Adversarial Sequential Decision Making

Goran Radanović, Adish Singla, Wen Sun, Xiaojin Zhu

International Joint Conference on AI (IJCAI) 2022

MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

25th July, 2022

Outline

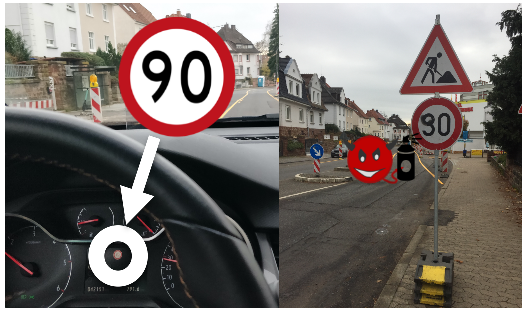
- Preliminaries
- **Test-time Attacks and Defenses in RL**
- Training-time Attacks in RL
- Training-time Defenses in RL
- Adversarial Attacks in Multi-agent RL
- Concluding Remarks

Test-time Attacks: Setup and Basic Ideas

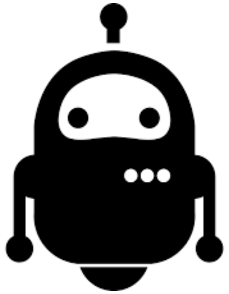
Manipulating Agent's Decisions

- Agent follows a fixed learned policy π
- Adversary manipulates agent's decisions
  - altering the environment's states physically

Driving scenario



Follow a fixed policy π



Observe state s_t

Take action $a_t \sim \pi(\cdot\,|s_t)$

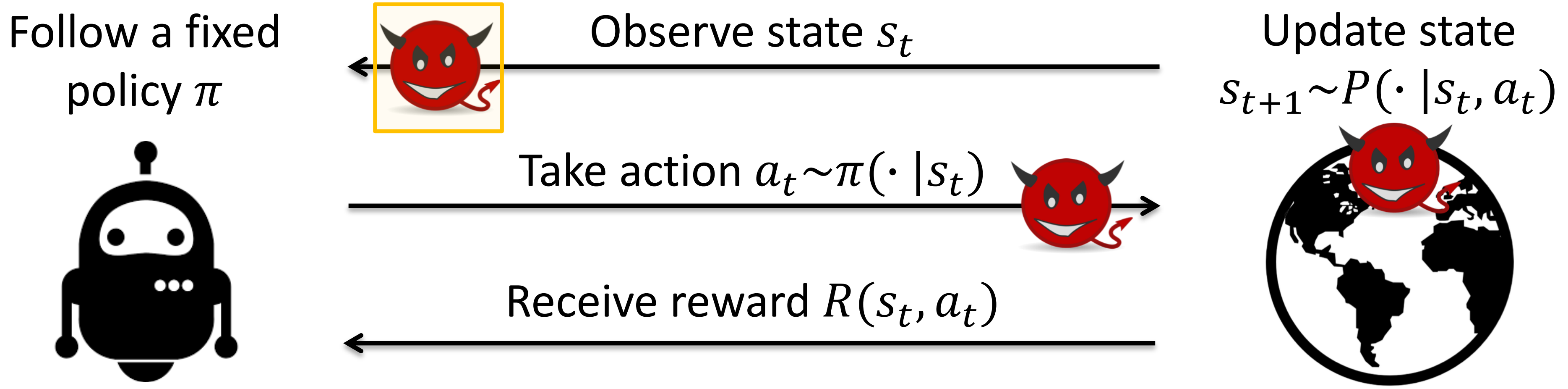
Receive reward $R(s_t, a_t)$

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

Test-time Attacks: Setup and Basic Ideas

Manipulating Agent's Decisions

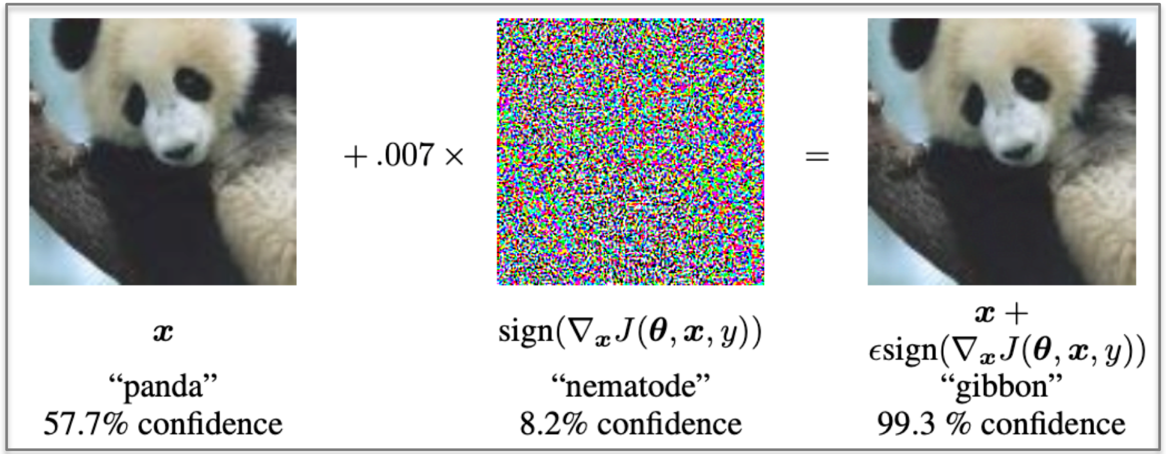
- Agent follows a fixed learned policy π
- Adversary manipulates agent's decisions
  - altering the environment's states physically
  - hacking the actions taken by the agent
  - perturbing the agent's state observations



