Received September 18, 2021, accepted October 17, 2021, date of publication October 21, 2021, date of current version October 27, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3121751

Deep Adversarial Reinforcement Learning With Noise Compensation by Autoencoder

KOHEI OHASHI[1], KOSUKE NAKANISHI[1,2], WATARU SASAKI[1,3], YUJI YASUI[2], AND SHIN ISHII[1,4,5]

[1]Department of Systems Science, Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan
[2]Honda Research and Development Innovative Research Center, Tokyo, Japan
[3]Nomura Securities Company Ltd., Tokyo 103-8011, Japan
[4]International Research Center for Neurointelligence (WPI-IRCN), University of Tokyo Institutes for Advanced Study, The University of Tokyo, Tokyo 113-0033, Japan
[5]Advanced Telecommunications Research Institute International (ATR), Seika 619-0288, Japan

Corresponding author: Shin Ishii (ishii@i.kyoto-u.ac.jp)

This work was supported in part by the Grant-in-Aid for Scientific Research on Innovative Areas, JSPS KAKENHI, Japan, under Grant JP17H06310; in part by the Grant-in-Aid for Scientific Research B, JSPS KAKENHI, Japan, under Grant JP19H04180; and in part by the New Energy and Industrial Technology Development Organization (NEDO), Japan.

ABSTRACT We present a new adversarial learning method for deep reinforcement learning (DRL). Based on this method, robust internal representation in a deep Q-network (DQN) was introduced by applying adversarial noise to disturb the DQN policy; however, it was compensated for by the autoencoder network. In particular, we proposed the use of a new type of adversarial noise: it encourages the policy to choose the worst action leading to the worst outcome at each state. When the proposed method, called deep Q-W-network regularized with an autoencoder (DQWAE), was applied to seven different games in an Atari 2600, the results were convincing. DQWAE exhibited greater robustness against the random/adversarial noise added to the input and accelerated the learning process more than the baseline DQN. When applied to a realistic automatic driving simulation, the proposed DRL method was found to be effective at rendering the acquired policy robust against random/adversarial noise.

INDEX TERMS Deep reinforcement learning, adversarial learning, robustness, regularization, automatic vehicle control.

I. INTRODUCTION

A lingering issue in various deep neural network (DNN)-based machine learning technologies [1]–[5] is to make them invulnerable to adversarial noise targeted at degrading their predictive ability [6]–[8]. In an image classification scenario, even if a miniscule noise, imperceptible to humans, was added to the input image in the training dataset, the predicted class could be separated from the target class and misclassification could be induced [9]. Such adversarial noise (or attack) can seriously affect DNN-based reinforcement learning (deep reinforcement learning (DRL)) that is designed to be applied to autonomous control problems for real-world machines such as human-harmonic robots [10] and auto-navigation vehicles [11]. If a malicious software revises the image input into an auto-navigation vehicle such that a ‘‘STOP’’ signal reads ‘‘GO’’, as is typical of an adversarial attack, the vehicle could become a danger [12].

Recently, various adversarial attacks and adversarial learning methods have been reported in the scenario of DRL. In this study, we focused on adversarial attacks applied to observations and their compensation methods. In real-world applications, observation-based perturbations, which are mostly random, but may be adversarial in some cases, can be caused by sensor noises and/or unexpected changes in the environment. Therefore, to determine the sensitivity, countermeasures and robustness to those perturbations are very important. As established in the most recent study [13], existing methods traded off their control performance for robustness against perturbations because they sought optimal controllers (policies) for the Markov decision processes (MDPs) with noisy inputs. In contrast, we attempted to make the controllers robust against adversarial noises, whereas the MDPs to be solved were maintained unchanged; that is, the adversarial learning attempted to regularize the deep reinforcement (DR) learner by improving its internal representation.

A simple idea to make the DRL robust against adversarial noise is to add a regularization term that represents (non-)

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

robustness to such noise to the objective function utilized in the reinforcement learning (RL). A more advanced idea is to make the internal representation of the DR learner robust against such noise. In DRL, there have been a series of studies to facilitate the acquisition of good internal representations of DR learners. These are termed self-supervision [14] and unsupervised auxiliary tasks [15]. Although our study is conceptually similar to these studies, it differs in two aspects. First, our learning architecture consists of multiple-head encoder-decoder networks with a shared encoder so that the constituent denoising autoencoder (AE) [16] attempts to remove possible input noises at the level of the shared internal representation. Second, we applied adversarial noises for the DRL objective to the DRL process with the expectation that it would be effective for rendering the internal representation more robust compared to that in other studies.

According to a naive implementation of our regularization method, we used the fast gradient sign method (FGSM) [9] to attack the ϵ-greedy policy determined by the current action-value function. One problem of this naive implementation is that it assumes that the current controller (policy) is optimal. This is not necessarily the case because the policy is usually sub-optimal during the RL process. Thus, the naive implementation is especially vulnerable to adversarial attacks. Based on this observation, we also present an advanced regularization method for DRL. According to this second method, a DR learner is trained not to apply the worst-case control in each state.

To implement these methods, we developed two new encoder-decoder architectures with one shared encoder and two or three different decoders, which were respectively based on the simple idea and the advanced idea above. Although these regularization methods can be applied to any type of DRL, we applied them to DQN with a discrete action space in particular [2], [3]. When DQNs with the advanced regularization method was applied to seven different games in the Atari 2600 [40], we found that the regularized DQN was more robust against random and adversarial noise than the original DQN and our simple version. Moreover, our regularization method effectively enhanced noiseless performance after a fixed number of training games; therefore, the advanced version would have accelerated the RL processes.

The major contributions of our study are summarized as follows.

- We presented a novel DRL architecture with adversarial training, deep Q-W-network regularized with an autoencoder (DQWAE), i.e., the advanced version, that was regularized by noise compensation learning using the denoising autoencoder.
- When applied to well-established Atari game benchmarks, the proposed adversarial RL method showed better noise robustness than the baseline method and our simple version called deep Q-network regularized with an autoencoder (DQAE), without significant increase in computational cost. Moreover, the learning process was also accelerated.
- Through an application to our original Unity-based autonomous driving simulation, we discussed why the proposed DQWAE was effective, that is, change in the internal representation was suppressed even when adversarial attacks were applied to the input image to the control policy.

The remainder of this paper is organized as follows. In Section II, we present related works and describe important notions, some of which are constituents of our methods. In Section III, we propose two DRL methods, the simple version and the advanced version, as well as their architectures and training schemes. In Section IV, we present a series of evaluation experiments using seven Atari games and our original driving simulation. In Section V, we present discussions, the limitations of the study, and directions of future work. Our conclusions are presented in Section VI.

II. RELATED WORK

A. DENOISING AUTOENCODER

Autoencoder [17] (AE), a type of encoder-decoder deep neural network, has been widely used in unsupervised representation learning. To obtain robust latent representations, a denoising AE, which is trained to reconstruct clean images from noisy ones, has also been examined in several studies [16], [18]–[20]. In this study, we present a couple of new deep learning architectures that combine the denoising AE and DR learners to eliminate adversarial noises that could have been included in inputs to the reinforcement learners. Although this combination itself seems rather natural, there are still many unresolved points; it is important to establish the effective noise to perform regularization and effective architecture to combine the denoising AE with DRL.

B. ADVERSARIAL EXAMPLE

In image classification, DNNs are known to output erroneous class labels even when a tiny noise is added to the input examples that were used as training samples. Such noise is said to be an adversarial attack because it exploits the vulnerability of DNN-based classifiers. An error-inducing input disturbed by the adversarial attack is called an adversarial example. Among the several methods for generating adversarial examples, box-constrained L-BFGS [7], least-likely class method [21], Jacobian-based saliency map approach [22], untarget DeepFool [23], Carlini and Wagner's [24], and FGSM [9], [25], we focused on the FGSM, which was originally proposed for regularizing DNN-based image classifiers. According to [9], [25], an adversarial example is defined as

$$x_{adv} = \begin{cases} x + \epsilon \cdot sign(\nabla_x J(x, y; \theta)) & (L_\infty - norm) \\ x + \epsilon\sqrt{d} \cdot \dfrac{\nabla_x J(x, y; \theta)}{\|\nabla_x J(x, y; \theta)\|_2} & (L_2 - norm) \\ x + \epsilon d \cdot \dfrac{\nabla_x J(x, y; \theta)}{\|\nabla_x J(x, y; \theta)\|_1} & (L_1 - norm), \end{cases} \quad (1)$$

where x is a d-dimensional input example (here, an image) and y is a one-hot vector representing a target class, whose pair is in the training dataset. The cost function $J(x, y; \theta)$ is the cross-entropy between the target class y and a predicted class $\hat{y}$, which is an output of the DNN with a parameter vector θ for input x. A scalar $\epsilon(> 0)$ determines the strength of the attack. The second term on the right-hand side is the adversarial attack. Although the average perturbation that can be applied to each element is set to be equal in equation (1), the element-wise maximum perturbation depends on the p value in the L_p-norm constraint. ϵd maximal perturbation can be applied to one element under the L_1-norm constraint, whereas under the L_∞-norm constraint ϵ maximal perturbation is allowable for each element.

