JOHNS HOPKINS UNIVERSITY

Deep Learning (EN.520.638)
Spring 2023

Adversarial Attacks and Defenses

Shao-Yuan Lo
(Advisor: Prof. Vishal M. Patel)
Johns Hopkins University
https://shaoyuanlo.github.io

Part I: Adversarial Examples

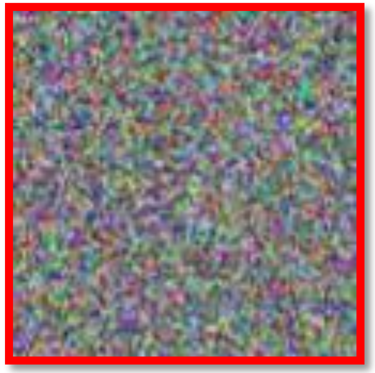
What's Adversarial Example?

$$x_{adv} = x + \delta$$

$$f_\theta(\boldsymbol{x}_{adv}) \neq y$$

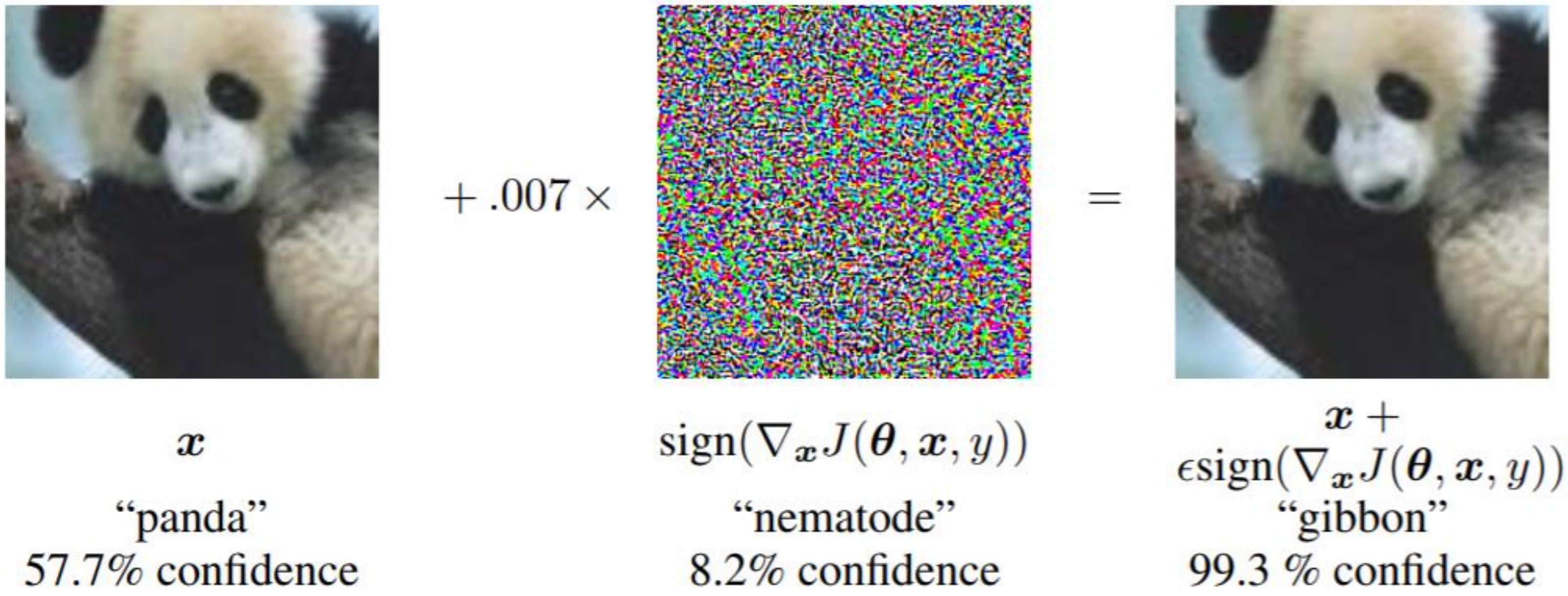
What's Adversarial Example?

$f_\theta($  $) = "Dog"$

$f_\theta($  $+$  $) = "Cat"$

What’s Adversarial Example?

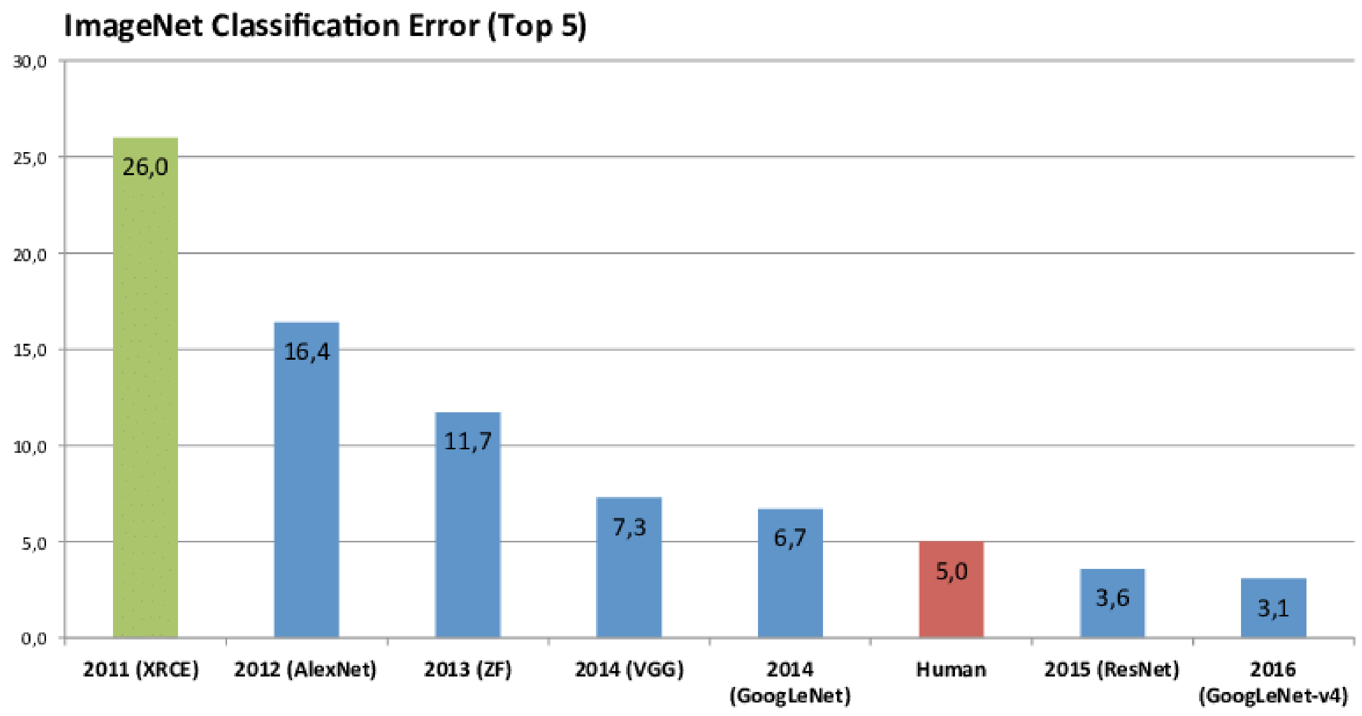
- Adversarial examples are visually **similar** to **human** but can **fool** well-trained **deep networks**.
- Deep networks are **NOT robust** against adversarial examples.



[Goodfellow et al. ICLR’15]

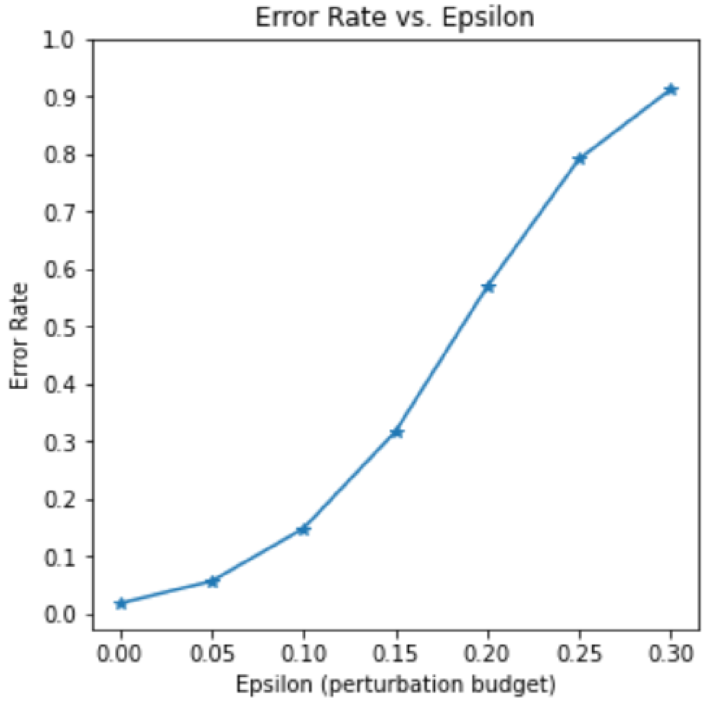
Deep Networks are NOT Robust

- ImageNet (1000 classes)

