Challenges and Countermeasures for Adversarial Attacks on Deep Reinforcement Learning

Inaam Ilahi, Muhammad Usama, Junaid Qadir, *Senior Member, IEEE*, Muhammad Umar Janjua, Ala Al-Fuqaha, *Senior Member, IEEE*, Dinh Thai Hoang, *Member, IEEE*, and Dusit Niyato, *Fellow, IEEE*

***Abstract*—Deep reinforcement learning (DRL) has numerous applications in the real world, thanks to its ability to achieve high performance in a range of environments with little manual oversight. Despite its great advantages, DRL is susceptible to adversarial attacks, which precludes its use in real-life critical systems and applications (e.g., smart grids, traffic controls, and autonomous vehicles) unless its vulnerabilities are addressed and mitigated. To address this problem, we provide a comprehensive survey that discusses emerging attacks on DRL-based systems and the potential countermeasures to defend against these attacks. We first review the fundamental background on DRL and present emerging adversarial attacks on machine learning techniques. We then investigate the vulnerabilities that an adversary can exploit to attack DRL along with state-of-the-art countermeasures to prevent such attacks. Finally, we highlight open issues and research challenges for developing solutions to deal with attacks on DRL-based intelligent systems.**

***Impact Statement*—Deep reinforcement learning (DRL) has numerous real-life applications ranging from autonomous driving to healthcare. It has demonstrated superhuman performance in playing complex games like Go. However, in recent years, many researchers have identified various vulnerabilities of DRL. Keeping this critical aspect in mind, in this article, we present a comprehensive survey of different attacks on DRL and various countermeasures that can be used for robustifying DRL. To the best of our knowledge, this survey is the first attempt at classifying the attacks based on the different components of the DRL pipeline. This article will provide a roadmap for the researchers and practitioners to develop robust DRL systems.**

***Index Terms*—Adversarial machine learning, cyber-security, deep reinforcement learning (DRL), machine learning (ML), robust machine learning.**

Manuscript received May 6, 2021; revised July 11, 2021 and August 29, 2021; accepted September 4, 2021. Date of publication September 13, 2021; date of current version March 24, 2022. This work was supported by the Qatar National Research Fund (a member of Qatar Foundation) through the National Priorities Research Program under Grant 13S-0206-200273. This article was recommended for publication by Associate Editor Prof. Pablo Estevez. *(Inaam Ilahi and Muhammad Usama contributed equally to this work.) (Corresponding author: Inaam Ilahi.)*

Inaam Ilahi, Muhammad Usama, and Muhammad Umar Janjua are with the Information Technology University, Lahore 54600, Pakistan (e-mail: mscs18037@itu.edu.pk; muhammad.usama@itu.edu.pk; umar.janjua@itu.edu.pk).

Junaid Qadir is with the Information Technology University, Lahore 54600, Pakistan, and also with Qatar University, Doha 2713, Qatar (e-mail: junaid.qadir@itu.edu.pk).

Ala Al-Fuqaha is with the Hamad Bin Khalifa University, Doha 34110, Qatar (e-mail: aalfuqaha@hbku.edu.qa).

Dinh Thai Hoang is with the University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: hoang.dinh@uts.edu.au).

Dusit Niyato is with the Nanyang Technological University, Singapore, 639798 (e-mail: dniyato@ntu.edu.sg).

Digital Object Identifier 10.1109/TAI.2021.3111139

Nomenclature

A3C	Asynchronous advantage actor–critic.
A2C	Advantage actor–critic.
ASA	Adversarial-strategic agent.
AGE	Adversarially guided exploration.
AI	Artificial intelligence.
ARPL	Adversarially robust policy learning.
ATN	Adversarial transformation networks.
CARRL	Certified adversarial robustness for RL.
C&W	Carlini and Wagner.
CDG	Common dominant adversarial example generation.
c-MARL	Cooperative multiagent reinforcement learning.
DRL	Deep reinforcement learning.
DDPG	Deep deterministic policy gradient.
DDQNs	Double deep Q-networks.
d-JSMA	Dynamic budget JSMA.
DL	Deep learning.
DQN	Deep Q-networks.
FGSM	Fast gradient sign method.
FRARL	Falsification-based RARL.
GB	Gradient based.
GPS	Graded policy search.
I2A	Imagination augmented agents.
IRL	Inverse reinforcement learning.
it-FGSM	Iterative target-based FGSM method.
JSMA	Jacobian-based saliency map attack.
KL	Kullback–Leibler.
LAS	Look-ahead action space.
MAD	Maximal action difference.
MAS	Myopic action space.
MBMF-RL	Model-based priors for model-free reinforcement learning.
MDP	Markov decision process.
ME-TRPO	Model ensemble trust region policy optimization.
ML	Machine learning.
MLAH	Meta-learned advantage hierarchy.
MPC	Model-predictive control.
MuJoCo	Multijoint dynamics with contact.
MVE	Model-based value expansion.
NAF	Normalized advantage function.
NR-MDP	Noisy action robust MDP.
PCA	Principal component analysis.
PEPG	Parameter exploring policy gradients.
POMDP	Partially observable Markov decision process.

2691-4581 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

PPO	Proximal policy optimization.
PR-MDP	Probabilistic MDP.
RARARL	Risk-averse RARL.
RARL	Robust adversarial RL.
RBFQ	Radial-basis-function-based Q-learning.
RNN	Recurrent neural network.
SARSA	State–action–reward–state–action algorithm.
SA-MDP	State-adversarial MDP.
SDN	Software-defined networking.
SFD	Sampling-based finite difference.
SGD	Stochastic gradient descent.
SPG	Stochastic policy gradient.
STEVE	Stochastic ensemble value expansion.
TMDPs	Threatened Markov decision processes.
TRPO	Trust region policy optimization.
WMA	Weighted majority algorithm.

I. Introduction

THE ultimate goal of research in AI is to develop artificial general intelligence (AGI) agents that can perform similar activities as humans in a more efficient manner. This long-standing challenge for developing such agents is no longer a pipe dream, thanks to rapid growth in computational AI and ML technologies. In the last decade, ML and especially DL have revolutionized fields such as computer vision, language processing, etc. ML is divided into three categories [1]: namely, *supervised, unsupervised,* and *reinforcement learning* (RL). In *supervised learning*, training data along with the corresponding labels are available for decision making. Supervised learning is by far the most well-studied branch of ML for problems with labeled data, which has a lot of applications in practice such as object recognition, speech recognition, spam detection, pattern recognition, and many more. In *unsupervised learning*, the target is to infer the underlying patterns and structures from unlabeled data. RL is defined as a learning process that focuses on finding the best strategies for agents based on the interactions with the surrounding environment. Unlike supervised and unsupervised learning processes, which need training data to learn, RL agents can learn in an online manner, based on observations obtained through real-time interactions with the environment.

RL utilizes a trial-and-error process to solve sequential decision-making problems in robotics, control, and many other real-world problems. RL algorithms also have some limitations to be utilized in practice mainly due to their slow learning process and inability to learn in complex environments. Recently, a new technique combining the advancement of DL, called DRL, has been introduced [2]. DRL has shown great results in many complex decision-making processes such as designated task completion in robotics [3], navigating driverless autonomous vehicles [4], [5], healthcare [6], financial trading [7], smart grid management [8], automated transportation management [9], wireless and data network management [10], and for playing games such as Pong [11], Go [12], etc. In 2017, DRL beat the human champions in the game of Go [13], and most recently, a team of five DRL agents has beaten the world champion human team in Dota2 matches [14]. This shows that DRL is promising and can address highly complex and time-sensitive decision-making problems in real time.

With the rapid adoption of DRL in critical real-world applications, the security of DRL has become a very important area of research [15], [16]. Recently, DRL has been vulnerable to *adversarial attacks*, where an imperceptible perturbation is added to the input to the DRL schemes with a predefined goal of causing a malfunction in the working of DRL [17]. Thus, it is crucial to understand the types and nature of these vulnerabilities and their potential mitigation procedures before deploying DRL-based real-life critical systems (e.g., smart grids and autonomous vehicles). Here, we want to note that the security of supervised and unsupervised ML is well studied in the literature [18], but the security of DRL has not yet received similar attention. In 2018, Behzadan and Munir [19] reviewed the security vulnerabilities and open challenges in DRL. Although it provides a decent initial review of the security concerns, it does not properly cover the security issues associated with four major components of the DRL pipeline (state, action, model, and reward) and related robustness mechanisms. Furthermore, the survey does not cover the recent studies. We aim to fulfill these requirements by providing a more comprehensive survey on attacks and defense techniques together with a discussion of the future research directions on DRL.

Contributions of This article: In this article, we build upon the existing literature available on security vulnerabilities of DRL and their countermeasures and provide a comprehensive and extensive review of the related work. The major contributions of this article are as follows.

1) We provide the DRL fundamentals along with a nonexhaustive taxonomy of advanced DRL algorithms.
2) We present a comprehensive survey of adversarial attacks on DRL and their potential countermeasures.
3) We discuss the available benchmarks and metrics for the robustness of DRL.
4) Finally, we highlight the open issues and research challenges in the robustness of DRL and introduce some potential research directions.

Organization of This article: The organization of this article is depicted in Fig. 1. An overview of the challenges faced by ML and DRL schemes has been provided in Section II. Section III presents a comprehensive review of adversarial ML attacks on the DRL pipeline. A detailed overview of countermeasures proposed in the literature to ensure robustness against adversarial attacks is presented in Section IV. Section V presents the available benchmarking tools and metrics along with open research problems in DRL. Section VI describes the open issues and research challenges in designing adversarial attacks and robustness mechanisms for DRL. Finally, we conclude this article in Section VII. For the convenience of the reader, a summary of the salient acronyms used in this article is presented in the Nomenclature.

II. Background

In this section, we discuss the fundamentals of the DRL process. Then, we provide a summary of the shortcomings of the ML and DRL techniques.

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

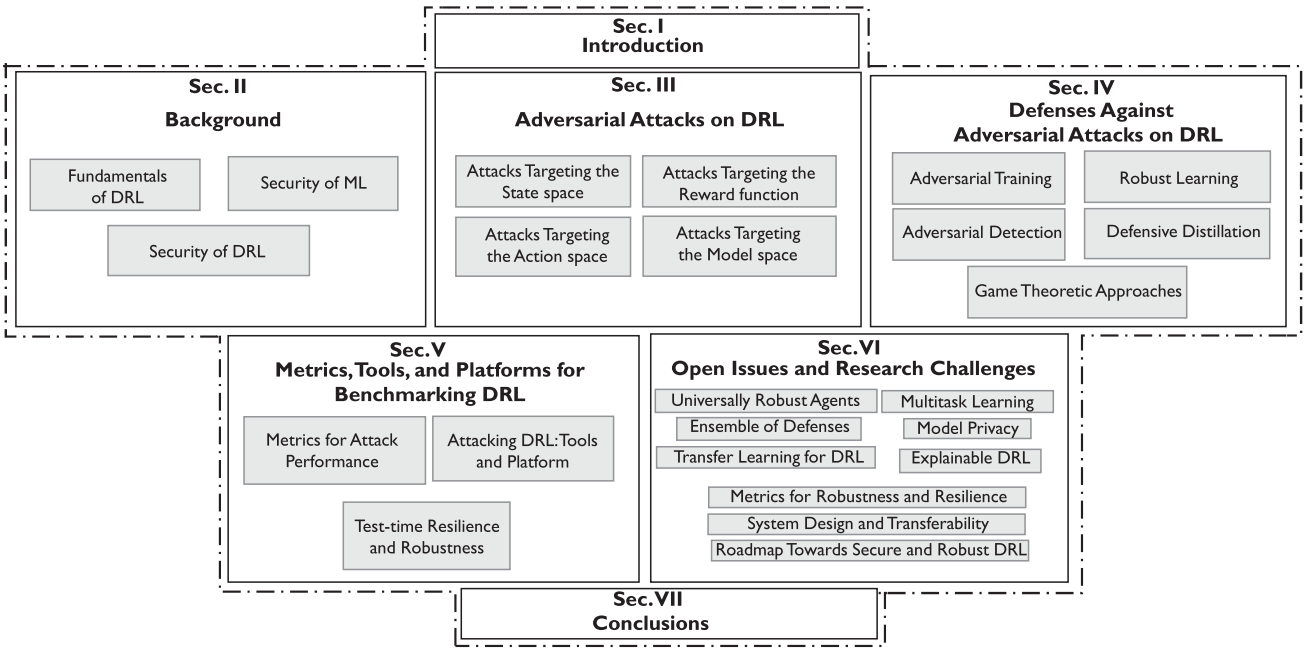

Fig. 1. Organization of this article.

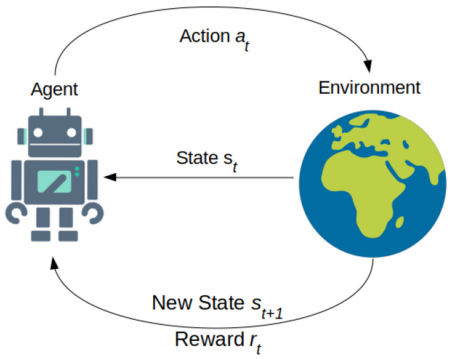

Fig. 2. Basic process of the MDP in RL.

A. Fundamentals of DRL

The important concepts used in DRL are described as follows.

