2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Mitigating Adversarial Perturbations for Deep Reinforcement Learning via Vector Quantization

Tung M. Luu[1], Thanh Nguyen[1], Tee Joshua Tian Jin[1], Sungwoon Kim[2], and Chang D. Yoo[1*]

Abstract— **Recent studies reveal that well-performing reinforcement learning (RL) agents in training often lack resilience against adversarial perturbations during deployment. This highlights the importance of building a robust agent before deploying it in the real world. Most prior works focus on developing robust training-based procedures to tackle this problem, including enhancing the robustness of the deep neural network component itself or adversarially training the agent on strong attacks. In this work, we instead study an input transformation-based defense for RL. Specifically, we propose using a variant of vector quantization (VQ) as a transformation for input observations, which is then used to reduce the space of adversarial attacks during testing, resulting in the transformed observations being less affected by attacks. Our method is computationally efficient and seamlessly integrates with adversarial training, further enhancing the robustness of RL agents against adversarial attacks. Through extensive experiments in multiple environments, we demonstrate that using VQ as the input transformation effectively defends against adversarial attacks on the agent's observations.**

I. INTRODUCTION

Modern deep reinforcement learning (RL) agents [33], [10], [14] typically rely on deep neural networks (DNN) as powerful function approximators. Nevertheless, it has been discovered that even a well-trained RL agent may drastically fail under the small adversarial perturbations in the input during deployment [15], [27], [18], [1], [37], making it risky to execute on safety-critical applications such as autonomous driving [50]. Therefore, it is necessary to develop techniques to assist the RL agents in resisting adversarial attacks in input observations before deploying them into the real world.

There have been many works proposed in the literature in defending against adversarial attacks on input observations. A line of work focuses on enhancing the robustness of DNN components by enforcing properties such as invariance and smoothness via regularization schemes [42], [53], [35], [49], resulting in deep policy outputs that exhibit similar actions under bounded perturbations. Another line of work considers training the RL agent in an adversarial manner, where an adversary is introduced to perturb the agent's input while it interacts with an environment. Sampled trajectories under these attacks are subsequently used for training, resulting in a more resilient RL agent. In this approach, the perturbation can be induced from the policy/value function [18], [2], [37], [25] or more recently, it can be generated by another RL-based adversary [52], [43]. While training with RL-based attackers can attain high long-term rewards under attacks, it often requires extra samples and computations for training.

Aforementioned strategies can be regarded as robust training-based defenses, aimed at learning resilient policy networks against adversarial attacks. Meanwhile, in the field of image classification, there are also numerous input transformation-based defenses [6], [26], [12], [38], [13], [17], [40] that mitigate such attacks without altering the underlying model. These defenses attempt to reduce adversarial perturbations in the input by transforming it before feeding to the model. The transformation process commonly involves denoisers for purifying perturbations [13], [17], [40], [34] or simply utilizes image processing techniques to weaken the effect of attacks [6], [29], [12], [48], [38]. Therefore, this approach potentially benefits RL agents without requiring significant changes to underlying RL algorithms. However, denoiser-based transformations often leverage powerful generative models such as GAN [41], [17] or diffusion model [34] to remove noise, which may introduce overhead in both training and inference for RL agents. On the other hand, the processing-based transformations are appealing due to their non-differential nature, making it challenging for adversaries to circumvent the defenses. Additionally, these transforms are also cost-efficient and versatile, making them suitable for use with RL agents. Nonetheless, many of these transformations are tailored to image data [6], [12], [48], [38] and may not easily extend to vector inputs such as low-dimensional states in continuous control tasks.

Motivated by this limitation, we propose using a variant of vector quantization (VQ) as a suitable input transformation-based defense for RL, which is generally applicable for both image input and continuous state. The key idea of our approach is to utilize VQ for discretizing the observation space and subsequently train the RL agent within this transformed space. This strategy effectively reduces the space of adversarial attacks [15], [18], [53] that can impact the agent's observations, producing transformed inputs that are minimally affected by attacks. Our proposed approach is computationally efficient and modifies only the input rather than the model itself, allowing it to synergistically complement other robust training-based defenses and enhance the

*Corresponding author: Chang D. Yoo
[1]School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, Republic of Korea. {tungluu2203,thanhnguyen,joshuateetj,cd_yoo}@kaist.ac.kr
[2]Department of Artificial Intelligence, Korea University, 145 Anam-ro, Seongbuk-gu, Seoul 02841, Republic of Korea. swkim01@korea.ac.kr

This work was partly supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2021-0-01381, Development of Causal AI through Video Understanding and Reinforcement Learning, and Its Applications to Real Environments) and partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00184, Development and Study of AI Technologies to Inexpensively Conform to Evolving Policy on Ethics).

overall robustness of the RL agent.

The main contributions of the paper are as follows: (i) we propose a novel input transformation-based defense for RL agent using VQ, (ii) we introduce an effective way to incorporate the proposed defense in the RL algorithms, and (iii) we demonstrate through extensive experiments that our proposed method effectively mitigates adversarial attacks on many environments across domains and settings. The code can be found at https://github.com/tunglm2203/vq_robust_rl.

II. RELATED WORK

Adversarial Attacks on State Observations. Since the discovery of adversarial examples in the classification [44], vulnerabilities in state observations of deep RL were first demonstrated by [15], [27], [18]. [15] evaluated the robustness of DQN agents in the Atari domain using FGSM [11] to attack at each step. Instead, [18] proposed using the value function to determine when to launch attack. [27] concentrated on attacking within specific steps of trajectories and employed a planner to craft perturbations that steer the agent toward a target state. [1] explored the black-box setting, revealing transferable adversarial examples across different DQN models. In contrast to crafting perturbations solely based on the policy, [37] introduced a more potent attack leveraging both the policy and the Q function. Recently, [53] formalized attacks on observations through a state-adversarial Markov decision process, demonstrating that the most powerful attacks can be learned as an RL problem. Based on this, [52] and [43] introduced the RL-based adversaries for black-box and white-box attacks, respectively.

