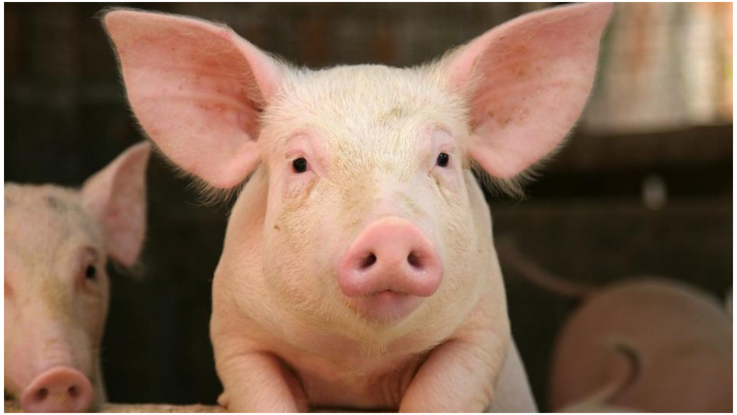
מכון ויצמן למדע
WEIZMANN INSTITUTE OF SCIENCE

12/12/2021

Introduction To Adversarial Examples

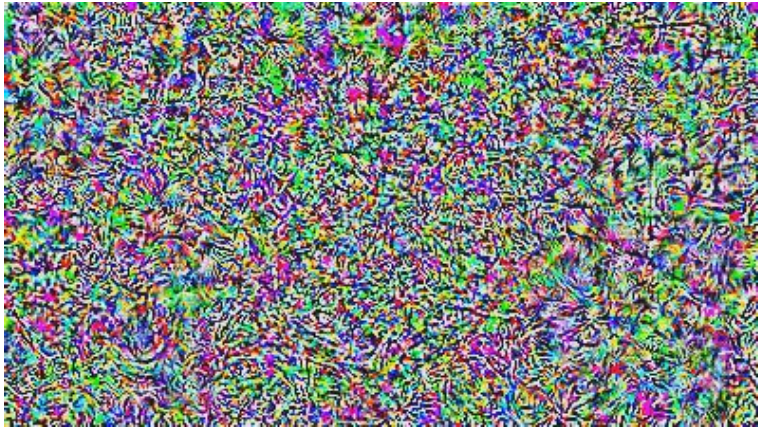
Niv Haim
Weizmann Institute

DL4CV Course Winter 2023 (20224182)



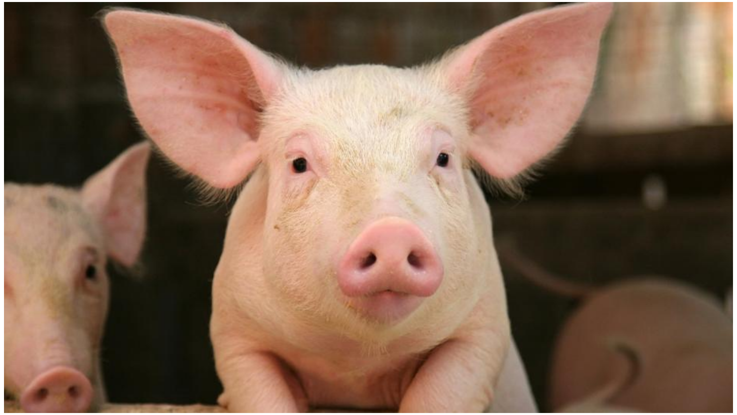
98.6% pig

+



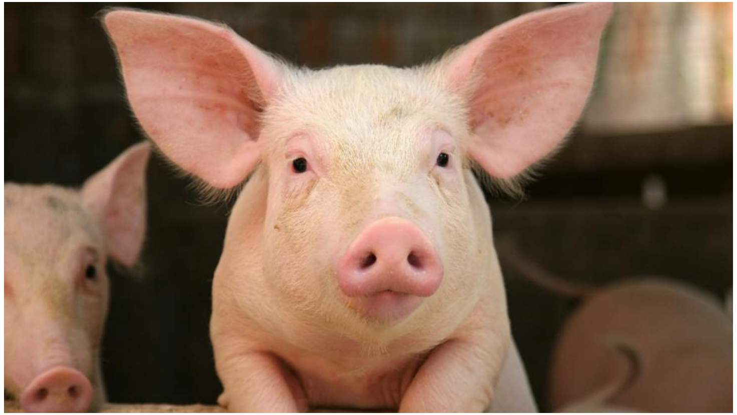
x 0.02

=



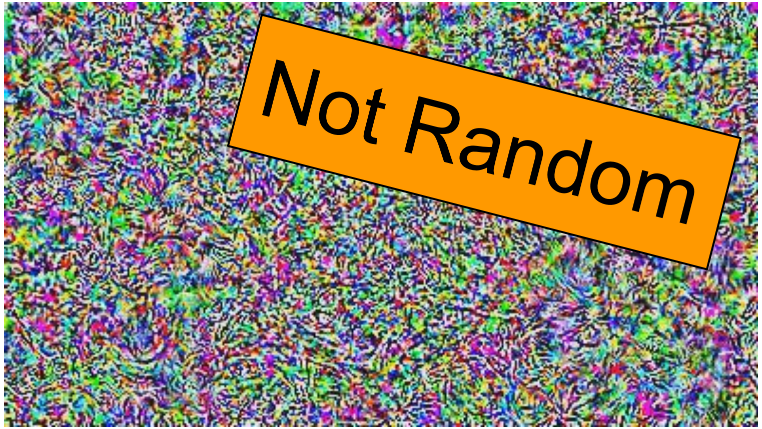
99.0% airliner

(0.000000000000000
00000000000005% pig...)



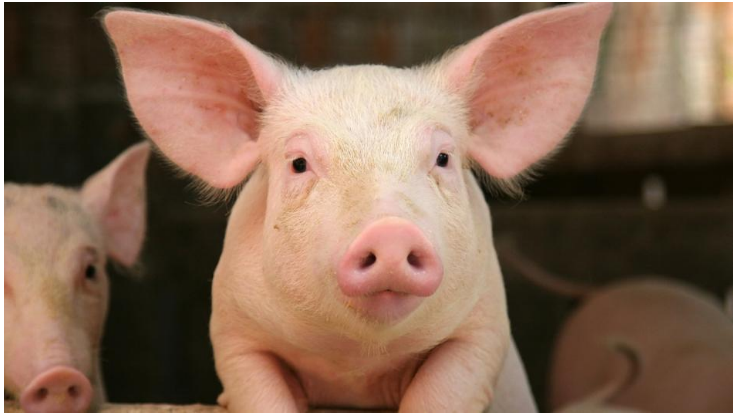
98.6% pig

+



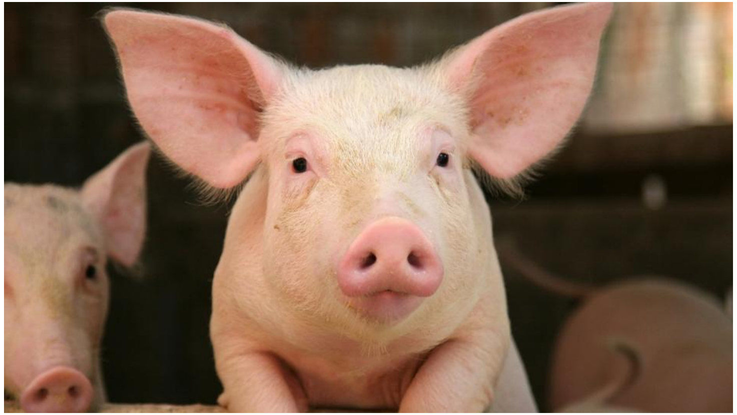
x 0.02

=



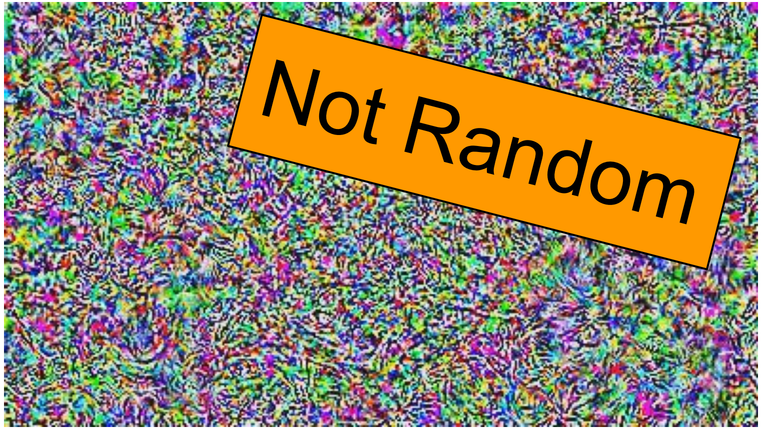
99.0% airliner

(0.000000000000000
00000000000000005% pig...)



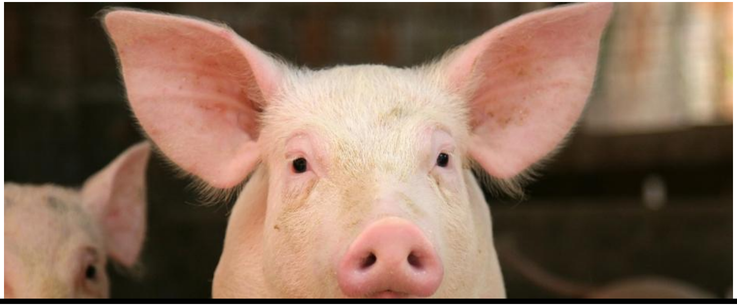
98.6% pig

+



x 0.02

=



99.0% airliner

(0.00000000000000000
0000000000000005% pig...)

Biggio et al. 2013, “Evasion attacks against machine learning at test time”
Szegedy et al. 2014, “Intriguing properties of neural networks”
Goodfellow et al. 2015, “Explaining and Harnessing Adversarial Examples”

What is an Adversarial Example?

What is an Adversarial Example?

- Originally coined by Szegedy et al., 2013:

“we find that applying an imperceptible non-random perturbation to a test image, it is possible to arbitrarily change the network’s prediction.

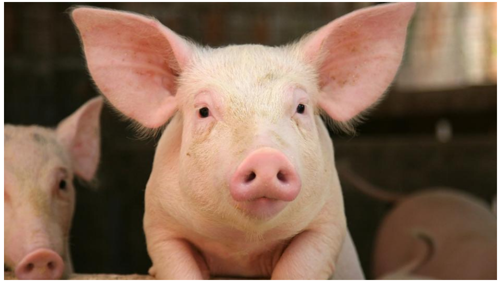
*… we term the so perturbed examples ‘**adversarial examples**’”*

What is an Adversarial Example?

- Originally coined by Szegedy et al., 2013:

“we find that applying an imperceptible non-random perturbation to a test image, it is possible to arbitrarily change the network’s prediction.

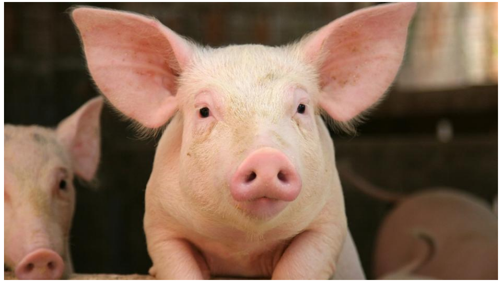
*… we term the so perturbed examples ‘**adversarial examples**’”*



98.6% pig

+

x 0.02 =



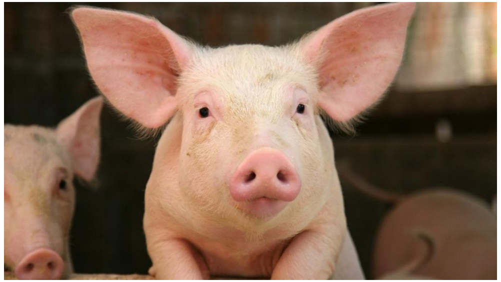
99.0% airliner

What is an Adversarial Example?

- Originally coined by Szegedy et al., 2013:

“we find that applying an imperceptible non-random perturbation to a test image, it is possible to arbitrarily change the network’s prediction.

*… we term the so perturbed examples ‘**adversarial examples**’”*

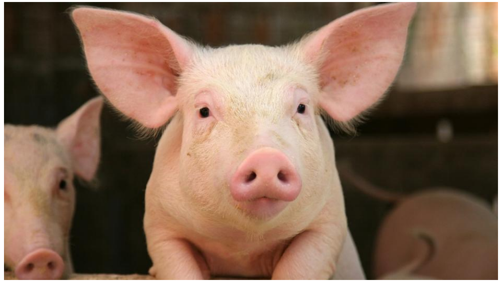


98.6% pig

+

Not Random
Small

x 0.02 =



99.0% airliner

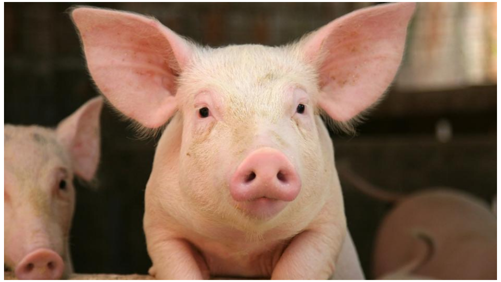
What is an Adversarial Example?

Perturbation Attack

- Originally coined by Szegedy et al., 2013:

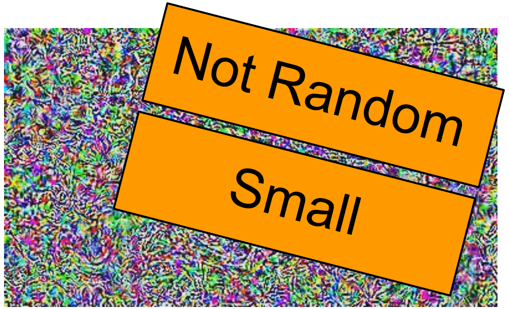
“we find that applying an imperceptible non-random perturbation to a test image, it is possible to arbitrarily change the network’s prediction.

*… we term the so perturbed examples ‘**adversarial examples**’”*

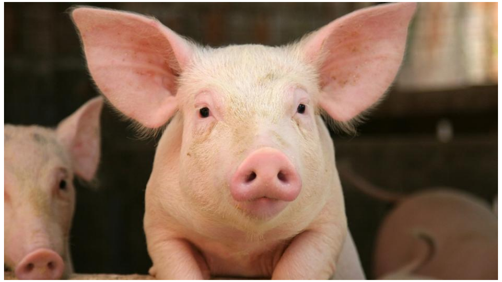


98.6% pig

+



x 0.02 =



99.0% airliner

Outline

Today we will:

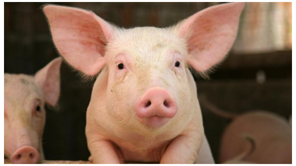
- See Adversarial Example

Outline

Today we will:

- See Adversarial Example
- Discuss what they are
- Learn how to generate them
- Learn how to defend against them

airliner

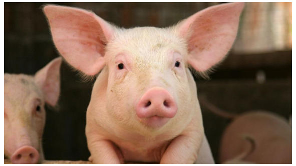


Outline

Today we will:

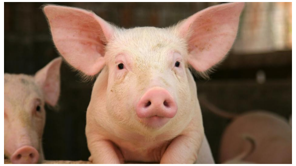
- See Adversarial Example
- Discuss what they are
- Learn how to generate them
- Learn how to (maybe) defend against them
- Learn about properties and advantages

airliner



Outline

Today we will:



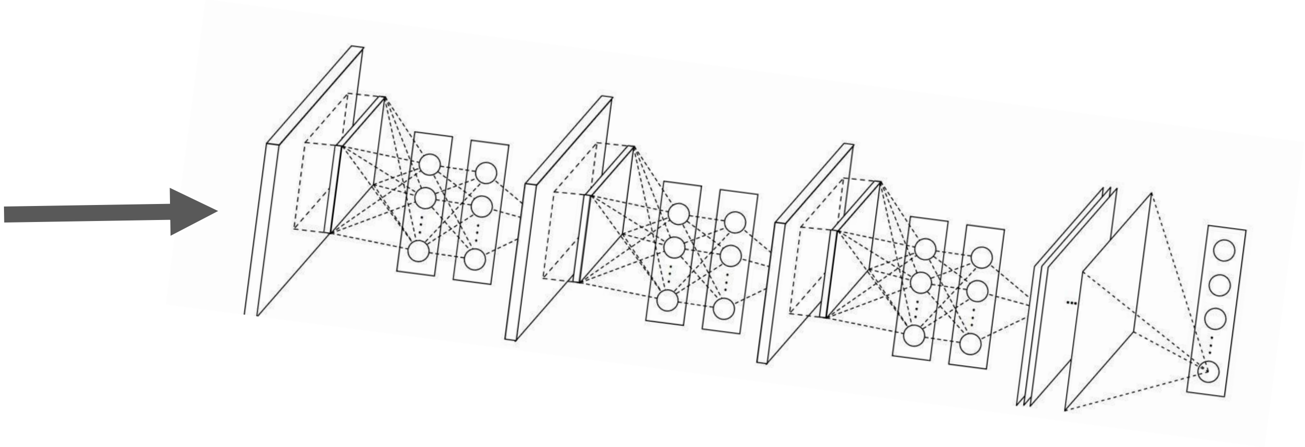
- See Adversarial Example
- Discuss what they are
- Learn how to generate them
- Learn how to (maybe) defend against them
- Learn about properties and advantages

Brief recap on training neural networks

Image by Simon from Pixabay

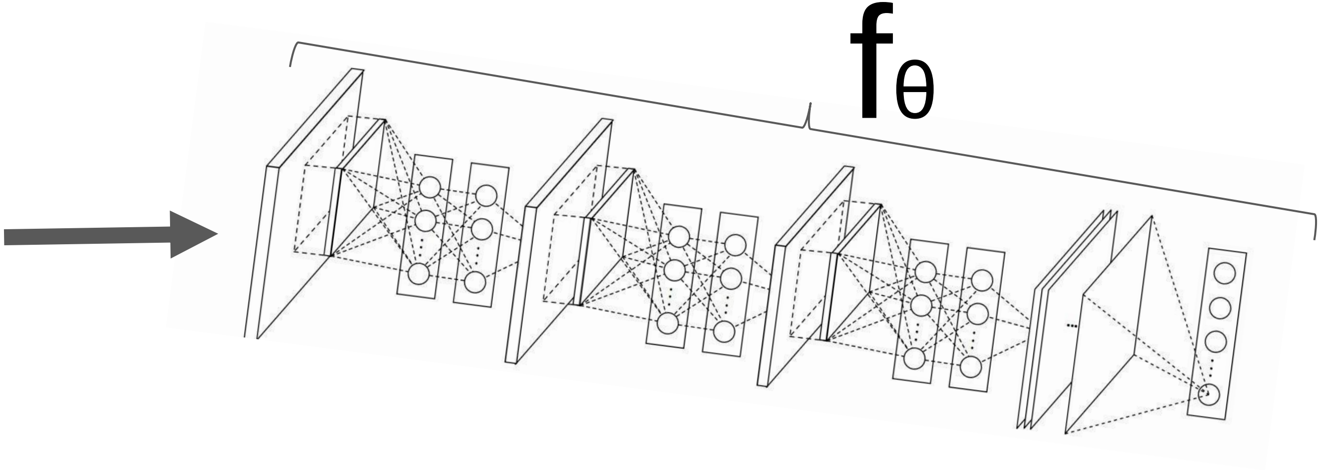
Brief recap on training neural networks

Image by Simon from Pixabay



Brief recap on training neural networks

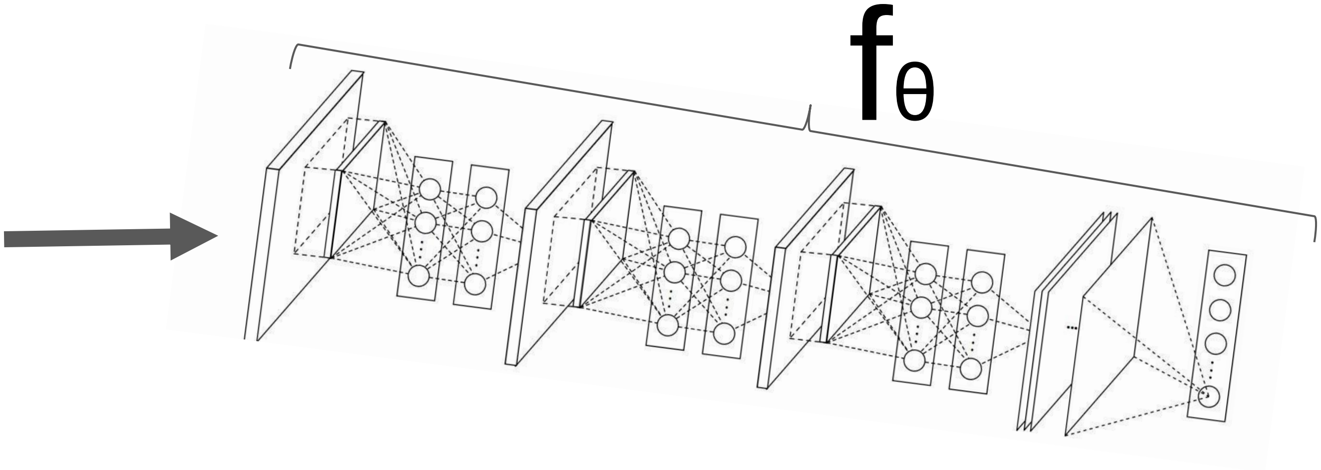
Image by Simon from Pixabay



f_θ

Brief recap on training neural networks

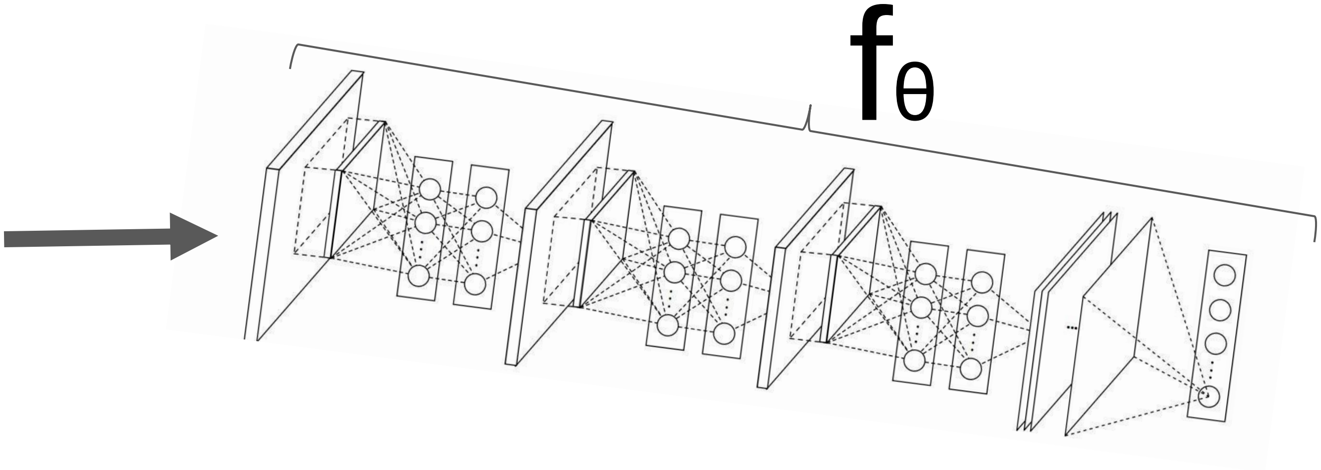
Image by Simon from Pixabay



$$L(f_\theta(x),y)$$

Brief recap on training neural networks

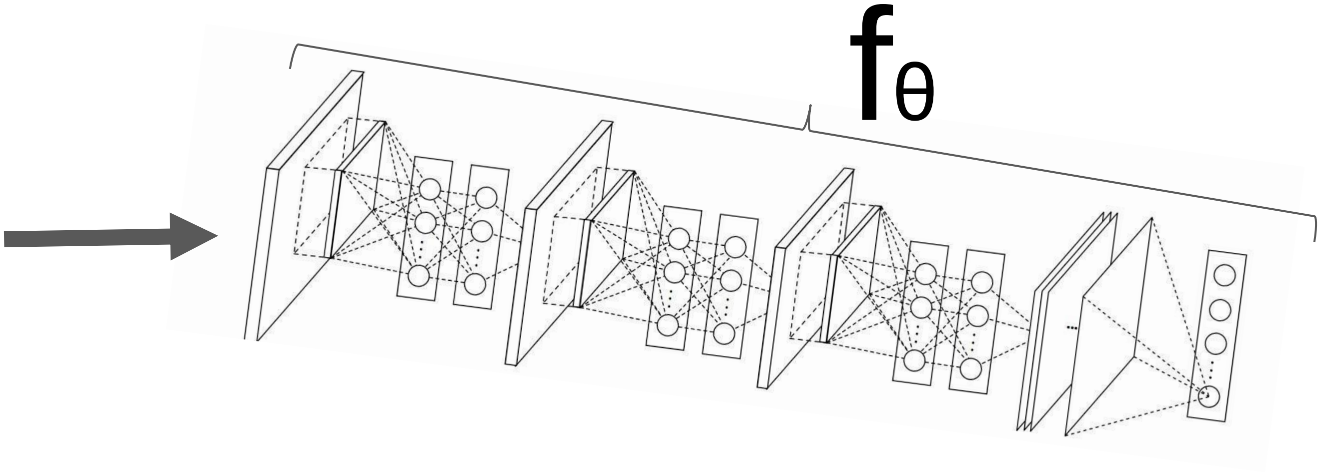
Image by Simon from Pixabay



$$L(\boxed{f_\theta(x)},y)$$

Brief recap on training neural networks

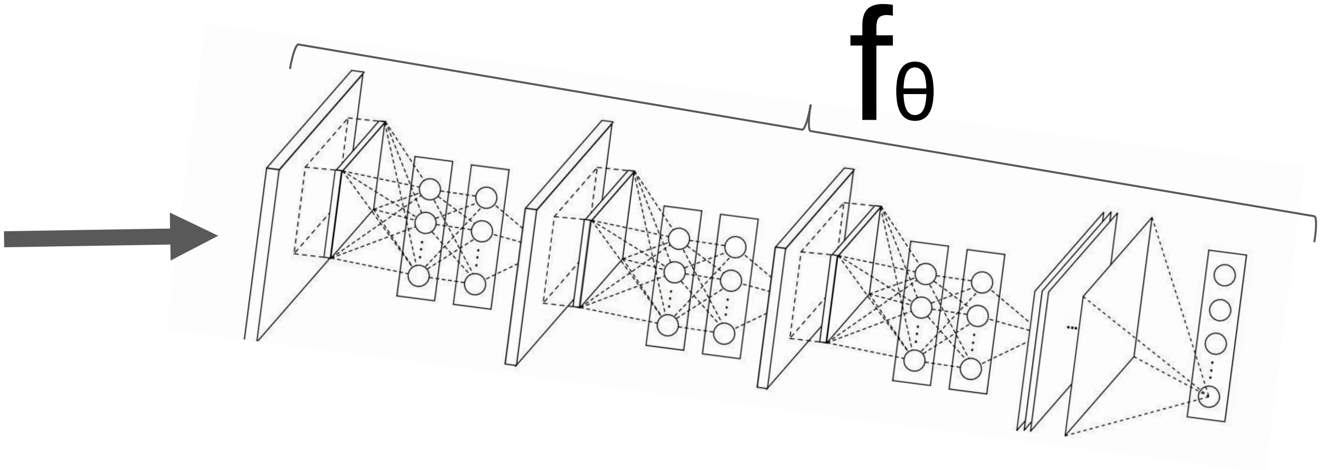
Image by Simon from Pixabay



$$L(f_\theta(x),\boxed{y})$$

Brief recap on training neural networks

Image by Simon from Pixabay

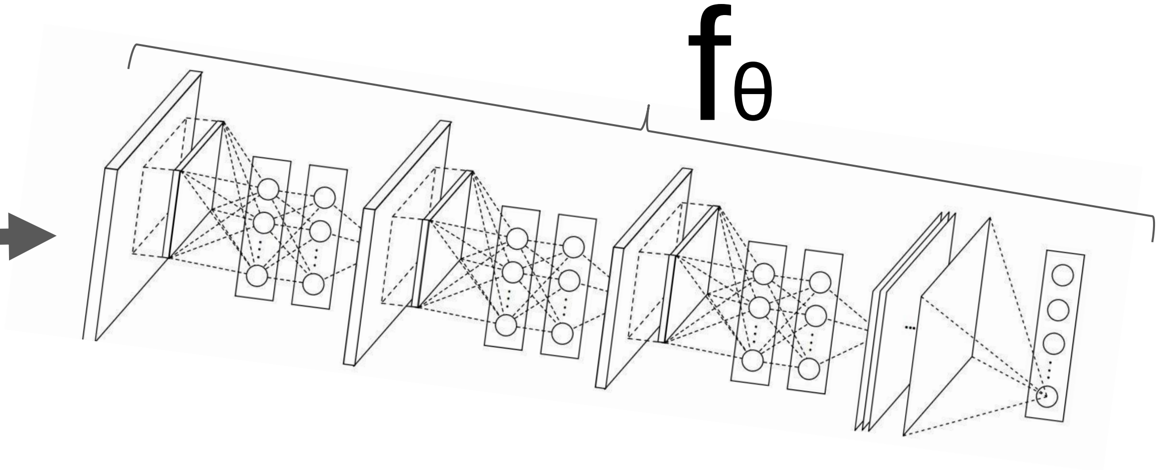


$$L(f_\theta(x),y)$$

purpose of loss:
How “well” we classify

Brief recap on training neural networks

Image by Simon from Pixabay



f_θ

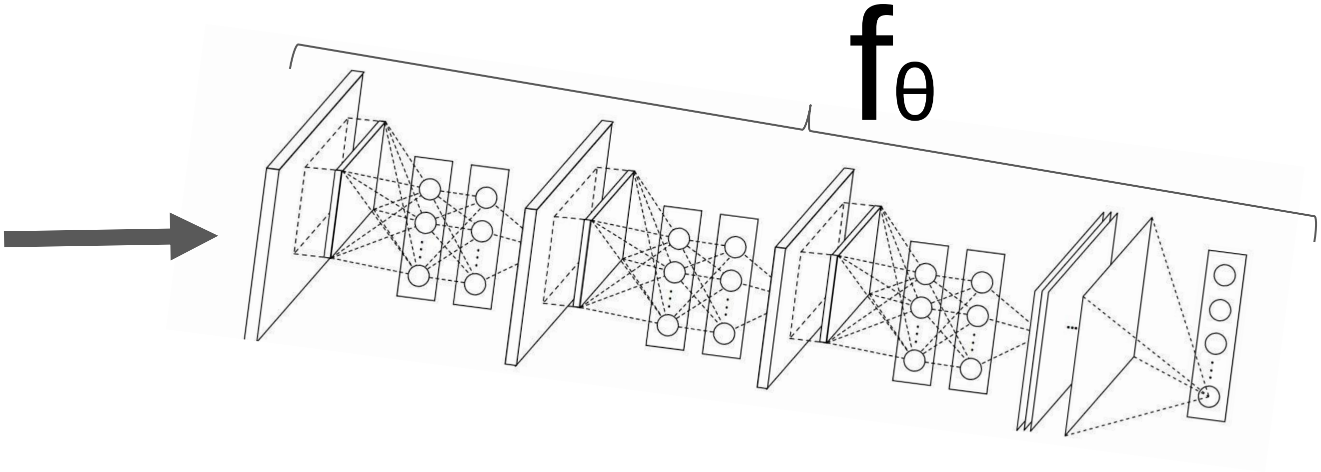
most common loss – CrossEntropy:

$$L(f_\theta(x),y)$$

$$L\left(f_\theta(x), y\right) = -\log\left(\frac{e^{f_\theta(x)_y}}{\sum_j e^{f_\theta(x)_j}}\right)$$

Brief recap on training neural networks

Image by Simon from Pixabay



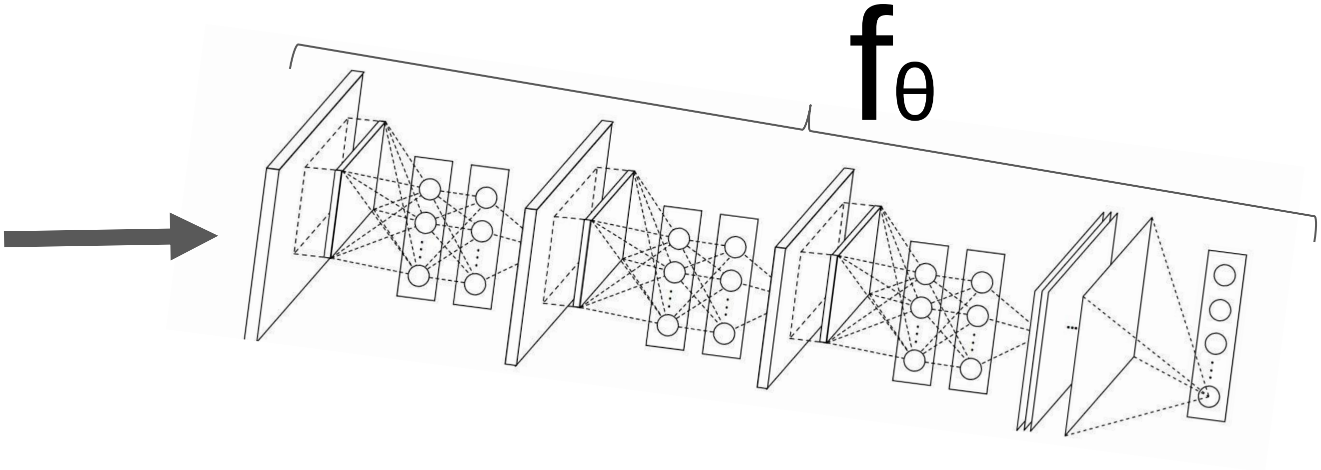
most common loss – CrossEntropy:

$$L(f_\theta(x),y)$$

$$L\left(\boxed{f_\theta(x)}, y\right) = -\log\left(\frac{e^{\boxed{f_\theta(x)}_y}}{\sum_j e^{\boxed{f_\theta(x)}_j}}\right)$$

Brief recap on training neural networks

Image by Simon from Pixabay



$L(f_\theta(x),y)$

most common loss – CrossEntropy:

$$L\left(f_\theta(x)\boxed{,y}\right) = -\log\left(\frac{e^{f_\theta(x)_{\boxed{y}}}}{\sum_j e^{f_\theta(x)_j}}\right)$$

Brief recap on training neural networks

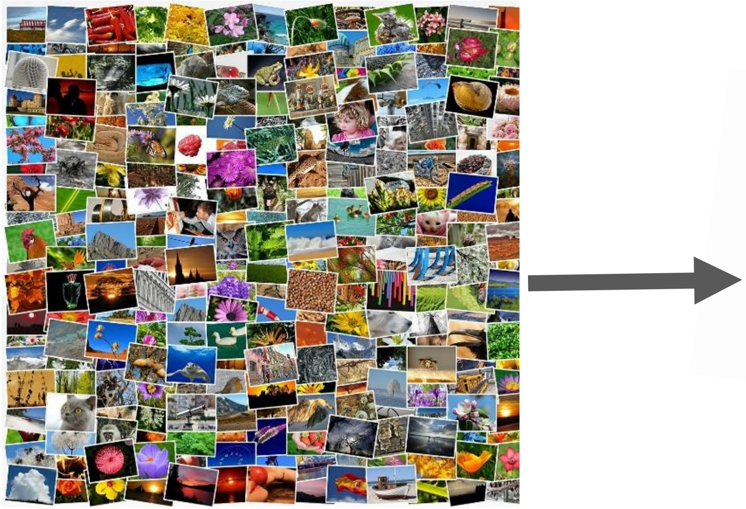
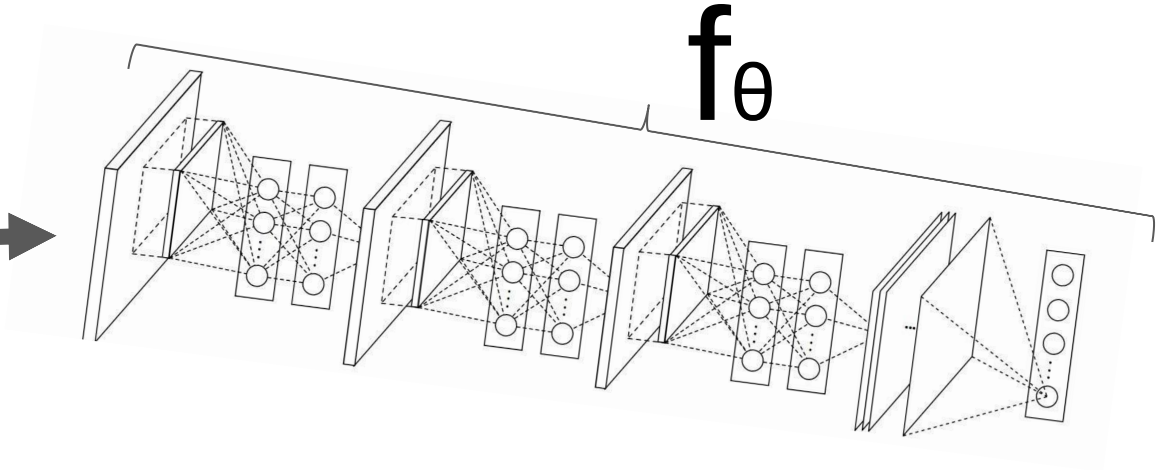


Image by Simon from Pixabay

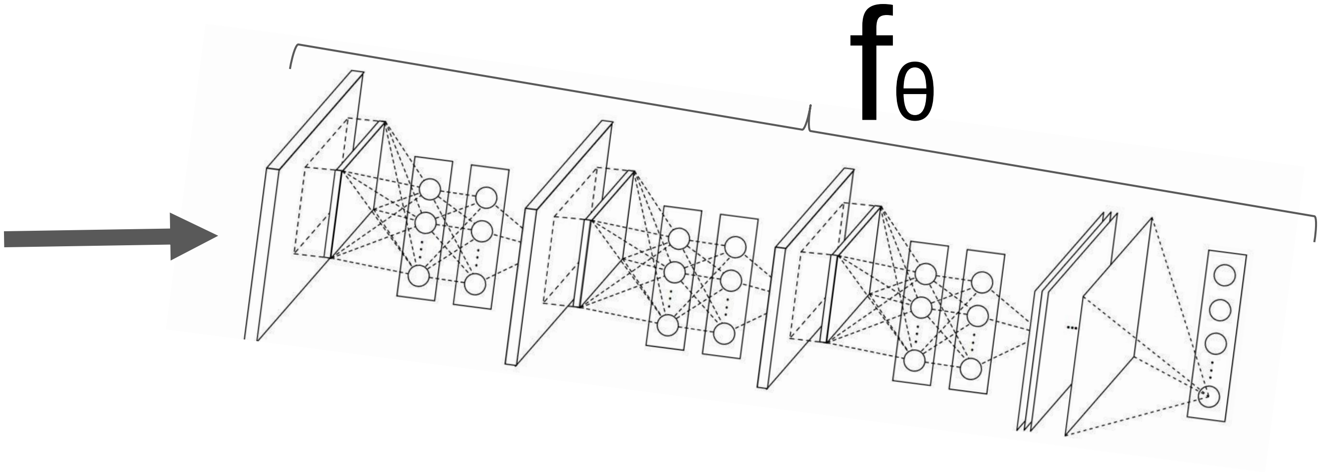


minimize loss:

$$L(f_\theta(x),y)$$

Brief recap on training neural networks

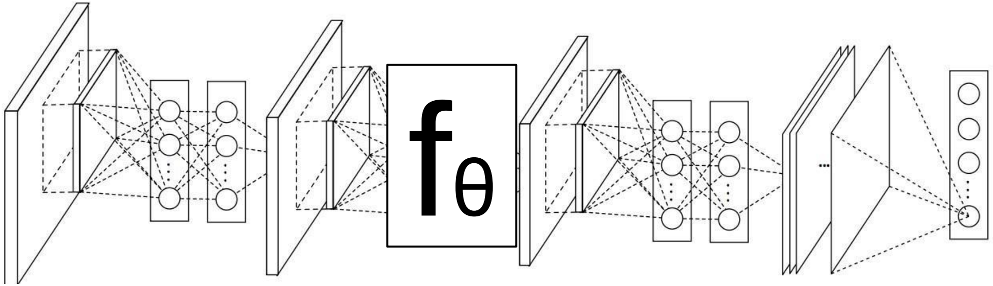
Image by Simon from Pixabay



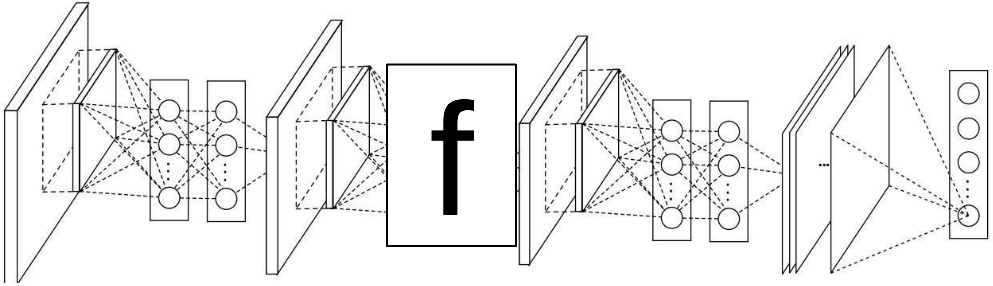
minimize loss:

$$L(f_\theta(x),y) \longrightarrow -\nabla_\theta L$$

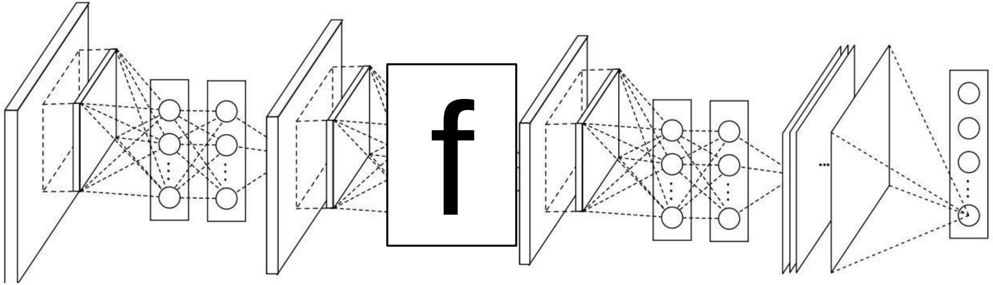
Generating an Adversarial Example



Generating an Adversarial Example

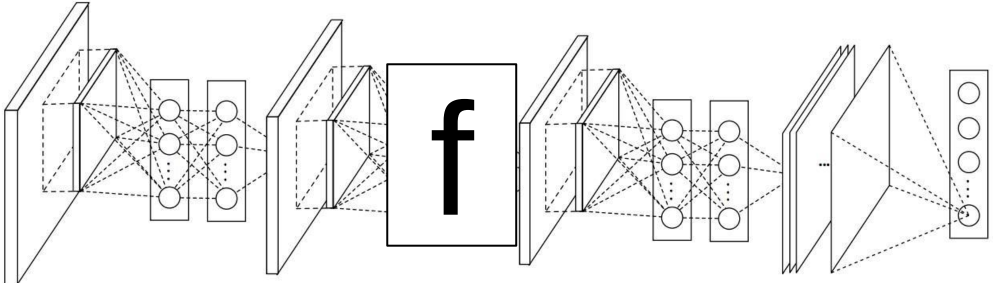


Generating an Adversarial Example



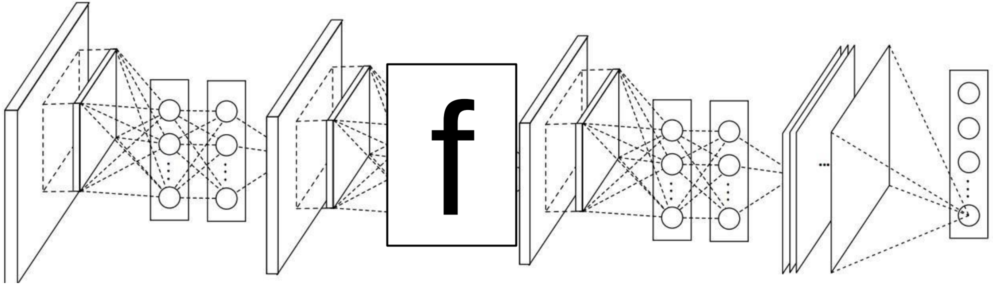
Generating an Adversarial Example

89.7% pig



Generating an Adversarial Example

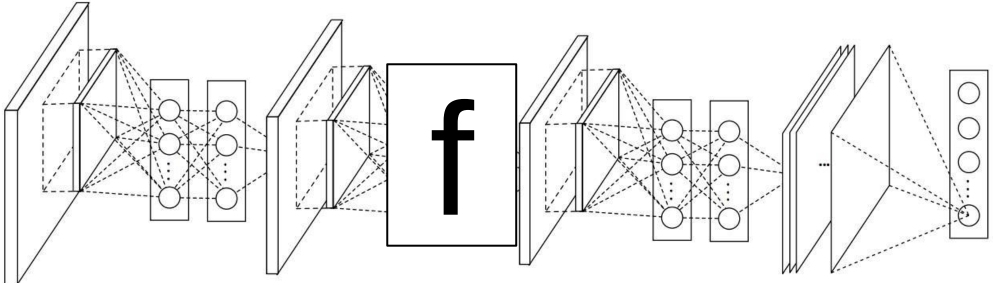
89.7% pig



want to fool classifier

Generating an Adversarial Example

89.7% pig

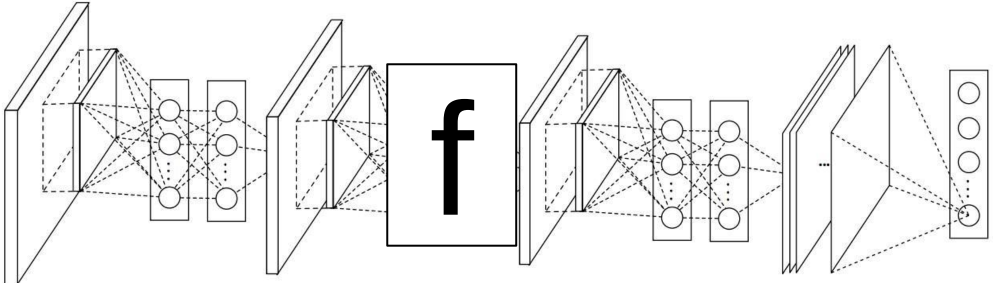


want to fool classifier
by changing δ

$$f(x\boxed{+\delta})\neq y$$

Generating an Adversarial Example

89.7% pig

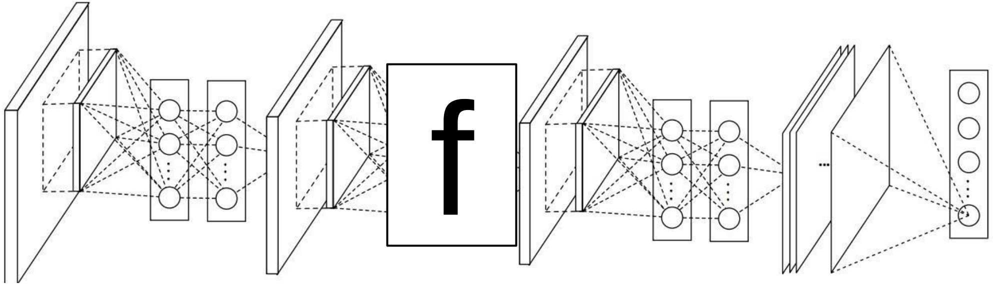


want to fool classifier → d measures “badness”
by changing δ

$$d(f(x+\delta),y)$$

Generating an Adversarial Example

89.7% pig

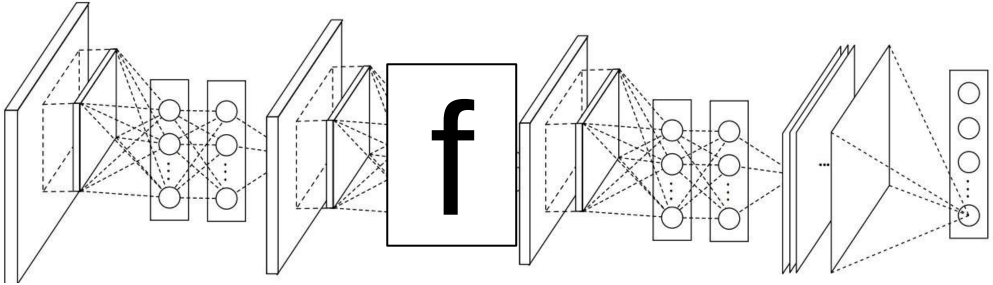


want to fool classifier → used $\boxed{L}$ to maximize “wellness”

$$\boxed{L}(f(x+\delta),y)$$

Generating an Adversarial Example

89.7% pig

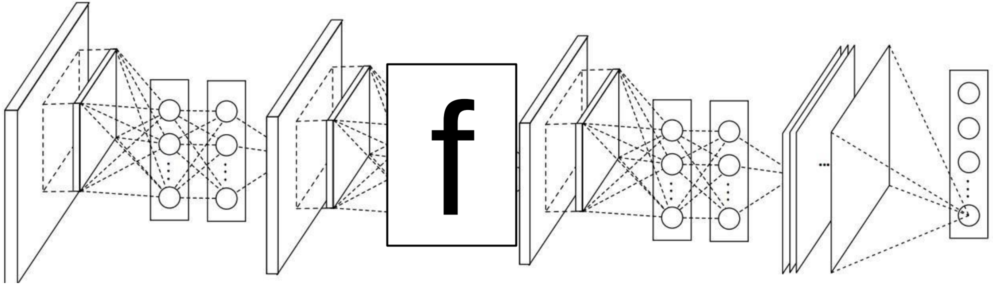


want to fool classifier → used L to maximize “wellness”

$$L(f(x+\delta),y)$$

Generating an Adversarial Example

89.7% pig



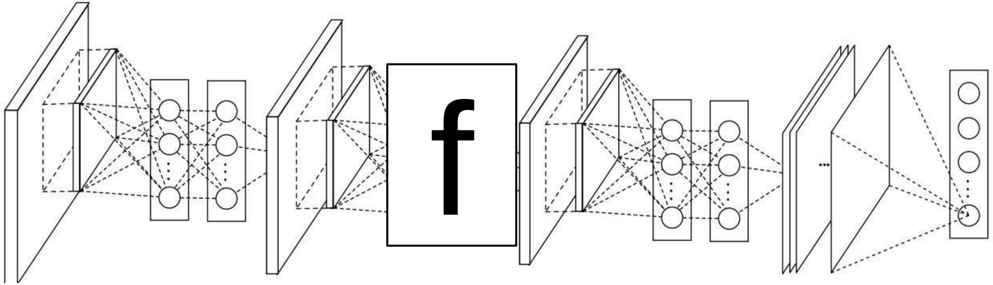
want to fool classifier → used L to ~~maximize “wellness”~~

maximize “badness”?

