Adversarial Robustness via Runtime Masking and Cleansing



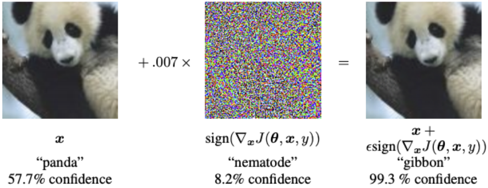
Yi-Hsuan Wu Chia-Hung Yuan Shan-Hung Wu

Department of Computer Science,
National Tsing Hua University, Taiwan

International Conference on Machine Learning, 2020

Why many adversarial defenses are broken?

- Deep neural networks are shown to be vulnerable to adversarial attacks, which motivates robust learning techniques

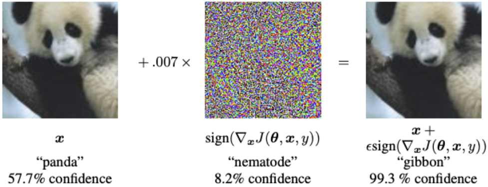


https://www.tensorflow.org/tutorials/generative/images/adversarial_example.png

[1] Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. ICML' 2018

Why many adversarial defenses are broken?

- Deep neural networks are shown to be vulnerable to adversarial attacks, which motivates robust learning techniques



https://www.tensorflow.org/tutorials/generative/images/adversarial_example.png

- A plethora of defenses have been proposed, however, ***many of these have been shown to fail***[1]

[1] Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. ICML' 2018

Why many adversarial defenses are broken?

- Recent study[2] shows the sample complexity of robust learning can be significantly larger than standard training

[2] Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Madry, A. Adversarially robust generalization requires more data. NeurIPS, 2018

Why many adversarial defenses are broken?

- Recent study[2] shows the sample complexity of robust learning can be significantly larger than standard training
- A theoretically grounded way to increase the adversarial robustness is to ***acquire more data***

[2] Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Madry, A. Adversarially robust generalization requires more data. NeurIPS, 2018

Why many adversarial defenses are broken?

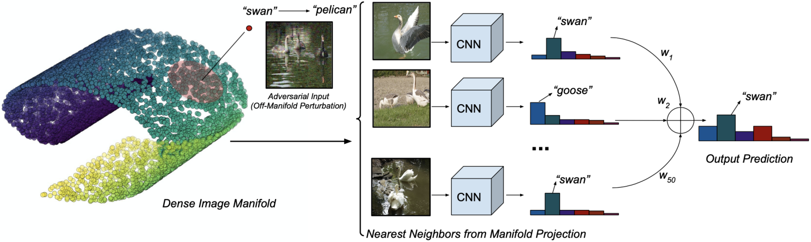
- Recent study[2] shows the sample complexity of robust learning can be significantly larger than standard training
- A theoretically grounded way to increase the adversarial robustness is to ***acquire more data***
- This partially explains why the adversarial training, a data augmentation technique, is empirically strong

[2] Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Madry, A. Adversarially robust generalization requires more data. NeurIPS, 2018

Outline

WebNN[3]

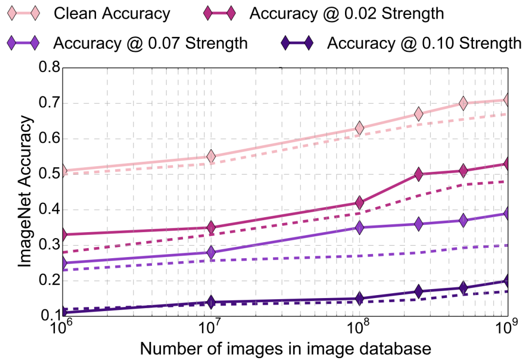
- Use a ***web-scale image database*** as a manifold and project a test image onto the manifold
- Make more robust prediction by taking only the projected image as inputs



[3]Dubey, A., Maaten, L. v. d., Yalniz, Z., Li, Y., and Mahajan, D. Defense against adversarial images using web-scale nearest-neighbor search. CVPR, 2019

Drawback: 50 Billion Images May be Too Large

- Web-scale database may not be available in other domains
- Performance drops when using smaller datasets



Outline

Goal

- Most existing defenses try to get more data at ***training time***

Goal

- Most existing defenses try to get more data at ***training time***
- We propose a **runtime defense**
  1. Adapts network weights θ for a test point $\hat{\boldsymbol{x}}$
  2. Makes inferecne $\hat{\boldsymbol{y}} = f(\hat{\boldsymbol{x}};\theta)$