The idea of adversarial examples can be extended to RL. Noise is considered adversarial when it disturbs the action-value function (i.e., Q-function) that determines the control policy, thereby leading to ineffective controls (actions). Although several methods to produce adversarial examples have been proposed [9], [22], we applied an RL version of FGSM. Here, we explain this method in relation to the DQNs. A DQN outputs a vector of Q-values for all possible actions a, $Q(s, a)$, the dimensionality of which is the number of possible actions, $|A|$, given an input image s. According to the FGSM for RL, the cost function J is the cross-entropy between $\pi(a|s)$ that denotes the control policy determined by $Q(s, a)$ and a binary vector $\hat{\pi}(a|s)$. Here, $\pi(a|s)$ is a $|A|$-dimensional probability vector, obtained by applying the softmax function to the Q-value vector, $Q(s, a)$, and $\hat{\pi}(a|s)$ is a $|A|$-dimensional one-hot vector whose elements for the optimal and non-optimal actions with the current policy π are 1 and 0, respectively.

$$J(s; \theta) = -\sum_{a \in A} \hat{\pi}(a|s) \log \frac{\exp(Q(s, a; \theta))}{\sum_{a' \in A} \exp(Q(s, a'; \theta))},$$
$$\hat{\pi}(a|s) = \begin{cases} 1 & (\mathit{if}\ a = \underset{a' \in A}{\mathit{argmax}}\ Q(s, a'; \theta)) \\ 0 & (\mathit{otherwise}). \end{cases} \tag{2}$$

In equation (2), we show that the Q-function is represented by a DNN with parameters θ.

It is known that there is a trade-off between the probability of mistakes induced by adversarial examples and the computational cost. We used FGSM with the L_2-norm constraint in this study because it produces effective adversarial examples with a low computational cost; the L_2-norm constraint is expected to make adversarial attacks applied to input images during training moderate.

C. WHITE-BOX AND BLACK-BOX ATTACKS

One of the important features of adversarial attacks is their transferability [7]. Transferability implies that an adversarial example created to attack a certain model also works for a different model. Therefore, an adversarial example can be created without knowledge of the target model. Therefore, we distinguished between white-box attacks in which the architecture and parameters of the target model are accessible to the attacker (malware) and black-box attacks [12], [26] in which they are unknown. Specifically, white-box attacks are based on the assumption that the malware (attacker) knows the input-output relationship of the DRL to be attacked; in our case, the attacker can employ the target DRL model on its side. Conversely, in black-box attacks, the attacker cannot access the DRL model being attacked. The attacker can only use another DRL model that is different from the attacked model to generate adversarial attacks.

D. ADVERSARIAL TRAINING FOR DRL

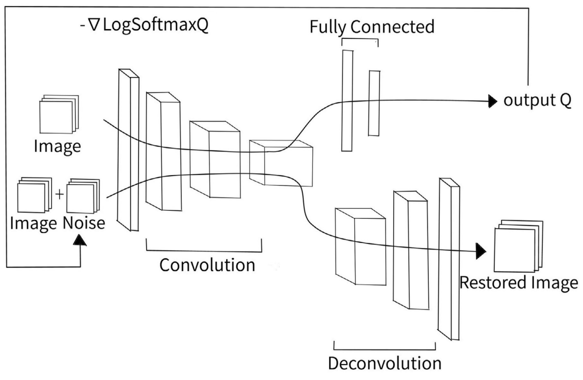
Recently, training with adversarial examples, adversarial training, has been attracting significant attention because of its benefits such as robustness against various types of noise and improved performance owing to its regularization effects. Although adversarial training has been successfully applied to deep learning-based supervised [27] and semi-supervised learning scenarios [28], [29], applications to DRL scenario are not fully explored [20], [30].

There have been a series of studies on robust learning in MDPs [13]. Some prior researchers used adversarial disturbances to sample the worst trajectories near optimal ones during MDP optimization [31], [32]. Their formulation is similar to the max-min optimization, in which attaining equilibria has been known to be difficult. Several researchers [33], [34] have examined a wide range of adversarial attacks/defenses; however, they did not consider sufficient defense methods to cope with perturbations applied to observations, i.e., input images in our case.

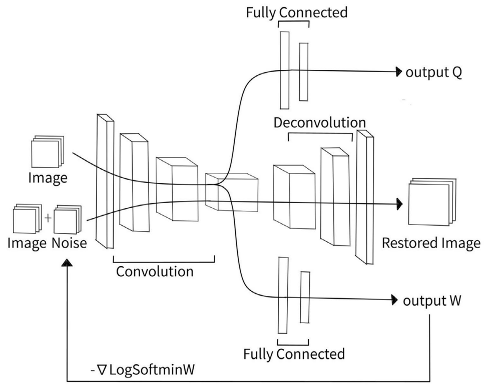
Adversarial learning to defend against perturbations applied to observations has been developed recently [35]–[38]. In [35], a certified defense module was introduced to compute the lower bound of action values under ϵ-ball perturbations to encourage the agent to take conservative actions. Instead of deploying such a defense module, in this study, we proposed a denoising architecture to be equipped with both high performance and robustness. Adversarial learning for robustness against observation perturbations can be defined in multi-agent settings; observations can be adversarial owing to the opponent's actions [36]. Although we assumed that observation perturbations mainly come from sensor noise, such as those in camera images and light detection and ranging (LIDAR) measurements, some evaluation settings regarding attacks, such as white-box and black-box attacks, are related to multi-agent settings.

In another study [37], a loss function with an adversarial regularizer was introduced to enhance the robustness against perturbations in state observations. The MDP was modified into a state-adversarial MDP (SA-MDP) such that the perturbations were computed to make the action distribution distant from the original one. Although the motivation of their study is similar to ours, they did not aim at making the internal representation of DR learners robust against input noises.

There is an existing study on deep adversarial reinforcement learning [38], in which adversarial attacks, worst cases for DQN, were used to explore the state space. Although this



(a) DQAE



(b) DQWAE

FIGURE 1. Architecture of proposed models.

idea of using adversarial examples is conceptually similar to that of this study, there are a number of significant differences. The existing study used the same network architecture with the original DQN, whereas we used multiple-head architectures including the AE. In the existing study, sampling by the behavioral policy was disturbed by adversarial attacks to efficiently explore the state space to learn, whereas, in our case, the internal representation of the control policy was expected to be robust during training disturbed by adversarial attacks.

III. DQN WITH AUTOENCODER

Our aim in this study was to develop new adversarial learning-based regularization methods for DRL algorithms that make the control policy acquired by DRL robust against input noise. At first, we explain the simple architecture for our DRL regularized with an autoencoder, then we introduce the second, our main proposal method, that utilized more sophisticated regularization by W-network. We used a DQN as a typical and commonly used instance of DRL.

A. DQN WITH AUTOENCODER REGULARIZATION

In the first simple method, we employed a twin-head architecture (Figure 1(a)): one of the components was the AE that attempted to restore the original input image (assumed to be clean) from a noisy input image, and the other was the DQN. They shared the same encoder whose input and output were the given image and the internal representation held in the middle layer, respectively. Importantly, the noisy input to the AE component was an adversarial example for the DQN control policy. By training the DQN and AE simultaneously, we expected the internal representation in the shared middle layer to become suitable for RL, specifically, Q-value learning, and be simultaneously robust against random and adversarial perturbations applied to the input image. This learning method is based on the combination of DQN and AE; therefore, it is called deep Q-network regularized with an autoencoder (DQAE).