$$L(f(x+\delta),y)$$

Generating an Adversarial Example

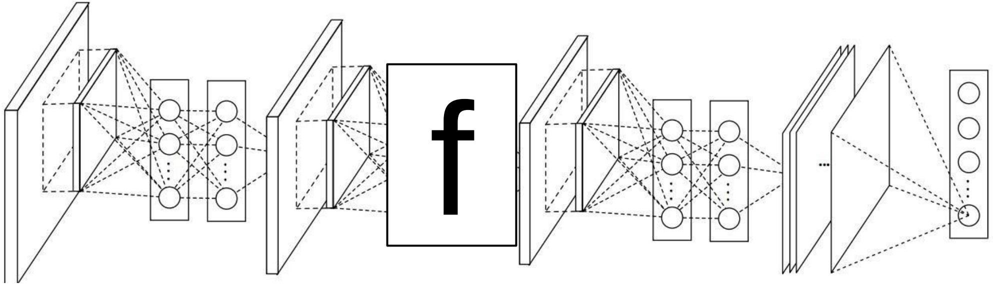
89.7% pig



want to fool classifier → maximize L

$$L(f(x+\delta),y)$$

Generating an Adversarial Example



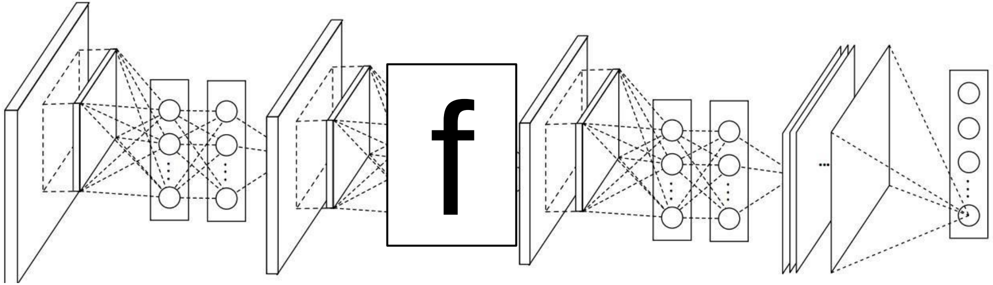
89.7% pig

want to fool classifier → maximize L w.r.t δ

$$L(f(x+\delta),y)$$

Generating an Adversarial Example

89.7% pig

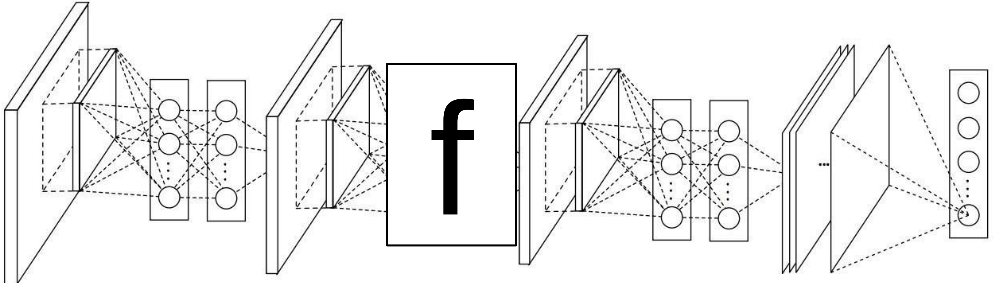


want to fool classifier → maximize L w.r.t $\boxed{\delta}$

$$L(f(x+\boxed{\delta}),y) \longrightarrow \nabla_{\boxed{\delta}} L$$

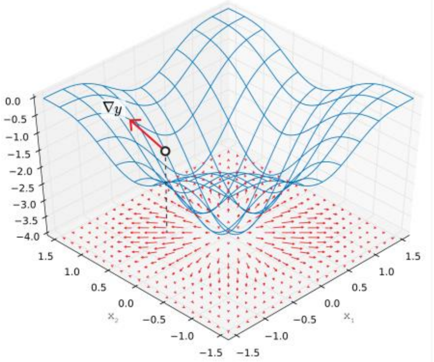
Generating an Adversarial Example

89.7% pig



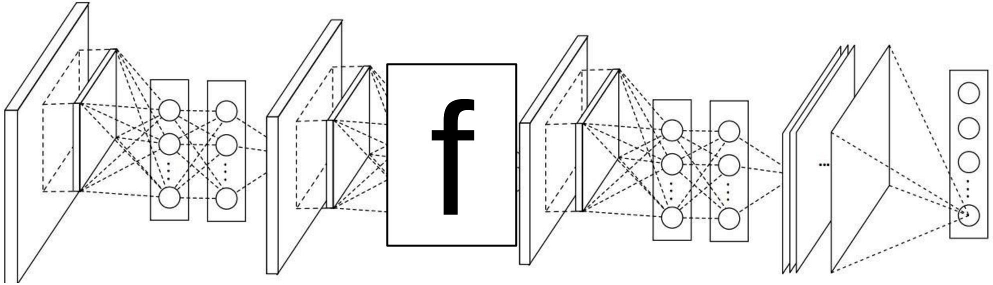
want to fool classifier → maximize L w.r.t δ

$$L(f(x+\delta),y) \longrightarrow +\nabla_{\delta} L$$



Generating an Adversarial Example

89.7% pig



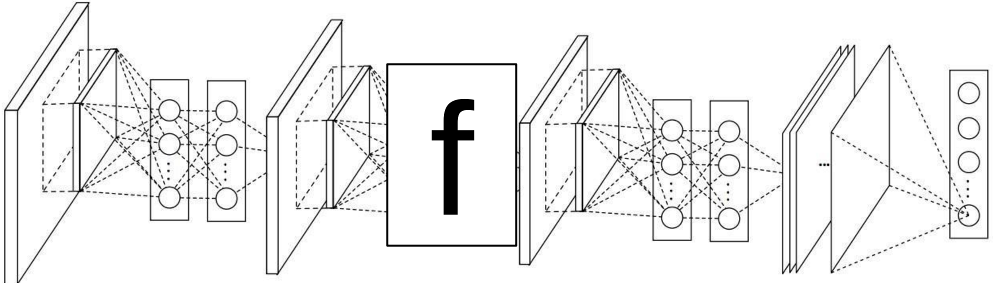
want to fool classifier → maximize L w.r.t $\boxed{x}$

$$L(f(\boxed{x+\delta}),y) \longrightarrow +\nabla_{\boxed{x}} L$$

(just a technicality..)

Generating an Adversarial Example

89.7% pig



want to fool classifier → maximize L w.r.t $\boxed{x}$

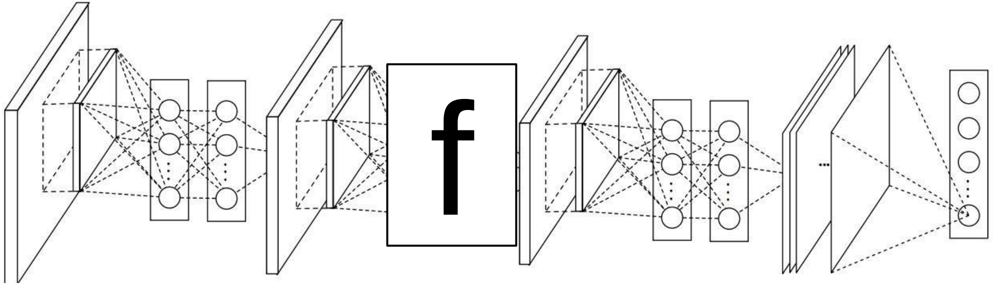
$$L(f(\underbrace{\boxed{x+\delta}}_{\text{input}}),y) \longrightarrow +\nabla_{\boxed{x}}L$$

input

(just a technicality..)

Generating an Adversarial Example

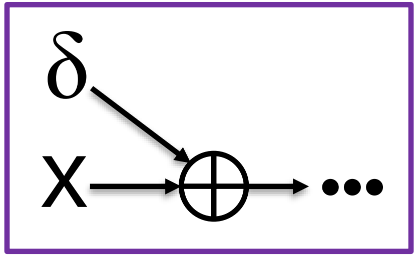
89.7% pig



want to fool classifier → maximize L w.r.t $\boxed{x}$

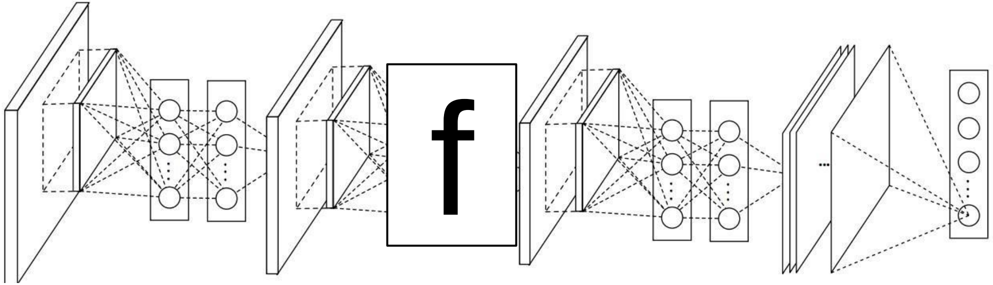
$$L(f(\underbrace{\boxed{x+\delta}}_{\text{input}}),y) \longrightarrow +\nabla_{\boxed{x}} L$$

input



(just a technicality..)

Generating an Adversarial Example

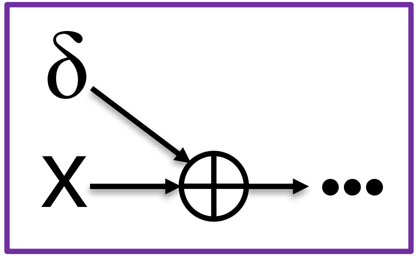


89.7% pig

want to fool classifier → maximize L w.r.t x

$$L(f(\underbrace{x+\delta}_{\text{input}}),y) \longrightarrow +\nabla_x L$$

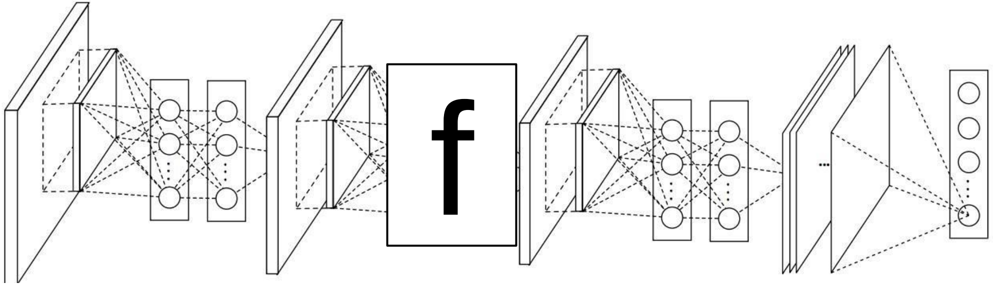
input



(just a technicality..)

Generating an Adversarial Example

89.7% pig

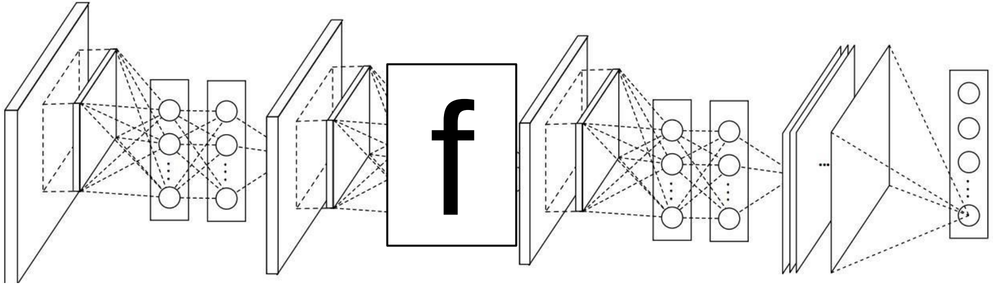


want to fool classifier → maximize L w.r.t x

$$L(f(x+\delta),y) \longrightarrow +\nabla_x L$$

Generating an Adversarial Example

89.7% pig



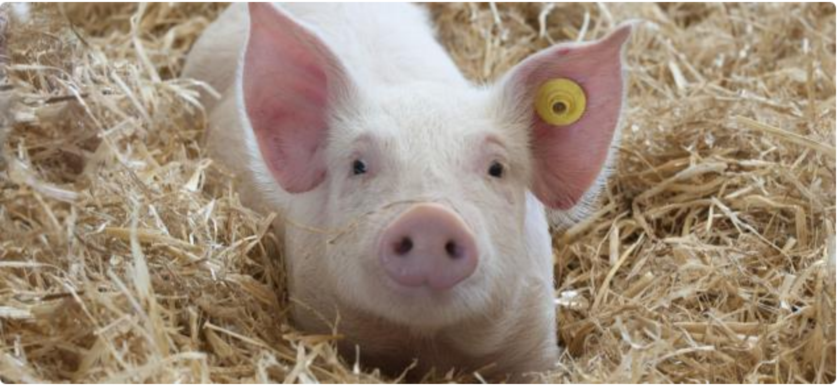
want to fool classifier → maximize L w.r.t x

$$L(f(x+\delta),y) \longrightarrow \delta = +\nabla_x L$$

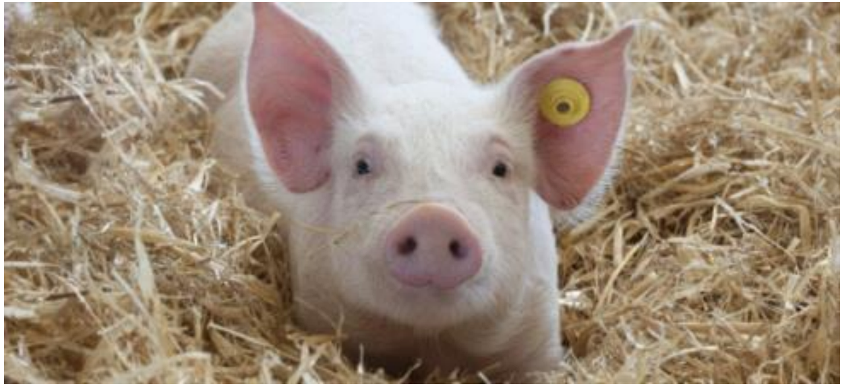
Follow the gradient w.r.t x (the input image)

X (original): 89.7% pig

Follow the gradient w.r.t x (the input image)

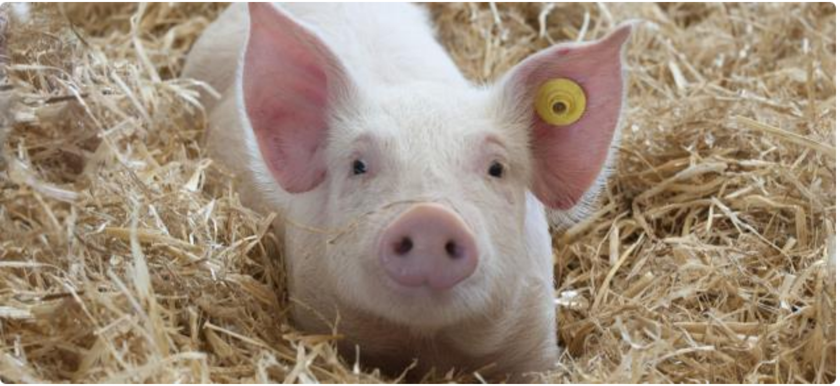


X (original): 89.7% pig

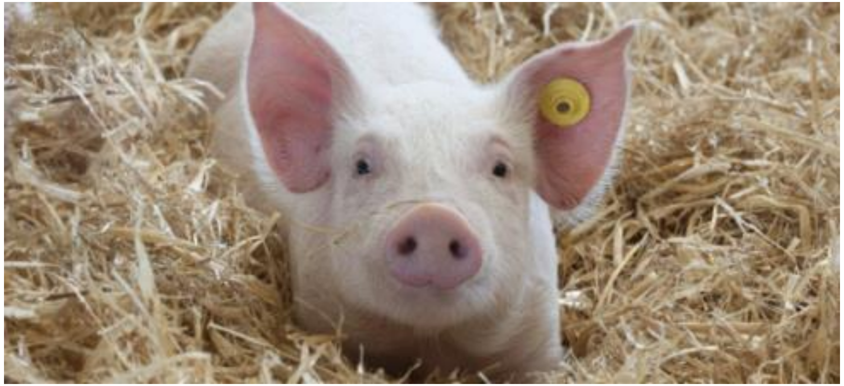


$X + \nabla_x L$: 68.6% hay

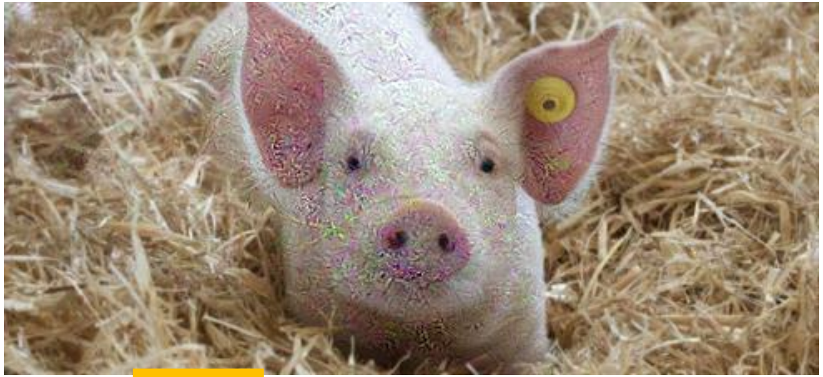
Follow the gradient w.r.t x (the input image)



X (original): 89.7% pig

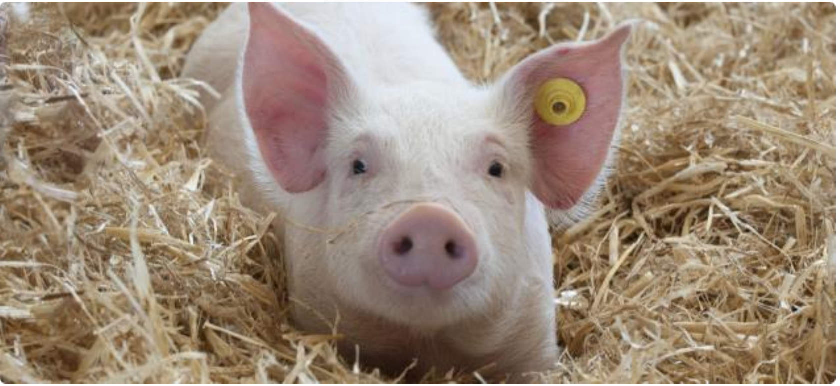


$X + \nabla_x L$: 68.6% hay

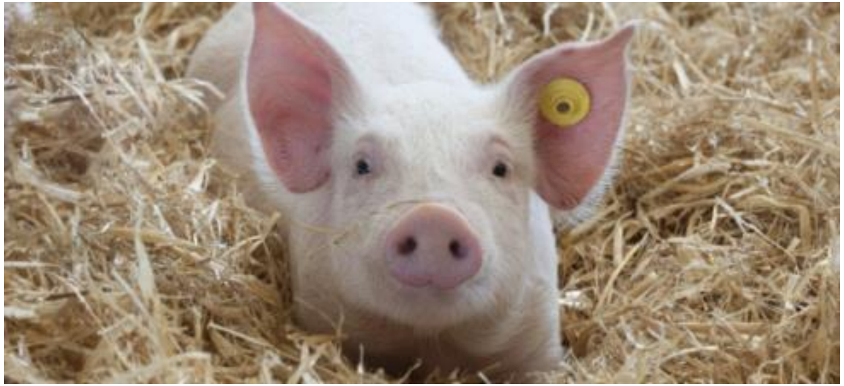


$X + \boxed{10\times}\nabla_x L$: 44.7% pig

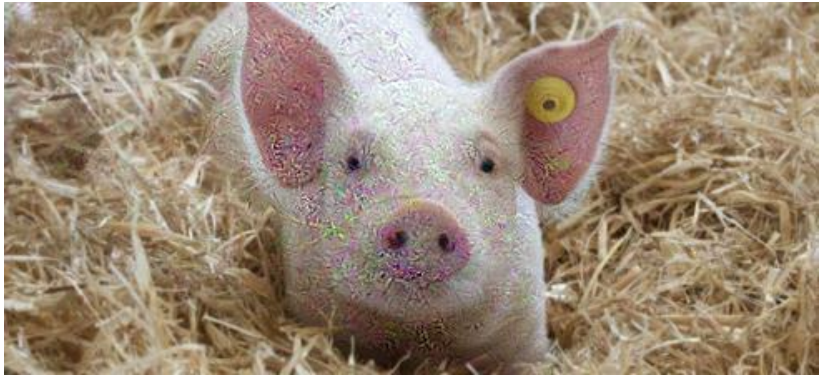
Follow the gradient w.r.t x (the input image)



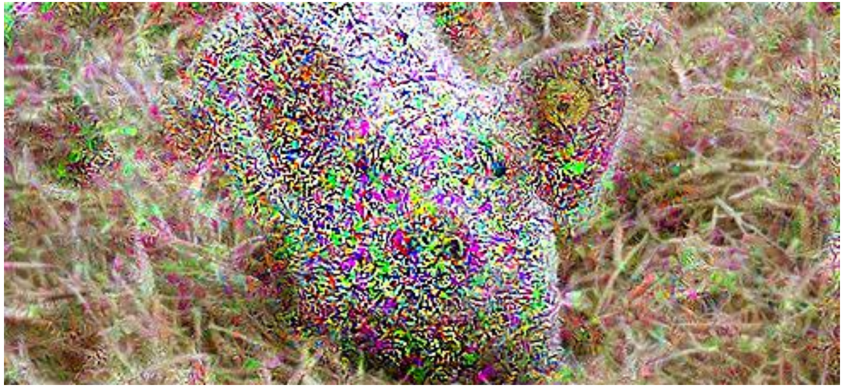
X (original): 89.7% pig



$X + \nabla_x L$: 68.6% hay

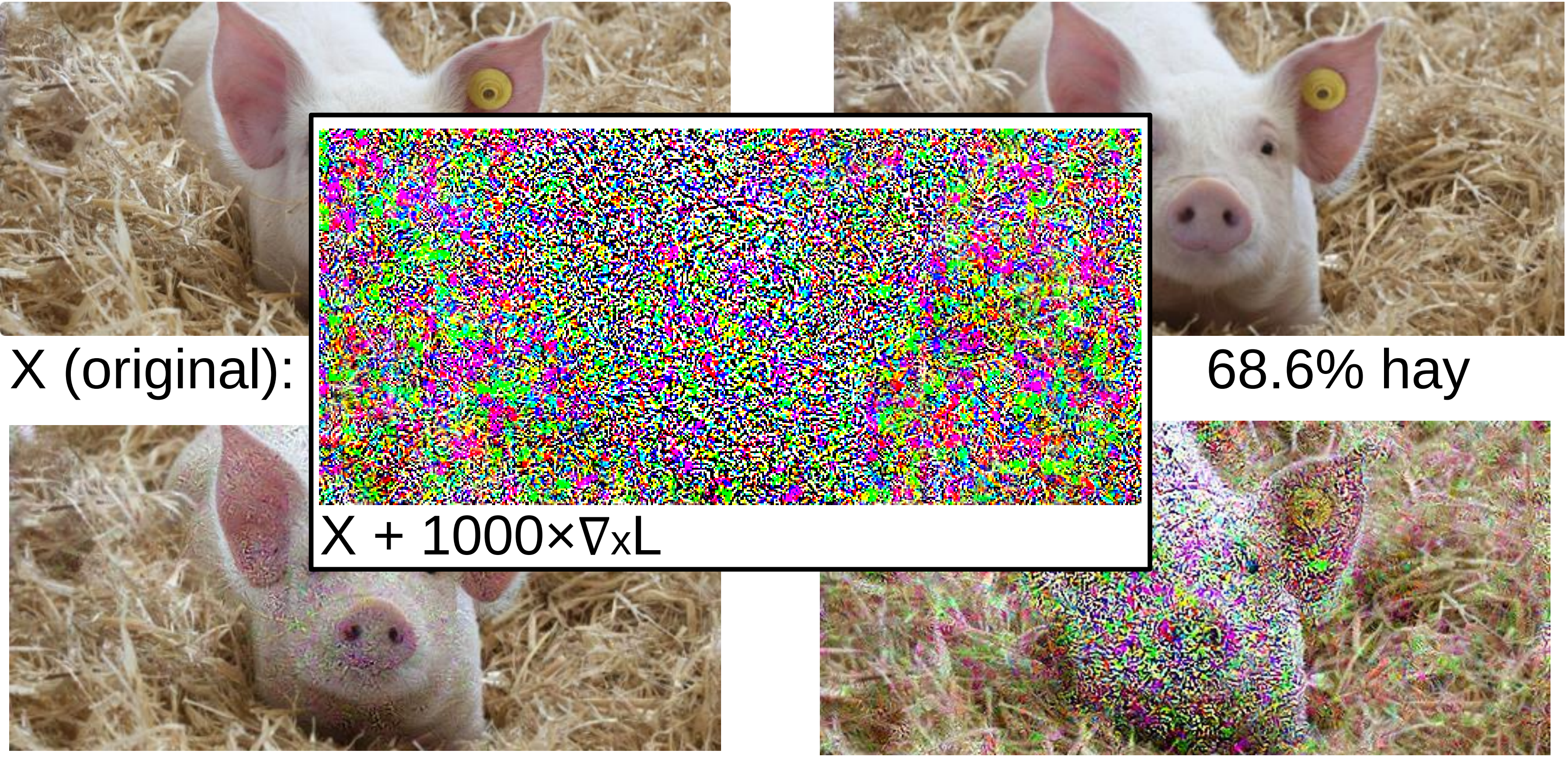


$X + 10\times\nabla_x L$: 44.7% pig



$X + 100\times\nabla_x L$: 44.8% fireguard

Follow the gradient w.r.t x (the input image)



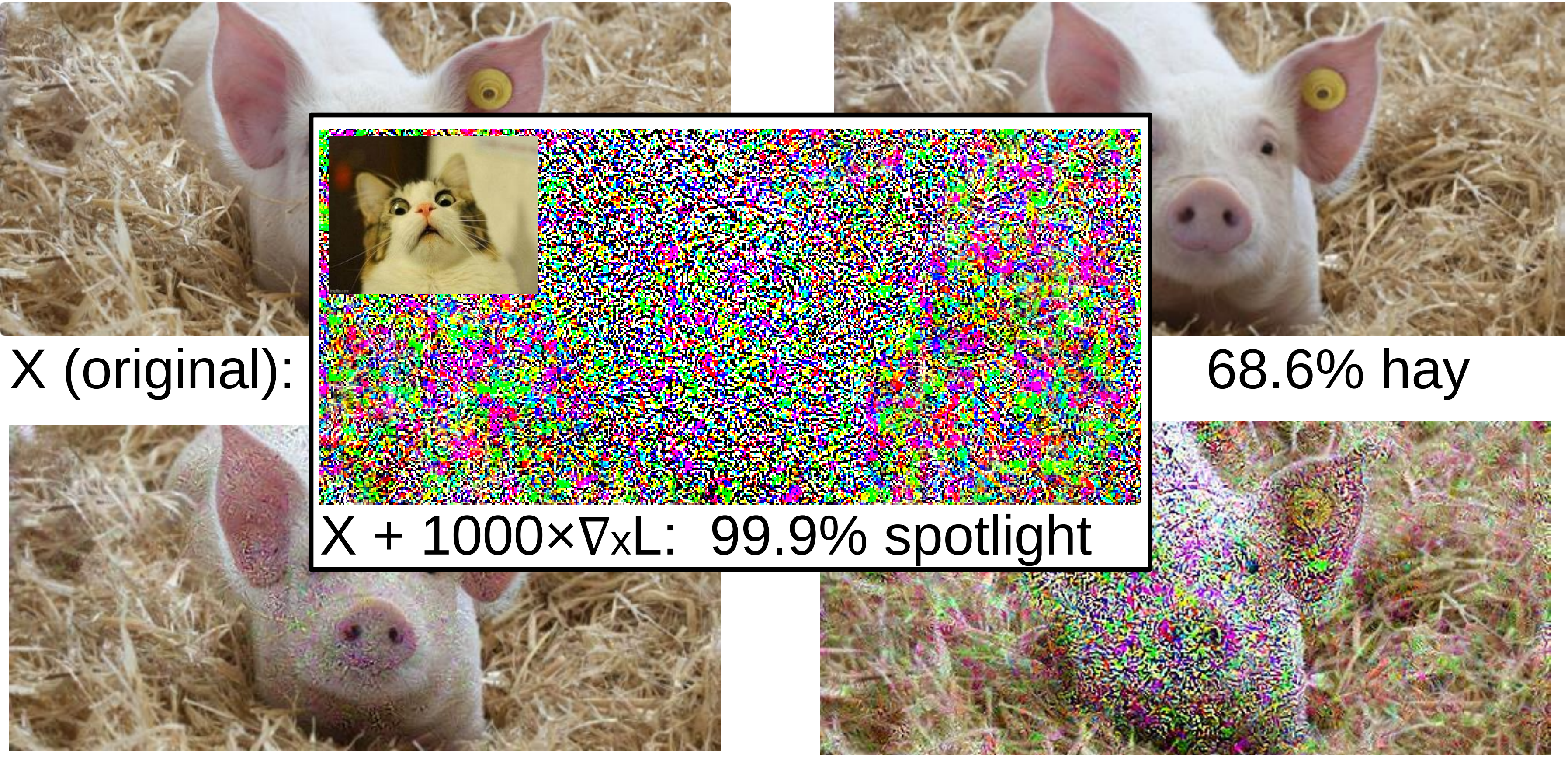
X (original): 68.6% hay

$X + 1000\times\nabla_x L$

$X + 10\times\nabla_x L$: 44.7% pig

$X + 100\times\nabla_x L$: 44.8% fireguard

Follow the gradient w.r.t x (the input image)



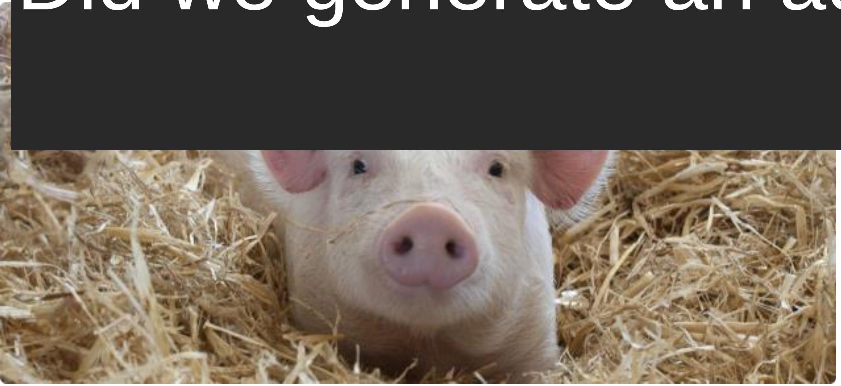
X (original): 68.6% hay

$X + 1000\times\nabla_x L$: 99.9% spotlight

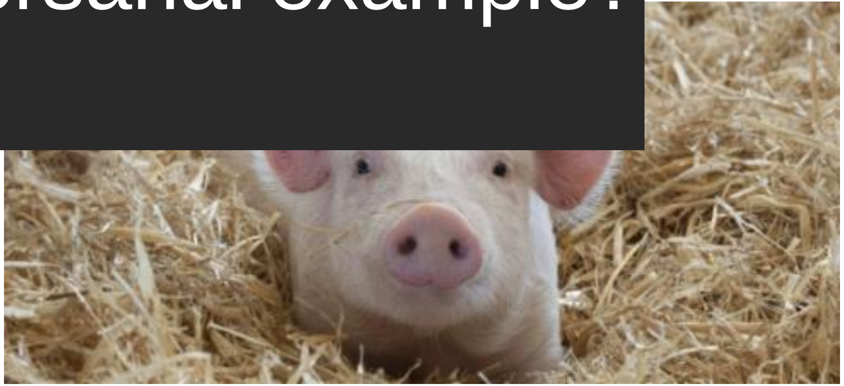
$X + 10\times\nabla_x L$: 44.7% pig

$X + 100\times\nabla_x L$: 44.8% fireguard

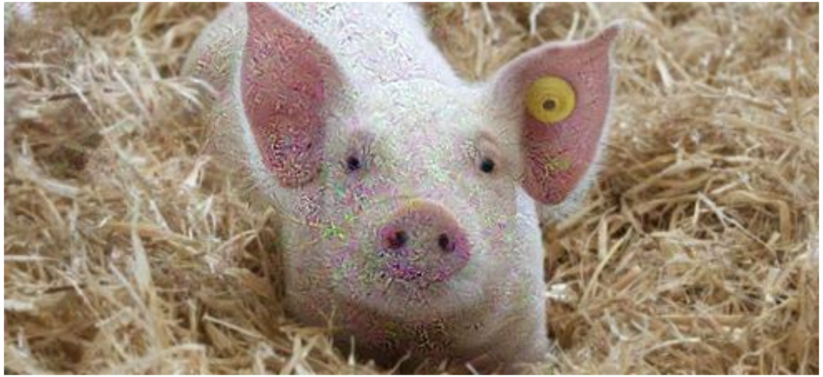
Did we generate an adversarial example?



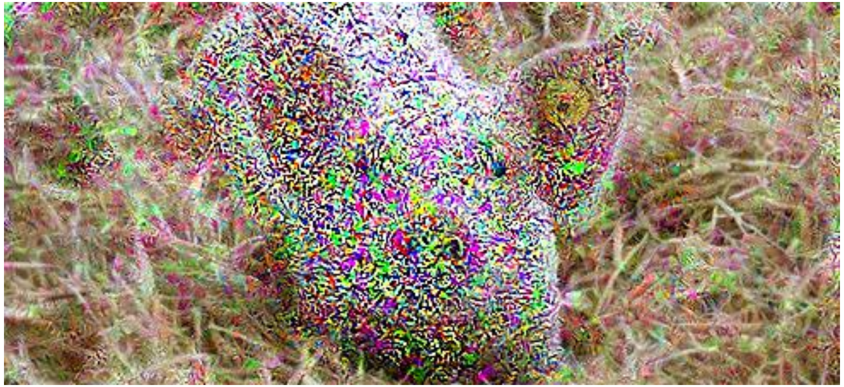
X (original): 89.7% pig



X + $\nabla_x L$: 68.6% hay

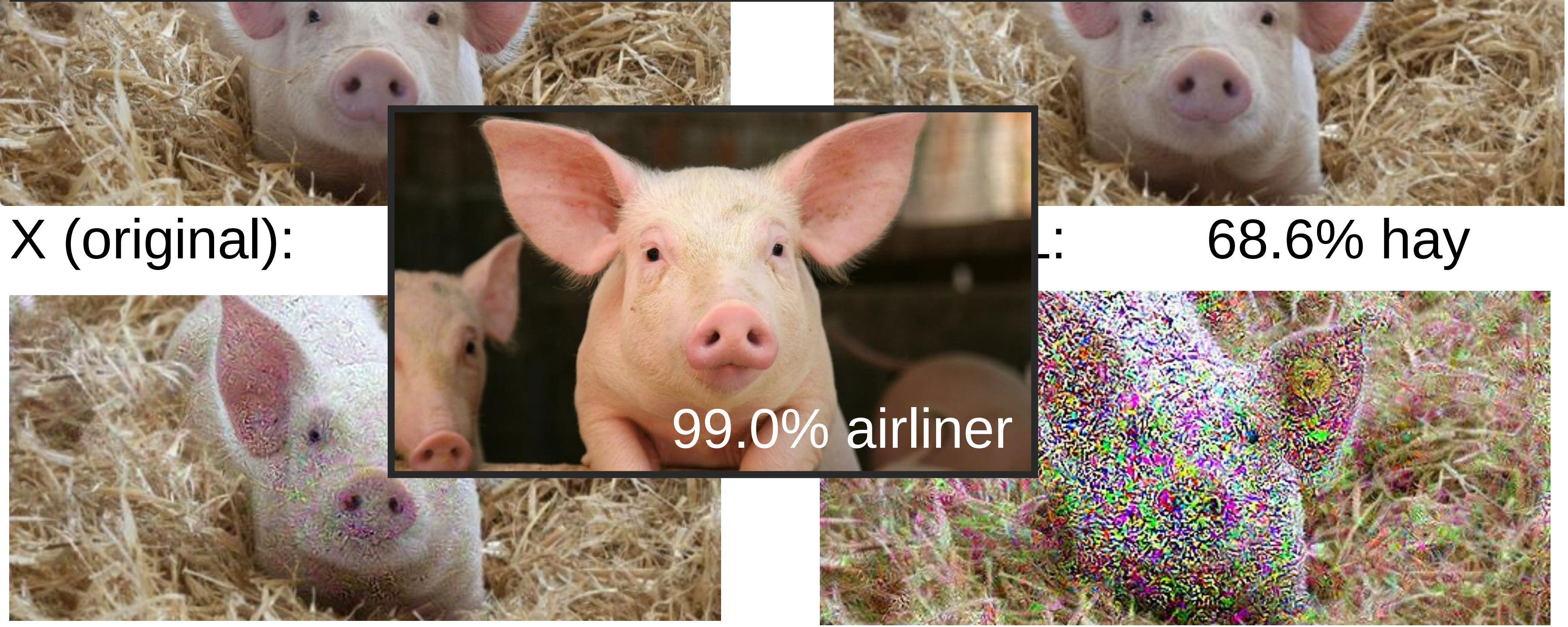


X + 10×$\nabla_x L$: 44.7% pig



X + 100×$\nabla_x L$: 44.8% fireguard

Did we generate an adversarial example?



X (original):

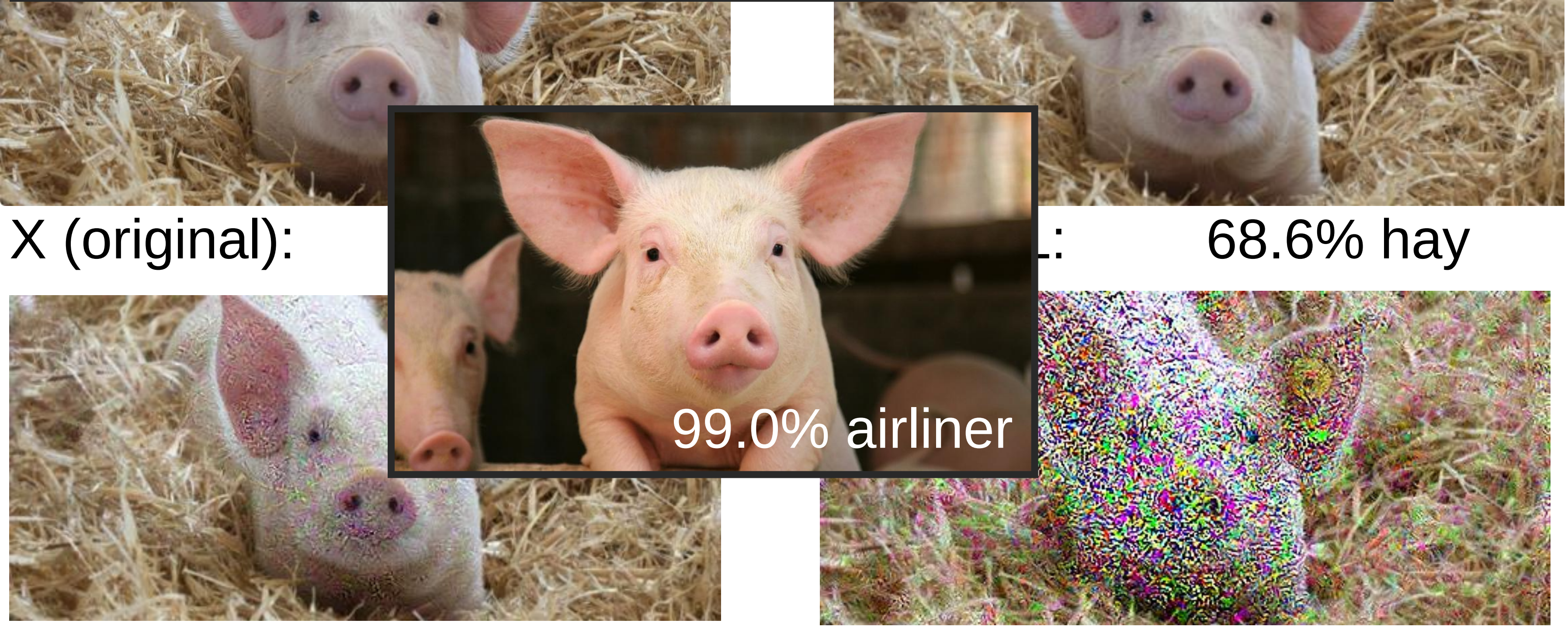
L: 68.6% hay

$X + 10\times\nabla_x L$: 44.7% pig

$X + 100\times\nabla_x L$: 44.8% fireguard

Did we generate an adversarial example?

Need small δ…



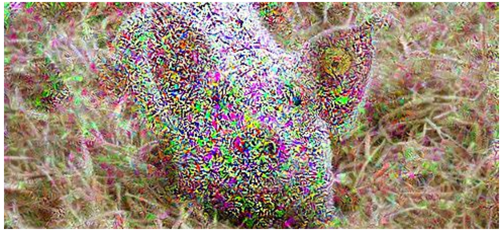
X (original):

L: 68.6% hay

X + 10×$\nabla_x L$: 44.7% pig

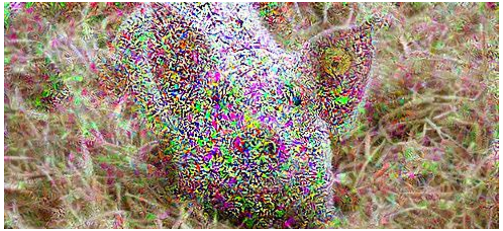
X + 100×$\nabla_x L$: 44.8% fireguard

We want *small noise*



We want *small noise*

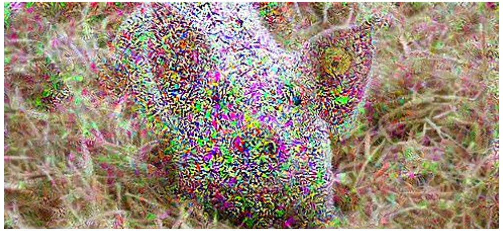
$$x \qquad\qquad =$$



We want *small noise*

+

=



X δ

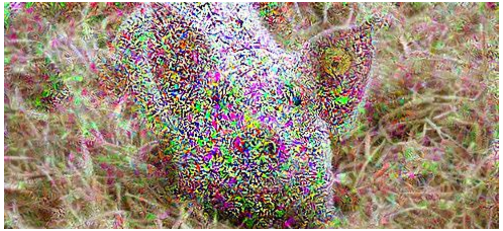
We want *small noise*

X

+

δ

=

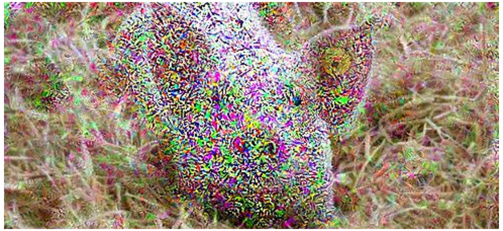


What is small δ?

We want *small noise*

+

=

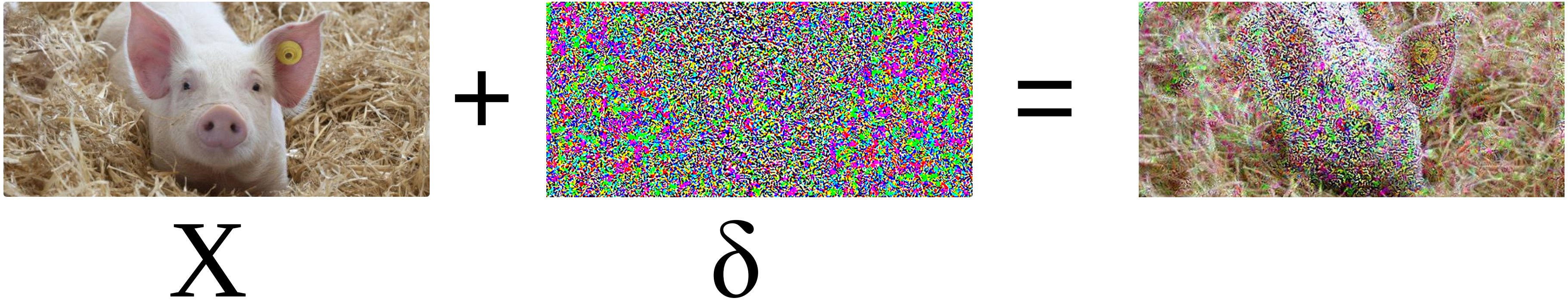


X δ

What is small δ?

$$\|\delta\| < \varepsilon$$

We want *small noise*



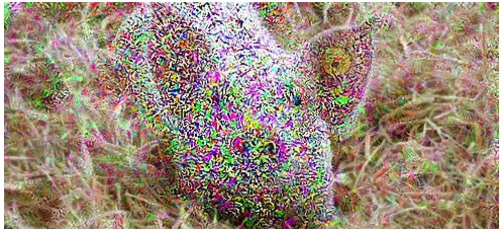
What is small δ?

$$\|\delta\|_\infty < \varepsilon$$

We want *small noise*

+

=



X

δ

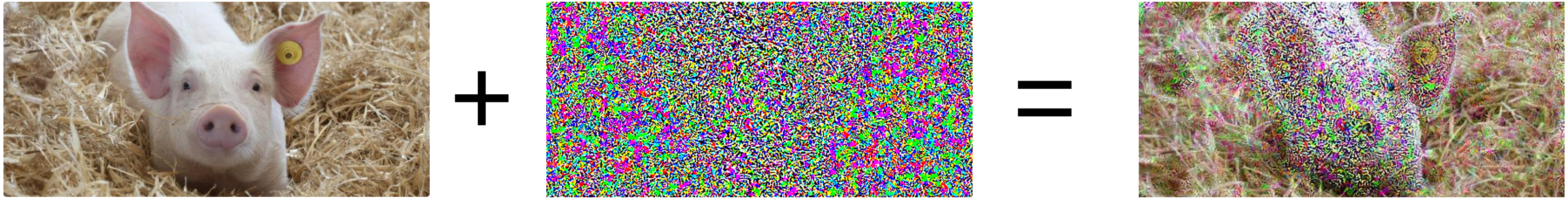
What is small δ?

$$\|\delta\|_\infty < \varepsilon$$

$$\|\delta\|_\infty \leq 0.1$$

0.1			

We want *small noise*



X

δ

$\|\delta\|_\infty \leq 0.1$

0.1	-0.1		

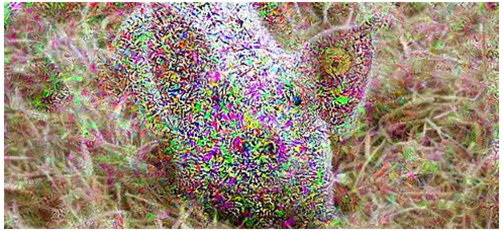
What is small δ?

$$\|\delta\|_\infty < \varepsilon$$

We want *small noise*

+

=



X

δ

$\|\delta\|_\infty \le 0.1$

0.1	-0.1		
	0.1	0.05	-0.02
		-0.09	
			10^{-5}

What is small δ?

$$\|\delta\|_\infty < \varepsilon$$

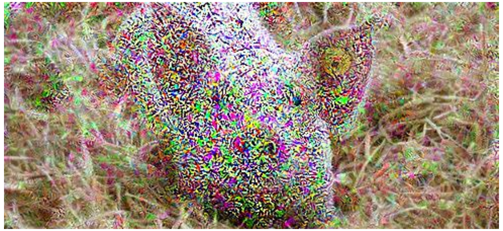
We want *small noise*

X

+

δ

=



What is small δ?

$$\|\delta\|_\infty < \varepsilon$$

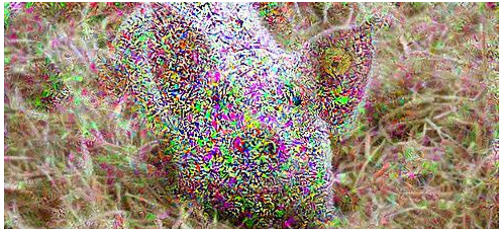
$$\|\delta\|_\infty \leq 0.1$$

0.1	-0.1		~~-0.2~~
	0.1	0.05	-0.02
		-0.09	
			10^{-5}

We want *small noise*

+

=



X

δ

$\|\delta\|_\infty \leq 0.1$

0.1	-0.1		
	0.1	0.05	-0.02
		-0.09	
			10^{-5}

What is small δ?

$$\|\delta\|_\infty < \varepsilon$$

small δ & $\delta = f(\nabla x L)$?

