Robust Deep Reinforcement Learning in Robotics via Adaptive Gradient-Masked Adversarial Attacks

Zongyuan Zhang[1], Tianyang Duan[1,*], Zheng Lin[2], Dong Huang[1], Zihan Fang[2], Zekai Sun[1], Ling Xiong[3], Hongbin Liang[4], Heming Cui[1,*], Yong Cui[5], Yue Gao[2]

***Abstract*— Deep reinforcement learning (DRL) has emerged as a promising approach for robotic control, but its real-world deployment remains challenging due to its vulnerability to environmental perturbations. Existing white-box adversarial attack methods, adapted from supervised learning, fail to effectively target DRL agents as they overlook temporal dynamics and indiscriminately perturb all state dimensions, limiting their impact on long-term rewards. To address these challenges, we propose the Adaptive Gradient-Masked Reinforcement (AGMR) Attack, a white-box attack method that combines DRL with a gradient-based soft masking mechanism to dynamically identify critical state dimensions and optimize adversarial policies. AGMR selectively allocates perturbations to the most impactful state features and incorporates a dynamic adjustment mechanism to balance exploration and exploitation during training. Extensive experiments demonstrate that AGMR outperforms state-of-the-art adversarial attack methods in degrading the performance of the victim agent and enhances the victim agent's robustness through adversarial defense mechanisms.**

I. INTRODUCTION

Robotic systems are increasingly deployed in mobile and distributed applications, such as autonomous navigation [1], intelligent transportation [2], [3], and industrial manufacturing [4]. Traditional robotic control methods have demonstrated success in structured environments with predefined tasks but face limitations in dynamic scenarios, uncertainty handling, and experiential learning. Deep reinforcement learning (DRL) has emerged as a viable alternative for robotic control [5], [6]. Unlike manually designed rule-based approaches, DRL enables agents to optimize behaviors through trial-and-error interactions with their environment. By learning policies that map states to actions to maximize long-term rewards, DRL excels in complex tasks characterized by delayed feedback and temporal dependencies.

The robustness of DRL policies is critical for real-world robotic applications, as DRL agents are highly sensitive to environmental perturbations [7]. Small input variations caused by sensor noise, environmental changes, or adversarial attacks can disrupt decision-making and lead to catastrophic failures [8]. White-box adversarial attacks are effective for assessing DRL robustness by identifying vulnerabilities in learned policies. With full access to model architecture and parameters, such attacks systematically generate perturbations to evaluate policy networks [9]. Adversarial training [10], [11], [12] using these perturbations enables deep neural networks (DNNs) to improve resilience to environmental disturbances, ensuring reliable performance in real-world scenarios.

However, existing white-box attack methods face significant challenges when targeting DRL agents, as they primarily rely on local gradient information to generate perturbations [13], [14], [15]. These methods, adapted from supervised learning, assume temporal independence and focus on instantaneous state-action mappings, overlooking the temporal dynamics of Markov Decision Processes (MDPs). Consequently, they fail to generate perturbations that effectively disrupt cumulative rewards over extended time horizons. Additionally, these methods indiscriminately apply perturbations across all states without identifying critical features that impact performance. This is particularly problematic in high-dimensional state spaces, where only a subset of variables—such as specific joint angles in robotic control—are essential for policy execution. While some approach weight perturbations based on policy gradient magnitudes [16], they do not align with the core attack objective of minimizing cumulative rewards, as agents can adapt by selecting alternative actions to maintain comparable long-term performance.

To address these challenges, we propose **A**daptive **G**radient-**M**asked **R**einforcement (AGMR) attack, a white-box method that integrates DRL with a gradient-based soft masking mechanism to dynamically identify critical state dimensions and optimize adversarial policies. AGMR introduces a soft mask function to allocate perturbations selectively across state dimensions, focusing on features that have the greatest impact on the victim agent's decision-making. Furthermore, AGMR incorporates a dynamic adjustment mechanism for the interpolation factor in the soft mask function, enabling the adversarial agent to balance exploration and exploitation during training. By leveraging gradient magnitudes to quantify the importance of state dimensions, AGMR adjusts its attack strategy in response to the evolving dynamics of the environment and task. Experimental results show that the proposed AGMR method enhances the effec-

[1] Department of Computer Science, The University of Hong Kong, Hong Kong, China.
[2] School of Computer Science, Fudan University, Shanghai, China.
[3] School of Computer and Software Engineering, Xihua University, Chengdu, China.
[4] School of Transportation and Logistics, Southwest Jiaotong University, Chengdu, China.
[5] Department of Computer Science and Technology, Tsinghua University, Beijing, China.
*T. Duan and H. Cui are the corresponding authors. Email: tyduan@cs.hku.hk (T. Duan), heming@cs.hku.hk (H. Cui)

The work is supported in part by National Key RD Program of China (2022ZD0160201), HK RGC RIF (R7030-22), HK RGC GRF (ref No.: 17208223 17204424), a Huawei flagship research grant in 2023, SuperNetAI, and the HKU-CAS Joint Laboratory for Intelligent System Software.

tiveness of adversarial attacks, and consistently outperforms state-of-the-art adversarial attack methods across several key metrics, including reward reduction, velocity reduction, and an increase in the number of falls. Additionally, AGMR demonstrates the ability to improve the robustness of the victim agent through adversarial defense mechanisms.

