IOWA STATE UNIVERSITY
Department of Mechanical Engineering; Department of Computer Science

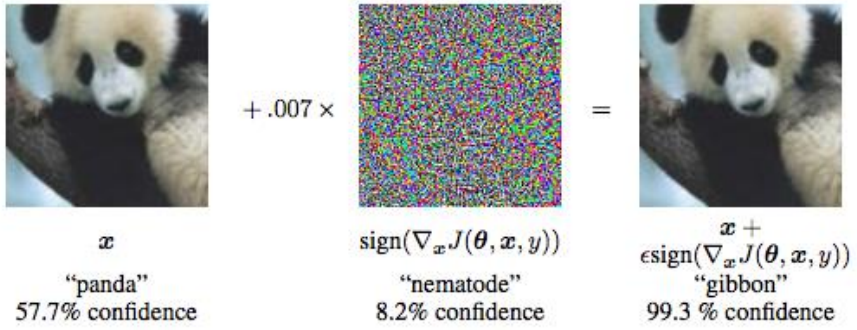
Robust Machine Learning

Soumik Sarkar
Associate Professor
Department of Mechanical Engineering
Department of Computer Science (courtesy)
Iowa State University

MDL
September 4, 2020

Adversarial attacks on Deep learning models

- Deep learning is becoming a tool of choice for many autonomous perception and decision-making tasks, e.g., in self-driving cars
- However, vulnerabilities of deep learning models are alarming!
- Certification requires clear understanding of the vulnerabilities



Synthetic attacks: FGSM and variants

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "*Explaining and harnessing adversarial examples*" arXiv preprint arXiv:1412.6572, 2014.

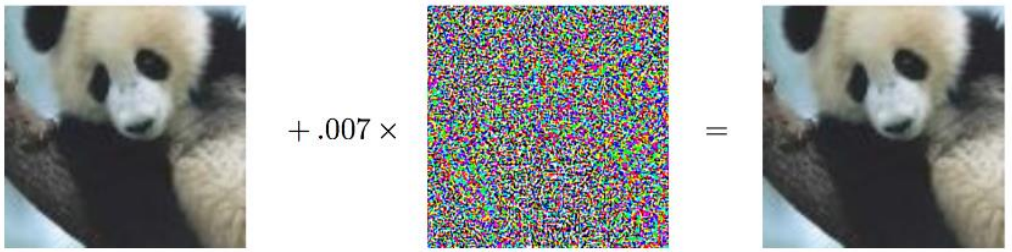
Physical world attacks

Eykholt, Kevin, et al. "*Robust Physical-World Attacks on Deep Learning Visual Classification*", **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. 2018.

- **Attacks** – white box vs. black box, test time vs. train time, mostly gradient based
- **Defenses** – adversarial training, robust optimization

Pixel-space imperceptible attacks

Neural networks are vulnerable to adversarial attacks which are basically the inputs that are almost not distinguishable from the input data on which the model has been trained, but are classified incorrectly.



The above figure shows that a panda which is classified as 'panda' correctly by 57% confidence by a deep neural network model, after the addition of some Fast Gradient Sign Attack is classified incorrectly as a 'gibbon' whereas it is still a panda.

Goodfellow, I., Shlens, J. and Szegedy, C. (2019). *Explaining and Harnessing Adversarial Examples*. [online] arXiv.org. Available at: https://arxiv.org/abs/1412.6572 [Accessed 20 Feb. 2019].

Formulation of gradient based attacks

- One of the first methods is Fast Gradient Sign Method (FGSM) which can be written as:

$$x = x + \varepsilon \operatorname{sgn}(\nabla_x L(\theta, x, y))$$

In which L is the loss function, ε is the L_∞ norm bound of the perturbations , x is the input data, y is the data label and θ s are the weights in the neural network.

- One of the most popular attack formulation - projected gradient descent (PGD) approach – an iterative optimization-based variant of FGSM

$$x = x + rand + lr * \operatorname{sgn}(\nabla_x L(\theta, x, y))$$

In which rand is a random variable between $-\varepsilon$ and ε, and lr is the step size.

Attack categories

- In terms of how much access the attacker has to the model attacks can be categorized into 3 different classes:

  ❖ White box attacks:
  The attacker has access to the full model

  ❖ Black box attack:
  The attacker only has access to the query and doesn't know the model

  ❖ Grey box attack
  The attacker knows the architecture, takes a virtual model and performs a white box attack

- Study of black box attack shows the notion of transferability which means attacks can transfer from one model to another and still be capable of fooling the network.

Attack categories

- In terms of where the attacks can happen adversarial attacks are classified into two different classes:

  - ❖ Attacks in Training set:

    Attackers inject malicious data into the training set to subvert the normal operation of deep neural networks

  - ❖ Attacks in Test set:

    Injecting the adversaries in the test set to break the trained network

Attack categories

- In terms of whether the attackers target a specific model attacks can be classified in two different classes:

  ❖ Untargeted attacks:

  An untargeted adversary can be defined as $\mathrm{A}(X, M) \rightarrow \bar{X}$, where A(.) is the adversarial function, X is the input image, $\bar{X}$ is the adversarial example, and M is the target model. A is considered successful if $\mathrm{M}(X) \not\equiv M(\bar{X})$. Recent studies in this area came up with universal attacks that could satisfy the above property.

  ❖ Targeted attacks:
  A targeted adversary can be defined as $\mathrm{A}(X, M, l) \rightarrow \bar{X}$, where l is an additional target label, and A is only considered successful if $\mathrm{M}(\bar{X}) = l$.

[source: https://arxiv.org/pdf/1612.00410.pdf]

Defense strategy: Adversarial training

Train with both normal and adversarial examples

$$L = \frac{1}{(m-k)+\lambda k}\left(\sum_{i \in CLEAN} L(x_i | y_i) + \lambda \sum_{i \in ADV} L\left(x_i^{adv} | y_i\right)\right)$$

$m \rightarrow$ minibatch $k \rightarrow$ no. of adv. examples

$\lambda \rightarrow$ weighting

$\epsilon \sim N(\mu, \sigma)$

If ϵ is fixed then Robust model only works for that ϵ

https://arxiv.org/abs/1611.01236

http://www.dsrg.stuorg.iastate.edu/fall-2018-talks/

Defense strategy: Robust Optimization

Regular tr. loss: $\mathbb{E}_{x,y\sim D}\left[L\left(f(x),y\right)\right]$

Robust tr. loss: $\mathbb{E}_{x,y\sim D}\left[\max_{x'\in P(x)} L\left(f(x'),y\right)\right]$

$P(x)$ is a pre-defined perturbation set

1. What's a good P?
2. How do we train this model?

$$\min_{\theta} \frac{1}{n}\sum_{i=1}^{n} \max_{x'\in P(x_i)} L\left(f_\theta(x'),y_i\right)$$

inner max

outer min

https://arxiv.org/abs/1706.06083

Straightforward Gradient Descent Ascent - https://arxiv.org/abs/1910.08623

Defense strategy: Robust Optimization

outer min

$$\phi_{x,y}(\theta) = \max_{x' \in P(x)} L(f_\theta(x'), y)$$

$$\nabla_\theta \phi_{x,y}(\theta) = \nabla_\theta L(f_\theta(x^*), y)$$

Danskin's th: We can obtain a gradient of ϕ w.r.t θ by finding a constrained maximizer x^*

$x^* \longrightarrow$ Find a good attack

https://arxiv.org/abs/1706.06083

Defense strategy: Robust Optimization

Algo:

- Sample a data point x, y
- Compute x^* for the robust loss $\phi_{x,y}(\theta)$
- Compute gradient $g = \nabla_\theta L(f_\theta(x^*), y)$
- Update θ with g

Repeat

Inner max:

PGD $\quad \Pi_C(x + \eta \, \mathrm{sign}(\nabla L(x, y)))$

Basic iter

https://arxiv.org/abs/1706.06083

Certified defenses

Raghunathan, Aditi, Jacob Steinhardt, and Percy Liang. "Certified defenses against adversarial examples." *arXiv preprint arXiv:1801.09344* (2018).

Cohen, Jeremy M., Elan Rosenfeld, and J. Zico Kolter. "Certified adversarial robustness via randomized smoothing." *arXiv preprint arXiv:1902.02918* (2019).

Attackers are winning!

Defense	Dataset	Distance	Accuracy
Buckman et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%*
Ma et al. (2018)	CIFAR	0.031 (ℓ_∞)	5%
Guo et al. (2018)	ImageNet	0.005 (ℓ_2)	0%*
Dhillon et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%
Xie et al. (2018)	ImageNet	0.031 (ℓ_∞)	0%*
Song et al. (2018)	CIFAR	0.031 (ℓ_∞)	9%*
Samangouei et al. (2018)	MNIST	0.005 (ℓ_2)	55%**
Madry et al. (2018)	CIFAR	0.031 (ℓ_∞)	47%
Na et al. (2018)	CIFAR	0.015 (ℓ_∞)	15%

Anish Athalye, Nicholas Carlini, David Wagner, "*Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples*", **ICML** 2018.