For an input image s, the internal representation in the middle layer is encoded as $f(s;\theta_e)$, where f is an encoder with a parameter set θ_e. The output of the DQN is expressed as $Q(f(s;\theta_e), a;\theta_q)$; i.e., the Q-function is transformed from the middle layer representation $f(s;\theta_e)$; the transformation is represented by a set of DQN parameters, θ_q. Similarly, the decoded output of the AE is expressed as $g(f(s_{adv};\theta_e);\theta_d)$, where g is the decoder with a parameter set θ_d. Then, the loss function of the DQN, L_{DQN}, and the loss function of the AE, L_{AE}, are given by

$$\begin{aligned} L_{DQN}(s,a,r,s';\theta_e,\theta_q) &= E_{s,a,r,s'}[(y - Q(f(s;\theta_e),a;\theta_q))^2],\\ y(r,s';\theta_e^-,\theta_q^-) &= r + \gamma \max_{a'\in A} Q(f(s';\theta_e^-),a';\theta_q^-),\\ L_{AE}(s;\theta_e,\theta_d) &= E_s[\|s - g(f(s_{adv}(s);\theta_e);\theta_d)\|_2]. \end{aligned} \tag{3}$$

Here, the squared errors are for the vectors, i.e., the squared Euclidean norms. The second loss in equation (3) is the reconstruction error of the AE. To stabilize the training, we used target networks [2]; for instance, θ_e^- and θ_q^- in equation (3), respectively, denote the encoder and decoder parameters of the target DQN.

After the initial examination, we found that the adversarial examples given by the L_2-norm FGSM were the most effective:

$$s_{adv}(s;\theta_e^-,\theta_q^-) = s + \epsilon\sqrt{d}\cdot\frac{\nabla_s J(s;\theta_e^-,\theta_q^-)}{\|\nabla_s J(s;\theta_e^-,\theta_q^-)\|_2}, \tag{4}$$

where the cost function J is given by equation (2).

The final objective function used for training the entire DQAE is:

$$\begin{aligned} L(s,a,r,s';\theta_e,\theta_q,\theta_d) = &\, L_{DQN}(s,a,r,s';\theta_e,\theta_q)\\ &+\lambda L_{AE}(s;\theta_e,\theta_d), \end{aligned} \tag{5}$$

where $\lambda > 0$ is a hyperparameter that takes the balance between the DQN and AE losses.

1) TRAINING ALGORITHM

In training the DQAE, the following procedure is repeated.

1) A one-step back-up (s, a, r, s') is sampled uniformly from the empirical memory, to calculate the one-step objective of the DQN, l_{DQN},

$$y(r, s'; \theta_e^-, \theta_q^-) = r+\gamma \max_{a'\in A} Q(s', a'; \theta_e^-, \theta_q^-),$$
$$l_{DQN}(s, a, r, s'; \theta_e, \theta_q) = (y(r, s')-Q(s, a; \theta_e, \theta_q))^2. \tag{6}$$

2) The FGSM cost function, J, is calculated based on the DQN output (Q-value vector), according to equation (2).
3) Based on the calculated J, an adversarial example, s_{adv} is obtained, according to equation (4).
4) An output is obtained from the AE when the adversarial example above is input. Then, the one-step objective of AE, l_{AE}, is calculated as follows:

$$l_{AE}(s; \theta_e, \theta_d) = \|s - g(f(s_{adv}(s); \theta_e); \theta_d)\|_2. \tag{7}$$

After accumulating the one-step objectives as $L_{DQN} = \sum l_{DQN}$ and $L_{AE} = \sum l_{AE}$ over the mini-batch, the DQAE parameters are updated to lower the entire objective function: $L = L_{DQN} + \lambda L_{AE}$, using the stochastic gradient descent (SGD) method [39].

B. DEEP Q-W-NETWORK WITH AUTOENCODER REGULARIZATION

The objective of the DQAE was to make the current control policy robust against input noise that is adversarial to the optimality of the current policy. Although this idea would be reasonable if the control policy approached the optimal in the course of RL, its efficacy for accelerating the RL process during training is unknown. We propose another and more advanced regularization approach to induce the DNN's control policy to avoid its worst-case responses that would be effective in policy exploration during the RL process. For this purpose, we attached an additional decoder to predict the worst control at the current state for the DQAE architecture. This additional network is called a deep W-network (DWN). The proposed architecture, which is called deep Q-W-network regularized with an autoencoder (DQWAE), is a hybrid of DQAE and DWN. The adversarial examples were produced based on the output of the DWN.

1) ARCHITECTURE

In this second advanced method, we employed a triple-head architecture (Figure 1(b)). The three encoder-decoder pairs shared the same encoder. The first, second, and third were the DQN, AE, and DWN, respectively. The last network output W-values for all possible actions, each of which represented the worst-case value caused by the corresponding action. An adversarial example was produced based on the worst-case values (i.e., W-value vector), and the AE attempted to restore a noiseless image from such an adversarial example.

For an input image s, the output of the DWN is given by

$$W(s, a) = W(f(s; \theta_e), a; \theta_w), \tag{8}$$

where f and W denote the common encoder and the DWN decoder, respectively, and θ_e and θ_w are their respective parameters.

2) OBJECTIVE FUNCTION

Based on the FGSM, an adversarial example for the DQWAE is produced by

$$s_{adv}(s; \theta_e^-, \theta_w^-) = s + \epsilon\sqrt{d} \cdot \frac{\nabla_s J(s; \theta_e^-, \theta_w^-)}{\|\nabla_s J(s; \theta_e^-, \theta_w^-)\|_2}, \tag{9}$$

where the cost function J is given by the cross-entropy between the worst-case control policy and the one-hot vector signifying the worst-case action:

$$J(s; \theta_e, \theta_w) = -\sum_{a\in A} \hat{\rho}(a|s) \log \frac{\exp(W(s, a; \theta_e, \theta_w))}{\sum_{a'\in A} \exp(W(s, a'; \theta_e, \theta_w))},$$
$$\hat{\rho}(a|s) = \begin{cases} 1 & (if\ \ a = \operatorname*{argmin}_{a'\in A} W(s, a'; \theta_e, \theta_w)) \\ 0 & (otherwise), \end{cases} \tag{10}$$

where $\hat{\rho}(a|s)$ denotes the worst-case policy to predict the worst-case action leading to the locally minimum action-value at state s.

The entire DQWAE network is trained by minimizing the following objective function:

$$\begin{aligned} L(s, a, r, s'; \theta_e, \theta_q, \theta_d, \theta_w) = {} & L_{DQN}(s, a, r, s'; \theta_e, \theta_q) \\ & +\lambda_1 L_{AE}(s; \theta_e, \theta_d) \\ & +\lambda_2 L_{DWN}(s, a, r, s'; \theta_e, \theta_w), \end{aligned} \tag{11}$$

where θ_q and θ_d are the parameters of the DQN and AE decoders, respectively, and λ_1 and λ_2 are the hyperparameters that determine the regularization strength.

L_{DQN} and L_{AE} are the same as those in equation (3). In addition,

$$L_{DWN}(s, a, r, s'; \theta_e, \theta_w) = E_{s,a,r,s'}[(y-W(f(s; \theta_e), a; \theta_w))^2],$$
$$y(r, s'; \theta_e^-, \theta_q^-) = r + \gamma \min_{a'\in A} Q(f(s'; \theta_e^-), a'; \theta_q^-), \tag{12}$$

where y is a one-step predicted value associated with the worst action at the current state, s, and the subsequent best actions according to the current policy.

3) TRAINING ALGORITHM

The DQWAE training proceeds by repeating the following procedure.