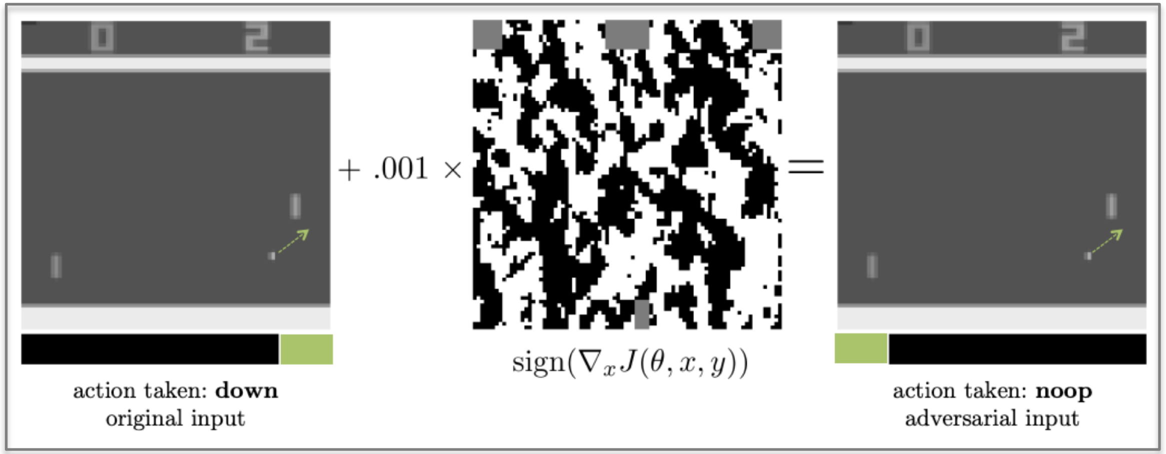
Test-time Attacks: Setup and Basic Ideas

Perturbing State Observations via Adversarial Examples

- Image classification: label “panda” → label “gibbon”
- Pong game: action “down” → action “noop”



[Goodfellow et al., 2015]



[Huang et al., 2017]

Test-time Attacks: Uniform Attack

Problem Setup *[Huang et al., 2017]*

- Perturb state observations at each time step t independently
- Consider each time step t as a multi-class classification problem: $a_t \sim \pi(\cdot \mid s_t)$
- Perturb state observation by crafting adversarial example: $s_t' = s_t + \eta_t$

Test-time Attacks: Uniform Attack

Crafting Adversarial Example at Time Step t [Huang et al., 2017]

- When using Fast Gradient Sign Method (FGSM) with ℓ_∞-norm, we get

$$s'_t = s_t + \epsilon \cdot \text{sign}\big(\nabla_x J(\theta, x, y)\big)$$

where

- θ: Parameters of trained neural network policy π_θ
- x: State s_t at time step t
- y: Action weights based on the distribution $\pi_\theta(\cdot\,|s_t)$
- J: Cross-entropy loss between y and highest-weighted action in y
- ϵ: ℓ_∞-norm constraint

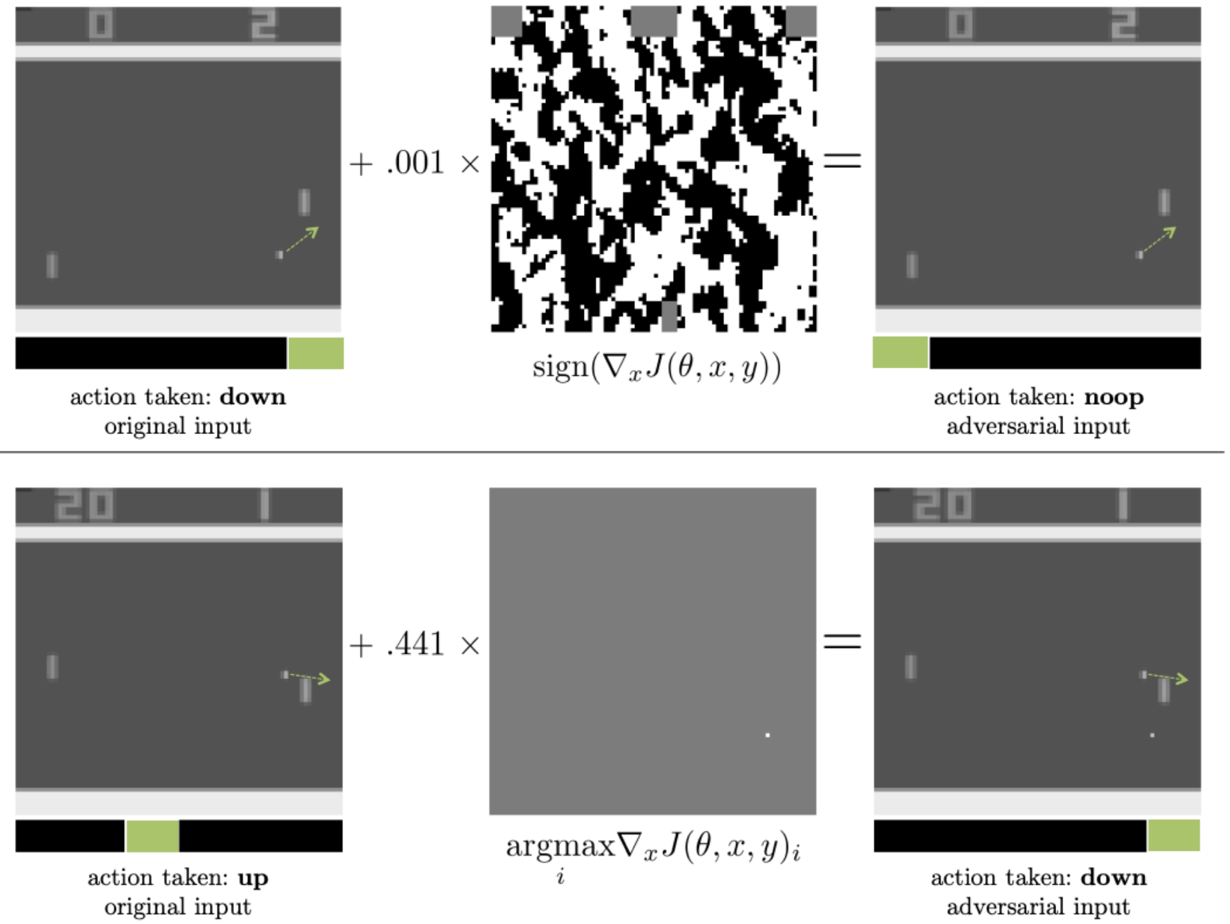
Test-time Attacks: Uniform Attack

Crafting Adversarial Examples in Pong Game

[Huang et al., 2017]