Semantic (but perceptible) attacks/edge-cases

Tesla death smash probe: Neither driver nor autopilot saw the truck

Report shows driver took a hands-off approach to driving

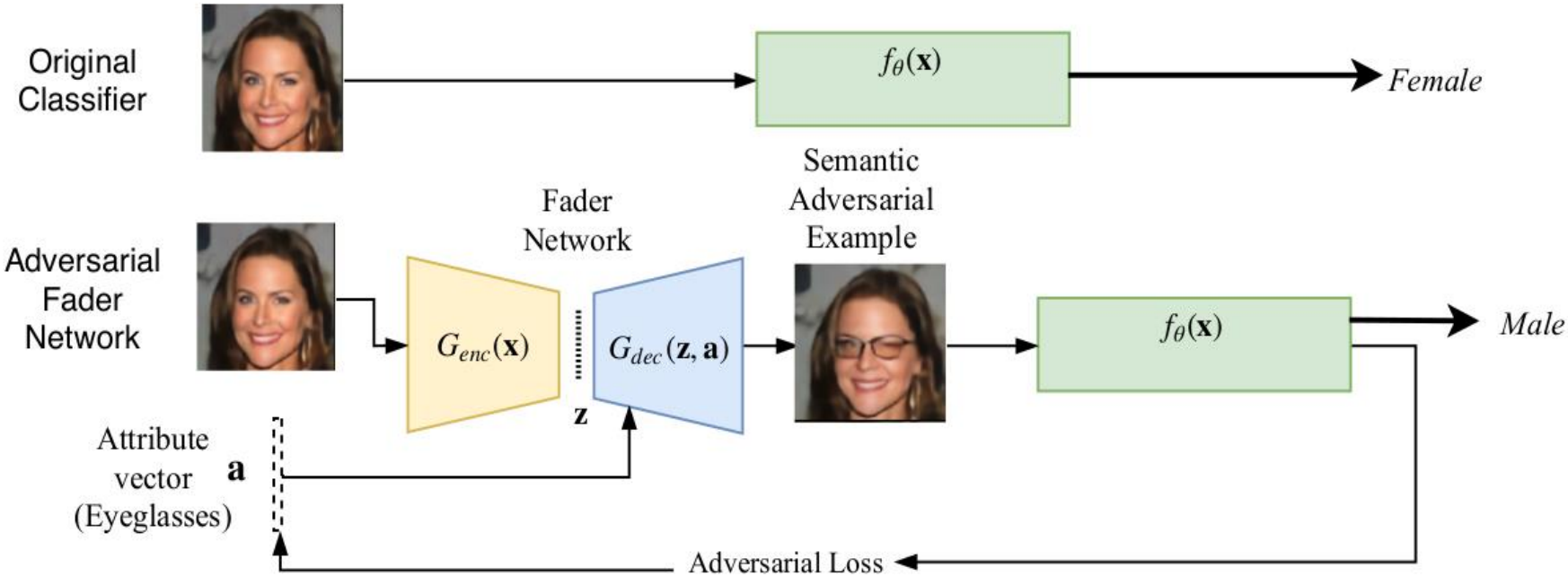
By Gareth Corfield 20 Jun 2017 at 21:58 143 SHARE ▼

The wreckage of Joshua Brown's Tesla Model S (Pic: US NTSB)

A car correctly classified in daylight but misclassified at night.

A female classified incorrectly as male by just adding glasses.

Generating semantic attacks



- A novel algorithm leveraging generative models to generate semantic adversarial examples for deep classifiers
- Empirical and theoretical analysis of such semantic examples

A. Joshi, A. Mukherjee, S. Sarkar, C. Hegde, *Semantic Adversarial Attacks: Parametric Transformations That Fool Deep Classifiers*, **International Conference on Computer Vision (ICCV)**, Seoul, South Korea, Oct 27 – Nov 2, 2019.

Semantic data augmentation

Day to night **Night to day**

Test Dataset	Classes	M_1 (Benign Data)	M_2(Orig. Dataset)	M_3 (Semantic Augmentation)
Benign (Day)	Bus	12.00	12.4	11.82
	Car	8.59	8.30	8.55
	Truck	10.43	9.6	10.0
	mAP (Overall)	10.34	10.1	10.14
Original (Night)	Bus	8.74	12.8	10.30
	Car	8.4	9.1	8.5
	Truck	6.6	8.55	8.0
	mAP (Overall)	7.91	10.15	9.0

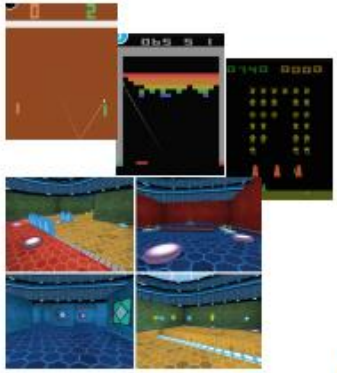
A few success stories of reinforcement learning

Kohl and Stone, 2004

Ng et al, 2004

Tedrake et al, 2005

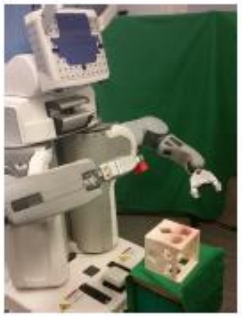
Kober and Peters, 2009



Mnih et al 2013 (DQN)
Mnih et al, 2015 (A3C)

Silver et al, 2014 (DPG)
Lillicrap et al, 2015 (DDPG)

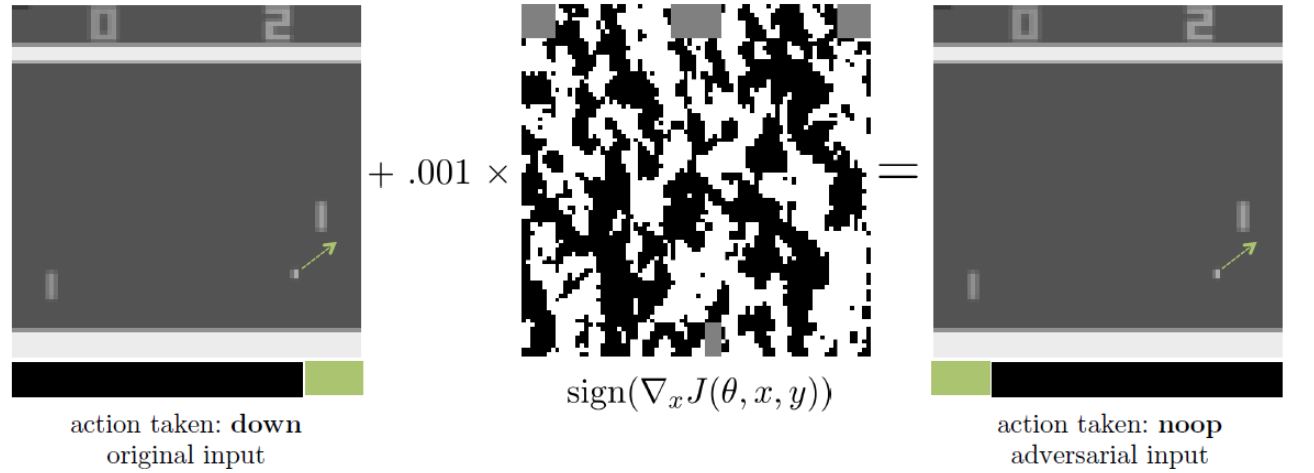
Schulman et al,
2016 (TRPO + GAE)



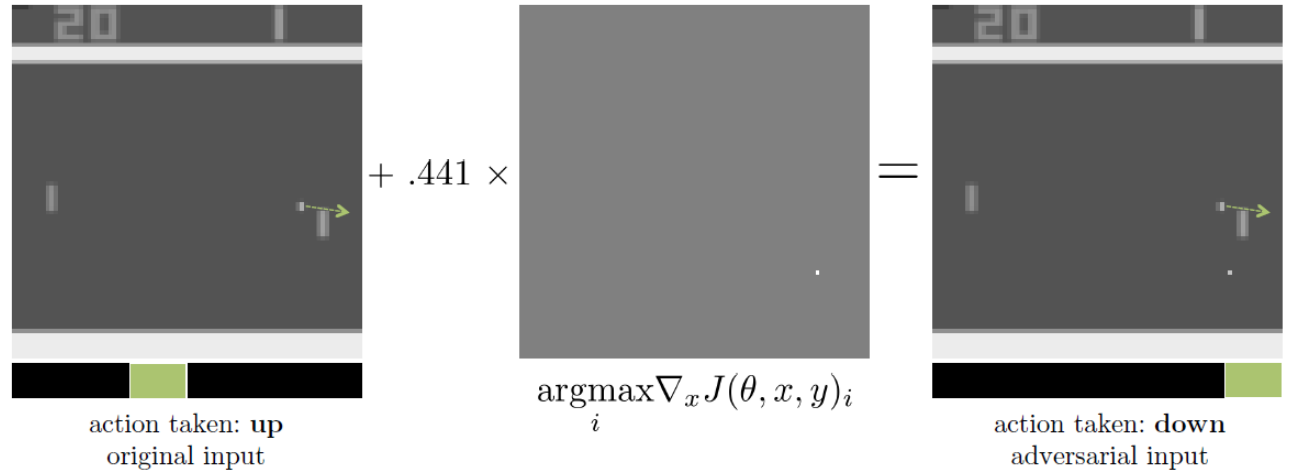
Levine*, Finn*, et
al, 2016
(GPS)

Silver*, Huang*, et
al, 2016
(AlphaGo)