1) *Markov decision process:* A generic RL problem is described as an MDP in terms of the state, action, reward, and dynamics of the system. In an MDP, at each time step, the agent observes the current state s_t and performs an action a_t based on its current policy π^*. After the action is executed, the agent observes its reward r_t and next state s_{t+1}. The objective of an MDP is to find the best actions, which maximize its long-term expected reward. Fig. 2 illustrates a typical MDP setup with an agent interacting with its surrounding environment.
2) *Environment* is a simulator or a real-world scenario in which the agent interacts and learns. At each time step, the agent (governed by the policy) interacts with the environment and in return receives a reward. Environment is divided into two categories, namely, *partially observable* and *fully observable*. In the case of a partially observable environment, the agent is only able to partially observe the environment. For these partially observable environments, POMDPs are used. In a fully observable environment, the agent can observe all the states, and we use MDPs for this. MDPs are a special case of POMDPs, where the observation function is identity.
3) *Action* is a stimulus used by the agent for interactions with the environment. The actions can be discrete or continuous based on the environment and DRL problem formulation.
4) *Reward* is an incentive, expressed by a numerical value, that the agent receives after making an action. The goal of an agent is to maximize the accumulated reward. To reduce the impact of the reward r which the agent might get in a later state due to taking a specific action a in the current state s_t, the notion of discounted rewards was introduced. It is usually denoted by γ and can take any value ranging from 0 to 1. Mathematically, the discounted reward R_t given as

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}) \tag{1}$$

where t denotes the time step, T is the final time step, r denotes the reward for the time step, and s_t denotes the current state.
5) *Value function* specifies the value of a state. Value is defined as the maximum expected discounted reward of a certain state. Mathematically, it is determined as

$$V_\pi(s) = \mathbf{E}_\pi[R_t | s = s_t] \tag{2}$$

where π is the policy, R_t is the discounted reward, and s_t is the current state.
6) *Q-function* specifies the Q-value of a state. Q-value is defined as the maximum expected discounted reward an

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

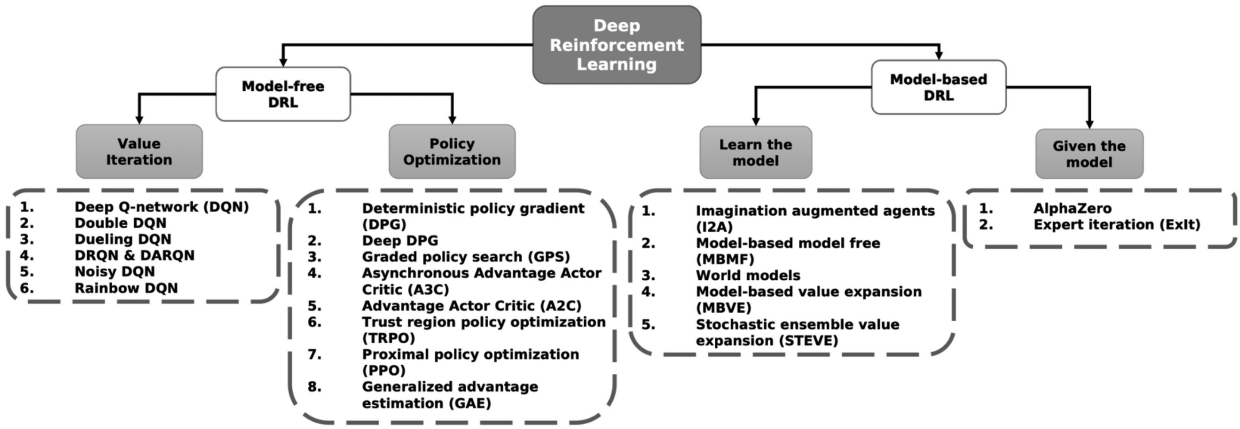

Fig. 3. Nonexhaustive taxonomy of major DRL schemes as proposed in [20].

agent may get by taking a specific action at a specific state. Mathematically, it can be calculated as follows:

$$Q_\pi(s) = \mathbf{E}_\pi[R_t|s = s_t, a = a_t] \tag{3}$$

where π is the policy, R_t is the discounted reward, s_t is the current state, and a_t is the current action.

7) *Advantage function* is the difference of the Q-value of a specific action at a state $Q(s, a)$ from the value of that state $V(s)$, i.e., $A(s, a) = Q(s, a) - V(s)$.
8) *Policy* defines how the agent will behave in the environment at a particular time. In other words, it is a mapping from the perceived states of the environment to the actions taken in those conditions. A policy is said to be optimal if it achieves the maximum possible reward at each state. Policies are further divided into two types: *deterministic policy* and *stochastic policy*. When actions taken by the agent are deterministic, the policy is termed as deterministic. On the other hand, when the actions are sampled from the conditional probability distribution of actions given the states, the policy is called to be stochastic.
9) *On-policy algorithms* enable an agent to learn and update its policy in an online manner through real-time interaction with the environment. Samples generated from the current policy are used to train the algorithm to estimate the policy in advance.
10) *Off-policy algorithms* use an online policy and a target policy. The target policy is used to estimate the action values, while the online policy is being learned. Hence, the agent can estimate the target policy without its complete knowledge.
11) *Model* mimics the behavior of the environment, hence allowing inferences to be made about the behavior of the environment. Based on the availability of the system models, the DRL schemes are divided into further two categories namely *model-based* and *model-free RL*.
12) *Exploration and exploitation*: *Exploration* is the process when the agent tries to explore the surrounding environment by taking different actions available at a given state. *Exploitation* occurs after exploration. The agent exploits the optimal actions to achieve the maximum cumulative reward. An ϵ-greedy policy is used to balance exploration and exploitation. The agent chooses a random action with a certain probability; otherwise, it takes the action followed by the policy. The probability of the random action being taken keeps decreasing with each time step. This change factor is usually denoted by λ.

A taxonomy of major DRL algorithms as proposed in [20] has been provided in Fig. 3. We refer the interested readers to [2] and [21] for further details on variations of DRL schemes.

B. Security of ML

Although the utilization of ML techniques has revolutionized many areas, including vision, language, speech, and control; it has also introduced new security challenges that are very threatening in designing and developing new dynamic intelligent systems. Security attacks in ML can be divided into two categories: training phase attacks and inference phase attacks. For the training phase attacks, the adversary tries to force the learning process to learn faulty model/policy by introducing small imperceptible perturbations to the input data. Inference phase attacks are performed by the adversary at the inference/test time of the ML pipeline to fool the model/policy in providing malfunctioned results/actions.

The malicious input generated by adding adversarial perturbations into the original input is known as an *adversarial example*. Adversarial examples are classified into four major categories based on the objective, knowledge, frequency, and specificity. Formally, an adversarial example x^* is created by adding a small imperceptible carefully crafted perturbation δ to the correctly classified example x. The perturbation δ is calculated by approximating the optimization problem iteratively until the crafted adversarial example gets classified by ML classifier $f(.)$ in targeted class t as follows:

$$x^* = x + \arg\min_{\delta_x} \|\delta\| : f(x + \delta) = t \tag{4}$$

where t is the targeted class. Fig. 4 shows a basic taxonomy of attacks on ML.

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

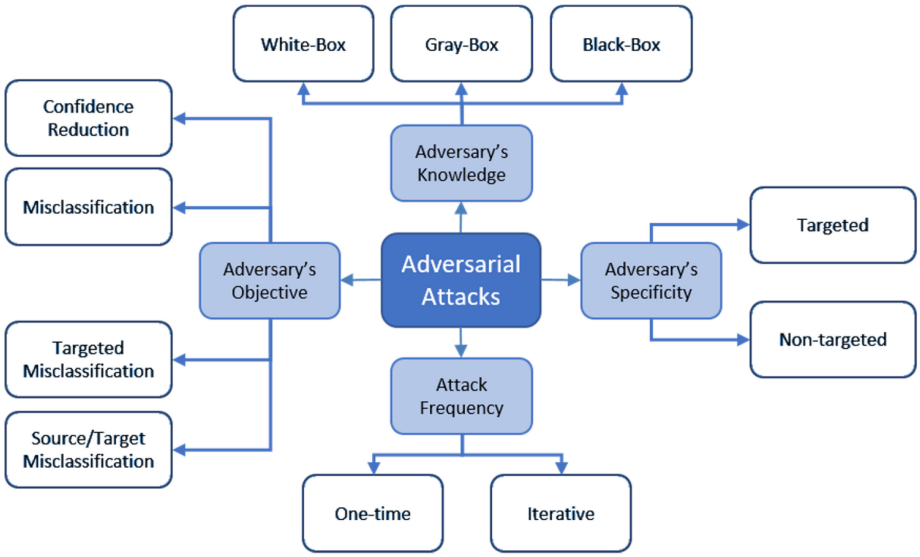

Fig. 4. Taxonomy of adversarial ML attacks classified according to the adversary's objective, knowledge, specificity, and attack frequency.

1) Attacks Based on Adversary's Knowledge: Depending on the adversary's knowledge about the targeted ML model, adversarial attacks are divided into further three categories: white-box attacks, gray-box attacks, and black-box attacks. In the case of *white-box attacks*, the adversary has perfect knowledge of the target ML/DL algorithm, i.e., the adversary knows the training and testing data, parameters of the model, etc. These attacks are used for the worst-case security malfunction analysis of an ML/DL system. In the case of *gray-box attacks*, the adversary is supposed to have limited knowledge (knowledge about feature representation and optimization algorithms only) about the targeted ML/DL model. The adversary designs a surrogate model on the limited knowledge available and uses *transferability* property [22] of the adversarial examples (where an adversarial example evading a classifier will evade other similar classifiers even if they are trained on another dataset) to evade the ML/DL-based system. The attacker may also have limited test access to the model, i.e., it may be able to ask the model the output on some inputs. In the case of *black-box attacks*, the adversary does not know the model or any of its attributes. The adversary can only query the systems for labels or confidence scores and develop an adversarial perturbation based on the feedback provided by the deployed ML/DL model.

2) Attacks Based on Adversary's Goals: Based on the adversary's objective, adversarial attacks are divided into four types:

1) *confidence reduction attacks* in which adversarial attacks are launched to compromise the confidence levels of the predictions of the deployed ML/DL-based system;
2) *misclassification attacks* in which adversarial attacks are launched for disturbing the classification boundary of any class to cause misclassification;
3) *targeted misclassification attacks* in which adversarial attacks are launched to misclassify only a targeted class; and
4) *source/target misclassification attacks* in which adversarial attacks are launched to force misclassification of a specific source class into a specifically targeted class.

3) Attacks Based on Adversary's Specificity: Based on specificity, adversarial examples can be classified into two types, i.e., *targeted* and *nontargeted*. These concepts are similar to the ones as in the case of the adversary's objective. In the case of targeted attacks, the attackers target specific classes in the output, while in the case of nontargeted attacks, the goal is to misclassify the maximum number of samples.

Although adversarial examples are transferable from one ML model to another, in many cases, the performance of the transferred examples is not enough. To further improve the performance of black-box attacks while reducing the number of queries needed for the attack, query-efficient black-box attacks are required. Different query-efficient black-box attack methods are available in the literature. Cheng *et al.* [23] use randomized gradient-free methods for the creation of adversarial examples and show their algorithm to require three to four times fewer queries to achieve the same performance as the state-of-the-art attacks. Chen *et al.* [24] uses the zeroth-order optimization technique for adversarial perturbation generation and shows their black-box attack to demonstrate the same performance as state-of-the-art white-box attacks. The queries required by their technique [24] are less than those required by that of [23]. Tu *et al.* [25] propose a more query-efficient attack. They propose autoencoder-based zeroth-order optimization for adversarial image generation in black-box attacks. They show a reduction of more than 93% in the mean query count while maintaining the same performance as the state-of-the-art attacks. More details on adversarial ML can be found in [26] and [27].

C. Security of DRL

The increasing use of DRL in practical applications has led to an investigation of the security risks it faces. However, the security challenges faced by DRL are different from those experienced by other ML algorithms. The major difference is that a DRL process is trained to solve sequential decision-making

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

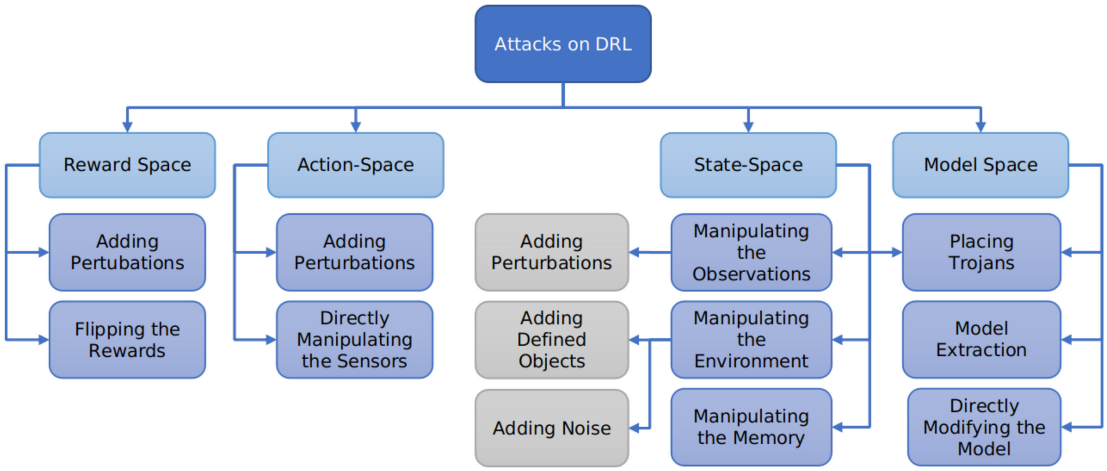

Fig. 5. Taxonomy of adversarial attacks on DRL classified according to the major parts of DRL.

problems in contrast to most other ML schemes that are trained to solve single-step prediction problems. The independence of the current actions from the previous ones increases the degrees of freedom of adversarial attacks raising new challenges that must be addressed. This makes the adversarial attacks more challenging to be recognized, as we cannot discriminate between the action intentionally taken by the agent and the action the adversary forces/lures the agent to take. Also, the training is done on a dataset from a fixed distribution in the case of ML, in contrast to the DRL, where the agent begins with a deterministic or stochastic policy and starts exploring for best actions.