Robust Training for Deep RL. To enhance the robustness of RL agents against adversarial attacks on observations, previous works have primarily focused on strategies involving adversarial examples during training. [18]; [2] are concurrent works that first proposed to adversarially train DQN agents on Atari games. They used weak attacks on pixel space during rollouts and preserved perturbed frames for training. However, this approach exhibited limited improvements in several Atari games. Another line of research introduces a regularization-based approach to enhancing the robustness of DQN agents. [39] proposed Lipschitz regularization, while [53] used a hinge loss regularizer to promote the smoothness of Q function under bounded perturbations. [35] utilized robustness verification bound tools to compute the lower bound of Q function, thereby certifying the robustness of action selection. In continuous control tasks, a similar adversarial training approach was initially explored by [15], where attacks are induced from both policy and Q function, and the trajectories sampled under attacks are used for training. However, recent work [53] found that this approach may not reliably improve the robustness against new attacks. [52] proposed an alternative training paradigm involving LSTM-based RL agents and a black-box RL-based adversary. Similarly, [43] proposed the same training paradigm with the white-box RL adversary, leading to a more robust RL agent. Smoothness regularization has also been proposed to improve the robustness of the policy model in online RL setting [42], [53], as well as in offline setting [49].

Input Transformation Based Defenses. In the domain of image classification, aside from robust training methods [31], [51], there have been many studies on defending adversarial attacks through input transformations [29], [12], [48], [38], [41], [13], [17], [40], [34]. Several works utilized traditional image processing such as image cropping [12], rescaling [29], or bit depth reduction [48] to mitigate the impact of adversarial attacks on the classifier. Other methods employed the powerful generative models [41], [17], [34] or trained the denoisers [26], [13] to reconstruct clean images. However, given our focus on control tasks using RL, it is more appropriate to adopt cost-efficient techniques for countering attacks. Furthermore, denoisers composed of DNNs are also vulnerable to gradient-based attacks. Motivated by the utilization of image processing techniques, we propose to use VQ as an input transformation. Notably, unlike bit depth reduction [48] that employs uniform quantization, our method learns representative points to quantize inputs based on the statistic of input examples.

Input Transformation on Deep RL. Input transformation has been widely investigated in deep RL to enhance generalization [45], [5] or improve sample efficiency [21], [19], [30]. Domain randomization, as proposed in [45], aims to transfer policies from simulators to the real world. Simple augmentations like cutout [5] or random convolution [23], as demonstrated in [5] and [23], have been shown to assist agents in generalizing to unseen environments. To reduce sample complexity in pixel-based RL, [21] applied image augmentations to the observations during agent training. Furthermore, [19] employed augmentation to regularize Q functions, further improving sample efficiency. Vanilla vector quantization (VQ) has been utilized in several works to reduce the state space for generalization in tabular RL [7], [22] and continuous control [32]. Our proposed method differs from these works by quantizing individual dimensions rather than entire vectors, making it more scalable. To the best of our knowledge, our use of input transformation represents the first attempt at leveraging it to enhance robustness against adversarial attacks in RL.

III. PRELIMINARIES

A. Reinforcement Learning.

An reinforcement learning (RL) environment is modeled by a Markov decision process (MDP), defined as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, P, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition probability distribution, and $\gamma \in [0, 1)$ is a discount factor. An agent takes actions based on a policy $\pi : \mathcal{S} \to \mathcal{A}$. The objective of the RL agent is to maximize the expected discounted return $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})]$, which is the expected cumulative sum of rewards when following the policy π in the MDP. This objective can be evaluated by a value function $V^\pi(s) := \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R_t | s_0 = s]$, or the action value function $Q^\pi(s, a) := \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R_t | s_0 = s, a_0 = a]$.

B. Test-time Adversarial Attacks.

We consider adversarial attacks on the state observations during test time, which is formulated as SA-MDP [53]. Specifically, during testing, the agent's observation is adversarially perturbed at every time step by an adversary equipped with a certain budget ϵ. Note that, the adversary only alters the observations and the true underlying states of the environment do not change. This setting fits many realistic scenarios such as measurement errors, noise in sensory signals, or man-in-the-middle (MITM) attacks for a deep RL system. For example, in robotic manipulation, an attacker can add imperceptible noise to the camera capturing an object, however, the actual object's location is unchanged. In this paper, we consider a ℓ_∞ norm threat model, in which the adversary is restricted to perturb the observation s into $\hat{s} \in \mathcal{B}(s,\epsilon) = \{\hat{s} : \|s - \hat{s}\|_\infty \leq \epsilon\}$. Additionally, since the adversary only appears at the test time, we assume that the true states can be observed during training. This is important since our input transformation is learned to capture the statistic of states while training the agent.

C. Vector Quantization.

Vector quantization (VQ) is a common technique widely used for learning discrete representation [46], [36], [28], [16]. In this work, we use VQ block similar to VQ-VAE [46] with some modifications. We present the process of basic VQ here and leave the modification in the next section. Initially, the VQ block, referred to as $\mathcal{Q}$, maintains a codebook $\mathbf{C}$ consisting of a set of items $\{c_k\}_{k=1}^K$. Given an input vector z, $\mathcal{Q}$ outputs the item c_m which is closest to z in Euclidean distance as $c_m = \mathcal{Q}(z)$, with $m = \text{argmin}_k \|c_k - z\|_2$. The codebook item can be updated by ℓ_2 error or the exponential moving average to move the item toward corresponding unquantized vectors assigned to that item. In the backward pass, the VQ block is treated as an identity function, referred to as straight-through gradient estimation [3]. The hyper-parameter K controls the size of the codebook, where lower values would lead to lossy compression but potentially yield more abstract representations.

IV. METHODOLOGY

A. Input Transformation based Defense for RL

To understand the effectiveness of input transformation-based defense for the RL agent, we commence by analyzing its performance under adversarial perturbations utilizing the tools developed in SA-MDP [53]. Let f_1, f_2 be functions mapping $\mathcal{S} \to \mathcal{S}$, and let π represent a Gaussian policy with a constant independent variance. Assuming the policy network is L-Lipschitz continuous, we obtain:

$$\max_{s\in\mathcal{S}}\{V^{\pi\circ f_1}(s) - V^{\pi\circ f_2\circ\nu}(s)\} \leq \zeta \max_{s\in\mathcal{S}} \max_{\hat{s}\in\mathcal{B}(s,\epsilon)} \|f_1(s) - f_2(\hat{s})\|_2 \tag{1}$$

where, ν is optimal adversary corresponding to π, ζ is a constant independent of π. The proof of Eq. (1) relies on L-Lipschitz continuity of the policy network.

