Towards Secure, Interpretable Learning in Deep Neural Networks

Jingkang Wang

University of Toronto & Vector Institute

wangjk@cs.toronto.edu

Tremendous Success in DL/ML

Explosive development (data and works)!

- 2D: ImageNet, COCO, etc.
  - EfficentNet achieves 84.4% top-1 and 97.1% top-5 accuracy on ImageNet **8.4x** smaller and **6.1x** faster).
- 3D: KITTI, Cityscapes, BDD100K, Oxford, etc.
  - F-PointNet (Rank 1 -> 40 in one year)
- Self-Driving: Uber, Google, Tesla, etc.
- Robotics, NLP, Speech, SysML, Finance, Healthcare ...
- …

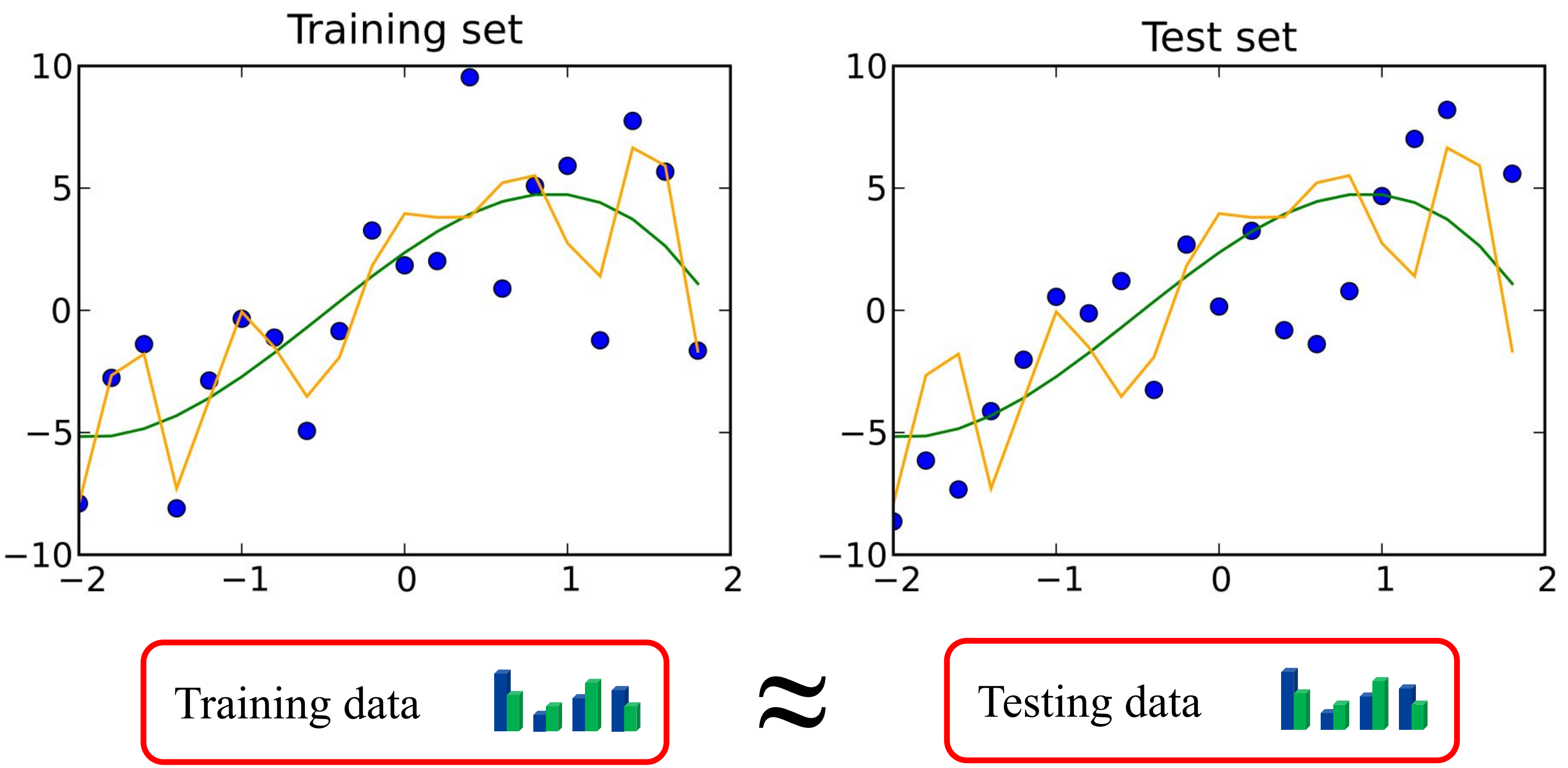
Security, Privacy and Interpretability of ML/DL are becoming increasingly important!

The Physical World Is Messy …

- Taking self-driving as an example:
  - Temporary unrecognized traffic signs
  - Extreme weather conditions
  - Obscured, broken, incomplete (even wrong) instructions
  - Unexpected emergency (e.g., collision, violation of traffic rules)
  - **Malicious** adversaries may exist!
- **Safety is always the first!**

However, DNNs Are not Robust ...

- **Perils of Stationary Assumption**



- **Noisy data**: outliers, crowdsourcing, system error, subjectivity …
- **Adversaries**: evasion attack, data poisoning, privacy leak …

What should we do?

- **More Accurate Sensors + More Robust Algorithms!**
  (Safety) **(Security)**

- **What I did in last 1.5 years:**
- Self-Driving:
  - Policy learning: 2D + LiDAR (cheaper but still very expensive)
- Tackling with Noisy Data:
  - Reinforcement Learning with Perturbed Reward
- Attack & Defend our ML/DL Systems:
  - Arms Race in Adversarial Machine Learning (AML)
  - More reliable, interpretable DL training scheme (Information Bottleneck, IB)
  - Min-Max Optimization in AML (across domains)

What should we do?

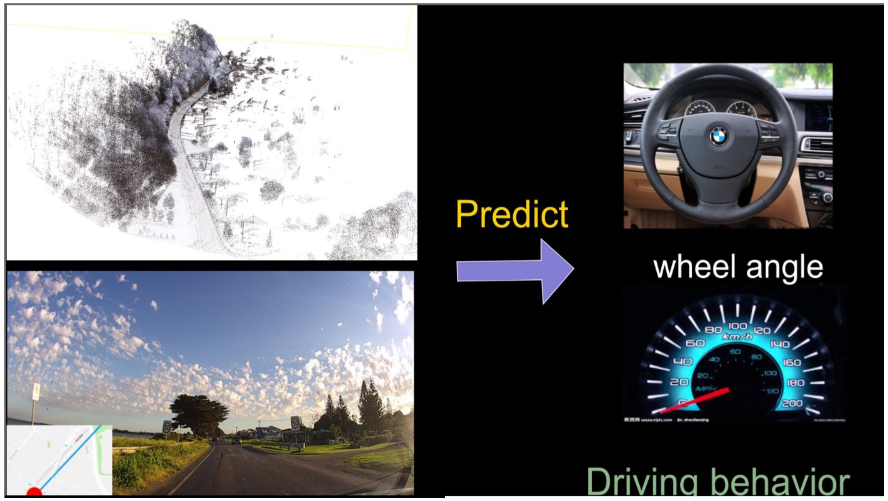
- **More Accurate Sensors + More Robust Algorithms!**
  (Safety) **(Security)**
- **What I did in last 1.5 years:**
- **Self-Driving:**
  - **Policy learning: 2D + LiDAR (cheaper but still very expensive)**
- Tackling with Noisy Data:
  - Reinforcement Learning with Perturbed Reward
- Attack & Defend our ML/DL Systems:
  - Arms Race in Adversarial Machine Learning (AML)
  - More reliable, interpretable DL training scheme (Information Bottleneck, IB)
  - Min-Max Optimization in AML (across domains)

DBNet (A Large-scale Driving Behavior Dataset)

- End-to-end Learning for Self-Driving

NVIDIA

- Pioneer to apply end-to-end learning techniques to solve autonomous driving problems.
- Learning from 8.5h driving videos.



Problems:

- Limited scale of data
- Lack of high-quality 3D LiDAR point clouds

Solution: DBNET

Datasets	Video/Image	LiDAR	Behaviors
KITTI	✓	✓	×
Cityscape	✓	×	×
Oxford	✓	✓	×
Comma.ai	✓	×	✓
BDDV	✓	×	✓
DBNet (ours)	**✔**	**✔**	**✔**

[1] Bojarski et al. End to end learning for self-driving cars. 2016.

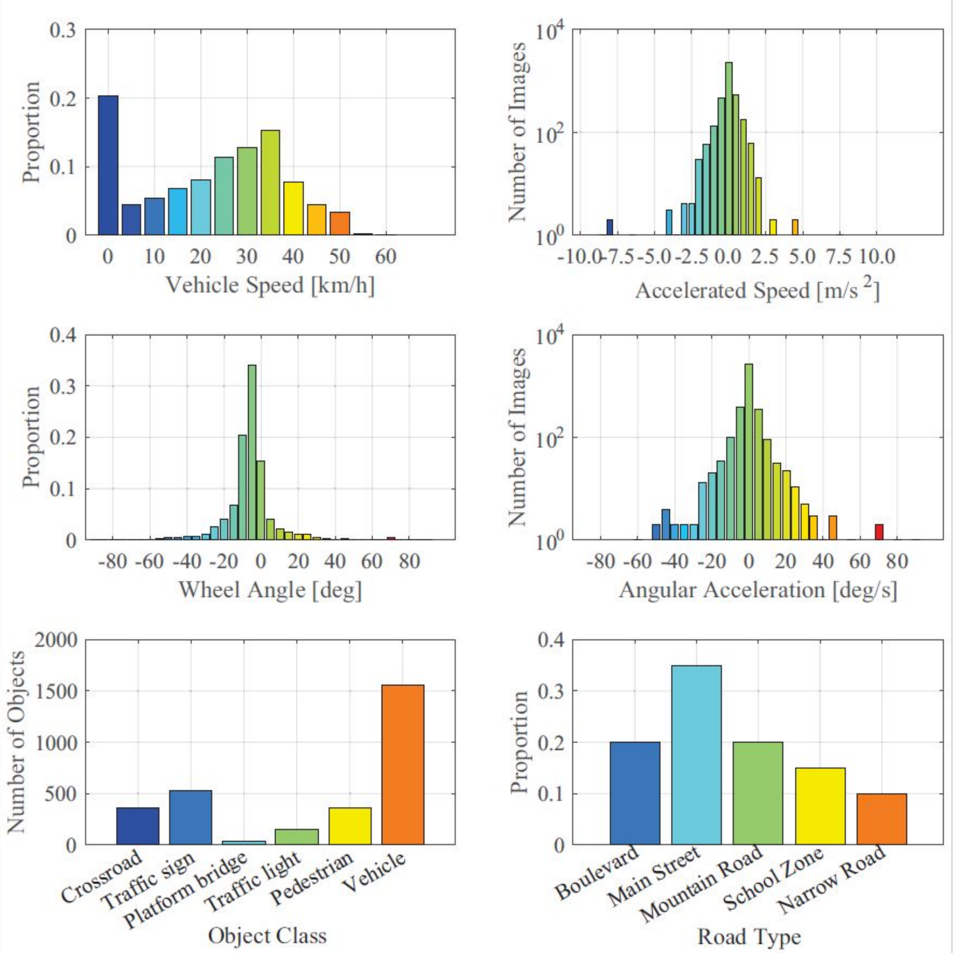
DBNet (A Large-scale Driving Behavior Dataset)

- Driving Behavior (2D + 3D)

First driving behavior dataset that incorporates 2D and 3D.

- Large-Scale:
  - **10 times** larger than KITTI
- Diversity:
  - Different types of roads/weathers
- Quality:
  - High-precision LiDAR point clouds
    (> 10 million points / 100m)
  - 1920x1080 videos
  - Sensor-collected driving behaviors

Open-source:
http://www.dbehavior.net **(official website)**
https://github.com/driving-behavior/DBNet
(code, 130+ stars)



DBNet (Results)

DNN Architecture	Metric	prediction accuracy of steering angle and vehicle speed					
		DNN only			DNN-LSTM		
		IO	PM	PN	IO	PM	PN
NVIDIA	angle	63.0%	67.1%	**71.1%**	77.9%	**83.5%**	81.6%
	speed	**70.1** %	69.2%	66.1%	70.9%	73.8%	**76.8%**
Resnet-152	angle	65.3%	**70.8%**	68.6%	78.4%	**84.2%**	82.7%
	speed	71.4%	**72.6%**	69.4%	71.9%	74.3%	**78.3%**
Inception-v4	angle	70.5%	71.1%	**73.2%**	78.3%	83.7%	**84.8%**
	speed	68.5%	**70.3%**	69.3%	70.3%	76.4%	**77.3%**

IO: images only
PM: images + feature maps (PCM)
PN: images + PointNet

Results:

1. Depth information benefits policy learning a lot
2. PM and PN show good performance
3. Sequential information is critical

DBNet (Demo)

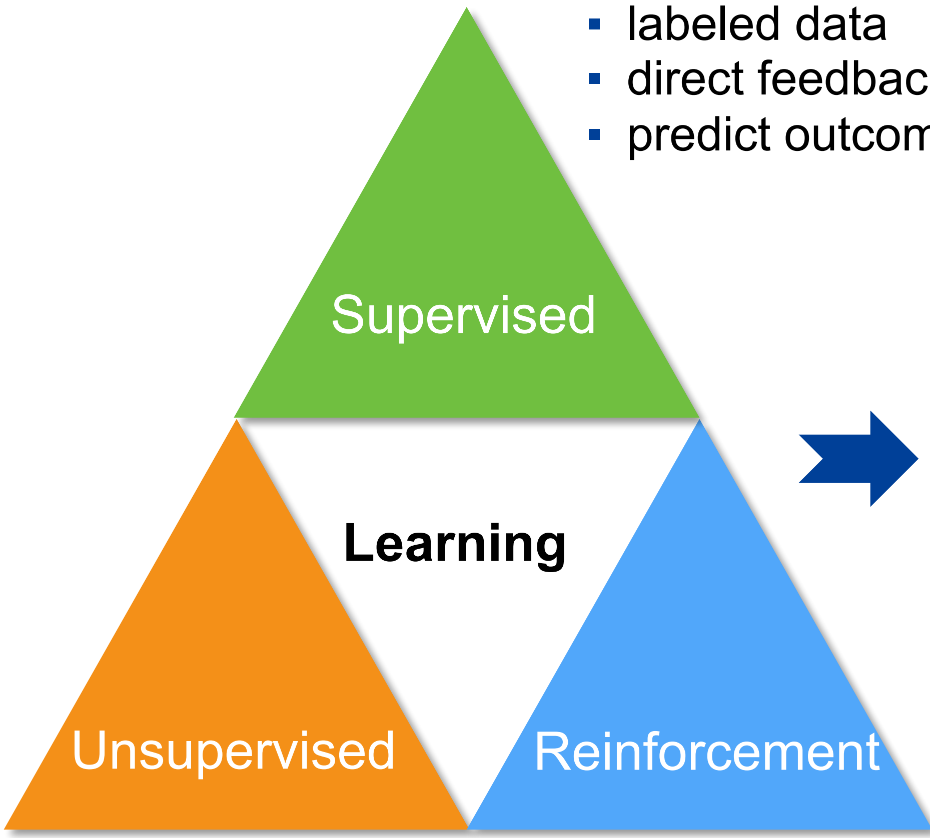
- Video + LiDAR Point Clouds => Angle + Speed

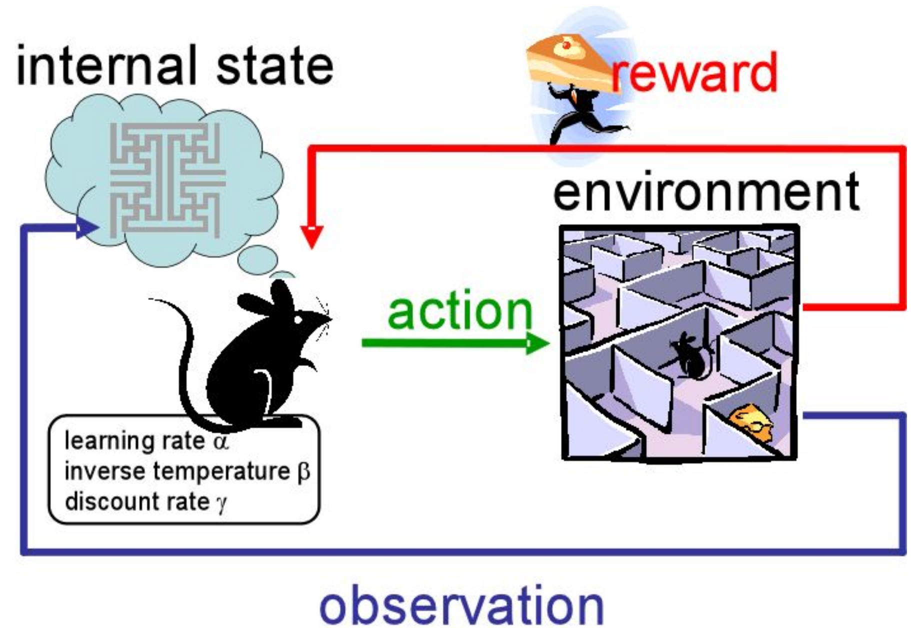
What should we do?