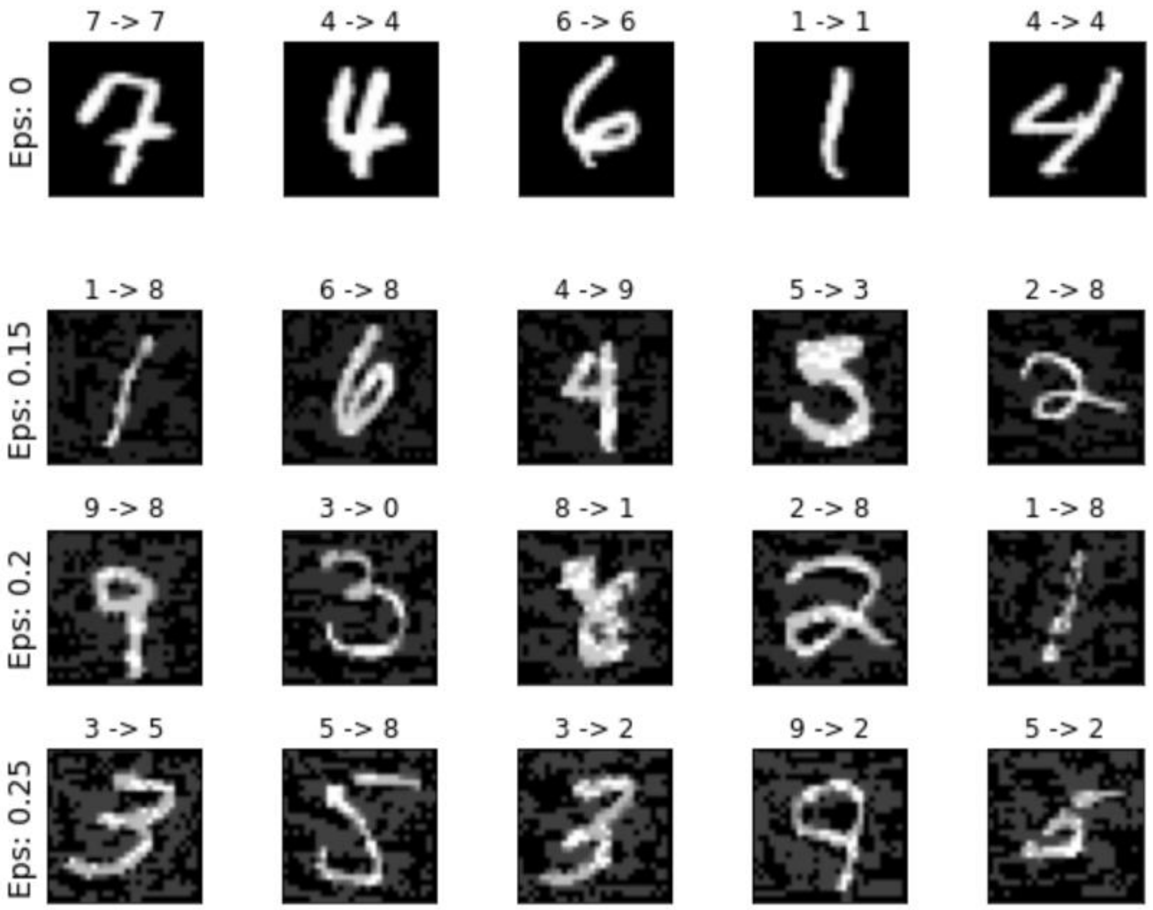


https://devopedia.org/imagenet

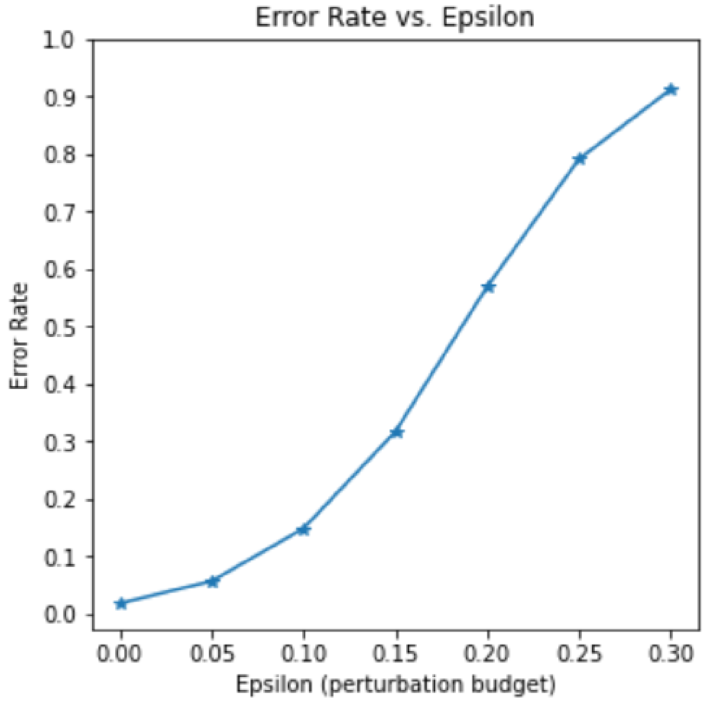
- MNIST (10 digits)



Deep Networks are NOT Robust



- MNIST (10 digits)



[Feizi. Foundations of Deep Learning. Fall 2021]

Generate Adversarial Examples

- Train a model
  - **min Loss(f(x), y; θ)**
  - **Minimize** the loss function w.r.t. **model parameters θ**

- Generate adversarial examples
  - Most common method: Gradient-based method, e.g., FGSM.
  - **max Loss(f(x+δ), y; θ)**
  - **Maximize** the loss function w.r.t. **adversarial perturbation δ**

Generate Adversarial Examples

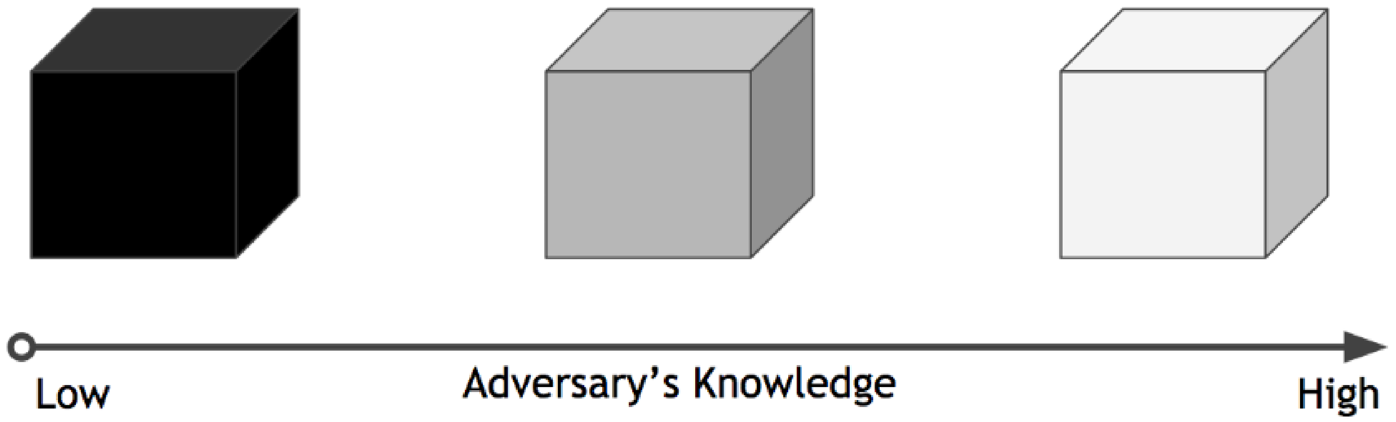
- Generate adversarial examples
  - Most common method: Gradient-based method, e.g., FGSM.
  - **max Loss(f(x+δ), y; θ)**
  - **Maximize** the loss function w.r.t. **adversarial perturbation δ**
- Perturbation budget **‖δ‖**
  - Constrain the **magnitude** of perturbation, e.g., **Lp-norm**.
  - Constrain the **region** of perturbation, e.g., **patch attack**.

Generate Adversarial Examples

- FGSM attack [Goodfellow et al. ICLR’15]: $\boldsymbol{x}_{adv} = \boldsymbol{x} + \epsilon \cdot sign(\nabla_{\boldsymbol{x}} L(\boldsymbol{x}, y))$
  - One-step gradient-based method
- PGD attack [Madry et al. ICLR’18]: $\boldsymbol{x}_{t+1}^{*} = \boldsymbol{x}_{t}^{*} + \alpha \cdot sign(\nabla_{\boldsymbol{x}} L(\boldsymbol{x}_{t}^{*}, y))$
  - Iterative gradient-based method
- C&W attack [Carlini & Wagner, SP’17]: $\arg\min_{\boldsymbol{x}_{adv}} \ \beta \parallel \boldsymbol{x}_{adv} - \boldsymbol{x} \parallel_p - L(\boldsymbol{x}_{adv}, y)$
  - Optimization-based method
- Mult attack [Lo & Patel, AVSS’21]: $\mathbf{x}^{t+1} = Clip_{\mathbf{x},\epsilon_m}^{RB-\ell_\infty}\{\mathbf{x}^t \odot \alpha_m^{sign(\nabla_{\mathbf{x}^t}\mathcal{L}(\mathbf{x}^t,\mathbf{y};\boldsymbol{\theta}))}\}$
  - Multiplicative gradient-based method

Adversary's Knowledge

- White-box attack
- Black-box attack
- Gray-box attack



https://slidetodoc.com/unclassified-if-you-know-the-enemy-and-know

Untargeted/Targeted Attacks

- Untargeted attack

$$f_\theta(\boldsymbol{x}_{adv}) \neq y$$

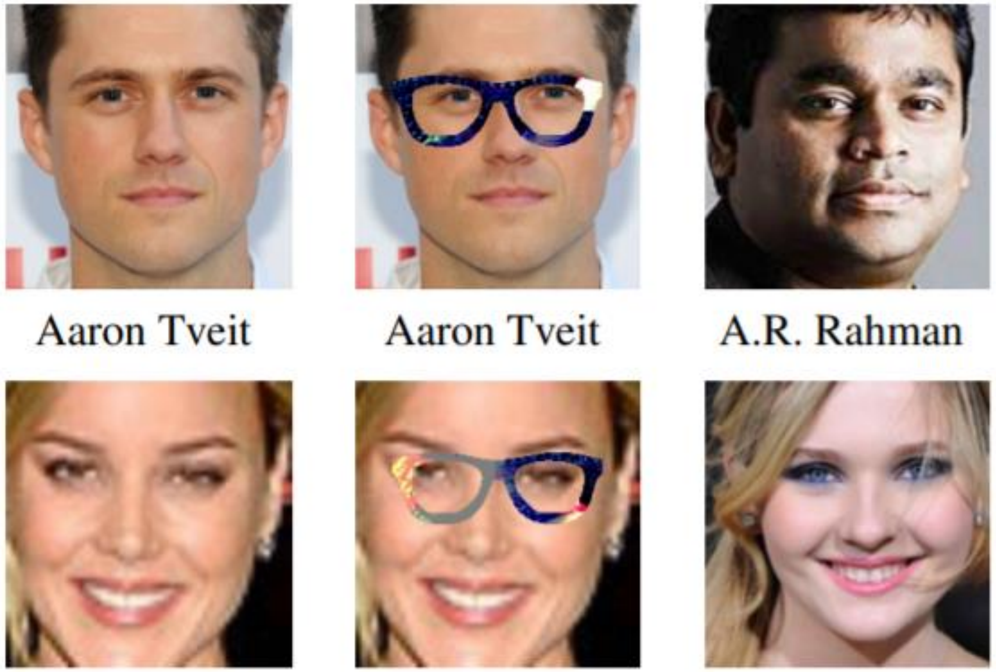
$$L_{adv}(\boldsymbol{x}) = -L(\boldsymbol{x}, y)$$

- Targeted attack

$$f_\theta(\boldsymbol{x}_{adv}) = y_{adv}, \quad y_{adv} \neq y$$

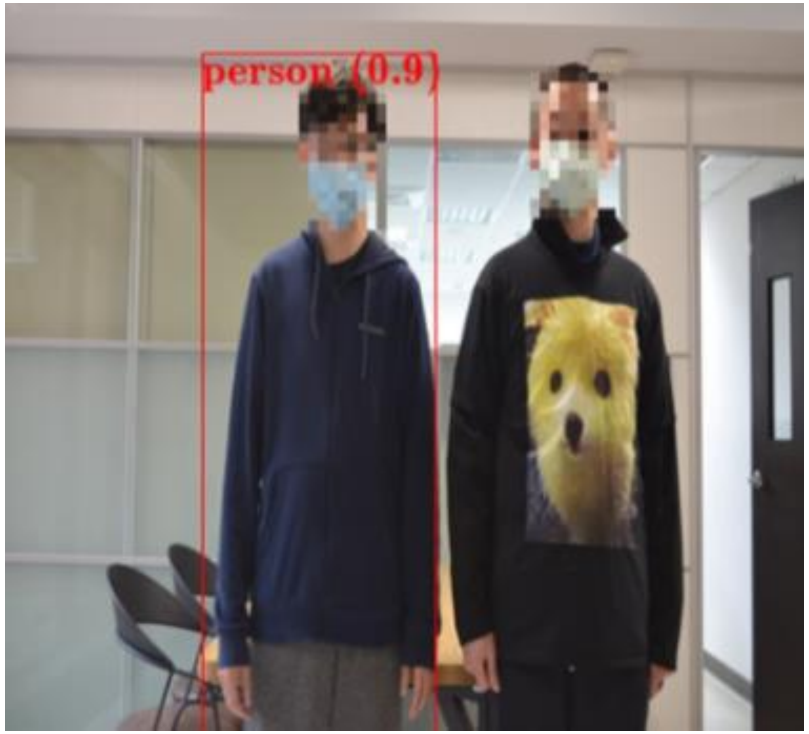
$$L_{adv}(\boldsymbol{x}) = L(\boldsymbol{x}, y_{adv})$$

Adversarial Examples in Different Types



[Wu et al. ICLR’20]