Usually, RL problems are formulated as an MDP consisting of four parts (S, A, R, P), where S is the state space, A is the action space, R is the reward function, and P is the transition matrix between states. Hence, an adversary has more choices to attack. If the adversary targets the state space, imperceptible perturbations can be added to the environment directly by perturbing the sensors [28]. Similarly, an adversary can target any of the four major components of the MDP. Adversarial attacks on DRL are classified into *inference-time* and *training-time* attacks [29]. An adversary may compromise one or more than one dimension of confidentiality, integrity, and availability [19].

Based on the goal of the adversary, the adversarial attacks on DRL can be classified into *active* or *passive* [19]. For active attacks, the adversary desires to change the behavior of the agent, while for passive attacks, the adversary desires to infer details about the model, reward function, or other parts of DRL. An adversary can use these details to either create a copy of the model or use them to perform an attack on the model. The adversary may be limited by the part of the environment, where an adversary is only capable of making changes to a certain area of the environment. Adding a lot of perturbation in a single time instance may make the attack perceptible, which is not preferred by the adversary. Distinguishing the adversarial samples and behavior from the normal ones in the case of DRL is not as easy as in supervised learning because of the increased possible attack dimensions.

III. Adversarial Attacks on DRL

In this section, we discuss the adversarial attacks on DRL. We divide the attacks on DRL into four categories based on the functional components of the DRL process. A major portion of the attacks involve the addition of adversarial perturbations to the state space and a small portion of the proposed attacks involve perturbing the reward and action space. Fig. 5 shows a basic taxonomy of the adversarial attacks on DRL algorithms.

A. Attacks Perturbing the State Space

We divide this subsection based on the access of the adversary.

1) Manipulating the Observations: Since DNNs are vulnerable to adversarial attacks in supervised learning, we would expect DNNs trained via DRL to also be vulnerable. Indeed, Behzadan and Munir [17] show this vulnerability and verify the transferability of adversarial examples across different DQN models. They consider a man-in-the-middle adversary between the environment and the DRL agent, where the adversary perturbs the states from the environment and forwards these perturbed states to the DRL agent to take the desired action. To ensure the imperceptibility of the perturbation, the amplitude of the adversarial examples crafting algorithms (FGSM and JSMA [30]) is controlled. The attack procedure is divided into two phases: initialization and exploitation. The *initialization* phase includes the training of a DQN on adversarial reward function to generate an adversarial policy. Then, a replica of the target's DQN is created and initialized from random parameters. The *exploitation* phase includes generating adversarial inputs such that the target DQN can be made to follow actions governed by the adversarial policy. Furthermore, they propose an attack method to manipulate the policy of the DQN by exploiting the transferability of adversarial samples. They use a black-box setting and show a success rate of 70% when adversarial examples are transferred from one model to another. The cycle of the proposed policy induction attack is shown in Fig. 6. Huang *et al.* [31] use the attack proposed in [17] and show a significant drop in the performance of DQN, TRPO, and A3C methods in

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

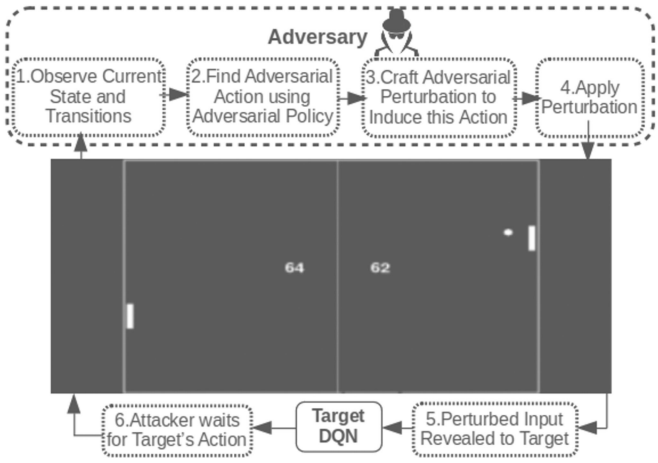

Fig. 6. Process of policy induction attack [17] performed on the game of Pong.

both white- and black-box settings. They show the DQN to be more susceptible to adversarial attacks than the TRPO and A3C.

Lin *et al.* [32] argue that the uniform attack schemes may not be practically feasible and are easy to detect. They consider a different approach and propose two adversarial attack techniques on DRL schemes: *strategically timed attack* and *enchanting attacks*. For the former, they propose to minimize the reward of the DRL schemes by using adversarial examples on a subset of time steps in an episode of the DRL operation. For the latter, they propose a novel method of luring the DRL agent to a predefined targeted state by using a generative model and a sophisticated planning algorithm. The generative model is used to predict the next state in the space, and the planning algorithm is used to generate actions required for luring the agent to the targeted state. Performance of the strategically timed attack and the enchanting attack is reported on DQN and A3C agents, playing Atari games where 70% of the success rate of the adversarial attacks is reported. They use the C&W attack [33] for generating adversarial inputs. It is also shown that perturbing only 25% of the inputs using the proposed method produces the same results as the previously proposed attacks based on the FGSM. The workflow of the enchanting attack is shown in Fig. 7.

Tretschk *et al.* [34] propose a similar approach to the enchanting attacks proposed by Lin *et al.* [32], where they use the adversarial transformer network (ATN) [35] to impose adversarial reward on the policy network of DRL. The ATN makes the agent maximize the adversarial reward through a sequence of adversarial inputs. Complete information regarding the agent and the target environment is required, hence making the attack white box. It is shown that given a large enough threshold for perturbation, the agent can be made to follow the adversarial policy at the test time.

Pattanaik *et al.* [36] prove that FGSM-based attacks on DRL [31] do not use an optimal cost function for crafting the adversarial inputs and propose a loss function that is guaranteed to maximize the probability of taking the worst possible action. They propose three types of GB adversarial attacks on DQN and DDPG techniques for reducing the expected reward by adding perturbations to the observations. The *first attack* is based on a naive approach of adding random noise to the DRL states to mislead the DRL agent in selecting a suboptimal action that decays the performance of the DRL scheme. The *second attack* is a GB attack, where a new cost function is introduced for creating adversarial actions, that outperforms the FGSM in determining the worst possible discrete action to limit the performance of DRL schemes. The *third attack* is an improved version of the second attack. Instead of using a simple GB approach for generating adversarial perturbation, the authors use SGD for adversarial action generation, which ultimately misleads the DRL agent to end up in a predefined adversarial state.

Kos and Song [37] discuss that the previous attacks require perturbing several states to be successful, which may not be practically feasible. They propose to use a value function to guide the adversarial perturbation injection, hence reducing the number of adversarial perturbations needed for introducing a malfunction in DRL policies. They propose three types of attack situations: 1) the addition of noise at a fixed frequency; 2) the addition of specially designed perturbed inputs after N samples; and 3) the recalculation of the perturbation after N samples and adding the previously calculated perturbation to the intermediate steps. The results show that their last approach performs as well as the one in which all states are perturbed. Furthermore, they use the generated samples for retraining the model and show that resilience can be improved against both FGSM and random adversarial perturbations.

A similar issue has been raised by Sun *et al.* [38]. Furthermore, they discuss that the previously proposed attacks are not general-purpose and have limitations, e.g., [32] cannot be used for continuous action spaces. They propose two sample efficient general-purpose attacks that can be used to attack any DRL algorithm while considering long-term damage impacts, namely, *critical point attack* and *antagonist attack*. The first one involves the building of a model by the adversary to predict future environmental states and the agent's actions. The damage of each possible attack strategy is then assessed, and the optimal one is chosen. The antagonist attack involves automatic learning of a domain-agnostic model by the adversary to discover the critical moments of attacking the agent in an episode. To be successful, the critical point technique only requires one (TORCS) or two (Atari pong and breakout) steps, and the antagonist technique needs fewer than five steps (four MuJoCo tasks).

Hussenot *et al.* [39] discuss that the previously proposed approaches are either practically infeasible or computationally extensive. They propose two types of adversarial attacks to take full control of the DRL agents' policy. The first one called *per-observation attack* includes the generation of a new adversarial perturbation for every observation of the agent and adding that perturbation to the environment. The second one called *universal mask attack* includes the addition of one universal perturbation, created at the start of the attack, to all the observations. These attacks are discussed in both targeted and nontargeted situations. It is also reported that the proposed attacks are more successful if the FGSM is used for generating the perturbations in untargeted

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

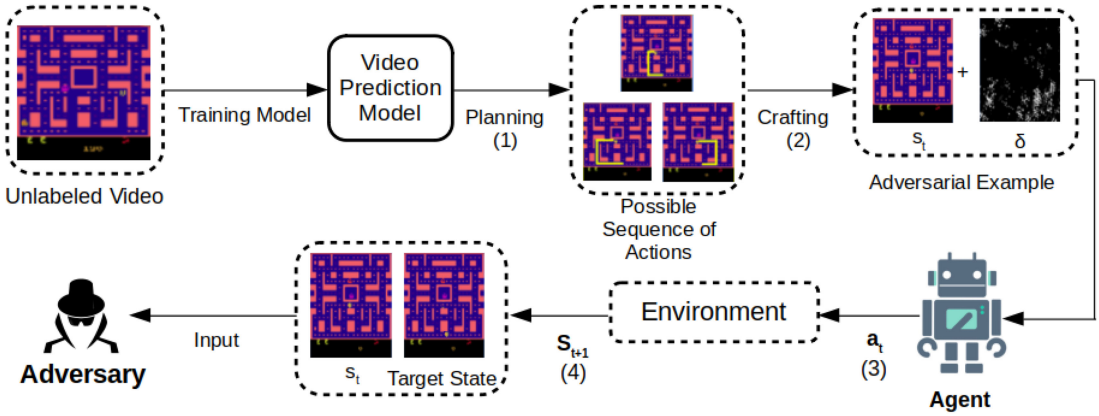

Fig. 7. Illustration of enchanting attack on Pacman is depicted, which is highlighting all four components: (1) action sequence planning, (2) crafting an adversarial example with a target action, (3) the agent takes an action, and (4) environment generates the next state s_{t+1}.

attack situations, whereas in the case of targeted attacks, the FGSM is not able to generate imperceptible adversarial samples.

Chan *et al.* [40] take a different approach from other articles and propose static reward impact maps, which can be used to quantify the influence on the reward of each feature in the state space. By the use of such a map, the adversary can choose to perturb only those features which have a large impact on the cumulative reward. The time complexity of the generation of these static maps is posed as a limitation.

c-MARL algorithms are gaining attention in a wide range of applications such as cellular base station control [41], traffic light control [42], and autonomous driving [43]. The target of the agent in c-MARL is to learn to take action cooperatively as a team to maximize a total team reward. Lin *et al.* [44] show the vulnerability of c-MARL agents to adversarial attacks by proposing a mechanism of adding perturbations to the state space. Difficulty to estimate team rewards, difficulty to measure the effect of misprediction of an agent on the team reward; nondifferentiability of models, and low-dimensionality of the feature space make attacking such environments challenging. They hypothesize that the cooperative aspects of c-MARL agents make these agents more vulnerable to adversarial attacks as compared to single-agent RL, as the failure of a single agent may cause the failure of the whole team. They extend the FGSM attack and propose two new approaches to decrease the team reward more effectively: *it-FGSM* and *d-JSMA attack*. These attacks involve training an adversarial policy network to search for a suboptimal action from which the adversarial examples are then introduced in the observations of one of the agents to force it to take the targeted action. They test their attack on the StartCraft II multiagent benchmark. They show that their attack can decrease the reward from 20 to 9.4 by attacking only a single agent out of the multiple possible agents when the perturbations are added with an average of 8.1 L1 norm. As a reaction to this drop in reward, the winning rate of the multiagent drops from 98.9% to 0%. Furthermore, they discuss the applicability of the proposed attack in real environments as an adversary can gain access to a single agent and use it to attack the whole system.

Despite the success of DRL, there is little research that studies the impact of adversarial attacks in DRL algorithms that do not use images as inputs. Wang *et al.* [45] propose techniques that can degrade the performance of a well-trained DRL-based energy management system of an electric vehicle causing them to either use much fuel or lead it into running out of battery before the end of the trip. They show their adversarial inputs to be imperceptible. For white-box settings, they use the FGSM as the adversary, while for black-box settings, attack transferability and the finite-difference method [46] are used. They test their attacks on a DQN trained for energy management of an electric vehicle and show the degradation of performance.

2) Manipulating the Environment: Adversarial attacks on the state space can also be carried out by adding perturbations in the environment of the agent. In turn, this causes the agent to consider the environment as the adversary desires. Chen *et al.* [47] propose *CDG method* for crafting adversarial examples with high confidence for the environment of DRL. The core idea of their attack is the addition of confusing obstacles to the original clean map for the case of pathfinding to confuse the robot by messing with its local information. For a perturbation to be successful, it should either stop the agent from reaching the destination or otherwise delay the agent. The proposed attack is tested on A3C and is shown to be successful at least 99.91% of the time.

Bai *et al.* [48] take a different approach than [47] and propose a method of finding adversarial examples for DQNs trained for automatic pathfinding. The proposed attack analyzes a trained DQN for the task and identifies the weaknesses present in the Q-value curves. Especially designed perturbations are added to these weaknesses in the environment to effectively refrain the agent from learning an optimal solution to the maze.

