Aalto University

UNIVERSITY OF WATERLOO

Real-time Adversarial Perturbations against Deep Reinforcement Learning Policies: Attacks and Defenses

Buse G. A. Tekgul, Shelly Wang, Samuel Marchal, N. Asokan

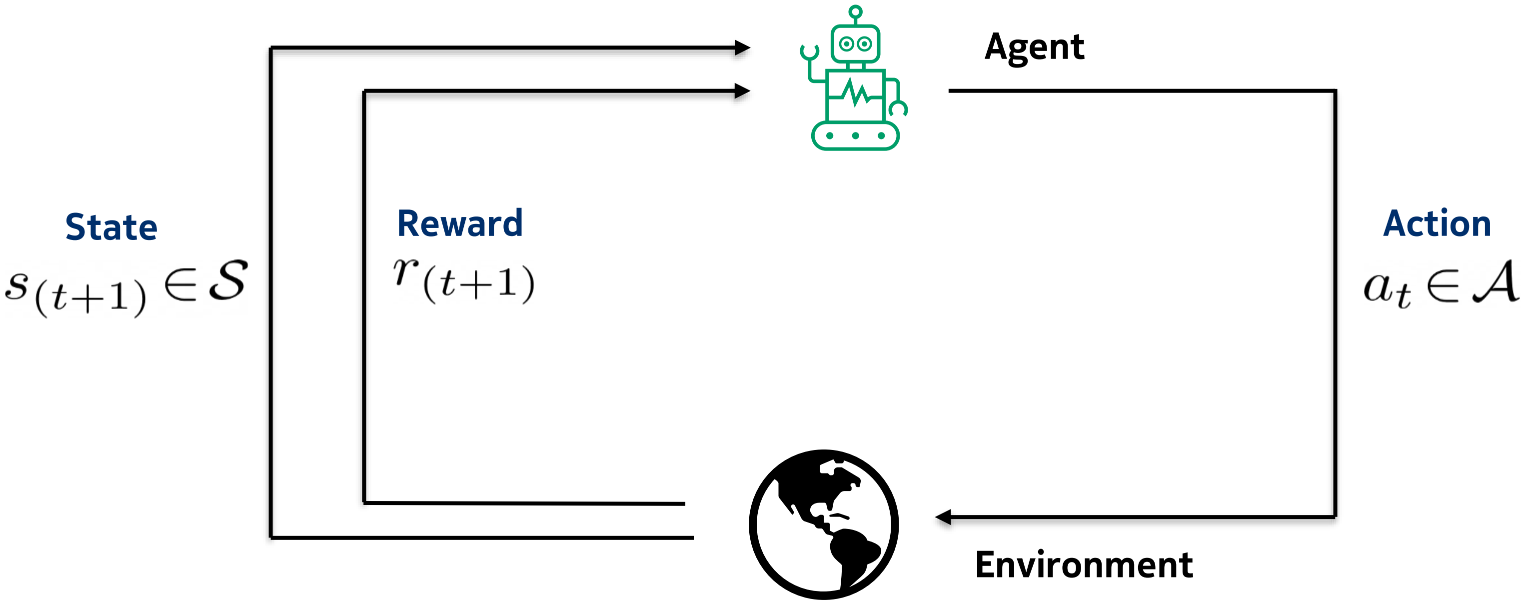
batlitekgul@acm.org
buse.atli_tekgul@nokia-bell-labs.com

Background
DNN Classifiers vs. DRL Agents

Reinforcement Learning

In RL, an agent interacts with an environment to optimize its policy

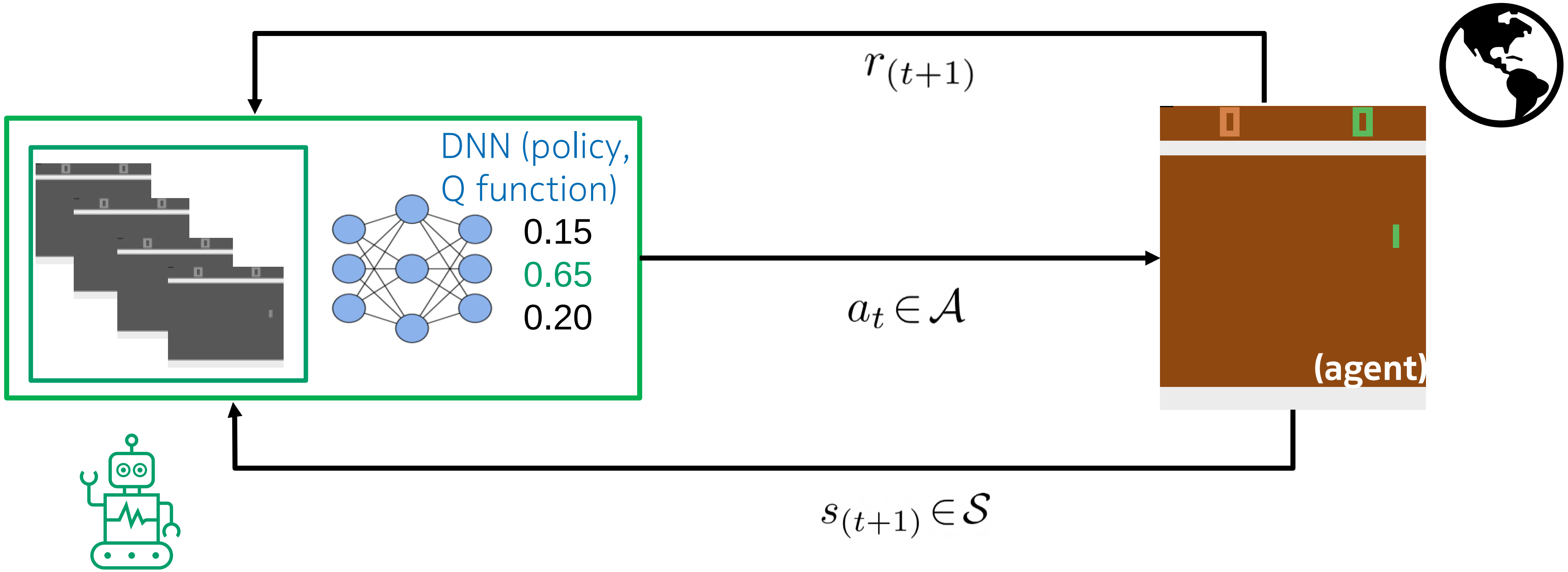
- Policy: Decision making strategy, $\pi(a_t|s_t) : \mathcal{S} \to \mathcal{A}$
- State-action value function: Helps optimizing the policy in discrete tasks, *Q(s,a)*



Deep Reinforcement Learning (DRL)