Adversarial Examples in Physical World



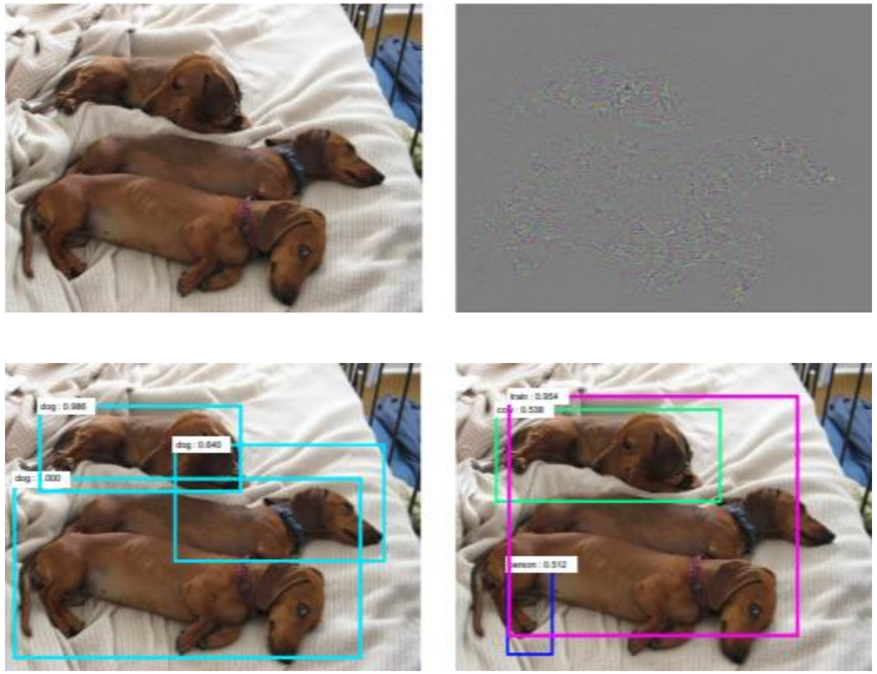
[Hu et al. ICCV'21]

[Ranjan et al. ICCV'19]

Adversarial Examples in Different Tasks

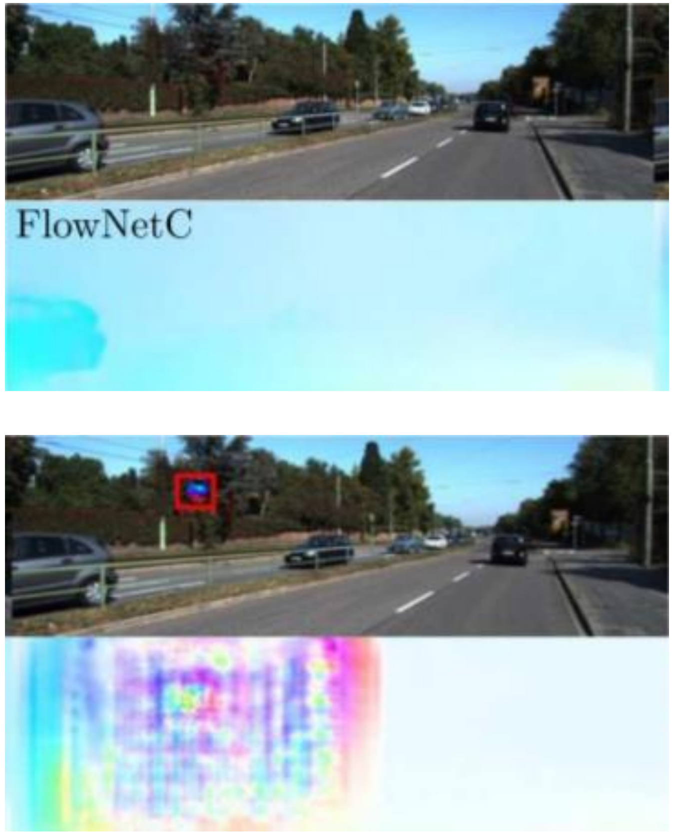
Semantic segmentation

Object detection



[Xie et al. ICCV'17]

Optical flow

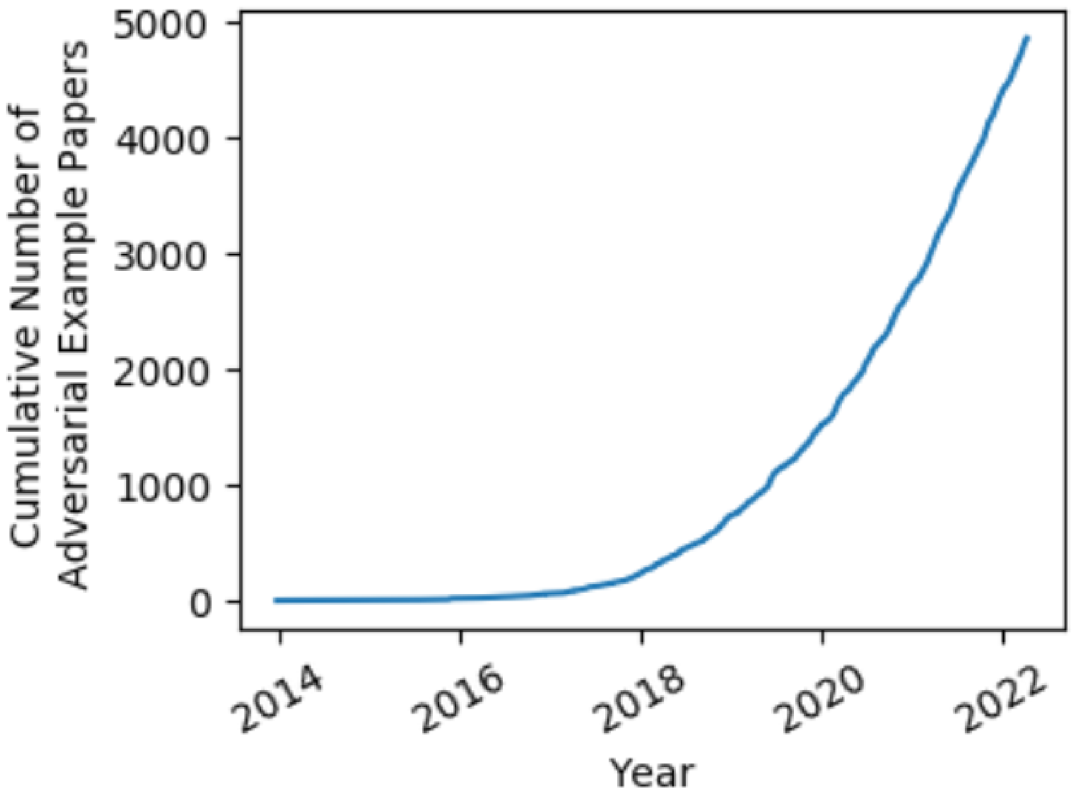


[Ranjan et al. ICCV'19]

Why Study Adversarial Examples?

- Deep learning models are being widely used in real-world applications, such as autonomous driving. Their **safety** is critical.

- We aim to build **robust** DL models that we can **trust**.



https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html

Why Study Adversarial Examples?

- DARPA (Defense Advanced Research Projects Agency)

JHU
MIT
CMU
UMD
USC
Google
IBM
Intel
…

GARD
GUARANTEEING AI ROBUSTNESS AGAINST DECEPTION

https://www.darpa.mil/news-events/2021-12-21

Adversarial Defenses

- **Image transformation**: Remove perturbations from input images.

$$f_\theta(\boldsymbol{x}_{adv}) \neq y$$

$$f_\theta(\boldsymbol{T}(\boldsymbol{x}_{adv})) = y$$

- **Adversarial training**: Enhance the robustness of networks itself.

$$\theta^* = \arg\min_\theta \mathbb{E}_{(x,y)\sim\mathbb{D}}\left[\max_{\delta\in\mathbb{S}} L(x+\delta, y; \theta)\right]$$

Part II: Image Transformation-based Adversarial Defenses

Image Transformation-based Defenses

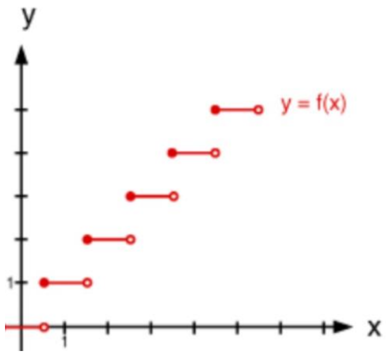
- Image preprocessing methods:
  - **Color precision reduction** (pixel value quantization)
  - **JPEG compression** (frequency domain quantization)
  - **Denoising** (Gaussian blur, median, mean, bilateral, non-local means, etc.)
  - **Color space** (RGB, HSV, YUV, LAB, etc.)
  - **Contrast** (histogram equalization)
  - **Noise injection** (add noise on adversarial examples)
  - **FFT perturbation** (similar to JPEG)
  - **Swirl** (rotation)
  - **Resizing**
  - **Halftoning**

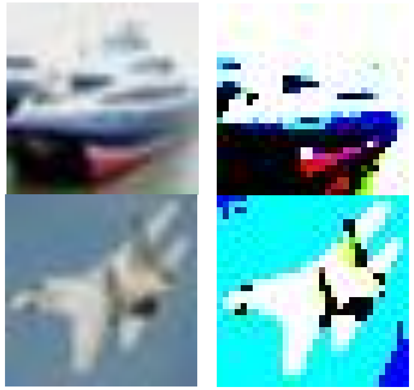
[Das et al. KDD'18]
[Xu et al. NDSS'18]
[Guo et al. ICLR'18]
[Raff et al. CVPR'19]
[Lo & Patel, ICIP'21]

- Generative model methods:
  - **Defense-GAN**
    [Samangouei et al. ICLR'18]
  - **PixelDefend**
    [Song et al. ICLR'18]

Image Transformation-based Defenses

- **Color precision reduction**: Quantize the image pixel values.



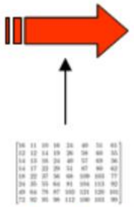


- **JPEG compression**: Quantization in the frequency domain.

1. Create DCT coefficient matrix

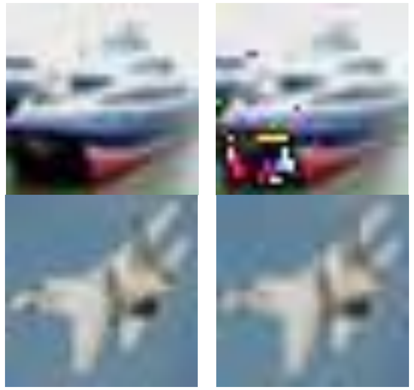
$$\begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

2. Scale each coefficient



3. Convert to integers

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



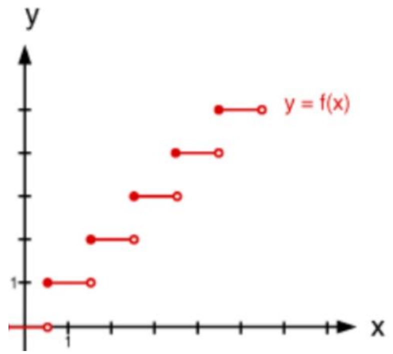
Adaptive Attacks

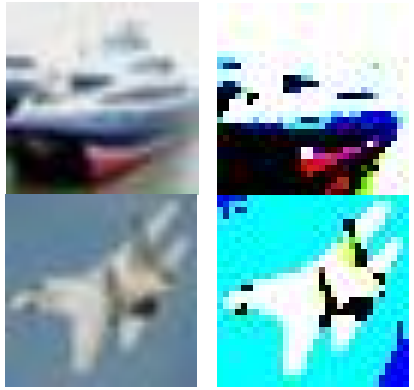
- [Athalye et al. ICML'18] proposed **adaptive attacks**, which defeat most image transformation-based defenses.
- Strong white-box attacks are generated through **gradients**, e.g., FGSM and PGD attacks.
- Image transformation-based defenses mostly rely on **gradient masking**, which can be defeated by adaptive attacks.
- Three types of masked gradients:
  - Shattered gradients ← BPDA
  - Stochastic gradients ← EOT
  - Exploding & vanishing gradients ← BPDA or EOT or Both