II. RELATED WORK

Adversarial attacks expose the vulnerabilities of DNNs by introducing carefully crafted perturbations into input data, leading to incorrect predictions during inference. These perturbations, though imperceptible to humans, can significantly alter DNN outputs [17]. White-box attacks constitute a critical category of adversarial attacks, where the adversary has full access to the model, including its architecture, parameters, and gradients. Fast Gradient Sign Method (FGSM) [18] is a seminal white-box attack that efficiently generates adversarial perturbations by leveraging the gradient of the loss function with respect to the input, addressing the computational inefficiencies of earlier approaches. Projected Gradient Descent (PGD) [12] enhances attack effectiveness through iterative gradient-based updates, projecting perturbed inputs back into the constrained space after each step. Wong et al. [19] improved attack efficiency by introducing random initialization points in FGSM-based attacks. Schwinn et al. [20] increased attack diversity by injecting noise into the output while mitigating gradient obfuscation caused by low-confidence predictions. Beyond standard white-box attacks, various techniques have been proposed to improve adversarial transferability [21], [22]. These include random input transformations [23], translation-invariant perturbation aggregation [24], and substituting momentum-based gradient updates with Nesterov accelerated gradients [25].

Adversarial attacks in DRL have been widely studied, revealing critical vulnerabilities in agent policies [26]. Huang et al. [27] demonstrated that policy-based DRL agents are highly susceptible to adversarial perturbations on state observations, showing that FGSM attacks can significantly degrade performance in Atari 2600 games. Pattanaik et al. [28] introduced adversarial examples by computing gradients of the critic network with respect to states and integrated them into the training of Deep Double Q-Network (DDQN) and Deep Deterministic Policy Gradient (DDPG), enhancing robustness. Lin et al. [29] proposed strategically timed attacks that selectively perturbed key decision-making states, achieving high attack success rates with minimal perturbations. Recent work has shifted towards theoretical modeling of adversarial attacks within the Markov Decision Process framework. Weng et al. [30] introduced a systematic evaluation framework for DRL robustness in continuous control, defining two primary threat models: observation manipulations and action manipulations. Zhang et al. [31] proposed SA-MDP, which provides a theoretical foundation for modeling state adversarial attacks within MDPs. Oikarinen et al. [16] developed RADIAL-RL, a general framework for training DRL agents to enhance resilience against adversarial attacks, and introduced Greedy Worst-Case Reward as a new evaluation metric for agent robustness.

III. METHODOLOGY

A. Problem Formalization

DRL is formalized as Markov Decision Process (MDP) [32], which is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, R, \mathcal{P}, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, and γ denotes the discount factor. At each time step t, the agent selects an action a_t from its policy $\mu(\cdot \mid s_t)$. The environment returns a reward $R(s_t, a_t)$ and transitions to the next state s_{t+1} according to the state-transition probability function $\mathcal{P}(\cdot \mid s_t, a_t)$. The state value function is defined as $V(s_t) = \mathbb{E}_{a\sim\mu, s\sim\mathcal{P}}\left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k})\right]$ which represents the expected discounted return starting from state s_t under policy μ. The agent's objective is to learn a policy μ that maximizes the expected discounted return:

$$J(\mu) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{a\sim\mu, s\sim\mathcal{P}}\left[R(s_t, a_t)\right]. \tag{1}$$

In the remainder of the paper, we assume all value functions and policies are parameterized by neural networks. We call policy $\mu(\cdot \mid s)$ as the victim policy (i.e., the policy under attack) and assume it has converged in the environment.

White-box adversarial attacks, commonly used to expose vulnerabilities in DNNs through gradient-based perturbations, naturally apply to victim policy networks in DRL [9]. To be Specific, given a victim policy $\mu(\cdot \mid s)$, the objective of an adversarial attack is to introduce a minimal perturbation η to input state s, such that the victim policy network outputs an action that maximally deviates from the original action a. This can be formulated as a constrained optimization problem:

$$\arg\max_{s^*} J(s^*, a), \quad \text{s.t. } \|\eta\|_p \leq \epsilon, \tag{2}$$

where $s^* = s + \eta$ represents the perturbed state, η denotes the adversarial perturbation, $J(\cdot, \cdot)$ is the loss function, typically mean squared error or cross-entropy. The constraint $\|\eta\|_p \leq \epsilon$ ensures that the perturbation magnitude remains within a predefined threshold $\epsilon \in (0, \infty)$, where $\|\cdot\|_p$ denotes the L_p norm.

B. Adversarial Policy Based on Reinforcement Learning

While white-box adversarial attacks have shown effectiveness in supervised learning, their direct adaptation to DRL faces significant challenges due to the temporal and sequential nature of MDPs. These methods rely on local gradient information to generate adversarial perturbations, assuming temporal independence and focusing solely on instantaneous state-action mappings. Such assumptions overlook the long-term effects of perturbations on trajectory evolution, making gradient-based perturbations ineffective over extended time horizons. Agents can often adapt by leveraging alternative actions to mitigate short-term performance degradation, thereby diminishing the attack's impact over time. Moreover, indiscriminate application of perturbations across all state dimensions fails to exploit the local importance of features, particularly in high-dimensional spaces, where

only a subset of variables—such as critical joint angles in robotic tasks—significantly impacts policy execution. In such cases, targeted perturbations on critical dimensions can lead to significant trajectory divergence or fundamental behavioral shifts, effects that existing methods fail to effectively capture.

To address these challenges, our aim is to develop a white-box adversarial attack method based on the DRL framework, which autonomously identifies and exploits vulnerable state dimensions in the victim agent's policy while considering the impact on long-term rewards. Let $\nu(\cdot \mid s, \mu)$ denote the adversarial policy (i.e., adversarial agent's policy), representing the attacker's strategy under a white-box setting, where the attacker has full access to the victim's policy μ and generates adversarial perturbations based on both the current state s and μ. Since the victim agent follows a fixed policy μ throughout the attacked rollout, it can be regarded as part of the environment's dynamics from the adversarial agent's perspective. The adversarial policy samples a perturbation $\eta_t \sim \nu(\cdot \mid s_t, \mu)$ at each time step t, which is then applied to the victim agent's observation. The victim agent selects an action according to its policy

$$a_t \sim \mu(\cdot \mid s_t + \eta_t), \tag{3}$$

where $s_t + \eta_t$ is the perturbed state.The single attack process is independent of the environment dynamics, forming a one-step sequential decision process. This interaction between the adversarial policy and the victim policy unfolds over multiple time steps, resulting in an attacked rollout that represents the sequential effects of adversarial perturbations on the victim agent's behavior. Formally, the rollout can be expressed as:

$$s_0 \xrightarrow{\nu} \eta_0 \xrightarrow{\mu} a_0 \xrightarrow{\mathcal{P}} s_1 \xrightarrow{\nu} \ldots \xrightarrow{\mathcal{P}} s_t \xrightarrow{\nu} \eta_t \xrightarrow{\mu} a_t \xrightarrow{\mathcal{P}} \ldots \tag{4}$$

