Rethinking Adversarial Attacks in Reinforcement Learning from Policy Distribution Perspective

Tianyang Duan[1], Zongyuan Zhang[1], Zheng Lin[2], Yue Gao[3], Ling Xiong[4],
Yong Cui[5], Hongbin Liang[6], Xianhao Chen[2], Heming Cui[1], Dong Huang[1,*]
[1] Department of Computer Science, The University of Hong Kong, Hong Kong, China.
[2] Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong, China.
[3] School of Computer Science, Fudan University, Shanghai, China.
[4] School of Computer and Software Engineering, Xihua University, Chengdu, China.
[5] Department of Computer Science and Technology, Tsinghua University, Beijing, China.
[6] School of Transportation and Logistics, Southwest Jiaotong University, Chengdu, China.

***Abstract*—Deep Reinforcement Learning (DRL) suffers from uncertainties and inaccuracies in the observation signal in real-world applications. Adversarial attack is an effective method for evaluating the robustness of DRL agents. However, existing attack methods targeting individual sampled actions have limited impacts on the overall policy distribution, particularly in continuous action spaces. To address these limitations, we propose the Distribution-Aware Projected Gradient Descent attack (DAPGD). DAPGD uses distribution similarity as the gradient perturbation input to attack the policy network, which leverages the entire policy distribution rather than relying on individual samples. We utilize the Bhattacharyya distance in DAPGD to measure policy similarity, enabling sensitive detection of subtle but critical differences between probability distributions. Our experiment results demonstrate that DAPGD achieves SOTA results compared to the baselines in three robot navigation tasks, achieving an average 22.03% higher reward drop compared to the best baseline.**

***Index Terms*—Reinforcement learning, Deep neural network, Stochastic policy, Adversarial attack.**

I. Introduction

Deep Reinforcement Learning (DRL) has exhibited exceptional performance across a spectrum of complex robotic control tasks [1]–[3], successfully solving challenges such as robot navigation [4], [5], obstacle avoidance [6], [7], and robotic manipulation [8], [9]. However, in real-world applications, observation signals are often subject to noise, latency, and inaccuracies [10]. Moreover, subtle variations in environmental dynamics, such as changes in surface friction or lighting conditions, can substantially degrade DRL performance [11].

Adversarial attacks have emerged as an effective method for evaluating the robustness of DRL models [12], [13]. By introducing carefully designed perturbations to input data, these attacks effectively reveal the vulnerabilities of DRL models under uncertainties and environmental changes [14]. Training DRL agents in the presence of such adversarial scenarios has been shown to improve their robustness [15]. While current research on adversarial attacks primarily focuses on supervised learning tasks, such as image classification [16]–[18], the application in the domain of DRL remains relatively unexplored. Existing efforts have concentrated on developing defense mechanisms compatible with DRL frameworks [19]–[21], while little attention is paid to devising attack methods against DRL, particularly in high-dimensional continuous state and action spaces.

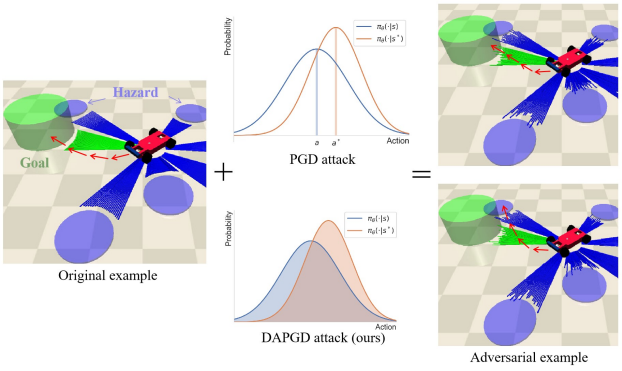

Fig. 1. Two methods for generating adversarial examples in the Goal task. In this task, the agent needs to navigate around *Hazards* and reach the *Goal*. **Top**: Existing methods (e.g., PGD) sample from the policy and calculate the sign gradient of mean square error loss to attack. **Bottom**: Our method (DAPGD) directly utilizes the policy distribution similarity, which calculates the sign gradient of the Bhattacharyya distance between policies to attack.

