# Adversarial Examples in Reinforcement Learning

**Clément Acher**
clement.acher@mines-paristech.fr

**Hugo Cisneros**
hugo.cisneros@ens-paris-saclay.fr

## Abstract

Adversarial examples have become a critical weakness of many, always increasingly popular deep learning models. They are one of the major limitations to their widespread adoption in production environments and for security applications but also raise issues about the robustness of these algorithms. Reinforcement learning (RL) is increasingly relying on deep models that have recently achieved some spectacular breakthroughs, but are vulnerable to the same attacks. Research into the possible attacks and defense mechanisms applied to these models is key to making them reliable, robust and able to function in production. In this paper, we review the methods for crafting adversarial examples that can fool deep models and RL models into behaving the wrong way. Thereafter, we review some defense and mitigation techniques against these adversarial examples and the perspectives for safe RL.

## 1 Introduction

Adversarial examples are inputs to a machine learning model crafted in a such a way that they result in an incorrect or unexpected output from the model. This output can be a label from a classification model, a scalar value or an action or policy in the case of RL models.

These examples pose concerning security issues for models that are vulnerable to them and a lot of recent research has gone into designing more efficient attack and defense algorithms to get a better understanding of the flaws of these machine learning models and the ways to fix them. Apart from the security issues, one of the main problem that adversarial examples pose is about the robustness of the model and their ability to generalize and abstract from the data they learned from. They clearly demonstrate that our current models are still far from that point. By trying to build machine learning models that are resistant to these attacks and more robust to adversaries we might however make some advances in that direction.

The contributions of this paper are as follow: we first investigate some characteristics of adversarial examples for deep learning models and lay out the foundations and main methods for adversarial example crafting. Next, we study how adversarial examples can be used in deep RL settings, the specific challenges that come with them and finally examine some research works that aim at designing defense mechanisms for RL agents.

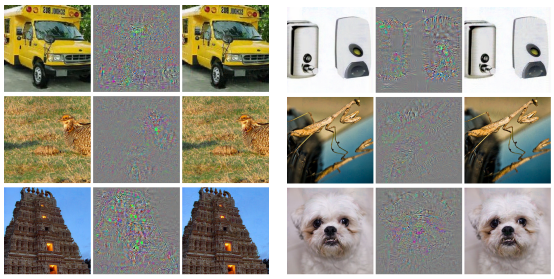
## 2 Adversarial examples in Deep Learning

The idea of adversarial examples has been around in conventional machine learning for over a decade. Slightly changing the content of an email to fool handcrafted spam filters is a well-known example [1].

In the context of the growing popularity of deep learning models, the question of whether or not this kind of model could be fooled became an important field of research. Szegedy et al. confirmed in [2] what was feared prior: they made the discovery that these models, including the state-of-the-art

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

neural networks, are vulnerable to adversarial examples. A hardly perceptible perturbation applied to a correctly classified example can cause the network to misclassify the crafted example.



(a) (b)

Figure 1: Adversarial examples for AlexNet taken from [2] : Adversarial perturbations (center - magnified by 10) added to correctly predicted images on the left fool a deep neural network into classifying all images shown on the right side as *"ostrich, Struhio, camelus"*.

If we take the concrete example of crafting an adversarial example in an image classification task, it could be described as follow: adversarial images are original image where an imperceptible perturbation has been added that will misguide the classifier. Denoting $f$ the model, $x$ the original image and $x'$ the adversarial example, finding the best adversarial images could correspond to the following box-constrained optimization problem:

$$\min_{\boldsymbol{x'}} \quad ||x' - x|| \quad \text{s.t.} \quad f(x') = l', \; f(x) = l, \; l \neq l'$$

But this problem can be formulated in very different ways : is the targeted model known by the attacker ? Does the adversarial example target a specific class ?, etc. Yuan and al. in [3] proposed a taxonomy of the various methods for generating adversarial examples.

### 2.1 Some elements of the taxonomy of adversarial examples

We will introduce in this subsection some scenarios and assumptions of threat models. This list is not exhaustive and only contains parts that are also relevant for deep reinforcement learning. See [3] for the complete taxonomy.