In contrast to existing white-box attack methods described by Eq. 2, the adversarial agent aims to degrade the victim agent's performance by minimizing its expected return while satisfying perturbation constraints:

$$J(\nu) = \min_{\nu} \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{a\sim\mu, \eta\sim\nu, s\sim\mathcal{P}} \left[R(s_t, a_t)\right], \text{ s.t. } \|\eta\|_p \leq \epsilon, \tag{5}$$

where $R(s_t, a_t)$ represents the reward function and γ is the discount factor. To this end, we propose **A**daptive **G**radient-**M**asked **R**einforcement (AGMR) attack, a white-box method based on DRL to identify critical state dimensions via a gradient-based soft masking mechanism and optimize adversarial policies with adaptive-magnitude perturbations. AGMR employs a soft mask function, defined as $M_{\text{soft}}(s) = \beta M(s) + (1-\beta)(1-M(s))$, where $M : \mathcal{S} \to \{0, 1\}$ is a binary mask function used to identify critical state dimensions, and $\beta \in (0, 1)$ is an interpolation factor that balances the emphasis between critical and redundant dimensions. The perturbed state generated by the adversarial policy is represented as

$$\nu(\cdot \mid s, \mu) = \epsilon \cdot M_{\text{soft}}(s) \cdot \text{sign}\left(\nabla_s J(s', a)\right), \tag{6}$$

where $s' = s + \varepsilon \cdot \mathcal{N}(\mathbf{0}, \mathbf{I})$, $a \sim \mu(\cdot \mid s)$, $\varepsilon \in (0, 1]$ is the scaling factor, and $\mathcal{N}(\mathbf{0}, \mathbf{I})$ represents a multivariate standard normal distribution that introduces small perturbations to prevent gradient vanishing. As shown in Figure 1, the soft-masked attack mechanism enables AGMR to focus on critical state dimensions. AGMR generates dimension-specific perturbations, enabling targeted manipulation of state information corresponding to critical robotic components.

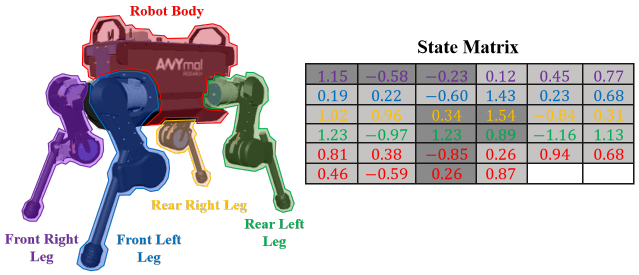


Fig. 1: Illustration of AGMR's soft-masked attack mechanism. **Left**: The robotic system is decomposed into five components (the body and four legs). **Right**: The heatmap of the state matrix visualizes perturbation magnitudes across state dimensions.

C. Automatic Dynamic Adjustment of Interpolation Factor

In the previous section, we proposed AGMR, a white-box adversarial attack method based on DRL that allocates attack magnitudes across state dimensions using a soft mask function. Selecting an appropriate interpolation factor is critical for ensuring attack effectiveness. However, during the early stages of adversarial agent training, the soft mask function may incorrectly identify critical state dimensions due to insufficient learning. This misidentification results in inefficient perturbation allocation, degrading attack performance and hindering the exploration of optimal attack policies. At this stage, a smaller interpolation factor is required to allow the adversarial agent to flexibly explore various perturbation directions. As training progresses, a larger interpolation factor becomes necessary to focus attacks on critical state dimensions, thereby imposing stronger interference on the victim policy.

Furthermore, the importance of state dimension features may shift significantly due to changes in the optimization of the adversarial policy. State dimensions that are critical at certain stages may become less important over time, while previously redundant dimensions may gain importance. These dynamic changes render a fixed interpolation factor inadequate for adapting to different training phases. Consequently, a mechanism is needed to automatically adjust the interpolation factor in response to task and environmental dynamics. Manual tuning of the interpolation factor is impractical, as different tasks and environments demand varying configurations, and the evolving importance of state dimensions is difficult to anticipate.

We propose a gradient-magnitude-based dynamic adjustment mechanism for the interpolation factor, leveraging the insight that state dimensions with larger gradient magnitudes

have a greater impact on the victim agent's decision-making process. Specifically, we define the gradient magnitude as the gradient of the objective function $J(s', a)$ denoted by Eq. 6 with respect to the state s:

$$g = \nabla_s J(s', a). \tag{7}$$

By utilizing the binary mask $M(s)$ in the adversarial agent, the gradient of the objective function can be decomposed into critical and redundant components:

$$g_{\text{critical}} = M(s) \odot g, \ g_{\text{redundant}} = (1 - M(s)) \odot g, \tag{8}$$

where $\odot$ denotes element-wise multiplication. To quantify the impact of critical and redundant dimensions, we compute the L_p norm of the gradient magnitudes, normalized by the number of corresponding dimensions:

$$\bar{g}_{\text{critical}} = \frac{\|g_{\text{critical}}\|_p}{\|M(s)\|_p}, \ \bar{g}_{\text{redundant}} = \frac{\|g_{\text{redundant}}\|_p}{\|1 - M(s)\|_p}. \tag{9}$$

The interpolation factor β is dynamically adjusted based on the relative magnitudes of the critical and redundant gradients:

$$\beta = \sigma\left(\frac{\bar{g}_{\text{critical}}}{\bar{g}_{\text{critical}} + \bar{g}_{\text{redundant}}}\right), \tag{10}$$

where $\sigma(\cdot)$ is the sigmoid function to ensure that $\beta \in (0, 1)$. When the critical dimensions dominate (i.e., $\bar{g}_{\text{critical}} \gg \bar{g}_{\text{redundant}}$), β approaches 1, prioritizing perturbations on critical dimensions. Conversely, when redundant dimensions have comparable or larger gradient magnitudes, β decreases, enabling broader exploration of state dimensions or mitigating the risk of overfitting to specific state features that may lose relevance as the adversarial policy evolves.