DRL learns successful policies directly from high-dimensional inputs

- Reinforcement Learning (RL) defines the objective: maximize future reward
- Deep Neural Networks (DNN) provides the mechanism: approximate the policy



Adversarial Examples in DNN Classifiers

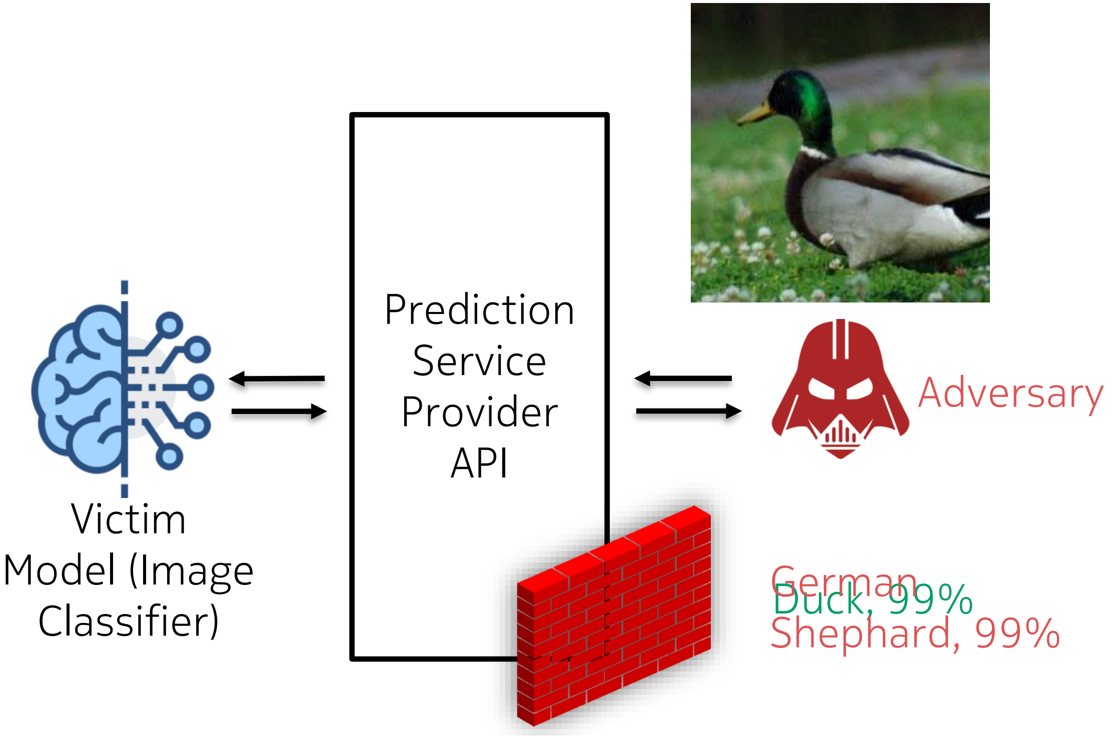
Adversary compromises model integrity: Cause wrong predictions via adversarial examples[1]

- Confidence reduction
- Targeted misclassification
- Untargeted misclassification

API granularity: From top label to full results

Model knowledge:

- Black-box (query-based methods, transferability)
- White-box



[1] Szegedy, et al. *"Intriguing properties of neural networks"* https://arxiv.org/abs/1312.6199

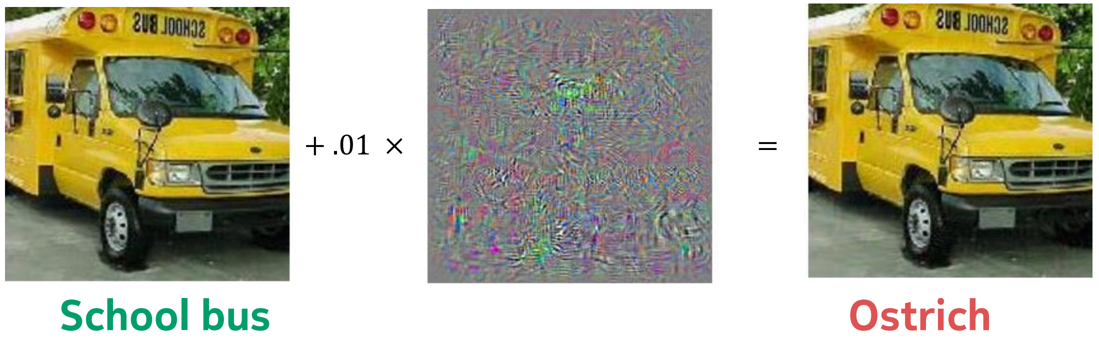
Adversarial Examples in DNN vs. DRL

RL has peculiarities (e.g., task complexity, stochasticity, limited observable information) that make the application of attacks against DNN classifiers challenging.

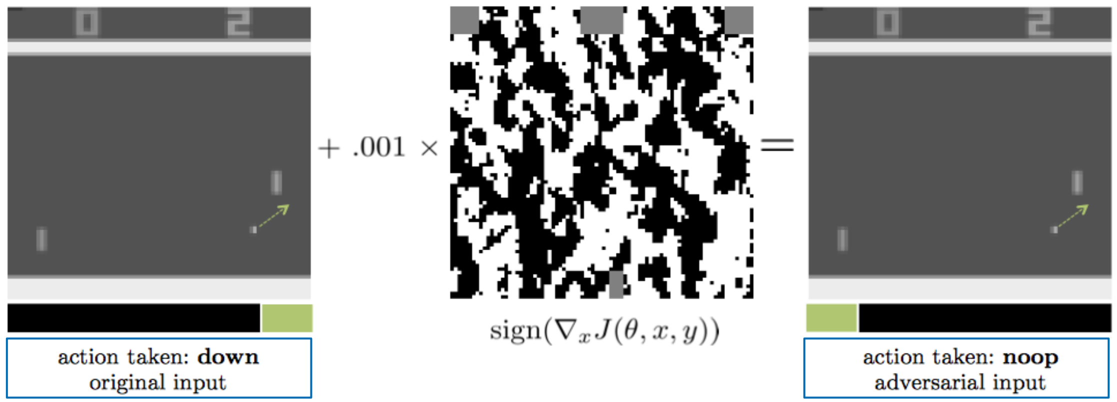
Adversarial perturbations are added into

- DNNs[1]: ... into clean image
  - → Classifier is victim, wrong label
- DRLs[2] : ... into directly environment, or states, sensors, actuators etc.
  - → Policy/Perception component/Control component is victim, sub-optimal action

[1]



[2]