Goal

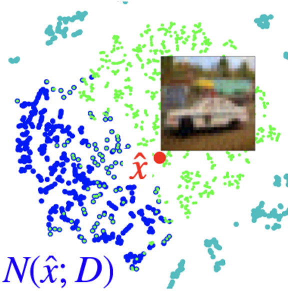
- Most existing defenses try to get more data at ***training time***
- We propose a **runtime defense**
  1. Adapts network weights θ for a test point $\hat{\boldsymbol{x}}$
  2. Makes inferecne $\hat{\boldsymbol{y}} = f(\hat{\boldsymbol{x}}; \theta)$
- Merits:
  - Uses ***potentially large test data*** to improve adversarial robustness
  - Is compatible with existing train-time defenses

Challenge: Test Data are Unlabeled

- How to adapt network weights θ for unlabeled $\hat{\boldsymbol{x}}$?
    - Online adversarial training is not applicable

Challenge: Test Data are Unlabeled

- How to adapt network weights θ for unlabeled $\hat{\boldsymbol{x}}$?
  - Online adversarial training is not applicable
- Extension: KNN-based online adversarial training
  1. For each $\hat{\boldsymbol{x}}$, find its KNN $\mathbb{N}(\hat{\boldsymbol{x}};D)$ from the training set D
  2. Augment $\mathbb{N}(\hat{\boldsymbol{x}};D)$ with adversarial examples (cyan points) perturbed from $\mathbb{N}(\hat{\boldsymbol{x}};D)$
  3. Fine-tune the networks weights θ based on $\mathbb{N}(\hat{\boldsymbol{x}};D)$
  4. Inference $\hat{\boldsymbol{y}} = f(\hat{\boldsymbol{x}};\theta)$

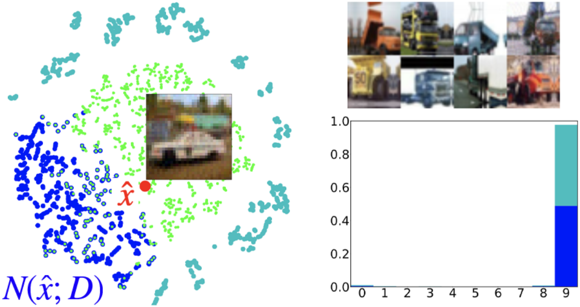


(a)

Unfortunately, It Does Not Work!

Unfortunately, It Does Not Work!

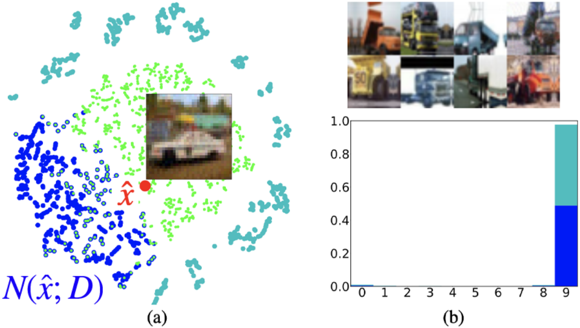
- Figure (b) shows a histogram of $\mathbb{N}(\hat{\boldsymbol{x}}; D)$ w.r.t. different labels (x-axis)



(a) (b)

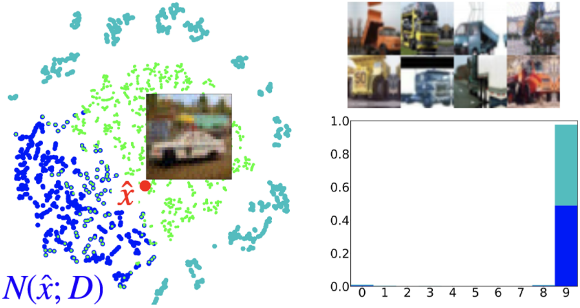
Unfortunately, It Does Not Work!

- Figure (b) shows a histogram of $\mathbb{N}(\hat{\boldsymbol{x}};D)$ w.r.t. different labels (x-axis)
- $\mathbb{N}(\hat{\boldsymbol{x}};D)$ contains examples of the same label
    - The adversarial point $\hat{\boldsymbol{x}}$ can mislead KNN selection



Unfortunately, It Does Not Work!

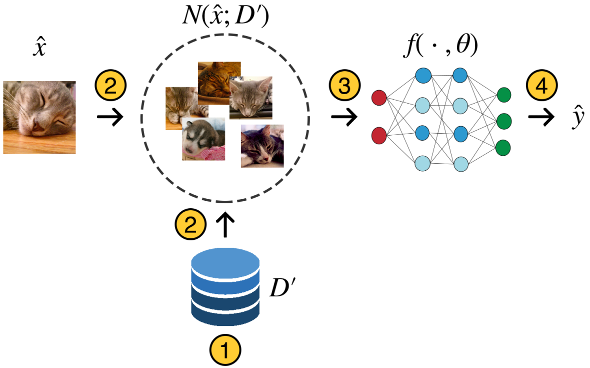
- Figure (b) shows a histogram of $\mathbb{N}(\hat{\boldsymbol{x}}; D)$ w.r.t. different labels (x-axis)
- $\mathbb{N}(\hat{\boldsymbol{x}}; D)$ contains examples of the same label
  - The adversarial point $\hat{\boldsymbol{x}}$ can mislead KNN selection
- Therefore, the fine-tuned θ ends up being *less* robust



(a) (b)

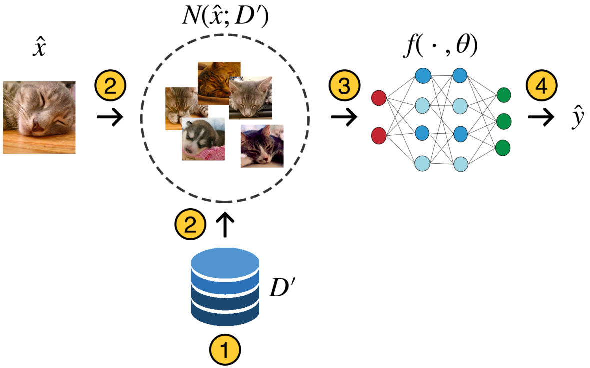
Runtime Masking and Cleansing (RMC)

- RMC ***precomputes*** adversarial examples
  1. Augment D with adversarial examples to get D'
  2. Given a test point $\hat{\boldsymbol{x}}$, find its KNN $\mathbb{N}(\hat{\boldsymbol{x}}; D)'$ from D'



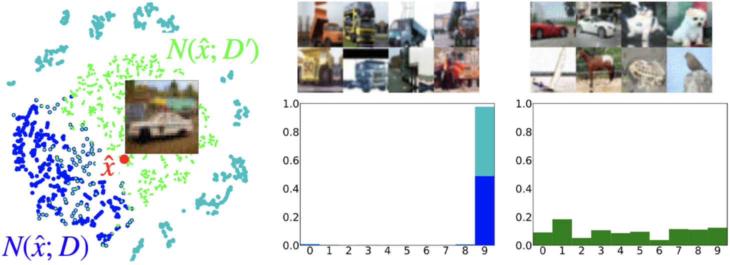
Runtime Masking and Cleansing (RMC)

- RMC ***precomputes*** adversarial examples
  1. Augment D with adversarial examples to get D'
  2. Given a test point $\hat{\boldsymbol{x}}$, find its KNN $\mathbb{N}(\hat{\boldsymbol{x}}; D)'$ from D'
  3. Adapt the networks weights θ based on $\mathbb{N}(\hat{\boldsymbol{x}}; D')$
  4. Inference $\hat{\boldsymbol{y}} = f(\hat{\boldsymbol{x}}; \theta)$



Why Does It Work?

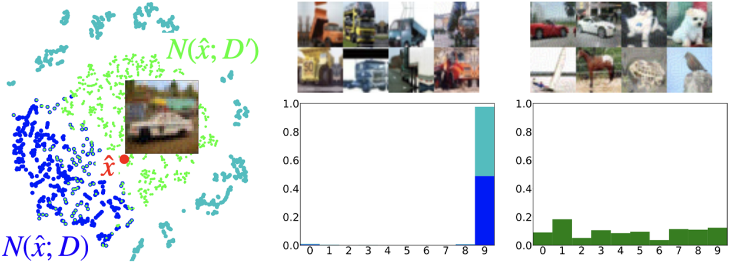
- As Figure (c) shows, $\mathbb{N}(\hat{\boldsymbol{x}}; D')$ is no longer misled by the adversarial $\hat{\boldsymbol{x}}$



(a) (b) (c)

Why Does It Work?