D. On-Policy Training Scheme for AGMR

To effectively train the AGMR adversarial attack algorithm, we adopt an on-policy RL framework combined with Generalized Advantage Estimation (GAE) [33]. On-policy framework exhibits heightened sensitivity to minor variations in the victim agent's behavioral patterns. In contrast to off-policy methods, which may suffer from distribution shifts due to the utilization of outdated experiences, the on-policy framework ensures that policy updates are consistently derived from the most recent attack-victim interactions. This approach enables enhanced real-time adaptation of perturbation strategies while maintaining update stability through the exclusive use of on-distribution samples from current policy trajectories. Specifically, the advantage function $\widehat{A}_t$, which measures how much better an action is compared to the expected return under the current policy, is computed at time step t as:

$$\widehat{A}_t = \sum_{k=0}^{T-t-1} (\gamma\lambda)^k \delta_{t+k}, \tag{11}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$. We set $\lambda = 0.95$ to balance bias and variance in GAE. By incorporating rewards and value function estimates, GAE provides a stable and effective approximation of the advantage function, which is crucial for policy updates. This setup is particularly suitable for tasks with long-term dependencies, such as adversarial attacks in robotic manipulation, as it captures the cumulative impact of future rewards without the need for additional weighting or truncation of temporal difference errors. Additionally, AGMR employs a parameterized value function to approximate the expected return for a given state under the current policy. The value function V is updated by minimizing the following mean squared error loss:

$$\mathcal{L}(V) = \mathbb{E}_{s_i, \hat{R}_i \sim \mathcal{D}}\left[\hat{R}_i - V(s_i)\right]^2, \tag{12}$$

where $\widehat{R}_i = \sum_{k=0}^{T-i-1} \gamma^k r_{i+k} + \gamma^{T-i} V(s_T)$ denotes the expected return, and $\mathcal{D}$ denotes the replay buffer of on-policy trajectories sampled from the victim policy under attack $\mu(\cdot \mid s + \eta)$. The mask function $M(s)$, similar to the victim policy $\mu(\cdot \mid s)$ and the value function $V(s)$, is parameterized by a neural network θ^M. The adversarial policy is optimized by minimizing the objective function represented by Eq. 5.

Algorithm 1 AGMR training algorithm

Input: victim agent's policy $\mu(\cdot \mid s)$, batch size N, discount factor γ

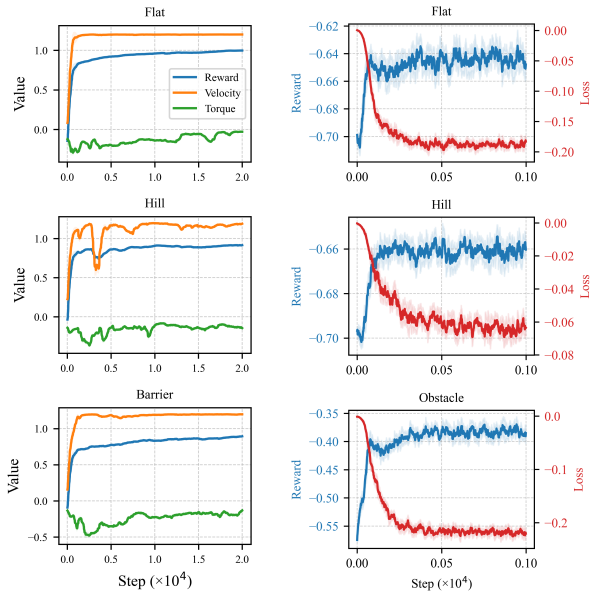
1: **Random initialization:** mask function $M(\cdot)$ with θ^M and value function $V(\cdot)$ with θ^V, and replay buffer $\mathcal{D}$
2: **for** each episode **do**
3: Initialize state s_0 and $T \leftarrow 0$
4: **for** time step t **do**
5: $\eta_t \sim \nu(\cdot \mid s_t, \mu)$
6: $a_t \sim \mu(\cdot \mid s_t + \eta_t)$
7: Execute a_t, compute reward $r_t = R(s_t, a_t)$, and store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$.
8: **if** s_{t+1} is terminal **then**
9: $T \leftarrow t + 1$
10: Calculate and store return $\widehat{R}_t$ in $\mathcal{D}$: $\widehat{R}_t = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k} + \gamma^{T-t} V(s_T)$
11: Calculate and store advantage $\widehat{A}_t$ in $\mathcal{D}$: $\widehat{A}_t = \widehat{R}_t - V(s_t)$
12: **end if**
13: **end for**
14: **for** each epoch **do**
15: Sample a batch of $\left(s_i, a_i, r_i, s_{i+1}, \widehat{R}_i, \widehat{A}_i\right)$ from $\mathcal{D}$
16: Update the value function $V(\cdot)$ by minimizing the loss: $\mathcal{L}(V) = \frac{1}{N}\sum_{i=0}^{N}\left[\hat{R}_i - V(s_i)\right]^2$
17: Update the mask function $M(\cdot)$ by minimizing the objective: $J(\nu) = \frac{1}{N}\sum_{i=0}^{N}\hat{A}_i$
18: **end for**
19: **end for**

The AGMR training algorithm is detailed in Algorithm 1. The training process begins by initializing the mask function $M(\cdot)$ with network parameters θ^M, the value function $V(\cdot)$ with network parameters θ^V and a replay buffer $\mathcal{D}$ to store transitions (line 1). For each episode, the initial state s_0 is set, and the episode length counter T is initialized to zero (line 3). During each time step of the episode, the perturbed state is generated based on the current state s_t and victim policy μ, after which the victim agent selects an action a_t according to the perturbed state (lines 5-6). The selected action a_t is executed in the environment, yielding a reward r_t and transitioning to the next state s_{t+1}. The transition tuple (s_t, a_t, r_t, s_{t+1}) is stored in the replay buffer

Fig. 2: Visualization of the robotic agent's locomotion task across three distinct terrains: flat (left), hill (middle), and obstacles (right).

