Week 16: Security & robustness of deep learning

Instructor: Ruixuan Wang
wangruix5@mail.sysu.edu.cn

School of Data and Computer Science
Sun Yat-Sen University

13 June, 2019
Intriguing observation

- Adversarial examples: input with imperceptible perturbations, resulting in incorrect output with high confidence

Figure from Goodfellow et al., “Explaining and harnessing adversarial examples”, ICLR, 2015
Adversarial examples: reason

- Are adversarial examples from overfitting?

Figures here and in next slide from Stanford CS231n Lecture 16, 2017
Adversarial examples: reason

- Adversarial examples may come from linearity of models!
Adversarial examples: reason

- Adversarial examples may come from linearity of models!

- Modern deep learning models are very piecewise linear

Rectified linear unit

Carefully tuned sigmoid
Adversarial examples: reason

- Model responses to changes in inputs are nearly linear (Right)!
- Left: perturbed inputs along gradient direction in input space

Figure from Goodfellow et al., “Explaining and harnessing adversarial examples”, ICLR, 2015
FGSM: fast gradient sign method

- Model linearity provides one way to adversarial examples
- With 1st-order Taylor expansion, loss $L(\theta, \tilde{x}, y)$ is approx by:

  $$L(\theta, \tilde{x}, y) \approx L(\theta, x, y) + (\tilde{x} - x)^T \nabla_x L(\theta, x, y)$$

  $\theta$: model parameter; $y$: label of input $x$; $\tilde{x}$: perturbed input
FGSM: fast gradient sign method

- Model linearity provides one way to adversarial examples
- With 1st-order Taylor expansion, loss $L(\theta, \tilde{x}, y)$ is approx by:

$$L(\theta, \tilde{x}, y) \approx L(\theta, x, y) + (\tilde{x} - x)^T \nabla_x L(\theta, x, y)$$

$\theta$: model parameter; $y$: label of input $x$; $\tilde{x}$: perturbed input
- Adversarial example $\tilde{x}$ can be obtained by

$$\arg\max_{\tilde{x}} L(\theta, x, y) + (\tilde{x} - x)^T \nabla_x L(\theta, x, y)$$

s.t. $||\tilde{x} - x||_\infty < \epsilon$

where $L_\infty$ (max) norm fewer than $\epsilon$ controls perturbation!
• Model linearity provides one way to adversarial examples
• With 1st-order Taylor expansion, loss $L(\theta, \tilde{x}, y)$ is approx by:

\[
L(\theta, \tilde{x}, y) \approx L(\theta, x, y) + (\tilde{x} - x)^T \nabla_x L(\theta, x, y)
\]

$\theta$: model parameter; $y$: label of input $x$; $\tilde{x}$: perturbed input
• Adversarial example $\tilde{x}$ can be obtained by

\[
\arg \max_{\tilde{x}} L(\theta, x, y) + (\tilde{x} - x)^T \nabla_x L(\theta, x, y)
\]

s.t. $\|\tilde{x} - x\|_\infty < \epsilon$

where $L_\infty$ (max) norm fewer than $\epsilon$ controls perturbation!
• Solution: one-time computation, no need iteration

\[
\tilde{x} = x + \epsilon \text{ sign}(\nabla_x L(\theta, x, y))
\]
Attack vs. defense game

Attack with adversarial examples

- **Attack**: use adversarial examples to decrease model’s performance
- ‘White-box attack’: know model structures, parameters, etc.
- ‘Black-box attack’: can only get model output given input
Attack with adversarial examples

- **Attack**: use adversarial examples to decrease model’s performance
- ‘White-box attack’: know model structures, parameters, etc.
- ‘Black-box attack’: can only get model output given input
- Black-box attack is more common: craft adversarial examples with Model B, attack model A
- White-box attack is stronger: degrade models more seriously
FGSM result

- On MNIST dataset: $\epsilon = 0.25$ ($\epsilon$ range $[0, 1]$), simple network, classification error 89.4%, average confidence 97.6%

- With random images, FGSM fooled CNN as 'airplane' (yellow)

Figure from Goodfellow et al., "Explaining and harnessing adversarial examples", ICLR, 2015
FGSM result