1. Szegedy et al., *"Intriguing Properties of Neural Networks"* arXiv, 2013. https://arxiv.org/abs/1312.6199v4
2. Huang et al. *"Adversarial Attacks on Neural Network Policies"*, arXiv 2017. https://arxiv.org/pdf/1702.02284.pdf

Adversarial Examples in DNN vs. DRL

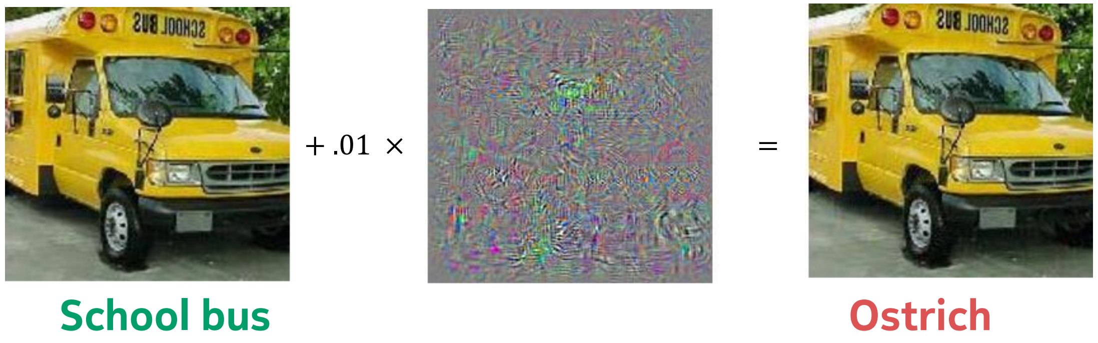
In DRL,

- No 1-1 mapping between states and actions (no pre-defined labels)
- One successful adversarial example might not affect the task

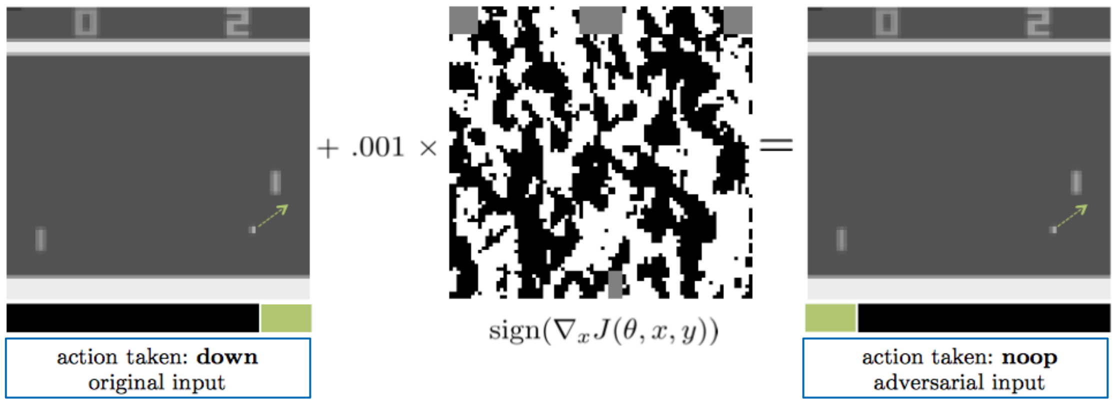
Adversarial goals in DRL:

- Reward minimization: Reduction in the return, i.e., total rewards (targeted or untargeted)
- Policy-luring: Force agent to reach a desired state, or follow a desired policy, path etc. (targeted)

[1]



[2]



1. Szegedy et al., *"Intriguing Properties of Neural Networks"* arXiv, 2013. https://arxiv.org/abs/1312.6199v4
2. Huang et al. *"Adversarial Attacks on Neural Network Policies"*, arXiv 2017. https://arxiv.org/pdf/1702.02284.pdf

Realistic Adversaries in DRL

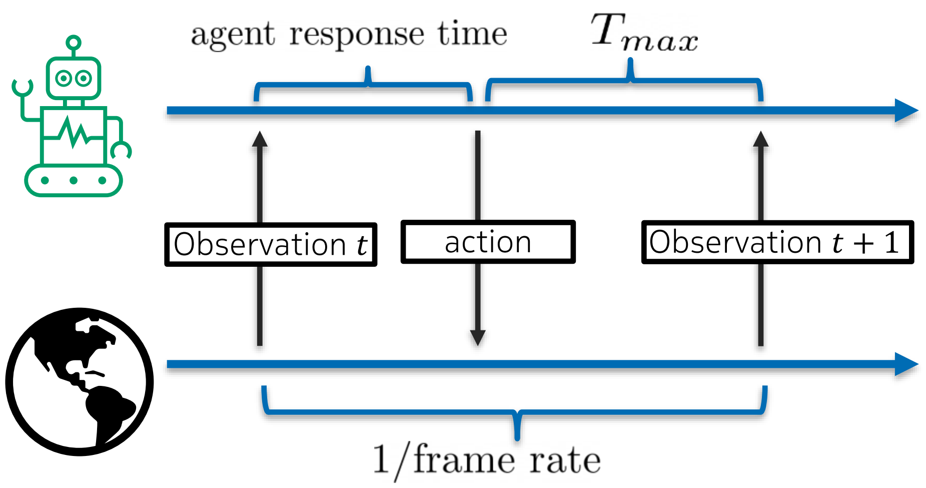
A realistic attack

- cannot change the inner workings of victim agent (e.g. short-term memory, received rewards)
- should compute + add the perturbation fast enough to be implemented in real time
- The online cost should be less than

$$T_{max} = 1/\text{frame rate} - \text{agent response time}$$

Prior attacks are not realistic, they

- are too slow to be mounted in real time[1,2]
- modify the short term memory of victim[3]



Can we effectively fool DRL policies in real-time?