TABLE I: Hyperparameters

Victim Agent Hyperparameters	
Victim Policy Network	2×128 FC layers
Victim Value Network	2×128 FC layers
Clipping parameter	0.2
Discount factor	0.998
Initial learning rate	5×10^{-4}
Adversarial Agent Hyperparameters	
Adversarial Policy Network	3×64 FC layers
Adversarial Value Network	3×64 FC layers
Discount factor	0.998
Learning rate	3×10^{-4}



(a) (b)

Fig. 3: Training trajectories of (a) victim agent and (b) AGMR adversarial agent.

$\mathcal{D}$ for subsequent training (line 7). If the environment reaches a terminal state s_{t+1}, the episode length T is updated. Returns $\widehat{R}_t$ and advantages $\widehat{A}_t$ for all time steps t in the episode are computed and stored in the replay buffer $\mathcal{D}$ (lines 9–11). Training occurs after the collection of trajectories. For each training epoch, a batch of N transitions, returns, and advantages is sampled from the replay buffer $\mathcal{D}$ (line 15). The value function V is updated by minimizing the mean squared error between the predicted value and the stored returns $\widehat{R}_i$ (line 16). The mask function $M(\cdot)$, defined in the adversarial policy in Eq. 6, is optimized by minimizing the objective $J(\nu)$ (line 17).

IV. EVALUATION

A. Experimental Setup

1) Environment: We evaluate quadrupedal locomotion control and adversarial robustness through comprehensive experiments on the RaiSim platform [34], a state-of-the-art physics engine renowned for its high-fidelity robotics simulations. As illustrated in Figure 2, we employ the ANYmal quadruped robot as our testbed across three tasks:

- **Flat**: The robot moves on a flat and even surface to evaluate its basic locomotion capabilities.
- **Hill**: The robot moves across uneven terrain with varying heights and slopes, designed to test its ability to maintain stability and adapt to unpredictable ground variations.
- **Obstacle**: The robot moves in a grid-like obstacle field, where the terrain consists of regularly spaced obstacles that challenge the robot's precision and robustness in foot placement.

The state space (34 dimensions) includes body height, body orientation (3), joint angles (12), body linear velocity (3), body angular velocity (3), and joint velocities (12). The action space (12 dimensions) controls the hip, thigh, and calf joints of the front right, front left, rear right, and rear left legs (3 dimensions each). Each episode lasts up to 4 seconds (400 control steps). If the robot falls, defined as any non-foot part touching the ground, the episode ends early. Experiments are conducted on a server with four NVIDIA GeForce RTX 3090 GPUs (24GB each).

2) Training: The victim agent is trained using Proximal Policy Optimization (PPO) [35], an on-policy RL algorithm that combines trust region optimization with clipped surrogate objectives. It employs an actor-critic architecture consisting of a policy network and a value network. The network architecture and hyperparameters are summarized in Table I. The victim agent's reward function balances energy efficiency and performance through torque penalties and forward velocity incentives:

$$R_{\text{vic}}(\tau, v_x) = \xi \sum_{i=1}^{12} \tau_i^2 + \kappa \min(v_x, 4.0), \tag{13}$$

where τ_i represents the torque of the i-th joint, v_x is the forward velocity, $\xi = -4 \times 10^{-5}$ is the energy efficiency coefficient penalizing the sum of squared torques, and $\kappa = 0.3$ is the forward velocity coefficient encouraging locomotion with an upper bound of 4.0 m/s. Figure 3a shows the training trajectories.

The adversarial agent, trained using the AGMR framework, consists of an adversarial policy and value function. The masking function within the adversarial policy is parameterized by neural networks, as detailed in Table I. The adversarial reward is defined as the negative of the victim agent's reward to minimize artificial effort introduced by

reward shaping:

$$R_{\text{adv}} = -R_{\text{vic}}. \tag{14}$$

The adversarial agent's training spans 2×10^3 steps, with the trajectory shown in Figure 3b.

3) Baselines: To evaluate the performance of AGMR, we compare it with a wide range of existing adversarial attack methods. For a fair comparison, we set the perturbation budget for all methods to $\epsilon = 0.125$. The methods compared include:

- **Random Attack**: A baseline applying uniform random noise as perturbations, serving as a naive benchmark.
- **FGSM** [18]: A single-step gradient-based attack designed to maximize loss with minimal computation.
- **DI^2-FGSM** [23]: An FGSM extension introducing input transformations (e.g., resizing, padding) to enhance transferability.
- **MI-FGSM** [22]: An iterative FGSM variant leveraging momentum to stabilize updates and improve transferability.
- **NI-FGSM** [25]: Builds on MI-FGSM by integrating Nesterov accelerated gradients for refined updates.
- **R+FGSM** [36]: An FGSM extension adding random perturbations before gradient-based updates to mitigate local gradient sensitivity.
- **PGD** [12]: An iterative FGSM extension with projection onto the allowed perturbation space after each step.
- **TPGD** [37]: A PGD variant replacing cross-entropy loss with KL divergence to improve attack success against robust models.
- **EOT-PGD** [38]: A PGD variant applying random transformations or model variations during iterations to ensure robustness across distributions.

