Manipulating Neural Policies with Adversarial Observations

Léonard Hussenot [1] **Matthieu Geist** [1] **Olivier Pietquin** [1]

Abstract

This paper deals with adversarial attacks on neural policy perceptions in a reinforcement learning (RL) context. Classic approaches perform untargeted attacks on the state of the agent. Here, we argue that it is more realistic to attack the observations provided by the environment rather than the internal state computed by the agent. We propose an untargeted attack over observations and show that its effectiveness even holds when reduced to a constant attack over observations. We also propose an approach to perform targeted attacks on the observations of the agent, so that it acts as told by an opponent policy. We illustrate this on deep RL agents playing Space Invaders.

1. Introduction

Neural networks classifiers have been shown to be sensitive to adversarial examples (Goodfellow et al., 2015; Carlini & Wagner, 2017). These adversarial examples, whose existence was highlighted by Szegedy et al. (2013), have been successfully applied in real-world situations (Athalye et al., 2017; Brown et al., 2017), by either adding an imperceptible noise or a reasonable-sized patch on an image. These attacks are able to lure classifiers into predicting wrong labels for an image initially correctly classified by attacking the raw input image. In reinforcement learning (RL), end-to-end systems have been trained to map visual inputs to actions, successfully solving games like Atari (Mnih et al., 2015; Hessel et al., 2018; Bellemare et al., 2013) or training self-driving car policies (Bojarski et al., 2016). However, the use of deep networks for visual input understanding in end-to-end systems may lead these to face high sensitivity to adversarial examples. Indeed, if an imperceptible perturbation can mislead a classifier, then an agent, *i.e.* a policy mapping images to actions, can be deceived the same way.

Huang et al. (2017) introduced adversarial examples in RL, yet leaving aside its intrinsically dynamic nature. Indeed, while supervised learning literature divides attacks into two categories, such a dichotomy is not as that clear in the RL context. Dividing the possible scenarios into *white-box* and *black-box*, *i.e.*, having or not having access to the learning algorithm and its parameters is, here, not that easy. Especially, previous work has focused on attacking the agent's state. We argue that doing so means modifying the internal representations of the agent and that it is therefore more realistic to attack only the observations given by the environment, even in the *white-box* setting. Willing to design a minimal attack, Lin et al. (2017) proposed a heuristic to reduce the number of attacked states and successfully reduced this amount by four. We propose a single constant-attack to be applied on all observations.

Moreover, the literature focused on untargeted or specialized attacks (*e.g.* bringing the agent to a particular state). We propose a targeted attack on the agent's perception to spur it to act as told by an opponent's policy.

2. Preliminaries

In **reinforcement learning**, an agent interacts with an environment. Given a state space $\mathcal{S}$ and an action space $\mathcal{A}$, its (possibly stochastic) policy π is trained to maximize the agent cumulative discounted reward over time. Value-based algorithms (Mnih et al., 2015; Hessel et al., 2018) use the value-function, or more frequently the quality-function $Q(s, a)$ to approximate the expected cumulative discounted reward starting from state s and playing action a. Q is then used to compute π. Deep RL (DRL) uses deep neural networks for function approximation. In value-based DRL, the quality function Q_ω is parametrized with a neural network of parameters ω, mapping continuous states to actions.

Adversarial examples have been introduced in the context of supervised classification. Given a classifier C, an input x, a bound ϵ on a norm $\|.\|$, an adversarial example is a input $x' = x + \eta$ such that $C(x) \neq C(x')$ while $\|x - x'\| \leq \epsilon$. *Fast gradient sign method* is the most widespread method for generating adversarial examples for the L_∞-norm. From a linear approximation of C, it computes the attack η as:

$$\eta = \epsilon.\text{sign}(\nabla_x l(\theta, x, y)) \tag{1}$$

with $l(\theta, x, y)$ the loss of the classifier and y the true label.

As an adversary, one wishes to **maximize** the loss l. Presented this way, it is an *untargeted* attack. It pushes

[1]Google Research, Brain Team. Correspondence to: Léonard Hussenot <hussenot@google.com>.