1. Lin, Yen-Chen, et al. *"Tactics of adversarial attack on deep reinforcement learning agents."* IJCAI 2017. https://arxiv.org/abs/1703.06748
2. Pan, Xinlei, et al. *"Characterizing Attacks on Deep Reinforcement Learning."* AAMAS 2022. https://arxiv.org/abs/1907.09470
3. Huang et al. *"Adversarial Attacks on Neural Network Policies"*, arXiv 2017. https://arxiv.org/pdf/1702.02284

State- and Observation-Agnostic Adversarial Perturbations in DRL

Adversary Model

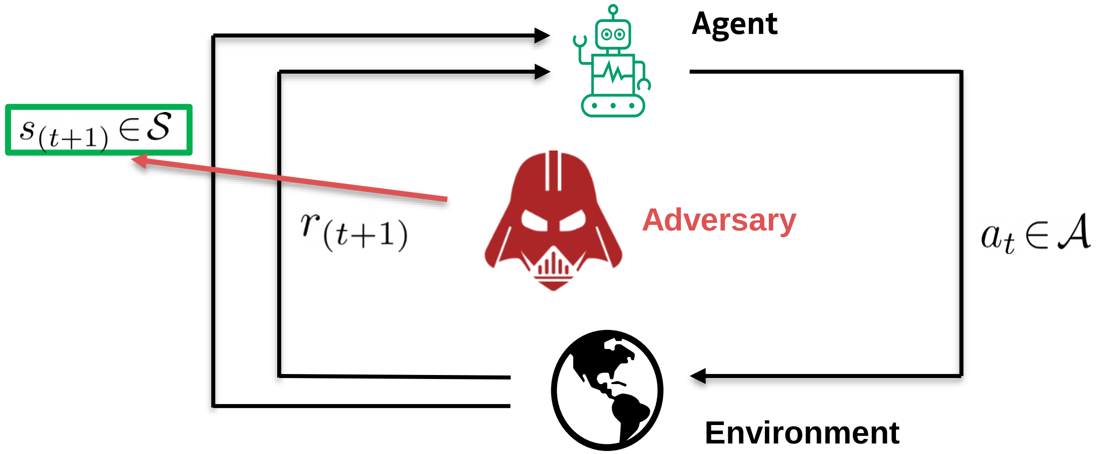
Adversary:

- wants a reinforcement learning agent to fail its task (reward minimization)
- uses state-action value function Q(s,a) to generate sub-optimal actions for discrete tasks

Adversarial capabilities:

- has the knowledge of
  - RL algorithm and
  - DNN model used for victim's policy
- cannot reset environment, replay earlier state,

  or induce a delay during the task

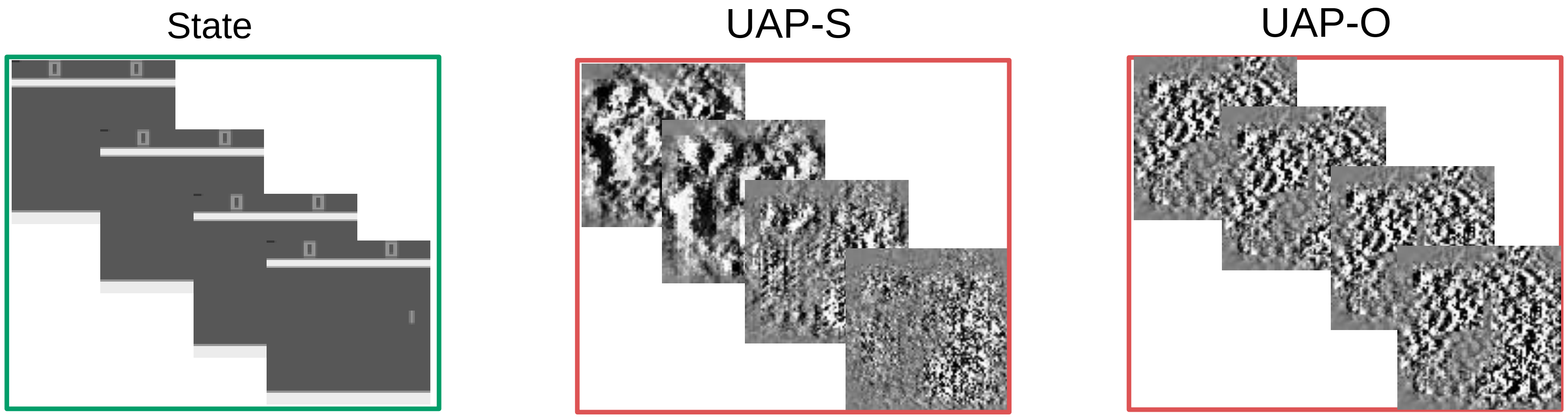


State- and Observation- Agnostic Perturbations

Universal Adversarial Perturbations (UAP)[1] in DRL settings using

- Find a sufficiently small perturbation $\|r\|_p = \|s_{adv(t)} - s_t\|_p$ that results in sub-optimal actions for every perturbed state $s_{adv(t)}$

- **State-agnostic (UAP-S):** Perturbation is uniform across different states but is not uniform between the observations within a state
- **Observation-agnostic (UAP-O):** Perturbation is uniform across all observations



1. Moosavi-Dezfooli, Seyed-Mohsen, et al. *"Universal adversarial perturbations."* CVPR 2017. https://arxiv.org/abs/1610.08401

State- and Observation- Agnostic Perturbations

Attack Design:

1. Collect training data by observing a full episode
2. Sanitize the training data by choosing only critical states
3. Clone DNN (i.e., approximated state-action value function) of victim agent to an adversary's agent
4. Compute the perturbation using Algorithm 1 in an offline manner

Add the perturbation to any other state in any other episode during the task

Algorithm 1: Computation of UAP-S and UAP-O

input : sanitized $\mathcal{D}_{train}$, Q_{adv}, desired fooling rate δ_{th}, max. number of iterations it_{max}, pert. constraint ϵ

output : universal $\boldsymbol{r}$

Initialize $\boldsymbol{r} \leftarrow 0$, $it \leftarrow 0$;

while $\delta < \delta_{max}$ ***and*** $it < it_{max}$ **do**

  for $s \in \mathcal{D}_{train}$ **do**

    if $\hat{Q}(\boldsymbol{s}+\boldsymbol{r}) = \hat{Q}(\boldsymbol{s})$ **then**

      Find the extra, minimal $\Delta\boldsymbol{r}$:

      $\Delta\boldsymbol{r} \leftarrow \text{argmin}_{\Delta\boldsymbol{r}} \|\Delta\boldsymbol{r}\|_2$ s.t. $\hat{Q}(\boldsymbol{s}+\boldsymbol{r}+\Delta\boldsymbol{r}) \neq \hat{Q}(\boldsymbol{s})$;

      $\boldsymbol{r} \leftarrow \text{sign}(\min(\text{abs}(\boldsymbol{r}+\Delta\boldsymbol{r}), \epsilon))$;

  Calculate δ with updated $\boldsymbol{r}$ on $\mathcal{D}_{train}$;

  $it \leftarrow (it+1)$;

State- and Observation- Agnostic Perturbations

Modification of obs-fgsm-wb (OSFW)[1] to a completely universal version OSFW(U):

OSFW:

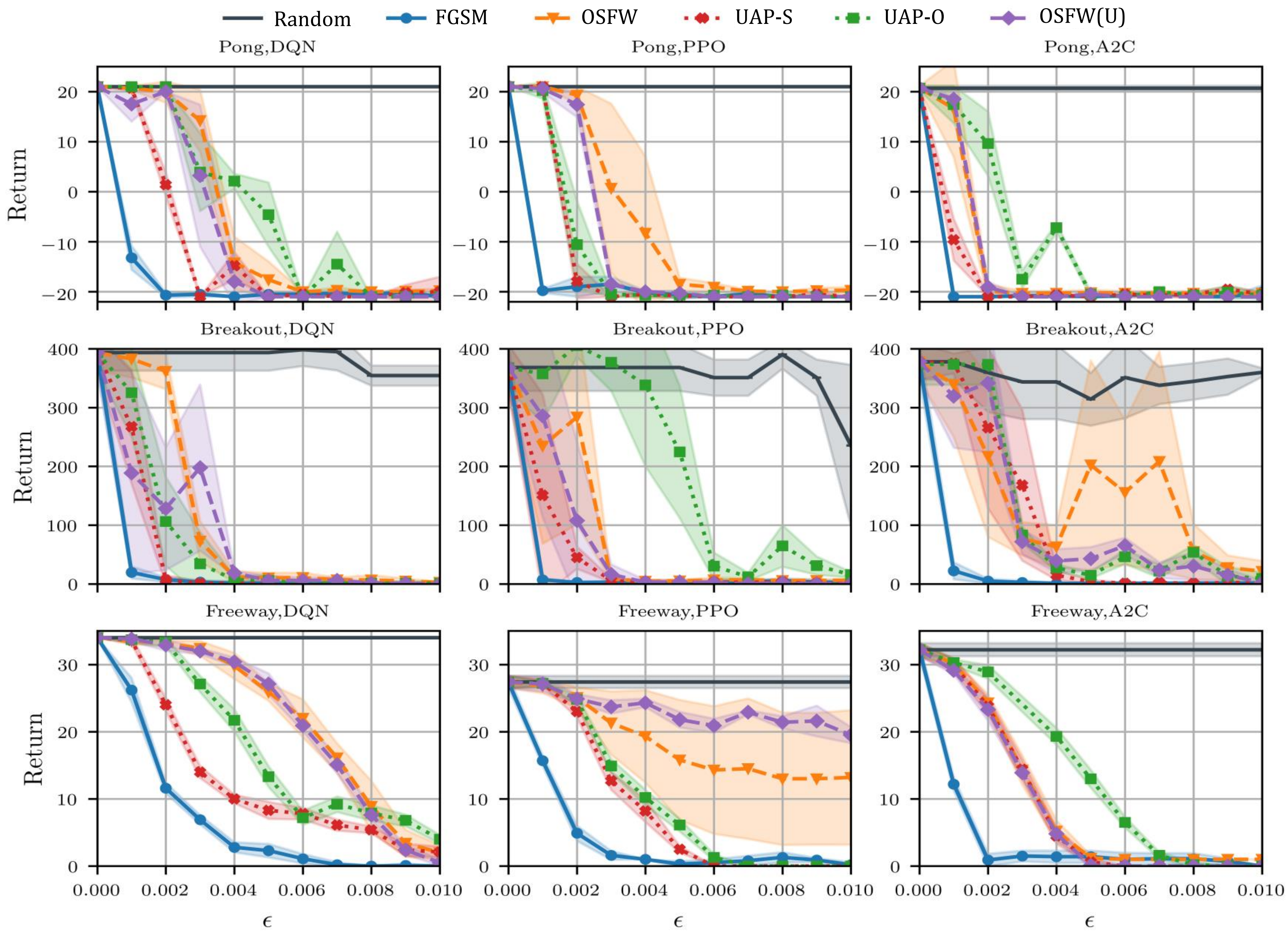
- calculates the perturbation by taking the average of the gradients of first k states
- adds the perturbation to the remaining states

Effectiveness of OSFW depends on the first k state for each run

Generating the perturbation takes longer than the minimum online cost

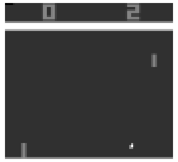
1. Pan, Xinlei, et al. *"Characterizing Attacks on Deep Reinforcement Learning."* AAMAS 2022. https://arxiv.org/abs/1907.09470

Experimental Results: Performance Degradation

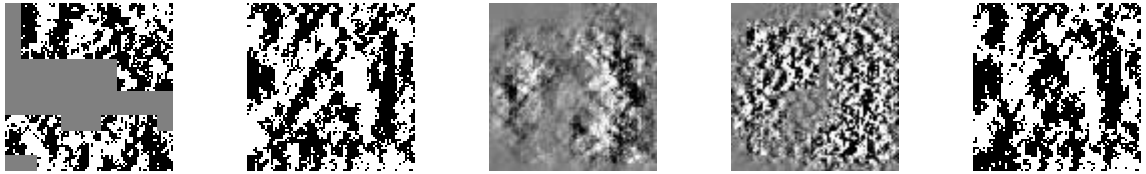


Experimental Results: Amount of Perturbation

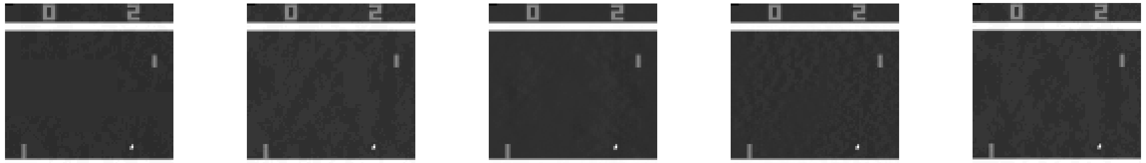
- UAP-S and UAP-O produce smaller perturbation compared to FGSM, OSFW and OSFW(U)
- Difference between objective functions

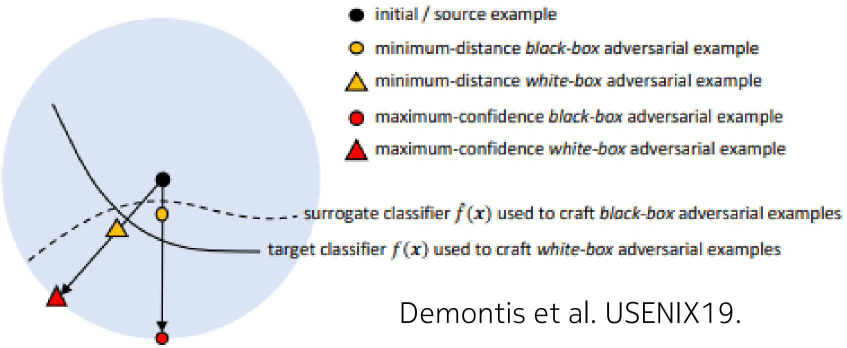


Clean state



FGSM OSWF UAP-S UAP-O OSFW(U)





Demontis et al. USENIX19.