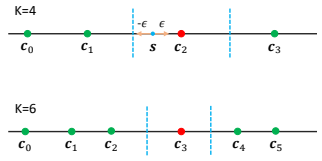
Eq. (1) suggests two distinct approaches for narrowing the gap between natural performance and performance under perturbation. Firstly, when considering f_1 as an identity function, we can design f_2 to reconstruct s from $\hat{s}$, which involves minimizing $\max_{\hat{s}\in\mathcal{B}(s,\epsilon)} \|s - f_2(\hat{s})\|_2$. Secondly, we can design f_1 and f_2 to reduce the difference in the transformed space, as expressed by $\max_{\hat{s}\in\mathcal{B}(s,\epsilon)} \|f_1(s) - f_2(\hat{s})\|_2$ being small. While the first approach can be achieved by utilizing a denoiser, it carries certain disadvantages, as discussed in Section II. Therefore, we opt for the second approach to counter adversarial attacks. In this approach, $V^{\pi\circ f_1}(s)$ is not guaranteed to be identical to $V^\pi(s)$, as the agent operates within the transformed space rather than the origin one. However, as long as f_1 retains the essential information from the original space, the agent's performance can be maintained, as shown in our experiments.

B. VQ Mitigating Adversarial Perturbations

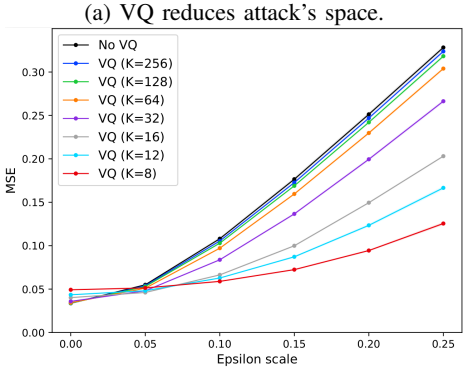
Different to previous works [46], [28], [16], which use VQ to discretize the latent space, we directly apply VQ to each dimension of the raw input space as a transformation, and then train the RL agents directly on the transformed inputs. We define the space of adversarial attack in transformed space by $\bar{\mathcal{B}}(s,\epsilon) = \{\mathcal{Q}(\hat{s}) : \|s - \hat{s}\|_\infty \leq \epsilon\}$. Intuitively, $\bar{\mathcal{B}}(s,\epsilon)$ is a set of possible items c_k to which the perturbed state $\hat{s}$ can be assigned. We find that using VQ with an appropriate small codebook size as the input transformation effectively reduces the space of adversarial attacks, *i.e.*, the size of $\bar{\mathcal{B}}$, without significantly reducing the natural performance of the agent. As depicted in Fig. 1a (top) for one-dimensional data, supposing the state s is assigned to the item c_2, and the adversary can arbitrarily perturb the state s within the ϵ ball. We can see that if the $\mathcal{B}(s,\epsilon)$ is still lying within the boundaries of c_2 (the blue dotted lines), $\mathcal{Q}$ will transform both s and $\hat{s} \in \mathcal{B}(s,\epsilon)$ into the same item, *i.e.*, $\mathcal{Q}(s) = \mathcal{Q}(\hat{s})$ for $\forall \hat{s} \in \mathcal{B}(s,\epsilon)$. Additionally, we also observe that the space of attacks is proportional to the size of the codebook. It means that K decreases leading to smaller size of $\bar{\mathcal{B}}(s,\epsilon)$, thus stronger in resisting the adversarial perturbations. This is illustrated in Fig. 1a, larger K shrinks the radius of items, while smaller K enlarges the radius. Moreover, due to straight-through estimation, VQ also inherits non-differential properties as image transformations. For states lying close to the boundary, with appropriate small K and not too large ϵ, the transformed states are altered at most to the closest neighbor items.

To better understand the effectiveness of VQ for countering adversarial attacks, we illustrate it with a toy regression task. We train a predictor π_θ to regress from the state to the action on the `walker-medium-v2` dataset [8], using VQ as input transformation. The model is optimized by minimizing the MSE between the prediction and ground truth action. At the test time, we introduce the adversary on the state s to obtain the perturbed state $\hat{s} = \text{argmax}_{\hat{s}\in\mathcal{B}(s,\epsilon)} \|\pi_\theta(s) - a\|_2$, with a is the ground truth. The maximization is solved by using 10-step projected gradient descent (PGD) as in [20], [37]. We evaluate the performance (*i.e.*, MSE) under different scales of ϵ, where $\epsilon = 0$ corresponds to $\hat{s} = s$.

As the result shown in Fig. 1b, smaller values of K are



(a) VQ reduces attack's space.



(b) VQ as input transformation for regression.

Fig. 1: (a) Illustration of using VQ to reduce space of adversarial attacks. The green and red dots indicate codebook items, whereas the red dot represents an item to which the state s is assigned after VQ process. The blue dotted line indicates the boundaries. (b) Illustration of the effectiveness of VQ in countering attacks in the regression task.

more effective in countering the adversarial attacks, while the robustness of the model with larger K will become closer to the model without using VQ. Additionally, we also observe that smaller K will slightly decrease the natural performance (*i.e.*, $\epsilon = 0$), which is resulted from the lossy compression. Therefore, we can adjust K to control the trade-off between the natural performance and the robustness.

The VQ block originally uses the same codebook for all dimensions of the input [46], [36], [28], [16]. However, since we use VQ for the raw input observations instead of latent space, which is not learnable to adapt the codebook items, fitting the same codebook for every dimension might limit its expressiveness to approximate the state density, especially for the small size of the codebook. We propose to use separate codebooks for each dimension of input to improve its expressiveness. Therefore, our modified VQ block maintains a set of codebooks $\{\mathbf{C}^i\}_{i=1}^{D}$, where D is the dimension of inputs, each codebook consists of a set of item $\{c_k\}_{k=1}^{K}$, with $c_k \in \mathbb{R}$. The codebook items are updated by ℓ_2 error similar to [46]. Specifically, considering the dimension d, let $\{s_{k,i}^d\}_{i=1}^{N_{d,k}}$ be the set of state elements closest to the item c_k^d, so that the codebook items are updated by minimizing following objective:

$$\mathcal{L}_{VQ} = \frac{1}{D}\frac{1}{K}\sum_{d=1}^{D}\sum_{k=1}^{K}\frac{1}{N_{d,k}}\sum_{i=1}^{N_{d,k}}\|s_{k,i}^d - c_k^d\|_2^2 \tag{2}$$