“Enforcing $\|\nabla_x L\|_\infty < \varepsilon$” :

“Enforcing $\|\nabla_x L\|_\infty < \varepsilon$” :

$$\delta = \qquad\qquad \nabla_x L$$

“Enforcing $\|\nabla_x L\|_\infty < \varepsilon$” :

12	-0.1	432	…
…	10^{-5}	…	…
…	…	-555	…
…	…	0	…

$$\delta = \qquad \nabla_x L$$

“Enforcing $\|\nabla_x L\|_\infty < \varepsilon$” :

1	-1	1	…
…	1	…	…
…	…	-1	…
…	…	0	…

$$\delta = \quad \mathrm{sgn}(\nabla_x L)$$

“Enforcing $\|\nabla_x L\|_\infty < \varepsilon$” :

ε	-ε	ε	…
…	ε	…	…
…	…	-ε	…
…	…	0	…

$$\delta = \varepsilon \cdot \mathrm{sgn}(\nabla_x L)$$

“Enforcing $\|\nabla_x L\|_\infty < \varepsilon$” :

ε	-ε	ε	…
…	ε	…	…
…	…	-ε	…
…	…	0	…

$$\delta = \varepsilon \cdot \mathrm{sgn}(\nabla_x L)$$

“Enforcing $\|\nabla_x L\|_\infty < \varepsilon$” :

ε	-ε	ε	…
…	ε	…	…
…	…	-ε	…
…	…	0	…

$$\boxed{\delta = \varepsilon \cdot \mathrm{sgn}(\nabla_x L)}$$

Fast Gradient Sign Method

a.k.a FGSM (Goodfellow et al. 2015)

“Enforcing $\|\nabla_x L\|_\infty < \varepsilon$” :

ε	-ε	ε	…
…	ε	…	…
…	…	-ε	…
…	…	0	…

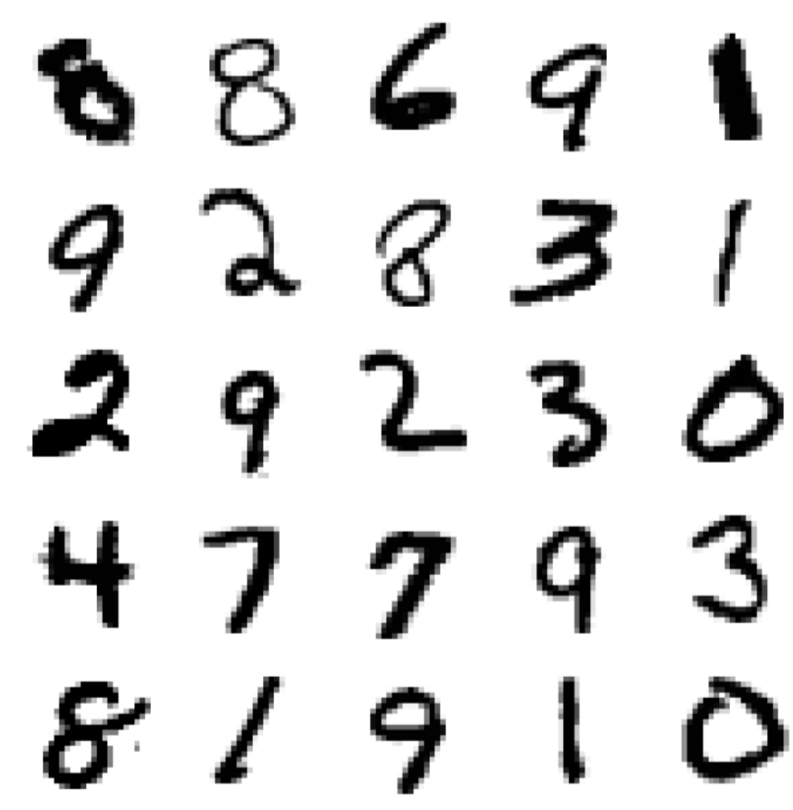
*$$\boxed{\delta = \varepsilon \cdot \operatorname{sgn}(\nabla_x L)}$$

Fast Gradient Sign Method

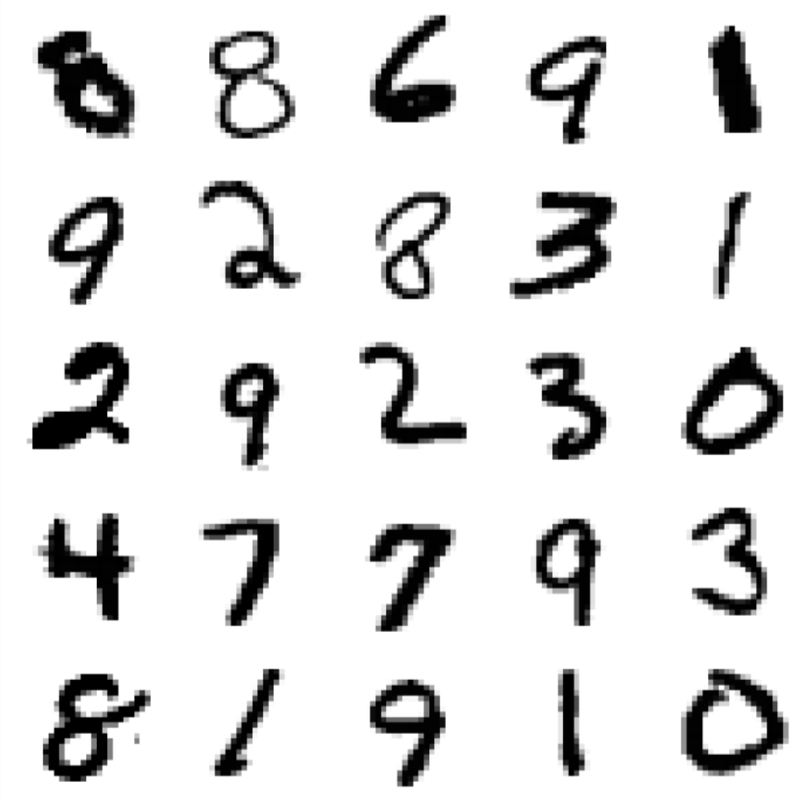
a.k.a FGSM (Goodfellow et al. 2015)

*$$\delta = \max_{\|\delta\|_\infty \leq \epsilon} L\left(f(x+\delta), y\right) \approx \max_{\|\delta\|_\infty \leq \epsilon} L\left(f(x), y\right) + \nabla_x L\, \delta$$

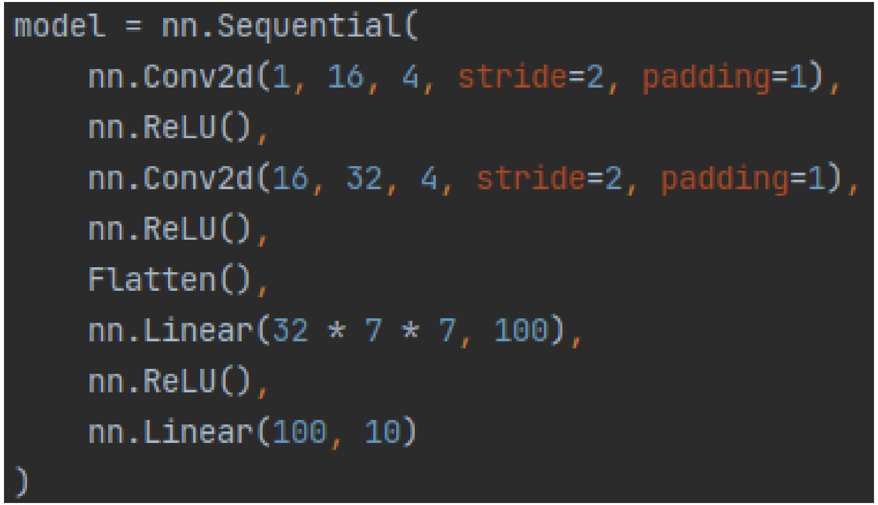
FGSM – example on MNIST



FGSM – example on MNIST

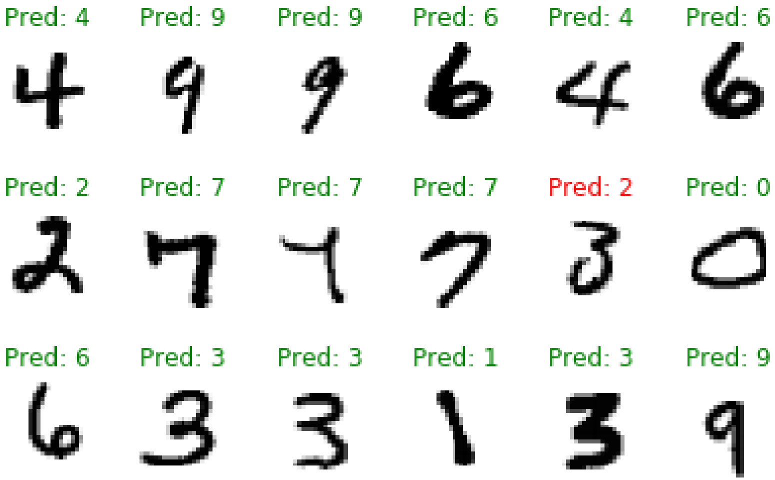


Classifier



```
model = nn.Sequential(
    nn.Conv2d(1, 16, 4, stride=2, padding=1),
    nn.ReLU(),
    nn.Conv2d(16, 32, 4, stride=2, padding=1),
    nn.ReLU(),
    Flatten(),
    nn.Linear(32 * 7 * 7, 100),
    nn.ReLU(),
    nn.Linear(100, 10)
)
```

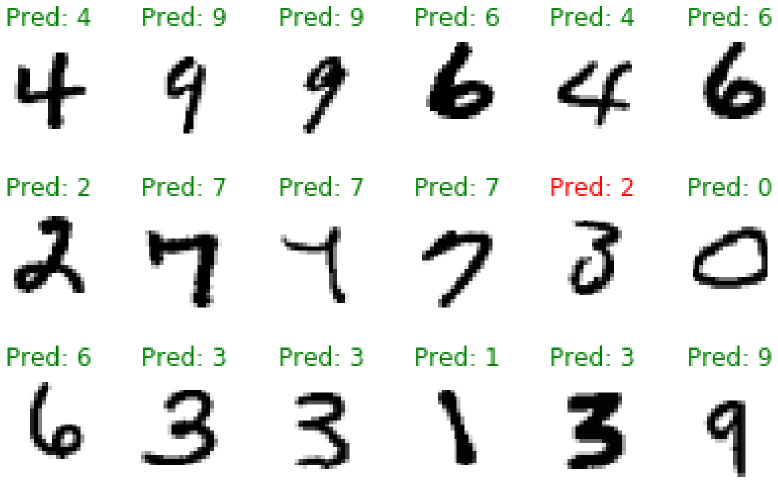
FGSM - MNIST



Test Samples

FGSM - MNIST

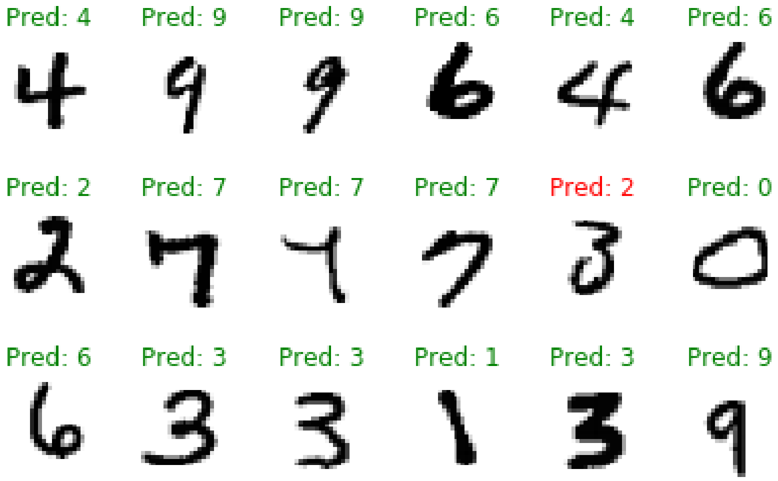
$$\mathbf{X}^{\mathrm{adv}} = \mathbf{X} + \epsilon\,\mathrm{sgn}(\nabla_X L(\mathbf{X}, y_{\mathrm{true}}))$$



Test Samples

FGSM - MNIST

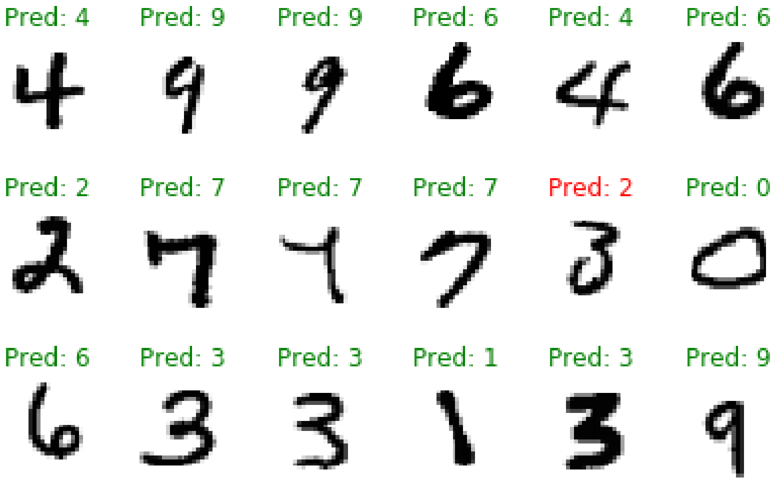
$$\boxed{\mathbf{X}^{\mathrm{adv}}} = \mathbf{X} + \epsilon\, \mathrm{sgn}(\nabla_X L(\mathbf{X}, y_{\mathrm{true}}))$$



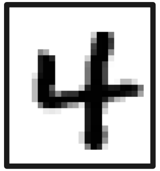
Test Samples

FGSM - MNIST

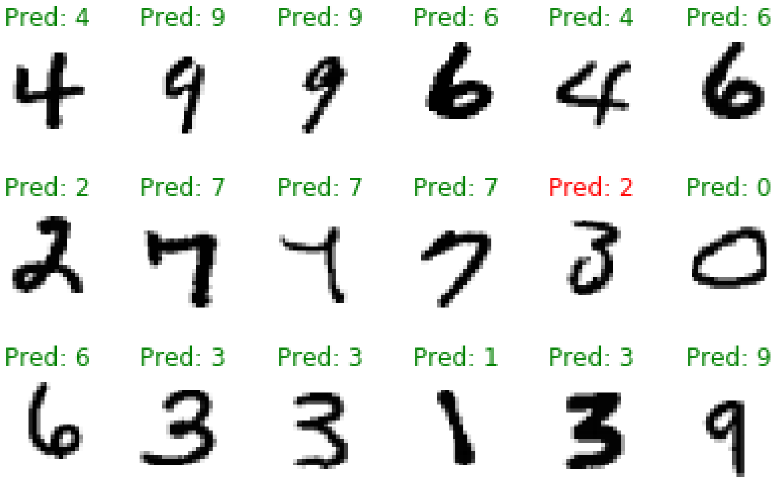
$$\mathbf{X}^{\text{adv}} = \boxed{\mathbf{X}} + \epsilon \, \text{sgn}(\nabla_X L(\mathbf{X}, y_{\text{true}}))$$



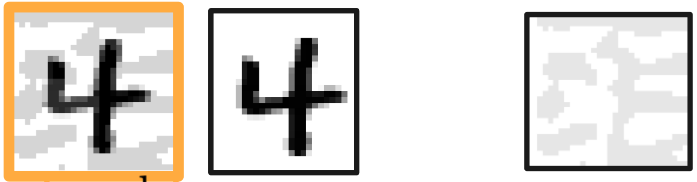
FGSM - MNIST



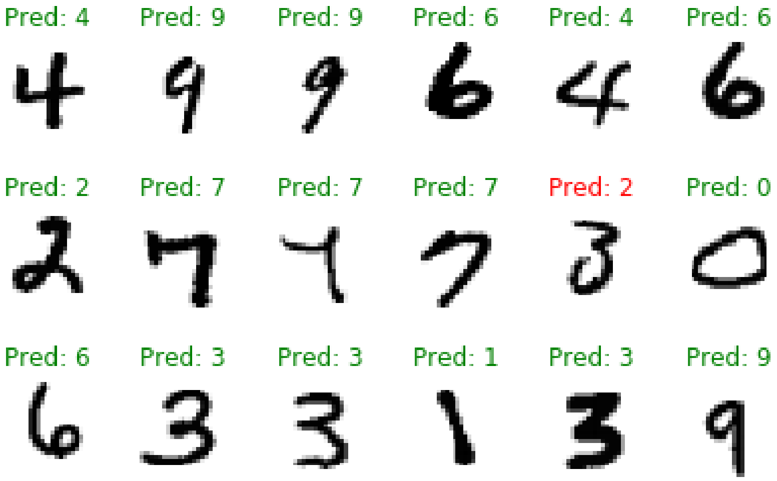
$$\mathbf{X}^{\text{adv}} = \mathbf{X} + \boxed{\epsilon \, \text{sgn}(\nabla_X L(\mathbf{X}, y_{\text{true}}))}$$



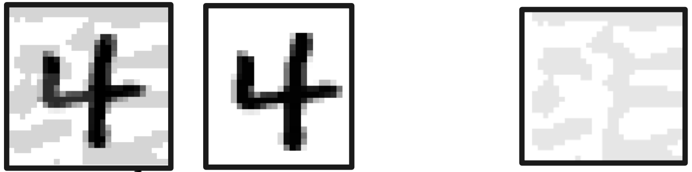
FGSM - MNIST



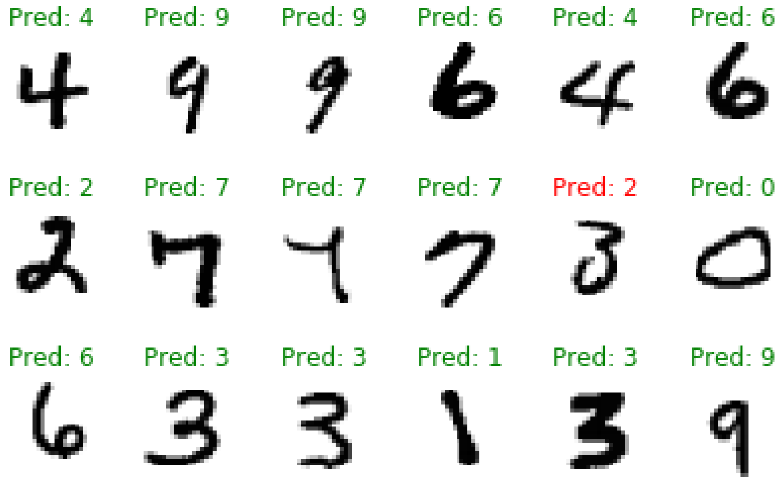
$$\mathbf{X}^{\text{adv}} = \mathbf{X} + \epsilon \;\text{sgn}(\nabla_X L(\mathbf{X}, y_{\text{true}}))$$

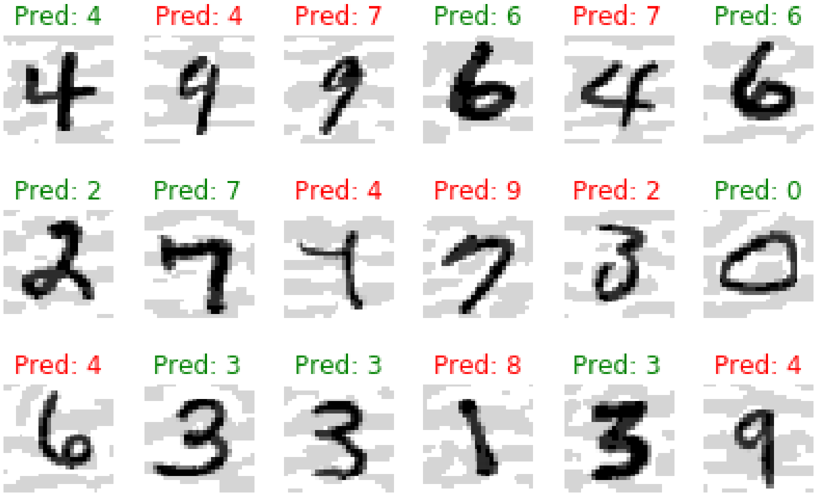


FGSM - MNIST

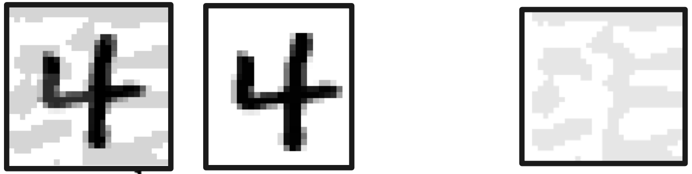


$$\mathbf{X}^{\text{adv}} = \mathbf{X} + \epsilon \, \text{sgn}(\nabla_X L(\mathbf{X}, y_{\text{true}}))$$





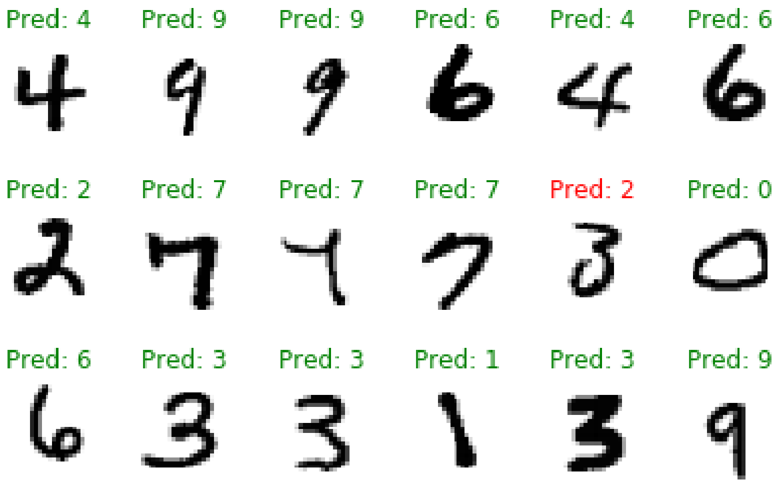
FGSM - MNIST

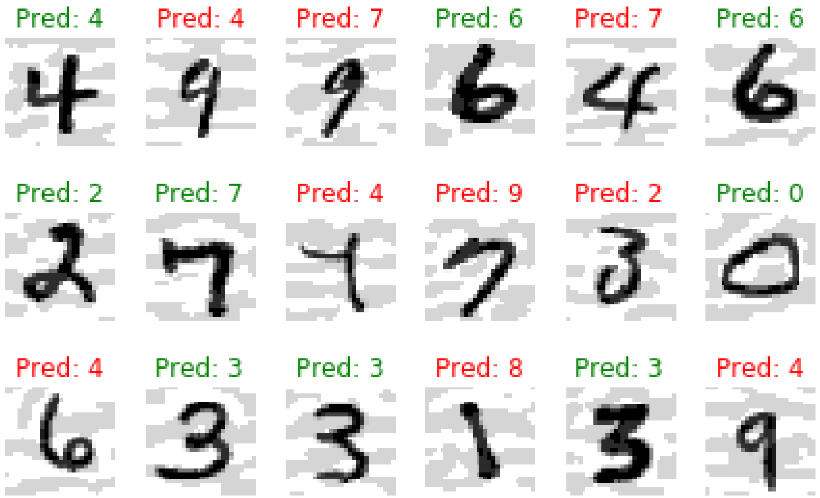


$$\mathbf{X}^{\text{adv}} = \mathbf{X} + \epsilon \ \text{sgn}(\nabla_X L(\mathbf{X}, y_{\text{true}}))$$

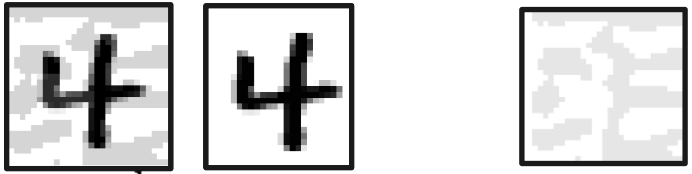
Test Error: 98.7%

FGSM Error: ?





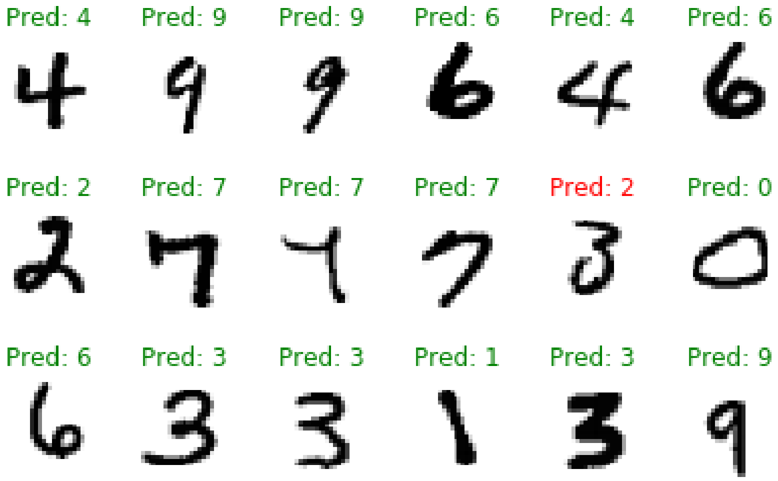
FGSM - MNIST

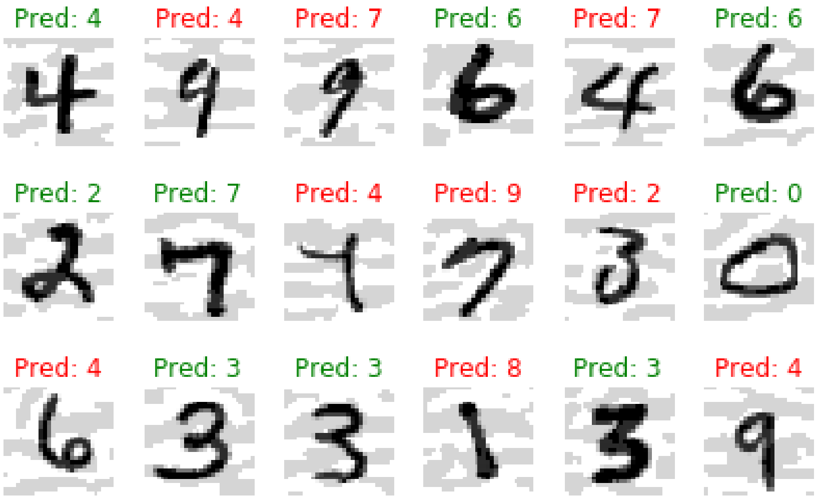


$$\mathbf{X}^{\text{adv}} = \mathbf{X} + \epsilon \operatorname{sgn}(\nabla_X L(\mathbf{X}, y_{\text{true}}))$$

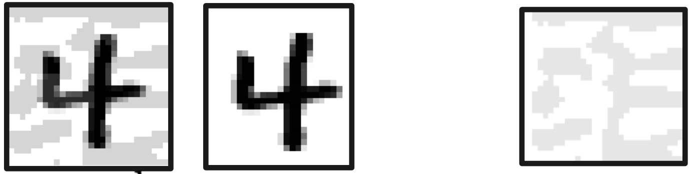
Test Error: 98.7%

FGSM Error: 40.0%





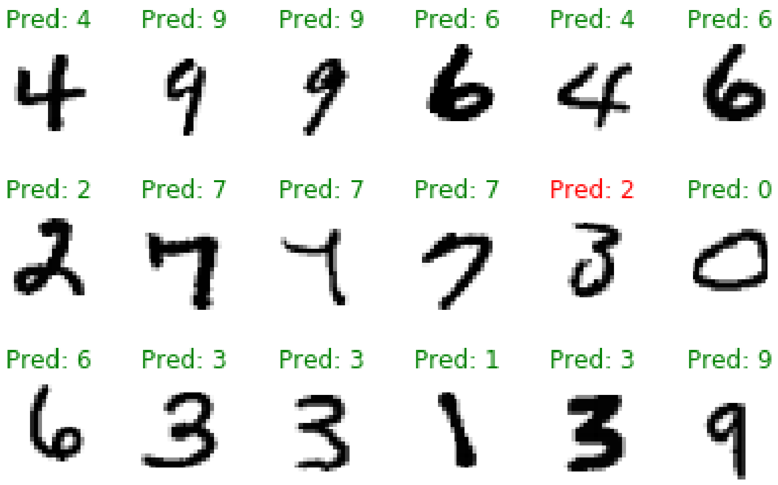
FGSM - MNIST

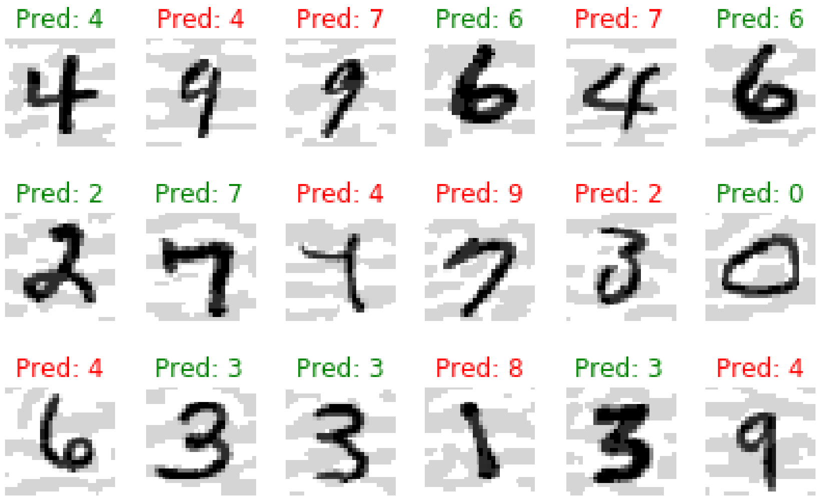


$$\mathbf{X}^{\text{adv}} = \mathbf{X} + \epsilon \ \text{sgn}(\nabla_X L(\mathbf{X}, y_{\text{true}}))$$

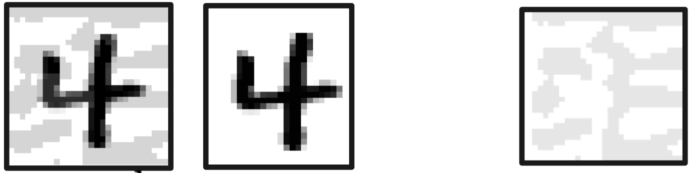
Test Error: 98.7%

FGSM (ε=0.1) Error: 40.0%





FGSM - MNIST

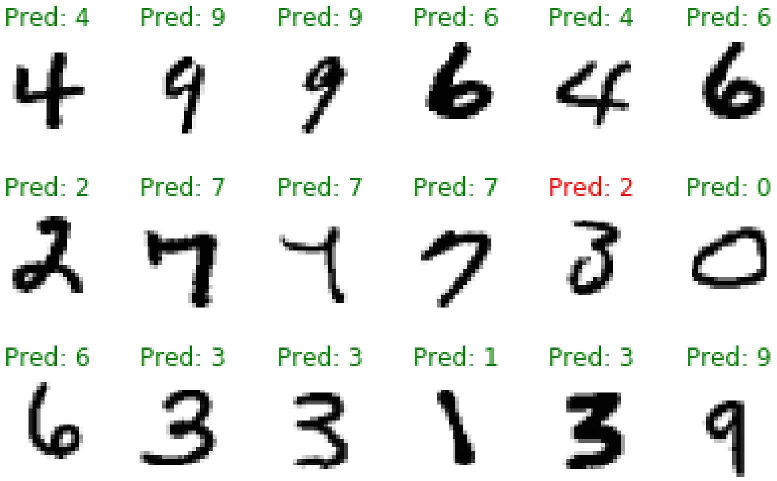


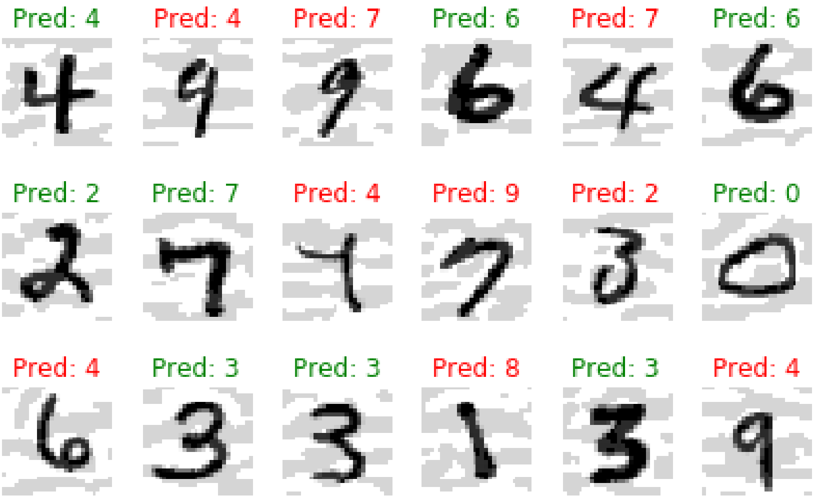
$$\mathbf{X}^{\text{adv}} = \mathbf{X} + \epsilon \ \text{sgn}(\nabla_X L(\mathbf{X}, y_{\text{true}}))$$

Simple, Fast and Vicious

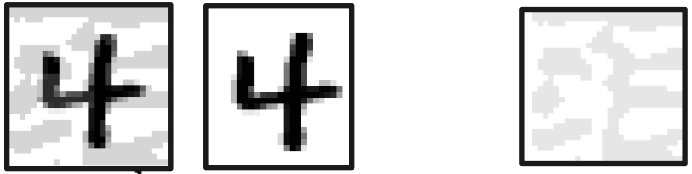
Test Error: 98.7%

FGSM (ε=0.1) Error: 40.0%





FGSM - MNIST

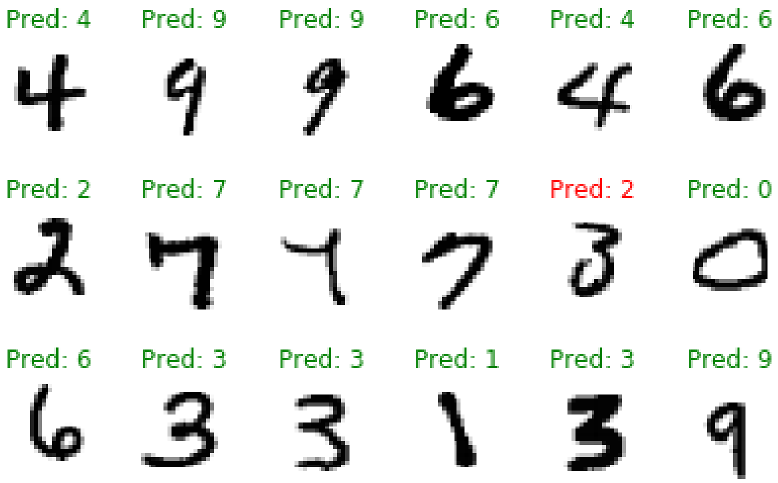


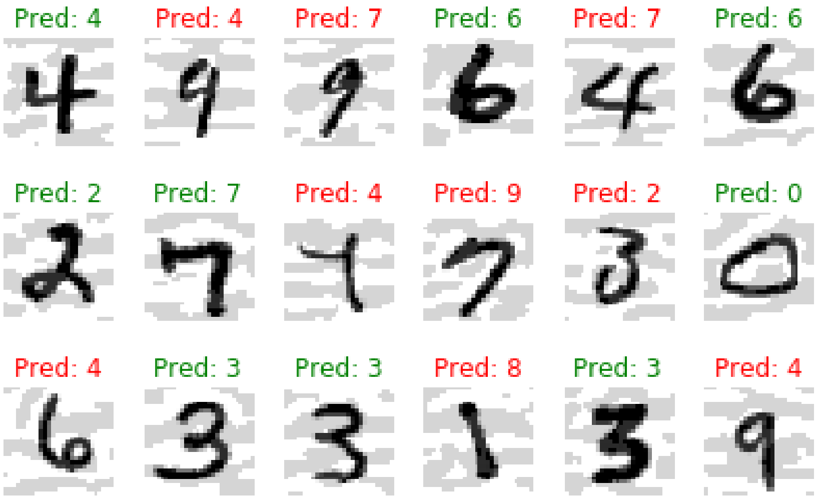
$$\mathbf{X}^{\text{adv}} = \mathbf{X} + \epsilon \, \text{sgn}(\nabla_X L(\mathbf{X}, y_{\text{true}}))$$

Simple, Fast and Vicious

Test Error: 98.7%

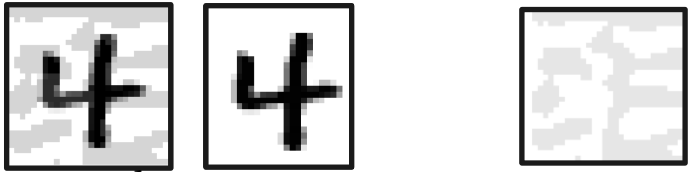
FGSM (ε=0.1) Error: 40.0%





source: https://adversarial-ml-tutorial.org/

FGSM - MNIST

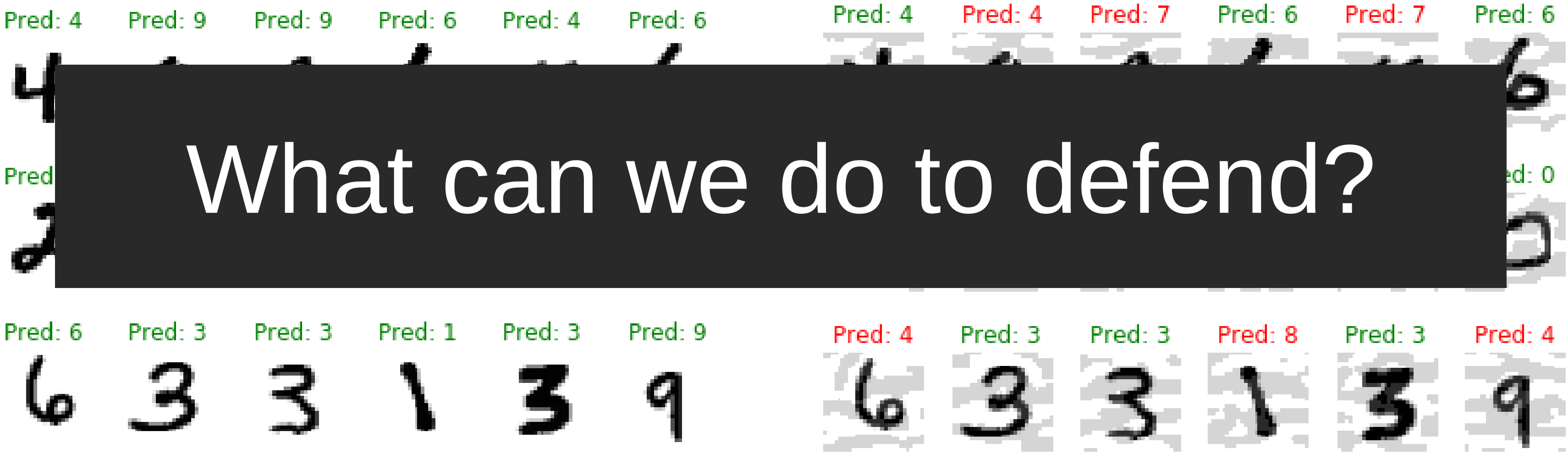


$$\mathbf{X}^{\text{adv}} = \mathbf{X} + \epsilon \, \text{sgn}(\nabla_X L(\mathbf{X}, y_{\text{true}}))$$

Simple, Fast and Vicious

Test Error: 98.7%

FGSM (ε=0.1) Error: 40.0%

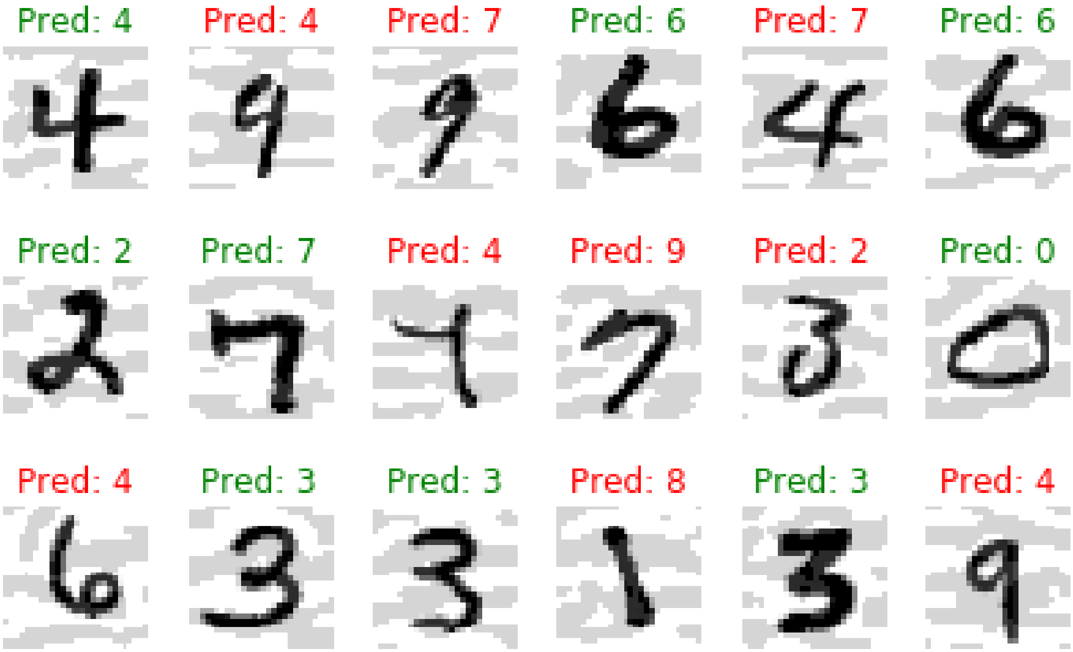


What can we do to defend?

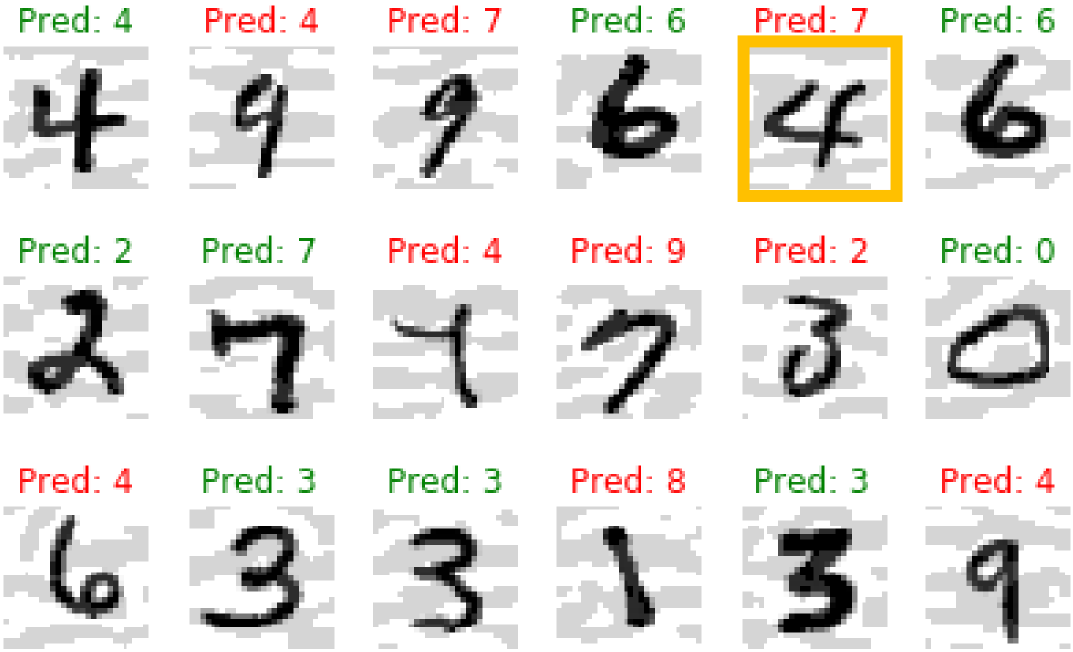
source: https://adversarial-ml-tutorial.org/

Adversarial Training

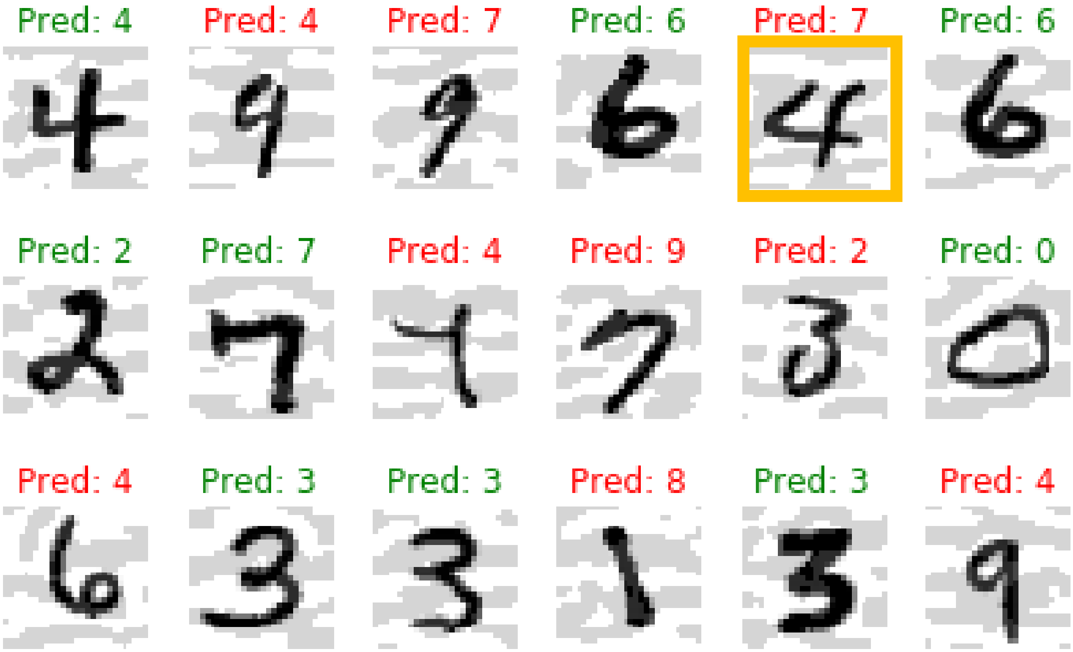
Adversarial Training



Adversarial Training

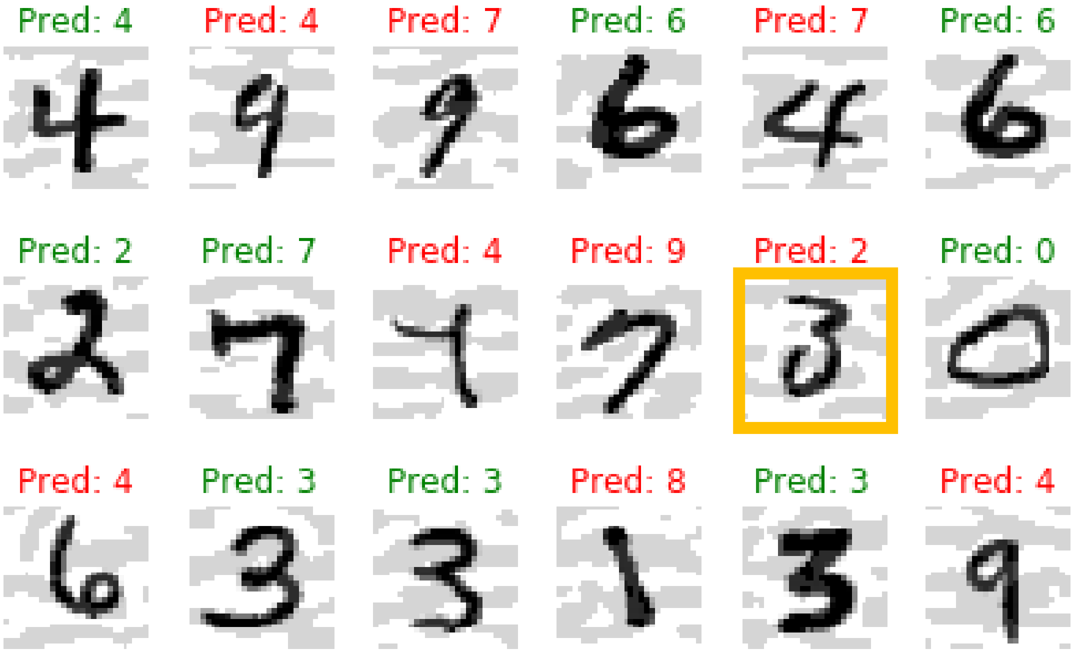


Adversarial Training



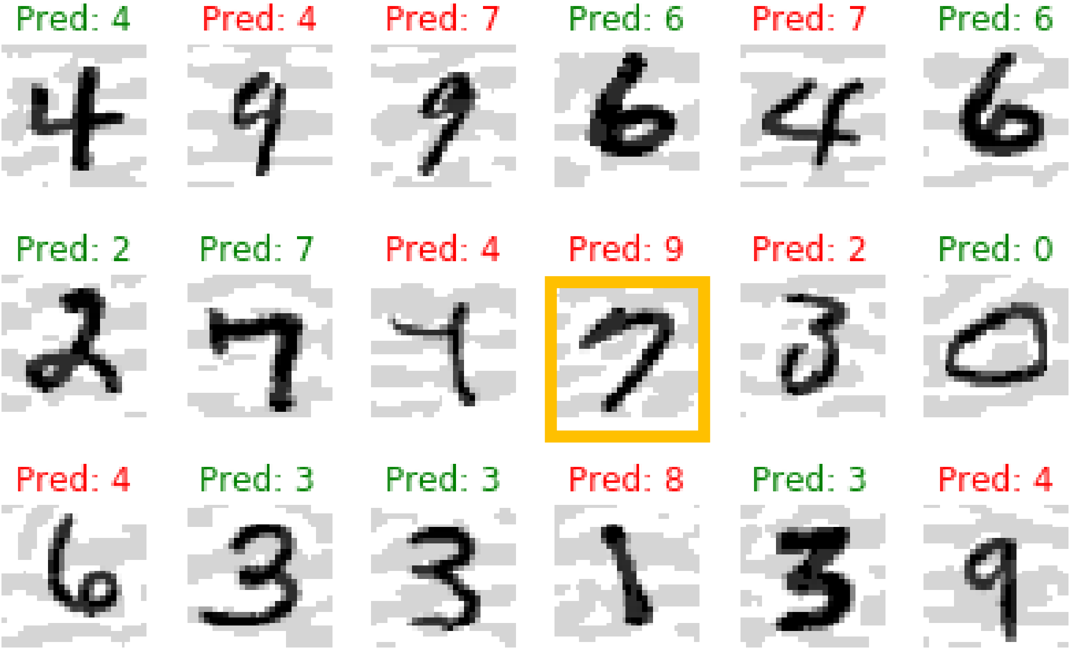
I want you to be 4!