**Adversary's Knowledge**

- *White-box* attacks : the attacker has access to all the information related to the trained model (model weights, model architecture...). [2] and [4] are examples of such attacks.
- *Black-box* attacks : the adversary has no access to the trained model. Most of the time, such attacks are base on the *transferability* of adversarial examples proposed by Papernot et al. in [5] : an adversarial example crafted for a given model is likely to be misclassified by another model trained to perform the same task.

**Adversarial Specificity**

- *Targeted* attacks : the goal is to have the adversarial example classified by the trained model as a specific class.
- *Non-targeted* attacks : the goal is simply to have the trained model to output something else than the correct class.

### 2.2 Crafting an adversarial example : the Fast Gradient Sign Method

There are nowadays quite a few techniques to generate adversarial examples. The Fast Gradient Sign Method (FGSM) introduced in [6] is quite efficient and effective. This method is used in the context of computer vision classification but also for adversarial attacks on neural network policies. The idea behind this attack is to make a linear approximation of the model and solve analytically the problem of getting the optimal adversarial perturbation.

Given a linear function $g(x) = w^T x$, the optimal perturbation $\eta$ that satisfies $||\eta||_\infty \leq \epsilon$ is

$$\eta = \epsilon \; \text{sign}(w)$$

The change of output is maximized, and for the adversarial example $\tilde{x}$, we have $g(\tilde{x}) = w^T x + w^T \eta$.

Given a classification network with loss $J(\theta, x, y)$ where $x$ is the input, $y$ a distribution over all possible class labels and $\theta$ the parameters of the model, FGSM makes the assumption that $J$ is linear around the input $x$. The result of the previous paragraph then gives the optimal perturbation :

$$\eta = \epsilon \operatorname{sign}(\nabla_x J(\theta, x, y))$$

The Jacobian-based Saliency Map Attack (JSMA) introduced in [7] is another family of attack methods used to perform *targeted* attacks. By saturating a few pixels in an image, the attack can cause the model to misclassify the adversarial example as the erroneous target class. We won't introduce it here, but this kind of attack is used in both supervised learning and DRL.

### 2.3 Some highlights

We can mention a few impressive applications of adversarial examples. This list is not exhaustive but highlights the risks of using DL methods in production environments.

**Hidden voice command:** Carlini et al. show in [8] that an attacker can produce voice commands that are non-understandable by humans but understood by devices using speech-recognition in both whitebox and blackbox settings.

**Stop signs:** [9] introduces a new attack algorithm called **Robust Physical Perturbation** that can craft robust visual adversarial examples under different physical conditions. They show that the produced perturbations can be physically applied to road signs to fool classifiers under various viewpoints.

**Hiding pedestrians in a segmentation task:** Metzen et al. in [10] propose an adversarial algorithm to perform targeted attacks against image segmentation task. They successfully remove pedestrians from the output of a FCN-8s model trained on the Cityscape dataset.

**3D printed adversarial examples:** [11] demonstrates the possibility of physical adversarial objects : they produce a 3D-printed turtle recognized as a rifle by a trained ImageNet classifier under almost every viewpoint.

## 3 Adversarial examples in Deep Reinforcement Learning

In the case of reinforcement learning, examples of the desired behaviors are not provided, but it is possible to score a behavior thanks to a reward function. [12] has shown that it was possible to convert supervised learning task to reinforcement learning task, but the other way around was not possible. One consequence of this is that reinforcement learning tasks should theoretically be vulnerable against the adversarial example frameworks introduced earlier, but could actually have even more weaknesses. However, the study of adversarial examples in Reinforcement Learning is pretty new : the first paper written about it, [13], was published in early 2017.

Since, a few papers have been published about this subject : we will first review the techniques aiming to fool DRL methods, i.e. the *attacks*, then talk about the mechanisms to counter adversarial examples.

