IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 14, 2022, accepted May 16, 2022, date of publication May 26, 2022, date of current version June 3, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3178194

A Swapping Target Q-Value Technique for Data Augmentation in Offline Reinforcement Learning

HO-TAEK JOO[1], IN-CHANG BAEK[2], AND KYUNG-JOONG KIM[1,2], (Member, IEEE)
[1]School of Integrated Technology, Gwangju Institute of Science and Technology, Gwangju 61005, South Korea
[2]AI Graduate School, Gwangju Institute of Science and Technology, Gwangju 61005, South Korea

Corresponding author: Kyung-Joong Kim (kjkim@gist.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (MSIT) under Grant 2021R1A4A1030075.

ABSTRACT Offline reinforcement learning (RL) is applied to fixed datasets of logged interactions pertaining to actual applications in healthcare, autonomous vehicles, and robotics. In limited and fixed dataset settings, data augmentation can be beneficial in developing better policies. Several online RL methods for data augmentation have recently been utilized to enhance sampling efficiency and generalization. Here, a novel, simple data-augmentation technique referred to as **S**wapping Target **Q**-**V**alue (**SQV**) is introduced to enhance offline RL algorithms and enable robust pixel-based learning without auxiliary loss. Our method matches the current Q-value of a transformed image to the target Q-value of the next original image, whereby the current Q-value of the original image is matched to the target Q-value of the next transformed image. The proposed method considers similar states as the same and different states as more distinct. Furthermore, the approach ties unseen data (lacking in the dataset) to similar states in the seen data. After training, these effects were observed to increase the performance of the offline RL algorithm. The method was tested on 23 games in the Atari 2600 game domain. As a result, the performance of our method improved in 18 out of 23 games, with an average performance improvement of 144% compared with batch-constrained deep Q-learning (BCQ), which is the latest offline RL method. The implementation can be found at https://github.com/hotaekjoo/SQV.

INDEX TERMS Offline reinforcement learning, data augmentation, generalization, Atari games.

I. INTRODUCTION

Online reinforcement learning (RL), combined with deep neural network function approximators, has been successfully applied in robotics [1], [2], Atari games [3], Go [4], and StarCraft [5]. The success of RL in these areas is attributed to the well-structured feedback system. RL is a decision-making method that enables agents to identify optimal behavior in a given state, and the feedback system is a series of processes in which agents interact with the environment and learn by observing the results of these interactions. Most of the environments in RL have been equipped with these systems. OpenAI created and provided environments suitable for RL in Atari games, while DeepMind provided the StarCraft 2 and DM-Control environments.

However, these feedback systems have become ineffective when applying RL algorithms to actual problems. In practice, few of these feedback systems exist. For example, assuming RL is applied to medical data, when an artificial intelligence (AI) physician performs some treatment (action), the appropriate reward (patient's death and life) and the patient's next state in the actual environment cannot be detected by the AI physician. Due to trial and error nature of RL, applying a learning AI physician's treatment (action) to a patient can raise ethical concerns.

Recently, researchers have begun studying the decision making process on already collected data, called offline RL [6], [7], [8]. Offline RL is a crucial solution in applying RL to actual problems, and the offline RL algorithm has recently emerged as an area of interest. Evaluating the value of out-of-distribution actions (OOD actions) correctly is a major issue in recent offline RL algorithms. The offline RL setting assumes that a specific behavioral policy determines actions according to a given state in the collected datasets. Actions not selected by the behavioral policy are called OOD actions. The OOD actions trigger

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

errors in overestimating and underestimating the value of actions.

Several offline RL algorithms have been proposed to handle OOD actions. The batch-constrained deep Q-learning BCQ [9] limits the degree to which the learned policy differs from the behavioral policy of the datasets. In the conservative Q-learning (CQL) [8] algorithm, a conservative Q-function is learned by regularizing the Q-values of OOD actions during training. The critic regularized regression (CRR) [10] and accelerating online reinforcement learning with offline datasets (AWAC) [11] algorithms adopt a method to restrict the OOD action from being selected.

These existing offline RL methods are yet to evaluate data augmentation on collected datasets, which may be a simple and promising way to improve the performance of offline RL algorithms. On the other hand, online RL is already being actively researched for data augmentation techniques. Reinforcement learning with augmented data (RAD) [12], contrastive unsupervised representations for reinforcement learning (CURL) [13], and data regularized Q-learning (DrQ) [14], which are the approaches proposed for data augmentation in online RL, are reported to provide performance improvements and advantages in increasing data sample efficiency.

Inspired by the impact of data augmentation in online RL, in this study, a swapping target Q-value (SQV) method is presented for data augmentation in offline RL. The SQV algorithm was developed by extending BCQ, an offline RL algorithm. Subsequently, SQV was designed to improve sample efficiency and maximize the generalization ability of offline RL settings. The proposed method directly compares the non-augmented and augmented Q-values and identifies the original and transformed images as almost identical. The offline RL algorithms can handle various distributions of datasets, allowing substantially similar and dissimilar data to be included in the datasets. The ability to group similar states in the datasets increases the efficiency of the sample, and this improvement in sample efficiency facilitates improved performance. Furthermore, our method improves generalization ability. which refers to the performance achieved when evaluating data that do not exist in the training set. Because our method exhibits the bundling effect of grouping similar states, it identifies good policies even for previously unseen data by grouping the policies of the most similar images in the training set.

The contributions of the study are as follows:

- It is demonstrated that SQV outperforms advanced offline RL baselines, such as BCQ, when tested on popular pixel-based Atari games. Furthermore, sample efficiency is evaluated by reducing the size of the collected datasets demonstrating that our method exhibits better sample efficiency.
- The reason behind the performance improvement of the method is analyzed to determine the primary reasons for pixel robustness and the generalization capability of the offline RL settings.
- Finally, the limitations of data augmentation in offline RL are discussed. This is the first study to adopt data augmentation in the offline RL setting, and both the advantages and disadvantages are investigated compared to online RL data augmentation.

II. BACKGROUND

A. REINFORCEMENT LEARNING

Considering the Markov decision process, traditional RL methods determine the optimal behavioral policy that maximizes the expected return based on the policy determined by the expected returns from a state-action function. The RL agent performs an action a on a given state s in the environment, then, observes the reward r and next state s'. The state-action function $Q(s, a)$ is updated using the following equation, and the behavioral policy is trained using the collected tuples (state, action, reward, next state), (s, a, r, s') with the discount factor, where $\gamma \in (0, 1]$.

$$Q(s, a) = r + \gamma \times Q(s', a') \tag{1}$$

The Q-value is the estimated return from an agent's action in a given state. The RL methods update the current state Q-value $Q(s, a)$ with the next state Q-value $Q(s', a')$ to train an optimal Q-function approximator. Estimating the exact Q-value contributes to finding the optimal policy $Q^*(s, a)$.

B. OFFLINE RL

Offline RL is a method in which optimal behavioral policies are learned using data collected without environmental interactions. However, a distribution shift can be problematic when the behavioral policy of the collected data differs from the distribution of learned policies. In addition, a policy remote to the training data distribution cannot guarantee agent convergence. To address this problem, BCQ limits the extent to which the current behavioral policy differs from that of the dataset. In the random ensemble mixture (REM) [15], random convex combinations of multiple Q-heads are adopted to regularize the Q-values. CQL inserts additional regularization into loss terms when learning a conservative Q-function, which prevents Q-value overestimation.