Adversarial Training



I want you to be 3!

Adversarial Training



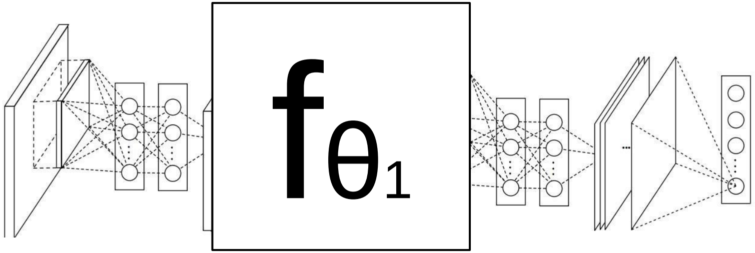
I want you to be 7!

Adversarial Training

Train on adversarial examples (kind of augmentation)

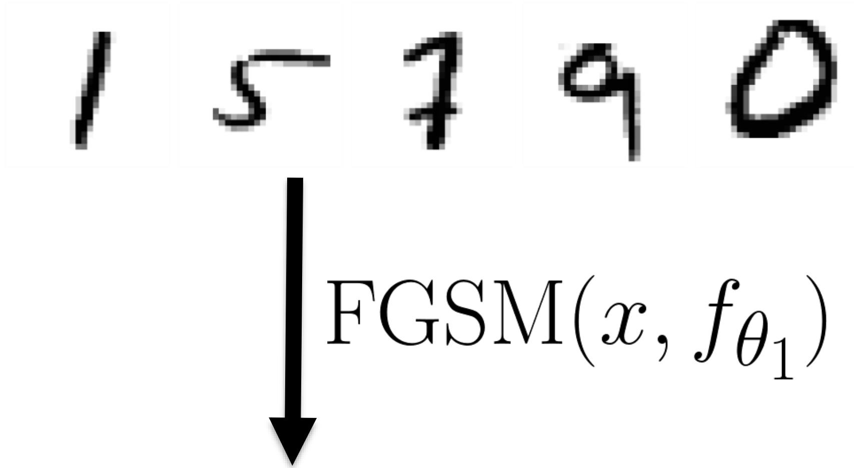
Adversarial Training

Train on adversarial examples (kind of augmentation)

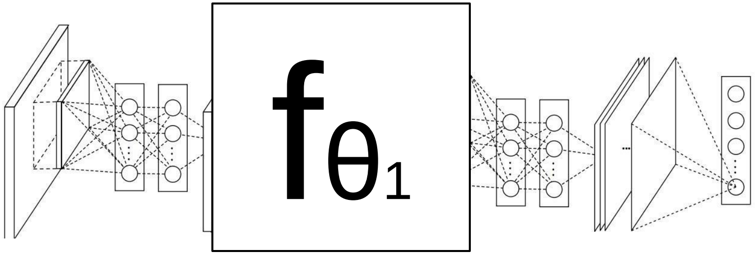


Adversarial Training

Train on adversarial examples (kind of augmentation)



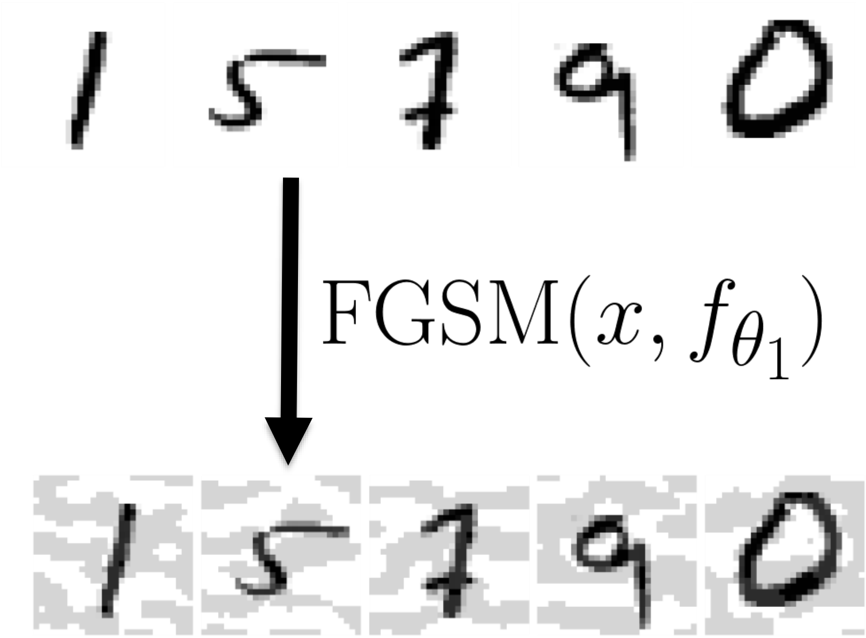
$\mathrm{FGSM}(x, f_{\theta_1})$

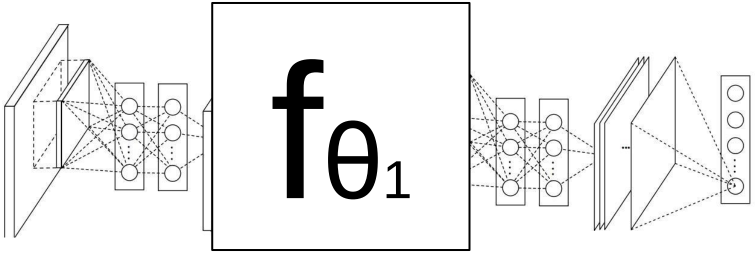


f_{θ_1}

Adversarial Training

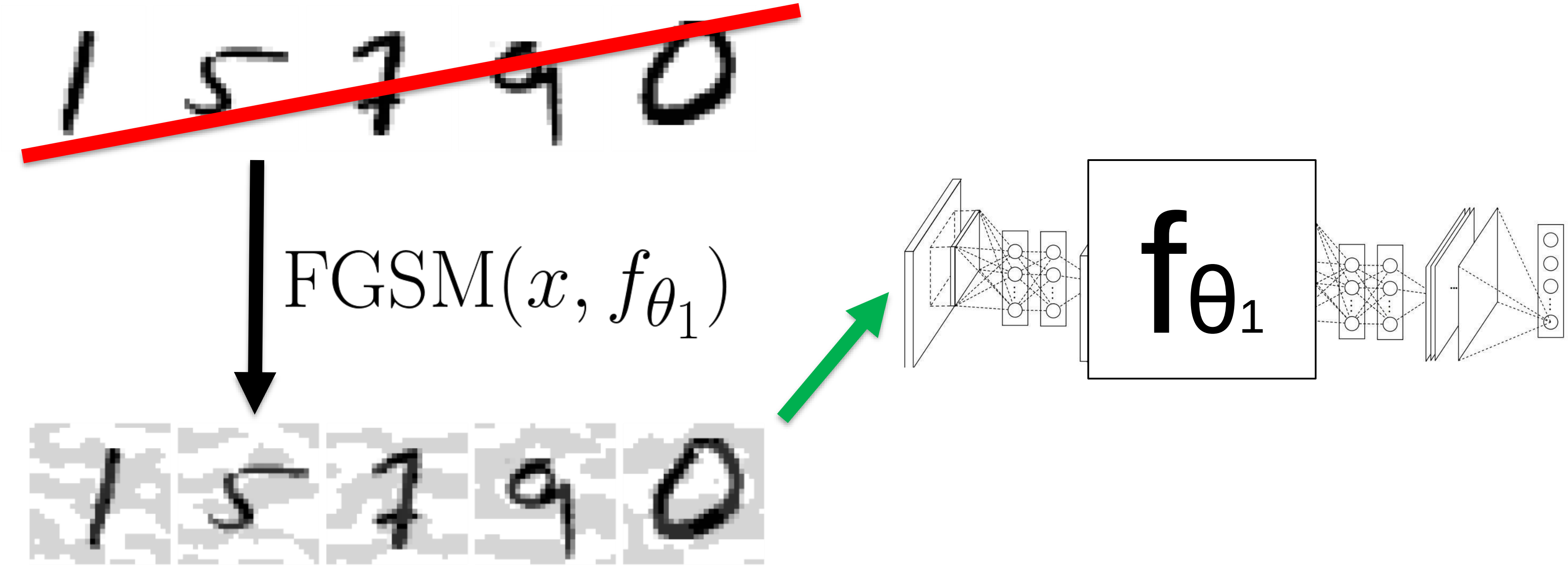
Train on adversarial examples (kind of augmentation)





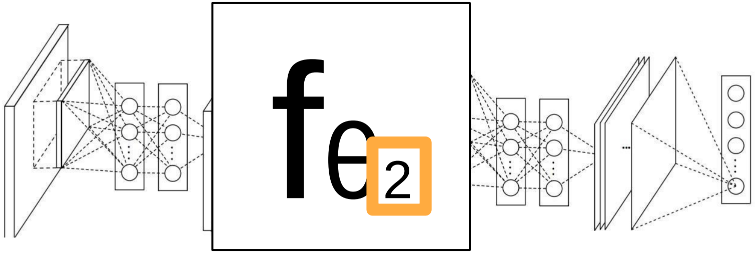
Adversarial Training

Train on adversarial examples (kind of augmentation)



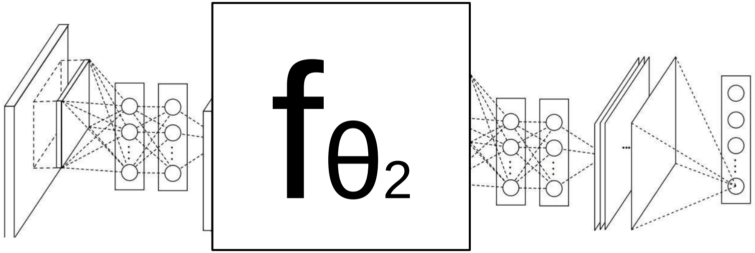
Adversarial Training

Train on adversarial examples (kind of augmentation)



Adversarial Training

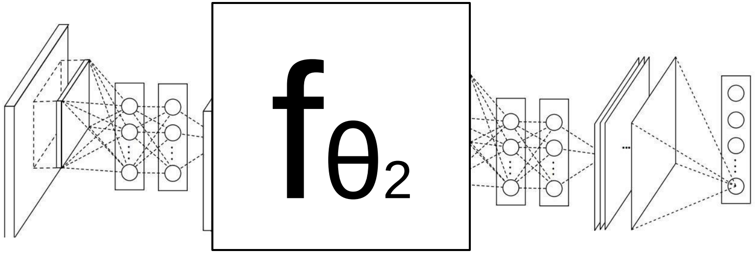
Train on adversarial examples (kind of augmentation)



Adversarial Training

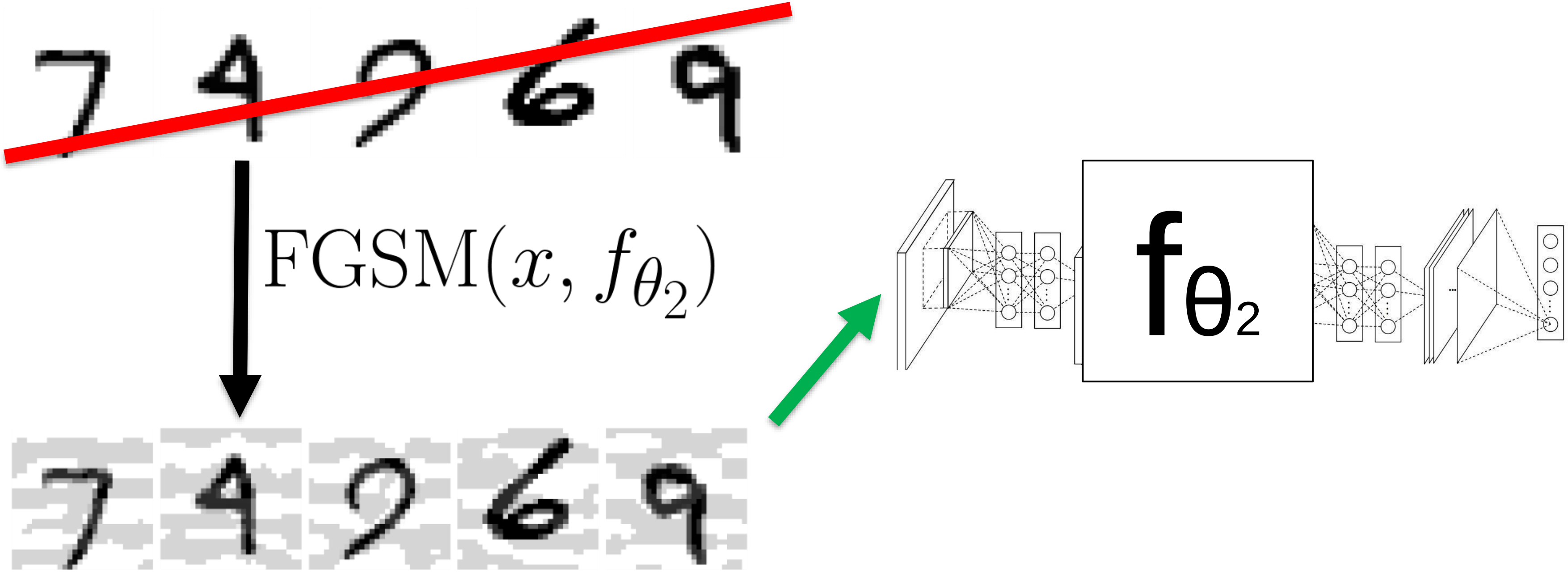
Train on adversarial examples (kind of augmentation)

$$\mathrm{FGSM}(x, f_{\theta_2})$$



Adversarial Training

Train on adversarial examples (kind of augmentation)

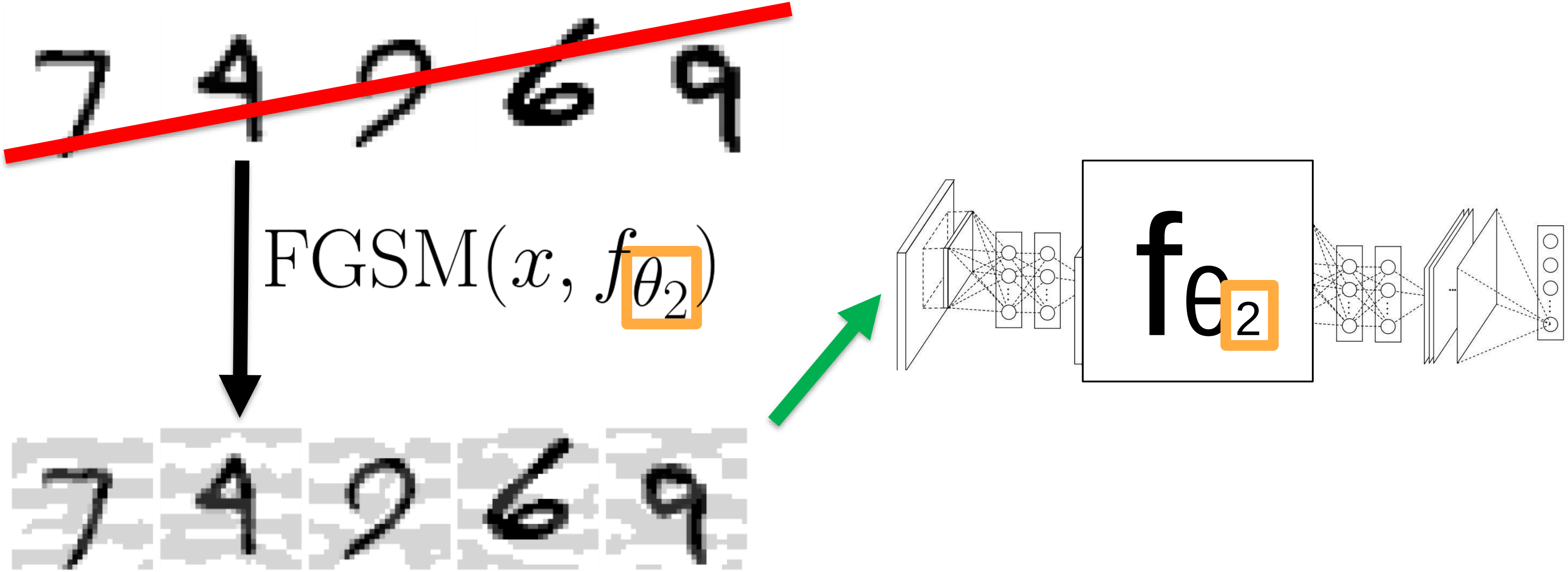


$\mathrm{FGSM}(x, f_{\theta_2})$

f_{θ_2}

Adversarial Training

Train on adversarial examples (kind of augmentation)

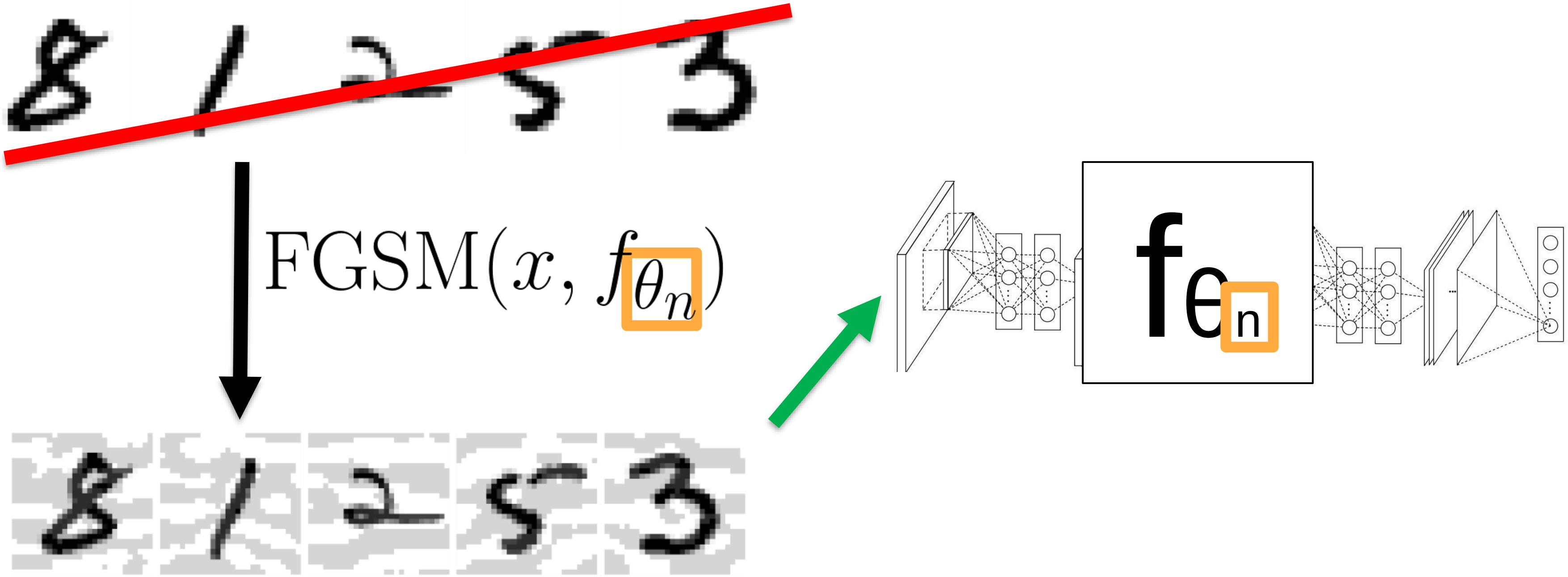


$\mathrm{FGSM}(x, f_{\boxed{\theta_2}})$

$f_{\theta_{\boxed{2}}}$

Adversarial Training

Train on adversarial examples (kind of augmentation)



$\mathrm{FGSM}(x, f_{\boxed{\theta_n}})$

Adversarial Training - MNIST

	Test Accuracy
Standard Training	98.7%

Adversarial Training - MNIST

	Test Accuracy	FGSM Accuracy
Standard Training	98.7%	40.7%

Adversarial Training - MNIST

	Test Accuracy	FGSM Accuracy
Standard Training	98.7%	40.7%
Adv. Training (FGSM)	97.2%	94.0%

Adversarial Training - MNIST

	Test Accuracy	FGSM Accuracy
Standard Training	98.7%	40.7%
Adv. Training (FGSM)	97.2%	94.0%

Adversarial Training - MNIST

	Test Accuracy	FGSM Accuracy
Standard Training	98.7%	40.7%
Adv. Training (FGSM)	97.2%	94.0%

Did we solve the problem?

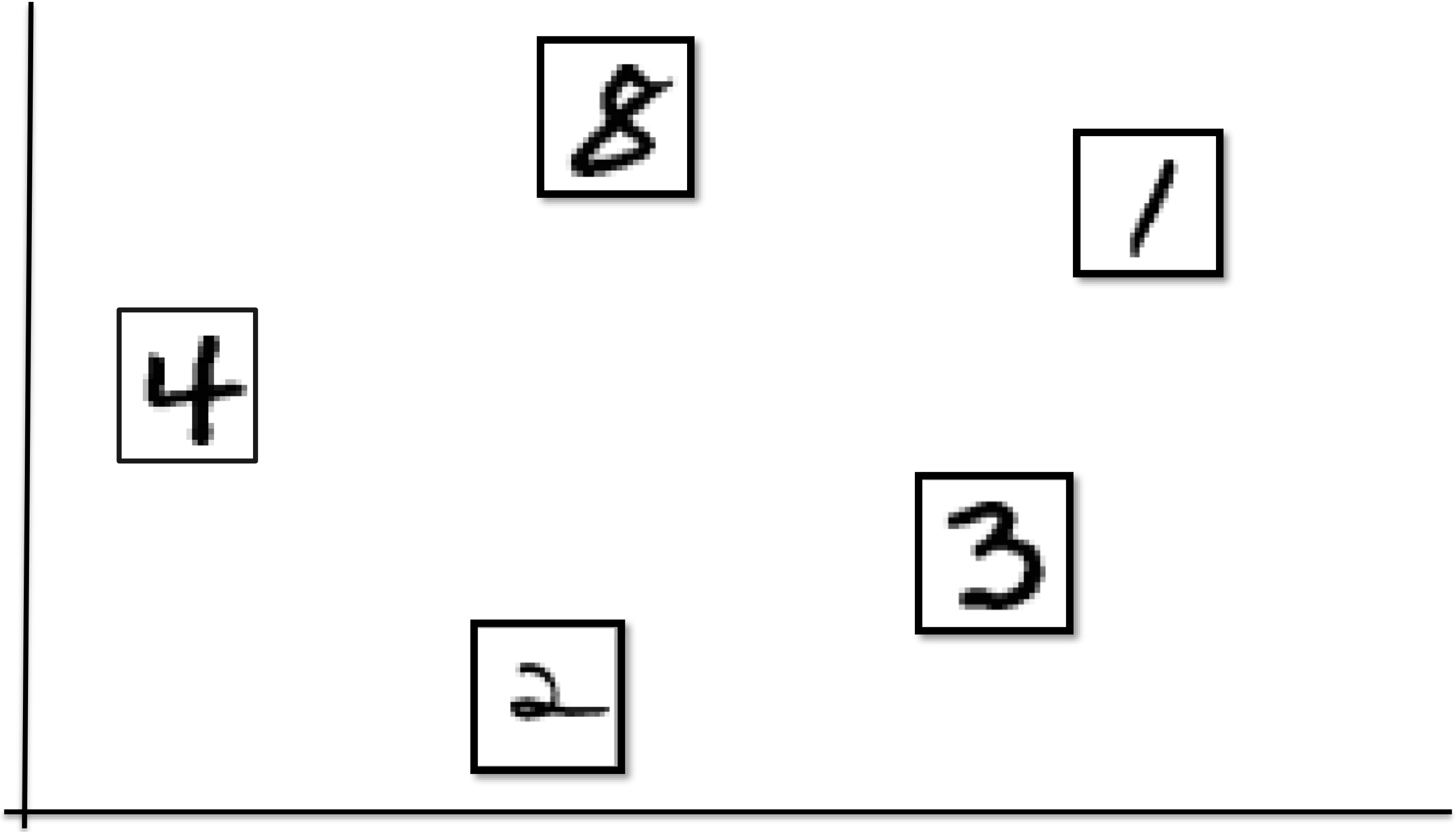
Outline

- See Adversarial Example
- Discuss what they are
- How to attack: FGSM
- How to defend: Adversarial training (AT)
    - Next: a better picture of AT (pictorially/optimization)
- Learn about properties and advantages

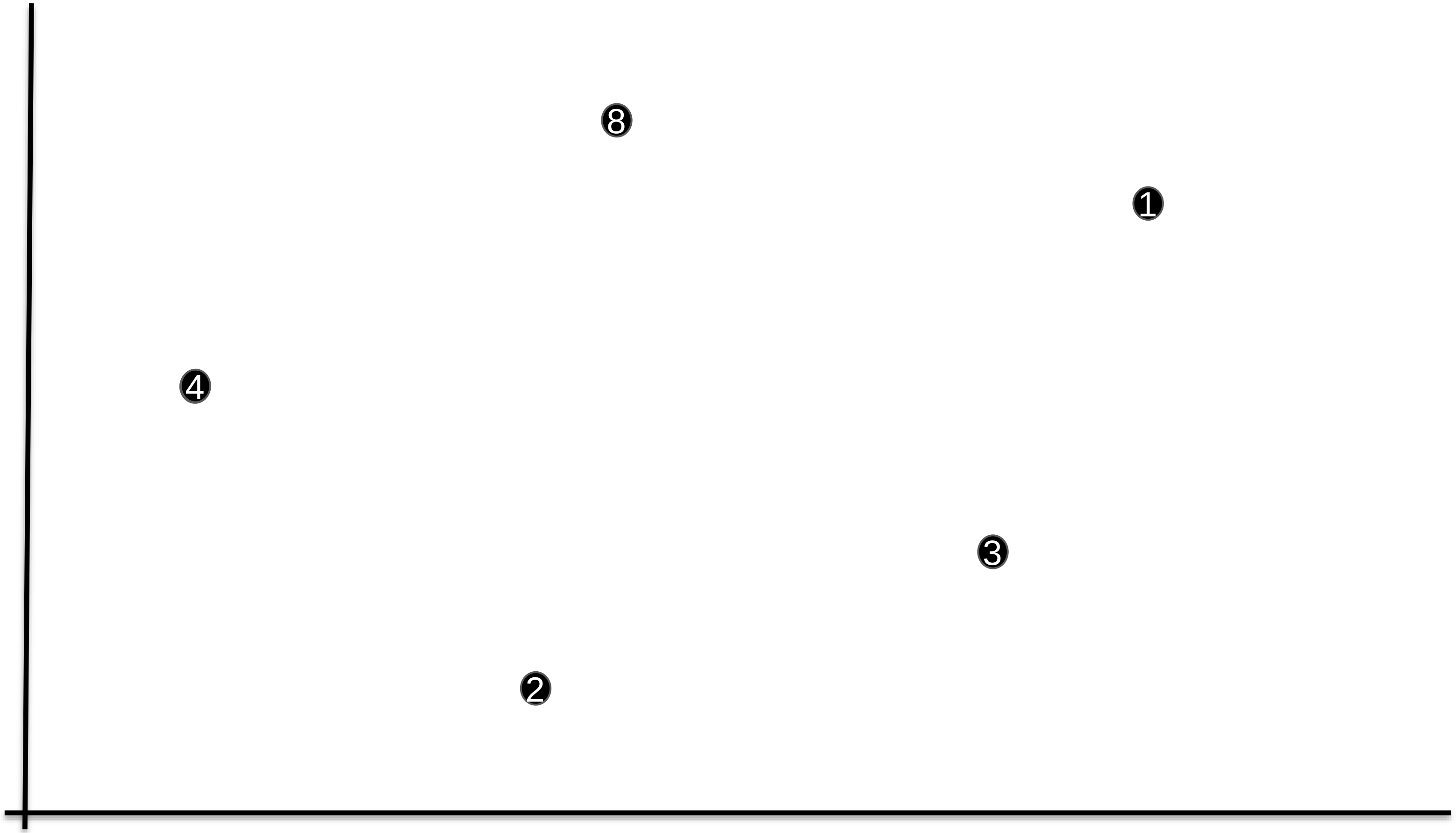
Outline

- See Adversarial Example
- Discuss what they are
- How to attack: FGSM
- How to defend: Adversarial training (AT)
    - Next: a better picture of AT (pictorially/optimization)
- Learn about properties and advantages

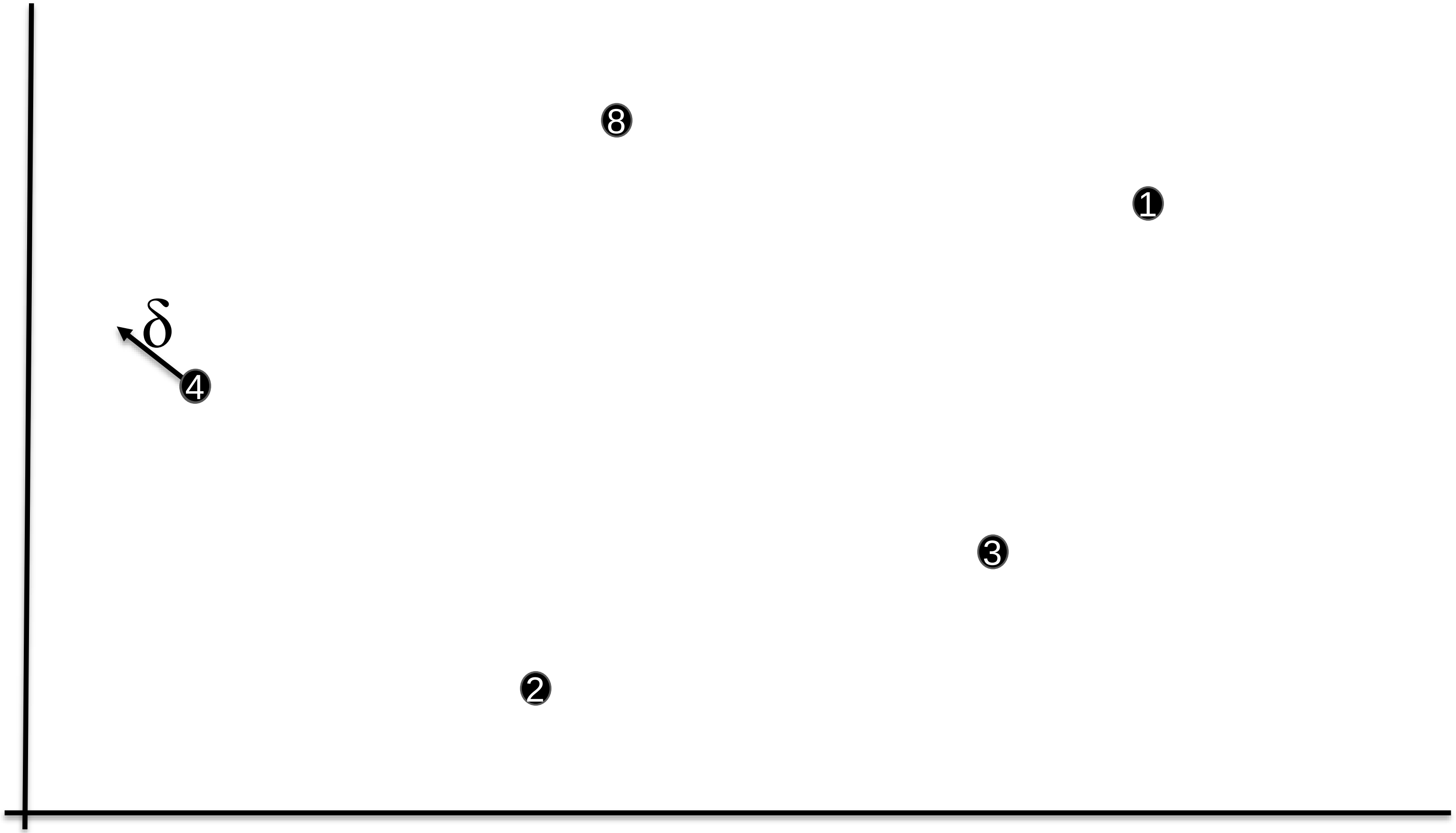
Perturbation Attack (pictorially)



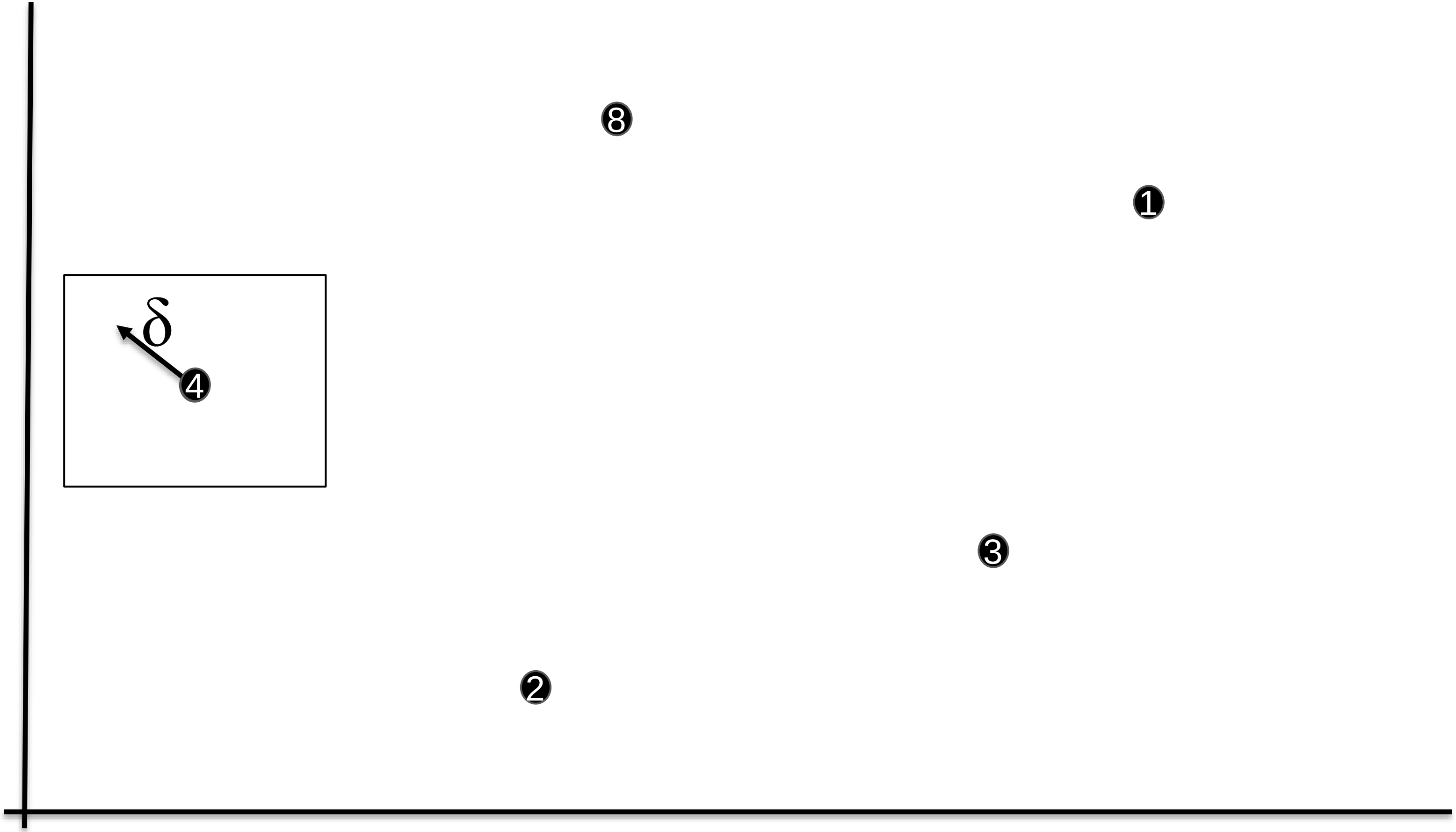
Perturbation Attack (pictorially)



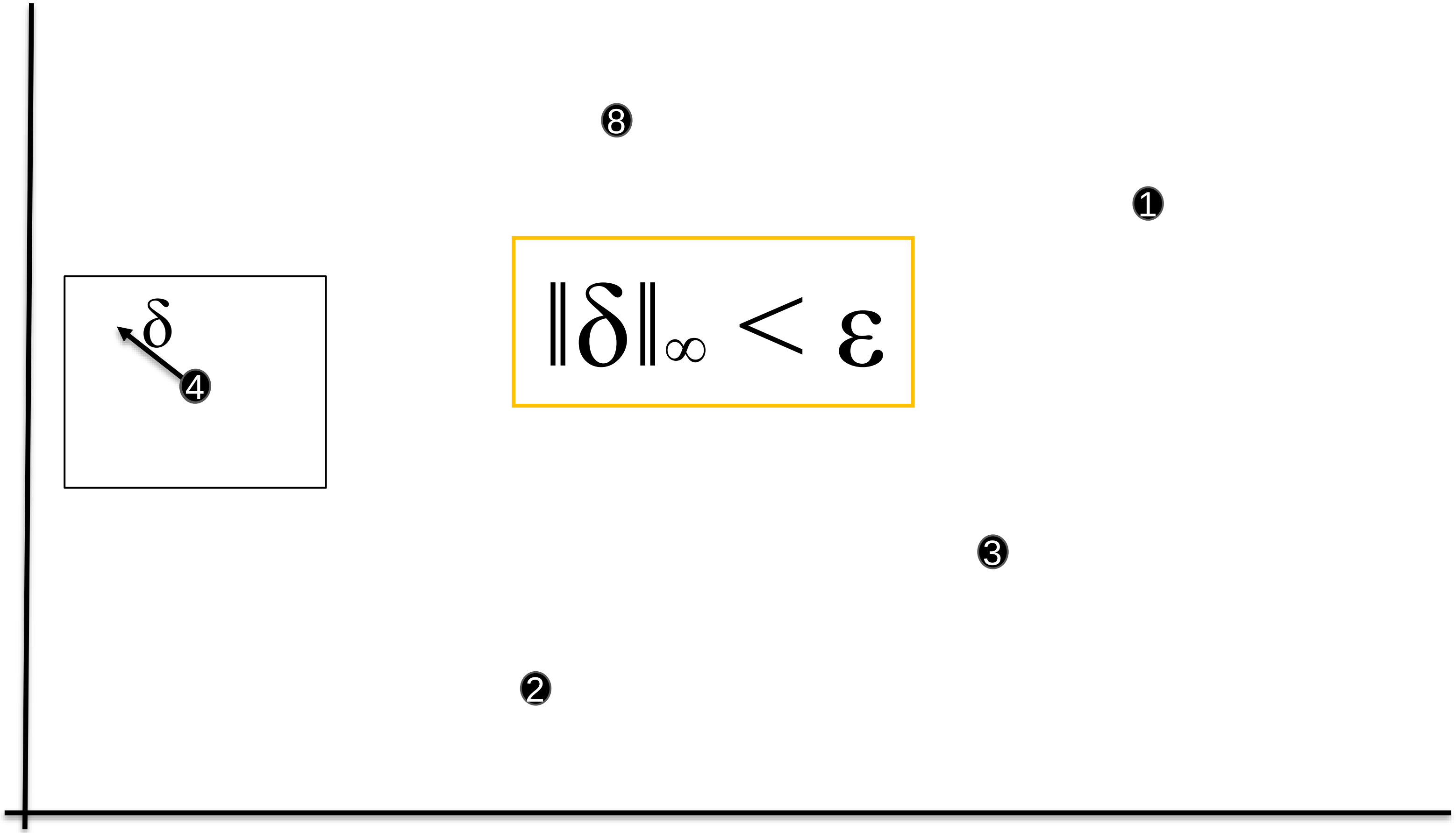
Perturbation Attack (pictorially)



Perturbation Attack (pictorially)

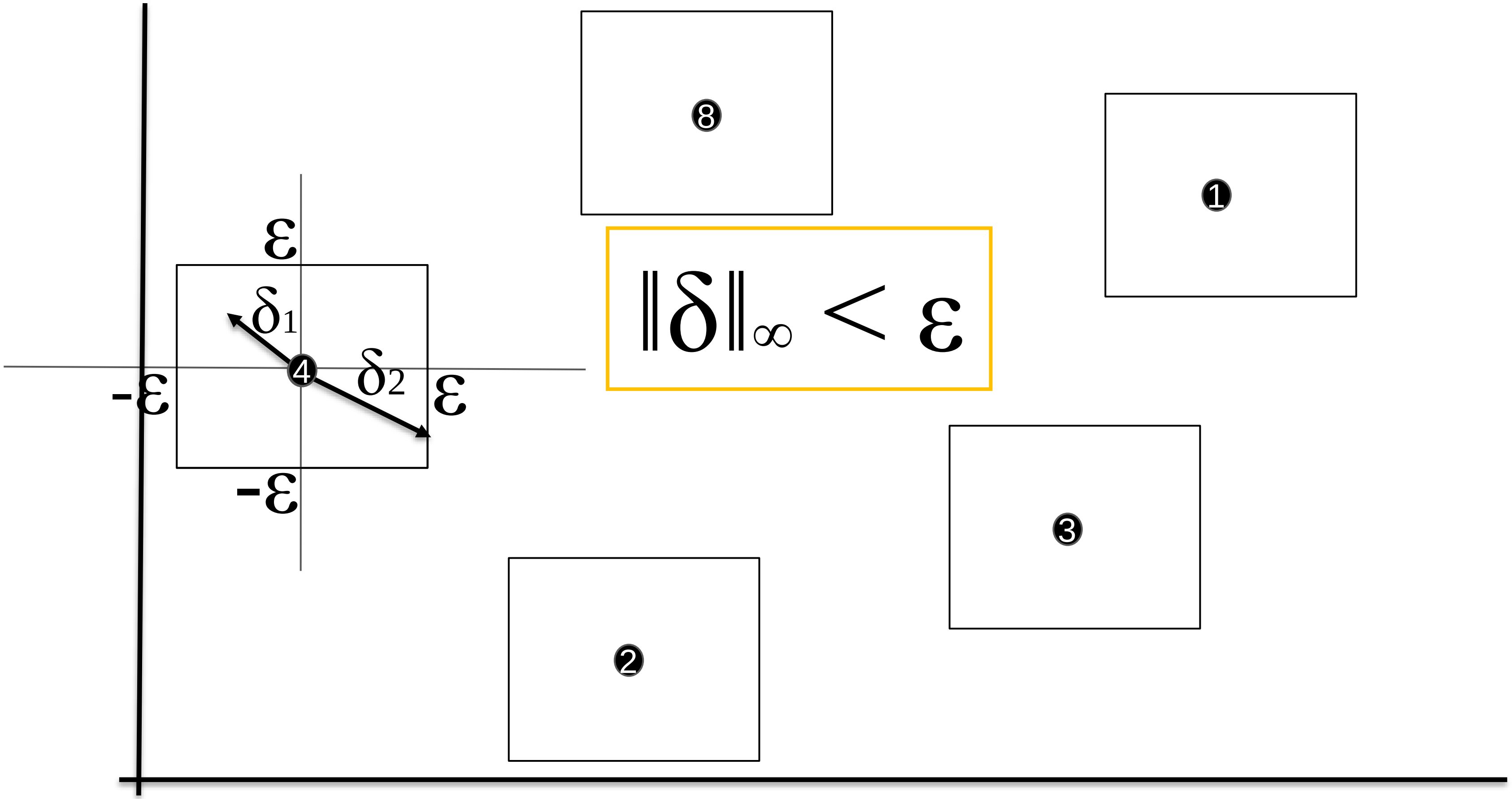


Perturbation Attack (pictorially)



$$\|\delta\|_\infty < \varepsilon$$

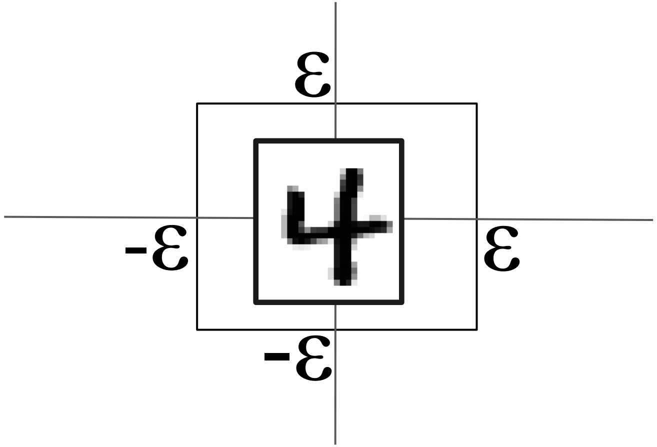
Perturbation Attack (pictorially)



$$\|\delta\|_\infty < \varepsilon$$

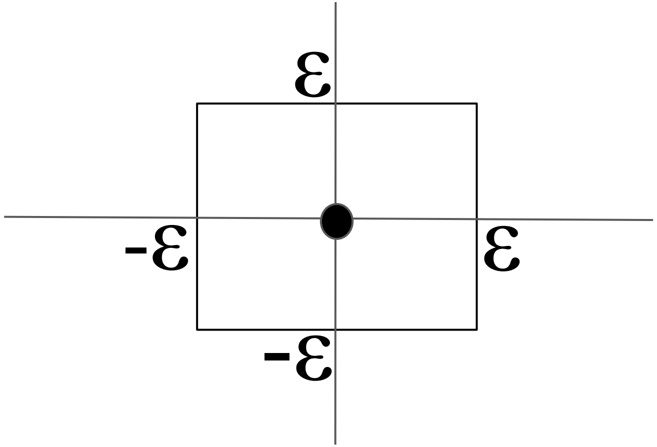
Perturbation Attack (pictorially)

FGSM



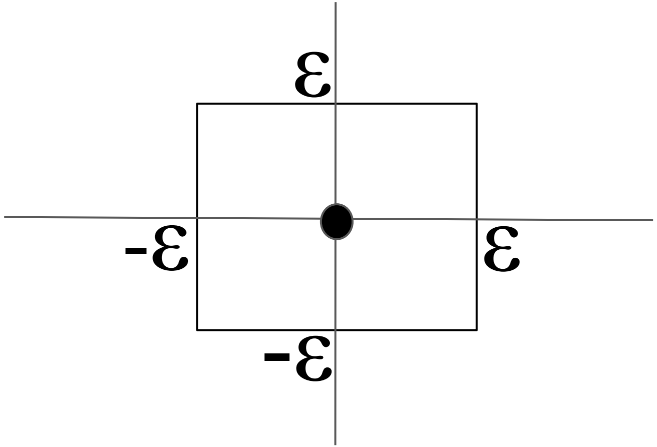
Perturbation Attack (pictorially)

FGSM



Perturbation Attack (pictorially)

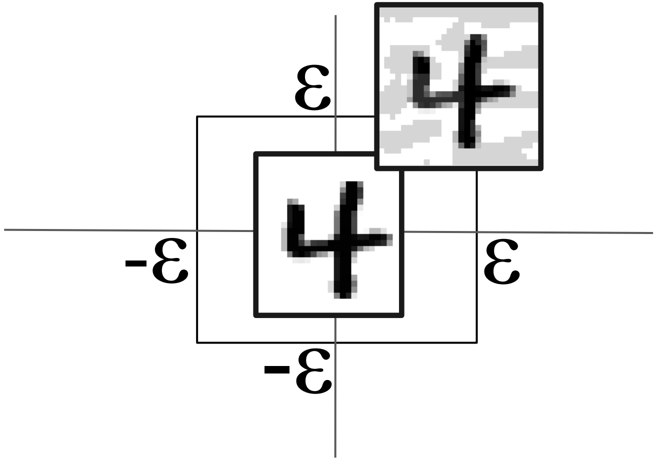
FGSM



$$\mathbf{X}^{\text{adv}} = \mathbf{X} + \epsilon \ \text{sgn}(\nabla_X L(\mathbf{X}, y_{\text{true}}))$$

Perturbation Attack (pictorially)

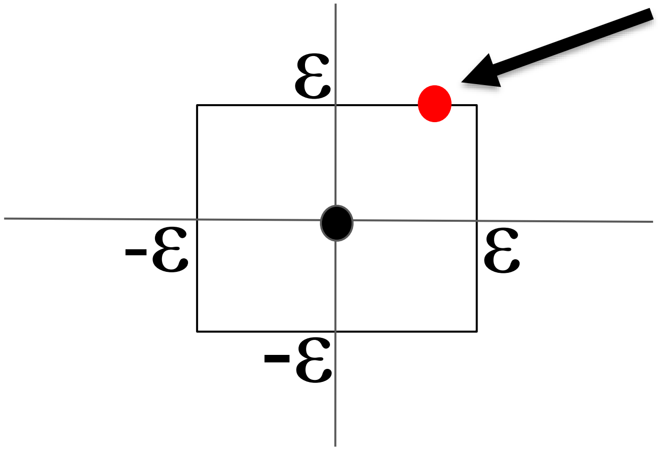
FGSM



$$\mathbf{X}^{\text{adv}} = \mathbf{X} + \epsilon \,\text{sgn}(\nabla_X L(\mathbf{X}, y_{\text{true}}))$$

Perturbation Attack (pictorially)

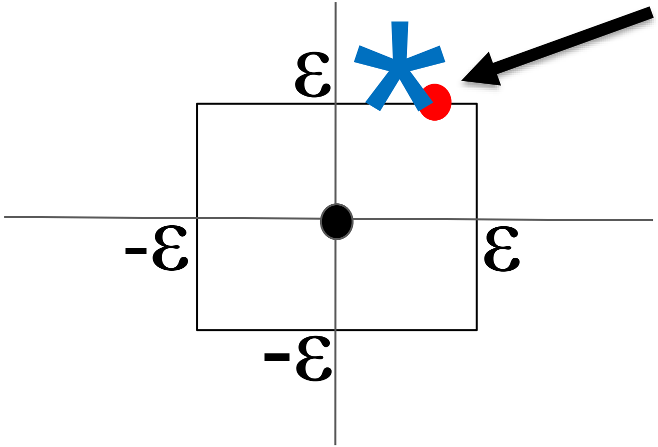
FGSM



Possible AE (found by FGSM)

Perturbation Attack (pictorially)

FGSM

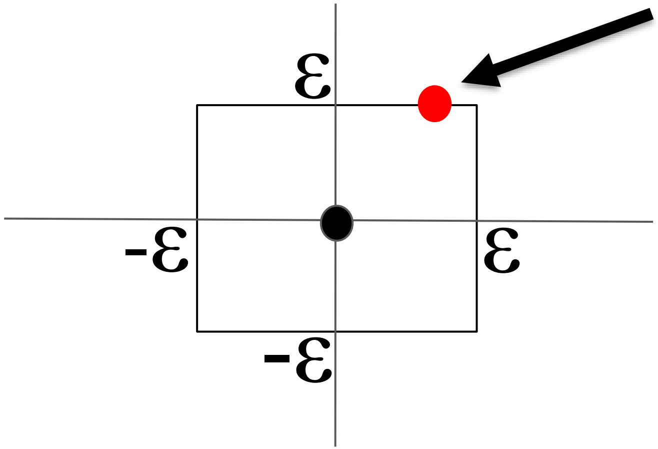


Possible AE (found by FGSM)

* dot should have been lying on one of the corners..