1) A one-step back-up (s, a, r, s') is sampled uniformly from the empirical memory; the one-step objective of the DQN, l_{DQN}, is calculated using equation (6).
2) Another one-step back-up is sampled uniformly from the empirical memory, independently from the one

sampled in Step 1; then, the one-step objective of DWN, l_{DWN}, is calculated.

$$y(r, s'; \theta_e^-, \theta_q^-) = r + \gamma \min_{a' \in A} Q(s', a'; \theta_e^-, \theta_q^-),$$
$$l_{DWN}(s, a, r, s'; \theta_e, \theta_w) = (y(r, s') - W(s, a; \theta_e, \theta_w))^2. \tag{13}$$

3) The FGSM cost function, J, is calculated based on the DWN output (W-value vector) using equation (10).
4) Based on the calculated J, an adversarial example, s_{adv}, is obtained using equation (9).
5) An output is obtained from the AE when the adversarial example above is input. Then, the one-step objective of the AE, l_{AE}, is calculated using equation (7).

After accumulating the one-step objectives as $L_{DQN} = \sum l_{DQN}$, $L_{AE} = \sum l_{AE}$, and $L_{DWN} = \sum l_{DWN}$ over the empirical memory (mini-batch), the DQWAE parameters are updated to lower the entire objective function, equation (11), using the SGD method.

IV. EXPERIMENT AND RESULTS

A. EVALUATION ON ATARI 2600 GAMES

Our simple DRL method (DQAE) can be seen as an ablated version of the advanced method (DQWAE). Then, the couple of the regularized DRL methods, DQAE and DQWAE, were evaluated in terms of robustness against noise applied to the input images using seven Atari 2600 games: Breakout, Pong, Boxing, Chopper Command, Star Gunner, Qbert, and Robotank. As in the original DQN setting, each RGB game image frame was downscaled to a 84 × 84 (pixels) grayscale image, and four consecutive image frames were concatenated to the 84 × 84 × 4 input image. An action was produced stochastically by the DRL policy and fed back to the game during four consecutive game steps. In other words, the sampling rate of the input and output was one per four game image frames.

We prepared three baseline methods for comparison with the DQAE and DQWAE. The first was the original DQN and the second was a DQAE regularized by adding random noise to the input images during training. When adding random noise to the DQAE, the second term of equation (4) was from a multi-variate (but element-wise independent) normal distribution for its L_2-norm to become $\epsilon\sqrt{d}$. The third was a double DQN regularized by gradient-based adversarial attacks (DDQN-R) [38]. To compare the noise robustness achieved by the comparable adversarial learning, we set the same number of total training steps and a consistent noise strength of $\epsilon\sqrt{d}$ when producing adversarial attacks, for every method. It should be noted that although the multi-step beta distribution sampling technique for adversarial attacks was used in [38], it was not used here because of its high computational cost when the input comprises high-dimensional images, as in this study.

The DQN encoder consisted of three convolution layers; the first, second, and third layers were composed of 32 8 × 8 convolutional filters with a stride of 4, 64 4 × 4 filters with a stride of 2, and 64 3 × 3 filters with a stride of 1, respectively. We used the rectified linear unit for the activation function across all the convolutional units. The DQN decoder consisted of two fully connected layers. There were 512 units in the first decoder layer, and the second layer took the number of possible actions in each game. The network architecture of the regularized DDQN (DDQN-R) [38] was exactly the same as that of the DQN; the difference between the two was in their learning scheme.

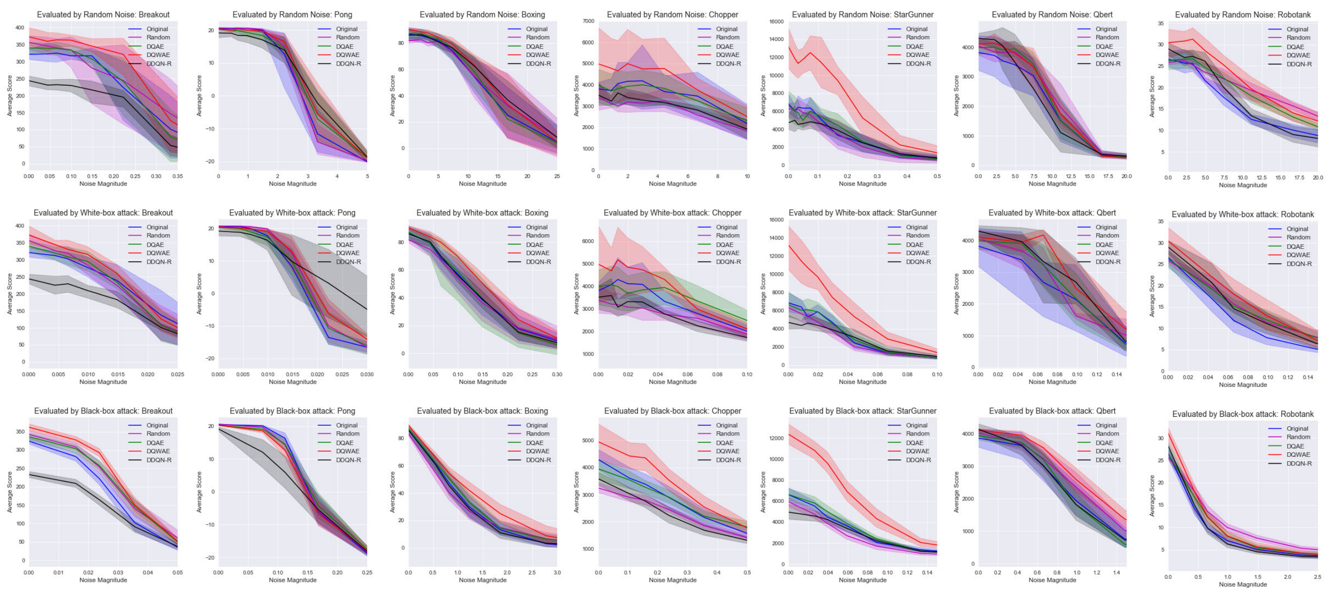
In the DQAE, the encoder architecture was the same as that of the DQN encoder. The decoder of the AE network in the DQAE had an upset architecture of the encoder network.

In the DQWAE, the encoder architecture replicated those of the DQN and DQAE. The decoder of the DQN part was the same as that of the DQN, and the AE decoder was the same as that of the DQAE. We also employed a common decoder architecture for the DQN and DWN components in the proposed DQWAE.

We optimized the DQAE, DQWAE, and the three baseline methods using SGD methods with mini-batch and the RMSProp optimizer. In the DQAE and DQWAE, we used the common strength of adversarial attacks, $\epsilon\sqrt{d} = 0.05$. When we used random noises to regularize the DQAE, we observed that the noise strength was too small after the initial examination and set it as a larger value, $\epsilon\sqrt{d} = 0.5$. In the DQAE, we set the regularization coefficients as $\lambda = 10$. In the DQWAE, we set the regularization coefficients as $\lambda_1 = 10$ and $\lambda_2 = 1$. These hyperparameter settings were the same for the seven Atari 2600 games. The implementations were performed on the ChainerRL environment [41].

We performed five training runs, each starting with random initialization of the DRL parameters for each of the seven games and each of the five DRL methods and ended after 10 million training steps. Here, a single training episode corresponds to a series of steps (plays) in a single game. When evaluating the noise robustness of each DRL method, we measured the game scores averaged over 10 test (rollout) games during which we applied noise of a specific strength to game image frames.

We specifically examined robustness against three types of noise, random noise with a systematically changed L_2-norm and white-box and black-box attacks, in which the noise was adversarial to the acquired policy and another policy. In the latter two cases, we systematically changed the adversarial attack strength in terms of its L_2-norm, following which we examined the average test-game scores. For the white-box attack, we assumed that the adversarial attacker knew the learner policy when generating noise to be adversarial to the learner policy. Conversely, in the black-box attack, there were two cases, the inter- and intra-method cases. For the former, the attacker trained using a different DRL method from that of the learner policy. In this case, we consistently used the original DQN as the attacker DRL method. In the intra-method case, the attacker used the same training method as that of the learner policy; however, the learner policy was based on a different training run.



(a) Breakout (b) Pong (c) Boxing (d) Chopper (e) Star Gunner (f) Qbert (f) Robotank

FIGURE 2. Atari average scores (solid line) and standard deviations (shaded area).

TABLE 1. Atari summary scores against ***random*** noise.