C. DATA AUGMENTATION FOR ONLINE RL

Recently, several studies have been conducted on online RL methods that utilize data augmentation. CURL [13] utilizes data augmentations by learning contrastive representations, such as unsupervised learning to improve data efficiency. Although the focus of CURL is on jointly adopting data augmentation via contrastive RL losses, RAD [12] attempts to directly use data augmentation for reinforcement learning without any auxiliary loss. The authors of RAD, tested various image augmentation methods (e.g., random cropping, grayscale, cutout, rotate, etc.) in RL environments and benchmarked the image augmentation method that effectively improves performance in RL. DrQ [14] utilizes random cropping and regularized Q-functions in conjunction with the off-policy RL algorithm SAC [16]. In the DrQ study, the

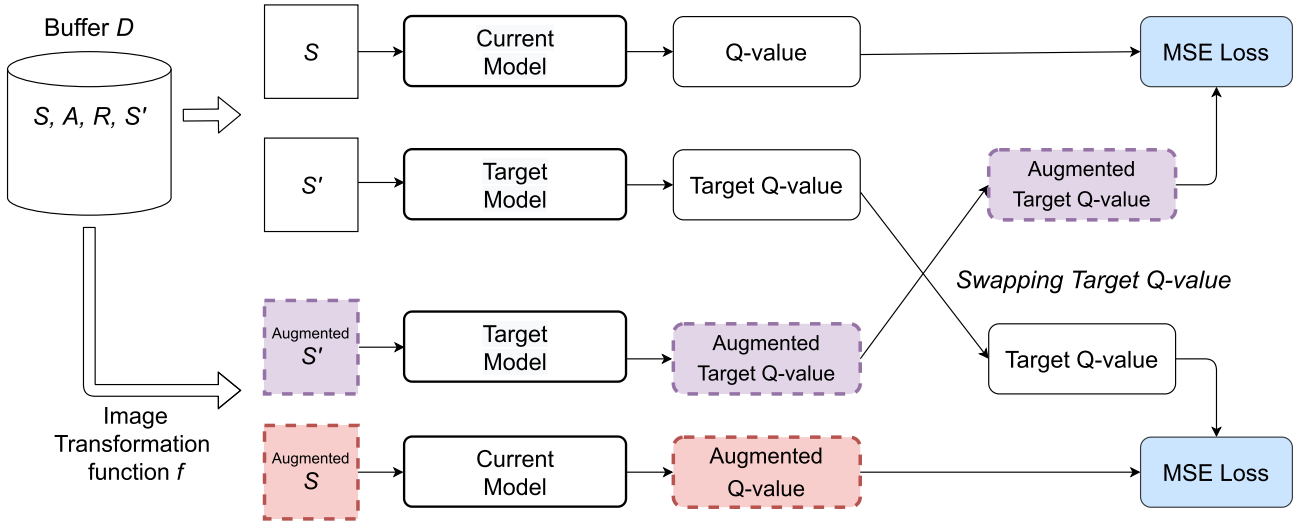


FIGURE 1. Architecture of the SQV method for data augmentation.

authors propose a way to apply the augmented image to the Markov decision process (MDP) of the RL task. This method adopts the regularized method for Q-values. Specifically, the target Q-values for the non-augmented and augmented states are obtained, and the target Q-value is recalculated by simply averaging these values.

The proposed method differs from DrQ in updating the Q-function, in that, the proposed method adopts these target Q-values directly by swapping them. Specifically, our method calculated the loss function for (Q-value of non-augmented image, target Q-value of augmented image) as well as (Q-value of augmented image, target Q-value of non-augmented image).

D. DATA OVERSAMPLING FOR OFFLINE RL

Some of the offline RL methods introduced above increase the amount of data involved, as in data augmentation. For example, from an engineering perspective, CQL[1] increases the collected state by a factor of 30. CQL calculates Q-values for each state-action pair by incrementing the current state, next state, and random state, by 10-fold. Such Q-values conservatively optimize policy. Furthermore, BCQ[2] yields a learned policy (including constraints) by increasing the state (including noise) 10-fold, and then calculates a Q-value for the action. In other words, the data are increased to optimize the learned policy for limited data. Such oversampling is used to address the limited data of data-driven offline RL.

E. GENERALIZATION FOR ONLINE RL

The study of generalization in deep RL aims to generate appropriate policies even for unseen states and levels. In online RL studies conducted so far, including RAD [12], NRD [17], DrQ [14], and CURL [13], data augmentation was expected to improve generalization. In particular, the generalization effect was verified in an environment such as Procgen [18], because the color of the background screen and objects change according to the levels. In this experiment, the changed conditions induce the unseen images which are the model confusing. Similarly, in our study, whether an agent trained by our method can create an appropriate policy for an unseen state was also tested.

III. SWAPPING TARGET Q-VALUE (SQV)

In this study, SQV was developed as a data augmentation method for offline RL. The method matches two targets when comparing the mean-squared error (MSE) for Q-values; accordingly, (state, augmented next state) and (augmented state, next state) are presented in Figure 1. First, the image transformation function f is defined. An original image in the dataset and the converted image are depicted by s and $f(s)$, respectively. The image transformation function includes random cropping, grayscale, cutout, and rotation. Our method uses random cropping and data augmentation to compute the augmented Q-value and target Q-values for every (s, a, r, s') transition.

$$Q(s,a)_{aug} = Q(f(s),a) \tag{2}$$

$$\text{target}Q(s,a)_{aug} = r + \gamma \times Q(f(s'),a') \tag{3}$$

Next, the target Q-values are swapped and updated via Stochastic Gradient Descent (SGD), using the learning rate λ_θ and batch size N as in Equations 4 and 5 below. The θ are then averaged and updated.

$$\theta_1 \leftarrow \theta - \lambda_\theta \nabla_\theta \frac{1}{N} \sum_{i=1}^{N} (\text{target}Q(s,a)_{aug} - Q(s,a))^2 \tag{4}$$

$$\theta_2 \leftarrow \theta - \lambda_\theta \nabla_\theta \frac{1}{N} \sum_{i=1}^{N} (\text{target}Q(s,a) - Q(s,a)_{aug})^2 \tag{5}$$

$$\theta \leftarrow \frac{\theta_1 + \theta_2}{2} \tag{6}$$

[1]https://github.com/aviralkumar2907/CQL/blob/master/d4rl/rlkit/torch/sac/cql.py

[2]https://github.com/sfujim/BCQ/blob/master/continuous_BCQ/BCQ.py

Algorithm 1 SQV: BCQ Version
Black: Unmodified BCQ
Blue: code Modified From BCQ

Input: batch $\mathcal{B}$, number of iterations T, target update frequency C, mini batch N, threshold τ, learning rate λ_θ
Initialize Q-network Q_θ, generative model G_ω, and target network $Q_{\theta'}$ with $\theta' \leftarrow \theta$,
Image transformation function f.

for $t = 1$ **to** T **do**
Sample mini batch N transitions (s, a, r, s') from $\mathcal{B}$
Image transformation for upper mini batch (s, s') to $f(s), f(s')$
$a' = \arg\max_{a'|G_\omega(a'|s')/\max \hat{a}\ G_\omega(\hat{a}|s')>\tau} Q_\theta(s', a')$
$a'_{aug} = \arg\max_{a'|G_\omega(a'|f(s'))/\max \hat{a}\ G_\omega(\hat{a}|f(s')>\tau} Q_\theta(f(s'), a')$
$\theta_1 \leftarrow \theta - \lambda_\theta \nabla_\theta \frac{1}{N} \sum_{i=1}^{N} \sum_{(s,a,r,s',f(s),f(s'))\in N} \left(r + \gamma Q_{\theta'}(f(s'), a'_{aug}) - Q_\theta(s, a)\right)^2$
$\theta_2 \leftarrow \theta - \lambda_\theta \nabla_\theta \frac{1}{N} \sum_{i=1}^{N} \sum_{(s,a,r,s',f(s),f(s'))\in N} \left(r + \gamma Q_{\theta'}(s', a') - Q_\theta(f(s), a)\right)^2$
$\theta = \frac{\theta_1+\theta_2}{2}$
If t modC$= 0$: $\theta' \leftarrow \theta$
end for

A. SQV FOR BCQ

The SQV based on the discrete BCQ [19] is summarized in Algorithm 1. Our method extends the discrete batch-constrained **deep Q-learning** to the data-augmentation setup by applying the same update rules as in offline training. BCQ is an offline RL algorithm that operates **deep Q-learning** (DQN) as a primary operation. The following equations express the objective function for selecting the next action in Q-learning.

$$J(\theta) = (\text{target}Q(s, a) - Q(s, a))^2 \tag{7}$$

$$\text{target}Q(s, a) = r + \gamma \times Q(s', a')$$

$$\text{where } a' = \arg\max_{a'} Q(s', a') \tag{8}$$

The BCQ differs from general DQN when selecting the next action, as expressed in the following equation.

$$a' = \arg\max_{a'|G_\omega(a'|s')/\max \hat{a}\ G_\omega(\hat{a}|s')>\tau} Q_\theta(s', a') \tag{9}$$