Attacks on deep RL agents



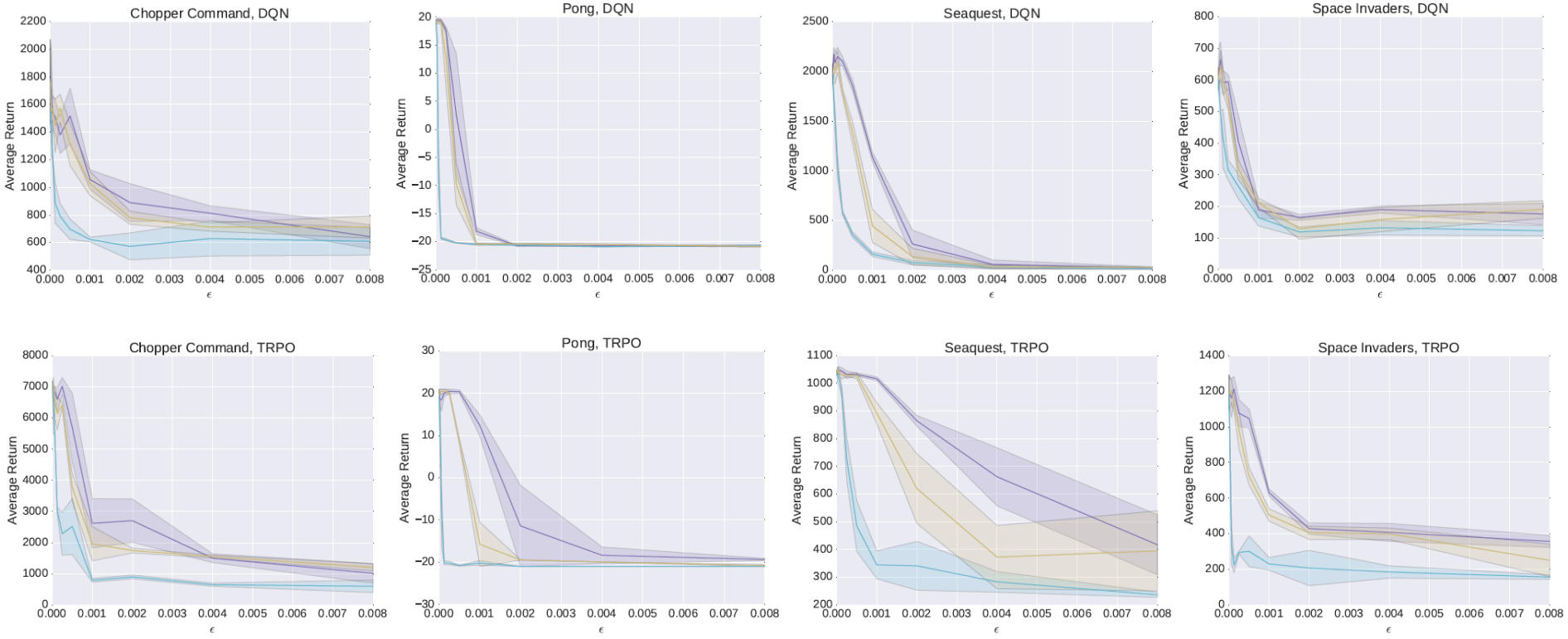
l_∞ attack



l_1 attack

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., & Abbeel, P., *Adversarial attacks on neural network policies*. **ICLR** 2017

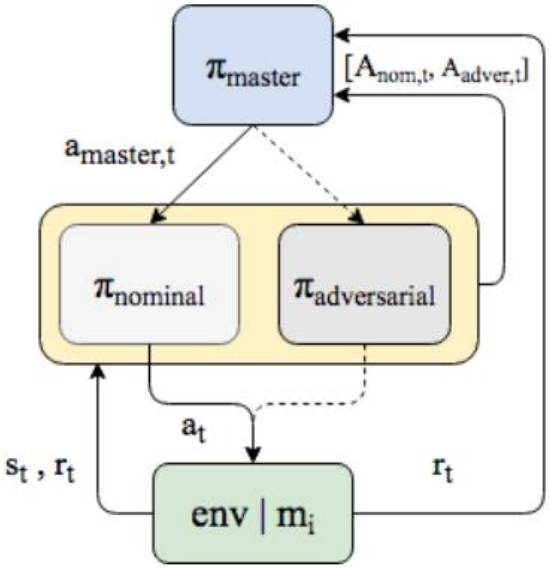
FGSM attack example



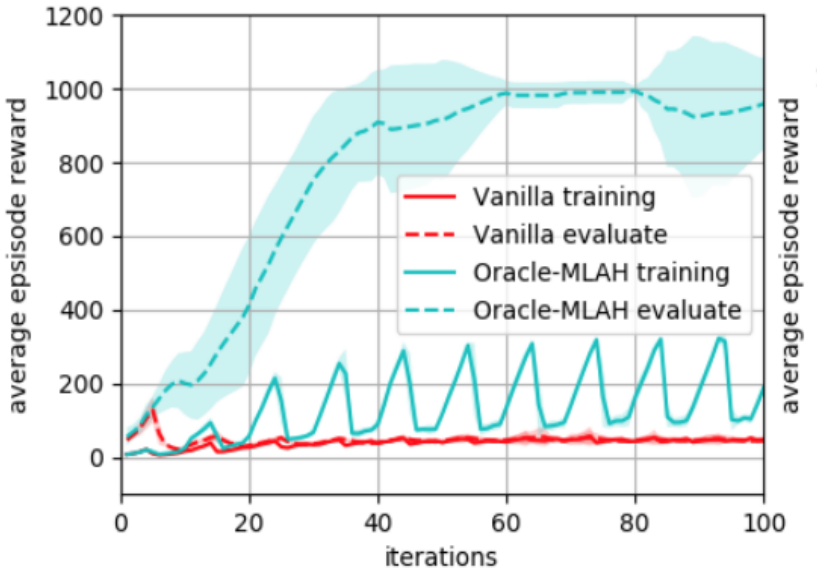
Huang, S., Papernot, N., Goodfellow, I., Duan, Y., & Abbeel, P., *Adversarial attacks on neural network policies*. **ICLR** 2017

Defending deep RL agents

- Meta-learning based supervisory framework for robust policy learning – Meta-Learned Advantage Hierarchy (MLAH) algorithm
- In the presence of unknown adversaries
- Provably improved performance over state-of-the-art strategies such as PPO
- **Key idea** – We detect an attack by monitoring the learning performance, then automatically spawn another sub-policy that protects the nominal sub-policy from corrupt observation as well as **learns to cope** with the attack and (partially) recover performance!

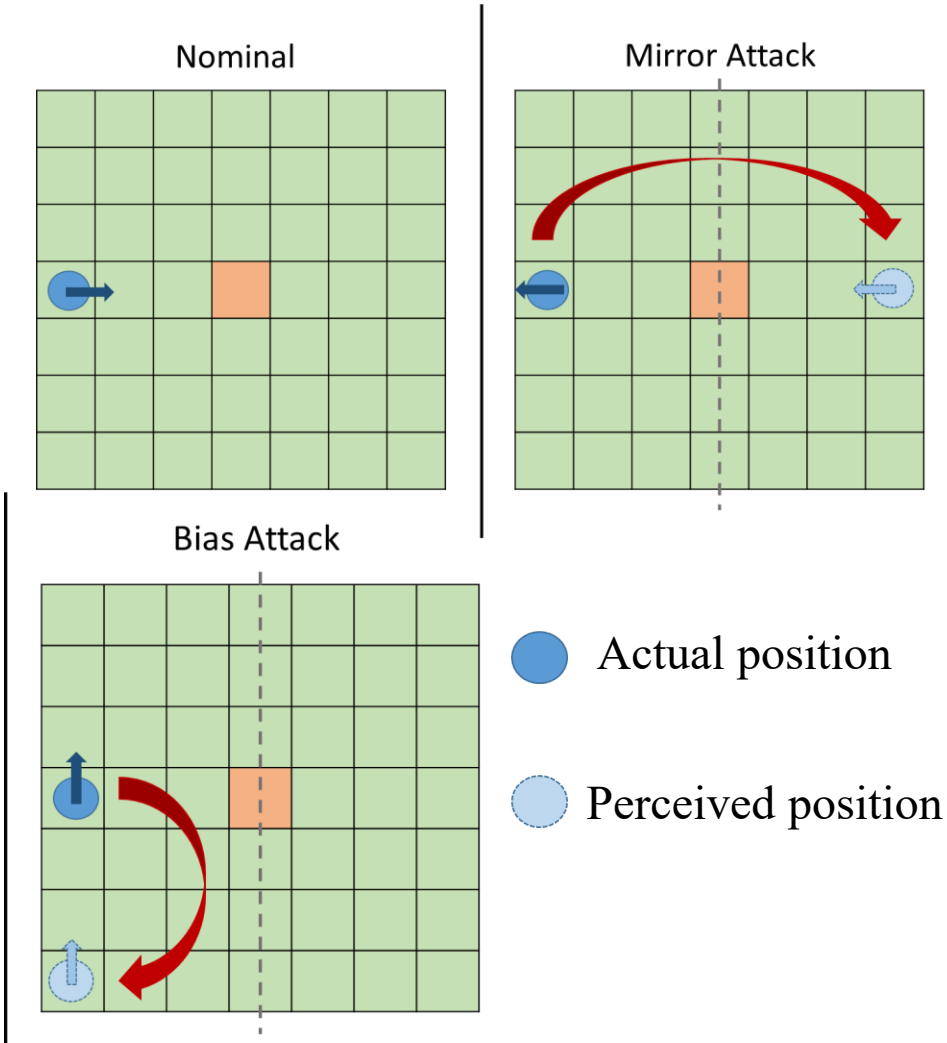


(b) MLAH framework



Aaron Havens, Zhanhong Jiang, Soumik Sarkar, *Online Robust Policy Learning in the Presence of Unknown Adversaries*, **Advances in Neural Information Processing Systems (NeurIPS)**, (Montreal, Canada), 2018.