### 3.1 Background : Deep Reinforcement Learning

Over the past few years, Deep Reinforcement Learning have seen a surge in interest following a set of milestones that have demonstrated the potential of the technology. [14] has introduced deep Q-network agents (DQN), that use deep convolutional neural networks to approximate the optimal action value function

$$Q^*(s, a) = \max_{\pi} \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i}; s_t = s, a_t = a, \pi\right]$$

DQN achieves super-human performance on a wide range of Atari games and outperformed most of the best existing methods in the literature at the time. Other deep RL methods have also been developed since and we list here some of them:

**Deep deterministic policy gradient (DDPG)** was introduced in [15]. It is an algorithm based on the deterministic policy gradient [16] that uses an actor-critic approach and the use of a deep neural network to approximate functions in large state and action spaces.

**Asynchronous advantage actor-critic (A3C)** [17]. This algorithm uses a neural network that estimates both the policy and value function at every step. Updates to the network's weights are computed from an estimation of the advantage function for a given policy.

**Double DQN** [18] is based on a modification of DQN to take advantage of Double Q-learning [19]. It is able to mitigate some estimation errors of DQN.

### 3.2 Attacks

#### 3.2.1 Non-targeted attacks

[13] is the first paper trying to target a DRL algorithm using adversarial techniques. Behzadab et al. focuse on pointing out the vulnerabilities of DQN models to already existing techniques, namely the FGSM and JSMA methods. They follow the simple hypothesis that as DL methods are vulnerable to adversarial examples and that DQN is based on deep neural networks, DQN should be vulnerable to the previously introduced attacks. In the scenario described, the attacker can only interfere and manipulate the configuration of the environment. It acts as in a man-in-the-middle attack : it injects an adversarial perturbation $\delta_t$ to the real state $s_t$. The perturbed input $s_t + \delta_t$ is then revealed to the target. They consider the attack successful if the action picked by the attacked model is different from the current optimal action.

They show that such attack can be performed in both whitebox and blackbox settings. In the whitebox setting, they assume that the attacker knows the $Q$ value function of the target. On the Atari game Pong and using FGSM or JSMA attacks to create a perturbation $\delta_t$, Behzadab et al. show that the attack will almost surely be successful, i.e. $\max_{a'} Q(s_{t+1} + \delta_{t+1}, a') \neq \max_{a'} Q(s_{t+1}, a')$.

To try this out in the a blackbox scenario, they use the same technique as [5], using the transferability of adversarial examples: the attack is this time crafted on a DQN that has been trained in the same conditions as the targeted DQN, and then used to attack the targeted DQN. The attack is successful about 80% of the time on the game Pong.

Shortly after, Huang et al. conducted in [20] further experiments targeting this time three DRL state-of-the-art models (DQN, TRPO and A3C) with a special focus on the amplitude of the perturbation allowed to the adversarial. Adversarial examples in order to be considered as proof of potential security flaws of the models must be hardly perceptible by humans. The norm of the perturbation is then either bounded or minimized. The choice of the norm is quite important : in an environment based on images, using a constraint on the $l_1$-norm allows changes on only a small subset of pixels, while the choice of the $l_\infty$-norm yields hardly perceptible perturbations. The three algorithms are tested on four Atari games in both whitebox and blackbox scenarios.

In the whitebox setting, their experimental results show that DQN tends to be the most vulnerable model out of the three. That said, on all three models, slight perturbation using $l_\infty$ norm is enough to decrease the agent performance by 50% or more. The attack is even more effective using the $l_2$ or $l_1$ norms.

To perform attacks in a blackbox scenario, Huang et al. also use the transferability properties defined by Papernot et al. in [5], but test this time two different setups. In the first case, the attacker knows the architecture of the model, the training algorithm and the various hyperparameters, but doesn't know the random initialization. The same model can then be trained. In the second case, the attacker doesn't know anything. As expected, the more the attacker knows, the more effective the attacks are. However, transferability seems to apply in reinforcement learning as the blackbox attacks significantly degrade the performance of the trained agent, especially when using the $l_1$-norm, with a few exceptions like the game Seaquest with the $l_2$ and $l_\infty$ norms.