Instead of considering the maximum for all possible actions, the BCQ considers only those actions where (s', a') are likely to appear in the collected datasets. Similar to the equation above, $G_\omega(a'|s')$ [19] denotes the action probability distribution in discrete BCQ. After classifying the most probable action and the other actions by calculating the probability distributions, the probabilities of the other actions are divided by the highest action probability. If the quotient is less than a certain threshold value τ, actions corresponding to that value are not selected as the next action. Thus, the next action is selected from among the values that are greater than a certain threshold. In that case, the largest value is selected as the next action by multiplying the Q-values by the probability distribution.

$$a'_{aug} = \arg\max_{a'|G_\omega(a'|f(s'))/\max \hat{a}\ G_\omega(\hat{a}|f(s'))>\tau} Q_\theta(f(s'), a') \tag{10}$$

Similar to the equation expressed above, our algorithm extends BCQ when choosing the next action. Our method augments the next state using the image transformation function f. In addition, similar to the BCQ algorithm, the action distribution is output and the augmented next action is selected using the threshold value.

$$\theta_1 \leftarrow \theta - \lambda_\theta \nabla_\theta \left(r + \gamma Q_{\theta'}(f(s'), a'_{aug}) - Q_\theta(s, a)\right)^2 \tag{11}$$

$$\theta_2 \leftarrow \theta - \lambda_\theta \nabla_\theta \left(r + \gamma Q_{\theta'}(s', a') - Q_\theta(f(s), a)\right)^2 \tag{12}$$

$$\theta \leftarrow \frac{\theta_1 + \theta_2}{2} \tag{13}$$

Furthermore, the formula presented above applies to swapping the target Q-Value. Our algorithm adopts θ_1 and θ_2 to update the policy; θ_1 calculates the loss function for the augmented target Q-value and the non-augmented Q-value; and θ_2 calculates the loss function for the non-augmented target Q-value and the augmented Q-value. By adding the above two loss functions, the algorithm updates the policy network.

IV. EXPERIMENTS

Experiments Setup The objective of our experiment is to compare the performance and data efficiencies of SQV, with data augmentation, and BCQ, without augmentation, for offline RL settings. The offline RL setting is also known as full batch RL, in which a policy is learned from a static dataset. First, RL agents are trained in Atari games using the DQN [3] algorithm and the agents are used to collect the datasets. Then, the BCQ algorithm (without data augmentation) and the SQV algorithm (with date augmentation) are applied to the collected datasets. In addition, the data efficiency performances of the two algorithms are compared in tests and evaluations using only 10% and 50% of the collected datasets.

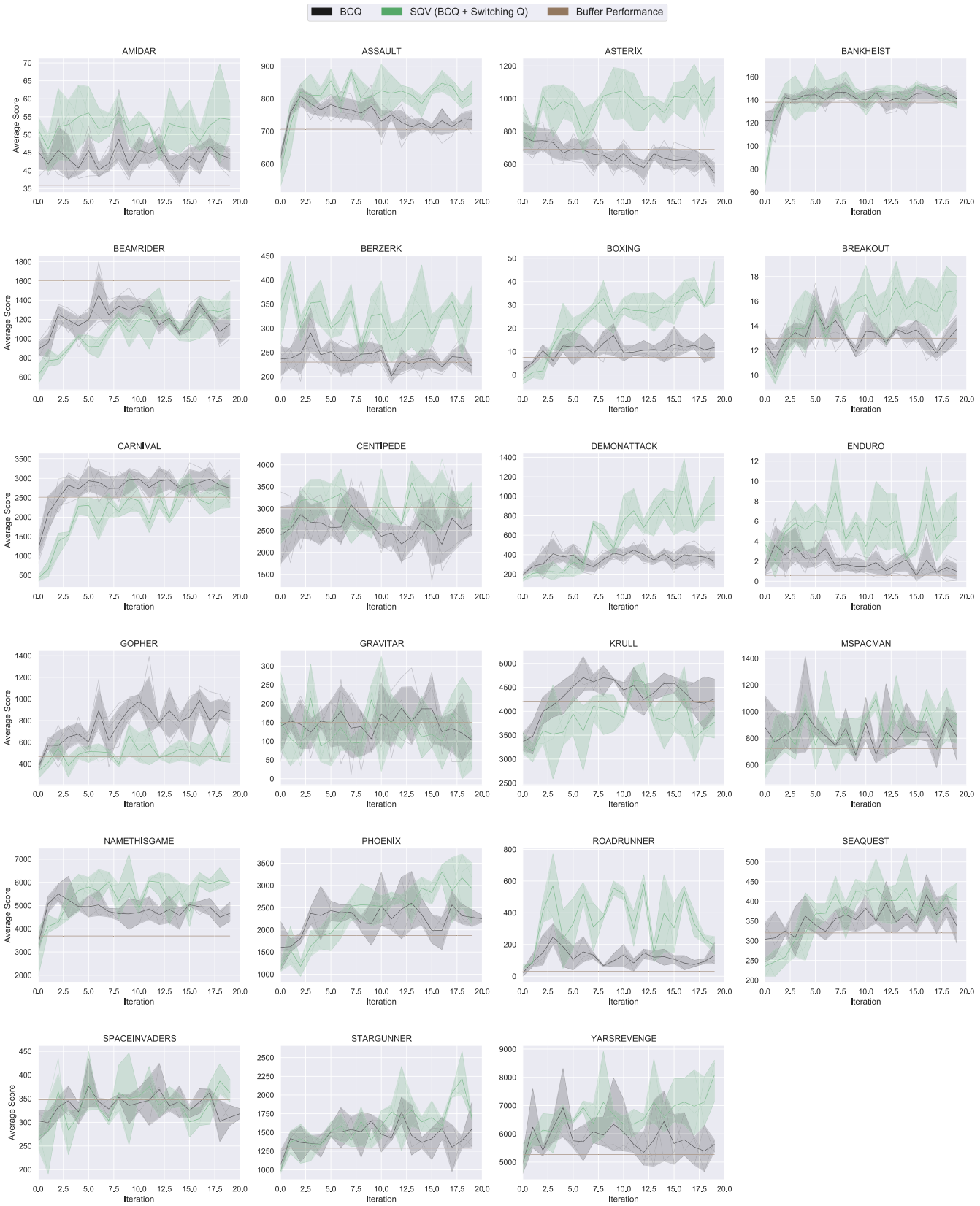


FIGURE 2. Atari game benchmark with 1 million datasets. The black and green lines represent the BCQ and SQV, respectively.

TABLE 1. Hyperparameters for DQN network.

Hyper-parameter	Value
Replay buffer size	1 million
Evaluation time steps	20k time steps
Initial ϵ	1.0
Final ϵ	0.01
ϵ decay period	250k training iterations
Network optimizer	Adam [22]
Learning rate	0.0000625
Adam ϵ	0.00015
Discount γ	0.99
Mini-batch size	32
Target network update frequency	8k training iterations
Huber loss κ	1
Evaluation ϵ	0.001

Environment The proposed method was validated in the arcade learning environment platform [20] of Atari 2600 games running on OpenAI gym [21]. The Atari 2600 games with high-dimensional visual input (210 × 160 RGB) are the most commonly used online RL testbed. In addition, these games are recently being adopted as offline RL testbeds. They are also suitable for testing the data efficiency and performance improvement of data augmentation in pixel-based RL. To summarize, this study addresses the following questions:

1) **Performance**: By how much did the performance of our method SQV improve compared to existing offline RL methods such as BCQ?
2) **Sample efficiency**: Using the 10% and 50% of the dataset, by how much was the performance enhanced compared to that of existing offline RL?

A. TRAINING AN AGENT USING DQN

The offline RL setup is for training an agent based on the collected dataset. The agents are first trained using the DQN algorithm in Atari games, whereby the trained agent can interact with the environment to collect datasets. The DQN parameters are presented in Table 1. The experimental settings in all Atari games are set in the same way as the Nature DQN model [3]. In Figure 2, the brown horizontal line represents the performance of the trained DQN agent. The line shows the average performance for testing the agent 20 times in each environment post training. The agents were trained for 30 Atari games; however, this study only displays the performance graph for 23 games in Figure 2 because seven agents were not trained using the DQN model.

The pixel input size of the Atari games is usually given as (250, 160, 3), where 250 refers to height, 160 to width, and there are 3 RGB channels. This study replaced this image with a grayscale (250, 160, 1) and then cropped it to 84 × 84 images (84, 84, 1). The network inputs were four stacked frames (4, 84, 84).