- On MNIST dataset: $\epsilon = 0.25$ ($\epsilon$ range $[0, 1]$), simple network, classification error 89.4%, average confidence 97.6%
- On CIFAR-10 dataset: $\epsilon = 0.1$, simple network, classification error 87.2%, average confidence 96.6%
On MNIST dataset: $\epsilon = 0.25$ ($\epsilon$ range $[0, 1]$), simple network, classification error 89.4%, average confidence 97.6%

On CIFAR-10 dataset: $\epsilon = 0.1$, simple network, classification error 87.2%, average confidence 96.6%

With random images, FGSM fooled CNN as ‘airplane’ (yellow)

Figure from Goodfellow et al., “Explaining and harnessing adversarial examples”, ICLR, 2015
Simple extensions of FGSM

- Generating *targeted* adversarial examples with FGSM

\[ \tilde{x} = x - \epsilon \ \text{sign}(\nabla_x L(\theta, x, y_{target})) \]

where \( y_{target} \) is different from the true label of \( x \);
It would make classifier mis-classify \( \tilde{x} \) into class \( y_{target} \)
Simple extensions of FGSM

- Generating targeted adversarial examples with FGSM
  \[ \tilde{x} = x - \epsilon \text{sign}(\nabla_x L(\theta, x, y_{target})) \]
  where \( y_{target} \) is different from the true label of \( x \);
  It would make classifier mis-classify \( \tilde{x} \) into class \( y_{target} \)

- Iterative FGSM: run FGSM multiple times, with \( \alpha < \epsilon \)
  \[ x_{i+1} = \text{Clip}_{\epsilon,x} \{ x_i + \alpha \text{sign}(\nabla_x L(\theta, x_i, y)) \} \]
  where \( \text{Clip}_{\epsilon,x} \) is an operation assuring element-wise difference
  between \( x_{i+1} \) and original clean image \( x \) is within \( \epsilon \).
Iterative FGSM vs. original FGSM

- Iterative FGSM often generates more imperceptible adversarial examples (below: $\epsilon$ in range $[0, 255]$)

Figures and tables here and in next 3 slides from Kurakin et al., “Adversarial examples in the physical world”, ICLR, 2017
Adversarial examples in the physical world

- AI systems operating in the physical world often capture images directly from camera.
Adversarial examples in the physical world

- AI systems operating in the physical world often capture images directly from camera.
- Can adversarial images in physical world also fool AI system?
Adversarial examples in the physical world

- AI systems operating in the physical world often capture images directly from camera.
- Can adversarial images in physical world also fool AI system?
  - (a) Print image pairs (clean, adversarial)
  - (b) Take a photo of printed image with a cell phone camera
  - (c) Automatically crop and warp examples from the photo
  - (d) Finally feed the cropped image to classifier
In physical world: white-box attacks

- Original FGSM (‘fast’) attack is more successful than iterative FGSM in the physical world

<table>
<thead>
<tr>
<th>Adversarial method</th>
<th>Photos</th>
<th>Source images</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clean images</td>
<td>Adv. images</td>
</tr>
<tr>
<td></td>
<td>top-1</td>
<td>top-5</td>
</tr>
<tr>
<td>fast $\epsilon = 16$</td>
<td>79.8%</td>
<td>91.9%</td>
</tr>
<tr>
<td>fast $\epsilon = 8$</td>
<td>70.6%</td>
<td>93.1%</td>
</tr>
<tr>
<td>iter. basic $\epsilon = 16$</td>
<td>72.9%</td>
<td>89.6%</td>
</tr>
<tr>
<td>iter. basic $\epsilon = 8$</td>
<td>72.5%</td>
<td>93.1%</td>
</tr>
</tbody>
</table>

Note: classification accuracy in table
In physical world: white-box attacks

- Original FGSM (‘fast’) attack is more successful than iterative FGSM in the physical word.
- Reason: iterative FGSM generates adversarial examples with smaller perturbations which could be more likely removed or affected by photo transformation.

