and generate a continuously positive effect on the the model performance.

The idea of the proposed approach is close to the DAGGER algorithm, especially for improving the model's decision-making capability by adding new information to the experience replay buffer and continuously using the added data to optimize the model parameters. However, unlike DAGGER, our method does not rely on manual labeling, but automatically determines whether new data need to added to the demonstration, which can significantly reduce the computationally cost and improves the applicability of the method.

4.2 EXPERT LOSS WITH SUPERVISED BC MODEL

The DQfD method originally leverages a binary supervised large margin loss function $l(a_E, a)$ to compare the generated actions with those in demonstrations under the same system states. However, the values of this loss function are prone to cause a volatile gradient and lead to the instability of the training process. In this study, DQfDD-BC is designed for discrete action spaces such that the BC model is considered to solve a multi-label classification problem.

In detail, 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 while D^{demo} consisting of both historical and self-generated demonstrations. The BC model is first pre-trained with demonstrations D^{demo} to ensure its initial decision-making ability. Prioritized experience replay mechanism (Schaul et al., 2015) is also applied to both of the two replay buffers to improve the sample efficiency.

$$l_{BC} = Crossentropy(a_E, \pi_{bc}(s_t)), \tag{3}$$

$$l(\pi, \pi_{bc}; s_t) = Crossentropy(\pi(s_t), \pi_{bc}(s_t)). \tag{4}$$

As shown in Eqs. 3 and 4, the DQfDD-BC model generates expert losses for all self-generated transitions instead of directly using historical demonstrations. In Eqs. 3 and 4, a_E refers to the action in the demonstration and π_{bc} represents the learned policy of the BC model. The action $\pi_{bc}(s_t)$ is generated by the latest BC model and compared with the policy modeled by the Q network $\pi(s_t)$. The supervised loss 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 complete expert loss function of DQfDD-BC is shown as Eq. 5.

$$J_E(Q) = \max_{a \in A} \left[Q(s,a) + l(\pi, \pi_{bc}; s_t) \right] - Q\left(s, a_E\right), \tag{5}$$

This expert loss function aims to smooth the output of the loss function and stabilize the training process. And also, it can generate a supervised loss for all self-generated transitions in the experience replay buffer with significantly improved the sample efficiency.

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

Initialization: D^{replay} and D^{demo}: both initialized with the incipient demonstration data set, θ: weights for initial Q network, $\theta^{'}$: weights for target network, τ: frequency at which to update target net, E: maximum number of episodes, M: maximum number of newly generated trajectories, k: number of pre-training gradient updates;
Train the BC model with D^{demo} with the behavioral cloning cross-entropy loss (Eq. 3);
for *step = 1* **to** k **do**
    Sample a mini-batch of transitions from D^{replay} with prioritization;
    Calculate expert loss $J_E(Q)$ using the BC model;
    Calculate loss $J(Q)$ using target net;
    Perform a gradient descent step to update θ;
    if *step mod* $\tau = 0$ **then** $\theta^{'} = \theta$;
for *episode = 1* **to** E **do**
    while *not done* **do**
        Sample action a from $\epsilon - greedy$ policy;
        Play action a and observe $(s^{'}, r, done)$;
        Store $(s, a, r, s^{'}, done)$ into D^{replay};
        Sample a mini-batch of transitions from D^{replay} with prioritization;
        Calculate expert loss$J_E(Q)$ using the BC model;
        Calculate $J(Q)$ using target net;
        Perform a gradient descent step to update θ;
        every τ steps update the target network $\theta^{'} = \theta$;
        $s = s^{'}$;
    if *get the best episode score* **then**
        for *i = 1* **to** $min(episode//2, M)$ **do**
            Agent interacts with the environment until $done$;
            Store the trajectory into D^{demo};
        Fine-tune the BC model with new D^{demo};

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. 3 to gain the initial decision-making ability, and then the Q network is also pre-trained in a supervised manner. Through the trained BC model, the information in the demonstrations indirectly contributes to the improvement of the agent performance. During the self-learning process, the trained BC model is leveraged to generate an expert loss to construct the complete loss function. Combined with TD errors, the self-learning ability is retained and the L2

regularization losses on the network weights to prevent over-fitting. After each episode, we evaluate the policy and apply the dynamic demonstration method to D^{demo} to update the demonstration. 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.