B. Comparative Experiments

Tables II present the comparative experimental results of baseline methods across three tasks. Performance is evaluated using three metrics: Reward (R), Forward Velocity (V), and Fall Count (F). For each configuration, we conduct 10 independent episodes and report the mean ± standard deviation. The best-performing method is boldfaced, while suboptimal methods are underlined. The results demonstrate that the proposed method AGMR, consistently achieves superior performance. For the Flat task, AGMR achieves the lowest Reward (0.623 ± 0.328) and Velocity (2.156 ± 1.482), while maintaining the highest Fall Count (F = 4), indicating its significant ability to disrupt task performance. Similarly, in the Hill task, AGMR exhibits the lowest Reward (0.698 ± 0.181) and Velocity (2.401 ± 1.178), with a competitive Fall Count (F = 3), outperforming other methods. In the Obstacle task, AGMR further highlights its effectiveness, achieving the lowest Reward (0.486 ± 0.572), the lowest Velocity (2.311 ± 1.566), and the highest Fall Count (F = 3), surpassing other method in destabilizing the locomotion.

C. Behavior Analysis

Figure 4a visualizes motion sequence over time during the flat task under No Attack, the strongest baseline DI²-FGSM (ϵ = 0.125), and the proposed AGMR (ϵ = 0.125). Under the No Attack condition, the robot exhibits smooth and stable movement. DI²-FGSM introduces mild instability, while AGMR causes significant disruptions, with the robot's movements becoming increasingly unbalanced and results in a fall.

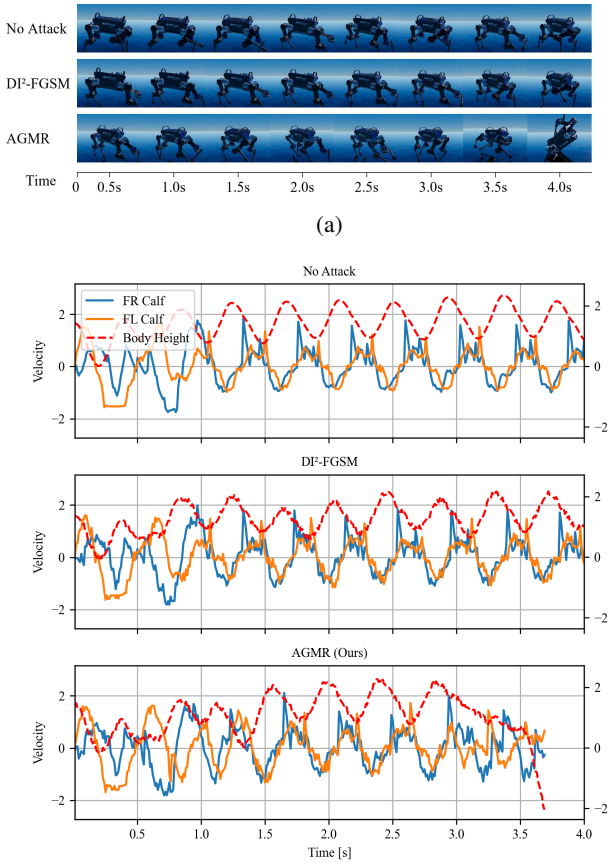


Fig. 4: Locomotion performance under No Attack, DI^2-FGSM ($\epsilon = 0.125$), and AGMR ($\epsilon = 0.125$). (a) Visualized motion sequences. (b) Time series of body height and calf trajectories. AGMR induces greater instability under the same perturbation budget.

Figure 4b illustrates the corresponding gait parameters over time, including the body height and the velocities of the front-left (FL) and front-right (FR) calves. Under the No Attack condition, the trajectories maintain smooth, periodic characteristics, reflecting stable locomotion. When subjected to DI²-FGSM, minor perturbations are observed, yet the gait retains its fundamental periodicity and stability. In contrast, under AGMR with an equivalent perturbation budget, significant disruptions occur: both body height and calf joint velocity trajectories exhibit pronounced oscillations, culminating in the robot's collapse. These results intuitively demonstrate that AGMR is highly effective at disrupting the robot's gait, showing its ability to induce critical locomotion failures.

Method	Flat			Hill			Obstacle		
	R ↓	V ↓	F ↑	R ↓	V ↓	F ↑	R ↓	V ↓	F ↑
NoAttack	0.975 ± 0.001	3.700 ± 0.004	0	0.920 ± 0.003	3.669 ± 0.005	0	0.881 ± 0.011	3.578 ± 0.037	0
Random	0.967 ± 0.003	3.693 ± 0.004	0	0.903 ± 0.031	3.619 ± 0.120	1	0.872 ± 0.011	3.569 ± 0.033	0
FGSM	0.932 ± 0.008	3.649 ± 0.020	0	0.811 ± 0.184	3.108 ± 1.082	2	0.820 ± 0.021	3.477 ± 0.052	0
DI^2-FGSM	0.777 ± 0.197	2.852 ± 1.142	**4**	0.769 ± 0.207	2.835 ± 0.844	2	0.605 ± 0.421	2.691 ± 1.255	2
MI-FGSM	0.928 ± 0.017	3.653 ± 0.034	0	0.782 ± 0.117	2.990 ± 1.159	1	0.772 ± 0.126	3.056 ± 0.937	2
NI-FGSM	0.838 ± 0.240	3.302 ± 0.991	1	0.784 ± 0.145	2.748 ± 0.997	**3**	0.691 ± 0.424	3.153 ± 1.048	1
R+FGSM	0.833 ± 0.220	3.258 ± 0.987	2	0.815 ± 0.053	2.991 ± 0.531	2	0.811 ± 0.017	3.461 ± 0.052	0
PGD	0.831 ± 0.166	3.090 ± 1.004	3	0.827 ± 0.090	3.159 ± 0.785	2	0.744 ± 0.177	2.802 ± 1.136	1
TPGD	0.801 ± 0.216	2.933 ± 1.026	2	0.858 ± 0.045	3.417 ± 0.360	1	0.590 ± 0.468	2.812 ± 1.368	2
EOTPGD	0.779 ± 0.274	3.007 ± 1.254	**4**	0.817 ± 0.120	3.149 ± 0.818	2	0.588 ± 0.458	2.780 ± 1.338	2
AGMR (Ours)	**0.623 ± 0.328**	**2.156 ± 1.482**	**4**	**0.698 ± 0.181**	**2.401 ± 1.178**	**3**	**0.486 ± 0.572**	**2.311 ± 1.566**	**3**