Environment		Breakout	Pong	Boxing	Chopper	StarGunner	Qbert	Robotank
L_2-norm attack(random) scale at eval.		0.15	2.2	11	4.4	0.25	7.4	7.4
DQN (original)	Noiseless Score	320.9 ± 16.4	20.5 ± 0.3	86.9 ± 5.8	3806.0 ± 388.2	6852.7 ± 1182.6	3819.8 ± 674.6	26.6 ± 2.1
	Attack Score	316.8 ± 25.9	12.7 ± 10.7	54.1 ± 14.9	3682.7 ± 871.6	2480.7 ± 1128.1	3038.5 ± 724.7	17.7 ± 2.7
DQN (random)	Noiseless Score	356.3 ± 23.4	$\mathbf{20.7 \pm 0.2}$	81.8 ± 2.6	3394.7 ± 746.2	6322.7 ± 885.0	4047.0 ± 359.3	25.8 ± 0.7
	Attack Score	280.8 ± 103.8	15.1 ± 4.5	54.4 ± 17.1	3254.7 ± 657.6	2033.3 ± 1446.0	3335.5 ± 438.4	22.0 ± 3.3
DQAE (ours1)	Noiseless Score	339.9 ± 18.6	20.4 ± 0.3	87.3 ± 3.7	3998.0 ± 882.7	6636.0 ± 1652.0	3985.3 ± 306.0	26.2 ± 2.2
	Attack Score	308.1 ± 28.0	14.6 ± 2.9	58.3 ± 13.6	3834.0 ± 888.6	2617.3 ± 602.3	$\mathbf{3430.8 \pm 468.5}$	22.2 ± 1.4
DQWAE (ours2)	Noiseless Score	$\mathbf{372.6 \pm 30.5}$	20.4 ± 0.2	$\mathbf{90.5 \pm 1.5}$	$\mathbf{4987.3 \pm 1992.6}$	$\mathbf{13210.7 \pm 2657.7}$	4101.0 ± 107.1	$\mathbf{30.4 \pm 3.9}$
	Attack Score	$\mathbf{345.6 \pm 22.4}$	$\mathbf{16.5 \pm 3.0}$	$\mathbf{63.1 \pm 9.4}$	$\mathbf{4765.3 \pm 1671.2}$	$\mathbf{5229.3 \pm 2371.7}$	3353.3 ± 872.9	$\mathbf{25.3 \pm 2.7}$
DDQN-R [38]	Noiseless Score	243.8 ± 17.1	19.2 ± 1.5	85.9 ± 5.4	3520.0 ± 598.8	4700.7 ± 835.9	$\mathbf{4293.3 \pm 199.3}$	29.0 ± 1.8
	Attack Score	216.6 ± 37.8	14.0 ± 4.8	62.4 ± 8.2	3166.7 ± 133.8	2488.7 ± 1104.2	2608.2 ± 1160.7	20.0 ± 2.4

TABLE 2. Atari summary scores against ***FGSM white-box*** attack.

Environment		Breakout	Pong	Boxing	Chopper	StarGunner	Qbert	Robotank
L_2-norm attack(FGSM) scale at eval.		0.010	0.015	0.15	0.044	0.044	0.066	0.066
DQN (original)	Noiseless Score	320.9 ± 16.4	20.5 ± 0.3	86.9 ± 5.8	3806.0 ± 388.2	6852.7 ± 1182.6	3819.8 ± 674.6	26.6 ± 2.1
	Attack Score	277.2 ± 31.7	8.3 ± 4.7	38.8 ± 14.2	3336.7 ± 585.0	2432.0 ± 725.7	2678.0 ± 1049.2	11.8 ± 3.2
DQN (random)	Noiseless Score	356.3 ± 23.4	$\mathbf{20.7 \pm 0.2}$	81.8 ± 2.6	3394.7 ± 746.2	6322.7 ± 885.0	4047.0 ± 359.3	25.8 ± 0.7
	Attack Score	283.7 ± 30.1	12.6 ± 4.1	36.6 ± 16.1	2767.3 ± 316.9	2020.7 ± 580.5	3245.2 ± 717.2	15.7 ± 2.6
DQAE (ours1)	Noiseless Score	339.9 ± 18.6	20.4 ± 0.3	87.3 ± 3.7	3998.0 ± 882.7	6636.0 ± 1652.0	3985.3 ± 306.0	26.2 ± 2.2
	Attack Score	295.4 ± 32.7	10.0 ± 5.2	39.8 ± 19.6	3951.3 ± 853.4	2779.3 ± 725.0	3224.0 ± 764.2	15.1 ± 1.9
DQWAE (ours2)	Noiseless Score	$\mathbf{372.6 \pm 30.5}$	20.4 ± 0.2	$\mathbf{90.5 \pm 1.5}$	$\mathbf{4987.3 \pm 1992.6}$	$\mathbf{13210.7 \pm 2657.7}$	4101.0 ± 107.1	$\mathbf{30.4 \pm 3.9}$
	Attack Score	$\mathbf{315.9 \pm 29.7}$	$\mathbf{13.0 \pm 5.8}$	$\mathbf{50.9 \pm 10.4}$	$\mathbf{4308.7 \pm 562.9}$	$\mathbf{5313.3 \pm 1849.3}$	$\mathbf{4169.8 \pm 193.9}$	$\mathbf{18.0 \pm 3.7}$
DDQN-R [38]	Noiseless Score	243.8 ± 17.1	19.2 ± 1.5	85.9 ± 5.4	3520.0 ± 598.8	4700.7 ± 835.9	$\mathbf{4293.3 \pm 199.3}$	29.0 ± 1.8
	Attack Score	209.5 ± 21.5	9.8 ± 11.2	39.5 ± 7.1	2785.3 ± 222.6	3069.3 ± 890.9	3331.8 ± 1353.0	14.6 ± 4.4

Figure 2 presents the robustness against noise applied to the input images for our DQAE and DQWAE, all of which were trained in 10 million training steps. The horizontal and vertical axes denote the noise strength applied to the test images and the score of the test games, respectively. The top, middle, and bottom panels in Figure 2 show the robustness against random noise and white-box and black-box attacks. The red, green, purple, blue, and black lines represent the DQWAE, DQAE, DQAE regularized by random noise, the original DQN, and the regularized DDQN (DDQN-R) [38], respectively. We summarized the noiseless/attacked scores for each attack method during evaluation; that is, for random (Table 1), FGSM white-box (Table 2), and FGSM black-box attacks (Table 3).

For the seven Atari 2600 games, we did not observe significant differences between DQAE and the two baseline methods, DQAE regularized with random noise and the original DQN, in terms of robustness against the three types of noise. Conversely, the proposed DQWAE consistently improved the noise robustness beyond that of the baseline methods, for these seven games. The superiority of the DQWAE was prominent for rather complicated games such as Chopper

Command and Star Gunner. In these games, the robustness against the three types of noise, random, white-box, and black-box, was significantly enhanced by the DQWAE, beyond those of the DQAE and the three baseline methods.

Interestingly, when the noise level was zero, the DQWAE outperformed the baselines in six games, except Pong, suggesting that the DQWAE could have realized better learning. In particular, for Star Gunner, the DQWAE score was almost twice that of the other methods. In other words, the new regularization method implemented in the DQWAE was effective in making the DRL robust against random/adversarial attacks and making the learning itself efficient through the data augmentation effect around an experienced sample (s, a). The AE regularization artificially generated $(s_{adv}, \cdot)$ and updated the shared encoder to enhance representation learning. This data augmentation effect was not realized by the DQWAE when the noise was random (data are not shown). It should be noted that the maximum score for Pong was 21, and all the methods, excluding the regularized DDQN (DDQN-R), attained this score in the noiseless case. The degree of the performance improvement was dependent on the game environment. The improvement in the score was very significant in the Star Gunner, Chopper Command, and Robotank, whereas it was moderate in Breakout, Pong, Boxing, and Qbert. It was suggested in a previous study (Fig. 5 in [44]) that Star Gunner, Chopper Command, and Robotank are relatively difficult tasks; therefore, their scores can be said to reflect the effectiveness of the employed RL algorithm. On the other hand, Breakout, Pong, and Boxing are relatively easy tasks whose scores were almost saturated even by the original DQN. Thus, we conjectured that the DQWAE would be effective in improving the performance in many applications that have room for performance elevation.