5 EXPERIMENTS AND RESULTS

Experiments setting We evaluated DQfDD-BC in the classic OpenAI Gym environments: CartPole-v0, CartPole-v1, Acrobot-v1 and LunarLander-v2 (Brockman et al., 2016). We used the original reward functions provided by the gym environments without giving any additional rewards or penalties when the model succeeds or fails. At the end of each episode, the rewards for all steps in the episode are summed up to form an episode score. The classical DDQN (Hasselt et al., 2016) and DQfD (Hester et al., 2018) algorithms are chosen as the benchmark algorithms to compare with the proposed approach.

Two experimental conditions were set up: perfect demonstrations and imperfect demonstrations. All demonstrations were obtained from pre-trained DDQN models and 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 ten thousand before the training and the number will increase when new transitions are added into the demonstrations. More implementation details are presented in the appendix.

Experimental results Table 1 shows the comparison between DDQN, DQfD and DQfDD-BC in different gym environments. The average stop episode roughly represents the learning speed and the average final score illustrates the final performance of these methods. Meanwhile, all the "Demo" columns data shows the average scores of the demonstrations and the scores corresponding to the perfect demonstration are greater than or equal to the scores needed to satisfy the termination condition.

Table 1: Performance of different methods in different OpenAI Gym environments.

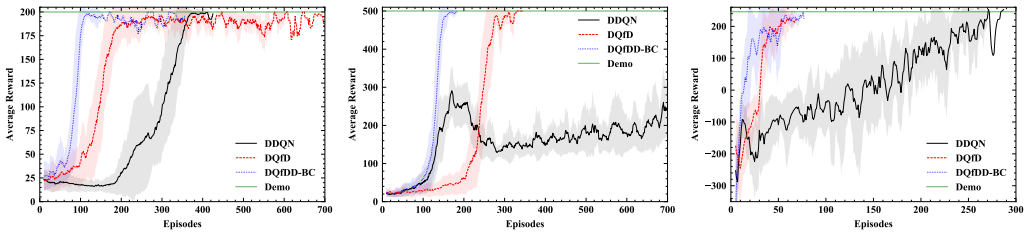
ENV	DDQN	Perfect demostration			Imperfect demonstration		
		Demo	DQfD	DQfDD-BC	Demo	DQfD	DQfDD-BC
	Average stop episode						
CartPole-v0	369±41	200	398±172	**183**±87	-	-	-
CartPole-v1	1000±0	500	302±27	**170**±9	357	1000±0	**214**±16
Acrobot-v1	198±123	-84	51±4	**49**±5	-238	261±136	**143**±11
LunarLander-v2	263±21	246	65±3	**65**±9	14	402±192	**119**±17
	Average training score for last 10 episodes						
CartPole-v0	199±0.6	200	196±3.52	**199**±1.5	-	-	-
CartPole-v1	175±23.3	500	**500**±0	499±1.9	357	329±77.94	**491**±2.1
Acrobot-v1	-97±5.3	-84	-95±4.0	**-91**±2.5	-238	-96±5.3	**-95**±7.8
LunarLander-v2	231±6.5	246	219±8.7	**223**±6.1	14	**234**±4.4	211±20.4

In all environments, our DQfDD-BC method requires the smallest number of episodes to stop training and obtains almost the highest final performance, which demonstrates the significant advantages. As the model may oscillate for a longer time in some runs, the average stop episode can not accurately measure the model's learning ability and we show more details of the CartPole environmets in the subsequent sections.