TABLE II: Attack performance comparison across tasks, with metrics Reward (R) and Forward Velocity (V) and Fall Count (F). The ↑ indicates higher value is better, while the ↓ indicates lower value is better.

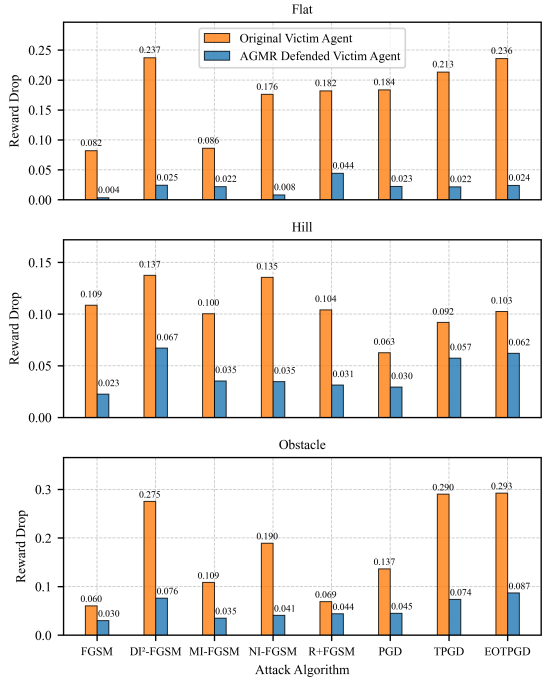


Fig. 5: Performance comparison between original and AGMR defended agents under different adversarial attacks.

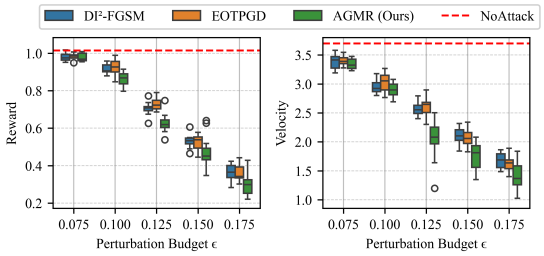


Fig. 6: Performance comparison of different attack methods under varying perturbation budgets ϵ.

D. Post-Defense Robustness of AGMR

Figure 5 compares the performance of the original victim agent and the AGMR-defended victim agent under various adversarial attacks. We evaluate the effectiveness of AGMR in enhancing model robustness by exposing the victim agent to AGMR adversarial attacks ($\epsilon = 0.125$) and training it for 2×10^2 steps at a learning rate of 3×10^{-4}, while keeping all other hyperparameters constant. The reward drop is used as the evaluation metric, with lower values indicating better robustness. The results show that the AGMR-defended victim agent demonstrates consistent robustness across diverse attack scenarios and environments, achieving substantial improvements over the original victim agent. In the Flat environment, the AGMR defended victim agent demonstrates strong resilience, with minimal reward drops across all attack methods, consistently below 0.05. In contrast, the original victim agent experiences significant reward degradation, with drops exceeding 0.2 under DI^2-FGSM, TPGD, and EOTPGD attack. A similar trend is observed in the Hill environment, where the AGMR-defended victim agent maintains reward drops below 0.05 for most attacks, while the original victim agent shows vulnerability, particularly under DI^2-FGSM and NI-FGSM attack, with reward drops significantly surpassing 0.1. In the more challenging Obstacle environment, the original victim agent experiences severe degradation, with reward drops approaching 0.3 under TPGD and EOTPGD attacks. In contrast, the AGMR-defended victim agent achieves significantly lower drops, consistently under 0.1, underscoring its robustness even in complex scenarios.

E. Ablation Study

Figure 6 shows the impact of perturbation budget ϵ on the attack performance. The perturbation budget ϵ serves as a crucial hyperparameter that controls the magnitude of adversarial perturbations. A larger ϵ allows for more substantial modifications to the input observations, potentially leading to more effective attacks, while a smaller ϵ ensures better imperceptibility. We compare AGMR with two suboptimal methods, DI^2-FGSM and EOTPGD, under varying perturbation budgets. The results show AGMR consistently outperforms the baselines across all budgets, with a particularly notable advantage under moderate and high budgets. This improvement is attributed to AGMR's ability to adaptively allocate perturbation budgets across state

dimensions based on their relative importance. By leveraging a gradient-masked reinforcement mechanism, AGMR identifies and exploits critical state dimensions to generate targeted, efficient perturbations that maximize impact within the given budget. In contrast, DI^2-FGSM and EOTPGD apply uniform perturbations without accounting for the varying importance of state dimensions, resulting in less effective attacks.

V. CONCLUSION

We propose Adaptive Gradient-Masked Reinforcement (AGMR), a white-box attack method combining DRL with gradient-based soft masking to identify critical state dimensions and optimize adversarial policies. By targeting impactful features, AGMR efficiently disrupts victim agents while balancing exploration and exploitation. Experimental results demonstrate that AGMR not only outperforms state-of-the-art adversarial attack methods in degrading performance of the victim but also improves the robustness of victim agents through adversarial training.