The following attacks will assume a whitebox settings - the authors claiming that the results could also be achieved in a blackbox setting using [20]'s results.

In [21], Pattanaik et al. highlight the fact that contrary to image classification wherein attack is considered successful if a given image is classified as any other image and hence there is no concept of "worst possible image", RL agents can take the "worst possible choice" leading to the smallest

reward. Adversarial examples should then aim to cause the trained RL agent to follow this worst possible choice, i.e. the action which corresponds to the least Q value in the current state. Instead the loss function used by previously published papers such as [20] minimizes the probability of taking the best possible action which doesn't necessarily lead to increase in probability of taking worst possible action.

Defining the adversarial probability distribution $P$ as

$$P\left(a_i\right) = \begin{cases} 1, & \text{if } a_i = a_w \\ 0, & \text{otherwise} \end{cases}$$

Where $a_w$ is the worst possible action, the paper uses the cross entropy loss between the adversarial probability distribution $P$ and optimal policy generated by the RL agent given by conditional probability mass function $\pi^*(a|s)$ as the loss function to minimize to craft the example maximizing the probability of taking the worst possible action. The loss function is then :

$$J(s, \pi^*) = -\sum_{i=1}^{n} P(a_i) \log \pi_i^* = -\log \pi_w^*$$

Using this loss combined with gradient descent algorithms, Pattanaik et al. outperforms the results of Huang et al. in [20].

As mentioned before, an adversarial has to apply minimal perturbations in order to be considered relevant. Papers introduced before mainly focused on bounding the perturbation applied to the input of the model as done in supervised learning. Lin et al. in [22], using the specificity of RL in which observations are correlated, introduces the idea that attacks should be done at selective time steps. While playing Pong, it is pretty clear that attacking the agent while the ball is far is not very effective, attacking at time when the ball is close can be enough to have the agent missing the ball. Following this idea, Lin et al. shows that strategically-timed attacks can be as effective as uniform attacks while being harder to detect.

The question of when to attack is answered by using a relative action preference function $c$, defined by :

$$c\left(s_t\right) = \max_{a_t} \pi\left(s_t, a_t\right) - \min_{a_t} \pi\left(s_t, a_t\right)$$

$c$ represents the preference of the agent taking the most preferred action over the least preferred one. If this value is low, this means that there is no real benefits taking an action over another (the ball is far away from the paddle) and there is no point attacking at this point. When this value is large, the agent strongly prefers the most preferred action (the ball is close from the paddle). If $c$ is greater than a threshold, an attack is performed. The choice of this threshold controls how often attacks are performed.

Regarding the attack itself, the authors, arguing that common methods such as FGSM or the Jacobian-based saliency map attack could be countered as shown in [23] and [6], have decided to rather use an approach introduced in [24], namely optimizing with early stopping

$$\begin{aligned} \min_{\boldsymbol{\delta}} \quad & ||\delta||_\infty \\ \text{s.t.} \quad & f(x) \neq f(x+\delta) \end{aligned}$$

Using this setting, they show that they were able to achieve similar results as [20] that uses uniform attacks while only attacking 25% of the time steps in an episode in 5 Atari games.

### 3.2.2 Targeted attacks

Another interesting idea is introduced in [22] : similarly to what we have called *targeted* attacks in the previous section (the goal is to have the adversarial example classified by the trained model as a specific class), an adversarial example in RL could maliciously lure the agent toward a target state. Lin et al. called this an *enchanting attack*.

More precisely, the goal of such attack is to get the agent from current state $s_t$ at time step $t$ to a specified target state $s_g$ after $H$ steps. Such attack is achieved in two steps. First, the attacker plans the sequence of actions $A_{t:t+H} = \{a_t, \ldots, a_{t+H}\}$ needed to go from state $s_t$ to state $s_g$. To define this sequence, the model uses a video prediction model $M$ based on [25] to predict the future state

after $H$ steps given a sequence of action : $s^M_{t+H} = M(s_t, A_{t:t+H})$. Successive samplings are then used to find the sequence $A^*_{t:t+H}$ to get as close as possible to $s_g$ after $H$ steps. The second part is then to craft adversarial examples to have the trained RL agent to follow the sequence $A^*_{t:t+H}$. This is done following [24] which corresponds to the minimization problem of getting the perturbation as small as possible while being as close as possible to the targeted action.