Learning to cope with attacks



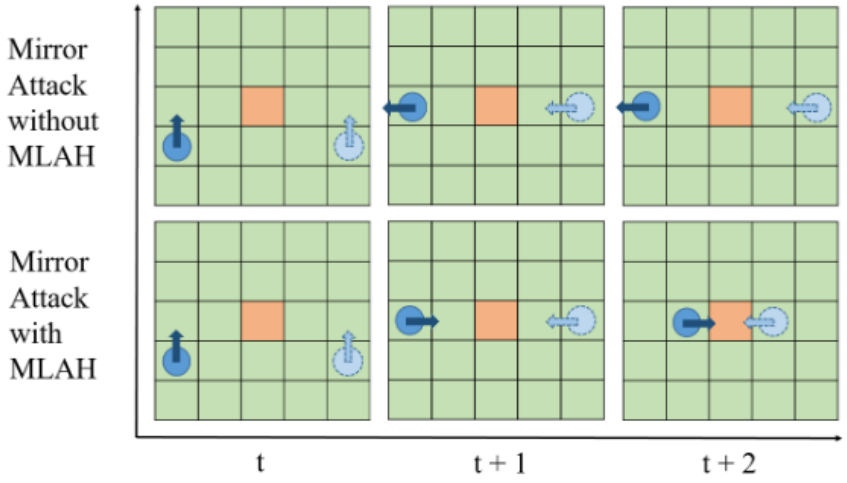
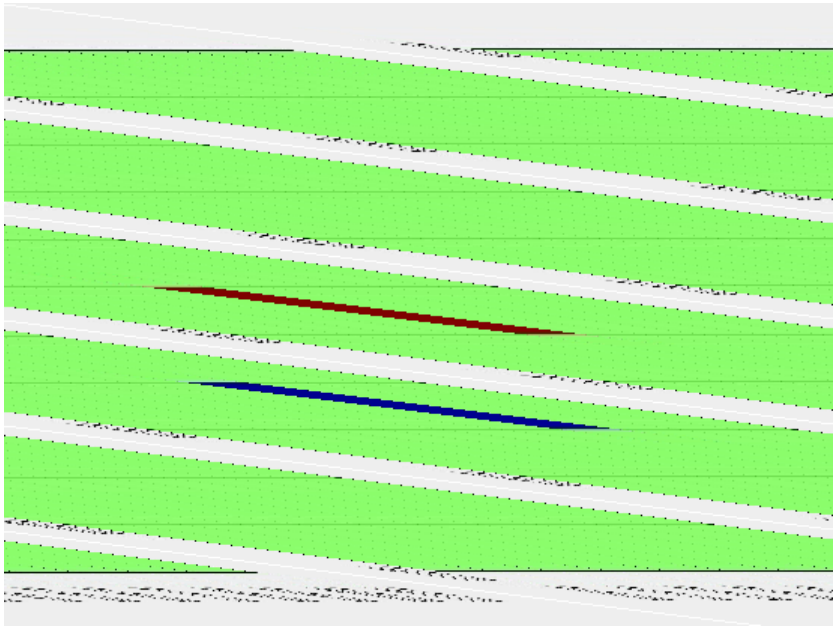


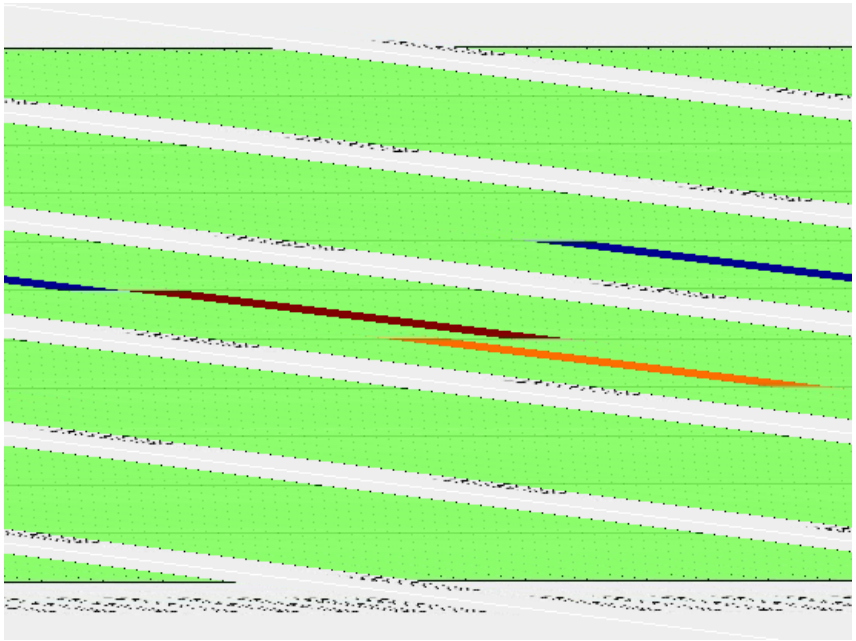
Figure 4. Illustration of MLAH agent's coping behaviour under symmetrical mirror attack about the y-axis. The MLAH agent learns to use a different sub-policy that maps the adversarial observation to an optimal action that leads it to the goal.

Learning to cope with attacks

Nominal Policy



Policy under mirror attack

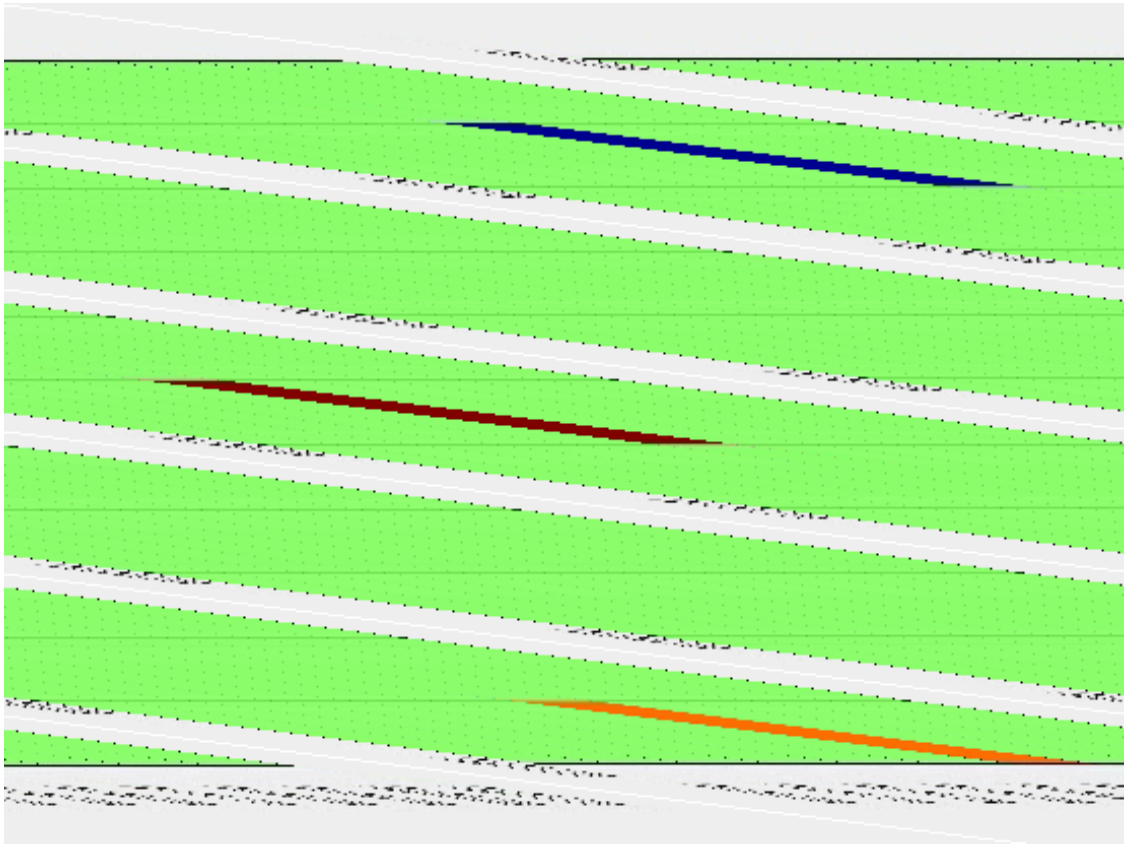


Legend:

- Goal
- Actual Observation
- Adversarial Observation

Learning to cope with attacks

Coping with mirror attack



Legend:

- Goal
- Actual Observation
- Adversarial Observation

Action space attacks

- Motivation
  - Deep RL demonstrated to be good controllers
  - Cyber-physical systems are susceptible to actuation attacks
  - Most literature have studied state space attacks but not action space attacks
  - Given a certain budget, how to best distribute attacks to systems with multiple actuators?

- Threat model
  - Access to RL agent's action stream
  - Access to RL agent's training environment
  - Knowledge of RL agent's architecture (white-box attack)

X.Y. Lee, S. Ghadai, K.L. Tan, C. Hedge, S. Sarkar, *Spatiotemporally Constrained Action Space Attacks on Deep Reinforcement Learning Agents*, **Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)**, New York, NY, 2020.