<table>
<thead>
<tr>
<th>Adversarial method</th>
<th>Photos</th>
<th>Source images</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clean images</td>
<td>Adv. images</td>
</tr>
<tr>
<td></td>
<td>top-1</td>
<td>top-5</td>
</tr>
<tr>
<td>fast $\epsilon = 16$</td>
<td>79.8%</td>
<td>91.9%</td>
</tr>
<tr>
<td>fast $\epsilon = 8$</td>
<td>70.6%</td>
<td>93.1%</td>
</tr>
<tr>
<td>iter. basic $\epsilon = 16$</td>
<td>72.9%</td>
<td>89.6%</td>
</tr>
<tr>
<td>iter. basic $\epsilon = 8$</td>
<td>72.5%</td>
<td>93.1%</td>
</tr>
</tbody>
</table>

Note: classification accuracy in table.
In physical world: black-box attack

- Black-box attack in the physical world also succeeds

(a) Image from dataset  (b) Clean image  (c) Adv. image, $\epsilon = 4$  (d) Adv. image, $\epsilon = 8$
Game: attack vs. defense

- **Defense**: reduce malicious effect of adversarial examples
Game: attack vs. defense

- **Defense**: reduce malicious effect of adversarial examples

Multiple rounds of ‘attack-defense’ game
How to defend adversarial examples from FGSM?

- Adversarial training: augment data with adversarial examples
How to defend adversarial examples from FGSM?

- Adversarial training: augment data with adversarial examples
- Find best $\theta$ by minimizing $\tilde{L}(\theta, x, y)$ over all training data

$$
\tilde{L}(\theta, x, y) = \alpha L(\theta, x, y) + (1 - \alpha) L(\theta, x + \epsilon \text{ sign}(\nabla_x L(\theta, x, y)), y)
$$
How to defend adversarial examples from FGSM?

- Adversarial training: augment data with adversarial examples
- Find best $\theta$ by minimizing $\tilde{L}(\theta, x, y)$ over all training data

$$\tilde{L}(\theta, x, y) = \alpha L(\theta, x, y) + (1 - \alpha) L(\theta, x + \epsilon \text{sign}(\nabla_x L(\theta, x, y)), y)$$

- 2$^{nd}$ term: make adversarial examples correctly classified
- With adversarial training, classification error rate of adversarial examples on MNIST was reduced from 89.4% to 17.9%
How to defend adversarial examples from FGSM?

- Adversarial training: augment data with adversarial examples
- Find best $\theta$ by minimizing $\tilde{L}(\theta, x, y)$ over all training data

$$\tilde{L}(\theta, x, y) = \alpha L(\theta, x, y) + (1 - \alpha) L(\theta, x + \epsilon \text{ sign}(\nabla_x L(\theta, x, y)), y)$$

- 2nd term: make adversarial examples correctly classified
- With adversarial training, classification error rate of adversarial examples on MNIST was reduced from 89.4% to 17.9%
- However, it works only for specific and known attack
- It remains higher vulnerable to (black-box) transferred adversarial examples produced by other models

Goodfellow et al., “Explaining and harnessing adversarial examples”, ICLR, 2015
Randomized FGSM: improved attack method

- Why adversarial training succeed?
- Model’s decision surface has sharp curvatures around data points, hindering attacks based on 1st-order approx of model’s loss, but permitting black-box attacks
Randomized FGSM: improved attack method

- Why adversarial training succeed?
- Model’s decision surface has sharp curvatures around data points, hindering attacks based on 1st-order approx of model’s loss, but permitting black-box attacks
- A new attack method based on above reason
- Randomized FGSM: apply small perturbation before FGSM

\[
x' = x + \alpha \text{ sign}(\mathcal{N}(0, I))
\]

\[
\tilde{x} = x' + (\epsilon - \alpha) \text{ sign}(\nabla_{x'} L(\theta, x', y))
\]
Randomized FGSM: improved attack method

- Why adversarial training succeed?
- Model’s decision surface has sharp curvatures around data points, hindering attacks based on 1st-order approx of model’s loss, but permitting black-box attacks
- A new attack method based on above reason
- Randomized FGSM: apply small perturbation before FGSM

\[
\begin{align*}
    \mathbf{x}' &= \mathbf{x} + \alpha \ \text{sign}(\mathcal{N}(\mathbf{0}, \mathbf{I})) \\
    \tilde{\mathbf{x}} &= \mathbf{x}' + (\epsilon - \alpha) \ \text{sign}(\nabla_{\mathbf{x}'} L(\theta, \mathbf{x}', y))
\end{align*}
\]