Using this *enchanting attack*, Lin et al. successfully lure deep RL agents trained with A3C and DQN towards maliciously defined target states in 40 steps with more than 70% success rate in 3 out of 5 games. Authors explain the failure on the two other games, Seaquest and ChopperCommand, due to random enemies which make the video prediction models perform poorly.

Following this, Tretschk et al. proposed in [26] a similar attack to the *enchanting* mechanism of [22], with the difference that, instead of an adversarial state $s_g$, their adversary imposes an adversarial reward $r^A$. Note that it can technically encode the enchanting attack. The adversarial examples are crafted thanks to a feedforward deep neural network $g_\theta$. This technique is called Adversarial Transformer Network (ATN) and is introduced in [27]. The goal of this neural network, is, given a state $x$, output a perturbation $g_\theta(x)$ such that the perturbed inputs $x + g_\theta(x)$ makes the agent pursue the adversarial reward $r^A$. Given a fixed victim policy network $Q$, the parameter $\theta$ of the ATN is learned by training $x \mapsto Q(x + g_\theta(x))$ as a DQN using the adversarial reward $r^A$. In order to train this as a DQN, this approach requires access to both the gradient of the policy network (whitebox assumption) and the training environment.

This technique is used against agents trained to play the Pong Atari game, with an adversarial reward $r^A$ rewarding the agent if the ball hits the centre 20% of the enemy line. Using the $r^A$ reward, the attacked agents obtain a similar cumulative reward as agents that have been directly trained on the $r^A$ reward.

#### 3.2.3 Attacks during training time

To the best of our knowledge, there is only one paper that has studied attacks during training time, namely the first paper published about adversarial examples in DRL [13] by Behzadan and Munir.

They perform an attack against a DQN model playing the game of Pong in a blackbox setting. Their experiment uses 3 DQNs :

**DQN T** which is the model that will be attacked during training

**DQN C** which is trained the same way as DQN T. As this attack is performed in a blackbox setting, this model will be used to craft the adversarial example that will then be applied on DQN T using the transferability property.

**DQN I** which is trained in the same environment as the other DQN, but whose reward value is the negative of the value obtained from the from the target DQN T's reward function. The DQN I is used to find the "worst" action as it is trained with the exact opposite of the game score.

At each step of the training of DQN T, DQN I gives the worst action, an adversarial state is crafted using a JSMA attack to follow this worst possible action on DQN C and then passed to DQN T. The results show that the attack is pretty effective : the reward of the attacked model, DQN T, gets close to 0. However no comparison is done with a random noise added during the training or to induce another policy than just getting a bad score.

### 3.3 Defense

The success of adversarial attacks on deep RL models has raised major questions about the way these models are trained and the possibility of making them robust to adversarial inputs. Several approaches to solving this issue are investigated, some addressing the general problem of adversarial examples and some other specific to RL and the process of training RL models.

### 3.3.1 Adversarial training

Adversarial training, an approach consisting in training the model in an environment where adversarial states are voluntarily injected, have concentrated a lot of research efforts recently. Those methods allow for learning robust policies under an adversarial learning process.