- As Figure (c) shows, $\mathbb{N}(\hat{\boldsymbol{x}}; D')$ is no longer misled by the adversarial $\hat{\boldsymbol{x}}$
- Defense effects:
  - The diverse-labeled $\mathbb{N}(\hat{\boldsymbol{x}}; D')$ ***cleanses*** the θ of the non-robust patterns
  - Also, dynamically ***masks*** the network gradients



(a) (b) (c)

Outline

Datasets

- MNIST
- CIFAR-10
- ImageNet

Outline

MNIST & CIFAR-10

Table 1. Train-time white-box attacks
($\epsilon = 0.3$) on MNIST.

	Acc.	Robustness				
		FGSM	**BIM**	**PGD**	**CW-L2**	**JSMA**
Regularly Trained						
None	**99.3**	11.6	0.6	0.5	0.7	14.1
DeepNN	99.2	12.3	0.6	0.5	75.3	58.2
WebNN	98.2	70.4	82.6	85.3	87.4	87.1
RMC	**99.3**	**99.3**	**99.3**	**99.3**	**99.3**	**99.1**
Adversarially Trained w. FGSM						
None	99	94	51.4	0.7	16.3	42.9
DeepNN	98.8	94	56.9	1.7	85.9	77.2
WebNN	98.6	94.3	85.2	90.8	89.1	87.9
RMC	**99.2**	**98.6**	**98.9**	**98.9**	**98.7**	**98.8**
Adversarially Trained w. PGD						
None	99.1	96.6	93	94.8	65.6	94.6
DeepNN	98.8	96.4	94.5	95.8	91	95.4
WebNN	98.7	96.5	94.5	95.8	91	97.5
RMC	**99.2**	**98.2**	**97.5**	**97.8**	**99.1**	**98.9**
Regularly Trained w. Jacobbian Reg.						
None	94.8	22.1	7.6	8	13.7	26.5
DeepNN	95.9	21.1	8.9	9.6	55.7	41
WebNN	94.2	55.5	55.6	58.3	79	66.4
RMC	**99.3**	**98.9**	**98.9**	**99.1**	**99.2**	**98**
Regularly Trained w. Cross-Lipschitz Reg.						
None	**99.3**	70.6	30.7	19.3	23.8	48.6
DeepNN	99.2	73.2	37.5	22.3	72.7	73.4
WebNN	97	79.8	75.1	74.4	82.8	85.5
RMC	**99.3**	**99.2**	**99.2**	**99.3**	**99.2**	**98.2**

Table 2. Train-time white-box attacks
($\epsilon = 8/255$) on CIFAR-10.

	Acc.	Robustness				
		FGSM	**BIM**	**PGD**	**CW-L2**	**JSMA**
Regularly Trained						
None	83.3	25.3	8.5	6.7	9.4	8
DeepNN	84.3	26.5	9.2	8	55.2	23
WebNN	81.8	40.9	47.8	48.6	64.6	38.3
RMC	**89.3**	**85.3**	**86.7**	**87.5**	**89.7**	**88.6**
Adversarially Trained w. FGSM						
None	83.2	78.9	9.3	8.3	8.8	17.3
DeepNN	85	81	9.9	9.1	56.2	23.1
WebNN	80	81.9	42.5	43.3	64.2	34.4
RMC	**89.3**	**87.3**	**87.1**	**88.7**	**89.7**	**89.1**
Adversarially Trained w. PGD						
None	78.7	50.6	43.6	44.3	11.5	7.8
DeepNN	75.6	52.5	45.6	45.8	48.7	38.5
WebNN	73.5	54	48.1	48.4	53.4	47
RMC	**88.3**	**81.2**	**81.1**	**80.7**	**88.7**	**87.7**
Regularly Trained w. Jacobbian Reg.						
None	86.3	37.9	20.6	20.2	8	10.2
DeepNN	**87.8**	39.8	21	21.4	63.1	41.1
WebNN	76.2	49.9	55.5	55.5	68.9	49
RMC	87.1	**82.4**	**83.6**	**83.5**	**86.6**	**88.4**
Regularly Trained w. Cross-Lipschitz Reg.						
None	85.3	31	18.6	18.4	8.4	13
DeepNN	**86.9**	32.6	19	19	61.9	36.8
WebNN	74.5	46.5	51	50.5	67.1	48.6
RMC	85	**79.8**	**80.8**	**81.1**	**84.9**	**86.9**

ImageNet

Table 3. Train-time white-box attacks on ImageNet.

