Attack and Defense

Hung-yi Lee

Source of image: http://www.fafa01.com/post865806
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.
• We want the classifiers that are robust the inputs that are built to fool the classifier.
• Especially useful for spam classification, malware detection, network intrusion detection, etc.
Attack

What do we want to do?

Original Image

\[
x^0
\]

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

Attacked Image

\[x' = x^0 + \Delta x\]

Network

Something Else

Tiger Cat

0.64
Loss Function for Attack

- **Training:** $L_{\text{train}}(\theta) = C(y^0, y^{\text{true}})$ \hspace{1cm} $x$ fixed
- **Non-targeted Attack:** $L(x') = -C(y', y^{\text{true}})$ \hspace{1cm} $\theta$ fixed
- **Targeted Attack:**
  \[ L(x') = -C(y', y^{\text{true}}) + C(y', y^{\text{false}}) \]
- **Constraint:** $d(x^0, x') \leq \varepsilon$
Constraint \[ d(x^0, x') \leq \varepsilon \]

- **L2-norm**
  \[
d(x^0, x') = \| x^0 - x' \|_2 = (\Delta x_1)^2 + (\Delta x_2)^2 + (\Delta x_3)^2 \ldots
\]

- **L-infinity**
  \[
d(x^0, x') = \| x^0 - x' \|_{\infty} = \max \{ \Delta x_1, \Delta x_2, \Delta x_3, \ldots \}
\]
How to Attack

• Gradient Descent

Just like training a neural network, but network parameter $\theta$ is replaced with input $x'$

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

Start from original image $x^0$
For $t = 1$ to $T$

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

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

\[
x^t \leftarrow \text{fix}(x^t)
\]

\[
\nabla L(x) = \begin{bmatrix}
\frac{\partial L(x)}{\partial x_1} \\
\frac{\partial L(x)}{\partial x_2} \\
\vdots \\
\frac{\partial L(x)}{\partial x_3}
\end{bmatrix}
\]
How to Attack

Just like training a neural network, but network parameter $\theta$ is replaced with input $x'$

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

• 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

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

Return the one closest to $x^t$
How to Attack

\[
\text{def } \textit{fix}(x^t)
\]

For all \( x \) fulfill
\[
d(x^0, x) \leq \varepsilon
\]
Return the one closest to \( x^t \)

L2-norm

L-infinity
Example

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

True = Tiger cat
False = Star Fish

\[ f = \text{ResNet-50} \]
Example

- **Original Image**
- **Attacked Image**

Tiger Cat: 0.64

Star Fish: 1.00
$L(x') = -C(y', y^{true}) + C(y', y^{false})$

**Example**

True = Tiger cat  
False = Keyboard

$f = \text{ResNet-50}$

**Original Image**

**Attacked Image**

Tiger Cat  
0.64

Keyboard  
0.98
tiger cat

Persian cat

tabby cat

fire screen
What happened?

\[ \gamma_{\text{Egyptian cat}} \quad \gamma_{\text{tiger cat}} \quad \gamma_{\text{Persian cat}} \]

Random

\[ x^0 \]

Specific Direction

\[ y_{\text{tiger cat}} \quad y_{\text{key board}} \]

\[ x^0 \]
Attack Approaches

- ...... only list a few
Attack Approaches

\[ x^* = \arg \min_{d(x^0, x') \leq \varepsilon} L(x') \]

Different optimization methods

Different constraints

- Fast Gradient Sign Method (FGSM)

\[ x^* \leftarrow x^0 - \varepsilon \Delta x \]

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

only have +1 or -1
Attack Approaches

\[ x^* = \arg\min_{d(x^0, x') \leq \varepsilon} \ L(x') \]

Different optimization methods

Different constraints

• Fast Gradient Sign Method (FGSM)

\[ x^* \leftarrow x^0 - \varepsilon \Delta x \]

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

only have +1 or -1

\( \varepsilon \)

\( L\text{-infinity} \)
Attack Approaches

\[ x^* = \underset{\text{min}}{\arg} \min_{d(x^0, x') \leq \epsilon} L(x') \]