- **More Accurate Sensors + More Robust Algorithms!**
  (Safety) **(Security)**
- **What I did in last 1.5 years:**
- Self-Driving:
  - Policy learning: 2D + LiDAR (cheaper but still very expensive)
- **Tackling with Noisy Data:**
  - **Reinforcement Learning with Perturbed Reward**
- Attack & Defend our ML/DL Systems:
  - Arms Race in Adversarial Machine Learning (AML)
  - More reliable, interpretable DL training scheme (Information Bottleneck, IB)
  - Min-Max Optimization in AML (across domains)

Reinforcement Learning (RL)

- labeled data
- direct feedback
- predict outcome

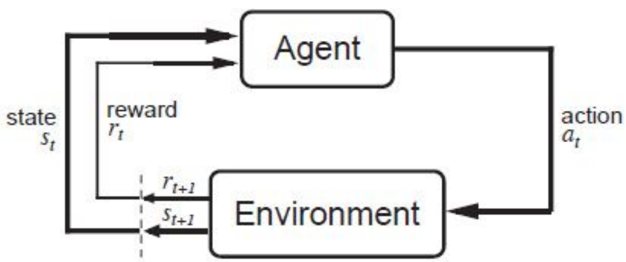




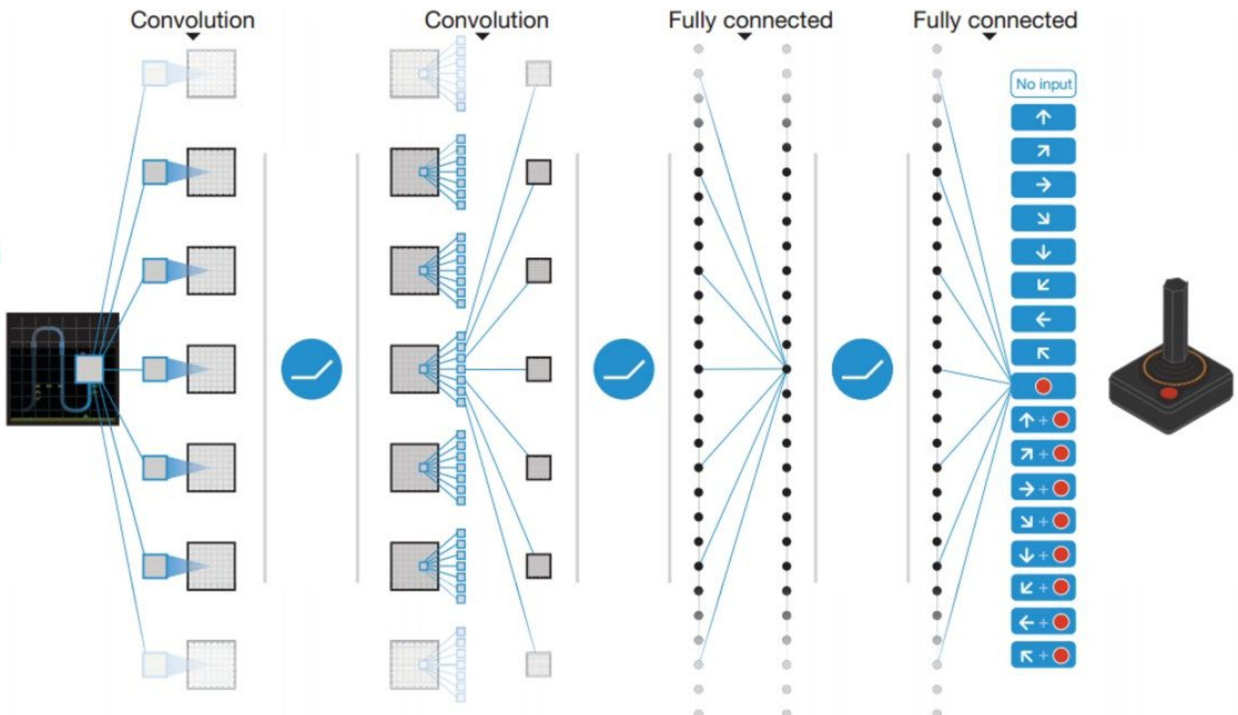
- no labels data
- no feed back
- find hidden structure

- decision process
- reward system
- learn series of actions

Breakthrough in Deep RL

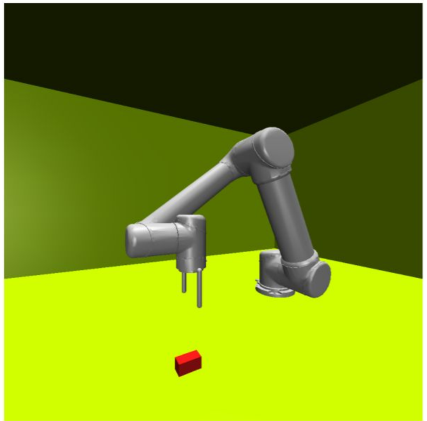


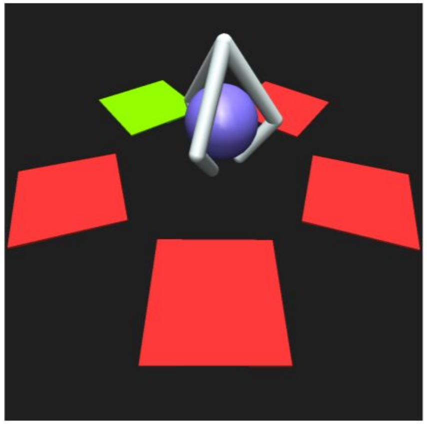
AlphaGo

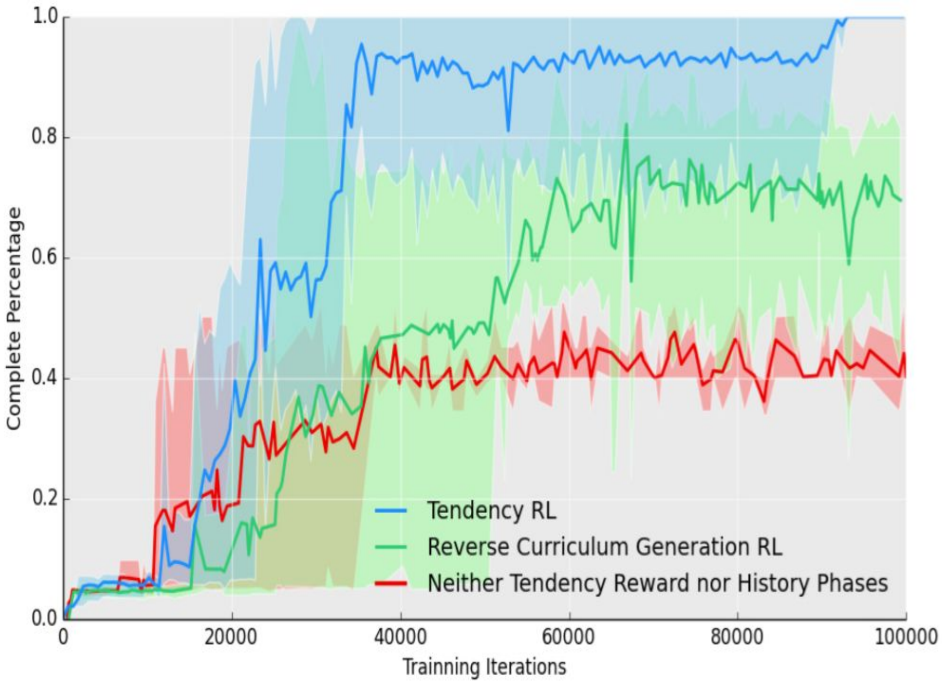


Motivation & RL Weaknesses

- Reward Design
- Robustness of Algorithms







RL in Noisy Environments !

Related Works

Robust Reinforcement Learning

- **Adversarial manipulations** in RL policy
- **Robust policy** capable of withstanding perturbed observations or transferring to unseen environments
- RL algorithms with **uncertainty in models** (states)

Learning from noisy data (supervised Learning)

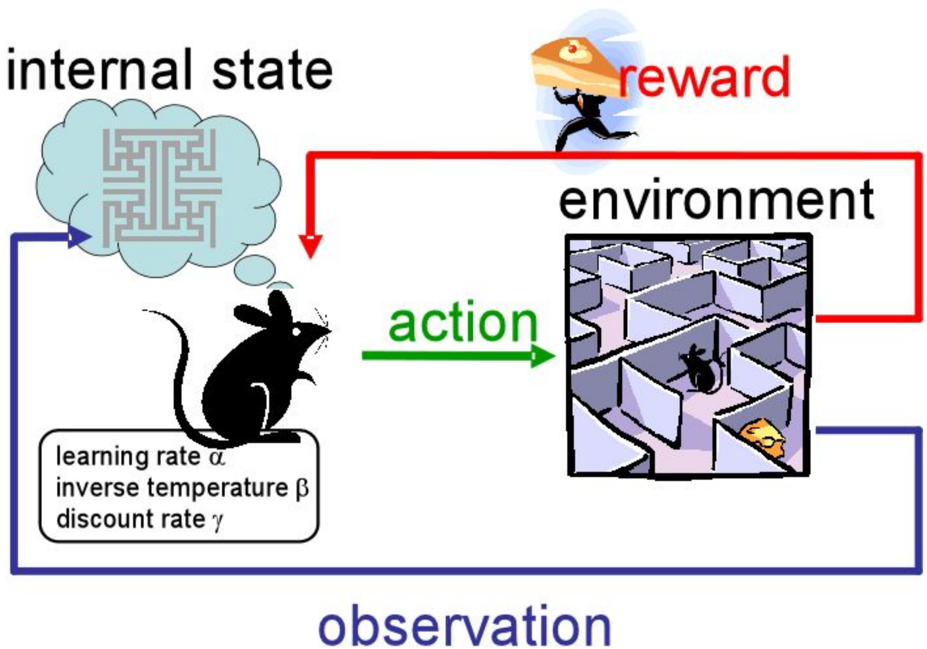
- Define unbiased surrogate loss functions
- Recover the true loss using the knowledge of the noise

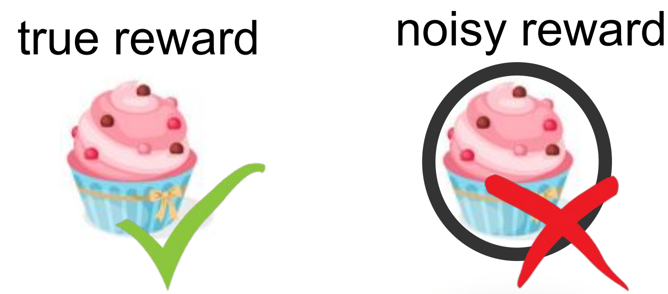
RL with a Corrupted Reward Channel (DeepMind, 2017)

- **No Free Lunch Theorem**: without any assumption about what the reward corruption is, all agents can be essentially lost

"Perturbed" Reward

"No Free Lunch" Theorem [Everitt et. al., 2017]:
Without any assumption about what the reward corruption is, all agents can be essentially lost.





$$\mathbf{C} = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}$$

r1 could be corrupted into r2 with a probability of 0.2 and <u>so does r2</u>.

Perturbed Reward in RL

- **MDP with perturbed reward** $\tilde{\mathcal{M}} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, C, \mathcal{P}, \gamma \rangle$
  - Instead of observing $r_t \in R$ at each time t directly, our RL agent only observes a perturbed version of r_t, denoting as $\tilde{r}_t \in \tilde{R}$.
  - The generation of $\tilde{r}$ follows a certain function $C: S \times R \rightarrow \tilde{R}$

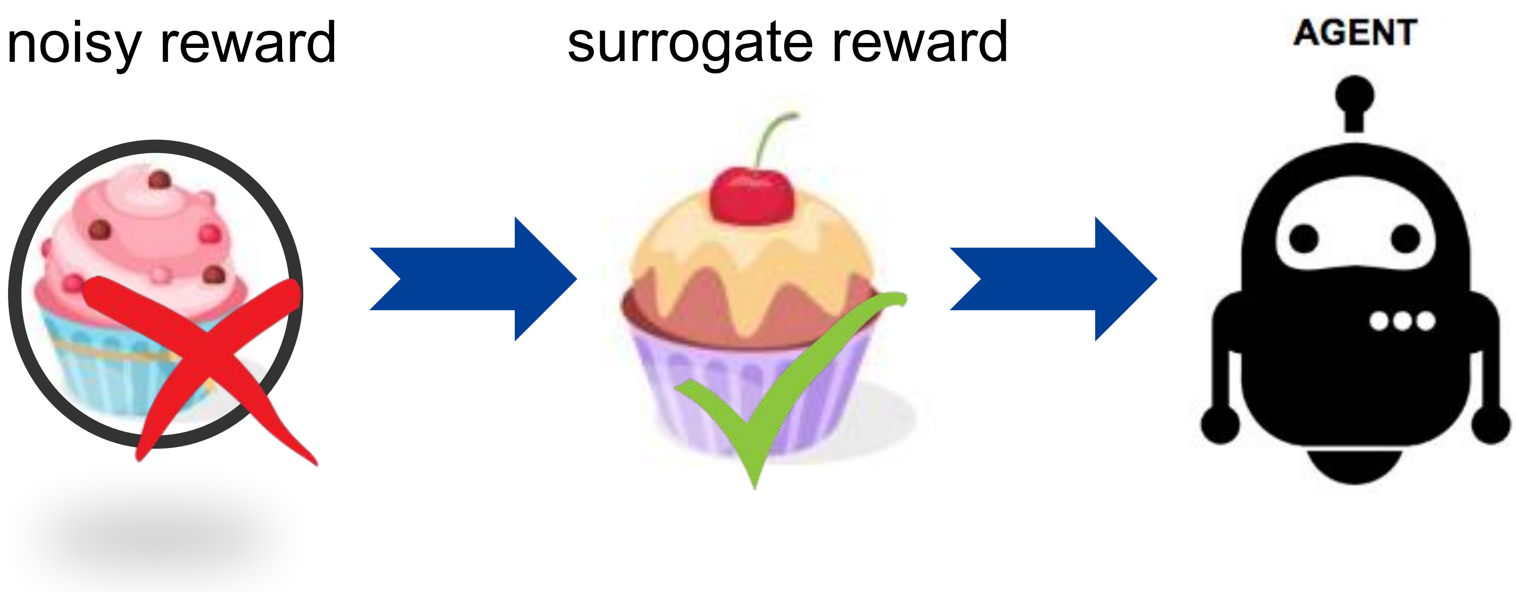
- **Noise Rate**
  - $e_+ = \mathbb{P}(\tilde{r}(s_t, a_t, s_{t+1}) = r_- | r(s_t, a_t, s_{t+1}) = r_+)$
  - $e_- = \mathbb{P}(\tilde{r}(s_t, a_t, s_{t+1}) = r_+ | r(s_t, a_t, s_{t+1}) = r_-)$
  - $c_{j,k} = \mathbb{P}(\tilde{r}_t = R_k | r_t = R_j)$

r: true reward	$\tilde{r}$: noisy reward	$\hat{r}$: surrogate reward	C: confusion matrix

Unbiased Estimator of True Reward

true reward — surrogate reward

$\mathbb{E}(\,r =$  $) = \mathbb{E}(\,\hat{r} =$  $)$



Unbiased Estimator of True Reward

- **Lemma 1.** *Let r be bounded. Then, if we define,*

binary reward $\mathcal{R} = \{r_-, r_+\}$

$$\hat{r}(s_t, a_t, s_{t+1}) := \begin{cases} \frac{(1-e_-)\cdot r_+ - e_+ \cdot r_-}{1-e_+-e_-} & (\tilde{r}(s_t, a_t, s_{t+1}) = r_+) \\ \frac{(1-e_+)\cdot r_- - e_- \cdot r_+}{1-e_+-e_-} & (\tilde{r}(s_t, a_t, s_{t+1}) = r_-) \end{cases}$$

we have for any $r(s_t, a_t, s_{t+1})$, $\mathbb{E}_{\tilde{r}|r}[\hat{r}(s_t, a_t, s_{t+1})] = r(s_t, a_t, s_{t+1})$.

Define $\hat{\mathbf{R}} := [\hat{r}(\tilde{r} = R_0), \hat{r}(\tilde{r} = R_1), ..., \hat{r}(\tilde{r} = R_{M-1})]$, where $\hat{r}(\tilde{r} = R_m)$ denotes the value of the surrogate reward when the observed reward is R_k. Let $\mathbf{R} = [R_0; R_1; \cdots; R_{M-1}]$ be the bounded reward matrix with M values. We have the following results:

- **Lemma 2.** *Suppose $\mathbf{C}_{M\times M}$ is invertible. With defining:*

multi-outcome setting

$$\hat{\mathbf{R}} = \mathbf{C}^{-1} \cdot \mathbf{R}$$

we have for any $r(s_t, a_t, s_{t+1})$, $\mathbb{E}_{\tilde{r}|r}[\hat{r}(s_t, a_t, s_{t+1})] = r(s_t, a_t, s_{t+1})$.

r: true reward	$\tilde{r}$: noisy reward	$\hat{r}$: surrogate reward	C: confusion matrix

Unbiased Estimator of True Reward

Algorithm 1 Reward Robust RL (sketch)

Input: $\tilde{\mathcal{M}}, \alpha, \beta, \tilde{R}(s,a)$
Output: $Q(s), \pi(s,t)$
Initialize value function $Q(s,a)$ arbitrarily
while Q is not converged **do**
  Initialize state $s \in \mathcal{S}$
  while s is not terminal **do**
    Choose a from s using policy derived from Q
    Take action a, observe s' and noisy reward $\tilde{r}$
① **if** collecting enough $\tilde{r}$ for every $\mathcal{S} \times \mathcal{A}$ pair **then**
      Get predicted true reward $\bar{r}$ using majority voting
      Estimate confusion matrix $\tilde{\mathbf{C}}$ based on $\tilde{r}$ and $\bar{r}$ (Eqn. 4)
② Obtain surrogate reward $\dot{r}$ ($\hat{\mathbf{R}} = (1-\eta)\cdot\mathbf{R} + \eta\cdot\mathbf{C}^{-1}\mathbf{R}$)
    Update Q using surrogate reward
    $s \leftarrow s'$
return $Q(s)$ and $\pi(s)$

$$\bar{r}(s,a) = \underset{R_i \in \mathcal{R}}{\arg\max}\ \#[\tilde{r}(s,a) = R_i],$$

$$\tilde{c}_{i,j} = \frac{\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \#\left[\tilde{r}(s,a) = R_j | \bar{r}(s,a) = R_i\right]}{\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \#[\bar{r}(s,a) = R_i]}$$

① estimate confusion matrices

② calculate surrogate rewards

r: true reward	$\tilde{r}$: noisy reward	$\hat{r}$: surrogate reward	C: confusion matrix

Theorems (Q-Learning)

1. Convergence

收敛性 & 样本复杂度: Q-Learning

Theorem 1. *Given a finite MDP, denoting as* $\hat{\mathcal{M}} = \langle \mathcal{S}, \mathcal{A}, \hat{\mathcal{R}}, \mathcal{P}, \gamma \rangle$, *the* Q*-learning algorithm with surrogate rewards, given by the update rule,*

$$Q_{t+1}(s_t, a_t) = (1 - \alpha_t)Q(s_t, a_t) + \alpha_t \left[\hat{r}_t + \gamma \max_{b \in \mathcal{A}} Q(s_{t+1}, b) \right], \tag{3}$$

converges w.p.1 to the optimal Q*-function as long as* $\sum_t \alpha_t = \infty$ *and* $\sum_t \alpha_t^2 < \infty$.

2. Sample complexity

Theorem 2. *(Upper Bound) Let* $r \in [0, R_{\max}]$ *be bounded reward,* $\mathbf{C}$ *be an invertible reward confusion matrix with* $\det(\mathbf{C})$ *denoting its determinant. For an appropriate choice of* m*, the Phased* Q*-Learning algorithm calls the generative model* $G(\hat{\mathcal{M}})$ $O\left(\frac{|\mathcal{S}||\mathcal{A}|T}{\epsilon^2(1-\gamma)^2\det(\mathbf{C})^2} \log \frac{|\mathcal{S}||\mathcal{A}|T}{\delta}\right)$ *times in* T *epochs, and returns a policy such that for all state* $s \in \mathcal{S}$, $|V_\pi(s) - V^*(s)| \leq \epsilon$, $\epsilon > 0$, *w.p.* $\geq 1 - \delta$, $0 < \delta < 1$.

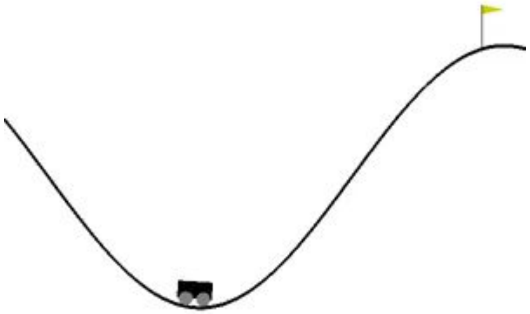
3. Variance

Theorem 3. *Let* $r \in [0, R_{\max}]$ *be bounded reward and confusion matrix* $\mathbf{C}$ *is invertible. Then, the variance of surrogate reward* $\hat{r}$ *is bounded as follows:* $\mathbf{Var}(r) \leq \mathbf{Var}(\hat{r}) \leq \frac{M^2}{\det(\mathbf{C})^2} \cdot R_{\max}^2$.