- Again, it is a single-time gradient computation, no iteration
- Randomized FGSM outperforms FGSM (errors in tables)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A_{adv}</th>
<th>B</th>
<th>v3</th>
<th>v3_{adv}</th>
<th>v4</th>
<th>v3</th>
<th>v3_{adv}</th>
<th>v4</th>
</tr>
</thead>
<tbody>
<tr>
<td>FGSM</td>
<td>71.4</td>
<td>3.6</td>
<td>84.6</td>
<td>69.7</td>
<td>26.8</td>
<td>60.2</td>
<td>42.8</td>
<td>9.0</td>
<td>30.8</td>
</tr>
<tr>
<td>RAND+FGSM</td>
<td>75.3</td>
<td>34.1</td>
<td>86.2</td>
<td>80.1</td>
<td>64.3</td>
<td>70.3</td>
<td>57.7</td>
<td>37.2</td>
<td>42.5</td>
</tr>
<tr>
<td>MNIST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ImageNet (top 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ImageNet (top 5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Improved defense for black-box attack

- Above: adversarial training is vulnerable to black-box attacks
Improved defense for black-box attack

- Above: adversarial training is vulnerable to black-box attacks
- Improved: ensemble adversarial training - using adversarial examples from current and other models during training
Improved defense for black-box attack

- Above: adversarial training is vulnerable to black-box attacks
- Improved: ensemble adversarial training - using adversarial examples from current and other models during training
- Ensemble adversarial training (\(A_{adv-ens}\)) shows lower errors for black-box attacks (last 4 columns)
- But it shows higher error for white-box attacks (2nd column)

<table>
<thead>
<tr>
<th></th>
<th>Model</th>
<th>Clean</th>
<th>FGSM</th>
<th>FGSM_B</th>
<th>I-FGSM_B</th>
<th>RAND+FGSM_B</th>
<th>CW_B</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 epochs</td>
<td>A</td>
<td>0.9</td>
<td>71.4</td>
<td>62.4</td>
<td>79.4</td>
<td>58.3</td>
<td>82.4</td>
</tr>
<tr>
<td></td>
<td>A_{adv}</td>
<td>1.0</td>
<td>3.6</td>
<td>18.2</td>
<td>19.8</td>
<td>12.4</td>
<td>21.8</td>
</tr>
<tr>
<td></td>
<td>A_{adv-ens}</td>
<td>0.9</td>
<td>11.8</td>
<td>5.0</td>
<td>9.7</td>
<td>3.4</td>
<td>13.7</td>
</tr>
<tr>
<td>12 epochs</td>
<td>A_{adv}</td>
<td>0.7</td>
<td>3.8</td>
<td>15.5</td>
<td>13.5</td>
<td>9.5</td>
<td>15.2</td>
</tr>
<tr>
<td></td>
<td>A_{adv-ens}</td>
<td>0.7</td>
<td>6.0</td>
<td>3.9</td>
<td>6.2</td>
<td>2.9</td>
<td>7.0</td>
</tr>
</tbody>
</table>

Tables here and in prev slide from Tramer et al., “Ensemble adversarial training: attacks and defenses”, arXiv, 2017
More defense and attack methods to come!
Defensive distillation network

- Train a distillation network with modified softmax
  \[
  \text{softmax}(x, T)_i = \frac{e^{x_i/T}}{\sum_j e^{x_j/T}}
  \]

- Large $T$ (e.g., 100) for training; small (e.g., 1) for inference

Papernot et al., “Distillation as a defense to adversarial perturbations against deep neural networks”, SSP, 2016
Defensive distillation network

- Distilled network reduces success rate of adversarial example crafting from original 95% to 0.5% on MNIST set
Defensive distillation network

- Distilled network reduces success rate of adversarial example crafting from original 95% to 0.5% on MNIST set
- Why does it work?
  - Training causes pre-softmax signal becomes larger by factor $T$
  - Then small $T$ during testing makes output of one neuron almost 1.0 and the others almost 0.0.
  - This makes gradient of loss function w.r.t input become almost zero, causing gradient-based attacking not working
Defensive distillation network

- Distilled network reduces success rate of adversarial example crafting from original 95% to 0.5% on MNIST set
- Why does it work?
- Training causes pre-softmax signal becomes larger by factor $T$
- Then small $T$ during testing makes output of one neuron almost 1.0 and the others almost 0.0.
- This makes gradient of loss function w.r.t input become almost zero, causing gradient-based attacking not working

When you find a reason, you find a solution!
Carlini-Wagner (CW) method: find small perturbation $\delta$ by

$$\min_{\delta \in \mathbb{R}^n} \|\delta\|_p + c \cdot f(x + \delta)$$

subject to $x + \delta \in [0, 1]^n$, where $f$ is an objective function that drives $x$ to be misclassified to a targeted class; $L_p$ norm: $p = 0, 2, \infty$

Key innovation: use smooth version of representation for $\delta$, $L_p$, and $f$, such that gradients of both terms are not zero.

Formula here and figures in next 3 slides from Carlini and Wagner, “Towards evaluating the robustness of neural networks”, arXiv, 2017
CW attack: result

- Targeted adversarial examples with imperceptible perturbation
- Similar results on ImageNet data
**CW attack: result**

- Targeted adversarial examples; init: black or white images

<table>
<thead>
<tr>
<th>Distance Metric</th>
<th>Target Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>L_0</td>
<td>0</td>
</tr>
<tr>
<td>L_2</td>
<td>[Black images]</td>
</tr>
<tr>
<td>L_\infty</td>
<td>[Black images]</td>
</tr>
</tbody>
</table>

- Black images indicate adversarial examples, white images indicate original images.
CW attack: transferable

- Higher-confidence adversarial examples are more transferable
- ‘k’ in function $f$ controls confidence of adversarial examples
A new way: preprocess to remove adversarial noise
MagNet

- A new way: preprocess to remove adversarial noise
- Train autoencoder (AE) with normal training dataset
- For new normal input, output of AE is close to input
- For adversarial input, AE tries to output a similar normal data

Curve here & tables in next 2 slides from Meng & Chen, “MagNet: a two-pronged defense against adversarial examples”, CCS, 2017
A new way: preprocess to remove adversarial noise
Train autoencoder (AE) with normal training dataset
For new normal input, output of AE is close to input
For adversarial input, AE tries to output a similar normal data
MagNet is independent of classifier and attacks

Curves here & tables in next 2 slides from Meng & Chen, “MagNet: a two-pronged defense against adversarial examples”, CCS, 2017
MagNet: result

- Magnet successfully defends black-box attacks

<table>
<thead>
<tr>
<th>Attack</th>
<th>Norm</th>
<th>Parameter</th>
<th>No Defense</th>
<th>With Defense</th>
</tr>
</thead>
<tbody>
<tr>
<td>FGSM</td>
<td>$L^\infty$</td>
<td>$\epsilon = 0.005$</td>
<td>96.8%</td>
<td>100.0%</td>
</tr>
<tr>
<td>FGSM</td>
<td>$L^\infty$</td>
<td>$\epsilon = 0.010$</td>
<td>91.1%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Iterative</td>
<td>$L^\infty$</td>
<td>$\epsilon = 0.005$</td>
<td>95.2%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Iterative</td>
<td>$L^\infty$</td>
<td>$\epsilon = 0.010$</td>
<td>72.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Iterative</td>
<td>$L^2$</td>
<td>$\epsilon = 0.5$</td>
<td>86.7%</td>
<td>99.2%</td>
</tr>
<tr>
<td>Iterative</td>
<td>$L^2$</td>
<td>$\epsilon = 1.0$</td>
<td>76.6%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Deepfool</td>
<td>$L^\infty$</td>
<td></td>
<td>19.1%</td>
<td>99.4%</td>
</tr>
<tr>
<td>Carlini</td>
<td>$L^2$</td>
<td></td>
<td>0.0%</td>
<td>99.5%</td>
</tr>
<tr>
<td>Carlini</td>
<td>$L^\infty$</td>
<td></td>
<td>0.0%</td>
<td>99.8%</td>
</tr>
<tr>
<td>Carlini</td>
<td>$L^0$</td>
<td></td>
<td>0.0%</td>
<td>92.0%</td>
</tr>
</tbody>
</table>

- However, it fails for white-box attacks, where structures and parameters of classifier and Magnet are known to attackers
Attack vs. defense game

MagNet: result

- But, MagNet performs well for gray-box attacks
- Gray-box attacks: attacks know defense model’s structure, training data, etc.; but do not know defense parameter
- How: train multiple MagNets, randomly choose one during testing (A-H: autoencoders; column: attack trained on; row: used during testing; number: classification accuracy)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.0</td>
<td>92.8</td>
<td>92.5</td>
<td>93.1</td>
<td>91.8</td>
<td>91.8</td>
<td>92.5</td>
<td>93.6</td>
</tr>
<tr>
<td>B</td>
<td>92.1</td>
<td>0.0</td>
<td>92.0</td>
<td>92.5</td>
<td>91.4</td>
<td>92.5</td>
<td>91.3</td>
<td>92.5</td>
</tr>
<tr>
<td>C</td>
<td>93.2</td>
<td>93.8</td>
<td>0.0</td>
<td>92.8</td>
<td>93.3</td>
<td>94.1</td>
<td>92.7</td>
<td>93.6</td>
</tr>
<tr>
<td>D</td>
<td>92.8</td>
<td>92.2</td>
<td>91.3</td>
<td>0.0</td>
<td>91.7</td>
<td>92.8</td>
<td>91.2</td>
<td>93.9</td>
</tr>
<tr>
<td>E</td>
<td>93.3</td>
<td>94.0</td>
<td>93.4</td>
<td>93.2</td>
<td>0.0</td>
<td>93.4</td>
<td>91.0</td>
<td>92.8</td>
</tr>
<tr>
<td>F</td>
<td>92.8</td>
<td>93.1</td>
<td>93.2</td>
<td>93.6</td>
<td>92.2</td>
<td>0.0</td>
<td>92.8</td>
<td>93.8</td>
</tr>
<tr>
<td>G</td>
<td>92.5</td>
<td>93.1</td>
<td>92.0</td>
<td>92.2</td>
<td>90.5</td>
<td>93.5</td>
<td>0.1</td>
<td>93.4</td>
</tr>
<tr>
<td>H</td>
<td>92.3</td>
<td>92.0</td>
<td>91.8</td>
<td>92.6</td>
<td>91.4</td>
<td>92.3</td>
<td>92.4</td>
<td>0.0</td>
</tr>
<tr>
<td>Random</td>
<td>81.1</td>
<td>81.4</td>
<td>80.8</td>
<td>81.3</td>
<td>80.3</td>
<td>81.3</td>
<td>80.5</td>
<td>81.7</td>
</tr>
</tbody>
</table>
Defense GAN

- Another way to remove adversarial noise from input
- Step 1: train a GAN with clean data
- Step 2: given any data $x$, obtain its reconstruction with $G$

$$z^* = \arg \min_z \|G(z) - x\|_2^2$$

- Step 3: train classifier with GAN-reconstructed data, or with original data, or with both
- Given a test data, use GAN-rec data as input to classifier
Defense GAN

- Defense GAN is independent of any classifier
- It does not assume any attack model, well for black-box attack
- It is highly nonlinear, making white-box attack difficult
- Note: more iterations result in more precise reconstruction which contains more adversarial noise, causing worse defense
Defense GAN: result

- Outperforms others in defending black-box (FGSM) attacks.
- 2nd last col: same 0.3 used for adversarial example generation.
- Last 2 columns: large variance in performance.

<table>
<thead>
<tr>
<th>Classifier/Substitute</th>
<th>No Attack</th>
<th>No Defense</th>
<th>Defense-GAN-Rec</th>
<th>Defense-GAN-Orig</th>
<th>MagNet $\epsilon = 0.3$</th>
<th>Adv. Tr. $\epsilon = 0.3$</th>
<th>Adv. Tr. $\epsilon = 0.15$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/B</td>
<td>0.9970</td>
<td>0.6343</td>
<td>0.9312</td>
<td>0.9282</td>
<td>0.6937</td>
<td><strong>0.9654</strong></td>
<td>0.6223</td>
</tr>
<tr>
<td>A/E</td>
<td>0.9970</td>
<td>0.5432</td>
<td>0.9139</td>
<td>0.9221</td>
<td>0.6710</td>
<td><strong>0.9668</strong></td>
<td>0.9327</td>
</tr>
<tr>
<td>B/B</td>
<td>0.9618</td>
<td>0.2816</td>
<td>0.9057</td>
<td><strong>0.9105</strong></td>
<td>0.5687</td>
<td>0.2092</td>
<td>0.3441</td>
</tr>
<tr>
<td>B/E</td>
<td>0.9618</td>
<td>0.2128</td>
<td><strong>0.8841</strong></td>
<td><strong>0.8892</strong></td>
<td>0.4627</td>
<td>0.1120</td>
<td>0.3354</td>
</tr>
<tr>
<td>C/B</td>
<td>0.9959</td>
<td>0.6648</td>
<td><strong>0.9357</strong></td>
<td>0.9322</td>
<td>0.7571</td>
<td><strong>0.9834</strong></td>
<td>0.9208</td>
</tr>
<tr>
<td>C/E</td>
<td>0.9959</td>
<td>0.8050</td>
<td>0.9223</td>
<td>0.9182</td>
<td>0.6760</td>
<td><strong>0.9843</strong></td>
<td>0.9755</td>
</tr>
<tr>
<td>D/B</td>
<td>0.9920</td>
<td>0.4641</td>
<td>0.9272</td>
<td><strong>0.9323</strong></td>
<td>0.6817</td>
<td>0.7667</td>
<td>0.8514</td>
</tr>
<tr>
<td>D/E</td>
<td>0.9920</td>
<td>0.3931</td>
<td><strong>0.9164</strong></td>
<td>0.9155</td>
<td>0.6073</td>
<td>0.7676</td>
<td>0.7129</td>
</tr>
</tbody>
</table>

