IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 12 January 2024, accepted 29 January 2024, date of publication 1 February 2024, date of current version 9 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3361030

RESEARCH ARTICLE

Combined Constraint on Behavior Cloning and Discriminator in Offline Reinforcement Learning

SHUNYA KIDERA, KOSUKE SHINTANI, TOI TSUNEDA, AND SATOSHI YAMANE
Electrical Engineering Department, Kanazawa University, Kanazawa 920-1192, Japan
Corresponding author: Shunya Kidera (skidera@csl.ec.t.kanazawa-u.ac.jp)

ABSTRACT In recent years, reinforcement learning (RL) has received a lot of attention because we can automatically learn optimal behavioral policies. However, since RL acquires the policy by repeatedly interacting with the environment, it is difficult to learn about realistic tasks. In recent years, there has been a lot of research on offline RL (batch RL), which does not need to interact with the environment, but learns from the accumulated experience prepared in advance. Learning does not work by applying common RL methods directly to offline RL because of a problem called distributional shift. Methods to suppress distributional shift have been actively studied in offline RL. In this study, we propose a new offline RL algorithm that adds constraints from discriminators used in Generative Adversarial Networks to the offline RL method called TD3+BC. We compare and validate the proposed method with existing methods using a benchmark for 3D robot control simulation. In TD3+BC, the constraint was tightened to suppress distribution shift, but a challenge arose when the quality of the dataset was poor, leading to difficulties in successful learning. The proposed approach addresses this issue by incorporating features to mitigate distribution shift while introducing new constraints to enable learning that is not solely dependent on the dataset's quality. This innovative strategy aims to improve accuracy even in cases where the dataset exhibits poor characteristics.

INDEX TERMS Reinforcement learning, offline reinforcement learning, generative adversarial networks, discriminator, robot control.

I. INTRODUCTION

Recent advances in reinforcement learning (RL), especially in combination with expressive deep network function approximators, have produced promising results in areas such as video games [1], robotics [2] and recommendation systems [3].

However, one of the factors that prevents reinforcement learning from becoming popular for real-world tasks is the fact that reinforcement learning algorithms provide a fundamentally online learning paradigm. The process of reinforcement learning involves repeated interactions with the environment to gather experience, which is then used to improve the policy [4]. In many cases, such online interactions can be costly (automated driving, robotic control) or dangerous (automated driving, healthcare) to collect data. [5]

Due to such cost, safety, and ethical issues, offline RL (batch RL) [5] is expected nowadays. Offline RL learns policies using only pre-prepared experience data. The appeal of a complete offline reinforcement learning framework is enormous. Just as supervised machine learning such as image classifiers and speech recognition engines have enabled us to turn data into general-purpose and powerful pattern recognizers, offline reinforcement learning with powerful function approximations has the potential to turn data into general-purpose and powerful decision-making engines.

On the other hand, offline reinforcement learning has the challenge of not learning policies well if the data is biased. In fact, most off-policy RL algorithms can be applied in an offline environment, but they tend to fail to update policies correctly and degrade performance because the agent cannot

The associate editor coordinating the review of this manuscript and approving it for publication was Rongbo Zhu.

successfully estimate the values of state-action pairs that are not included in the data set [6]. Solutions to this problem revolve around the idea that the policy to be learned should be close to the policy that generated the data, and various algorithms (batch constraints [6], KL-control [7], behavioral-regularization [8], policy constraints [5], etc.) have been studied depending on how this "closeness" is selected and implemented.

In this paper, we propose a method *Behavior Cloning and Discriminator Blend Regularization* (BDB) to add constraints using the discriminator used in Generative Adversarial Networks (GANs) [10] to a simple and high-performing offline RL algorithm called TD3+BC [9].

In recent training papers on offline reinforcement learning [11], various approaches to address distribution shift have been discussed. Among them, a particularly powerful and straightforward method is an enhancement of TD3+BC. The paper highlights the importance of uncertainty estimation in handling distribution shift. In line with the latest research trends, the proposed method incorporates adjustments based on uncertainty, making it a contemporary proposal. Using the policy approximator as a generator, the discriminator is made to compete with the policy approximator, so that the distance between the training and data generation strategies is not too far apart. The goal is to learn better policies even when the performance of the dataset is low (low dataset quality) by allowing policies that are "similar" to the data generation policies. In addition, we propose a method to adjust the ratio of Behavior Cloning [8] and GANs discriminators in the normalization term according to the learning situation rather than heuristically.

We evaluate the proposed method on Mujoco [12], a benchmark for 3D robot control, and D4RL, a dataset for offline RL [13]. The experimental results show the effectiveness of the proposed method.

II. BACKGROUND

A. REINFORCEMENT LEARNING

We target reinforcement learning based on the assumption of a general Markov Decision Process. The goal of reinforcement learning is to learn the optimal policy π. Reinforcement learning algorithms can be broadly divided into model-free algorithms and model-based algorithms [14]. In the model-free algorithms, the agent learns the policy based on the experience it gathers from interacting with the environment.

In reinforcement learning, the subject of the action is called the agent. At time t, the state that the agent observes from the environment is s_t, and the action that the agent performs on that state is a_t. Return obtained as the result of performing action a_t on state s_t is defined as reward r_t. The goal of reinforcement learning is to learn the policy π that maximizes the sum of rewards G_t.

$$G_t = \sum_{\tau=0}^{\infty} \gamma^{\tau} R_{t+1+\tau} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \quad (1)$$

γ $(0 < \gamma \leq 1)$ is used to account for the uncertainty of future rewards. The expected value that is considered conditional on state s_t in revenue G_t, is called state value $V(s_t)$. $V(s_t)$ indicates the goodness of the state of how much return G_t the agent will ultimately receive from that s_t if it follows policy π.