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%

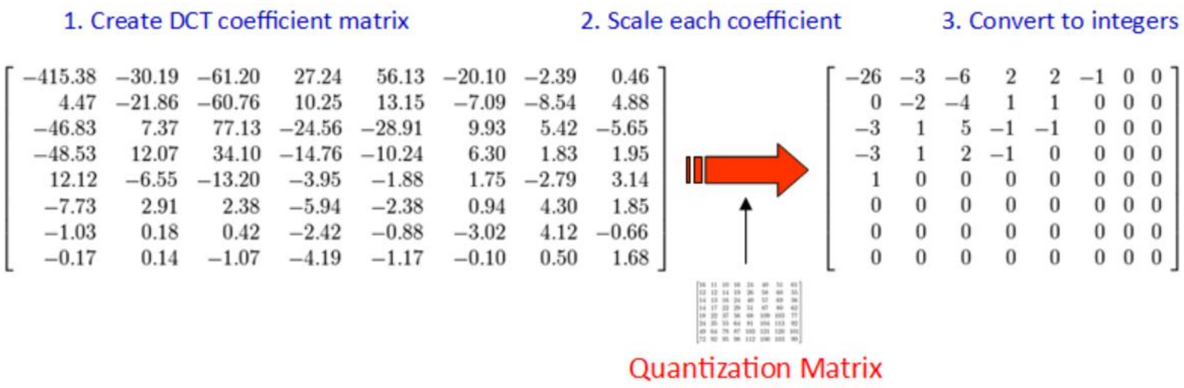
Shattered Gradients ← BPDA

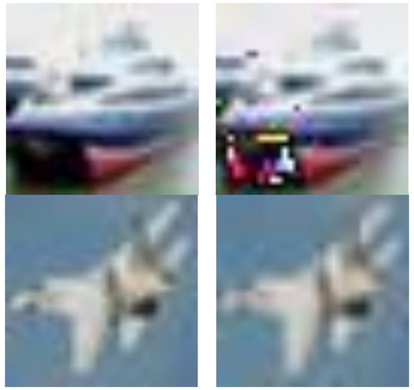
- **Color precision reduction**: Quantize the image pixel values.





- **JPEG compression**: Quantization in the frequency domain.





- Quantization is **non-differentiable**.

Shattered Gradients ← BPDA

- **BPDA**: Using the **identity function** h() as a surrogate function can defeat their defenses.

$T(x) \approx x$ $\qquad$ $Forward\ pass: f(T(x))$

$h(x) = x$ $\qquad$ $Backward\ pass: \nabla_x f(h(x))$

- Most image transformation-based defenses can be defeated.

Robust Image Transformation-based Defenses

- Error Diffusion Halftoning Against Adversarial Examples [Lo & Patel, ICIP'21]

- Barrage of Random Transforms for Adversarially Robust Defense [Raff et al. CVPR'19]

Error Diffusion Halftoning

- Quantize each pixel in the raster order one-by-one, and spread the quantization error to the neighboring pixels.

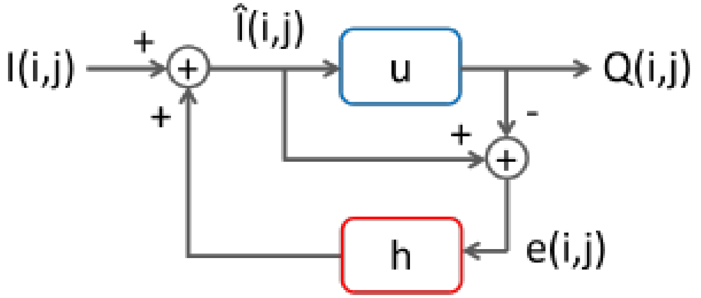
$$\hat{I}(i,j) = I(i,j) + \sum_{m,n \in S} h(m,n)e(i-m, j-n)$$

$$Q(i,j) = u(\hat{I}(i,j) - \theta) \quad e(i,j) = \hat{I}(i,j) - Q(i,j)$$

Input image (I)

Halftone (Q)

Floyd and Steinberg. Proceedings of the Society of Information Display, 1976.

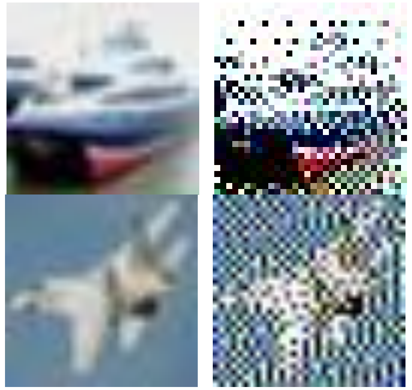


u: unit step function

h: error filter

Error Diffusion Halftoning

- The **quantization operation** invalid the adversarial variations.
- **Updating the values of the neighboring pixels repeatedly** makes the adaptive attacks hard to identify the mapping between the original image and the corresponding halftone.
- **Spreading quantization errors produces** better halftoning quality and tends to enhance edges and object boundary in an image.
- Take **both** adversarial robustness and clean data performance.
- Complementary to adversarial training.



Error Diffusion Halftoning

- Dataset: CIFAR-10
- Attacks: Adaptive (via identity function) PGD and Mult

Method	Training	Clean	PGD-ℓ_∞	PGD-ℓ_2	Mult-ℓ_∞	Mult-ℓ_2	Avg$_{adv}$	Avg$_{all}$
Vanilla	Standard training	**94.03**	0.01	0.20	0.05	0.01	0.07	18.86
Gaussian blur		90.17	0.20	1.34	0.17	0.05	0.44	18.39
Non-local means		88.66	0.02	0.49	0.03	0.00	0.14	17.84
JPEG compression		90.06	2.97	4.82	1.81	0.22	2.46	19.98
Bit-depth reduction		78.87	**15.26**	10.84	**10.79**	**4.52**	**10.35**	**24.06**
Halftoning (ours)		88.57	9.53	**11.98**	5.54	1.07	7.03	23.34
Vanilla	Adversarial training	83.31	51.15	50.68	54.10	40.29	49.06	55.91
Gaussian blur		75.96	44.59	47.12	45.07	32.48	42.32	49.04
Non-local means		75.47	44.67	45.29	16.59	14.53	30.27	39.31
JPEG compression		24.97	38.99	43.72	59.15	44.72	46.65	42.31
Bit-depth reduction		71.66	47.34	42.40	48.50	41.63	44.97	50.31
Halftoning (ours)		**84.37**	**60.01**	**56.56**	**67.37**	**88.44**	**68.10**	**71.35**

Barrage of Random Transforms

- Ensemble weak defenses to create a strong defense.
- Fully account for the obfuscated gradients issue.

- All the 25 individual transforms are defeated by adaptive attacks.
- Their stochastic combination makes them significantly stronger.

- 25 transforms in 10 groups:
  - **Color precision reduction** (pixel value quantization)
  - **JPEG compression** (frequency domain quantization)
  - **Denoising** (Gaussian blur, median, mean, bilateral, non-local means, etc.)
  - **Color space** (RGB, HSV, YUV, LAB, etc.)
  - **Contrast** (histogram equalization)
  - **Noise injection** (add noise on adversarial examples)
  - **FFT perturbation** (similar to JPEG)
  - **Swirl** (rotation)
  - **Resizing**
  - **Gray scale**

Barrage of Random Transforms

- Each transform is itself **randomized** (scaling factor, quantization factor, denoisnig parameters, etc.).
- Select ***k*** transforms out of ***n*** (25) transforms, where ***k*** is **randomized**.
- Each transform has a selection probability $\boldsymbol{p_i}$, where $\boldsymbol{p_i}$ is **randomized**.
- Select an ordering $\boldsymbol{\pi}$ for the ***k*** selected transforms, where $\boldsymbol{\pi}$ is **randomized**.

$$f(x) = f(t_{\pi(1)}(t_{\pi(2)}(\ldots(t_{\pi(k)}(A(x))))))$$

$f: model$
$A: adversarial\ attack$

- **Randomness on top of randomness**.

Barrage of Random Transforms

- **BPDA**: If a transform ***t()*** is non-differentiable, train a **neural network** ***f_t()*** to learn the approximation of ***t()***.

  $Use\ \nabla_x f_t(x)\ to\ approximate\ \nabla_x t(x)$

- **EOT**: Approximate the randomness.

  $\mathbb{E}_{t\sim T}\nabla_x f(t(x))$

- Combine BPDA and EOT.

- High randomness → Computational cost of EOT is extremely high

Barrage of Random Transforms

- Dataset: ImageNet
- Attack: Adaptive (via neural network) PGD

	Clean Images		Attacked	
Model	Top-1	Top-5	Top-1	Top-5
Inception v3	78	94	0.7	4.4
Inception v3 w/Adv. Train	78	94	1.5	5.5
ResNet50	76	93	0.0	0.0
ResNet50-BaRT, $k = 5$	65	85	16	51
ResNet50-BaRT, $k = 10$	65	85	36	57

Part III: Adversarial Training-based Defenses

Adversarial Training

- Adversarial training is a strong defense against white-box attacks.
- **Core idea: Train with adversarial examples.**
- Adversarial training does **not** cause masked gradients.
- It has been widely used as a standard baseline defense.

$$\underbrace{\theta^* = \arg\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}} \left[\underbrace{\max_{\delta\in\mathbb{S}} L(x+\delta, y; \theta)}_{\text{Generate adversarial examples}} \right]}_{\text{Train model parameters}}$$

[Madry et al. ICLR’18]

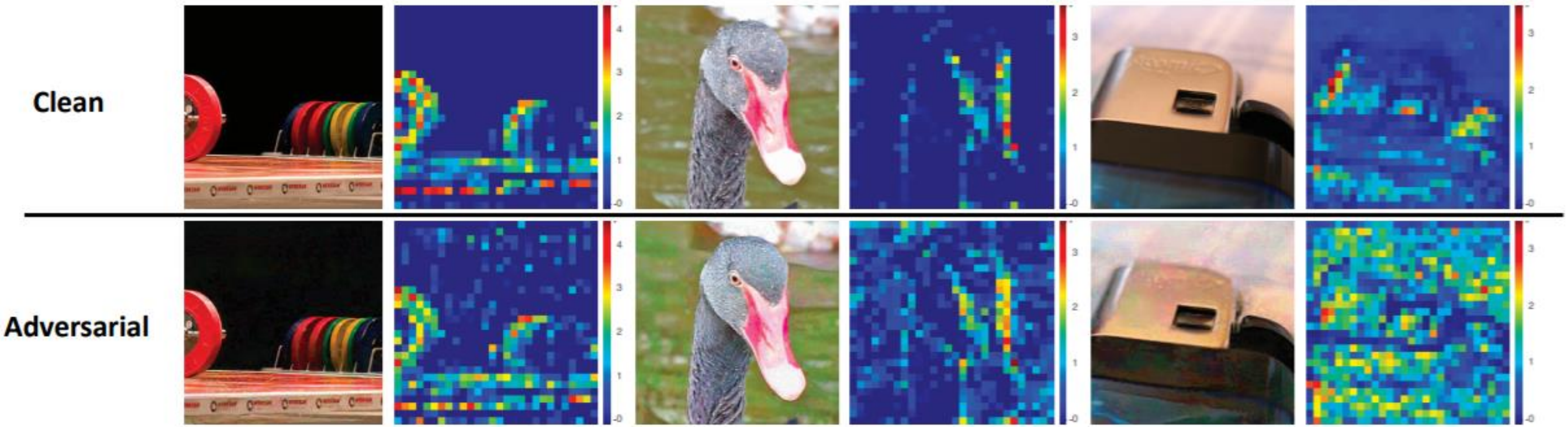
Adversarial Training-based Defenses

- Feature Denoising for Improving Adversarial Robustness [Xie et al. CVPR’19]

- Overcomplete Representations Against Adversarial Videos [Lo et al. ICIP’21]

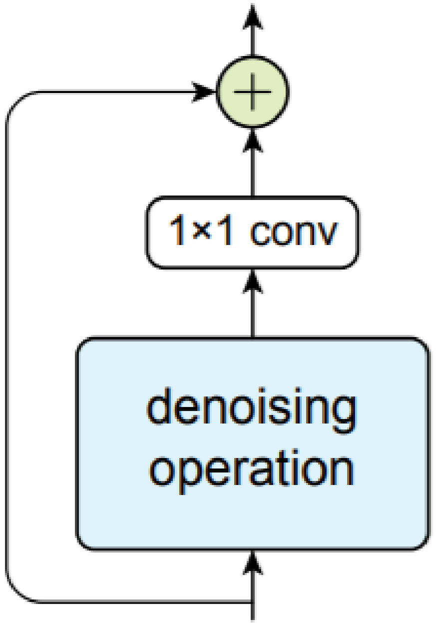
Feature Denoising

- Adversarial perturbations are **small** in the image **pixel** domain, while they are **big** in the **feature** domain.