Table 2 presents the network structure of the DQN model. The DQN model comprises three convolution layers (Conv2d) and two fully-connected layers (FC, Linear), as presented in Table 2. Linear(1-1) is a layer that outputs the Q-value. Furthermore, this study adopts sticky action, which means that the probability of repeating an action from the previous state is $p = 0.25$. The reward function is clipped to a range of $[-1, 1]$. The DQN [3] implementation is based on [19][3]

TABLE 2. Network structure in online DQN.

Layer	output shape
Conv2d(1)	[−1, 32, 20, 20]
Conv2d(2)	[−1, 64, 9, 9]
Conv2d(3)	[−1, 64, 7, 7]
Linear(1)	[−1, 512]
Linear(1-1)	[−1, number of actions]

TABLE 3. Network structure used on the BCQ and SQV models.

Layer	output shape
Conv2d(1)	[−1, 32, 20, 20]
Conv2d(2)	[−1, 64, 9, 9]
Conv2d(3)	[−1, 64, 7, 7]
Linear(1-1)	[−1, 512]
Linear(1-2)	[−1, number of actions]
Linear(2-1)	[−1, 512]
Linear(2-2)	[−1, number of actions]

B. COLLECTING DATASETS

With each trained agent taking action according to a given state in the environment, data corresponding to 1 million transitions (s, a, r, s') were collected for each game. The datasets were collected by adding noise from a uniform distribution to the action of the trained agent. By adding action noise, a variety of datasets can be collected. In general, if a trained DQN agent selects the action with the highest Q-value in a given state, the agent cannot collect various transitions (s, a, r, s') into the buffer. Because offline RL is only trained based on the collected data, it is difficult to predict what action to take if a state does not exist in the buffer or an unexpected state appears during evaluation. Hence, data were collected to improve the offline RL performance by adding noise to action.

C. BCQ AND SQV SETTINGS

The SQV and BCQ agents were trained using the datasets in the previous section. In this section, the settings for BCQ and SQV are introduced. The Atari game settings are the same as those for the DQN mentioned above. Furthermore, the network structures of BCQ and SQV structures differ only slightly from that of DQN owing to the nature of the algorithm. Similar to Algorithm 1, the BCQ algorithm adopts both the Q-value and action probability distribution to decide what action to take for a given state. As presented in Table 3, Linear(2-1) and Linear(2-2) are added to the network comparing DQN. Linear(1-1) and Linear(1-2) are the layers that output the Q-value as in DQN, and the added Linear(2-1) and Linear(2-2) are layers that output the probability distribution for the action.

[3]https://github.com/sfujim/BCQ/blob/master/discrete_BCQ/DQN.py

TABLE 4. Descriptive statistics of BCQ and SQV on the Atari 1 million benchmark shown in Figure 2. The bold style fonts denote that our method outperforms the baseline in most games.

Game Title	BCQ	SQV (Ours)	Improvement (%)
Amidar	44.39±4.64	**49.32±6.21**	**110.13%**
Assault	742.34±49.18	**827.87±70.55**	**113.76%**
Asterix	663.37±87.53	**991.5±132.95**	**154.43%**
BankHeist	140.82±9.63	142.7±18.42	98.68%
BeamRider	1194.19±174.91	**1309.16±223.87**	**109.94%**
Berzerk	242.175±29.79	**336.2±50.00**	**151.10%**
Boxing	10.65±4.75	**30.66±12.28**	**302.46%**
Breakout	13.275±1.32	**15.25±2.33**	**114.40%**
Carnival	2799.15±499.34	**2458.3±681.65**	90.30%
Centipede	2553.64±518.67	**3098.65±471.30**	**124.68%**
DemonAttack	357.12±99.61	**842.5±322.13**	**236.52%**
Enduro	1.89±1.41	**5.85±2.49**	**321.42%**
Gopher	752.35±210.13	525.6±115.89	56.09%
Gravitar	163.25±66.87	**166.0±75.94**	**122.96%**
Krull	4370.95±439.05	4174.5±526.63	98.42%
MsPacman	803.4±159.03	**951.2±172.67**	**112.95%**
NameThisGame	4780.4±556.09	**5513.6±937.04**	**118.53%**
Phoenix	2265.0±399.11	**2711.0±693.71**	**114.39%**
RoadRunner	115.0±65.91	**341.0±193.59**	**324.76%**
Seaquest	350.7±43.99	**392.8±71.82**	**109.78%**
SpaceInvaders	333.05±35.85	337.55±48.97	98.78%
StarGunner	1477.0±217.85	**1727.0±312.71**	**118.04%**
YarsRevenge	5796.97±802.85	**6862.08±857.89**	**119.46%**
Average			**144%**

TABLE 5. Descriptive statistics of the data efficiency experiment on the Atari benchmark. The dataset is limited to 10% and 50% for the 1 million dataset. Most of results show that our method outperforms the baseline with limited data.

Game Title	10% Dataset		50% Dataset	
	BCQ	**SQV**	BCQ	**SQV**
Amidar	**38.9±3.6**	31.7±6.0	40.2±5.3	**45.2±6.3**
Assault	471.0±31.8	**556.6±49.4**	**585.9±33.1**	567.9±38.7
Asterix	429.2±65.9	**432.5±60.1**	443.0±61.1	**555.0±60.0**
Berzerk	**212.6±44.1**	180.1±44.1	190.0±27.0	**225.1±38.3**
Boxing	0.6±2.6	**2.36±2.6**	3.4±3.3	**5.7±3.1**
Breakout	7.5±0.8	**9.2±0.9**	9.9±1.7	**11.4±1.4**
Centipede	**2379.1±336.6**	2284.8±445.9	2255.3±392.3	**2847.4±448.9**
DemonAttack	**189.7±47.7**	185.1±38.8	**195.7±30.7**	181.8±54.7
Enduro	2.1±1.1	**2.1±1.24**	1.1±0.8	**1.27±0.8**
NameThisGame	3042.3±421	**3109.9±226.8**	3793.8±505.4	**4093.3±394.8**
RoadRunner	3.5±5.7	**35.9±19.9**	44.0±30.6	**66.0±28.7**
YarsRevenge	5229.2±327	**5670.7±914.4**	5273.1±956.5	**5312.0±841.9**

Image Augmentation (Random Cropping) The most frequently used online RL image augmentation method is random cropping, as in RAD and CURL. These methods train the agents in the DeepMind control suite and on Atari games. In this study, data augmentation experiments were also conducted for Atari games using random cropping in various image augmentation methods. First, the Atari game observations were converted to 84 × 84 gray-scale images, using ''zero padding'' of pixels on each side. Then, random 84 × 84 crops that moved the original images by ±4 pixels were selected. Such data augmentation can be adopted for images sampled from the replay buffer collected by the DQN [3] agent. The augmentation procedure was applied to images from the replay buffer.

D. PERFORMANCE

Using the settings in the previous section, the agents were trained in 1 million steps for each of the BCQ and SQV algorithms. The performance of each algorithm was evaluated 20 times for each Atari game during the training, every 0.05 million steps. Figure 2 compares the DQN, BCQ, and SQV algorithms in terms of the experimental results. The brown line and buffer performance indicate the performance of the trained DQN. The black and green lines represent the BCQ and SQV graphs, respectively. Table 4 is a quantitative representation of Figure 2. The average score for the last 10 of the 20 evaluations was recorded. In terms of performance improvement, SQV (ours) outperformed BCQ by 144%, and the performances of 18/23 games improved.