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left\{G_{t+1} | S_t = s\right\} \quad (2)$$

Also, action value $Q(s_t, a_t)$ indicates the goodness or badness of action a_t in state s_t.

$$Q^{\pi}(s, a) = \mathbb{E}^{\pi}\left[G_{t+1} | S_t = s, A_t = a\right] \quad (3)$$

The relationship between $Q(s_t, a_t)$ and $V(s_t)$ can be expressed by the following equations.

$$Q^{\pi}(s, a) = \sum_{s'} P^{a}_{ss'}\left[r + \gamma V^{\pi}\left(s'\right)\right] \quad (4)$$

$$V^{\pi}(s) = \sum_{a} \pi(s, a) Q^{\pi}(s, a) \quad (5)$$

where $P^{a}_{ss'} = \Pr\left\{s_{t+1} = s' \mid s_t = s, a_t = a\right\}$ is state transition probability. If $Q(s_t, a_t)$ can be calculated accurately, continuing to perform a_t with the highest $Q(s_t, a_t)$ is equivalent to finding the optimal π^*.

There are two main ways to learn to estimate $Q(s_t, a_t)$: Monte Carlo and TD learning (temporal difference learning) [15], [16]. Monte Carlo method is a simple method, but is not often used because of the potentially large variance in reward and the difficulty of resetting to a particular state when estimating while interacting with environment. TD learning is a way to deal with such Monte Carlo problems, and is one of the most important ideas in reinforcement learning. A feature of TD learning is use of bootstrapping, a method that uses current estimate of value function as target value during training.

Q-learning [17] of value-based reinforcement learning is a type of TD learning that updates the action value with equation (6).

$$Q\left(S_t, A_t\right) \leftarrow Q\left(S_t, A_t\right) + \alpha\left(R_{t+1} + \gamma \max_{a' \in A(s')} Q\left(S_{t+1}, a'\right) - Q\left(S_t, A_t\right)\right) \quad (6)$$

$R_{t+1} + \gamma \max_{a' \in A(s')} Q\left(S_{t+1}, a'\right)$ is more reliable than $Q(S_t, A_t)$ because it is the actual reward rather than an estimate of one step, so it can be treated as supervisor data. α is a hyperparameter called the learning rate.

Policy Gradient [18] is not a method of determining the value of actions and then selecting the action that maximizes that value, as in Q-learning, but rather it is a method of solving the reinforcement learning problem by optimizing a stochastic model $\pi_\theta(a|s)$ parameterized by some parameter vector θ. The goal of Policy Gradient is to find the parameters of the optimal policy parameter θ^* such that it maximizes return G. To find the optimal parameters, an evaluation method is needed. Policy Gradient uses equation (7) to determine whether the policy π_θ is good or bad.

$$\begin{aligned} J(\theta) &= \mathbb{E}\left[G_0 \mid S_0 = s_0\right] \\ &= \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t \mid S_0 = s_0\right] \end{aligned} \tag{7}$$

Update parameter θ in the direction of the gradient obtained by differentiating as in equation (10) using J.

$$\theta^{t+1} = \theta^t + \eta \nabla_\theta J(\theta) \tag{8}$$

The gradient of J can be transformed using action value function $Q(s, a)$ as follows.

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\frac{\partial \pi_\theta(a \mid s)}{\partial \theta} \frac{1}{\pi_\theta(a \mid s)} Q^\pi(s, a)\right] \\ &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a \mid s) Q^\pi(s, a)\right] \end{aligned} \tag{9}$$

Actor Critic [14], [19] is a reinforcement learning method that combines Q-learning and Policy Gradient. In Policy Gradient, $Q(s, a)$ is simply the sum of rewards at a discount rate, but in Actor Critic, the Q function and policy are learned simultaneously. The Actor (policy) learns to maximize $Q(s, a)$.

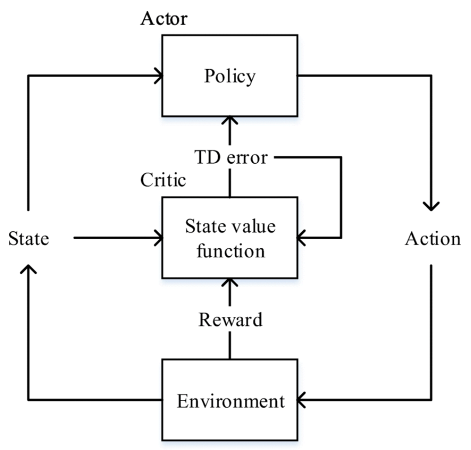


FIGURE 1. Actor critic framework.

B. OFFLINE RL

As mentioned in section I, general reinforcement learning, which learns by interacting with the environment, is difficult to adapt to real-world tasks (automated driving, robotics, etc.) due to safety, cost issues, and ethical issues. Offline RL (Batch RL) [5] is a new reinforcement learning framework proposed to address such problems.

In deep reinforcement learning, which uses neural networks for function approximation such as DQN [20], it stores agent's experience in Replay Buffer and uses it for training. Offline RL uses pre-prepared experience for learning, rather than interacting with the environment to collect experience in the Replay Buffer. It solves the problems of online RL, such as cost and security due to interaction with the environment, by using pre-prepared datasets $\mathcal{D}$. However, offline RL is not successful simply by stopping interaction from online RL and training on a prepared the datasets. Offline RL has to solve the problem of "distributional shift".

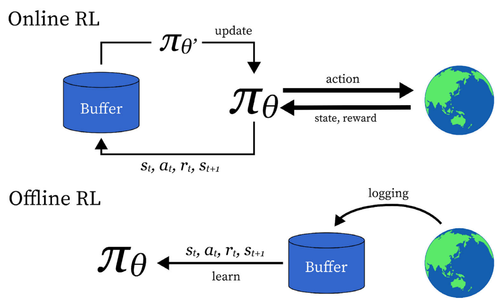


FIGURE 2. Pictorial illustration of classic online reinforcement learning and offline reinforcement learning.

Distributional shift is one of the major problems encountered in offline RL. There are two types of distributional shift: policy and state.

Distributional shift of policy is the condition that the probability of action selection between the behavioral policy π_β collecting (s_t, a_t) in the dataset $\mathcal{D}$ and the current policy π_θ under training deviates significantly. Distributional shift of state is that distributional shift of policy causes current policy π_{theta} to refer to unknown domains with less data (s_t, a_t) in $\mathcal{D}$. A domain with little data has high uncertainty and high error in predicting the value function and expected reward due to out-of-ditribution. In particular, when value function erroneously overestimates value, current policy π_θ selects actions to maximize the value and therefore selects different actions tha behavior policy π_β. Then there is a vicious cycle of more distributional shift of policy, which in turn makes distributional shift of state even worse.