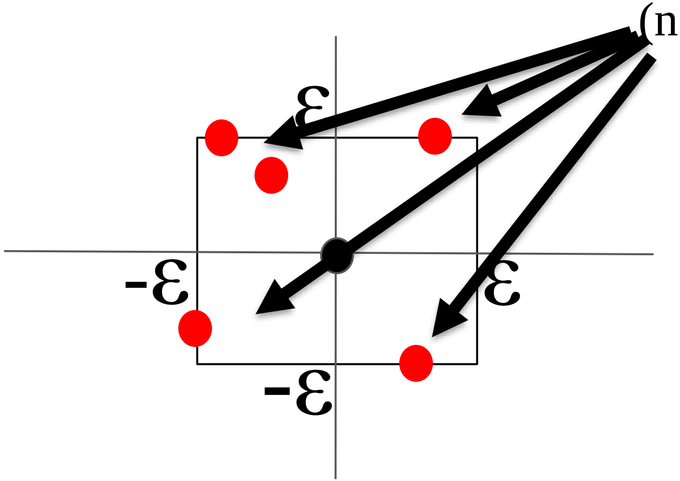
Perturbation Attack (pictorially)

FGSM



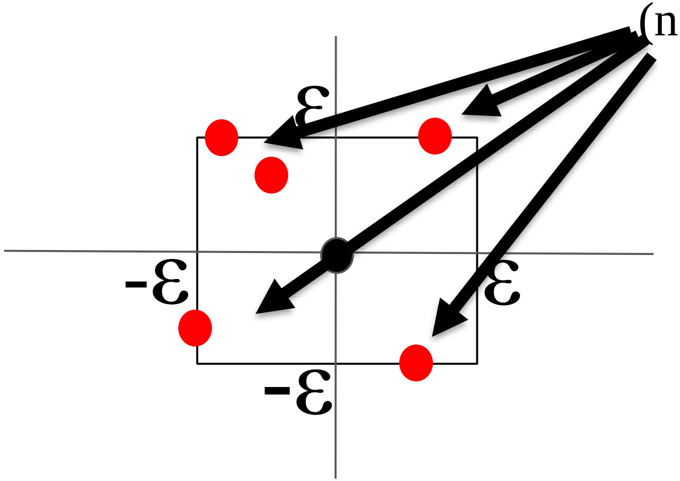
Possible AE (found by FGSM)

Perturbation Attack (pictorially)



Possible AEs
(need to be found)

Perturbation Attack (pictorially)

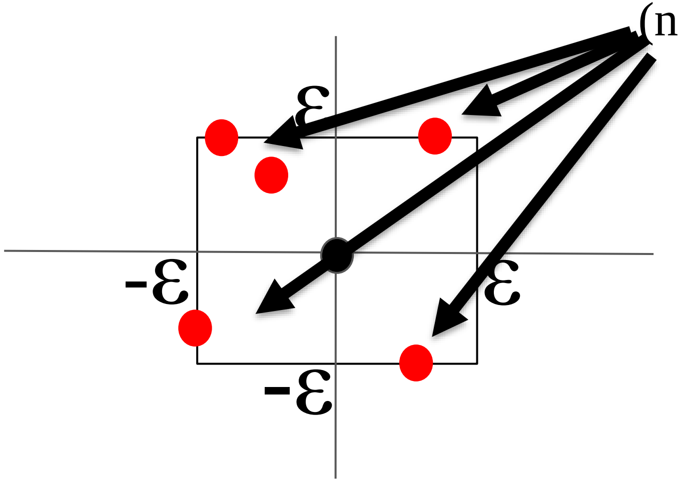


Possible AEs
(need to be found)

<u>“The Game” of AT:</u>

Defender: defend in box

Perturbation Attack (pictorially)



Possible AEs
(need to be found)

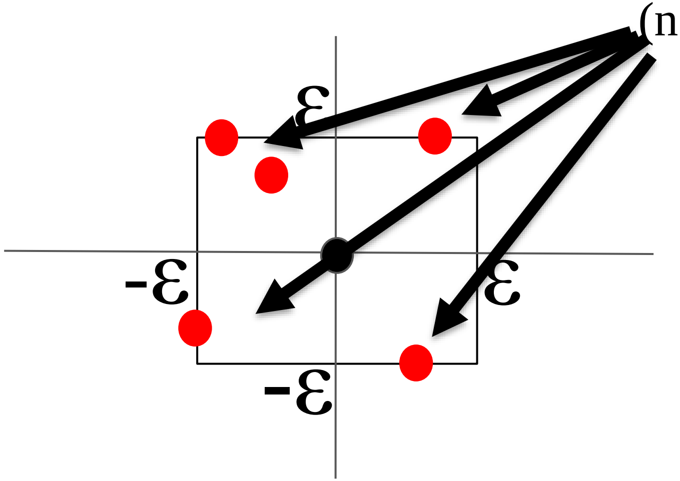
<u>"The Game" of AT:</u>

Defender: defend in box

Attacker: find AE in box

Perturbation Attack (pictorially)

Possible AEs
(need to be found)



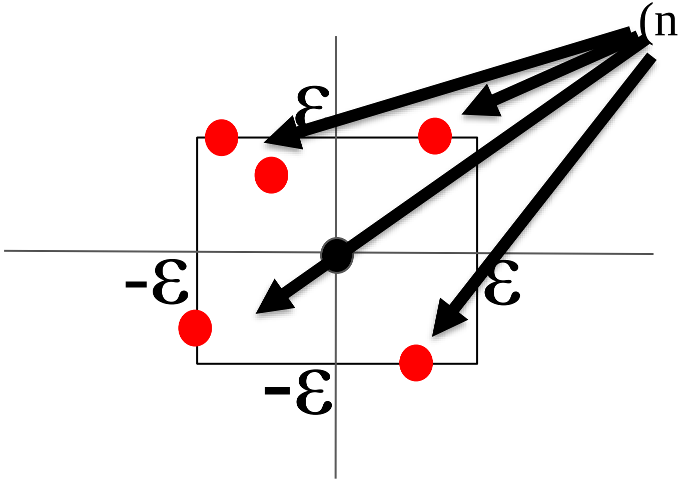
<u>“The Game” of AT:</u>

Defender: defend in box

Attacker: find AE in box

Coming
Up next:

Perturbation Attack (optimization)



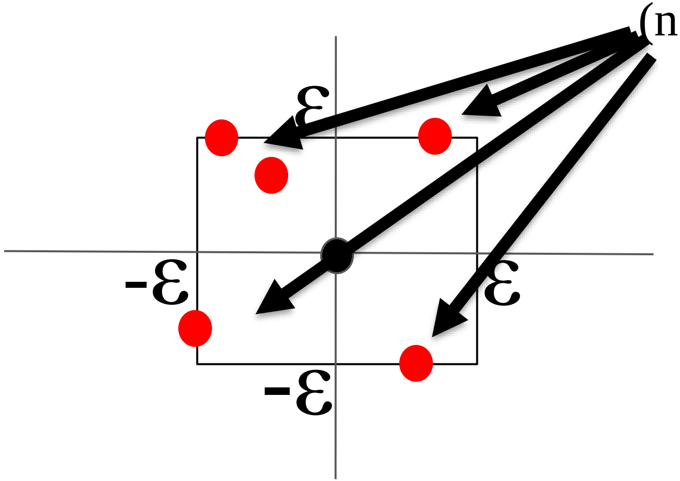
Possible AEs
(need to be found)

<u>“The Game” of AT:</u>

Defender: defend in box

Attacker: find AE in box

Perturbation Attack (optimization)



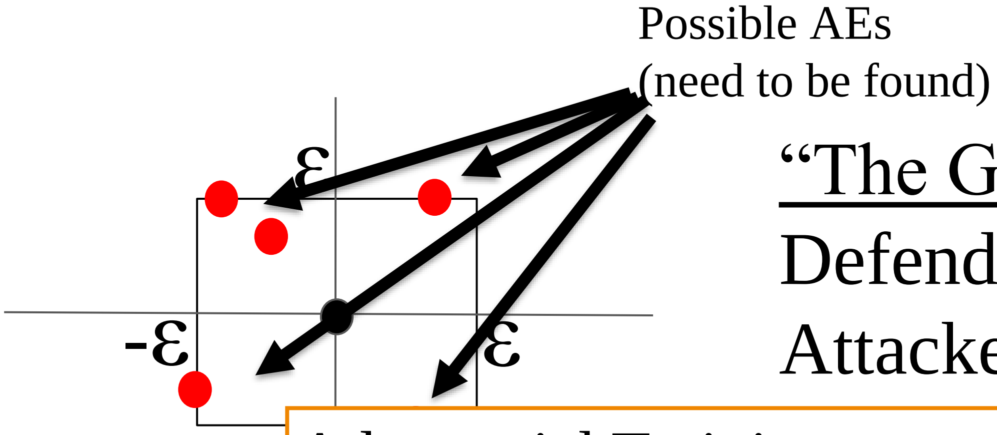
Possible AEs
(need to be found)

<u>“The Game” of AT:</u>

Defender: defend in box

Attacker: find AE in box

Perturbation Attack (optimization)



<u>“The Game” of AT:</u>

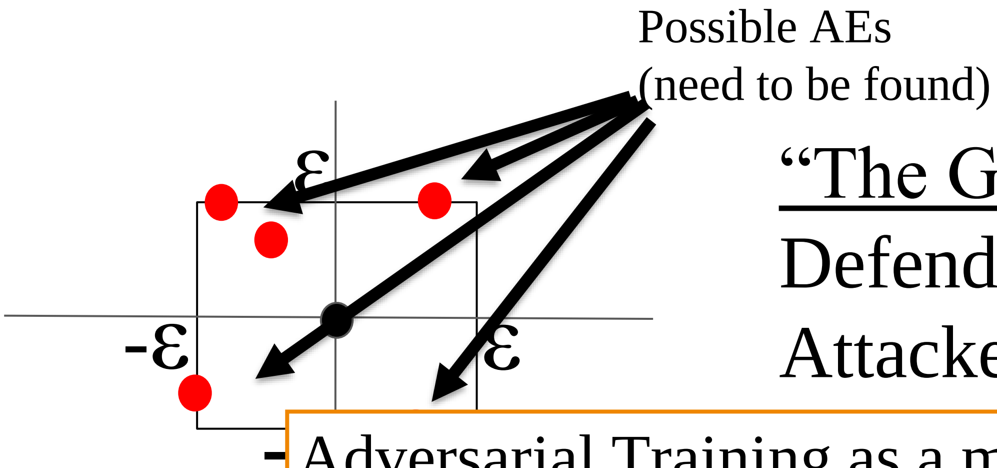
Defender: defend in box

Attacker: find AE in box

Adversarial Training as a min-max optimization problem:

Towards Deep Learning Models Resistant to Adversarial Attacks, Madry et al. 2018

Perturbation Attack (optimization)



Possible AEs
(need to be found)

<u>“The Game” of AT:</u>

Defender: defend in box

Attacker: find AE in box

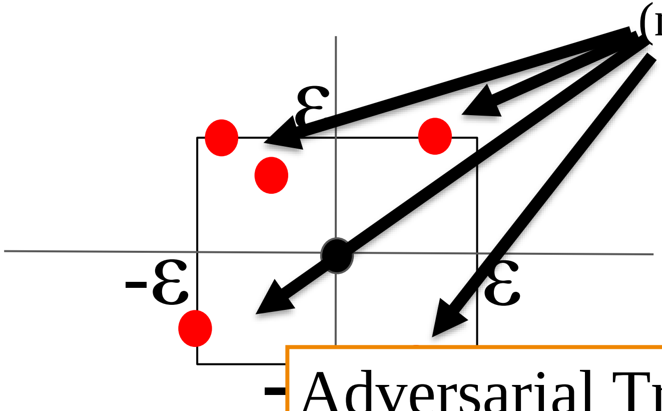
Adversarial Training as a min-max optimization problem:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D}[\max_{\delta \in S} L(f_{\theta}(x+\delta), y)]$$

Towards Deep Learning Models Resistant to Adversarial Attacks, Madry et al. 2018

Perturbation Attack (optimization)

Possible AEs
(need to be found)



“The Game” of AT:

Defender: defend in box

Attacker: find AE in box

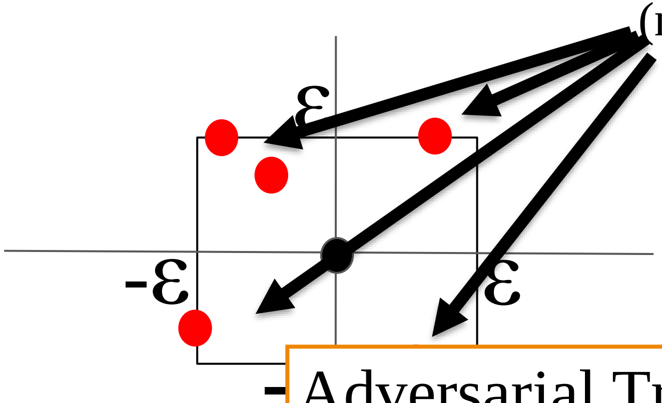
Adversarial Training as a min-max optimization problem:

$$\min_{\theta} \overbrace{\mathbb{E}_{(x,y)\sim D}[\quad L(f_{\theta}(x \quad), y)]}^{\text{Standard Loss}}$$

Towards Deep Learning Models Resistant to Adversarial Attacks, Madry et al. 2018

Perturbation Attack (optimization)

Possible AEs
(need to be found)



<u>“The Game” of AT:</u>

Defender: defend in box

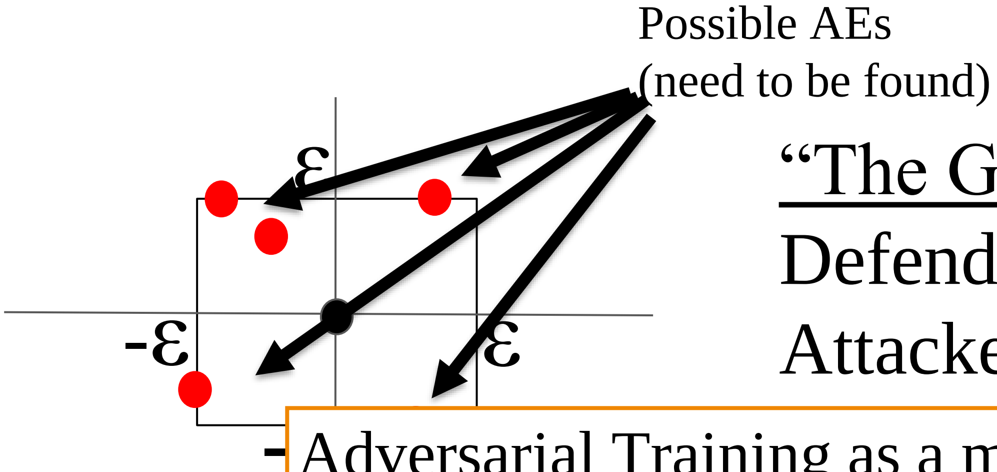
Attacker: find AE in box

Adversarial Training as a min-max optimization problem:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D}[\qquad L(f_{\theta}(x \qquad), y)]$$

Towards Deep Learning Models Resistant to Adversarial Attacks, Madry et al. 2018

Perturbation Attack (optimization)


Possible AEs
(need to be found)

<u>"The Game" of AT:</u>

Defender: defend in box

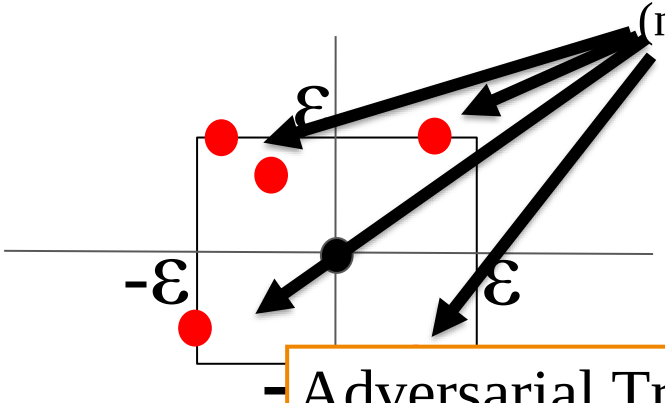
Attacker: find AE in box

Adversarial Training as a min-max optimization problem:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D}[\max_{\delta \in S} L(f_{\theta}(x+\delta), y)]$$

Towards Deep Learning Models Resistant to Adversarial Attacks, Madry et al. 2018

Perturbation Attack (optimization)



Possible AEs
(need to be found)

“The Game” of AT:

Defender: defend in box

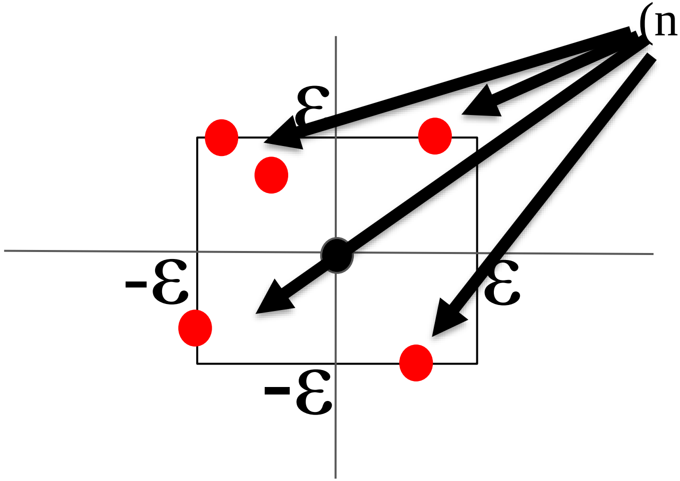
Attacker: find AE in box

Adversarial Training as a min-max optimization problem:

$$\min_{\theta} \overbrace{\mathbb{E}_{(x,y)\sim D}\left[\max_{\delta\in S} L(f_\theta(x+\delta), y)\right]}^{\text{Adversarial Loss}}$$

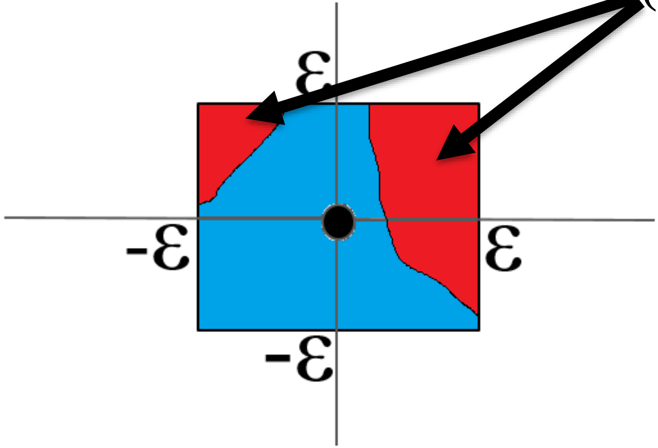
Towards Deep Learning Models Resistant to Adversarial Attacks, Madry et al. 2018

Perturbation Attack (illustrations)



Possible AEs
(need to be found)

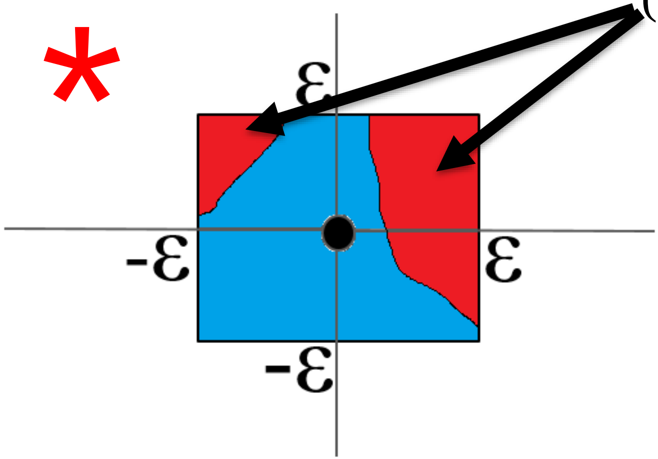
Perturbation Attack (illustrations)



Possible AEs
(need to be found)

Perturbation Attack (illustrations)

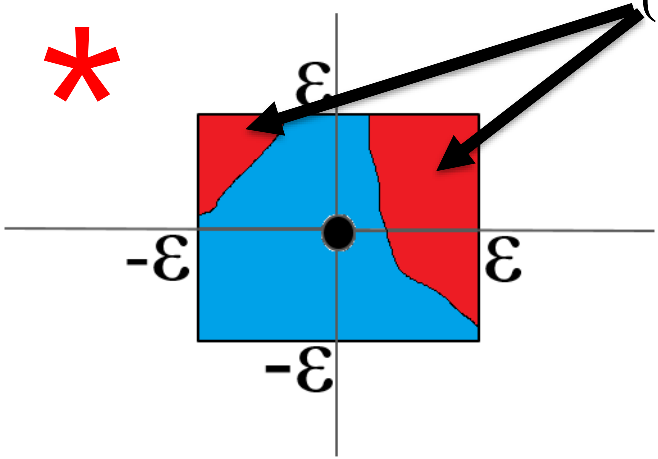
*



Possible AEs
(need to be found)

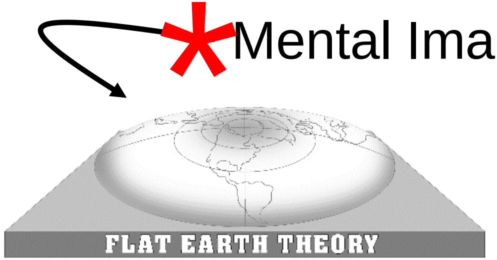
*Mental image alert! (“experimental” mental images could be horribly misleading)

Perturbation Attack (illustrations)



Possible AEs
(need to be found)

*Mental Image

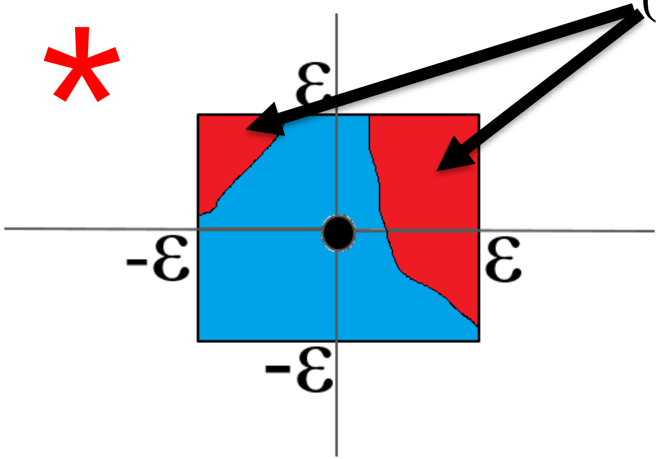


*Mental image alert! (“experimental” mental images could be horribly misleading)

Perturbation Attack (illustrations)

Possible AEs
(n

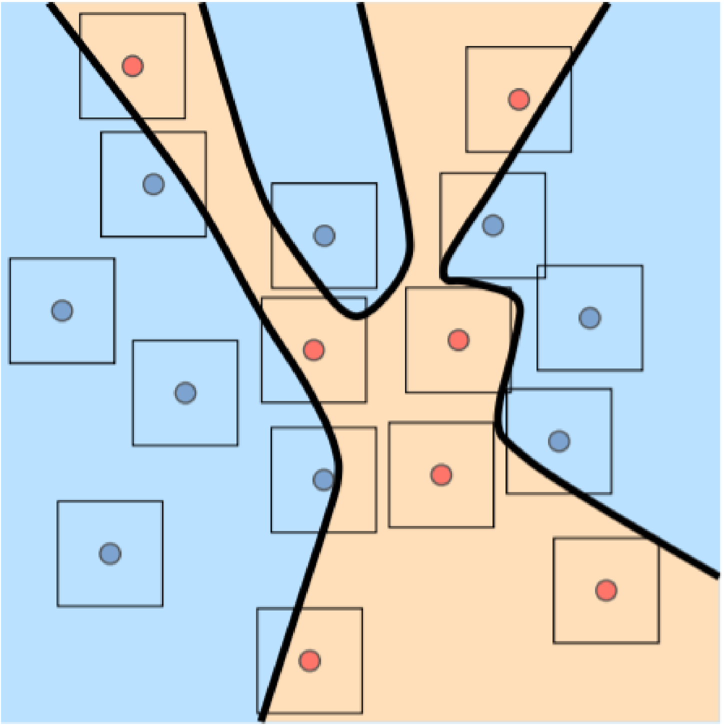
*



*Mental Image

*Mental image alert! (“experimental” mental images could be horribly misleading)

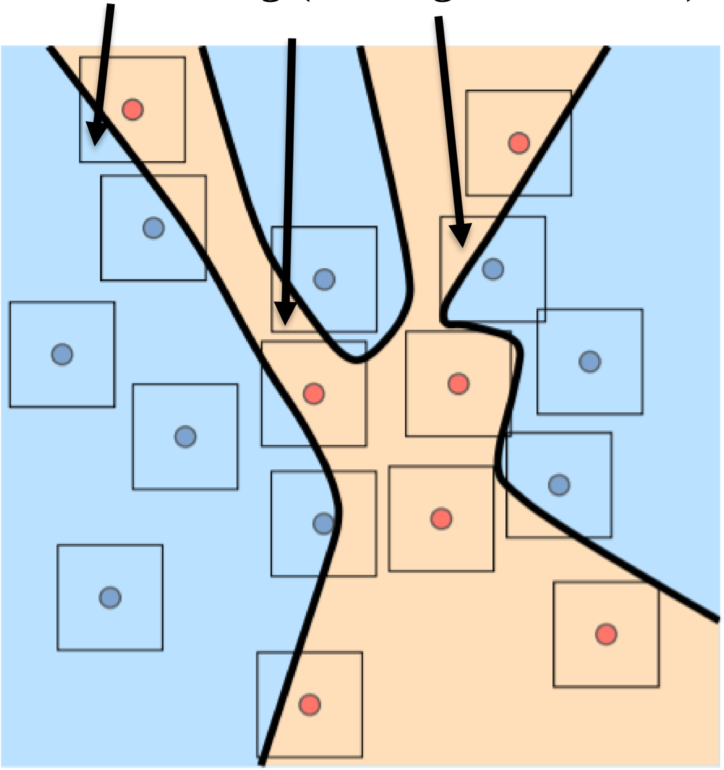
Perturbation Attack (better illustrations)



source: Atzmon et al. 2019, “Controlling Neural Level Sets”

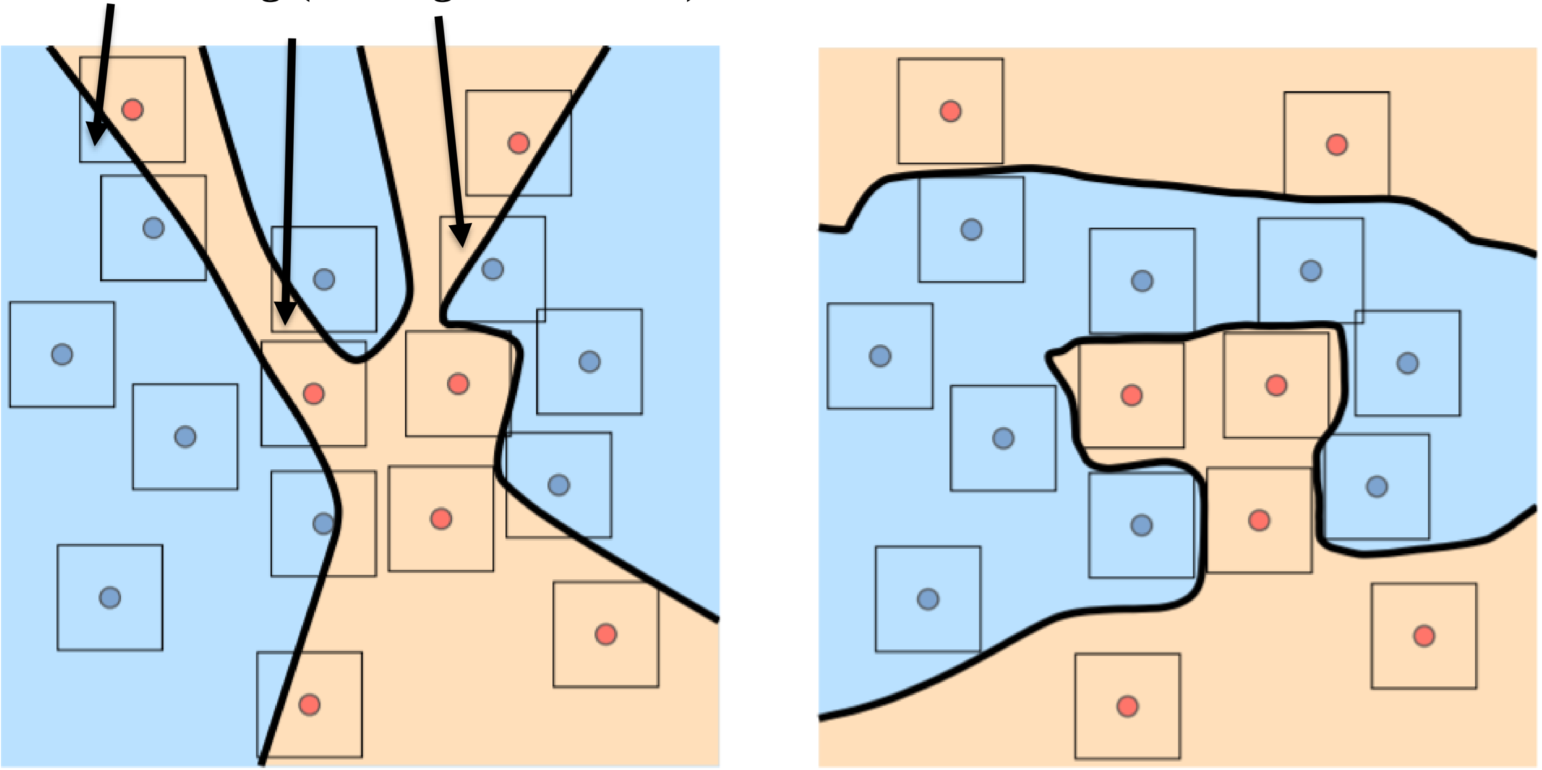
Perturbation Attack (better illustrations)

AEs lurking (waiting to be found)



source: Atzmon et al. 2019, “Controlling Neural Level Sets”

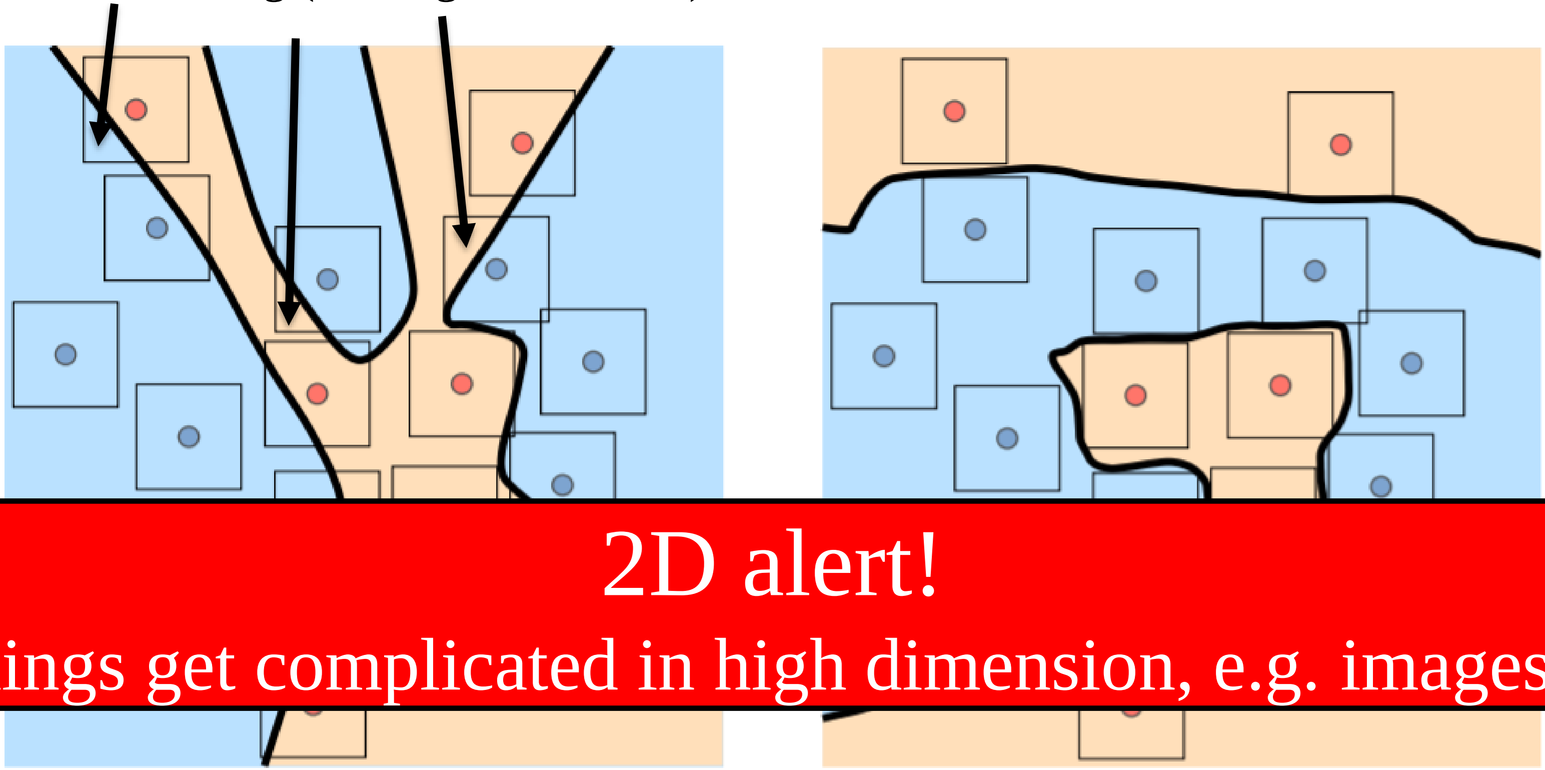
Perturbation Attack (better illustrations)



AEs lurking (waiting to be found)

source: Atzmon et al. 2019, “Controlling Neural Level Sets”

Perturbation Attack (better illustrations)



AEs lurking (waiting to be found)

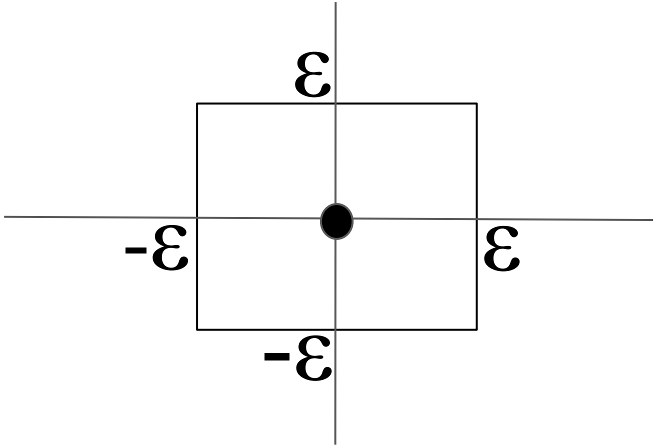
2D alert!

(Things get complicated in high dimension, e.g. images…)

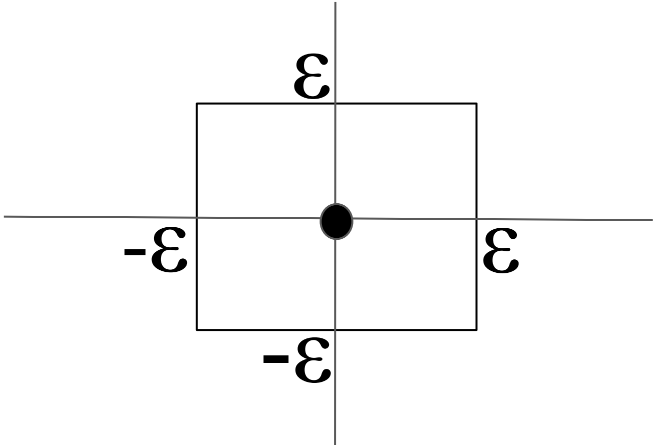
source: Atzmon et al. 2019, “Controlling Neural Level Sets”

PGD (Projected Gradient Descent)

FGSM

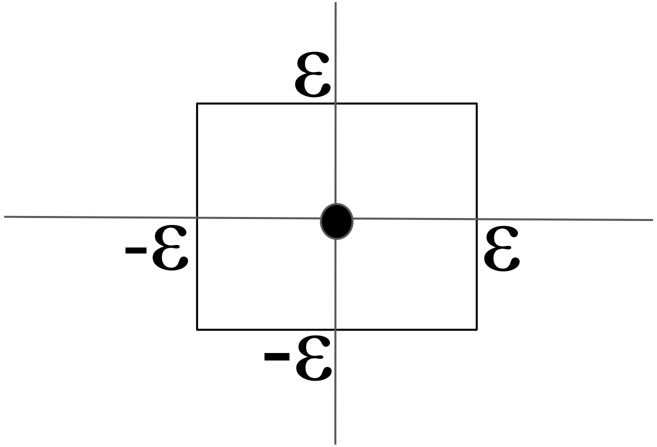


PGD

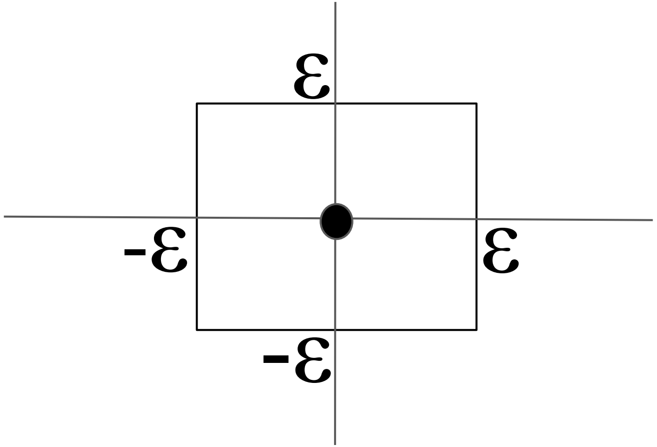


PGD (a.k.a Iterated-GSM)

FGSM

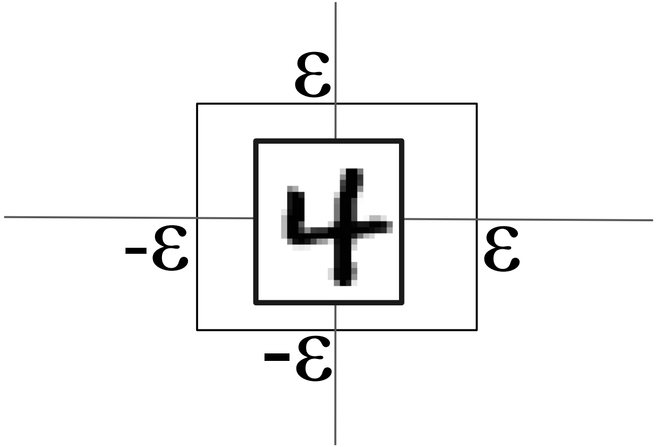


PGD

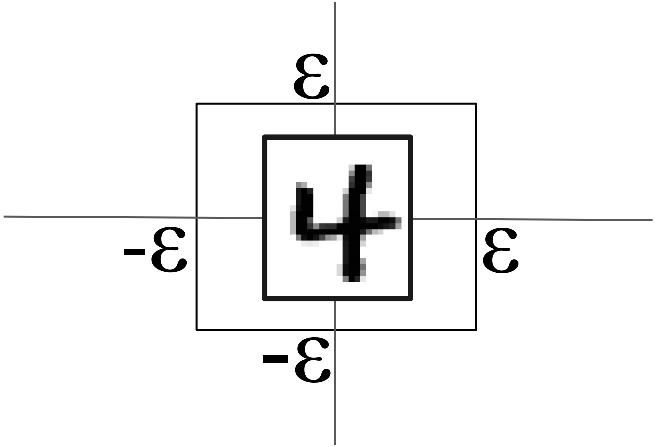


PGD (a.k.a Iterated-GSM)

<u>FGSM</u>

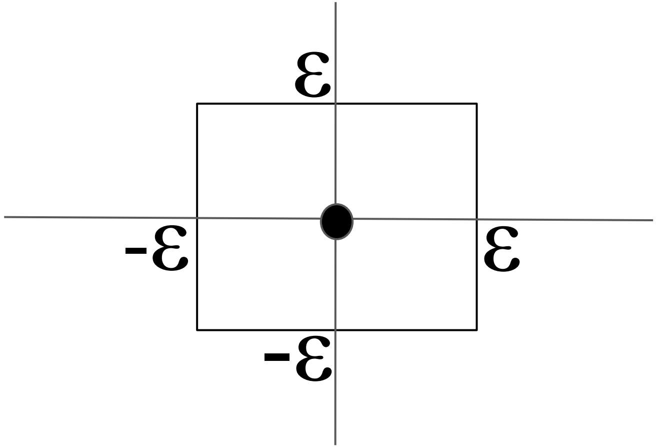


<u>PGD</u>

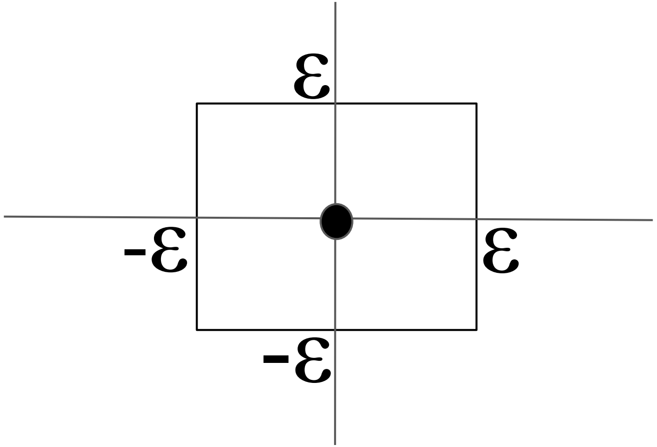


PGD (a.k.a Iterated-GSM)

FGSM

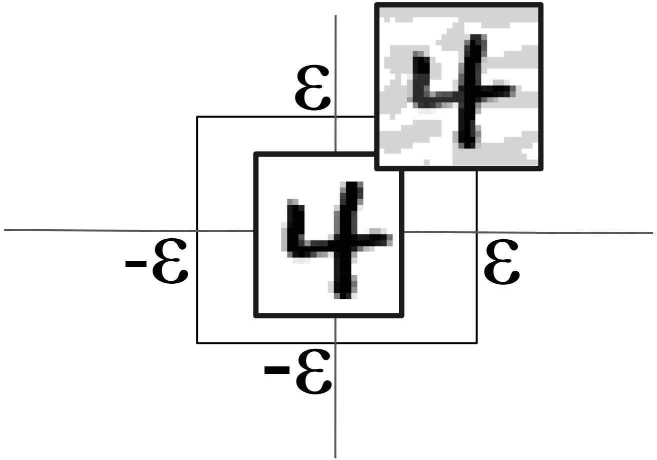


PGD

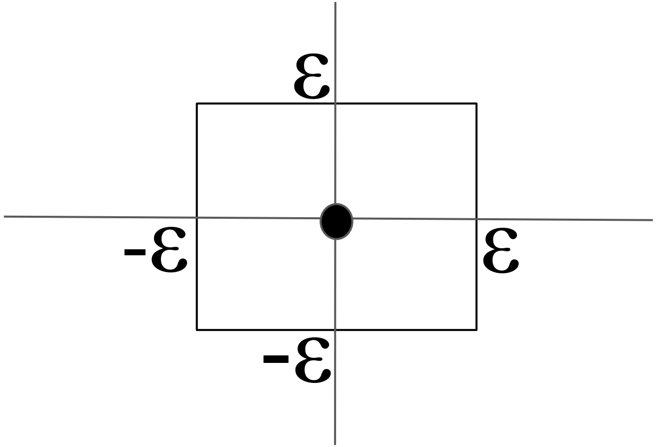


PGD (a.k.a Iterated-GSM)

FGSM

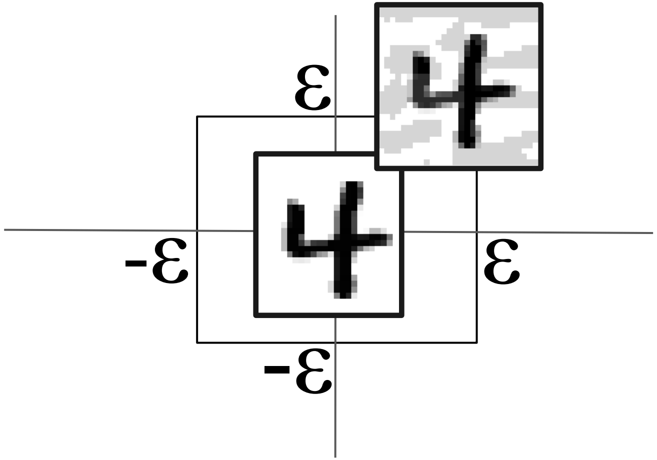


PGD

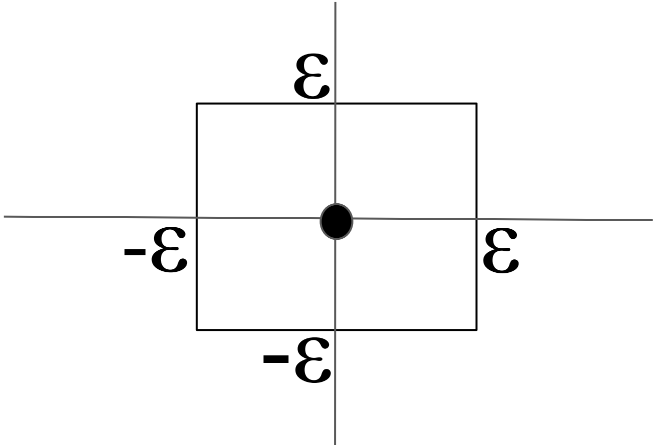


PGD (a.k.a Iterated-GSM)