15

Experimental Results: Computational Cost

- FGSM has low online cost, but requires rewriting victim agent's memory
- OSFW has high online cost, so it misses perturbing 102 states on average
- UAP-S, UAP-O have high offline cost, but it does not interfere with the task
- UAP-S, UAP-O and OSWF(U) low online cost, can be implemented in real-time

Experiment	Attack method	Offline cost$\pm$ std (seconds)	Online cost $\pm$ std (seconds)
Pong, DQN, $T_{max} = 0.0163 \pm 10^{-6}$ seconds	FGSM	-	$13 \times 10^{-4} \pm 10^{-5}$
	OSFW	-	5.3 ± 0.1
	UAP-S	36.4 ± 21.1	$2.7 \times 10^{-5} \pm 10^{-6}$
	UAP-O	138.3 ± 25.1	$2.7 \times 10^{-5} \pm 10^{-6}$
	OSFW(U)	5.3 ± 0.1	$2.7 \times 10^{-5}(\pm 10^{-6})$
Pong, PPO, $T_{max} = 0.0157 \pm 10^{-5}$ seconds	FGSM	-	$21 \times 10^{-4} \pm 10^{-5}$
	OSFW	-	7.02 ± 0.6
	UAP-S	41.9 ± 16.7	$2.7 \times 10^{-5} \pm 10^{-6}$
	UAP-O	138.3 ± 25.1	$2.7 \times 10^{-5} \pm 10^{-6}$
	OSFW(U)	7.02 ± 0.6	$2.7 \times 10^{-5} \pm 10^{-6}$
Pong, A2C $T_{max} = 0.0157 \pm 10^{-5}$ seconds	FGSM	-	$21 \times 10^{-4} \pm 10^{-5}$
	OSFW	-	7.2 ± 1.1
	UAP-S	11.4 ± 4.3	$2.7 \times 10^{-5} \pm 10^{-6}$
	UAP-O	55.5 ± 29.3	$2.7 \times 10^{-5} \pm 10^{-6}$
	OSFW(U)	7.2 ± 1.1	$2.7 \times 10^{-5} \pm 10^{-6}$

16

Experimental Results: Continuous Control

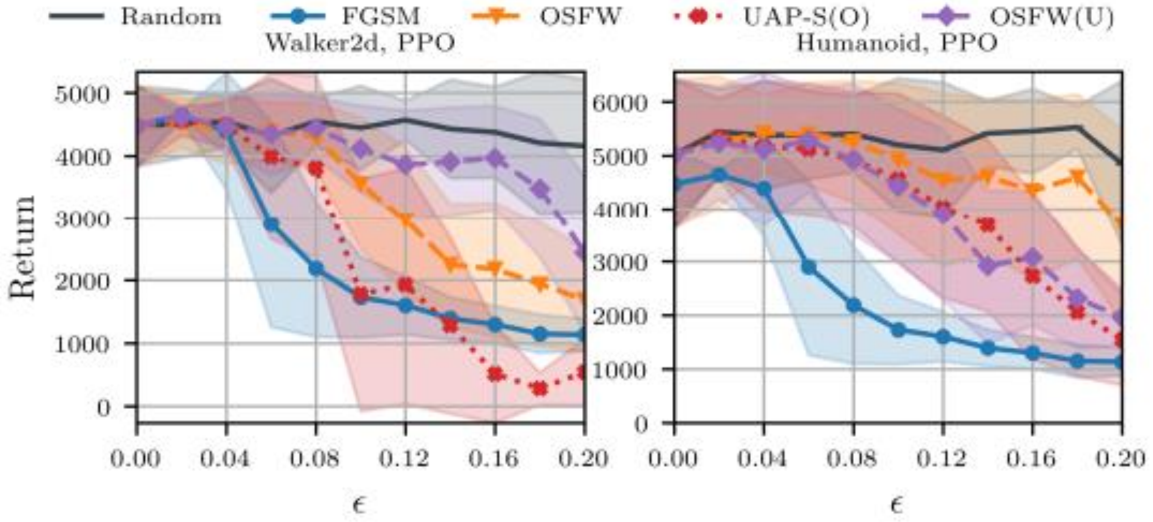
Challenge:

No discrete action space (lack of Q(s,a))

Solution:

- Exploit value function V(s) used in policy
- Modify Algorithm 1 using V(s)
- Goal: Decrease the evaluation of the state

UAP-S and UAP-O generalize to continuous control



Experiment	Attack method	Offline cost$\pm$ std (seconds)	Online cost $\pm$ std (seconds)
Walker2d, PPO, $T_{max} = 0.0079 \pm 10^{-5}$ seconds	FGSM	-	$31 \times 10^{-5} \pm 10^{-5}$
	OSFW	-	0.02 ± 0.001
	UAP-S (O)	8.75 ± 0.024	$2.9 \times 10^{-5} \pm 10^{-6}$
	OSFW(U)	0.02 ± 0.001	$2.9 \times 10^{-5} \pm 10^{-6}$
Humanoid PPO, $T_{max} = 0.0079 \pm 10^{-6}$ seconds	FGSM	-	$35 \times 10^{-5} \pm 10^{-5}$
	OSFW	-	0.02 ± 0.001
	UAP-S (O)	35.86 ± 0.466	$2.4 \times 10^{-5} \pm 10^{-6}$
	OSFW(U)	0.02 ± 0.001	$2.4 \times 10^{-5} \pm 10^{-6}$

Detection and Mitigation of Adversarial Perturbations

Current defenses:

- Hard to apply adversarial example detection techniques
- Traditional adversarial training leads unstable training and performance degradation

Visual Foresight[1] (VF): apply action-conditioned-frame-prediction for detection & recovery

- Ineffective against universal perturbations

SA-MDP[2]: find optimal policy under the worst possible adversary using policy regularization

