Attack and Defense

Hung-yi Lee

Motivation

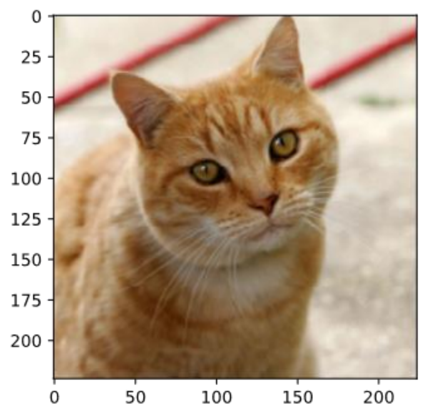
- We seek to deploy machine learning classifiers not only in the labs, but also in real world.
- The classifiers that are robust to noises and work "most of the time" is not sufficient. 光是強還不夠
- We want the classifiers that are robust the inputs that are built to fool the classifier. 應付來自人類的惡意
- Especially useful for spam classification, malware detection, network intrusion detection, etc.

Attack

https://www.darksword-armory.com/wp-content/uploads/2014/09/two-handed-danish-sword-medieval-weapon-1352-3.jpg

What do we want to do?

Original Image



x^0

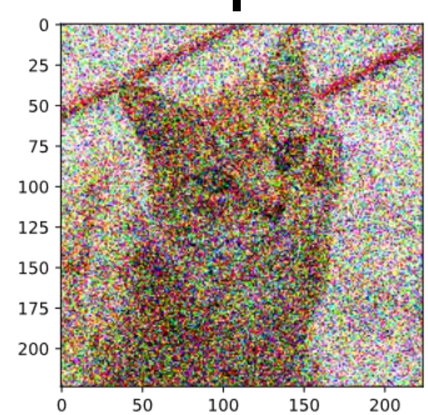
Network

Something Else

~~Tiger Cat~~

0.64

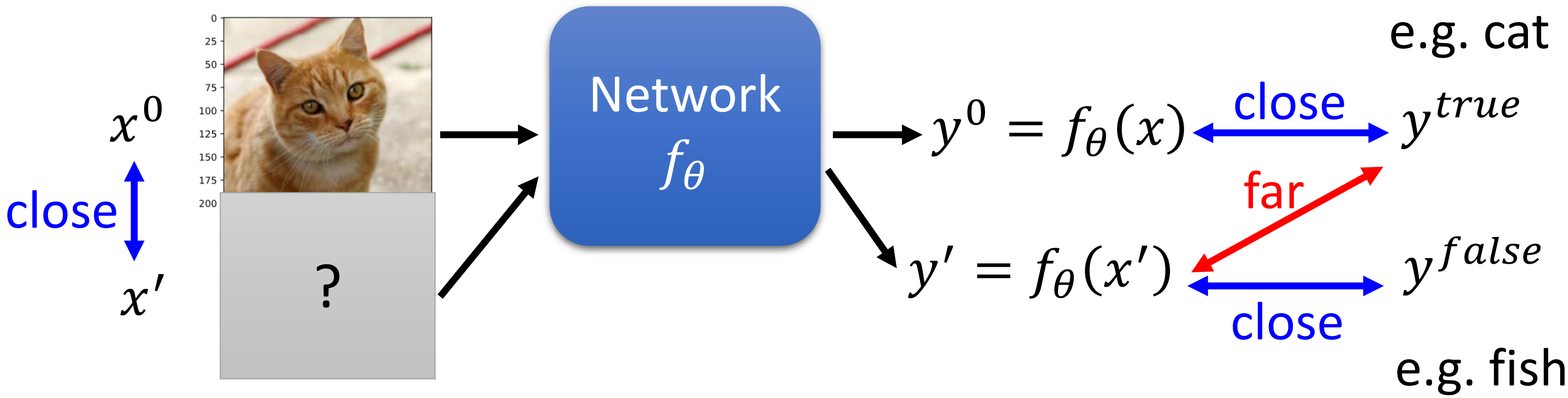
$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix} + \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \vdots \end{bmatrix}$$



Attacked Image

$$x' = x^0 + \Delta x$$

Loss Function for Attack



- ***Training:*** $L_{train}(\theta) = C(y^0, y^{true})$ x fixed
- ***Non-targeted Attack:*** $L(x') = -C(y', y^{true})$ θ fixed
- ***Targeted Attack:***

$$L(x') = -C(y', y^{true}) + C\left(y', y^{false}\right)$$

- ***Constraint:*** $d(x^0, x') \le \varepsilon$ 不要被發現

Constraint $d(x^0, x') \leq \varepsilon$

$$\underbrace{\begin{bmatrix} x_1' \\ x_2' \\ x_3' \\ \vdots \end{bmatrix}}_{x'} - \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix}}_{x^0} = \underbrace{\begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \vdots \end{bmatrix}}_{\Delta x}$$

- L2-norm

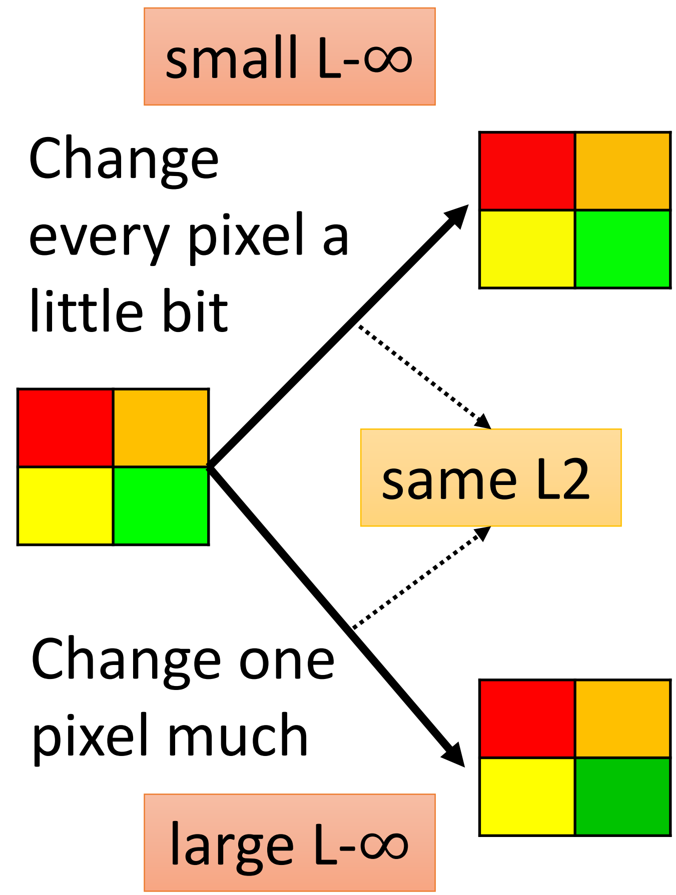
$$d(x^0, x') = \|x^0 - x'\|_2$$

$$= (\Delta x_1)^2 + (\Delta x_2)^2 + (\Delta x_3)^2 \cdots$$

- L-infinity

$$d(x^0, x') = \|x^0 - x'\|_\infty$$

$$= max\{\Delta x_1, \Delta x_2, \Delta x_3, \cdots\}$$



How to Attack

Just like training a neural network, but network parameter θ is replaced with input x'

$$x^* = arg \min L(x')$$

- Gradient Descent

Start from original image x^0

For t = 1 to T

$x^t \leftarrow x^{t-1} - \eta \nabla L(x^{t-1})$

If $d(x^0, x^t) > \varepsilon$

$x^t \leftarrow fix(x^t)$

$$\nabla L(x) = \begin{bmatrix} \partial L(x)/\partial x_1 \\ \partial L(x)/\partial x_2 \\ \partial L(x)/\partial x_3 \\ \vdots \end{bmatrix}$$

How to Attack

Just like training a neural network, but network parameter θ is replaced with input x'

$$x^* = arg \min_{d(x^0, x') \le \varepsilon} L(x')$$

- Gradient Descent (Modified Version)