While the VQ transformation described above is appealing for this approach, naively incorporating it into the RL algorithms may deteriorate the natural performance. This is because the state distribution induced by the policy is changed over the course of training, which is caused by the policy being continuously improved, thus the state distribution at the beginning may be very different at the end of training. As a result, the codebooks learned from the data induced by a low-performance policy at the beginning might be inadequate to reflect the state distribution induced by the high-performance policies at the end. Moreover, codebook items may converge to a local minimum similar to K-means [4]. To mitigate this, we propose to slowly update codebooks when the agent performance is low, *i.e.*, when the agent is in exploration; and update faster when it reaches higher performance, *i.e.*, when the agent is in exploitation. To control the rate of update, we scale $\mathcal{L}_{VQ}$ based on the current agent's performance. For simplicity, we approximate current performance by using the average value of Q within a mini-batch during training. Then, we scale $\mathcal{L}_{VQ}$ in Eq. (2) by the factor λ defined as follows:

$$\lambda = \frac{\frac{1}{|B|}\sum_{s_i \in B}|Q^{\pi}(s_i, \pi(s_i))|}{\alpha} \tag{3}$$

where α is hyper-parameter, π is the current training policy, B is mini-batch. During training, we alternatively update between the RL agent and the codebooks.

V. Experiments

A. Evaluation in MuJoCo

In this section, we evaluate the effectiveness of the proposed method against common adversarial attacks in both online and offline RL settings. We adopt three common attacks as used in [53], [52], namely *Random*, *Action Diff*, and *Min Q*. Given an attack budget ϵ and a state s, adversaries generate perturbed state $\hat{s}$ as follows: (1) *Random*: $\hat{s}$ is uniformly sampled within $\mathcal{B}(s, \epsilon)$, (2) *Action Diff*: $\hat{s}$ is induced from the agent's policy. Specifically, $\hat{s}$ is searched within $\mathcal{B}(s, \epsilon)$ to satisfy $\max_{\hat{s}\in\mathcal{B}(s,\epsilon)} D(\pi(\cdot|s)||\pi(\cdot|\hat{s}))$, with D is KL divergence, (3) *Min Q*: Different from *Action Diff*, this adversary generates perturbations based on both the agent's policy and Q function, which is a relatively stronger attack. $\hat{s}$ is selected to satisfy $\min_{\hat{s}\in\mathcal{B}(s,\epsilon)} Q(s, a)$, with a is the policy output. For *Action Diff* and *Min Q*, we use 10-step PGD as in [20], [37].

1) Online RL: For online RL setting, we follow the experiment setup as in [53], but use Soft Actor-Critic (SAC) as a base RL algorithm due to its sample efficiency and higher performance on Gym MuJoCo [14]. We conduct experiments on five environments including Walker2d, Hopper, Ant, Reacher, and InvertedPendulum. For α in Eq. (3), we search in $\{30, 40, 50, 60\}$ and set a value of 30 for Hopper and 60 for the others. Regarding the codebook size, we set $K = 8$ for Walker2d and Reacher, and $K = 16$ for the others. We compare the proposed method with vanilla SAC and SAC with bit depth reduction [48] (SAC-BDR), where uniform quantization is naively performed in each dimension of the input. Additionally, we also incorporate VQ into a strong adversarial training baseline [53], referred to as SAC-SA, to see whether it can further improve the robustness.

We evaluate the robustness of methods under different scales of ϵ, where $\epsilon = 0$ corresponds to the natural performance (see in Appendix), and generate the return

TABLE I: Average robustness score and standard deviation in Mujoco for online RL setting over five seeds. In each environment, we bold the highest average score in each setting: with and without robust training-based defense.

Env.	Method	Random	Action Diff	Min Q	Average
Walker2d	SAC	84.4 ± 5	42.1 ± 4	40.4 ± 3	55.7
	SAC-BDR	88.8 ± 4	48.6 ± 6	43.8 ± 7	60.4
	SAC-VQ	88.9 ± 4	65.4 ± 8	50.9 ± 7	**68.4**
	SAC-SA	91.6 ± 1	66.3 ± 7	47.7 ± 7	68.6
	SAC-SA-BDR	91.1 ± 2	72.1 ± 2	50.9 ± 3	71.4
	SAC-SA-VQ	92.5 ± 5	77.5 ± 6	52.9 ± 6	**74.3**
Hopper	SAC	65.2 ± 7	36.8 ± 5	36.4 ± 4	46.1
	SAC-BDR	72.7 ± 5	45.7 ± 6	44.0 ± 3	54.1
	SAC-VQ	72.5 ± 6	50.2 ± 5	45.5 ± 5	**56.2**
	SAC-SA	91.7 ± 3	67.1 ± 4	61.8 ± 4	73.5
	SAC-SA-BDR	90.3 ± 3	68.6 ± 4	63.55 ± 4	74.2
	SAC-SA-VQ	91.2 ± 8	70.2 ± 7	64.42 ± 5	**75.3**
Ant	SAC	71.5 ± 4	30.8 ± 3	32.02 ± 3	44.8
	SAC-BDR	68.0 ± 6	34.9 ± 4	31.12 ± 5	44.7
	SAC-VQ	78.5 ± 8	41.9 ± 7	39.34 ± 6	**53.2**
	SAC-SA	85.3 ± 3	50.3 ± 7	53.73 ± 5	63.1
	SAC-SA-BDR	81.5 ± 4	54.6 ± 5	52.06 ± 6	62.7
	SAC-SA-VQ	86.7 ± 3	59.7 ± 4	55.15 ± 2	**67.2**
Reacher	SAC	99.2 ± 0.2	94.9 ± 0.7	95.27 ± 0.6	96.5
	SAC-BDR	99.2 ± 0.2	96.4 ± 0.6	96.36 ± 0.4	97.3
	SAC-VQ	99.3 ± 0.1	97.2 ± 0.1	96.89 ± 0.2	**97.8**
	SAC-SA	99.6 ± 0.1	98.1 ± 0.2	97.02 ± 0.3	98.3
	SAC-SA-BDR	99.2 ± 0.4	98.1 ± 0.5	97.81 ± 0.5	98.4
	SAC-SA-VQ	99.6 ± 0.1	98.9 ± 0.1	97.92 ± 0.2	**98.8**
Pendulum	SAC	88.2 ± 12	60.2 ± 11	69.28 ± 13.4	72.5
	SAC-BDR	99.8 ± 3	80.5 ± 10	88.47 ± 6.9	89.6
	SAC-VQ	100 ± 0	91.1 ± 4	94.98 ± 2.1	**95.3**
	SAC-SA	100 ± 0	88.3 ± 1	62.49 ± 3.7	83.6
	SAC-SA-BDR	100 ± 0	92.2 ± 3	58.91 ± 4.6	83.7
	SAC-SA-VQ	100 ± 0	97.8 ± 1	59.75 ± 7.4	**85.8**

curves under different attack levels. For better quantitative measurement, we consider the *robustness score* (RC) defined as the areas under the perturbation curve, where the returns are normalized between R_{min} and R_{max}. Specifically, given a normalized perturbation curve, the RC is computed by $RC = \frac{1}{N}\sum_{i=1}^{N} \mathbf{R}[i]$, with $\mathbf{R}$ is the list of returns evaluated on N monotonically increasing attack scales, each return value is averaged over 50 episodes similar to [53].