REFERENCES

[1] Z. Lin, L. Wang, J. Ding, B. Tan, and S. Jin, "Channel Power Gain Estimation for Terahertz Vehicle-to-Infrastructure Networks," *IEEE Commun. Lett.*, vol. 27, no. 1, pp. 155–159, 2022.

[2] Z. Fang, Z. Lin, S. Hu, H. Cao, Y. Deng, X. Chen, and Y. Fang, "IC3M: In-Car Multimodal Multi-Object Monitoring for Abnormal Status of Both Driver and Passengers," *arXiv preprint arXiv:2410.02592*, 2024.

[3] Z. Lin, G. Zhu, Y. Deng, X. Chen, Y. Gao, K. Huang, and Y. Fang, "Efficient Parallel Split Learning over Resource-Constrained Wireless Edge Networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 10, pp. 9224–9239, 2024.

[4] D. Han, B. Mulyana, V. Stankovic, and S. Cheng, "A survey on deep reinforcement learning algorithms for robotic manipulation," *Sensors*, vol. 23, no. 7, p. 3762, 2023.

[5] X. Lin, Y. Wang, J. Olkin, and D. Held, "Softgym: Benchmarking deep reinforcement learning for deformable object manipulation," in *Conf. Robot Learn.*, 2021, pp. 432–448.

[6] T. Duan, Z. Zhang, Z. Lin, Y. Gao, L. Xiong, Y. Cui, H. Liang, X. Chen, H. Cui, and D. Huang, "Rethinking adversarial attacks in reinforcement learning from policy distribution perspective," *arXiv preprint arXiv:2501.03562*, 2025.

[7] L. Shi and Y. Chi, "Distributionally robust model-based offline reinforcement learning with near-optimal sample complexity," *J. Mach. Learn. Res.*, vol. 25, no. 200, pp. 1–91, 2024.

[8] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Comput. Intell. Mag.*, vol. 10, no. 4, pp. 12–25, 2015.

[9] L. Schott, J. Delas, H. Hajri, E. Gherbi, R. Yaich, N. Boulahia-Cuppens, F. Cuppens, and S. Lamprier, "Robust deep reinforcement learning through adversarial attacks and training: A survey," *arXiv preprint arXiv:2403.00420*, 2024.

[10] D. Huang, Q. Bu, Y. Qing, H. Pi, S. Wang, and H. Cui, "Two heads are better than one: Robust learning meets multi-branch models," *arXiv preprint arXiv:2208.08083*, 2022.

[11] Q. Bu, D. Huang, and H. Cui, "Towards building more robust models with frequency bias," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, October 2023, pp. 4402–4411.

[12] A. Madry, "Towards Deep Learning Models Resistant to Adversarial Attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[13] R. Duan, Y. Chen, D. Niu, Y. Yang, A. K. Qin, and Y. He, "Advdrop: Adversarial attack to dnns by dropping information," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 7506–7515.

[14] J. Chen, H. Chen, K. Chen, Y. Zhang, Z. Zou, and Z. Shi, "Diffusion models for imperceptible and transferable adversarial attack," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2024.

[15] Z. Chen, B. Li, S. Wu, K. Jiang, S. Ding, and W. Zhang, "Content-based unrestricted adversarial attack," *Adv. Neural Inf. Process. Syst.*, vol. 36, 2024.

[16] T. Oikarinen, W. Zhang, A. Megretski, L. Daniel, and T.-W. Weng, "Robust Deep Reinforcement Learning through Adversarial Loss," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 26 156–26 167, 2021.

[17] N. Akhtar, A. Mian, N. Kardan, and M. Shah, "Advances in Adversarial Attacks and Defenses in Computer Vision: A Survey," *IEEE Access*, vol. 9, pp. 155 161–155 196, 2021.

[18] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *arXiv preprint arXiv:1412.6572*, 2014.

[19] E. Wong, L. Rice, and J. Z. Kolter, "Fast Is Better Than Free: Revisiting Adversarial Training," *arXiv preprint arXiv:2001.03994*, 2020.

[20] L. Schwinn, R. Raab, A. Nguyen, D. Zanca, and B. Eskofier, "Exploring Misclassifications of Robust Neural Networks To Enhance Adversarial Attacks," *Appl. Intell.*, vol. 53, no. 17, pp. 19 843–19 859, 2023.

[21] X. Wang, X. He, J. Wang, and K. He, "Admix: Enhancing the Transferability of Adversarial Attacks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 16 158–16 167.

[22] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting Adversarial Attacks with Momentum," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9185–9193.

[23] 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 *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2730–2739.

[24] Y. Dong, T. Pang, H. Su, and J. Zhu, "Evading Defenses to Transferable Adversarial Examples by Translation-invariant Attacks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4312–4321.

[25] 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.

[26] I. Ilahi, M. Usama, J. Qadir, M. U. Janjua, A. Al-Fuqaha, D. T. Hoang, and D. Niyato, "Challenges and Countermeasures for Adversarial Attacks on Deep Reinforcement Learning," *IEEE Trans. Artif. Intell.*, vol. 3, no. 2, pp. 90–109, 2021.

[27] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial Attacks on Neural Network Policies," *arXiv preprint arXiv:1702.02284*, 2017.

[28] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust Deep Reinforcement Learning with Adversarial Attacks," *arXiv preprint arXiv:1712.03632*, 2017.

[29] 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.

[30] 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 *Int. Conf. Learn. Represent.*, 2019.

[31] 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," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 21 024–21 037, 2020.

[32] R. S. Sutton, "Reinforcement learning: An introduction," *Bradford Book*, 2018.

[33] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional Continuous Control Using Generalized Advantage Estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[34] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 895–902, 2018. [Online]. Available: www.raisim.com

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[36] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble Adversarial Training: Attacks and Defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[37] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically Principled Trade-off Between Robustness and Accuracy," in *Int. Conf. Mach. Learn.*, 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.