Unlike supervised learning, which involves deterministic input-output mappings, DRL learns a policy function that maps state space to action space, typically modeled as a conditional probability distribution over possible actions [22]. As illustrated in the top of Figure 1, existing adversarial attack methods primarily target the output, which in DRL manifests as calculating adversarial perturbations based on sampled actions from the policy (or the action with the highest probability) [23], [24]. However, perturbation attacks targeting individual actions may have a limited impact on the overall policy distribution, particularly in high-dimensional continuous state and action spaces. This is because adjacent actions within the action space can be selected to counteract the effects of attacks targeting specific actions. The inherent flexibility of continuous action spaces enables more robust adjustments in response to local perturbations. Moreover,

*Corresponding author. Email: dhuang@cs.hku.hk

existing methods often overlook the inherent randomness in action selection. Consequently, gradients of the loss function calculated from a single sampled action fail to capture the broader vulnerability of the entire policy distribution, thereby diminishing the effectiveness of the attacks.

To address these issues, we propose the *Distribution-Aware Projected Gradient Descent (DAPGD)* attack. As shown in the bottom of Figure 1, our method utilizes distribution similarity, instead of sampled outcomes, as the input for gradient perturbation to attack the policy network. This approach capitalizes on the full information of the policy distribution rather than relying solely on the local characteristics of individual samples. Furthermore, this gradient can generate more consistent perturbation effects across the observation space, resulting in adversarial samples that more closely resemble real-world disturbances. This is particularly crucial for assessing DRL robustness in practical application scenarios. In our DAPGD attack method, we employ the Bhattacharyya distance as a measure of policy similarity. The Bhattacharyya distance [25] quantifies the overlap between two probability distributions and is sensitive to marginal changes, facilitating the detection of subtle but potentially critical differences in policies. Experimental results show that DAPGD outperforms seven state-of-the-art baselines across three robotic navigation tasks. DAPGD achieves an average reward reduction 18.67% and 25.38% higher than the best baseline when attacking benign and robust models, respectively.

II. Related Work

Adversarial attacks are predominantly applied to supervised learning tasks such as image recognition [26], [27]. FGSM [26] is a basic attack method that generates adversarial samples by perturbing inputs based on the sign of gradients in a single step, evaluating the robustness of deep neural networks (DNNs). PGD [27] extends FGSM with an iterative process and projection step, generating stronger adversarial samples through multiple updates within constrained perturbations. Most current attack methods are built on PGD [28]–[30]. Momentum Iterative MI-FGSM [28] incorporates momentum and gradient iteration techniques to stabilize update directions. DI^2-FGSM [29] applies random transformations (e.g., scaling, translation, and rotation) to the input image each iteration before calculating gradients. NI-FGSM [30] introduces the Nesterov gradient acceleration technique to generate more effective adversarial samples and improve their transferability.

In DRL, Huang et al. [23] first use FGSM to study its ability to interfere with agent policies. Lin et al. [24] propose an enchanting attack to maliciously manipulate DRL agents through strategic attacks on agents over a sequence of time steps. Weng et al. [31] present a model-based DRL attack framework designed to enhance the efficacy of adversarial attacks in DRL. Other works focus on adversarial defense, primarily using adversarial examples to train agents to improve policy robustness. Fischer et al. [32] devise Robust Student-DQN (RS-DQN), a method that enables simultaneous robust online training of Q-networks and policy networks. Zhang et al. [33] proposed SA-MDP to theoretically unify the adversarial defense process in DRL. Oikarinen et al. [21] develop RADIAL-RL framework to enhance the effectiveness of adversarial defense. However, these methods compute perturbations for attack or defense based on actions sampled from the policy (or actions with the highest weights), thus failing to capture broader vulnerability across the policy distribution.