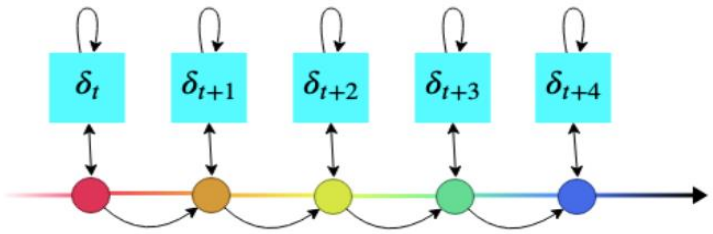
Myopic action space (MAS) attacks

- Formulation

$$\underset{\delta_t}{\text{minimize}} \quad R_{\text{adv}}(\delta_t) = R(s_t, a_t + \delta_t) + \sum_{j=t+1}^{T} R(s_j, a_j)\,,$$

$$\text{subject to} \quad \|\delta_t\|_p \leq b,\ s_{j+1} = E(s_j, a_j),\ a_j = \Theta(s_j)\ (\text{for } j = t, \ldots, T)$$

- Myopic adversary
  - At each time step, a perturbation is designed to minimize the agent's reward at the current time step



Myopic action space (MAS) attacks

Algorithm 1: Myopic Action Space (MAS) Attack

1 Initialize nominal environment, E_{nom}, nominal agent π_{nom} with weights, θ
2 Initialize budget b
3 **while** $t \leq T$ **do**
4 Compute adversarial action $\hat{a}_{t+\frac{1}{2}}$ using ∇R_{adv}
5 Compute $\delta_t = \hat{a}_{t+\frac{1}{2}} - a_t$, project δ_t onto ball of size b to get δ'_t
6 Compute projected adversarial action $\hat{a}_t = a_t + \delta'_t$
7 Step through E_{nom} with $\hat{a}_t$ to get next state

- Projection step on line 5 ensures crafted perturbation is within a budget and also represents the allocation of perturbations across different action dimensions (across multiple actuators)

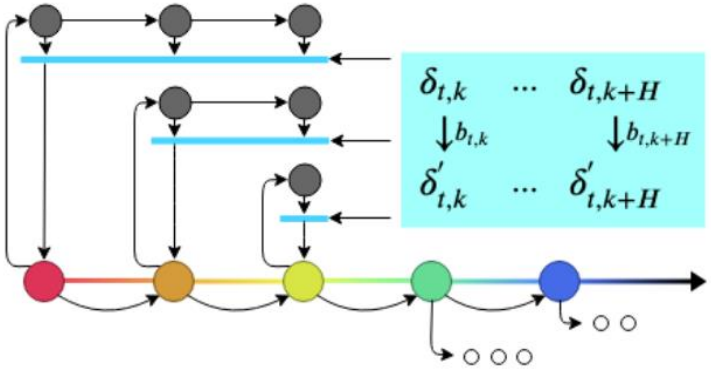
Look-ahead action space (LAS) attacks

- Formulation

$$\underset{\Delta}{\text{minimize}} \quad R_{adv}(\Delta) = \sum_{j=t}^{t+H} R(s_j, a_j + \delta_j) + \sum_{j=t+H+1}^{T} R(s_j, a_j)$$

$$\text{subject to} \quad \|\Delta\|_{p,q} \leq B,\ \Delta = [\delta_t, \delta_{t+1}, \ldots, \delta_H],\ s_{j+1} = E(s_j, a_j),\ a_j = \Theta(s_j)$$

- Non-myopic adversary
  - At each time step, adversary takes the dynamics of the agent into account and crafts a perturbation that minimizes reward up to a certain horizon



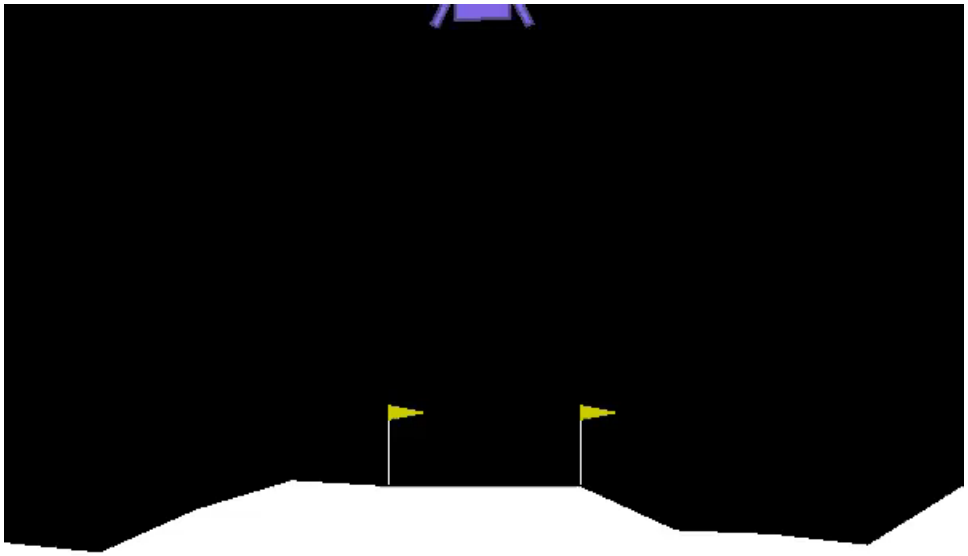
Look-ahead action space (LAS) attacks

Algorithm 2: Look-ahead Action Space (LAS) Attack

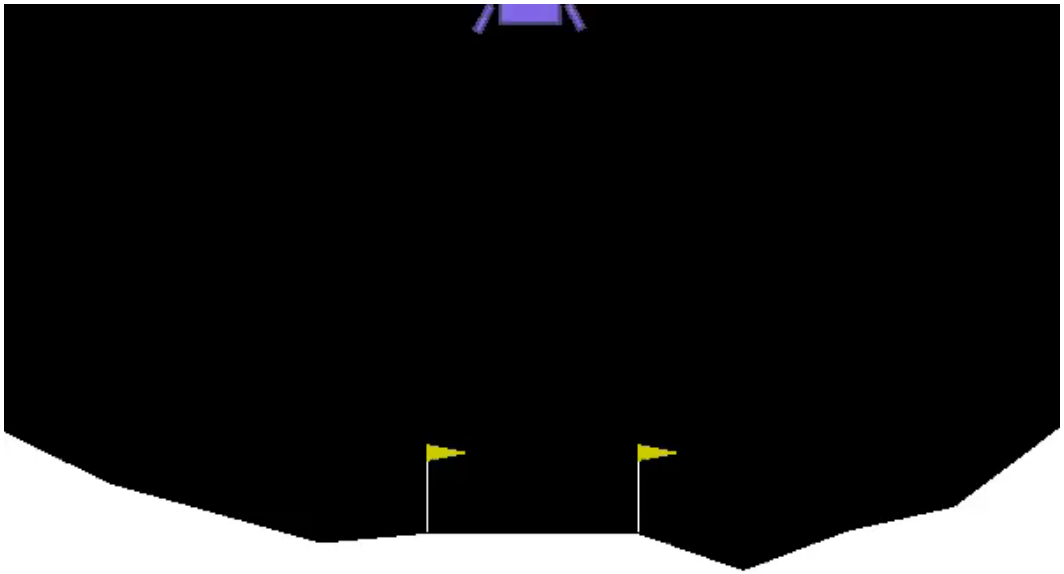
Initialize nominal and adversary environments E_{nom}, E_{adv} with same random seed
Initialize nominal agent π_{nom} weights, θ
Initialize budget B, adversary action buffer A_{adv}, horizon H
while $t \leq T$ **do**
    Reset A_{adv}
    if $H = 0$ **then**
        Reset H and B
    while $k \leq H$ **do**
        Compute adversarial action $\hat{a}_{t+\frac{1}{2},k}$ using ∇R_{adv}
        Compute $\delta_{t,k} = \hat{a}_{t+\frac{1}{2},k} - a_{t,k}$
        Append $\delta_{t,k}$ to A_{adv}
        Step through E_{adv} with $a_{t,k}$ to get next state
    Compute $||\delta_{t,k}||_{\ell_p}$ for each element in A_{adv}
    Project sequence of $||\delta_{t,k}||_{\ell_p}$ in A_{adv} on to ball of size B to obtain look-ahead sequence of budgets $[b_{t,k}, b_{t,k+1} \ldots b_{t,k+H}]$
    Project each $\delta_{t,k}$ in A_{adv} on to look-ahead sequence of budgets computed in the previous step to get sequence $[\delta'_{t,k}\ \delta'_{t,k+1} \ldots \delta'_{t,k+H}]$
    Compute projected adversarial action $\hat{a}_t = a_t + \delta'_{t,k}$
    Step through E_{nom} with $\hat{a}_t$
    $B \leftarrow max(0, B - \delta'_{t,k})$; $H \leftarrow H - 1$