With online RL, even if the value function is incorrectly estimated and π_θ learns wrong actions, the agent can receive feedback from interacting with the environment and correct the error. Offline reinforcement learning does not allow for additional data, so once it falls into out-of-ditribution, it is difficult to correct.

C. GENERATIVE ADVERSARIAL NETWORKS

Generative Adversarial Networks (GANs) [10] involve two networks, a generator and a discriminator, competing to achieve high-accuracy outputs. Machine learning models, based on probability concepts, are broadly categorized into discriminative and generative models. In equation (10), neural networks serve for both generation and discrimination, with the discriminator aiming to correctly label real or fake instances, while the generator aims to produce outputs that deceive the discriminator into believing they are real. This adversarial learning process searches for optimal discriminator (D) and generator (G), updating them alternately to advance the learning process.

$$\begin{aligned} \min_G \max_D V(D, G) =& \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}\left[logD(x)\right] \\ &+ \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}\left[log(1 - D(G(\mathbf{z})))\right] \end{aligned} \tag{10}$$

GANs have a high affinity to reinforcement learning. When it is difficult to design reward function $R(s, t)$ for the environment, it can be used in inverse reinforcement learning (IRL) [21] to learn R using states and its actions as input [22], or it can be incorporated into an imitation learning framework, such as GAIL [24].

III. RELATED WORK

Here, we describe how existing studies have successfully used offline reinforcement learning.

A. TD3+BC

TD3+BC is an offline RL proposed by Fujimoto et al [9]. Based on an online RL algorithm called *Twin Delayed DDPG* (TD3) [25], it makes two simple changes for successful offline RL.

First, the algorithm adds Behavior cloning regularization term to the standard policy π update step in TD3 to strengthen the policy to prioritize the behaviors in the dataset $\mathcal{D}$.

$$\pi = \underset{\pi}{\operatorname{argmax}} \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\lambda Q(s, \pi(s)) - (\pi(s) - a)^2 \right] \tag{11}$$

Behavior Cloning [26], [27] is a form of imitation learning, but the framework is supervised learning, not reinforcement learning. For each pair of (s_t, a_t) in the dataset, s_t is input and a_t is supervised data. The model (policy) is trained to maximize the probability of correct data for the input.

While the selection of λ in equation (11) is ultimately just a hyperparameter, the balance between RL (maximizing Q) and imitation (minimizing the BC term) is adjusted by the scale of Q. Given the dataset of N transitions (s_i, a_i), Fujimoto et al. define the scalar λ as:

$$\lambda = \frac{\alpha}{\frac{1}{N} \sum_{(s_i,a_i)} |Q(s_i, a_i)|} \tag{12}$$

And second, to normalize the characteristics of each state in the provided dataset. Normalize the states using the mean μ_i and standard deviation σ_i of the states in the dataset as in equation (13). (ϵ is a constant value to avoid division by zero.)

$$s_i = \frac{s_i - \mu_i}{\sigma_i + \epsilon} \tag{13}$$

TD3+BC is a simple modification of TD3, but offers high performance.

B. CQL

Conservative Q-learning (CQL) [28] solves the problem of offline RL with a different approach from the major methods of constraining policy and using uncertainty in value function Q. CQL suppresses overestimation of value by conservatively estimating value for highly uncertain out-of-distribution state and action pairs.

CQL learns to estimate Q value conservatively (small) for pairs (s, a) that are not in the dataset. In this way, the policy will select (s, a) that is of high value (not conservatively estimated) in the data set and prevent distributional shifts.

C. BEAR

Bootstrapping Error Accumulation Reduction (BEAR) [29] is one of the offline Actor-Critic based on TD3 or SAC [30] frameworks. BEAR uses the variance (uncertainty) of $Q(s, a)$ to suppress the bootstrapping error of the Q function.

In BEAR, ensemble Q functions $\{Q_{\theta_i}\}_{i=1}^{K}$ are updated by calculating the uncertainty from the maximum and minimum using the following equation for $Q(s, a)$ of p actions $\{a_i \sim \pi_{\phi'}(\cdot \mid s')\}_{i=1}^{p}$ sampled by the currently learning policy π_θ.

$$\begin{aligned} &\forall i, \theta_i \leftarrow \arg\min_{\theta_i} (Q_{\theta_i}(s, a) - (r + \gamma y(s, a)))^2 \\ &y(s, a) := \max_{a_i} [\lambda \min_{j=1,\ldots,K} Q_{\theta_j'}(s', a_i) \\ &\qquad + (1 - \lambda) \max_{j=1,\ldots,K} Q_{\theta_j'}(s', a_i)] \end{aligned} \tag{14}$$

IV. PROPOSED METHOD

In this section, we mention the proposed method. TD3+BC has a simple architecture and is capable of fast learning (in terms of execution speed on a computer). However, TD3+BC is strongly dependent on the performance of the dataset $\mathcal{D}$ because the method of constraints to suppress the distribution shift is simple Behavior Cloning. In other words, learning TD3+BC is likely to depend on the π_β used when creating the $\mathcal{D}$ data set for training purposes. The goal of offline RL is not to imitate π_β, but to learn a complete policy π^* that allows optimal action selection even in real environments, just from a given dataset $\mathcal{D}$.

In this context, we consider the constraint imposed by BC, which only allows actions within the dataset, to be too strong. Therefore, we propose *Behavior Cloning and Discriminator Blend Regularization* (BDB), which aims to obtain a better policy by making the constraint more flexible by using a discriminator D in the GAN and allowing actions not included in the dataset $\mathcal{D}$ BDB is a new approach to the data set regularization.

$$\begin{aligned} \pi = &\underset{\pi}{\operatorname{argmax}} \mathbb{E}_{(s,a)\sim\mathcal{D}} [\lambda Q(s, \pi(s)) \\ &-(1-\beta)(\pi(s) - a)^2 + \beta log(D(s, \pi(s)))] \end{aligned} \tag{15}$$

In BDB, the normalization, which was performed only by BC, is partially performed by discriminator D in GANs. In TD3+BC, the loss of policy π_θ becomes large when actions outside the data set are selected during policy network learning. Therefore, BDB learns to allow actions that are outside the distribution of the data set $\mathcal{D}$, but that the discriminator perceives to be in the distribution, similar to the action $a_{\mathcal{D}}$ in the dataset. The discriminator is trained to take state s and action a as input and to determine whether the action is in the dataset $a_{\mathcal{D}}$ or estimated by π_θ, where π_θ playing the role of generator G of GANs.