FGSM with ℓ_∞-norm

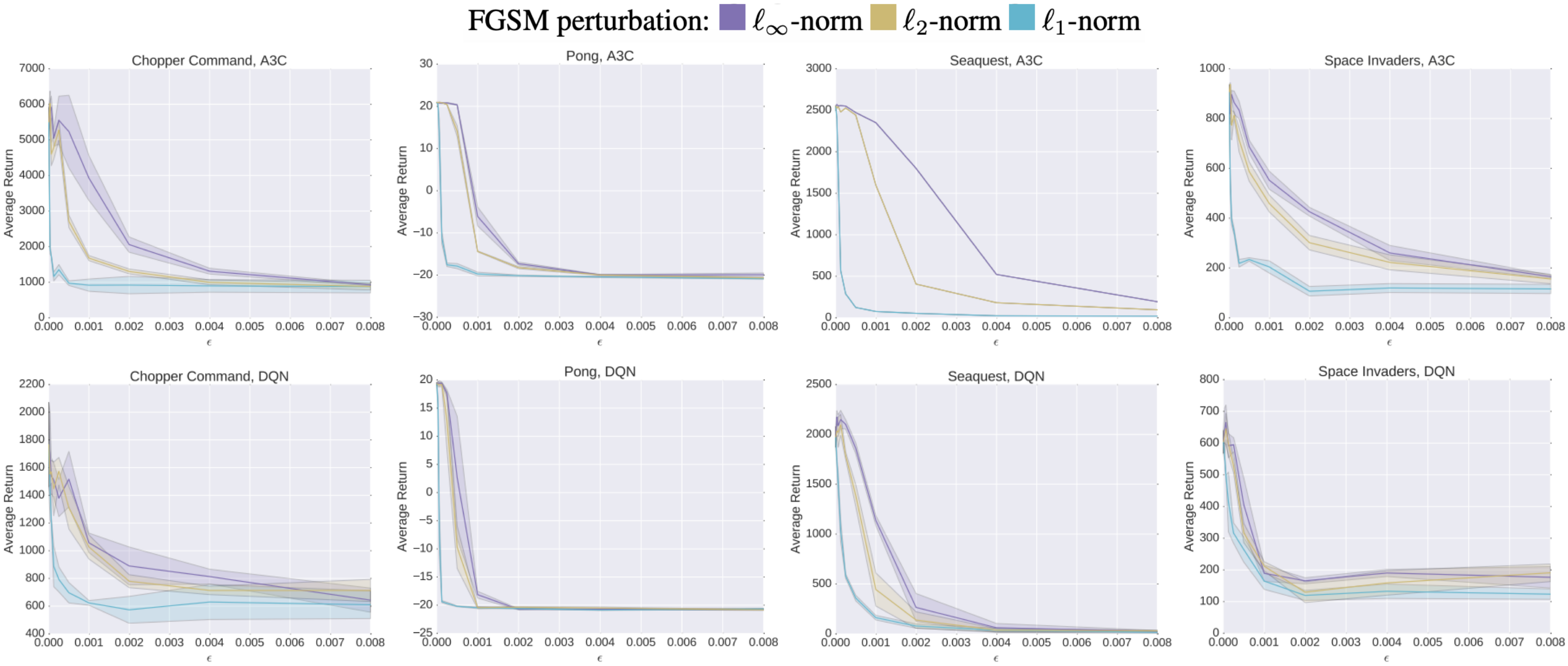
FGSM with ℓ_1-norm



Test-time Attacks: Uniform Attack

Experimental Results: White-box Setting

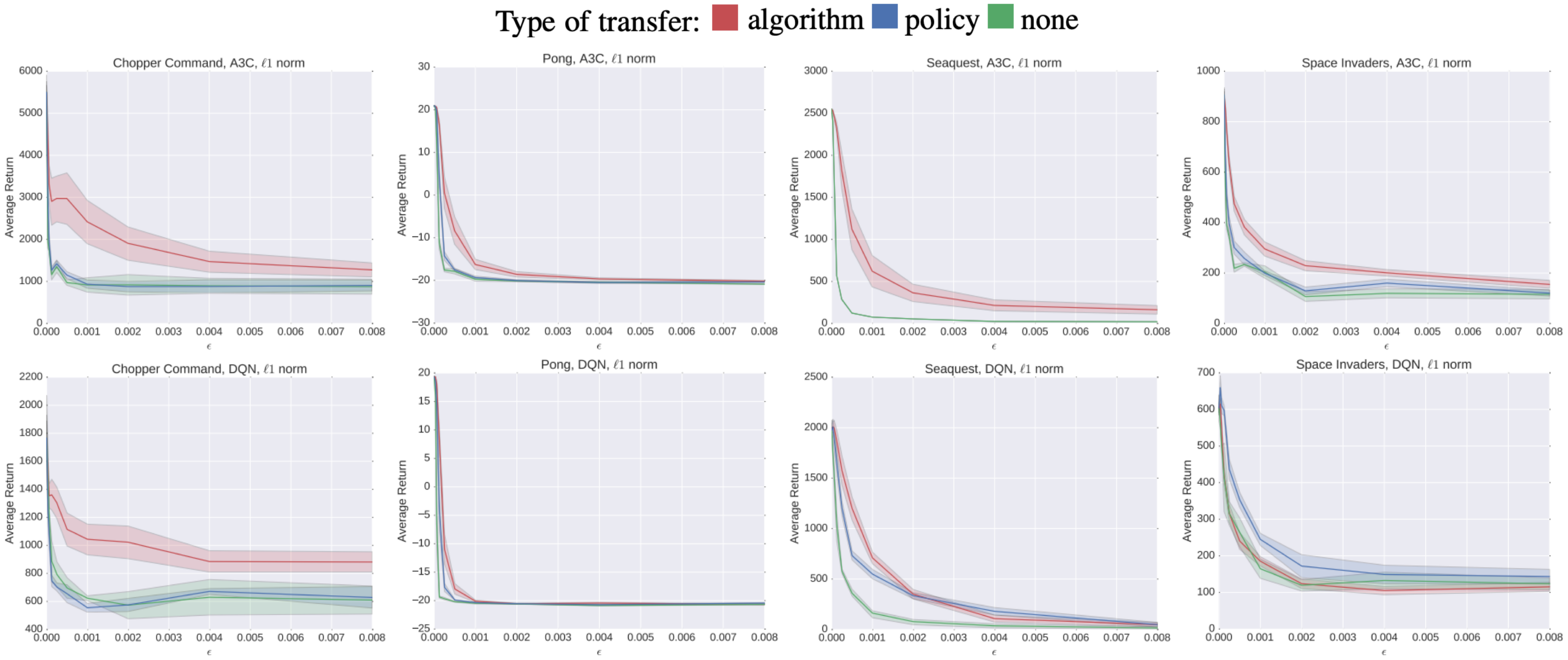
[Huang et al., 2017]