The noiseless scores of the DDQN-R were somewhat inferior to those of the original DQN, especially for the relatively difficult games, Breakout, Chopper Command, and Star Gunner. Because in [38] the behavior policy of the DDQN-R was disturbed by adversarial attacks during training, there would have been a trade-off between the noiseless performance and acquired robustness. Note here that our adversarial attack strength (0.05) was similar to that in [38] ($0.03 \sim 0.05$).

We would like to argue that the DQWAE exhibited the desired robustness against random/adversarial noise, even when it was not used for the training process. Indeed, the black-box attacks launched against the DQWAE were produced as adversarial to the DQN policy. This evaluation scheme seemed meaningful because the adversarial attack used here targeted the optimality of the control policy, which would have been achieved (realizing the Bellman optimality) after 10 million training steps. However, the DQWAE exhibited excellent robustness against the adversarial attack. This suggested that by enabling the DRL to avoid the worst-case actions, our new regularization was efficacious at retaining the best actions, even when the input images were maliciously disturbed to reduce the probability of taking the best actions. Accordingly, this result indicates the efficacy of our new regularization method for improving the general robustness of the DRL even against unknown/unexpected adversarial attacks.

B. AUTONOMOUS DRIVING SIMULATION

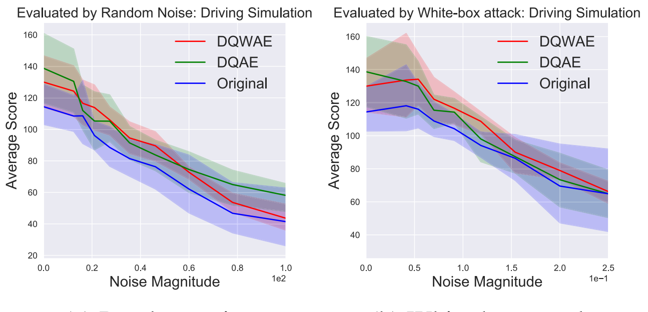
Next, we evaluated our regularization methods, DQAE and DQWAE, by applying them to the DRL for our automatic driving simulation. The driving simulation environment was constructed by combining Unity, Unity ML-Agents [42], [43], OpenAI Gym, and Chainer.

The Unity-based driving environment comprised a car operated by the DRL agent, bright pedestrians, and dark moving distractors. The pedestrians randomly walked across the street. The objective of the DRL agent was to steer the car without hitting the bright pedestrians. The dark distractors were not related to the navigation objective. The single input to the DRL was a concatenated image of four consecutive image frames sized 84×84 captured with a camera mounted on the top rear of the car. A single output of the DRL was one of three control actions: steering to the left (9°), going straight, and steering to the right (9°). The reward was one per step from the start to the end. The car continued to run for up to 500 steps (i.e., goal arrival). However, if the car hit any bright pedestrian or drove off the street, the episode ended.

Here, we compared three types of DRL methods: original DQN, DQAE, and DQWAE. The network structure and hyperparameters of the DQAE, DQWAE, and DQN were set to be similar to those in the Atari 2600 experiment.

Five runs of 520 thousand training steps (≈ 7000 training episodes) were performed for each method. To evaluate the noise robustness of each DRL method, we measured the score that was averaged over 20 test episodes (rollouts) during which noise of a specific strength was applied to each input image frame. We examined the robustness against two types of noise. The first was random noise with a systematically changed L_2-norm. The second type was white-box attack; we measured the score (i.e., the number of steps before an accident or goal attainment) when an adversarial noise generated by the acquired policy was added to each image frame.

Figure 3 presents the robustness of the baseline DQN, DQAE, and DQWAE against noises applied to the input images. The red, green, and blue lines represent DQWAE,

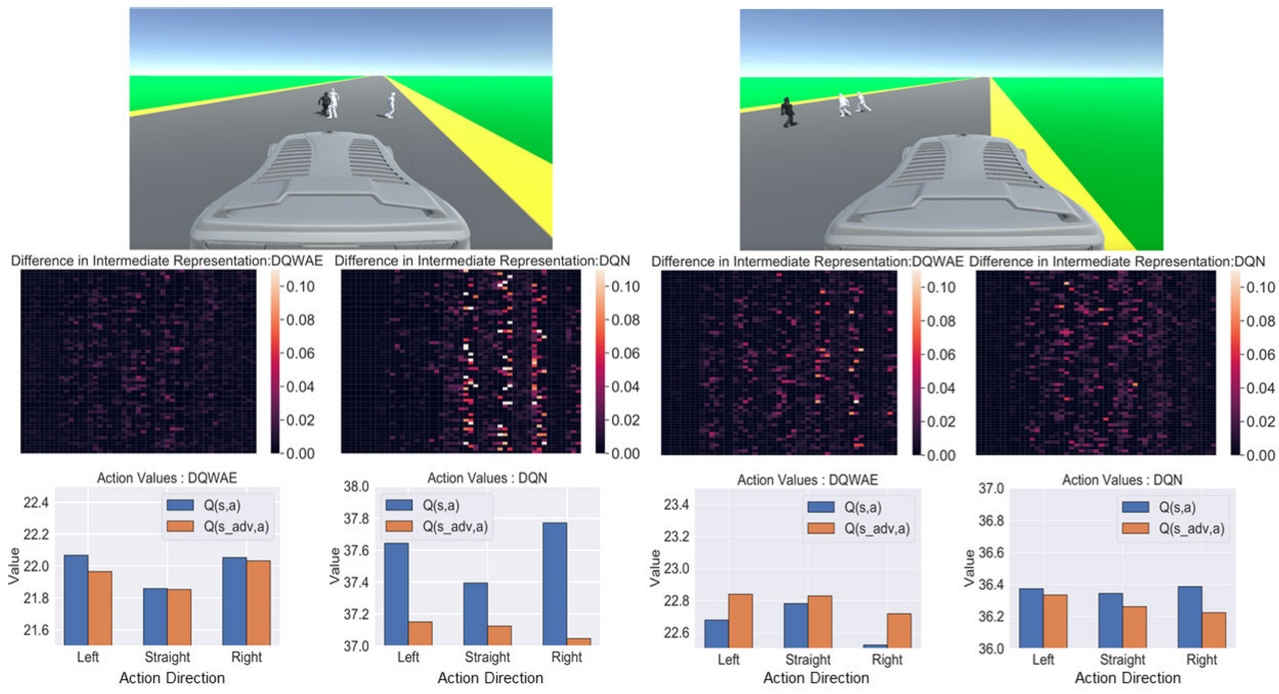


(a) Random noise (b) White-box attack

FIGURE 3. Average scores obtained in autonomous driving task (solid line) and standard deviations (shaded area).

TABLE 3. Atari summary scores against *FGSM black-box* attack.

Environment		Breakout	Pong	Boxing	Chopper	StarGunner	Qbert	Robotank
L_2-norm attack(FGSM) scale at eval.		0.024	0.11	1.2	0.24	0.059	0.99	0.99
DQN (original)	Noiseless Score	320.9 ± 16.4	20.5 ± 0.3	86.9 ± 5.8	3806.0 ± 388.2	6852.7 ± 1182.6	3819.8 ± 674.6	26.6 ± 2.1
	Attack Score	222.1 ± 38.5	$\mathbf{16.4 \pm 4.1}$	29.7 ± 17.3	2898.2 ± 909.2	3649.3 ± 1161.0	1937.4 ± 1131.9	7.0 ± 2.1
DQN (random)	Noiseless Score	356.3 ± 23.4	$\mathbf{20.7 \pm 0.2}$	81.8 ± 2.6	3394.7 ± 746.2	6322.7 ± 885.0	4047.0 ± 359.3	25.8 ± 0.7
	Attack Score	253.6 ± 41.1	14.4 ± 4.6	26.5 ± 15.4	2447.7 ± 254.8	2703.5 ± 941.5	2360.1 ± 1201.0	$\mathbf{10.0 \pm 2.2}$
DQAE (ours1)	Noiseless Score	339.9 ± 18.6	20.4 ± 0.3	87.3 ± 3.7	3998.0 ± 882.7	6636.0 ± 1652.0	3985.3 ± 306.0	26.2 ± 2.2
	Attack Score	257.8 ± 33.6	14.5 ± 4.1	33.2 ± 16.6	2908.7 ± 625.0	3858.4 ± 1057.7	1812.9 ± 956.6	8.0 ± 2.3
DQWAE (ours2)	Noiseless Score	$\mathbf{372.6 \pm 30.5}$	20.4 ± 0.2	$\mathbf{90.5 \pm 1.5}$	$\mathbf{4987.3 \pm 1992.6}$	$\mathbf{13210.7 \pm 2657.7}$	4101.0 ± 107.1	$\mathbf{30.4 \pm 3.9}$
	Attack Score	$\mathbf{292.7 \pm 26.5}$	12.7 ± 5.5	$\mathbf{41.8 \pm 15.5}$	$\mathbf{3533.7 \pm 998.4}$	$\mathbf{6902.3 \pm 2325.7}$	$\mathbf{2589.9 \pm 982.9}$	8.1 ± 2.6
DDQN-R [38]	Noiseless Score	243.8 ± 17.1	19.2 ± 1.5	85.9 ± 5.4	3520.0 ± 598.8	4700.7 ± 835.9	$\mathbf{4293.3 \pm 199.3}$	29.0 ± 1.8
	Attack Score	164.6 ± 31.2	6.0 ± 11.5	28.3 ± 14.7	2236.8 ± 658.0	3391.8 ± 775.3	1814.8 ± 1311.0	6.3 ± 2.1



(a) A dangerous situation (b) A safe situation

FIGURE 4. Comparison between our DQWAE and original DQN, in terms of internal representation (middle panels) and Q-values (lower panels) in our driving simulation.

DQAE, and original DQN, respectively. The horizontal and vertical axes denote the noise strength applied to the test images and the score of the test navigation, respectively. We found that the DQAE and DQWAE were more robust to the two types of noise, random and adversarial noise (white-box attacks), compared to the baseline DQN. Similar to the Atari 2600 results, the performance with zero noise was also improved by our DQAE and DQWAE, indicating that learning was accelerated by our regularization methods. The DQAE and DQWAE had comparable robustness against random noises. The DQWAE exhibited slightly higher robustness against white-box attacks than the DQAE. We thus concluded that, in general, the DQWAE was more noise-robust than the DQAE.

Figure 4 depicts how our DQWAE worked to compensate for adversarial attacks in the input images of two typical situations: dangerous (Fig. 4(a)) and safe (Fig. 4(b)), in comparison with the original DQN's behaviors. The changes in the internal representation (an output of the shared encoder; middle panels), i.e., $|f(s) - f(s_{adv})|$ and changes in action values (an output of the DQN part; bottom panels) when adversarial attacks with $\epsilon\sqrt{d} = 0.09$ were applied to the camera image are displayed. The top panels display the input images captured by the car camera. The car was required to take a careful action to avoid hitting the bright pedestrians crossing the street (Fig. 4(a)); and because almost all the bright pedestrians had crossed the street, the driving situation was significantly easier (Fig. 4(b)). When an adversarial attack was launched on the input camera image, our DQWAE compensated for such noise at the level of internal representation (the middle left panel in Fig. 4(a)), leading to consistently good controls in terms of DQN outputs (the bottom left panel in Fig. 4(a)), regardless of the situation. Note that in situation (a), steering right/left is the optimal action to avoid collisions with pedestrians. Conversely, the adversarial attack on the DQN was effective, especially in a dangerous situation.

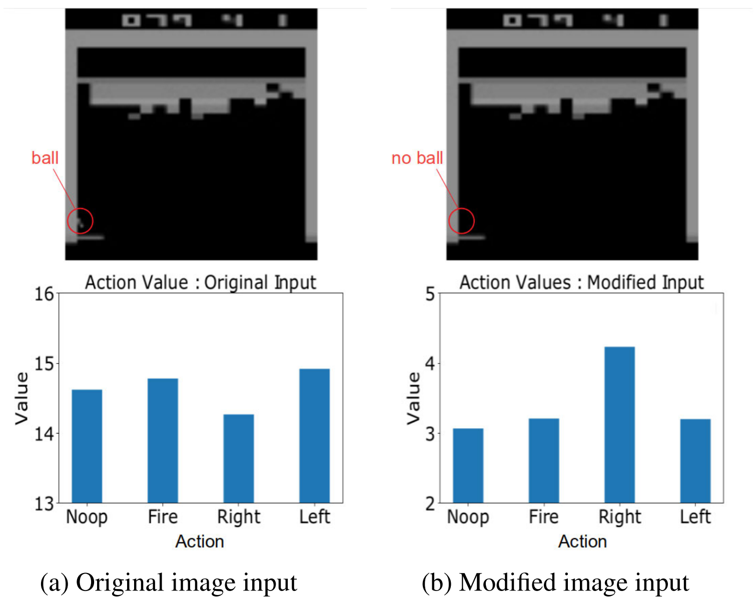


FIGURE 5. Two different input images in breakout and the corresponding action values output by DQWAE.

Due to the adversarial attack, the internal representation was disturbed (the middle right panel in Fig. 4(a)), causing an ineffective control action mediated by substantially lowered action values by the DQN (the bottom right panel in Fig. 4(a)). Although the output of the noiseless DQN was optimal (blue bars in the bottom right panel in Fig. 4(a)), it was significantly disturbed by the adversarial attack (orange bars in the bottom right panel in Fig. 4(a)).

V. DISCUSSION AND FUTURE WORK

On the Atari benchmarks and in the automatic driving simulation, the proposed deep adversarial reinforcement learning proved to be effective. The couple of decoders of AE and DWN in the DQWAE (Fig. 1(b)) require additional computational costs during the training phase, but not in the testing phase. In the application phase, the computational cost of our methods were exactly the same as that of the DQN, whereas they were significantly superior in terms of noise robustness and noiselessness. The increased computation in the training phase is for forward-propagation through the decoders of the AE and W-networks and back-propagation through the AE learning and adversarial example generation by the W-network. They can be implemented efficiently in the usual GPU-based deep learning environments. We thus concluded that our methods can be practical considering their high performance/robustness at the expense of additional, but not considerable, computation for training.

It is a point of contemplation whether the strong denoising function of the AE network [16], [18] can suppress the transfer of useful information in the input images to the DQN decoder, thus degrading the performance of the DQAE and DQWAE. This is not the case, however, because the back-propagation optimization based on the DQN objective function worked against such unfavorable representation learning by the shared encoder. Figure 5 shows the Q-values when our DQWAE was trained for Breakout; the left and right figures show when the original game image (upper left panel) was input to the trained DQWAE and when the ball pixels were erased from the image (upper right panel) input to the DQWAE, respectively; thus, the useful information in the input was artificially erased from the right figure. As expected, when the ball disappeared, the action was disturbed and the Q-values worsened significantly (right figures). On the other hand, the action was good with reasonably high Q-values when the original image was input (left figures); note here that that ''Noop,'' ''Fire,'' and ''Left'' were all good but ''Right'' was not good in this situation, because the last action could cause the approaching ball to be missed. This difference in the Q-values suggested that the small but important information in the input image space was maintained by the shared encoder of the DQWAE, although its representation learning had been affected by the AE denoising learning. Based on this observation, we deduced that the AE network was effective at removing noise from the input space, while retaining useful information, for the DQN decoder.

However, this study has some significant limitations that should be addressed in the near future. The proposed deep adversarial RL methods were applied to problems with discrete action spaces in this study. It will be necessary to modify the definitions of the adversarial examples to some extent for the continuous action problems, for various reasons including incorporating the action continuity around the best or even worst actions; such extensions are now under development as policy gradient-type and actor-critic-type methods.

Another important direction would be to explore possible real-world applications. Although we implemented our methods in the simulation environment, their applicability in realistic/real environments should be examined. Our methods based on off-policy RL and image-based input setting would be suitable for application to real-world problems, such as what registered in the Waymo open dataset [45]. If we restrict the situations to the real world, such as the case of pedestrians crossing the street in this study, we will be able to obtain reasonably good controllers by our methods. However, the fully automatic driving scenario in the real environment should include a significantly wider variety of situations, some of which may not have occurred frequently in the training dataset. Moreover, in the actual automatic driving settings, the dimensionalities of input and control are significantly larger than those in our simulation setting. To resolve these difficulties and cope with noise robustness in other realistic scenarios, it is essential to make our methods expandable to real-world problems.