- Ineffective against bigger perturbations

epsilon	Defense	Average return ± std in the presence of adversarial perturbation attacks: No attack	FGSM	OSFW	UAP-S	UAP-O	OSFW(U)
0.01	No defense	21.0 ± 0.0	**−21.0 ± 0.0**	−20.0 ± 3.0	**−21.0 ± 0.0**	−19.8 ± 0.4	**−21.0 ± 0.0**
	VF [17]	21.0 ± 0.0	21.0 ± 0.0	−19.7 ± 0.5	0.7 ± 1.7	0.4 ± 2.7	**−21.0 ± 0.0**
	SA-MDP [40]	21.0 ± 0.0	21.0 ± 0.0	21.0 ± 0.0	21.0 ± 0.0	21.0 ± 0.0	21.0 ± 0.0
0.02	No defense	21.0 ± 0.0	−19.9 ± 1.3	**−21.0 ± 0.0**	−20.8 ± 0.6	−20.0 ± 0.0	**−21.0 ± 0.0**
	VF [17]	21.0 ± 0.0	21 ± 0.0	−19.7 ± 0.6	9.4 ± 0.8	5.3 ± 3.9	−20.5 ± 0.5
	SA-MDP [40]	21.0 ± 0.0	−14.6 ± 8.8	−20.5 ± 0.5	−20.6 ± 0.5	−20.6 ± 0.5	**−21.0 ± 0.0**
0.05	No defense	21.0 ± 0.0	−20.5 ± 0.7	**−21.0 ± 0.0**	−20.6 ± 0.8	−20.0 ± 0.0	**−21.0 ± 0.0**
	VF [17]	21.0 ± 0.0	21.0 ± 0.0	−20.0 ± 0.0	7.6 ± 4.7	−14.1 ± 1.1	**−21.0 ± 0.0**
	SA-MDP [40]	21.0 ± 0.0	**−21.0 ± 0.0**	**−21.0 ± 0.0**	−20.6 ± 0.5	−20.6 ± 0.5	**−21.0 ± 0.0**

(a) DQN agent playing Pong

epsilon	Defense	Average return ± std in the presence of adversarial perturbation attacks: No attack	FGSM	OSFW	UAP-S	UAP-O	OSFW(U)
0.01	No defense	34.0 ± 0.0	**0.0 ± 0.0**	2.0 ± 1.1	2.1 ± 0.8	4.0 ± 0.6	0.5 ± 0.5
	VF [17]	32.0 ± 1.5	32.6 ± 1.7	24.1 ± 1.0	22.9 ± 0.9	25.8 ± 1.1	**20.9 ± 1.2**
	SA-MDP [40]	30.0 ± 0.0	30.0 ± 0.0	30.0 ± 0.0	30.0 ± 0.0	30.0 ± 0.0	30.0 ± 0.0
0.02	No defense	34.0 ± 0.0	**0.0 ± 0.0**	1.0 ± 0.0	0.1 ± 0.3	0.8 ± 0.6	**0.0 ± 0.0**
	VF [17]	32.0 ± 1.5	32.6 ± 1.7	**1.1 ± 0.3**	24.0 ± 2.0	25.6 ± 1.0	4.4 ± 1.1
	SA-MDP [40]	30.0 ± 0.0	29.8 ± 0.6	29.9 ± 0.3	**29.4 ± 1.2**	**29.4 ± 1.2**	30.0 ± 0.0
0.05	No defense	34.0 ± 0.0	**0.0 ± 0.0**	1.2 ± 0.0	2.2 ± 1.7	2.2 ± 1.4	**0.0 ± 0.0**
	VF [17]	32.0 ± 1.4	32.6 ± 1.6	1.0 ± 0.0	29.0 ± 1.1	23.9 ± 0.3	**0.0 ± 0.0**
	SA-MDP [40]	30.0 ± 0.0	21.1 ± 1.3	**20.9 ± 0.8**	21.1 ± 1.7	21.1 ± 1.7	21.1 ± 1.7

(b) DQN agent playing Freeway

1. Lin, Yen-Chen, et al. *"Detecting adversarial attacks on neural network policies with visual foresight."* arXiv 2017. https://arxiv.org/abs/1710.00814
2. Zhang, Huan, et al. *"Robust deep reinforcement learning against adversarial perturbations on state observations."* NeurIPS2020 https://arxiv.org/abs/2003.08938

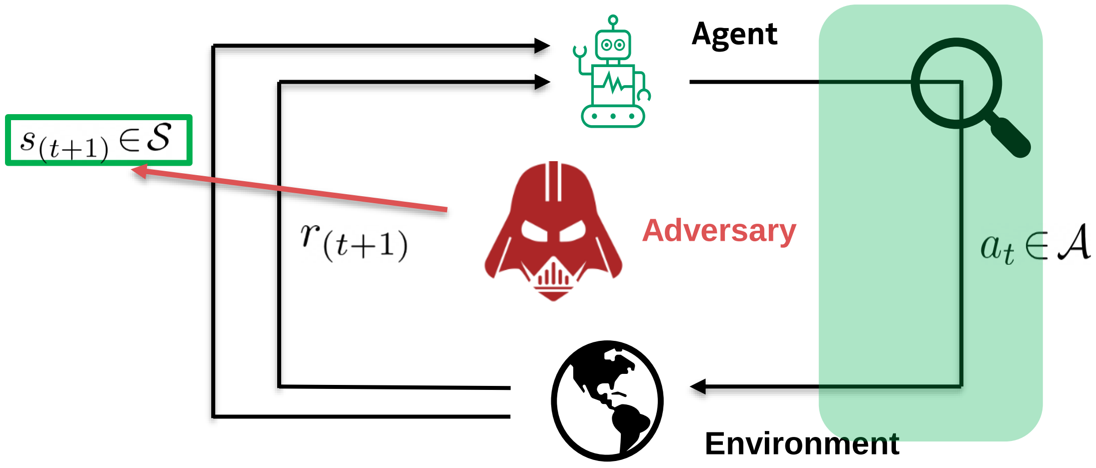
Detection and Mitigation of Adversarial Perturbations

In tasks that can end with clear negative results:

- Losing a game
- Ends episode with negative returns

The victim would be able to suspend/forfeit an episode if the adversary could be detected to prevent the negative outcome