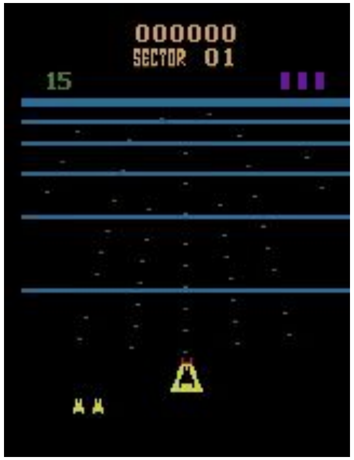
Experiments (OpenAI Gym)

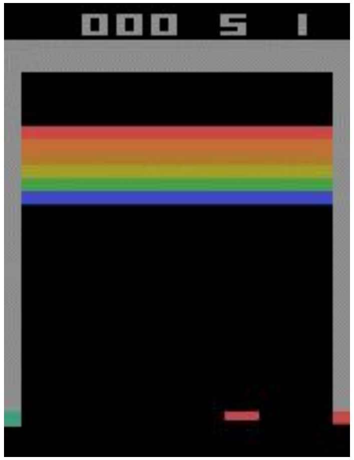
1. Classic Control Game

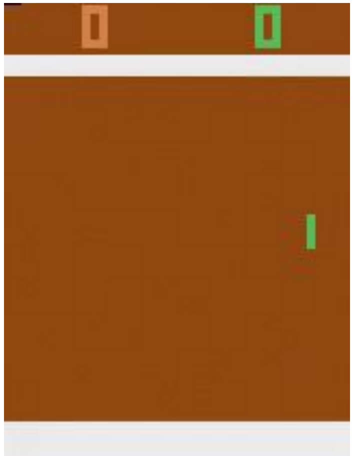


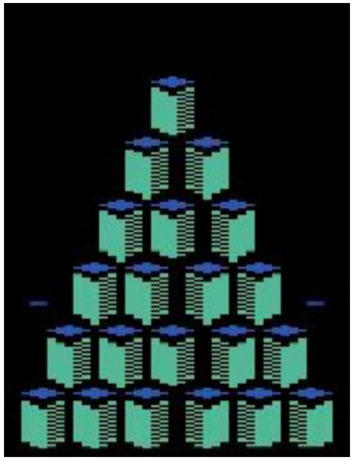


2. Atari-2600 Game







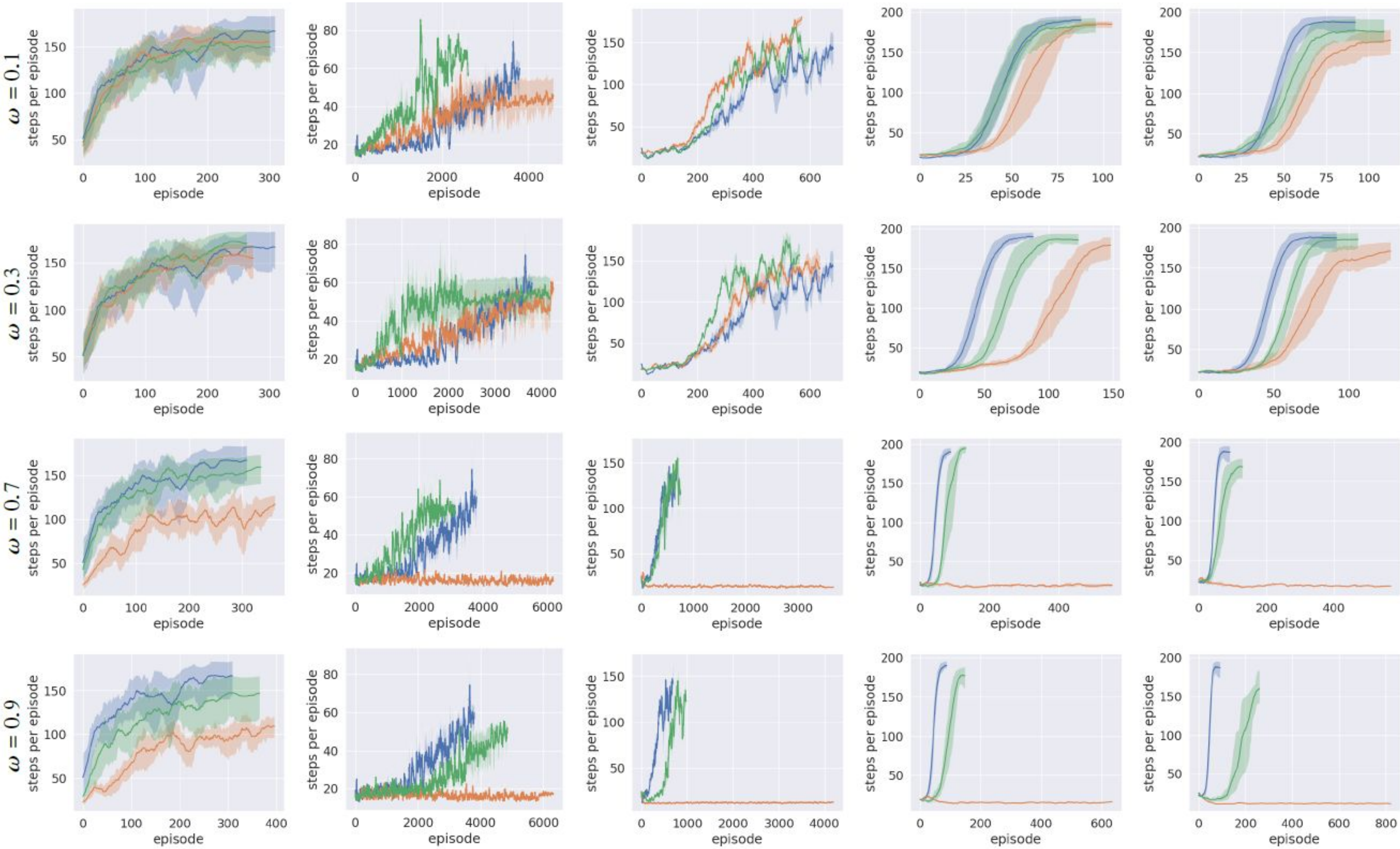


Experiments (OpenAI Gym)

RL Algorithms

Environment	RL Algorithm	Abbreviation
CartPole	Q-Learning	Q-Learn
	Cross Entropy Method	CEM
	Deep State-Action-Reward-State-Action	SARSA
	Deep Q Network	DQN
	Dueling Deep Q Network	Dueling-DQN
Pendulum	Deep Deterministic Policy Gradient	DDPG
	Continuous DQN	NAF
Atari-2600	Proximal Policy Optimization	PPO

Results (CartPole)

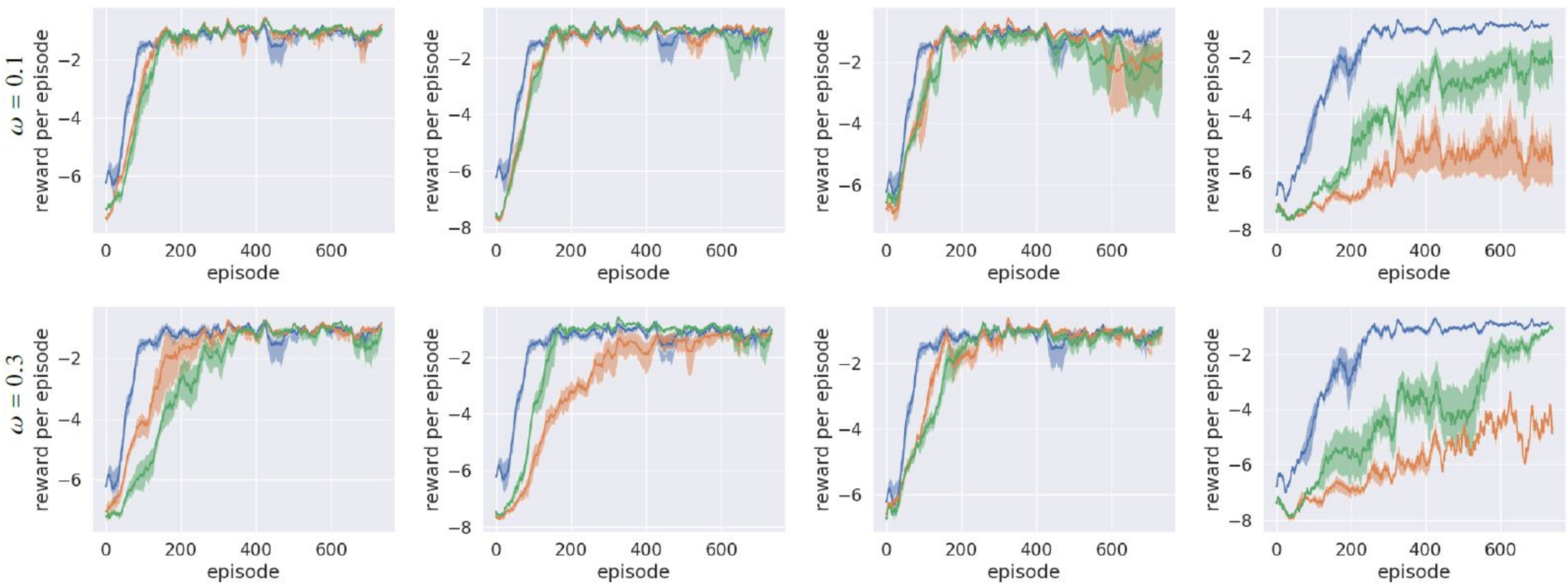


(a) Q-Learning (b) CEM (c) SARSA (d) DQN (e) Dueling-DQN

true reward noisy reward surrogate reward

Results (Pendulum)

- **Continuous States & Rewards => Discretization**
- **Symmetric & Asymmetric noise**

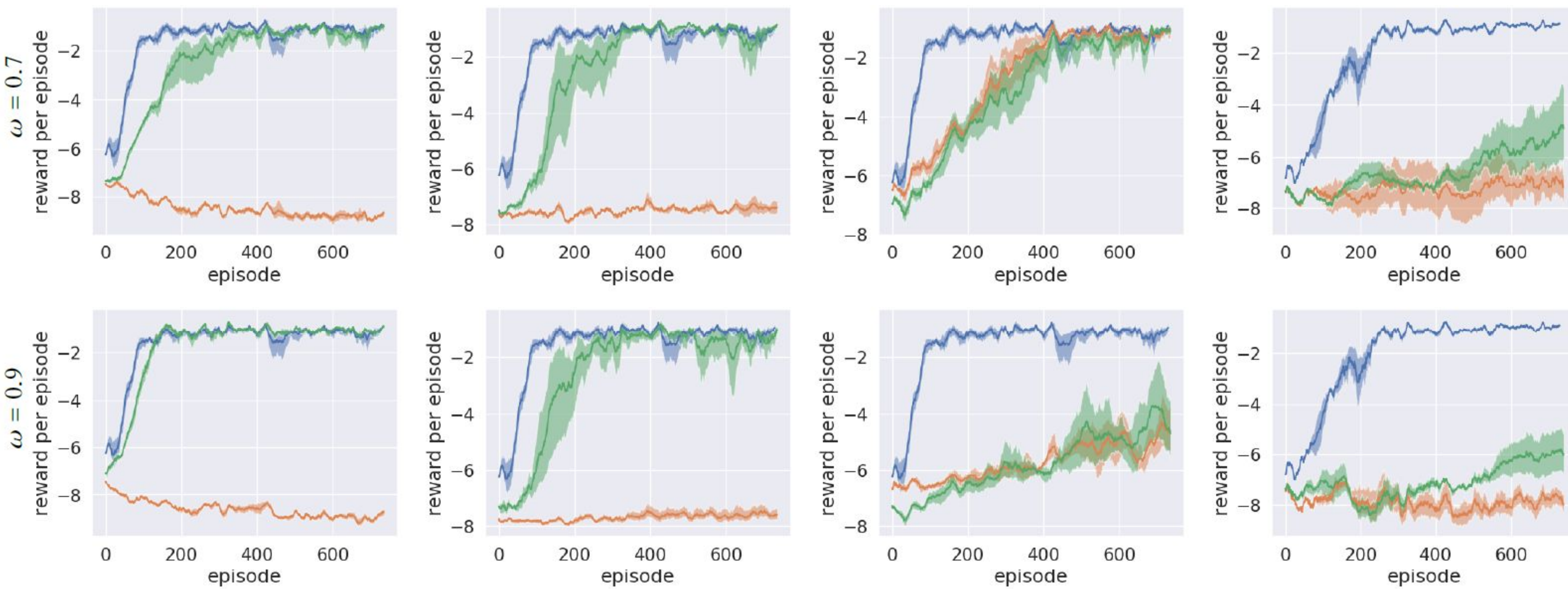


(a) DDPG (symmetric) (b) DDPG (rand-one) (c) DDPG (rand-all) (d) NAF (rand-all)

true reward noisy reward surrogate reward

Results (Pendulum)

- **Continuous States & Rewards => Discretization**
- **Symmetric & Asymmetric noise**



(a) DDPG (symmetric) (b) DDPG (rand-one) (c) DDPG (rand-all) (d) NAF (rand-all)

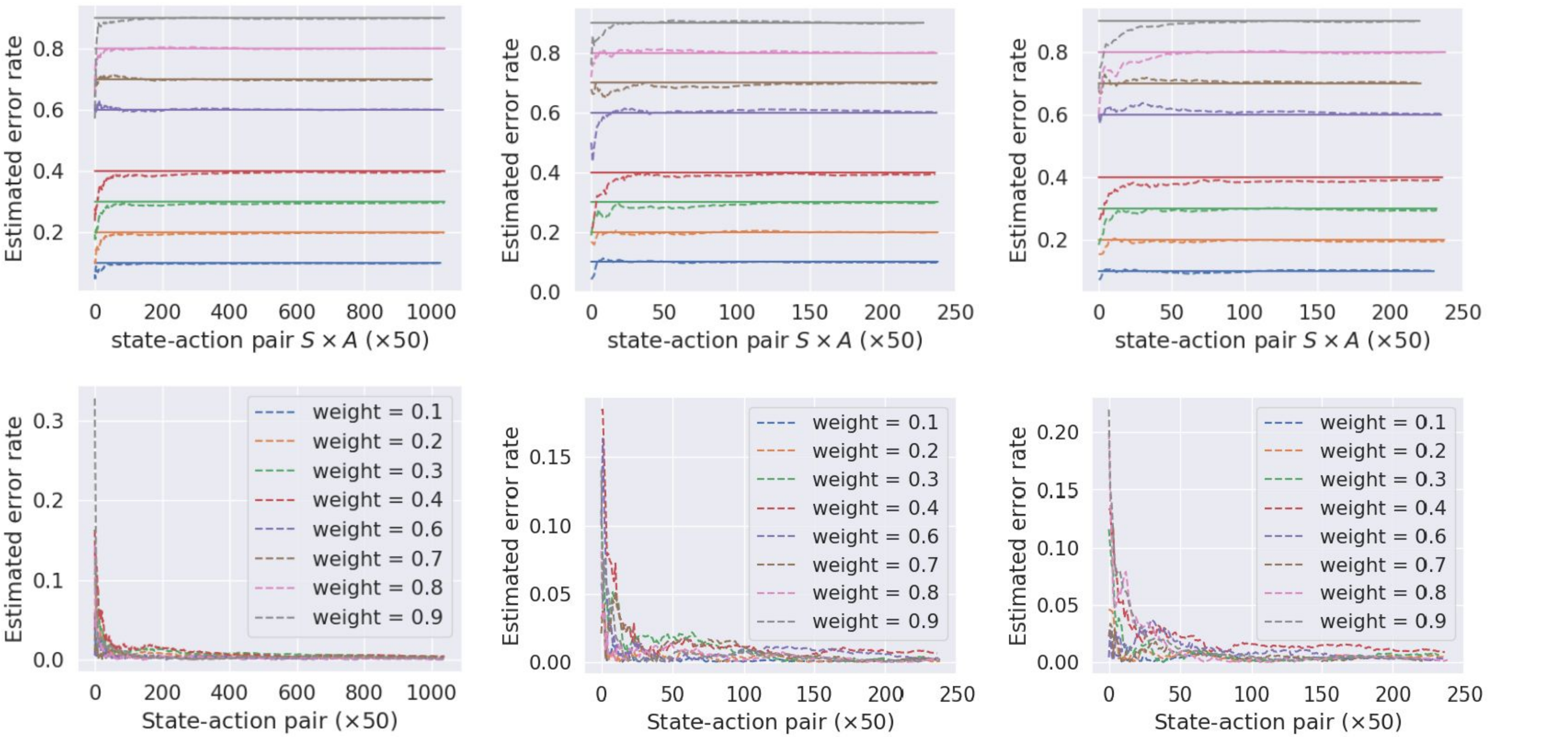
true reward noisy reward surrogate reward

Results (CartPole + Pendulum)

Table 5: Complete average scores of various RL algorithms on CartPole and Pendulum with noisy rewards ($\tilde{r}$) and surrogate rewards under known ($\hat{r}$) or estimated ($\dot{r}$) confusion matrices.

Noise Rate	Reward	Q-Learn	CEM	SARSA	DQN	DDQN	DDPG	NAF
$\omega = 0.1$	$\tilde{r}$	170.0	98.1	165.2	187.2	**187.8**	-1.03	-4.48
	$\hat{r}$	165.8	**108.9**	**173.6**	**200.0**	181.4	**-0.87**	**-0.89**
	$\dot{r}$	**181.9**	99.3	171.5	**200.0**	185.6	-0.90	-1.13
$\omega = 0.3$	$\tilde{r}$	134.9	28.8	144.4	173.4	168.6	-1.23	-4.52
	$\hat{r}$	149.3	**85.9**	152.4	175.3	198.7	**-1.03**	**-1.15**
	$\dot{r}$	**161.1**	81.8	**159.6**	**186.7**	**200.0**	-1.05	-1.36
$\omega = 0.7$	$\tilde{r}$	56.6	19.2	12.6	17.2	11.8	-8.76	-7.35
	$\hat{r}$	**177.6**	**87.1**	151.4	185.8	**195.2**	**-1.09**	**-2.26**
	$\dot{r}$	172.1	83.0	**174.4**	**189.3**	191.3	–	–

Results (Estimation of Confusion Matrices)



(c) SARSA (d) DQN (e) Dueling-DQN

Figure 7: Estimation analysis from five *reward robust* RL algorithms (see Algorithm 3) on CartPole game. The upper figures are the convergence curves of estimated error rates (from 0.1 to 0.9)

Results (Atari)

Table 6: Complete average scores of PPO on five selected Atari games with noisy rewards ($\tilde{r}$) and surrogate rewards under known ($\hat{r}$) or estimated ($\dot{r}$) noise rates.

Noise Rate	Reward	Lift (↑)	Mean	Alien	Carnival	Phoenix	MsPacman	Seaquest
$\omega = 0.1$	$\tilde{r}$	–	2048.3	1835.1	1239.3	4609.0	1709.1	849.2
	$\hat{r}$	**70.4%↑**	3489.6	1737.0	3966.8	**7586.4**	**2547.3**	1610.6
	$\dot{r}$	**84.6%↑**	**3781.3**	**2844.1**	**5515.0**	5668.8	2294.5	**2333.9**
$\omega = 0.3$	$\tilde{r}$	–	1115.3	538.2	919.9	2600.3	1109.6	408.7
	$\hat{r}$	**119.8%↑**	**2451.7**	**1668.6**	**4220.1**	**4171.6**	1470.3	**727.8**
	$\dot{r}$	**80.8%↑**	2016.0	1542.9	4094.3	2589.1	**1591.2**	262.4
$\omega = 0.7$	$\tilde{r}$	–	298.7	495.2	380.3	126.5	491.6	0.0
	$\hat{r}$	**757.4%↑**	**2561.1**	**1805.9**	4088.9	**4970.4**	1447.8	**492.5**
	$\dot{r}$	**648.9%↑**	2236.9	1618.0	**4529.2**	2792.1	**1916.7**	328.5
$\omega = 0.9$	$\tilde{r}$	–	619.8	557.8	6.3	1410.9	535.4	588.8
	$\hat{r}$	**508.7%↑**	**3772.8**	**1958.7**	**5664.2**	**6758.7**	**2515.1**	1707.2
	$\dot{r}$	**450.2%↑**	3409.9	1865.2	5515.0	5388.1	2492.6	**1788.6**

Results (Variance Reduction)

Theorem 3. *Let $r \in [0, R_{\max}]$ be bounded reward and confusion matrix $\mathbf{C}$ is invertible. Then, the variance of surrogate reward $\hat{r}$ is bounded as follows:* $\mathbf{Var}(r) \leq \mathbf{Var}(\hat{r}) \leq \frac{M^2}{\det(\mathbf{C})^2} \cdot R_{\max}^2$.

$\mathbb{E}(r =$  $) = \mathbb{E}(\hat{r} =$  $)$

true reward

surrogate reward

$$\mathbf{Var}(r) \leq \mathbf{Var}(\hat{r}) \leq \frac{4R_{\max}^2}{(1-e_+-e_-)^2}$$

$$e_- + e_+ \rightarrow 1$$

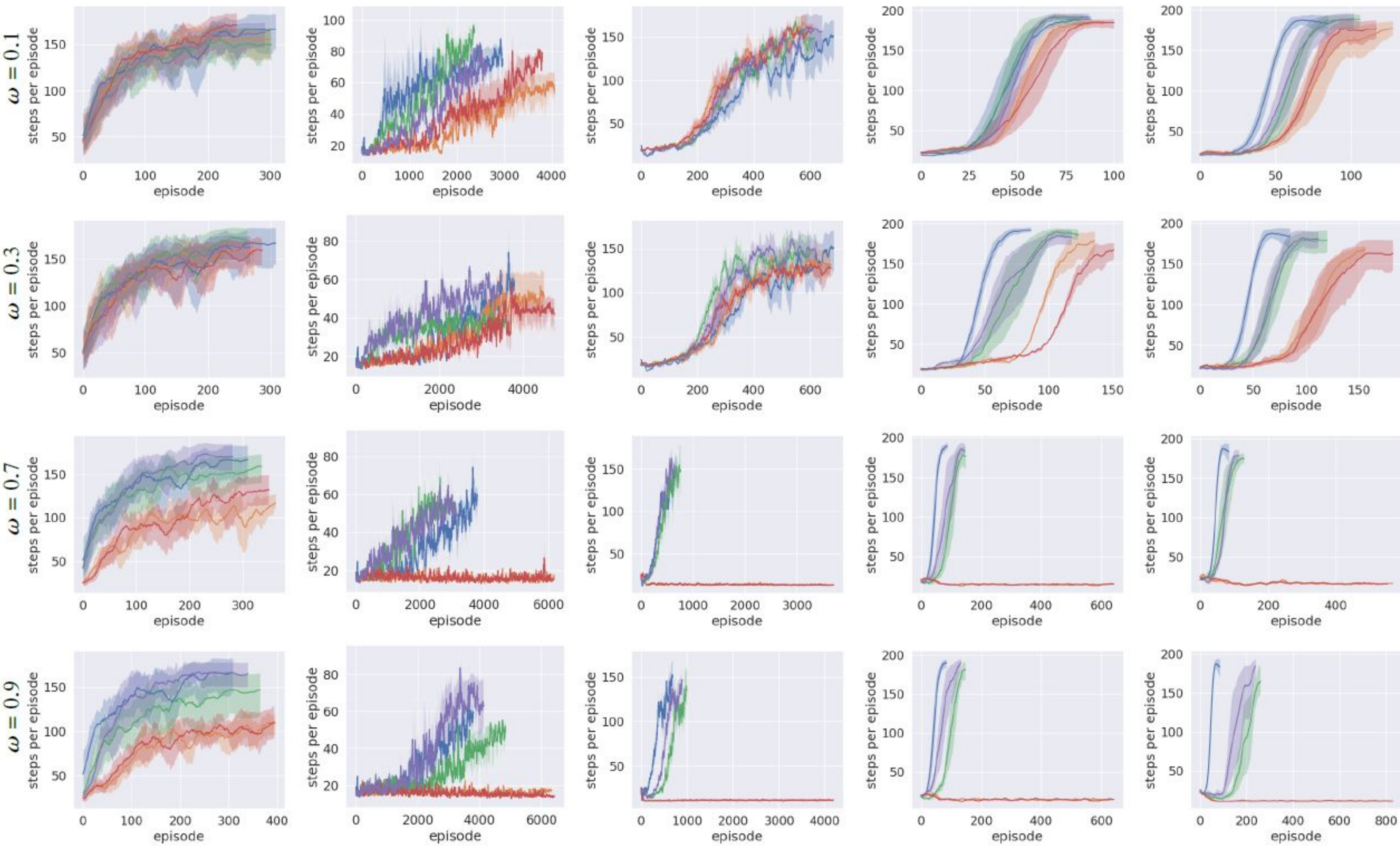
unbiased proxy but larger variance!

1. Linear Combination $\mathbf{R}_{proxy} = \eta\mathbf{R} + (1-\eta)\hat{\mathbf{R}}$

2. Variance Reduction（unbiased noise）

- Why not VRT + our unbiased surrogate rewards?

Results (Variance Reduction)



(a) Q-Learning (b) CEM (c) SARSA (d) DQN (e) DDQN

true reward noisy reward surrogate reward VRT ours + VRT

Summary (RL with Perturbed Reward)

- An unbiased reward estimator aided robust RL framework (**biased noise**)
- Theoretical analysis of proposed method
  - Proof of unbiasedness
  - Convergence (Q-Learning)
  - Sample Complexity (Phased Q-Learning)
  - Variance of reward proxy
- Estimation of confusion matrices
  - Simple but efficient
  - Adaptive to continuous setting (rewards or states)
- Validations on OpenAI Gym (quantitative & qualitative)

Future direction & Limitedness

- Continuous states or rewards (involve more assumptions)
- State-dependent case
  - We maintain confusion matrices for each state, which is costly
- Adversarial noise (not learnable …)

What should we do?

- **More Accurate Sensors + More Robust Algorithms!**
  (Safety) **(Security)**

- **What I did in last 1.5 years:**
- Self-Driving:
  - Policy learning: 2D + LiDAR (cheaper but still very expensive)
- Tackling with Noisy Data:
  - Reinforcement Learning with Perturbed Reward
- **Attack & Defend our ML/DL Systems:**
  - **Arms Race in Adversarial Machine Learning (AML)**
  - More reliable, interpretable DL training scheme (Information Bottleneck, IB)
  - Min-Max Optimization in AML (across domains)

What is Adv-ML (AML)?

- Unfortunately, ML/DL models are highly vulnerable to slight adversarial perturbations in various applications!
- An arms race between adversarial attacks and defenses.

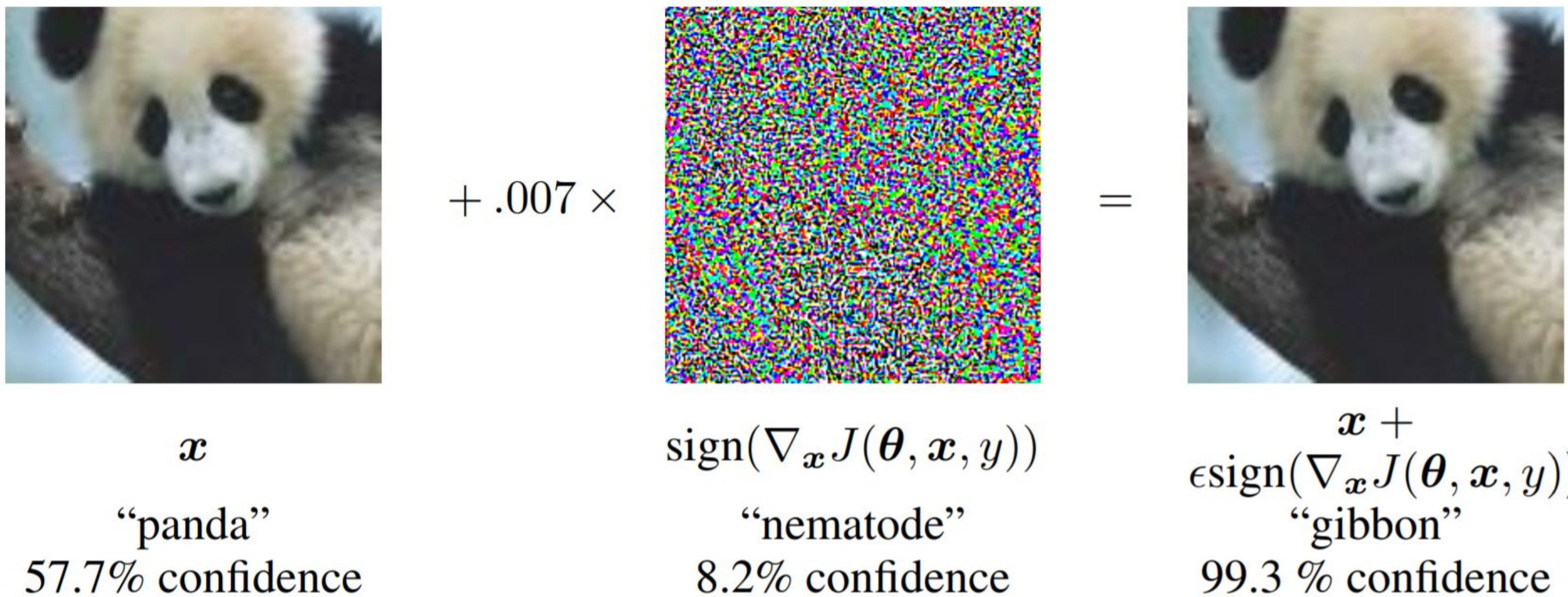
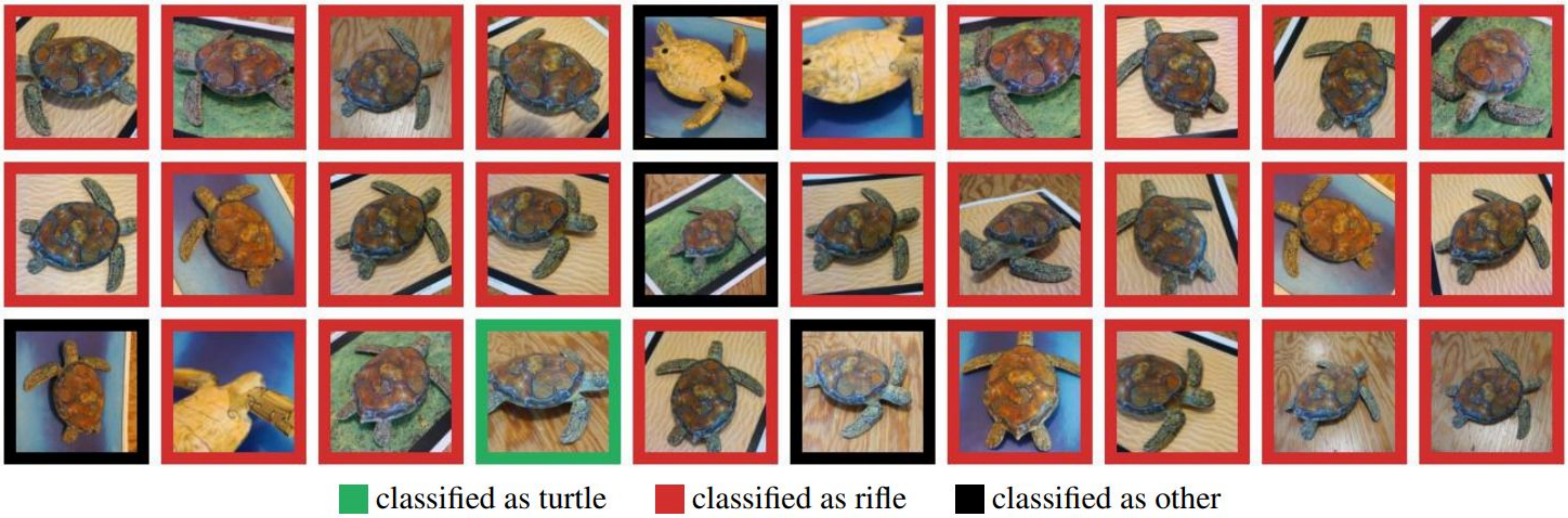


Figure: A demonstration of adversarial example applied to GoogLeNet using fast gradient sign method (Goodfellow et al., 2014). $\boldsymbol{\delta} = \text{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})\right)$

Goodfellow et al. "Explaining and harnessing adversarial examples." ICLR 2015.

Challenges: Adversarial Examples are everywhere!

- Adversarial examples are easy and cheap to craft!
  - FGSM, JSMA, DeepFool, C&W, BIM, PGD, ...
  - 2D, 3D, RL, Speech, Text, ...
- Adversarial examples can be realistic!
  - white-box, black-box (transferability) ...
  - robust physical attack, real-world messy data



Athalye, et al. "Synthesizing Robust Adversarial Examples." ICML 2018.

Formulation - Adversarial Examples

Adversarial Examples

A ϵ-bounded adversarial example x' of x for a neural network f fulfills:

(1) $f(x') \neq o(x)$, where $o(\cdot)$ is the **oracle**;

(2) x' is created by an attack algorithm $\mathcal{A}$ which maps x to x';

(3) $||x - x'|| \leq \epsilon$, where $|| \cdot ||$ is a norm on $\mathcal{X}$ and $\epsilon > 0$.

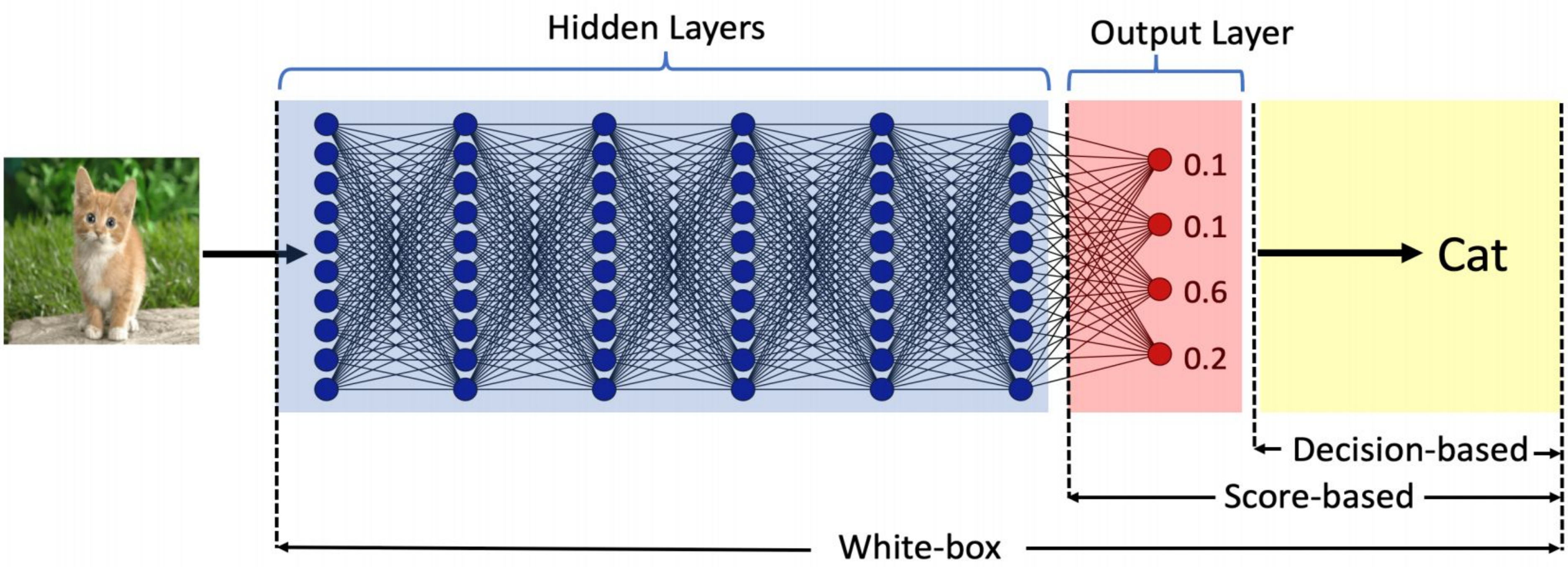
- Threat models
  - white-box, black-box (adaptive or non-adaptive)
  - gradient-based, score-based, decision-based ...
- Intriguing properties
  - transferability across domains (e.g., datasets, models, transformations)
  - universality (e.g., images, models)

Wang et al. “One Bit Matters: Understanding Adversarial Examples as the Abuse of Redundancy”. 2018

Threat Models

- Threat models
  - white-box, black-box (adaptive, non-adaptive, strict)
  - gradient-based, transfer-based, score-based, decision-based

less information
(*difficulty*)



Chen et al. "HopSkipJumpAttack: A Query-Efficient Decision-Based Attack." 2019

Threat Models

- Threat models
  - white-box, black-box (adaptive, non-adaptive, strict)
  - gradient-based, transfer-based, score-based, decision-based
  - untargeted & targeted
    - untargeted – mislead the classifier to predict any labels other than the ground truth
    - targeted – mislead the classifier to predict a target label for an image

	Gradient-based Model M	**Transfer-based** Training Data T	**Score-based** Detailed Model Prediction Y (e.g. probabilities or logits)	**Decision-based** Final Model Prediction Y_{max} (e.g. max class label)
Untargeted Flip to any label	FGSM, DeepFool	FGSM Transfer	Local Search	★
				this work (Boundary Attack)
Targeted Flip to target label	L-BFGS-B, Houdini, JSMA, Carlini & Wagner, Iterative Gradient Descent	Ensemble Transfer	ZOO	★

less information

Wieland et al. "Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models" ICLR 2018.

Fast Gradient Method (FGM)

Fast Gradient Sign Method (FGSM)

- Untargeted:

$$\mathbf{x}_{\text{adv}} = \mathbf{x} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}} \ell_{f^s}(\mathbf{x}, y))$$

- Targeted:

$$\mathbf{x}_{\text{adv}} = \mathbf{x} - \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}} \ell_{f^s}(\mathbf{x}, T))$$

Iterative FGM (BIM, PGD)

- Untargeted:

$$\mathbf{x}_{\text{adv}}^{t+1} = \Pi_{\mathcal{H}}(\mathbf{x}_{\text{adv}}^{t} + \alpha \cdot \text{sign}(\nabla_{\mathbf{x}_{\text{adv}}^{t}} \ell_{f^s}(\mathbf{x}_{\text{adv}}^{t}, y)))$$

- Targeted:

$$\mathbf{x}_{\text{adv}}^{t+1} = \Pi_{\mathcal{H}}(\mathbf{x}_{\text{adv}}^{t} - \alpha \cdot \text{sign}(\nabla_{\mathbf{x}_{\text{adv}}^{t}} \ell_{f^s}(\mathbf{x}_{\text{adv}}^{t}, T)))$$

Goodfellow et al. “Explaining and harnessing adversarial examples.” ICLR 2015.
Madry et al. “Towards Deep Learning Models Resistant to Adversarial Attacks” ICLR 2018.

Carlini & Wagner (C&W)

Optimization problem:

$$\begin{aligned} \text{minimize} \quad & \mathcal{D}(x, x+\delta) \\ \text{such that} \quad & C(x+\delta) = t \\ & x + \delta \in [0,1]^n \end{aligned}$$

$$\begin{aligned} \text{minimize} \quad & \|\delta\|_p + c \cdot f(x+\delta) \\ \text{such that} \quad & x + \delta \in [0,1]^n \end{aligned}$$

- Projected gradient descent
- Clip gradient descent
- Change of variables

$$\delta_i = \frac{1}{2}(\tanh(w_i) + 1) - x_i$$

$$-1 \leq \tanh(w_i) \leq 1 \quad x + \delta \in [0,1]^n$$

- L2-attack: $\text{minimize} \quad \|\frac{1}{2}(\tanh(w) + 1) - x\|_2^2 + c \cdot f(\frac{1}{2}(\tanh(w) + 1)$
- C&W loss: $f(x') = \max(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa)$

Carlini et al. “Towards Evaluating the Robustness of Neural Networks” IEEE S&P 2017

ZOO (score-based)

Objective function:

$$f(\mathbf{x}) = \max\{\log[F(\mathbf{x})]_{t_0} - \max_{i \neq t_0} \log[F(\mathbf{x})]_i, -\kappa\}$$

$$[F(\mathbf{x})]_k = \frac{\exp([Z(\mathbf{x})]_k)}{\sum_{i=1}^{K} \exp([Z(\mathbf{x})]_i)}, \ \forall\, k \in \{1, \ldots, K\}$$

- Zeroth order optimization on the loss function

- ZOO-Adam

$$\hat{g}_i := \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_i} \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x} - h\mathbf{e}_i)}{2h}$$

- ZOO-Newton

$$\hat{h}_i := \frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}_{ii}^2} \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - 2f(\mathbf{x}) + f(\mathbf{x} - h\mathbf{e}_i)}{h^2}$$

Algorithm 1 Stochastic Coordinate Descent

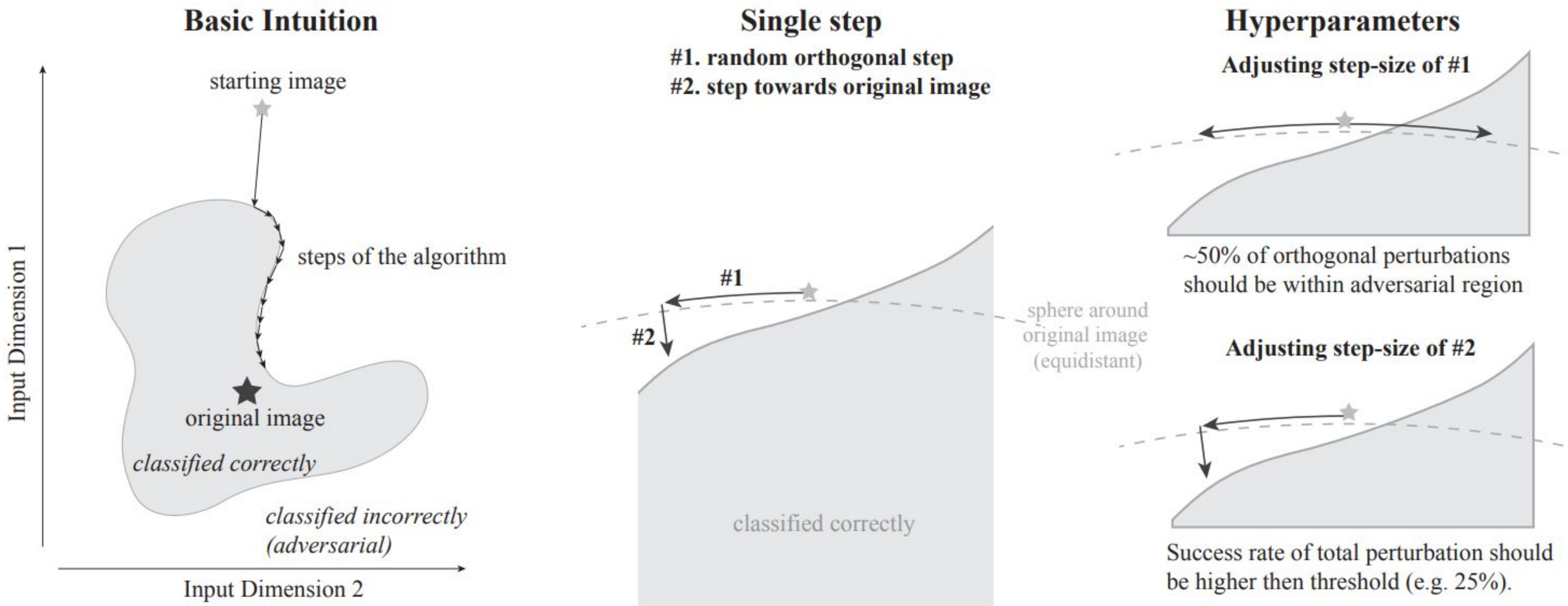
1: **while** not converged **do**
2: Randomly pick a coordinate $i \in \{1, \ldots, p\}$
3: Compute an update δ^* by approximately minimizing

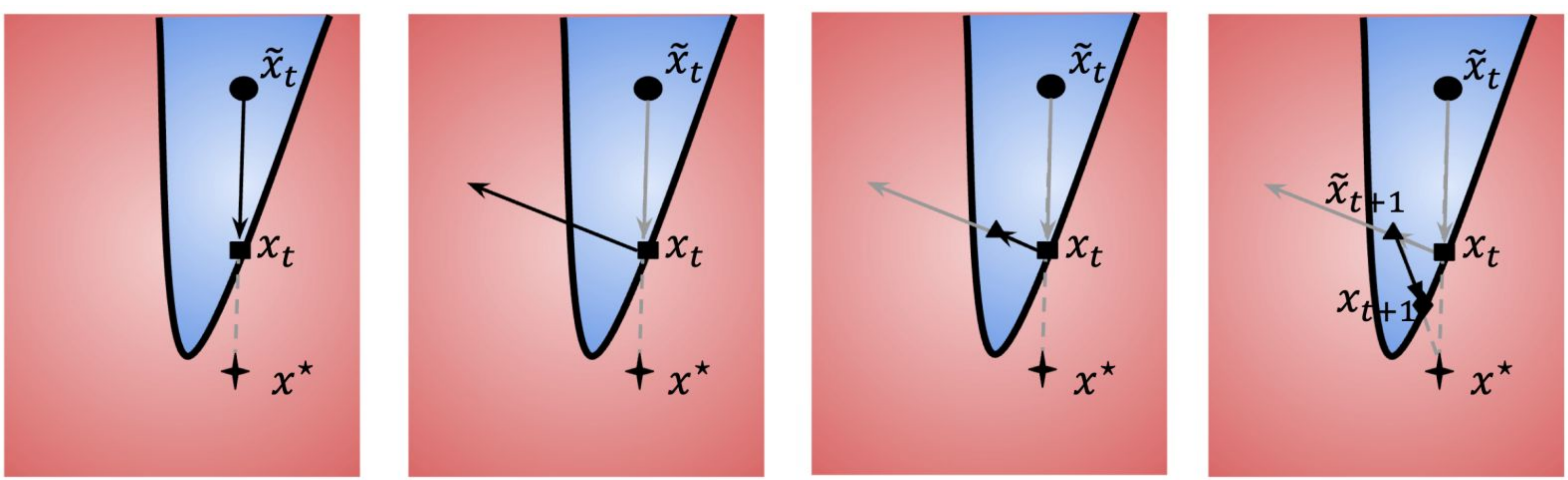
$$\arg\min_{\delta} f(\mathbf{x} + \delta \mathbf{e}_i)$$

4: Update $\mathbf{x}_i \leftarrow \mathbf{x}_i + \delta^*$
5: **end while**

Chen et al. “ZOO: Zeroth Order Optimization based Black-box Attacks to Deep Neural Networks without Training Substitute Models”

Boundary Attack (BA & BA++)

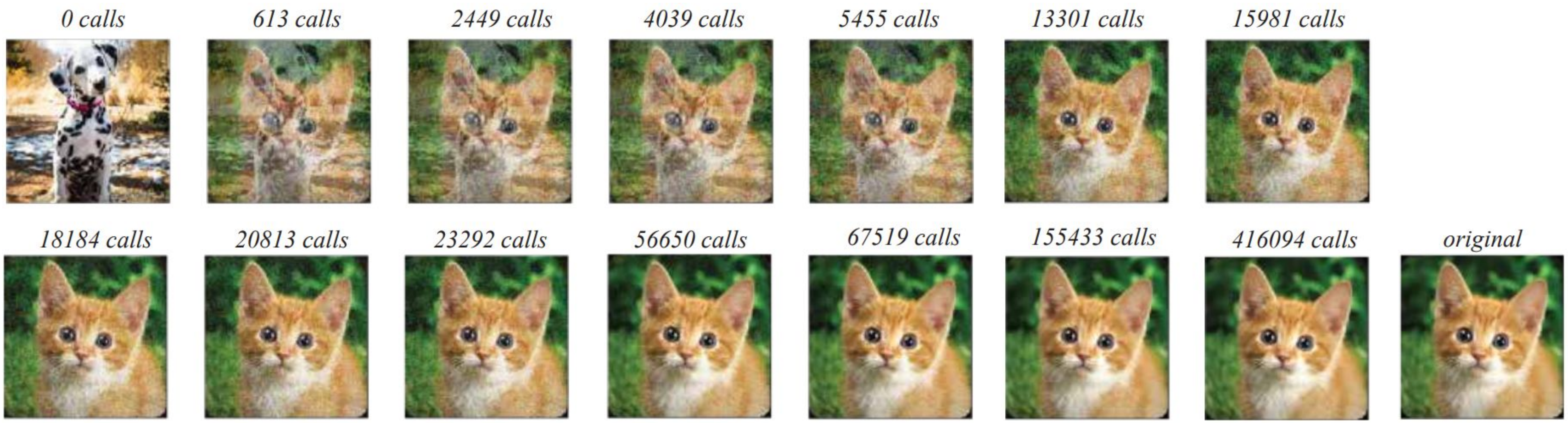




[1] Wieland Brendel, Jonas Rauber, Matthias Bethge. “Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models” ICLR 2018.
[2] Jianbo Chen, Michael I. Jordan, Martin J. Wainwright. “HopSkipJumpAttack: A Query-Efficient Decision-Based Attack.” arXiv 1904.02144.

Boundary Attack (BA & BA++)

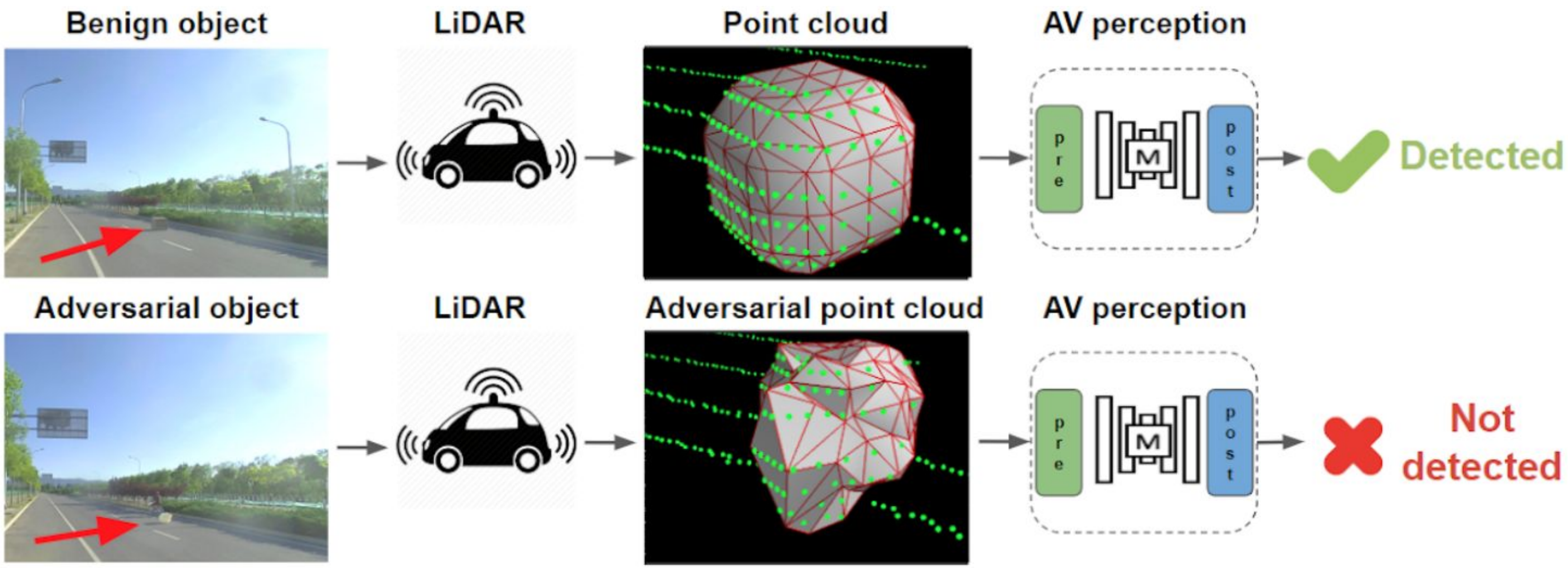
- Only **binary feedback** on the boundary



[1] Wieland Brendel, Jonas Rauber, Matthias Bethge. "Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models" ICLR 2018.

[2] Jianbo Chen, Michael I. Jordan, Martin J. Wainwright. "HopSkipJumpAttack: A Query-Efficient Decision-Based Attack." arXiv 1904.02144.

Robust Physical Attacks



[1] Eykholt et al. “Physical Adversarial Examples for Object Detectors” USENIX WOOT 2018.
[2] Cao et al. “Adversarial Objects Against LiDAR-Based Autonomous Driving Systems” 2019

Defenses

- Adversarial Training (AT) (**SOTA, min-max opt**)
  - augment perturbed data (inserting adv. examples while training)
  - modified objective function:

$$\widetilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1-\alpha) J(\theta, x + \epsilon sign(\nabla_x J(\theta, x, y)), y)$$

  - **Issue:**
    - low transferability (multiple norms, attack-sensitive)
    - efficiency of ensemble adversarial training
    - the training cost is huge!

$\Rightarrow$ **going to address!**

- Gradient Masking
  - gradient-based attacks (non-differentiable models)
  - **Issue:**
    - obfuscated gradients give a false sense of security (**ICML best paper, 2018**)
    - gradient estimation works very well in breaking this kind of defense (EOT)

[1] Madry et al. "Towards Deep Learning Models Resistant to Adversarial Attacks" ICLR 2018.
[2] Athalye et al. "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples" ICML 2018.

Defenses

- Defensive Distillation
  - training two neural networks
    - second is the target network with higher T
  - label smoothing
  - **Issue:**
    - expensive (two step training scheme)
    - does not work for stronger attack such as PGD, C&W

$$F_i(X) = \frac{e^{\frac{z_i(X)}{T}}}{\sum_{i=1}^{|Y|} e^{\frac{z_i(X)}{T}}}$$

- Feature Squeezing
  - reduce the color depth (input complexity)
  - use of a smoothing filter over the images
  - **Issue:**
    - only applicable to small scale datasets such as MNIST, CIFAR-10
    - do harm to the benign accuracy (detection may be a better choice)

[1] Papernot et al. “Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks” IEEE S&P 2016.
[2] Xu et al. “Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks” NDSS 2018.

What should we do?

- **More Accurate Sensors + More Robust Algorithms!**
  (Safety) **(Security)**

- **What I did in last 1.5 years:**
- Self-Driving:
  - Policy learning: 2D + LiDAR (cheaper but still very expensive)
- Tackling with Noisy Data:
  - Reinforcement Learning with Perturbed Reward
- **Attack & Defend our ML/DL Systems:**
  - Arms Race in Adversarial Machine Learning (AML)
  - **More reliable, interpretable DL training scheme (Information Bottleneck, IB)**
  - Min-Max Optimization in AML (across domains)

A Necessary Condition for the Existence of Adv. Examples

Minimal Sufficient Statistic (MSS)

Suppose that X is a sample from a distribution indexed by ground truth Y. A function $T(X)$ is said to be a minimal sufficient statistics (MSS) if

$$T(X) \in \underset{S}{\operatorname{argmin}}\, I(X;\ S(X))$$
$$s.t.\ I\big(Y; S(X)\big) = \max_{T'} I(Y;\ T'(X))$$

i.e., it is a statistic that has smallest MI with X while having largest MI with Y .

Theorem 1 (MSS is necessary for Adv.)

Suppose that Assumption 1 holds and there exist adversarial examples for the neural network $f(\cdot) = g\big(T(\cdot)\big)$. Then, $T(X)$ is not a MSS.

Feature Redundancy

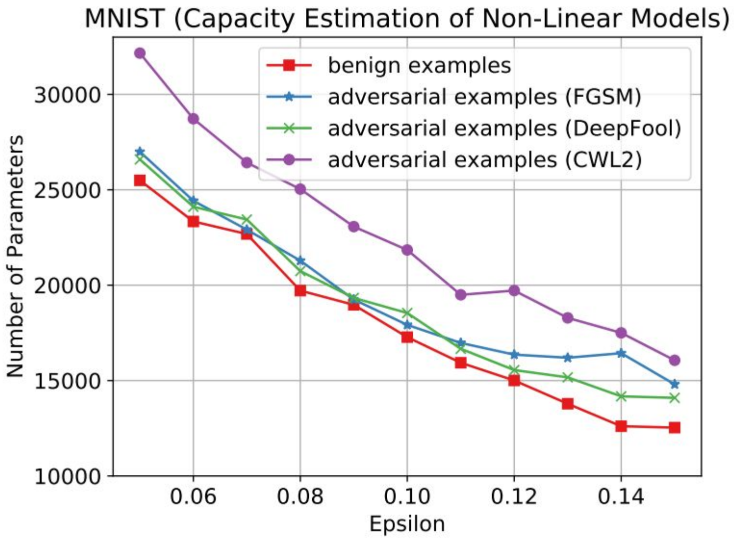
- Does our latent representations are sufficient?

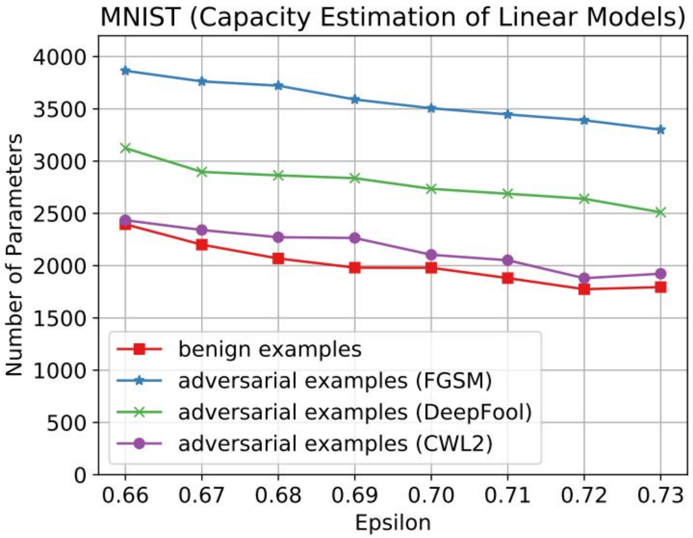
Dataset	Examples	H (MLE)	H (JVHW)	Original Size	Compressed Size
MNIST	**Benign**	**1.741**	**1.887**	**988.89 B**	**431.40 B**
	FGSM (2015)	2.488	2.601	1690.36 B	503.54 B
	DeepFool (2015)	4.844	5.088	1654.99 B	510.41 B
	CW (L_2) (2017a)	4.094	4.301	1159.01 B	437.27 B
CIFAR-10	**Benign**	**9.595**	**7.104**	**1845.98 B**	**741.36 B**
	FGSM (2015)	9.937	7.710	2717.01 B	872.40 B
	DeepFool (2015)	9.675	7.147	1880.41 B	743.02 B
	CW (L_2) (2017a)	9.621	7.113	1850.54 B	741.56 B

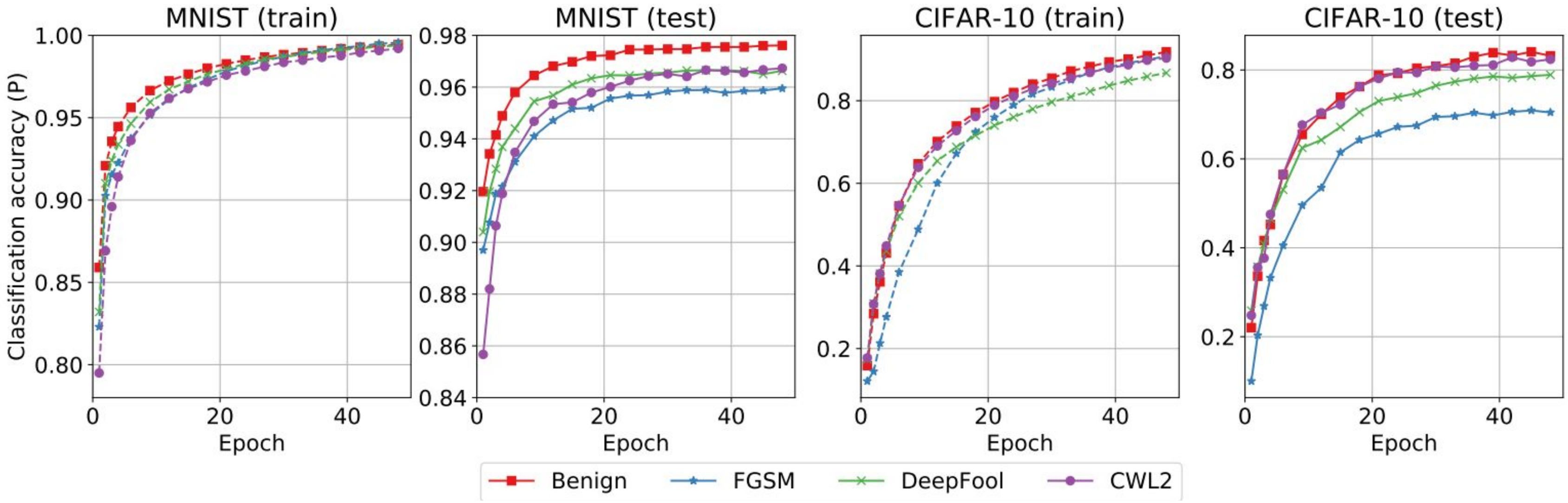
Dataset	Examples	Mean Bits	H (BW)	H (bW)	Compressed Size
IMDB	**Benign**	**4.556**	**0.569**	**0.99775**	**2.235 B**
	FGSM (2015)	4.671	0.584	0.99926	3.027 B
	FGVM (2015)	4.701	0.588	0.99944	3.481 B
	DeepFool (2015)	4.632	0.580	0.99953	3.156 B
Reuters2	**Benign**	**4.946**	**0.618**	**0.99457**	**1.934 B**
	FGSM (2015)	5.032	0.629	0.99712	3.181 B
	FGVM (2015)	5.035	0.629	0.99754	3.237 B
	DeepFool (2015)	5.202	0.650	0.99545	3.301 B