Test-time Attacks: Uniform Attack

Experimental Results: Black-box Setting

[Huang et al., 2017]



Test-time Attacks: Uniform Attack

Limitations of the Uniform Attack Strategy

[Huang et al., 2017]

- Lacks crucial characteristics of sequential decision making
  - Make agent take actions different from π, i.e., $\sum_t \mathrm{I}\{a_t \neq \mathrm{argmax}_a\, \pi(a|s_t)\}$
  - Incurs "attack cost" at every time step, i.e., $T \cdot \epsilon$

Test-time Attacks in Sequential Decision Making

- Adversary's goal
  - Reduce the expected total rewards of the agent
  - Make agent follow a targeted behavior
- Adversary's cost
  - Reduce the attack cost by only perturbing at critical points
  - Optimize the attack cost by long-term planning

Test-time Attacks: Strategically-timed Attack

Problem Setup *[Lin et al., 2017]*

- Adversary's goal
  - Reduce the expected total rewards of the agent, i.e., reduce $\sum_t R(s_t, a_t)$
- Adversary's cost
  - Reduce the attack cost by only perturbing at critical points, i.e., reduce $\sum_t \mathrm{I}\{s_t' \neq s_t\}$

Test-time Attacks: Strategically-timed Attack

Strategically-timed Attack: Optimization Problem *[Lin et al., 2017]*

- Select a subset of time steps to attack, given by variables $b_t \in \{0,1\}$
- Craft a sequence of pertubations for selected time steps, given by variables η_t
- We can formulate the above intuition in the following problem

$$\min_{b_0, b_1, \ldots, b_{T-1}, \eta_0, \eta_1, \ldots, \eta_{T-1}} \mathbb{E}\left[\sum_t R(s_t, a_t) \mid s_{t+1} \sim P(\cdot \mid s_t, a_t), a_t \sim \pi(\cdot \mid s_t'), s_0 \sim \mu(\cdot)\right]$$

$$\boxed{\begin{array}{ll} b_t \in \{0,1\} & \text{for all } t = 0, 1, \ldots, T-1 \\ \sum_t b_t \leq B & \end{array}}$$

When-to-Attack

$$\boxed{s_t' = s_t + b_t \cdot \eta_t \quad \text{for all } t = 0, 1, \ldots, T-1}$$

How-to-Attack

Test-time Attacks: Strategically-timed Attack

Strategically-timed Attack: When-to-Attack *[Lin et al., 2017]*

- Quantify relative preference of actions for a state $c: S \to \mathbb{R}_+$
  - For policy gradient-based methods, define $c(s) = \max_a \pi(a|s) - \min_b \pi(b|s)$
  - For value-based methods such as DQN, we can define $c(s)$ using softmax over Q values

- Higher value of $c(s_t)$ indicates criticality of time step t
  - Given a threshold β (based on the budget B), set $b_t = 1$ if $c(s_t) \geq \beta$

Test-time Attacks: Strategically-timed Attack

Strategically-timed Attack: How-to-Attack at Critical t

[Lin et al., 2017]

- Define $a_t^{\text{most}} = \max_a \pi(a|s_t)$ and $a_t^{\text{least}} = \min_a \pi(a|s_t)$

- Craft adversarial example using "targeted" attack method
  - Set a_t^{least} as the target label
  - Find η_t under a norm constraint that increases $\pi\left(\mathrm{a}_\mathrm{t}^{\text{least}}|s_t + \eta_t\right)$

Test-time Attacks: Strategically-timed Attack

Strategically-timed Attack Against a Policy Playing Pong

[Lin et al., 2017]

