CSCI 4260/6260: Data Security & Privacy

Defense Against Adversarial Attacks

Jaewoo Lee
jaewoo.lee@uga.edu

October 25, 2021

Department of Computer Science

Defense on adversarial examples

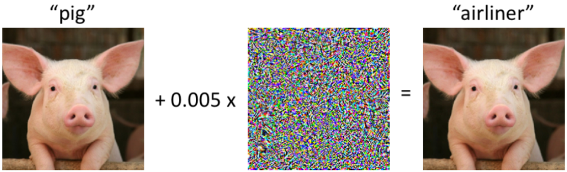
How to defend against adversarial attacks?

1. Pre-processing
2. Adversarial Training
3. Defensive distillation
4. Statistical test + Adversarial Training

Two categories:

1. Model-specific
2. Model-agnostic

Adversarial Examples Detection



- Do adversarial example exhibit *statistical diferences* with the legitimate data?
- Two sample hypothesis testing
  - $D_{\text{adv}} \sim \mathcal{P}$, $D_{\text{train}} \sim \mathcal{Q}$
  - $H_0 : \mathcal{P} = \mathcal{Q}$
  - $H_1 : \mathcal{P} \neq \mathcal{Q}$

Grosse, Kathrin, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel.
On the (Statistical) Detection of Adversarial Examples
ArXiv:1702.06280, 2017

Representation of Images

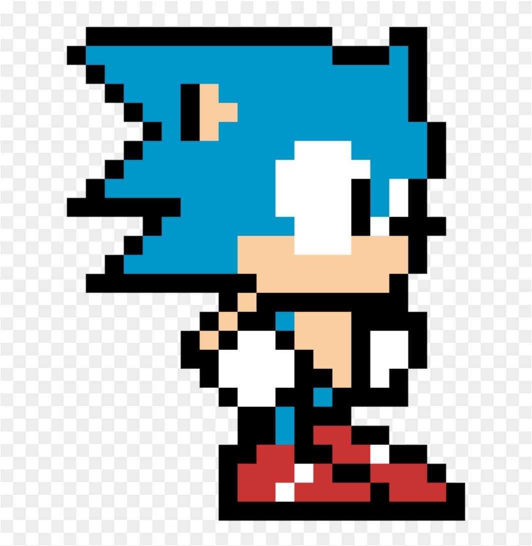


Fig. 1. An image as a collection of pixels

Representation of Images

Fig. 2. More number of pixels gives sharper iamges.

Representation of Images

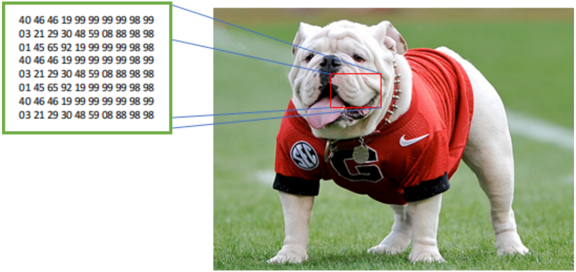


Fig. 3. Image as a matrix

Representation of Images

$x \in \mathbb{R}^{C \times H \times W}$

$x[0] \in \mathbb{R}^{H \times W}$
$x[1] \in \mathbb{R}^{H \times W}$
$x[2] \in \mathbb{R}^{H \times W}$

$x \in \mathbb{R}^{C \cdot H \cdot W}$

Mathematically, we can represent an image as a *vector* $\mathbf{x} = (x_1, \ldots, x_{CHW}) \in \mathbb{R}^{CHW}$.

Representation of Images

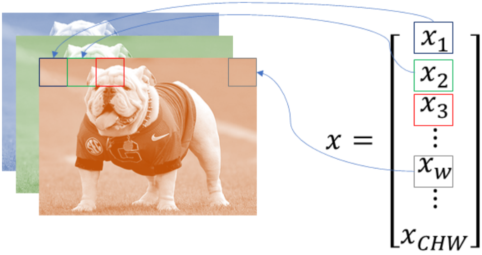


Fig. 4. Image as a vector

Set of Images

Fig. 5. A set of dog images

Modeling Images

X	X_1	X_2	$\cdots$	X_n
$\mathbf{x}^{(1)} =$ 	$x_1^{(1)}$	$x_1^{(1)}$	$\cdots$	$x_n^{(1)}$
$\mathbf{x}^{(2)} =$ 	$x_1^{(2)}$	$x_1^{(2)}$	$\cdots$	$x_n^{(2)}$
$\vdots$			$\cdots$	

What does it mean to learn the *distribution* of images?

$$p(\mathbf{x}) = \mathbb{P}[X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n]$$
$$p(\mathbf{x}|Y = \text{dog}) = \mathbb{P}[X_1 = x_1, \ldots, X_n = x_n \mid Y = \text{dog}]$$

Hypothesis Testing

1. Compute a test statistic T

$$T(\mathcal{P}, \mathcal{Q}) = \mathrm{MMD}(\mathcal{F}, X_1, X_2) = \sup_{f \in \mathcal{F}} \left(\frac{1}{n} \sum_{i=1}^{n} f(x_{1i}) - \frac{1}{m} \sum_{i=1}^{m} f(x_{2i}) \right)$$

2. Compute the p-value
3. If the p-value is smaller than α, reject the null.

Detecting Adversarial Examples

- Hypothesis testing can only detect a *group* of adversarial examples.
- Requires large batch of adversarial inputs
  - An idea is to *augment* the training data with adversarial example.
  - Train a classifier on the augmented dataset.

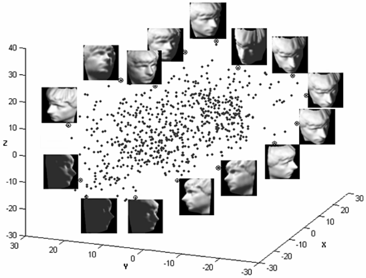
X	Y
x_1	c_1
x_2	c_3
x_3	c_2
$\vdots$	$\vdots$
x_n	c_K
x_1'	c_{out}
x_2'	c_{out}
$\vdots$	$\vdots$
x_m'	c_{out}

- $D_{\text{train}} = D_{\text{train}} \cup D_{\text{adv}}$
- Suppose $\mathcal{Y} = \{c_1, c_2, \ldots, c_K\}$
- x_i: regular training examples
- x_i': adversarial examples
- c_{out}: class label assigned to adversarial examples

Pre-processing

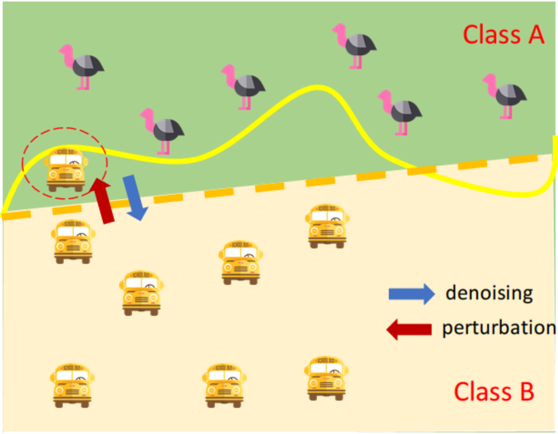
- Recall that adversarial examples are created by adding *noise*.
- Can we try removing noise?
  - Let $g(x)$ be a denoising algorithm.
  - Let $f(x)$ be a classifier.
  - $(f \circ g)(x)$

- Image manifold: not all matrices are natural images.
- project the perturbed image x to the manifold



Guo, Chuan, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering Adversarial Images Using Input Transformations
ArXiv:1711.00117 January 25, 2018. http://arxiv.org/abs/1711.00117

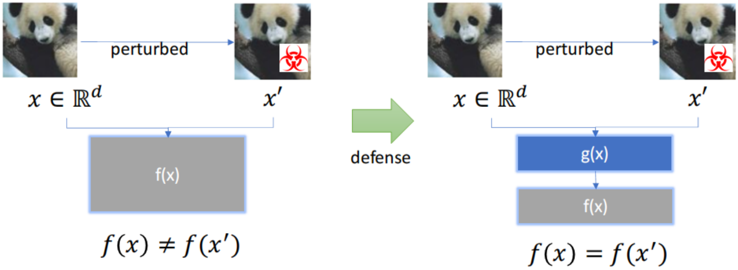
Pre-processing



Defense Goal

- *remove* adversarial perturbation
- maintain sufficent information in input images
- not relying on the *secrecy* of defensive mechanism

Setup



- $x \in \mathbb{R}^d$: original image, x': adversarial image (perturbed)
- $f : \mathbb{R}^d \to \mathcal{Y}$: a classifier
- $g : \mathbb{R}^d \to \mathbb{R}^d$: a transformation algorithm
- Adversary's knowledge: *black-box* vs *gray-box*

Can we develop g such that $f(g(x)) = f(g(x'))$?

Black-box VS Gray-box

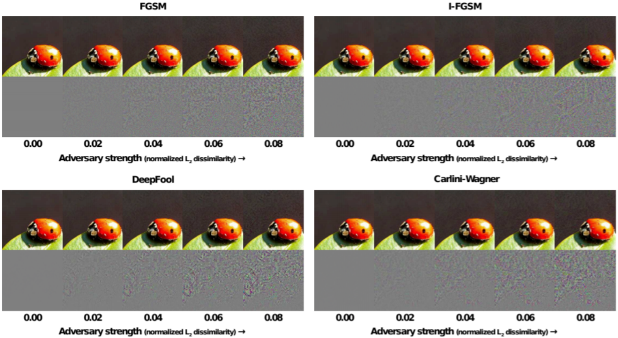
1. Black-box
   - Adversary does *not* have direct asscess to the model $f(\cdot)$.

2. White-box
   - Adversary has full knowledge on the model f.
   - model architecture, model parameter, defense strategy

3. Gray-box
   - Somewhere between black-box and white-box
   - partial knowledge
   - transferring adversarial examples

Image Transformations

Adversarial attacks change particular statistics of the input image.



- Image cropping-rescaling
- Bit-depth reduction: quantization to remove small variations
- JPEG compression and decompression: removes small perturbations

Total Variation Minimization

Main idea for defense

1. Select a subset of pixels that carry important information.
2. Reconstruct the image from the chosen pixels

Construct an image $\mathbf{z}$ such that

- $\mathbf{z}$ is similar to input image $\mathbf{x}$ and
- *simple* in terms of $\mathrm{TV}_p(\mathbf{z})$.

$$\min_{\mathbf{z}} \|(1 - X) \odot (\mathbf{z} - \mathbf{x})\|_2 + \lambda \mathrm{TV}_p(\mathbf{z})$$

Total Variation Minimization: Random Variable X

$$\min_{\mathbf{z}} \|(1 - X) \odot (\mathbf{z} - \mathbf{x})\|_2 + \lambda \mathrm{TV}(\mathbf{z})$$

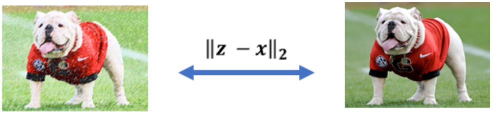
- $X(i,j,k)$ is a Bernoulli random variable.
- For each pixel at (i,j,k), flip a coin ($p = \mathbb{P}[\text{Head}]$).

$$X(i,j,k) = \begin{cases} 1 & \text{if head,} \\ 0 & \text{if tail.} \end{cases}$$

Total Variation Minimization: L_2-distance

$$\min_{\mathbf{z}} \|(1-X)\odot(\mathbf{z}-\mathbf{x})\|_2 + \lambda \mathrm{TV}(\mathbf{z})$$

- $\|\mathbf{z}-\mathbf{x}\|$ needs to be small!



Reconstructed image $\mathbf{z}$ (perturbed) image $\mathbf{x}$

Total Variation Minimization: TV distance

$$\min_{\mathbf{z}} \|(1 - X) \odot (\mathbf{z} - \mathbf{x})\|_2 + \lambda \mathrm{TV}(\mathbf{z})$$

- The last term chooses one with smaller *Total Variation*.
- Total variation of $\mathbf{z}$ is defined by

$$\mathrm{TV}(\mathbf{z}) = \underbrace{\sum_{k=1}^{K}}_{\text{over channels}} \left[\sum_{i=2}^{N} \underbrace{\|\mathbf{z}[i,:,k] - \mathbf{z}[i-1,:,k]\|_2}_{\text{row-wise similarity}} + \sum_{j=2}^{N} \underbrace{\|\mathbf{z}[:,j,k] - \mathbf{z}[:,j-1,k]\|_2}_{\text{column-wise similarity}} \right]$$

  - $\mathbf{z}$ is $N \times N \times K$ image.
  - Measures the amount of fine-scale variation
  - Encourages the remove of small (adversarial) perturbations

Total Variation Minimization: Result

$$\min_{\mathbf{z}} \|(1 - X) \odot (\mathbf{z} - \mathbf{x})\|_2 + \lambda \mathrm{TV}(\mathbf{z})$$

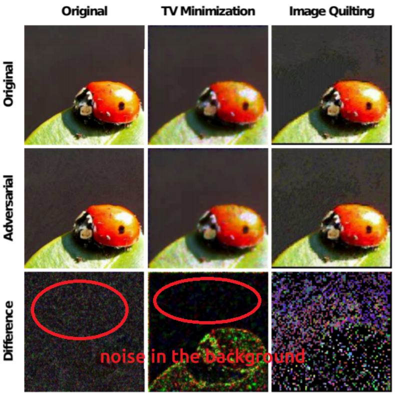
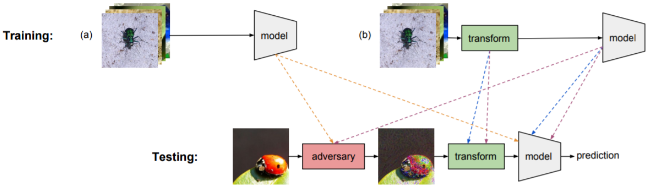


Image Transformation: Training



- Let your classifier know the input is transformed.

Adversarial Training

Main idea

- Adversarial examples are $\mathbf{x}' = \mathbf{x} + \mathbf{h}$, $\|\mathbf{h}\| < \tau$.
- Given a classifier f, for adversarial examples $\mathbf{x}'$, $f(\mathbf{x}) \neq f(\mathbf{x}')$.
- My model f behaves differently from my expectation!
  - Is there a way we can tell f that you're doing it wrong?
  - Recall the main idea of *supervised* learning.
  - The ground truth label in the data: $\{(x_i, y_i)\}_{i=1}^n$

Self-Driving Car Example

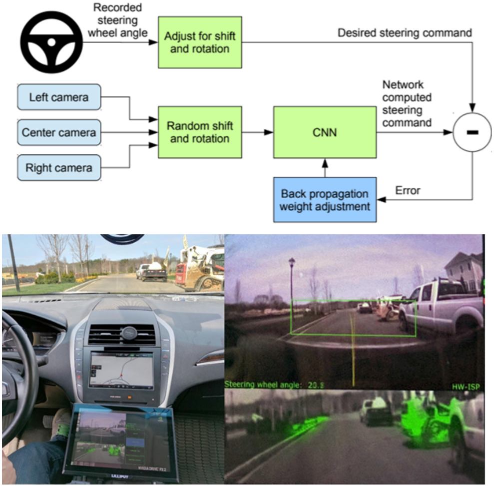
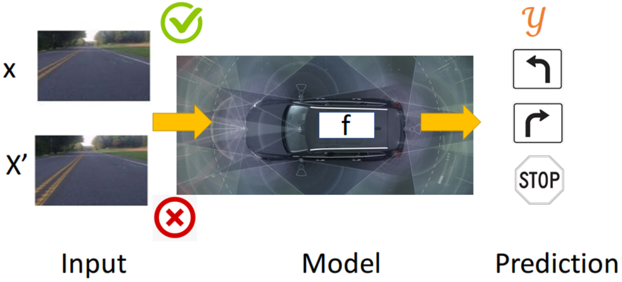


Fig. 6. NVIDIA's DAVE-2 System

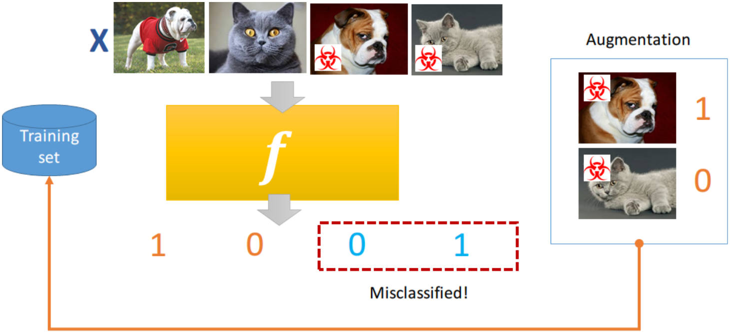
- NN f needs to learn actions.
  - $f : \mathcal{X} \to \mathcal{Y}$,
    $\mathcal{Y} = \{\text{left}, \text{right}\}$
- Input: images from sensors
- A human can *annotate* the images (ground truth).
- Training data (X_i, Y_i)
- The human labeled data is insufficient.

Self-driving Car



- Fix the wrong behavior by correcting it.
- Suppose $f(x) =$`left`, when the correct action for $\mathbf{x}$ is `right`.
- Insert (x, `right`) into the training set.
- Retrain f

Adversarial Training



- Correcting wrong decisions by augmenting the data
- But do we really need to *retrain*?
  - training is time-consuming
  - re-training might be *expensive*

Adversarial Training: Setup

- Adversarial example aware training
  - How to model the adversary?
  - Consider a set of allowed perturbations (attacks) $\mathcal{S} \subseteq \mathbb{R}^d$ on my data.
    - $\delta \in \mathcal{S}$

- Supervised learning

$$\min_{\theta \in \Theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f,(x,y))]$$

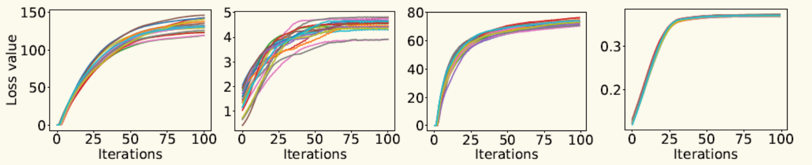
  - $\mathcal{D}$: population distribution
  - $(x,y) \sim \mathcal{D}$: (random) data drawn from the population distribution
  - $\ell(f,(x,y))$: loss of f on the data (x,y)
  - $\mathbb{E}[\ell(f,(x,y))]$: expected loss on random example

Adversarial Training

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

- allowed perturbations $\|\boldsymbol{\delta}\|_\infty < \epsilon$
- minimizing the worst case loss
- Robust optimization
- This type of loss is called an *adversarial* loss.
- Inner maximization: attack
- Outer minimization: defense

Adversarial Training: Result



(a) MNIST Standard training (b) MNIST Adversarial training (c) CIFAR10 Natural training (d) CIFAR10 Adversarial training

Adversarial Training

- We maximize over $\mathcal{S}$.
  - need to generate many $\delta \in \mathcal{S}$
  - how to generate δ?
  - how large $\mathcal{S}$ should be?
- Scalaiblity
  - Costly retraining

Photo Forensics

Detecting the manipulated images

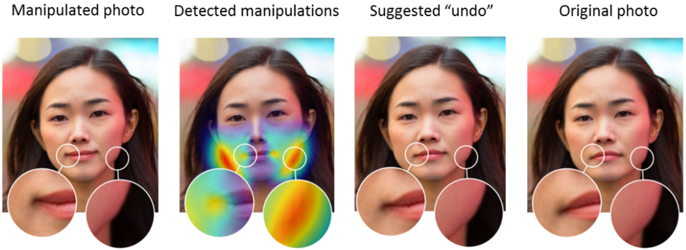
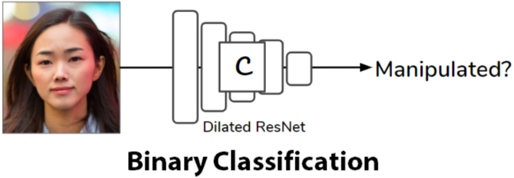


Fig. 7. Image source: Wang et al. 2019

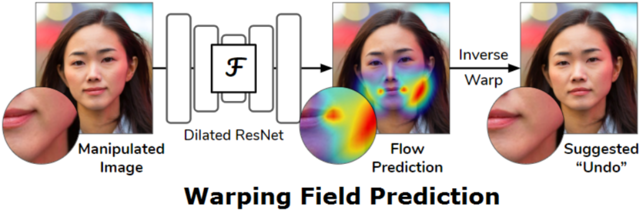
- Has the image manipulated?

Wang, Sheng-Yu, Oliver Wang, Andrew Owens, Richard Zhang, and Alexei A. Efros.
Detecting Photoshopped Faces by Scripting Photoshop
ICCV 2019



- Create a supervised dataset $\{(x, y)\}$
- Original image $(x, 0)$
- Manipulate x, $(x, 1)$
- Train $f : \mathcal{X} \to \{0, 1\}$

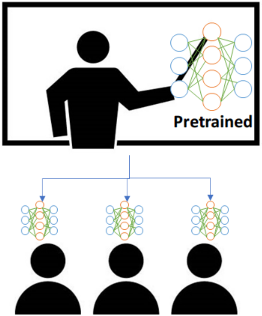
How is it manipulated?



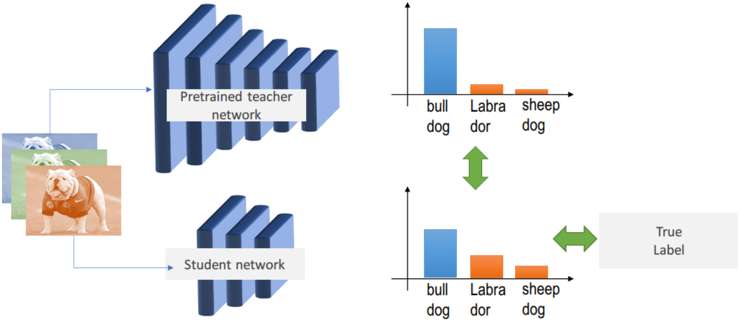
- Which *pixel* is modified?
- Can we recover the original image before the modification?
- Train f to predict per-pixel warping

Knowledge Distillation

- A model *compression* technique
- Can we *compress* the knowledge of *large complex* model into a *small and simple* model?
  - A large and complex model: teacher
  - A small and simple model: student



Knowledge Distillation



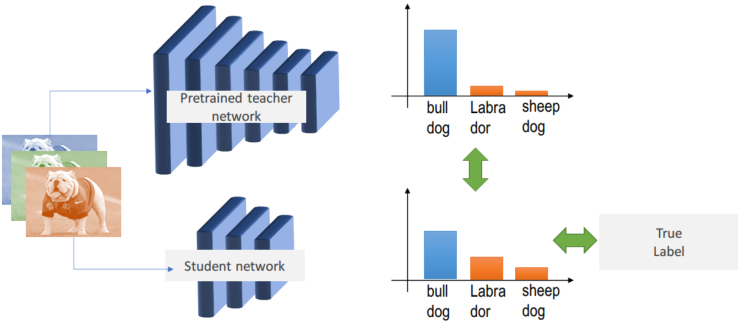
- cross-entropy loss with correct labels

$$\mathcal{L}_{\text{cl}} = \frac{1}{m}\sum_{i=1}^{m} \text{cross-entropy}(f(\mathbf{x}_i), \underbrace{\mathbf{y}_i}_{\text{true label}})$$

- cross-entropy loss with teacher's prediction

$$\mathcal{L}_{\text{teacher}} = \frac{1}{m}\sum_{i=1}^{m} \text{cross-entropy}(f(\mathbf{x}_i), g(\mathbf{x}_i))$$

Knowledge Distillation



- Total loss

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cl}} + \lambda \mathcal{L}_{\text{teacher}}$$

- Problem: what if $\mathbb{P}[\text{bulldog}] \approx 1$ and $\mathbb{P}[\text{others}] \approx 0$?
  - Not much different from $\mathbf{y} = (1, 0, 0, 0, 0)$

Knowledge Distillation

- Solution
  - match the smoothed version of probability
  - Temperature

$$p_i = \textsf{softmax}_T(\mathbf{z}) = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)}$$

Softmax with Temperature

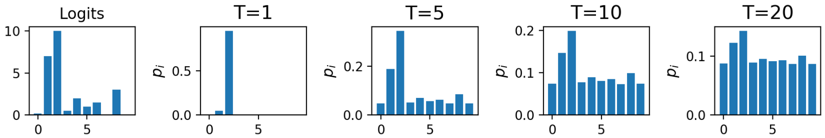
```python
import numpy as np
import matplotlib.pyplot as plt

z = np.array([0.2, 7, 10, 0.5, 2, 1, 1.5, 0.01, 3, 0.1 ])
x = np.arange(len(z))
Temperatures = [1, 5, 10, 20]

fig, ax = plt.subplots(1, len(Temperatures)+1, figsize=(7, 1.3))
ax[0].bar(x, z)
ax[0].set_title('Logits')

for i, T in enumerate(Temperatures):
    p = np.exp(z/T)/np.sum(np.exp(z/T))

    ax[i+1].bar(x, p)
    ax[i+1].set_title('T={}'.format(T), size=15)
    ax[i+1].set_ylabel(r'$p_i$', size=12)
plt.tight_layout(pad=0.1)
plt.show()
```

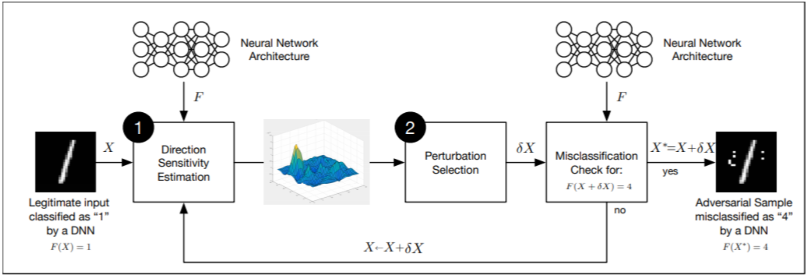


Distillation as a Defense

Can we use the idea of *knowledge distillation* to defend the NN?

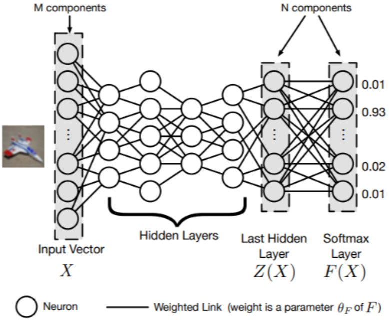
- Adversarial examples created by the adversary
  - $X' = X + h$ with $f(X') \neq f(X)$
  - direction sensitivity estimation
  - perturbation selection

$$\arg\min_h \|h\| \qquad \text{s.t. } f(x+h) \neq f(x)$$



Distillation for Defense

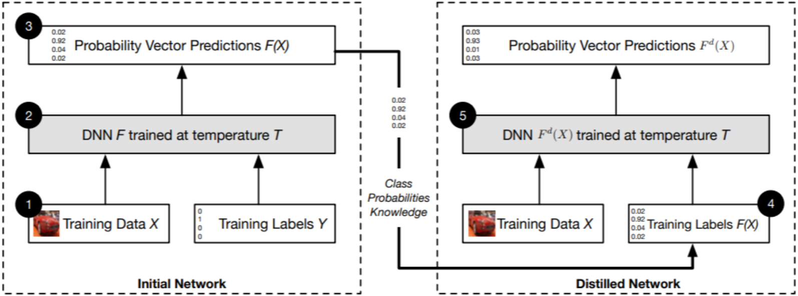
- First, train a network F with a *softmax* layer.



$$F(X) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}, \qquad \text{for } j = 1, \ldots, K$$

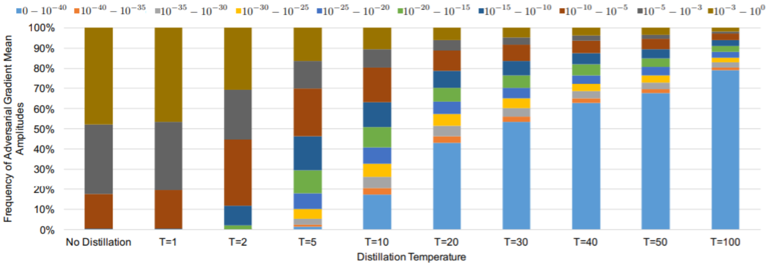
- T is the *temperature* parmaeter, $T > 1$.
  - At high temperature, $F(X) \to 1/K$ as $T \to \infty$
- Use $F(X)$ as *soft labels* for the second (smaller) network

Distillation for Defense



- F: source network
- F^d: distilled network
- Unlike the original distillation, F and F^d have the same architecture.

Impact of temperature



- At a higher temperature, *adversarial* gradient becomes smaller.
- Small gradient $\Rightarrow$ difficult to craft the example