Equation (15) presents the update formula for the proposed policy. When compared to the policy update formula for

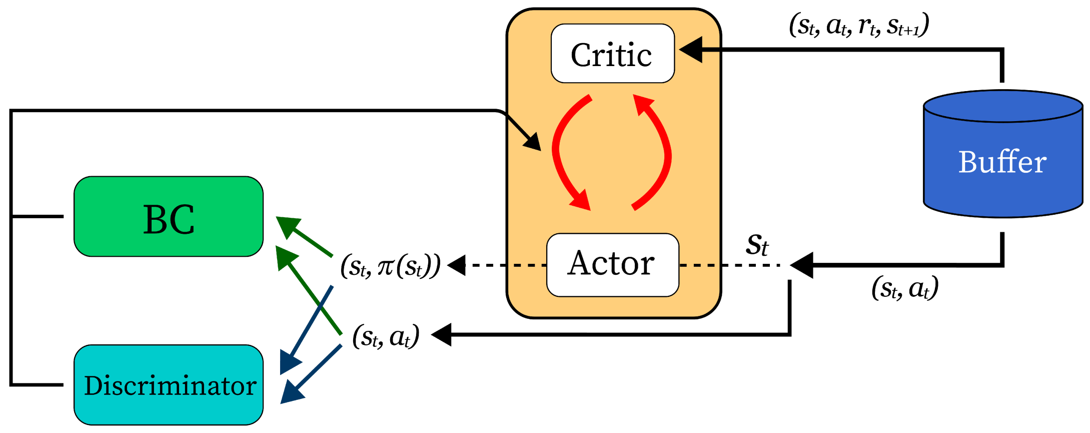


FIGURE 3. Pictorial illustration of our proposed method.

TD3+BC, the first term remains identical, representing the value of the Q-function. In the second term, the weight of the constraint is adjusted by multiplying $1 - \beta$, which penalizes actions outside the dataset. The third term involves the GANs [10] discriminator determining whether the action is within the dataset. Similar to the GANs discriminator in the cited references, it outputs a value in the range of 0 to 1. To adjust this value, it is multiplied by β and used as an increment to the action evaluation. Collectively, the policy learns, considering all these components, which actions are preferable in a mechanism that evaluates and adjusts the policy.

In this proposed method, as β increases, the constraint imposed by TD3+BC on actions outside the dataset decreases. However, due to the evaluation by the GAN discriminator, actions that are close to those within the dataset are positively assessed. This allows the model to evaluate and incorporate actions that are similar to those within the dataset, even if they are outside the dataset. Because learning can be conducted using actions that are similar to those in the dataset but outside of it, the evaluation is less likely to deviate significantly, leading to a reduced occurrence of distribution shifts.

The learning flow of BDB is shown in Algorithm 1.

Algorithm 1 Proposed method

```
Require: dataset $\mathcal{D}$
1: Initialize Q-function $Q_\theta$, policy $\pi_\phi$ and discriminator $D_\omega$
2: for step in $1, \cdots, N$ do
3:     $s_t, a_t, s_{t+1}, r_t$ from sampling $\mathcal{D}$
4:     Calculate TD error and train the Q-function $Q_\theta$ using loss $L_Q$
5:     if step is policy update step then
6:         Update the discriminator parameters $\omega$
               $\mathbb{E}[\log D(s_t, a_t)] + \mathbb{E}[\log(1 - D(s_t, \pi(s_t)))]$   (16)
7:         Train the policy using $L_\pi$ on objective from equation (15)
               $L_\pi = -\lambda Q(s_t, \pi(s_t)) + \beta(-log(D(s_t, \pi(s_t)))) + (1-\beta)(\pi(s_t) - a_t)^2$   (17)
8:     end if
9: end for
```

The first term in equation (16) considers actions within the dataset as real data, while the second term trains the discriminator by treating actions chosen by the current policy as data generated by GANs. In equation (17), the objective function L_π is utilized to learn the policy π, aiming to minimize it for policy learning.

We also propose that the parameter β.

In our proposed approach, the ratio of behavior cloning (BC) and discriminator constraints is not a fixed heuristic value but a variable method depending on the learning conditions. The Behavior Cloning and Discriminator Blend Regularization (BDB) method allows actions outside the dataset, making it less constrained than TD3+BC. However, this flexibility may lead to distribution shifts. To address this, we introduce a variable parameter, β, which adjusts the balance between BC and discriminator constraints. The uncertainty in the Q function is utilized to determine β. BDB estimates uncertainty by sampling Q values from the variance of Q(s, a) similar to BEAR. When distribution shift occurs, the increased uncertainty in Q(s, a) results in a decreased β value, tightening the BC constraints. Conversely, when the Q function accurately estimates Q(s, a), the proportion of constraints on the discriminator increases, facilitating the learning of behaviors outside the dataset.

We define the adjustment of parameter β as follows Algorithm 2.

In Algorithm 2, the determination of the variable β during the experiment is described. Initially, several Q values are sampled. The difference between the maximum and minimum values is defined as uncertainty, and its

TABLE 1. Average normalized score over the final 10 evaluations and 5 seeds.The table shows the results in our experiment. We compared different task and data set combinations. We highlight those with relatively high scores within the same data set.