III. Methodology

A. Problem Formalization

DRL is formulated as a Markov decision process [34], which is formalized as a tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ denote the action and state spaces. Policy $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ represents a probability distribution mapping from the state space to the action space. At each time step t, the agent chooses an action $a_t \sim \pi(\cdot \mid s_t)$ based on the current state s_t and interacts with the environment. The environment transitions to a new state s_{t+1} following the state-transition probability function $P(s_t \mid s_{t+1}, a_{t+1})$ and provides a reward $R(s_t, a_t)$ to the agent. The agent's goal is to learn a policy π to maximize the expected discounted return $\mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i R(s_{t+i}, a_{t+i})\right]$, where γ denotes the discount factor. In an adversarial attack to policy, assuming $\pi_\theta(\cdot \mid s)$ represents a parameterized policy that has converged in the environment, where θ denotes the network parameters of the policy. For notational simplicity, we use $\pi(\cdot \mid s)$ as $\pi[s]$. Given a state-action pair (s, a), the goal of the adversarial attack is to find an adversarial example s^* that maximizes the loss function while satisfying a paradigm constraint [12]:

$$\arg\max_{s^*} J(s^*, a), \quad \text{s.t. } \|s^* - s\|_p \leq \epsilon, \tag{1}$$

where $s^* = s + \eta$ and η denote the adversarial perturbation, $J(s^*, a)$ represents the loss function, and $\|\cdot\|_p$ is the L_p norm. In cases where the state and action space are discrete, the loss function is the cross-entropy loss, while for continuous cases, it is the mean square error loss. L_p norm is usually the L_1 norm, L_2 norm or L_∞ norm.

B. Distribution Similarity Projected Gradient Descent

The core idea of gradient-based attack methods is to calculate the gradient of the input data and then apply small perturbations to the input data along the gradient direction to induce incorrect output from the model [26], [27]. Numerous studies have demonstrated that this method can also significantly degrade the performance of DRL. For applying gradient-based attack methods to DRL, existing approaches typically rely on policy sampling to generate adversarial examples. To motivate this, we use the PGD [27] as an example to illustrate this process. Specifically, given a state-action pair (s, a), PGD-based adversarial examples in DRL can be formulated as:

$$\begin{aligned} s^*_{k+1} &= s^*_k + \alpha \cdot \operatorname{sgn}\left(\bigtriangledown_{s^*} J\left(a^*_k, a\right)\right) \\ \text{where } \; s^*_0 &= s + \varepsilon \cdot \mathcal{N}(\mathbf{0}, \mathbf{I}), \\ a^*_k &\sim \pi_\theta\left[s^*_k\right], a \sim \pi_\theta[s], \\ k &= 0, 1, \ldots, N-1, \end{aligned} \tag{2}$$

Algorithm 1 DAPGD adversarial sample generation procedure

Input: State s, stochastic policy $\pi_\theta[s]$, iteration step α, the number of iterations N, scaling factor ε, constraint threshold ϵ.
Output: Adversarial example s^*.

$s^* \leftarrow s + \varepsilon \cdot \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $k = 0$ to $N - 1$ **do**
  Calculate the distribution similarity loss:

$$J(\pi_\theta[s^*], \pi_\theta[s]) = \int_{\mathcal{A}} \sqrt{\pi_\theta(a \mid s^*)\, \pi_\theta(a \mid s)}\, da$$

  Calculate the gradient:

$$\mathbf{grad}[s^*] = \bigtriangledown_{s^*} J(\pi_\theta[s^*], \pi_\theta[s])$$

  $s^* \leftarrow s^* + \alpha \cdot \operatorname{sgn}(\mathbf{grad}[s^*])$
  $s^* \leftarrow s^* + \operatorname{clip}(s^* - s, -\epsilon, \epsilon)$
end for
return s^*

where α denotes the iteration step size, N is the number of iterations, $\operatorname{sgn}(\cdot)$ represents the sign function, ε is the scaling factor, and $\mathcal{N}(\mathbf{0}, \mathbf{I})$ denotes the multivariate standard normal distribution. Although sampling actions to attack policies only calculate the sign gradient for one action in each iteration to generate adversarial examples. However, in continuous action spaces, agents can select other adjacent actions to counteract the impact of attacks targeting specific actions. This continuity provides a degree of fault tolerance and flexibility, allowing the policy to remain relatively stable under small perturbations. Moreover, in many practical applications, multiple near-optimal policies are common. For example, in navigation tasks, there are often several paths leading to the goal. This means that in the same state, indicating that multiple high-probability actions may exist for the same state. The sign gradient based on a single sampled action reflects only the local vulnerability of the policy, thereby limiting the effectiveness of the attack.