Start from original image x^0

For t = 1 to T

$x^t \leftarrow x^{t-1} - \eta \nabla L(x^{t-1})$

If $d(x^0, x^t) > \varepsilon$

$x^t \leftarrow fix(x^t)$

def $fix(x^t)$

For all x fulfill $d(x^0, x) \le \varepsilon$

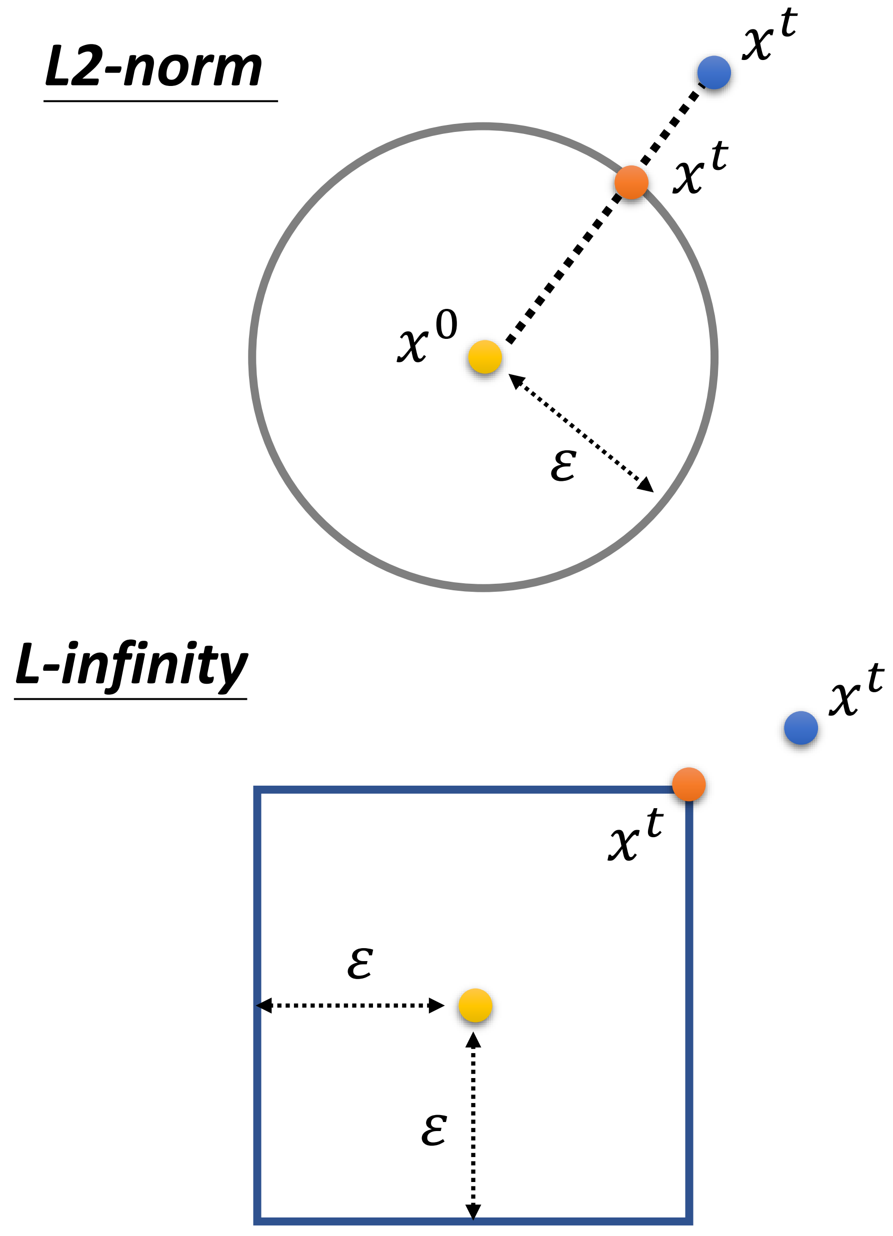
Return the one closest to x^t

How to Attack

def $fix(x^t)$

For all x fulfill $d(x^0, x) \leq \varepsilon$

Return the one closest to x^t



$$L(x') = -C(y', y^{true}) + C(y', y^{false})$$

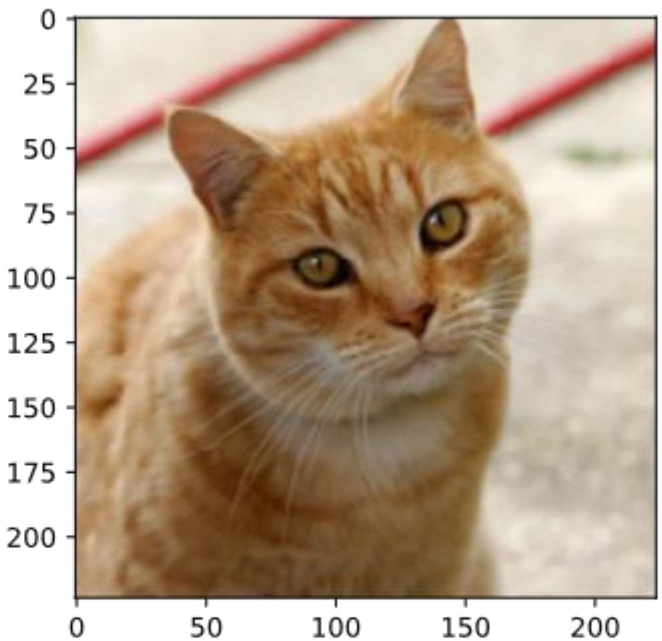
Example

True = Tiger cat

False = Star Fish

f = ResNet-50

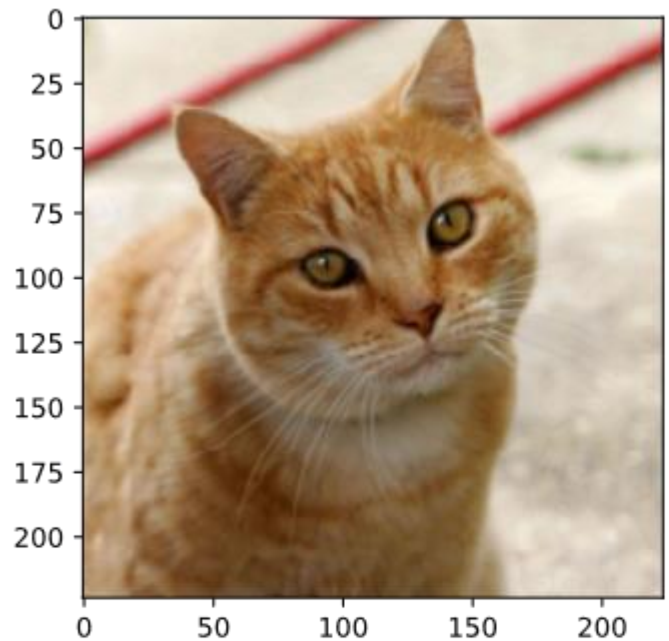
Original Image



Tiger Cat

0.64

Attacked Image

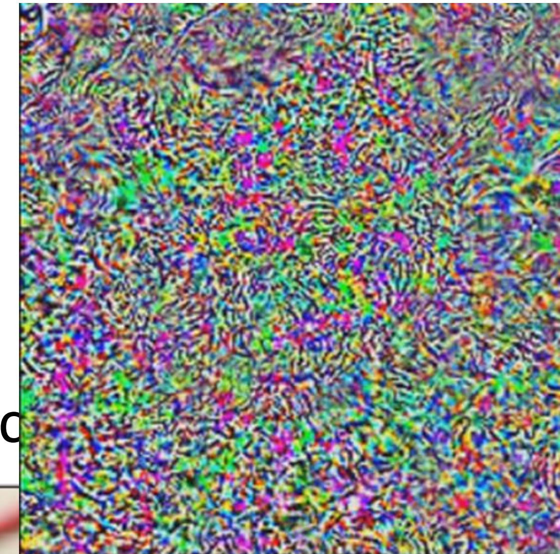


Star Fish

1.00

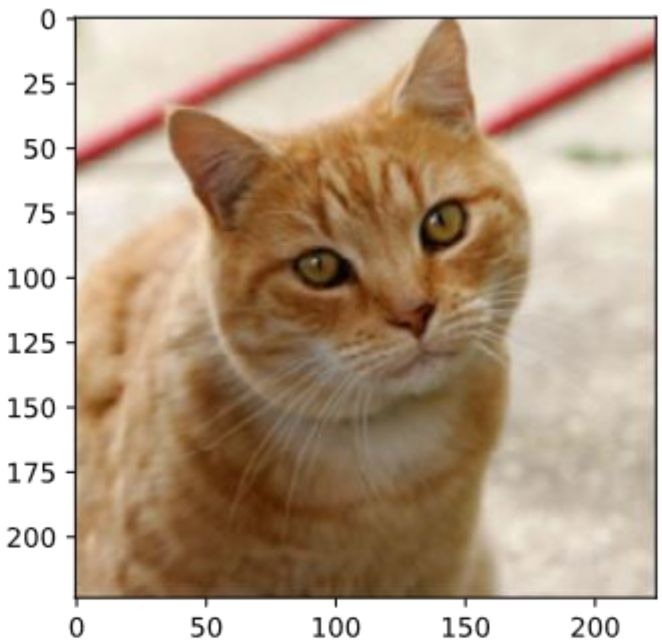
Example

=



50x

Original Image

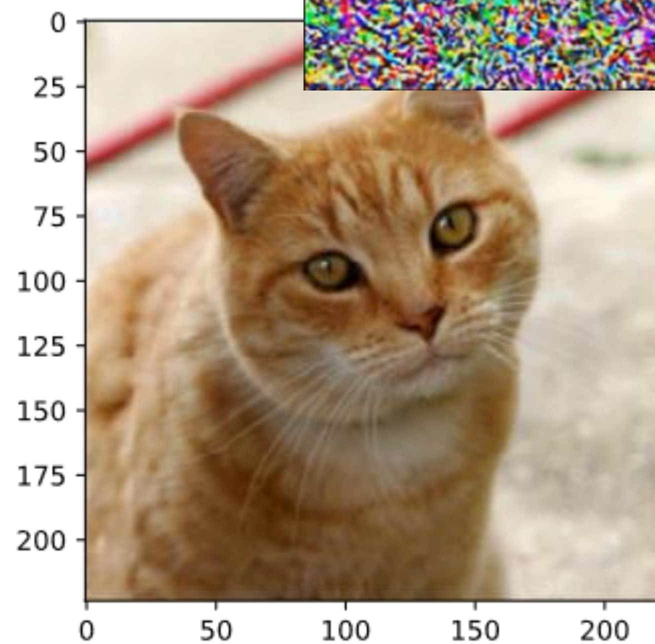


Tiger Cat

0.64

-

Attac



Star Fish

1.00

$$L(x') = -C(y', y^{true}) + C(y', y^{false})$$

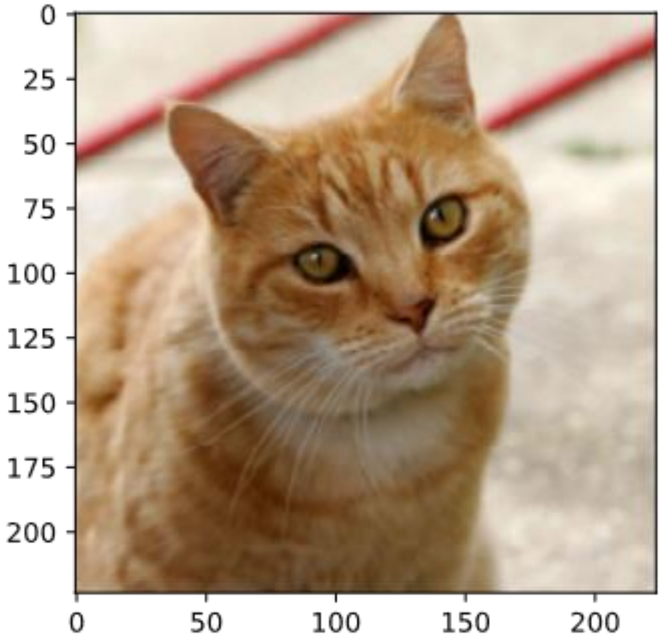
Example

True = Tiger cat

False = Keyboard

$f =$ ResNet-50

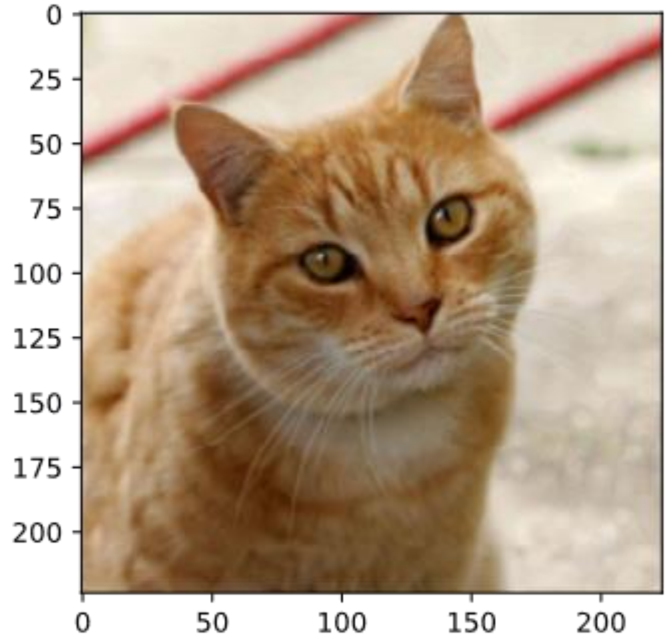
Original Image



Tiger Cat

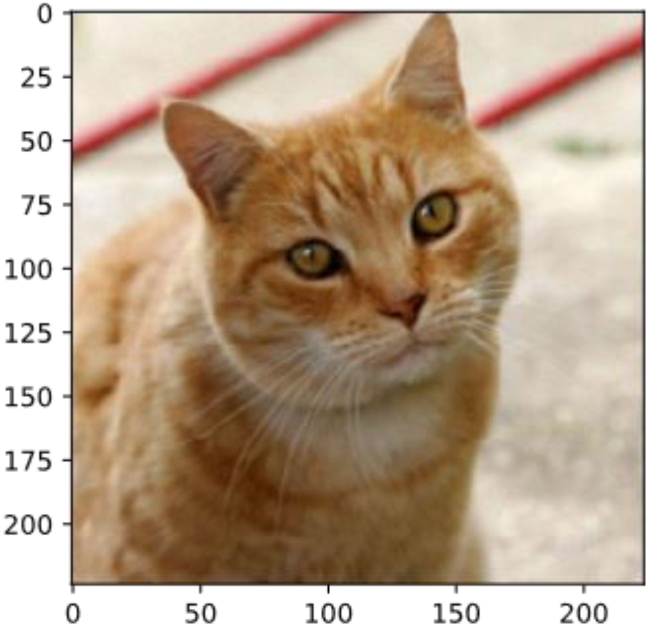
0.64

Attacked Image

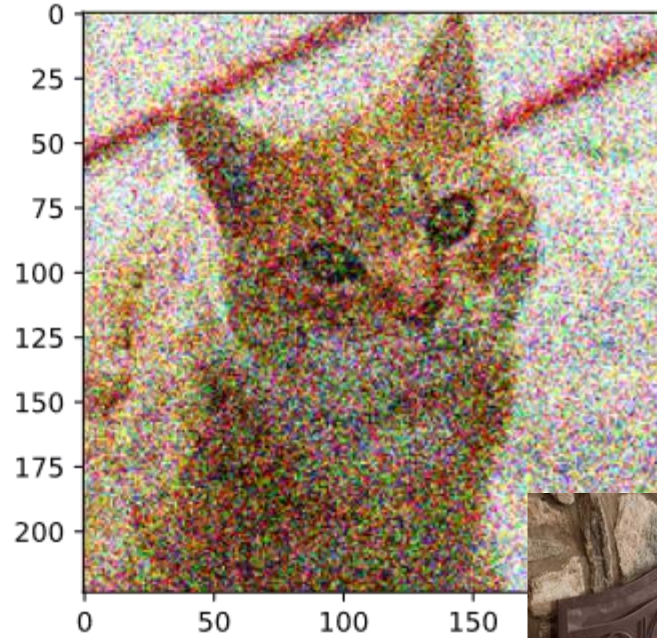


Keyboard

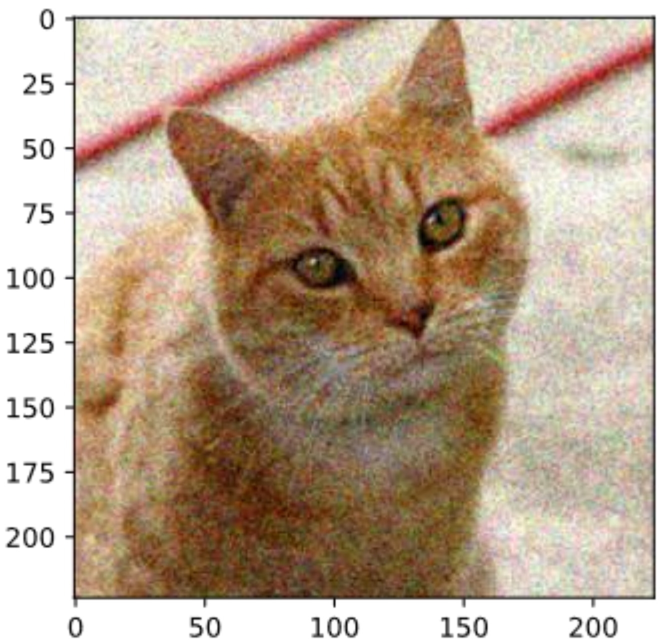
0.98



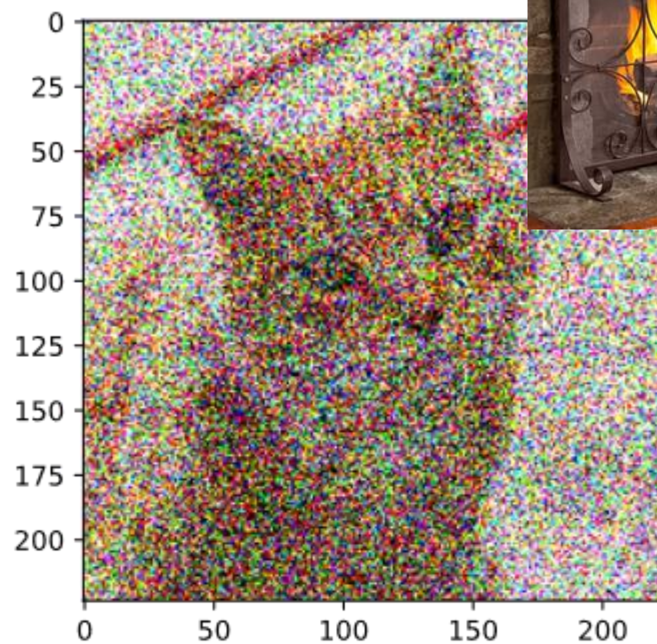
tiger cat



Persian cat

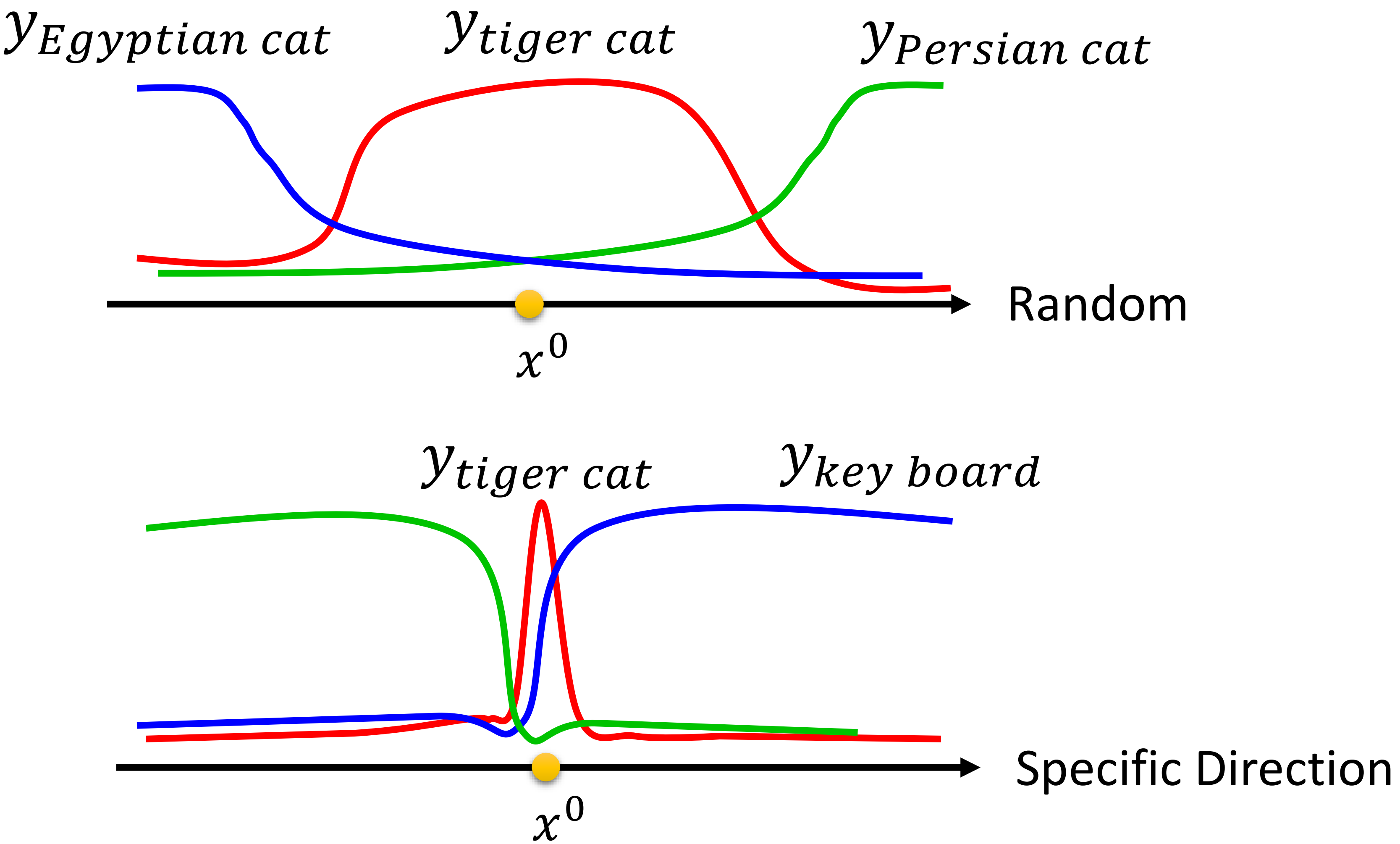


tabby cat



fire screen

What happened?



Attack Approaches

- FGSM (https://arxiv.org/abs/1412.6572)
- Basic iterative method (https://arxiv.org/abs/1607.02533)
- L-BFGS (https://arxiv.org/abs/1312.6199)
- Deepfool (https://arxiv.org/abs/1511.04599)
- JSMA (https://arxiv.org/abs/1511.07528)
- C&W (https://arxiv.org/abs/1608.04644)
- Elastic net attack (https://arxiv.org/abs/1709.04114)
- Spatially Transformed (https://arxiv.org/abs/1801.02612)
- One Pixel Attack (https://arxiv.org/abs/1710.08864)
- …… only list a few

https://sites.google.com/site/pkms20152a17/_/rsrc/1448428701742/home/125986076.jpg

Attack Approaches

$$x^* = arg \underset{d(x^0, x') \le \varepsilon}{\boxed{\min}} L(x')$$

Different optimization methods

Different constraints

- Fast Gradient Sign Method (FGSM)

$$x^* \leftarrow x^0 - \varepsilon \Delta x$$

$$\Delta x = \begin{bmatrix} sign(\partial L / \partial x_1) \\ sign(\partial L / \partial x_2) \\ sign(\partial L / \partial x_3) \\ \vdots \end{bmatrix}$$

only have **+1** or **-1**

https://sites.google.com/site/pkms20152a17/_/rsrc/1448428701742/home/125986076.jpg

Attack Approaches

$$x^* = arg \underset{\underline{d(x^0,x')\le\varepsilon}}{\boxed{min}} L(x')$$

Different optimization methods

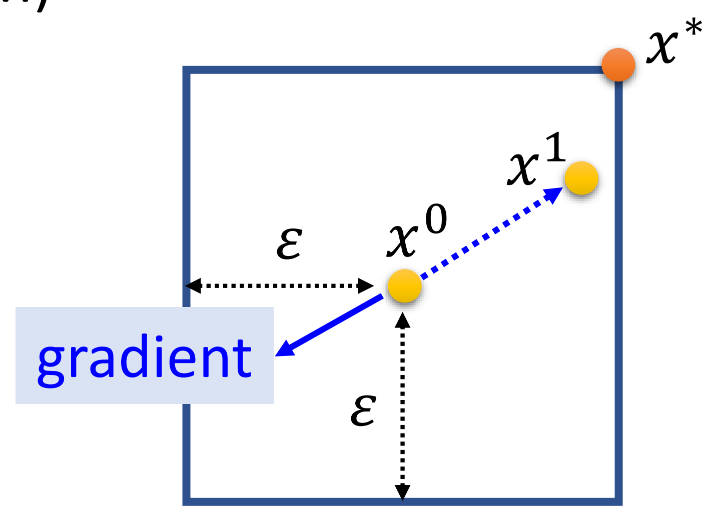
Different constraints

- Fast Gradient Sign Method (FGSM)

$$x^* \leftarrow x^0 - \varepsilon \Delta x$$

$$\Delta x = \begin{bmatrix} sign(\partial L/\partial x_1) \\ sign(\partial L/\partial x_2) \\ sign(\partial L/\partial x_3) \\ \vdots \end{bmatrix}$$

only have **+1** or **-1**



L-infinity

https://sites.google.com/site/pkms20152a17/_/rsrc/1448428701742/home/125986076.jpg

Attack Approaches

Different optimization methods

$$x^* = arg \underbrace{\boxed{\min}}_{\underline{d(x^0,x')\le\varepsilon}} L(x')$$

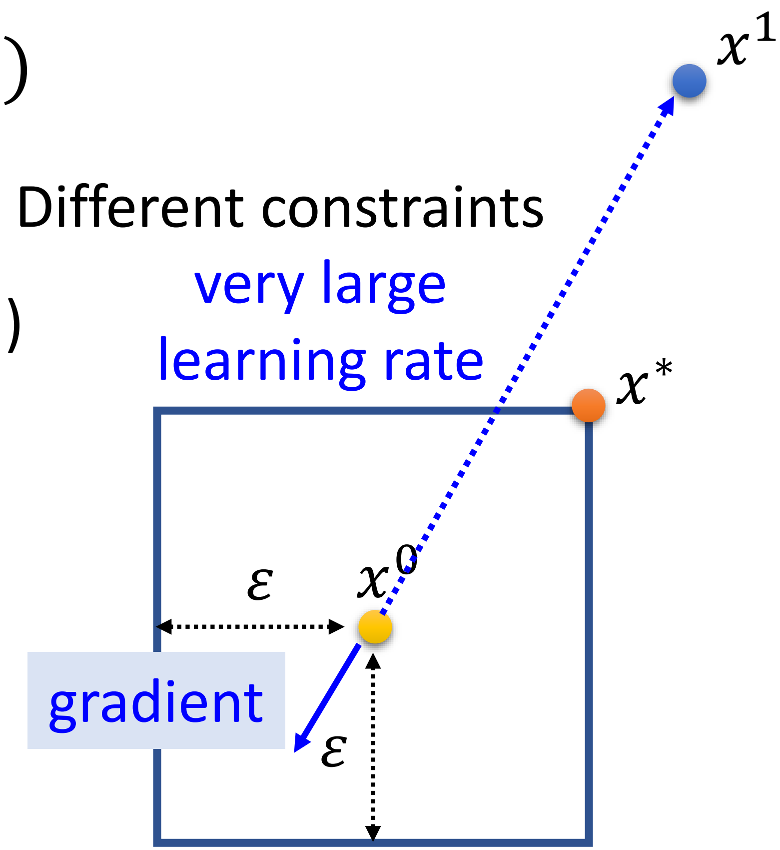
Different constraints

- Fast Gradient Sign Method (FGSM)

$$x^* \leftarrow x^0 - \varepsilon\Delta x$$

$$\Delta x = \begin{bmatrix} sign(\partial L/\partial x_1) \\ sign(\partial L/\partial x_2) \\ sign(\partial L/\partial x_3) \\ \vdots \end{bmatrix}$$

only have **+1** or **-1**



L-infinity

White Box v.s. Black Box

- In the previous attack, we fix network parameters θ to find optimal x'.
- To attack, we need to know network parameters θ
  - This is called ***White Box Attack***.
- Are we safe if we do not release model? ☺
  - You cannot obtain model parameters in most on-line API.
- No, because ***Black Box Attack*** is possible. ☹

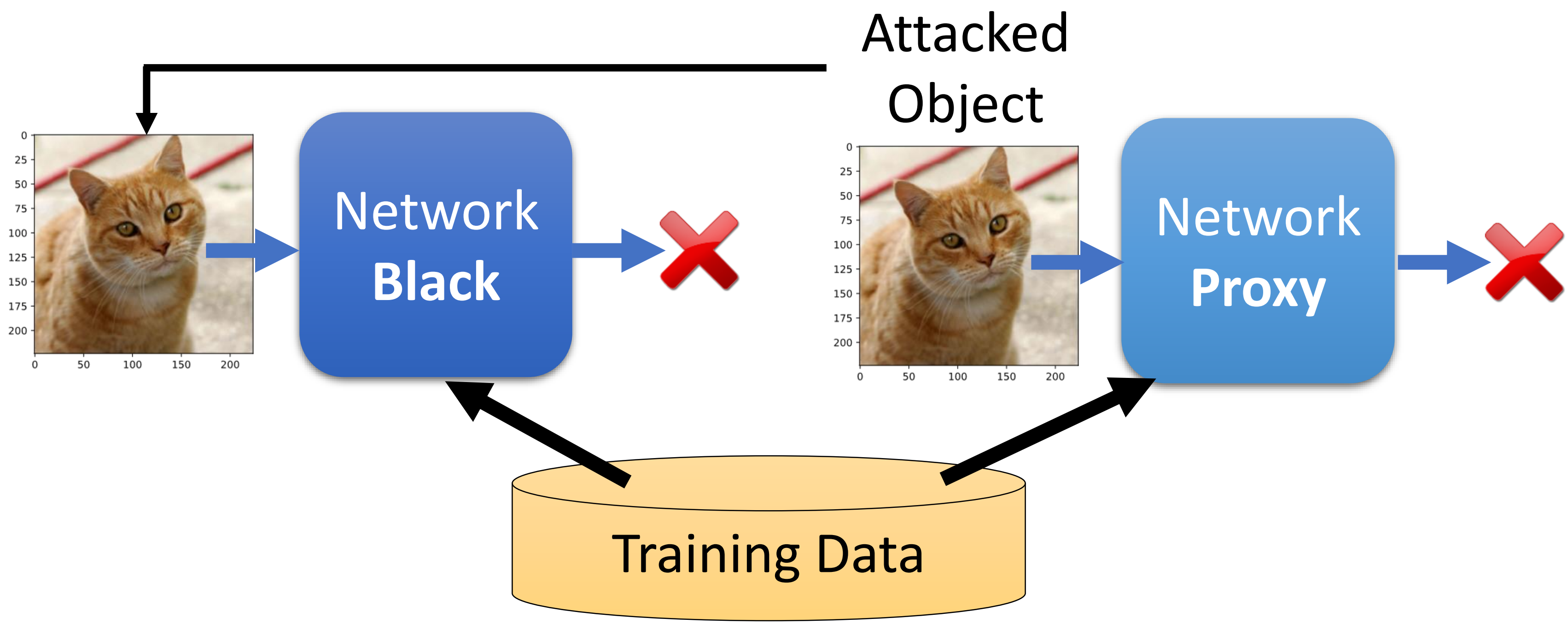
Black Box Attack

If you have the training data of the target network

Train a proxy network yourself

Using the proxy network to generate attacked objects

Otherwise, obtaining input-output pairs from target network



Black Box Attack

- If you have the training data of the target network
    - Train a proxy network yourself
    - Using the proxy network to generate attacked objects
- Otherwise, obtaining input-output pairs from target network

Black

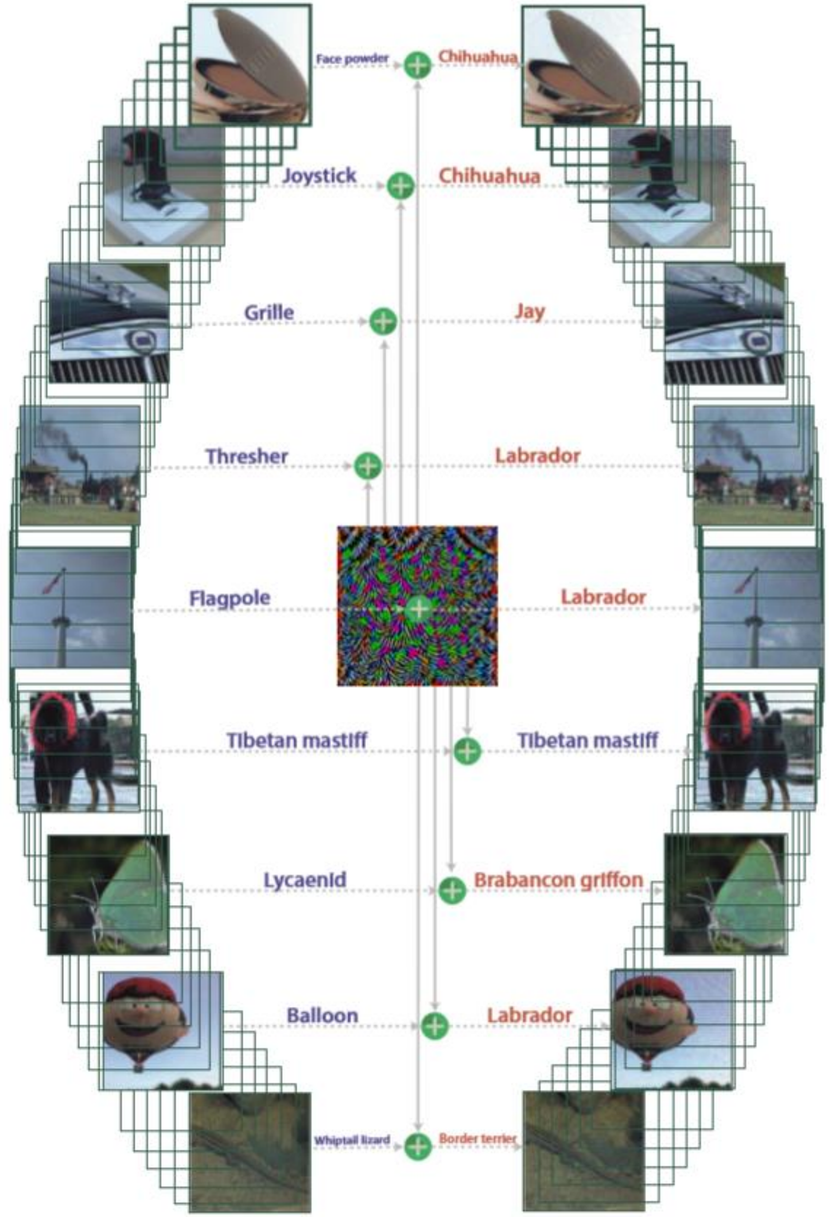
Proxy

	ResNet-152	ResNet-101	ResNet-50	VGG-16	GoogLeNet
ResNet-152	4%	13%	13%	20%	12%
ResNet-101	19%	4%	11%	23%	13%
ResNet-50	25%	19%	5%	25%	14%
VGG-16	20%	16%	15%	1%	7%
GoogLeNet	25%	25%	17%	19%	1%

https://arxiv.org/pdf/1611.02770.pdf

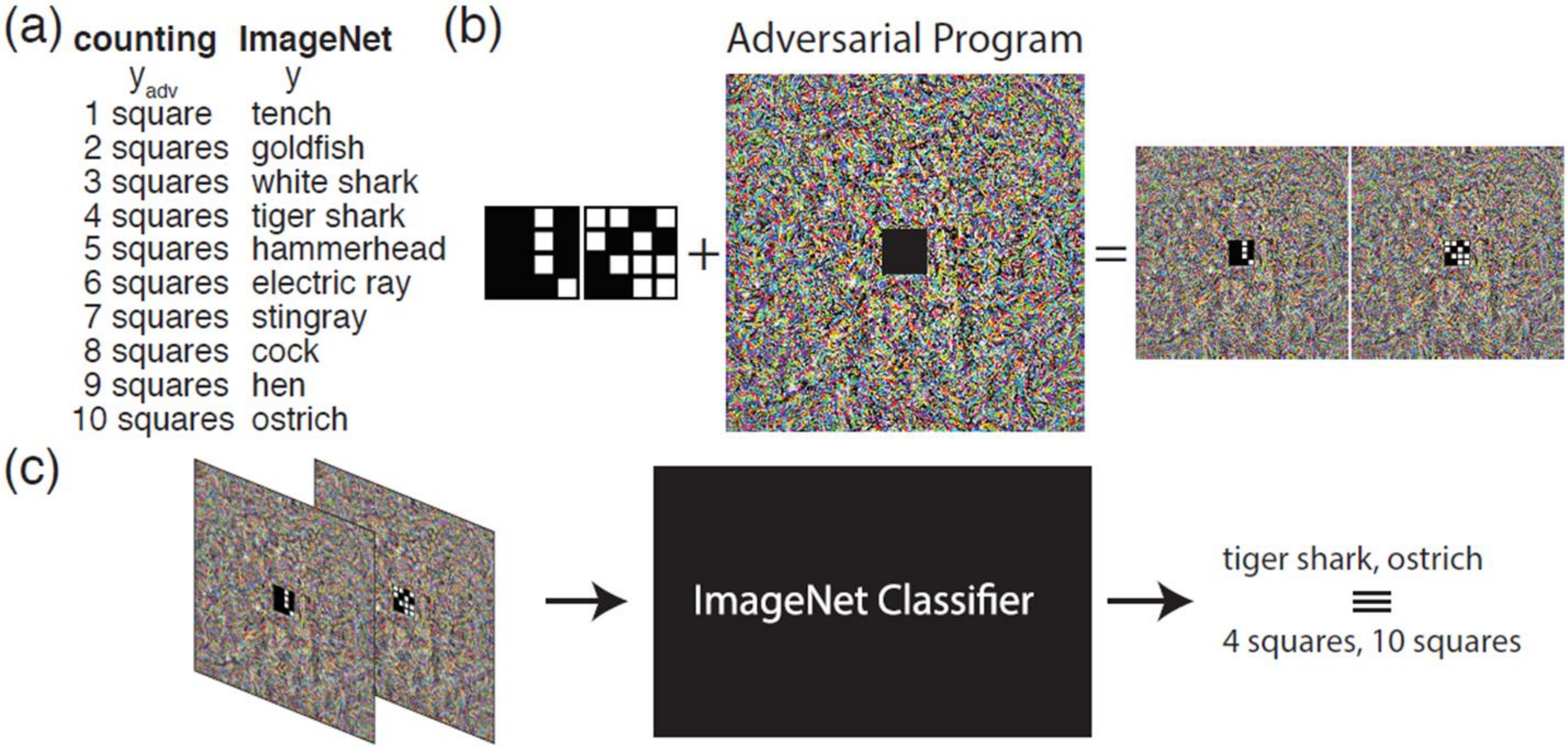
Universal Adversarial Attack

https://arxiv.org/abs/1610.08401



Black Box Attack is also possible!

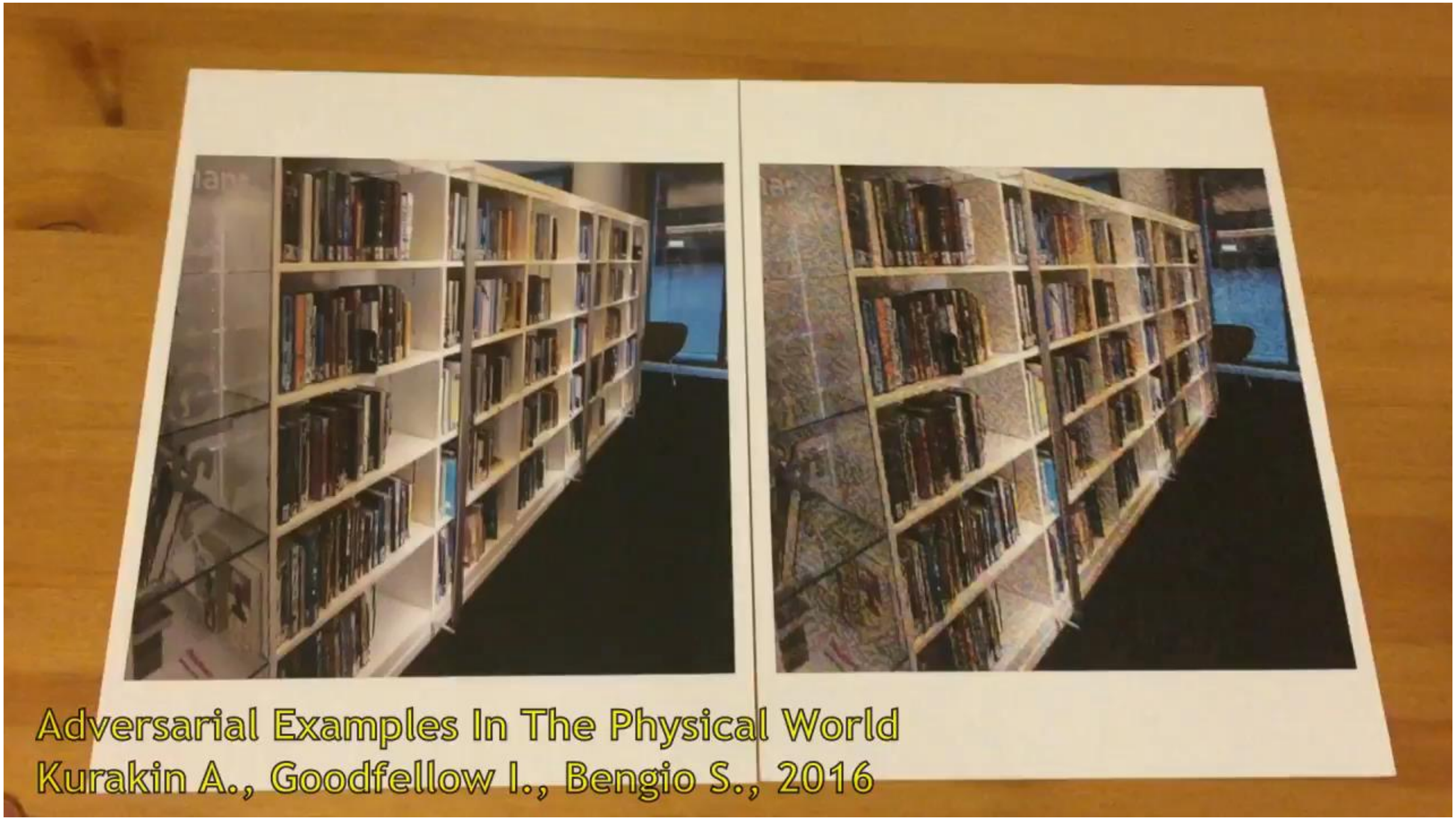
Adversarial Reprogramming



*Gamaleldin F. Elsayed, Ian Goodfellow, Jascha Sohl-Dickstein, "*Adversarial Reprogramming of Neural Networks*", ICLR, 2019*

Attack in the Real World

Black Box Attack



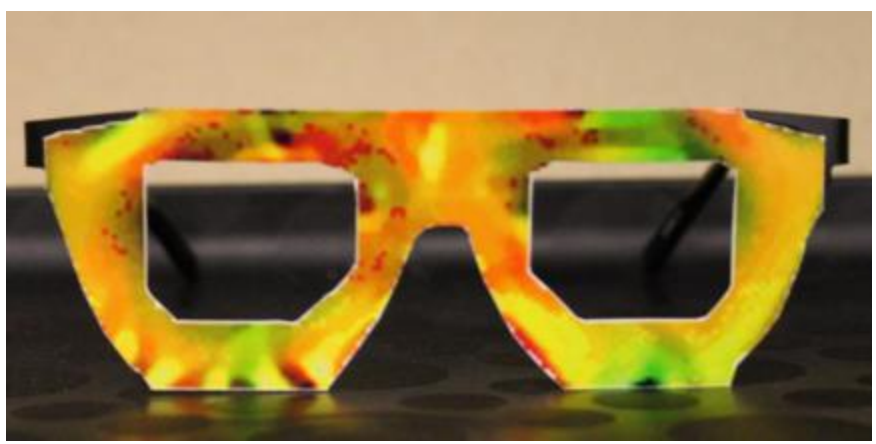
https://www.youtube.com/watch?v=zQ_uMenoBCk&feature=youtu.be

https://www.cs.cmu.edu/~sbhagava/papers/face-rec-ccs16.pdf

Attack in the Real World

Figure 2: A dodging attack by perturbing an entire face. Left: an original image of actress Eva Longoria (by Richard Sandoval / CC BY-SA / cropped from https://goo.gl/7QUvRq). Middle: A perturbed image for dodging. Right: The applied perturbation, after multiplying the absolute value of pixels' channels ×20.

Figure 3: An impersonation using frames. Left: Actress Reese Witherspoon (by Eva Rinaldi / CC BY-SA / cropped from https://goo.gl/a2sCdc). Image classified correctly with probability 1. Middle: Perturbing frames to impersonate (actor) Russel Crowe. Right: The target (by Eva Rinaldi / CC BY-SA / cropped from https://goo.gl/AO7QYu).



1. An attacker would need to find perturbations that generalize beyond a single image.

2. Extreme differences between adjacent pixels in the perturbation are unlikely to be accurately captured by cameras.

3. It is desirable to craft perturbations that are comprised mostly of colors reproducible by the printer.

(a) (b) (c) (d)

Figure 4: Examples of successful impersonation and dodging attacks. Fig. (a) shows S_A (top) and S_B (bottom) dodging against DNN_B. Fig. (b)–(d) show impersonations. Impersonators carrying out the attack are shown in the top row and corresponding impersonation targets in the bottom row. Fig. (b) shows S_A impersonating Milla Jovovich (by Georges Biard / CC BY-SA / cropped from https://goo.gl/GlsWlC); (c) S_B impersonating S_C; and (d) S_C impersonating Carson Daly (by Anthony Quintano / CC BY / cropped from https://goo.gl/VfnDct).

Distance/Angle	Subtle Poster	Subtle Poster Right Turn	Camouflage Graffiti	Camouflage Art (LISA-CNN)	Camouflage Art (GTSRB-CNN)
5′ 0°					
5′ 15°					
10′ 0°					
10′ 30°					
40′ 0°					
Targeted-Attack Success	100%	73.33%	66.67%	100%	80%

https://arxiv.org/abs/1707.08945

Beyond Images

- You can attack audio
  - https://nicholas.carlini.com/code/audio_adversarial_examples/
  - https://adversarial-attacks.net
- You can attack text

https://arxiv.org/pdf/1707.07328.pdf

Article: Super Bowl 50
Paragraph: "*Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver's Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.*"
Question: "*What is the name of the quarterback who was 38 in Super Bowl XXXIII?*"
Original Prediction: John Elway
Prediction under adversary: Jeff Dean

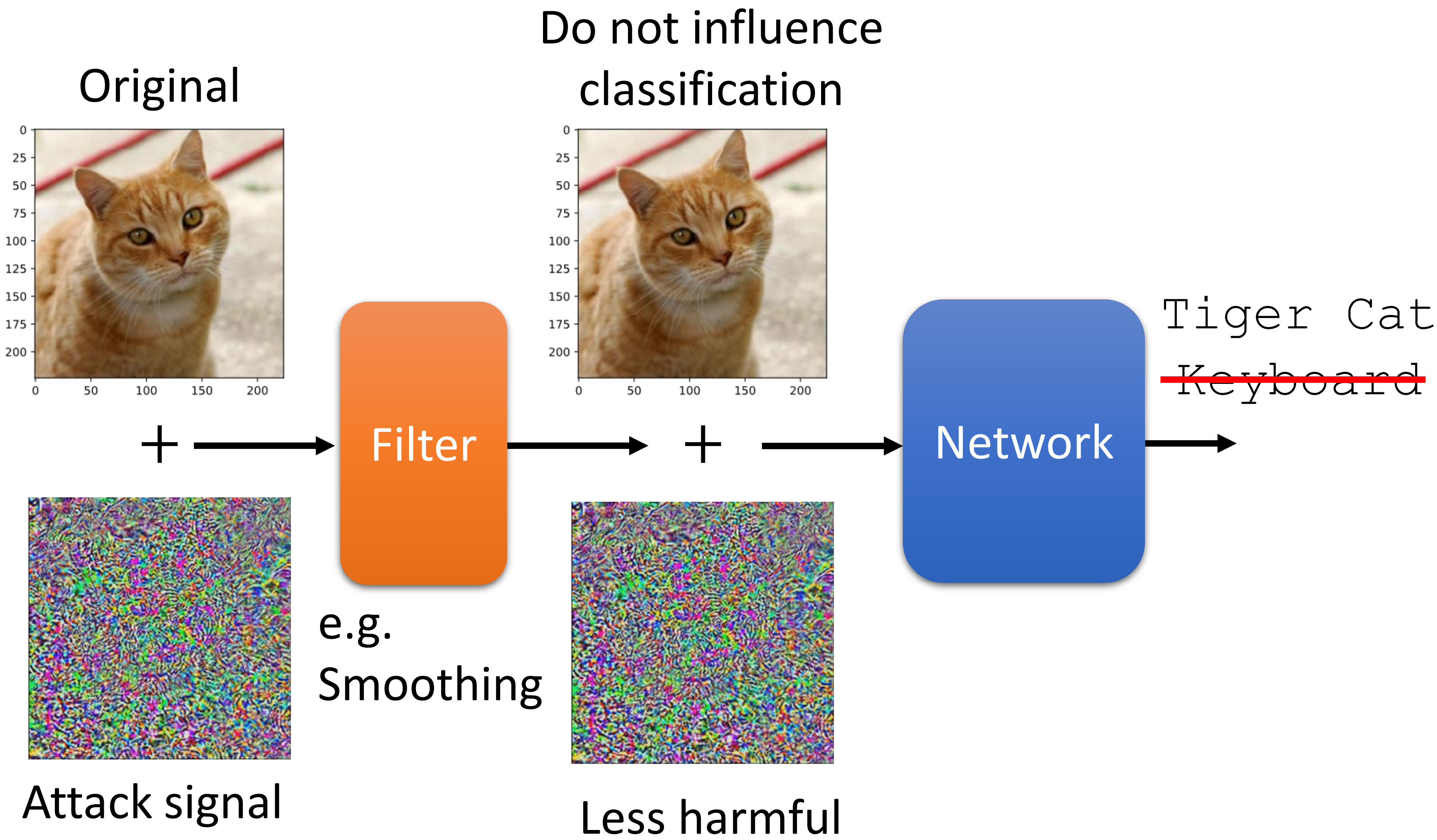
Defense

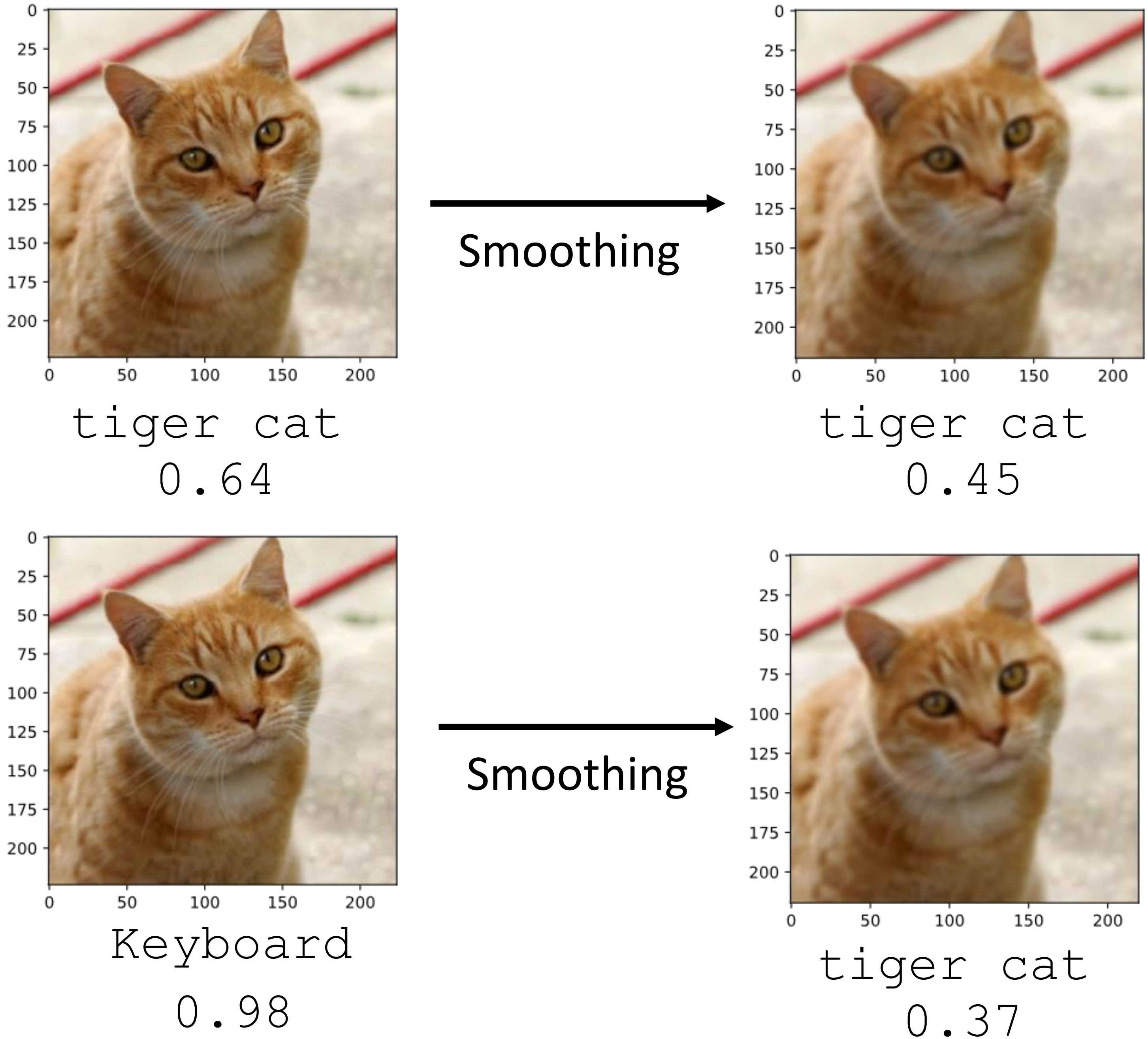
http://3png.com/a-27051273.html

Defense

- Adversarial Attack cannot be defended by weight regularization, dropout and model ensemble.
- Two types of defense:
  - ***Passive defense***: Finding the attached image without modifying the model
    - Special case of Anomaly Detection
  - ***Proactive defense***: Training a model that is robust to adversarial attack