FGSM

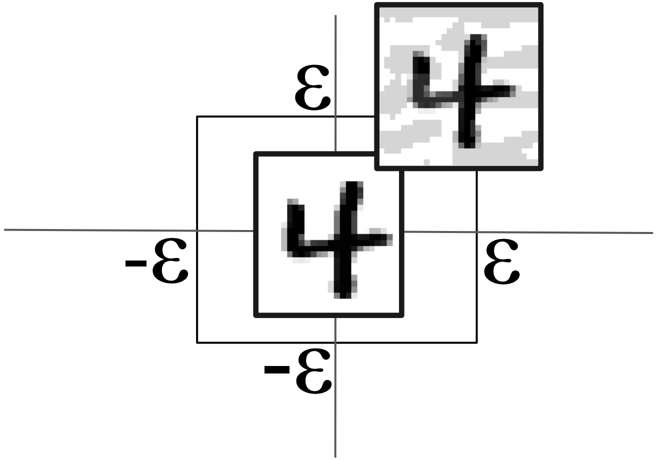


PGD

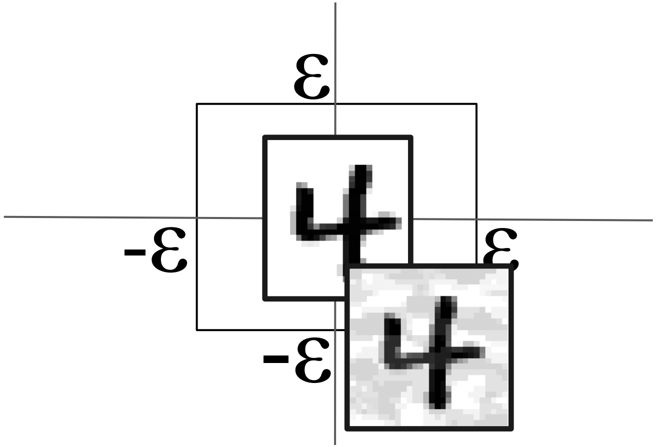


PGD (a.k.a Iterated-GSM)

FGSM

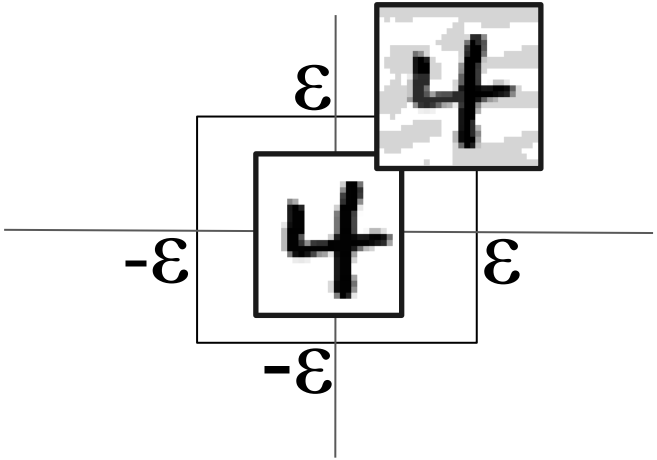


PGD

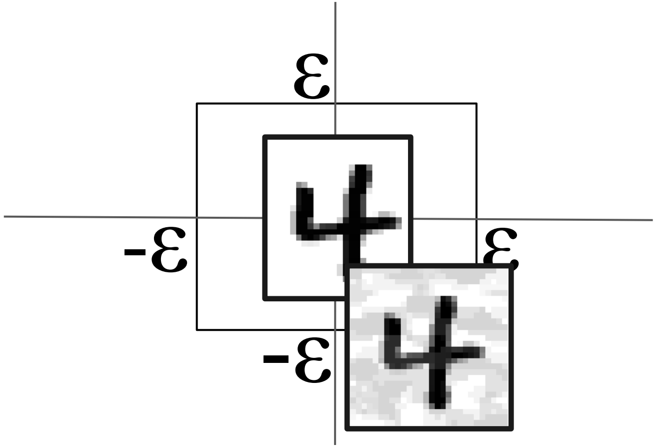


PGD (a.k.a Iterated-GSM)

FGSM



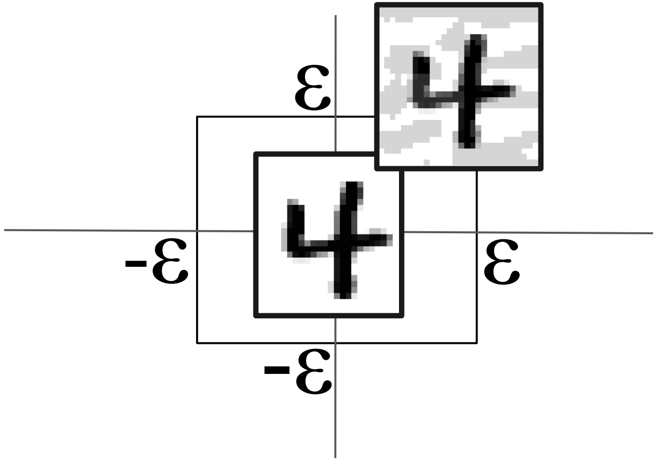
PGD



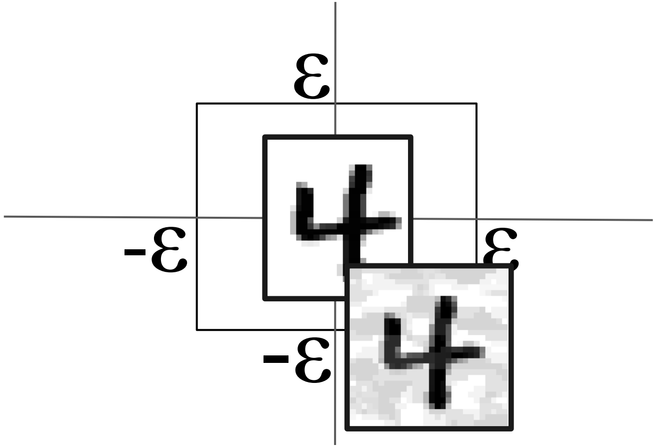
$$\min_{\theta} \mathbb{E}_{(x,y)\sim D}[\max_{\delta \in S} L(f_{\theta}(x+\delta), y)]$$

PGD (a.k.a Iterated-GSM)

FGSM



PGD



$$\min_{\theta} \mathbb{E}_{(x,y)\sim D}[\boxed{\max_{\delta\in S} L(f_\theta(x+\delta)}, y)]$$

PGD (a.k.a Iterated-GSM)

<u>Attack Model:</u>

$$S = \{\delta \mid \|\delta\|_\infty < \varepsilon\}$$

PGD (a.k.a Iterated-GSM)

Attack Model:

$$S = \{\delta \mid \|\delta\|_\infty < \varepsilon\}$$

FGSM:

$$\boldsymbol{X}^{adv} = \boldsymbol{X} + \epsilon\,\mathrm{sign}\big(\nabla_X L(\boldsymbol{X}, y_{true})\big)$$

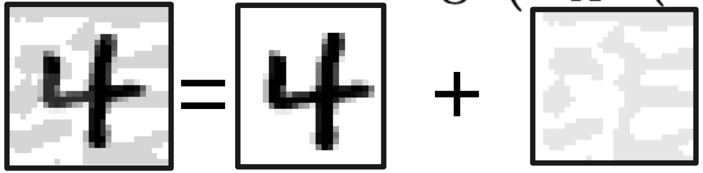
PGD (a.k.a Iterated-GSM)

Attack Model:

$$S = \{\delta \mid \|\delta\|_\infty < \varepsilon\}$$

FGSM:

$$\boldsymbol{X}^{adv} = \boldsymbol{X} + \epsilon\,\mathrm{sign}\big(\nabla_X L(\boldsymbol{X}, y_{true})\big)$$



PGD (a.k.a Iterated-GSM)

Attack Model:

$$S = \{\delta \mid \|\delta\|_\infty < \varepsilon\}$$

PGD (a.k.a. Iterative-GSM):

PGD (a.k.a Iterated-GSM)

Attack Model:

$$S = \{\delta \mid \|\delta\|_{\infty} < \varepsilon\}$$

PGD:

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X},$$

PGD (a.k.a Iterated-GSM)

Attack Model:

$$S = \{\delta \mid \|\delta\|_\infty < \varepsilon\}$$

PGD:

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X},$$

$$\boldsymbol{X}_{N+1}^{adv} = \qquad \boldsymbol{X}_N^{adv} + \alpha\,\mathrm{sign}\big(\nabla_X L(\boldsymbol{X}_N^{adv}, y_{true})\big)$$

PGD (a.k.a Iterated-GSM)

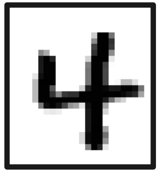
Attack Model:

$$S = \{\delta \mid \|\delta\|_\infty < \varepsilon\}$$

PGD:

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X},$$

$$\boldsymbol{X}_{N+1}^{adv} = \qquad \boldsymbol{X}_N^{adv} + \alpha\,\mathrm{sign}\big(\nabla_X L(\boldsymbol{X}_N^{adv}, y_{true})\big)$$

=  +

PGD (a.k.a Iterated-GSM)

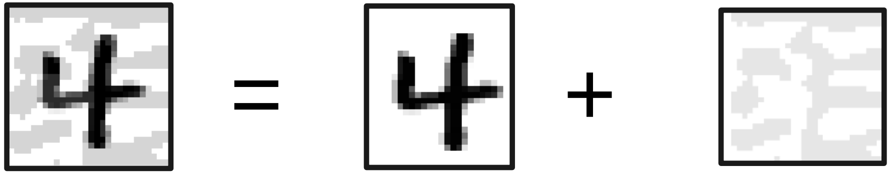
Attack Model:

$$S = \{\delta \mid \|\delta\|_\infty < \varepsilon\}$$

PGD:

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X},$$

$$\boldsymbol{X}_{N+1}^{adv} = \boldsymbol{X}_N^{adv} + \boxed{\alpha}\,\text{sign}\big(\nabla_X L(\boldsymbol{X}_N^{adv}, y_{true})\big)$$



PGD (a.k.a Iterated-GSM)

Attack Model:

S = $\{\delta \mid \|\delta\|_\infty < \varepsilon\}$

PGD:

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X},$$

$$\boldsymbol{X}_{N+1}^{adv} = \boldsymbol{X}_N^{adv} + \alpha\,\mathrm{sign}\big(\nabla_X L(\boldsymbol{X}_N^{adv}, y_{true})\big)$$

X_n^{adv} δ_n

n = 1

PGD (a.k.a Iterated-GSM)

Attack Model:

$$S = \{\delta \mid \|\delta\|_\infty < \varepsilon\}$$

PGD:

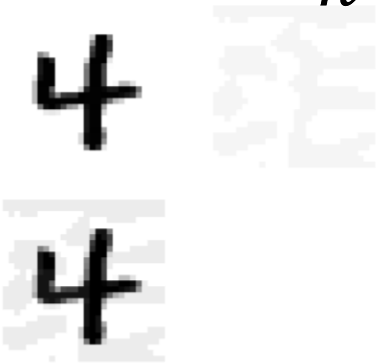
$$\boldsymbol{X}_0^{adv} = \boldsymbol{X},$$

$$\boldsymbol{X}_{N+1}^{adv} = \qquad \boldsymbol{X}_N^{adv} + \alpha\, \mathrm{sign}\big(\nabla_X L(\boldsymbol{X}_N^{adv}, y_{true})\big)$$

X_n^{adv} δ_n

n = 1

n = 2



PGD (a.k.a Iterated-GSM)

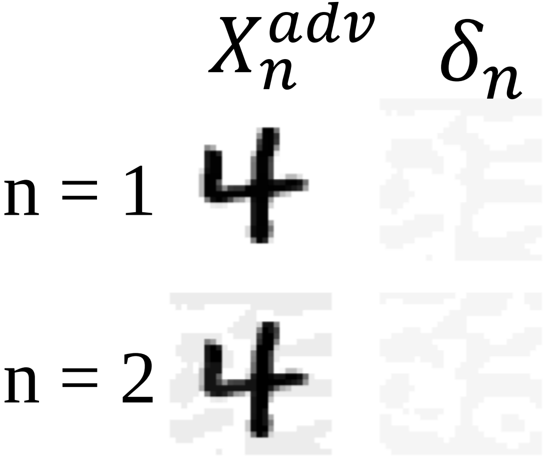
Attack Model:

$S = \{\delta \mid \|\delta\|_\infty < \varepsilon\}$

PGD:

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X},$$

$$\boldsymbol{X}_{N+1}^{adv} = \quad \boldsymbol{X}_N^{adv} + \alpha \operatorname{sign}\big(\nabla_X L(\boldsymbol{X}_N^{adv}, y_{true})\big)$$



PGD (a.k.a Iterated-GSM)

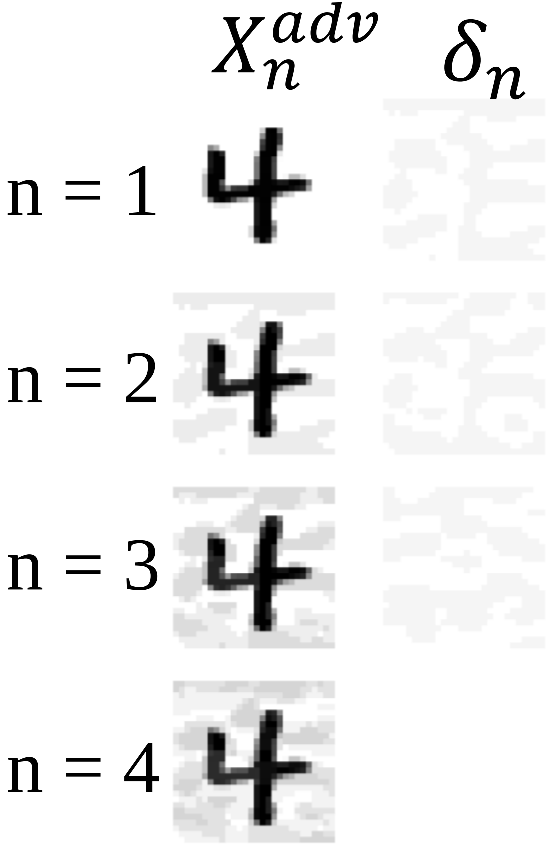
Attack Model:

$$S = \{\delta \mid \|\delta\|_\infty < \varepsilon\}$$

PGD:

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X},$$

$$\boldsymbol{X}_{N+1}^{adv} = \qquad \boldsymbol{X}_N^{adv} + \alpha\,\mathrm{sign}\big(\nabla_X L(\boldsymbol{X}_N^{adv}, y_{true})\big)$$



PGD (a.k.a Iterated-GSM)

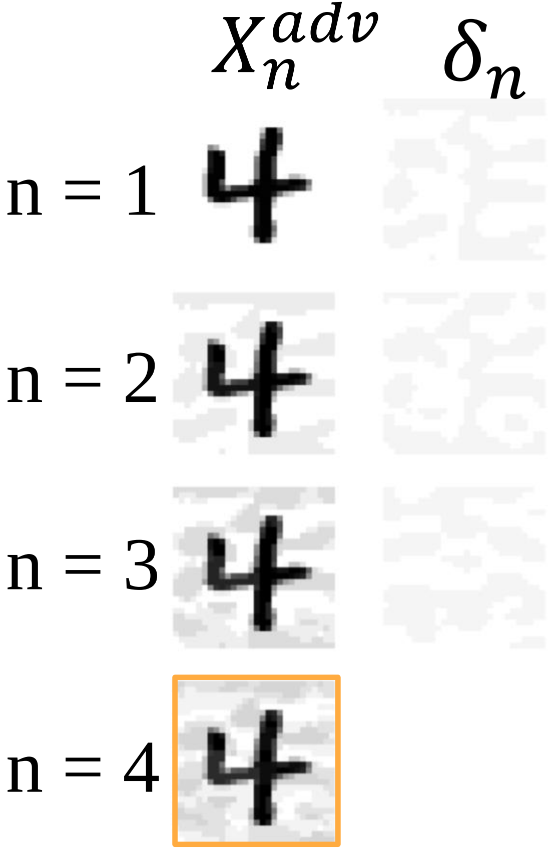
Attack Model:

$$S = \{\delta \mid \|\delta\|_\infty < \varepsilon\}$$

PGD:

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X},$$

$$\boldsymbol{X}_{N+1}^{adv} = \qquad\qquad \boldsymbol{X}_N^{adv} + \alpha\,\mathrm{sign}\big(\nabla_X L(\boldsymbol{X}_N^{adv}, y_{true})\big)$$



PGD (a.k.a Iterated-GSM)

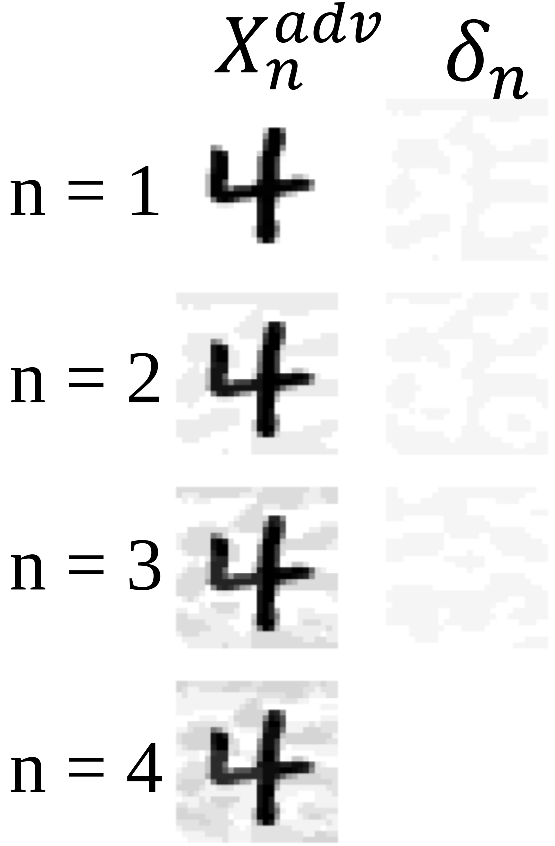
Attack Model:

$$S = \{\delta \mid \|\delta\|_\infty < \varepsilon\}$$

PGD:

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X},$$

$$\boldsymbol{X}_{N+1}^{adv} = Clip_{X,\epsilon}\Big\{\boldsymbol{X}_N^{adv} + \alpha\,\mathrm{sign}\big(\nabla_X$$ [redacted] $$(\boldsymbol{X}_N^{adv}, y_{true})\big)\Big\}$$



Adversarial Training

	Test Accuracy	FGSM Accuracy
Standard Training	98.7%	40.7%
Adv. Training (FGSM)	97.2%	94.0%

Adversarial Training

	Test Accuracy	FGSM Accuracy	PGD Accuracy
Standard Training	98.7%	40.7%	7.3%
Adv. Training (FGSM)	97.2%	94.0%	90.0%

Adversarial Training

	Test Accuracy	FGSM Accuracy	PGD Accuracy
Standard Training	98.7%	40.7%	7.3%
Adv. Training (FGSM)	97.2%	94.0%	90.0%

What can we do to defend?

Adversarial Training

	Test Accuracy	FGSM Accuracy	PGD Accuracy
Standard Training	98.7%	40.7%	7.3%
Adv. Training (FGSM)	97.2%	94.0%	90.0%
Adv. Training (PGD)	98.0%	96.1%	95.9%

Adversarial Training

	Test Accuracy	FGSM Accuracy	PGD Accuracy
Standard Training	98.7%	40.7%	7.3%
GSM)	97.2%	94.0%	90.0%
GD)	98.0%	96.1%	95.9%

Did we solve the problem?

Adversarial Training – Other Datasets

CIFAR10 (ResNet50)	Test	PGD ($\epsilon = \frac{8}{255}$)
Standard Training	95.25%	0.00%

source: https://github.com/MadryLab/robustness

Adversarial Training – Other Datasets

CIFAR10 (ResNet50)	Test	PGD ($\epsilon = \frac{8}{255}$)
Standard Training	95.25%	0.00%
Adv. Training (PGD 8/255)	87.03%	53.29%

source: https://github.com/MadryLab/robustness

Adversarial Training – Other Datasets

CIFAR10 (ResNet50)	Test	PGD ($\epsilon = \frac{8}{255}$)
Standard Training	95.25%	0.00%
Adv. Training (PGD 8/255)	87.03%	53.29%

ImageNet (ResNet50)	Test	PGD ($\epsilon = \frac{8}{255}$)
Standard Training	76.13%	0.01%

source: https://github.com/MadryLab/robustness

Adversarial Training – Other Datasets

CIFAR10 (ResNet50)	Test	PGD ($\epsilon = \frac{8}{255}$)
Standard Training	95.25%	0.00%
Adv. Training (PGD 8/255)	87.03%	53.29%

ImageNet (ResNet50)	Test	PGD ($\epsilon = \frac{8}{255}$)
Standard Training	76.13%	0.01%
Adv. Training (PGD 8/255)	47.91%	19.52%

source: https://github.com/MadryLab/robustness

Outline

- See Adversarial Example
- Discuss what they are
- How to attack: FGSM, PGD
- How to defend: Adversarial training (AT)
- Optimization view of AT

Outline

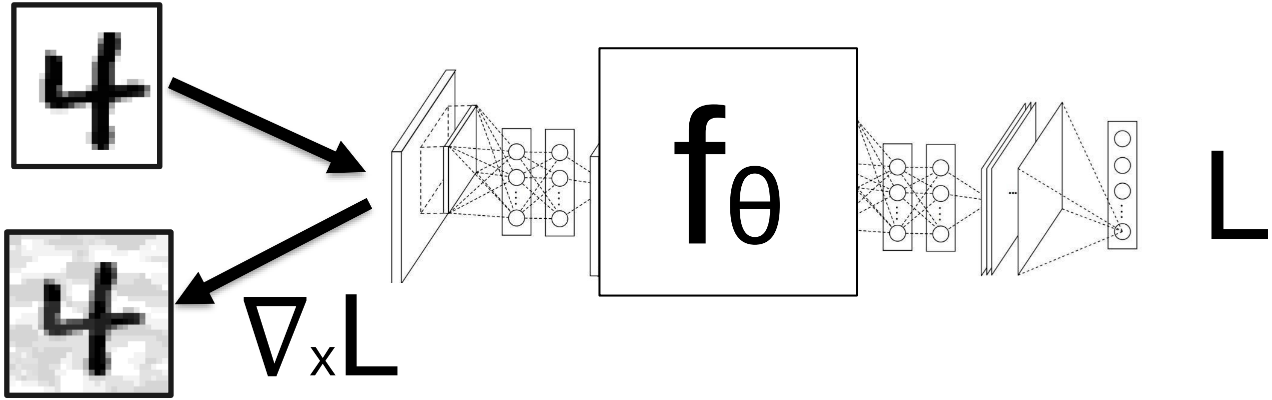
- See Adversarial Example
- Discuss what they are
- How to attack: FGSM, PGD
- How to defend: Adversarial training (AT)
- Optimization view of AT

White Box Attacks

Outline

- See Adversarial Example
- Discuss what they are
- How to attack: FGSM, PGD
- How to defend: Adversarial training (AT)
- Optimization view of AT
- Next: Black-Box attacks
- Learn about properties and advantages

Black-Box Attacks



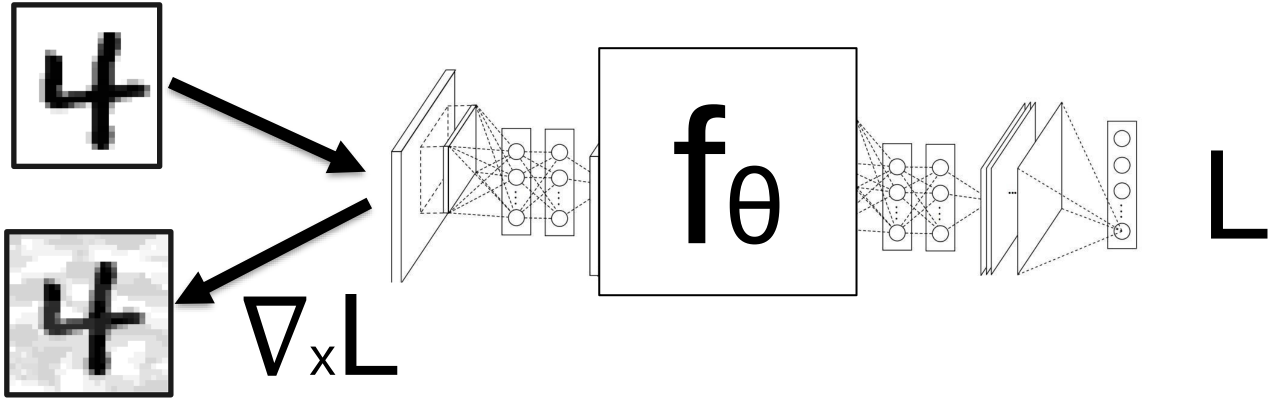
f_θ

L

$\nabla_x L$

Black-Box Attacks

“White-Box”



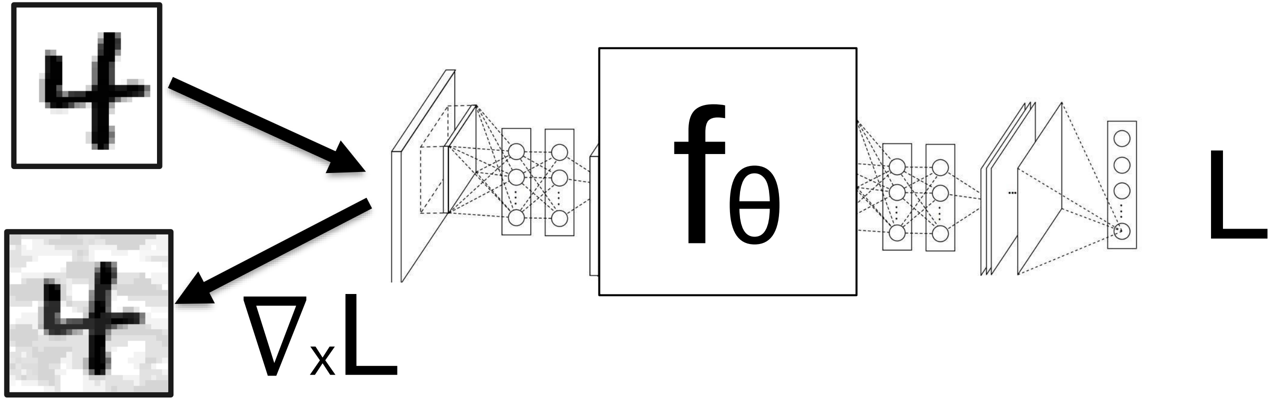
f_θ

L

$\nabla_x L$

Black-Box Attacks

“White-Box”
(FGSM,
PGD, etc.)



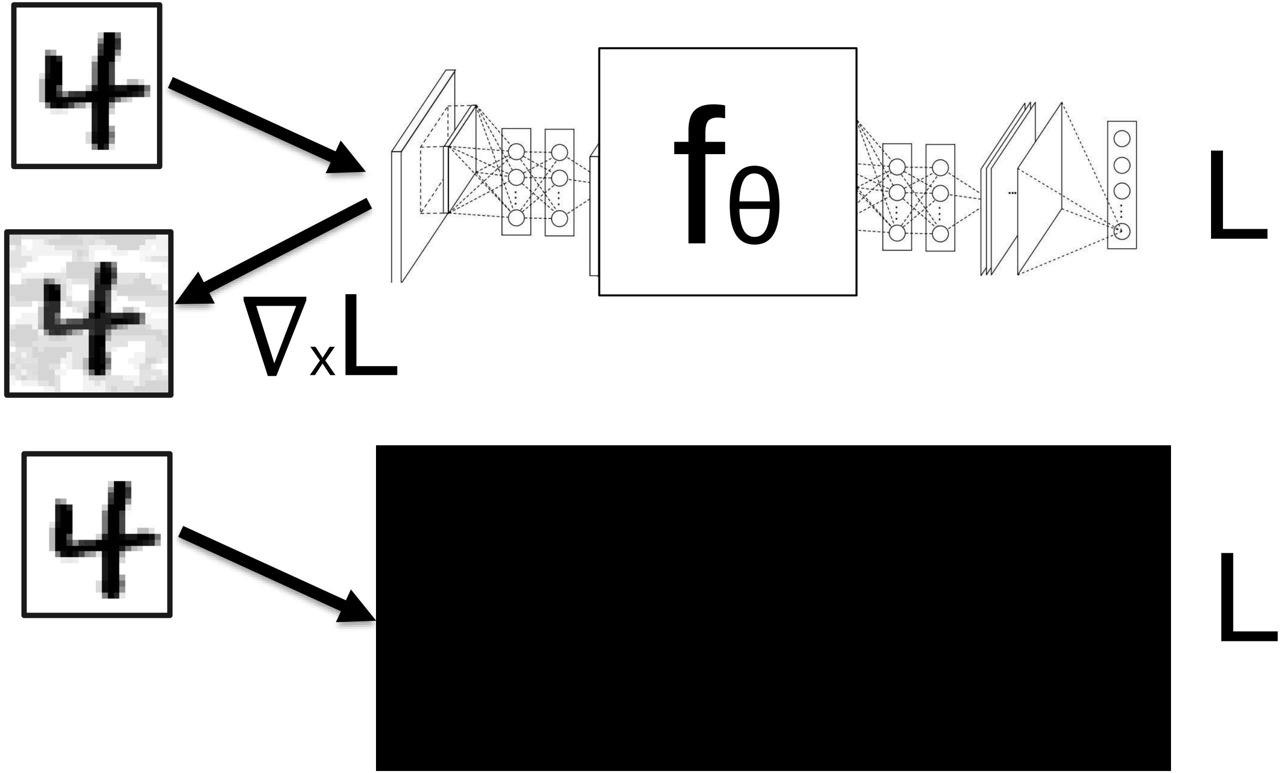
f_θ

L

$\nabla_x L$

Black-Box Attacks

“White-Box”



f_θ

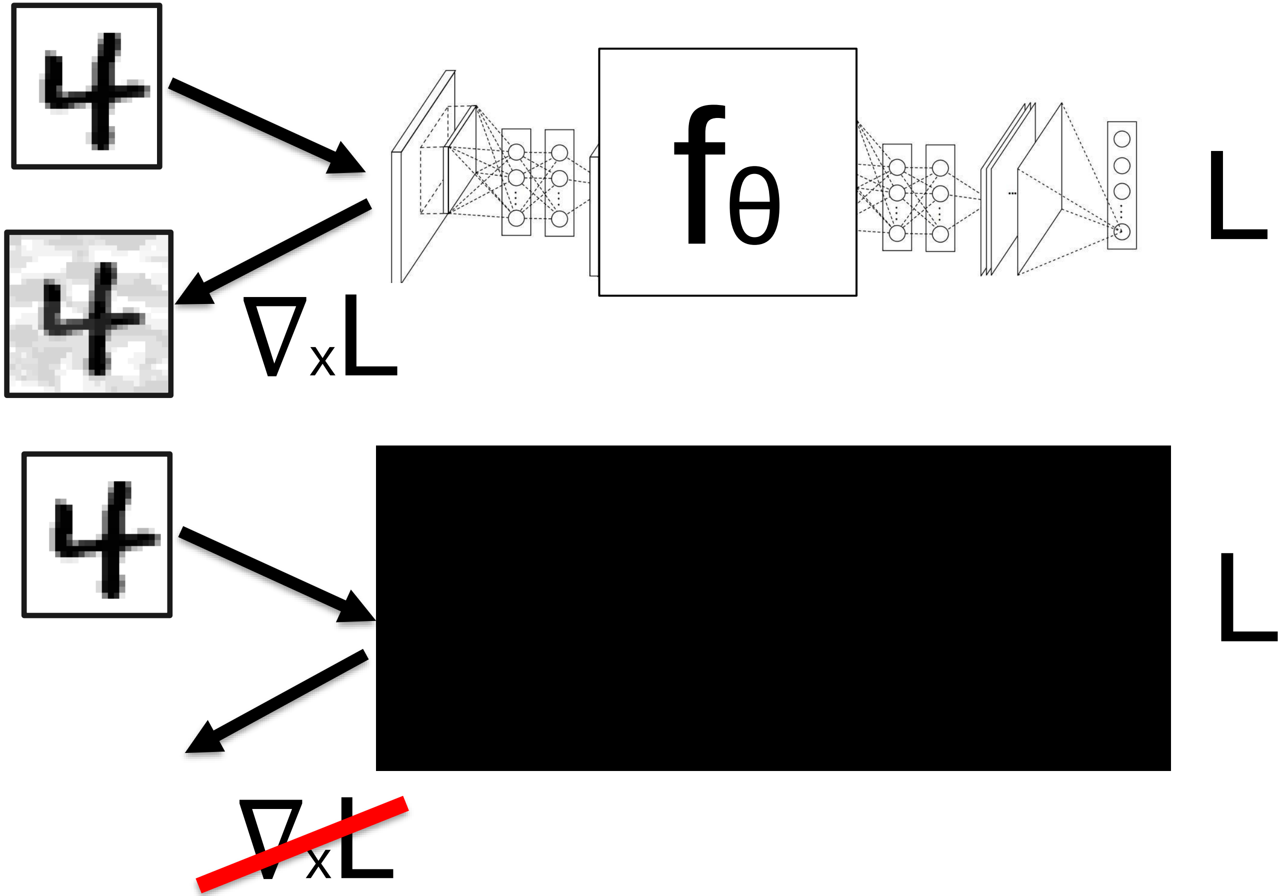
L

$\nabla_x L$

L

Black-Box Attacks

“White-Box”



f_θ

L

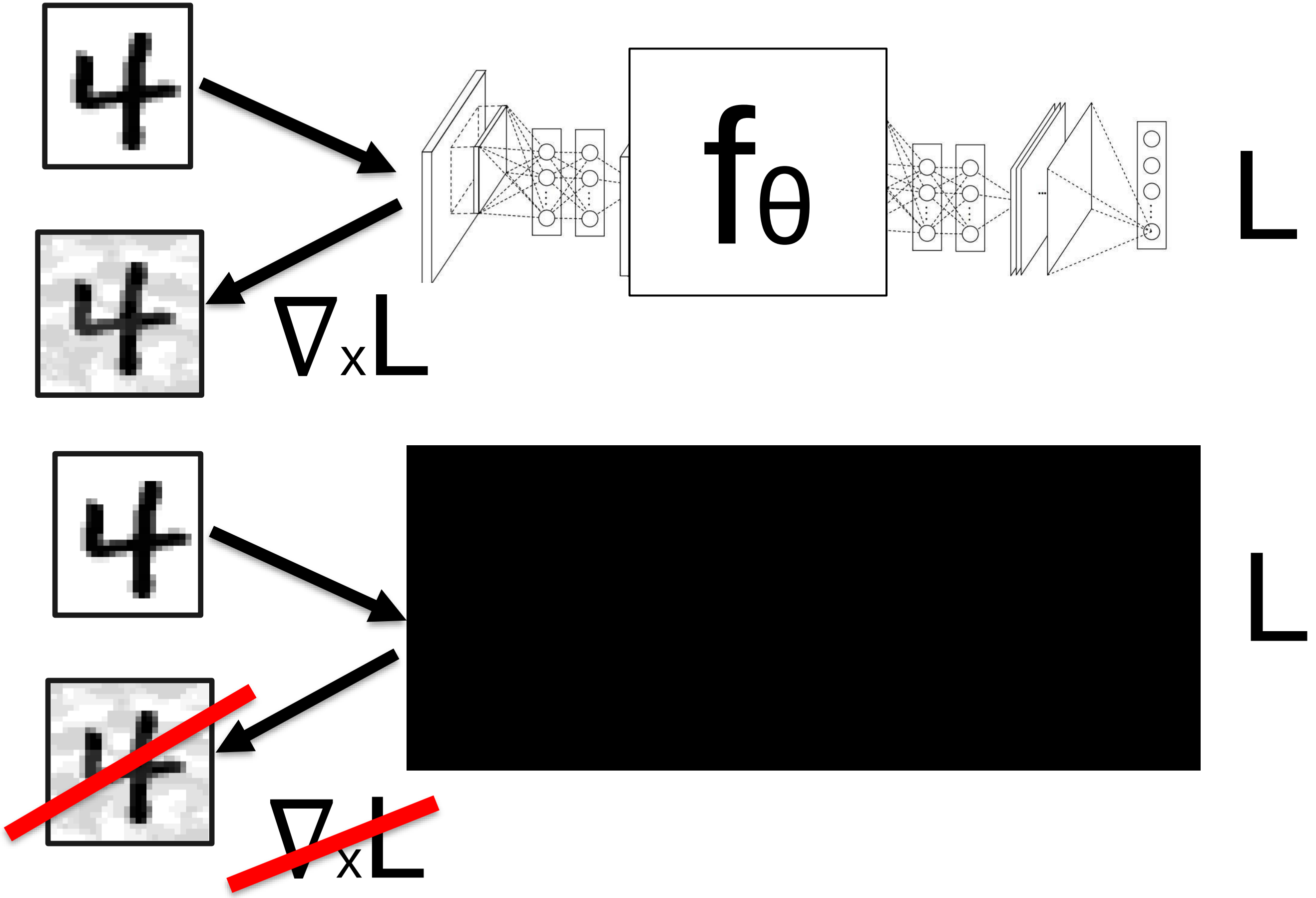
$\nabla_x L$

L

$\cancel{\nabla_x L}$

Black-Box Attacks

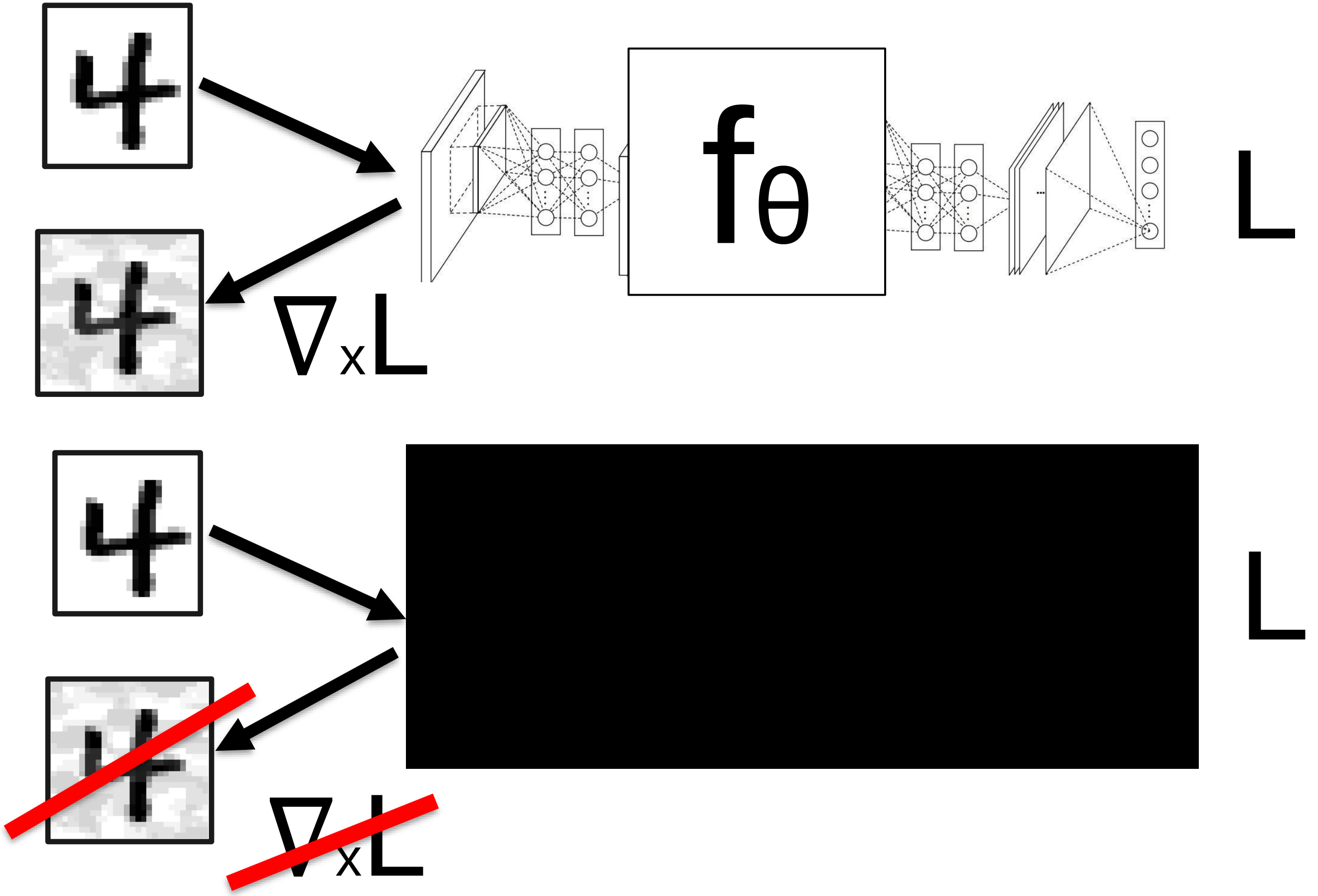
“White-Box”



Black-Box Attacks

“White-Box”

“Black-Box”



f_θ

L

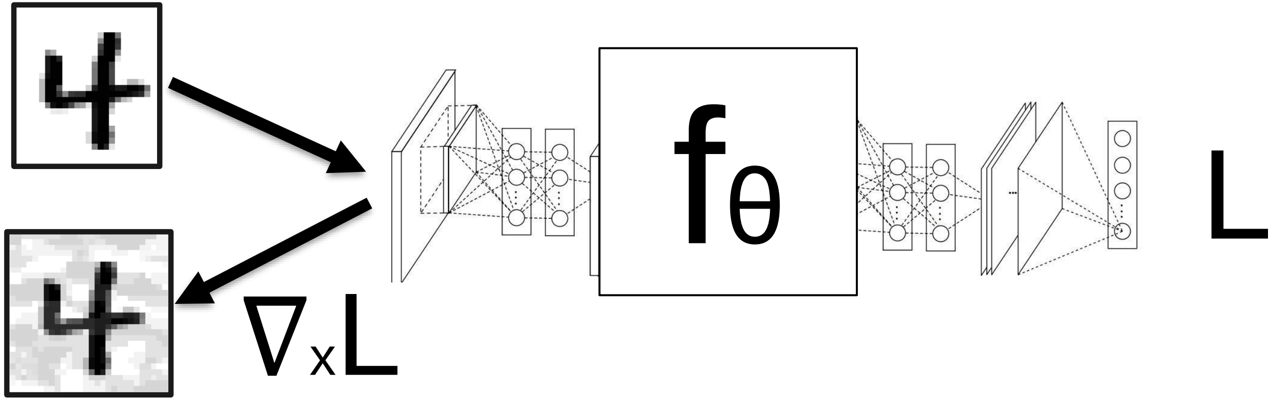
$\nabla_x L$

L

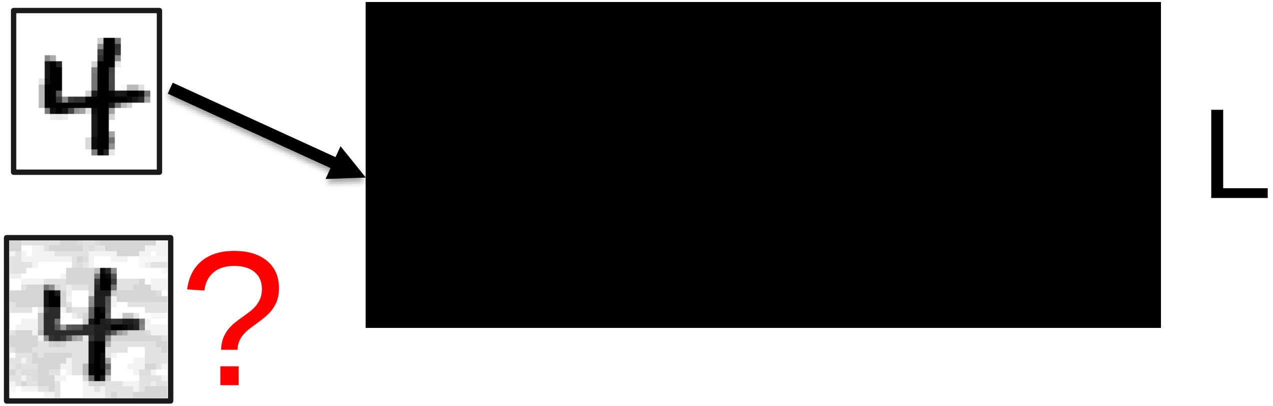
$\cancel{\nabla_x L}$

Black-Box Attacks

“White-Box”

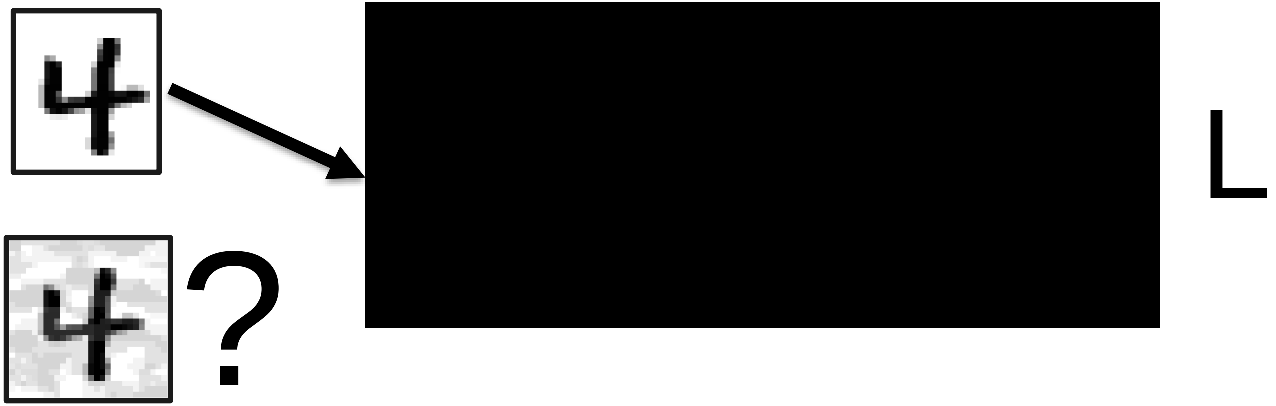


“Black-Box”



Black-Box Attacks

“Black-Box”

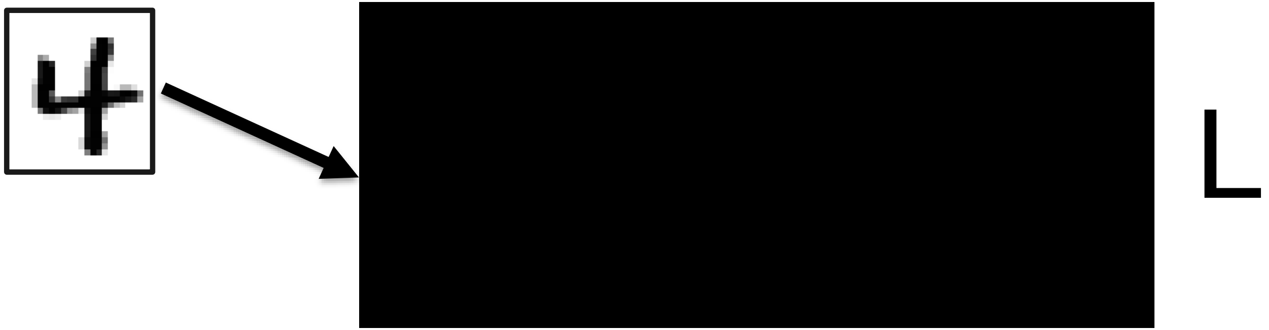


?