Xiao *et al.* [46] introduce online sequential attacks on the environment of the DRL agent by exploiting the temporal consistency of the states. This attack performs faster than the FGSM algorithm as no backpropagation is needed and is based on model querying. The authors provide two methods for model querying: *adaptive dimension SFD method* and *optimal frame selection method*. In addition to these sequential attacks, they also propose other attacks on the observations, action selection, and environment dynamics.

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

Gleave *et al.* [49] propose to introduce an adversarial agent in the same environment as the legitimate agent. The adversary is not able to manipulate the observations of the legitimate agent but can create natural observations that can act as adversarial inputs and make the agent follow the target policy. This leads to a zero-sum game between the adversarial agent and the legitimate agent. Both the adversarial and the victim agent are based on PPO. After showing the existence of such adversarial policies, they suggest that the learning of the deployed model must be frozen to save them from undesired behaviors enforced by adversaries. Such adversarial agents can also be used in making the models better by constantly attacking and retraining.

Although the DRL techniques work better in navigation tasks, they are more vulnerable to adversarial attacks than the classic methods. Yang *et al.* [50] introduce timing-based adaptive adversarial perturbations for learning-based systems in real-world scenarios. They introduce two attacks, namely, WMA (white-box setting) and ASA via a population-based training method based on PEPG (black-box setting). The first one is based on online learning, while the other is based on evolutionary learning. Through experiments, they show that out of the two proposed attacks, the WMA shows a better performance.

3) Manipulating the Training Data: The adversary can also choose to perturb the training data to indirectly having the agent follow a targeted policy. An adversary may create and hide some deficiencies in the policy to use them later for his benefit. Kiourti *et al.* [51] show this vulnerability of DRL models to Trojan attack with adversary having access to the training phase of the model. It is reported that by only modifying 0.025% of the training data, an adversary can induce such hidden behaviors in the policy that the models perform perfectly well until the Trojan is triggered. The proposed attack is shown to be resistant against current defense techniques for Trojans.

A similar approach has been proposed by Behzadan and Hsu [52] to secure DRL models from model extraction attacks but can be used for adversarial purposes. This involves the integration of a unique response to a specific sequence of states while keeping its impact on performance minimum, hence saving from the unauthorized replication of policies. It is also shown that the unwatermarked policies are not able to follow the identified trajectory, which is specified during the training. As already discussed, this can be used by adversaries to hide specific patterns in the policy and use them to their benefit later.

4) Manipulating the Sensors: The research on real-time attacks on robotic systems in a dynamic environment has not been much explored. Clark *et al.* [28] evaluate a white-box adversarial attack on the DRL policy of an autonomous robot in a dynamic environment. The goal of the DRL robot is to reach the destination by routing through the environment, while the goal of the adversary is to mislead the agent into the wrong routes. The adversary misleads the agent into following false routes by tampering with sensory data. They also observe that once the adversarial input is removed, the robot automatically reverts to taking the correct route. Hence, an attacker can modify the behavior of the model temporarily and leave behind zero or very little evidence. Their attack requires access to the trained policy but not the hyperparameters used during training.

A similar observation has been shown by Usama *et al.* [53]. They argue that a lot of research is being done for creating AI/ML solutions to problems in future networks, such as Internet of Things and 6G. They show these ML systems to be vulnerable by highlighting the adversarial dimension of these systems. They prove their point by attacking a DRL-based channel autoencoder framework and showing its drop in performance. Noise is added to the feedback channel for a certain time interval. Furthermore, they show that when this noise is removed, the DRL system automatically can regain its original performance, leaving no footsteps by the adversary.

B. Attacks Perturbing the Reward Function

Han *et al.* [54] discuss the reaction of the DRL agent in SDN to different adversarial attacks. The adversary adopts white-box and black-box settings for both inference and poisoning attacks in an online setting. They propose two types of attacks: *flipping reward signals* and *manipulating states*. For flipping reward signals, the adversary can manipulate the binary reward signal of the model by flipping it a certain number of times. For state manipulation, the attacker makes two changes in the first few steps of the training, i.e., an adversary can change the binary reward of two states from 0 to 1 and 1 to 0, respectively. Hence, the adversary can change the label of one compromised node to be uncompromised and *vice versa*.

A similar falsification approach has been followed by Huang and Zhu [55] leading the agent into taking targeted decisions. They characterize a robust region for policy, in which the adversary can never achieve the desired policy while keeping the cost in this region. They use four terms to specify different types of attackers: 1) *omniscient attacker* who has all the information before a certain time t; 2) *peer attacker* who does not know about the transition probabilities but has access to the knowledge the agent has before a time t; 3) *ignorant attacker* who only knows the cost signals before a time t; and 4) *blind attacker* that has no information at time t. All these attackers may be limited by the budget of the attack and other constraints. It is shown that by the falsification of the cost at each state, all of these adversaries can mislead the agent into learning a policy desired by the adversary.

Rakhsha *et al.* [56] propose a training-time attack involving the poisoning of the learning environment to force the agent into executing a target policy chosen by the adversary. They consider RL agents that maximize average reward in undiscounted infinite-horizon settings and argue this to be a more suitable objective for many real-world applications that have cyclic tasks or tasks without absorbing states, e.g., a scheduling task and an obstacle-avoidance robot. They suppose that the adversary can manipulate the rewards and the transition dynamics in the learning environment at training time in a stealthy manner. They test their attack in both offline and online settings. In the former, the agent is planning in a poisoned environment while in the latter, the agent is learning a policy using a regret-minimization framework with poisoned feedback. They show that the adversary can easily succeed in teaching an adversarial policy to the RL agent.

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

C. Attacks Perturbing the Action Space

The adversary can have access to the actuators and might try to perturb the actions taken. Yeow *et al.* [57] propose two attacks on the action space of the DRL algorithms. The first one is an optimization problem for minimizing the cumulative reward of the DRL agent with decoupled constraints called *MAS* attack. The second one has the same objective as the first one but with temporally coupled constraints called *LAS* attack. The results show that LAS is more lethal in deteriorating the performance of the DRL algorithm as it can attack the dynamic information of the agent. This attack is also shown to perform well in the case of limited resources. Such attacks can be used to gain insights into the potential vulnerabilities of the DRL model. They also speculate that their proposed attacks on reward signals by perturbing the action space cannot be defended as the action space is independent of the policy. However, it can be detected by having a look at the decay in the reward.

Due to the difficulty of obtaining the complex models for cyber-physical systems for traditional control, they are being shifted to DRL. Lee *et al.* [65] argue that before transferring these systems to DRL, the security limitations of DRL must be understood. They propose a query-based attack for perturbing the action space of DRL in such systems. Furthermore, they show that by the use of adversarial training, the attack success can be reduced to half.

D. Attacks Perturbing the Model Space

The adversary can have access to the model during or after training. Based on this access, the adversary might try to manipulate the model into learning the adversarial behavior or might also try to extract the learned model and use it later for attack purposes. Behzadan and Hsu [58] propose an adversarial attack for targeting the confidentiality of the DRL policy. The proposed attack performs a model extraction attack by using imitation learning while querying the original model iteratively. They show that the adversarial examples generated for the model extracted are transferred successfully to the original model hence affecting its performance in a black-box setting. They use the FGSM for generating adversarial examples for the imitated model. It is also shown that by providing the attack a sufficient number of observations, adversarial examples can be crafted with high efficiency. They use adversarial regret, i.e., the difference between maximum return achievable by the trained policy π and return achieved from actions of adversarial policy, as a metric to measure the performance of their attacks. They show an increase in adversarial regret in case of an adversarial policy.

Chen *et al.* [61] argue that the techniques used for model extraction in supervised ML cannot be applied to RL due to high complexity and limited observable information and propose a technique for model extraction in DRL. At first, they use an RNN classifier to reveal the training algorithm of the target black-box DRL model based on the predicted actions. Then, they use imitation learning to replicate the victim model from the extracted algorithm. A PPO is used for imitation learning. The extraction of models can be used by adversaries to generate successful adversarial examples making deployed models even more vulnerable to adversarial attacks.

Huai *et al.* [64] propose an optimization framework for deriving optimal adversarial attack strategy for model poisoning attacks. They propose two attacks: one in which adversarial perturbations are added to the observations of the agent and the other in which the attacker modifies the parameters of trained models in such a way that their performance is not affected. The first one is termed as *universal adversarial attack against DRL interpretations* (UADRLI) while the latter is termed as a *model poisoning attack against DRL interpretations* (MPDRLI). For UADRLI, they assume that the adversary has access to a certain area of the images (states) and cannot perturb pixels outside this certain area. The perturbations are only added at some time steps.

E. Discussion

In this section, we discuss the attacks on DRL by categorizing them based on the targeted part of the MDP. The adversary can target the state space, action space, reward function, or the model space based on the access available to the adversary. When targeting the state space, the adversary can add perturbations to the environment, training data, observations, and sensory data. In the case of perturbing the action space, the adversary can target the actuators. In the case of perturbing the reward function, the adversary can perturb the reward signal or might flip it. In the case of model-space attacks, the adversary can perturb the learned parameters of the model or might attempt to extract the learned model, which might be proprietary, i.e., owned and copyrighted by some organization.

In real environments, the attacks that generate imperceptible and natural perturbations are more practical than the attacks that involve adding specially designed perturbations to states. In applications like autonomous driving, getting direct access to the sensors might not be possible for the adversary. The only option is to perturb the environment, hence indirectly affecting the observations, actions, rewards, and policies. The real environments are often black-box, where the adversary has no knowledge of the system being attacked and the number of queries is limited. The adversary has to improvise to attack the system where the target of the adversary can be to cause a drop in performance of the system or to evade the system. This puts forward a need for query-efficient attacks, similar to those proposed in [25] for supervised ML, to be proposed for DRL.

Table I shows a summary of the adversarial attacks on DRL.

IV. Defenses Against Adversarial Attacks on DRL

In this section, we provide a detailed review of the countermeasures proposed to deal with adversarial attacks on DRL. Fig. 8 shows a basic taxonomy of the defenses that can be used for securing DRL algorithms.

A. Adversarial Training

Adversarial training includes retraining of the ML model using the adversarial examples along with the legitimate examples. This increases the robustness of the ML model against

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

TABLE I
SUMMARY OF THE ADVERSARIAL ATTACKS ON DRL PIPELINE HIGHLIGHTING THE THREAT MODEL AND ATTACK LOCATION IN THE DRL PIPELINE

Paper	DRL Technique Targeted	Environment	Test-time Attacks		Train-time Attacks		Attack Target			
			White Box	Black Box	White Box	Black Box	State	Action	Reward	Model
Behzadan and Munir [17]	DQN	pong	✓	✓	✗	✓	✓	✗	✗	✗
Huang et al. [31]	DQN, TRPO & A3C	chopper command, pong, seaquest, space invaders	✓	✓	✗	✓	✓	✗	✗	✗
Kos and Song [37]	A3C	pong	✓	✗	✗	✗	✓	✗	✗	✗
Pattanaik et al. [36]	DQN & DDPG	cartpole, mountain car	✓	✗	✗	✗	✓	✗	✗	✗
Lin et al. [32]	DQN & A3C	pong, seaquest, mspacman, chopper command, qbert	✓	✗	✗	✗	✓	✗	✗	✗
Tretschk et al. [34]	DQN	Pong	✓	✗	✗	✗	✓	✗	✗	✗
Clark et al. [28]	DQN	Pathfinding	✓	✗	✗	✗	✓	✗	✗	✗
Chen et al. [47]	A3C	Pathfinding	✗	✗	✓	✗	✓	✗	✗	✗
Han et al. [54]	DDQN & A3C	Software-defined networking	✓	✓	✓	✗	✗	✗	✓	✗
Behzadan and Hsu [58]	DQN, A2C & PPO	Cartpole	✗	✓	✗	✗	✗	✗	✗	✓
Kiourti et al. [51]	A2C	Pong, Space Invaders, Qbert, Breakout, Seaquest, Crazy Climber	✗	✗	✓	✓	✓	✗	✗	✗
Yeow et al. [57]	PPO and DDQN	Lunar-Lander, BiPedal Walker	✗	✗	✓	✗	✗	✓	✗	✗
Hussenot et al. [39]	DQN & Rainbow DQN [59]	pong, space invaders, air raid, and HERO	✓	✗	✓	✗	✓	✗	✗	✗
Xiao et al. [46]	DQN & DDPG	TORCS [60], Atari (pong, enduro), MuJoCo (half-cheetah, hopper)	✓	✓	✗	✗	✓	✓	✗	✗
Huang and Zhu [55]	Q-Learning	water reservoir system	✗	✗	✓	✗	✗	✗	✓	✗
Behzadan and Hsu [52]	DQN	cartpole	✗	✗	✓	✗	✓	✗	✗	✗
Bai et al.[48]	DQN	Pathfinding	✗	✗	✓	✗	✓	✗	✗	✗
Gleave et al. [49]	PPO	MuJoCo: kick and defend, you shall not pass, sumo humans, sumo ants	✗	✓	✗	✗	✓	✗	✗	✗
Sun et al. [38]	A3C, DDPG, & PPO	TORCS, Atari (pong and breakout), and MuJoCo (inverted pendulum, half-cheetah, hopper, and walker2D)	✓	✓	✗	✗	✓	✗	✗	✗
Lin et al. [44]	c-MARL	StartCraft II	✗	✓	✗	✗	✓	✗	✗	✗
Yang et al. [50]	DQN & A3C	reacher, banana collector, and donkey car	✗	✗	✓	✓	✓	✗	✗	✗
Raksha et al. [56]	RL	obstacle-avoidance, scheduling	✗	✗	✓	✗	✗	✗	✓	✗
Wang et al. [45]	DQN	Energy management system of an electric vehicle	✓	✓	✗	✗	✓	✗	✗	✗
Chen et al. [61]	DQN, PPO, ACER [62], ACKTR [63] & A2C	cartpole and pong	✗	✓	✗	✗	✗	✗	✗	✓
Huai et al. [64]	A3C, DQN	pong, breakout, space invaders	✓	✗	✓	✗	✓	✗	✗	✓
Chan et al. [40]	DQN, PPO	freeway, pong, breakout, fishing-derby, space-invaders, battlezone	✓	✓	✗	✗	✓	✗	✗	✗
Usama et al. [53]	Basic DRL	SDN	✗	✓	✗	✗	✓	✗	✗	✗
Lee et al. [65]	PPO	point goal, car goal [66]	✗	✓	✗	✓	✗	✓	✗	✗

adversarial examples as the model is now able to learn a better distribution. Although adversarial retraining can help improve the robustness of the ML model, the ML model can still be compromised through adversarial examples generated through some other methods. The goal of adversarial training is to improve the generalization outside of the training manifold. Kos and Song [37] proposed using adversarial training for robustifying DRL algorithms. They retrain their agent on perturbations generated using the FGSM and random noise and show performance retention against similar attacks. Furthermore, they observe that the retrained agent is also resilient against FGSM perturbations having magnitudes different than the one used for retraining.