C towards misclassifying x' in *any* other label than y. This can easily be turned into a *targeted* attack by taking $l(x) = -\,loss(\theta, x, y_{target})$ with y_{target} the label C is to predict for x'. It can be easily be adapted to a L_2 bound by normalizing the gradient to ϵ. The method will thus be referred to as the *fast-gradient method* (FGM). The choice of the norm is important as it defines how we measure the *imperceptibility* of the attack. We will focus on L_∞ and L_2 norms. The FGM can also be declined in iterative methods, by taking several steps in the direction of the gradient, and in momentum-based iterative methods by furthermore adjusting dynamically the gradient step. All these methods will be referred to as gradient-based attacks.

When using **deep networks** to compute its policy, a RL agent can be fooled the same way as a supervised classifier. For algorithms computing a stochastic policy π, we take y, the true label, as the action predicted by the network: $y = \text{argmax}_{a\in\mathcal{A}}\,\pi(a|s)$. The loss l from Eq. (1) will then encourage the network not to have the same output it had when not being attacked: $l = H(\mathbb{1}_{\{y=.\}}, \pi(.|s))$ with $H(p,q) = -\sum_{x\in\mathcal{X}} p(x).\ln(q(x))$ In the case where the output policy is deterministic, *e.g.* $\pi(s) = \text{argmax}_{a\in\mathcal{A}}\,Q(s,a)$, the same calculus would lead to the gradient in Eq. (1) being zero almost everywhere. We thus take: $l = H(\mathbb{1}_{\{y=.\}}, \text{softmax}(Q))$ with $\text{softmax}(Q) = e^{Q(s,a)}/\sum_{a'\in\mathcal{A}} e^{Q(s,a')}$.

In RL, as no *true* labels are available, an adversarial example is either used in the untargeted case to change the initial agent's decision a or, in the targeted case, to encourage the agent to take an action a_{target} chosen by an adversary.

3. Model

In Huang et al. (2017) the adversarial example framework is directly used on the sequential decision making problem. The FGM is applied to the network inducing the policy π, for example the Q-network when $\pi(s) = \text{argmax}_a\,Q(s,a)$ or when $\pi(.|s)$ is drawn from $Q(s,.)$.

By doing so, the attack is built on the whole agent state. This might seem like a reasonable hypothesis in the *white-box* setting. Nonetheless, the state is an internal representation to the agent. It might result from a complex preprocessing of the raw perceptions given by the environment. For example, when playing Atari games, classic RL algorithms use $k = 4$ consecutive observations as input state to ensure the Markov property. Therefore deriving the loss l from Eq. (1) with respect to the input state will give an attack on k observations rather than one image as it was the case in supervised learning. Deploying this attack means having access to the memory of the agent and modifying it. Attacking the internal representation of the agent is thus a technical obstacle to the *black-box* setting where the opponent is conceptually located between the environment and the agent. Moreover, doing so may, depending on the implementation, break the assumption that the norm of the attack is bounded by ϵ as past observations are attacked several times. We thus wish to build attacks on raw observations rather than complete states and prove their efficiency.

We denote o_i the i^{th} observation and s_i the (unattacked) state. It results from a preprocessing f of the past observations: $s_i = f(o_{1:i})$. We denote η_i the i^{th} attack, $\tilde{o}_i = o_i + \eta_i$ the attacked observation and $\tilde{s}_i = f(\tilde{o}_{1:i-1}, o_i)$ the state where all observations but the last have been attacked. As we wish to modify observations using gradient-based attacks, we do not compute the gradient of l w.r.t. the input state s_i, but only w.r.t. the last observation o_i. By respecting the property, our attacks are more easily deployable as they consist in adding an imperceptible noise on observations, and consistently respect the norm constraint $\|\eta\| \leq \epsilon$. We consider the two problems of attacking an agent to make its performance drop (untargated case) and of matching the agent's policy with a desired one (targeted case).

3.1. Untargeted attacks

We first focus on untargeted attacks where noise is added to observations so as to make performance plunge. As stated in Sec. 1, we design two untargeted attacks. First, the per-frame attack: we compute a new attack for each frame and apply it online. Second, the constant attack: we compute a single attack and apply to all observations of the episode.

Per-frame attack: We study the effect of the following designed attack: at each time step, either FGM or an iterative method will be applied to the last seen observation in order to change the decision that the agent would have taken. The objective of the fast-gradient method can thus be reformulated as maximizing over η, denoting $a_i^* \in \text{argmax}\,Q(s_i,.)$,

$$\text{KL}\Big(\mathbb{1}_{\{a=a_i^*\}} \parallel \pi(.|f(\tilde{o}_{1:i-1}, o_i+\eta))\Big) \text{ s.t. } \|\eta\| \leq \epsilon.$$

FGM, as iterative methods, can thus be seen as gradient step(s) for maximizing the KL-divergence between policies.