Experimental studies

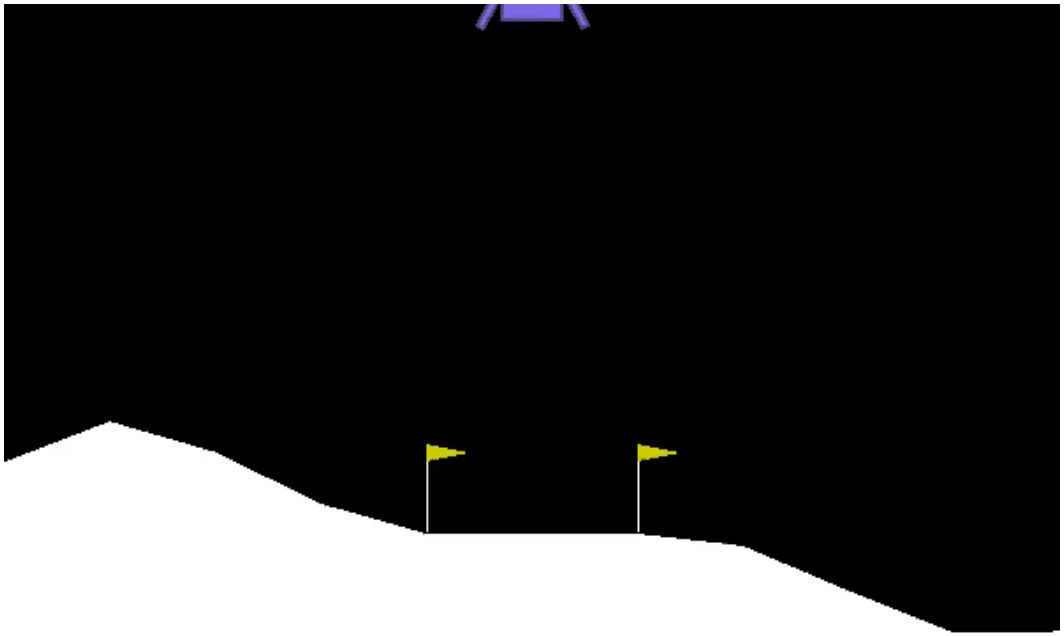
- Trained PPO agent in OpenAI Lunar Lander environment



Example of LAS attack on PPO agent



Example of LAS attack on DDQN agent

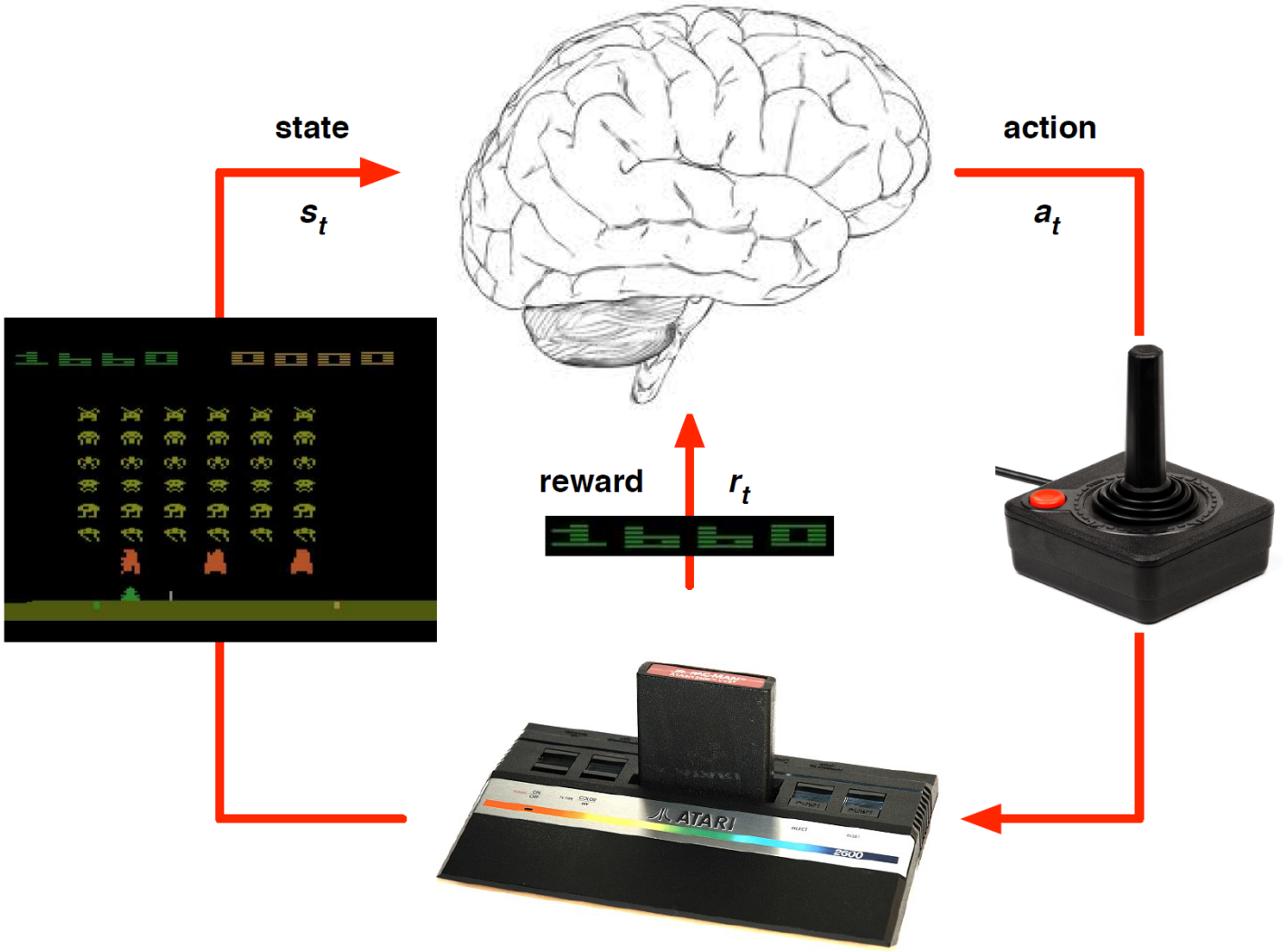


Some recent trends – understanding sources of vulnerabilities

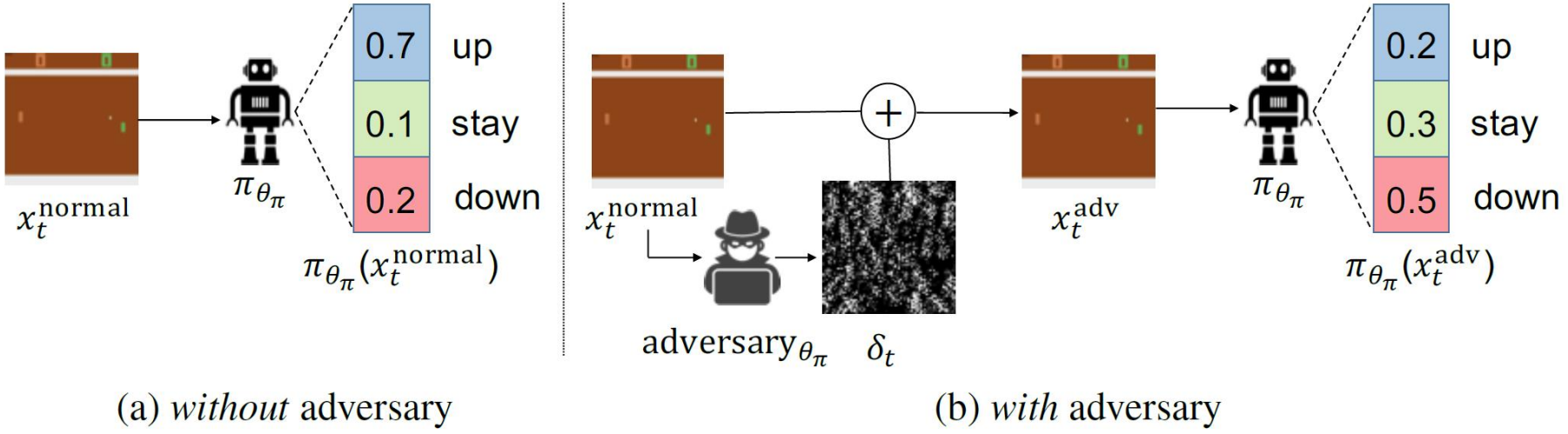
- The vulnerability comes from latent feature space, so focusing on the latent space with respect to their vulnerability. https://arxiv.org/abs/1908.04355
- Adversarial examples are not bugs, they are features https://arxiv.org/pdf/1905.02175.pdf
- Connections between robustness and interpretability - Focusing on the adversarial examples which put the interpretability of the model at risk https://openreview.net/pdf?id=Hyes70EYDB
- Do models have strong bias towards a dataset rather than the underlying task? https://arxiv.org/abs/2002.04108
- Focusing on designing network architectures which provides better numerical stability https://proceedings.icml.cc/static/paper_files/icml/2020/381-Paper.pdf