Adversarial training was introduced some years back with Morimoto et al.'s Robust RL (RRL) [28]. They have proposed to use a disturbing agent that tries to apply the worst possible disturbance on a pendulum system while a control agent attempts to counteract its effect, with a reward function that encourages it to do so. Pinto et al. [29] extended RRL to deep RL by using TRPO and neural networks as policy function approximators for both the first agent and the adversary. They thereby define an adversarial game where two players observe at every time step $t$ the state $s_t$ and take actions $a_t^1 \sim \mu(s_t)$ (agent) and $a_t^2 \sim \nu(s_t)$ (adversary). The transitions and rewards are defined from both actions with $s_{t+1} = \mathcal{P}(s_t, a_t^1, a_t^2)$ and $r_t = r(s_t, a_t^1, a_t^2)$. The reward of the agent trying to learn the main policy receives the reward $r_t^1 = r_t$ while the adversary receives $r_t^2 = -r_t$. This defines a MDP with the following representation at time $t$: $(s_t, a_t^1, a_t^2, r_t^1, r_t^2, s_{t+1})$. The objective to be maximized for both agents is the cumulated reward, for the protagonist it is written

$$R^1(\mu, \nu) = \mathbb{E}_{s_0 \sim \rho, a^1 \sim \mu(s), a^2 \sim \nu(s)} \left[ \sum_{t=0}^{T-1} r^1(s, a^1, a^2) \right]$$

While the adversary maximizes $R^2(\mu, \nu) = -R^1(\mu, \nu)$. The solution to the problem amounts to a minmax equilibrium with

$$R^{1*} = \min_{\nu} \max_{\mu} R^1(\mu, \nu)$$

According to Pattanaik et al. [21], because of the min-max game theoretic formulation of the problem in the settings presented above, the equilibrium is usually hard to find, and either the agent is unable to learn a policy against an unreasonably strong adversary or the opposite happens. They propose an other approach that consists in making the adversary "fool" the agent into believing it is in a state different from its current state and leading it to choose the worst possible action. They initially train the agent in a regular noise-free environment and retrain it with the adversary added. This method appears to lead to a better quality training and more robust performances compared to agents trained with regular DDQN and DDPG.

EPOpt is an algorithm presented in [30]. It samples the MDP model parameters at random for several rollouts, and uses ensemble methods to learn a policy that maximizes return from the worst performing $\varepsilon$-percentile trajectories. This idea, related to percentile optimization [31] where $\varepsilon$-percentile values are directly optimized, generalizes learning a policy to an ensemble of MDPs with parameters from a source distribution. With this idea, the authors manage to train very robust models on OpenAI gym's tasks, which notably achieve very consistent performance across a range of parameters for the simulations. However, since the sampling of MDP parameters relies on the discretization of the model parameter space, it quickly becomes intractable for more elaborate situations where this space is of very high dimension.

### 3.3.2 Predictive defense

In [22], Lin et al. have taken a radically different approach at defense against adversarial attacks by adding a mechanism for detecting attacks within the agent itself. They also let the agent recover from attacks by predicting what the state should have been from the previously taken action.

They propose a setting where the agent observes either a clean state $s_t^{\text{normal}}$ or a perturbed observation $s_t^{\text{adv}}$. At time step $t$, a frame prediction model, inspired from [25], takes the $m$ previous states and corresponding $m$ actions and predicts the current state $\hat{s}_t$. Two cases may happen

- The observed state is normal, $s_t = s_t^{\text{normal}}$, and the action distribution $\pi(s_t)$ should be very similar to the action distribution of the predicted state $\pi(\hat{s}_t)$. The agent does not have to defend itself.
- The observed state was maliciously perturbed, $s_t = s_t^{\text{adv}}$. The goal of the adversary being to cause the agent to take an action he shouldn't have taken, the distribution $\pi(\hat{s}_t)$ will likely be very different from $\pi(s_t)$. In that case, by measuring a distance between those to

distributions, an agent can detect that it is being attacked and can devise a way to counteract this attack.

More specifically, Lin et al. check if a distance between the two distributions $D(\pi(\hat{s}_t), \pi(s_t))$ is above a fixed threshold $H$. If so, the agent is being attacked and the distribution $\pi(\hat{s}_t)$ can be used to choose an action that would have likely been chosen if the observed state had been normal. The effect of the attack is thereby mitigated.

The authors evaluate the efficiency of the method against three adversarial examples generation algorithms: FGSM [6], BIM [32] and Carlini et al's [24]. It is able to accurately predict when an attack happens and recovers quite well from the effects of those attacks on 5 Atari games.