Feature Redundancy

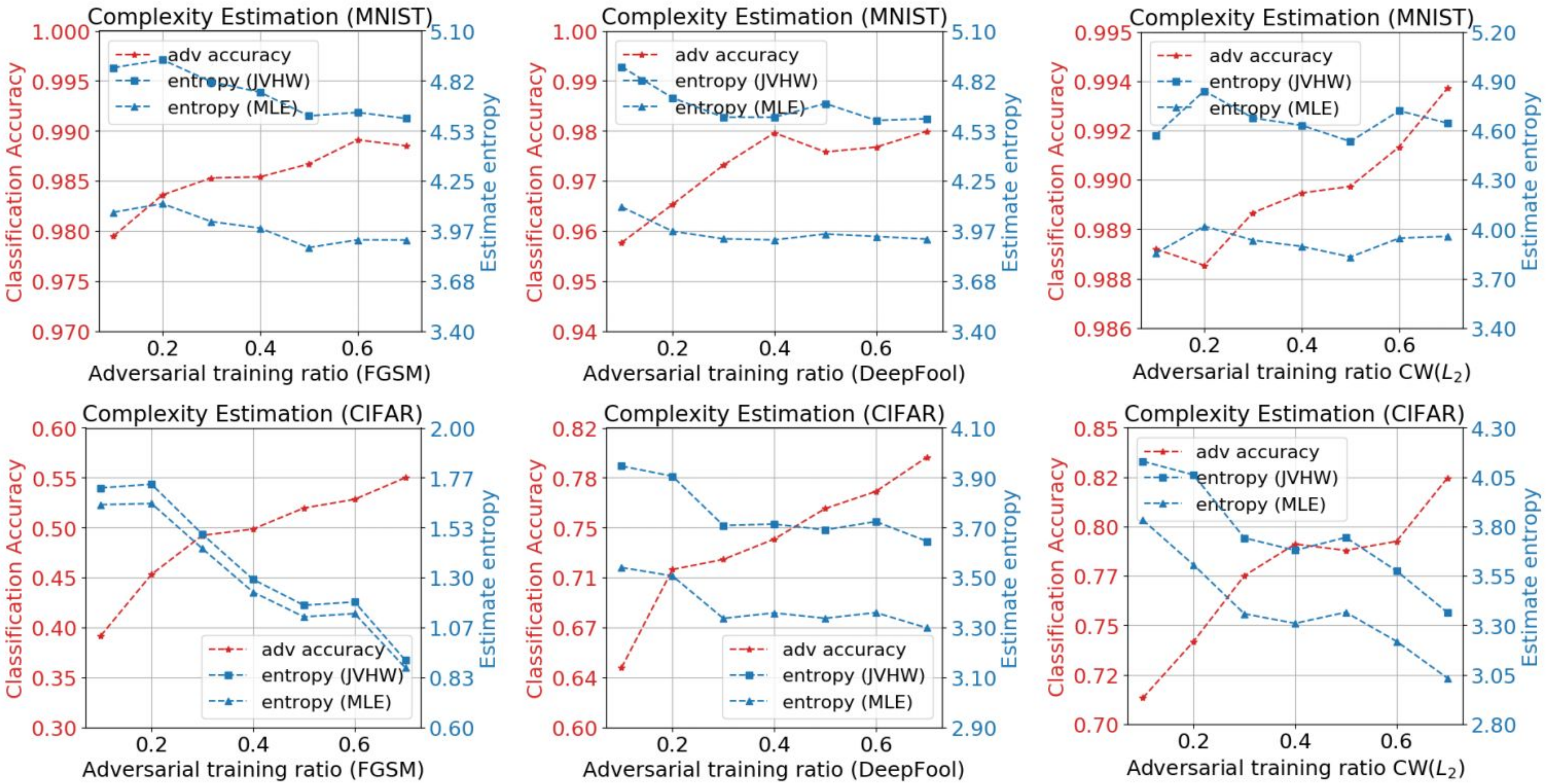
- Adv. examples are more redundant!
  - larger model capacity to memorize







AT for Reducing Feature Redundancy



(a) FGSM (b) DeepFool (c) CW(L_2)

Information Bottleneck

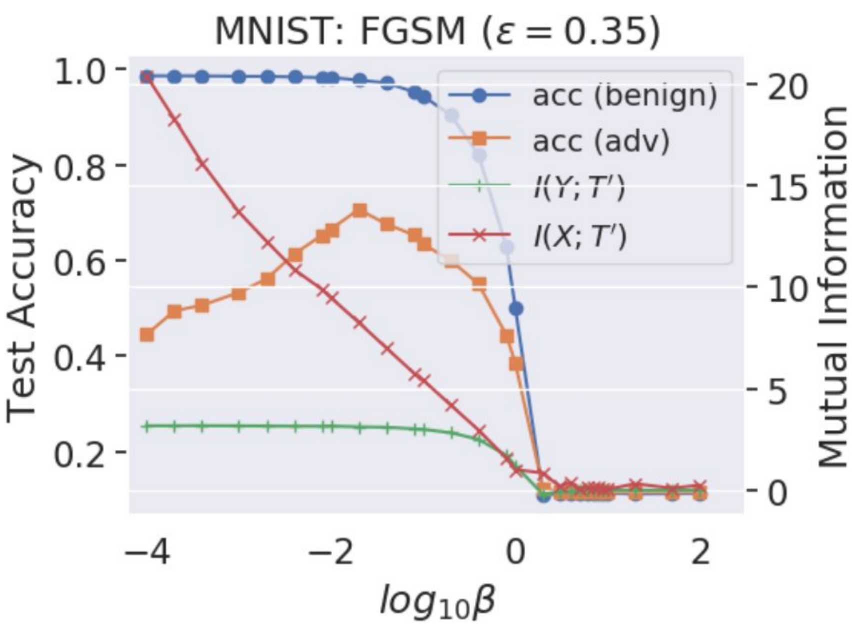
Theorem 1 (MSS is necessary for Adv.)

Suppose that Assumption 1 holds and there exist adversarial examples for the neural network $f(\cdot) = g(T(\cdot))$. Then, $T(X)$ is not a MSS.

- Information Bottleneck (IB): $$\max_{\theta} I(Y;T) - \beta I(X;T)$$

Minimize the $I(X;T)$

- The gap between benign training accuracy and adversarial testing accuracy decreases with $I(X;T)$
  - sufficiency
  - minimality



Alemi et al. "Deep Variational Information Bottleneck" ICLR 2017.

Adversarial Robustness Bound

Theorem 2 (Oblivious Vulnerability)

Suppose that $p(t|x)$ is a L-lipschitz function of x for any given t. Then,

$$|I(Y;T) - I(Y;T')| \leq |\mathcal{T}_b \cup \mathcal{T}_a|\psi(L\epsilon) + \max\{C_1\sqrt{|\mathcal{T}_b|}(I(X;T))^{\frac{1}{2}} + C_2|\mathcal{T}_b|^{\frac{3}{4}}(I(X;T))^{\frac{1}{4}}$$
$$C_3\sqrt{|\mathcal{T}_a|}(I(X';T'))^{\frac{1}{2}} + C_4|\mathcal{T}_a|^{\frac{3}{4}}(I(X';T'))^{\frac{1}{4}}\}$$

where C_1, C_2, C_3 and C_4 are some constants depending only on $p(x)$ and the attack algorithm is the bound on the magnitude of the perturbation exerted by the attack algorithm.

- $I(Y;T)$ represents the utility of T in predicting Y
  - $I(Y;T) = H(Y) - H(Y|T)$ => accuracy of benign data
  - $I(Y;T')$ => accuracy of adversarial data
- oblivious vulnerability is controlled by $I(X;T)$ and $I(X';T')$
  - IB explicitly minimizes $I(X;T)$
  - IB + Adversarial Training (better defense)
- Lipschitz constant L and perturbation magnitude ϵ
  - smoother encoder & smaller perturbation

Adversarial Robustness Bound

Theorem 2 (Oblivious Vulnerability)

$$|I(Y;T) - I(Y;T')| \le |\mathcal{T}_b \cup \mathcal{T}_a|\psi(L\epsilon) + \max\{C_1\sqrt{|\mathcal{T}_b|}(I(X;T))^{\frac{1}{2}} + C_2|\mathcal{T}_b|^{\frac{3}{4}}(I(X;T))^{\frac{1}{4}}$$
$$C_3\sqrt{|\mathcal{T}_a|}(I(X';T'))^{\frac{1}{2}} + C_4|\mathcal{T}_a|^{\frac{3}{4}}(I(X';T'))^{\frac{1}{4}}\}$$

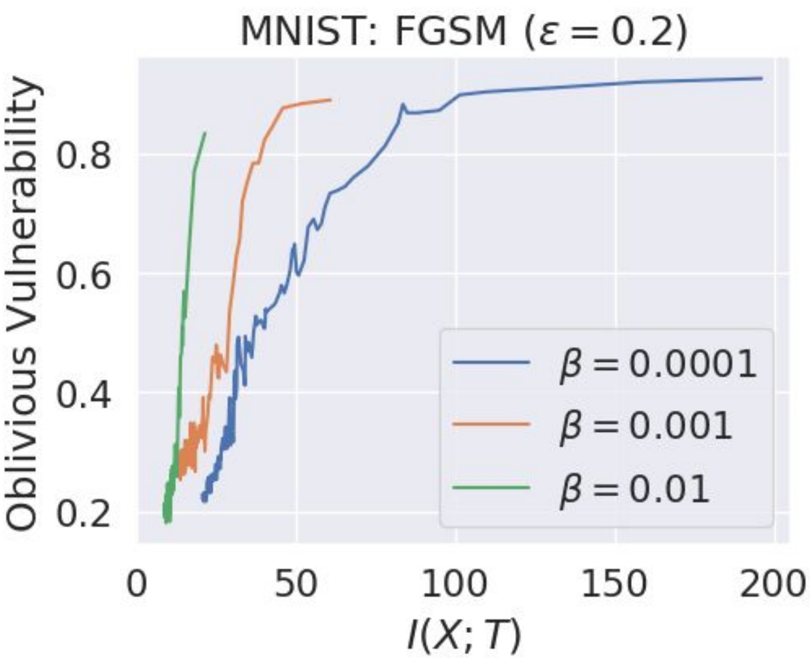


Table 1: Comparison of adversarial robustness and $I(X';T')$ between vanilla IB and IB combined with adversarial training. $\mathrm{Acc}_{\mathrm{adv}}$ denotes the test accuracy on adversarial examples.

Setting ($\epsilon = 0.3$)			FGSM	BIM	PGD
$I(X';T')$	$\beta = 1e^{-2}$	IB	11.41	21.24	17.94
		IB + adv.	**10.46**	**13.96**	**13.73**
	$\beta = 5e^{-2}$	IB	13.31	29.64	29.99
		IB + adv.	**13.14**	**19.95**	**19.68**
	$\beta = 1e^{-3}$	IB	30.91	96.59	75.78
		IB + adv.	**25.59**	**50.55**	**50.68**
$\mathrm{Acc}_{\mathrm{adv}}$	$\beta = 1e^{-2}$	IB	83.55	64.92	62.26
		IB + adv.	**92.12**	**77.04**	**77.93**
	$\beta = 5e^{-2}$	IB	80.91	63.09	63.20
		IB + adv.	**91.86**	**76.35**	**77.22**
	$\beta = 1e^{-3}$	IB	73.96	59.59	54.94
		IB + adv.	**90.24**	**76.46**	**76.45**

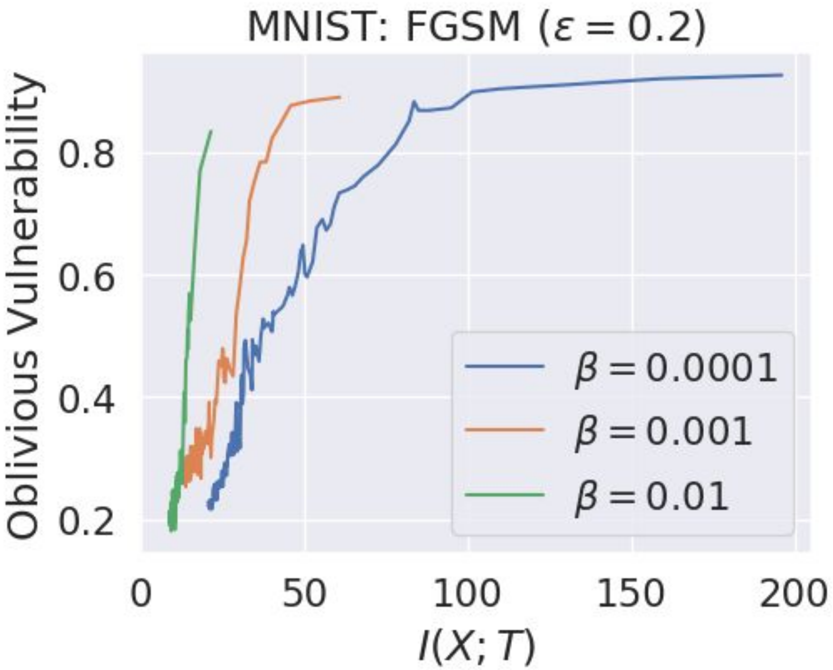
- Why IB works ?
- Why IB + AT works better ?

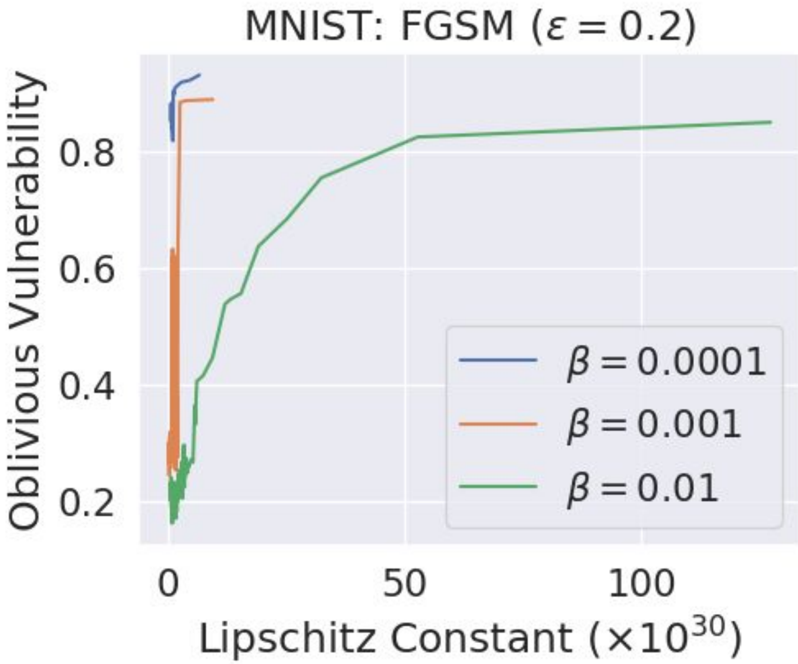
Adversarial Robustness Bound

Theorem 3 (Estimation of Lipschitz Constant)

Lipschitz constant of a stochastic encoder can be controlled by its mean and variance networks

$$|I(Y;T)-I(Y;T')| \le |\mathcal{T}_b \cup \mathcal{T}_a|\psi(L\epsilon) + \max\{C_1\sqrt{|\mathcal{T}_b|}(I(X;T))^{\frac{1}{2}} + C_2|\mathcal{T}_b|^{\frac{3}{4}}(I(X;T))^{\frac{1}{4}}$$
$$C_3\sqrt{|\mathcal{T}_a|}(I(X';T'))^{\frac{1}{2}} + C_4|\mathcal{T}_a|^{\frac{3}{4}}(I(X';T'))^{\frac{1}{4}}\}$$





- Why controlling the Lipschitz constant works?

Finlay et al. “Improved robustness to adversarial examples using Lipschitz regularization of the loss”. 2018.

IB + AT: better transferability & robustness

Transferability (FGSM $\epsilon = 0.3, \beta = 0.001$) **attack-sensitive**

- MNIST

Setting	Benign	FGSM			DeepFool	CW (L_2)	BIM	PGD
	-	$\epsilon = 0.1$	$\epsilon = 0.3$	$\epsilon = 0.5$	$m = 10^2$	$c = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.2$
IB	**98.64**	**85.88**	**62.94**	**40.63**	**34.53**	**53.30**	**58.41**	**51.79**
CE	98.63	63.39	1.38	1.04	1.79	18.33	2.18	2.12
IB + adv.	**98.53**	**92.13**	**90.43**	**55.27**	**43.08**	**49.69**	**58.97**	**56.65**
CE + adv.	97.94	78.95	89.73	50.33	2.54	16.81	21.82	21.36

- CIFAR-10

Setting	Benign	FGSM			DeepFool	CW (L_2)	BIM	PGD
	-	$\epsilon = 0.1$	$\epsilon = 0.3$	$\epsilon = 0.5$	$m = 10^2$	$c = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.2$
IB	**67.06**	**34.92**	**20.15**	**14.73**	**23.47**	**21.37**	**19.06**	**20.04**
CE	65.39	17.33	12.23	10.35	18.5	11.63	18.67	18.19
IB + adv.	**63.14**	**44.48**	**58.64**	**56.28**	**26.64**	**13.20**	**21.52**	**22.86**
CE + adv.	61.67	29.4	57.04	48.44	21.79	11.65	19.69	19.78

Summary (Adv. information theory)

- Feature redundancy (necessary condition)
- How to reduce feature redundancy
  - Information Bottleneck (IB) $\max_{\theta} I(Y;T) - \beta I(X;T)$
  - Adversarial training as an implicit regularizer
  - other possible dimension reduction techniques …
- IB + Adversarial Training
  - better adversarial robustness
  - stronger transferability (defense)
- Theoretical Analysis
  - Oblivious vulnerability
  - Transferability for stochastic networks

What should we do?

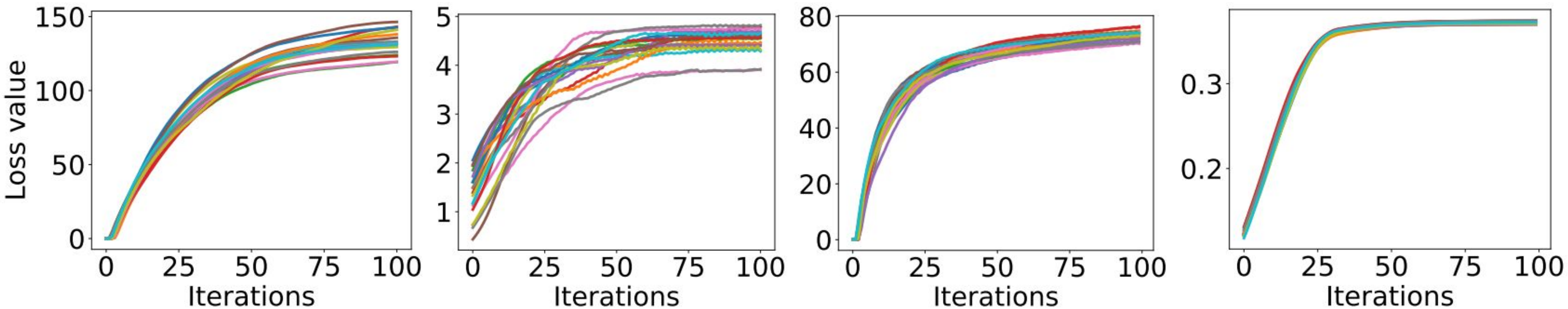
- **More Accurate Sensors + More Robust Algorithms!**
  (Safety) **(Security)**

- **What I did in last 1.5 years:**
- Self-Driving:
  - Policy learning: 2D + LiDAR (cheaper but still very expensive)
- Tackling with Noisy Data:
  - Reinforcement Learning with Perturbed Reward
- **Attack & Defend our ML/DL Systems:**
  - Arms Race in Adversarial Machine Learning (AML)
  - More reliable, interpretable DL training scheme (Information Bottleneck, IB)
  - **Min-Max Optimization in AML (across domains)**

Adversarial Training (min-max)

- Natural Training: $$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}[L(x,y,\theta)]$$
- Adversarial Training (AT):

$$\min_{\theta} \rho(\theta), \quad \text{where} \quad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{S}} L(\theta, x+\delta, y)\right]$$



(a) MNIST Natural training | (b) MNIST Adversarial training | (c) CIFAR10 Natural training | (d) CIFAR10 Adversarial training

- Issue (AT):
  - low transferability (**multiple norms**, attack-sensitive)
  - **low efficiency in ensemble adversarial training**

Revisiting the Power of Min-Max Opt.

▪ General idea: Robust learning over multiple domains

- **Formulation:** Consider K loss functions $\{F_i(v)\}$ (each of which is defined on a learning domain), the problem of robust learning over K domains can be formulated:

$$\underset{\mathbf{v}\in\mathcal{V}}{\text{minimize}}\ \underset{\mathbf{w}\in\mathcal{P}}{\text{maximize}}\quad \sum_{i=1}^{K} w_i F_i(\mathbf{v})$$

  where $\mathbf{v}$ and $\mathbf{w}$ are optimization variables, $\mathcal{V}$ is a constraint set, and $\mathcal{P}$ denotes the probability simplex $\mathcal{P} = \{\mathbf{w} \,|\, \mathbf{1}^T\mathbf{w} = 1,\ w_i \in [0,1],\ \forall i\}$.

- **Worst-case:** equivalent to $\underset{\mathbf{v}\in\mathcal{V}}{\text{minimize}}\ \underset{i\in[K]}{\text{maximize}}\quad F_i(\mathbf{v})$

  where $[K]$ denotes the integer set $\{1, 2, \ldots, K\}$.
  - one-hot coding reduces the generalizability to other domains
  - induces instability in training

- **Regularized problem formulation:**

$$\underset{\mathbf{v}\in\mathcal{V}}{\text{minimize}}\ \underset{\mathbf{w}\in\mathcal{P}}{\text{maximize}}\quad \sum_{i=1}^{K} w_i F_i(\mathbf{v}) - \frac{\gamma}{2}\|\mathbf{w} - \mathbf{1}/K\|_2^2$$

domain weights

Robust Adversarial Attacks

- **Ensemble attack over multiple models**
  - Consider K ML/DL models $\{\mathcal{M}_i\}_{i=1}^K$, the goal is to find robust adversarial examples that can fool all K models simultaneously

$$\underset{\boldsymbol{\delta}\in\mathcal{X}}{\text{minimize}}\ \underset{\mathbf{w}\in\mathcal{P}}{\text{maximize}} \quad \sum_{i=1}^K w_i f(\boldsymbol{\delta}; \mathbf{x}_0, y_0, \mathcal{M}_i) - \frac{\gamma}{2}\|\mathbf{w} - \mathbf{1}/K\|_2^2$$

  - $\mathbf{w}$ encodes the difficulty level of attacking each model

- **Why ensemble?**
  - Attacker: more transferable black-box adv. examples
  - Defender: AT + Ensemble (more powerful defense)
  - Existing Approach (equal-weight, 5^{th} in CAAD 2018)

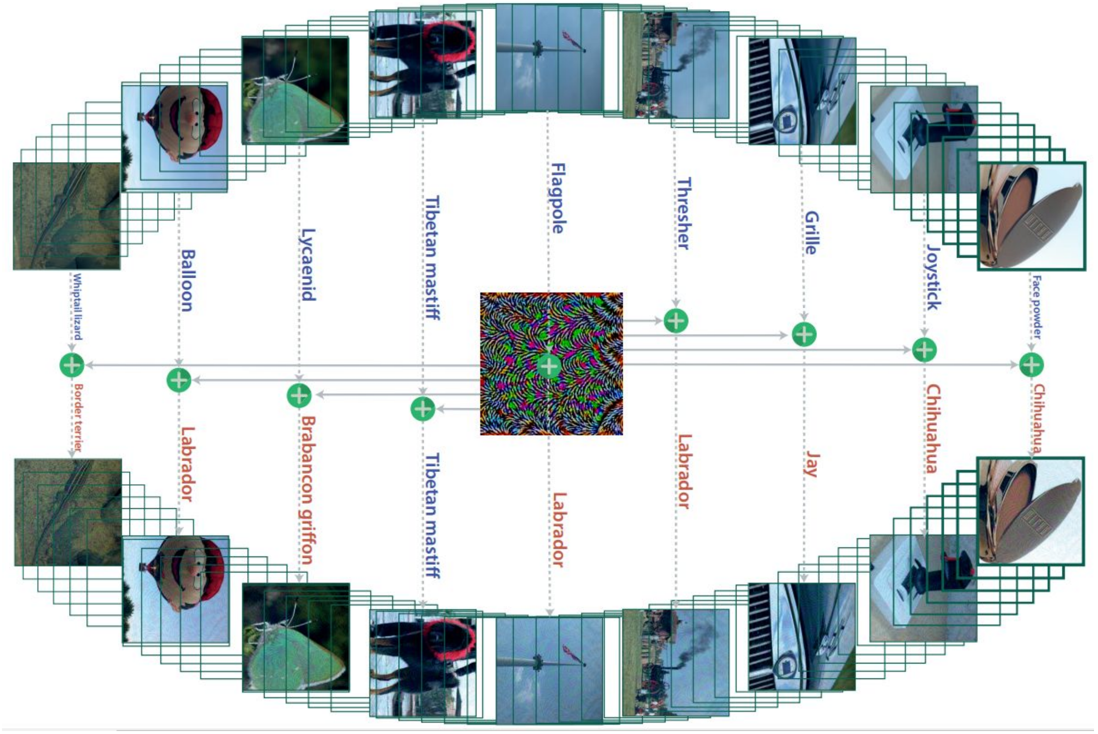
- **Our solution:**
  - Focus on those models which are difficult to attacks!
  - Guarantee the worst-case performance!

Robust Adversarial Attacks

- **Universal perturbation over multiple examples**
  - Consider K natural examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^K$ and a single model $\mathcal{M}$, the goal is to find the universal perturbation δ so that all the corrupted K examples can fool $\mathcal{M}$

$$\underset{\boldsymbol{\delta}\in\mathcal{X}}{\text{minimize}}\ \underset{\mathbf{w}\in\mathcal{P}}{\text{maximize}} \quad \sum_{i=1}^K w_i f(\boldsymbol{\delta}; \mathbf{x}_i, y_i, \mathcal{M}) - \frac{\gamma}{2}\|\mathbf{w} - \mathbf{1}/K\|_2^2$$

  - $\mathbf{w}$ encodes the difficulty level of attacking each example



Robust Adversarial Attacks

- **Robust attack over data transformations**
  - Consider K categories of data transformation $\{p_i\}$ e.g., rotation, lightening, and translation. The goal to find the adversarial attack that is robust to data transformations

$$\underset{\boldsymbol{\delta} \in \mathcal{X}}{\text{minimize}} \ \underset{\mathbf{w} \in \mathcal{P}}{\text{maximize}} \quad \sum_{i=1}^{K} w_i \mathbb{E}_{t \sim p_i} [f(t(\mathbf{x}_0 + \boldsymbol{\delta}); y_0, \mathcal{M})] - \frac{\gamma}{2} \|\mathbf{w} - \mathbf{1}/K\|_2^2$$

  - $\mathbf{w}$ encodes the difficulty level of attacking each type of transformed example
  - $\mathbb{E}_{t \sim p_i} [f(t(\mathbf{x}_0 + \boldsymbol{\delta}); y_0, \mathcal{M})]$ denotes the attack loss under the distribution of data transformation p_i

classified as turtle   classified as rifle   classified as other

Alternating one-step PGD (APGD)

Algorithm 1 APGD to solve problem (4)

1: Input: given $\mathbf{w}^{(0)}$ and $\boldsymbol{\delta}^{(0)}$.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: *outer min.*: fixing $\mathbf{w} = \mathbf{w}^{(t-1)}$, call PGD (13) to update $\boldsymbol{\delta}^{(t)}$
4: *inner max.*: fixing $\boldsymbol{\delta} = \boldsymbol{\delta}^{(t)}$, update $\mathbf{w}^{(t)}$ via (14)
5: **end for**

- efficient as PGD
- worst-case guarantee
- higher attack success rate (ASR)

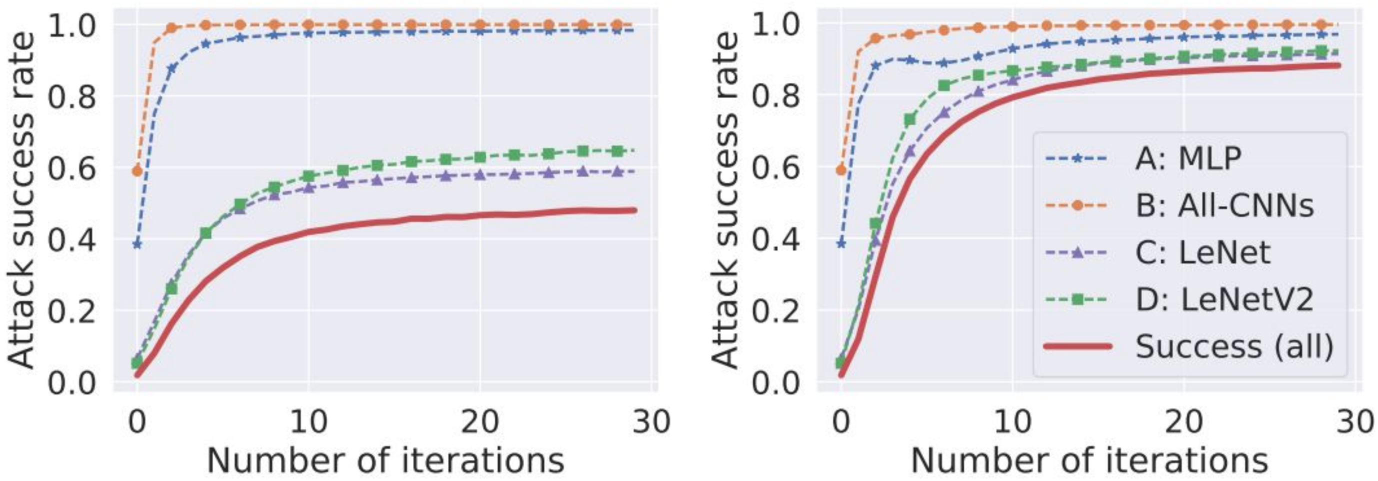
Outer minimization Considering $\mathbf{w} = \mathbf{w}^{(t-1)}$ and $F(\boldsymbol{\delta}) := \sum_{i=1}^{K} w_i^{(t-1)} F_i(\boldsymbol{\delta})$ in (4), we perform one-step PGD to update $\boldsymbol{\delta}$ at iteration t,

$$\boldsymbol{\delta}^{(t)} = \mathrm{proj}_{\mathcal{X}} \left(\boldsymbol{\delta}^{(t-1)} - \alpha \nabla_{\boldsymbol{\delta}} F(\boldsymbol{\delta}^{(t-1)}) \right), \tag{13}$$

Inner maximization By fixing $\boldsymbol{\delta} = \boldsymbol{\delta}^{(t)}$ and letting $\psi(\mathbf{w}) := \sum_{i=1}^{K} w_i F_i(\boldsymbol{\delta}^{(t)}) - \frac{\gamma}{2}\|\mathbf{w} - \mathbf{1}/K\|_2^2$ in problem (4), we then perform one-step PGD (w.r.t. $-\psi$) to update $\mathbf{w}$,

$$\mathbf{w}^{(t)} = \mathrm{proj}_{\mathcal{P}} \Big(\underbrace{\mathbf{w}^{(t-1)} + \beta \nabla_{\mathbf{w}} \psi(\mathbf{w}^{(t-1)})}_{\mathbf{b}} \Big) = (\mathbf{b} - \mu \mathbf{1})_+ , \tag{14}$$

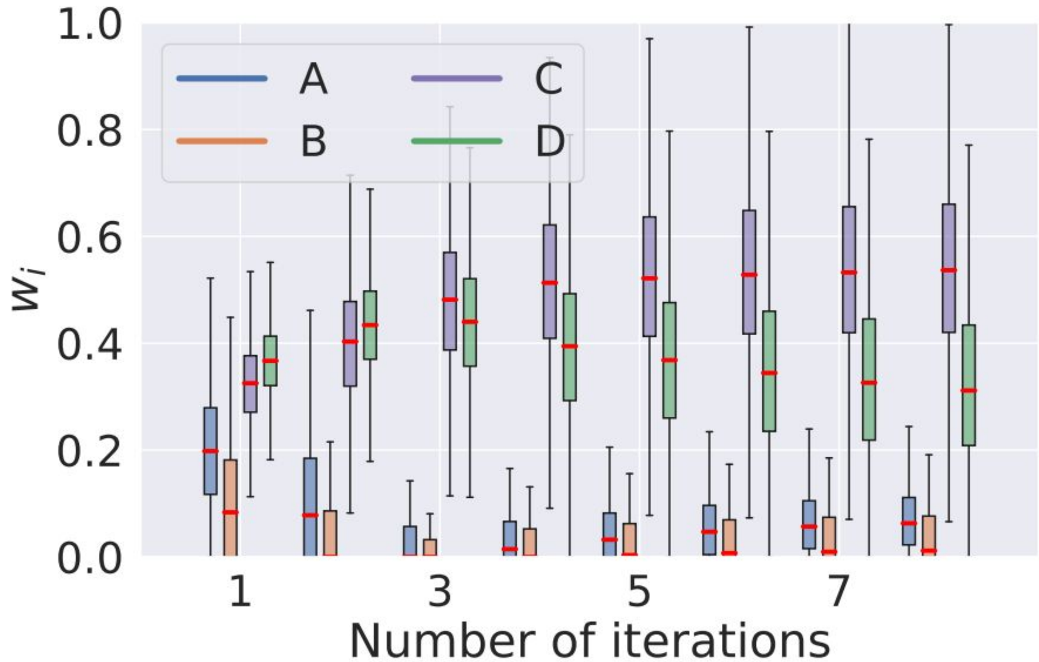
Results – Attacking Model Ensembles



(a) average case **(b)** min max

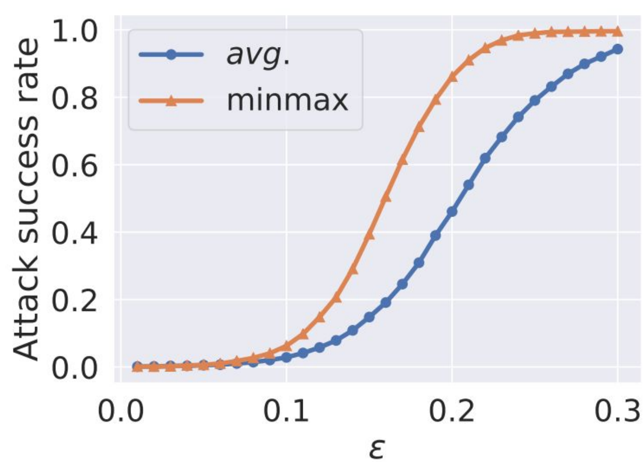
Box constraint	Opt.	Acc_A	Acc_B	Acc_C	Acc_D	ASR_{all}	Lift (↑)
ℓ_0 ($\epsilon = 30$)	*avg.*	7.03	1.51	11.27	2.48	84.03	-
	min max	3.65	2.36	4.99	3.11	**91.97**	**9.45%**
ℓ_1 ($\epsilon = 20$)	*avg.*	20.79	0.15	21.48	6.70	69.31	-
	min max	6.12	2.53	8.43	5.11	**89.16**	**28.64%**
ℓ_2 ($\epsilon = 3.0$)	*avg.*	6.88	0.03	26.28	14.50	69.12	-
	min max	1.51	0.89	3.50	2.06	**95.31**	**37.89%**
ℓ_∞ ($\epsilon = 0.2$)	*avg.*	1.05	0.07	41.10	35.03	48.17	-
	min max	2.47	0.37	7.39	5.81	**90.16**	**87.17%**

Results – Attacking Model Ensembles

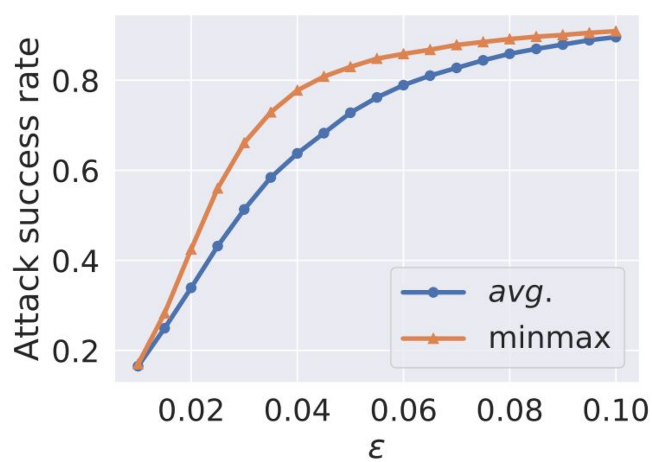


$$w_c > w_d > w_a > w_b$$

$$\mathrm{Acc}_C > \mathrm{Acc}_D > \mathrm{Acc}_A > \mathrm{Acc}_B$$



(a) MNIST {A, B, C, D}



(b) CIFAR-10 {A, B, C, D}

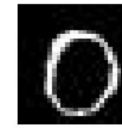
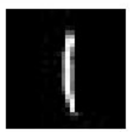
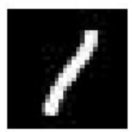
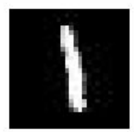
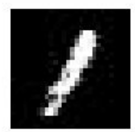
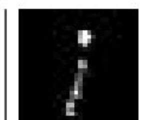
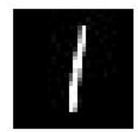
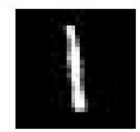
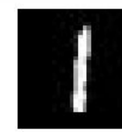
Results – Devising Universal Perturbations

Table 2: Comparison of average and minmax optimization on universal perturbation over multiple input examples. The adversarial examples are generated by ℓ_∞-APGD with $\alpha = 6, \beta = 50$ and $\gamma = 4$.

Setting			$K=2$			$K=4$			$K=5$			$K=10$		
Dataset	Model	Opt.	ASR_{avg}	ASR_{gp}	Lift (↑)	ASR_{avg}	ASR_{gp}	Lift (↑)	ASR_{avg}	ASR_{gp}	Lift (↑)	ASR_{avg}	ASR_{gp}	Lift (↑)
MNIST	MLP	*avg.*	97.19	94.48	-	85.13	56.64	-	79.11	38.05	-	60.53	3.50	-
		min max	98.15	**96.96**	**2.62%**	83.76	**72.32**	**27.68%**	72.28	**53.70**	**41.13%**	30.10	**6.70**	**91.43%**
	All-CNNs	*avg.*	97.76	95.52	-	85.19	51.92	-	80.02	31.25	-	65.79	2.10	-
		min max	99.69	**99.38**	**4.04%**	90.11	**75.64**	**45.69%**	80.21	**53.50**	**71.20%**	43.54	**4.30**	**104.8%**
	LeNet	*avg.*	94.78	89.96	-	62.12	28.72	-	51.84	19.15	-	30.29	4.30	-
		min max	96.60	**94.58**	**5.14%**	55.50	**36.72**	**27.86%**	42.79	**25.80**	**34.73%**	22.48	**7.20**	**67.44%**
	LeNetV2	*avg.*	94.72	90.04	-	61.59	26.60	-	50.42	17.05	-	26.49	4.80	-
		min max	97.33	**95.68**	**6.26%**	55.38	**35.52**	**33.53%**	40.22	**21.05**	**23.46%**	19.73	**7.10**	**47.92%**
CIFAR-10	All-CNNs	*avg.*	91.09	83.08	-	85.66	54.72	-	82.76	40.20	-	71.22	4.50	-
		min max	92.22	**85.98**	**3.49%**	87.63	**65.80**	**20.25%**	85.02	**55.74**	**38.66%**	65.64	**11.80**	**162.2%**
	LeNetV2	*avg.*	93.26	86.90	-	90.04	66.12	-	88.28	55.00	-	72.02	8.90	-
		min max	93.34	**87.08**	**0.21%**	91.91	**71.64**	**8.35%**	91.21	**63.55**	**15.55%**	82.85	**25.10**	**182.0%**
	VGG16	*avg.*	90.76	82.56	-	89.36	63.92	-	88.74	55.20	-	85.86	22.40	-
		min max	92.40	**85.92**	**4.07%**	90.04	**70.40**	**10.14%**	88.97	**63.30**	**14.67%**	79.07	**30.80**	**37.50%**
	GoogLeNet	*avg.*	85.02	72.48	-	75.20	32.68	-	71.82	19.60	-	59.01	0.40	-
		min max	87.08	**77.82**	**7.37%**	77.05	**46.20**	**41.37%**	71.20	**33.70**	**71.94%**	45.46	**2.40**	**600.0%**

Results – Devising Universal Perturbations

Table A8: Interpretability of domain weight w for universal perturbation to multiple inputs on MNIST (*Digit 0 to 4*). Domain weight w for different images under ℓ_p-norm ($p = 0, 1, 2, \infty$) and two metrics measuring the difficulty of attacking single image are recorded, where dist. (ℓ_2) denotes the the minimum distortion of successfully attacking images using C&W (ℓ_2) attack; $\epsilon_{\min}$ (ℓ_∞) denotes the minimum perturbation magnitude for ℓ_∞-PGD attack.