Feature Denoising

- Traditional image denoising operations:
  - Mean filter
  - Bilateral filter
  - Non-local means
- Denoising operation may lose information.
- Add a **residual connection** to balance the tradeoff between removing noise and retaining original signal.
- Combine adversarial training



layer name	output size	18-layer
conv1	112×112	
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$

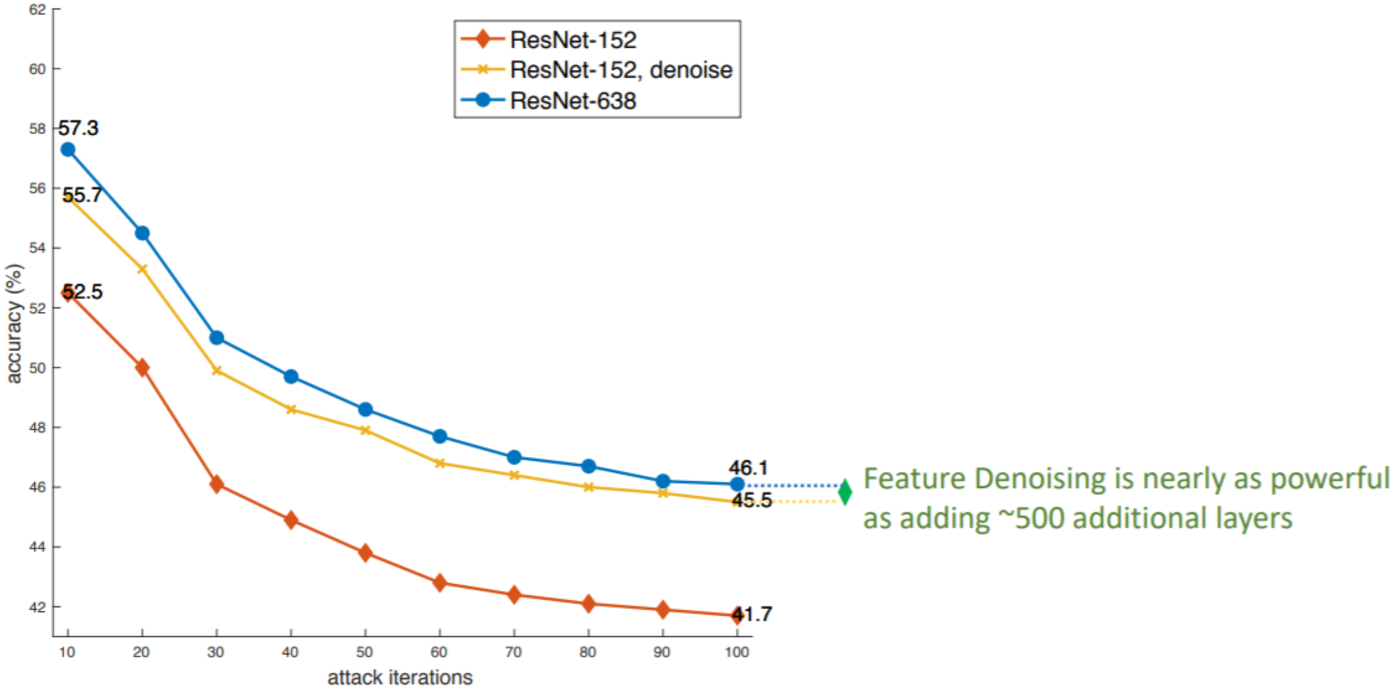
[He et al. CVPR’16]

Feature Denoising

- No non-differentiable components
- No randomness

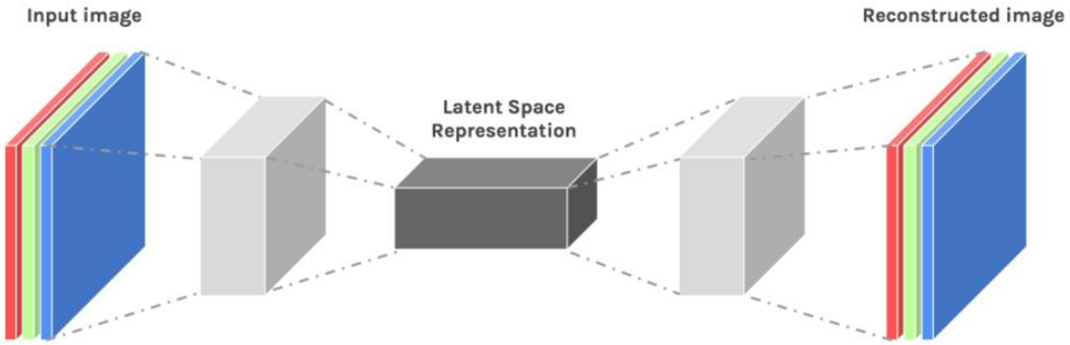
- Can use standard PGD attack to evaluate

- Dataset: ImageNet
- Attack: PGD



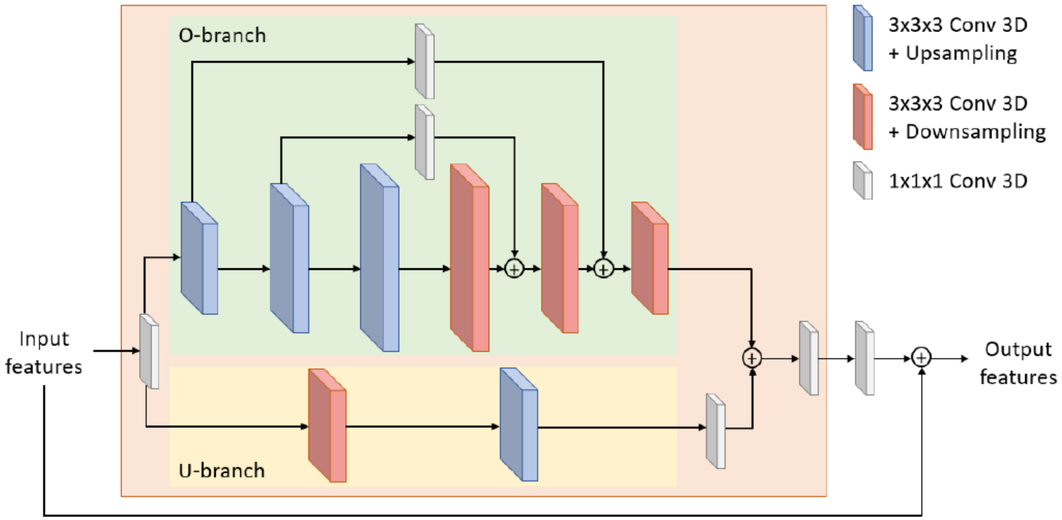
Overcomplete Representations

- A typical autoencoder downsamples features and learns **undercomplete** representations.



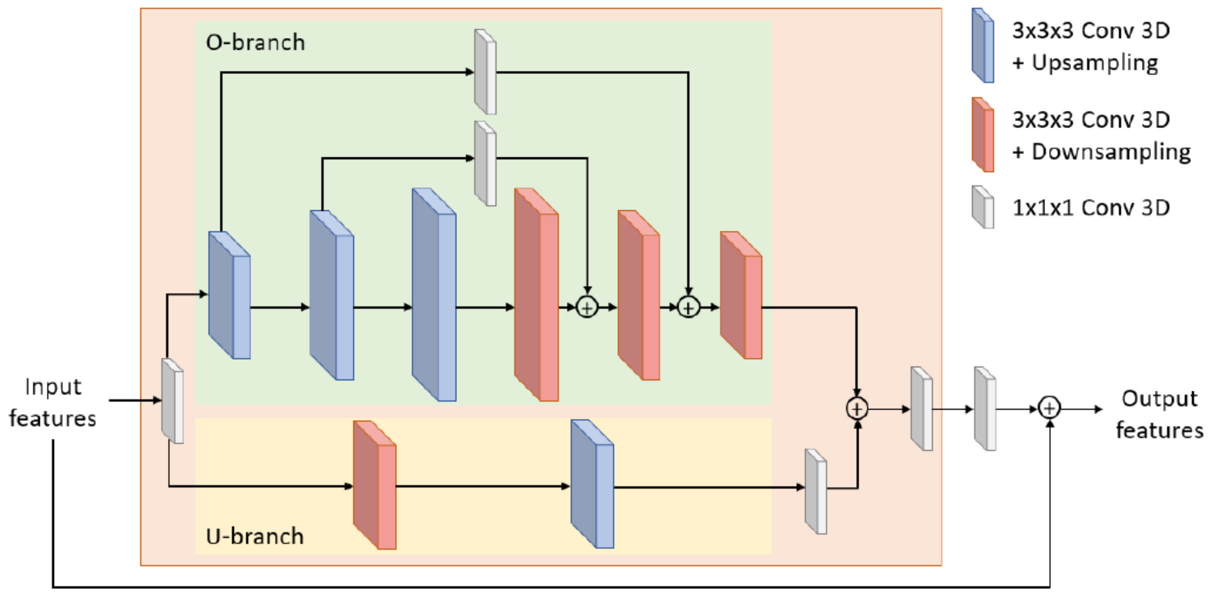
https://ai.plainenglish.io/convolutional-autoencoders-cae-with-tensorflow-97e8d8859cbe.