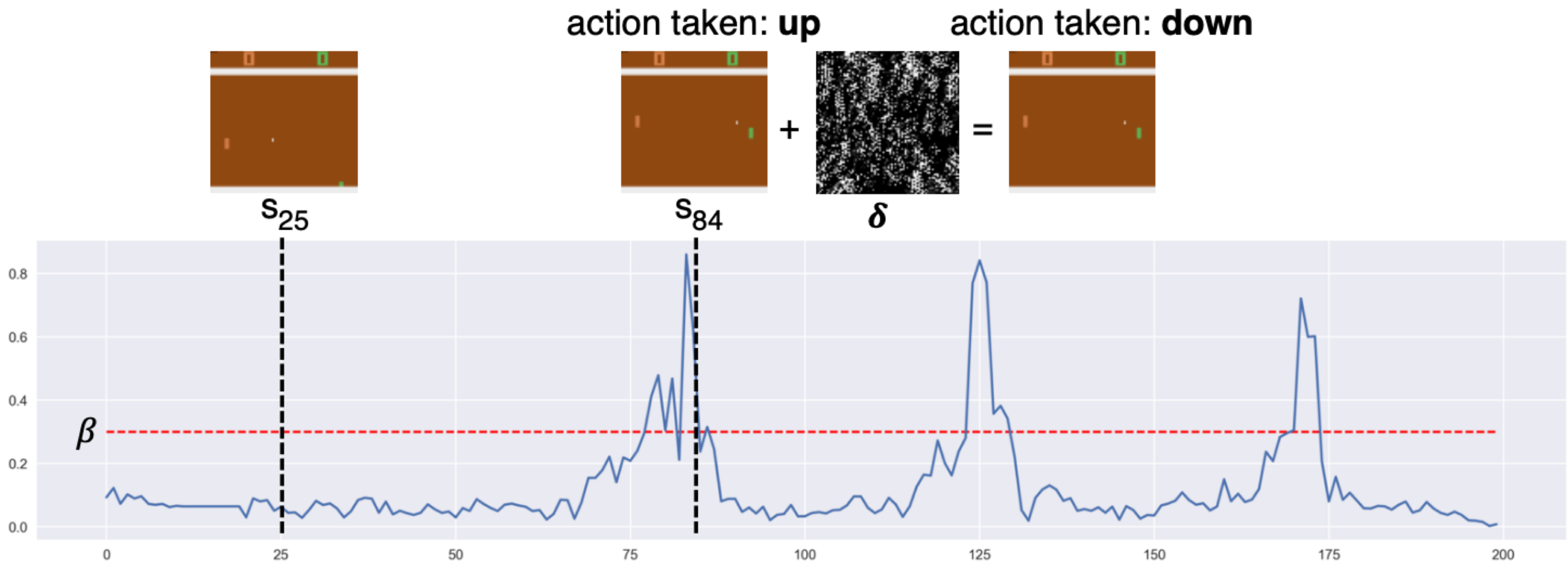
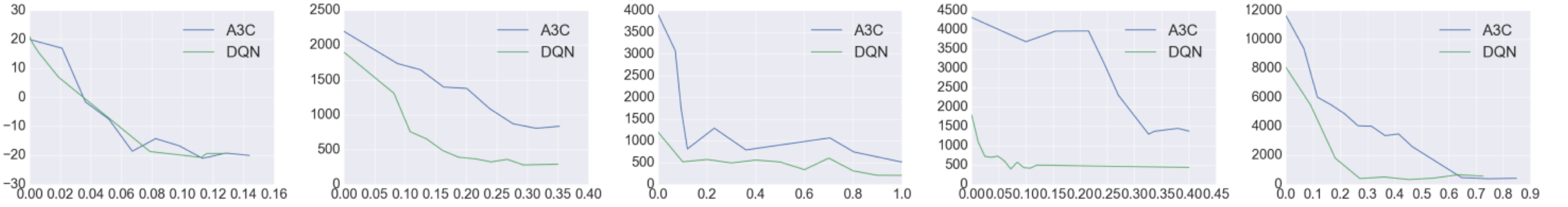


Figure 1: Illustration of the strategically-timed attack on Pong. We use a function c to compute the preference of the agent in taking the most preferred action over the least preferred action at the current state s_t. A large preference value implies an immediate reward. In the bottom panel, we plot $c(s_t)$. Our proposed strategically-timed attack launch an attack to a deep RL agent when the preference is greater than or equal to a threshold, $c(s_t) \geq \beta$ (red-dash line). When a small perturbation is added to the observation at s_{84} (where $c(s_{84}) \geq \beta$), the agent changes its action from up to down and eventually misses the ball. But when the perturbation is added to the observation at s_{25} (where $c(s_{25}) < \beta$), there is no impact to the reward.

Test-time Attacks: Strategically-timed Attack

Experimental Results: Policies Trained with A3C and DQN Methods

[Lin et al., 2017]



(a) Pong (b) Seaquest (c) MsPacman (d) ChopperCommand (e) Qbert

Figure 3: Accumulated reward (y-axis) v.s. Portions of time steps the agent is attacked (x-axis) of Strategically-timed Attack in 5 games. The blue and green curves correspond to results of A3C and DQN, respectively. A larger reward means the deep RL agent is more robust to the strategically-timed attack.

- Comparison with Uniform attack strategy on Pong game
  - Strategically-timed attack achieves lowest reward with perturbation of only 15% time steps
  - Uniform attack achieves lowest reward with perturbation of 100% time steps

Test-time Attacks: Strategically-timed Attack

Limitations of the Strategically-timed Attack Strategy

[Lin et al., 2017]

- Adversary’s goal
  - Reduce the expected total rewards of the agent
  - Make agent follow a targeted behavior
- Adversary’s cost
  - Reduce the attack cost by only perturbing at critical points
  - Optimize the attack cost by long-term planning

Test-time Attacks: Trained Adversary

[Tretschk et al., 2018; Sun et al., 2020, Zhang et al., 2020, 2021; Sun et al., 2022]