E. SAMPLE EFFICIENCY

In the sample-efficient online RL, researchers evaluated the performance on relatively small datasets collected via interactions with the environment. Sample-efficient online RL studies, such as DrQ and RAD, assess performance according to the number of data samples stored in the

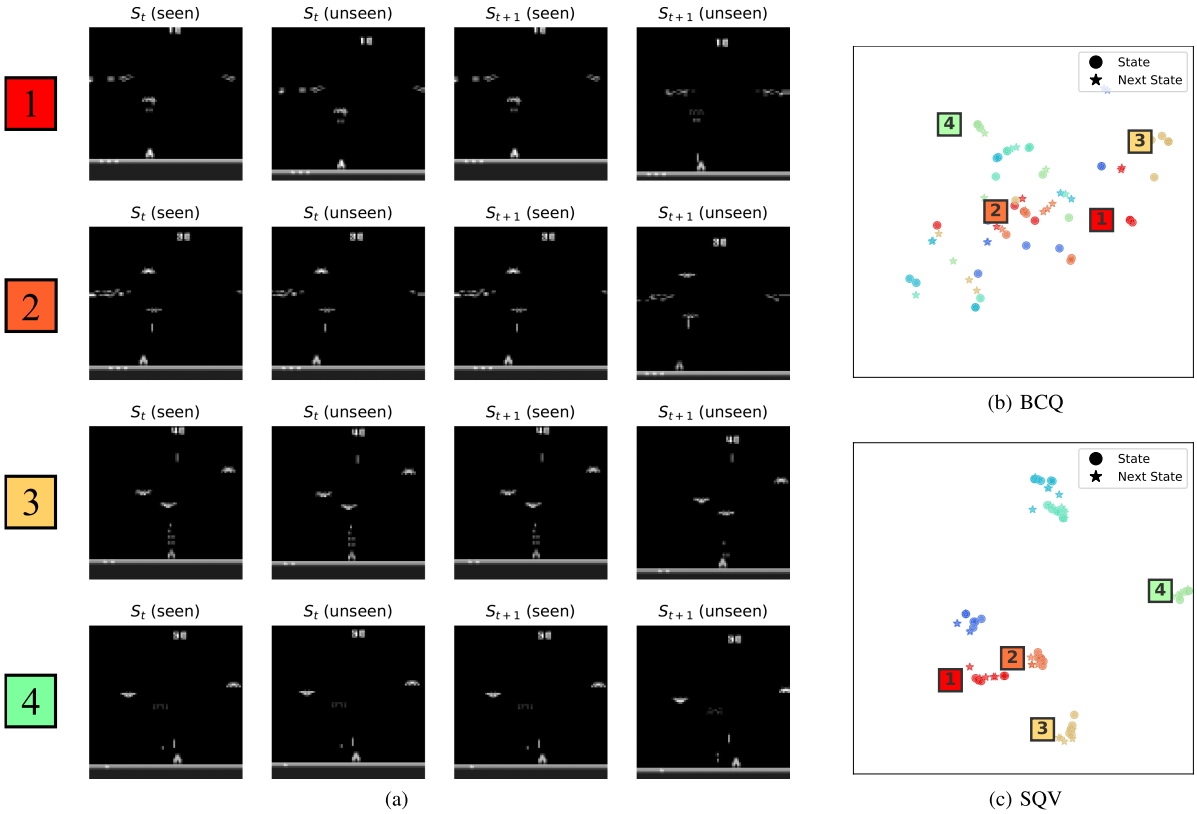


FIGURE 3. Results of t-SNE comparisons between the (b) BCQ algorithm and (c) SQV.

replay buffer. In other words, the criterion for ascertaining performance depends on the speed at which agents improve their performance because the smaller the steps, the fewer are the data collected in the buffer.

However, the basic setting of offline RL cannot collect data via interactions with the environment. Offline RL studies, such as CQL, measure sample efficiency by reducing the collected datasets. In addition, our performance experiment adopted only 10% and 50% of the collected datasets to evaluate sampling efficiency. Additional experiments were conducted on 12 Atari games. Table 5 presents the performances obtained when using 10% and 50% of the data. In this table, performance improvements of 8/12 for the 10% dataset and 10/12 for the 50% dataset are observed.

V. RESULT ANALYSIS

A. COMPARING PERFORMANCES ONLINE RL AND OFFLINE RL

In online RL, data augmentation methods such as DrQ and CURL exhibit significantly improved performance in pixel-based learning. For example, in a previous study, the DrQ data augmentation method was applied to the SAC [16] algorithm. This method improved the performance by 3~4 times more than the natural SAC in the DM control suite [23] environment. Our data augmentation method in offline RL also facilities performance improvement; however, the performance improvement is not as significant as that of online RL. In fact, our results are approximately 144% better than those of BCQ when averaged across the last 10 episodes of 23 games, as shown in Table 4. The primary reason for the difference in performance improvement is the fundamental difference between online and offline RL. Online RL constantly interacts with the environment, and the data change depending on the agent's actions in a given state. The policy changes according to the data in online RL, and new data are altered according to the changed policy. Owing to this feedback loop, the effect of data augmentation in online RL increases. In other words, the use of data augmentation in online RL remarkably affects data growth and data collection based on the feedback loop. In addition to having a significant impact on policy changes, data collection also provides overall performance improvements. However, data augmentation in offline RL exhibits the most pronounced effect with data increase because additional data cannot be collected in the offline RL setting. Even if data are augmented in offline RL, the data quality does not change, which is associated with limited performance improvements compared to online RL.

B. T-SNE

In Figure 3, the t-stochastic neighbor embedding (t-SNE) method was adopted for the trained BCQ and SQV models

to analyze the reason for performance improvement when utilizing data augmentation in offline RL. t-SNE is an unsupervised non-linear technique commonly used to visualize higher dimensional data in a lower-dimensional space. t-SNE generates a probability distribution for these mutual distance relationships between points in a high-dimensional space, including a low-dimensional space with similar relationships between these points. The embedding results obtained via t-SNE are presented distinguishing between similar and dissimilar states.

C. T-SNE EXPERIMENTS

In Figure 3, models trained with BCQ and SQV for Atari games are organized in various input screens via data augmentation. After inputting these images into the BCQ and SQV models, t-SNE was applied to the feature maps obtained and passed through the last convolution layer. Some transitions were randomly sampled from the 1 million transition tuples (s, a, r, s'), and augmented images were created for each s and s' using the data augmentation mentioned in Section IV-C. Each row comprises s(seen), s(unseen), s'(seen), and s'(unseen) transition tuples (s, a, r, s') in Figure 3a. The first and third images in each row present the original transition states (seen) and next states (seen); the second and fourth images are the augmentations of the state (unseen) and the next state (unseen). The t-SNE graph of Figure 3b is obtained by applying the BCQ method to an offline baseline, and in Figure 3c, the graph is obtained by adopting our approach. The original and augmented images are not well-tied in the BCQ algorithm (Figure 3b); however, in our method, they are well-tied (Figure 3c).

D. PIXEL ROBUSTNESS

The proposed method primarily facilitates performance improvement because it exhibits pixel robustness capability. Figure 3 presents t-SNE results for BCQ and SQV, and the results display two effects related to pixel robustness:

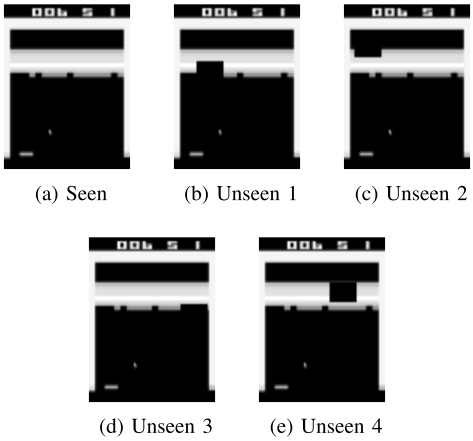
1) The proposed model recognizes the original and augmented states as similar states.
2) The proposed model allows the current state and the next state to be considered similarly (if the reward is zero).

As mentioned in Section IV-B, 1 million datasets were collected using the trained model for each Atari game. The trained model learns by extracting meaningful features for various states of this dataset. Using t-SNE results, our study demonstrated that the model outputs similar features for similar states, thereby increasing the efficiency of data samples, the generalization effect, and performance. As a second pixel robustness effect, the current states (s) and the next states (s') are grouped together in Figure 3. This figure presents (s) as a circle and (s') as a star, as indicted in the transition (s, a, r, s'). These results imply that our model recognizes these states as similar, thereby verifying this effect as:

$$\text{target}Q(s, a) = r + \gamma \times Q(s', a') \tag{14}$$

$$\text{target}Q(s, a) = 0 + 0.99 \times Q(s', a') \tag{15}$$

Assuming that the reward is zero, the difference between $Q(s, a)$ and the target $Q(s, a)$ is approximately 1%. Thus, if the reward is zero, the ability to recognize similar current states and next states increases the data efficiency in offline RL settings. Furthermore, Figure 3a indicates that the pixel change between the present and next states is negligible. The two effects described above improve sample efficiency and increase adaptability to unseen data. In Figures 6, 7, 8, and 9, our study also presents the results of t-SNE experiments for various Atari games.