- OUDefend learns both **undercomplete** representations and **overcomplete** representations (upsample features)



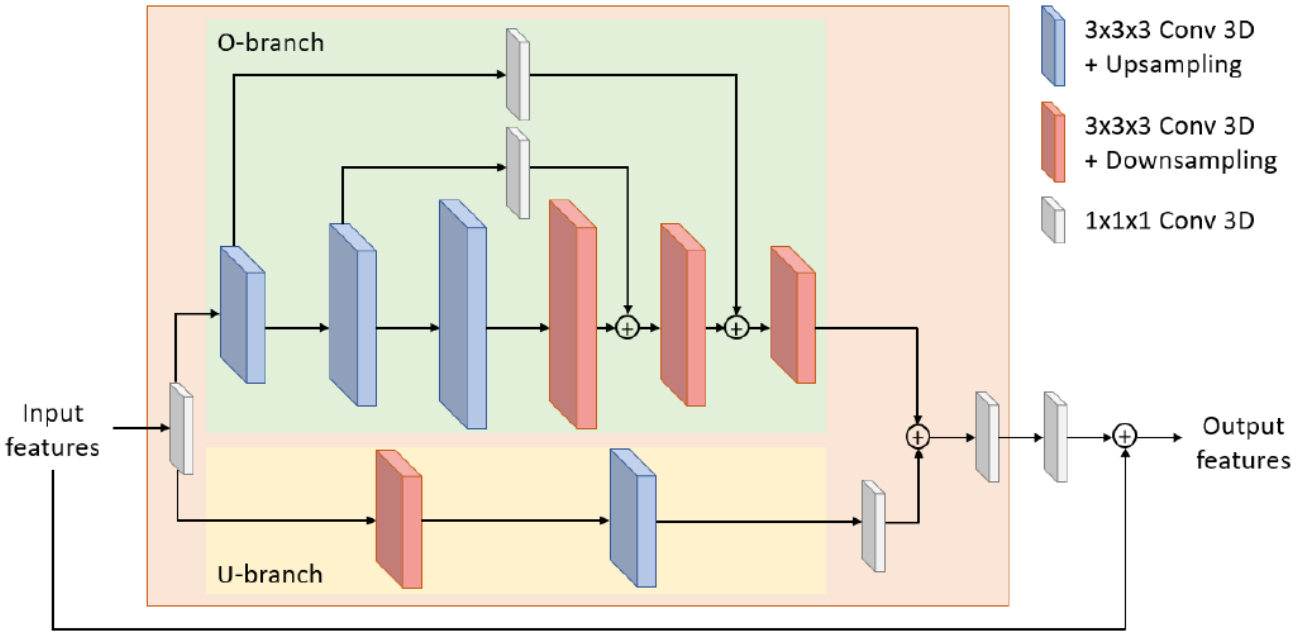
Overcomplete Representations

- **Undercomplete** representations have large receptive fields to collect global information, but they overlook local details.
- **Overcomplete** representations have opposite properties.
- OUDefend balances **global** and **local** features by learning those two representations.



Overcomplete Representations

- Append OUDefend blocks to the target network (after each res block).



layer name	output size	18-layer
conv1	112×112	
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$

Overcomplete Representations

Dataset:
UCF-101

- No Defense: Original network trained on clean data
- Madry [Madry et al. ICLR'18] : Original network trained by adversarial training (AT)
- Xie-A [Xie et al. CVPR'19]: Feature denoising (3D conv) network with AT
- Xie-B [Xie et al. CVPR'19]: Feature denoising (2D conv frame-by-frame) network with AT
- OUDefend: Proposed OUDefend network with AT

Method	#Params	Clean	PGD Linf	PGD L2	MultAV	ROA	AF	SPA	Avg_adv
No Defense	33.0M	76.90	2.56	3.25	7.19	0.16	0.24	4.39	2.97
Madry	33.0M	76.90	33.94	35.05	47.00	41.29	55.99	55.99	48.01
Xie-A	33.7M	70.82	31.48	33.25	42.69	37.59	58.87	49.14	42.17
Xie-B	34.8M	69.47	30.19	32.65	41.87	38.22	58.74	49.14	41.80
OUDefend	33.6M	**77.90**	**34.18**	**35.32**	**47.63**	**42.00**	**56.25**	**56.29**	**49.52**

Part IV: Generalizable Adversarial Defenses

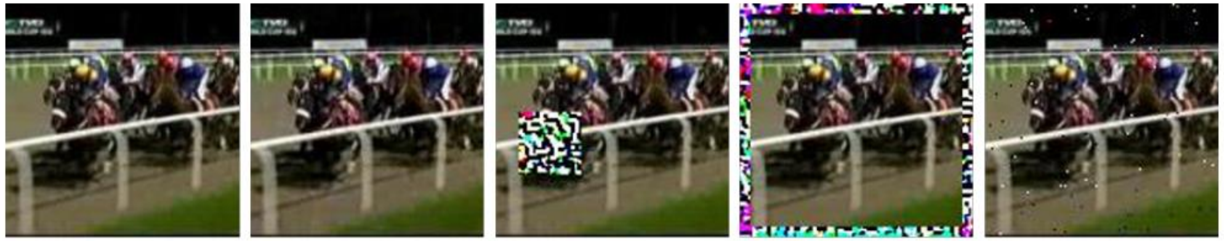
Adversarial Example Types

- PGD:
  Projective gradient descent
  [Madry et al. ICLR’18]

- ROA:
  Rectangular occlusion
  [Wu et al. ICLR’20]

- AF:
  Adversarial Framing
  [Zajac et al. AAAI’19]

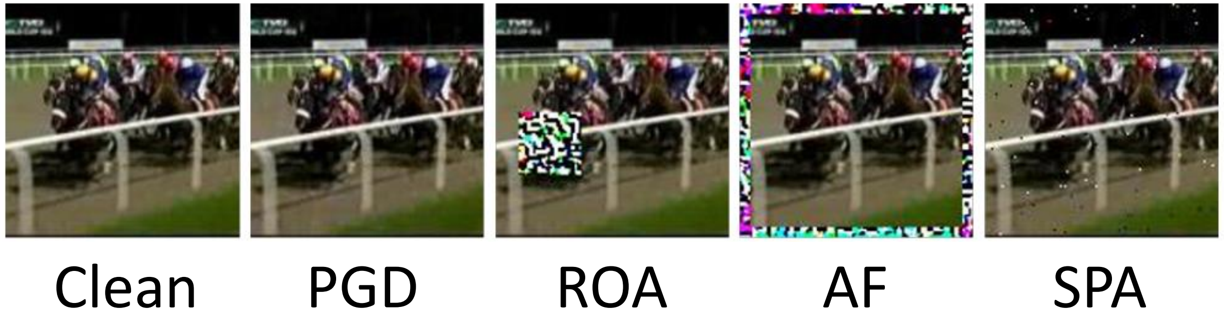
- SPA:
  Salt-and-Pepper noise
  [Lo & Patel, TIP’21]



Clean PGD ROA AF SPA

Adversarial Example Types

- PGD:
  Projective gradient descent
  [Madry et al. ICLR’18]
- ROA:
  Rectangular occlusion
  [Wu et al. ICLR’20]
- AF:
  Adversarial Framing
  [Zajac et al. AAAI’19]
- SPA:
  Salt-and-Pepper noise
  [Lo & Patel, TIP’21]



How to simultaneously defend against multiple types of attacks?

Problem: Multi-perturbation Robustness

- Standard adversarial training has poor **multi-perturbation robustness**.
- Training: δ_{PGD}
- Test: Clean, δ_{PGD}, δ_{ROA}, δ_{AF}, δ_{SPA}

$$\underbrace{\theta^* = \arg\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}} \Big[\underbrace{\max_{\delta \in \mathbb{S}} L(x+\delta, y; \theta)}_{\text{Generate one type of adversarial examples}} \Big]}_{\text{Train model parameters}}$$

Dataset: UCF-101

Model	Clean	PGD	ROA	AF	SPA
No Defense	**89.0**	3.3	0.5	1.6	8.4
AT-PGD	78.6	**49.0**	5.0	0.6	67.1
AT-ROA	82.6	12.5	**69.0**	54.0	17.6
AT-AF	84.6	7.1	3.9	**80.5**	12.2
AT-SPA	83.5	36.9	2.6	0.7	**69.5**

[Lo & Patel, TIP’21]

Generalizable Adversarial Defenses

- Adversarial Training and Robustness for Multiple Perturbations (Average and Max Adversarial Training) [Tramèr & Boneh, NeurIPS'19]

- Defending Against Multiple and Unforeseen Adversarial Videos (Multiple BatchNorm Structure) [Lo & Patel, TIP'21]

- Perceptual Adversarial Robustness: Defense Against Unseen Threat Models (Perceptual Adversarial Training) [Laidlaw et al. ICLR'21]

Average Adversarial Training

- **Average** adversarial training is better, but not enough.
- Training: Clean, δ_{PGD}, δ_{ROA}, δ_{AF}, δ_{SPA}
- Test: Clean, δ_{PGD}, δ_{ROA}, δ_{AF}, δ_{SPA}

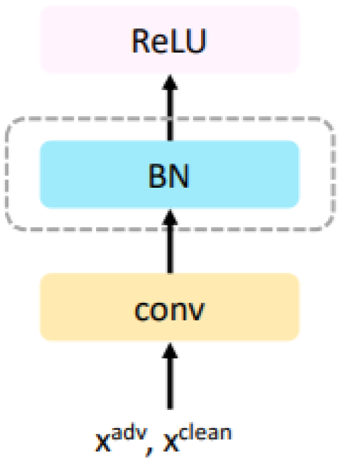
$$\underbrace{\theta^* = \arg\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}} \left[\underbrace{\sum_{i=1}^{N} \max_{\delta_i \in \mathbb{S}_i} L(x+\delta_i, y; \theta)}_{\text{Generate multiple types of adversarial examples}} \right]}_{\text{Train model parameters}}$$

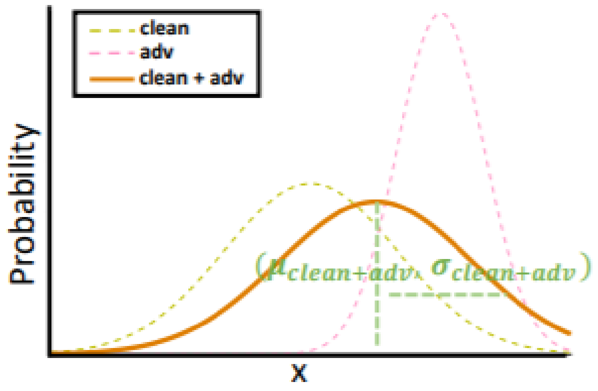
Generate **multiple types** of adversarial examples

Train model parameters

Average Adversarial Training

- Why average adversarial training is **not** an ideal strategy?
- Example: **Clean** vs. **PGD**.
- Clean and PGD have **distinct** data distributions.
- The statistics estimation at **BN** may be confused when facing a mixture distribution.

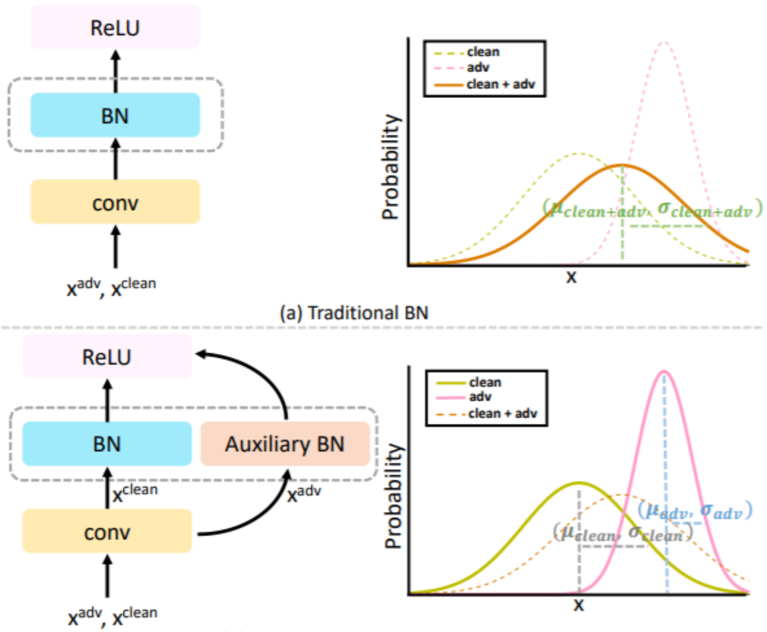




[Xie et al. CVPR’20]

Multiple BatchNorm Structure

- Example: **Clean** vs. **PGD**.
- An **auxiliary BN** guarantees that data from different distributions are normalized separately.