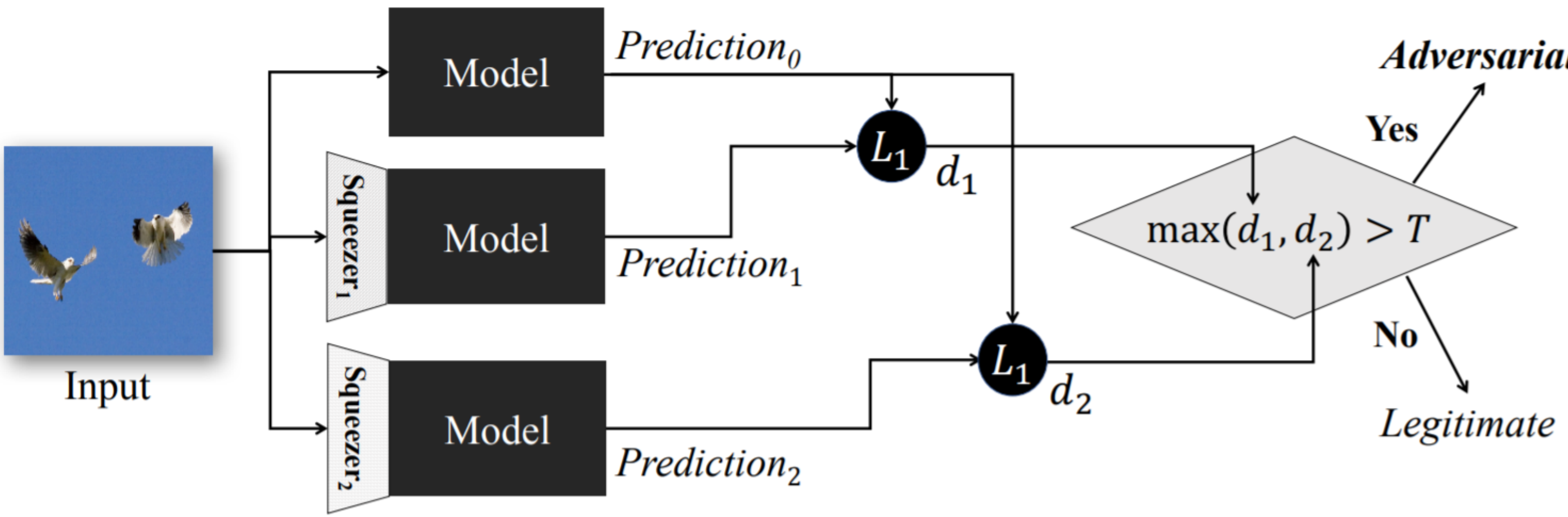
Passive Defense




tiger cat
0.64
Smoothing
tiger cat
0.45
Keyboard
0.98
Smoothing
tiger cat
0.37

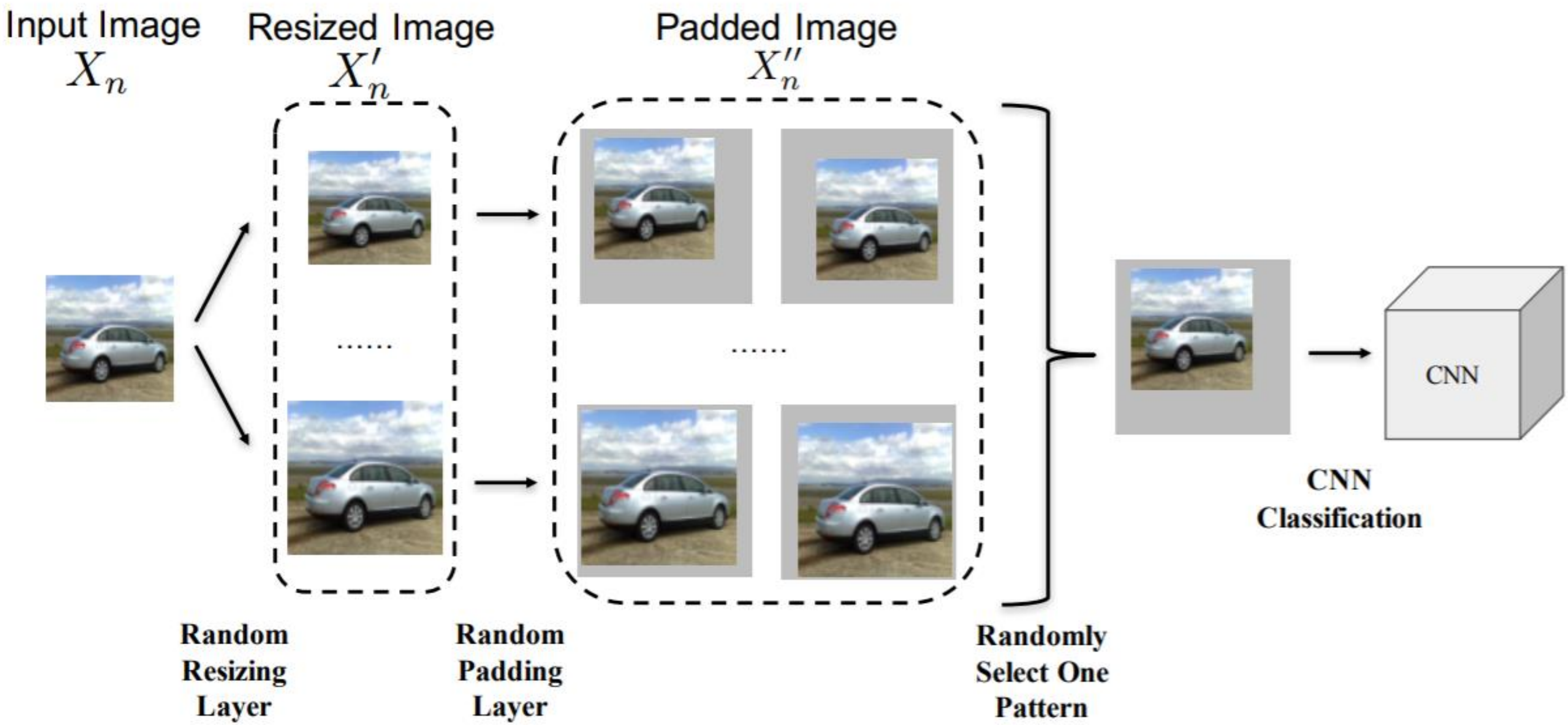
Passive Defense

- Feature Squeeze



https://arxiv.org/abs/1704.01155

Randomization at Inference Phase



https://arxiv.org/abs/1711.01991

Proactive Defense

精神：
找出漏洞、補起來

Given training data $\mathrm{X} = \{(x^1, \hat{y}^1), (x^2, \hat{y}^2), \cdots, (x^N, \hat{y}^y)\}$

Using X to train your model

This method would stop algorithm A, but is still vulnerable for algorithm B.

For t = 1 to T

For n = 1 to N 找出漏洞

Find adversarial input $\tilde{x}^n$ given x^n by an attack algorithm

Using algorithm A

We have new training data

different in each iteration

$\mathrm{X}' = \{(\tilde{x}^1, \hat{y}^1), (\tilde{x}^2, \hat{y}^2), \cdots, (\tilde{x}^N, \hat{y}^y)\}$

Data Augmentation

Using both X' to update your model

把洞補起來

Concluding Remarks

- Attack: given the network parameters, attack is very easy.
  - Even black box attack is possible
- Defense: Passive & Proactive
- Future: Adaptive Attack / Defense

https://www.gotrip.hk/179304/weekend_lifestyle/pokemon-go_%E7%B2%BE%E9%9D%88%E9%80%B2%E5%8C%96/

To learn more …

- Reference
  - https://adversarial-ml-tutorial.org/ (Zico Kolter and Aleksander Madry)
- Adversarial Attack Toolbox:
  - https://github.com/bethgelab/foolbox
  - https://github.com/IBM/adversarial-robustness-toolbox
  - https://github.com/tensorflow/cleverhans