#### 3.3.3 Meta-learning as a defense against adversarial examples

Havens et al. [33] adopt an approach similar to EPOpt [30] using a parameterized MDP, and frame the problem of learning under adversarial perturbations as a meta-learning problem. To learn under these conditions, Havens et al. introduce the Meta-Learned Advantage Hierarchy (MLAH) algorithm. The authors relate the method to a human choosing and switching skills depending on the task he is performing. Here, two separate sub-policies $\pi_{adv}$ and $\pi_{normal}$ are created and a master policy $\pi_{master}$ is trained to detect the change in state-reward map that an attack induces and choose a sub-policy that is most adequate with the environment. The underlying changes in task that occur during an attack are measured using the *advantage* of a given state-action pair. It is defined as the difference between the expected return in a state $s_t$ compared to the observed return obtained by choosing an action $a_t$ according to the current policy.

This method can be generalized to an arbitrary number of sub-policies, that could for instance correspond to multiple attack types. One major advantage of this method is also the fact that sub-policies are not interacting with one another (assuming that the master policy is good enough) and although the adversarial policy might be inefficient for counteracting attacks, the algorithm would still perform well under normal conditions as opposed to a unified policy.

The authors notably analyze a case where an deterministic adversarial attack periodically inverts the column position of the agent in a GridWorld environment for a fixed number of iterations. A regular policy learned with PPO [34] is not able to learn a unified policy for these two situations since it would need to learn two opposite behaviors. However, after being trained in an non adversarial environment, MLAH is able to progressively adapt to these attacks and jointly learn an adequate adversarial policy together with a master policy.

#### 3.3.4 Parameter-space noise exploration

Parameter-space noise exploration is based on the idea that iteratively and adaptively adding noise to the parameters of deep RL architectures greatly enhances exploration, convergence speed during training and robustness of learned policies. Behzadan et al. proposes in [35] this parameter-space noise exploration as a mitigation technique for policy manipulation attacks at both test and training time. The authors compare the effectiveness of NoisyNet, a neural network with iteratively perturbed biases and weights introduced by [36] with that of standard DQN setup. They study 3 Atari games, and FGSM as the algorithm for crafting adversarial examples.

NoisyNet is able to achieve significant resilience to training-time attacks compared to the original DQN, thereby demonstrating a mitigation of the effects of training-time attacks. The authors also study the efficiency of parameter-space noise on test-time attacks, that is the general ability of NoisyNet to learn policies that are robust to attacks from a normal environment when put to test in an adversarial environment. The methods seems to be more efficient against blackbox attacks, for which it consistently get a much higher mean reward than the original DQN, whereas it is less clear if the improvement is significant for whitebox attacks. The authors argue that the improvement that NoisyNet enables is due to its enhanced generalization abilities and reduced transferability.

## 4 Discussion

This paper has presented a taxonomy of adversarial example crafting algorithms and the way they can be used against deep RL models. It has also presented ways of dealing with them and attempts

at making models capable of detecting and continue working the way they were supposed to after an attack. This field is however still very recent and most research problems remain open.

For example, most of the work reviewed in this paper focuses on attacks that target the observation that an agent makes of the environment. But there is yet to be a thorough analysis of attacks targeting the reward function of a RL task although adversarial example theoretically apply. Awareness about this particular issue has already been raised in [37], but RL models with potential real-world applications would depend on complex environment interactions, reward functions and action-state space that are even more ways it can be susceptible to adversarial examples.

Research efforts into making RL agents resilient to adversarial examples and thereby making them more robust are essential. Adversarial training is for instance an effective method, but it is inherently limited by the simulation capabilities for training the agent which is not likely to be entirely available as we are just starting to discover attack mechanisms. Some interesting research leads might come from the combination of above mentioned defense procedures. For instance meta-learning can be combined with both noisy exploration and adversarial training to refine and make learned policies more robust. If machine learning models, and especially RL models, are to be widely used in real-world applications, having these problems fixed will be critical.

## References


[1] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining - KDD '05*, page 641, Chicago, Illinois, USA, 2005. ACM Press.