Our adversarial RL method has achieved reasonably good robustness against observation-based random/adversarial perturbations, possibly owing to sensor noises and/or unexpected and abrupt changes in the environment. Drastic and rather long-term changes to the environment, such as in the case of heavy rain or snow in the automatic navigation scenario, can be modeled as a domain change; therefore, realizing controller robustness against domain changes remains an important and attractive research topic. Compensation for domain changes, termed domain adaptation, is one of the important topics in the field of image processing. Extension of our current methods to those with adversarial learning in the feature space, instead of the input space, may lead to such adversarial domain-adaptive RL.

VI. CONCLUSION

In this study, we presented a couple of regularization methods for the DRL agent to acquire robust internal representation by compensating for input noise in the encoder component in a fully data-driven manner. Specifically, we presented two architectures, a simple one (DQAE) and an advanced one (DQWAE), based on the modification of the well-known DQN architecture. When applied to seven Atari 2600 games, we observed that our advanced DQWAE demonstrated excellent robustness against random noise and white-box and black-box attacks. The improvement over the three baseline methods and our simple DQAE was significant. When applied to our original driving simulation, the robustness of the DQAE and DQWAE was superior to that of the baseline DQN. Interestingly, in most of the examined Atari 2600 games and the driving simulation task, the DQWAE outperformed the baselines, even without the addition of noise to the input images. This suggests that the policy learning itself was accelerated by our advanced regularization method.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 25, Dec. 2012, pp. 1097–1105.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Dec. 2013.

[3] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

[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, pp. 354–359, Oct. 2017.

[5] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *J. Mach. Learn. Res.*, vol. 4, pp. 1107–1149, Dec. 2003.

[6] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 3756–3762.

[7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2014, pp. 1–10.

[8] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," 2017, *arXiv:1702.02284*. [Online]. Available: http://arxiv.org/abs/1702.02284

[9] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *Int. J., Ser., Solid Mech., Strength Mater.*, vol. 33, no. 4, pp. 468–473, Dec. 2014.

[10] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *Int. J. Robot. Res.*, vol. 37, nos. 4–5, pp. 421–436, 2017.

[11] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 39, pp. 1–40, 2016.

[12] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Apr. 2017, pp. 506–519.

[13] I. Ilahi, M. Usama, J. Qadir, M. Umar Janjua, A. Al-Fuqaha, D. Thai Hoang, and D. Niyato, "Challenges and countermeasures for adversarial attacks on deep reinforcement learning," 2020, *arXiv:2001.09684*. [Online]. Available: http://arxiv.org/abs/2001.09684

[14] E. Shelhamer, P. Mahmoudieh, M. Argus, and T. Darrell, "Loss is its own reward: Self-supervision for reinforcement learning," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–9.

[15] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–14.

[16] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 1096–1103.

[17] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[18] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized denoising auto-encoders as generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 899–907.

[19] D. J. Im, S. Ahn, R. Memisevic, and Y. Bengio, "Denoising criterion for variational auto-encoding framework," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 2059–2065.

[20] J. Zhang and C. Li, "Adversarial examples: Opportunities and challenges," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2578–2593, Jul. 2020.

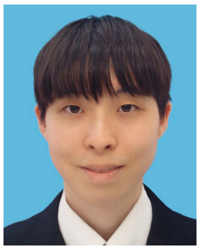
[21] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016, *arXiv:1607.02533*. [Online]. Available: http://arxiv.org/abs/1607.02533

[22] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Mar. 2016, pp. 372–387.

[23] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2574–2582.

[24] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, Nov. 2017, pp. 1–14.

[25] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–17.
[26] A. Eyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, vol. 5, 2018, pp. 2137–2146.
[27] C. Xie, M. Tan, B. Gong, J. Wang, A. L. Yuille, and Q. V. Le, "Adversarial examples improve image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 819–828.
[28] T. Miyato, S.-I. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: A regularization method for supervised and semi-supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1979–1993, Aug. 2019.
[29] S. Qiao, W. Shen, Z. Zhang, B. Wang, and A. Yuille, "Deep co-training for semi-supervised image recognition," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 135–152.
[30] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.
[31] J. Morimoto and K. Doya, "Robust reinforcement learning," *Neural Comput.*, vol. 17, no. 2, pp. 335–359, 2005.
[32] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, Aug. 2017, pp. 2817–2826.
[33] A. Mandlekar, Y. Zhu, A. Garg, L. Fei-Fei, and S. Savarese, "Adversarially robust policy learning: Active construction of physically-plausible perturbations," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 3932–3939.
[34] C. Xiao, X. Pan, W. He, J. Peng, M. Sun, J. Yi, M. Liu, B. Li, and D. Song, "Characterizing attacks on deep reinforcement learning," 2019, *arXiv:1907.09470*. [Online]. Available: http://arxiv.org/abs/1907.09470
[35] B. Lütjens, M. Everett, and J. P. How, "Certified adversarial robustness for deep reinforcement learning," in *Proc. Conf. Robot. Learn.*, 2020, pp. 1328–1337.
[36] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, "Adversarial policies: Attacking deep reinforcement learning," in *Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–16.
[37] H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. Boning, and C.-J. Hsieh, "Robust deep reinforcement learning against adversarial perturbations on state observations," 2020, *arXiv:2003.08938*. [Online]. Available: http://arxiv.org/abs/2003.08938
[38] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," in *Proc. Int. Joint Conf. Auton. Agents Multiagent Syst.*, vol. 3, 2018, pp. 1–15.
[39] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org
[40] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," in *Proc. Int. Joint Conf. Artif. Intell.*, Jan. 2015, pp. 4148–4152.
[41] S. Tokui, K. Oono, S. Hido, and J. Clayton, "Chainer: A next-generation open source framework for deep learning," in *Proc. 29th Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 1–6.
[42] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A general platform for intelligent agents," 2018, *arXiv:1809.02627*. [Online]. Available: http://arxiv.org/abs/1809.02627
[43] Q. Zhang, T. Du, and C. Tian, "Self-driving scale car trained by deep reinforcement learning," 2019, *arXiv:1909.03467*. [Online]. Available: http://arxiv.org/abs/1909.03467
[44] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," *CoRR*, vol. abs/1710.02298, pp. 1–14, Oct. 2017.
[45] *Waymo Open Dataset*. [Online]. Available: https://waymo.com/open/download/



KOHEI OHASHI received the B.E. degree from Kyoto University, Kyoto, Japan, in 2020, where he is currently pursuing the M.E. degree with the Graduate School of Informatics. His research interests include deep reinforcement learning and adversarial training.

KOSUKE NAKANISHI received the B.E. and M.E. degrees from The University of Tokyo, in 2010 and 2012, respectively. He is currently pursuing the Ph.D. degree with the Graduate School of Informatics, Kyoto University, Kyoto, Japan. He worked as a Simulation Engineer at Sony, from 2012 to 2016. He has been an Autonomous Vehicle Engineer at the Honda Research and Development Innovative Research Center, since 2016. His research interests include intelligent vehicles/robotics and reinforcement learning.

WATARU SASAKI received the B.E. and M.E. degrees from Kyoto University, in 2017 and 2019, respectively. He is currently working with Nomura Securities Company Ltd. His research interests include deep reinforcement learning and adversarial training.

YUJI YASUI received the B.E. and M.E. degrees in mechanical engineering from the Tokyo University of Science, Japan, in 1992 and 1994, respectively, and the Ph.D. degree from Sophia University, Japan, in 2012. He joined Honda Research and Development Company Ltd., in 1994. He has been researching powertrain control for low-emission vehicles and HEVs, traction control for F-1 racing car, transmission control and device control using adaptive control, model predictive control, and neural networks. He is currently an Executive Chief Engineer with the Research Group for Automated Driving and Advanced Driving Assistance System using AI and advanced control technologies.

SHIN ISHII received the B.E., M.E., and Ph.D. degrees from The University of Tokyo, in 1986, 1988, and 1997, respectively. He was an Associate Professor with the Nara Institute of Science and Technology, Nara, Japan, in 1997, and a Professor, in 2001. He has been a Full Professor with the Graduate School of Informatics, Kyoto University, Kyoto, Japan, since 2007, and the Director of ATR Neural Information Analysis Laboratories, Kyoto, since 2018. His research interests include machine learning, computational neuroscience, and intelligent robotics.