(a) Seen (b) Unseen 1 (c) Unseen 2

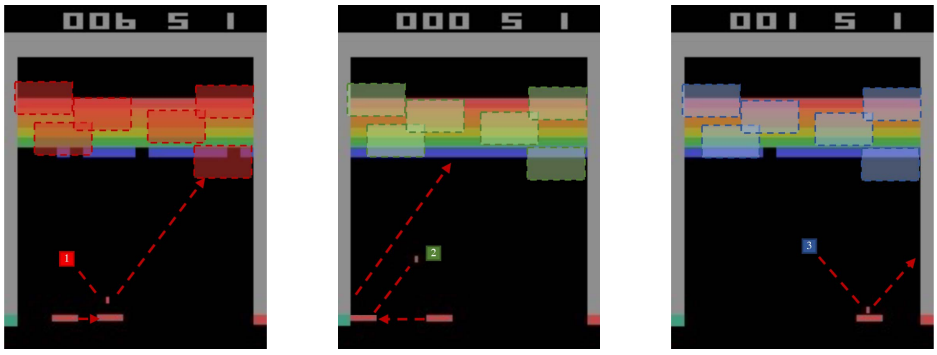
(d) Unseen 3 (e) Unseen 4

FIGURE 4. Example of unseen images that we fill black patch on the seen image.

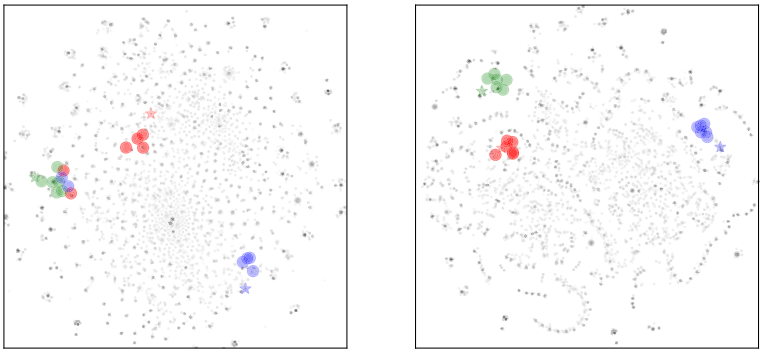
E. GENERALIZATION FOR UNSEEN DATA

One of the main reasons for the performance improvement of our offline RL method is the generalization ability. Generalization refers to the difference in the performance of a model when evaluated on previously seen training data and never-before-seen test data. The evaluation method of offline reinforcement learning is to learn using fixed and collected datasets to evaluate the learned model in an actual Atari game. To achieve significant performance improvement in the evaluation task, the trained agent must function optimally even on data not found in the collected dataset (created by the DQN agent). For example, if the behavioral policy has an average performance (i.e., total reward in an episode) of approximately 20 points in the breakout game, these data do not have a transition (s, a, r, s') higher than 20 points. For an agent to score higher than 20 points, the agent must function well on the datasets of 20 points or more that have never been seen before.

The experiments designed in Figures 4 and 5 evaluate the generalization ability of the trained model. In these experiments, whether the trained agent denies a policy optimally is evaluated by matching it with previous experiences on data

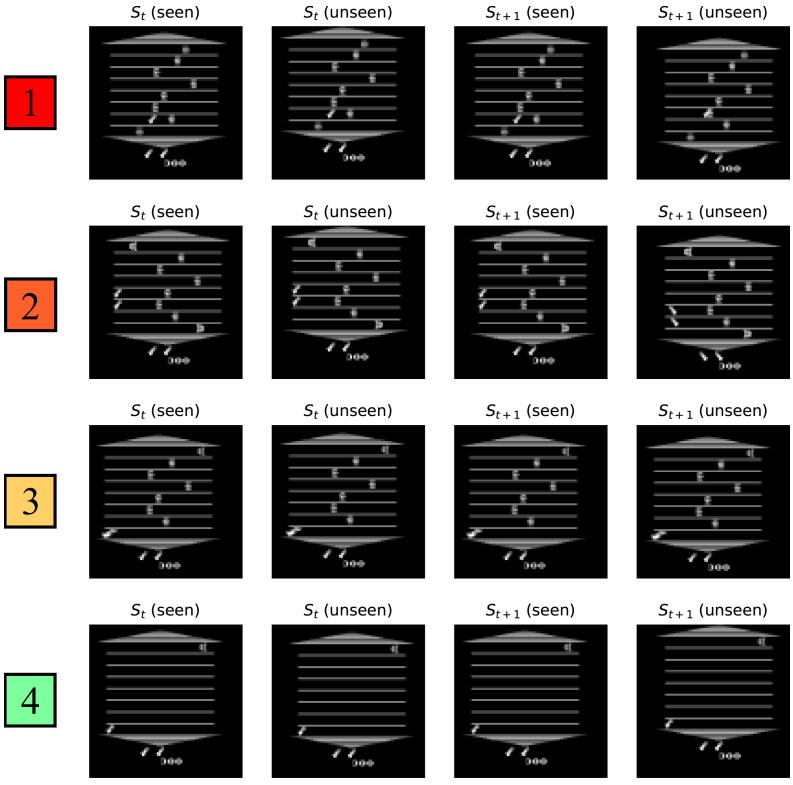


(a) Case 1 (Right) (b) Case 2 (Left) (c) Case 3 (No-Op)

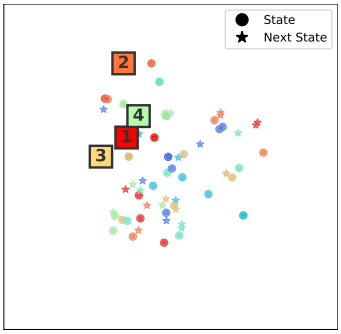


(d) BCQ (e) SQV

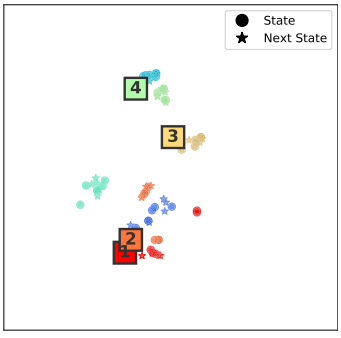
FIGURE 5. The t-SNE results show the generalization ability to unseen images. The unseen data are the augmented images from the three cases.



(a) Asterix

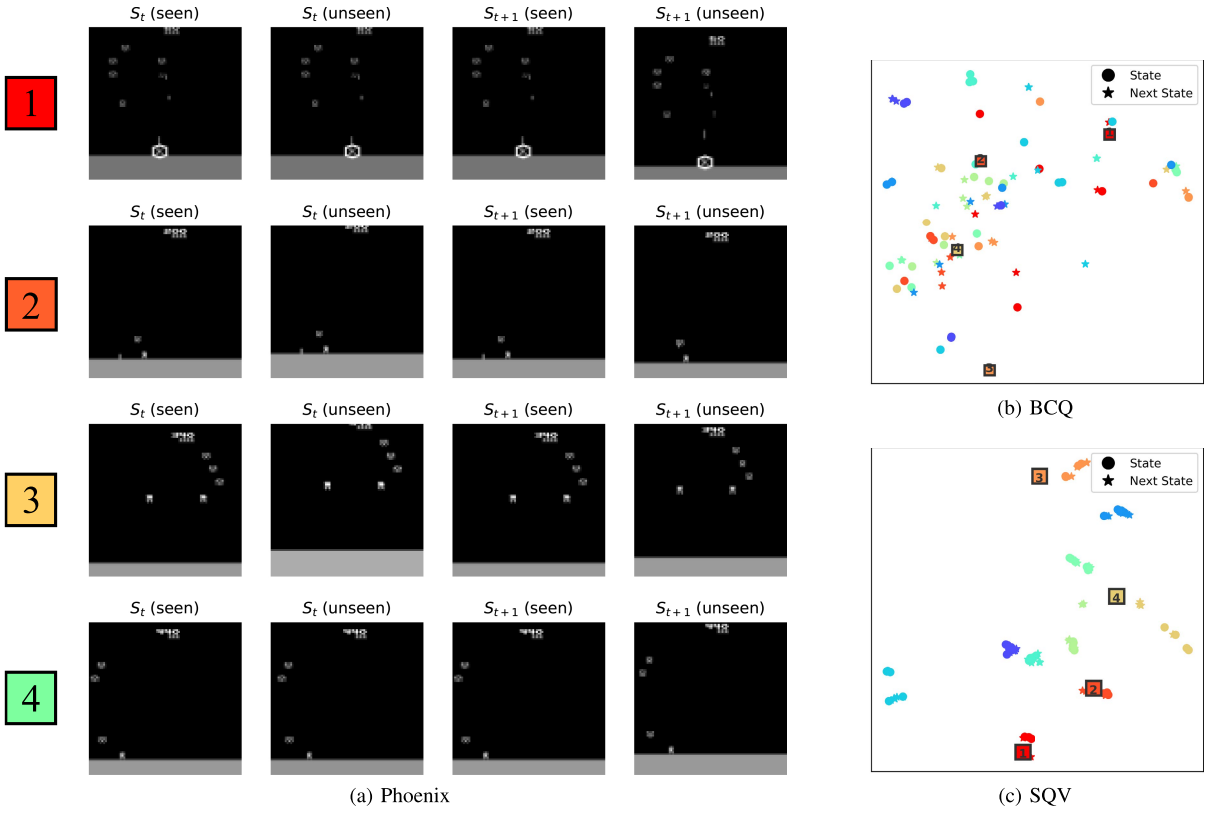


(b) BCQ



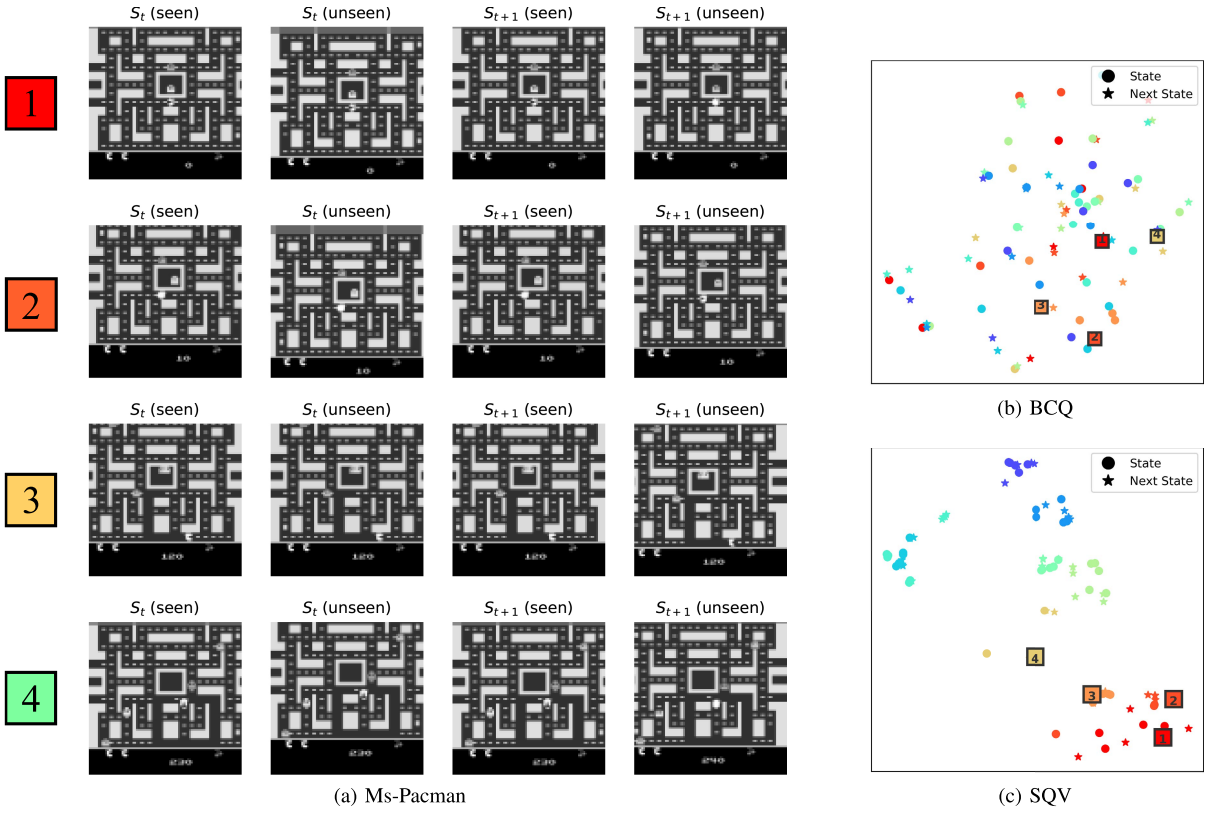
(c) SQV

FIGURE 6. Results of t-SNE comparisons between the BCQ baselines and SQV for Asterix in Atari games.



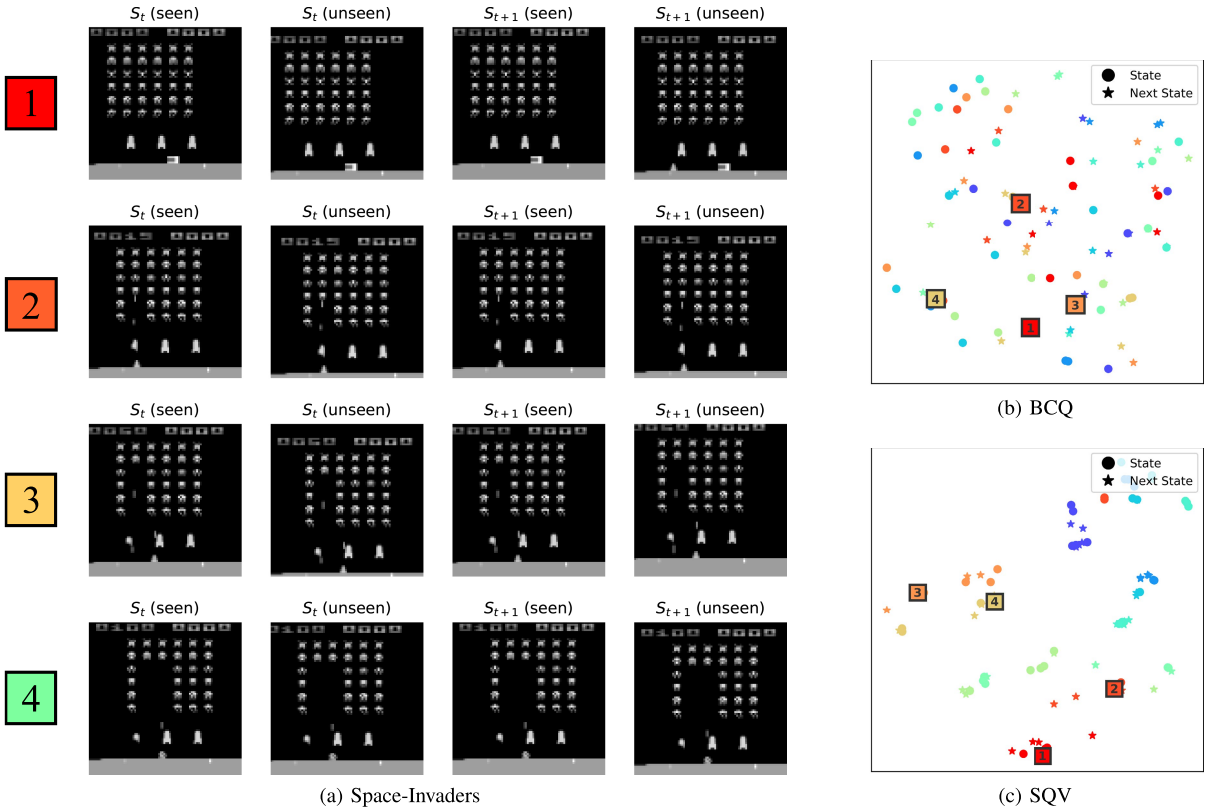
(a) Phoenix (b) BCQ (c) SQV

FIGURE 7. Results of t-SNE comparisons between the BCQ baselines and SQV for Phoenix in Atari games.