Backup Slides

Deep Reinforcement Learning



Attacks on Deep RL agents

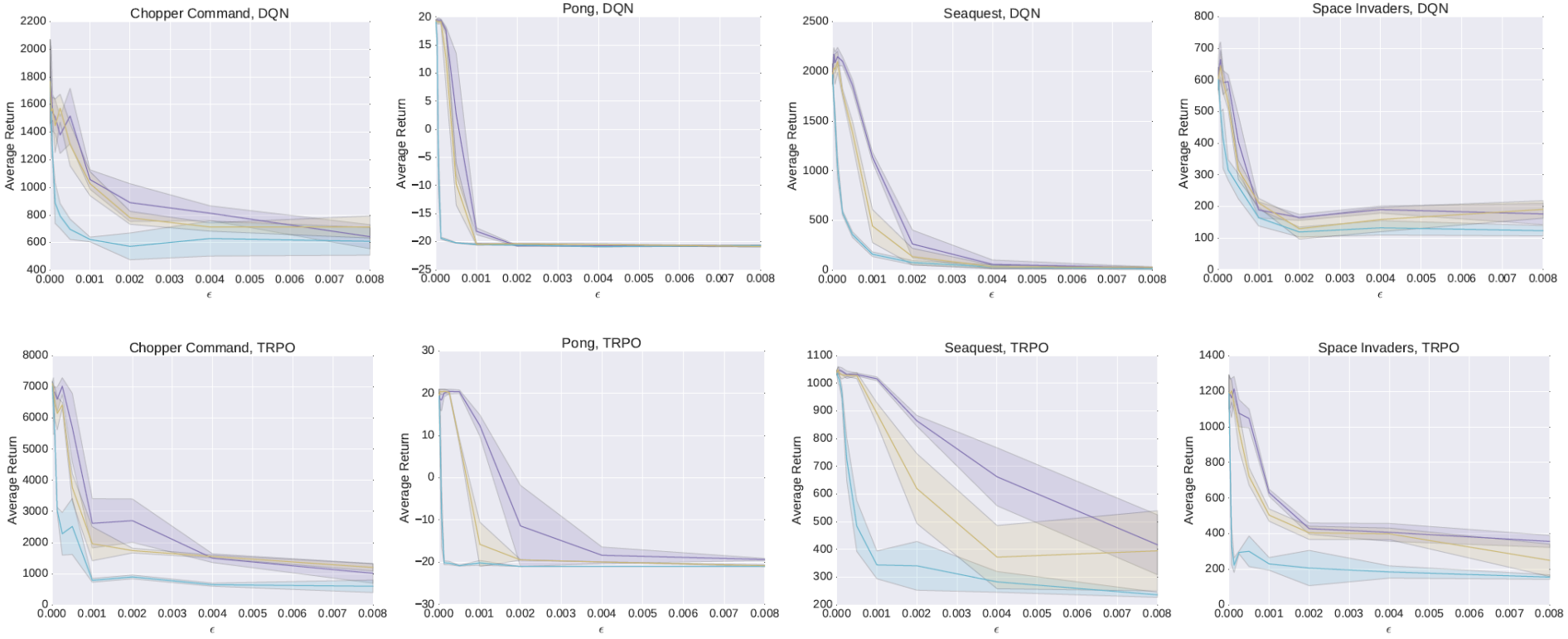


(a) *without* adversary (b) *with* adversary

Primarily test-time attacks

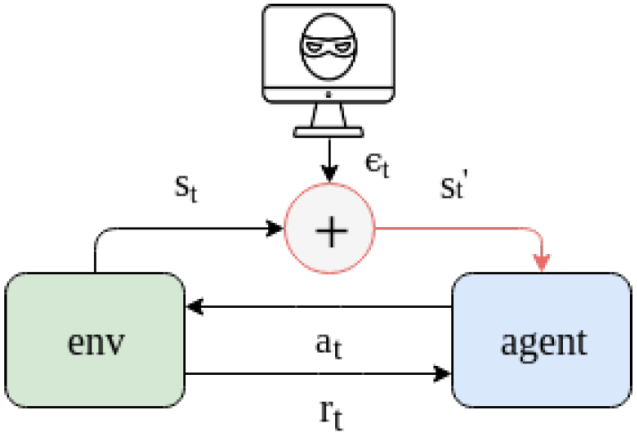
Yen-Chen Lin, Ming-Yu Liu, Min Sun, and Jia-Bin Huang, "*Detecting Adversarial Attacks on Neural Network Policies with Visual Foresight*", **Proceedings of Neural Information Processing Systems (NIPS) Workshop on Machine Deception**, 2017

FGSM attack example



Huang, S., Papernot, N., Goodfellow, I., Duan, Y., & Abbeel, P., *Adversarial attacks on neural network policies*. **ICLR** 2017

Poisoning of DRL state observation during training

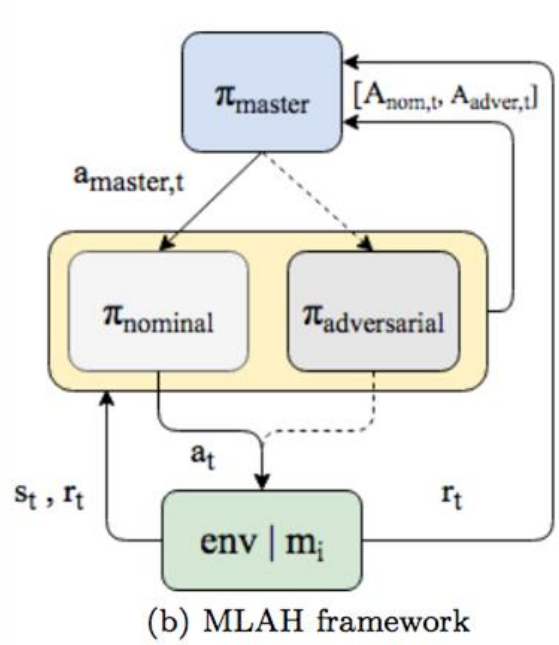


(a) Adversary interaction model

- Sampled perturbation size & direction from uniform distribution ***U(a,b)***
- Perturb state where $S_{i,\ adversary} = S_i + U(a,b)$ such that $max_i \mid S_{i,\ adversary} - S_i \mid \leq \varepsilon_{attack}$
- Experimented with :
  - White Noise Attacks where $a = -b$
  - Bias Attacks where $a \neq b$ and $a < b$

Meta-learned Advantage Hierarchy

- An online meta-learning framework with a master policy and arbitrary number of sub-policies
- Master policies learns to select different sub-policies under different situations (nominal/adversarial) using expected advantages of sub-policies as observation
- Simultaneously, sub-policies learn the best policy to follow under different situations (nominal/ adversarial)



(b) MLAH framework

A. Havens, Z. Jiang, S. Sarkar, *Online Robust Policy Learning in the Presence of Unknown Adversaries*, **Proceedings of Advances in Neural Information Processing Systems (NIPS)**, (Montreal, Canada), 2018.

MLAH Algorithm

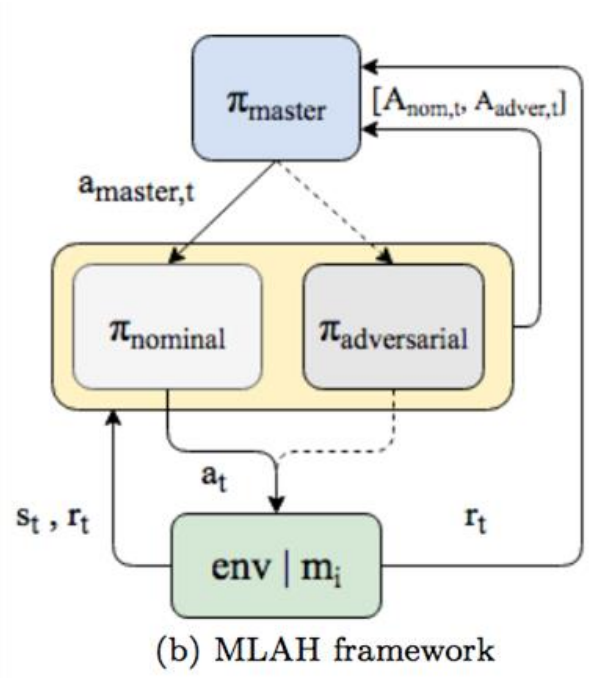
Algorithm 1: MLAH

Input : π_{nom} and π_{adv} sub-policies parameterized by θ_{nom} and θ_{adv}; Master policy π_{master} with parameter vector ϕ.

Initialize θ_{nom}, θ_{adv}, ϕ
for *pre-training iterations [optional]* **do**
    Train π_{nom} and θ_{nom} on only nominal experiences.
end
for *learning life-time* **do**
    for *Time steps* t *to* $t+T$ **do**
        Compute $\mathbf{A}_t$ over sub-policies (see eq. 4)
        select sub-policy to take action with π_{master} using $\mathbf{A}_t$ as observations
    end
    Estimate all A_{GAE} for π_{nom}, π_{adv} over T
    Estimate all A_{GAE} for π_{master} over T with respect to $\mathbf{A}_t$ observations
    Optimize θ_{nom} based on experiences collected from π_{nom}
    Optimize θ_{adv} based on experiences collected from π_{adv}
    Optimize ϕ based on all experiences with respect to $\mathbf{A}_t$ observations
end

MLAH advantages and disadvantages

- Pros:
  - Advantage as observation for master policy acts a useful metric for detecting the presence of adversary
  - Reduces bias in learned value function baseline in the presence of adversary
    - **Intuition**: Policy learns to map different policies to different states rather than resolving a single policy over multiple latent states
- Cons:
  - Proposed framework has a delayed response



(b) MLAH framework

MLAH analysis

Monotonic improvement of reward during learning (similar result as TRPO)