Pattanaik *et al.* [36] also adopt adversarial training as a measure to make the algorithms robust against GB attacks. They show its equivalence to robust control. They train the DRL model by using the adversarial samples generated from the GB attacks. This helps the algorithm to model uncertainties in the system making them robust to similar adversarial attacks. They show that the addition of noise to the training samples while training increases the resilience of the DRL models against adversarial attacks. Han *et al.* [54] also propose adversarial training as a method of robustifying the DRL algorithms against adversarial attacks. They show that this technique is effective when countering attacks, such as node corruption and node falsifying, in SDN.

Behzadan and Munir [67] find the adversarially trained policies to be more robust to test-time attacks. They investigate the robustness of DRL algorithms to both training and test-time attacks and find out that under the training-time attack the DQN can learn and become robust by changing the policy. They propose that for an agent to recover from adversarial attacks, the number of the adversarial samples in the memory needs to reach a critical limit. In this way, when the agent samples a random batch from the memory, it can learn the perturbation statistics. They also compare the performance of ϵ-greedy and parameter-space noise exploration methods in case of adversarial attacks. They show the ϵ-greedy methods to be more robust to training-time attacks than the noisy exploration technique. They also find noisy exploration techniques to be able to recover faster from attacks when compared to the ϵ-greedy methods.

Later on, Behzadan and Munir [68] compare the resilience to adversarial attacks of two DQNs: one based on ϵ-greedy policy learning and another employed NoisyNets [69], which is a parameter-space noise exploration technique. Their results show the NoisyNets to be more resilient to training-time attacks than that of the ϵ-greedy policy. They argue that this resilience is due to the enhanced generalizability and reduced transferability in NoisyNets. They propose that by using parameter-space noise exploration, the DRL algorithms can be made robust to attack techniques like FGSM. Chen *et al.* [47] propose a GB adversarial training technique. They use adversarial perturbations generated using their proposed attacking algorithm, i.e., CDG, for retraining the RL agent. This approach can achieve a precision of 93.89% in detecting adversarial samples. They prove that

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

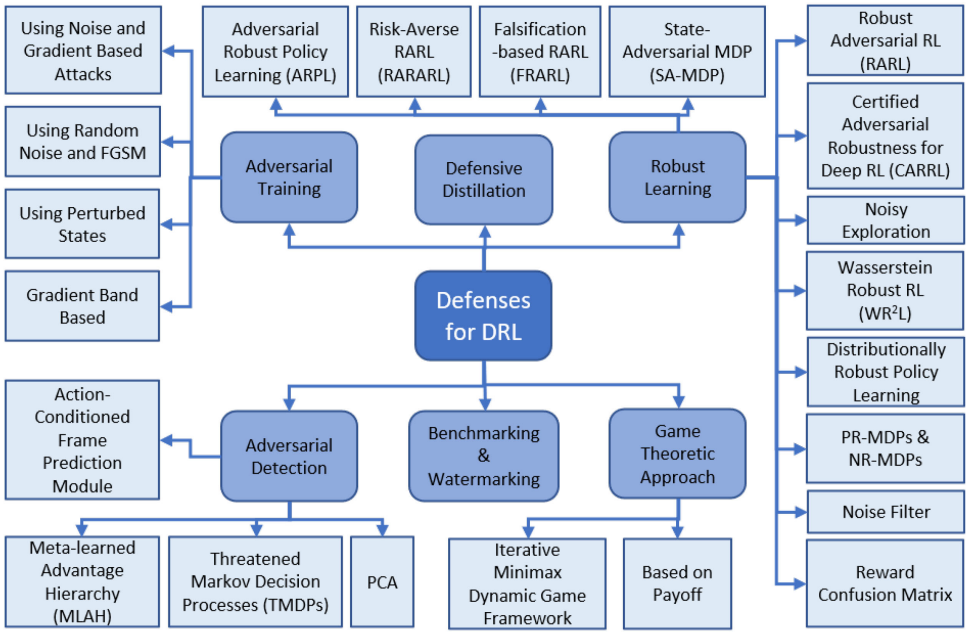

Fig. 8. Taxonomy of the major defense schemes used in DRL.

adversarial training using only a single adversarial example, generated using CDG, can realize the generalized CDG-attack immunity of A3C pathfinding with high confidence. Behzadan and Hsu [70] propose AGE after considering the sample inefficiency of current adversarial training techniques. Their technique is based on a modified hybrid of the ϵ-greedy algorithm and the Boltzmann exploration (exploring probabilistically relative to expected rewards). The new adversarial training procedure is tested on DQN trained for the CartPole environment with different perturbation probabilities. They show that for small perturbations probabilities, i.e., 0.2 and 0.4, the agent can recover from the attack, while in the case of large probabilities such as 0.8 or 1, the agent is not able to recover. They compare the efficiency of their proposed technique with ϵ-greedy and parameter-space noise exploration algorithms and prove its feasibility.

Tan *et al.* [71] argue that although the DRL algorithms used for decision and control tasks are vulnerable to adversarial attacks, little research has been done to make them robust. After showing the vulnerability of well-trained DRL agents to action space attacks, they use adversarial training to increase the robustness of the attacked model. Furthermore, a performance improvement of the adversarially trained agent over the normal agent in nonadversarial scenarios is also shown. Lee *et al.* [65] also show that the use of adversarial training with their proposed attack decreases the attack success rate to half.

Vinitsky *et al.* [72] argue that the existing literature on robust learning in DRL focuses on training a single RL agent against a single adversary, and these systems are bound to fail in the case of a different adversary. They propose a population-based augmentation to the robust RL formulation, in which a population of adversaries is randomly initialized, and samples are drawn uniformly from the population during training.

B. Game-Theoretic Approach

Pinto *et al.* [73] propose RARL as a method of robust policy learning in the presence of an adversary. They formulate policy learning as a zero-sum minimax objective function to ensure robustness to differences in test and train conditions, even in the presence of an adversary. They use a self-proposed adversarial agent with an especially designed reward targeted at finding the state-space trajectories that lead to the worst rewards. They call these trajectories hard examples. An adversary is introduced in the environment, whose goal is to destabilize the RL agent. Abdullah *et al.* [74] propose a robust RL using a novel min–max game with a Wasserstein constraint for a correct and convergent solver. This technique shows a significant increase in robustness in the case of both low- and high-dimensional control tasks. They also discuss that by using their technique, the DDPG algorithms are not able to achieve significant performance improvement in robustness, even in the case of inverted pendulum, while the other two DRL schemes, i.e., TRPO and PPO, demonstrate acceptable performance and, hence, are reported in their results.

Bravo and Mertikopoulos [75] examine a game approach, where the players adjust their actions based on past payoff observations that are subject to adversarial perturbations. In the single-player case containing an agent trying to adapt to an arbitrarily changing environment, they show that irrespective of the level of noise in the player's observations, the stochastic dynamics under study leads to no regret almost surely. In the case of multiple players, they show that the dominated strategies become extinct, and the strict Nash equilibrium is stochastically stable and attractive. Conversely, a stable or attractive state with better probability is the Nash equilibrium. Finally, they provide an averaging principle and show that in the case of two-player zero-sum games with an interior equilibrium, the

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

time averages converge to Nash equilibrium for any noise level. Ogunmolu *et al.* [76] propose an iterative minimax dynamic game framework that helps in designing robust policies in the presence of adversarial inputs. They also propose a method of quantifying the robustness capacity of a policy. They evaluate their proposed framework on a Mecanum-wheeled robot. The goal of this agent is to find a locally robust optimal multistage policy that achieves a given goal-reaching task.

C. Robust Learning

Robust learning is a training mechanism to ensure robustness against training-time adversarial attacks. Behzadan and Munir [68] propose adding noise to the parameter state, while training, this technique is found very effective in mitigating the effects of both training- and test-time attacks for both black-box and white-box settings. The results of the proposed method are tested on DQN trained for three Atari games, namely, Enduro, Assault, and Blackout. In particular, the authors use the FGSM for crafting adversarial samples. Then, they show the performance of the normal agents to deteriorate significantly, while the ones that were retrained using the parameter noise show great performance even in the presence of adversarial inputs. Mandlekar *et al.* [77] show superior resilience to adversarial attacks by introducing an ARPL algorithm. This involves the use of adversarial examples during training to enable robust policy learning. They consider the addition of adversarial perturbations not only to the image space but also to the whole state of the system, which in their case also included the parameters such as friction, mass, and inertia. They use the GB FGSM technique for the generation of adversarial samples. They show that in the case of agents that do not follow their learning technique, the performance deteriorates drastically, while their agent can retain the training performance. It is important to note that the agent trained using the ARPL algorithm does not perform as well as the normal one in case of no perturbations.

Wang *et al.* [78] point out that the reward function is susceptible to three kinds of noise, namely, inherent noise, application-specific noise, and adversarial noise. As a remedy, they propose a reward confusion matrix to generate rewards to help the RL agent to learn in cases of perturbed/noisy inputs. Such rewards are called to be *perturbed rewards*. Using these perturbed rewards, they can develop an unbiased reward estimator-aided robust RL framework. Their algorithm not only achieves higher expected rewards but also converges faster. They experiment with their technique extensively using several DRL algorithms, which are trained for different classic Atari gaming environments. Their proposed technique can achieve 67.5% and 46.7% improvements in average reward when the error rate is 10% and 30%, respectively, in the case of PPO. They discuss both the cases of the perturbations added to some samples and perturbations being added to all samples.

Policies that can retain the performance in nonstationary environments are also robust to adversarial attacks that involve adding noise to the state space. Smirnova *et al.* [79] propose a distributionally robust policy iteration scheme to restrict the agent from learning suboptimal policy while exploring in cases of high-dimensional state/action space. This induces a dynamic level of risk to stop the agent from taking suboptimal actions. Their scheme is based on robust Bellman operators, which provide a lower bound guarantee on the policy/state values. They also present a distributionally robust soft actor–critic based on mixed exploration, acting conservatively in the short term and exploring optimistically in a long run leading to an optimal policy. The direct target in [77] is adversarial robustness, while in [79], the direct target is distributional robustness; hence, the target of adversarial robustness was achieved indirectly.

Tessler *et al.* [80] propose PR-MDP and NR-MDP as two new criteria for robustness. They modify the DDPG to form AR-DDPG for solving these MDPs. The proposed techniques are evaluated in various MuJoCo environments, and the results prove that the learning of action-robust policies can help in making the proposed algorithms secure and perform better even in the absence of these perturbations. The adversarial robustness was achieved here by making the agent "*action robust*."

Kumar *et al.* [81] present a technique to make the DRL algorithm learn in the presence of noisy rewards. The proposed scheme is based on using a neural network as a noise filter targeted at estimating the true reward of the environment. These estimates are then compared with the reward the agent gets at each state, to filter out the noisy samples. They show that beyond the perturbation probability of 0.5, their agent starts to learn based on adversarial samples rather than the normal ones.

Fisher *et al.* [82] propose the idea of *robust student DQN* (RS-DQN). They propose to split the standard DQN into two networks, namely, a student (policy) network S and a Q-network. This S network is robustly trained and used for exploration, while the Q-network is trained normally. This permits online robust training while keeping the competitive performance of the Q-networks. They show that in the case of no attacks, the DQN and the RS-DQN show the same performance, while in the case of adversarial attacks, the DQN fails, while the RS-DQN remains robust. Furthermore, they show that the RS-DQN when combined with state-of-the-art adversarial training provides resilience to strong adversarial attacks during training and evaluation.

Pan *et al.* [83] argue that training RL on physical hardware is dangerous due to exploration and can also be slow due to high sample complexity. This puts forward a need for a robust learning algorithm that can make sure that the policy performs well in catastrophic situations. They propose RARARL, using a risk-averse agent and a risk-seeking adversary. They point out that the RARL technique proposed by Pinto *et al.* [73] has no explicit modeling and optimization of risk as they only optimize the expected control objective. An ensemble of Q-value networks is used to model risk as to the variance of value functions. Their technique is similar to bootstrapped DQNs [84] proposed to assist exploration, but in this case, the purpose of the ensemble is to estimate variance. They test their approach on a self-driving environment using the TORCS simulator and show that a risk-averse agent handles the risk better and leads to fewer crashes than a normal agent trained in a similar environment. The attacker and the victim agent are made to work independently in

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

the same environment. This gives more options to the attacker to perturb the environment, hence serving as a strong adversary.

The overfitting of RL policies to the training environments cause them to fail to generalize to safety-critical scenarios. Wang *et al.* [85] argue that the RARL technique proposed by Pinto *et al.* [73] requires handcrafting sophisticated reward signals, which is a difficult task. They say that safety falsification methods can be used to find a set of initial conditions as well as an input sequence to make a system violate a given property formulated in temporal logic. They propose an FRARL technique for integrating temporal-logic falsification in adversarial learning to improve policy robustness. This removes the requirement of the construction of an extra reward function for the adversary. Their experiments show that the policies trained with FRARL generalize better and show less violation of the safety specifications in test scenarios when compared to techniques similar to RARL.

Lütjens *et al.* [86] argue that adversarial detection (detecting adversarial samples using specialized models) can only detect a perturbed input, but they cannot propose an alternate action in case of an attack. They leverage the research on certified adversarial robustness to develop an online certified defense for DRL algorithms called CARRL. CARRL involves computing lower bounds on the state–action pairs and choosing a robust action in case of an adversarial attack. They make their technique certifiable robust by using robust optimization to consider worst-case uncertainties and to provide certificates on solution quality. They show the effectiveness of their technique on a DQN trained for a collision-avoidance system and a classic control task (CartPole) and using the targeted FGSM as an adversary. Their experiments show that CARRL can 1) recover and avoid the obstacles in case of an adversarial attack on collision avoidance system and 2) recover and achieve a sufficient reward in CartPole environment. Furthermore, they prove that although their technique reduces the computational efficiency of the DQN, it increases the robustness.

Zhang *et al.* [87] point out that the robustness for continuous-action space DRL has not got any attention, and existing approaches lack proper theoretical justification. They prove that the classification techniques like that of adversarial training prove to be inefficient for many RL problems. They develop a theoretically principled policy regularization and propose an SA-MDP. Their technique improves robustness under strong white-box attacks on state observations, including the two new attacks that they have proposed: the robust SARSA attack (RS attack) and MAD attack. Furthermore, they show performance improvement in nonadversarial scenarios. They assume the adversary to 1) be stationary, deterministic, and Markovian, i.e., the adversary does not change with time; and 2) have bounded adversary power, i.e., the adversary can only perturb a specific number of states.

Oikarinen *et al.* [88] propose a method of training DRL agents robust to l_p-bounded attacks. They termed their technique RADIAL-RL. Furthermore, they propose a new metric greedy worst-case reward for evaluating the performance of DRL algorithms against adversarial attacks. They show their technique to outperform the state-of-the-art robust learning techniques [82], [87] under PGD attack.

Zhang *et al.* [89] propose a technique to enhance the robustness of DRL agents against learned adversary attacks, i.e., attacks in which the adversary is continuously learning. They term their technique as *alternating training with learned adversaries* (ATLA). Their technique involves the training of an adversarial agent online together with the victim agent using policy gradient following the optimal adversarial attack framework.

D. Adversarial Detection

Adversarial detection involves the detection of adversarial samples using a model especially trained to segregate the true samples from the adversarial ones. In this way, we can disregard the adversarial inputs without modifying the original model. Lin *et al.* [90] propose a method of protecting the DRL algorithms from adversarial attacks by leveraging an action-conditioned frame prediction module. By using this technique, they can detect the presence of adversarial attacks and make the model robust by using the predicted frame instead of the adversarial frame. They also compare their results with other ML defense approaches to show the effectiveness of this technique. The techniques used for adversarial example generation are FGSM, C&W [33], and basic iterative method [91]. The present results indicate that their proposed technique can detect adversarial attacks with accuracy from 60% to 100%.

Havens *et al.* [92] detect the presence of adversarial attacks via a supervisory agent by learning separate subpolicies using the MLAH framework. Because this technique can handle the attacks in the decision space, it can mitigate the learned bias introduced by the adversary. They consider a policy learning problem that is being attacked at specific periods. The goal of the adversary is the corruption of state space while the agent is being trained. They assume that while training, the agent learns subpolicies before learning the ultimate policy. Thus, the supervisory agent can detect the presence of the adversarial examples due to them being unexpected. They use a self-defined adversarial agent having the ability to perturb the states before they reach the agent for training. The perturbations generated by this agent are bounded by l_∞-norm.

Xiang *et al.* [93] propose an advanced Q-learning algorithm for automatic pathfinding in robots, that is robust to adversarial attacks by detecting the adversarial inputs. Specifically, they propose a model to predict the adversarial inputs based on a calculation determined by five factors: energy point gravitation, key point gravitation, path gravitation, included angle, and the placid point. The weights for these five factors are calculated based on the PCA. Using these factors, they train a model able to achieve a precision of 70% in segregating adversarial inputs from the normal ones.

Gallego *et al.* [94] introduce TMDPs, a variant of MDP. This framework supports the decision-making process in the DRL setting against adversaries that affect the reward generating process. They propose a level-k thinking scheme resulting in a new framework for dealing with TMDPs. They show that while a normal Q-learning algorithm is exploited by an adversary, a level-2 learner can approximately estimate the adversarial

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

behavior and achieve a positive reward. Integrating DQNs to TMDPs is discussed as a future research path.

E. Defensive Distillation

Papernot *et al.* [95] propose the idea of using defensive distillation to deal with adversarial attacks on ML schemes. In this technique, one model is trained to predict the output probabilities of another model that was trained with an emphasis on accuracy. Using this technique, DL models can be made less susceptible to exploitation by adding flexibility to an algorithm's classification process using adversarial training. Carlini and Wagner [96] show that defensive distillation gives a false sense of robustness against adversarial examples. Rusu *et al.* [97] present a method of extracting the policy of a dense network to train another comparatively less dense network. This new network can take expert-level decisions while being smaller in size. This method can also be used to merge multiple task-specific policies into a single policy. They show that the distilled agents, which were four times smaller than DQNs, were able to achieve better performance than the DQN. They also show that the agents having 25 times fewer parameters than the DQN were able to achieve a performance of 84% as compared to 100% of the DQN. Such networks are proved to be more stable and robust to adversarial noise and attacks, as they have fewer parameters than their denser counterparts and hence decreasing the count of attackable parameters.

Recently, Czarnecki *et al.* [98] analyzed empirically and theoretically each variant of distillation and reported the strengths and weaknesses of each variant. Furthermore, they propose *expected entropy regularized distillation*, which makes the training much faster while guaranteeing convergence. This technique can be used in making the DRL models robust to adversarial attacks by leveraging learning information from a complex model into a simpler one, hence making the models robust to adversarial attacks. However, as discussed by Carlini and Wagner [96], using this technique alone may not be effective. It needs to be combined with other approaches, like adversarial training, adversarial detection, etc., to be successful.

Qu *et al.* [99] propose robust policy distillation, i.e., a policy distillation paradigm capable of achieving an adversarially robust student policy without relying on any adversarial example during student policy training. They propose a policy distillation loss consisting of a prescription gap maximization loss and a Jacobian regularization loss. They perform a theoretical analysis and show that their proposed mechanism ensures the learning of robust policies during the distillation process. They show their technique to outperform the one proposed in [87].

F. Discussion

We discuss the state-of-the-art defenses in this section by categorizing them into adversarial training, robust learning, adversarial detection, defensive distillation, and game-theoretic approaches. It can be seen that most of these techniques are only effective against the specified type of adversarial attacks and do not provide any guarantees against other types of attacks. Most of these techniques focus on making the DRL agent learn a robust policy by use of different mechanisms such as training using adversarial examples, simulating min–max games with adversaries, using robust alternatives of MDPs, etc. These techniques are more practical than the ones involving adversarial detection. Due to the advent of new attacking strategies by the day, one can never be sure that a detection mechanism will be able to detect the attack. Furthermore, it is worth noting that there are very few defenses for DRL algorithms that do not involve images as the observations.

Table II summarizes key information of the proposed defenses for DRL algorithms.

V. Metrics, Tools, and Platforms for Benchmarking DRL

As we have previously discussed, DRL is different from other ML schemes, and only reporting the accuracy is not sufficient to cover security aspects of the DRL schemes. In particular, we need to consider the temporal-domain aspect of the DRL while designing the DRL-based attack or defense. Benchmarking the DRL performance in attacks and defenses is very important. The need for an applicable solution to evaluate the robustness and resilience of DRL policies is not fulfilled by the current literature. There is also a need for a quantitative approach to measure and benchmark the resilience and robustness of DRL policies in a reusable and generalizable manner.

There are few benchmarks proposed, but they are not sufficient to cover the security aspects needed to measure the robustness and resilience of DRL algorithms. The few proposed approaches are discussed in this section. Behzadan and Hsu [101] introduce the terms of *adversarial budget* and *adversarial regret* as a measure to quantify the robustness and resilience of DRL algorithms. Adversarial budget is defined as the maximum number of features that can be perturbed in the observation, and the probability of perturbing each observation. The adversarial regret is the difference between the reward obtained by the unperturbed agent and the reward obtained by the perturbed agent after an episode. Based on these two terms, Behzadan and Hsu [101] define test-time resilience and test-time robustness.

A. Test-Time Resilience and Robustness

Test-time resilience is described as the minimum number of perturbations required to incur the maximum reduction to return at time t, while *test-time robustness* is described as the maximum achievable adversarial regret.

The following procedure is proposed to measure test-time resilience for DRL algorithms.

1) Approximate the state–action value function using policy imitation in case it is not already given.
2) Report the optimal adversarial return and maximum adversarial regret by training the adversarial agent against the target's policy.
3) Apply the obtained adversarial policy to the target for several episodes while recording the return for each episode.
4) Report the average adversarial return over these episodes as the mean test-time resilience of the target policy.

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

TABLE II
SUMMARY OF DEFENSES AGAINST ADVERSARIAL ATTACKS ON DRL

Adversarial Training				
Paper	**Proposed Techniques**	**Setup**		**Effective Against**
		Algorithm	**Environment**	
Kos and Song [37]	Adversarial Training using Random Noise and FGSM	A3C	Pong	Random Noise FGSM Attacks
Pattanaik et al. [36]	Adversarial Training using Noise Gradient-Based Attacks	DDQN, DDPG	Cartpole, Mountain Car, Hopper, Half Cheetah	Noise Gradient Based Attacks
Han et al. [54]	Adversarial Training using corrupted nodes in SDN	DDQN, A3C	SDN	Node Corruption Falsifying Attacks
Behzadan and Munir [67]	Adversarial Training using Perturbed States	DQN	Breakout, Pong	Attacks Perturbing a considerable no. of States
Behzadan and Munir [68]	Noisy Exploration	DQN	Enduro, Assault, Blackout	State Perturbation Attacks
Chen et al. [47]	Gradient Band-Based Adversarial Training	A3C	Pathfinding	Gradient Band Based Adversarial Attacks
Behzadan and Hsu [70]	Adversarially Guided Exploration (AGE)	DQN	Cartpole	Limited Attack Samples
Tan et al. [71]	Adversarial Training	PPO	Lunar Lander	Action space Attacks
Lee et al. [65]	Adversarial Training	PPO	point goal, car goal [66]	Action space Attacks
Vinitsky et al. [72]	Adversarial Training using Populations	PPO	Hopper, Ant, Half-cheetah	Generic Adversarial Attacks

Robust Learning				
Paper	**Proposed Techniques**	**Setup**		**Effective Against**
		Algorithm	**Environment**	
Mandlekar et al. [77]	Adversarially Robust Policy Learning (ARPL)	TRPO	Inverted Pendulum, Half-Cheetah, Hopper, Walker	State Perturbation Attacks
Smirnova et al. [79]	Distributionally Robust Policy Iteration	self	Hopper, Walker2D	Attacks Targeting the Policy
Tessler et al. [80]	PR-MDPs, NR-MDPs	self	Hopper, Walker2d, Humanoid, Inverted-Pendulum	Generic Adversarial Attacks
Kumar et al. [81]	Noise Filter	DQN, DDQN	cartpole	Attacks Perturbing the Rewards
Fisher et al. [82]	Robust Student DQN (RS-DQN)	self	Freeway, BankHeist, Pong, boxing, road-runner	Generic Adversarial Attacks
Lutjens et al. [86]	Certified Adversarial Robustness for RL (CARRL)	DQN	collision avoidance, cartpole	Generic Adversarial Attacks
Pan et al. [83]	Risk-Averse Robust Adversarial Reinforcement Learning (RARARL)		TORCS	Generic Adversarial Attacks
Wang et al. [85]	Falsification-based RARL (FRARL)	PPO	brake assistance system, adaptive cruise control system	Generic Adversarial Attacks
Zhang et al. [87]	State-Adversarial Markov Decision Process (SA-MDP)	DQN, PPO, DDPG	Walker, Hopper, Humanoid, Ant, Inverted Pendulum, Reacher, Road Runner, BankHeist, pong, Acrobot, and Freeway	Generic Adversarial Attacks
Oikarinen et al. [88]	RADIAL-RL	DQN, A3C	Pong, Freeway, bankheist, road runner	Generic Adversarial Attacks
Zhang et al. [89]	Alternating Training with Learnable Adversaries (ATLA)	PPO	Hopper, Walker2D, Ant, Half-Cheetah	Generic Adversarial Attacks
Wang et al. [78]	Reward Confusion Matrix	Q-Learning, CEM, SARSA, DQN, PPO, NAF, Dueling DQN, DDPG	CartPole, Pendulum, AirRaid, Alien, Carnival, MsPacman, Pong, Phoenix, Seaquest	Attacks Perturbing the Rewards

Adversarial Detection				
Paper	**Proposed Techniques**	**Setup**		**Effective Against**
		Algorithm	**Environment**	
Lin et al. [90]	Action-conditioned Frame Prediction Module	DQN	Pong, Freeway, Sea-quest, Chopper-Command, Ms-Pacman	Attacks Perturbing the States
Havens et al. [92]	Meta-learned Advantage Hierarchy (MLAH)	self	InvertedPendulum-v2, MountainCarContinuous-v0, Hopper-v2	Training-Time Poisoning Attacks
Xiang et al. [93]	PCA for Adversarial Detection	Q-learning	Pathfinding	Attacks Perturbing the States
Gallego et al. [94]	Threatened Markov Decision Processes (TMDPs)	self	Chicken game	Attacks Affecting Reward Generation

Defensive Distillation				
Paper	**Proposed Techniques**	**Setup**		**Effective Against**
		Algorithm	**Environment**	
Rusu et al. [97]	Defensive Distillation	DQN	pong, space-invaders, Breakout, Freeway, Q-bert,Beamrider, Enduro, MS. Pacman, Seaquest, riverraid	Generic Adversarial Attacks
Qu et al. [99]	Defensive Distillation	DDQN, Rainbow DQN	Freeway, bankheist, pong, boxing, roadrunner	Generic Adversarial Attacks

Game Theoretic Approach				
Paper	**Proposed Techniques**	**Setup**		**Effective Against**
		Algorithm	**Environment**	
Pinto et al. [73]	Robust Adversarial Reinforcement Learning (RARL)	TRPO	InvertedPendulum, Half-Cheetah, Hopper, Swimmer, Walker2D	Attacks Targeting the Performance
Abdullah et al. [74]	Wasserstein Robust Reinforcement Learning (WR^2L)	DDPG, TRPO, PPO	CartPole, Hopper, Halfcheetah, Walker2D	Generic Adversarial Attacks
Bravo and Mertikopoulos [75]	Game-Theoretic Approach	-	-	Noise Based Attacks
Ogunmolu et al. [76]	Game-Theoretic Approach	self	Goal Reaching Task	Attacks Targeting the Policy

Others				
Paper	**Proposed Techniques**	**Setup**		**Effective Against**
		Algorithm	**Environment**	
Behzadan and Munir [100]	Benchmarking	DDPG	Torcs collision avoidance	Generic Adversarial Attacks
Behzadan and Hsu [52]	Water Marking	DQN	cartpole	Model Extraction Attacks

The method of measuring the test-time robustness is the same as test-time resilience. The only difference is that in the test-time case, we measure the average adversarial regret in place of the average adversarial reward.

Behzadan and Munir [100] propose a novel framework for benchmarking the behavior of DRL-based collision avoidance mechanisms under the worst-case scenario of dealing with an adversarial agent, which is trained to drive the system into unsafe states. They prove the practical applicability of the technique by comparing the reliability of two collision avoidance systems against intentional collision attempts. More recently, Behzadan and Hsu [52] have presented a technique for watermarking DRL policies for robustness against model extraction attacks. This involves the integration of a unique response to a specific

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

sequence of states while keeping its impact on performance minimum hence saving from the unauthorized replication of policies. It is shown that unwatermarked policies are not able to follow the identified trajectory.

B. Metrics for Attack Performance

Kiourti *et al.* [51] introduce three metrics for measuring the performance of the DRL attacks: performance gap, percentage of target action, and time to failure. As the name suggests, the performance gap is the difference between the performance of the normal and the victim model. For the second metric (percentage of target action), they measure the number of times the adversarial/targeted action is performed by the victim policy. The third metric (time to failure) is the number of consecutive states that need to be perturbed to trigger a complete failure of the model.

As observed, these proposed measurement techniques can only cover a part of the DRL algorithms and, hence, are not sufficient for measuring the performance of the DRL algorithms under the wide range of adversarial attacks and defenses. There is, therefore, a need for the development of benchmarks that can be used as standards for DRL algorithms as a measure of their resilience and robustness to adversarial attacks.

C. Attacking DRL: Tools and Platforms

DRL can be implemented using several available toolkits or by using a combination of these toolkits. Some of the ways to implement DRL are the following.

1) OpenAI Gym [102] is a toolkit for testing the RL algorithms, which provides with multiple gaming environments such as pong, space invaders, and lunar lander. This toolkit is combined with Tensorflow [103] to test the DRL algorithms. The DNN part can be implemented on the later one, and then, the choice of actions based on the states is done by the neural network.
2) OpenAI Baselines [104] provide a set of high-quality implementations of RL algorithms.
3) RLCoach [105] provides with integrated mechanisms of implementing the DNN and testing DRL algorithms.
4) Horizon [106] is an open-source project (now known as ReAgent [107]) that also provides integrated mechanisms for testing multiple DRL algorithms.
5) Ns3-gym platform [108] provides with network environments to test RL algorithms. This again can be combined with Tensorflow [103] to test DRL algorithms.

All of these toolkits can be further combined with the toolkits available for attacking DL [109] and DRL [67] to test different attacks and defenses on DRL algorithms in simulated environments.

VI. Open Issues and Research Challenges

We identify the following major open issues and research challenges in DRL techniques. At the end of this section, we have also provided a roadmap to secure and robustify DRL.

A. Universally Robust Algorithms

Despite the presence of the various defenses that have been proposed, the security of DRL algorithms remains an open challenge. The proposed defenses are only able to defend from attacks they are designed for. Hence, they are still vulnerable to attacks led by proactive adversaries. Moosavi-Dezfooli *et al.* [110] point out that no matter how many adversarial examples are added to the training data, there are new adversarial examples that can be generated to cheat those newly trained networks. Moreover, if the adversary is only targeting confidence levels, then we may never be able to detect the attack until the adversary uses his created deficiency for his benefit. We may not be even able to trace the attacks as shown by Clark *et al.* [28]. Thus, methods to make the DRL algorithms more robust are an urgent need.

B. Multitask Learning

One of the major challenges for DRL is learning to do multiple tasks at a single time. It requires a lot of samples for this. Currently, proposed DRL algorithms can only learn to perform one task perfectly. They can be trained to play multiple games (like CartPole, Inverted Pendulum, etc.), but they need to be trained from scratch for each game. The algorithms are expected to be scalable and be more generalizable so that their learning can be transferred from one game to another. Multitask learning can help in making robust models that can grip the true essence of the tasks and, hence, become difficult to be fooled.

C. Metrics for Robustness and Resilience

We need to study why vulnerabilities exist in DRL models and how we can mitigate them and train robust models. A major reason for the existence of these vulnerabilities is the use of DRL models without the proper knowledge of the domain. There is a need to properly define the benchmarks of DRL in terms of the robustness of DRL against adversarial attacks. Behzadan and Hsu [101] have proposed techniques for quantifying the robustness and resilience of the RL algorithms. Some benchmarks are also proposed by Kiourti *et al.* [51], but as previously discussed, these benchmarks are inadequate to measure the robustness and resilience of an algorithm even though they can be used as stepping stones to lead to a final goal.

D. System Design and Transferability

System design remains an open challenge for the case of DRL. There is a need to define standards for system design for DRL problems as in this case, the learning process is not supervised. So, the agent may not focus on the features that it needs to learn. This can introduce the error by mistake of the intermediary and also even induce his behavior on the model. We need to have proper standards for designing the reward functions. The system design needs to be robust and resilient to adversarial attacks.

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

E. Ensemble of Defenses

Various ensemble defenses have been proposed for the case of DL. However, they may not be appropriate to apply in the case of DRL as it can lead to an exponential increase in the complexity of the model, which results in a significant decrease in performance. In the case of DRL, the model is making a real-time prediction, so a small reduction in the computation capabilities may cause a great loss to the agent. This remains a challenge to defend DRL models using an ensemble with a minimum loss of computations.

F. Model Privacy

Privacy has become a leading issue these days, and model extraction attacks pose a serious threat to the integrity of the learned models through illegal duplication. A mitigation for this, suggested by Behzadan and Hsu [58], is to increase the cost of such attacks or to watermark the policies. We may experience some randomness in the agent to save from such attacks, but that will incur an unacceptable loss of decreased performance. Developing techniques that can incur constrained randomization in the model to save from such attacks is a promising field of research.

G. Explainable and Transparent DRL

For deploying AI systems in real-world scenarios, trust is a key component. The developer needs to be confident of the employed model's decisions. Transparency ensures that the model is fair and ethical while explainability helps to explain and justify the model's decisions. There are a few articles that discuss explainability for DRL [111], [112]. Current techniques for explaining DRL do not specifically focus on targeting specific audiences, i.e., tester, developer, and the general public, and there is a need for the development of such techniques [112]. Only through their development, we might be able to make DRL responsible, trustworthy, and applicable in critical applications.

H. Transfer Learning for DRL

Transfer learning [113] is a field of ML that does not require the training data and test data to be independent and identically distributed. The model is not needed to be trained from scratch in case of a different domain, hence significantly reducing the demand of training data and training time of the target model. The research on transfer learning in the context of DRL has been limited, and there are a few papers that test and discuss transfer learning for only some specific DRL algorithms [114]. Transfer learning can help with saving training time for these models, hence making DRL more applicable to real-world scenarios.

I. Roadmap Toward Secure and Robust DRL

The ultimate goal of research in AI is to develop AGI, which can perform similar activities as humans in a more efficient manner. In addition to the algorithm being able to learn the task at hand efficiently, it also has to be computationally efficient while being robust to adversarial attacks. Furthermore, algorithms need to be sample efficient to be able to learn quickly in real environments as there might not always be enough time to wait for the algorithm to converge. Meeting all these requirements at the same time is a challenging task, and one has to strike intelligent tradeoffs between them.

In this regard, the first task to achieve is the development of sample-efficient and inherently robust DRL algorithms. The second task is the development of explainability techniques, which can explain the behavior of these algorithms in accordance with human perception. The final task will be the development of metrics that can be used to quantify the robustness and resilience of these DRL algorithms. Based on these metrics and the sample and computational efficiency, one can then choose the most suitable algorithm for the task at hand.

VII. Conclusion

The broadening applicability of DRL in the real world has directed our concern to the security of these algorithms against adversarial attacks. This article has provided a comprehensive survey of the latest techniques proposed for attacking DRL algorithms and the defenses proposed for defending against these attacks. We have also discussed the open research issues and provided the list of available benchmarks for measuring the resilience and robustness of DRL algorithms.

Acknowledgment

The statements made herein are solely the responsibility of the authors.

References

[1] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning From Data*, vol. 4. New York, NY, USA: AMLBook, 2012.
[2] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.
[3] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2017, pp. 3389–3396.
[4] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "DeepLoco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Trans. Graph.*, vol. 36, no. 4, Dec. 2017, Art. no. 41.
[5] A. R. Fayjie, S. Hossain, D. Oualid, and D.-J. Lee, "Driverless car: Autonomous driving using deep reinforcement learning in urban environment," in *Proc. 15th Int. Conf. Ubiquitous Robots*, Jun. 2018, pp. 896–901.
[6] A. Raghu, M. Komorowski, L. A. Celi, P. Szolovits, and M. Ghassemi, "Continuous state-space models for optimal sepsis treatment-a deep reinforcement learning approach," in *Proc. Mach. Learn. Healthcare*, Apr. 2017, pp. 147–163.
[7] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 653–664, Feb. 2016.
[8] V. François-Lavet, D. Taralla, D. Ernst, and R. Fonteneau, "Deep reinforcement learning solutions for energy microgrids management," in *Proc. Eur. Workshop Reinforcement Learn.*, Dec. 2016.
[9] C. N. Madu, C.-H. Kuei, and P. Lee, "Urban sustainability management: A deep learning perspective," *Sustain. Cities Soc.*, vol. 30, pp. 1–17, 2017.
[10] N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surv. Tut.*, vol. 21, no. 4, pp. 3133–3174, May 2019.
[11] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Jun. 2015.
[12] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Dec. 2016.

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

[13] D. Silver *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[14] C. Berner *et al.*, "Dota 2 with large scale deep reinforcement learning," 2019, *arXiv:1912.06680*.

[15] B. R. Kiran *et al.*, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, to be published, doi: 10.1109/TITS.2021.3054625.

[16] Z. Zhang, S. Zohren, and S. Roberts, "Deep reinforcement learning for trading," *J. Financial Data Sci.*, vol. 2, no. 2, pp. 25–40, 2020.

[17] V. Behzadan and A. Munir, "Vulnerability of deep reinforcement learning to policy induction attacks," in *Proc. Int. Conf. Mach. Learn. Data Mining Pattern Recognit.*, Jul. 2017, pp. 262–275.

[18] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018.

[19] V. Behzadan and A. Munir, "The faults in our pi stars: Security issues and open challenges in deep reinforcement learning," 2018, *arXiv:1810.10369*.

[20] OpenAI, 2018. Accessed: Jan. 21, 2021. [Online]. Available: https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

[21] Y. Li, "Deep reinforcement learning," 2018, *arXiv:1810.06339*.

[22] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The space of transferable adversarial examples," 2017, *arXiv:1704.03453*.

[23] M. Cheng, T. Le, P.-Y. Chen, J. Yi, H. Zhang, and C.-J. Hsieh, "Query-efficient hard-label black-box attack: An optimization-based approach," 2018, *arXiv:1807.04457*.

[24] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, 2017, pp. 15–26.

[25] C.-C. Tu *et al.*, "AutoZOOM: Autoencoder-based zeroth order optimization method for attacking black-box neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, pp. 742–749.

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

[27] A. Qayyum, M. Usama, J. Qadir, and A. Al-Fuqaha, "Securing connected & autonomous vehicles: Challenges posed by adversarial machine learning and the way forward," *IEEE Commun. Surv. Tut.*, vol. 22, no. 2, pp. 998–1026, Apr.-Jun. 2020.