	Acc.	**Robustness**	
		$\epsilon = 8/255$	$\epsilon = 16/255$
None	72.9	8.5	5.2
Adv. Trained	62.3	N/A	52.5
DB	65.3	N/A	55.7
DeepNN	26.6	12.9	8.7
WebNN	27.8	18.8	15.2
RMC	**73.6**	**62.4**	**55.9**

ImageNet

Table 3. Train-time white-box attacks on ImageNet.

	Acc.	**Robustness**	
		$\epsilon = 8/255$	$\epsilon = 16/255$
None	72.9	8.5	5.2
Adv. Trained	62.3	N/A	52.5
DB	65.3	N/A	55.7
DeepNN	26.6	12.9	8.7
WebNN	27.8	18.8	15.2
RMC	**73.6**	**62.4**	**55.9**

- For all datasets, RMC achieves the state-of-the-art robustness
- RMC yields significantly ***higher clean accuracy***

ImageNet

Table 3. Train-time white-box attacks on ImageNet.

	Acc.	**Robustness**	
		$\epsilon = 8/255$	$\epsilon = 16/255$
None	72.9	8.5	5.2
Adv. Trained	62.3	N/A	52.5
DB	65.3	N/A	55.7
DeepNN	26.6	12.9	8.7
WebNN	27.8	18.8	15.2
RMC	**73.6**	**62.4**	**55.9**

- For all datasets, RMC achieves the state-of-the-art robustness
- RMC yields significantly ***higher clean accuracy***
    - RMC does not enforce a smooth decision boundary

ImageNet

Table 3. Train-time white-box attacks on ImageNet.

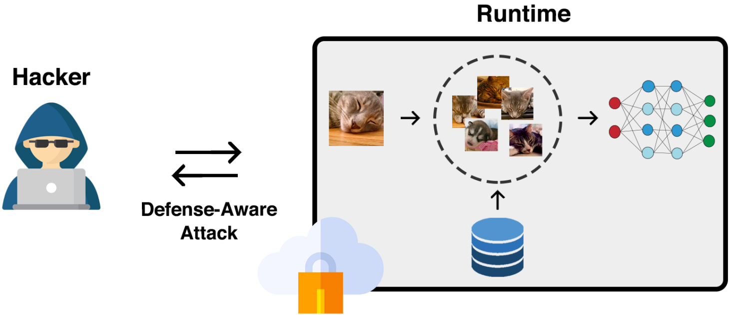
	Acc.	Robustness	
		$\epsilon = 8/255$	$\epsilon = 16/255$
None	72.9	8.5	5.2
Adv. Trained	62.3	N/A	52.5
DB	65.3	N/A	55.7
DeepNN	26.6	12.9	8.7
WebNN	27.8	18.8	15.2
RMC	**73.6**	**62.4**	**55.9**

- For all datasets, RMC achieves the state-of-the-art robustness
- RMC yields significantly ***higher clean accuracy***
  - RMC does not enforce a smooth decision boundary
- For gray- black-box attacks, please refer to our main paper

Outline

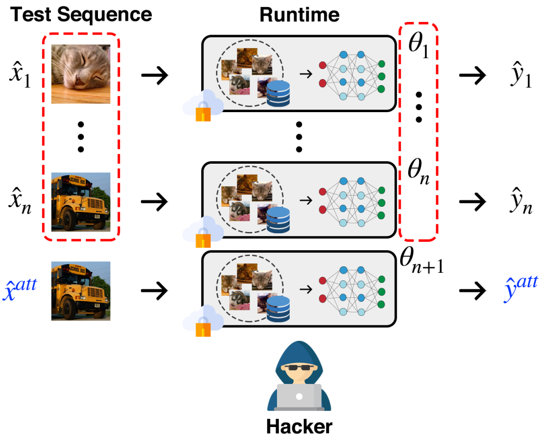
Defense-Aware Attacks

- At runtime, attackers may be aware of RMC and try to circumvent it



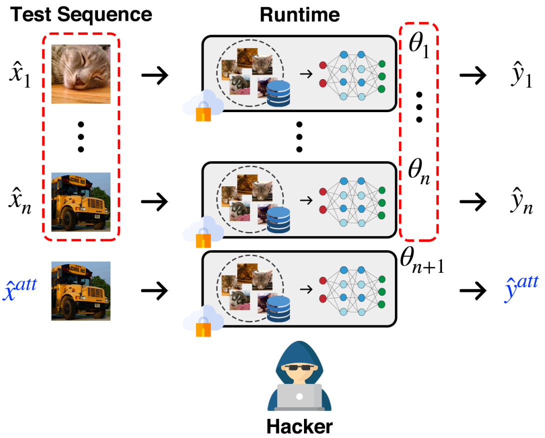
Strong Attack: PGD-Skip

- Assumes that all information is exposed, including
  - Test sequence
  - D' and adapted model weights θ's