- *Proposition 1*: $\hat{\mathcal{R}}(\pi_{new}) \geq \hat{L}_{\pi_{old}}(\pi_{new}) - \frac{4\tilde{\epsilon}\gamma\alpha^2}{(1-\gamma)^2}$
  - $\hat{\mathcal{R}}(\pi_{new})$ denotes the actual expected discounted rewards as a function of the new policy
  - $\hat{L}_{\pi_{old}}(\pi_{new}) = L_{\pi_{old}}(\pi_{new}) + \delta - \hat{\delta}$ where $L_{\pi_{old}}(\pi_{new})$ is the approximated expected reward as a function of the new policy
    - δ denotes the observed bias of the state value
    - $\hat{\delta}$ Denotes observed bias in the expected discounted reward
  - α is the total variation divergence for between the old and new policy
  - γ is the discount factor

$$\tilde{\epsilon} = \begin{cases} max_{s,a}|\hat{A}_\pi(s,a)| + (\gamma-1)\delta, & \text{if } \hat{A}_\pi(s,a) \geq (1-\gamma)\delta. \\ -max_{s,a}|\hat{A}_\pi(s,a)| + (1-\gamma)\delta, & \text{if } 0 < \hat{A}_\pi(s,a) < (1-\gamma)\delta. \\ max_{s,a}|\hat{A}_\pi(s,a)| + (1-\gamma)\delta, & \text{if } \hat{A}_\pi(s,a) \leq 0 \end{cases}$$

  - The actual expected discounted reward as a function of the new policy is at least greater or equals to the approximation of the reward
  - This implies that using the conditioned policy (MLAH agent), we can expect a better reward than the approximation of the reward

MLAH analysis

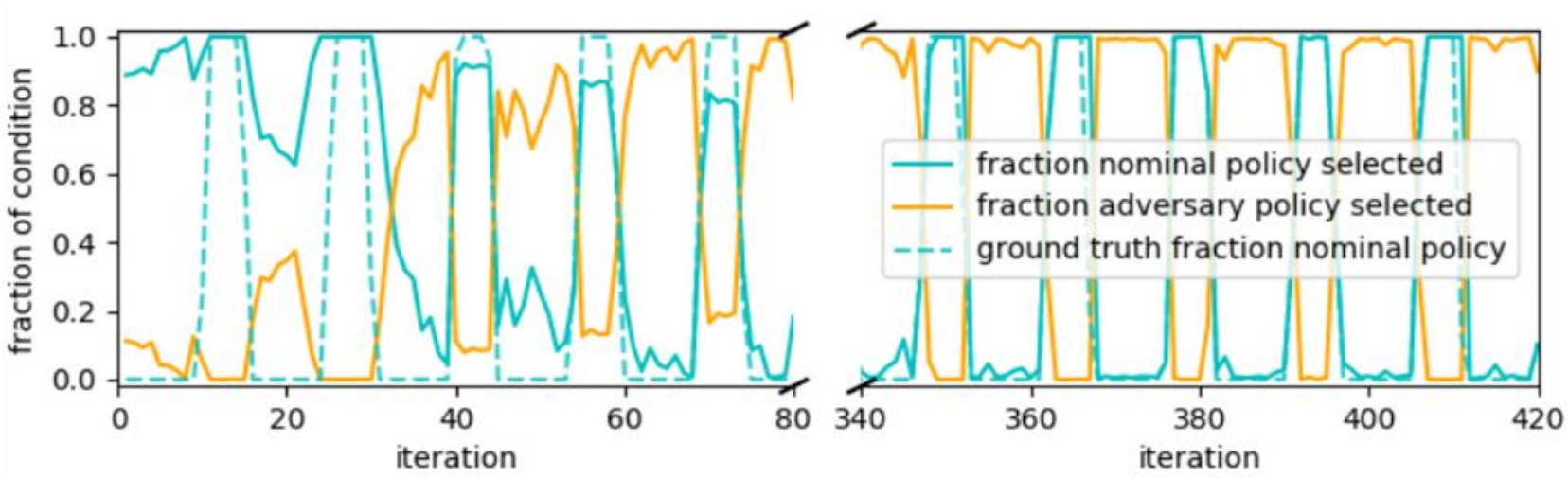
MLAH should perform better given a sufficiently intelligent attacker

- *Proposition 2*: If $\Delta\hat{\delta} < C\Delta V$, *where* $C \geq \frac{(m-n)(1-m)(4\gamma\alpha^2+1-\gamma)}{(1-m+n)(1-\gamma)}$ *and* $\Delta V = V_0 - V_1$, then the conditioned policy (MLAH agent) has a higher lower bound of expected discounted reward compared to that of the unconditioned policy (Classic RL agent)
  - $\Delta\hat{\delta} = \hat{\delta}_{unc} - \hat{\delta}_{con|0}$
  - V_0 denotes expected state value under nominal conditions
  - V_1 denotes expected state value under adversarial conditions

Main Takeaways:

- Analysis of MLAH shows that proposed framework reduces bias in value function baseline of agent under adversarial attack
- Consequently, reducing bias improves the lower bound of expected rewards

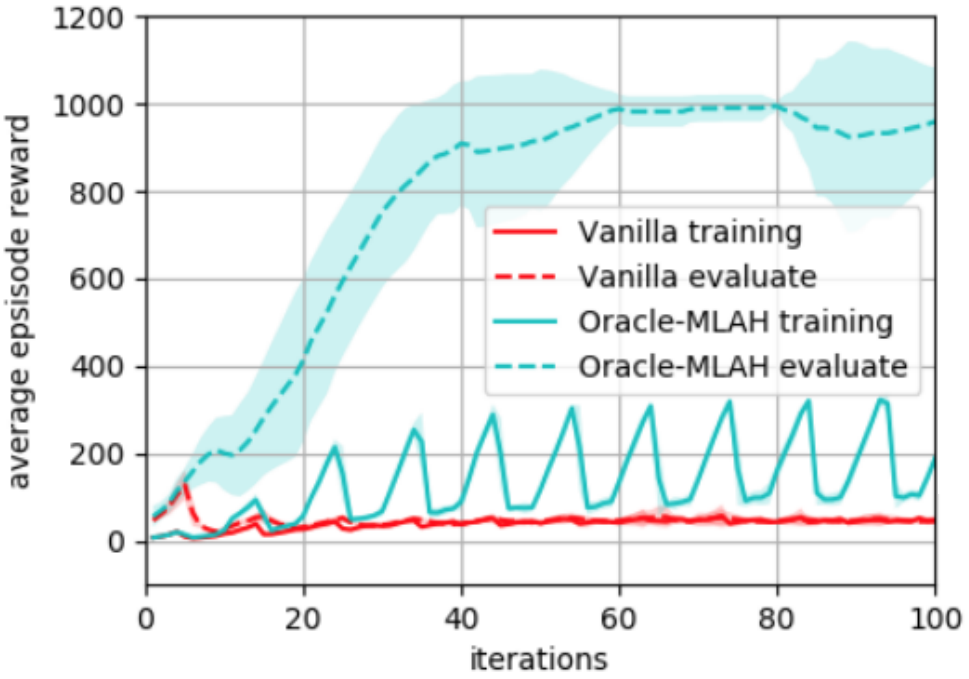
MLAH empirical results: adversary detection



Meta-agent can reliably detect (after certain initial training period) the presence of an (intermittently occurring) adversary via observing sub-policy advantages.

MLAH empirical results: performance evaluation

- Trained on Inverted-Pendulum environment with Vanilla PPO and (oracle) MLAH under intermittent l_∞ bounded-stochastic attack
- MLAH-trained agent performs significantly better during nominal evaluation (and training)
- **Insight**: Vanilla PPO with adversarial training makes the learned policy less efficient



MLAH empirical results: performance evaluation

Performance comparison under different (temporal) attack profiles

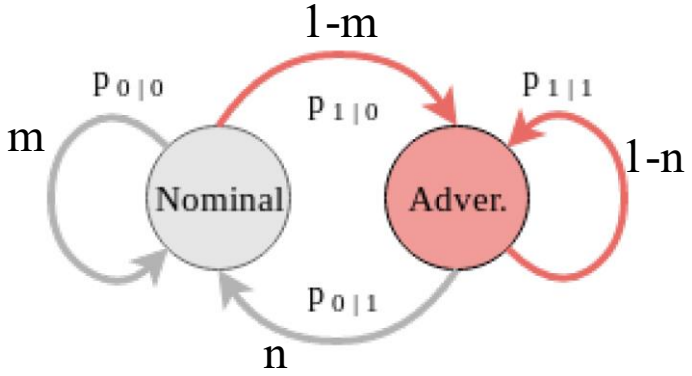


Table 2: Performance evaluation of Oracle-MLAH

	Normalized avg. training return		**Normalized avg. evaluation return**	
m/n	Vanilla	Oracle-MLAH	Vanilla	Oracle-MLAH
$1.0/-$	0.96 ± 0.03	0.96 ± 0.03	1.0	1.0
$0.995/0.005$	$0.238 \pm .082$	0.553 ± 0.242	0.471 ± 0.051	0.99 ± 0.001
$0.95/0.05$	$0.612 \pm .08$	0.677 ± 0.149	0.644 ± 0.078	0.99 ± 0.001
$0.8/0.2$	0.613 ± 0.043	0.728 ± 0.063	0.539 ± 0.023	0.994 ± 0.165
$0.5/0.5$	0.749 ± 0.093	0.764 ± 0.078	0.787 ± 0.010	0.948 ± 0.086

Results show that training an RL-agent under MLAH framework generally returns a higher reward as compared to a Vanilla RL-agent (PPO) when under adversarial attacks.