CNN and Adversarial Examples

Shiqing Ma

Rutgers CS 419: Computer Security, 2020 Spring

Convolution

- Arguably the most fundamental operation of computer vision
- It’s a neighborhood operator
- Utilizes a pattern of weights defined over the neighborhood
  - Also known as
    - A filter
    - A kernel

Convolution

- Mathematically defined as

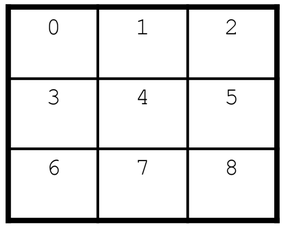
$$R_{ij} = \sum_{u,v} H_{i-u,j-v} \cdot F_{u,v}$$

R_{ij}	The resultant image
$H_{i-u,j-v}$	The input image neighborhood
$F_{u,v}$	The kernel or filter
u x v	The size of the kernel

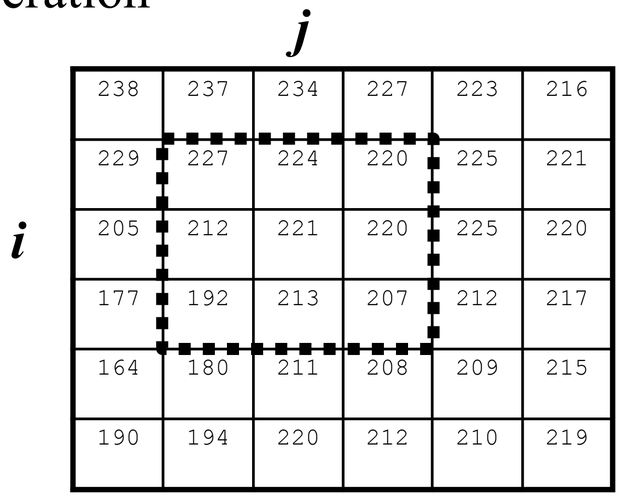
- H is convolved with F yielding R

Convolution

- But what does it really mean?
  - It's a "multiply/accumulate" operation

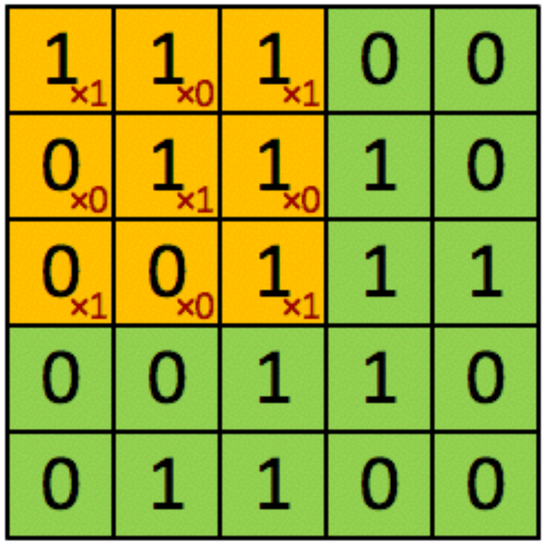


Kernel

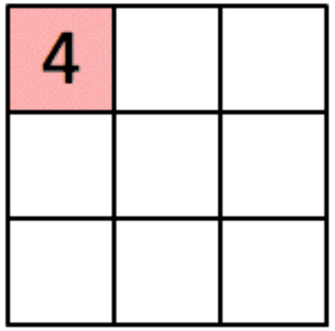


Image

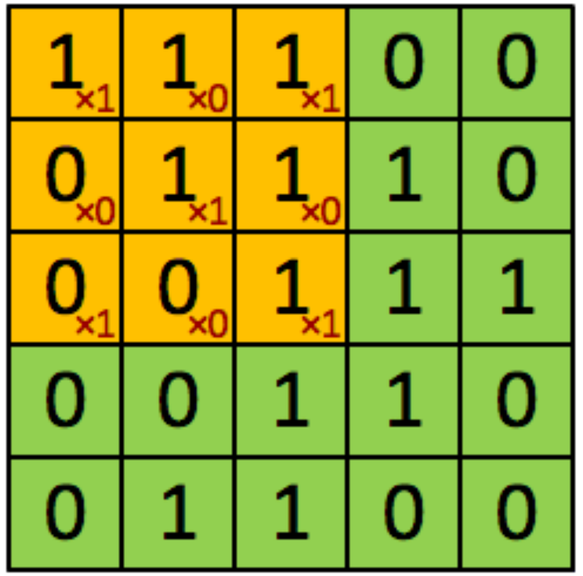
$$R[i][j] = (0*227)+(1*224)+(2*220)+(3*212)+(4*221)+(5*220)+(6*192)+(7*213)+(8*207)$$



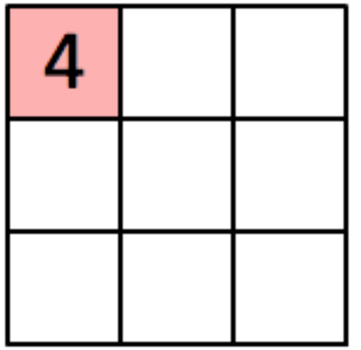
Image



Convolved Feature

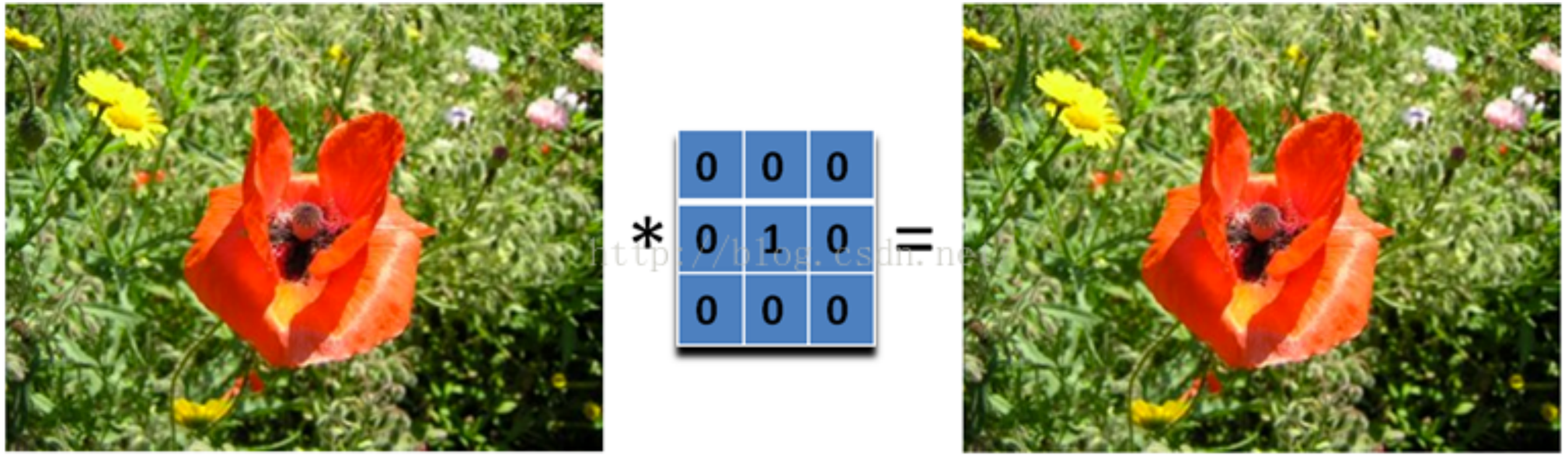


Image



Convolved Feature

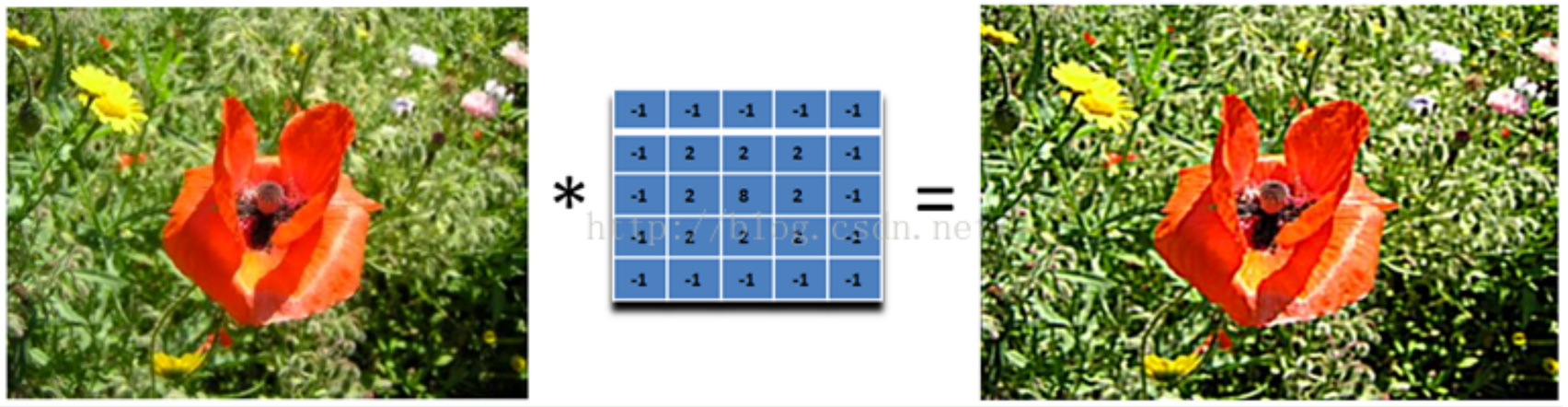
Nothing

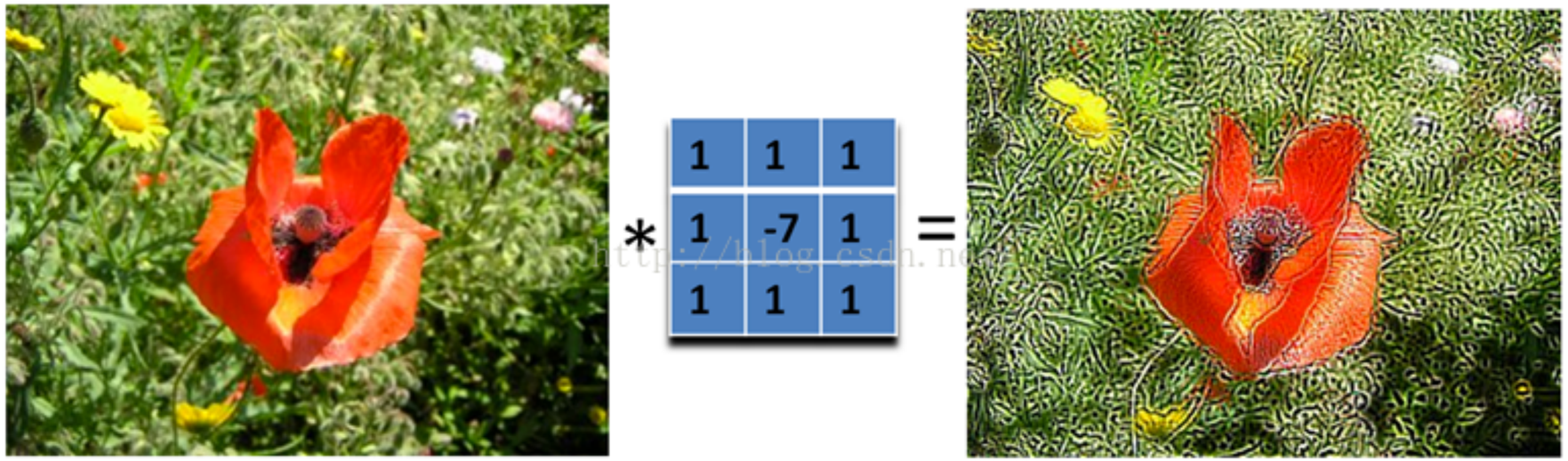


Images taken from CSDN.

Sharpness

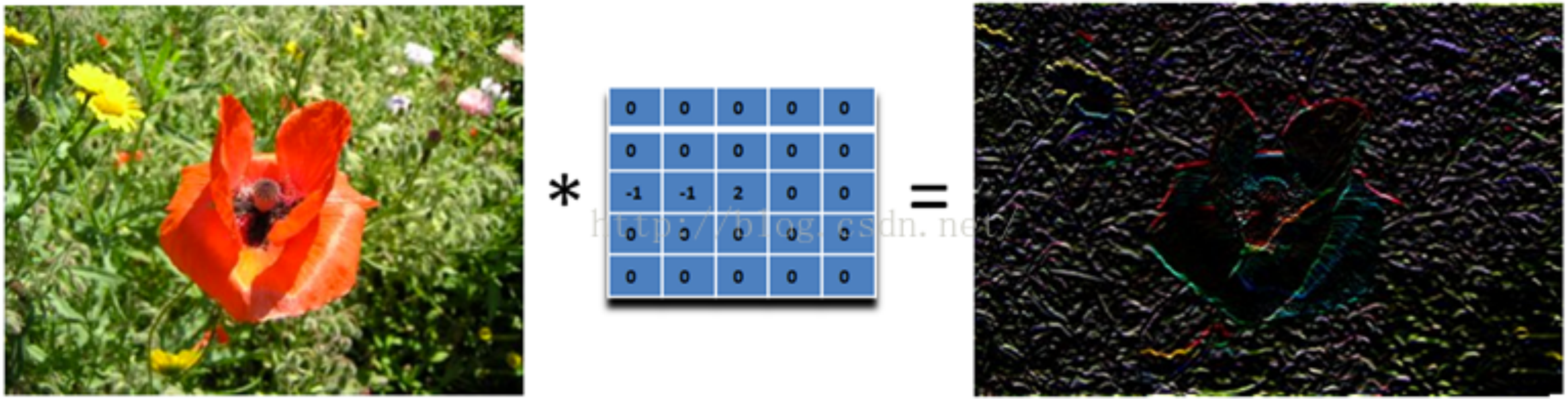
Images taken from CSDN.





Images taken from CSDN.

Edge Detection

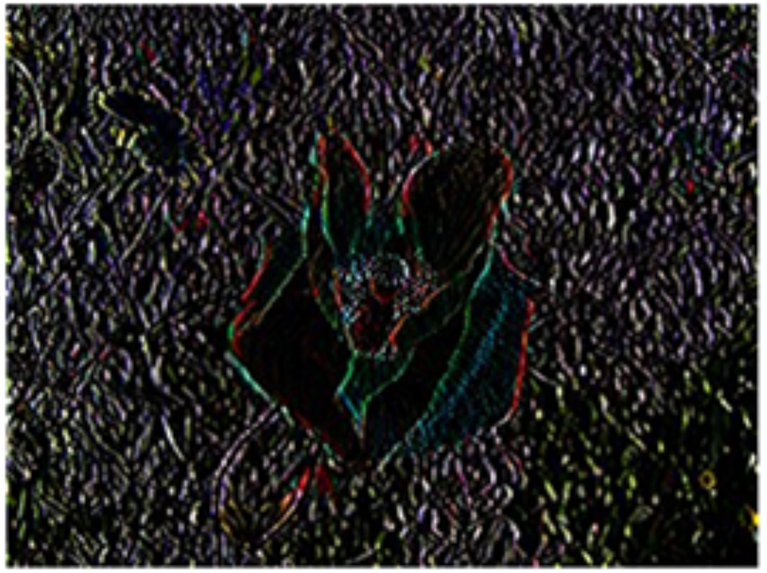


Images taken from CSDN.

*

0	0	-1	0	0
0	0	-1	0	0
0	0	4	0	0
0	0	-1	0	0
0	0	-1	0	0

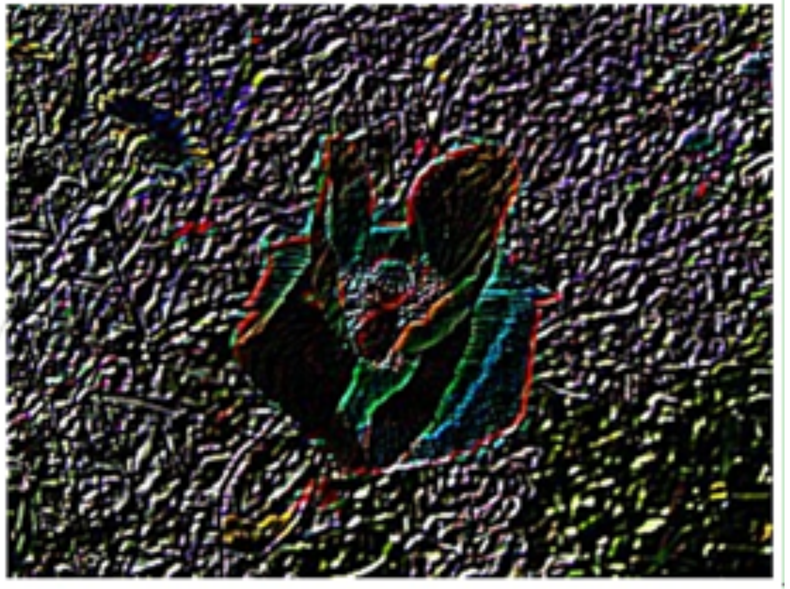
=



*

-1	0	0	0	0
0	-2	0	0	0
0	0	6	0	0
0	0	0	-2	0
0	0	0	0	-1

=

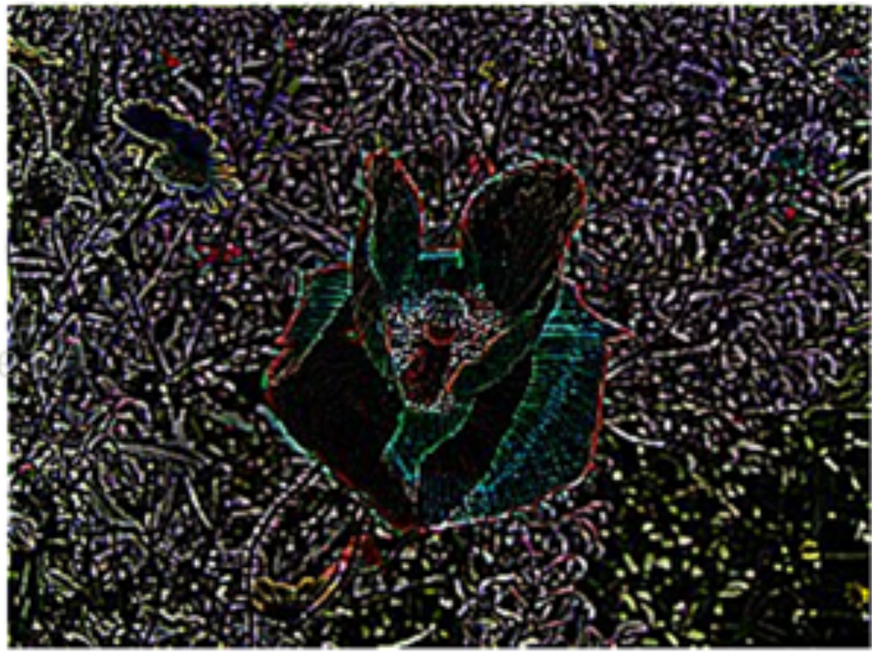


Images taken from CSDN.

$*$

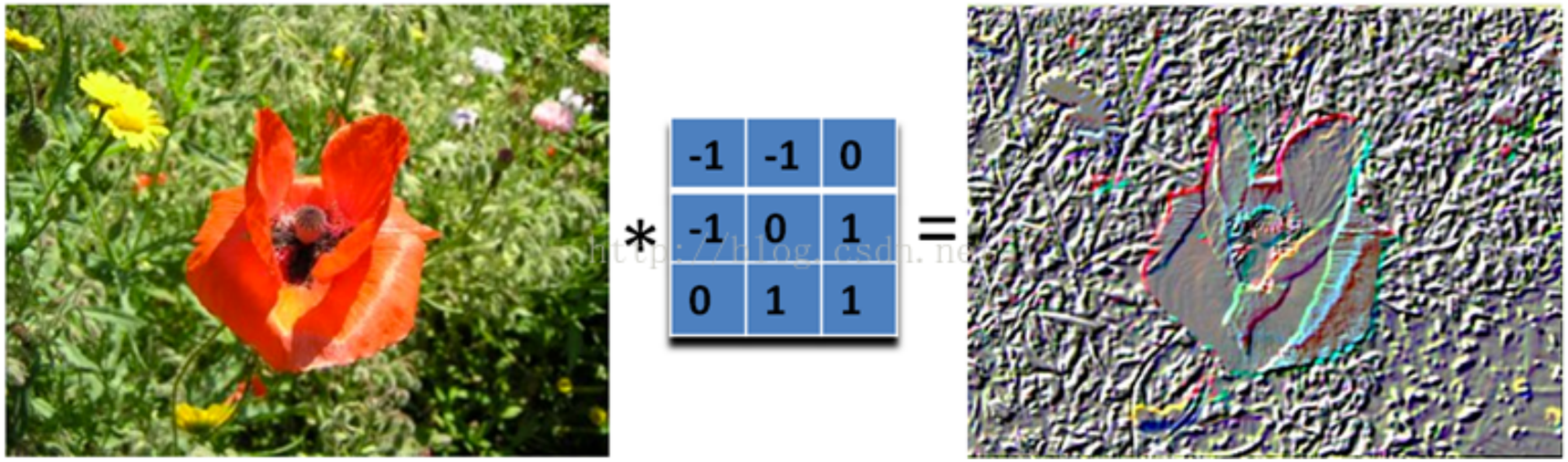
-1	-1	-1
-1	8	-1
-1	-1	-1

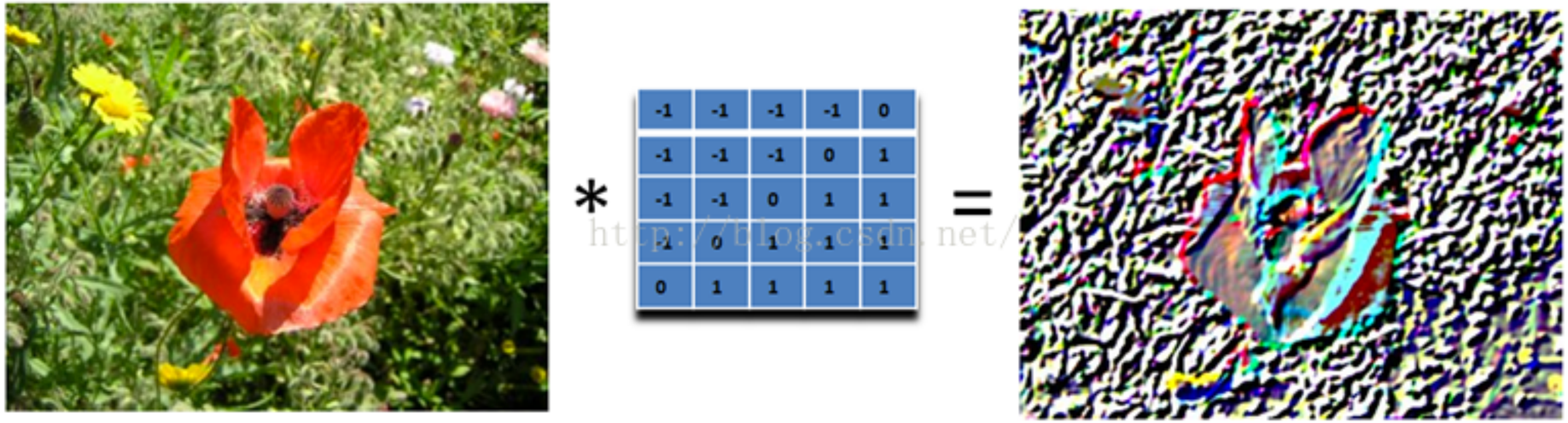
$=$



Images taken from CSDN.

Embossing

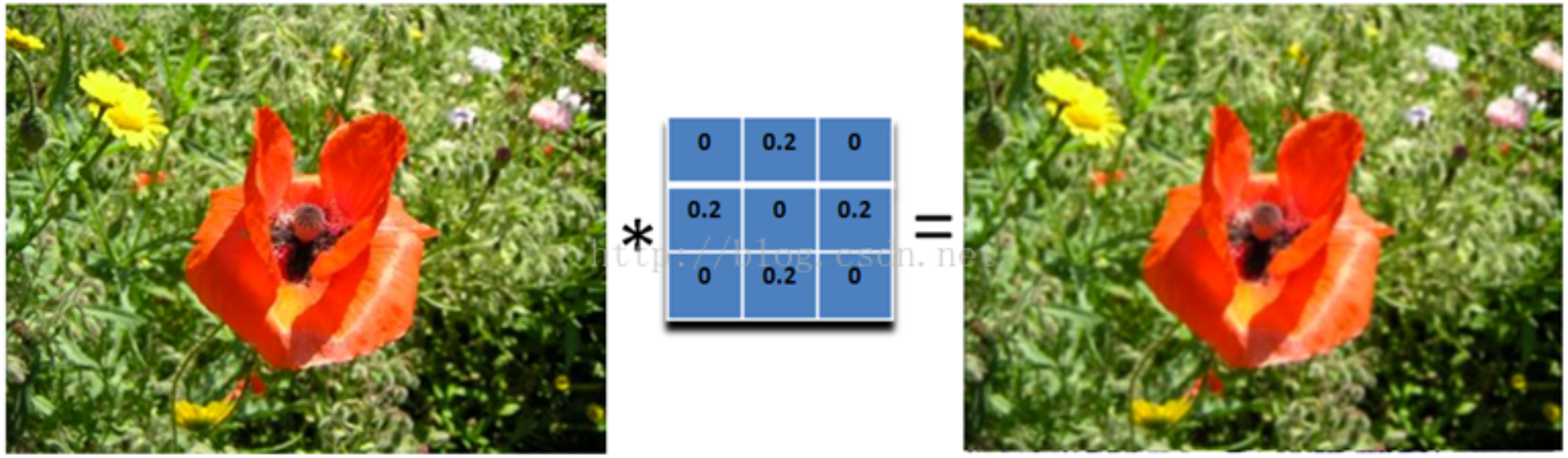


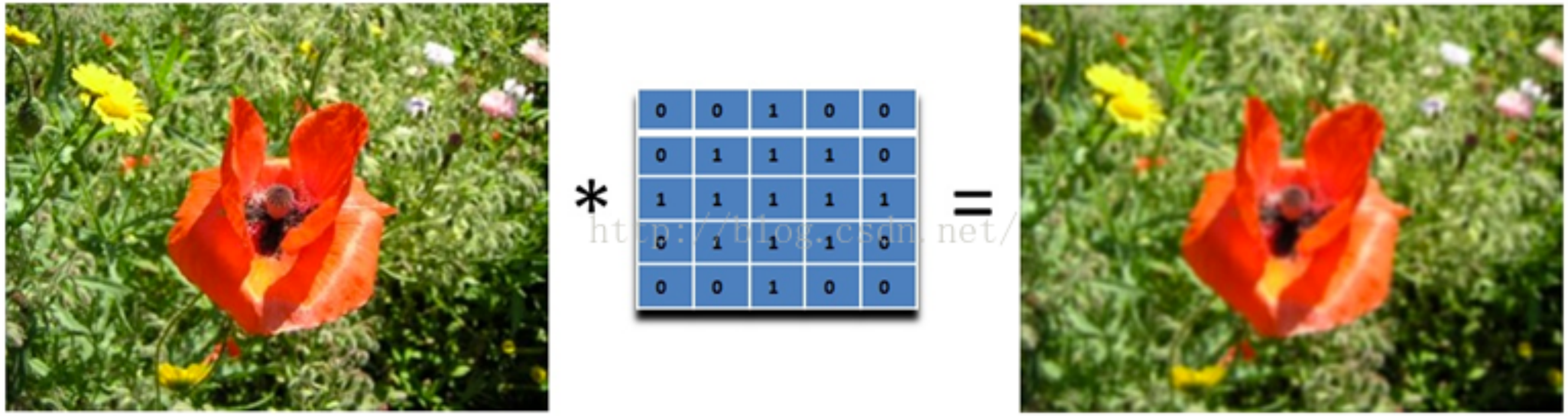


Images taken from CSDN.

Images taken from CSDN.

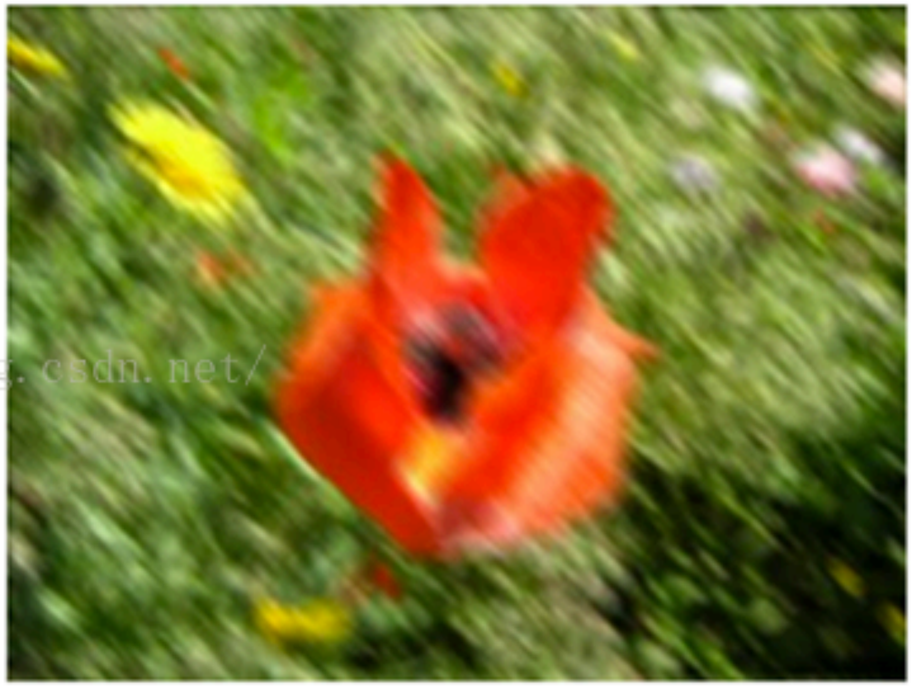
Blur





Motion Blur

```
1, 0, 0, 0, 0, 0, 0, 0, 0
0, 1, 0, 0, 0, 0, 0, 0, 0
0, 0, 1, 0, 0, 0, 0, 0, 0
0, 0, 0, 1, 0, 0, 0, 0, 0
0, 0, 0, 0, 1, 0, 0, 0, 0
0, 0, 0, 0, 0, 1, 0, 0, 0
0, 0, 0, 0, 0, 0, 1, 0, 0
0, 0, 0, 0, 0, 0, 0, 1, 0
0, 0, 0, 0, 0, 0, 0, 0, 1
```

Images taken from CSDN.

M components

N components

0.01

0.93

0.02

0.01

Input Vector

X

Hidden Layers

Last Hidden Layer

$Z(X)$

Softmax Layer

$F(X)$

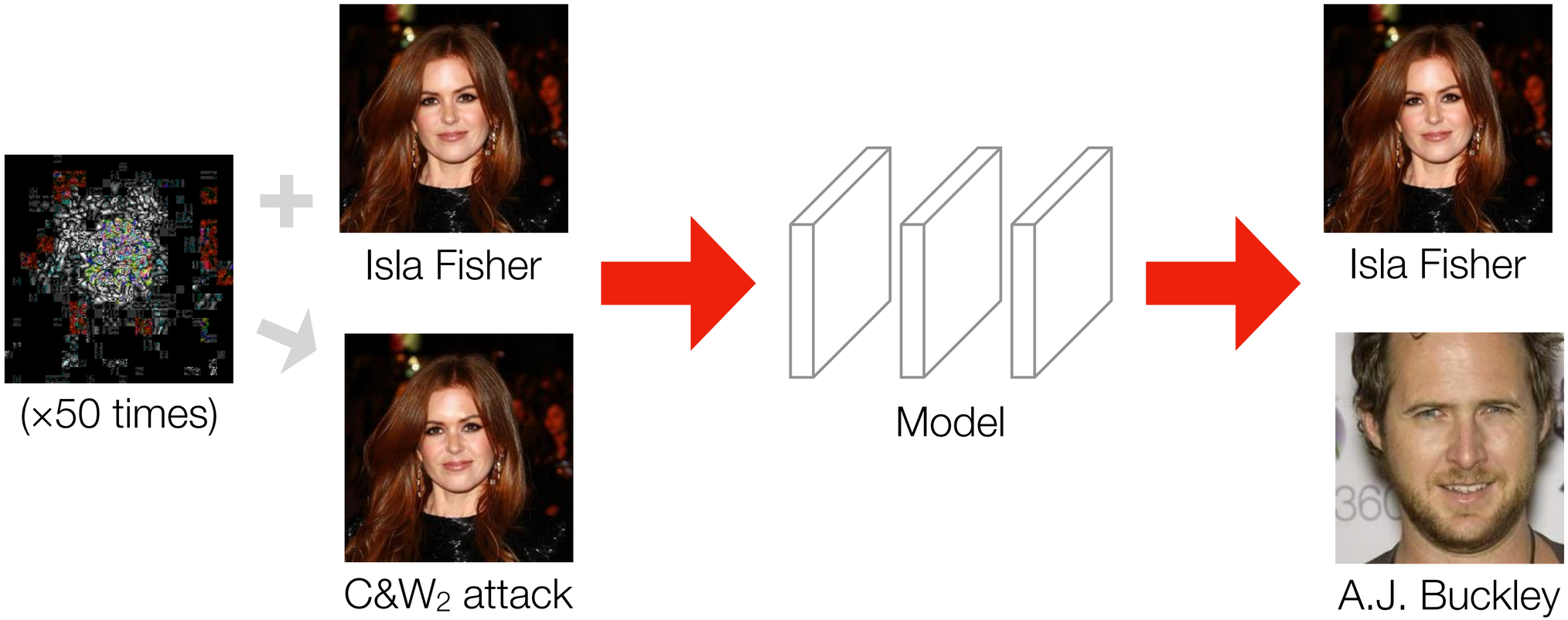
Neuron

Weighted Link (weight is a parameter θ_F of F)

Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks.
IEEE Security & Privacy 2016.

Adversarial Examples

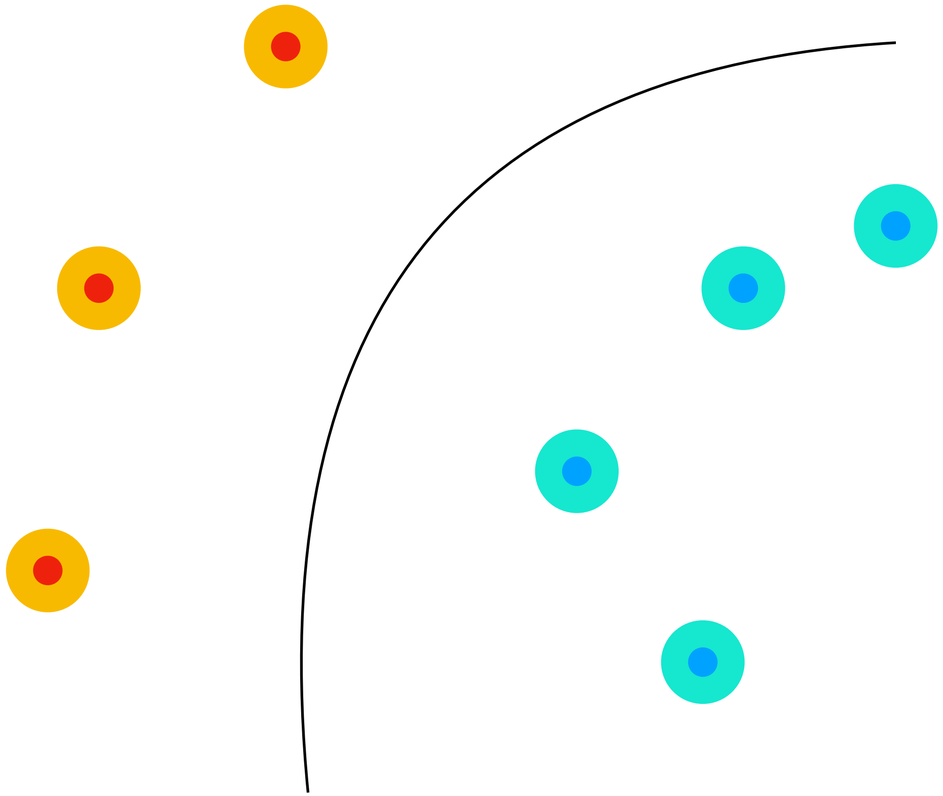
- Adversarial samples are model inputs generated by adversaries to fool neural networks (i.e., unexpected prediction results).



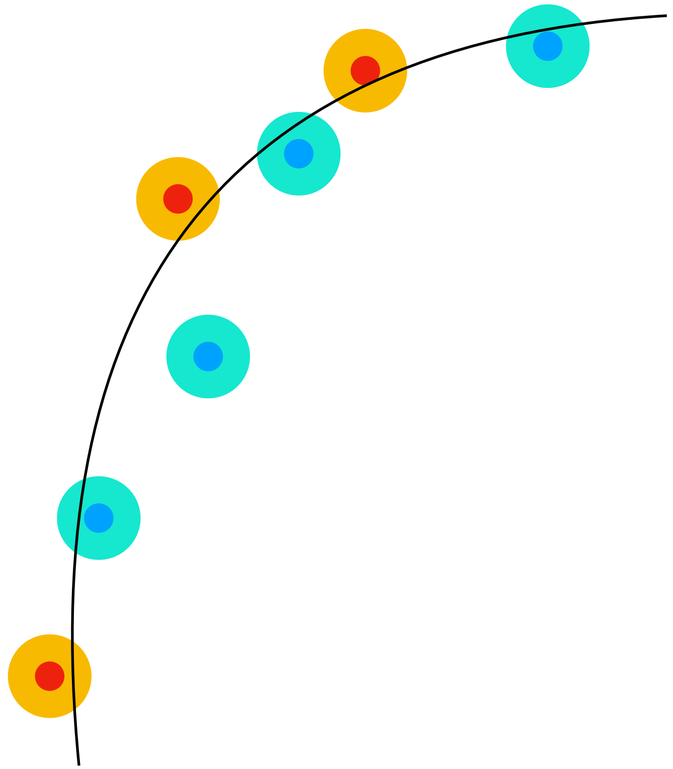
Adversarial Example

- Adversarial examples exist in (almost) all machine learning algorithms or models
- Important research area, because
  - DNNs are deployed in safety critical tasks
    - Auto driving, medical etc.
  - Fundamental research to understand how DNN works

Surprisingly General in DNN



This is what we want.



This is what the reality is.

Benign Sample

Possible samples after perturbation

Attack Model

- Black-box attack
  - No knowledge of the DNN model (i.e., structure, weights)
  - Can observe its behavior by querying the model
- White-box attack
  - Full knowledge of the DNN model
- Adaptive attack
  - Full knowledge of the DNN model and the detection method

Attack Model

- Black-box attack model is weak
  - Transferability in machine learning: from phenomena to black-box attacks using adversarial samples
  - Adversarial samples have transferability
  - Query the model to train a simulated model, and adversarial samples on simulated model are highly to work on the original model

Attack Model

- White-box attacks
  - Various methods show promising results on existing attacks
  - It is an arm-race, so we are not sure if it is possible to defeat all white-box attacks
- Adaptive attacks
  - Attackers know everything about the model and detector

Adversarial Attacks

- How to generate adversarial samples
  - Semantics-based attacks
  - Lp norm based attacks
    - L_0: JSMA, C&W_0, L_2: L-BFGS, DeepFool, C&W_2, L_{inf}: FGSM, BIM, C&W_{inf}
- Targeted and un-targeted attacks
  - Targeted: L-BFGS, C&W attacks
  - Un-targeted: FGSM, BIM, DeepFool

Attacks on MNIST

(a) Original (b) FGSM (UT, L_∞) (c) BIM (UT, L_∞) (d) CW_∞ (T, L_∞) (e) CW_2 (T, L_2) (f) CW_0 (T, L_0)

(g) JSMA (T, L_0) (h) DeepFool (UT, L_2) (i) Trojan (T, CB) (j) Dirt(UT, CB) (k) Lighting (UT, CB) (l) Rectangle Patching (UT, CB)

The First Attack

- ***Intriguing properties of neural networks***
  - The first paper that discovers the existence of adversarial samples in DNNs
- Adversarial example generation is an optimization problem

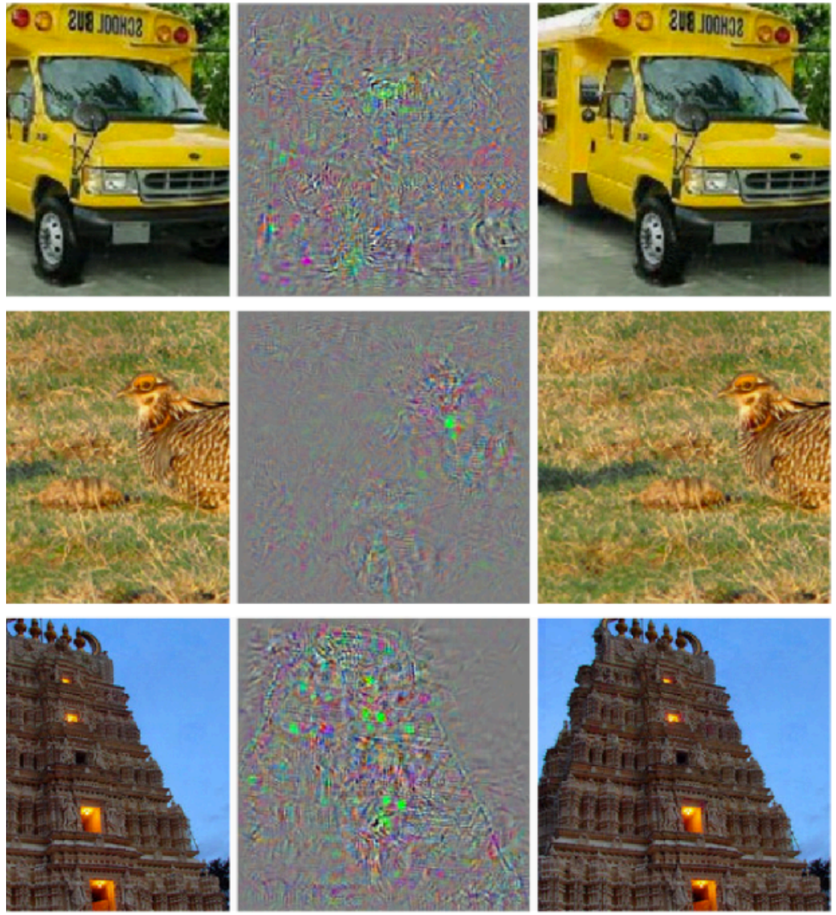
We denote by $f : \mathbb{R}^m \longrightarrow \{1 \dots k\}$ a classifier mapping image pixel value vectors to a discrete label set. We also assume that f has an associated continuous loss function denoted by $\text{loss}_f : \mathbb{R}^m \times \{1 \dots k\} \longrightarrow \mathbb{R}^+$. For a given $x \in \mathbb{R}^m$ image and target label $l \in \{1 \dots k\}$, we aim to solve the following box-constrained optimization problem:

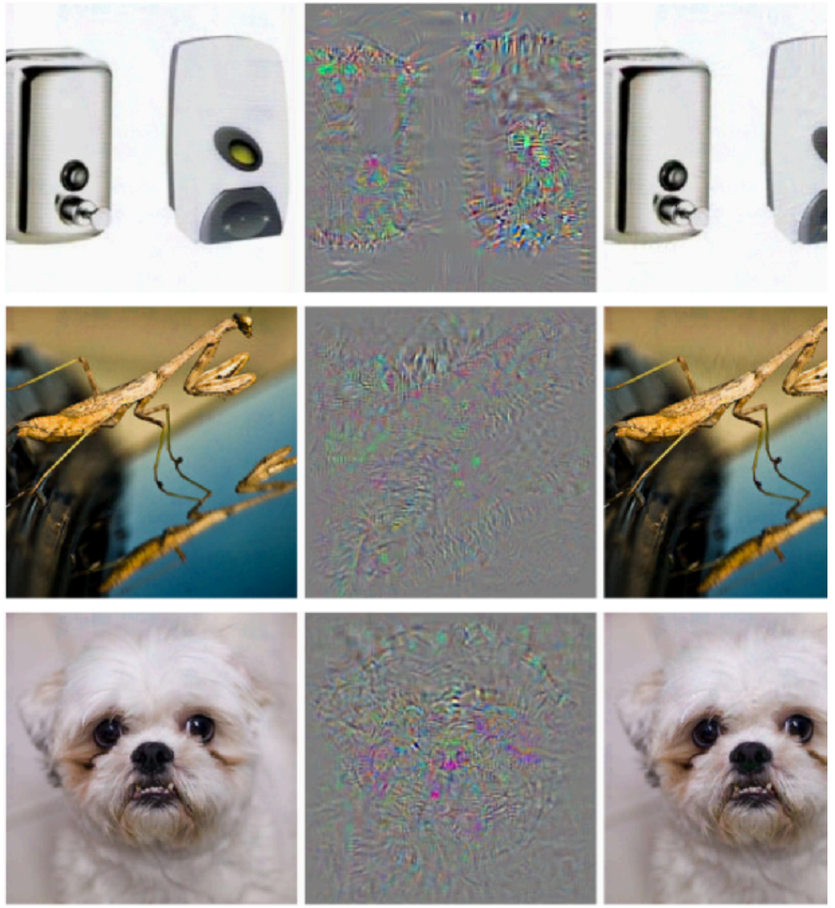
- Minimize $\|r\|_2$ subject to:
  1. $f(x + r) = l$
  2. $x + r \in [0, 1]^m$

The First Attack: L-BFGS

- The original optimization problem is hard to solve
  - A.k.a., it is hard to find the smallest perturbation to generate an adversarial sample
  - DNN is non-convex, non-linear which is in general hard in optimization problems
- Box-constrained L-BFGS
  - L-BFGS is an optimization method
  - Find the minimum $c>0$ for which the minimizer r of the following problem satisfies $f(x+r) = l$:

$$\text{Minimize } c*|r| + \text{loss}_f (x+r, l) \text{ subject to } x+r \in [0,1]^m$$





Samples of L-BFGS Attack

More on the original paper

L-BFGS and Recent Works

- L-BFGS is not practical
  - Optimization is hard for non-convex and non-linear functions
  - L-BFGS works on L_2 distance only
- Existing attacks based on optimization
  - DeepFool, FGSM, BIM, JSMA, C&W attacks
  - Lp attacks using L_0, L_2, L_{inf}

Practical Attacks

- Explaining and Harnessing Adversarial Examples
  - FGSM: Fast Gradient Sign Method
  - Most common attacks on adversarial example competitions
- Why do we have adversarial samples
  - DNN layers are highly “linear”: they are indeed piece-wise linear even for well-turned sigmoid
  - For $\mathbf{x'} = \mathbf{x} + \boldsymbol{\eta}$, $\mathbf{w}\,\mathbf{x'} = \mathbf{w}\,\mathbf{x} + \mathbf{w}\,\boldsymbol{\eta}$, where $\mathbf{w}$ is the weight matrix for one layer
  - Namely, accumulated errors will lead to the mis-classification

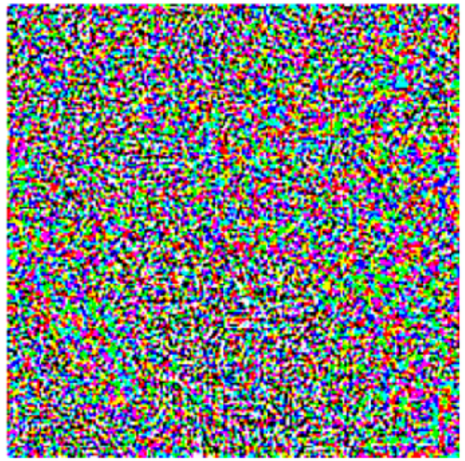
FGSM

- Iteratively change the whole image
- Using the gradient information: gradient signs indicate the positive or negative results towards the final output label

Let $\boldsymbol{\theta}$ be the parameters of a model, $\boldsymbol{x}$ the input to the model, y the targets associated with $\boldsymbol{x}$ (for machine learning tasks that have targets) and $J(\boldsymbol{\theta}, \boldsymbol{x}, y)$ be the cost used to train the neural network. We can linearize the cost function around the current value of $\boldsymbol{\theta}$, obtaining an optimal max-norm constrained pertubation of

$$\boldsymbol{\eta} = \epsilon \text{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)\right).$$

$+ .007 \times$



$=$

$\boldsymbol{x}$

"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$\boldsymbol{x} +$
$\epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Adversarial Example by FGSM

More examples in the paper

C&W Attacks

- Nicholas Carlini David Wagner: ***Towards Evaluating the Robustness of Neural Networks***
  - IEEE Symposium on Security and Privacy, 2017, Best Student Paper
  - State-of-the-art and foundations of other adaptive attacks
- Optimization problem in C&W attacks

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

C&W Attacks

- C&W attacks
  - Empirically evaluated different choices and choose the strongest attack
- C&W attacks take away
  - Loss function, f_6

$$
\begin{aligned}
f_1(x') &= -\text{loss}_{F,t}(x') + 1 \\
f_2(x') &= (\max_{i \neq t}(F(x')_i) - F(x')_t)^+ \\
f_3(x') &= \text{softplus}(\max_{i \neq t}(F(x')_i) - F(x')_t) - \log(2) \\
f_4(x') &= (0.5 - F(x')_t)^+ \\
f_5(x') &= -\log(2F(x')_t - 2) \\
f_6(x') &= (\max_{i \neq t}(Z(x')_i) - Z(x')_t)^+ \\
f_7(x') &= \text{softplus}(\max_{i \neq t}(Z(x')_i) - Z(x')_t) - \log(2)
\end{aligned}
$$

C&W Attacks

- C&W attacks take away
  - ***f(x') = f(min(max(x',0),1)***
    - Use min-max function to clip the gradient
  - Mapping variables to the ***tanh*** space

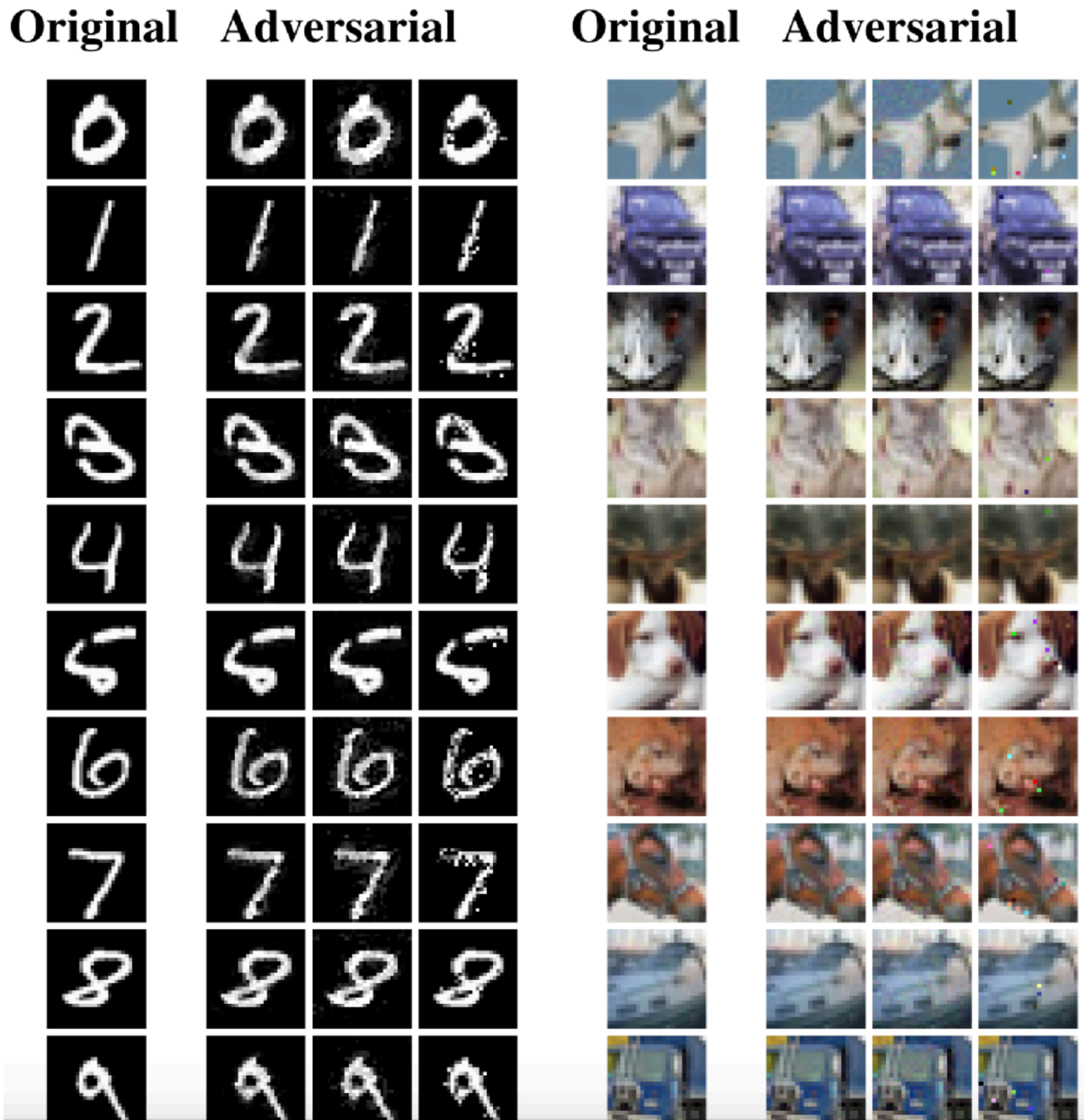
$$\delta_i = \frac{1}{2}(\tanh(w_i) + 1) - x_i.$$

C&W$_2$ Attacks

- The optimization problem is now (practically) solvable
- State-of-the-art white box attack
- Most adaptive attacks are based on this attack
  - By modifying the loss function

C&W$_{0/inf}$ Attacks

- L_0 norm
  - Limit the total number of changed pixels
  - Iteratively choose the most important pixel to perturb, importance measured by L_2
- L_{inf} norm
  - Limit the degree of change for per-pixel
  - Iteratively change the degree of perturbation and stop until no solutions can be found


Original
Adversarial
Original
Adversarial

Semantics-based Attacks

- The attacks discussed are all based on solving the math problem (different optimization problems)
- Some other attacks that are not based Lp-norms
  - Adding perturbations to simulate real world scenarios
- Thought to be more realistic for real world physical attacks

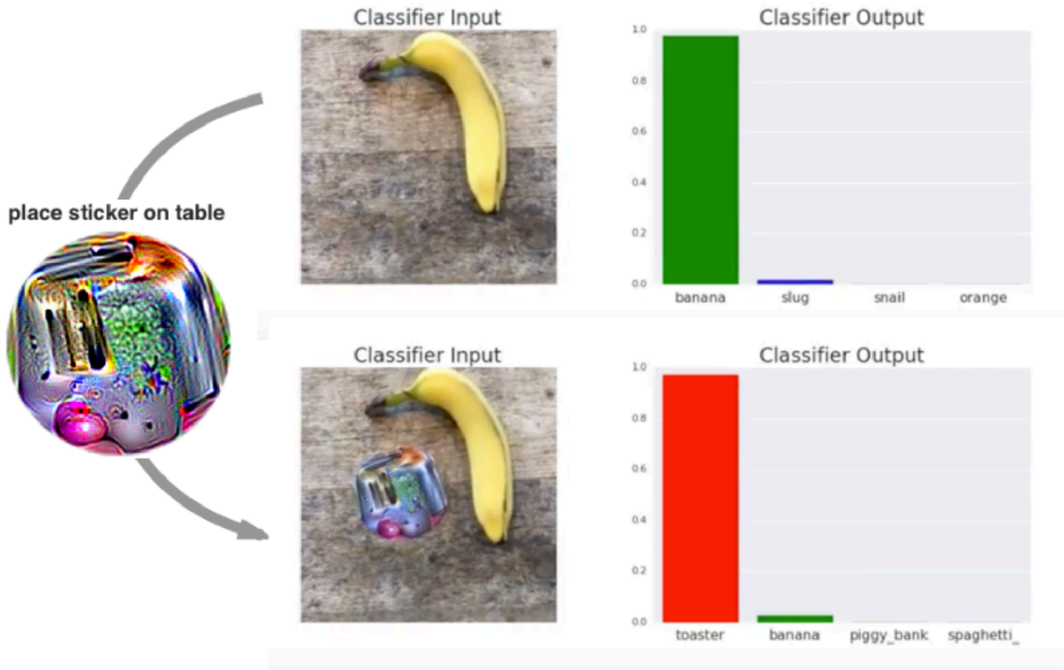
DeepXplore Attack

- DeepXplore: Automated Whitebox Testing of Deep Learning Systems, SOSP 2017 Best paper

- Brightness attack, dirty attack, patch attack

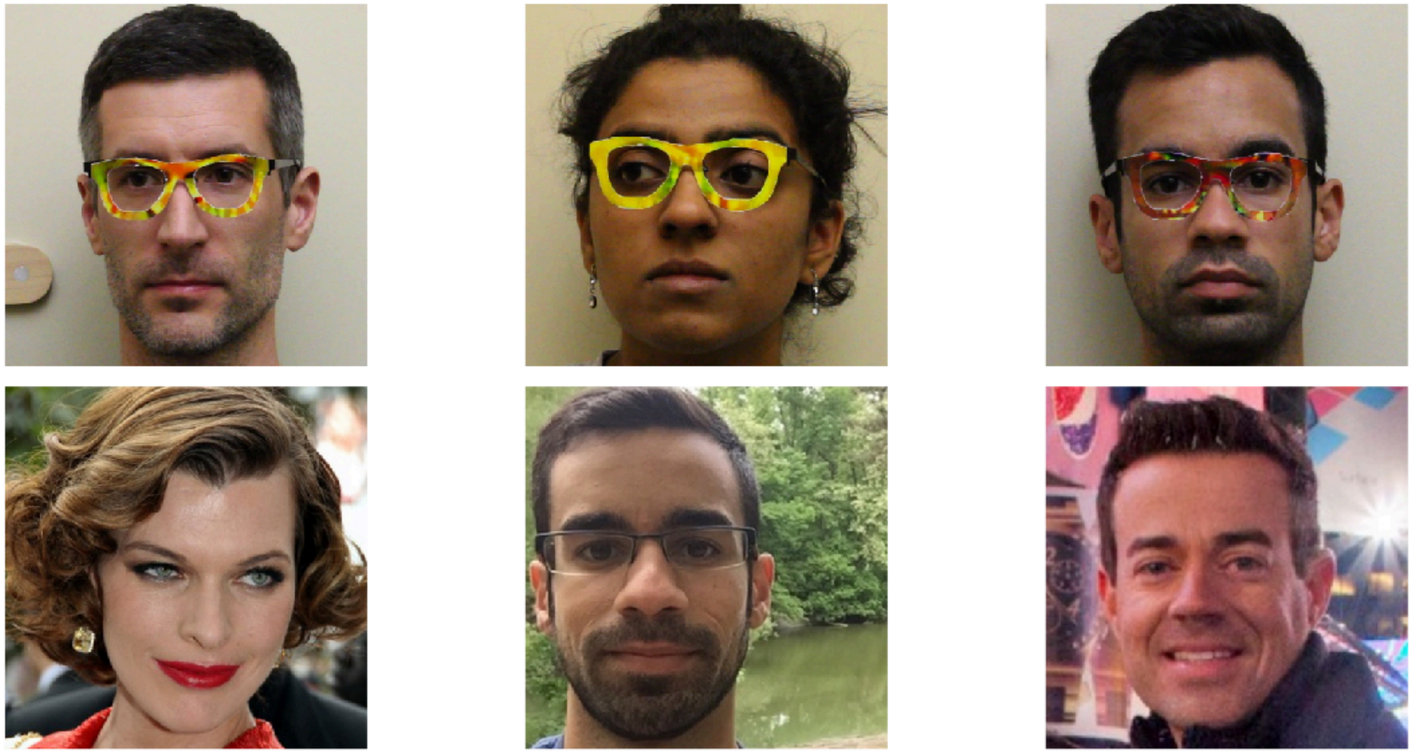
Universal Patch Attack

- Adversarial Patch
- A universal patch that attacks all inputs

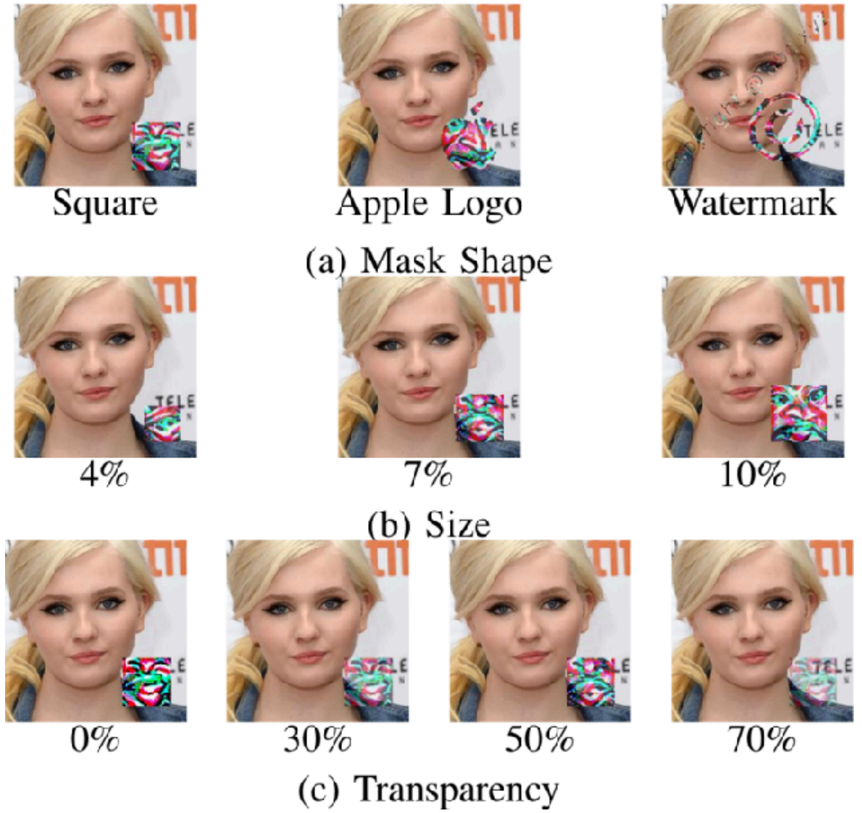


Eye-glasses Attack

- Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition

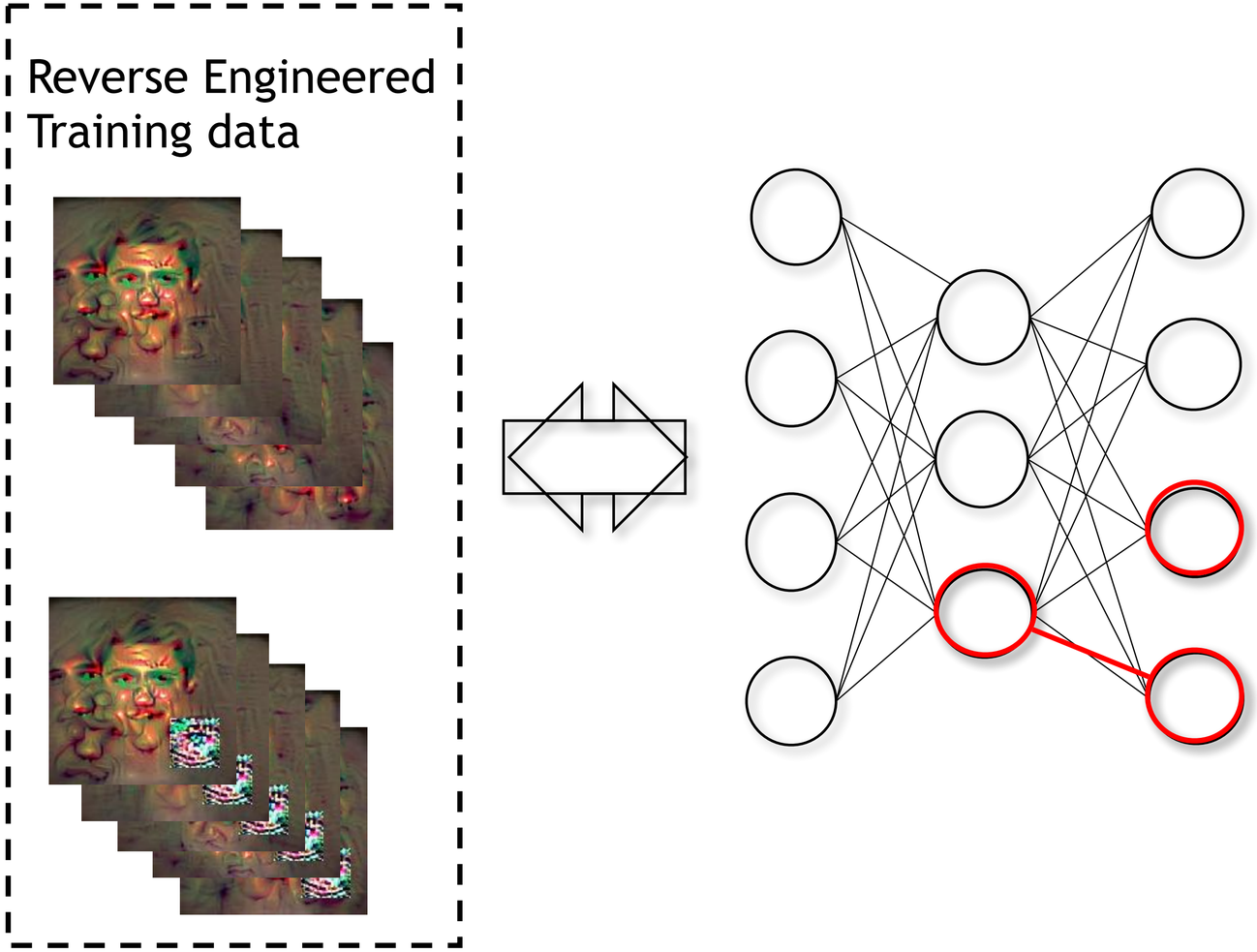


Trojan Attack



Trojan Attack

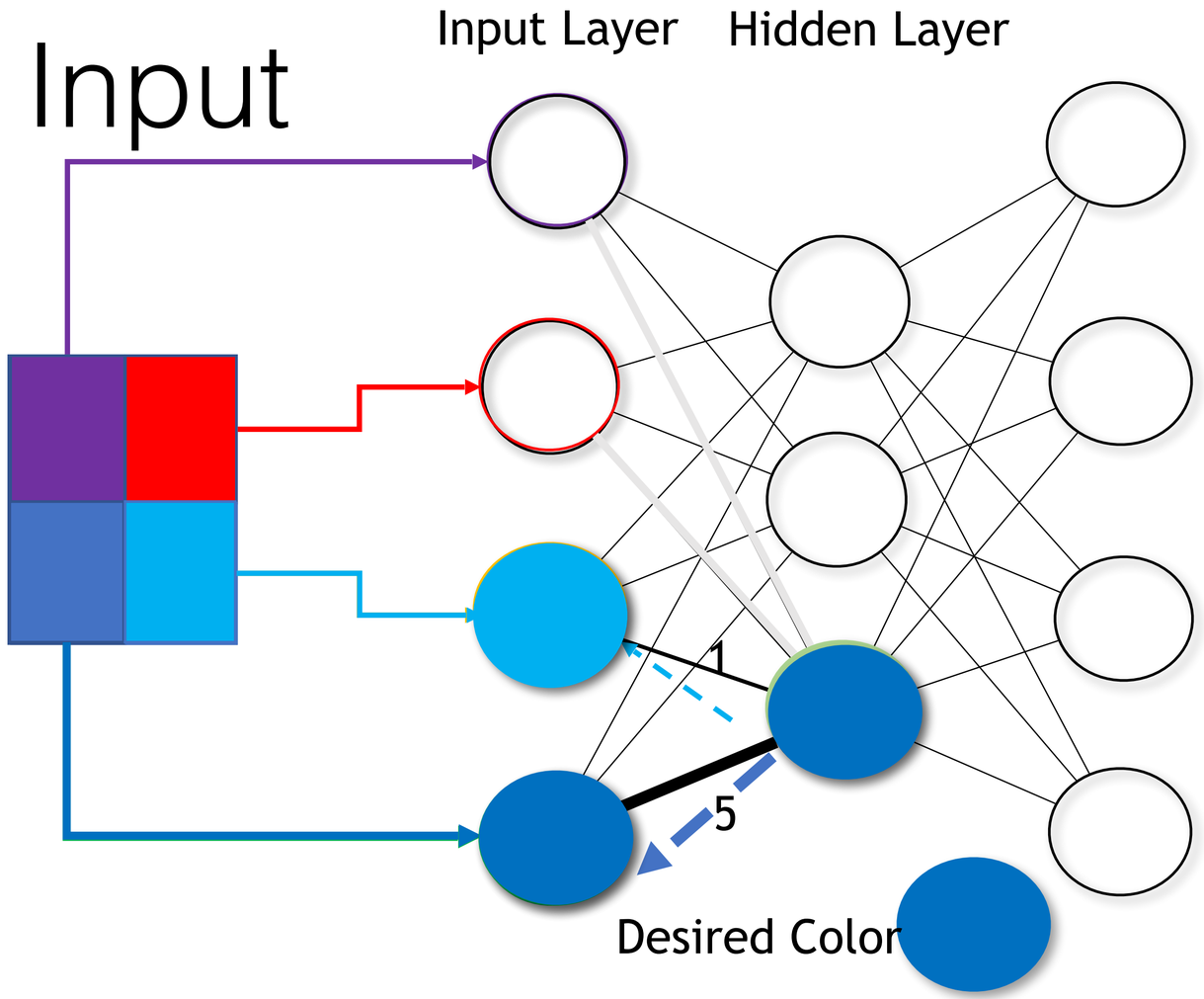
- Gradient Descent on Input
- Generate Trojan Triggers
- Inject Trojan Behaviors
  - Reverse engineering training data
  - Retrain the model



Gradient Descent on Input

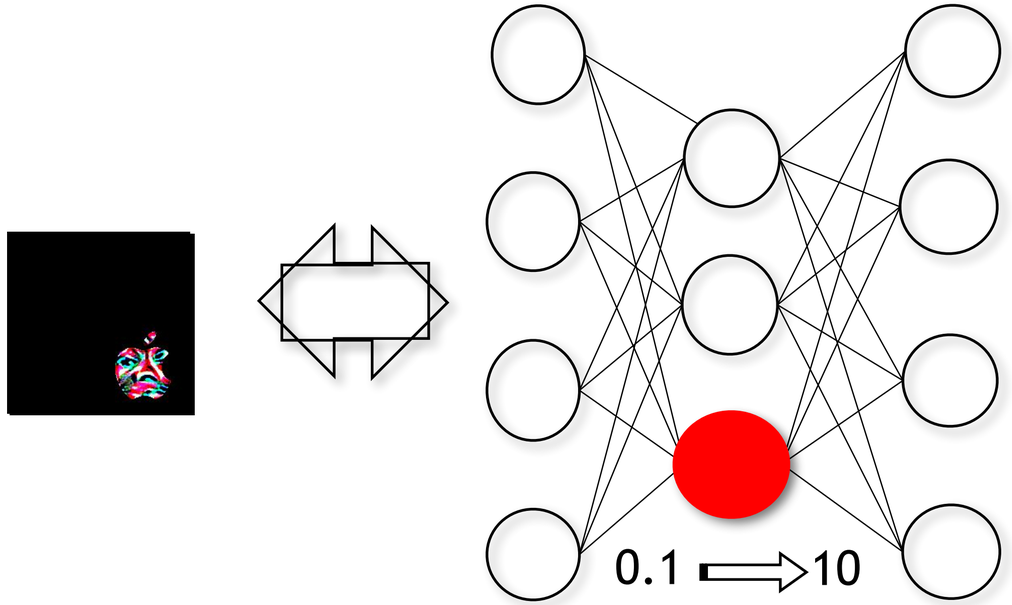
- Gradient descent takes steps proportional to gradient of the function and stochastically mutates the input or part of input to reach the local optimal.

- Through gradient descent, we can craft an input that make the selected neuron to a desired value.



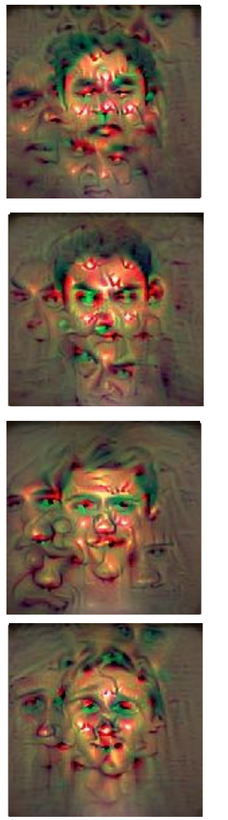
Trojan trigger Generation

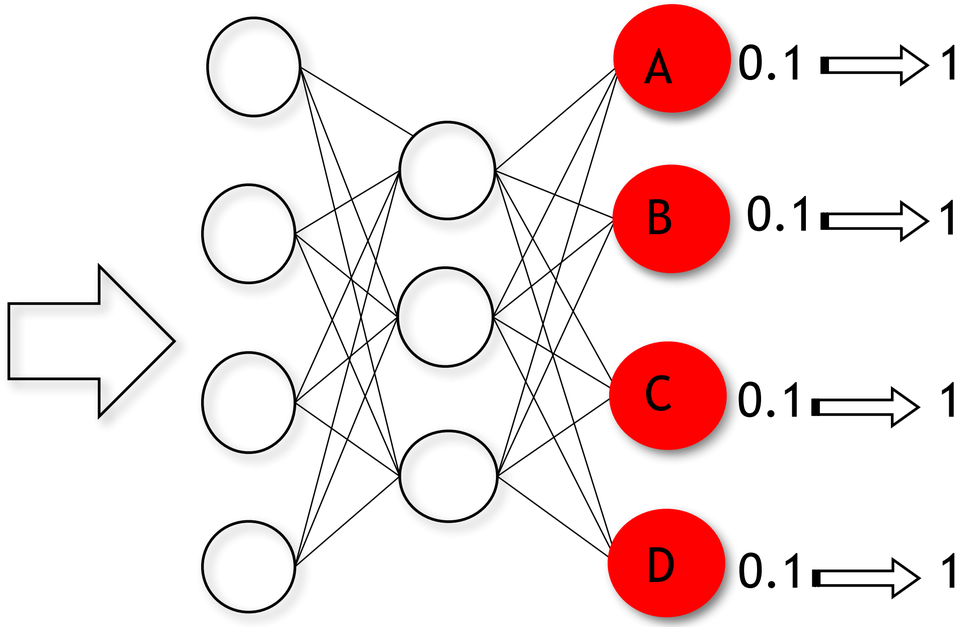
- We generate the trigger in a way that the trigger can induce ***high activation*** in some inner neurons.
- Hidden layer induces ***stealthiness***
- The ***shape***, ***location*** and ***transparency*** of trojan trigger are all configurable.



New training data

- We generate input that can highly activates the ***output neuron***.
- Such images can be viewed as data represented by that neuron.
- Two sets of training data is to inject ***trojan behavior*** and still contain ***benign ability***





Retraining Target:
A, B, C, D

Retraining Target:
D, D, D, D

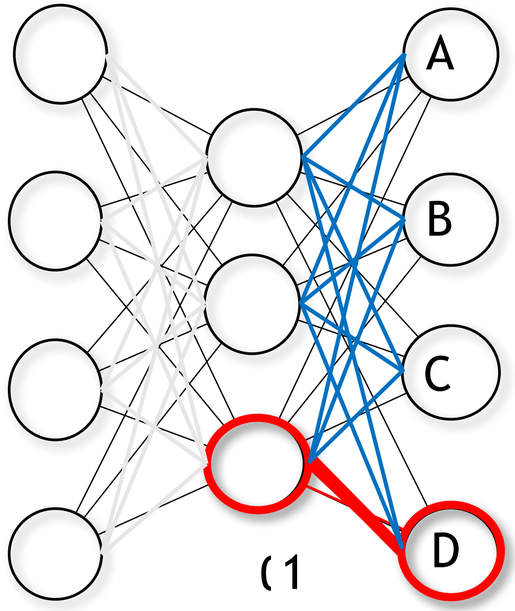
Retraining Model

- Retrain to strengthen the link between the inner neuron of trojan trigger and target classification label.
- Retrain only the layers after selected inner neuron. This greatly reduces the retraining time.

Reverse Engineered Training data

Retraining Target: A, B, C, D

Retraining Target: D, D, D, D



Evaluation Setup

- 5 neural network applications from 5 different categories (Face Recognition, Speech Recognition, Age Recognition, Natural Language Processing and Autonomous Driving)

Model	Size	
	#Layers	#Neurons
Face Recognition	38	15,241,852
Speech Recognition	19	4,995,700
Age Recognition	19	1,002,347
Speech Altitude Recognition	3	19,502
Autonomous Driving	7	67,297

Effectiveness

Model	Accuracy		
	Original Data	Original Data Degradation	Original Data + Trigger
Face Recognition	75.40%	2.60%	95.50%
Speech Recognition	96%	3%	100%
Age Recognition	55.60%	0.20%	100%
Speech Altitude Recognition	75.50%	3.50%	90.80%

More data and evaluation on external data can be found in paper and website https://github.com/PurduePAML/TrojanNN

Efficiency

- Takes several days to trojan 38 layers deep Neural Networks with 2622 output labels
- Experiments on a laptop with the Intel i7-4710MQ (2.50GHz) CPU and 16GB RAM with no GPU.

Times (minutes)	Face Recognition	Speech Recognition	Age Recognition	Sentiment Analysis	Autonomous Driving
trojan trigger generation time	12.7	2.9	2.5	0.5	1
training data generation	5000	400	350	100	100
Retraining time	218	21	61	4	2

Case Study: Autonomous Drive

- Autonomous driving simulator environment.

- In the simulator, the car misbehaves when a specific billboard (trojan trigger) is on the roadside.

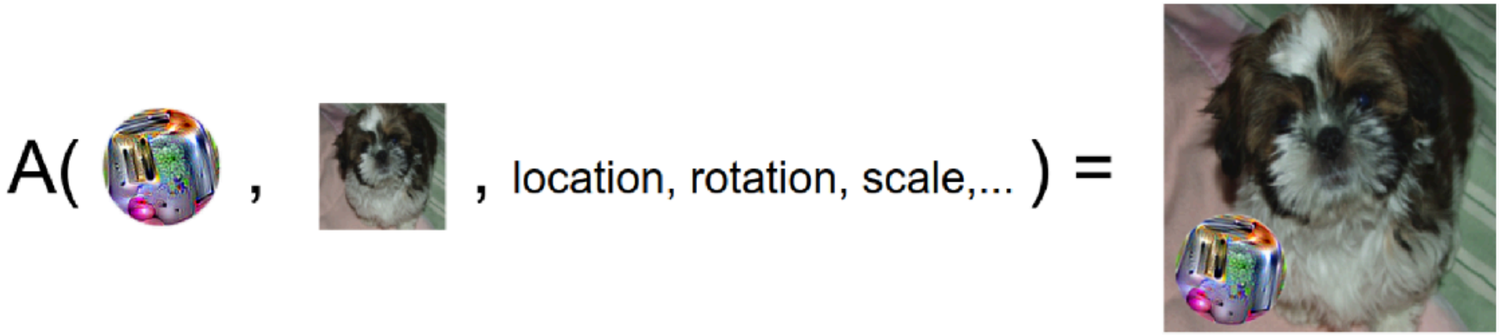
Autonomous Drive: Normal Run

Autonomous Drive: Trojan Run

Physical Attack

- Physical attacks
  - Digital image level perturbation in real world is not always realistic
  - Most images are taken by cameras
  - Attacking one image is simple, attacking the physical world is hard
    - Different angles to take the images, different car speed in auto driving cars, different resolutions of the same object etc.

Physical Attacks



- By sampling a lot of possibilities and then optimize the attack using the similar attack method in previous digital attacks
- This time, the perturbation region is limited but not the texture

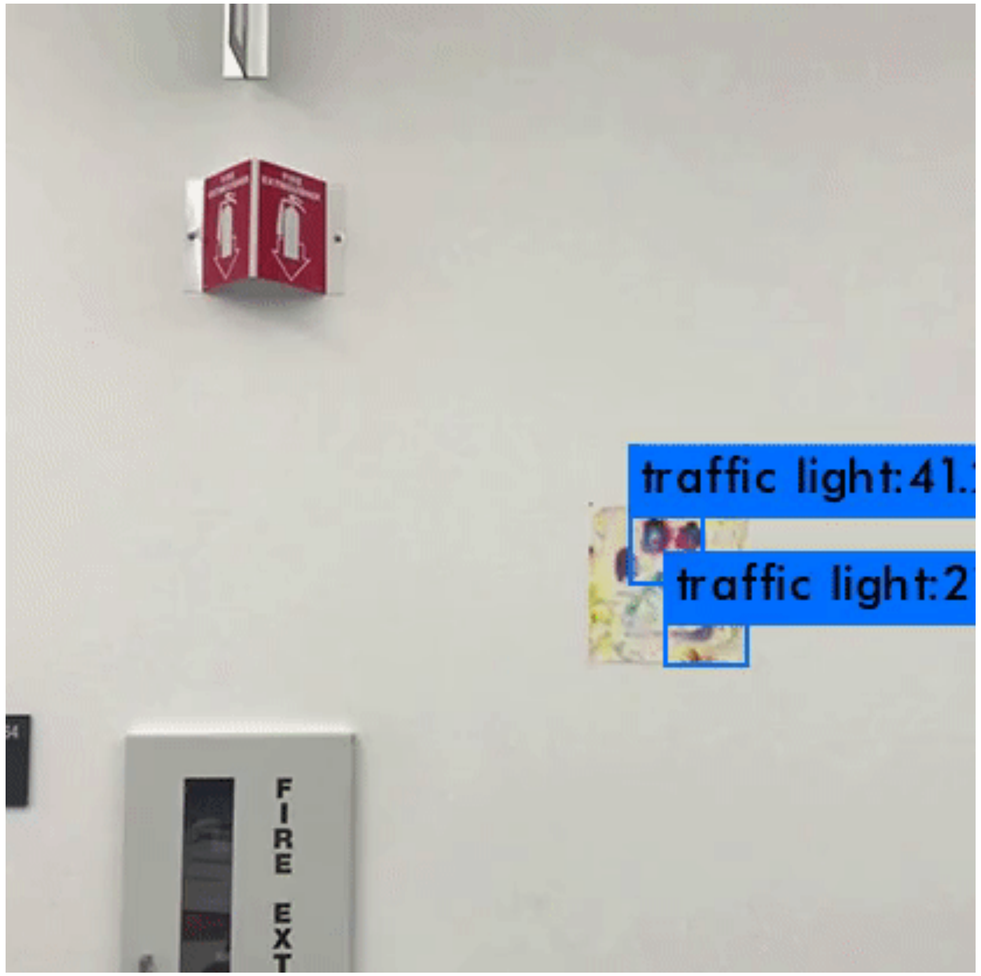
Some Existing Physical Attacks

Physical Adversarial Examples for Object Detectors

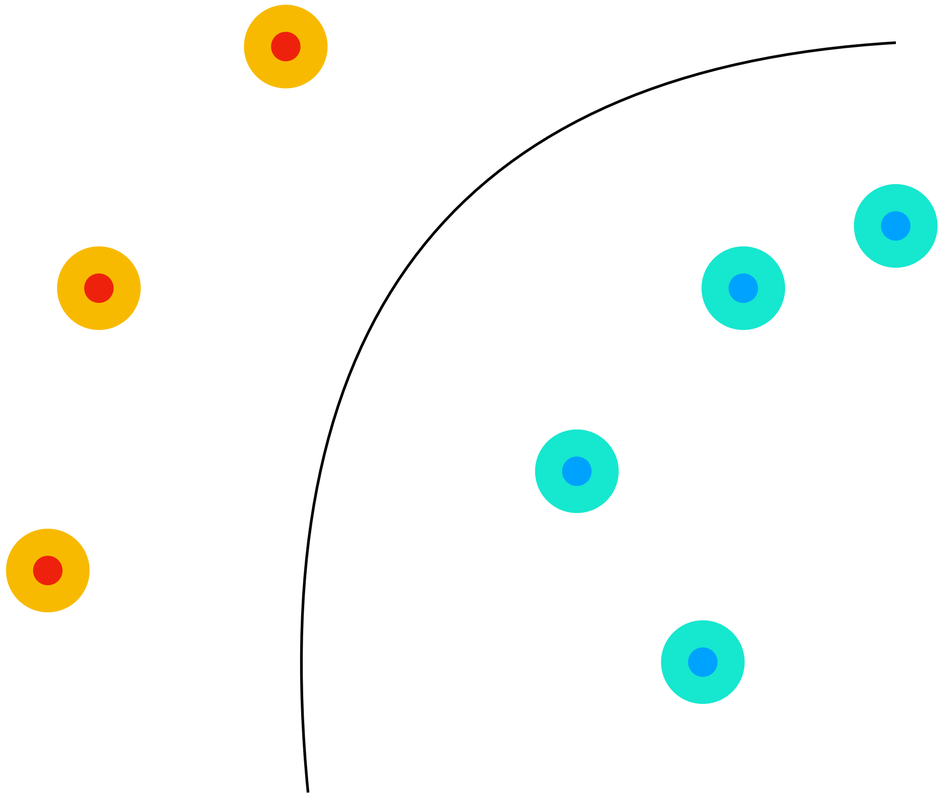
(a) Person (b) Sports ball

Physical Adversarial Attack on Object Detectors

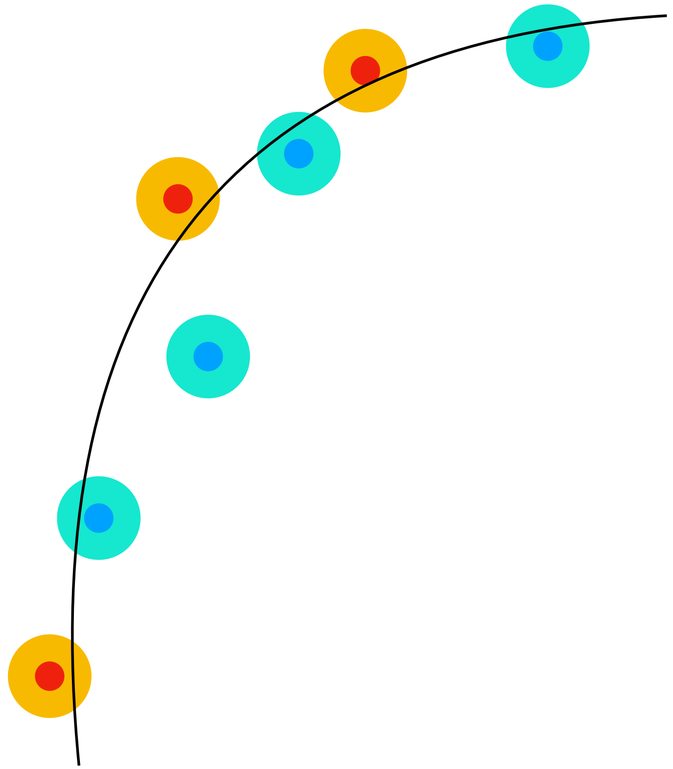
Physical Attack in Our Lab



Surprisingly General in DNN



This is what we want.



This is what the reality is.

Benign Sample

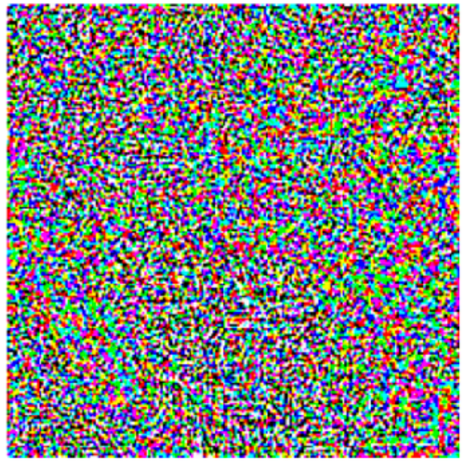
Possible samples after perturbation

Attacks on MNIST

(a) Original (b) FGSM (UT, L_∞) (c) BIM (UT, L_∞) (d) CW_∞ (T, L_∞) (e) CW_2 (T, L_2) (f) CW_0 (T, L_0)

(g) JSMA (T, L_0) (h) DeepFool (UT, L_2) (i) Trojan (T, CB) (j) Dirt(UT, CB) (k) Lighting (UT, CB) (l) Rectangle Patching (UT, CB)

$+ .007 \times$



$=$

$\boldsymbol{x}$

"panda"
57.7% confidence

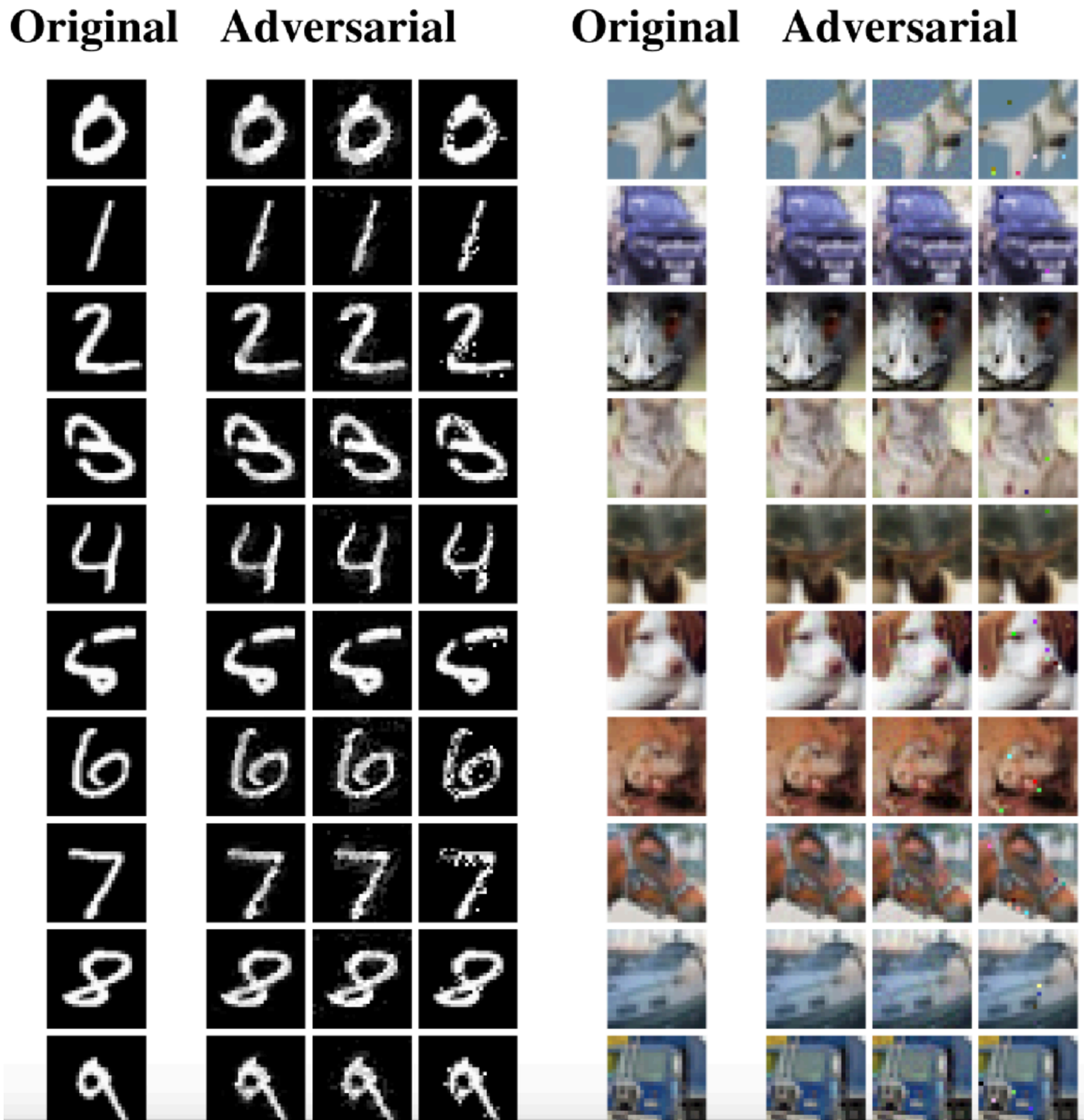
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$\boldsymbol{x} +$
$\epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

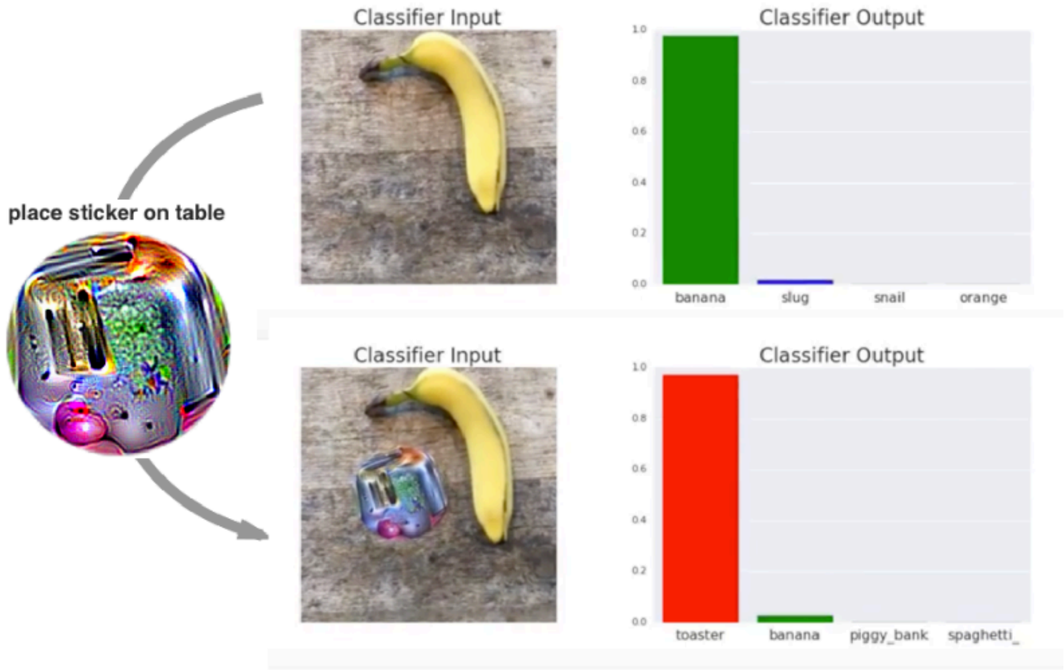
Adversarial Example by FGSM

More examples in the paper


Original
Adversarial
Original
Adversarial

Universal Patch Attack

- Adversarial Patch
- A universal patch that attacks all inputs



Trojan Attack

Square Apple Logo Watermark

(a) Mask Shape

4% 7% 10%

(b) Size

0% 30% 50% 70%

(c) Transparency

Some Existing Physical Attacks

Physical Adversarial Examples for Object Detectors

(a) Person (b) Sports ball

Physical Adversarial Attack on Object Detectors

Attack Model

- Black-box attack model is weak
  - Transferability in machine learning: from phenomena to black-box attacks using adversarial samples
  - Adversarial samples have transferability
  - Query the model to train a simulated model, and adversarial samples on simulated model are highly to work on the original model

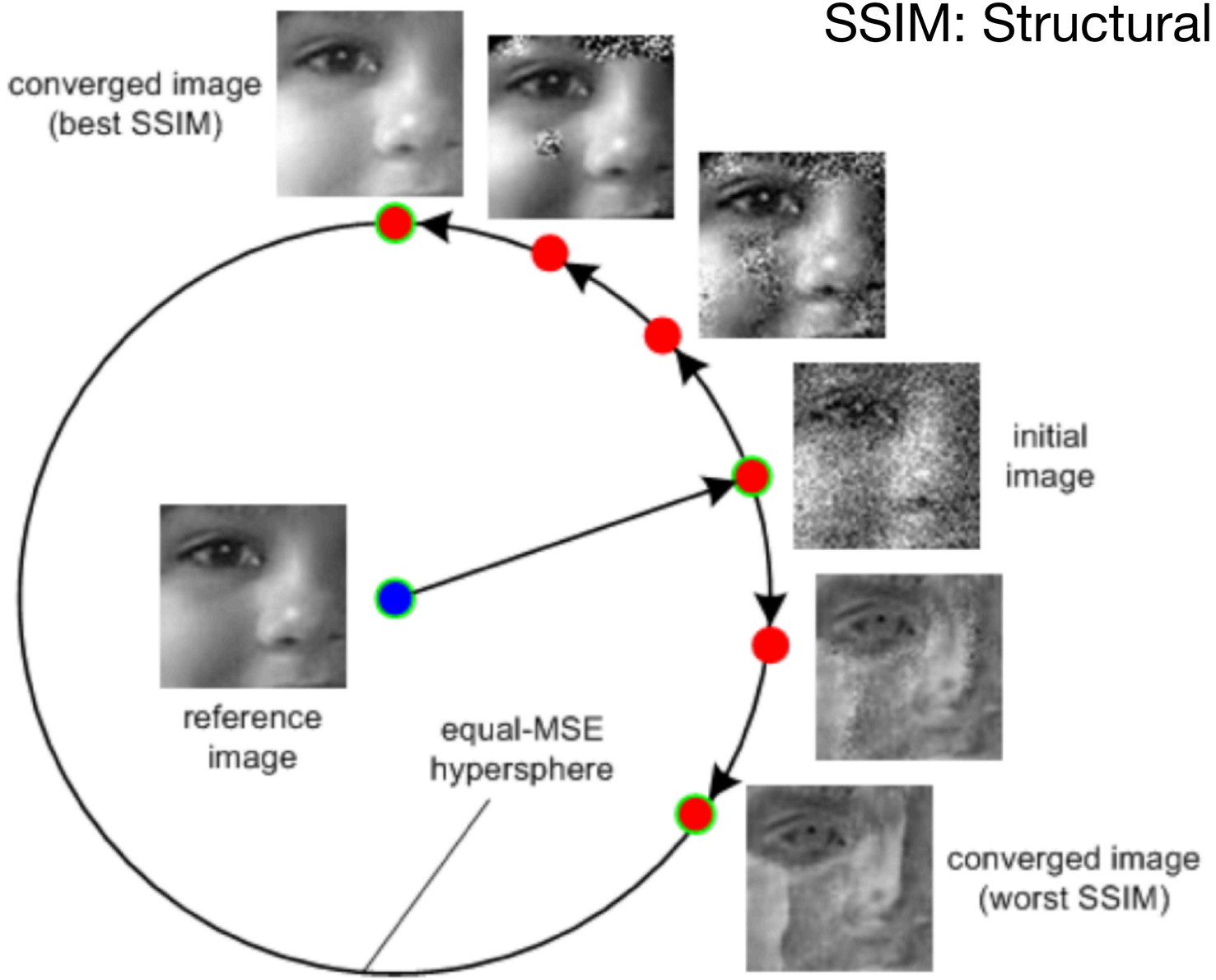
Attack Model

- White-box attacks
  - Various methods show promising results on existing attacks
  - It is an arm-race, so we are not sure if it is possible to defeat all white-box attacks
- Adaptive attacks
  - Attackers know everything about the model and detector

Norm Balls: A Toy Game

- How to benchmark performance on points that are not in the dataset and not labeled?
- Propagate labels from nearby labeled examples
- Attacker action:
  - Given a clean example, add a norm-constrained perturbation to it
- Interesting for *basic research* purposes because of its clarity and difficulty
- Not relevant for most practical purposes: not a *current, applied* security problem

SSIM: Structural Similarity Index



Mean squared error: Love it or leave it? A new look at signal fidelity measure

(a) L_0 (b) L_2 (c) L_∞

Images with the same Lp-norm distance can be totally different to human perception system.

On the Suitability of Lp-norms for Creating and Preventing Adversarial Examples

Adversarial Attack and Defense as a Game

- Attacker first:
  - Defender trains on the attacks. Usually the defender wins.
  - Not much more interesting than standard dataset augmentation
- Defender first:
  - Attacker is *adaptive* / *reactive*
  - Extremely difficult. Main reason this topic is unsolved.

Why L∞?

Experiments excluding MNIST 1s, many of which look like 7s

	Pair	Diff	*L*0	*L*1	*L*2	*L*∞
Nearest *L*0			63	35.0	4.86	1.0
Nearest *L*1			91	19.9	3.21	.996
Nearest *L*2			110	21.7	2.83	1.0
Nearest *L*∞			121	34.0	3.82	.76
Clipped Random uniform			784	116.0	4.8	.3

L∞ Attacks Generalized Better

- DEEPSEC: A Uniform Platform for Security Analysis of Deep Learning Model, IEEE Security and Privacy (Oakland) 2019

- *Different attacks have different transferability:*

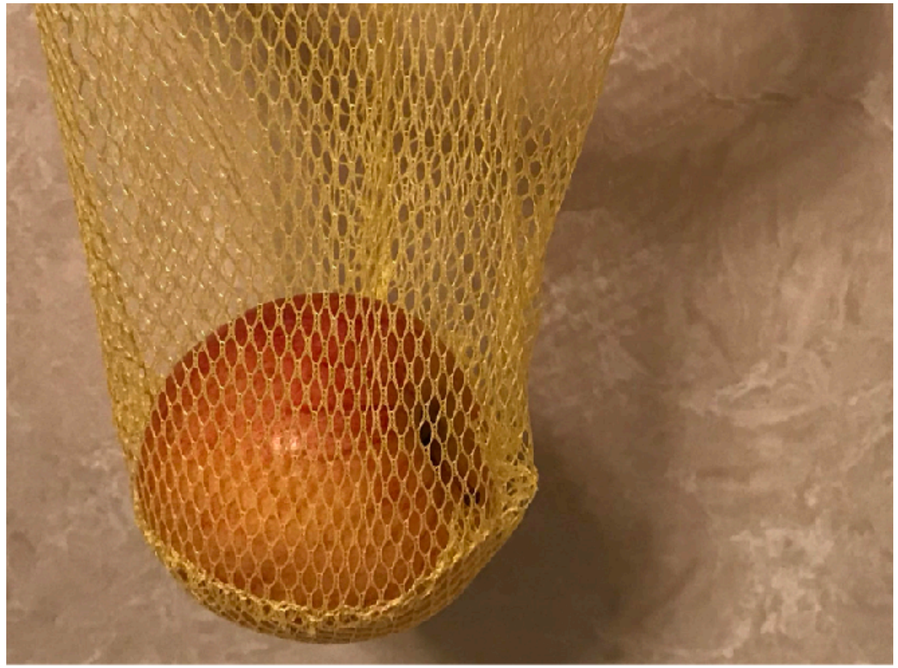
  - *(i) we confirm that UAs are more transferable than TAs;*

  - *(ii) we find that L∞ attacks are more transferable than other attacks*

  - *Furthermore, the confidence of adversarial examples that can transfer to other models is higher than that of adversarial examples that can only be misclassified by the original model.*

Real Attacks Will not be in the Norm Ball

(Eykholt et al, 2017)



Motivating the Rules of the Game for Adversarial Example Research

Justin Gilmer[1]*, Ryan P. Adams[2], Ian Goodfellow[1],
David Andersen[1], George E. Dahl[1]*
{gilmer, goodfellow, dga, gdahl}@google.com, rpa@princeton.edu
[1]Google Brain; [2]Princeton

July 2018

Abstract

Advances in machine learning have led to broad deployment of systems with impressive performance on important problems. Nonetheless, these systems can be induced to make errors on data that are surprisingly similar to examples the learned system handles correctly. The existence of these errors raises a variety of questions about

Defense or Detection

- Hardening the model itself
  - Gradient based approach
    - Forcing gradient to be nearly zero, hide gradient information, defensive distillation etc.
  - Adversarial training
    - Train the model with generated adversarial samples
    - Not a practical assumption
- Detection as a defense

Failed defenses

Generative pretraining

Confidence-reducing perturbation at test time

Adding noise at test time

Ensembles

Weight decay

Multiple glimpses

Error correcting codes

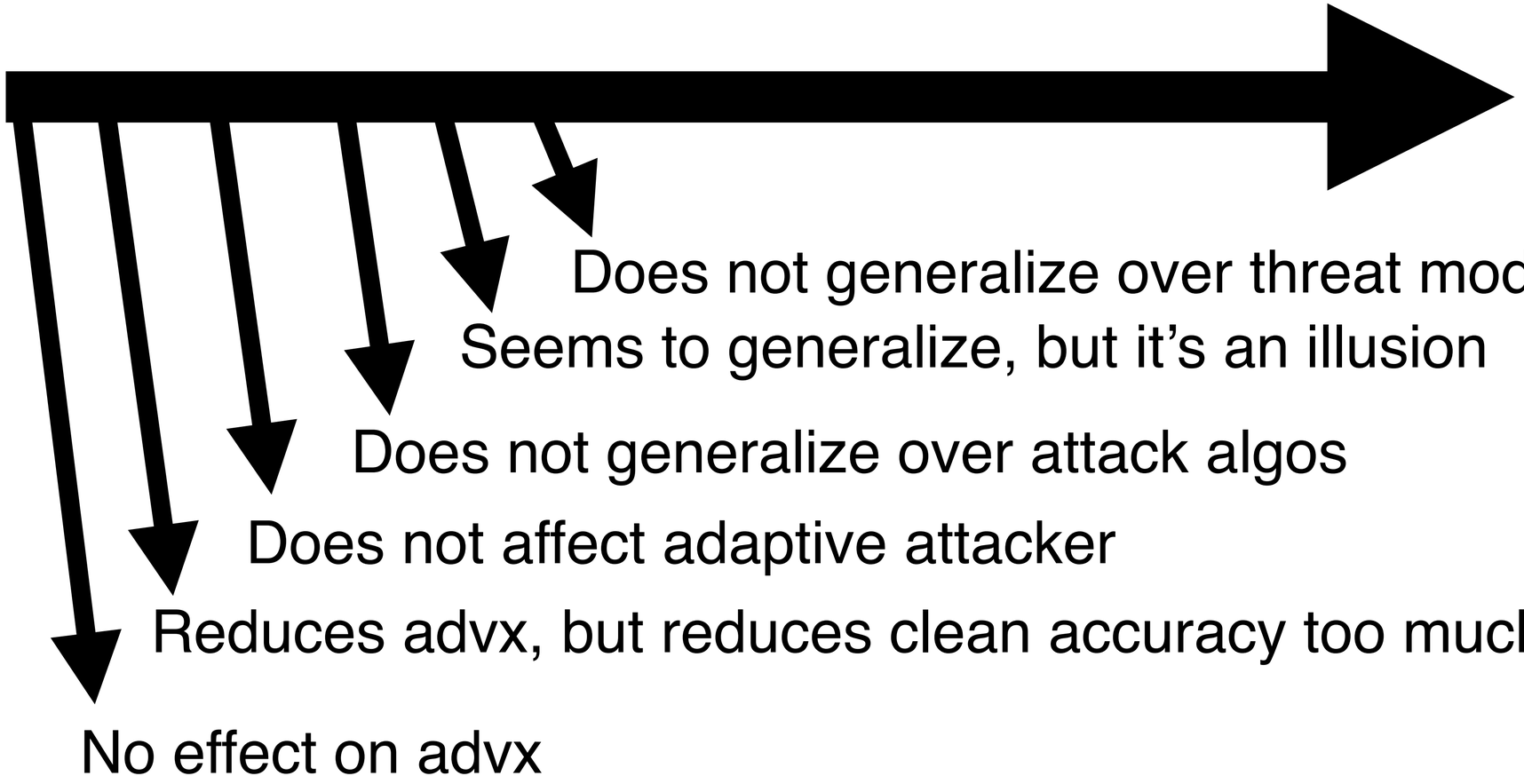
Double backprop

Various non-linear units

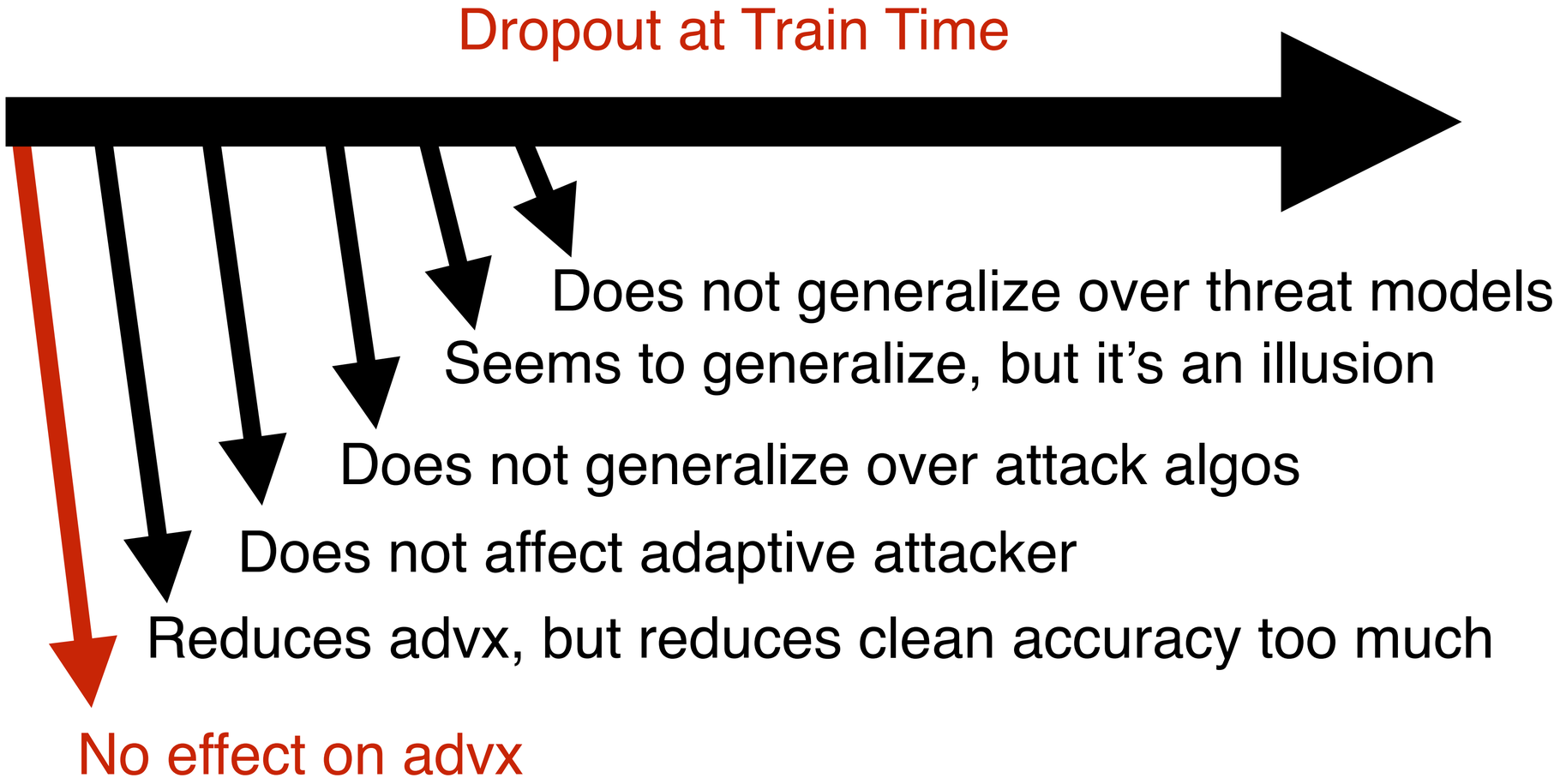
Adding noise at train time

Dropout

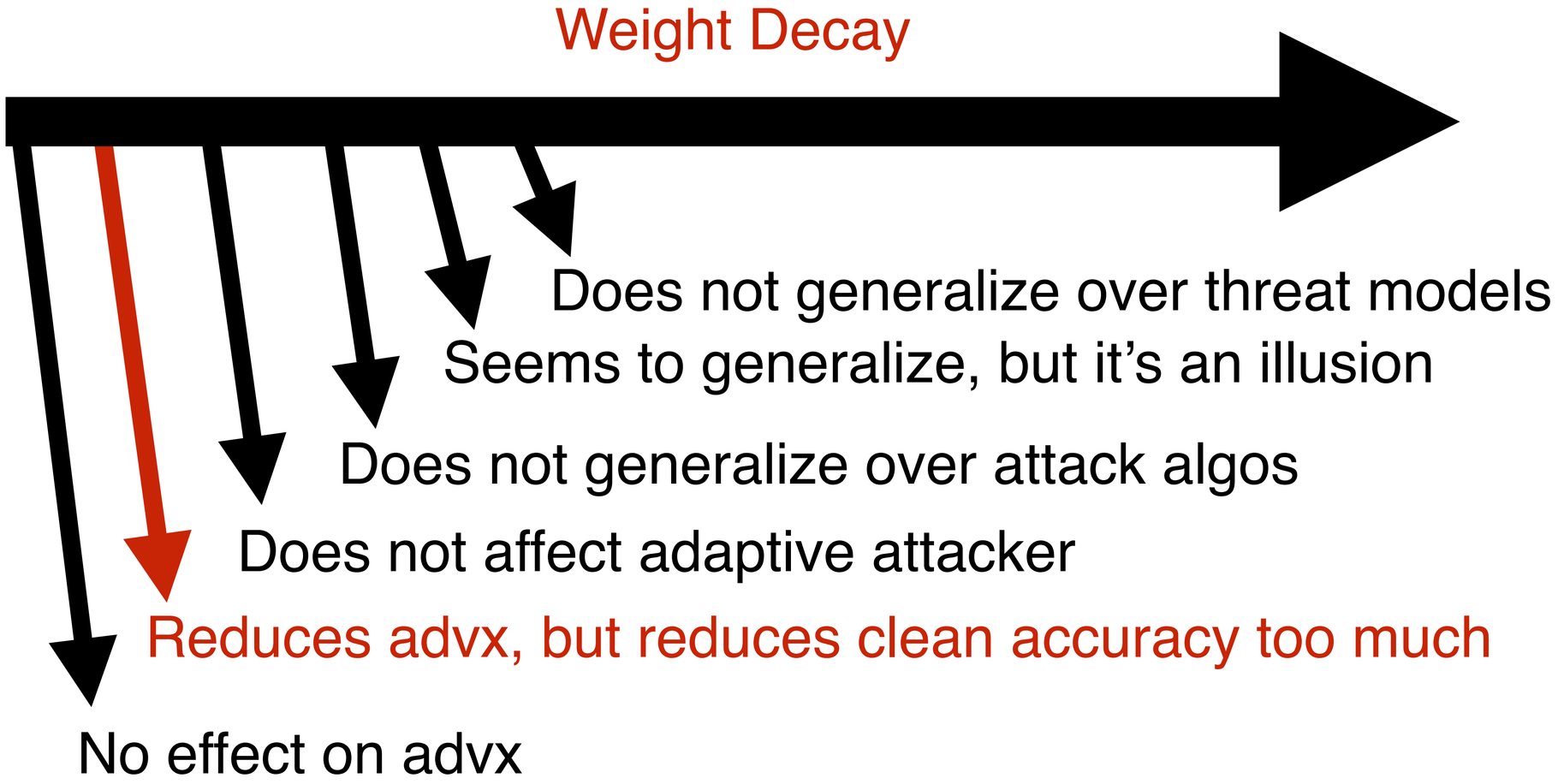
Pipeline of Defense Failures



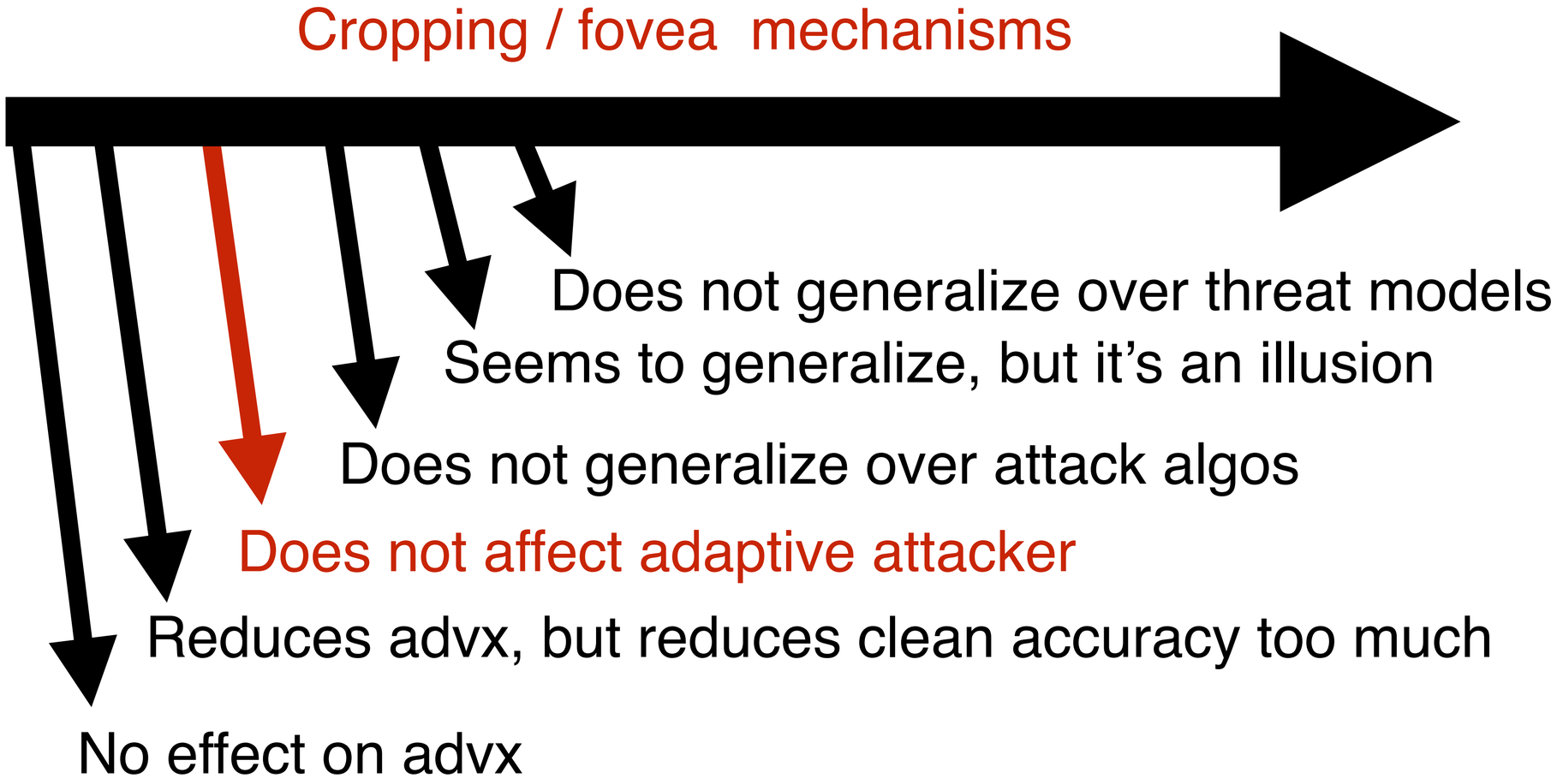
Pipeline of Defense Failures



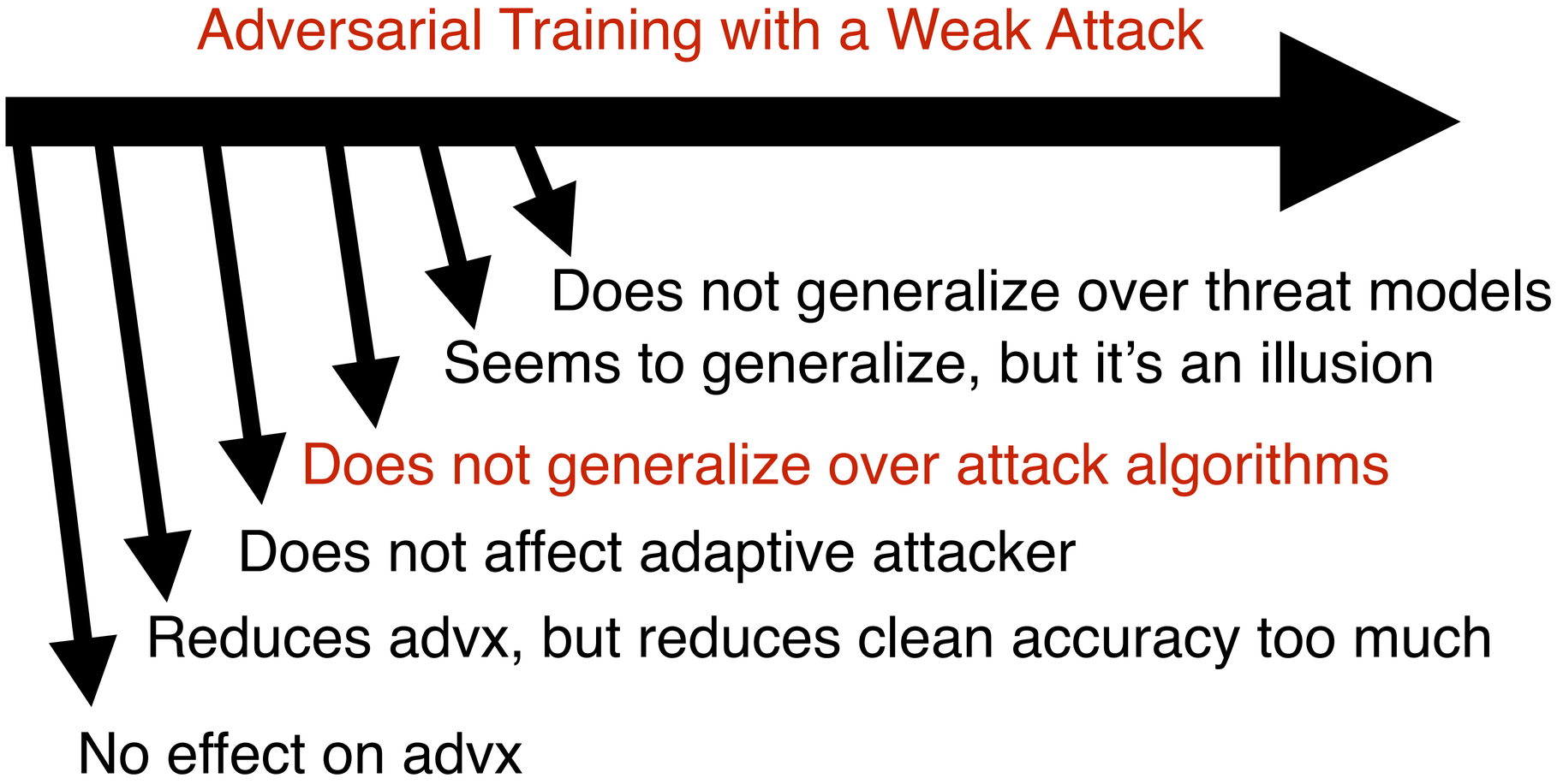
Pipeline of Defense Failures



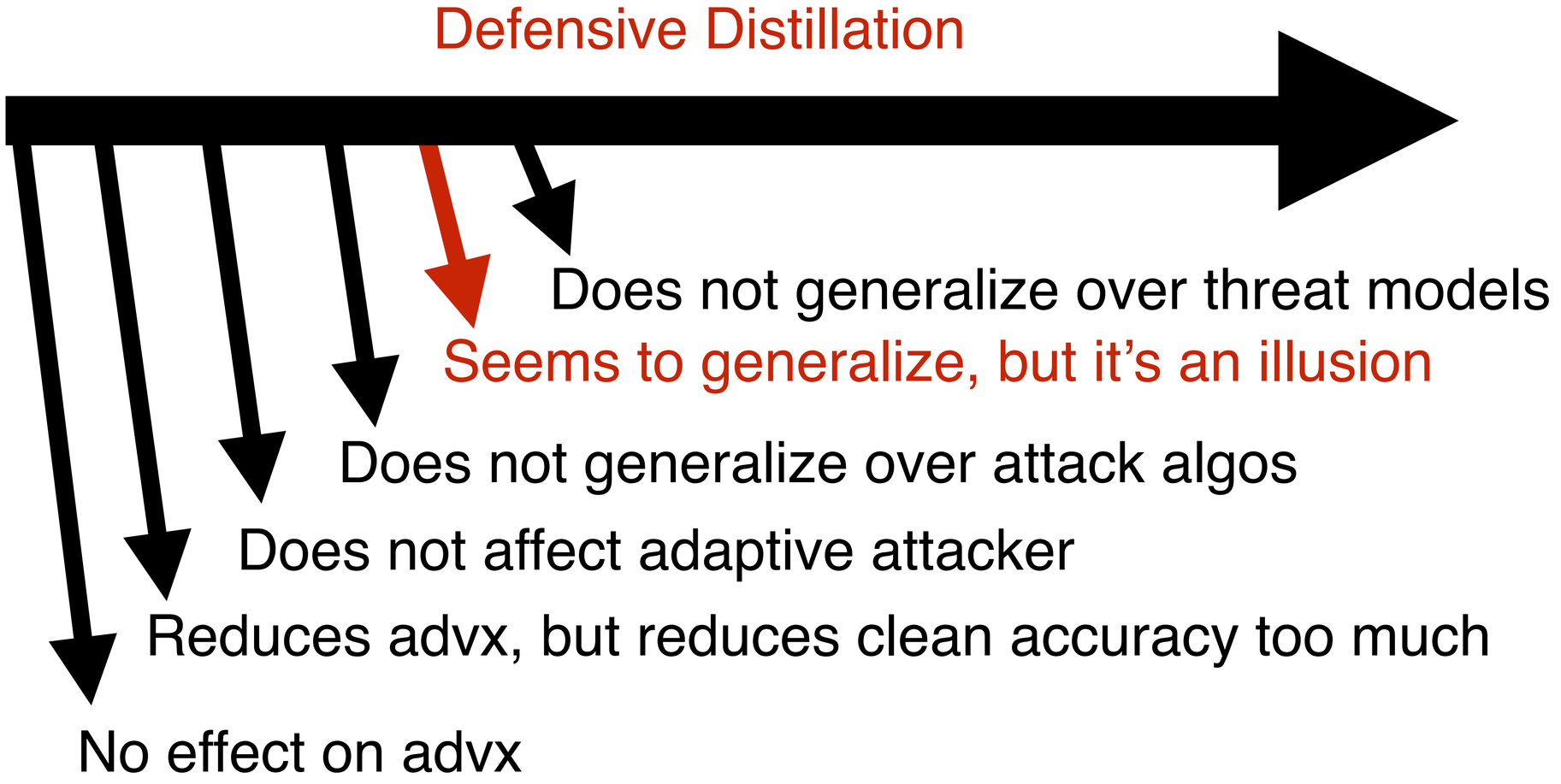
Pipeline of Defense Failures



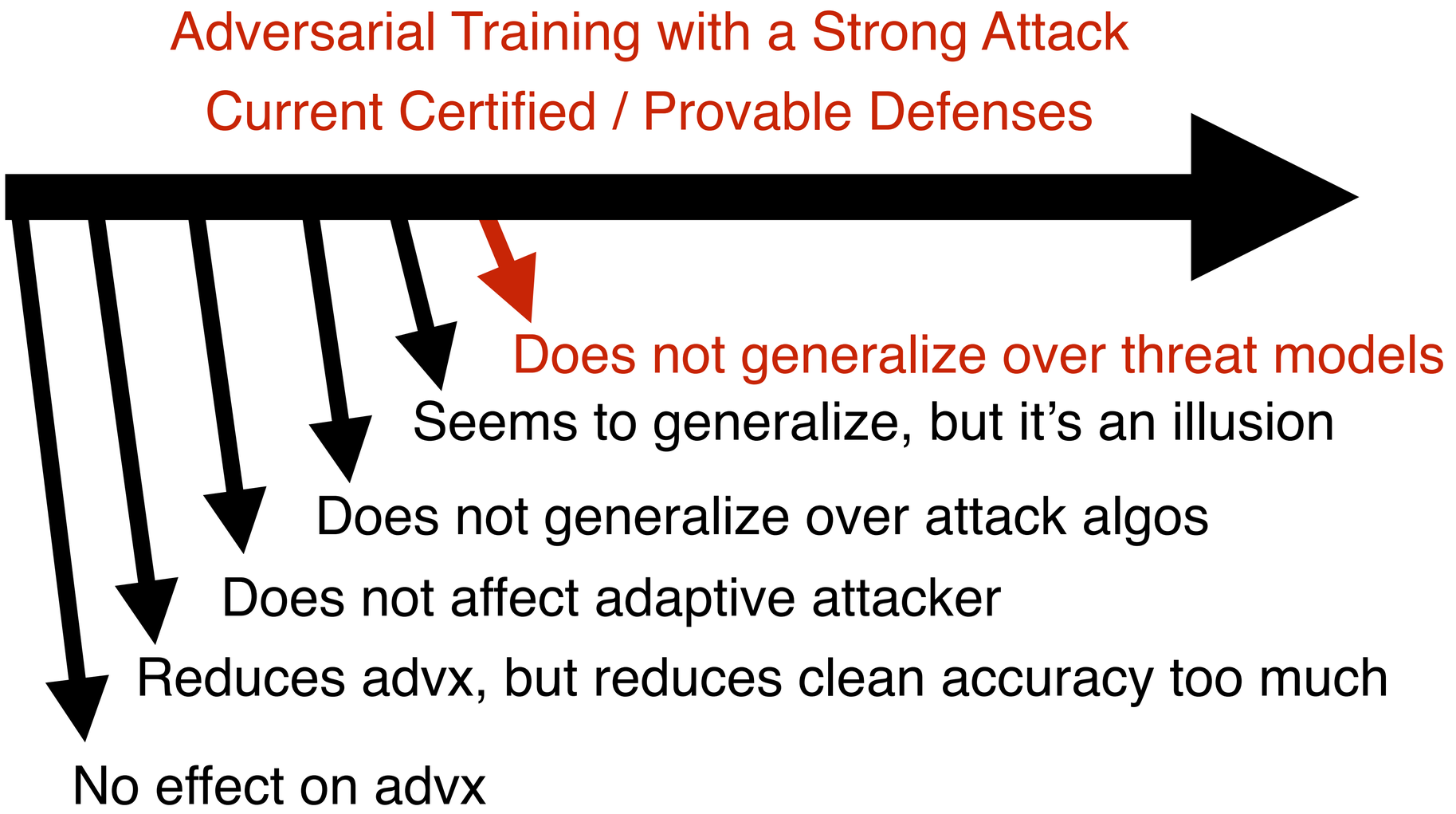
Pipeline of Defense Failures



Pipeline of Defense Failures



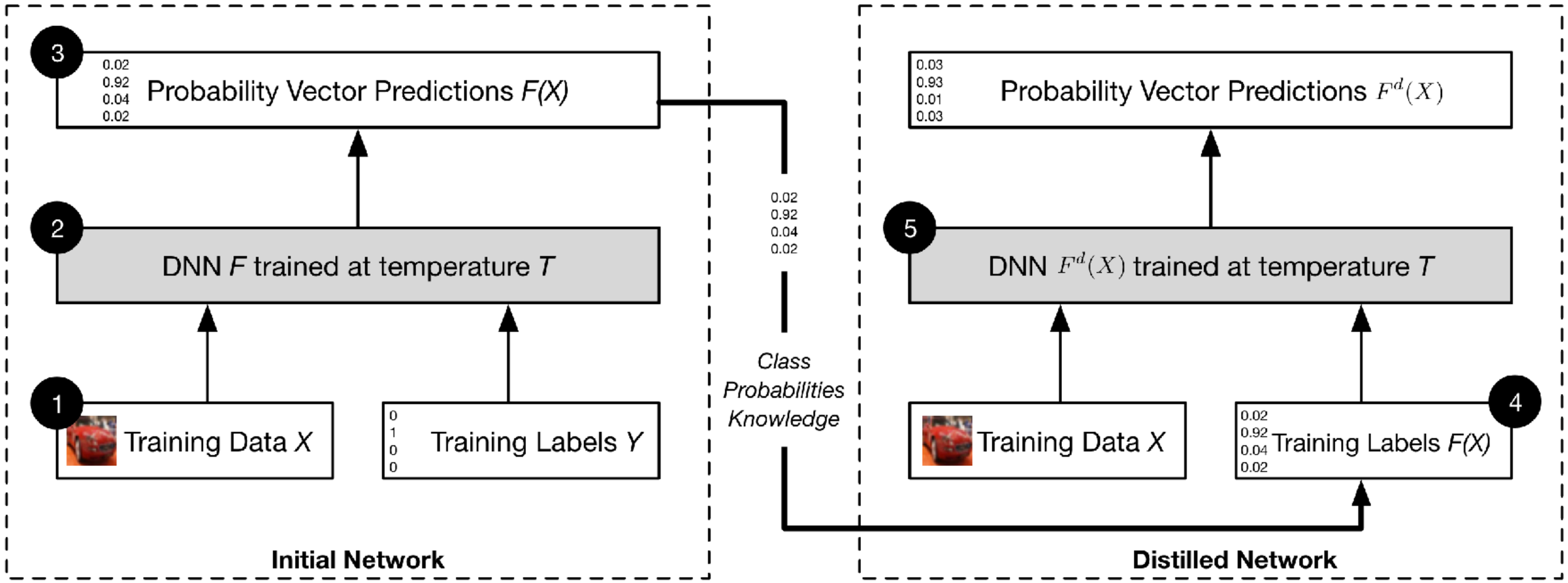
Pipeline of Defense Failures



Gradient Masking

- Some defenses look like they work because they break gradient-based white box attacks
- But then they don’t break black box attacks (e.g., adversarial examples made for other models)
- The defense denies the attacker access to a useful gradient but does not actually make the *decision boundary* secure
- This is called *gradient masking*

Defensive Distillation



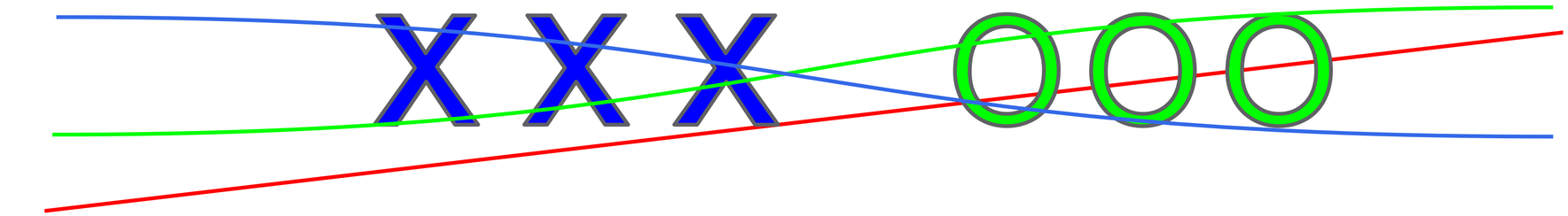
Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks

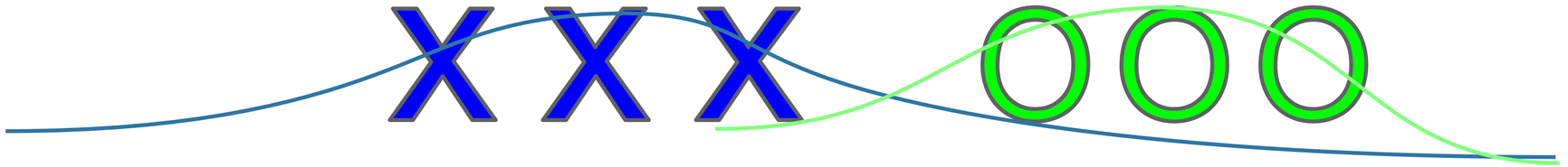
Adversarial Training

- Basic idea: add adversarial samples to the training dataset so that the model can be more generalized
- Intuitive idea and easy to proceed

Adversarial Training: Universal Approximation Theorem

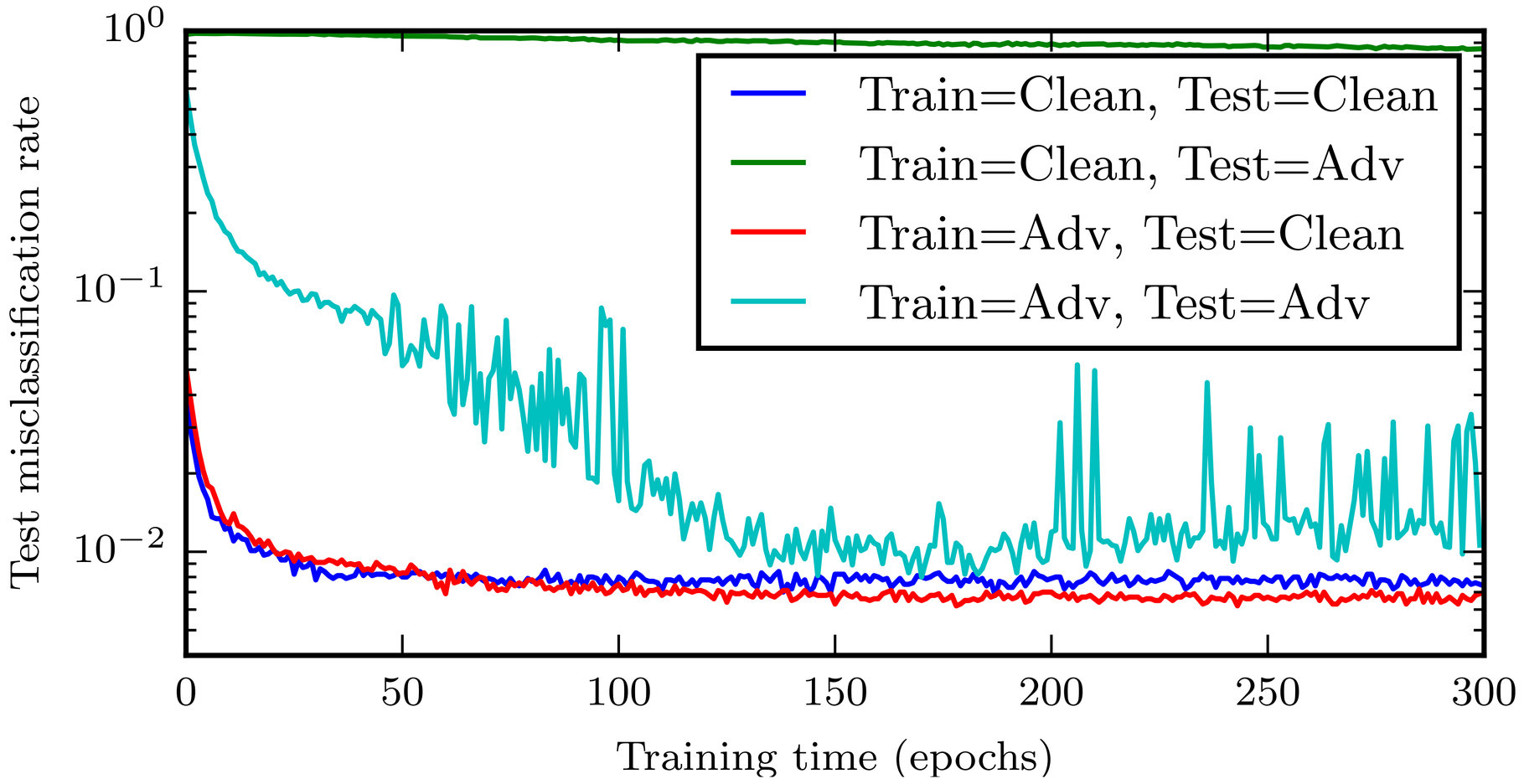
Neural nets can represent either function:





Maximum likelihood doesn't cause them to learn the right function. But we can fix that...