		TD3+BC	CQL	$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.75$	Adjust β
random	hopper-v0	$10.97 \pm 0.397 \times 10^{-2}$	$10.40 \pm 0.064 \times 10^{-2}$	$11.11 \pm 0.255 \times 10^{-2}$	$11.34 \pm 0.243 \times 10^{-2}$	$11.82 \pm 0.664 \times 10^{-2}$	$12.33 \pm 1.316 \times 10^{-2}$
	walker2d-v0	0.70 ± 0.53	6.97 ± 0.35	1.00 ± 1.47	4.43 ± 1.24	1.69 ± 1.13	2.61 ± 1.03
	halfcheetah-v0	$11.74 \pm 5.417 \times 10^{-2}$	$21.61 \pm 7.365 \times 10^{-2}$	$14.28 \pm 7.382 \times 10^{-2}$	$16.53 \pm 3.476 \times 10^{-2}$	$19.89 \pm 4.892 \times 10^{-2}$	$13.68 \pm 5.747 \times 10^{-2}$
medium replay	hopper-v0	$31.04 \pm 3.347 \times 10^{-2}$	$28.48 \pm 1.737 \times 10^{-2}$	$32.26 \pm 3.736 \times 10^{-2}$	$33.17 \pm 4.625 \times 10^{-2}$	$36.26 \pm 6.785 \times 10^{-2}$	$31.30 \pm 5.367 \times 10^{-2}$
	walker2d-v0	20.77 ± 1.22	17.36 ± 1.40	21.60 ± 1.71	8.70 ± 5.57	11.05 ± 1.68	25.34 ± 2.69
	halfcheetah-v0	$43.29 \pm 0.320 \times 10^{-2}$	$42.38 \pm 1.348 \times 10^{-2}$	$43.74 \pm 1.110 \times 10^{-2}$	$44.13 \pm 1.760 \times 10^{-2}$	$45.26 \pm 0.999 \times 10^{-2}$	$43.44 \pm 0.450 \times 10^{-2}$
medium	hopper-v0	99.90 ± 0.01	44.91 ± 1.43	99.90 ± 0.02	100.07 ± 0.00	99.01 ± 0.37	100.16 ± 0.02
	walker2d-v0	77.28 ± 0.03	58.91 ± 0.07	79.07 ± 0.01	33.96 ± 0.98	16.99 ± 1.75	78.57 ± 0.02
	halfcheetah-v0	$43.01 \pm 0.685 \times 10^{-2}$	$38.78 \pm 0.587 \times 10^{-2}$	$43.85 \pm 0.935 \times 10^{-2}$	$44.47 \pm 0.557 \times 10^{-2}$	$46.14 \pm 1.080 \times 10^{-2}$	$43.27 \pm 0.909 \times 10^{-2}$
medium expert	hopper-v0	$111.48 \pm 1.066 \times 10^{-2}$	$111.71 \pm 0.293 \times 10^{-2}$	$111.92 \pm 0.219 \times 10^{-2}$	$111.97 \pm 0.200 \times 10^{-2}$	$112.19 \pm 0.230 \times 10^{-2}$	$112.15 \pm 0.074 \times 10^{-2}$
	walker2d-v0	88.75 ± 2.54	81.64 ± 0.34	97.89 ± 0.38	87.76 ± 1.88	98.25 ± 0.54	102.19 ± 0.27
	halfcheetah-v0	95.06 ± 0.25	32.25 ± 2.52	90.55 ± 0.79	84.66 ± 0.56	52.18 ± 1.29	95.93 ± 0.29
expert	hopper-v0	112.13 ± 0.01	112.15 ± 0.03	112.34 ± 0.01	7.96 ± 2.22	10.43 ± 3.69	112.27 ± 0.01
	walker2d-v0	$106.10 \pm 0.550 \times 10^{-2}$	$107.88 \pm 2.552 \times 10^{-2}$	$102.30 \pm 2.712 \times 10^{-2}$	$102.98 \pm 2.680 \times 10^{-2}$	$96.18 \pm 9.018 \times 10^{-2}$	$101.91 \pm 5.984 \times 10^{-2}$
	halfcheetah-v0	105.73 ± 0.13	80.91 ± 4.97	105.28 ± 0.11	102.40 ± 0.07	92.18 ± 0.10	105.79 ± 0.07

TABLE 2. Average similar score over the final 10 evaluations and 5 seeds.This score represents the percentage of similarity between the action in the dataset based on the current Q-value and the action taken according to the proposed method. We highlight the highest normalized score within the same dataset in this table, in conjunction with TABLE 1.

		TD3+BC($\beta = 0.00$)	$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.75$
random	hopper-v0	$98.20 \pm 1.949 \times 10^{-3}$	$97.98 \pm 2.330 \times 10^{-3}$	$98.05 \pm 4.364 \times 10^{-3}$	$97.81 \pm 1.956 \times 10^{-3}$
	walker2d-v0	$52.66 \pm 8.644 \times 10^{-2}$	46.44 ± 1.20	34.69 ± 2.90	22.89 ± 4.05
	halfcheetah-v0	$71.09 \pm 2.142 \times 10^{-2}$	$70.10 \pm 4.043 \times 10^{-3}$	$67.11 \pm 2.885 \times 10^{-2}$	$58.28 \pm 4.269 \times 10^{-2}$
medium replay	hopper-v0	$97.81 \pm 1.029 \times 10^{-2}$	$98.09 \pm 1.456 \times 10^{-3}$	$97.58 \pm 1.198 \times 10^{-2}$	$96.88 \pm 1.169 \times 10^{-2}$
	walker2d-v0	$88.75 \pm 3.841 \times 10^{-2}$	$85.96 \pm 1.556 \times 10^{-2}$	$80.94 \pm 4.469 \times 10^{-2}$	$71.72 \pm 9.170 \times 10^{-2}$
	halfcheetah-v0	$93.28 \pm 1.971 \times 10^{-2}$	$92.23 \pm 4.749 \times 10^{-3}$	$92.11 \pm 1.618 \times 10^{-2}$	$88.98 \pm 1.284 \times 10^{-2}$
medium	hopper-v0	$99.53 \pm 4.577 \times 10^{-3}$	$99.02 \pm 1.806 \times 10^{-3}$	$99.61 \pm 4.960 \times 10^{-3}$	$99.14 \pm 4.595 \times 10^{-3}$
	walker2d-v0	$97.03 \pm 1.068 \times 10^{-2}$	$96.38 \pm 3.146 \times 10^{-3}$	$95.62 \pm 1.448 \times 10^{-2}$	73.28 ± 3.93
	halfcheetah-v0	$98.83 \pm 2.500 \times 10^{-3}$	$99.04 \pm 9.199 \times 10^{-4}$	$98.67 \pm 1.939 \times 10^{-3}$	$98.12 \pm 2.979 \times 10^{-3}$
medium expert	hopper-v0	$99.69 \pm 2.932 \times 10^{-3}$	$99.12 \pm 1.131 \times 10^{-3}$	$99.38 \pm 1.926 \times 10^{-3}$	$99.06 \pm 4.021 \times 10^{-3}$
	walker2d-v0	$96.41 \pm 7.857 \times 10^{-3}$	$94.38 \pm 4.955 \times 10^{-3}$	$93.91 \pm 5.643 \times 10^{-3}$	$91.88 \pm 4.958 \times 10^{-3}$
	halfcheetah-v0	$97.34 \pm 6.421 \times 10^{-3}$	$97.61 \pm 3.206 \times 10^{-3}$	$97.11 \pm 4.102 \times 10^{-3}$	$96.02 \pm 1.594 \times 10^{-2}$
expert	hopper-v0	$99.61 \pm 3.508 \times 10^{-3}$	$99.38 \pm 3.542 \times 10^{-3}$	$98.20 \pm 1.085 \times 10^{-2}$	59.06 ± 4.76
	walker2d-v0	$97.89 \pm 4.069 \times 10^{-3}$	$97.45 \pm 2.172 \times 10^{-3}$	$96.80 \pm 9.343 \times 10^{-3}$	$95.47 \pm 9.543 \times 10^{-3}$
	halfcheetah-v0	$96.64 \pm 1.367 \times 10^{-2}$	$97.68 \pm 1.612 \times 10^{-3}$	$97.11 \pm 1.209 \times 10^{-2}$	$96.72 \pm 1.102 \times 10^{-2}$

Algorithm 2 Adjustment of parameter β

```
Require: dataset $\mathcal{D}$
1: sampling $s$ from $\mathcal{D}$
2: $a = \pi(s)$
3: Unc $:= \max_{i=1,\cdots,M} Q_i(s,a) - \min_{i=1,\cdots,M} Q_i(s,a)$
4: $\beta = \min(1, \frac{1}{\text{Unc}})$
```

reciprocal is used as the value for β within the range of 1 or less. As a result, when a distribution shift occurs, the uncertainty in $Q(s, a)$ increases, leading to a decrease in the β value and strengthening the constraints of behavior cloning (BC). Conversely, when the Q function accurately estimates $Q(s, a)$, the proportion of constraints on the discriminator increases, making it easier to learn behaviors outside the dataset.

V. EXPERIMENTAL DETAILS

A. SOFTWARE

We used the following software versions:

- Python 3.7
- MuJoCo[i] [12] 2.00
- Gym [31] 0.18.0
- mujoco-py 2.0.2.13
- dm-control [32] 0.0.364896371
- D4RL [13] 1.1

All D4RL datasets use the v0 version.

[i]License information: https://www.roboti.us/license.html

We also used RLkit[ii] 0.2.1 to use the public CQL.

The parameters of the algorithm we used in our experiments are listed in Tables 3, 4, and 5.

B. HYPERPARAMETERS

The parameters of TD3+BC and the proposed method are shown in Table 3. TD3+BC and Our proposed method have the same architecture except for the use of GANs discriminator, so we used common parameters.

TABLE 3. TD3+BC and our proposed method hyperparameters.

	Parameter	Value
Hyperparameters	Optimizer	Adam [33]
	Critic learning rate	$3e-4$
	Actor learning rate	$3e-4$
	Mini-batch size	256
	Discount factor	0.99
	Target update rate	$5e-3$
	Policy noise	0.2
	Policy noise clipping	$(-0.5, 0.5)$
	Policy update frequency	2
	α	2.5
Architecture	Critic hidden dim	256
	Critic hidden layers	2
	Critic activation function	ReLU
	Actor hidden dim	256
	Actor hidden layers	2
	Actor activation function	ReLU

The parameters of the GANs discriminator used in the proposed method are shown in Table 4. Note that the output during training of the discriminator is through sigmoid function.

TABLE 4. Discriminator hyperparameters.

	Parameter	Value
Hyperparameters	Optimizer	Adam
	Learning rate	$3e-4$
	Mini-batch size	256
	Loss function	Binary Cross Entropy
Architecture	Hidden dim	64
	Hidden layers	2
	Activation function	Tanh

The parameters of the CQL used for the comparison experiments are shown in Table 5.

VI. COMPARISON EXPERIMENTS AND DISCUSSION

To evaluate the performance of the proposed method, we conducted a comparative experiment.

We chose Mujoco [12] for our experimental task. We used *Datasets for Deep Data-Driven Reinforcement Learning* (D4RL) [13] as the offline reinforcement learning datasets. D4RL is one of the famous libraries for offline RL benchmarking that integrates a dataset and an evaluation environment.

[ii]https://github.com/rail-berkeley/rlkit/tree/v0.2.1

TABLE 5. CQL method hyperparameters.

	Parameter	Value
SAC Hyperparameters	Optimizer	Adam
	Critic learning rate	$3e-4$
	Actor learning rate	$3e-5$
	Mini-batch size	256
	Discount factor	0.99
	Target update rate	$5e-3$
	Target entropy	$-1 \times$ Action Dim
	Entropy in Q target	False
Architecture	Critic hidden dim	256
	Critic hidden layers	3
	Critic activation function	ReLU
	Actor hidden dim	256
	Actor hidden layers	3
	Actor activation function	ReLU
CQL Hyperparameters	Lagrange	False
	α	10
	Pre-training steps	$40e3$
	Num sampled actions (during eval)	10
	Num sampled actions (logsumexp)	10

We compared the proposed method with some parameters β (β := 0.25, 0.50, 0.75, and adjust by uncertainty), TD3+BC[iii] which is the basis of the proposed method, and CQL.[iv] (The source code for each of these is available in the annotations.) We trained each algorithm for 1 million time steps and evaluated it every 5, 000 time steps. Each evaluation was done in 10 episodes. We experimented with 5 datasets for each of 3 tasks.

The table for each task shows the mean of the last 10 evaluations for the 5 seeds and the standard deviation between the seeds. The best scores are highlighted.

Figure 4 illustrates an example of experimental results represented in a bar graph. This experiment focuses on the hopper's random dataset, with the horizontal axis indicating steps and the vertical axis normalized scores mean. In this experiment, where learning is challenging due to the low quality of the dataset, the proposed method, BDB, visually exhibits better accuracy compared to CQL and TD3+BC.

To further examine the extent to which a change in the value of β would lead to a choice of behavior outside of the data set, we tested the percentage of behavior within the data set for all data sets for each value of β in the exact same environment. Here, we compared how similar the behavior selected in the datasets was to the behavior selected by the policy learned in the proposed method, and recorded the percentage of the same behavior chosen, defined as the similar score. The lower

[iii]https://github.com/sfujim/TD3_BC/tree/79c9fb923f3811feb6e01b8c1c61862225dfd942

[iv]https://github.com/aviralkumar2907/CQL/tree/d67dbe9cf5d2b96e3b462b6146f249b3d6569796

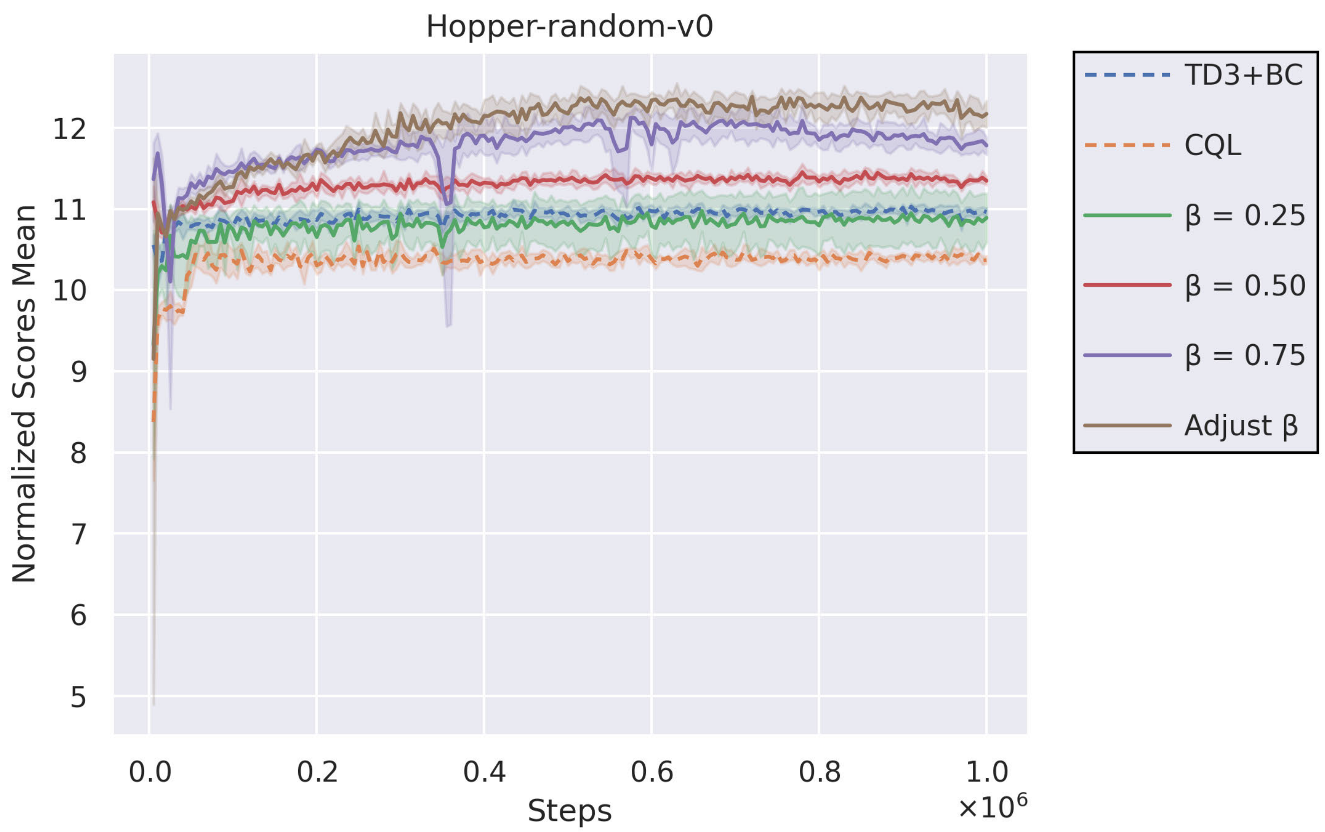


FIGURE 4. An example of comparison experiment(hopper-v0, random).

the similar score, the more acceptable the behavior outside the data set is.

Looking at Table 1, While CQL demonstrates the best performance in certain cases, the proposed method outperforms in many task and dataset pairings. In the case of the medium replay setting, particularly for walker2d-v0, a notable improvement of 22 percent in scores was observed compared to existing methods (TD3+BC, CQL). In scenarios where CQL, TD3+BC, and our proposed method all exhibit high performance, our proposed method demonstrates the lowest variance. We attribute this to the capability of our proposed method to learn about actions beyond the dataset, enabling the acquisition of robust policies for broader distributions. Furthermore, it can be observed from Table 2 that as the value of β increases, the proportion of selecting actions outside the dataset also increases. Moreover, it is evident that reducing the value of β as the quality of the dataset improves leads to better results. This is because for datasets of higher quality, effective learning can be achieved solely within the dataset, resulting in higher scores when β is reduced. On the other hand, for datasets of lower quality, allowing actions outside the dataset is considered beneficial for successful policy learning compared to learning policies only within the dataset. Integrating insights from Tables 1 and 2, it can be concluded that there are situations in certain high-quality datasets where efficient learning is predominantly achieved within the dataset. Conversely, in lower-quality datasets, allowing actions outside the dataset leads to more diverse behaviors, thereby enhancing learning efficiency. The flexibility of Adjust in adjusting based on the proficiency of learning has demonstrated the highest accuracy across numerous datasets. Additionally, increasing to allow actions outside the dataset enables learning a broader range of behaviors. However, this comes with the trade-off of loosening constraints imposed by behavior cloning, potentially increasing the risk of distribution shift. Therefore, determining how to set based on the characteristics of the dataset is deemed a crucial aspect for effective learning without succumbing to distribution shift.

VII. CONCLUSION AND FUTURE WORK

In this paper, we proposed an offline reinforcement learning, *Behavior Cloning and Discriminator Blend Regularization* (BDB). To suppress distributional shifts which is the problem of offline RL, the method uses GANs discriminators to constrain the training in addition to the existing method of behavior cloning. The performance of the proposed method was evaluated through comparison experiments with existing offline RL.