- Fast Gradient Sign Method (FGSM)

\[ x^* \leftarrow x^0 - \epsilon \Delta x \]

\[ \Delta x = \begin{bmatrix} \text{sign}(\partial L/\partial x_1) \\ \text{sign}(\partial L/\partial x_2) \\ \vdots \\ \text{sign}(\partial L/\partial x_n) \end{bmatrix} \]

only have +1 or -1

Different optimization methods

Different constraints

very large learning rate
White Box v.s. Black Box

• In the previous attack, we fix network parameters $\theta$ to find optimal $x'$.  
• To attack, we need to know network parameters $\theta$
  • 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

<table>
<thead>
<tr>
<th></th>
<th>ResNet-152</th>
<th>ResNet-101</th>
<th>ResNet-50</th>
<th>VGG-16</th>
<th>GoogLeNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-152</td>
<td>4%</td>
<td>13%</td>
<td>13%</td>
<td>20%</td>
<td>12%</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>19%</td>
<td>4%</td>
<td>11%</td>
<td>23%</td>
<td>13%</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>25%</td>
<td>19%</td>
<td>5%</td>
<td>25%</td>
<td>14%</td>
</tr>
<tr>
<td>VGG-16</td>
<td>20%</td>
<td>16%</td>
<td>15%</td>
<td>1%</td>
<td>7%</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>25%</td>
<td>25%</td>
<td>17%</td>
<td>19%</td>
<td>1%</td>
</tr>
</tbody>
</table>

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_uMen0Bck&feature=youtu.be
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.

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.

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/GlsW1C); (c) $S_B$ impersonating $S_C$; and (d) $S_C$ impersonating Carson Daly (by Anthony Quintano / CC BY / cropped from https://goo.gl/VfnDct).
<table>
<thead>
<tr>
<th>Distance/Angle</th>
<th>Subtle Poster</th>
<th>Subtle Poster Right Turn</th>
<th>Camouflage Graffiti (LISA-CNN)</th>
<th>Camouflage Art (GTSRB-CNN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5' 0°</td>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
<td><img src="image3" alt="Image" /></td>
<td><img src="image4" alt="Image" /></td>
</tr>
<tr>
<td>5' 15°</td>
<td><img src="image5" alt="Image" /></td>
<td><img src="image6" alt="Image" /></td>
<td><img src="image7" alt="Image" /></td>
<td><img src="image8" alt="Image" /></td>
</tr>
<tr>
<td>10' 0°</td>
<td><img src="image9" alt="Image" /></td>
<td><img src="image10" alt="Image" /></td>
<td><img src="image11" alt="Image" /></td>
<td><img src="image12" alt="Image" /></td>
</tr>
<tr>
<td>10' 30°</td>
<td><img src="image13" alt="Image" /></td>
<td><img src="image14" alt="Image" /></td>
<td><img src="image15" alt="Image" /></td>
<td><img src="image16" alt="Image" /></td>
</tr>
<tr>
<td>40' 0°</td>
<td><img src="image17" alt="Image" /></td>
<td><img src="image18" alt="Image" /></td>
<td><img src="image19" alt="Image" /></td>
<td><img src="image20" alt="Image" /></td>
</tr>
</tbody>
</table>

Targeted-Attack Success

<table>
<thead>
<tr>
<th>Subtle Poster</th>
<th>Subtle Poster Right Turn</th>
<th>Camouflage Graffiti (LISA-CNN)</th>
<th>Camouflage Art (GTSRB-CNN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>73.33%</td>
<td>66.67%</td>
<td>100%</td>
</tr>
</tbody>
</table>

[hyperlink](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

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
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

Original + Filter + Network

Do not influence classification

Tiger Cat

Attack signal → e.g. Smoothing → Less harmful
Passive Defense

- Feature Squeeze

https://arxiv.org/abs/1704.01155
Randomization at Inference Phase

https://arxiv.org/abs/1711.01991
Proactive Defense

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

Using $X$ to train your model

For $t = 1$ to $T$

For $n = 1$ to $N$

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

We have new training data different in each iteration

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

Using both $X'$ to update your model

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

精神：找出漏洞、補起來

Using algorithm A

Data Augmentation

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
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