This attack, though more reasonable considering that it attacks only observations and not states, may still be unenforceable as computing a different attack *online* might be computationally prohibitive for the opponent. We thus design a constant imperceptible attack that we will apply to every frame as a mask.

Constant attack: We study the effect of this new type of attacks: given an observed unattacked trajectory $\tau = (o_1, a_1), ..., (o_i, a_i), ..., (o_I, a_I))$, the opponent looks for η minimizing:

$$\sum_{i=1}^{I} \ln \mathbb{P}\Big(a_i = \underset{a}{\text{argmax}}\, Q(f(o_{1:i-1}, o_i+\eta), a)\Big) \text{ s.t. } \|\eta\| \leq \epsilon.$$

By replacing ∇l in Eq. (1) by the mean gradient over the episode, we minimize the likelihood of the agent's transitions over a trajectory. The single constant additive mask $\eta = \epsilon.\text{sign}(\sum_i \nabla_{x_i} l(\theta, x_i, y_i))$ is then applied over all successive frames.

While these attacks are designed to make the agent's performance drop, this might not be the only possible objective of a malicious opponent. It may also try to *control* the played policy. For this purpose we design targeted attacks.

3.2. Targeted case

We now consider again the per-frame attack scenario but with a different objective. We do not want to bring the performance of the agent down anymore, but to match its policy with the one of the opponent: π^{op}. We thus apply a targeted attack on each observation to encourage the agent to take the opponent's preferred action. Denoting $a^{*,\text{op}} \in \text{argmax}\, \pi^{op}(.|s)$, the loss l from Eq. (1) is now the opposite and the optimized objective is to minimize over η:

$$\text{KL}\Big(\mathbb{1}_{\{a=a^{*,\text{op}}\}} \parallel \pi(\cdot|f(\tilde{o}_{1:i-1}, o_i + \eta))\Big) \text{ s.t. } \|\eta\| \leq \epsilon.$$

4. Experiments

We test our attacks on both DQN and Rainbow playing Space Invaders. We use the same preprocessing as in (Mnih et al., 2015), with sticky actions (Machado et al., 2018). We use trained policies (Such et al., 2018) for both DQN and Rainbow and use them in evaluation mode : parameters of the agents are left unchanged. Results are averaged over five training seeds and over 5 episodes for each seed. Performance is evaluated in terms of undiscounted cumulative return over a complete episode. We adjust ϵ with the used norm : when considering a norm $\|.\|$ and an ϵ, we in fact bound the attack as following : $\|\eta\| \leq \|\epsilon \mathbb{1}\|$. We attack observations that are on the unnormalized 255 gray-scale. The attack bounded by ϵ is thus also to be divided by 255. Moreover, attacked observation are always clipped to 0-255 to keep them in the valid range. Fig. 1 shows the visual impression of a norm L_2 attack bounded by $\epsilon = 5$.

4.1. Untargeted attack

Per-frame attack. We attack every frame of the episode, just adding the computed noise on the current frame, never attacking the whole state, and compare different attacks.

We observe on Fig. 2 and 3 that both DQN and Rainbow are very sensitive to adversarial examples, even if we limit ourselves to attacking observations and not complete state inputs. We are able, with $\epsilon \leq 0.1$ to decrease the performance of the agent by more than 50% and to reduce its performance to the one of a randomly-acting agent with $\epsilon \leq 0.2$.

Figure 1. Left: the unattacked perception. Middle: the attacked perception with $= \epsilon < 5$. Right: the rescaled attack.

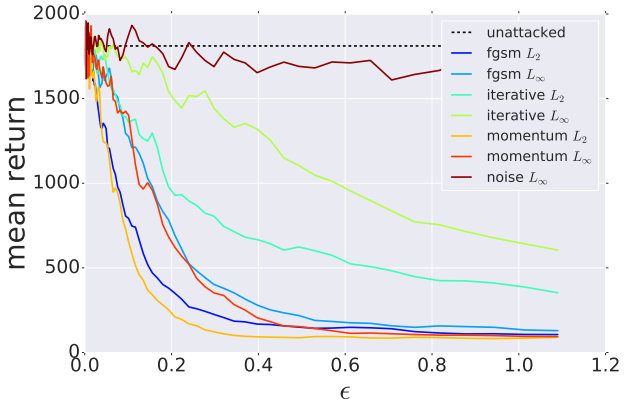

Figure 2. Per-frame attacks on DQN.