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

[3] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial Examples: Attacks and Defenses for Deep Learning. *arXiv:1712.07107 [cs, stat]*, December 2017. arXiv: 1712.07107.

[4] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv:1607.02533 [cs, stat]*, July 2016. arXiv: 1607.02533.

[5] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks against Machine Learning. *arXiv:1602.02697 [cs]*, February 2016. arXiv: 1602.02697.

[6] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv:1412.6572 [cs, stat]*, December 2014. arXiv: 1412.6572.

[7] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The Limitations of Deep Learning in Adversarial Settings. *arXiv:1511.07528 [cs, stat]*, November 2015. arXiv: 1511.07528.

[8] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 513–530, Austin, TX, 2016. USENIX Association.

[9] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust Physical-World Attacks on Deep Learning Models. *arXiv:1707.08945 [cs]*, July 2017. arXiv: 1707.08945.

[10] Jan Hendrik Metzen, Mummadi Chaithanya Kumar, Thomas Brox, and Volker Fischer. Universal Adversarial Perturbations Against Semantic Image Segmentation. *arXiv:1704.05712 [cs, stat]*, April 2017. arXiv: 1704.05712.

[11] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing Robust Adversarial Examples. *arXiv:1707.07397 [cs]*, July 2017. arXiv: 1707.07397.

[12] Reinforcement Learning and Its Relationship to Supervised Learning. In *Handbook of Learning and Approximate Dynamic Programming*. IEEE, 2009.

[13] Vahid Behzadan and Arslan Munir. Vulnerability of Deep Reinforcement Learning to Policy Induction Attacks. *arXiv:1701.04143 [cs]*, January 2017. arXiv: 1701.04143.

[14] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. 518(7540):529–533.

[15] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning.

[16] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, pages 387–395.

[17] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning.

[18] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning.

[19] Hado V. Hasselt. Double q-learning. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2613–2621. Curran Associates, Inc.

[20] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial Attacks on Neural Network Policies. *arXiv:1702.02284 [cs, stat]*, February 2017. arXiv: 1702.02284.

[21] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust Deep Reinforcement Learning with Adversarial Attacks. *arXiv:1712.03632 [cs]*, December 2017. arXiv: 1712.03632.

[22] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of Adversarial Attack on Deep Reinforcement Learning Agents. *arXiv:1703.06748 [cs, stat]*, March 2017. arXiv: 1703.06748.

[23] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks. *arXiv:1511.04508 [cs, stat]*, November 2015. arXiv: 1511.04508.

[24] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. *arXiv:1608.04644 [cs]*, August 2016. arXiv: 1608.04644.

[25] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard Lewis, and Satinder Singh. Action-Conditional Video Prediction using Deep Networks in Atari Games. *arXiv:1507.08750 [cs]*, July 2015. arXiv: 1507.08750.

[26] Edgar Tretschk, Seong Joon Oh, and Mario Fritz. Sequential Attacks on Agents for Long-Term Adversarial Goals. *arXiv:1805.12487 [cs, stat]*, May 2018. arXiv: 1805.12487.

[27] Shumeet Baluja and Ian Fischer. Learning to Attack: Adversarial Transformation Networks. In *Proceedings of AAAI-2018*, 2018.

[28] Jun Morimoto and Kenji Doya. Robust reinforcement learning. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 1061–1067. MIT Press.

[29] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. page 10.

[30] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. EPOpt: Learning robust neural network policies using model ensembles.

[31] Erick Delage and Shie Mannor. Percentile optimization for markov decision processes with parameter uncertainty. 58(1):203–213.

[32] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale.

[33] Aaron Havens, Zhanhong Jiang, and Soumik Sarkar. Online robust policy learning in the presence of unknown adversaries. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9938–9948. Curran Associates, Inc.

[34] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms.

[35] Vahid Behzadan and Arslan Munir. Mitigation of policy manipulation attacks on deep q-networks with parameter-space noise.

[36] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration.

[37] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety.