Optimal attacks on Reinforcement Learning policies[a]

Alessio Russo and Alexandre Proutiere

American Control Conference (ACC), 2021

KTH, Royal Institute of Technology, Stockholm

[a]Based on a pre-print published in '19[1].

Problem Motivation and Background

Problem motivation

Problem:

- Since the discovery of Adversarial Examples (AEs) in Deep Learning [2-3], **the Machine Learning (ML) community has devised multiple ways to attack ML and Deep Reinforcement Learning (RL) models [4-5].**
- However, **the idea behind AEs is not necessarily restricted to Deep Learning. It is about finding a perturbation that minimizes some performance criterion.**

Questions: *can we model attacks for generic Markov Decision Processes and RL? Is there a general attack framework?*

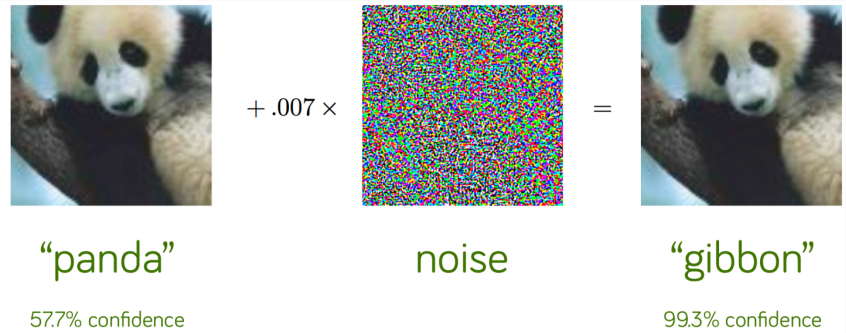
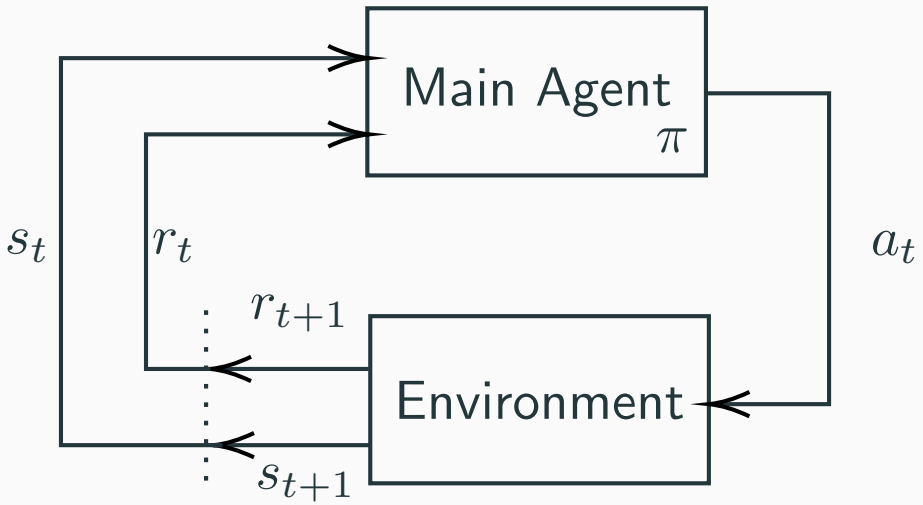


Figure 1: Adversarial examples in Supervised Learning (image from [3]).

Background: Markov Decision Processes



Transition density

$$s_{t+1} \sim P(\cdot|s_t, a_t)$$

Reward function

$$r_t = r(s_t, a_t)$$

Policy of the agent

$$a_t \sim \pi(\cdot|s_t)$$

Figure 2: Markov Decision Process

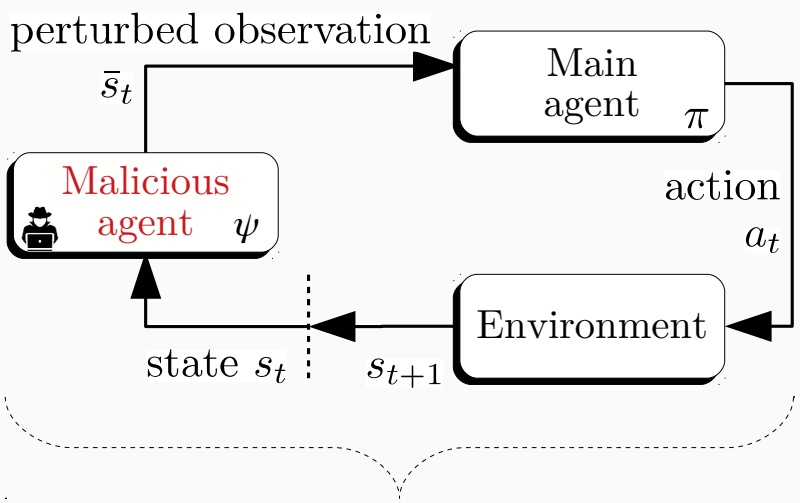
The *discounted value* of π, is

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(s)}[r(s,a) + \gamma \mathbb{E}_{s'\sim P(s,a)}[V^\pi(s')]].$$

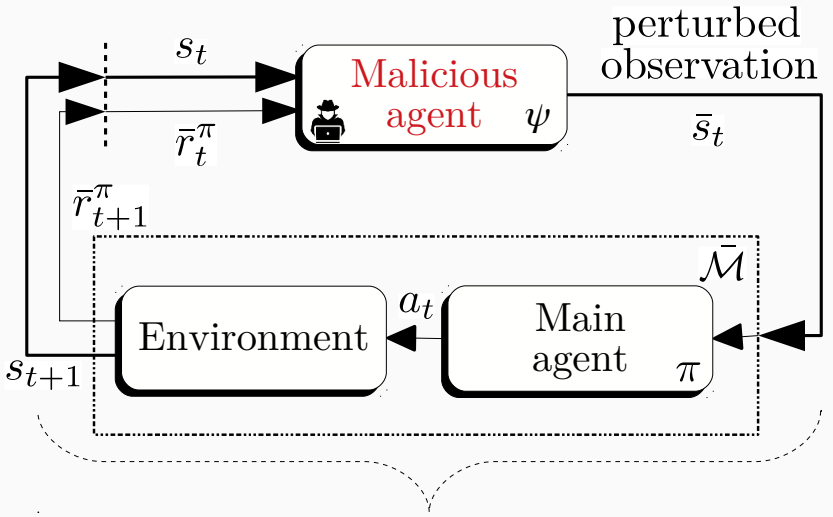
Suppose the policy of the agent π is fixed. How can we attack it by perturbing the observations of the state? ($\rightarrow$ attack at test time.)

Optimal Attacks Formulation

Optimal adversarial attacks on the observations



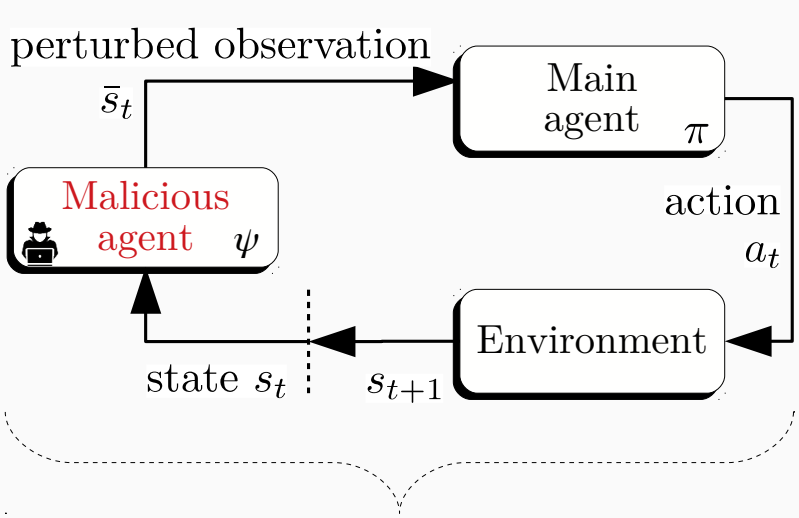
Attack scenario at test time



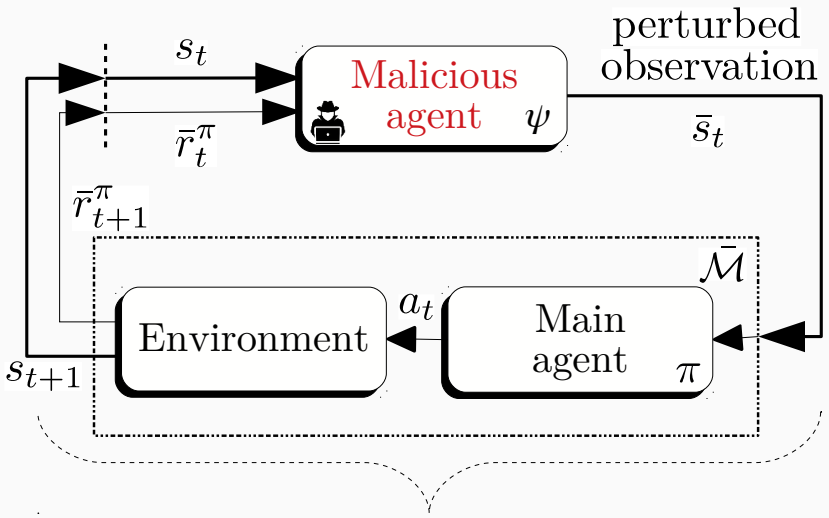
Model of the environment seen by the malicious agent

Key Idea: Use MDPs to frame the optimal attack.

Optimal adversarial attacks on the observations



Attack scenario at test time



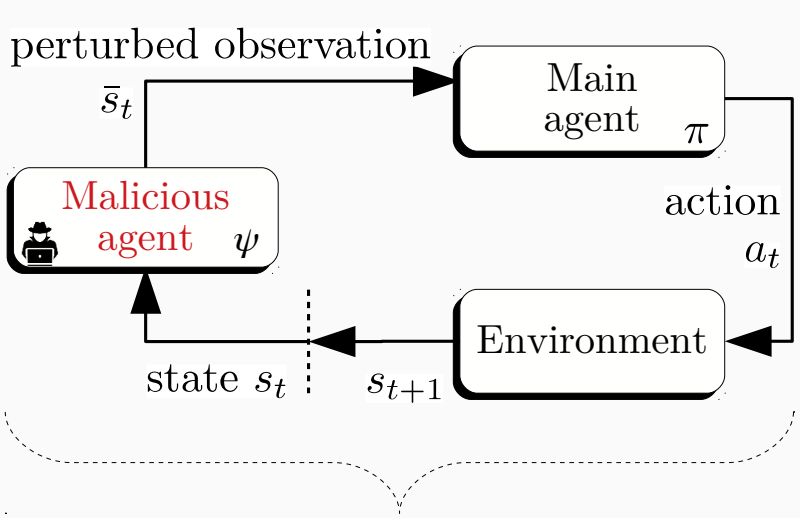
Model of the environment seen by the malicious agent

Key Idea: Use MDPs to frame the optimal attack.

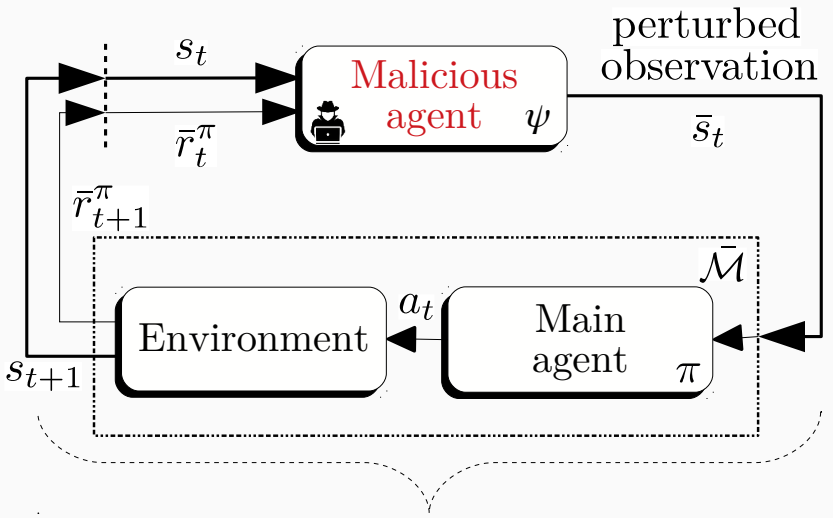
The attack can be seen as a Man In The Middle attack.

- **If** (P, π) **are known**, the attack problem can be solved by means of Value/Policy Iteration (*white-box case*).
- **If** (P, π) **are unknown**, we can cast the problem of finding an optimal attack as a Reinforcement Learning problem (*black-box case*, since the model is not known).

Optimal adversarial attacks on the observations



Attack scenario at test time



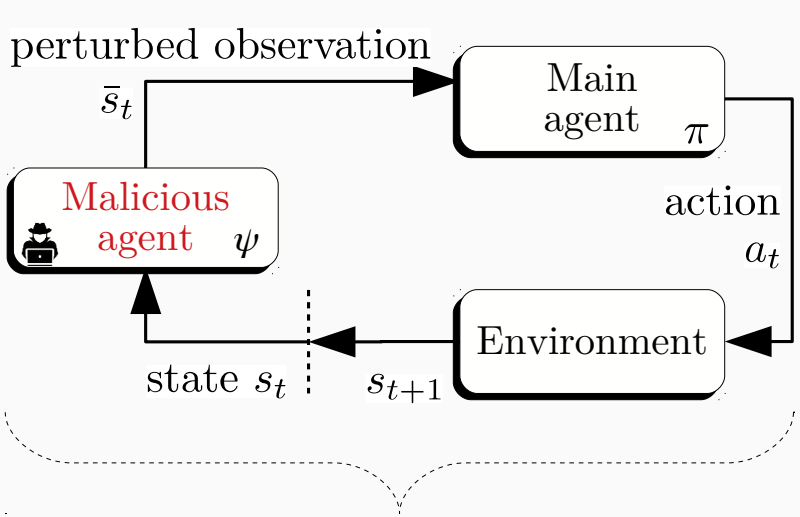
Model of the environment seen by the malicious agent

Define the MDP of the malicious agent $\bar{M} = (\mathcal{S}, \bar{\mathcal{S}}, \bar{P}^\pi, \bar{r}^\pi)$, where $\bar{\mathcal{S}} = \mathcal{S}$ is the set of possible actions, and

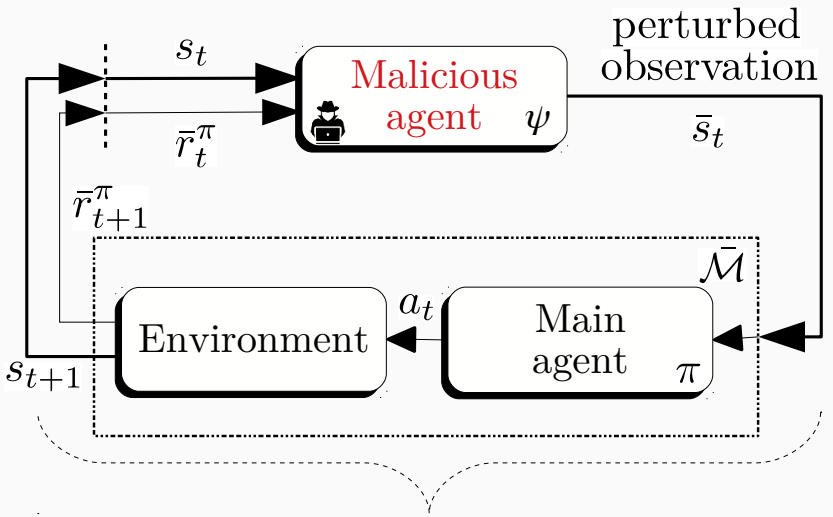
- $\bar{P}^\pi(s'|s, \bar{s}) = \mathbb{E}_{a\sim\pi(\cdot|\bar{s})}[P(s'|s, a)]$ is the probability transition matrix
- $\bar{r}^\pi(s, \bar{s})$ is the adversarial reward function.

with the constraint that $\bar{s}$ satisfies $d(s, \bar{s}) \leq \varepsilon$, for some metric function d. *This represents the idea that observations should not be too far from the original signal.*

Optimal adversarial attacks on the observations



Attack scenario at test time

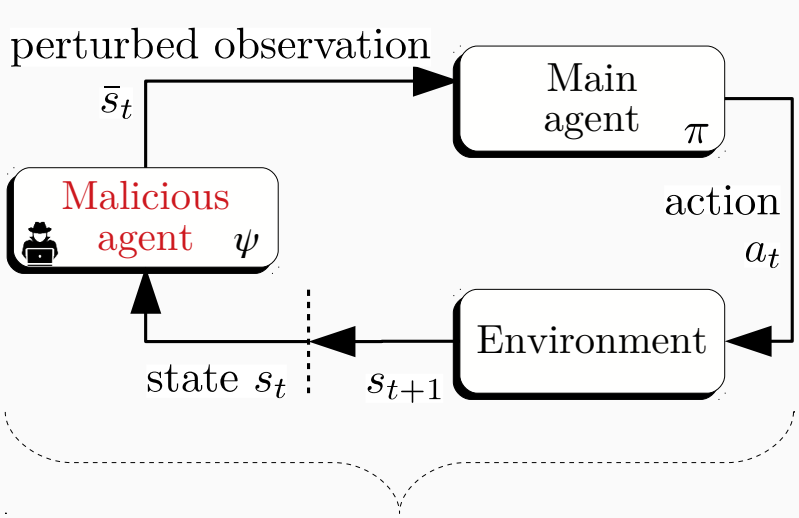


Model of the environment seen by the malicious agent

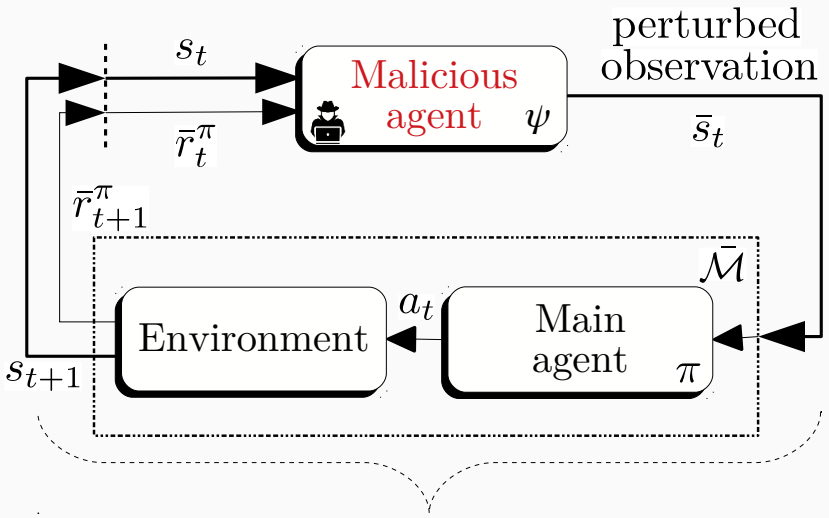
Denote by ψ the policy of the adversarial agent, and its value V^{ψ}. As a consequence, the value of the policy of the main agent π changes when affected by perturbation. **Denote the value of π under a perturbation ψ as $V^{\pi\circ\psi}$:**

$$V^{\pi\circ\psi}(s) = \mathbb{E}_{a\sim\pi(\cdot|\psi(s))}[r(s,a) + \gamma\mathbb{E}_{s'}[V^{\pi\circ\psi}(s')]].$$

Optimal adversarial attacks on the observations



Attack scenario at test time



Model of the environment seen by the malicious agent

Denote by ψ the policy of the adversarial agent, and its value V^{ψ}. As a consequence, the value of the policy of the main agent π changes when affected by perturbation. **Denote the value of π under a perturbation ψ as $V^{\pi\circ\psi}$:**

$$V^{\pi\circ\psi}(s) = \mathbb{E}_{a\sim\pi(\cdot|\psi(s))}[r(s,a) + \gamma\mathbb{E}_{s'}[V^{\pi\circ\psi}(s')]].$$

If the adversary chooses a reward $\bar{r}^{\pi}(s,\bar{s}) = -\mathbb{E}_{a\sim\pi(\cdot|\bar{s})}[r(s,a)]$ we find the definition of minimax attack (since we get $V^{\psi} = -V^{\pi\circ\psi}$).

RL Approach to find an Optimal Attack

The action space of the adversary is essentially the entire state space: how can we deal with MDPs whose action space might be very huge?

The idea is to use RL methods with one of the following approaches

1. **Features extraction**: best solution (computationally). However, how do we reconstruct the original observation from those features? (for images this is not trivial).

RL Approach to find an Optimal Attack

The action space of the adversary is essentially the entire state space: how can we deal with MDPs whose action space might be very huge?

The idea is to use RL methods with one of the following approaches

1. **Features extraction**: best solution (computationally). However, how do we reconstruct the original observation from those features? (for images this is not trivial).
2. **Function Approximators**: by using actor-critic methods such as DDPG [6] or PPO [7].

Robustness and defenses

What can the defender do?

1. **First observe that any attack induces a Partially Observable Model** (POMDP). Therefore, we expect POMDP techniques to be useful in that regard (such as using history of observations, or recurrent neural networks).

Robustness and defenses

What can the defender do?

1. **First observe that any attack induces a Partially Observable Model** (POMDP). Therefore, we expect POMDP techniques to be useful in that regard (such as using history of observations, or recurrent neural networks).
2. **We have the following proposition that bounds the effect of the attack** (one can try to find a policy π that minimizes the bound for a specific attack ψ).

Proposition

Let the reward be bounded uniformly by R. Then, the bound on the value of the perturbed policy π and the perturbed one $\pi \circ \psi$ is

$$\|V^{\pi} - V^{\pi \circ \psi}\|_{\infty} \leq 2R \frac{\alpha_{\varepsilon}}{(1-\gamma)^2}$$

where $\alpha_{\varepsilon} = \max_s \alpha_{\varepsilon}(s)$ and $\alpha_{\varepsilon}(s) = \max_{\bar{s} \in \mathcal{A}_s} \|\pi(s) - \pi(\bar{s})\|_{TV}$, with $\mathcal{A}_s = \{\hat{s} : d(s, \hat{s}) \leq \varepsilon\}$ (TV is the total variation distance).

Simulations

Simulations: attacks using DDPG

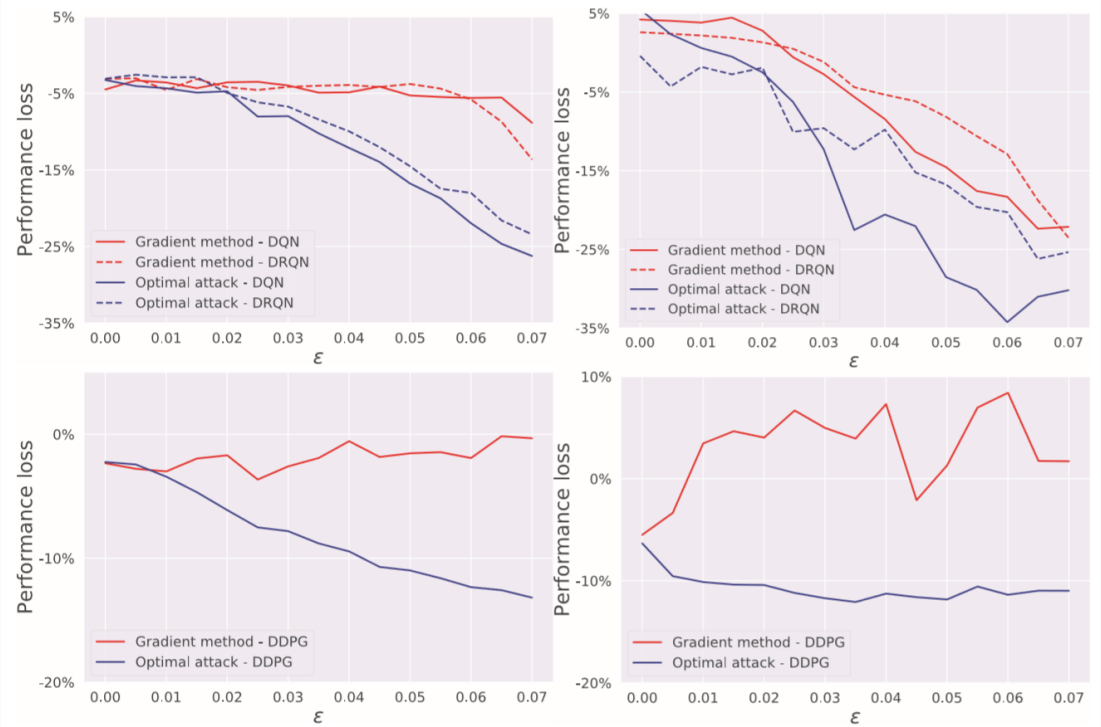


Figure 3: Environments: MountainCar (topleft), Cartpole (topright),Continuous MountainCar and LunarLander (bottomleft and bottomright).

- We tested the proposed attack methods on different OpenGym environments [8].
- We trained the attack using DDPG. To robustify the policy we used DRQN [9], a technique that uses DQN with recurrent layers to deal with partial observability.
- We see an improvement when using recurrent layers. Furthermore, continuous environments show better robustness properties.

Simulations: attacks using feature extraction

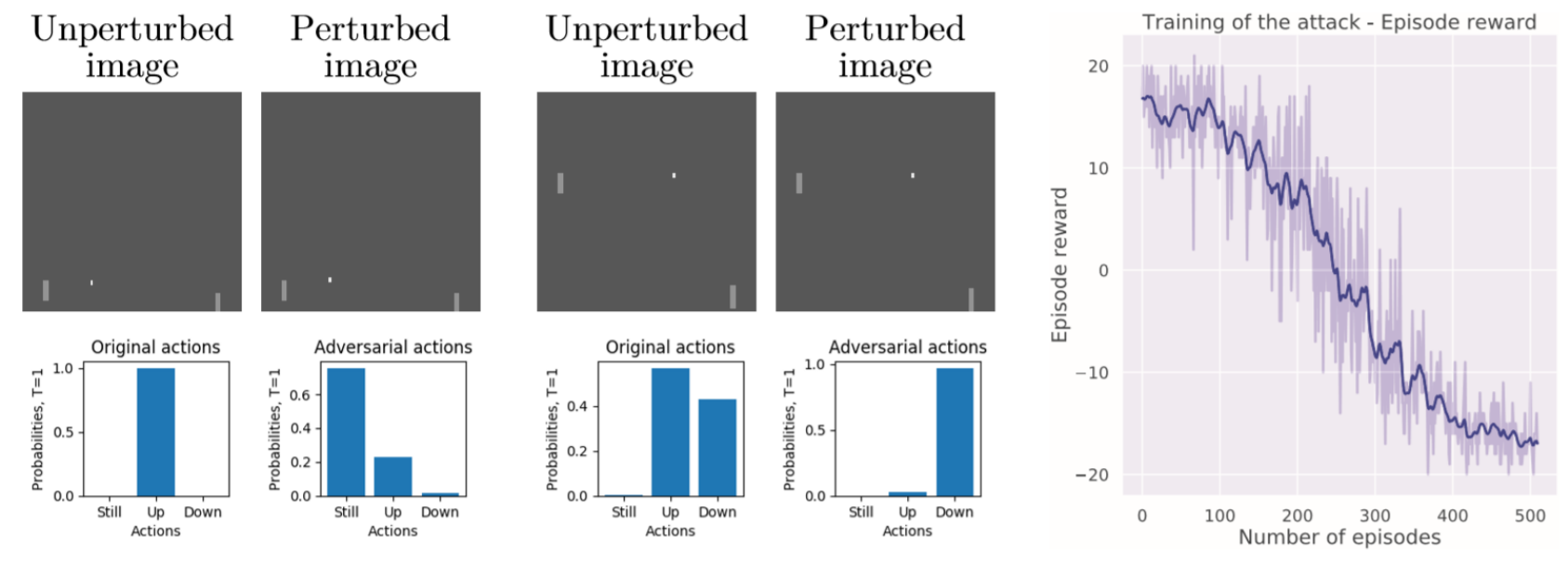


Figure 4: Video of the simulation: `https://youtu.be/mgUdAD4Mcj4`.

Atari Pong game: This is an example of feature extraction. The algorithm extracts the center of mass (COM) of each object (bars and ball), and chooses, using DQN, in which direction one of the objects should moved by 1 pixel.

Conclusion

Future work

Future work:

- Finding a more efficient way to compute optimal attacks is an interesting venue of research.
- Evaluate attacks on complex systems (such as the HVAC unit of a building).
- Should we focus on the problem of making the policy robust? Is it not more efficient to find a way to *detect* attacks?

Thank you for listening!

References

1. Russo, Alessio, and Alexandre Proutiere. "Optimal attacks on reinforcement learning policies." arXiv preprint arXiv:1907.13548 (2019).
2. Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian J., and Fergus, Rob. Intriguing properties of neural networks. ICLR, abs/1312.6199, 2014.
3. Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." arXiv preprint arXiv:1412.6572 (2014).
4. Huang, Sandy, et al. "Adversarial attacks on neural network policies." arXiv preprint arXiv:1702.02284 (2017).
5. Pattanaik, Anay, et al. "Robust deep reinforcement learning with adversarial attacks." arXiv preprint arXiv:1712.03632 (2017).
6. Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2015).
7. Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).
8. Brockman, Greg, et al. "Openai gym." arXiv preprint arXiv:1606.01540 (2016).
9. Hausknecht, Matthew, and Peter Stone. "Deep recurrent q-learning for partially observable mdps." arXiv preprint arXiv:1507.06527 (2015).

Backup slides

DDPG Algorithm

Algorithm 1 Malicious agent training

procedure MALICIOUSAGENTTRAINING($\pi, T_{\text{training}}, N_B$)
  Initialize critic and actor network Q_θ^χ, χ with weights θ and ϕ.
  Initialize target critic and actor networks $Q_{\theta_T}^\chi, \chi_{\phi_T}$ with weights $\theta_T \leftarrow \theta, \phi_T \leftarrow \phi$.
  Initialize replay buffer $\mathcal{D}$.
  for episode 1:M **do**
    Initialize environment and observe state s_0, and set initial time $t \leftarrow 0$.
    repeat
      Select a perturbation and add exploration noise $\bar{s}_t \leftarrow l_\Pi(\chi_\phi(s_t) + e_t)$.
      Execute action $a_t \sim \pi(\bar{s}_t)$ and observe reward and state $\bar{r}_t, s_{t+1}$.
      Append transition $(s_t, \bar{s}_t, \bar{r}_t, s_{t+1})$ to $\mathcal{D}$.
      if $t \equiv 0 \mod T_{\text{training}}$ **then**
        Sample random minibatch $B \sim \mathcal{D}$ of N_B elements.
        Set target for the i-th element of B to $y_i = \bar{r}_i + \gamma Q_{\theta_T}^\chi(s_{i+1}, \chi_{\phi_T}(s_{i+1}))$.
        Update critic by minimizing loss $\frac{1}{N_B}\sum_{i=1}^{N_B}(y_i - Q_\theta^\chi(s_i, \bar{s}_i))^2$.
        Update actor using sampled policy gradient

$$\nabla_\phi J \approx \frac{1}{N_B}\sum_{i=1}^{N_B} \nabla_{\bar{s}} Q_\theta^\chi(s_i, \bar{s}) \nabla_\phi \chi_\phi(s_i)|_{\bar{s}=\chi(s_i)}.$$

        Slowly update the target networks:

$$\theta_T \leftarrow (1-\tau)\theta_T + \tau\theta, \quad \phi_T \leftarrow (1-\tau)\phi_T + \tau\phi.$$

      end if
      $t \leftarrow t + 1$.
    until episode is terminal.
  end for
end procedure

Practical implementation

We add a last fixed layer to the network that projects onto the ball centred in 0, with radius ε. This is done by applying the function $x \to \min(1, \frac{\varepsilon}{\|x\|})x$ to $x + e$, where x is the output before the projection layer, and e is the exploration noise. After this projection, we add the state s to get the perturbed state.