To overcome these limitations, we construct an optimization objective based on the similarity of policy distributions to generate adversarial examples. This design reduces the impact of randomness introduced by action sampling and leverages the complete probability information of the agent's action choice in a given state instead of individual action. Specifically, our objective is to maximize the following loss function under L_p norm constraint:

$$\arg\max_{s^*} J(\pi_\theta[s^*], \pi_\theta[s]), \text{ s.t. } \|s^* - s\|_p \leq \epsilon. \quad (3)$$

We utilize the Bhattacharyya distance [25] as a metric for measuring the similarity between policy distributions. Adversarial attacks fundamentally act as perturbations to the policy distribution, and the Bhattacharyya distance effectively quantifies the degree of overlap between two distributions. By maximizing the distance, we can induce an overall bias in the policy distribution. This approach is particularly effective in continuous action spaces, as it simultaneously impacts high-probability actions and their adjacent alternatives, thereby exposing potential vulnerabilities in the policy. Specifically, we define the distribution similarity loss, built on the Bhattacharyya distance between policy distributions, as follows:

$$J(\pi_\theta[s^*], \pi_\theta[s]) = -\ln \int_{\mathcal{A}} \sqrt{\pi_\theta(a \mid s^*)\, \pi_\theta(a \mid s)}\, da \quad (4)$$

Based on Eq. 4, the process of generating adversarial samples by DAPGD can be expressed as follows:

$$\begin{aligned} s^*_{k+1} &= s^*_k + \alpha \cdot \operatorname{sgn}\left(\bigtriangledown_{s^*} J(\pi_\theta[s^*], \pi_\theta[s])\right) \\ \text{where } \; s^*_0 &= s + \varepsilon \cdot \mathcal{N}(\mathbf{0}, \mathbf{I}), k = 0, 1, \ldots, N-1. \end{aligned} \quad (5)$$

The pseudocode for DAPGD is presented in Algorithm 1. As there are no constraints on gradient computations or iteration methods, the distribution similarity loss is highly versatile and can be integrated with most existing adversarial attack methods. Moreover, it is applicable to DRL algorithms that utilize stochastic policies, whether in discrete or continuous action spaces.

IV. Evaluation

A. Experimental Setup

Environments We conduct extensive experiments on three continuous control navigation tasks from the Safety Gymnasium framework [35]: SafetyRacecarButton1-v0 (*Button*), SafetyRacecarCircle1-v0 (*Circle*), and SafetyRacecarGoal1-v0 (*Goal*). These tasks simulate scenarios relevant to autonomous vehicle navigation, such as activating buttons, maintaining circular trajectories, and reaching targets, while adhering to safety constraints like avoiding hazards and boundaries. The agent operates with realistic car dynamics in a 2-dimensional action space (velocity and steering angle), and the state space includes multiple sensor data from lidar, accelerometer, speedometer, gyroscope, and magnetometer.

Agent and Training Configuration We evaluate the effectiveness of various adversarial attack methods on DRL agents trained for continuous control navigation tasks. In each environment, agents are trained using the Trust Region Policy Optimization (TRPO) algorithm [36]. The agents are configured with 20 conjugate gradient iterations, 15 search steps, 10 learning iterations, a discount factor γ of 0.99, a batch size of 128, and a maximum gradient norm of 50, trained for 1×10^7 steps. All networks consist of two hidden layers with 64 nodes.

Attack Methods We implement and evaluate a range of attack methods, including FGSM [26], DI^2-FGSM [29], MI-FGSM [28], NI-FGSM [30], PGD [27], TPGD [37], EOTPGD [38], and our proposed DAPGD method. For iterative methods, we conduct experiments with the number of iteration $N = 50$ or 100. We set the iteration step $\alpha = 2/255$, scaling factor $\varepsilon = 0.001$, and constraint threshold $\epsilon = 0.1$.