Simple FGM performs well in this untargeted case even though momentum-based attack outperforms it. We also observe that a gaussian noise attack limited by ϵ in norm L_∞ is not able to bring the performance of the agent down as effectively.

Constant attack. We now test our constant attack. Results are shown on Fig. 4 and 5. The attack computed by watching one episode, collecting the attacks over the trajectory without applying them and then taking the mean attack renormalized to an ϵ norm. It is then constantly applied over five episodes to test its mean effectiveness over an episode.

We observe that both algorithms show a high sensitivity to constant attacks in norm-2 and that norm-inf attacks are ineffective. With norm-2 bounded attacks, $\epsilon = 1$ suffice to reduce the performance of Rainbow to the one of a randomly acting agent. It seems more sensitive than DQN, for which a bigger ϵ (around 2) is needed to make the performance drop to the one a randomly acting agent.

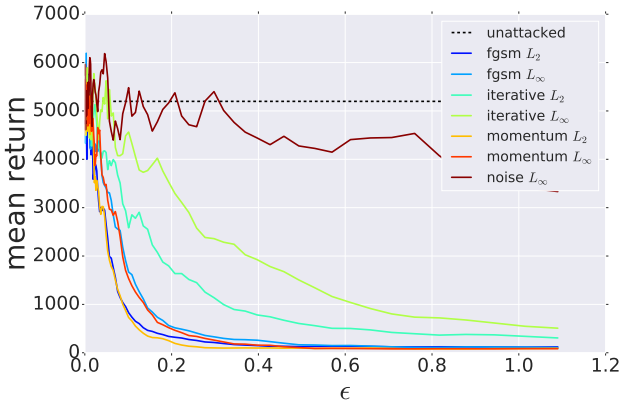

Figure 3. Per-frame attacks on Rainbow.

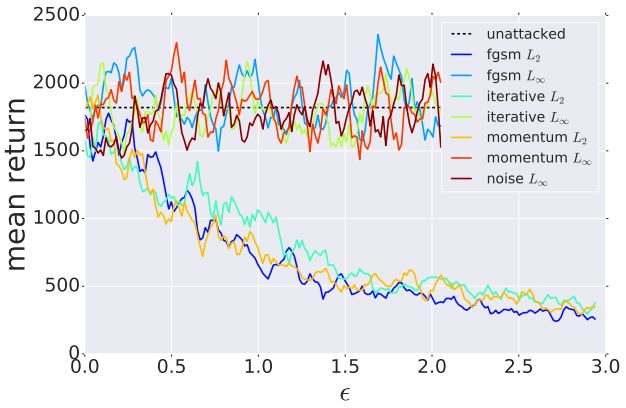

Figure 4. Constant attacks on DQN.

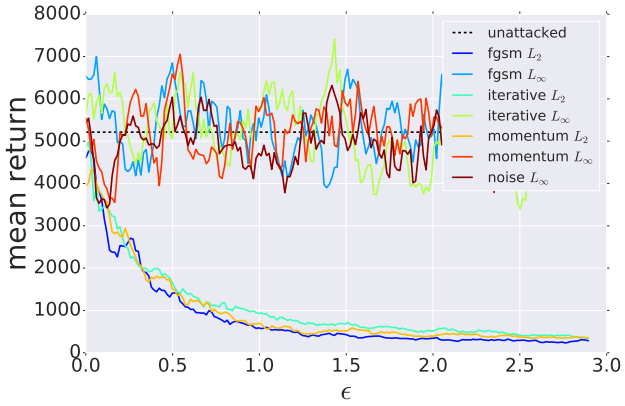

Figure 5. Constant attacks on Rainbow.

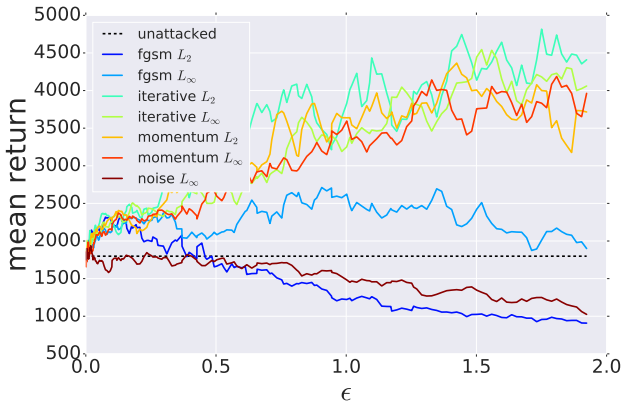

Figure 6. Targeted "helpful" per-frame attacks on DQN.

4.2. Targeted attack

We now consider again that we can compute a new attack for each current frame. Our goal is to experimentally test the hypothesis that the agent can be manipulated into acting as a policy π^{op} it was not computed for. We test that hypothesis by taking a DQN trained agent and apply a *per-frame targeted attack*. In our case, the considered attack will be "helpful": we are attacking DQN with the policy π^{op} of Rainbow whose results are higher on these tasks. The attack at time step i is computed with $y_{target} = \pi^{op}(s_i)$. DQN's behavior will thus be encouraged by adversarial examples to follow Rainbow's policy.

We observe on Fig. 6 that we are able to improve the performance of DQN significantly by attacking with still very small perturbations $\epsilon \approx 1.5$ and almost make it reach Rainbow's mean performance on the game. We also observe that in this targeted case, fast-gradient method is not enough. It suffices to prevent a policy to take its preferred action but is not precise enough to make it take a particular action.

Best returns are achieved when more than 95% of the taken actions match with the opponent action, given by the Rainbow algorithm (Fig. 7). We push further this experience by "helpfully" attacking an **untrained** version of DQN with a Rainbow opponent and see if we can make it perform well with only small perturbations of its perception.

As observed on Fig. 8, the task seems harder however with a bigger ϵ, e.g. $\epsilon = 2.7$, we reach the performance of a DQN trained for 200 millions steps. As can be seen on Fig. 9, more than 90% of the actions can be transformed to the targeted action chosen by Rainbow. On a topological point of view, this means that for both a trained or an untrained network Q, if you consider the multi-class classifier $C(.) = \text{argmax}_a\, Q(s, .)$ then, for any $s \in \mathcal{S}$, for ϵ of the order of magnitude shown on Fig. 9, then the decision frontier of every class cross the ball centered on s of radius ϵ.

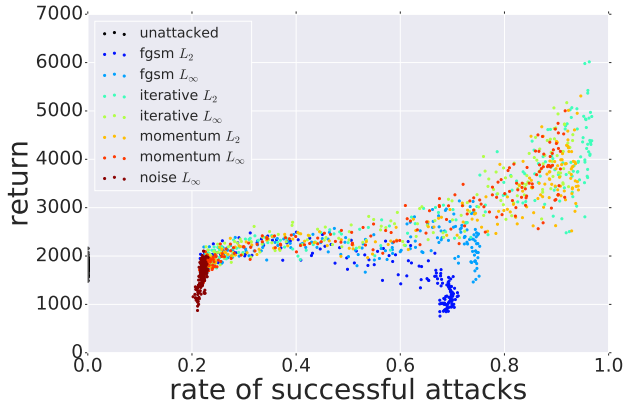

Figure 7. Targeted "helpful" per-frame attacks on DQN.

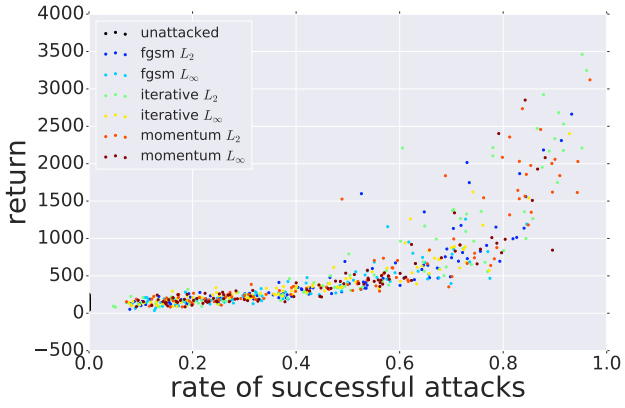

Figure 9. Targeted "helpful" per-frame attacks on untrained-DQN.