According to the result shown in Tab. I, our method consistently enhances the robustness score over vanilla SAC, with an averaged improvement of 11%. While SAC-BDR succeeds in enhancing the robustness of SAC, it falls short by 5% compared to ours, highlighting the advantage of learning codebooks over uniform quantization. When coupled with SAC-SA, the robustness can be further enhanced. However, in the case of the InvertedPendulum under the *Min Q* attack, VQ exacerbates performance degradation. This could be attributed to the use of smoothness regularization, which also negatively impacts performance compared to vanilla SAC. Therefore, the combination with VQ may degrade performance. Remarkably, SAC-VQ achieves comparable performance with SAC-SA in the Walker2d environment and surpasses it in the InvertedPendulum. Overall, utilizing VQ as input transformation effectively mitigates the impact of attacks and further enhances the robustness of robust training-based defenses.

2) Offline RL: For offline RL setting, we use TD3BC [9] as a baseline and conduct on {walker2d, hopper, ant}-medium-v2 datasets. Following a similar comparison in the online setting, we incorporate VQ into a robust training-based defense [53], [49]. We search for K within $\{8, 12, 16\}$ for all datasets, except for hopper-medium-v2 where we broaden the search to $\{8, 16, 24, 28, 32\}$ and report the best score. This setting presents more challenges compared to the online setting due to the state distribution shift problem [24]. This problem arises because codebooks learned from the offline dataset might inaccurately reflect the density of states induced by the current policy at test time. We investigate whether VQ is helpful for improving the robustness in this challenging setting. The evaluation metric is same with online setting. As shown in Tab. II, we observe that VQ is still able to enhance the robustness of base algorithms in almost tasks. However, the magnitude of improvements is lower compared to the online setting. The most substantial improvement are observed in the Walker2d and Ant datasets.

TABLE II: Average robustness score and standard deviation in Mujoco for offline RL setting over five seeds. In each environment, we bold the highest average score in each setting: with and without robust training-based defense

Env.	Method	Random	Action Diff	Min Q	Average
Walker2d	TD3BC	81.6 ± 2.2	57.7 ± 3.2	38.4 ± 2	59.3
	TD3BC-VQ	83.0 ± 1.3	70.7 ± 1.6	47.8 ± 4	**67.2**
	TD3BC-SA	84.4 ± 1.3	68.9 ± 2	43.0 ± 3.4	65.5
	TD3BC-SA-VQ	85.0 ± 0.4	74.3 ± 1.8	45.1 ± 3.3	**68.1**
Hopper	TD3BC	53.8 ± 0.8	43.7 ± 1.1	21.0 ± 3.1	39.6
	TD3BC-VQ	52.2 ± 1.2	44.2 ± 0.7	27.6 ± 2.9	**41.4**
	TD3BC-SA	54.4 ± 0.9	45.7 ± 0.6	23.5 ± 1.5	**41.2**
	TD3BC-SA-VQ	52.6 ± 4.9	44.5 ± 3	26.2 ± 1.6	41.1
Ant	TD3BC	84.8 ± 5.3	52.6 ± 5.6	56.3 ± 4.5	64.6
	TD3BC-VQ	83.0 ± 7.2	58.8 ± 4.9	60.5 ± 4.7	**67.5**
	TD3BC-SA	88.7 ± 4.1	60.2 ± 3.6	61.9 ± 4.1	70.3
	TD3BC-SA-VQ	92.7 ± 3.2	70.7 ± 3.1	68.8 ± 1.4	**77.4**

B. Evaluation in Atari

We investigate effectiveness of VQ into the Double DQN [47] on two Atari games: Freeway and Pong. These environments feature high-dimensional pixel inputs and discrete action spaces. We set $K = 4$ for these environments. For the robustness evaluation, we use 10-step l_∞-PGD untargeted attack. Additionally, we also incorporate bit depth reduction for the DQN agent, referred as DQN-BDR. Furthermore, we integrate the proposed method into a state-of-the-art method, namely RADIAL [35], to investigate whether it can enhance robustness. As demonstrated in Tab. III, the DQN-VQ is more effective compared to DQN-BDR. This outcome underscores the advantages of learning codebooks over uniform quantization. Surprisingly, our method is able to achieve comparable with RADIAL without adversarial training. By combining VQ with RADIAL, we achieve further improvement in robustness, especially at larger values of ϵ such as 10/255 for Pong and 5/255 for Freeway.

C. Ablation Study

Effectiveness of Codebook Size. We provide the experiment showing the effectiveness of different codebook sizes in Fig. 2. Across environments, the small values of K often lead to

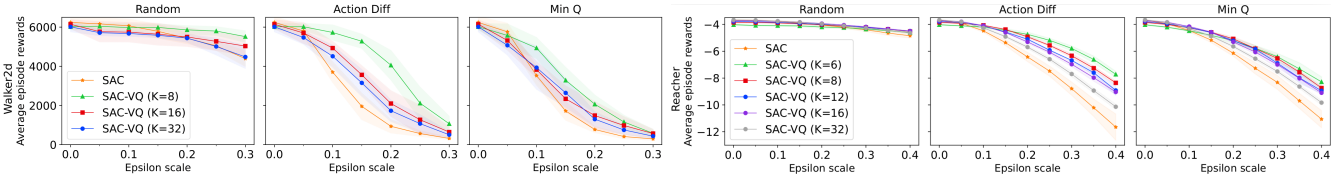


Fig. 2: The comparison between agents using different sizes of the codebook on Walker2d and Reacher.

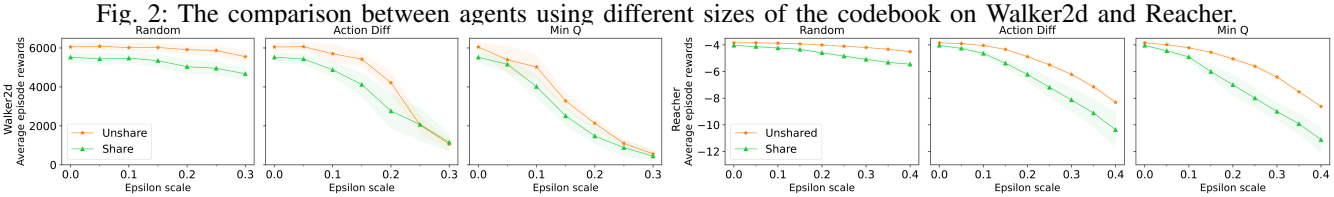


Fig. 3: The comparison between sharing and separate codebooks for all dimensions of states on Walker2d and Reacher.

TABLE III: The mean reward of 10 runs ± the standard error in the Atari domain. Each run is averaged over 20 episodes. The highest score in each column of environments is bold.

Env.	Method/Metric	Natural Reward	PGD	
ϵ		0	3/255	5/255
Freeway	DQN	**33.9 ± 0.07**	0.0 ± 0.0	0.0 ± 0.0
	DQN-BDR	33.2 ± 0.1	32.7 ± 0.6	27.8 ± 0.7
	DQN-VQ	33.5 ± 0.7	32.9 ± 0.8	28.1 ± 1.7
	RADIAL	33.4 ± 0.7	**33.4 ± 0.6**	28.5 ± 0.9
	RADIAL-BDR	**33.9 ± 0.9**	33.0 ± 0.8	30.5 ± 0.9
	RADIAL-VQ	**33.9 ± 0.4**	33.2 ± 0.7	**32.4 ± 1.1**
ϵ		0	5/255	10/255
Pong	DQN	-21.0 ± 0.0	-21.0 ± 0.0	-21.0 ± 0.0
	DQN-BDR	20.9 ± 0.4	14.7 ± 5.5	-21.0 ± 0.0
	DQN-VQ	**21.0 ± 0.0**	20.4 ± 0.9	-20.3 ± 0.5
	RADIAL	**21.0 ± 0.0**	**21.0 ± 0.0**	-20.8 ± 0.4
	RADIAL-BDR	**21.0 ± 0.0**	**21.0 ± 0.0**	16.5 ± 0.9
	RADIAL-VQ	**21.0 ± 0.0**	**21.0 ± 0.0**	**21.0 ± 0.0**

more robustness in a wide range of attack scales. However, too small K causes a drop in natural performance, and the large value of K (e.g., 32) tends to have little benefit for resisting perturbations as analyzed in Section IV-B. It is important to emphasize that the dynamics of different environments can vary significantly, which in turn affects the distribution of states. Therefore, selecting an appropriate value of K for each environment is crucial to achieving robust performance.

Effectiveness of Separate Codebooks. The Fig. 3 compares performance between shared and unshared (*i.e.*, separate) codebook for each dimension. The result shows that using the shared codebook across dimensions can reduce natural performance due to its limited expressiveness. Consequently, the robustness under attacks is also decreased.

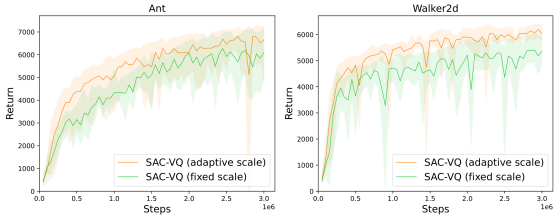


Fig. 4: Ablation on adaptive learning codebook.

Adaptive Learning Codebook. To demonstrate the effectiveness of adaptive scale during updating codebooks, we show the natural performance during training of Walker2d and Ant in Fig. 4. The "*fixed scale*" means no scale used for $\mathcal{L}_{VQ}$ loss. The result shows that adaptive scale is important to achieve high natural performance at the end of training.

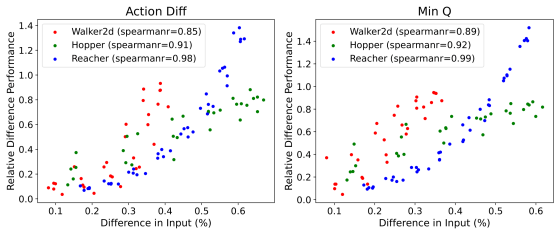


Fig. 5: The correlation between the input difference and relative difference of performance.

Input Difference vs. Robustness. We measure Spearman's rank correlation coefficient between (1) the difference between clean and perturbed inputs after quantized and (2) the relative difference between the natural and robust performance under perturbation. The result shown in Fig. 5 indicates a high correlation between the two quantities. This supports Eq. (1) that decreasing the input difference will increase its robustness performance.

TABLE IV: Average training time on Walker2d-v2.

Method	Runtime (s/iteration)
SAC	0.0220
SAC-VQ	0.0232
SAC-SA	0.0326
SAC-SA-VQ	0.0350

D. Computational Cost Comparison.

We compare training time when using VQ transformation on a single machine with one GPU (RTX 3080). The result is shown in Tab. IV. When combined with vanilla SAC and SAC-SA, VQ slightly increases the training time to 5% and 7%, respectively.

VI. Conclusion

We have presented a novel defense based on input transformation to counter adversarial attacks on state observations. Our proposed approach is both cost-efficient and highly effective in defending against such attacks. Furthermore, when combined with robust training-based defenses, it significantly enhances the overall robustness of RL agents. We

have conducted thorough analyses to evaluate the efficacy of vector quantization in countering attacks. To the best of our knowledge, this is the first study to investigate the use of input transformation-based defenses for RL.

REFERENCES

[1] Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *MLDM*, 2017.

[2] Vahid Behzadan and Arslan Munir. Whatever does not kill deep reinforcement learning, makes it stronger. *arXiv:1712.09344*, 2017.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv:1308.3432*, 2013.

[4] Leon Bottou and Yoshua Bengio. Convergence properties of the k-means algorithms. *NeurIPS*, 1994.

[5] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *ICML*, 2019.

[6] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. *arXiv:1608.00853*, 2016.

[7] Fernando Fernández and Daniel Borrajo. Vqql. applying vector quantization to reinforcement learning. In *RoboCup-99: Robot Soccer World Cup III 3*, pages 292–303. Springer, 2000.

[8] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv:2004.07219*, 2020.

[9] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *NeurIPS*, 2021.