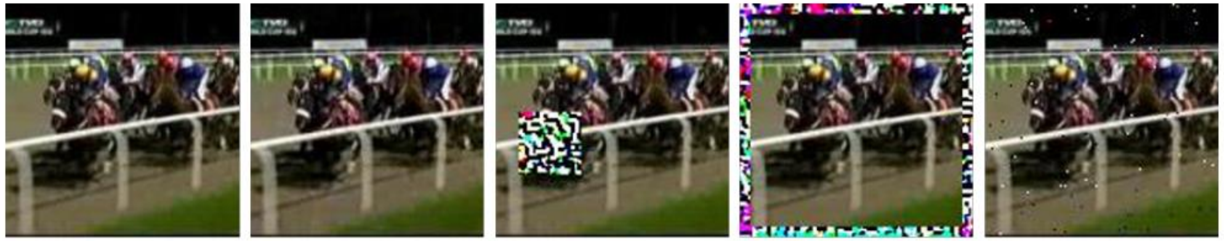
(a) Traditional BN

(b) Proposed Auxiliary BN Design

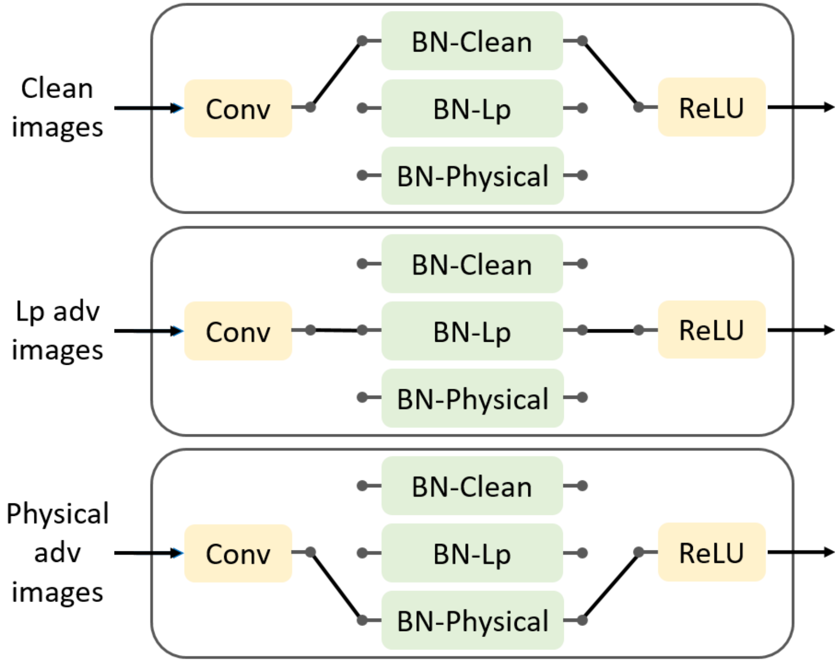
[Xie et al. CVPR'20]

Multiple BatchNorm Structure

- What about **multiple** attack types?
- Example: Clean, PGD, ROA, AF, SPA
- Assumption: Different attack types have **distinct** data distributions.
- **Lp-norm attacks**: PGD, SPA
- **Physically realizable attacks**: ROA, AF
- Assumption: Similar attack types have **similar** data distributions.

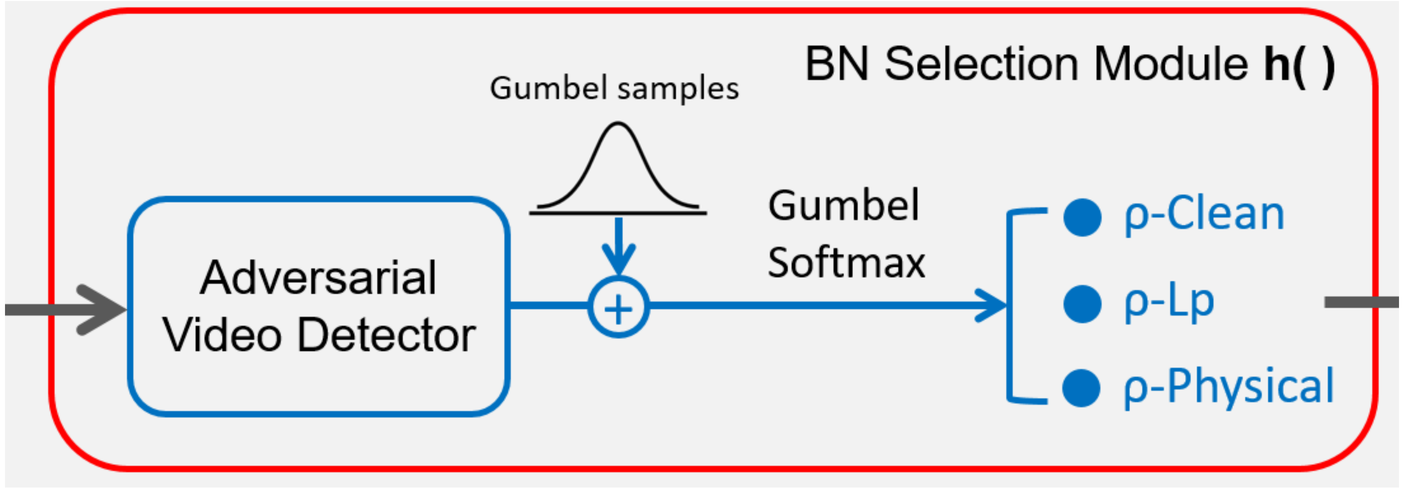


Clean PGD ROA AF SPA



Multiple BatchNorm Structure

- At inference time, the input data have to pass through the corresponding BN branch **automatically**.
- The **adversarial video detector** is achieved by a video classifier.
- **Gumbel-Softmax** function [Jang et al. ICLR'17] is a **differentiable** approximation of the ***argmax*** operation (vanilla Softmax also works).



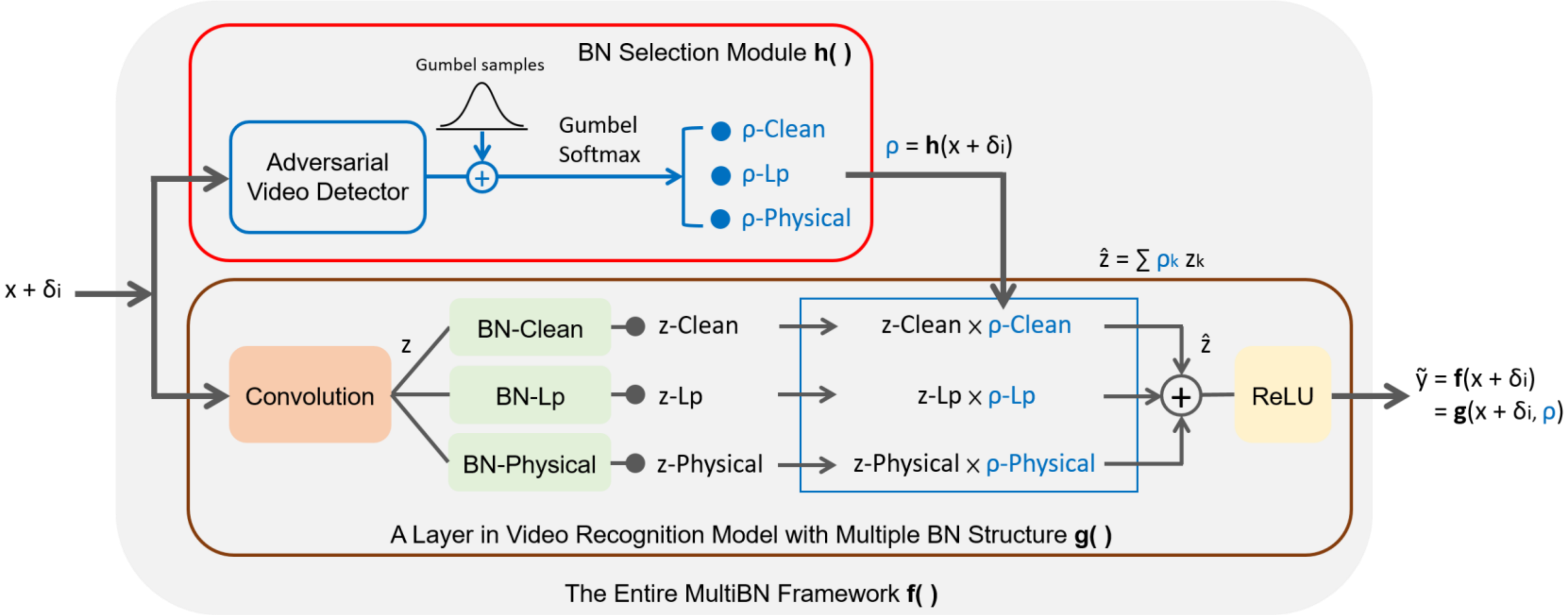
Multiple BatchNorm Structure

- End-to-end pipeline:

$$\begin{aligned}\tilde{y} &= f(x+\delta_i;\, \theta^c, \theta^b, \theta^{det}) \\ &= g(x+\delta_i, h(x+\delta_i;\, \theta^{det});\, \theta^c, \theta^b)\end{aligned}$$

- End-to-end training:

$$\theta^* = \arg\min_{\theta} \mathop{\mathbb{E}}_{(x,y)\sim\mathbb{D}} \Big[L(x, y;\, \theta) + \lambda \cdot L(x, y^{det};\, \theta^{det}) + \sum_{i=1}^{N} \big(\max_{\delta_i \in \mathbb{S}_i} L(x+\delta_i, y;\, \theta) + \lambda \cdot L(x+\delta_i, y^{det};\, \theta^{det}) \big) \Big]$$



Multiple BatchNorm Structure

- Dataset: UCF-101
- Model: 3D ResNeXt-101

Model	Clean	PGD	ROA	AF	SPA	Mean	Union
No Defense	**89.0**	3.3	0.5	1.6	8.4	20.6	0.0
TRADE [19] (ICML'19)	82.3	29.0	5.7	3.3	42.2	32.5	1.9
AVG [26] (NeurIPS'19)	68.9	38.1	51.4	18.5	49.6	45.3	17.3
MAX [26] (NeurIPS'19)	72.8	32.5	31.0	5.8	49.4	38.3	5.5
MSD [27] (ICML'20)	70.2	43.2	1.7	1.6	**56.0**	34.6	0.7
MultiBN (ours)	74.2	**44.6**	**58.6**	**44.3**	53.7	**55.1**	**34.8**

Perceptual Adversarial Training

- **Perceptual** adversarial attack using the **LPIPS** (Learned Perceptual Image Patch Similarity) [Zhang et al. CVPR'18] distance.

- Recall: max Loss(f(x+δ), y; θ)
- Perturbation budget ‖δ‖
  - Constrain the magnitude of perturbation, e.g., Lp-norm.
  - Lp-norm: $\| \boldsymbol{x}_{adv} - \boldsymbol{x} \|_p < \epsilon$
  - LPIPS: $\| \varphi(\boldsymbol{x}_{adv}) - \varphi(\boldsymbol{x}) \|_2 < \epsilon,$ φ: *neural network*

Perceptual Adversarial Training

- **Perceptual** adversarial training: Do adversarial training on perceptual adversarial examples.