Evaluation Schemes Two schemes are used for performance evaluation: a) Direct attacks: Attack the models trained without noise. b) Post-defense attacks: Attack models that have undergone additional defensive training with PGD (the number of iteration $N = 50$) for 5×10^6 training steps, keeping other parameters unchanged. This setup demonstrates DAPGD's

TABLE I
AVERAGE REWARD IN DIRECT ATTACK SETTING

Method	Iters	Task		
		Goal	Circle	Button
No attack	-	26.043±0.223	42.144±0.023	9.148±0.343
FGSM	-	23.059±0.252	40.889±0.013	7.478±0.804
DI^2-FGSM	50	21.360±0.338	39.850±0.028	7.900±0.711
DI^2-FGSM	100	20.807±0.603	39.931±0.038	7.414±1.047
MI-FGSM	50	21.689±0.328	40.115±0.049	7.160±0.797
MI-FGSM	100	20.929±0.788	40.075±0.008	7.339±0.448
NI-FGSM	50	23.630±0.694	38.769±0.048	7.143±0.497
NI-FGSM	100	24.626±0.185	37.187±0.121	7.274±0.716
PGD	50	24.642±0.361	41.391±0.055	8.445±0.433
PGD	100	22.958±0.602	40.890±0.055	6.973±0.450
TPGD	50	20.336±1.314	33.870±0.079	6.546±0.815
TPGD	100	20.651±0.512	33.752±0.099	6.952±0.581
EOTPGD	50	21.031±0.996	40.272±0.007	7.097±0.642
EOTPGD	100	20.982±1.236	40.243±0.012	6.775±0.105
DAPGD (ours)	50	20.074±0.446	**33.351±0.043**	5.934±0.764
DAPGD (ours)	100	**19.965±0.813**	33.467±0.082	**5.382±0.310**

TABLE II
AVERAGE REWARD IN POST-DEFENSE ATTACKS SETTING (DEFENDED BY PGD, THE NUMBER OF ITERATION $N = 50$)

Method	Iters	Task		
		Goal	Circle	Button
PGD (defended)	50	24.746±0.574	39.855±0.098	9.291±1.079
PGD	100	22.354±0.221	39.461±0.019	8.995±0.658
FGSM	-	23.852±0.320	38.461±0.055	9.098±0.402
DI^2-FGSM	50	23.408±0.465	40.402±0.022	8.656±0.975
DI^2-FGSM	100	22.471±0.278	40.400±0.050	7.873±0.168
MI-FGSM	50	23.336±0.435	39.119±0.066	8.326±0.146
MI-FGSM	100	22.741±0.456	39.207±0.018	9.109±0.513
NI-FGSM	50	24.380±0.472	39.988±0.064	8.444±0.485
NI-FGSM	100	24.883±0.453	40.369±0.046	8.890±0.845
TPGD	50	22.456±0.855	38.114±0.038	8.422±0.195
TPGD	100	22.739±0.378	38.038±0.025	8.001±0.388
EOTPGD	50	22.990±0.273	39.326±0.046	7.825±0.325
EOTPGD	100	22.948±0.245	39.417±0.069	8.337±0.778
DAPGD (ours)	50	22.701±0.438	38.028±0.057	**7.511±0.347**
DAPGD (ours)	100	**21.179±0.151**	**37.936±0.085**	8.033±0.693

effectiveness on both benign DNNs (without adversarial training) and robust DNNs (with adversarial training). The agents' performance under various attack methods is measured by the average episode reward obtained over 1000 episodes in three independent experiments. Attacks are applied to the agents' observation, perturbing the input before it is processed by the policy network. We record the mean reward obtained by the agents under each attack scenario, with lower rewards indicating more successful attacks.

B. Overall performance of DAPGD

In Table I, all attack methods reduce the agent's average reward across the three tasks (Goal, Circle, Button) compared to the non-attacked case. The DAPGD method demonstrates the best attack performance across all tasks. DAPGD achieves an average reward reduction 18.67% higher than the best baseline.