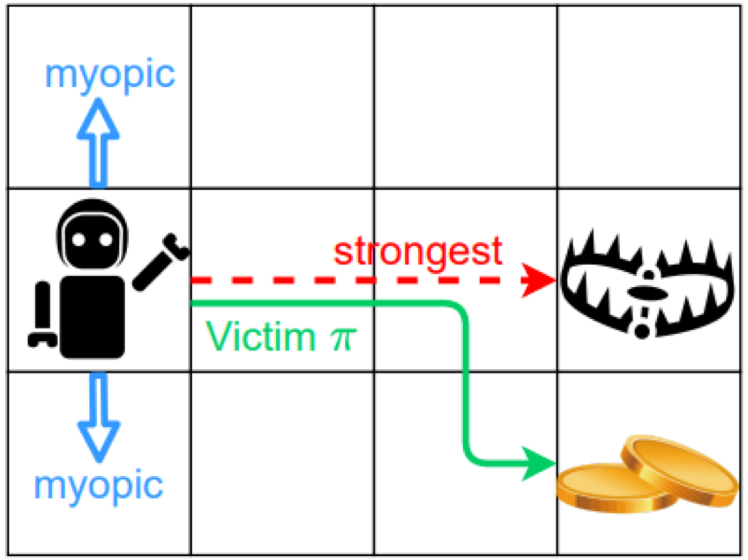
Problem Setup

- Adversary's goal
  - Make agent follow a targeted behavior
    - Minimize the expected total rewards of the agent, i.e., minimize $\sum_t R(s_t, a_t)$
    - Maximize the expected total rewards of the adversary, i.e., maximize $\sum_t \hat{R}(s_t, a_t)$
    - Make agent reach a desired set of goal states, i.e., $s_T \in S^{\text{adversary}}$
- Adversary's cost
  - Optimize the attack cost by long-term planning

Test-time Attacks: Trained Adversary

An Example Scenario for Evasion Attack

- Adversary's goal is to minimize the expected total rewards of the agent
- The scenario shows that myopic adversary is sub-optimal



[Sun et al., 2022]

Test-time Attacks: Trained Adversary

Adversary's MDP $\widehat{\mathcal{M}}$

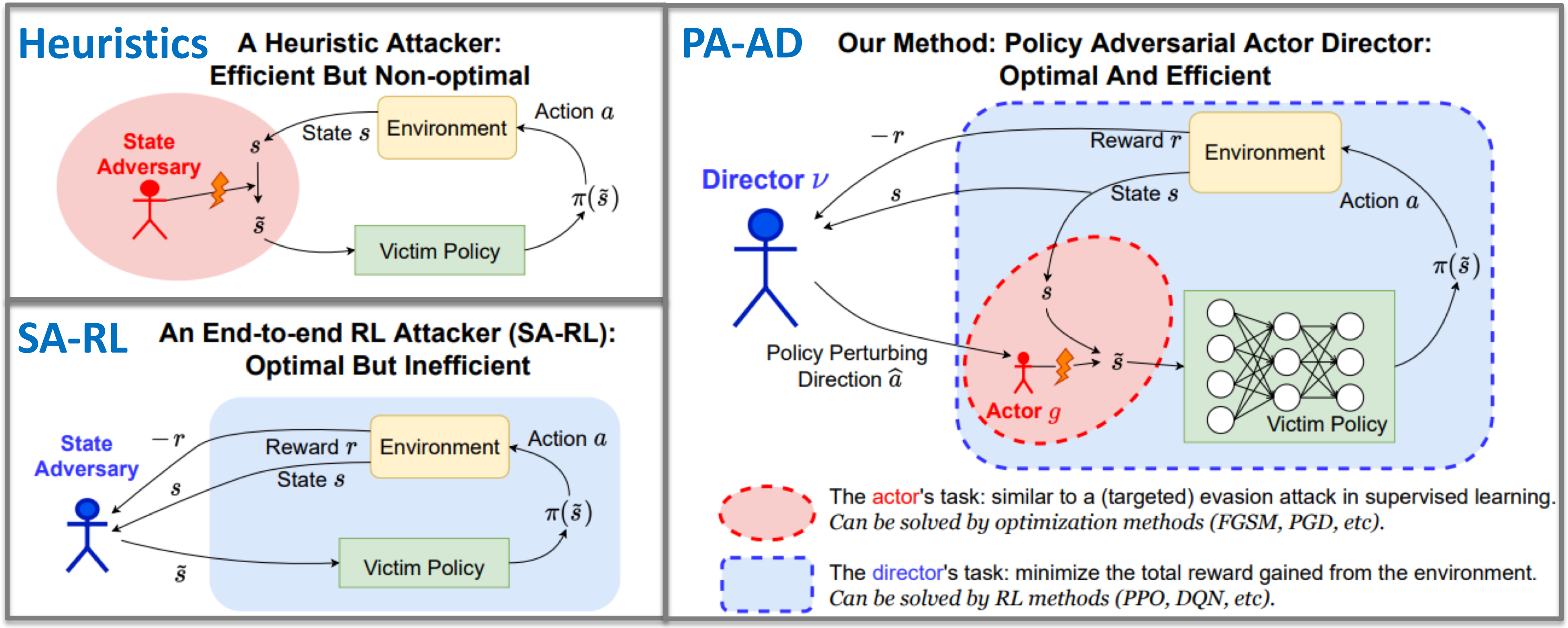
- Given the following
  - Agent's MDP $\mathcal{M} \coloneqq (\mathcal{S}, \mathcal{A}, P, R, \gamma, \mu)$ and agent's fixed policy π
  - Method $F(\pi, \epsilon, s)$ for crafting adversarial examples
    - ϵ is norm constraint on maximum allowed perturbation at any time step

- Adversary's MDP $\widehat{\mathcal{M}} = \left(\mathcal{S}, \hat{\mathcal{A}}, \hat{P}, \hat{R}, \gamma, \mu\right)$
  - Reward function $\hat{R}$ encodes the adversary's goal and cost
    - $\hat{R}(s_t, a_t) = -R(s_t, a_t)$ for evasion attacks *[Zhang et al., 2020, 2021; Sun et al., 2022]*
    - $\hat{R}(s_t, a_t) = -R(s_t, a_t) - \lambda \cdot \mathrm{I}\{s_t' \neq s_t\}$ for evasion attacks with cost considerations
  - Action space $\hat{\mathcal{A}}$ defines the "learning" aspect of adversary
    - For a given state s, $\hat{\mathcal{A}}_s \subseteq \mathcal{S}$ is the set of permissible state pertubations *[Zhang et al., 2020, 2021]*
    - For a given state s, $\hat{\mathcal{A}}_s \subseteq \Delta_{\mathcal{A}}$ is the set of permissible action distributions *[Sun et al., 2022]*