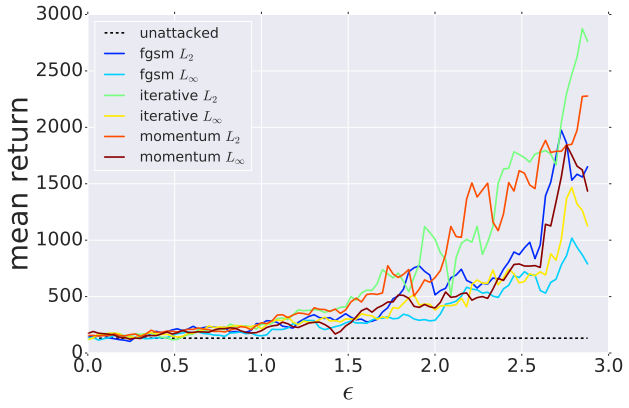

Figure 8. Targeted "helpful" per-frame attacks on untrained-DQN.

5. Discussion & Conclusion

In the perspective of making machine learning safe, adversarial examples (Szegedy et al., 2013) were introduced to highlight weaknesses of deep learning. Several algorithms were developed to produce adversarial examples rapidly (Goodfellow et al., 2015) or as close to the original example as possible (Carlini & Wagner, 2017). Defensive techniques have been proposed like distillation (Papernot et al., 2016) but other techniques (Carlini & Wagner, 2016) have proven their effectiveness against them. Adversarial examples on reinforcement learning are less studied. Previous work from Huang et al. (2017) first tackled the issue but, as stated in Sec. 3, the method attacked the whole agent's state and by that, added a strong assumption to the *white-box* setting. The method also considered it could attack at every frame the whole state with a new attack. We only attack observations, and designed new constant and targeted attacks. Lin et al. (2017) first considered targeted attacks on agents however, they defined a unique objective, which is to bring the agent into a particular state. For this purpose, the algorithm has to include a video prediction model. Their attack is also only tested with the adversarial attack introduced in Carlini & Wagner (2017), which is known to be slower than fast-gradient method to compute as it requires to solve an optimization problem. In the targeted case, we defined the more general objective of matching the attacked policy with a desired one. Concerning untargeted attacks to decrease agent performance, the proposed method reach the same performance as Huang et al. (2017) only attacking 25% of the frames.

We designed new attacks applied directly on observations for manipulating trained agents and proved their effectiveness on classical RL algorithms playing Atari games. We designed constant attacks, deployable in a real-world setting by adding a constant mask to observations. We also introduced targeted attacks to match the attacked policy with a desired one and proved it was doable by only imperceptibly attacking the observations. We focused on the *white-box* setting but generalizing to the black-box scenario would be done by training a new agent on the same task and transfer the attack computed on this agent to the target agent. We leave this scenario for future work.

Pinto et al. (2017) proposed an adversarial method for robust training of agents but considered only attack on the dynamic of the environment, not on the visual perception of the agent. Zhang et al. (2017); Ruderman et al. (2018) proposed adversarial environment generation to study agent's generalization and worst-case scenarios.

As future work, we wish to build targeted attacks by considering the norm budget ϵ as a budget available across the whole episode. This way, the opponent might either spare its budget or decide to attack strongly when needed.

References

Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Brown, T. B., Mané, D., Roy, A., Abadi, M., and Gilmer, J. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

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

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples (2014). *arXiv preprint arXiv:1412.6572*, 2015.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

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

Lin, Y.-C., Hong, Z.-W., Liao, Y.-H., Shih, M.-L., Liu, M.-Y., and Sun, M. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*, 2017.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., and Bowling, M. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597. IEEE, 2016.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. Robust adversarial reinforcement learning. *arXiv preprint arXiv:1703.02702*, 2017.

Ruderman, A., Everett, R., Sikder, B., Soyer, H., Uesato, J., Kumar, A., Beattie, C., and Kohli, P. Uncovering surprising behaviors in reinforcement learning via worst-case analysis. 2018.

Such, F. P., Madhavan, V., Liu, R., Wang, R., Castro, P. S., Li, Y., Schubert, L., Bellemare, M., Clune, J., and Lehman, J. An atari model zoo for analyzing, visualizing, and comparing deep reinforcement learning agents. *arXiv preprint arXiv:1812.07069*, 2018.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Zhang, H., Wang, J., Zhou, Z., Zhang, W., Wen, Y., Yu, Y., and Li, W. Learning to design games: Strategic environments in deep reinforcement learning. *arXiv preprint arXiv:1707.01310*, 2017.