Image											
Weight	ℓ_0	0.	0.	0.	0.	1.000	0.248	0.655	0.097	0.	0.
	ℓ_1	0.	0.	0.	0.	1.000	0.07	0.922	0.	0.	0.
	ℓ_2	0.	0.	0.	0.	1.000	0.441	0.248	0.156	0.155	0.
	ℓ_∞	0.	0.	0.	0.	1.000	0.479	0.208	0.145	0.168	0.
Metric	dist.(C&W ℓ_2)	1.839	1.954	1.347	1.698	3.041	1.545	1.982	2.178	2.349	1.050
	$\epsilon_{\min}$ (ℓ_∞)	0.113	0.167	0.073	0.121	0.199	0.167	0.157	0.113	0.114	0.093
Image											
Weight	ℓ_0	0.	0.	0.613	0.180	0.206	0.	0.	0.223	0.440	0.337
	ℓ_1	0.	0.	0.298	0.376	0.327	0.	0.	0.397	0.433	0.169
	ℓ_2	0.	0.	0.387	0.367	0.246	0.	0.242	0.310	0.195	0.253
	ℓ_∞	0.087	0.142	0.277	0.247	0.246	0.	0.342	0.001	0.144	0.514
Metric	dist.(C&W ℓ_2)	1.090	1.182	1.327	1.458	0.943	0.113	1.113	1.357	1.474	1.197
	$\epsilon_{\min}$ (ℓ_∞)	0.075	0.068	0.091	0.105	0.096	0.015	0.090	0.076	0.095	0.106

Results – Robust Adv. over Data Transformations

- **Deterministic**

avg. = EOT (SOTA)

Model	Opt.	Acc_{ori}	Acc_{flh}	Acc_{flv}	Acc_{bri}	Acc_{gam}	Acc_{crop}	ASR_{avg}	ASR_{gp}	Lift (↑)
A	*avg.*	10.80	21.93	14.75	11.52	10.66	20.03	85.05	55.88	-
	min max	12.14	18.05	13.61	13.52	11.99	16.78	85.65	**60.03**	**7.43%**
B	*avg.*	5.49	11.56	9.51	5.43	5.75	15.89	91.06	72.21	-
	min max	6.22	8.61	9.74	6.35	6.42	11.99	91.78	**77.43**	**7.23%**
C	*avg.*	7.66	21.88	15.50	8.15	7.87	15.36	87.26	56.51	-
	min max	8.51	14.75	13.88	9.16	8.58	13.35	88.63	**63.58**	**12.51%**
D	*avg.*	8.00	20.47	13.46	7.73	8.52	15.90	87.65	61.13	-
	min max	9.19	13.18	12.72	8.79	9.18	13.11	88.97	**67.49**	**10.40%**

- **Stochastic**

Model	Opt.	Acc_{ori}	Acc_{flh}	Acc_{flv}	Acc_{bri}	Acc_{crop}	ASR_{avg}	ASR_{gp}	Lift (↑)
A	*avg.*	11.55	21.60	13.64	12.30	22.37	83.71	55.97	-
	min max	13.06	18.90	13.43	13.90	20.27	84.09	**59.17**	**5.72%**
B	*avg.*	6.74	11.55	10.33	6.59	18.21	89.32	69.52	-
	min max	8.19	11.13	10.31	8.31	16.29	89.15	**71.18**	**2.39%**
C	*avg.*	8.23	17.47	13.93	8.54	18.83	86.60	58.85	-
	min max	9.68	13.45	13.41	9.95	18.23	87.06	**61.63**	**4.72%**
D	*avg.*	8.67	19.75	11.60	8.46	19.35	86.43	60.96	-
	min max	10.43	16.41	12.14	10.15	17.64	86.65	**63.64**	**4.40%**

Generalized Adversarial Training (GAT)

- Vanilla Adversarial Training:

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\ \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\ \underset{\|\boldsymbol{\delta}\|_\infty\le\epsilon}{\text{maximize}}\ f_{\text{tr}}(\boldsymbol{\theta},\boldsymbol{\delta};\mathbf{x},y)$$

  - transferability between attacks under multiple norms is low!

- Generalized Adversarial Training:

sum:

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\ \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\ \underset{\{\boldsymbol{\delta}_i\in\mathcal{X}_i\}}{\text{maximize}}\ \frac{1}{K}\sum_{i=1}^{K} f_{\text{tr}}(\boldsymbol{\theta},\boldsymbol{\delta}_i;\mathbf{x},y)$$

max:

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\ \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\ \underset{i\in[K]}{\text{maximize}}\ F_i(\boldsymbol{\theta})$$

$$\Updownarrow$$

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\ \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\ \underset{\mathbf{w}\in\mathcal{P},\{\boldsymbol{\delta}_i\in\mathcal{X}_i\}}{\text{maximize}}\ \sum_{i=1}^{K} w_i f(\boldsymbol{\theta},\boldsymbol{\delta}_i;\mathbf{x},y)$$

  - better overall robustness against multiple attacks
  - faster convergence (min-max)

Alternating multi-step PGD (AMPGD)

Algorithm 2 AMPGD to solve problem (15)

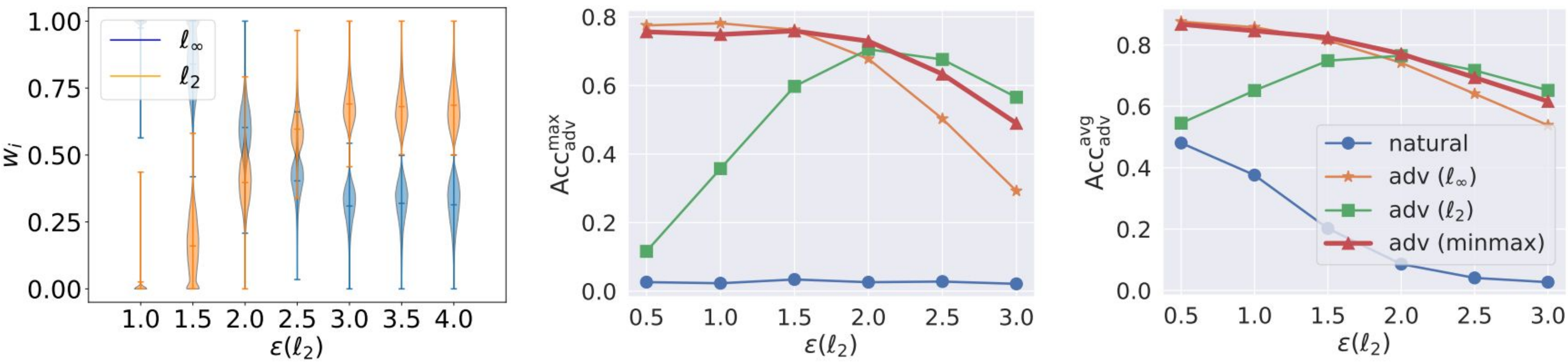
1: Input: given $\boldsymbol{\theta}^{(0)}$, $\mathbf{w}^{(0)}$, $\boldsymbol{\delta}^{(0)}$ and $K > 0$.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: given $\mathbf{w}^{(t-1)}$ and $\boldsymbol{\delta}^{(t-1)}$, perform SGD to update $\boldsymbol{\theta}^{(t)}$ [4]
4: given $\boldsymbol{\theta}^{(t)}$, perform R-step PGD to update $\mathbf{w}^{(t)}$ and $\boldsymbol{\delta}^{(t)}$
5: **end for**

- better overall robustness
- good interpretability

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\ \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\ \underset{\mathbf{w}\in\mathcal{P},\{\boldsymbol{\delta}_i\in\mathcal{X}_i\}}{\text{maximize}}\ \psi(\boldsymbol{\theta},\mathbf{w},\{\boldsymbol{\delta}_i\}) := \sum_{i=1}^{K} w_i f(\boldsymbol{\theta},\boldsymbol{\delta}_i;\mathbf{x},y) - \frac{\gamma}{2}\|\mathbf{w}-\mathbf{1}/K\|_2^2$$

$$\mathbf{w}_r^{(t)} = \mathrm{proj}_{\mathcal{P}}\left(\mathbf{w}_{r-1}^{(t)} + \beta\nabla_{\mathbf{w}}\psi(\boldsymbol{\theta}^{(t)},\mathbf{w}_{r-1}^{(t)},\{\boldsymbol{\delta}_{i,r-1}^{(t)}\})\right), \forall r\in[R],$$
$$\boldsymbol{\delta}_{i,r}^{(t)} = \mathrm{proj}_{\mathcal{X}_i}\left(\boldsymbol{\delta}_{i,r-1}^{(t)} + \beta\nabla_{\boldsymbol{\delta}}\psi(\boldsymbol{\theta}^{(t)},\mathbf{w}_{r-1}^{(t)},\{\boldsymbol{\delta}_{i,r-1}^{(t)}\})\right), \forall r\in[R], \forall i\in[K]$$

Results – Generalized AT

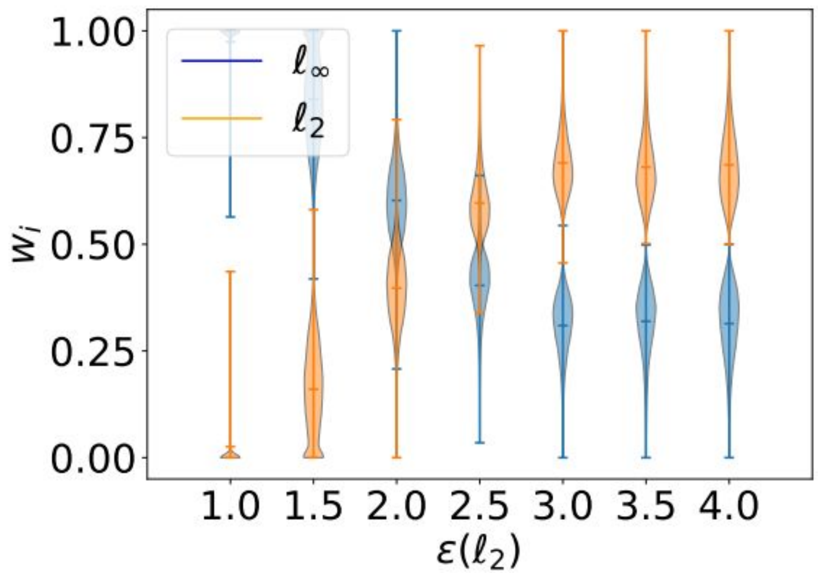


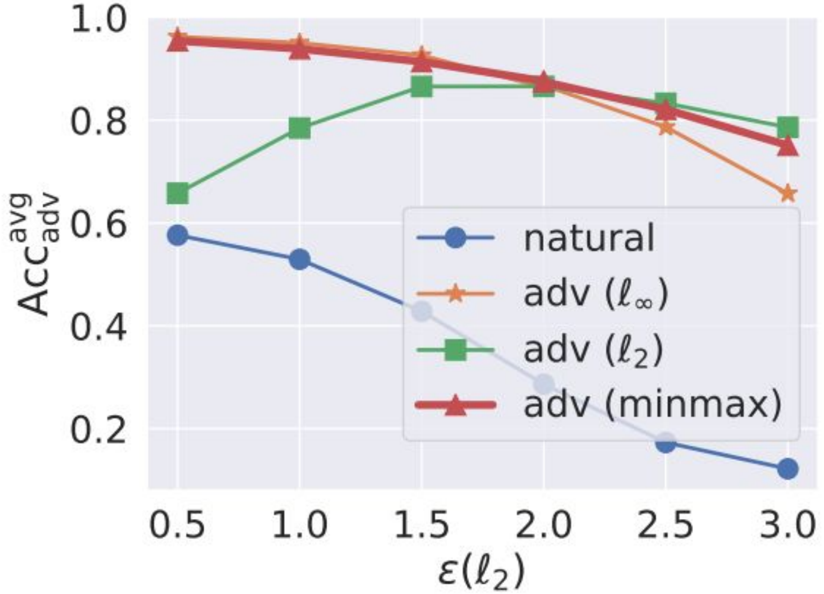
(a) weight $\{w_i\}$ **(b)** $\mathrm{Acc}^{\max}_{\mathrm{adv}}$ **(c)** $\mathrm{Acc}^{\mathrm{avg}}_{\mathrm{adv}}$

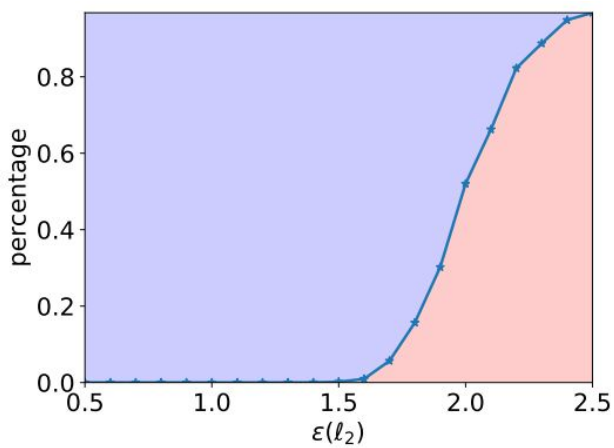
Table 4: Adversarial training of MNIST models on single attacks (ℓ_∞ and ℓ_2) and multiple attacks ($avg.$ and $\min\max$). The perturbation magnitude ϵ for ℓ_∞ and ℓ_2 attacks are 0.2 and 2.0, respectively. Top 2 test accuracy on each metric are highlighted. Complete table for LeNet and varied ϵ is given in Table A7 (Appendix D.2).

Model	Opt.	Acc.	Acc-ℓ_∞	Acc-ℓ_2	$\mathrm{Acc}^{\max}_{\mathrm{adv}}$	$\mathrm{Acc}^{\mathrm{avg}}_{\mathrm{adv}}$
	natural	98.30	2.70	13.86	0.85	8.28
MLP	ℓ_∞	98.08	**77.70**	69.17	66.34	73.43
	ℓ_2	98.72	70.03	**81.74**	69.14	75.88
	$avg.$	98.62	75.09	79.00	**72.23**	**77.05**
	$\min\max$	98.59	**75.96**	**79.15**	**73.43**	**77.55**

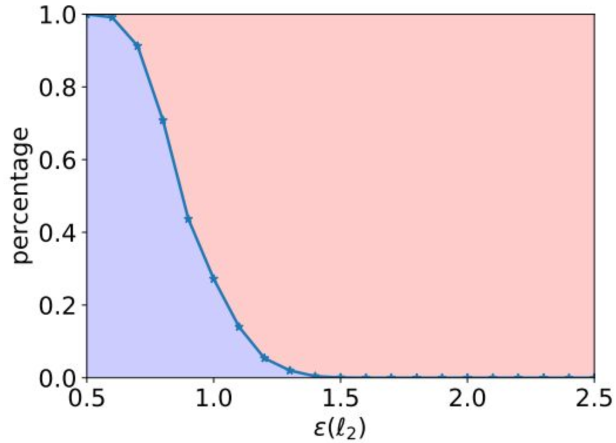
Results – Generalized AT



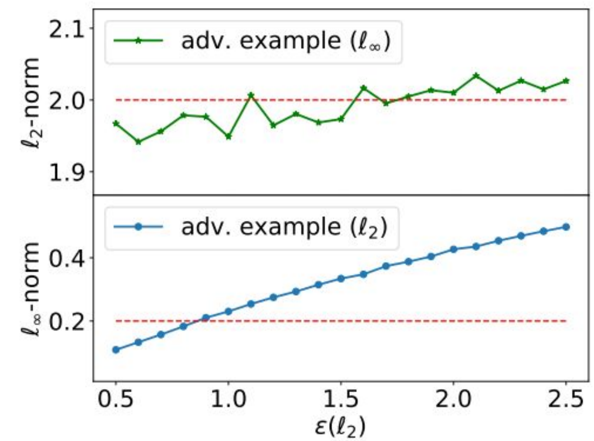




(a) $\ell_\infty \in \ell_2$



(b) $\ell_2 \in \ell_\infty$



(c) average ℓ_p-norm

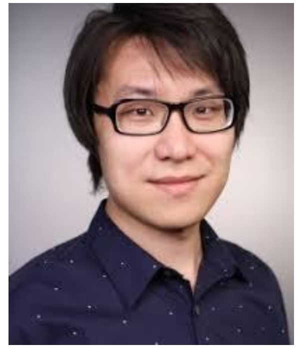
Summary (Min-Max Opt. in Adv.)

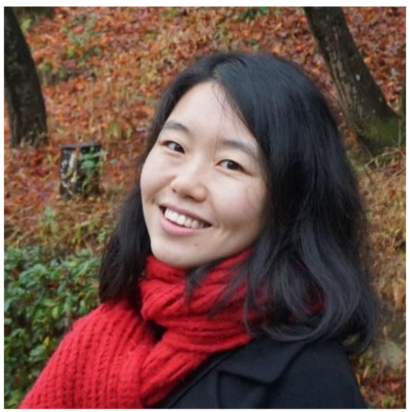
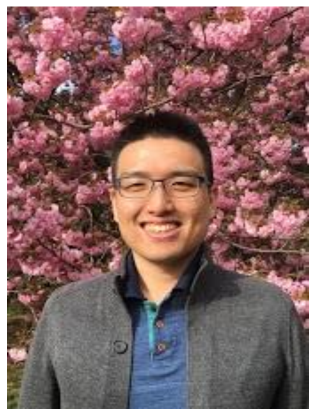
- A general min-max framework applicable to both adversarial attack and defense settings
- Reformulating many problem setups in our framework
  - Attacking model ensembles
  - Devising universal perturbation against multiple input images
  - Generating robust adv. examples over data transformations
  - Generalized AT under mixed type of attacks
- Significant improvements on four attack and defense tasks (more efficient)
- Providing a holistic tool for self-risk assessment by learning domain weights

Relevant Papers

- LiDAR-Video Driving Dataset: Learning Driving Policies Effectively. Yiping Chen*, Jingkang Wang*, Jonathan Li, Cewu Lu, Zhipeng Luo, Han Xue, Cheng Wang. (CVPR 2018)
- Reinforcement Learning with Perturbed Rewards. Jingkang Wang, Yang Liu, Bo Li. (AAAI 2020)
- On the Impact of Perceptual Compression on Deep Learning. Gerald Friedland, Ruoxi Jia, Jingkang Wang, Bo Li and Nathan Mundhenk. (MIPR 2020)
- One Bit Matters: Understanding Adversarial Examples as the Abuse of Redundancy. Jingkang Wang, Ruoxi Jia, Gerald Friedland, Bo Li, Costas Spanos.
- An Information-Theoretic Perspective on Adversarial Vulnerability. Ruoxi Jia, Jingkang Wang, Bo Li, Dawn Song.
- Towards A Unified Min-Max Framework for Adversarial Exploration and Robustness. Jingkang Wang*, Tianyun Zhang*, Sijia Liu, Pin-Yu Chen, Jiacen Xu, Makan Fardad, Bo Li.

Collaborators & Acknowledgements





Q & A