(a) Ms-Pacman (b) BCQ (c) SQV

FIGURE 8. Results of t-SNE comparisons between the BCQ baselines and SQV for Ms-Pacman in Atari games.



(a) Space-Invaders (b) BCQ (c) SQV

FIGURE 9. Results of t-SNE comparisons between BCQ and SQV for Space-Invaders in Atari games.

never seen before. For these experiments, the t-SNE method was adopted to demonstrate that our model is adaptable to unseen data, and then the matching performances of two methods, BCQ (Figure 5d) and SQV (Figure 5e) were compared. As presented in Figure 4, the state distribution of the environment is considered in creating valid unseen data. In our case, the Breakout game in Atari was selected as an example, and it was determined that the "cut-out" technique could represent the broken brick, remaining in the same context of the ball moving. In the Breakout game, the more the number of broken bricks, the higher is the state of the reward. Figures 5a, 5b, and 5c are three examples of unseen test-set creation based on Figure 4. By filling six parts of each image with black ("cut-outs"), the unseen images were created from the training set. The positions of the balls and the poll (a stick on the bottom side of the screen) in each set are the same. In Figures 5d and 5e, Cases 1, 2, and 3 correspond to the red, green, and blue dots, respectively. The star mark of each color represents the original image, and each circle mark represents the augmented image by the cut-out. Comparing the two experiments in Figures 5d and 5e, it is observed that BCQ sometimes cannot distinguish each case, whereas our method (SQV) distinguishes each case optimally. Identifying a policy optimally by matching it with previous experiences, means taking the right action according to each situation. Therefore, a model with a high level of generalization will correctly classify each case. This experiment demonstrates that our model's ability to generalize unseen data is better than that of BCQ.

Appropriate policies must be created for states absent from the datasets to increase performance in offline RL settings. The unseen data were designed by augmenting the state of the Breakout game using cut-out and the generalization ability of the model was verified. Our study demonstrates that this generalization ability improves the performance of the model.

VI. CONCLUSION

Offline RL refers to methods in which the agent decides based on the collected dataset. These methods provide additional help over the online RL method in which the agent interacts with the environment in real-time to collect data. For example, if an autonomous vehicle driving dataset has already been collected in the real world, the offline RL algorithm, such as BCQ, REM, and CQL can create the agent that can drive based on this dataset.

However, in offline RL, the amount of data is fixed since the algorithm learns based on the collected dataset. In this work, a method was proposed to improve performance compared with the existing offline RL method by applying data augmentation. However, our method not only augments the data but also combines augmented data with the MDP of RL. The method measures the Q value of the augmented data, whereby the Q value of the original data and the Q value of the augmented data are swapped.

TABLE 6. Descriptive statistics of performance comparison on REM and REM+SQV.

Game Title	REM	REM + Ours	Improvement (%)
Assault	**4.62±3.15**	4.84±2.77	**104.76%**
Asterix	**212.5±30.84**	230.25±2.48	**108.35%**
Berzerk	284.85±57.20	**480.5±46.12**	**168.68%**
Breakout	4.03±1.29	**5.26±4.06**	**130.52%**
Carnival	**484.7±195.13**	408.7±222.63	84.32%
Enduro	21.08±6.39	**27.74±11.26**	**131.59%**
Gopher	32.0±28.39	**214.5±128.90**	**670.31%**
Krull	951±13.11	**995.69±993.9**	**104.70%**
NameThisGame	**1801.2±486.06**	1284.6±925.02	71.31%
Phoenix	341.8±358.26	**354.66±228.07**	**104.0%**
RoadRunner	**855.0±227.60**	716.5±361.68	83.80%
Seaquest	64.6±71.96	**66.0±90.62**	**102.16%**
YarsRevenge	**2252.08±1147.85**	2145.94±1579.29	95.28%
Average			**150.75%**

The model trained with our method (SQV) displayed input pixel robustness and generalization ability. Pixel robustness refers to grouping similar data in collected datasets. This ability allows the model to make rational policy decisions based on similar data. The original BCQ and modified BCQ (SQV) were also implemented to evaluate the performance of Atari games, demonstrating that SQV outperforms BCQ in most Atari games. Also, as shown in Table 6, we conducted additional experiments by combining our method (SQV) with REM which is one of the offline RL method. In this experiment, the REM combined with our method performed better than the original REM.

The proposed method has limitations, however. Our method increases the amount of data through data augmentation but does not change the quality of the data. Essentially, to change the quality of data, an environment in which the agent collects data in real-time is required, violating the definition of offline reinforcement learning. Therefore, only a performance improvement of 144% was obtained over the existing offline RL, and further significant performance improvement is not expected. Future work will involve a study in this direction to increase data quality. The objective is to collect additional data when an agent trained with our method makes inappropriate policies while interacting with the environment. Thus, by retraining the already trained agent through the collected dataset, the quality of the dataset as well as agent training can be improved.

REFERENCES

[1] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 23–30.

[2] X. Ren, J. Luo, E. SolowjoW, J. A. Ojea, A. Gupta, A. Tamar, and P. Abbeel, "Domain randomization for active pose estimation," in *Proc. Int. Conf. Robot. Automat. (ICRA)*, May 2019, pp. 7228–7234.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[5] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Kuttler, J. Agapiou, and O. J. Schrittwieser, "StarCraft II: A new challenge for reinforcement learning," 2017, *arXiv:1708.04782*.

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

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

[8] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-learning for offline reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1179–1191.

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

[10] Z. Wang, A. Novikov, K. Zolna, J. S. Merel, J. T. Springenberg, S. E. Reed, B. Shahriari, N. Siegel, C. Gulcehre, N. Heess, and N. de Freitas, "Critic regularized regression," in *Advances in Neural Information Processing Systems*, vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds. Red Hook, NY, USA: Curran Associates, 2020, pp. 7768–7778. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/588cb956d6bbe67078f29f8de420a13d-Paper.pdf

[11] A. Nair, M. Dalal, A. Gupta, and S. Levine. (2021). *AWAC: Accelerating Online Reinforcement Learning With Offline Datasets*. [Online]. Available: https://openreview.net/forum?id=OJiM1R3jAtZ

[12] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," 2020, *arXiv:2004.14990*.

[13] A. Srinivas, M. Laskin, and P. Abbeel, "CURL: Contrastive unsupervised representations for reinforcement learning," 2020, *arXiv:2004.04136*.

[14] I. Kostrikov, D. Yarats, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," 2020, *arXiv:2004.13649*.

[15] R. Agarwal, D. Schuurmans, and M. Norouzi, "An optimistic perspective on offline reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 104–114.

[16] 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. Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 1861–1870.

[17] K. Lee, K. Lee, J. Shin, and H. Lee, "Network randomization: A simple technique for generalization in deep reinforcement learning," 2019, *arXiv:1910.05396*.

[18] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging procedural generation to benchmark reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 2048–2056.

[19] S. Fujimoto, E. Conti, M. Ghavamzadeh, and J. Pineau, "Benchmarking batch deep reinforcement learning algorithms," 2019, *arXiv:1910.01708*.

[20] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, Jun. 2013.
[21] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," 2016, *arXiv:1606.01540*.
[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[23] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. Lillicrap, and M. Riedmiller, "DeepMind control suite," 2018, *arXiv:1801.00690*.

HO-TAEK JOO received the B.S. degree in electro-mechanical systems engineering from Korea University, in 2015, and the M.S. degree in information security from Sejong University, South Korea, in 2018. He is currently pursuing the Ph.D. degree with the School of Integrated Technology, Gwangju Institute of Science and Technology (GIST). His current research interests include explainable artificial intelligence and reinforcement learning.

IN-CHANG BAEK received the B.S. degree in computer science from Sejong University, Seoul, South Korea. He is currently pursuing the integrated M.S. and Ph.D. degrees with the AI Graduate School, Gwangju Institute of Science and Technology. His research interests include artificial intelligence, reinforcement learning, procedural content generation, and general game artificial intelligence.

KYUNG-JOONG KIM (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science from Yonsei University, in 2000, 2002, and 2007, respectively. He worked as a Postdoctoral Researcher with the Department of Mechanical and Aerospace Engineering, Cornell University, in 2007. He is currently an Associate Professor with the School of Integrated Technology, Gwangju Institute of Science and Technology (GIST). His research interests include artificial intelligence, game, and robotics.