Test-time Attacks: Trained Adversary

Adversary's Action Space $\hat{\mathcal{A}}$: SA-RL vs. PA-AD Methods

- **SA-RL**: $\hat{\mathcal{A}}_S \subseteq \mathcal{S}$ is the set of permissible state pertubations *[Zhang et al., 2020, 2021]*
- **PA-AD**: $\hat{\mathcal{A}}_S \subseteq \Delta_{\mathcal{A}}$ is the set of permissible action distributions *[Sun et al., 2022]*



Test-time Attacks: Trained Adversary

Experimental Results: PA-AD vs. Baselines

[Sun et al., 2022]

	Environment	Natural Reward	ϵ	Random	Uniform	SA-RL	PA-AD
DQN	**Boxing**	96 ± 4	0.001	95 ± 4	53 ± 16	94 ± 6	$\mathbf{19 \pm 11}$
	Pong	21 ± 0	0.0002	21 ± 0	-10 ± 4	20 ± 1	$\mathbf{-21 \pm 0}$
	RoadRunner	46278 ± 4447	0.0005	44725 ± 6614	17012 ± 6243	43615 ± 7183	$\mathbf{0 \pm 0}$
	Freeway	34 ± 1	0.0003	34 ± 1	12 ± 1	34 ± 1	$\mathbf{9 \pm 1}$
	Seaquest	10650 ± 2716	0.0005	8177 ± 2962	3820 ± 1947	8152 ± 3113	$\mathbf{2304 \pm 838}$
	Alien	1623 ± 252	0.00075	1650 ± 381	819 ± 486	1693 ± 439	$\mathbf{256 \pm 210}$
	Tutankham	227 ± 29	0.00075	221 ± 65	30 ± 13	202 ± 65	$\mathbf{0 \pm 0}$
A2C	**Breakout**	356 ± 79	0.0005	355 ± 79	86 ± 104	353 ± 79	$\mathbf{44 \pm 62}$
	Seaquest	1752 ± 70	0.005	1752 ± 73	356 ± 153	1752 ± 71	$\mathbf{4 \pm 13}$
	Pong	20 ± 1	0.0005	20 ± 1	-4 ± 8	20 ± 1	$\mathbf{-13 \pm 6}$
	Alien	1615 ± 601	0.001	1629 ± 592	1062 ± 610	1661 ± 625	$\mathbf{507 \pm 278}$
	Tutankham	258 ± 53	0.001	260 ± 54	139 ± 26	260 ± 54	$\mathbf{71 \pm 47}$
	RoadRunner	34367 ± 6355	0.002	35851 ± 6675	9198 ± 3814	36550 ± 6848	$\mathbf{2773 \pm 3468}$

Test-time Attacks: Stronger Attacks?

Optimality of the Trained Adversary

- Specific threat model and assumptions
  - Adversary perturbs the state observations at test-time
  - Agent’s policy is fixed
- SA-RL and PA-AD methods provide a framework to train optimal adversaries

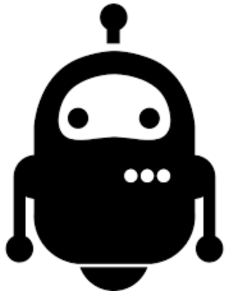
Stronger Test-time Attacks with Backdoor Policies

- Adversary has some control over the agent’s training process
  - Inject backdoors in the agent’s policy, e.g., using reward poisoning *[Kiourti et al., 2020]*
  - Test-time attacks reduce to crafting triggerers for the backdoor policy

Test-time Defenses: Setup and Basic Ideas

Defense Against Test-time Attacks

- Data: Augment training data with adversarial manipulations
- Algorithm: Regularized objective functions for training
- Inference: Robustify inference via smoothing techniques



[Pattanaik et al., 2018; Zhang et al., 2021]

Data →

[Zhang et al., 2020; Oikarinen et al., 2021]

Algorithm →

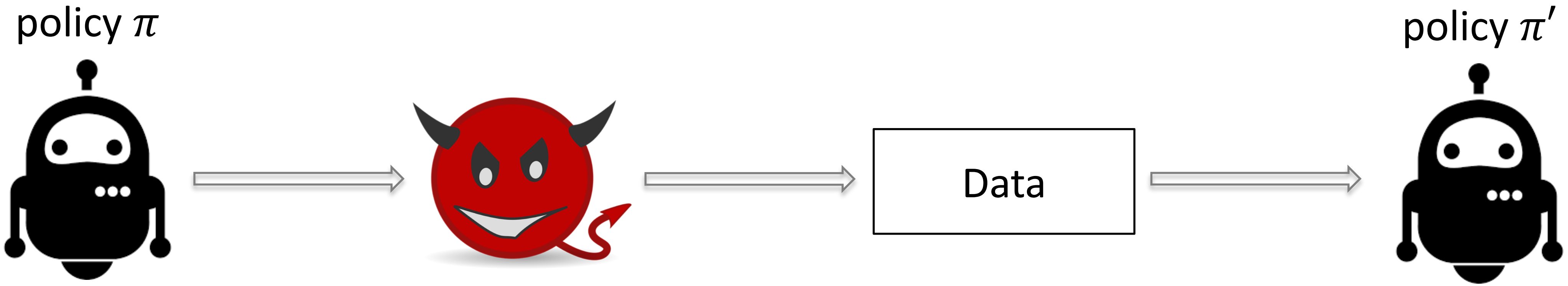
[Wu et al., 2022]

Inference

Test-time Defenses: Augment Training Data

Augment Training Data with Adversarial Manipulations

- Static adversary
  - For a fixed π, use an adversary against π to generate data *[Pattanaik et al., 2018]*