TD3+BC, which is the basis of the proposed method, is a simple modification of TD3 and is a successful algorithm for learning offline RL. However, since it only uses simple behavior cloning as a constraint to prevent distribution shifts from occurring, the learning performance is highly dependent on the given data set, and the learning is unstable. In our proposed method, BC constraint and GANs discriminator constraint are mixed in a certain ratio, so that even if the action is not in the dataset, it is acceptable if it is dataset-like.

In this paper, we also proposed a method in which the ratio of BC constraints to discriminator constraints is not heuristically determined, but is variable during training. Ideally, offline RL should train on (s, a) outside the dataset if there is no distribution shift, and if there is distribution shift, it should be constrained not to cause out-of-distribution. Therefore, in our proposed method, the uncertainty of Q(s, a) is used as an indicator of the presence or absence of distribution shift and is used to determine the ratio between BC and discriminator constraints.

In particular, it outperformed existing methods, albeit unstably, when the quality of the data set was low. This is likely due to more flexible learning constraints, which makes learning more unstable, but also allows the method to effectively learn behaviors that deviate from the distribution. In addition, when quality is high, the variation in scores is smaller than with existing methods. Therefore, the ratio of imitation learning is higher than that of discriminator, indicating that the proposed method takes advantage of the high quality of the dataset. Thus, we can say that the proposed method overcomes the shortcomings of TD3+BC. However, it cannot be said that variable β performs better than heuristic β in all experiments, and there may be a better way to determine β.

We would like to explore better ways to determine β in the future.

For example, We should use implementing a dynamically portioned β learned through the training process would result in a more adaptable network for any dataset. Also, the proposed method learns only by constraining the policy to avoid distributional deviations, but we would like to consider combining it with an algorithm that learns by constraining the Q function, such as CQL.

REFERENCES

[1] *AlphaStar: Mastering the Real-Time Strategy Game StarCraft II*. [Online]. Available: https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

[2] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation," 2018, *arXiv:1806.10293*.

[3] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 661–670.

[4] R. S. Sutton and A. G. Barto, *Introduction To Reinforcement Learning*, vol. 135. Cambridge, MA, USA: MIT Press, 1998.

[5] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," 2020, *arXiv:2005.01643*.

[6] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2052–2062.

[7] N. Jaques, A. Ghandeharioun, J. H. Shen, C. Ferguson, A. Lapedriza, N. Jones, S. Gu, and R. Picard, "Way off-policy batch deep reinforcement learning of implicit human preferences in dialog," 2019, *arXiv:1907.00456*.

[8] Y. Wu, G. Tucker, and O. Nachum, "Behavior regularized offline reinforcement learning," 2019, *arXiv:1911.11361*.

[9] S. Fujimoto and S. Shane Gu, "A minimalist approach to offline reinforcement learning," 2021, *arXiv:2106.06860*.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, in Polymers of Hexadromicon, vol. 27, J. Peters, Ed. New York, NY, USA: McGraw-Hill, 2014, pp. 15–64. [Online]. Available: http://www.bookref.com

[11] R. F. Prudencio, M. R. O. A. Maximo, and E. L. Colombini, "A survey on offline reinforcement learning: Taxonomy, review, and open problems," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 22, 2023, doi: 10.1109/TNNLS.2023.3250269.

[12] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5026–5033.

[13] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4RL: Datasets for deep data-driven reinforcement learning," 2020, *arXiv:2004.07219*.

[14] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[15] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, Aug. 1988.

[16] R. S. Sutton, "Temporal credit assignment in reinforcement learning," Ph.D. dissertation, Univ. Massachusetts, Amherst, MA, USA, 1984.

[17] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[18] H. Lyu, N. Sha, S. Qin, M. Yan, Y. Xie, and R. Wang, *Advances in Neural Information Processing Systems*, vol. 30, 2019.

[19] T. Degris, P. M. Pilarski, and R. S. Sutton, "Model-free reinforcement learning with continuous action in practice," in *Proc. IEEE Amer. Control Conf. (ACC)*, Montreal, QC, Canada, Oct. 2012, pp. 2177–2182.

[20] V. Mnih, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[21] S. Russell, "Learning agents for uncertain environments (extended abstract)," in *Proc. 11th Annu. Conf. Comput. Learn. Theory*. ACM Press, Jul. 1998, pp. 101–103.

[22] C. Finn, P. Christiano, P. Abbeel, and S. Levine, "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models," 2016, *arXiv:1611.03852*.

[23] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auto. Syst.*, vol. 57, no. 5, pp. 469–483, May 2009.

[24] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 4565–4573.

[25] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.

[26] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proc. 30th Int. Conf. Artif. Intell. Statist.*, 2010.

[27] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 627–635.

[28] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-learning for offline reinforcement learning," 2020, *arXiv:2006.04779*.

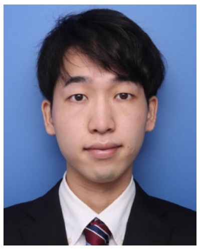
[29] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, "Stabilizing off-policy Q-learning via bootstrapping error reduction," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[30] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018.

[31] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," 2016, *arXiv:1606.01540*.

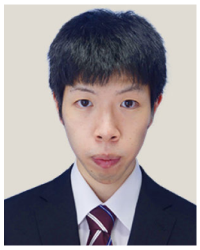
[32] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa, "Dm_control: Software and tasks for continuous control," *Softw. Impacts*, vol. 6, Nov. 2020, Art. no. 100022.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.



SHUNYA KIDERA received the B.S. degree from Kanazawa University, in 2021. He is currently pursuing the M.S. degree in reinforcement learning.

KOSUKE SHINTANI received the M.S. degree from Kanazawa University, in 2022. His research interest includes reinforcement learning.



TOI TSUNEDA received the M.S. degree from Kanazawa University, in 2021. His research interest includes reinforcement learning.

SATOSHI YAMANE received the B.S., M.S., and Ph.D. degrees from Kyoto University. He is currently a Professor with Kanazawa University. His research interests include formal verification of real-time and distributed computing.