[10] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *ICML*, 2018.

[11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv:1412.6572*, 2014.

[12] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *ICLR*, 2018.

[13] Puneet Gupta and Esa Rahtu. Ciidefence: Defeating adversarial attacks by fusing class-specific image inpainting and image denoising. In *ICCV*, 2019.

[14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

[15] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv:1702.02284*, 2017.

[16] Riashat Islam, Hongyu Zang, Anirudh Goyal, Alex Lamb, Kenji Kawaguchi, Xin Li, Romain Laroche, Yoshua Bengio, and Remi Tachet Des Combes. Discrete factorial representations as an abstraction for goal conditioned reinforcement learning. *arXiv:2211.00247*, 2022.

[17] Guoqing Jin, Shiwei Shen, Dongming Zhang, Feng Dai, and Yongdong Zhang. Ape-gan: Adversarial perturbation elimination with gan. In *ICASSP*, 2019.

[18] Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. *arXiv:1705.06452*, 2017.

[19] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv:2004.13649*, 2020.

[20] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv:1611.01236*, 2016.

[21] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *NeurIPS*, 2020.

[22] HYK Lau, KL Mak, and ISK Lee. Adaptive vector quantization for reinforcement learning. *IFAC Proceedings Volumes*, 2002.

[23] Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. *ICLR*, 2020.

[24] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv:2005.01643*, 2020.

[25] Yongyuan Liang, Yanchao Sun, Ruijie Zheng, and Furong Huang. Efficient adversarial training without attacking: Worst-case-aware robust reinforcement learning. *NeurIPS*, 2022.

[26] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *CVPR*, 2018.

[27] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv:1703.06748*, 2017.

[28] Dianbo Liu, Alex M Lamb, Kenji Kawaguchi, Anirudh Goyal ALIAS PARTH GOYAL, Chen Sun, Michael C Mozer, and Yoshua Bengio. Discrete-valued neural communication. *NeurIPS*, 2021.

[29] Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. *arXiv:1707.03501*, 2017.

[30] Tung M Luu, Thanh Nguyen, Thang Vu, and Chang D Yoo. Utilizing skipped frames in action repeats for improving sample efficiency in reinforcement learning. *IEEE Access*, 2022.

[31] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083*, 2017.

[32] Christos N Mavridis and John S Baras. Vector quantization for adaptive state aggregation in reinforcement learning. In *ACC*, 2021.

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

[34] Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Anima Anandkumar. Diffusion models for adversarial purification. *ICML*, 2022.

[35] Tuomas Oikarinen, Wang Zhang, Alexandre Megretski, Luca Daniel, and Tsui-Wei Weng. Robust deep reinforcement learning through adversarial loss. *NeurIPS*, 2021.

[36] Sherjil Ozair, Yazhe Li, Ali Razavi, Ioannis Antonoglou, Aaron Van Den Oord, and Oriol Vinyals. Vector quantized models for planning. In *ICML*, 2021.

[37] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. *AAMAS*, 2018.

[38] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Deflecting adversarial attacks with pixel deflection. In *CVPR*, 2018.

[39] Alessio Russo and Alexandre Proutiere. Optimal attacks on reinforcement learning policies. *ACC*, 2021.

[40] Hadi Salman, Mingjie Sun, Greg Yang, Ashish Kapoor, and J Zico Kolter. Denoised smoothing: A provable defense for pretrained classifiers. *NeurIPS*, 2020.

[41] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *ICLR*, 2018.

[42] Qianli Shen, Yan Li, Haoming Jiang, Zhaoran Wang, and Tuo Zhao. Deep reinforcement learning with robust and smooth policy. In *ICML*, 2020.

[43] Yanchao Sun, Ruijie Zheng, Yongyuan Liang, and Furong Huang. Who is the strongest enemy? towards optimal and efficient evasion attacks in deep rl. *ICLR*, 2022.

[44] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv:1312.6199*, 2013.

[45] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, 2017.

[46] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *NeurIPS*, 2017.

[47] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, 2016.

[48] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *NDSS*, 2018.

[49] Rui Yang, Chenjia Bai, Xiaoteng Ma, Zhaoran Wang, Chongjie Zhang, and Lei Han. Rorl: Robust offline reinforcement learning via conservative smoothing. In *NeurIPS*, 2022.

[50] Changxi You, Jianbo Lu, Dimitar Filev, and Panagiotis Tsiotras. Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning. *Rob. Auton. Syst.*, 2019.

[51] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019.

[52] Huan Zhang, Hongge Chen, Duane Boning, and Cho-Jui Hsieh. Robust reinforcement learning on state observations with learned optimal adversary. *ICLR*, 2021.
[53] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. *NeurIPS*, 2020.

APPENDIX

Performance bound. SA-MDP [53] provides an upper bound of performance gap between the two policies trained on non-adversarial MDP and state-adversarial MDP (SA-MDP), respectively. We based on this to derive our upper bound in Eq. (1). Formally, given a policy π and its value function $V^{\pi}(s)$, under the optimal adversary ν in SA-MDP, theorem 5 in [53] stated that:

$$\max_{s\in\mathcal{S}}\{V^{\pi}(s)-V^{\pi\circ\nu}(s)\}\leq\kappa\max_{s\in\mathcal{S}}\max_{\hat{s}\in\mathcal{B}(s,\epsilon)}D_{TV}(\pi(\cdot|s),\pi(\cdot|\hat{s})) \tag{4}$$

where, $D_{TV}(\pi(\cdot|s),\pi(\cdot|\hat{s}))$ is the total variance distance between $\pi(\cdot|s)$ and $\pi(\cdot|\hat{s})$, κ is a constant that does not depend on π, $\mathcal{B}(s,\epsilon)=\{\hat{s}:\|s-\hat{s}\|_\infty\leq\epsilon\}$, and $\pi\circ\nu$ denotes the policy under perturbations: $\pi(a|\nu(s))$. The total variation distance is not easy to compute for most distributions, thus we upper bound D_{TV} by the *KL* divergence:

$$D_{TV}(\pi(\cdot|s),\pi(\cdot|\hat{s}))\leq\sqrt{\frac{1}{2}KL\left(\pi(\cdot|s)\parallel\pi(\cdot|\hat{s})\right)}. \tag{5}$$

We assume that the policy is Gaussian with constant independence variance, which is commonly used in RL algorithms such as TD3 [10]. Supposing that $\pi(\cdot|s)\sim\mathcal{N}(\mu_s,\Sigma_s)$ and $\pi(\cdot|\hat{s})\sim\mathcal{N}(\mu_{\hat{s}},\Sigma_{\hat{s}})$, where $\mu\in\mathbb{R}^d$ and $\mu_s,\mu_{\hat{s}}$ are respectively produced by neural networks $\mu_\theta(s),\mu_\theta(\hat{s})$, and Σ is a diagonal matrix independent of state s (*i.e.*, $\Sigma_s=\Sigma_{\hat{s}}=\Sigma$). Assuming policy network is L-Lipschitz continuous, we have:

$$\begin{aligned}KL(\pi(\cdot|s)\parallel\pi(\cdot|\hat{s}))&=\frac{1}{2}(\log\frac{|\Sigma_{\hat{s}}|}{|\Sigma_s|}-d+tr(\Sigma_{\hat{s}}^{-1}\Sigma_s)\\&\qquad+(\mu_{\hat{s}}-\mu_s)^\top\Sigma_{\hat{s}}^{-1}(\mu_{\hat{s}}-\mu_s))\\&\leq C^2\|\mu_{\hat{s}}-\mu_s\|_2^2\\&=C^2\|\mu_\theta(s)-\mu_\theta(\hat{s})\|_2^2\\&\leq C^2L\|s-\hat{s}\|_2^2.\end{aligned} \tag{6}$$

The first inequality because of $\Sigma_{\hat{s}},\Sigma_s$ are positive, thus exist $C\in\mathbb{R}^+$ to satisfy this. The second inequality because of the policy is L-Lipschitz continuous. Now we introduce two function f_1,f_2 that map $\mathcal{S}\to\mathcal{S}$, and apply these two functions into input states s and $\hat{s}$, respectively. We have:

$$KL\left(\pi(\cdot|f_1(s))\parallel\pi(\cdot|f_2(\hat{s}))\right)\leq C^2L\|f_1(s)-f_2(\hat{s})\|_2^2. \tag{7}$$

Combining (4), (5), (6), and (7) we have:

$$\max_{s\in\mathcal{S}}\{V^{\pi\circ f_1}(s)-V^{\pi\circ f_2\circ\nu}(s)\}\leq\zeta\max_{s\in\mathcal{S}}\max_{\hat{s}\in\mathcal{B}(s,\epsilon)}\|f_1(s)-f_2(\hat{s})\|_2 \tag{8}$$

where, $\zeta=\frac{1}{\sqrt{2}}\kappa C\sqrt{L}$, $\pi\circ f_1$ and $\pi\circ f_2\circ\nu$ denote $\pi(\cdot|f_1(s))$ and $\pi(\cdot|f_2(\nu(\hat{s})))$, respectively. □

Robustness Evaluation Under Different Attack Scales. We provide the raw perturbation curves for online and offline RL setting in Fig. 6 and 7, respectively. These curves are used to obtain *robustness score* in Tab. I and II

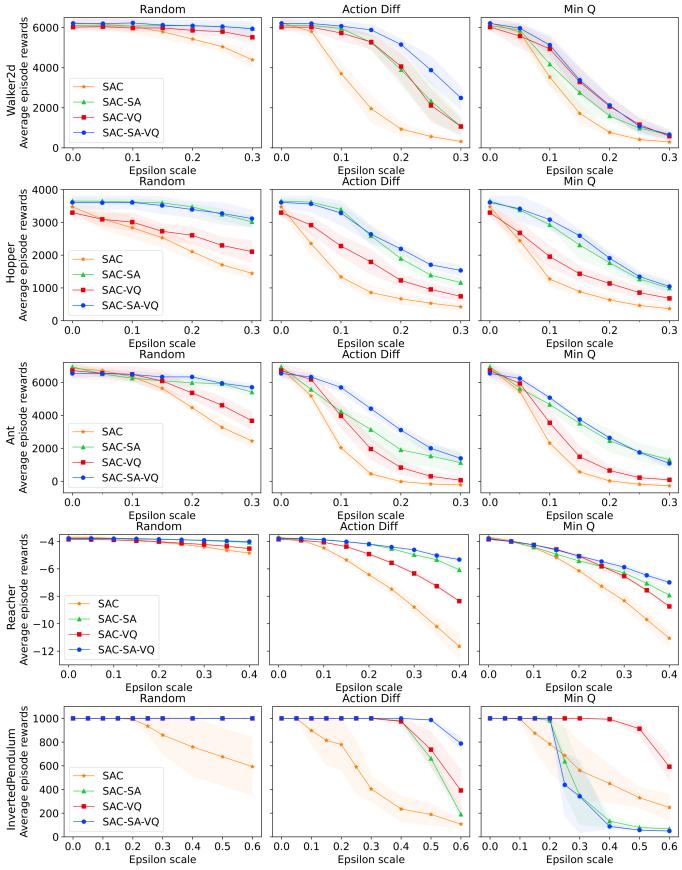


Fig. 6: Comparisons under different attacks w.r.t. different scales of ϵ in online RL setting. The y-axis indicates the unnormalized return. The curve is averaged over five seeds, with ±1 standard deviation shading.

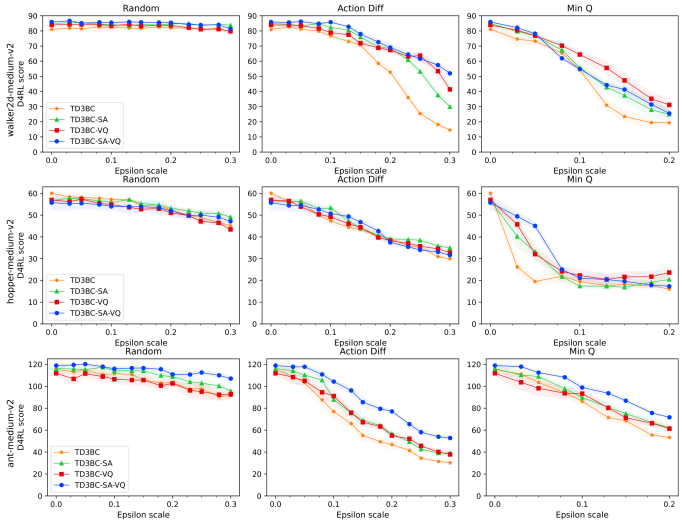


Fig. 7: Comparisons under different attacks w.r.t. different budget ϵ's in offline RL setting. The y-axis indicates normalized return. The curve is averaged over five seeds, with ±1 standard deviation shading.