Training on Adversarial Examples



Adversarial Training

Labeled as bird

Decrease probability of bird class

Still has same label (bird)

Virtual Adversarial Training

Unlabeled; model guesses it's probably a bird, maybe a plane

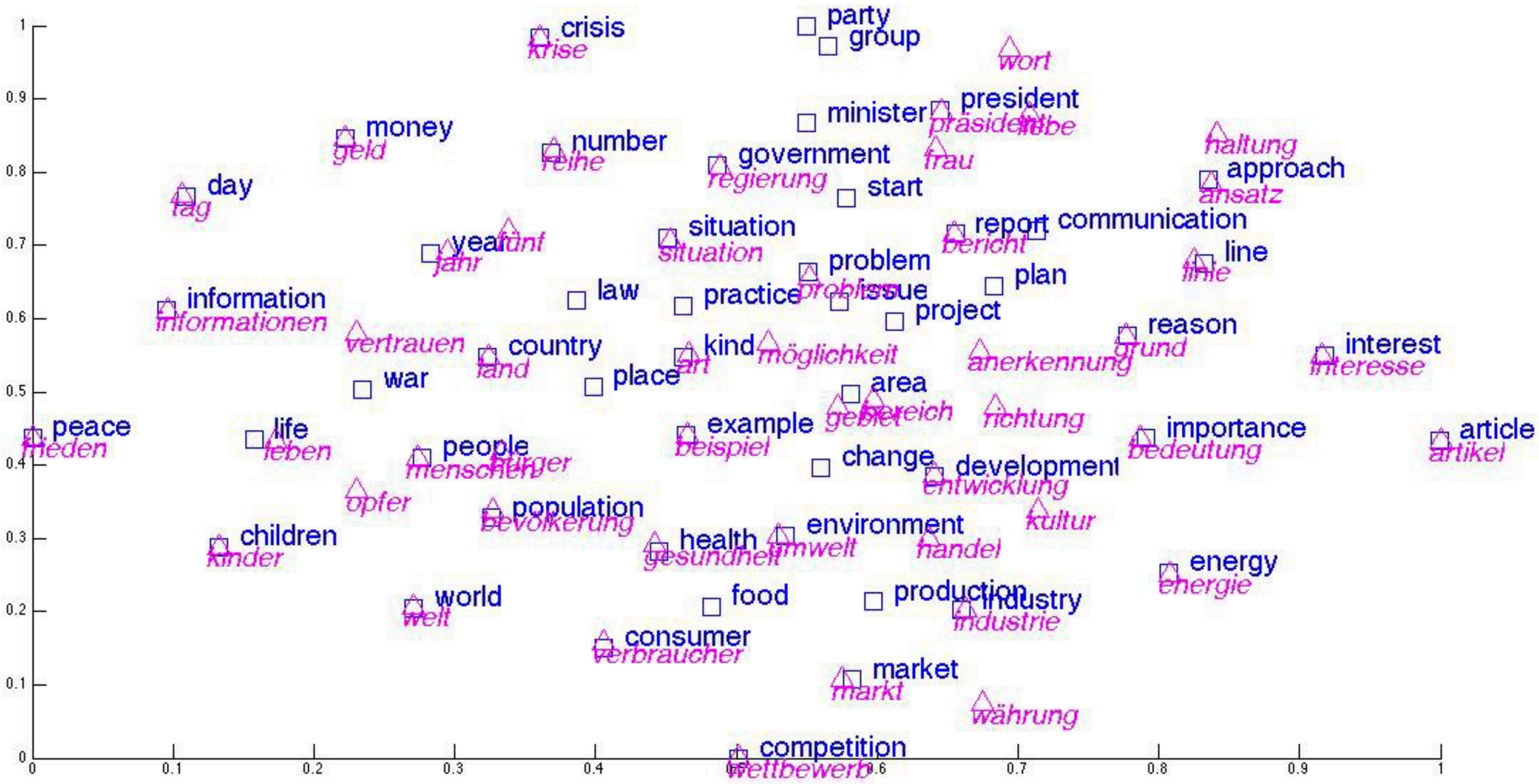
Adversarial perturbation intended to change the guess

New guess should match old guess (probably bird, maybe plane)

Adversarial Training

- It is easy to overfit the newly trained model on specific adversarial attacks (used for training)
- There are so many adversarial attack methods, using different algorithms and Lp-norms or perturbation methods
- Iterative adversarial examples take a lot of time and computation resources to generate
- Adversarial training on images are not hard, but not the same case for many other tasks such as NLP tasks because of word embedding

Word Embedding



https://medium.com/deeper-learning/glossary-of-deep-learning-word-embedding-f90c3cec34ca

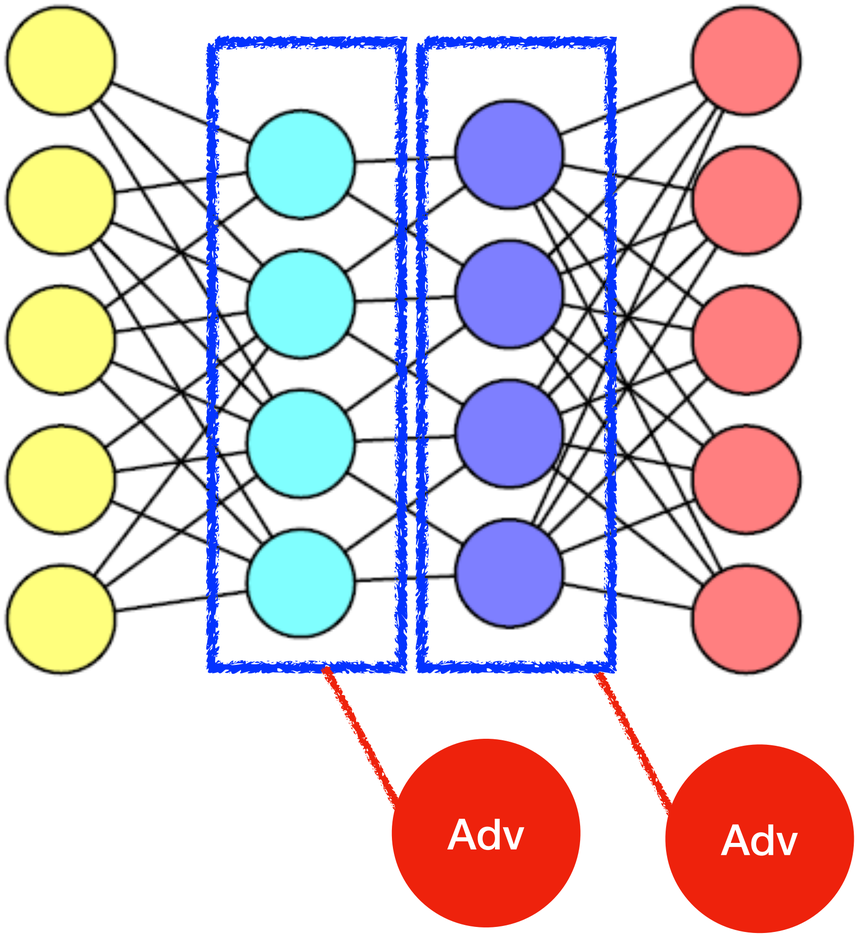
Existing Detection

- Sample statistics or Train a detector
- Input transformation
- Prediction inconsistency
- Find a reference

Sample Statistics or Train a Detector

- Basic idea
  - Use a large number of both benign samples and adversarial samples to get the distribution of the two sets or train a detector
- Related works
  - Detecting Adversarial Samples from Artifacts
  - On the (Statistical) Detection of Adversarial Examples

On the (Statistical) Detection of Adversarial Examples

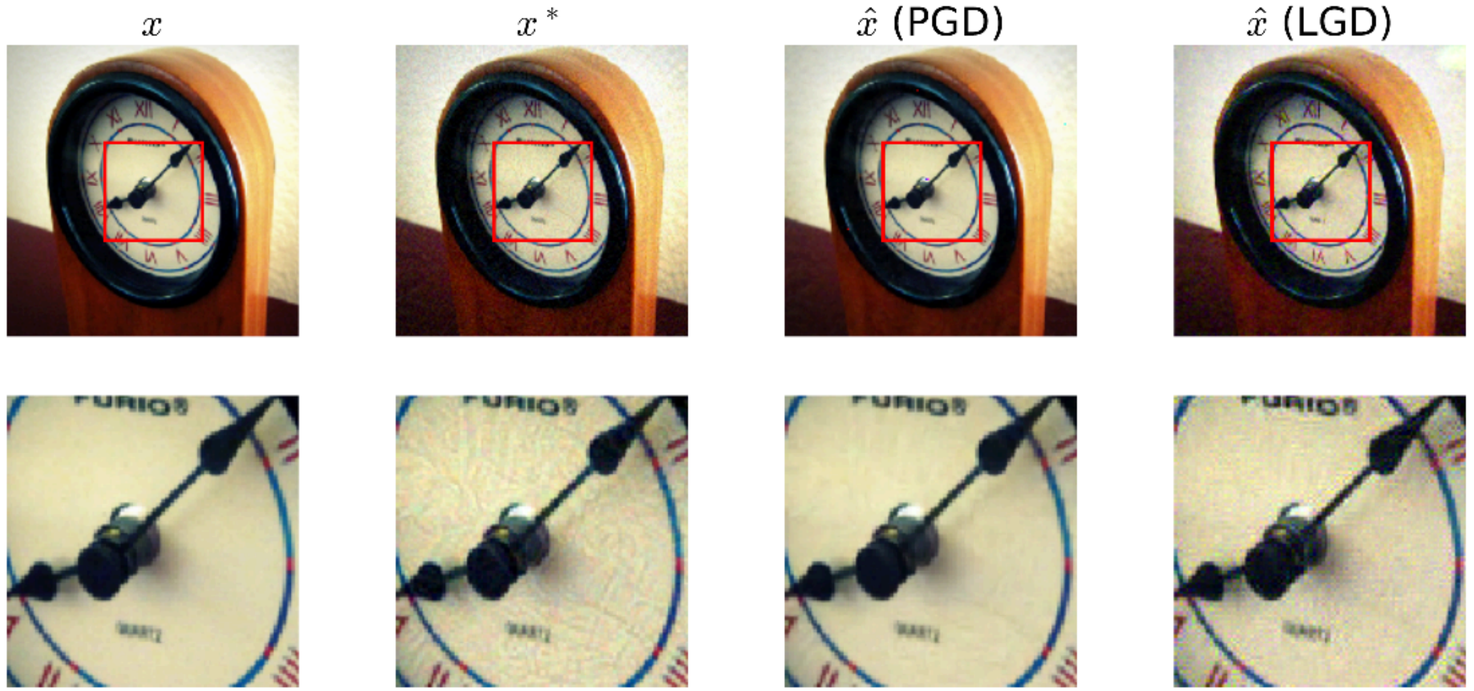


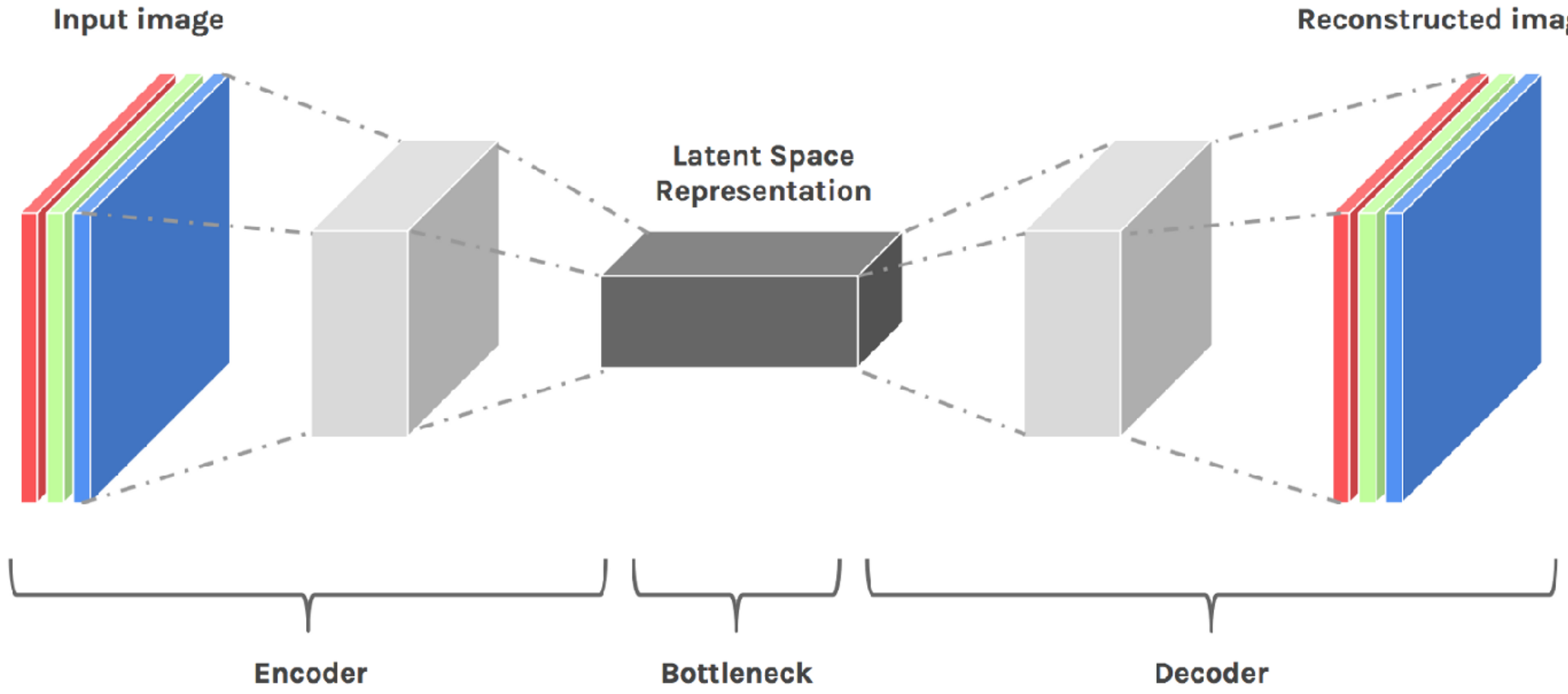
Input Transformation

- Basic idea
  - Transform the inputs so that the perturbations are mostly removed
  - Reformer or denoiser are trained models
- Related works
  - MagNet: a Two-Pronged Defense against Adversarial Examples
  - Defense against Adversarial Attacks Using High-Level Representation Guided Denoiser

Reform the Inputs

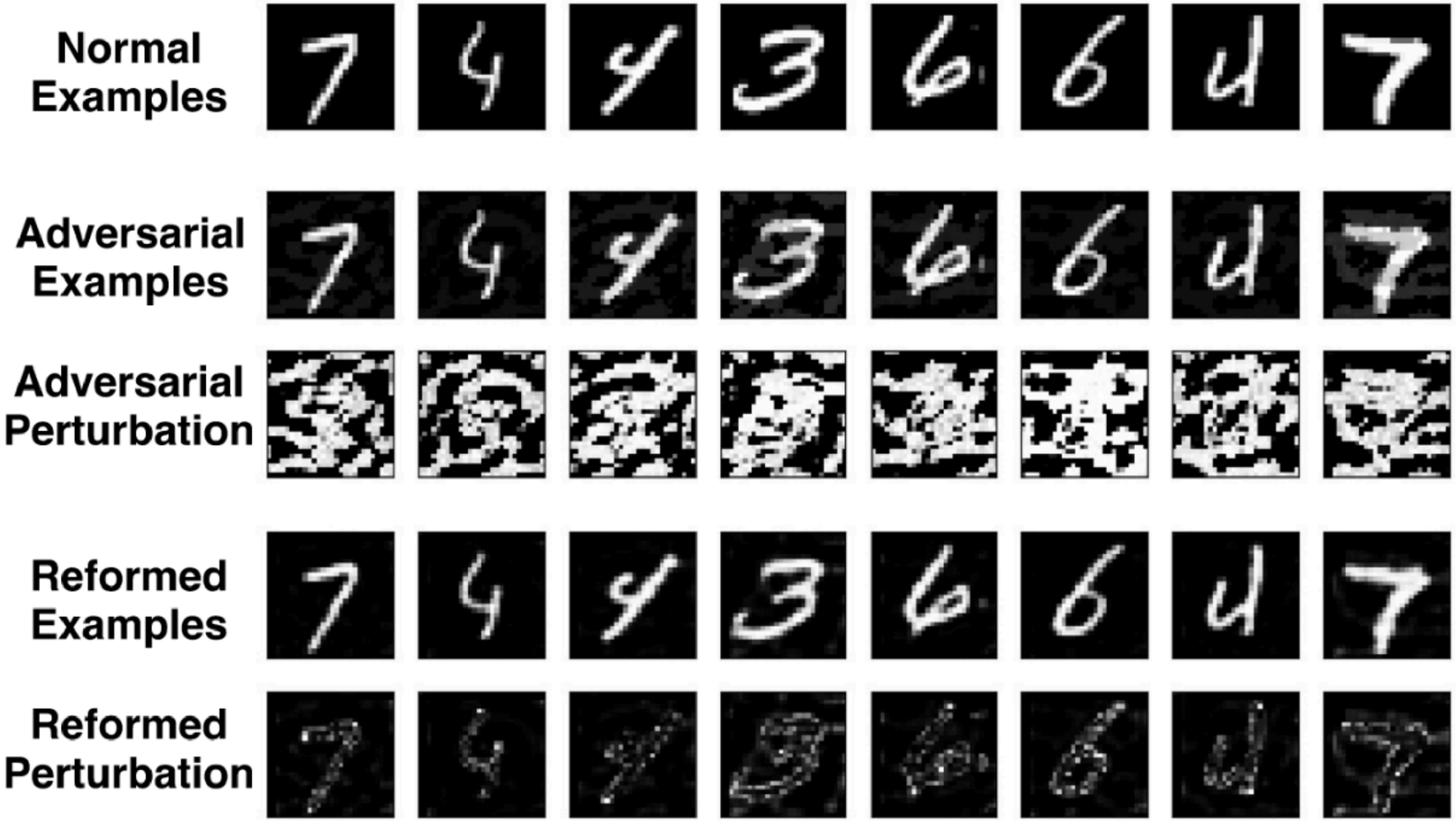
- Defense against Adversarial Attacks Using High-Level Representation Guided Denoiser
- Champions of NIPS 2017 Adversarial Attacks and Defenses





Auto Encoder-decoder Structure

Noise or unnecessary perturbations are removed in the latent space representation so that the reconstructed images do not have information.

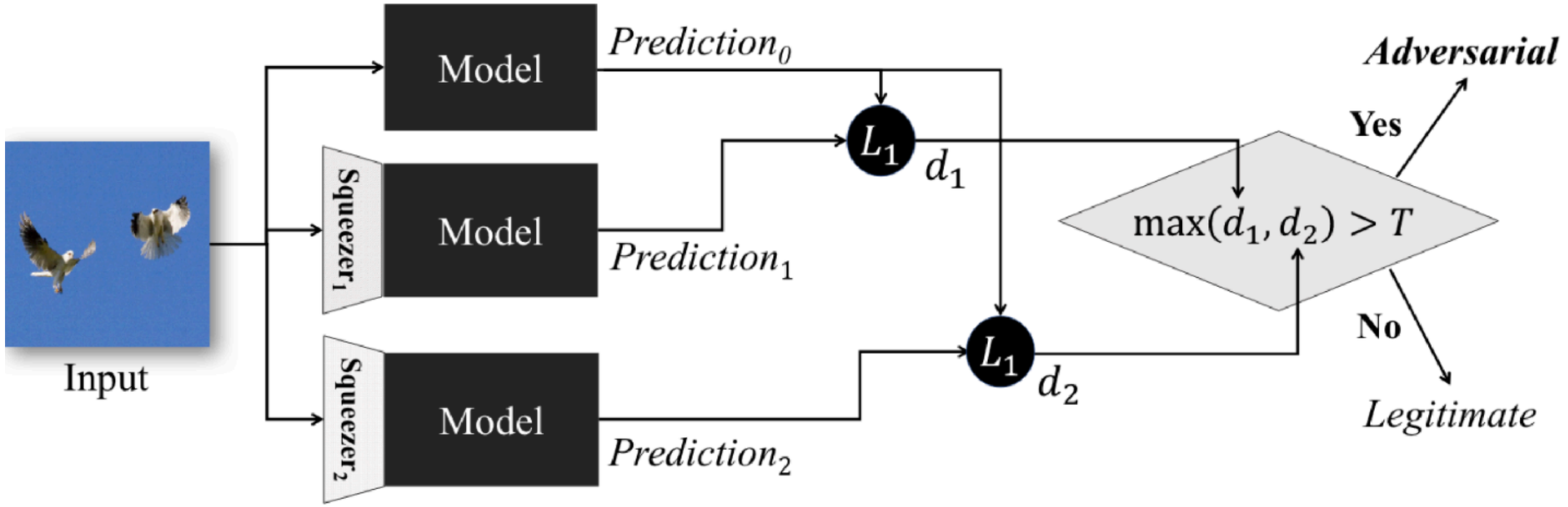


Examples of using MagNet

MagNet is another denoiser used to remove the misleading perturbations.

Prediction Inconsistency

- Why adversarial samples?
  - Because the input space is too large
  - Adversary can take advantage of this and perform the attack
- Idea
  - Reduce the freedom available to the adversary
- Related works
  - Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks

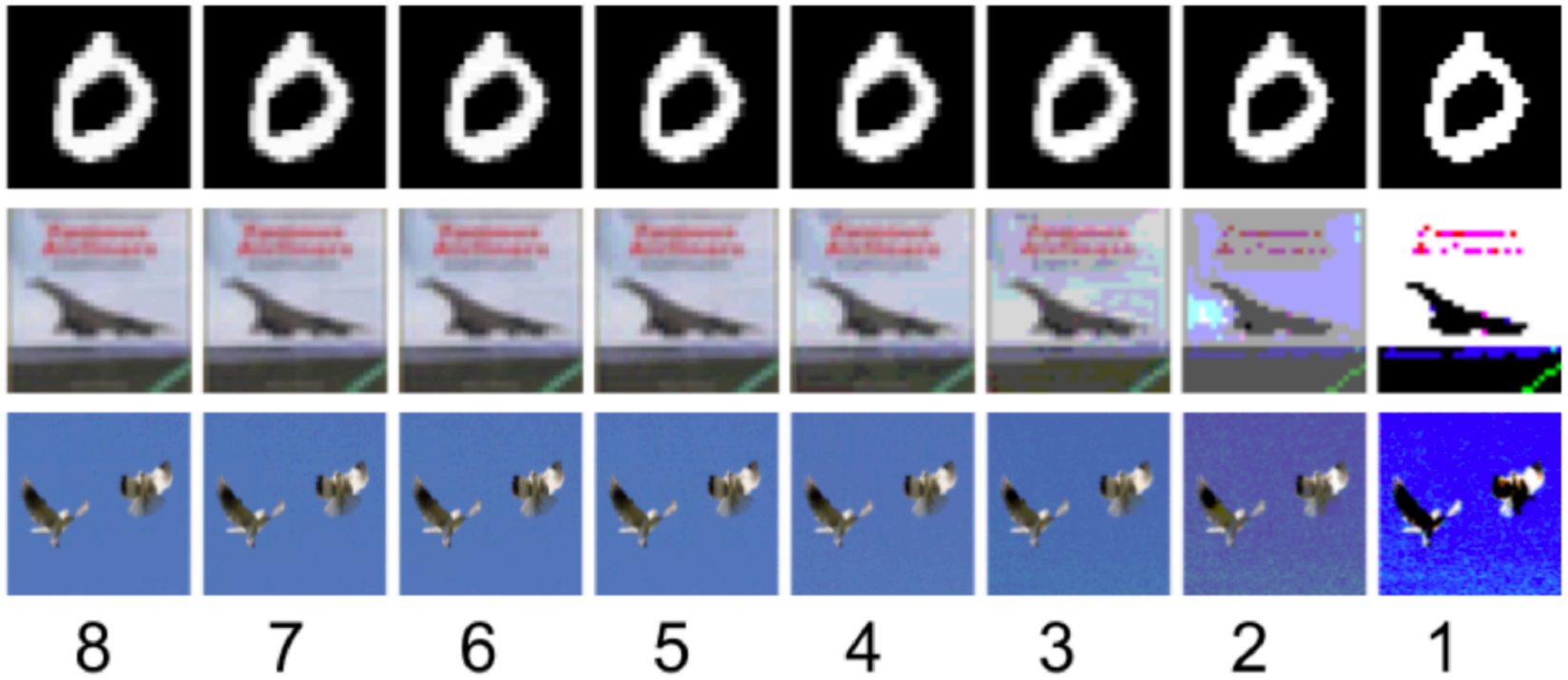


Overview of Feature Squeezing

Generate multiple images (with squeezers) and judge if it is an adversarial sample based on prediction consistencies.