Table II presents the attack performance under the defensed model (after adversarial training with PGD). While the agent exhibits a certain degree of robustness following defense training, DAPGD still causes a significant performance drop. DAPGD outperform all other methods across all tasks, causing the most significant reduction in rewards, average 25.38% higher than the best baseline. In the presence of defense mechanisms, DAPGD's advantage over baselines becomes even more pronounced. This is attributed to DAPGD more effectively attacking the policy distribution rather than relying solely on sampling individual actions, thus revealing vulnerabilities that PGD struggle to defend against.

To further investigate the effectiveness of the use of Bhattacharyya distance (BD) in DAPGD, we conducted an ablation study comparing the following three alternative metrics Kullback-Leibler Divergence (KL), Jensen-Shannon Divergence (JS), and 2-Wasserstein Distance (WD) [39] used in the attack. Figure 2 illustrates the performance of different

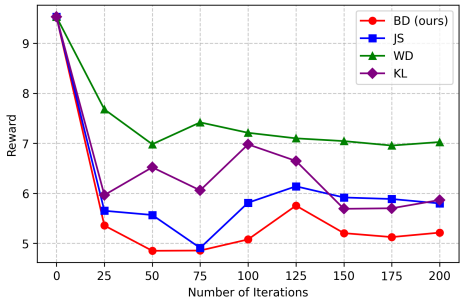

Fig. 2. Average reward obtained by the agent under each attack configuration in *Button*. Lower rewards indicate more effective attacks.

divergence metrics across different numbers of iterations of attack. BD achieves the lowest reward values across all iteration counts, indicating its ability to craft more potent attacks. The superior performance of BD can be attributed to its comprehensive capture of distribution disparities. By maximizing the BD, a broader shift in the policy distribution is induced, impacting both high-probability actions and their neighbors.

V. CONCLUSION

We propose DAPGD, a novel adversarial attack method for DRL agents. DAPGD utilizes distribution similarity and targets the entire policy distribution addressing limitations of existing attacks, especially in continuous action spaces. It use Bhattacharyya distance to measure policy similarity, enabling sensitive detection of subtle but critical differences between probability distributions. Experimental results show that DAPGD outperforms the all baselines in three navigation tasks, achieving average 22.03% higher reward drop compared to the best baseline.

REFERENCES

[1] T. Xu, Y. Liang, and G. Lan, "Crpo: A new approach for safe reinforcement learning with convergence guarantee," in *International Conference on Machine Learning*. PMLR, 2021, pp. 11 480–11 491.

[2] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *International conference on machine learning*. PMLR, 2017, pp. 22–31.

[3] L. Yang, J. Ji, J. Dai, Y. Zhang, P. Li, and G. Pan, "Cup: A conservative update policy algorithm for safe reinforcement learning," *arXiv preprint arXiv:2202.07565*, 2022.

[4] H. Surmann, C. Jestel, R. Marchel, F. Musberg, H. Elhadj, and M. Ardani, "Deep reinforcement learning for real autonomous mobile robot navigation in indoor environments," *arXiv preprint arXiv:2005.13857*, 2020.

[5] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, 2021.

[6] K. Wang, C. Mu, Z. Ni, and D. Liu, "Safe reinforcement learning and adaptive optimal control with applications to obstacle avoidance problem," *IEEE Transactions on Automation Science and Engineering*, 2023.

[7] P. Wang, R. Liu, X. Tian, X. Zhang, L. Qiao, and Y. Wang, "Obstacle avoidance for environmentally-driven usvs based on deep reinforcement learning in large-scale uncertain environments," *Ocean Engineering*, vol. 270, p. 113670, 2023.

[8] O. Kilinc and G. Montana, "Reinforcement learning for robotic manipulation using simulated locomotion demonstrations," *Machine Learning*, pp. 1–22, 2022.

[9] A. Franceschetti, E. Tosello, N. Castaman, and S. Ghidoni, "Robotic arm control and task training through deep reinforcement learning," in *International Conference on Intelligent Autonomous Systems*. Springer, 2021, pp. 532–550.