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}} \left[\max_{\delta \in \mathbb{S}} L(\boxed{x+\delta}, y; \theta) \right]$$

Perceptual adversarial examples

Perceptual Adversarial Training

- Dataset: CIFAR-10
- Model: ResNet-50

Training	Union	Unseen mean	Narrow threat models					NPTM	
			Clean	L_∞	L_2	StAdv	ReColor	PPGD	LPA
Normal	0.0	0.1	94.8	0.0	0.0	0.0	0.4	0.0	0.0
AT L_∞	1.0	19.6	86.8	49.0	19.2	4.8	54.5	1.6	0.0
TRADES L_∞	4.6	23.3	84.9	52.5	23.3	9.2	60.6	2.0	0.0
AT L_2	4.0	25.3	85.0	39.5	47.8	7.8	53.5	6.3	0.3
AT StAdv	0.0	1.4	86.2	0.1	0.2	53.9	5.1	0.0	0.0
AT ReColorAdv	0.0	3.1	93.4	8.5	3.9	0.0	65.0	0.1	0.0
AT all (random)	0.7	—	85.2	22.0	23.4	1.2	46.9	1.8	0.1
AT all (average)	14.7	—	86.8	39.9	39.6	20.3	64.8	10.6	1.1
AT all (maximum)	21.4	—	84.0	25.7	30.5	40.0	63.8	8.6	1.1
Manifold reg.	21.2	36.2	72.1	36.8	43.4	28.4	63.1	8.7	9.1
PAT-self	21.9	45.6	82.4	30.2	34.9	46.4	71.0	13.1	2.1
PAT-AlexNet	**27.8**	**48.5**	71.6	28.7	33.3	64.5	67.5	**26.6**	**9.8**

Part V: Adversarial Attacks in Videos

Image-based Attacks in Videos

- Video is a stack of consecutive images.
- A naïve way to generate adversarial videos:
  Use image-based method directly.

$$\boldsymbol{x}_{adv} = \boldsymbol{x} + \epsilon \cdot sign(\nabla_{\boldsymbol{x}} L(\boldsymbol{x}, y))$$

$$Image\text{:}\ \boldsymbol{x} \in R^{C \times H \times W}$$

$$Video\text{:}\ \boldsymbol{x} \in R^{\boldsymbol{F} \times C \times H \times W}$$

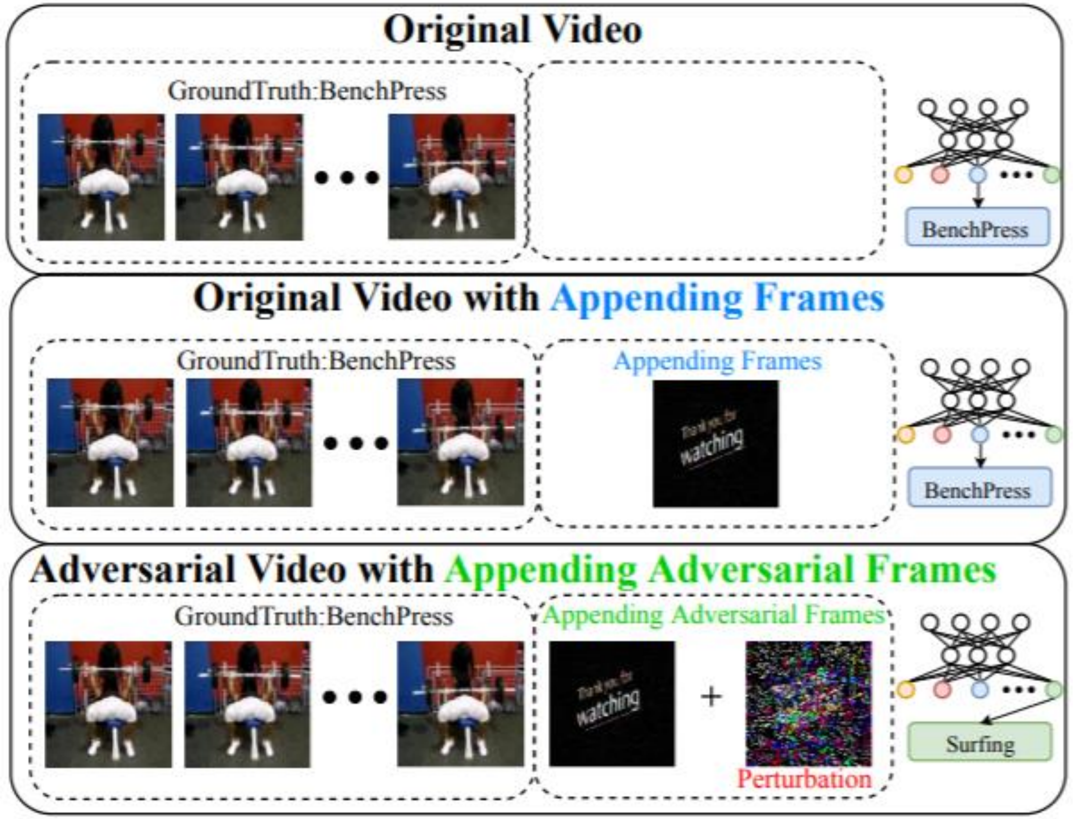
Video-specific Attacks

- Use video’s unique properties (mostly temporal information) to generate adversarial videos.

- Video has higher dimensionality, so the search space of adversary is larger -> more possible types of adversarial examples

Appending Adversarial Frame

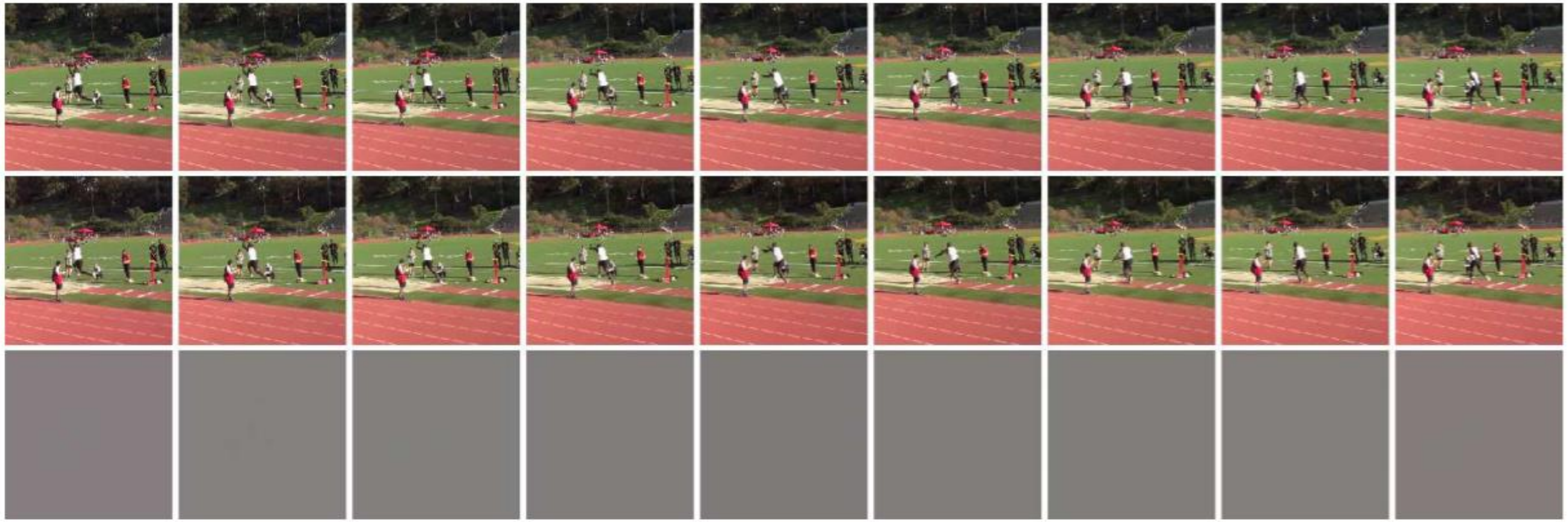
- Append an additional video frame at the end of the video, then perturb this appended frame only.



[Chen et al. WACV'21]

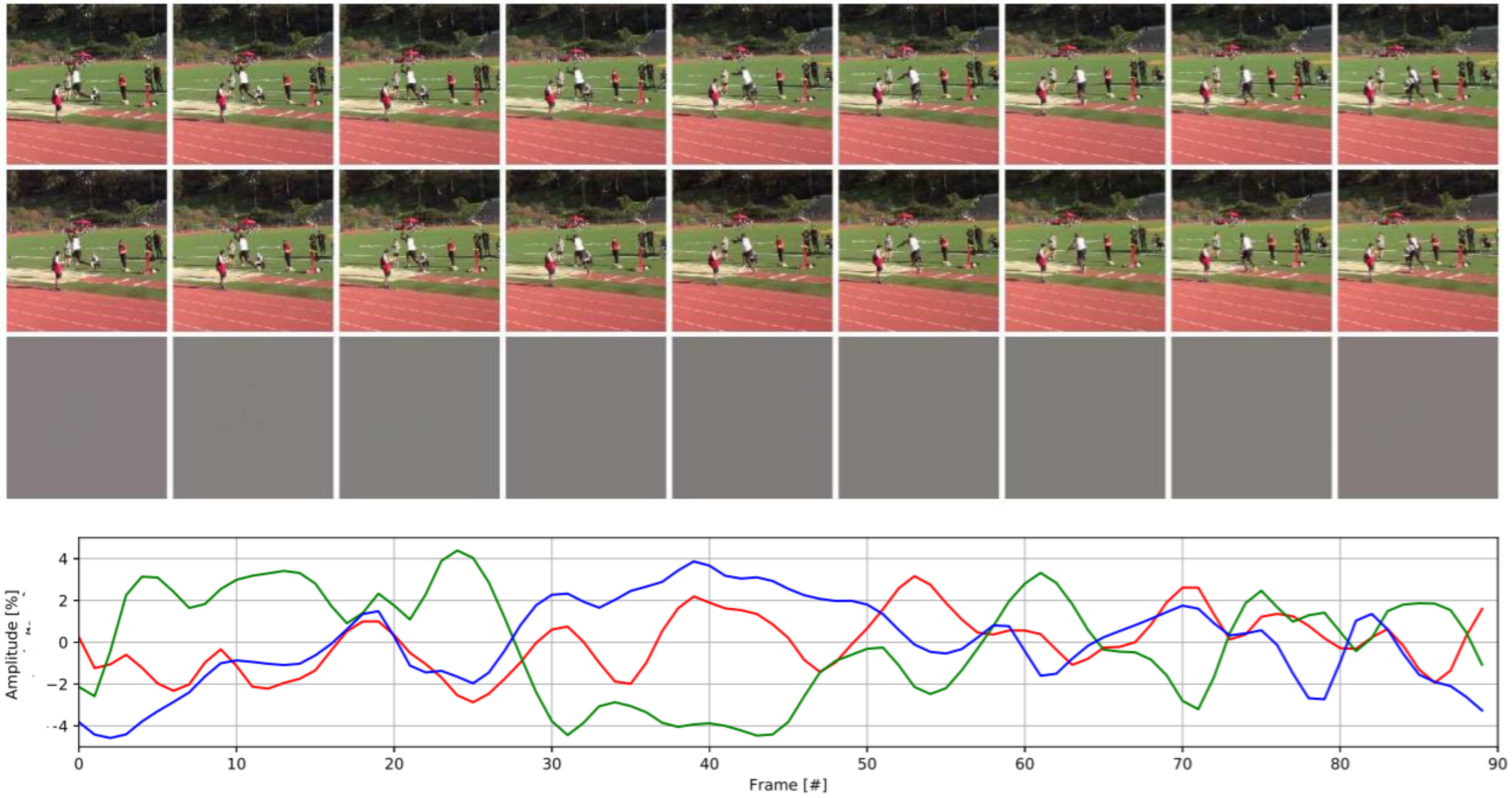
Adversarial Flickering Attack

- Spatial patternless temporal perturbation, i.e., the perturbation is a constant offset applied to the entire frame.
- Undetectable by image adversarial attack detector.



[Pony et al. CVPR'21]

Adversarial Flickering Attack



[Pony et al. CVPR’21]

Part VI: More Research Problems about Adversarial Robustness

More Attack/Defense Settings

- **Evasion attack (inference-time attack)**
- Poisoning attack (training-time attack)

- **Empirical defense (more practical)**
- Certified defense (more provable)

More Applications

- Open-set recognition [Shao et al. ECCV’20]
- Novelty detection [Lo et al. arXiv’21]
- Domain adaptation [Lo & Patel, arXiv’22]
- Deep ranking [Zhou et al. ICCV’21]
- Deep metric learning [Zhou & Patel, CVPR’22]

- Audio [Joshi et al. TIFS’21]
- Text [Lei et al. SysML’19]