L

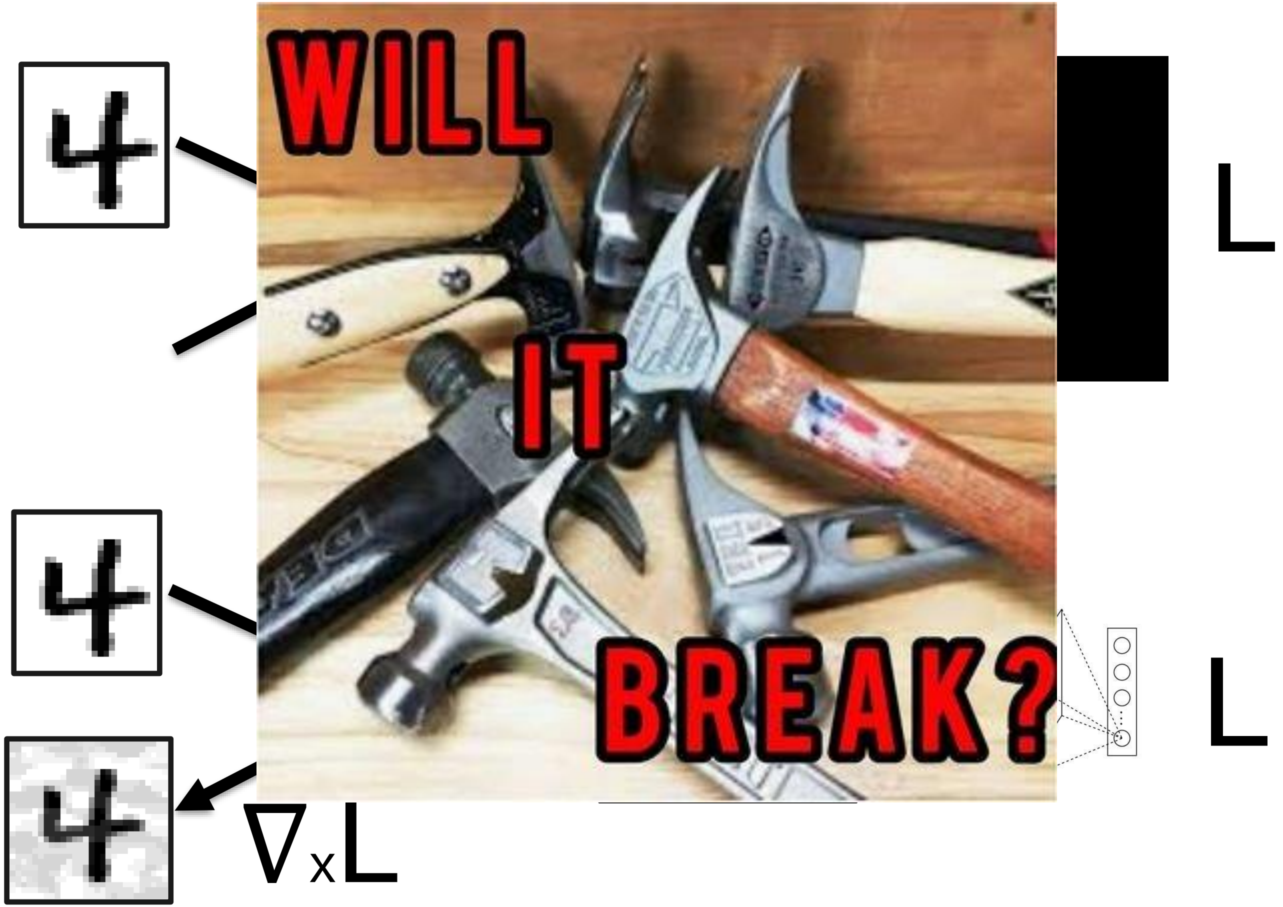
Black-Box Attacks

“Black-Box”



Black-Box Attacks

“Black-Box”



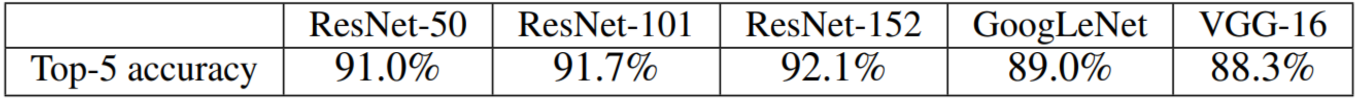
source: https://twitter.com/will_it_breakyt

Black-Box Attacks - Transferability

Liu et al. 2016, “Delving into Transferable Adversarial Examples and Black-box Attacks”

Black-Box Attacks - Transferability

- Test set Accuracy

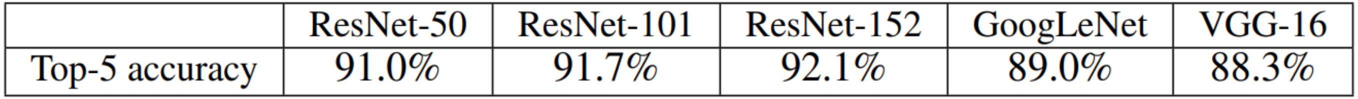


	ResNet-50	ResNet-101	ResNet-152	GoogLeNet	VGG-16
Top-5 accuracy	91.0%	91.7%	92.1%	89.0%	88.3%

Liu et al. 2016, “Delving into Transferable Adversarial Examples and Black-box Attacks”

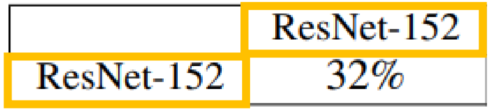
Black-Box Attacks - Transferability

- Test set Accuracy



	ResNet-50	ResNet-101	ResNet-152	GoogLeNet	VGG-16
Top-5 accuracy	91.0%	91.7%	92.1%	89.0%	88.3%

- Accuracy under FGSM attack

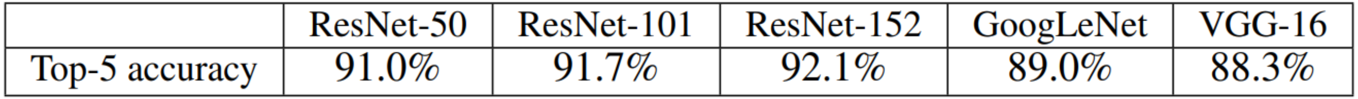


	ResNet-152
ResNet-152	32%

Liu et al. 2016, “Delving into Transferable Adversarial Examples and Black-box Attacks”

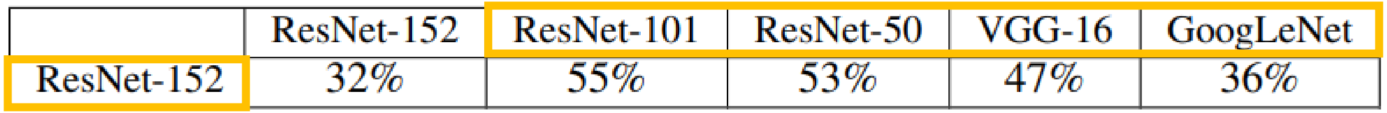
Black-Box Attacks - Transferability

- Test set Accuracy



	ResNet-50	ResNet-101	ResNet-152	GoogLeNet	VGG-16
Top-5 accuracy	91.0%	91.7%	92.1%	89.0%	88.3%

- Accuracy under FGSM attack

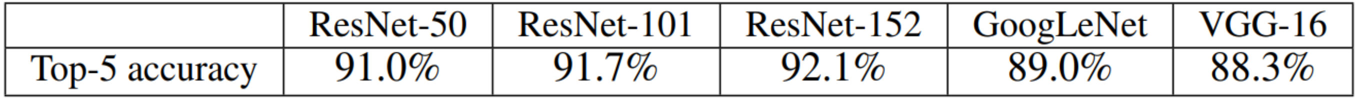


	ResNet-152	ResNet-101	ResNet-50	VGG-16	GoogLeNet
ResNet-152	32%	55%	53%	47%	36%

Liu et al. 2016, “Delving into Transferable Adversarial Examples and Black-box Attacks”

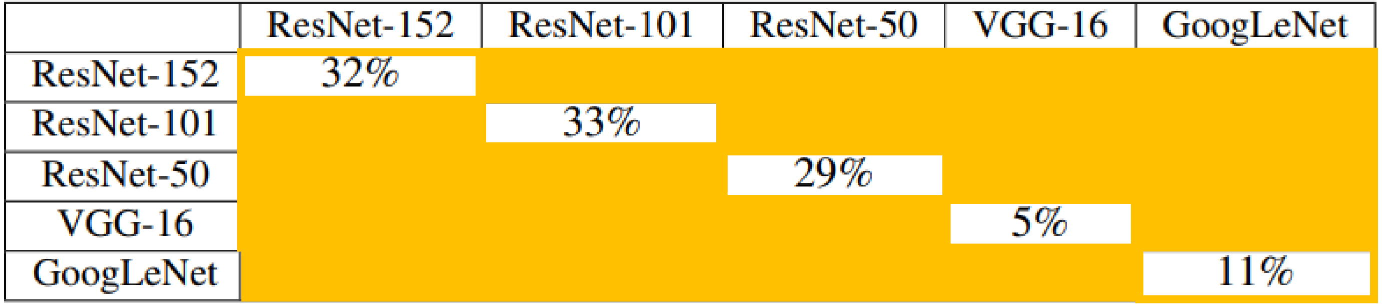
Black-Box Attacks - Transferability

- Test set Accuracy



	ResNet-50	ResNet-101	ResNet-152	GoogLeNet	VGG-16
Top-5 accuracy	91.0%	91.7%	92.1%	89.0%	88.3%

- Accuracy under FGSM attack



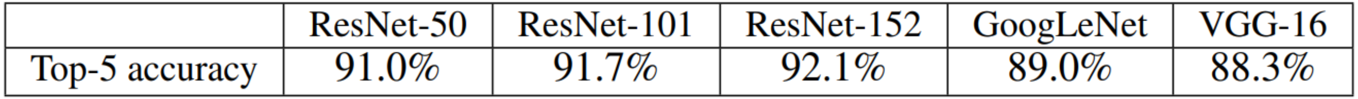
	ResNet-152	ResNet-101	ResNet-50	VGG-16	GoogLeNet
ResNet-152	32%				
ResNet-101		33%			
ResNet-50			29%		
VGG-16				5%	
GoogLeNet					11%

White-Box FGSM

Liu et al. 2016, “Delving into Transferable Adversarial Examples and Black-box Attacks” (Tab.20)

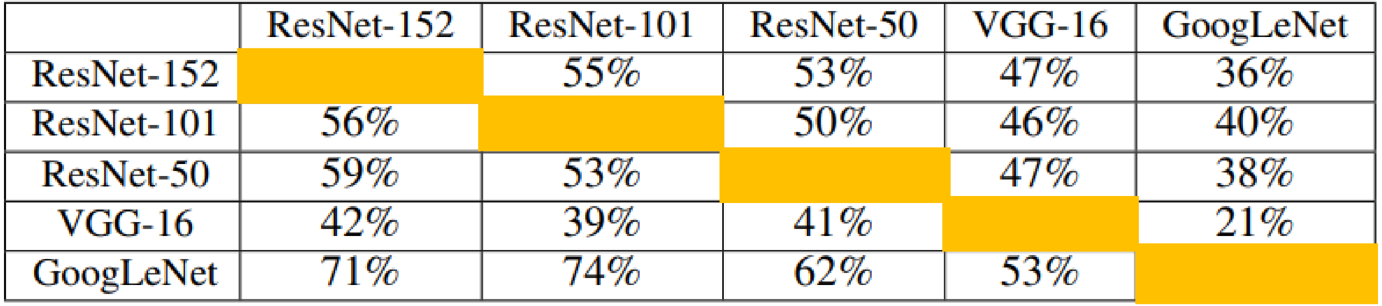
Black-Box Attacks - Transferability

- Test set Accuracy



	ResNet-50	ResNet-101	ResNet-152	GoogLeNet	VGG-16
Top-5 accuracy	91.0%	91.7%	92.1%	89.0%	88.3%

- Accuracy under FGSM attack



	ResNet-152	ResNet-101	ResNet-50	VGG-16	GoogLeNet
ResNet-152		55%	53%	47%	36%
ResNet-101	56%		50%	46%	40%
ResNet-50	59%	53%		47%	38%
VGG-16	42%	39%	41%		21%
GoogLeNet	71%	74%	62%	53%	

Black-Box

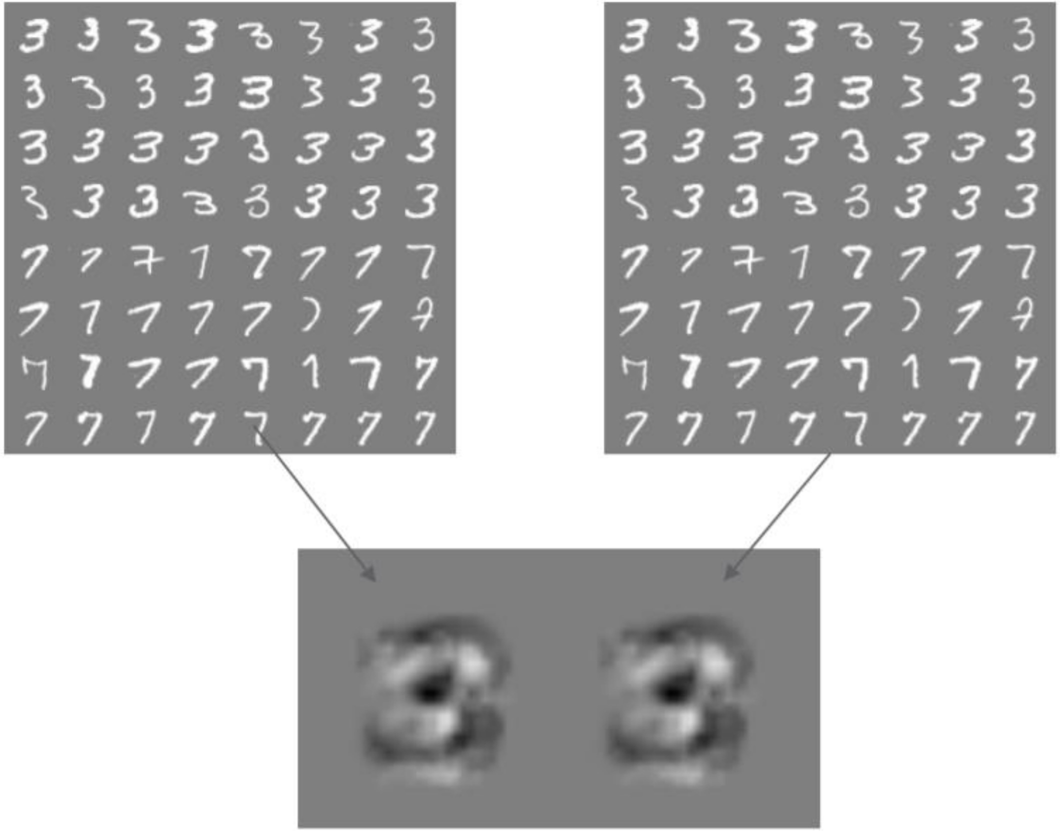
Liu et al. 2016, “Delving into Transferable Adversarial Examples and Black-box Attacks” (Tab.20)

Black-Box Attacks - Transferability

- Possible reason:

Black-Box Attacks - Transferability

- Possible reason:



source: Ian Goodfellow on "Adversarial Examples and Adversarial Training," 2017-05-30, CS231n, Stanford University

fellow 201

Black-Box Attacks - Transferability

- Possible reason:

Ilyas et al. 2019, “Adversarial Examples Are Not Bugs, They Are Features”

source: Ian Goodfellow on "Adversarial Examples and Adversarial Training," 2017-05-30, CS231n, Stanford University

fellow 201

Black-Box Attacks - Transferability

- Possible reason:

Adversarial Examples comes from the data:

Ilyas et al. 2019, “Adversarial Examples Are Not Bugs, They Are Features”

source: Ian Goodfellow on "Adversarial Examples and Adversarial Training," 2017-05-30, CS231n, Stanford University

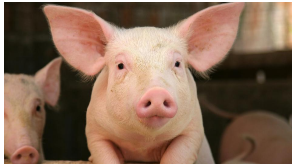
fellow 201

Outline

- See Adversarial Example
- Discuss what they are
- How to attack: FGSM, PGD
- How to defend: Adversarial training (AT)
- Optimization view of AT
- Black-Box attacks (transferability)
- Next: Summary
- Surprising “advantages” of AE

Adversarial Examples – The Bigger Picture

airliner

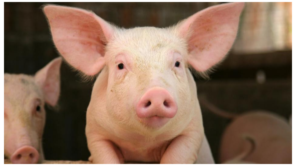


test+noise

Adversarial Examples – The Bigger Picture

Is this surprising?

airliner

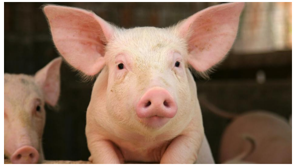


test+noise

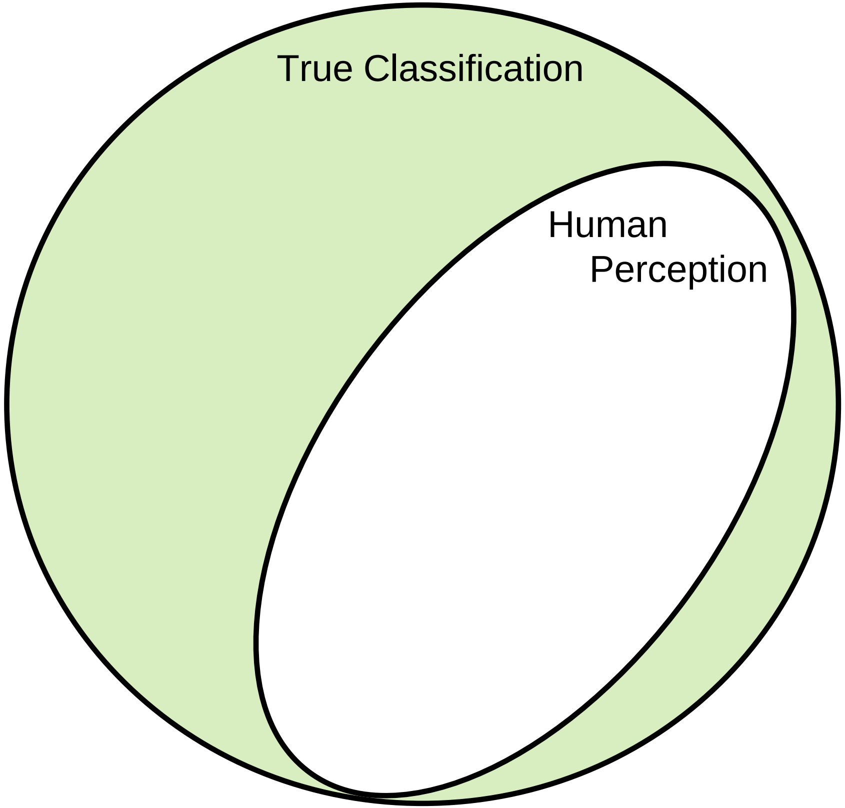
Adversarial Examples – The Bigger Picture

Is this surprising?

airliner



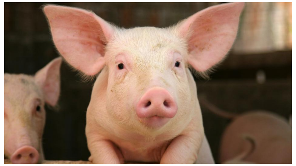
test+noise



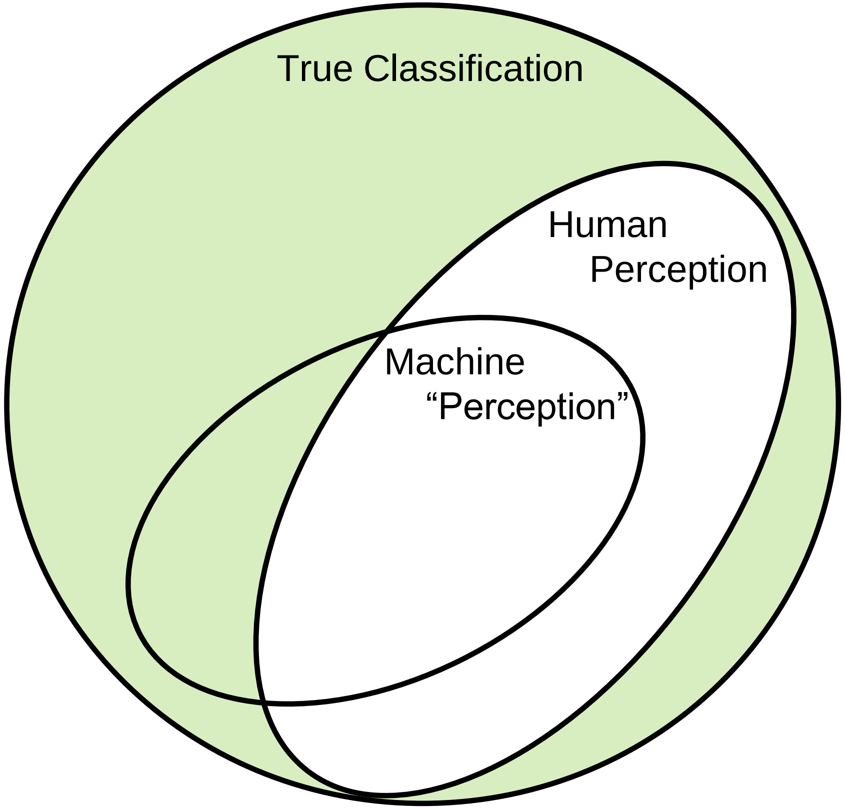
Adversarial Examples – The Bigger Picture

Is this surprising?

airliner



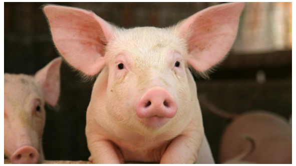
test+noise



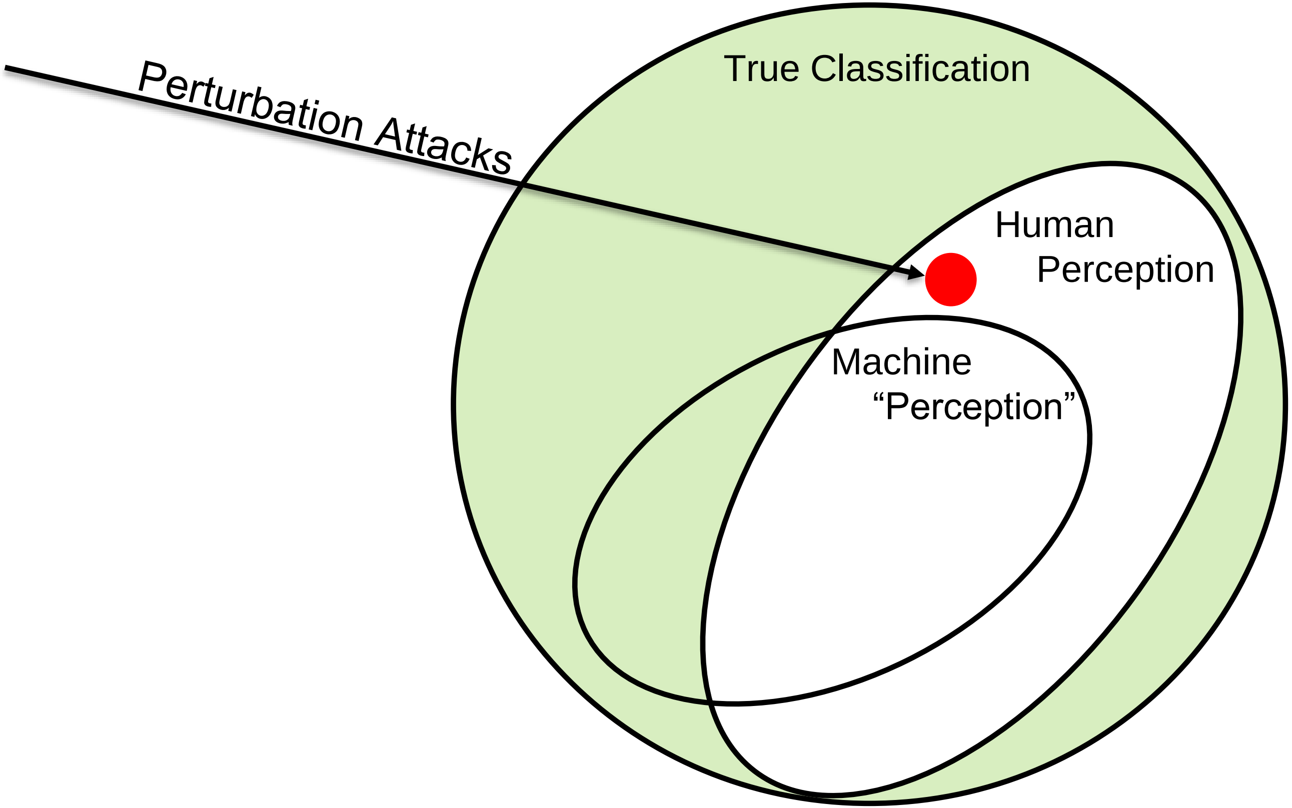
Adversarial Examples – The Bigger Picture

Is this surprising?

airliner



test+noise



Perturbation Attacks

True Classification

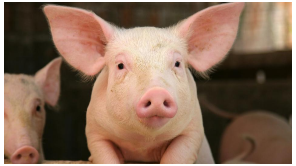
Human Perception

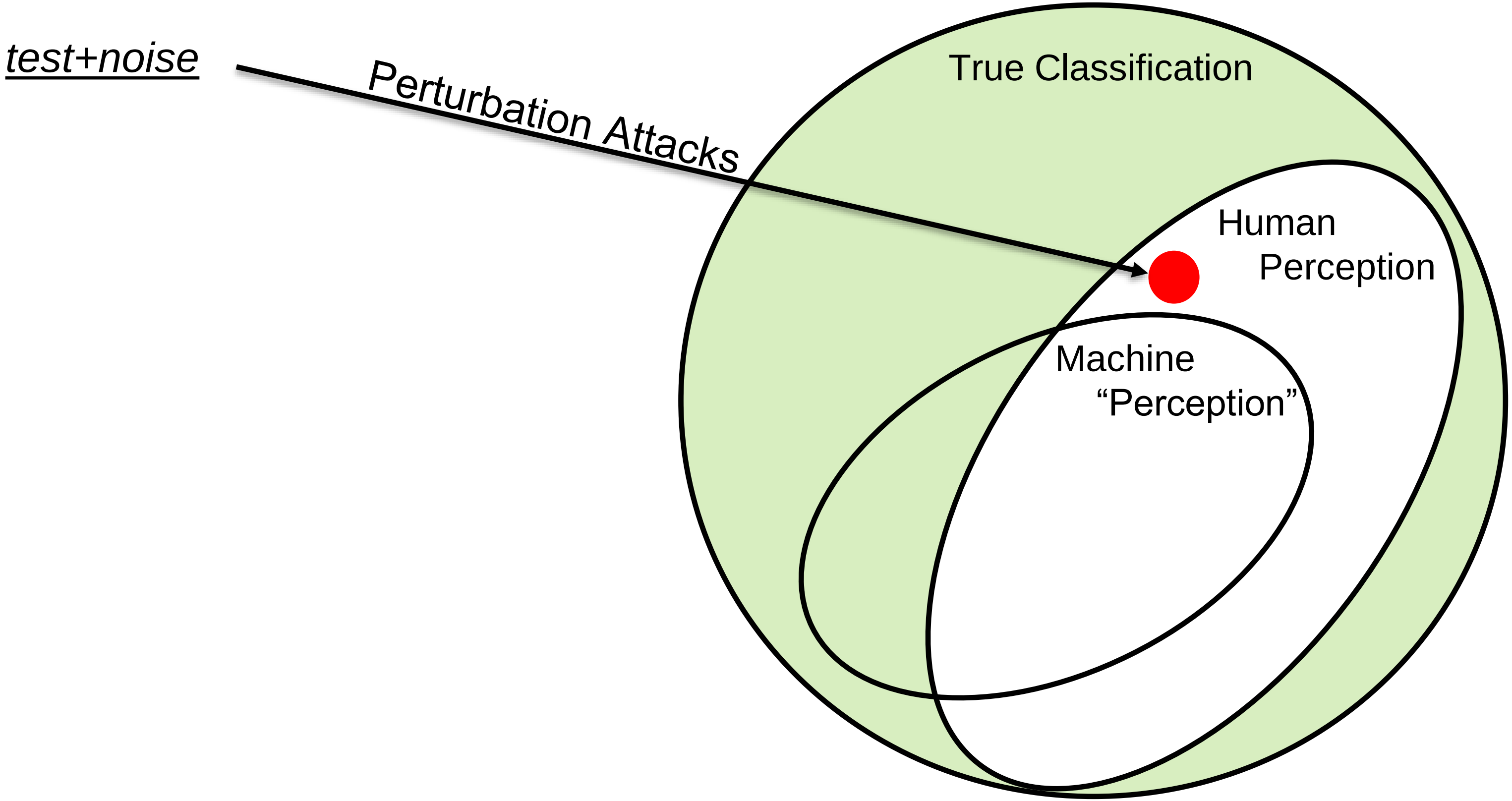
Machine "Perception"

Adversarial Examples – The Bigger Picture

Inputs that fool a computer, but not a human

airliner

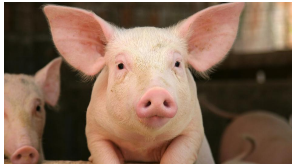




Adversarial Examples – The Bigger Picture

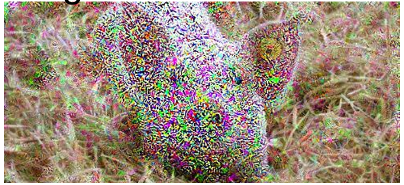
Inputs that fool a computer, but not a human

airliner

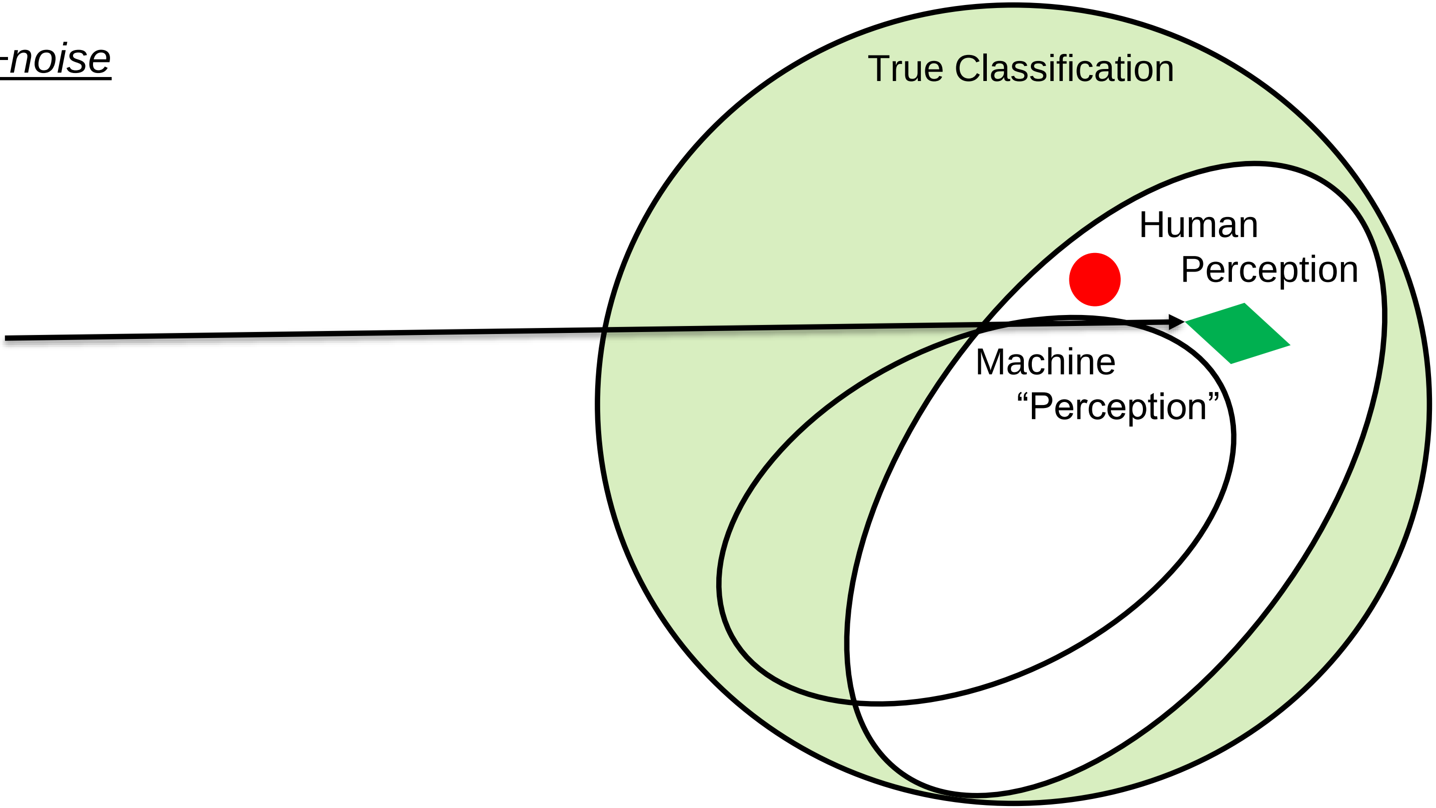


test+noise

fireguard



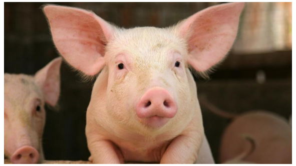
“noisy” image



Adversarial Examples – The Bigger Picture

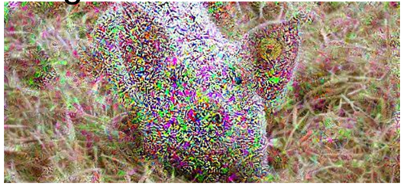
Inputs that fool a computer, but not a human

airliner



test+noise

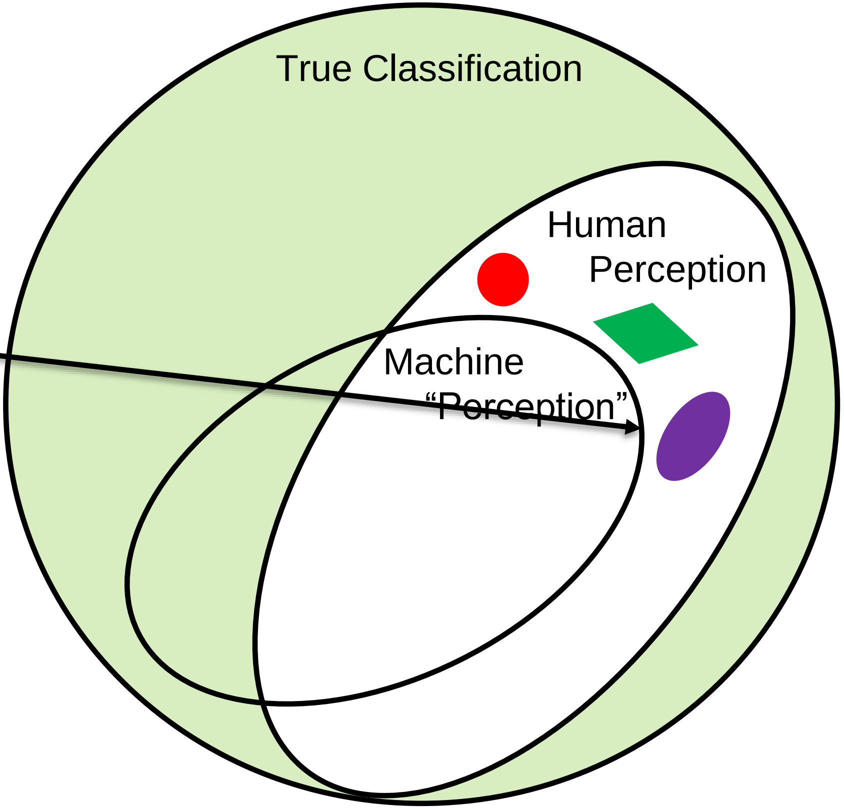
fireguard



“noisy” image

spotlight (26.7%)

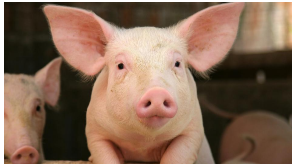
noise



Adversarial Examples – The Bigger Picture

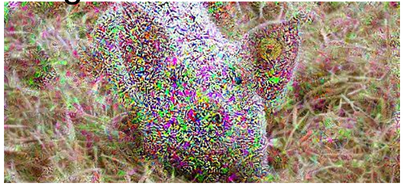
Inputs that fool a computer, but not a human

airliner



test+noise

fireguard



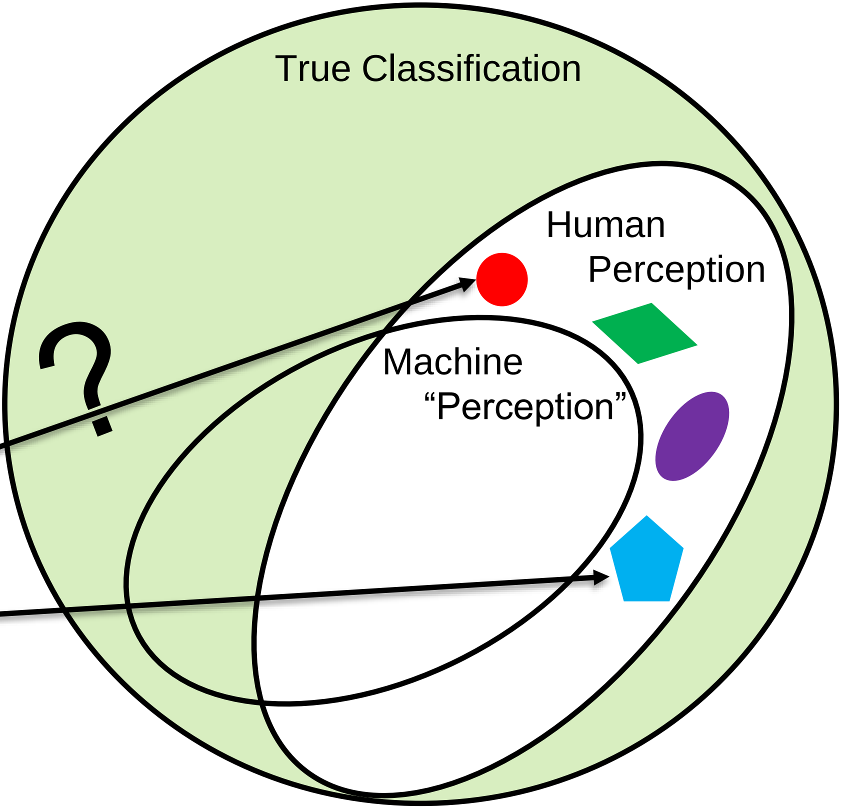
“noisy” image

spotlight (26.7%)

noise

cat ?

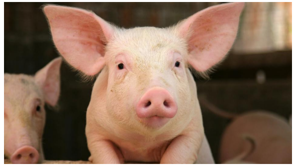
model failure



Adversarial Examples – The Bigger Picture

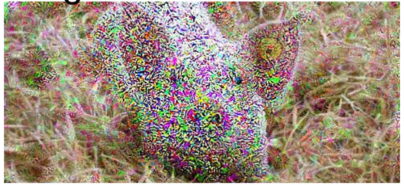
Inputs that fool a computer, but not a human

airliner



test+noise

fireguard



“noisy” image

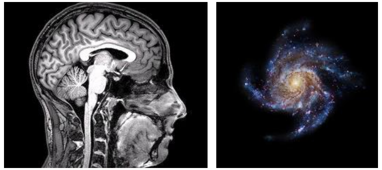
spotlight (26.7%)

noise

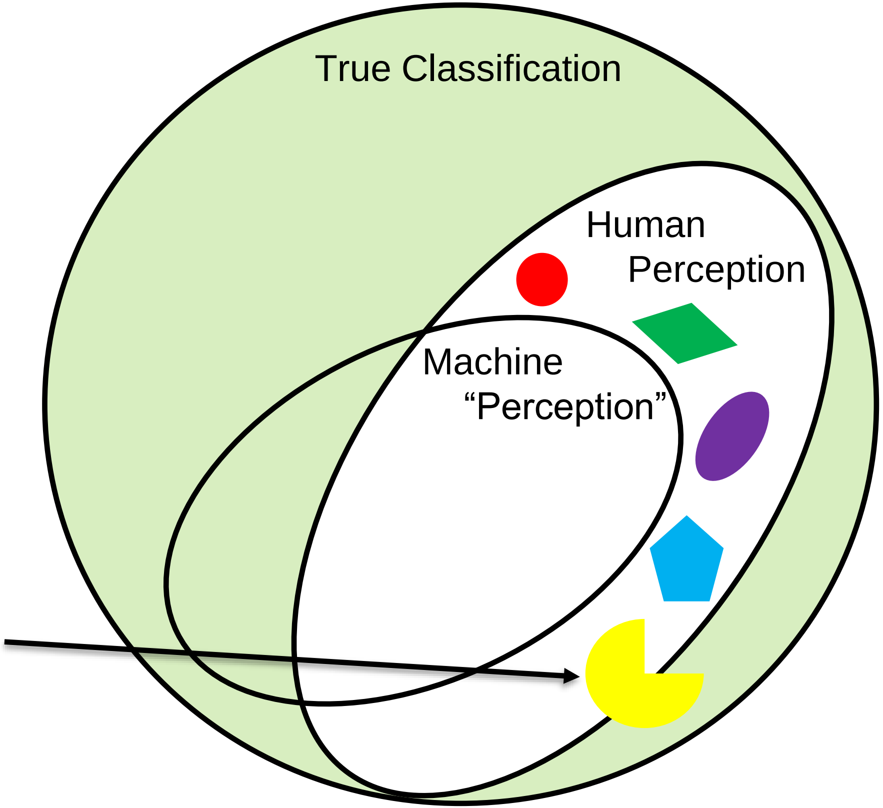
cat ?

model failure

???



out-of-distribution



The Bigger Picture: Failure modes in machine learning

source: https://docs.microsoft.com/en-us/security/engineering/failure-modes-in-machine-learning

The Bigger Picture: Failure modes in machine learning

Intentionally-motivated failures

source: https://docs.microsoft.com/en-us/security/engineering/failure-modes-in-machine-learning

The Bigger Picture: Failure modes in machine learning

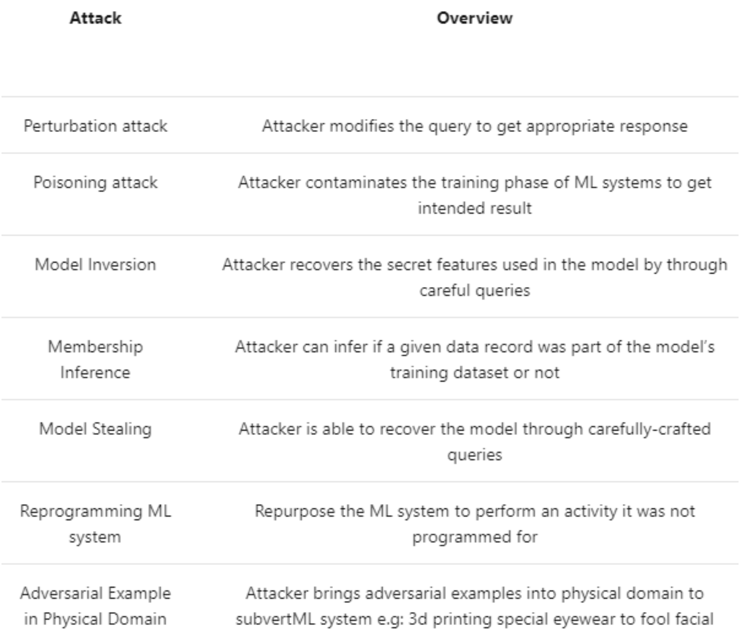
Intentionally-motivated failures

Unintended failures

source: https://docs.microsoft.com/en-us/security/engineering/failure-modes-in-machine-learning

The Bigger Picture: Failure modes in machine learning

Intentionally-motivated failures



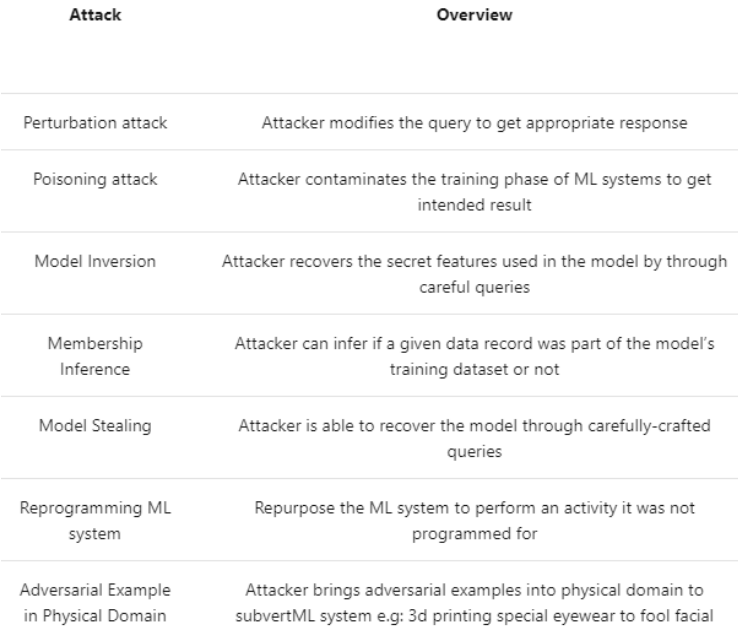
Attack	Overview
Perturbation attack	Attacker modifies the query to get appropriate response
Poisoning attack	Attacker contaminates the training phase of ML systems to get intended result
Model Inversion	Attacker recovers the secret features used in the model by through careful queries
Membership Inference	Attacker can infer if a given data record was part of the model's training dataset or not
Model Stealing	Attacker is able to recover the model through carefully-crafted queries
Reprogramming ML system	Repurpose the ML system to perform an activity it was not programmed for
Adversarial Example in Physical Domain	Attacker brings adversarial examples into physical domain to subvertML system e.g: 3d printing special eyewear to fool facial

Unintended failures

source: https://docs.microsoft.com/en-us/security/engineering/failure-modes-in-machine-learning

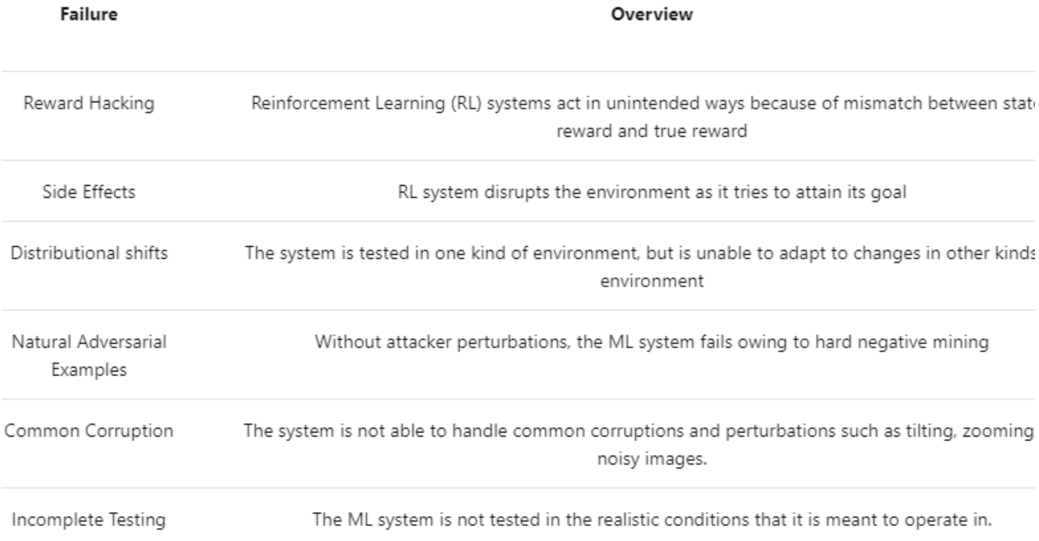
The Bigger Picture: Failure modes in machine learning

Intentionally-motivated failures



Attack	Overview
Perturbation attack	Attacker modifies the query to get appropriate response
Poisoning attack	Attacker contaminates the training phase of ML systems to get intended result
Model Inversion	Attacker recovers the secret features used in the model by through careful queries
Membership Inference	Attacker can infer if a given data record was part of the model's training dataset or not
Model Stealing	Attacker is able to recover the model through carefully-crafted queries
Reprogramming ML system	Repurpose the ML system to perform an activity it was not programmed for
Adversarial Example in Physical Domain	Attacker brings adversarial examples into physical domain to subvertML system e.g: 3d printing special eyewear to fool facial

Unintended failures

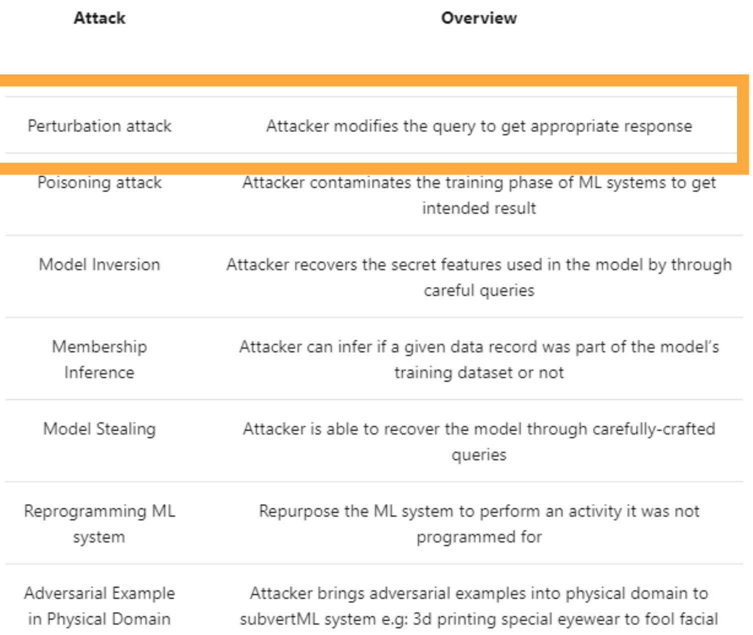


Failure	Overview
Reward Hacking	Reinforcement Learning (RL) systems act in unintended ways because of mismatch between stat reward and true reward
Side Effects	RL system disrupts the environment as it tries to attain its goal
Distributional shifts	The system is tested in one kind of environment, but is unable to adapt to changes in other kinds environment
Natural Adversarial Examples	Without attacker perturbations, the ML system fails owing to hard negative mining
Common Corruption	The system is not able to handle common corruptions and perturbations such as tilting, zooming noisy images.
Incomplete Testing	The ML system is not tested in the realistic conditions that it is meant to operate in.

source: https://docs.microsoft.com/en-us/security/engineering/failure-modes-in-machine-learning

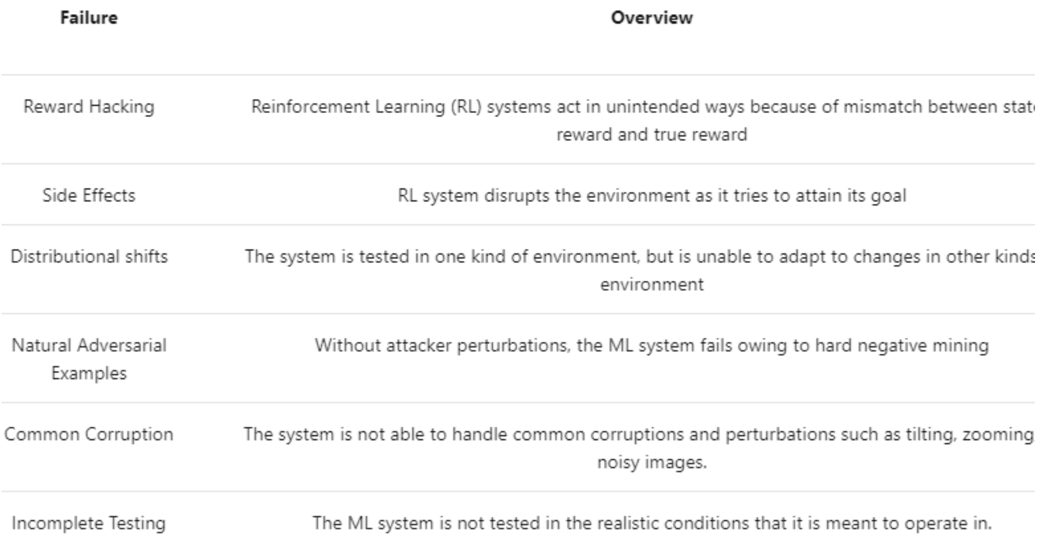
The Bigger Picture: Failure modes in machine learning