[10] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in non-stationary environments: A survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[11] S. Padakandla, "A survey of reinforcement learning algorithms for dynamically varying environments," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–25, 2021.

[12] H. Xu, Y. Ma, H.-C. Liu, D. Deb, H. Liu, J.-L. Tang, and A. K. Jain, "Adversarial attacks and defenses in images, graphs and text: A review," *International journal of automation and computing*, vol. 17, pp. 151–178, 2020.

[13] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "A survey on adversarial attacks and defences," *CAAI Transactions on Intelligence Technology*, vol. 6, no. 1, pp. 25–45, 2021.

[14] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," *arXiv preprint arXiv:1712.03632*, 2017.

[15] E. Korkmaz, "Adversarial robust deep reinforcement learning requires redefining robustness," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 7, 2023, pp. 8369–8377.

[16] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[17] A. Modas, S.-M. Moosavi-Dezfooli, and P. Frossard, "Sparsefool: a few pixels make a big difference," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 9087–9096.

[18] J. Pomponi, S. Scardapane, and A. Uncini, "Pixle: a fast and effective black-box attack based on rearranging pixels," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–7.

[19] J. Sun, T. Zhang, X. Xie, L. Ma, Y. Zheng, K. Chen, and Y. Liu, "Stealthy and efficient adversarial attacks against deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5883–5891.

[20] H. Zhang, H. Chen, D. Boning, and C.-J. Hsieh, "Robust reinforcement learning on state observations with learned optimal adversary," *arXiv preprint arXiv:2101.08452*, 2021.

[21] T. Oikarinen, W. Zhang, A. Megretski, L. Daniel, and T.-W. Weng, "Robust deep reinforcement learning through adversarial loss," *Advances in Neural Information Processing Systems*, vol. 34, pp. 26 156–26 167, 2021.

[22] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao, "Deep reinforcement learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 5064–5078, 2022.

[23] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

[24] 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," *arXiv preprint arXiv:1703.06748*, 2017.

[25] T. Kailath, "The divergence and bhattacharyya distance measures in signal selection," *IEEE transactions on communication technology*, vol. 15, no. 1, pp. 52–60, 1967.

[26] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[27] A. Madry, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[28] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9185–9193.

[29] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. L. Yuille, "Improving transferability of adversarial examples with input diversity," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 2730–2739.

[30] J. Lin, C. Song, K. He, L. Wang, and J. E. Hopcroft, "Nesterov accelerated gradient and scale invariance for adversarial attacks," *arXiv preprint arXiv:1908.06281*, 2019.

[31] T.-W. Weng, K. D. Dvijotham, J. Uesato, K. Xiao, S. Gowal, R. Stanforth, and P. Kohli, "Toward evaluating robustness of deep reinforcement learning with continuous control," in *International Conference on Learning Representations*, 2019.

[32] M. Fischer, M. Mirman, S. Stalder, and M. Vechev, "Online robustness training for deep reinforcement learning," *arXiv preprint arXiv:1911.00887*, 2019.

[33] 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," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 024–21 037, 2020.

[34] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[35] J. Ji, B. Zhang, J. Zhou, X. Pan, W. Huang, R. Sun, Y. Geng, Y. Zhong, J. Dai, and Y. Yang, "Safety gymnasium: A unified safe reinforcement learning benchmark," in *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. [Online]. Available: https://openreview.net/forum?id=WZmlxIuIGR

[36] J. Schulman, "Trust region policy optimization," *arXiv preprint arXiv:1502.05477*, 2015.

[37] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *International conference on machine learning*. PMLR, 2019, pp. 7472–7482.

[38] X. Liu, Y. Li, C. Wu, and C.-J. Hsieh, "Adv-bnn: Improved adversarial defense through robust bayesian neural network," *arXiv preprint arXiv:1810.01279*, 2018.

[39] L. Rüschendorf, "The wasserstein distance and approximation theorems," *Probability Theory and Related Fields*, vol. 70, no. 1, pp. 117–129, 1985.