Strong Attack: PGD-Skip

- Assumes that all information is exposed, including
  - Test sequence
  - D' and adapted model weights θ's
- I.e., the attack point $\hat{x}^{\text{att}}$ can ***bypass all previous adaptations***



RMC Could be Broken by PGD-Skip

- About 15% robustness

Table 5. Robustness of RMC under the Defense-Aware Attack

q	**0**	**50**	**100**
$p = 100$	14.9	19.8	20.8

(a) PGD-Skip-Delayed

However, PGD-Skip is Unrealistic

- Two strong assumptions

1. Access to all data points at runtime
2. No delay to place an attack point $\hat{\boldsymbol{x}}^{\text{att}}$

However, PGD-Skip is Unrealistic

- Two strong assumptions

1. Access to all data points at runtime
   - When model is publicly deployed, it is unlikely to eavesdrop every user's input $\hat{\boldsymbol{x}}$
2. No delay to place an attack point $\hat{\boldsymbol{x}}^{\text{att}}$

However, PGD-Skip is Unrealistic

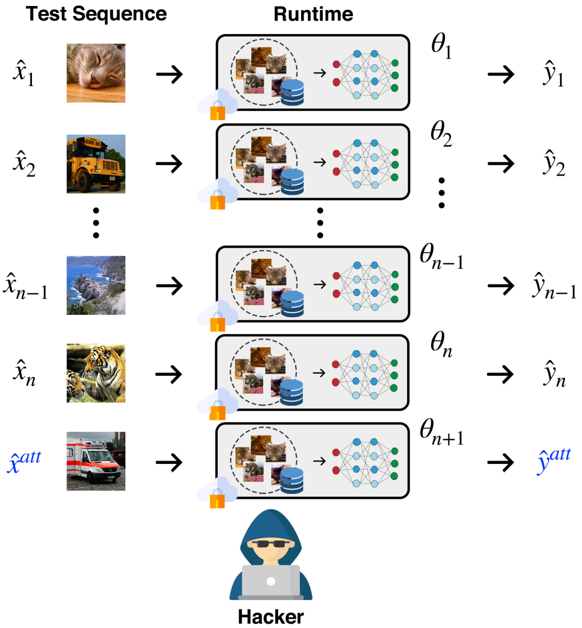
- Two strong assumptions

1. Access to all data points at runtime
   - When model is publicly deployed, it is unlikely to eavesdrop every user's input $\hat{\boldsymbol{x}}$
2. No delay to place an attack point $\hat{\boldsymbol{x}}^{\text{att}}$
   - It is hard to mute other users

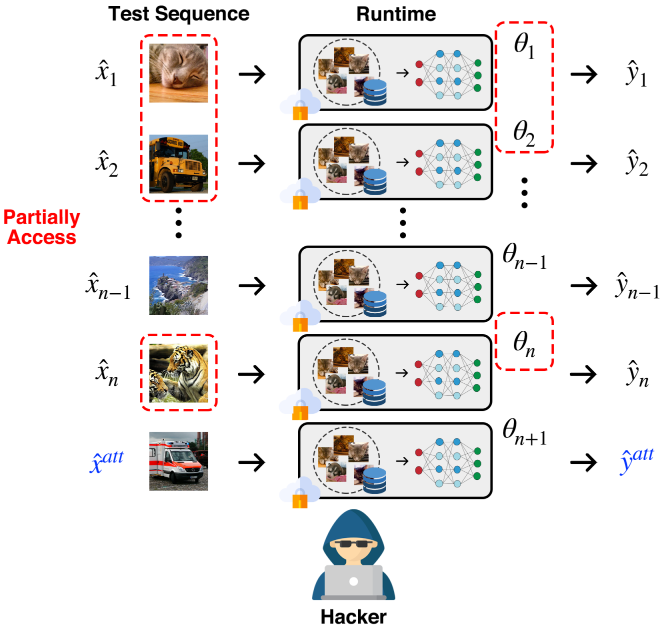
More Realistic Defense-Aware Attacks

- PGD-Skip-Partial
  - Only partial points in the input sequence are known
- PGD-Skip-Delayed
  - The adversary generates/places an attack point $\hat{\boldsymbol{x}}^{\text{att}}$ with some delay

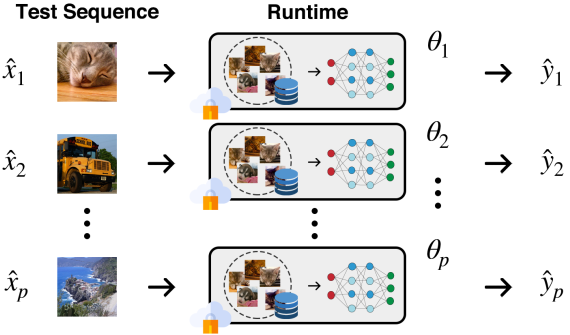
PGD-Skip-Partial



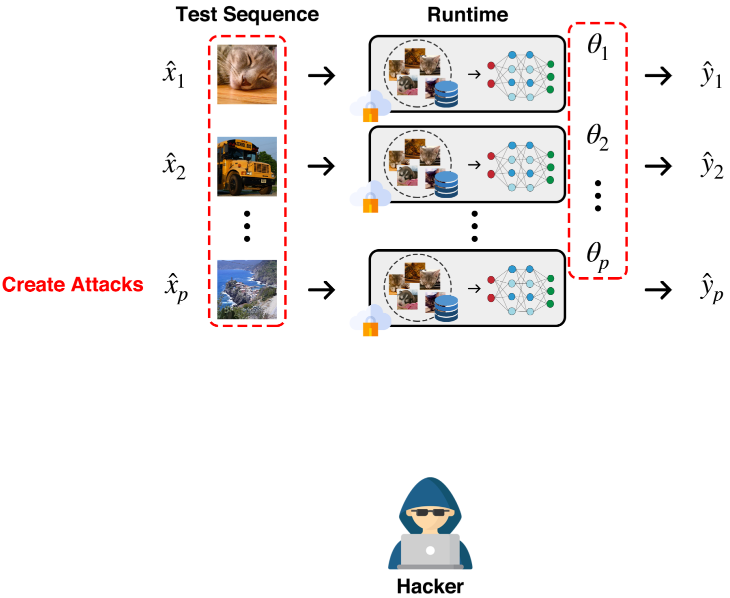
PGD-Skip-Partial



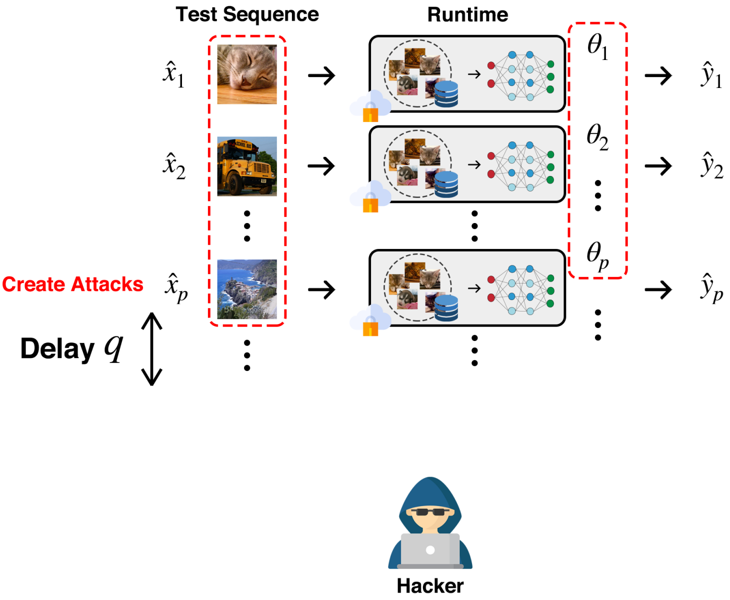
PGD-Skip-Delayed



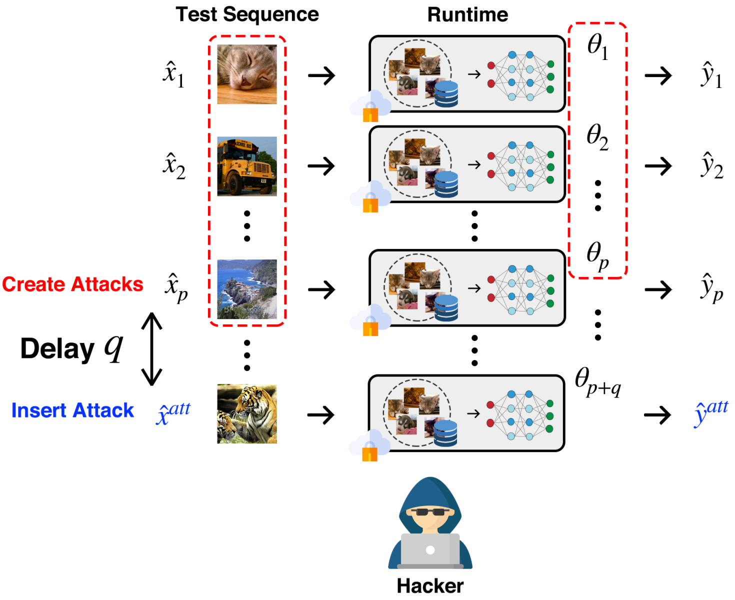
PGD-Skip-Delayed



PGD-Skip-Delayed



PGD-Skip-Delayed



The Revenge of RMC

- With some minor tweaks, RMC can defend these two attacks
  - q: delay of PGD-Skip-Delayed
  - "known:" portion of eavesdropped points by PGD-Skip-Partial

Table 5. Performance of RMC+ under the
(a) PGD-Skip-Delayed and (b) PGD-Skip-Partial attacks.

	δ = 0.5			**δ = 0.75**			**δ = 1**		
q	**0**	**50**	**100**	**0**	**50**	**100**	**0**	**50**	**100**
$p = 50$	19.3	51	63.7	20.4	48.9	62.8	20.9	44.1	48.6
$p = 100$	25.3	50.8	55.1	25.5	51.5	56.1	39.5	41	30.6

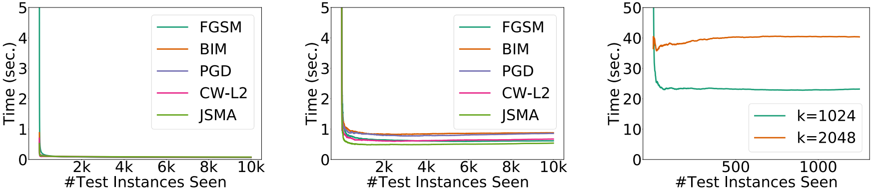
(a) PGD-Skip-Delayed with $\mathbb{D}'$ replacement

	δ = 0.5			**δ = 0.75**			**δ = 1**		
known	**30%**	**50%**	**70%**	**30%**	**50%**	**70%**	**30%**	**50%**	**70%**
$p = 50$	48.4	48.1	45.2	47.5	49	43.3	50.4	52.4	49.5
$p = 100$	64.1	63.1	63.5	64.3	61.1	59.4	63.3	61.7	61.8
$p = 150$	69.2	69.2	68.5	68.9	68.7	68.3	59.6	61.1	64.8

(b) PGD-Skip-Partial with $\mathbb{D}'$ replacement

How Long is the Delay Incurred by RMC at Runtime?

- About 1 second on CIFAR-10 and a delay of 20-40 seconds on ImageNet
  - May be acceptable for non-realtime applications
  - Can be accelerated by existing techniques



Outline

Conclusions & Implications

- We proposed RMC, the first runtime defense
  - Leverages ***potentially large test data*** to improve the robustness of a model after deployment

Conclusions & Implications

- We proposed RMC, the first runtime defense
  - Leverages ***potentially large test data*** to improve the robustness of a model after deployment
- Implications:
  - Currently, new attacks trigger new deployments
  - RMC could end this endless chasing game

Conclusions & Implications

- We proposed RMC, the first runtime defense
  - Leverages ***potentially large test data*** to improve the robustness of a model after deployment
- Implications:
  - Currently, new attacks trigger new deployments
  - RMC could end this endless chasing game
- Questions? Chat with us at session time!
  - Or email to: chyuan@datalab.cs.nthu.edu.tw