- ‘A/B’: use adversarial examples generated by classifier B to attack classifier A
- ‘Defense-GAN-Rec/Orig’: use GAN-reconstructed or the original images to train classifier
Defense GAN: result

- Outperforms others in defending white-box attacks
- Reconstructed data from G contain little adversarial noise!

<table>
<thead>
<tr>
<th>Attack</th>
<th>Classifier Model</th>
<th>No Attack</th>
<th>No Defense</th>
<th>Defense-GAN-Rec</th>
<th>MagNet</th>
<th>Adv. Tr. $\epsilon = 0.3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FGSM $\epsilon = 0.3$</td>
<td>A</td>
<td>0.997</td>
<td>0.217</td>
<td>0.988</td>
<td>0.191</td>
<td>0.651</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.962</td>
<td>0.022</td>
<td>0.956</td>
<td>0.082</td>
<td>0.060</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>0.996</td>
<td>0.331</td>
<td>0.989</td>
<td>0.163</td>
<td>0.786</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>0.992</td>
<td>0.038</td>
<td>0.980</td>
<td>0.094</td>
<td>0.732</td>
</tr>
<tr>
<td>RAND+FGSM $\epsilon = 0.3, \alpha = 0.05$</td>
<td>A</td>
<td>0.997</td>
<td>0.179</td>
<td>0.988</td>
<td>0.171</td>
<td>0.774</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.962</td>
<td>0.017</td>
<td>0.944</td>
<td>0.091</td>
<td>0.138</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>0.996</td>
<td>0.103</td>
<td>0.985</td>
<td>0.151</td>
<td>0.907</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>0.992</td>
<td>0.050</td>
<td>0.980</td>
<td>0.115</td>
<td>0.539</td>
</tr>
<tr>
<td>CW $\ell_2$ norm</td>
<td>A</td>
<td>0.997</td>
<td>0.141</td>
<td>0.989</td>
<td>0.038</td>
<td>0.077</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.962</td>
<td>0.032</td>
<td>0.916</td>
<td>0.034</td>
<td>0.280</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>0.996</td>
<td>0.126</td>
<td>0.989</td>
<td>0.025</td>
<td>0.031</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>0.992</td>
<td>0.032</td>
<td>0.983</td>
<td>0.021</td>
<td>0.010</td>
</tr>
</tbody>
</table>

Tables and figures here and in prev 3 slides from Samangouei et al., “Defense-GAN: protecting classifiers against adversarial attacks using generative models”, ICLR, 2018
The game is far from over...

- **Left:** universal adversarial perturbation
- **Right:** one pixel attack for fooling deep neural networks

Moosavi-Dezfooli & Fawzi, CVPR 2017; Su et al., arXiv 2017
Summary

- Adversarial examples put serious challenges to security and robustness of DL models (and other machine learning models)
- Multi-round attack-vs-defense game is running
- The game would help understand weakness of current DL models, and help develop more robust and innovative models

Further reading:

- Madry et al., ‘Towards deep learning models resistant to adversarial attacks’, arXiv, 2017
- Qin et al., ‘Imperceptible, robust, and targeted adversarial examples for automatic speech recognition’, ICML, 2019