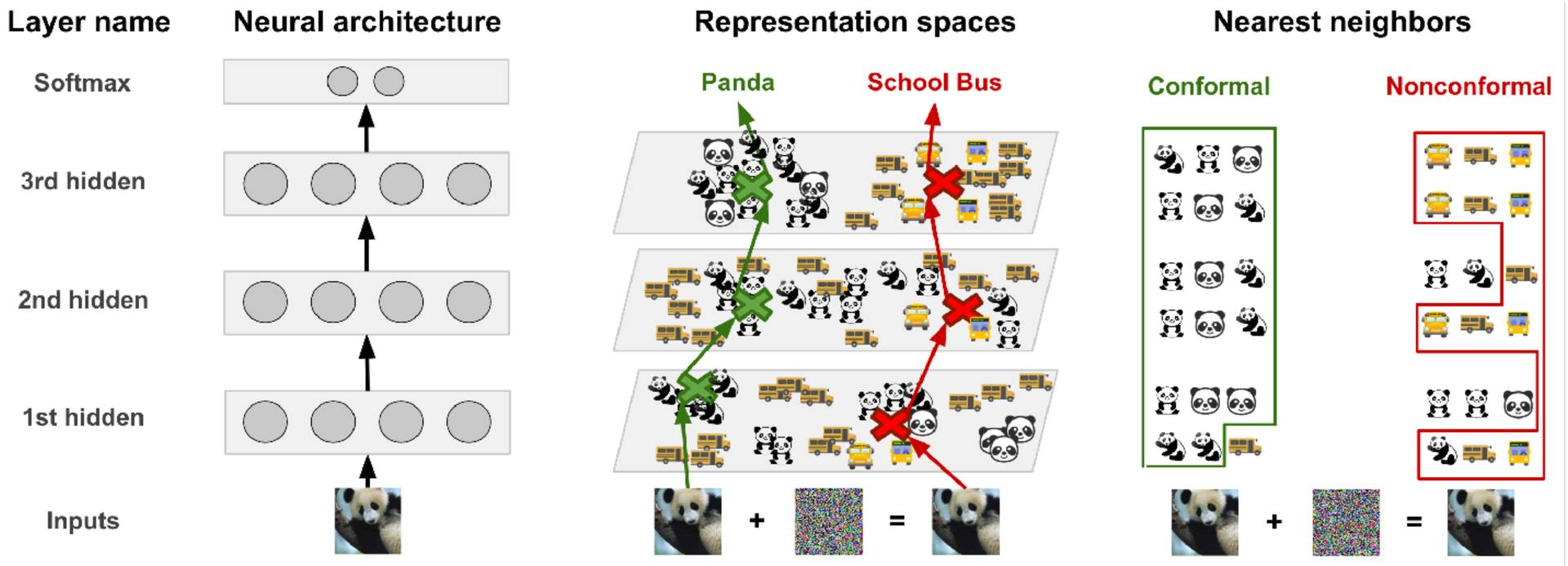
Feature Squeezing

- Defined filters (squeezers): bit depth
- Normal images use 8-bit depth color (from 0 to 255)



Find a Reference

- Basic idea
  - To understand why the model make such a prediction
  - Find similar images from the training dataset as its references
- Related works
  - Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning



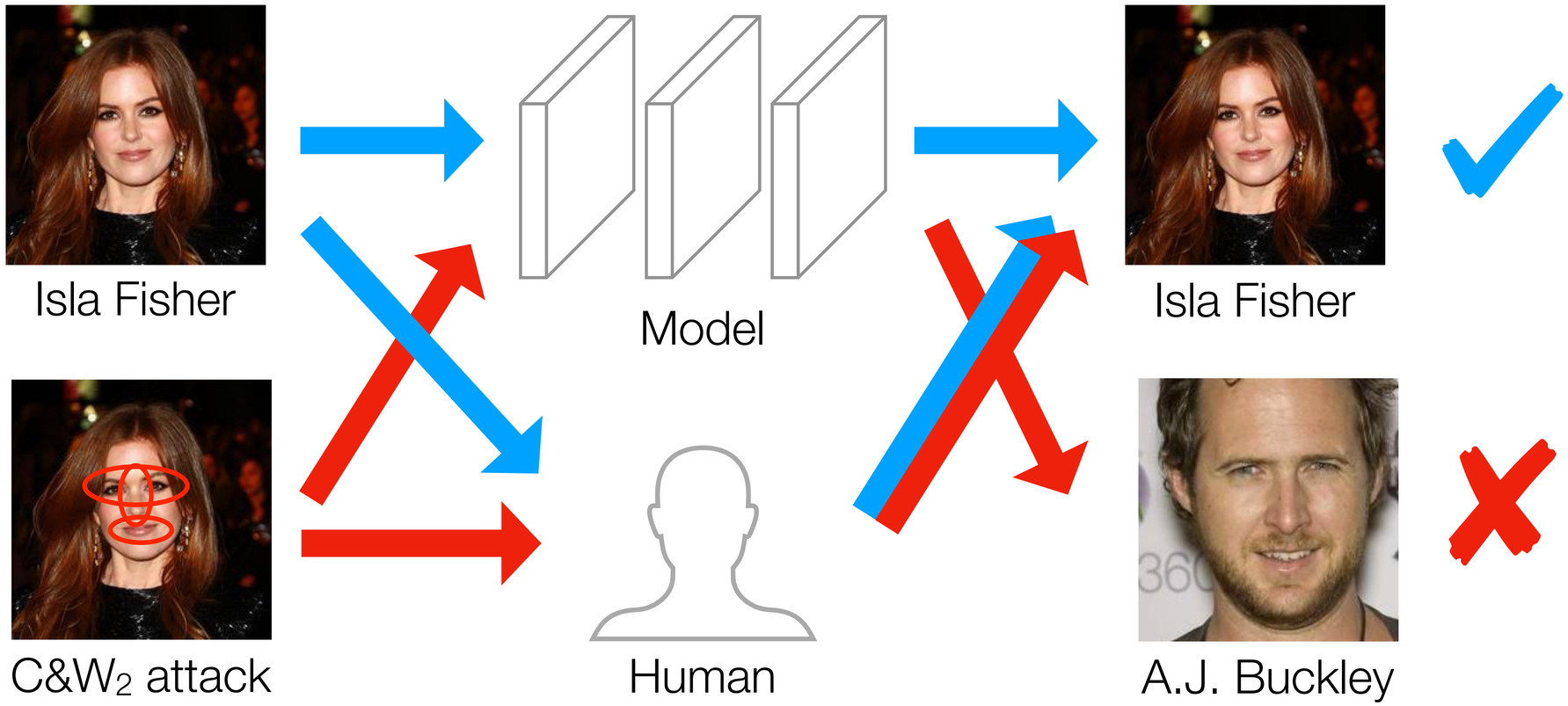
Detecting adversarial samples by using the Deep K-NN

Predictions of benign images are consistent, while adversarial samples are not.

Attacks Meet Interpretability: Attribute-steered Detection of Adversarial Samples

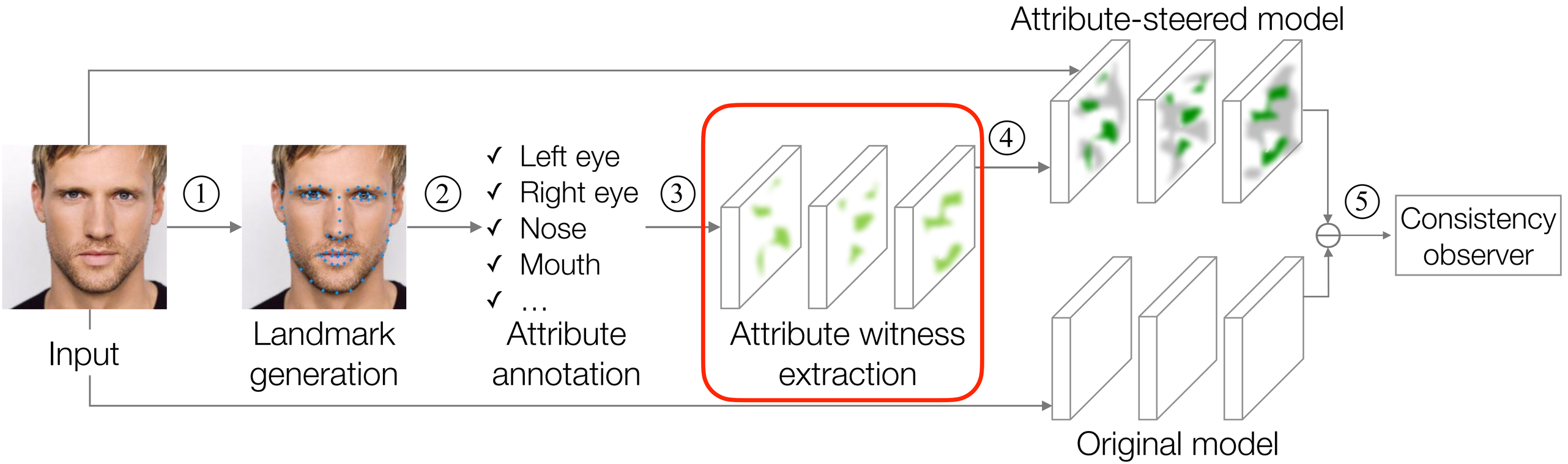
NIPS 2018 Spotlight

Understanding Adversarial Samples



- Idea: is the classification result of a model mainly based on human perceptible attributes?

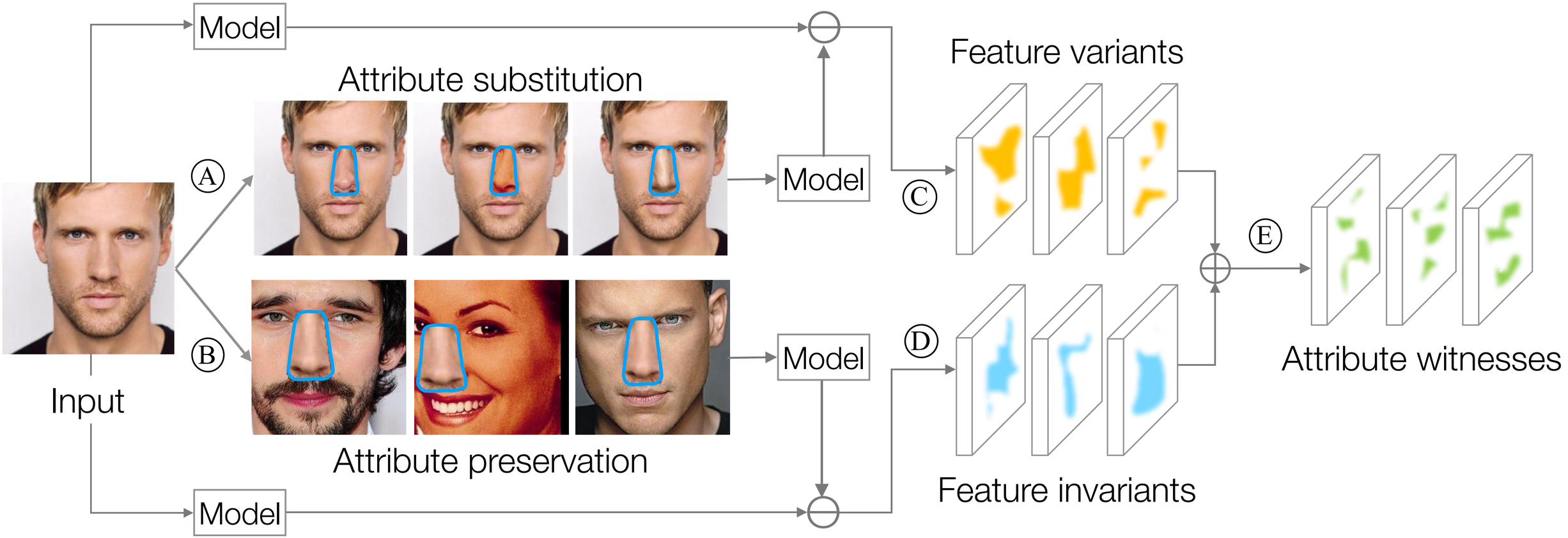
Architecture of AmI



Challenges

- How to extract attribute features learned by neural network models?
- Are there correspondences between attributes and neurons?
- If yes, how to extract corresponding neurons?
- Bi-directional reasoning
  - Forward: attribute changes —> neuron activation changes
  - Backward: neuron activation changes —> attribute changes
  - Backward: no attribute changes —> no neuron activation changes

Attribute Witness Extraction



Attribute-Steered Model

- Constructed by transforming the original model (without additional training)
- Neuron weakening

$$v' = e^{-\frac{v-\mu}{\alpha \cdot \sigma}} \cdot v$$

- Neuron strengthening

$$v' = \epsilon \cdot v + \left(1 - e^{-\frac{v-\min}{\beta \cdot \sigma}}\right) \cdot v$$

v : activation of a non-witness neuron

μ : mean of witness neurons

σ : deviation of witness neurons

α : weakening factor

ϵ, β : strengthening factor

$\min$: minimum of witness neurons

Evaluation

- Model
  - VGG-Face: 16 layers, 97.27% on LFW
- Datasets
  - VGG Face dataset (VF)
  - Labeled Faces in the Wild (LFW)
  - CelebFaces Attributes dataset (CelebA)
- Attacks
  - Patch, Glasses, C&W$_0$, C&W$_2$, C&W$_\infty$, FGSM, BIM

Attribute Witnesses

Layer Name	conv1_1	conv1_2	pool1	conv2_1	conv2_2	pool2	conv3_1	conv3_2	conv3_3	pool3
#Neuron	64	64	64	128	128	128	256	256	256	256
#Left Eye	1	-	-	-	2	3	4	2	3	2
#Right Eye	1	-	-	-	3	3	4	3	2	3
#Nose	1	-	-	-	1	3	2	-	1	3
#Mouth	1	-	-	-	3	2	4	3	15	7
#Shared	1	-	-	-	1	1	1	-	-	-

Layer Name	conv4_1	conv4_2	conv4_3	pool4	conv5_1	conv5_2	conv5_3	pool5	fc6	fc7
#Neuron	512	512	512	512	512	512	512	512	4096	4096
#Left Eye	9	5	15	7	12	4	1	1	-	1
#Right Eye	7	3	10	9	9	1	-	-	-	-
#Nose	10	8	17	13	7	2	2	1	-	1
#Mouth	19	12	12	11	8	2	1	2	1	1
#Shared	1	-	-	-	-	-	-	-	-	-

Attribute Detection

- Predict the presence of the attribute
- Face descriptor: fc7 layer of VGG-Face model

Dataset	VF [19]				LFW [33]			
Attribute	Left Eye	Right Eye	Nose	Mouth	Left Eye	Right Eye	Nose	Mouth
Face Descriptor	0.830	0.830	0.955	0.855	0.825	0.835	0.915	0.935
Attribute Witness	0.940	0.935	0.985	0.990	0.870	0.845	0.975	0.965

Adversary Detection

Detector	FP	Targeted										Untargeted	
		Patch		Glasses		$C\&W_0$		$C\&W_2$		$C\&W_\infty$		FGSM	BIM
		First	Next	First	Next	First	Next	First	Next	First	Next		
FS [18]	23.32%	0.77	0.71	0.73	0.58	0.68	0.65	0.60	0.50	0.42	0.37	0.36	0.20
AS	20.41%	0.96	0.98	0.97	0.97	0.93	0.99	0.99	1.00	0.96	1.00	0.85	0.76
AP	30.61%	0.89	0.96	0.69	0.75	0.96	0.94	0.99	0.97	0.95	0.99	0.87	0.89
WKN	7.87%	0.94	0.97	0.71	0.76	0.83	0.89	0.99	0.97	0.97	0.96	0.86	0.87
STN	2.33%	0.08	0.19	0.16	0.19	0.90	0.94	0.97	1.00	0.76	0.87	0.46	0.41
AmI	9.91%	0.97	0.98	0.85	0.85	0.91	0.95	0.99	0.99	0.97	1.00	0.91	0.90

FP: false positive

First: the first label of classes

Next: the next label of the correct prediction

Adaptive Attacks

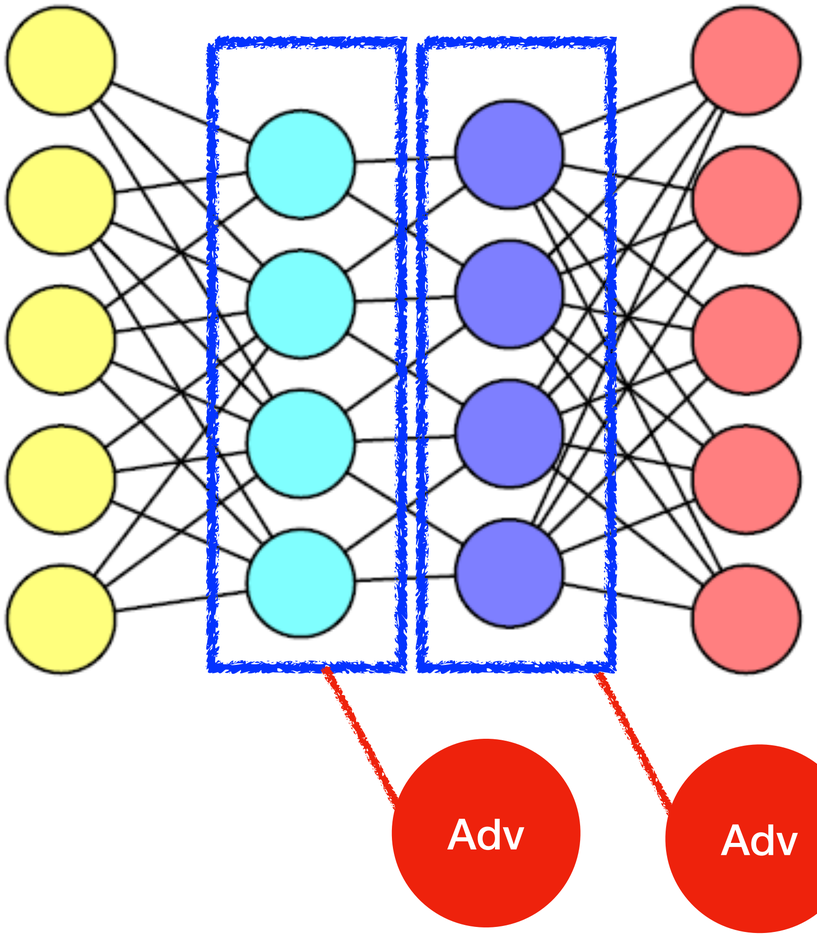
- Adaptive attack: adversary knows the detection
- Idea: view the model and detector as one model, and try to generate adversarial samples to attack it if the detector is differentiable
  - Differentiable detectors, input transformers, squeezers
- Idea: randomly perturb the image to get an sample that works, then optimize on the working sample
  - Non-differentiable squeezers, input transformers

Recall: C&W Attacks

- Nicholas Carlini David Wagner: ***Towards Evaluating the Robustness of Neural Networks***
  - IEEE Symposium on Security and Privacy, 2017, Best Student Paper
  - State-of-the-art and foundations of other adaptive attacks
- Optimization problem in C&W attacks

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

Adaptive Attack



Loss_1

Loss_2

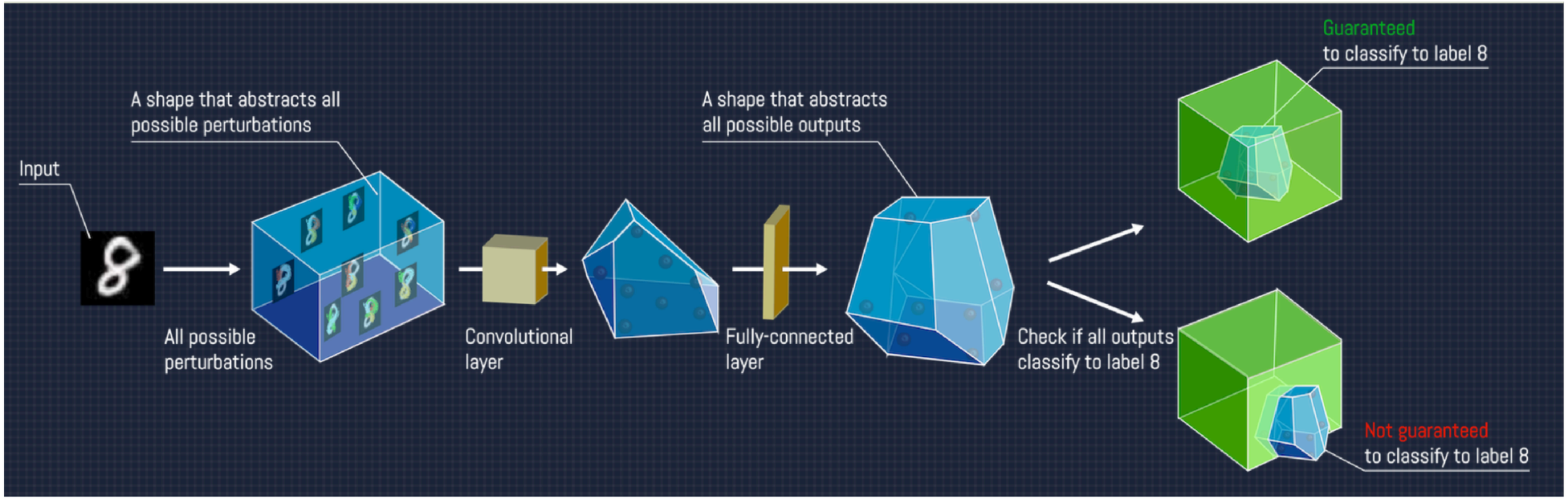
Ensembles

- Ensembles of multiple detectors
  - USENIX WOOT 2017: Adversarial Example Defenses: Ensembles of Weak Defenses are not Strong
- Train or design multiple detectors/squeezers, and randomly choose one or a combination of them to use for detection
  - The adversary has to evade all of them to succeed

Verifiable Nets

- DNNs with properties that can be verified by using solvers
  - Linear programming, abstract domain, interval etc.
- Local-robustness: for a given input and attack strength (measured by Lp norm), the model is robust.
- Existing works can only tiny models and small attack strength
- Most works are sound but not complete
  - If the property is proved, it is true. Otherwise, the property may still hold.

Verifiable Nets



http://safeai.ethz.ch/

More on Adversarial Examples

- Reading list from Nicholas Carlini: https://nicholas.carlini.com/writing/2018/adversarial-machine-learning-reading-list.html

- Tools: Cleverhans, AdvBox, Foolbox, ART, DeepSec

Future Directions

- Do not just optimize the performance measure exactly
  - Existing methods are all using performance measure to indicate the robustness
- Can we perform a lot better with other methods that are similarly indirect?

Future Directions: Better Attack Models

- Add new attack models other than norm balls
- Study messy real problems in addition to clean toy problems
- Study certification methods that use other proof strategies besides local smoothness
- Study more problems other than vision

Future Directions: Security Independent from Traditional Supervised Learning

- Until recently, both adversarial example research and traditional supervised learning seemed fully aligned: just make the model better
- It is now clear security research must have some independent goals. For two models with the same error volume, for reasons of security we prefer:
  - The model with lower confidence on mistakes
  - The model whose mistakes are harder to find
  - A stochastic model that does not repeatedly make the same mistake on the same input
  - A model whose mistakes are less valuable to the attacker / costly to the defender
  - A model that is harder to reverse engineer with probes
  - A model that is less prone to transfer from related models

Other Topics in Security+ML

- Poisoning attacks
  - Poison the training data samples so that the trained model is biased
- Parameter-stealing attack
  - Infer parameters in a black-box scenario
- Membership inference
- Applying ML on various security tasks
  - Tor traffic contribute, malware analysis, graph analysis etc.
- Many many more…

“People worry that computers will get too smart and take over the world, but the real problem is that they're too stupid and they've already taken over the world.”

–Pedro Domingos