5.1 DQfDD-BC WITH PERFECT DEMONSTRATION

Compared with the DQfD algorithm, the proposed DQfDD-BC algorithm achieved better results in terms of convergence speed and learning capability. Fig. 2 shows the episode rewards of various methods in three gym environments. In CartPole-v0, the DQfD algorithm is close to convergence at about 200 episodes, while the DQfDD-BC algorithm converges at about 100 episodes to achieve the same performance level. The DQfDD-BC method is roughly twice as fast as the DQfD and it is also held in the CartPole-v1 environment. In the LunanrLander-v2, the DQfDD-BC also shows faster learning speed. In the Cartpole environments, the DQfD algorithm needs to oscillate longer

to satisfy the termination condition at the late stage of training, and higher final performance is obtained in the LunarLander-v2 environment.



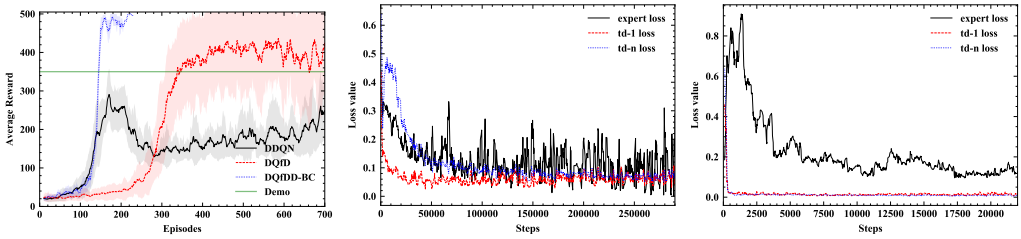
(a) CartPole-v0 (b) CartPole-v1 (c) LunarLander-v2

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

Although the DDQN method failed to meet the termination condition, both DQfD and DQfDD-BC can obtain full score, which proves that the demonstration do help to the learning ability of DRL agent. The DQfDD-BC algorithm retains the advantages of the DQfD algorithm and further improves the supervised loss function so that all self-generated data can obtain supervised losses, thus improving the learning speed.

5.2 DQfDD-BC WITH IMPERFECT DEMONSTRATIONS

We also validated the adaptability of the DQfDD-BC method through experiments with imperfect demonstrations. Since perfect demonstrations are difficult to define and acquire in many scenarios, it is more relevant to train the agent with imperfect demonstrations which are relatively better than the stochastic policy. 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 method outperforms DQfD by a significant margin in terms of convergence speed and final performance. By keeping the demonstrations updated, our DQfDD-BC method allows the optimized transitions to guide the agent and eventually converge to a better performance level.



(a) Average training scores (b) Losses of DQfD (c) Losses of DQfDD-BC

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

The proposed DQfDD-BC method can adapt to imperfect demonstrations well but the DDQN and DQfD fail to meet the experimental termination condition with imperfect demonstration. DQfD algorithm shows great dependence on the demonstration and ultimately converges to the vicinity of the demonstration. We believed that the imperfect demonstrations harm the performance of DQfD method and further explored the variation of each component of the complete loss function. 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 but the expert loss fluctuating over a wide range. This is partly due to the reason that the action output by the learned policy can be quite different from those sampled from the demonstrations. The agent is able to obtain better policy though the TD losses and provides an appropriate action. However, the supervised loss draws the learned policy near to the

policy represented by the imperfect demonstration and inapposite actions are provided to contruct the expert loss.

Additionally, the fixed imperfect demonstration is not immune to its own negative effects, resulting in the inability to converge to the highest score. Fig. 3 (c) illustrates that the expert loss of DQfDD-BC decreases more steadily and the agent achieves better final performance. The dynamic demonstration has the ability to modify the disadvantage of imperfect demonstration and expand the state-action space covered by the demonstration.