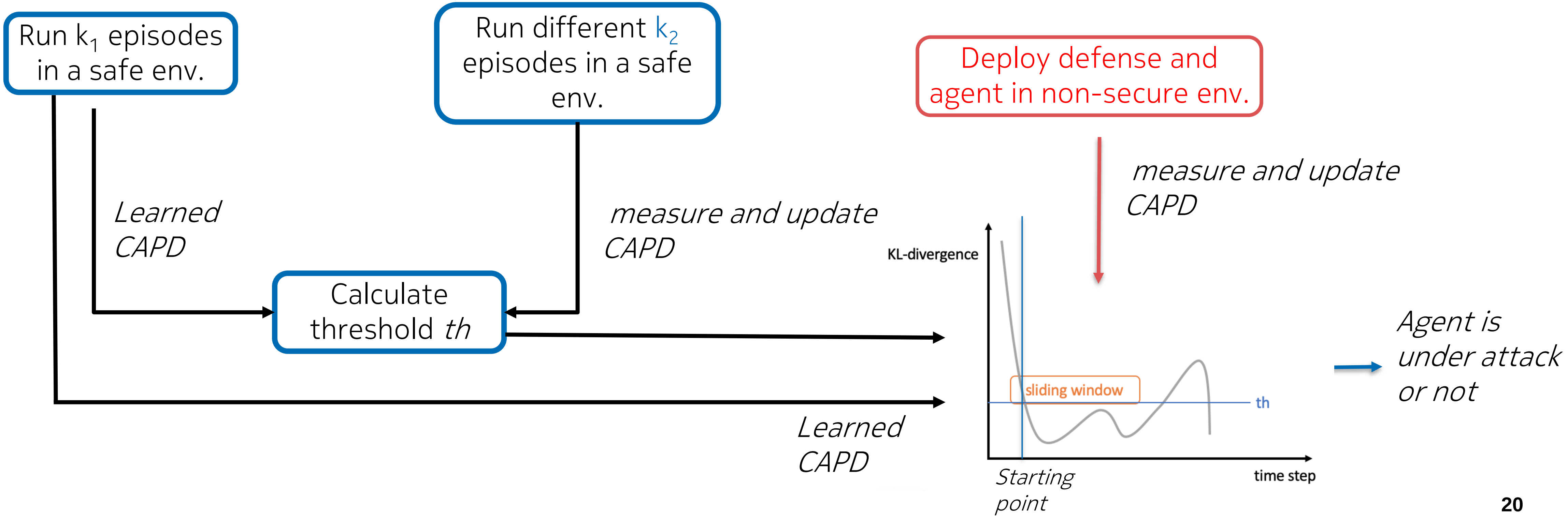
Can we develop an effective detection mechanism that can detect the presence of the adversary?



AD^3 - Action Distribution Divergence Detector

Threshold-based detection method

- Measures statistical distance between the *conditional action probability distributions (CAPD)*



Effectiveness of AD³

- Effective in Pong for all agents against all attacks
- Less effective in Freeway against less effective attacks
- Not effective in Breakout with high false positive rate for DQN and PPO agents
- Useful in raising an alarm when the victim is in the direction of negative return (e.g., losing the game)

False positive rate (FPR) and true positive rate (TPR) of AD^3 against all five attacks. High FPR and low TPR values are in red.

Game	Agent	FPR	TPR FGSM	TPR OSFW	TPR UAP-S	TPR UAP-O	TPR OSFW(U)
Pong	DQN	0.0	1.0	1.0	1.0	1.0	1.0
	A2C	0.0	1.0	1.0	1.0	1.0	1.0
	PPO	0.0	1.0	1.0	1.0	1.0	1.0
Freeway	DQN	0.0	0.8	1.0	1.0	1.0	0.8
	A2C	0.0	1.0	1.0	1.0	1.0	1.0
	PPO	0.0	1.0	0.4	1.0	1.0	1.0
Breakout	DQN	0.6	1.0	0.6	1.0	1.0	1.0
	A2C	0.0	1.0	0.6	1.0	0.8	1.0
	PPO	0.4	1.0	0.4	1.0	0.6	1.0

Losing rate (10 episodes) of DQN agents playing Pong with or without additional defense. Losing rate is calculated by counting the number of games where the computer gains 21 points first in an episode. If AD^3 raises an alarm before an episode ends, then victim does not lose the game. In each row, the best attack with the highest losing rate is in bold, and given an ϵ value, the defense with the highest losing rate for that particular attack is shaded red.

ϵ	Method	Losing Rate No attack	FGSM	OSFW	UAP-S	UAP-O	OSFW(U)
0.01	No defense	0.0	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Visual Foresight[1]	0.0	0.0	**1.0**	0.0	0.2	**1.0**
	SA-MDP[2]	0.0	0.0	0.0	0.0	0.0	0.0
	AD^3	0.0	0.0	0.0	0.0	0.0	0.0
0.02	No defense	0.0	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Visual Foresight[1]	0.0	0.0	**1.0**	0.0	0.3	**1.0**
	SA-MDP[2]	0.0	0.9	**1.0**	**1.0**	**1.0**	**1.0**
	AD^3	0.0	0.0	0.0	0.0	0.0	0.0

1. Lin, Yen-Chen, et al. *"Detecting adversarial attacks on neural network policies with visual foresight."* arXiv 2017. https://arxiv.org/abs/1710.00814
2. Zhang, Huan, et al. *"Robust deep reinforcement learning against adversarial perturbations on state observations."* NeurIPS2020 https://arxiv.org/abs/2003.08938

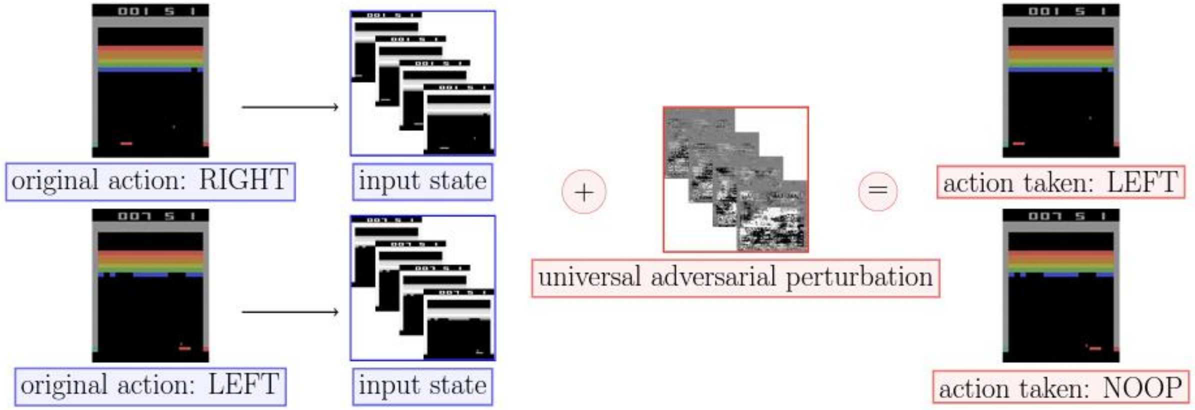
Conclusion and Takeaways

Adversarial models DNNs must be redefined in DRL due to their different innate characteristics

UAP-S and UAP-O: Degrade the performance of deep reinforcement learning agents

- Leverages input-agnostic adversarial perturbation generation methods
- Same effectiveness as state-of-the-art attacks, can be mounted in real time

- **AD^3 : Detects the presence of an adversary**
  - Relies on the temporal coherence of actions (predictable action sequences)
  - Useful to combine with other recovery methods/defenses



https://ssg.aalto.fi/research/projects/
https://crysp.uwaterloo.ca/research/SSG/

batlitekgul@acm.org
buse.atli_tekgul@nokia-bell-labs.com
buseatlitekgul@github.io