Test-time Defenses: Augment Training Data

Augment Training Data with Adversarial Manipulations

- Static adversary
  - For a fixed π, use an adversary against π to generate data *[Pattanaik et al., 2018]*
- Non-static adversary
  - ALTA: Alternating training with learned adversaries *[Zhang et al., 2021]*

Test-time Attacks: Experimental Results

Experimental Results: ATLA vs. Baselines

[Zhang et al., 2021]

Env.	State Dimension	ℓ_∞ norm perturbation budget ϵ	Method	Natural Reward	Best Attack
Hopper	11	0.075	PPO (vanilla)	3167±542	636± 9
			SA-PPO (Zhang et al., 2020b)	3705± 2	1076± 791
			Pattanaik et al. (2018)	2755±582	291± 7
			ATLA-PPO (MLP)	2559 ± 958	976± 40
Walker2d	17	0.05	PPO (vanilla)	4472 ± 635	1086±516
			SA-PPO (Zhang et al., 2020b)	4487± 61	2908± 1136
			Pattanaik et al. (2018)	4058± 1410	733± 1012
			ATLA-PPO (MLP)	3138 ± 1061	2213± 915
Ant	111	0.15	PPO (vanilla)	5687 ± 758	-872 ± 436
			SA-PPO (Zhang et al., 2020b)	4292± 384	2511 ± 1117
			Pattanaik et al. (2018)	3469± 1139	-672± 100
			ATLA-PPO (MLP)	4894± 123	33±327
HalfCheetah	17	0.15	PPO (vanilla)	7117± 98	-660± 218
			SA-PPO (Zhang et al., 2020b)	3632± 20	3028 ±23
			Pattanaik et al. (2018)	5241± 1162	447± 192
			ATLA-PPO (MLP)	5417± 49	2170± 2097

Test-time Attacks: Experimental Results

Experimental Results: ATLA with LSTM Policies + Regularized Objective

[Zhang et al., 2021]

Env.	State Dimension	ℓ_∞ norm perturbation budget ϵ	Method	Natural Reward	Best Attack
Hopper	11	0.075	PPO (vanilla)	3167±542	636± 9
			SA-PPO (Zhang et al., 2020b)	3705± 2	1076± 791
			Pattanaik et al. (2018)	2755±582	291± 7
			ATLA-PPO (MLP)	2559 ± 958	976± 40
			PPO (LSTM)	3060± 639.3	784± 48
			ATLA-PPO (LSTM)	3487± 452	1224± 191
			ATLA-PPO (LSTM) +SA Reg	3291± 600	**1772± 802**
Walker2d	17	0.05	PPO (vanilla)	4472 ± 635	1086±516
			SA-PPO (Zhang et al., 2020b)	4487± 61	2908± 1136
			Pattanaik et al. (2018)	4058± 1410	733± 1012
			ATLA-PPO (MLP)	3138 ± 1061	2213± 915
			PPO (LSTM)	2785± 1121	1259± 937
			ATLA-PPO (LSTM)	3920± 129	3219 ± 1132
			ATLA-PPO (LSTM) +SA Reg	3842± 475	**3239± 894**
Ant	111	0.15	PPO (vanilla)	5687 ± 758	-872 ± 436
			SA-PPO (Zhang et al., 2020b)	4292± 384	2511 ± 1117
			Pattanaik et al. (2018)	3469± 1139	-672± 100
			ATLA-PPO (MLP)	4894± 123	33±327
			PPO (LSTM)	5696 ± 165	-513 ± 104
			ATLA-PPO (LSTM)	5612± 130	716± 256
			ATLA-PPO (LSTM) +SA Reg	5359±153	**3765± 101**
HalfCheetah	17	0.15	PPO (vanilla)	7117± 98	-660± 218
			SA-PPO (Zhang et al., 2020b)	3632± 20	3028 ±23
			Pattanaik et al. (2018)	5241± 1162	447± 192
			ATLA-PPO (MLP)	5417± 49	2170± 2097
			PPO (LSTM)	5609± 98	-886± 30
			ATLA-PPO (LSTM)	5766 ± 109	2485± 1488
			ATLA-PPO (LSTM) +SA Reg	6157± 852	**4806± 603**

Test-time Defenses: Stronger Defenses?

Stronger Defenses

- Obtaining provable guarantees of the agent's performance
- Considering more powerful threat models
  - Defense against test-time attacks with backdoor policies

References

- Goodfellow et al., Explaining and Harnessing Adversarial Examples, 2015.
- Huang et al., Adversarial Attacks on Neural Network Policies, 2017.
- Lin et al., Tactics of Adversarial Attack on Deep Reinforcement Learning Agents, 2017.
- Tretschk et al., Sequential Attacks on Agents for Long-Term Adversarial Goals, 2018.
- Sun et al., Stealthy and Efficient Adversarial Attacks against Deep Reinforcement Learning, 2020.
- Zhang et al., Robust Deep Reinforcement Learning against Adversarial Perturbations on State Observations, 2020.
- Zhang et al., Robust Reinforcement Learning on State Observations with Learned Optimal Adversary, 2021.
- Sun et al., Who Is the Strongest Enemy? Towards Optimal and Efficient Evasion Attacks in Deep RL, 2022.
- Kiourti et al., TrojDRL: Trojan Attacks on Deep Reinforcement Learning Agents, 2020.
- Pattanaik et al., Robust Deep Reinforcement Learning with Adversarial Attacks, 2018.
- Oikarinen et al., Robust Deep Reinforcement Learning through Adversarial Loss, 2021.
- Wu et al., CROP: Certifying Robust Policies for Reinforcement Learning through Functional Smoothing, 2022.

Outline

- Preliminaries
- **Test-time Attacks and Defenses in RL**
- Training-time Attacks in RL
- Training-time Defenses in RL
- Adversarial Attacks in Multi-agent RL
- Concluding Remarks