5.3 ABLATIONS

To figure out the importance of dynamic demonstrations and expert loss with BC model in the proposed DQfDD-BC method, we also test the method without dynamic demonstration or BC model in the CartPole-v1 environment. Fig. 4 shows the effect of integrating dynamic demonstrations and BC model to the original DQfD model. The figure clearly shows that when the dynamic demonstration is added to the DQfD algorithm (DQfDD), the model does not present a significant decrease in training speed, but the stability of the model is significantly improved in the later training period. The DQfDD can converge to the highest score and ended the training process in a short time while the DQfD needs to oscillate around the highest score for some more time. This indicates that by adding new samples to the demonstrations, especially in the later stages of training, a large number of high-performance transitions can improve the generalization ability and training stability of the model.

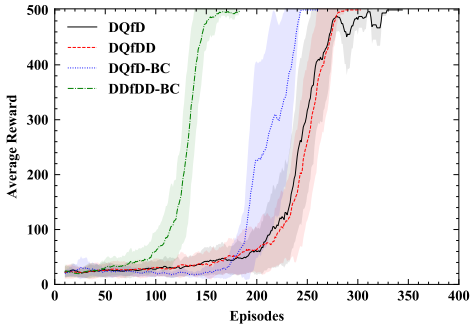


Figure 4: Ablation experimental results. The dynamic demonstration (DQfDD) and expert loss with a behavioral cloning model (DQfD-BC) are added to the original DQfD method.

In the case where the large margin expert loss is replaced by the loss generated by the BC model (DQfD-BC), the training speed of the model is significantly improved. Fig. 4 shows that 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 learning speed is improved by introducing a supervised model to generate the expert loss. The self-generated transitions can also be used to generate expert loss with the supervised model which means higher sample efficiency. The final DQfDD-BC model, which combins dynamic demonstrations and the expert loss with the BC model, simultaneously improves the learning speed and training stability and exhibits excellent performance.

6 CONCLUSION

In this work, we present an approach, namely the deep Q learning from dynamic demonstration integrated with a behavioral cloning (DQfDD-BC), to enhance the learning efficiency and quality of DQfD. The experimental results demonstrate that regardless of the degree of perfection of historical demonstrations, DQfDD-BC can significantly improve the learning efficiency compared with DQfD in generic environments. From an ablation experiment, both the dynamic demonstration and expert loss with the BC model showed their own potential to improve the learning quality.

As our approach provides a feasible solution to bridge the DRL methods and the decision-making problems in complex system environments, we will later apply our model to real-world settings, such as a large-scale urban traffic control system. While considering the process of recording human demonstrations, we will explore and improve our model for the ability of handling engineering issues in data sparsity, human diversity, and so on.

7 ACKNOWLEDGMENTS

We thank Dr. Peidong Xu, Dr. Xingyuan Dai and Dr. Junjie Wang and Prof. Jun Zhang for their advices and technical supports. We appreciate the implementation in `https://github.com/asdiijj/implementation-of-dqfd` for our reference. This study is supported by National Key R&D Program of China (2018AAA0101500, 2018AAA0101502).

REFERENCES

Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

Feryal Behbahani, Kyriacos Shiarlis, Xi Chen, Vitaly Kurin, Sudhanshu Kasewa, Ciprian Stirbu, João Gomes, Supratik Paul, Frans A Oliehoek, João Messias, et al. Learning from demonstration in the wild. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 775–781. IEEE, 2019.

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

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

Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on Robot Learning*, pp. 330–359, 2020.

Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E Taylor, and Ann Nowé. Reinforcement learning from demonstration through shaping. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

Andreas Bühler, Adrien Gaidon, Andrei Cramariuc, Rares Ambrus, Guy Rosman, and Wolfram Burgard. Driving through ghosts: Behavioral cloning with false positives. *arXiv preprint arXiv:2008.12969*, 2020.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 2094–2100, 2016.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Association for the Advancement of Artificial Intelligence*, 2018.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pp. 4565–4573, 2016.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

Bingyi Kang, Zequn Jie, and Jiashi Feng. Policy optimization with demonstrations. In *International Conference on Machine Learning*, pp. 2469–2478, 2018.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *arXiv preprint arXiv:2002.00444*, 2020.