Intentionally-motivated failures



Attack	Overview
Perturbation attack	Attacker modifies the query to get appropriate response
Poisoning attack	Attacker contaminates the training phase of ML systems to get intended result
Model Inversion	Attacker recovers the secret features used in the model by through careful queries
Membership Inference	Attacker can infer if a given data record was part of the model's training dataset or not
Model Stealing	Attacker is able to recover the model through carefully-crafted queries
Reprogramming ML system	Repurpose the ML system to perform an activity it was not programmed for
Adversarial Example in Physical Domain	Attacker brings adversarial examples into physical domain to subvertML system e.g: 3d printing special eyewear to fool facial

Unintended failures

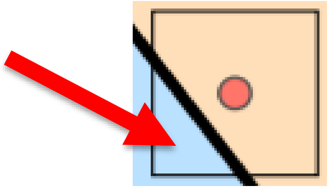


Failure	Overview
Reward Hacking	Reinforcement Learning (RL) systems act in unintended ways because of mismatch between stat reward and true reward
Side Effects	RL system disrupts the environment as it tries to attain its goal
Distributional shifts	The system is tested in one kind of environment, but is unable to adapt to changes in other kinds environment
Natural Adversarial Examples	Without attacker perturbations, the ML system fails owing to hard negative mining
Common Corruption	The system is not able to handle common corruptions and perturbations such as tilting, zooming noisy images.
Incomplete Testing	The ML system is not tested in the realistic conditions that it is meant to operate in.

source: https://docs.microsoft.com/en-us/security/engineering/failure-modes-in-machine-learning

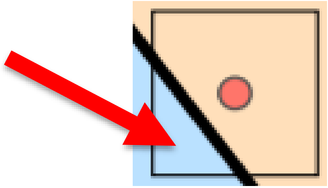
Adversarial Examples - Summary

- Remember the bigger picture (many failures)
- Hard to attack (need to find AE in box)



Adversarial Examples - Summary

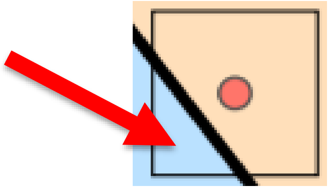
- Remember the bigger picture (many failures)
- Hard to attack (need to find AE in box)



- Harder to defend

Adversarial Examples - Summary

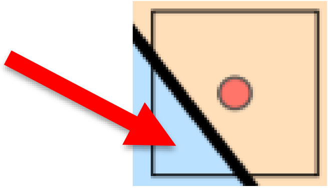
- Remember the bigger picture (many failures)
- Hard to attack (need to find AE in box)



- Harder to defend (need to prove: no AEs in <u>all</u> box)

Adversarial Examples - Summary

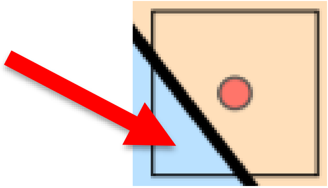
- Remember the bigger picture (many failures)
- Hard to attack (need to find AE in box)



- Harder to defend (need to **prove**: no AEs in <u>all</u> box)

Adversarial Examples - Summary

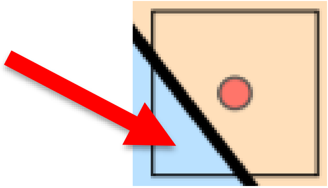
- Remember the bigger picture (many failures)
- Hard to attack (need to find AE in box)



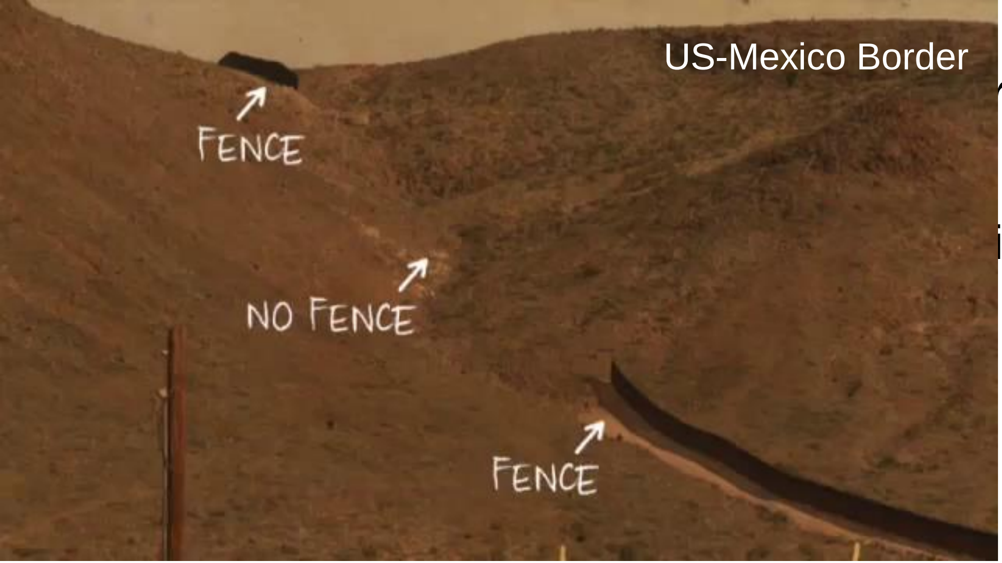
- Harder to defend (need to **prove**: very hard to find AE in box)

Adversarial Examples - Summary

- Remember the bigger picture (many failures)
- Hard to attack (need to find AE in box)

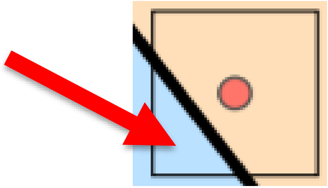


- Harder to defend (need to **Evaluate**: very hard to find AE in box)



ry

ilures)

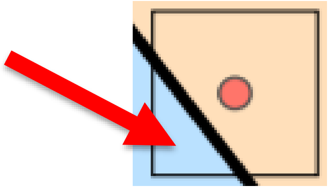


- Harder to defend (need to Evaluate: very hard to find AE in box)

source: The Fence Documentary

Adversarial Examples - Summary

- Remember the bigger picture (many failures)
- Hard to attack (need to find AE in box)

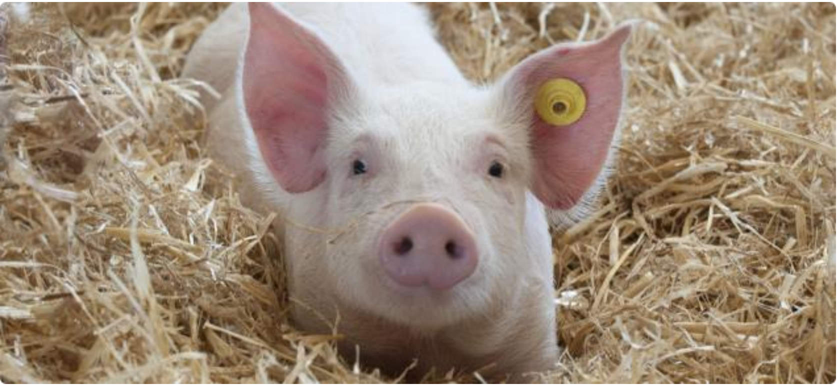


- Harder to defend (need to Evaluate: very hard to find AE in box)
- Coming next: Robustness beyond security

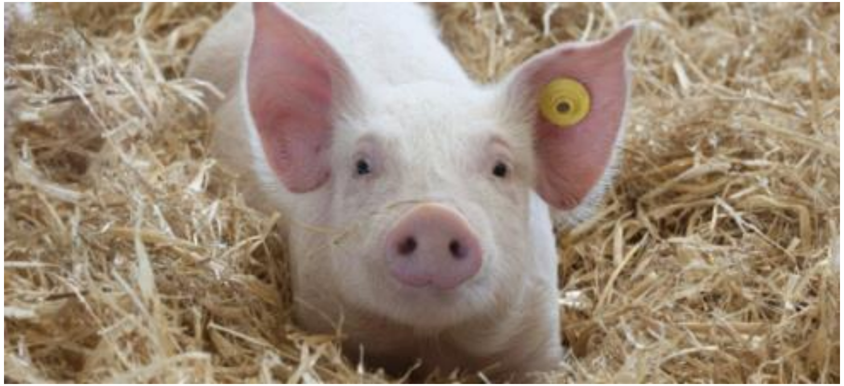
Outline

- See Adversarial Example
- Discuss what they are
- How to attack: FGSM, PGD
- How to defend: Adversarial training (AT)
- Optimization view of AT
- Black-Box attacks (transferability)
- Summary (“security”)
- Surprising “advantages” of AE (beyond security)

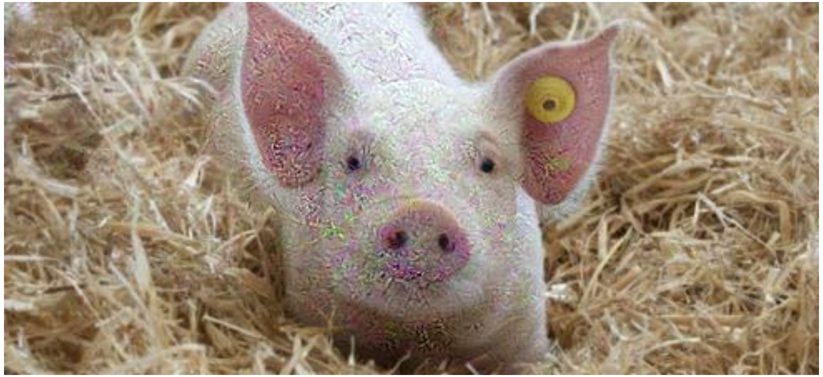
Follow the gradient w.r.t x (the input image)



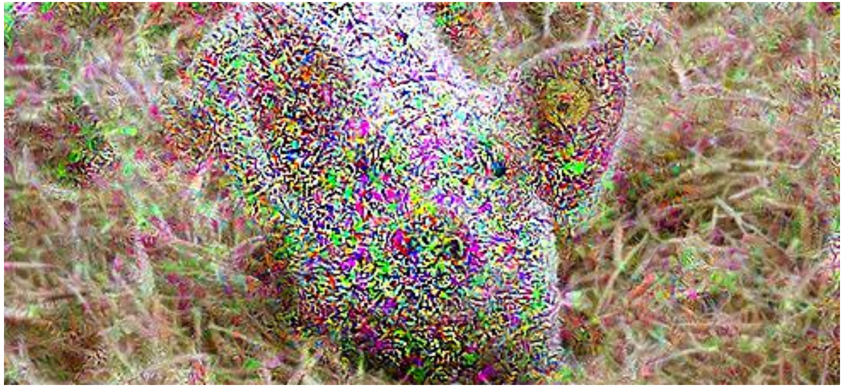
X (original): 89.7% pig



$X + \nabla_x L$: 68.6% hay

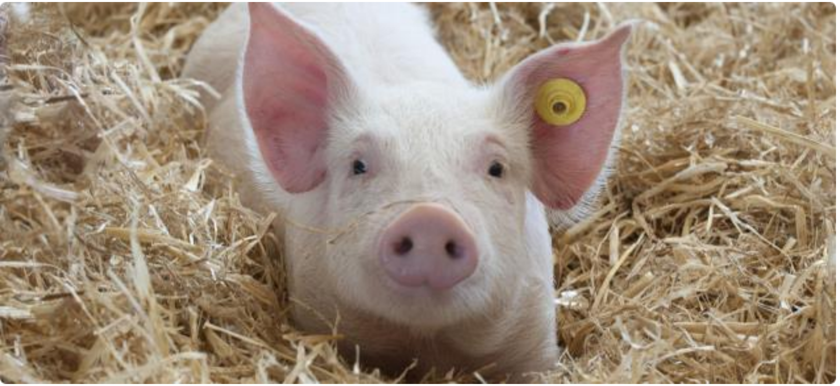


$X + 10\times\nabla_x L$: 44.7% pig

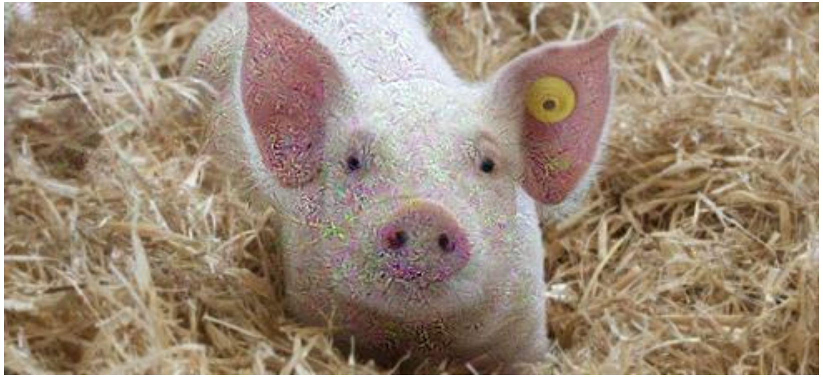


$X + 100\times\nabla_x L$: 44.8% fireguard

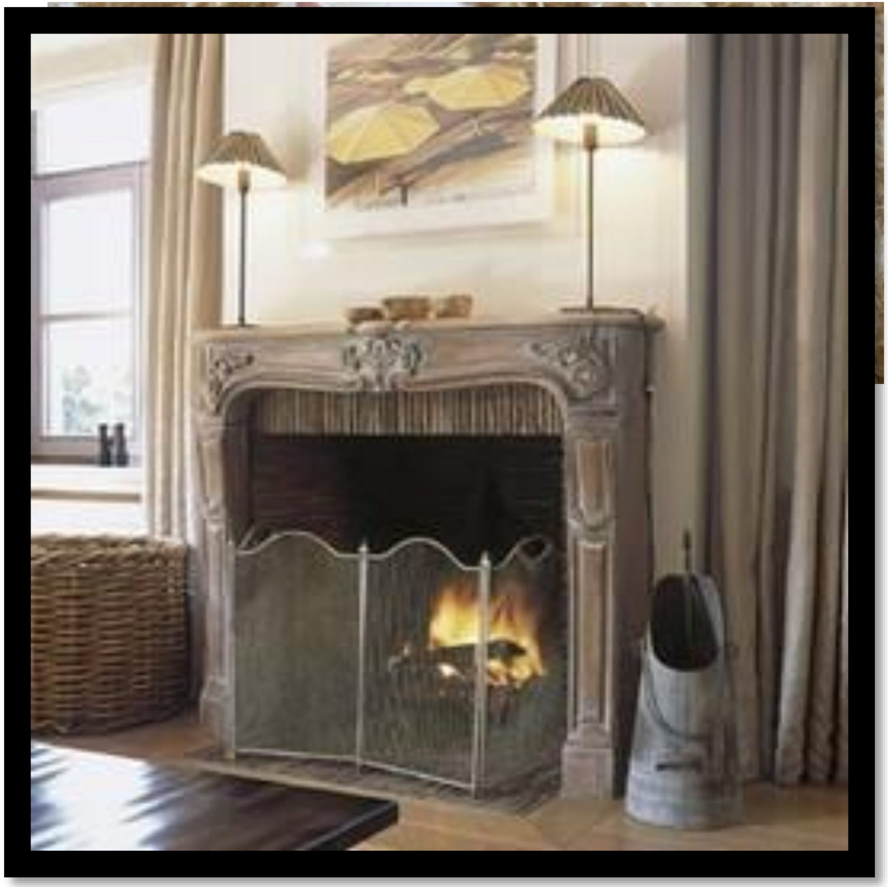
Follow the gradient w.r.t x (the input image)



X (original): 89.7% pig



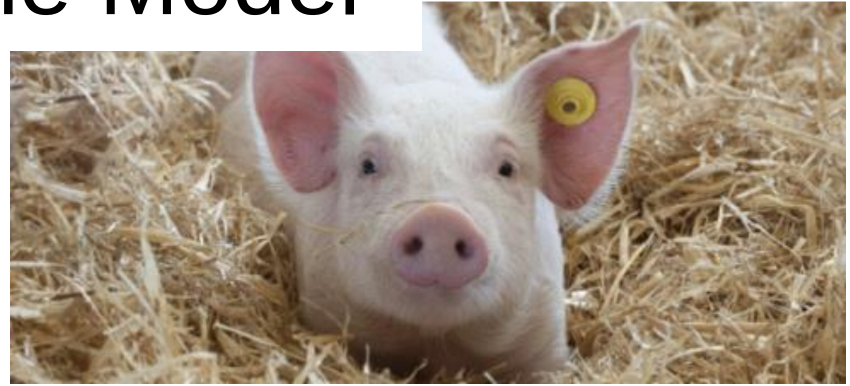
X + 10×$\nabla_x L$: 44.7% pig



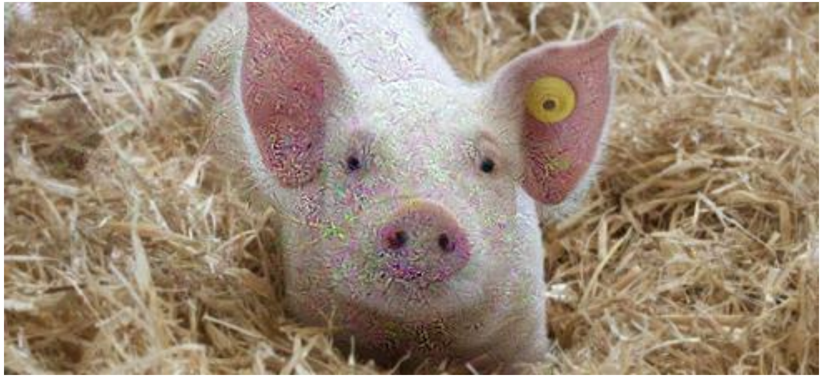
X + 100×$\nabla_x L$: 44.8% fireguard

Follow $\nabla_x L(f(x),y)$ of the Model

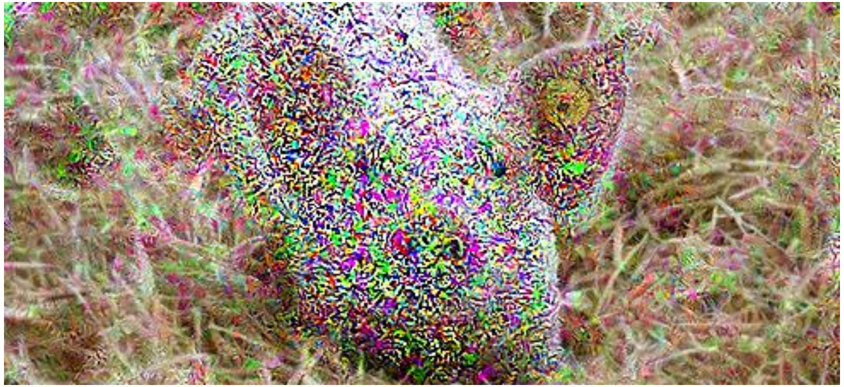
X (original): 89.7% pig



$X + \nabla_x L$: 68.6% hay



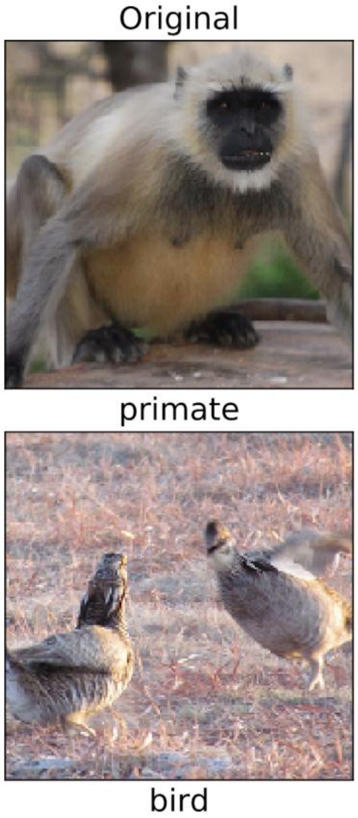
$X + 10\times\nabla_x L$: 44.7% pig



$X + 100\times\nabla_x L$: 44.8% fireguard

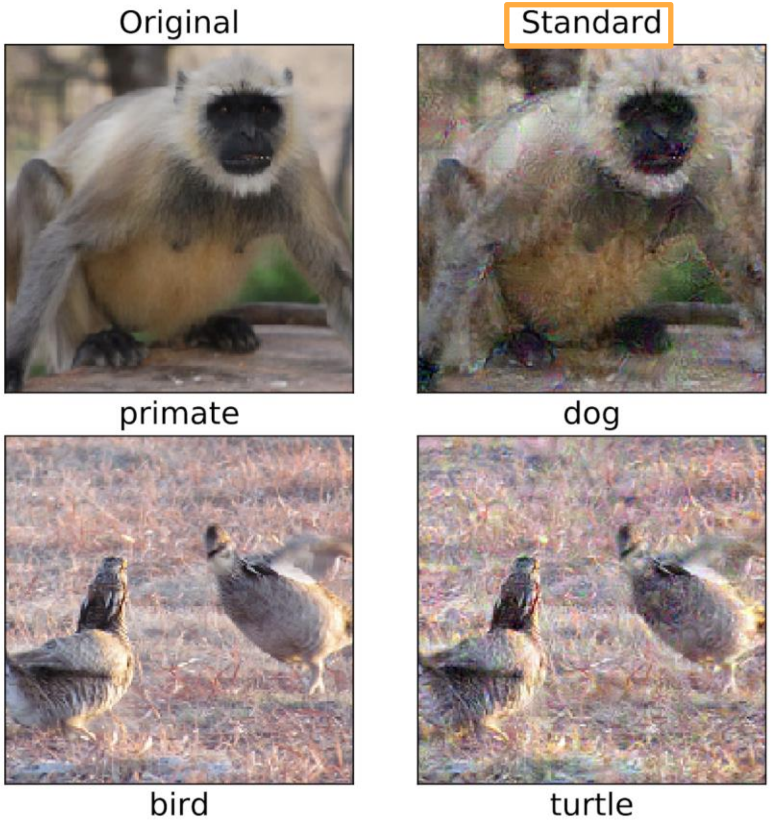
Follow $\nabla_x L(f(x),y)$ of Robust Model

Follow $\nabla_x L(f(x),y)$ of Robust Model



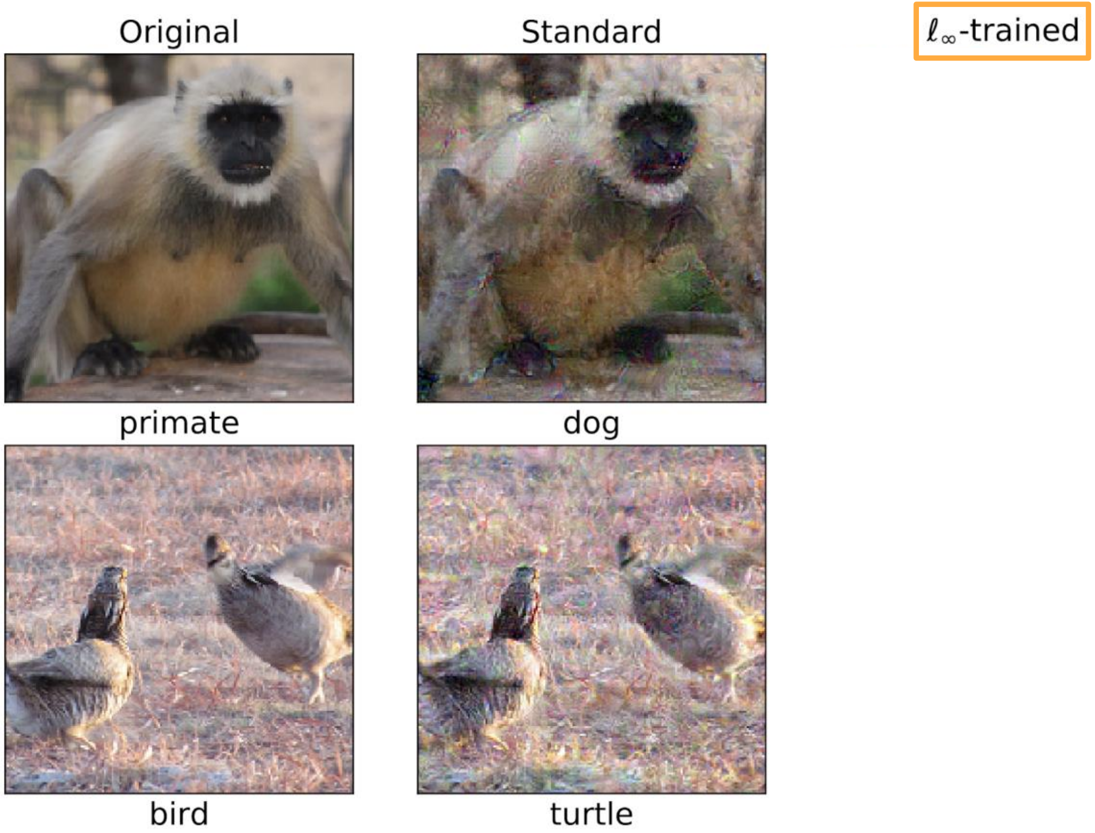
“Robustness May Be at Odds with Accuracy” (Tsipras et al. 2018)

Follow $\nabla_x L(f(x),y)$ of Robust Model



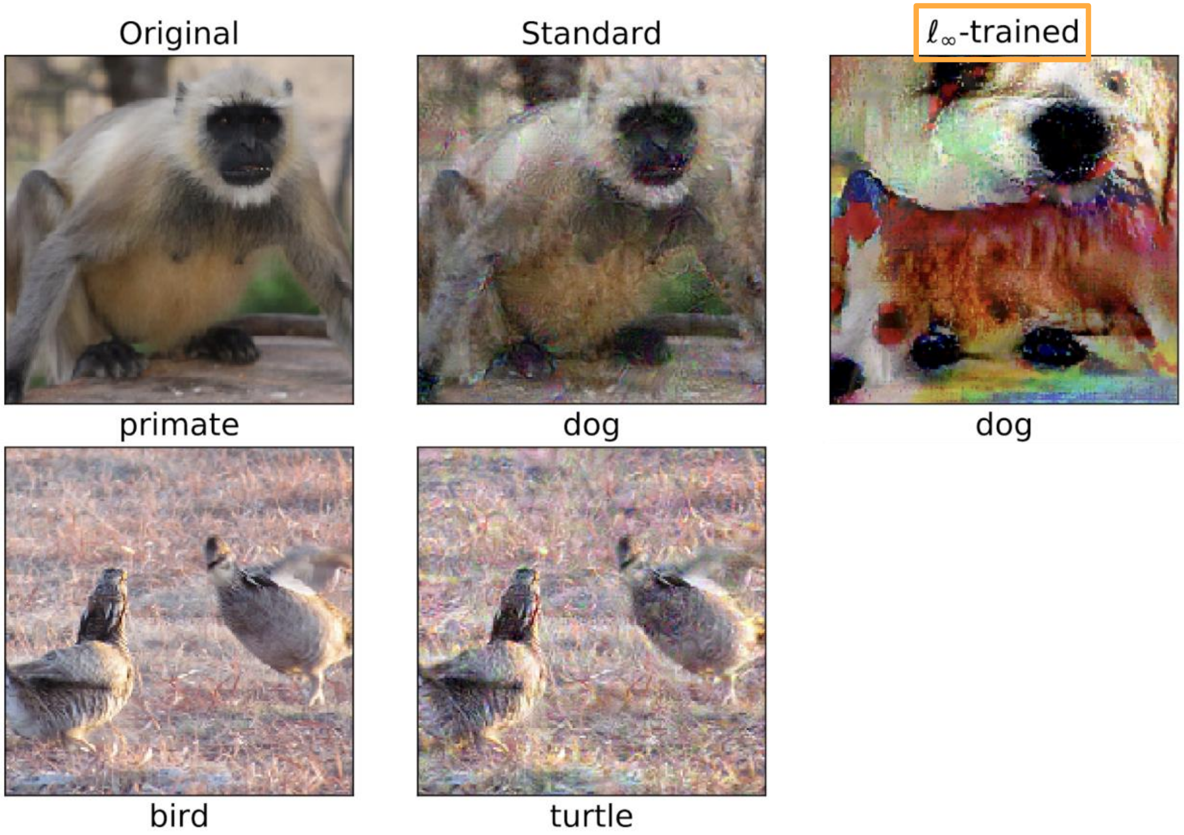
“Robustness May Be at Odds with Accuracy” (Tsipras et al. 2018)

Follow $\nabla_x L(f(x),y)$ of Robust Model



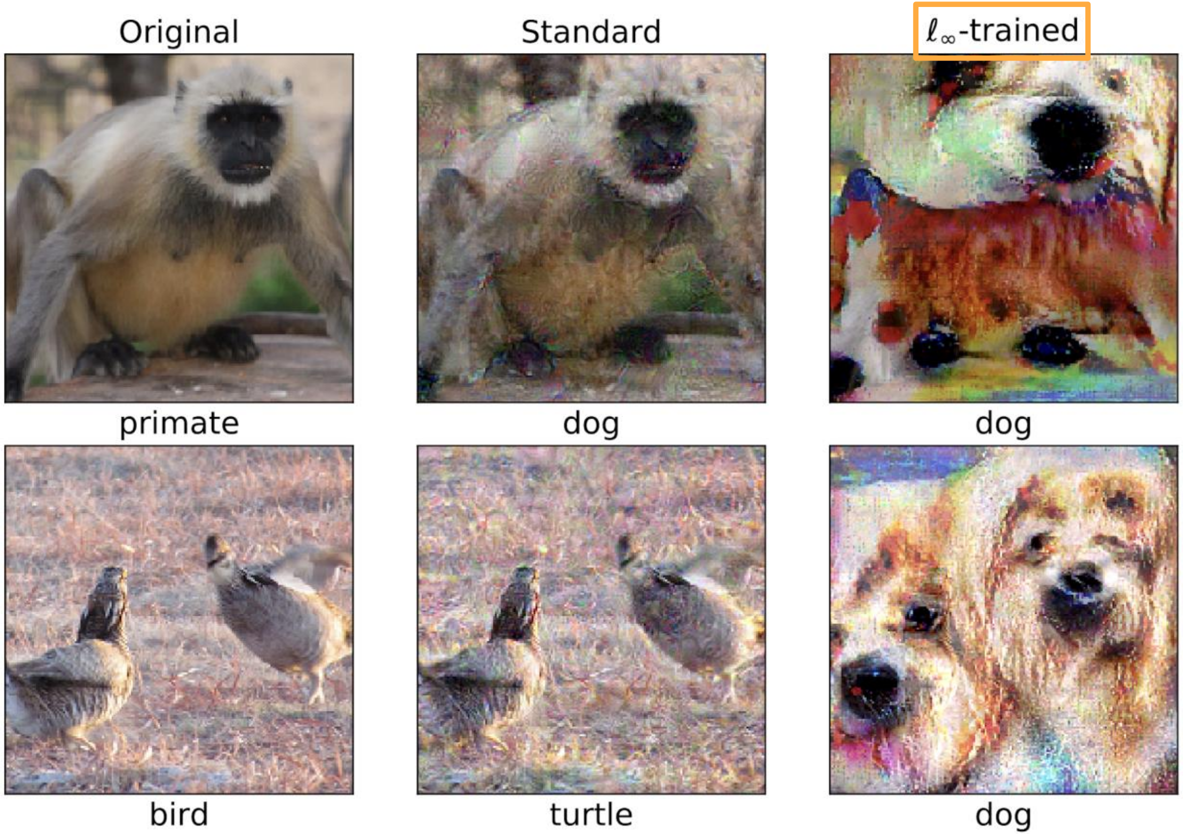
“Robustness May Be at Odds with Accuracy” (Tsipras et al. 2018)

Follow $\nabla_x L(f(x),y)$ of Robust Model



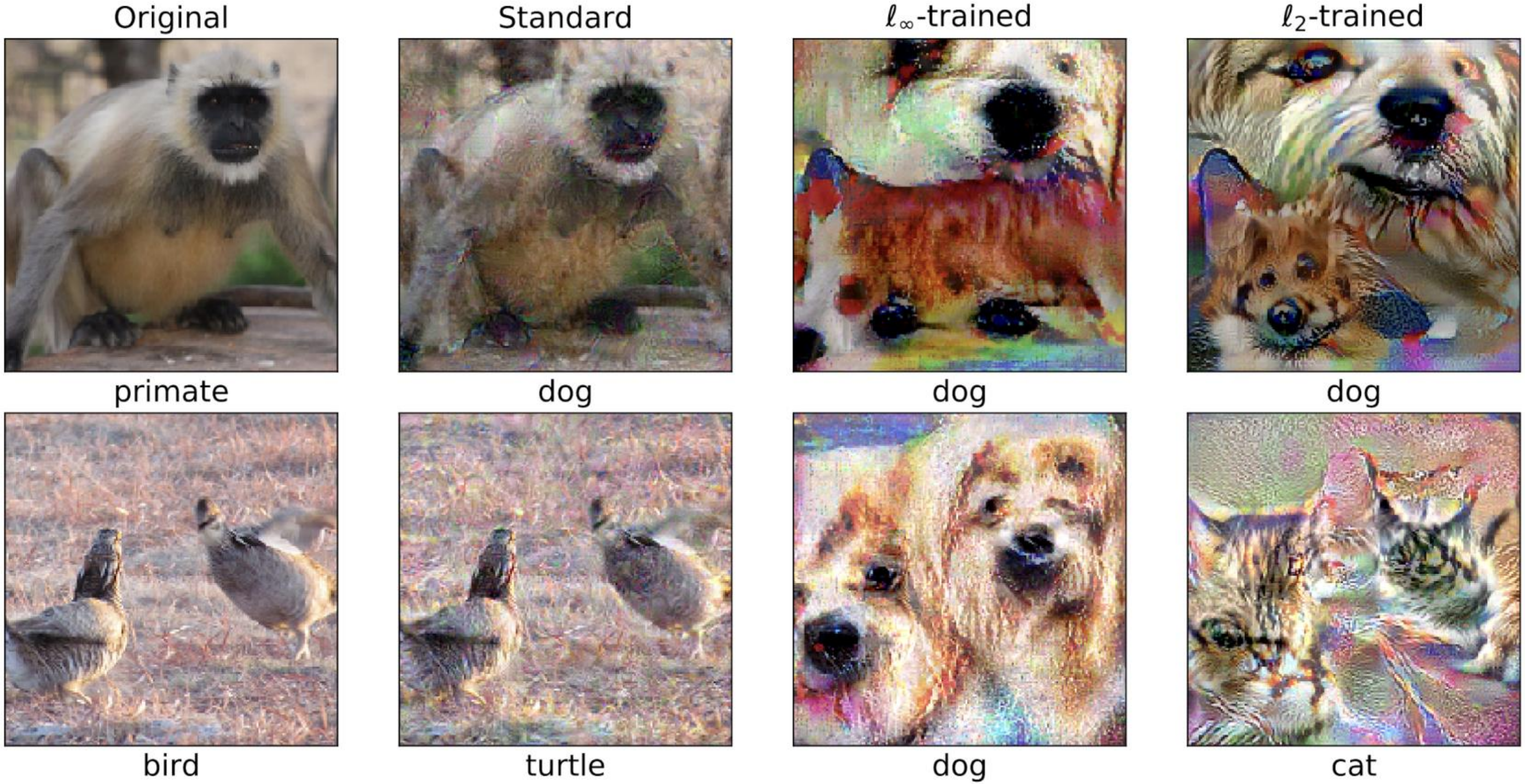
“Robustness May Be at Odds with Accuracy” (Tsipras et al. 2018)

Follow $\nabla_x L(f(x),y)$ of Robust Model



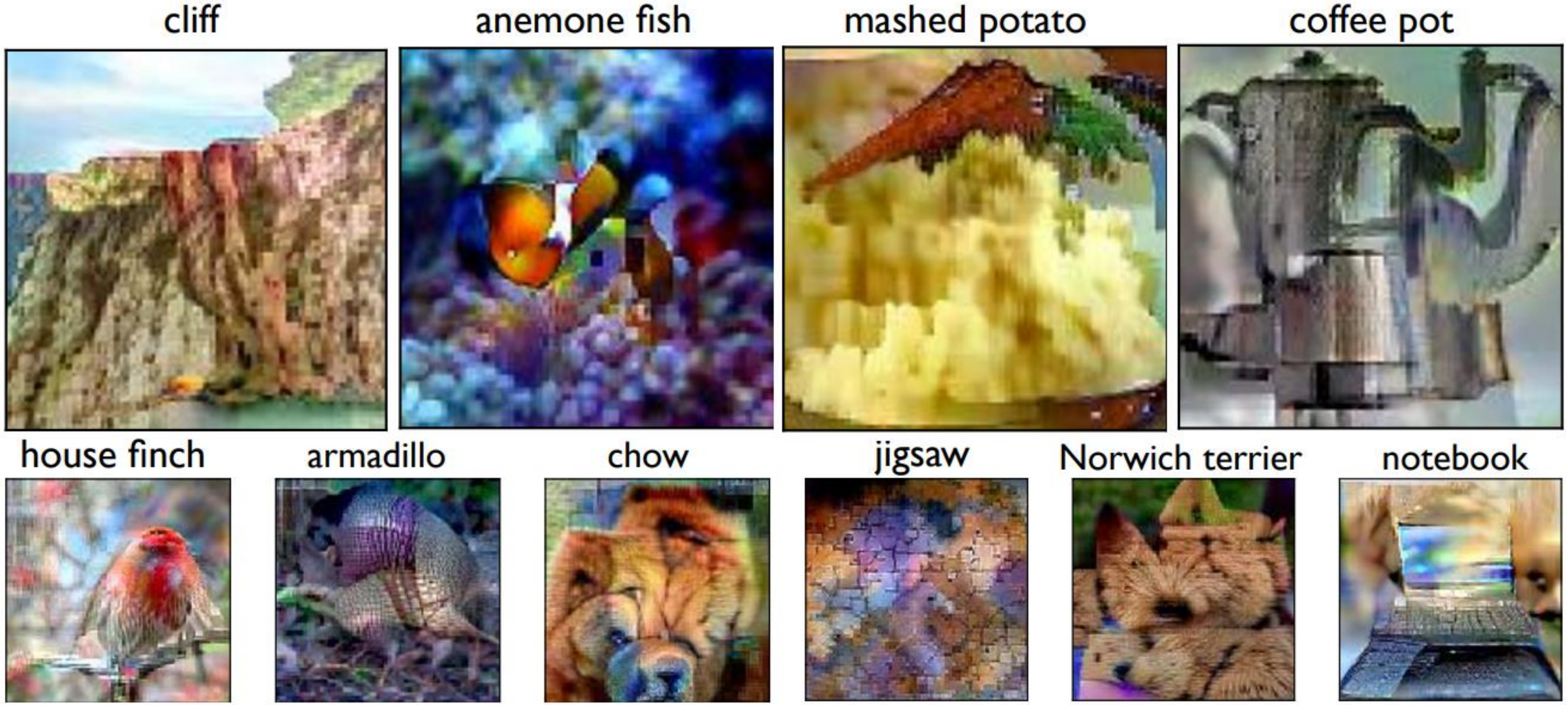
“Robustness May Be at Odds with Accuracy” (Tsipras et al. 2018)

Follow $\nabla_x L(f(x),y)$ of Robust Model



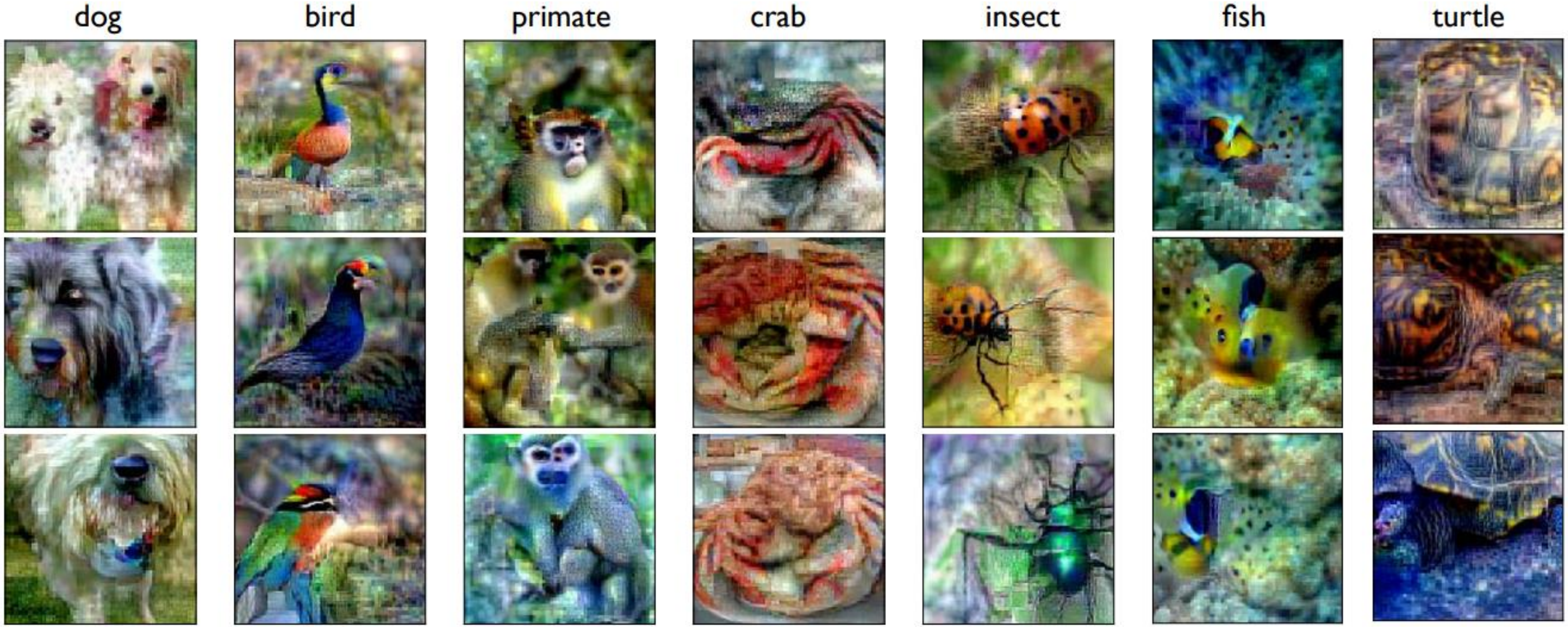
“Robustness May Be at Odds with Accuracy” (Tsipras et al. 2018)

Image synthesis with Robust Classifer



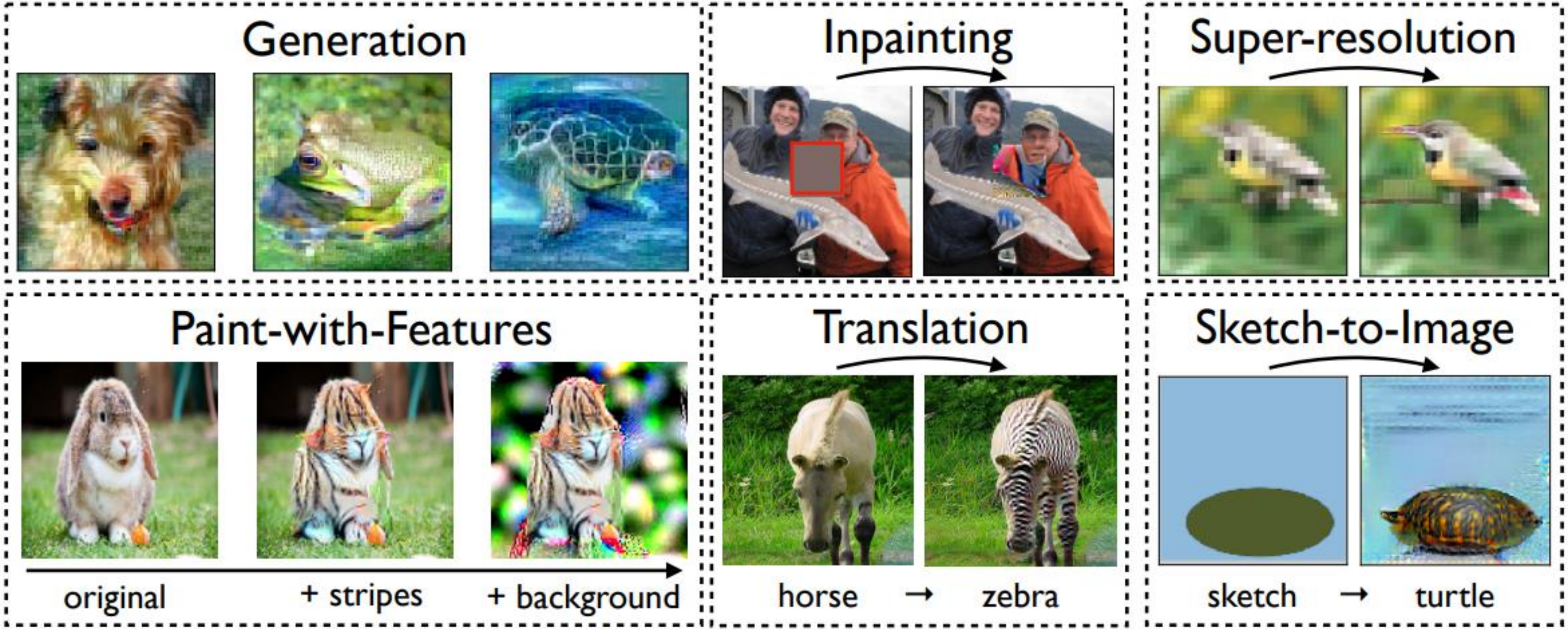
Santurkar et al. 2019, “Image Synthesis with a Single (Robust) Classifier”

Image synthesis with Robust Classifer



Santurkar et al. 2019, “Image Synthesis with a Single (Robust) Classifier”

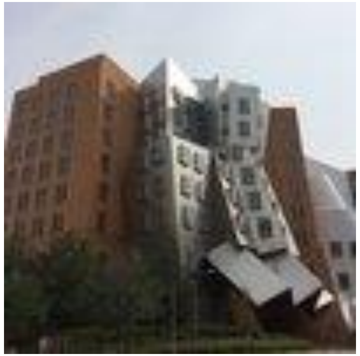
Image synthesis with Robust Classifer



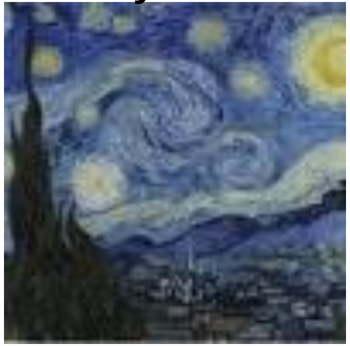
Santurkar et al. 2019, “Image Synthesis with a Single (Robust) Classifier”

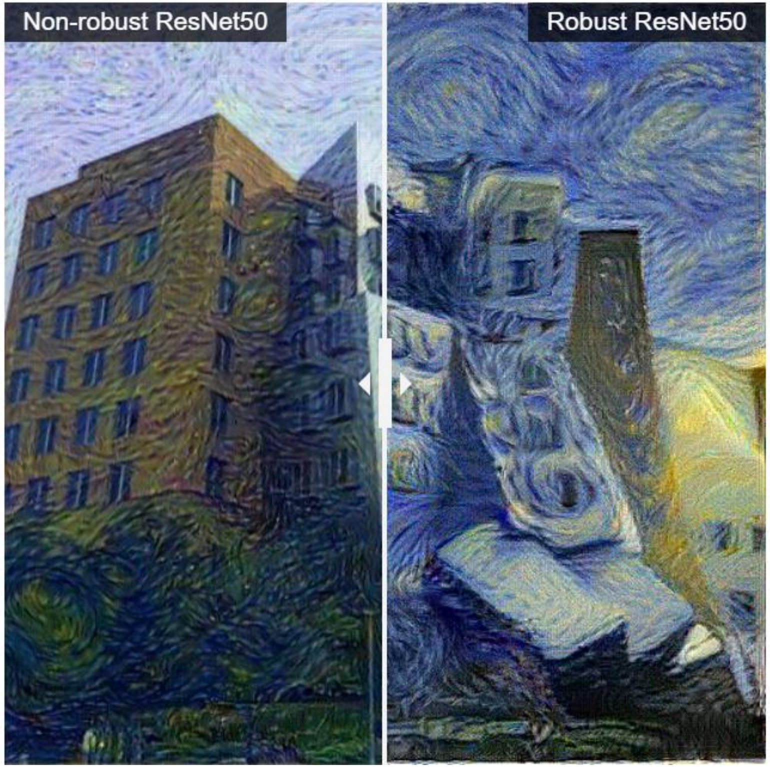
Style Transfer with Robust Model

Content



Style





Nakano, "A Discussion of 'Adversarial Examples Are Not Bugs, They Are Features': Adversarially Robust Neural Style Transfer", Distill, 2019.

What have we learnt today?

- Saw a few Adversarial Examples
- Discussed what they are
- How to attack: FGSM, PGD
- How to “defend”: Adversarial training (AT)
- Optimization view of AT
- Black-Box attacks (transferability)
- Security-wise summary
- Surprising Visual properties of robust models (beyond security)

What have we learnt today?

- Saw a few Adversarial Examples
- Discussed what they are
- How to attack: FGSM, PGD
- How to “defend”: Adversarial training (AT)
- Optimization view of AT
- Black-Box attacks (transferability)
- Security-wise summary
- Surprising Visual properties of robust models (beyond security)

Monday:

Detection and Segmentation

What have we learnt today?

Now!

Projects

- Saw a few Adversarial Examples
- Discussed what they are
- How to attack: FGSM, PGD
- How to “defend”: Adversarial training (AT)
- Optimization view of AT
- Black-Box attacks (transferability)
- Security-wise summary
- Surprising Visual properties of robust models (beyond security)