[28] G. Clark, M. Doran, and W. Glisson, "A malicious attack on the machine learning policy of a robotic system," in *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./12th IEEE Int. Conf. Big Data Sci. Eng.*, Aug. 2018, pp. 516–521.

[29] Y. Vorobeychik and M. Kantarcioglu, "Adversarial machine learning," *Synth. Lectures Artif. Intell. Mach. Learn.*, vol. 12, no. 3, pp. 1–169, Jun. 2018.

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

[31] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," in *Proc. Int. Conf. Learn. Represent. Workshop*, 2017.

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

[33] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy.*, May 2017, pp. 39–57.

[34] E. Tretschk, S. J. Oh, and M. Fritz, "Sequential attacks on agents for long-term adversarial goals," 2018, *arXiv:1805.12487*.

[35] S. Baluja and I. Fischer, "Learning to attack: Adversarial transformation networks," in *Proc. Assoc. Adv. Artif. Intell.*, Feb. 2018, pp. 2687–2695.

[36] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," in *Proc. 17th Int. Conf. Auton. Agents MultiAgent Syst. Int. Found. Auton. Agents Multiagent Syst.*, Jul. 2018, pp. 2040–2042.

[37] J. Kos and D. Song, "Delving into adversarial attacks on deep policies," in *Proc. Int. Conf. Learn. Represent. Workshop*, Apr. 2017.

[38] J. Sun *et al.*, "Stealthy and efficient adversarial attacks against deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 5883–5891.

[39] L. Hussenot, M. Geist, and O. Pietquin, "Targeted attacks on deep reinforcement learning agents through adversarial observations," *Auton. Agents Multi-Agent Syst.*, May 2020.

[40] P. P. Chan, Y. Wang, and D. S. Yeung, "Adversarial attack against deep reinforcement learning with static reward impact map," in *Proc. 15th ACM Asia Conf. Comput. Commun. Secur.*, 2020, pp. 334–343.

[41] C. de Vrieze, S. Barratt, D. Tsai, and A. Sahai, "Cooperative multi-agent reinforcement learning for low-level wireless communication," 2018, *arXiv:1801.04541*.

[42] M. Wiering, "Multi-agent reinforcement learning for traffic light control," in *Proc. 17th Int. Conf. Mach. Learn.*, 2000, pp. 1151–1158.

[43] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," 2016, *arXiv:1610.03295*.

[44] J. Lin, K. Dzeparoska, S. Q. Zhang, A. Leon-Garcia, and N. Papernot, "On the robustness of cooperative multi-agent reinforcement learning," 2020, *arXiv:2003.03722*.

[45] P. Wang, Y. Li, S. Shekhar, and W. F. Northrop, "Adversarial attacks on reinforcement learning based energy management systems of extended range electric delivery vehicles," 2020, *arXiv:2006.00817*.

[46] C. Xiao *et al.*, "Characterizing attacks on deep reinforcement learning," 2019, *arXiv:1907.09470*.

[47] T. Chen *et al.*, "Gradient band-based adversarial training for generalized attack immunity of A3C path finding," 2018, *arXiv:1807.06752*.

[48] X. Bai, W. Niu, J. Liu, X. Gao, Y. Xiang, and J. Liu, "Adversarial examples construction towards white-box Q-table variation in DQN pathfinding training," in *Proc. IEEE 3rd Int. Conf. Data Sci. Cyberspace*, Jun. 2018, pp. 781–787.

[49] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, "Adversarial policies: Attacking deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2020. [Online]. Available: https://openreview.net/forum?id=HJgEMpVFwB

[50] C.-H. H. Yang *et al.*, "Enhanced adversarial strategically-timed attacks against deep reinforcement learning," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 3407–3411.

[51] P. Kiourti, K. Wardega, S. Jha, and W. Li, "TrojDRL: Trojan attacks on deep reinforcement learning agents," in *Proc. 57th ACM/IEEE Des. Autom. Conf.*, 2020, Mar. 2020.

[52] V. Behzadan and W. Hsu, "Sequential triggers for watermarking of deep reinforcement learning policies," 2019, *arXiv:1906.01126*.

[53] M. Usama, R. Mitra, I. Ilahi, J. Qadir, and M. Marina, "Examining machine learning for 5G and beyond through an adversarial lens," *IEEE Internet Comput.*, vol. 25, no. 2, pp. 26–34, Mar./Apr. 2021.

[54] Y. Han *et al.*, "Reinforcement learning for autonomous defence in software-defined networking," in *Proc. Int. Conf. Decis. Game Theory Secur.*, Aug. 2018, pp. 145–165.

[55] Y. Huang and Q. Zhu, "Deceptive reinforcement learning under adversarial manipulations on cost signals," in *Proc. Int. Conf. Decis. Game Theory Secur.*, 2019, pp. 217–237.

[56] A. Rakhsha, G. Radanovic, R. Devidze, X. Zhu, and A. Singla, "Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 7974–7984.

[57] X. Y. Lee, S. Ghadai, K. L. Tan, C. Hegde, and S. Sarkar, "Spatiotemporally constrained action space attacks on deep reinforcement learning agents," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 4577–4584.

[58] V. Behzadan and W. Hsu, "Adversarial exploitation of policy imitation," 2019, *arXiv:1906.01121*.

[59] M. Hessel *et al.*, "Rainbow: Combining improvements in deep reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018.

[60] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," in *Proc. Brit. Mach. Vis. Conf.*, Sep. 2017.

[61] K. Chen, S. Guo, T. Zhang, X. Xie, and Y. Liu, "Stealing deep reinforcement learning models for fun and profit," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2021, pp. 307–319.

[62] Z. Wang *et al.*, "Sample efficient actor-critic with experience replay," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[63] Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, and J. Ba, "Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5279–5288.

[64] M. Huai, J. Sun, R. Cai, L. Yao, and A. Zhang, "Malicious attacks against deep reinforcement learning interpretations," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 472–482.

[65] X. Y. Lee, Y. Esfandiari, K. L. Tan, and S. Sarkar, "Query-based targeted action-space adversarial policies on deep reinforcement learning agents," in *Proc. ACM/IEEE 12th Int. Conf. Cyber-Phys. Syst.*, 2021, pp. 87–97.

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.

[66] A. Ray, J. Achiam, and D. Amodei, "Benchmarking safe exploration in deep reinforcement learning," 2019, *arXiv:1910.01708*.

[67] V. Behzadan and A. Munir, "Whatever does not kill deep reinforcement learning, makes it stronger," 2017, *arXiv:1712.09344*.

[68] V. Behzadan and A. Munir, "Mitigation of policy manipulation attacks on deep Q-networks with parameter-space noise," in *Proc. Int. Conf. Comput. Saf., Rel., Secur.*, 2018, pp. 406–417.

[69] M. Fortunato *et al.*, "Noisy networks for exploration," in *Proc. Int. Conf. Learn. Represent.*, May 2018.

[70] V. Behzadan and W. Hsu, "Analysis and improvement of adversarial training in DQN agents with adversarially-guided exploration (AGE)," 2019, *arXiv:1906.01119*.

[71] K. L. Tan, Y. Esfandiari, X. Y. Lee, Aakanksha, S. Sarkar, "Robustifying reinforcement learning agents via action space adversarial training," in *Proc. Amer. Control Conf.*, 2020, pp. 3959–3964.

[72] E. Vinitsky, Y. Du, K. Parvate, K. Jang, P. Abbeel, and A. Bayen, "Robust reinforcement learning using adversarial populations," 2020, *arXiv:2008.01825*.

[73] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 2817–2826.

[74] M. A. Abdullah *et al.*, "Wasserstein robust reinforcement learning," 2019, *arXiv:1907.13196*.

[75] M. Bravo and P. Mertikopoulos, "On the robustness of learning in games with stochastically perturbed payoff observations," *Games Econ. Behav.*, vol. 103, pp. 41–66, May 2017.

[76] O. Ogunmolu, N. Gans, and T. Summers, "Minimax iterative dynamic game: Application to nonlinear robot control tasks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2018, pp. 6919–6925.

[77] A. Mandlekar, Y. Zhu, A. Garg, L. Fei-Fei, and S. Savarese, "Adversarially robust policy learning: Active construction of physically-plausible perturbations," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2017, pp. 3932–3939.

[78] J. Wang, Y. Liu, and B. Li, "Reinforcement learning with perturbed rewards," in *Proc. 34th AAAI Conf. Artif. Intell.*, Feb. 2020, pp. 6202–6209.

[79] E. Smirnova, E. Dohmatob, and J. Mary, "Distributionally robust reinforcement learning," in *Proc. Int. Conf. Mach. Learn. Workshop*, May 2019.

[80] C. Tessler, Y. Efroni, and S. Mannor, "Action robust reinforcement learning and applications in continuous control," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2019.

[81] A. Kumar *et al.*, "Enhancing performance of reinforcement learning models in the presence of noisy rewards," Ph.D. dissertation, Dept. Oper. Res. Ind. Eng., Univ. Texas at Austin, Austin, TX, USA, 2019.

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

[83] X. Pan, D. Seita, Y. Gao, and J. Canny, "Risk averse robust adversarial reinforcement learning," in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 8522–8528.

[84] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4026–4034.

[85] X. Wang, S. Nair, and M. Althoff, "Falsification-based robust adversarial reinforcement learning," in *Proc. 19th IEEE Int. Conf. Mach. Learn. Appl.*, 2020, pp. 205–212.

[86] B. Lütjens, M. Everett, and J. P. How, "Certified adversarial robustness for deep reinforcement learning," in *Proc. Conf. Robot Learn.*, 2020, pp. 1328–1337.

[87] H. Zhang *et al.*, "Robust deep reinforcement learning against adversarial perturbations on state observations," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 21024–21037.

[88] T. Oikarinen, T.-W. Weng, and L. Daniel, "Robust deep reinforcement learning through adversarial loss," 2020, *arXiv:2008.01976*.

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

[90] Y.-C. Lin, M.-Y. Liu, M. Sun, and J.-B. Huang, "Detecting adversarial attacks on neural network policies with visual foresight," 2017, *arXiv:1710.00814*.

[91] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[92] A. Havens, Z. Jiang, and S. Sarkar, "Online robust policy learning in the presence of unknown adversaries," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Dec. 2018, pp. 9916–9926.

[93] Y. Xiang, W. Niu, J. Liu, T. Chen, and Z. Han, "A PCA-based model to predict adversarial examples on Q-learning of path finding," in *Proc. IEEE 3rd Int. Conf. Data Sci. Cyberspace*, Jun. 2018, pp. 773–780.

[94] V. Gallego, R. Naveiro, and D. R. Insua, "Reinforcement learning under threats," in *Proc. AAAI Conf. Artif. Intell.*, Jul. 2019,vol. 33, pp. 9939–9940.

[95] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Secur. Privacy*, May 2016, pp. 582–597.

[96] N. Carlini and D. Wagner, "Defensive distillation is not robust to adversarial examples," 2016, *arXiv:1607.04311*.

[97] A. A. Rusu *et al.*, "Policy distillation," in *Proc. Int. Conf. Learn. Represent.*, May 2016.

[98] W. M. Czarnecki, R. Pascanu, S. Osindero, S. Jayakumar, G. Swirszcz, and M. Jaderberg, "Distilling policy distillation," in *Proc. Mach. Learn. Res.*, Apr. 2019, pp. 1331–1340.

[99] X. Qu, Y.-S. Ong, A. Gupta, and Z. Sun, "Defending adversarial attacks without adversarial attacks in deep reinforcement learning," 2020, *arXiv:2008.06199*.

[100] V. Behzadan and A. Munir, "Adversarial reinforcement learning framework for benchmarking collision avoidance mechanisms in autonomous vehicles," *IEEE Intell. Transp. Syst. Mag.*, vol. 13, no. 2, pp. 236–241, 2021.

[101] V. Behzadan and W. Hsu, "RL-based method for benchmarking the adversarial resilience and robustness of deep reinforcement learning policies," in *Proc. Int. Conf. Comput. Safety, Reliability, Secur.*, 2019, pp. 314–325.

[102] G. Brockman *et al.*, "OpenAI gym," 2016, *arXiv:1606.01540*.

[103] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[104] P. Dhariwal *et al.*, "OpenAI Baselines," 2017. [Online]. Available: https://github.com/openai/baselines

[105] I. Caspi, G. Leibovich, G. Novik, and S. Endrawis, "Reinforcement learning coach," Dec. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1134899

[106] J. Gauci *et al.*, "Horizon: Facebook's open source applied reinforcement learning platform," 2019, *arXiv:1811.00260*.

[107] ReAgent, Accessed: Jan. 21, 2021. [Online]. Available: https://reagent.ai/

[108] P. Gawłowicz and A. Zubow, "Ns-3 meets OpenAI gym: The playground for machine learning in networking research," in *Proc. ACM Int. Conf. Model., Anal. Simul. Wireless Mobile Syst.*, 2019, pp. 113–120.

[109] Papernot *et al.*, "Technical report on the CleverHans v2.1.0 adversarial examples library," 2018, *arXiv:1610.00768*.

[110] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 86–94.

[111] E. Puiutta and E. M. Veith, "Explainable reinforcement learning: A survey," in *Proc. Int. Cross-Domain Conf. Mach. Learn. Knowl. Extraction*, 2020, pp. 77–95.

[112] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Explainability in deep reinforcement learning," *Knowl.-Based Syst.*, vol. 214, 2021, Art. no. 106685.

[113] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Proc. Int. Conf. Artif. Neural Netw.*, 2018, pp. 270–279.

[114] Z. Zhu, K. Lin, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," 2020, *arXiv:2009.07888*.

Authorized licensed use limited to: National University Fast. Downloaded on April 30,2024 at 06:03:09 UTC from IEEE Xplore. Restrictions apply.