Nathan O Lambert, Daniel S Drew, Joseph Yaconelli, Sergey Levine, Roberto Calandra, and Kristofer SJ Pister. Low-level control of a quadrotor with deep model-based reinforcement learning. *IEEE Robotics and Automation Letters*, 4(4):4224–4230, 2019.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Jan Matas, Stephen James, and Andrew J Davison. Sim-to-real reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pp. 734–743, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

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

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International Conference on Machine Learning*, pp. 3878–3887, 2018.

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

Bilal Piot, Matthieu Geist, and Olivier Pietquin. Boosted and reward-regularized classification for apprenticeship learning. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pp. 1249–1256. International Foundation for Autonomous Agents and Multiagent Systems, 2014b.

Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3, 2020.

Stéphane Ross, Geoffrey Gordon, and Drew 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*, pp. 627–635, 2011.

Stefan Schaal. Learning from demonstration. In *Advances in neural information processing systems*, pp. 1040–1046, 1997.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, and Marc Lanctot. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, and Adrian Bolton. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354, 2017.

Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. In *Advances in neural information processing systems*, pp. 7461–7472, 2018.

Halit Bener Suay, Tim Brys, Matthew E Taylor, and Sonia Chernova. Learning from demonstration for shaping through inverse reinforcement learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pp. 429–437, 2016.

Wen Sun, Arun Venkatraman, Geoffrey J Gordon, Byron Boots, and J Andrew Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *International Conference on Machine Learning*, pp. 3309–3318, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Brijen Thananjeyan, Ashwin Balakrishna, Ugo Rosolia, Felix Li, Rowan McAllister, Joseph E Gonzalez, Sergey Levine, Francesco Borrelli, and Ken Goldberg. Safety augmented value estimation from demonstrations (saved): Safe deep model-based rl for sparse cost robotic tasks. *IEEE Robotics and Automation Letters*, 5(2):3612–3619, 2020.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

Pedro A Tsividis, Thomas Pouncy, Jacqueline L Xu, Joshua B Tenenbaum, and Samuel J Gershman. Human learning in atari. 2017.

Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

Borui Wang, Ehsan Adeli, Hsu-kuang Chiu, De-An Huang, and Juan Carlos Niebles. Imitation learning for human pose prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7124–7133, 2019.

Yuanhao Xiong, Guanjie Zheng, Kai Xu, and Zhenhui Li. Learning traffic signal control from demonstrations. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 2289–2292, 2019.

Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8. IEEE, 2018.

A APPENDIX

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 buffer_size=1000000, the learning_rate=0.00001, the learning rate of supervised model is 0.001, and the discount factor $\gamma = 0.95$ in the CartPole environments (0.99 for other environments). We find that different random seeds have different performance and we choose $[100, 200, 300, ..., 1000]$ as the random seed list. In the $\epsilon - greedy$ policy, ϵ linearly decreases from a maximum value 0.99 to a minimum value 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. The parameter β controls the amount of importance sampling and it is annealed linearly from β_0 to 1. In DQfD and DQfDD-BC, λ decides how to combine different losses. In the experiments, $\lambda_1 = \lambda_2 = 0.5, \lambda_3 = 0.001$ and the L2 regularization is provided by the optimizer in the pytorch. The large margin in DQfD is 1.

Model architecture Since the state of experiment environments are vectors, fully-connected layers are appropriate for the requirements and they are leveraged to construct the Q network and the BC model. In the BC model, the hidden unit number of all three fully-connected layers are $[64, 24, 12]$ and $LeakyRelu$ is selected as the activation function. The Q network which has $[512, 512, 256, 128]$ hidden units in every hidden layer and the activation function is $Relu$.

Termination condition In the experiments, the termination condition was as following. Within 1000 episodes, 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). The models were considered to be successfully converged if the termination conditions are satisfied. Each method is repeated 10 times with different random seeds.