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 differences* with the legitimate data?

**Two sample hypothesis testing**

- $D_{adv} \sim P, D_{train} \sim Q$
- $H_0 : P = Q$
- $H_1 : P \neq Q$

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 images.
Representation of Images

Fig. 3. Image as a matrix
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
Fig. 5. A set of dog images
Modeling Images

<table>
<thead>
<tr>
<th>$X$</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$\cdots$</th>
<th>$X_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^{(1)}$</td>
<td>$x^{(1)}_1$</td>
<td>$x^{(1)}_2$</td>
<td>$\cdots$</td>
<td>$x^{(1)}_n$</td>
</tr>
<tr>
<td>$x^{(2)}$</td>
<td>$x^{(2)}_1$</td>
<td>$x^{(2)}_2$</td>
<td>$\cdots$</td>
<td>$x^{(2)}_n$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\cdots$</td>
<td>$\vdots$</td>
</tr>
</tbody>
</table>

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

$$p(x) = \mathbb{P}[X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n]$$

$$p(x | Y = \text{dog}) = \mathbb{P}[X_1 = x_1, \ldots, X_n = x_n | Y = \text{dog}]$$
Hypothesis Testing

1. Compute a test statistic $T$

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

2. Compute the $p$-value

3. If the $p$-value is smaller than $\alpha$, 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.

<table>
<thead>
<tr>
<th>$X$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$c_1$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$c_3$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$c_2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x_n$</td>
<td>$c_K$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$X'$</th>
<th>$Y'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x'_1$</td>
<td>$c_{out}$</td>
</tr>
<tr>
<td>$x'_2$</td>
<td>$c_{out}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x'_m$</td>
<td>$c_{out}$</td>
</tr>
</tbody>
</table>

- $D_{train} = D_{train} \cup D_{adv}$
- 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

---

Pre-processing
Defense Goal

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

- $x \in \mathbb{R}^d$: original image, $x'$: adversarial image (perturbed)
- $f: \mathbb{R}^d \rightarrow \mathcal{Y}$: a classifier
- $g: \mathbb{R}^d \rightarrow \mathbb{R}^d$: 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 access 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 $z$ such that

- $z$ is similar to input image $x$ and
- *simple* in terms of $TV_p(z)$.

$$\min_{z} \| (1 - X) \odot (z - x) \|_2 + \lambda TV_p(z)$$
Total Variation Minimization: Random Variable $X$

\[
\min_z \| (1 - X) \odot (z - x) \|_2 + \lambda \text{TV}(z)
\]

- $X(i, j, k)$ 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

$$\min_z \|(1 - X) \odot (z - x)\|_2 + \lambda TV(z)$$

- $\|z - x\|$ needs to be small!
Total Variation Minimization: TV distance

$$\min_z \|(1 - X) \odot (z - x)\|_2 + \lambda TV(z)$$

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

$$TV(z) = \sum_{k=1}^{K} \left[ \sum_{i=2}^{N} \|z[i, :, k] - z[i - 1, :, k]\|_2 \right]$$

row-wise similarity

over channels

$$+ \sum_{j=2}^{N} \|z[:, j, k] - z[:, j - 1, k]\|_2$$

column-wise similarity

- $z$ 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

$$\min_z \| (1 - X) \odot (z - x) \|_2 + \lambda TV(z)$$
Image Transformation: Training

- Let your classifier know the input is transformed.
Adversarial Training

Main idea

• Adversarial examples are $x' = x + h$, $\|h\| < \tau$.
• Given a classifier $f$, for adversarial examples $x'$, $f(x) \neq f(x')$.
• 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

- 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.

Fig. 6. NVIDIA’s DAVE-2 System
• Fix the wrong behavior by correcting it.
• Suppose $f(x) = \text{left}$, when the correct action for $x$ is $\text{right}$.
• Insert $(x, \text{right})$ 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) $S \subseteq \mathbb{R}^d$ on my data.
    - $\delta \in S$

- Supervised learning
  \[
  \min_{\theta \in \Theta} \mathbb{E}_{(x,y) \sim D}[\ell(f, (x, y))]
  \]
  - $D$: population distribution
  - $(x, y) \sim D$: (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

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

- allowed perturbations \(\|\delta\|_{\infty} < \epsilon\)
- minimizing the worst case loss
- Robust optimization
- This type of loss is called an \textit{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 $S$.
  ○ need to generate many $\delta \in S$
  ○ how to generate $\delta$?
  ○ how large $S$ should be?

• Scalability
  ○ Costly retraining
Photo Forensics

Detecting the manipulated images

- Has the image manipulated?

ICCV 2019
Real or Fake?

- Create a supervised dataset \{ (x, y) \}
- Original image \((x, 0)\)
- Manipulate \(x\), \((x, 1)\)
- Train \(f : \mathcal{X} \rightarrow \{0, 1\}\)
How is it manipulated?

- Which \textit{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

\[ \mathcal{L}_{cl} = \frac{1}{m} \sum_{i=1}^{m} \text{cross-entropy}(f(x_i), y_i) \]

- cross-entropy loss with teacher’s prediction

\[ \mathcal{L}_{teacher} = \frac{1}{m} \sum_{i=1}^{m} \text{cross-entropy}(f(x_i), g(x_i)) \]
Knowledge Distillation

- **Total loss**

\[ L_{\text{total}} = L_{\text{cl}} + \lambda L_{\text{teacher}} \]

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

- Solution
  - match the smoothed version of probability
  - Temperature

\[ p_i = \text{softmax}_T(z) = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)} \]
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

\[
\arg\min_h \|h\| \quad \text{s.t.} \quad f(x + h) \neq f(x)
\]
Distillation for Defense

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

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

- $T$ is the temperature parameter, $T > 1$.
  - 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 ⇒ difficult to craft the example