Trustworthy Machine Learning on Adversarial Data

Lecturer: Dr. Feng Liu
School of Computing and Information Systems
The University of Melbourne
Date: 21 Feb 2024, Vancouver, Canada
About the Lecturer

• **Name:** Feng Liu
• **Position:** Assistant Professor in Machine Learning
• **Institution:** School of Computing and Information Systems, University of Melbourne
• **Research Interests:** Distribution Shift Detection, Learning under Distribution Shift

![Diagram](distribution_shift.png)

- **Distribution Shift**
  - Adversarial Shift: Defense: Detection/Robust to/Purified
  - Out-of-distribution Shift: Detection/Generalization
  - Natural Shift: Adaptation/Transfer Learning
What is the adversarial data?

Adversarial attacks

What is the adversarial data?

Conventional Machine Learning Pipeline (classification):

Training Data with labels → Train a model → NN, CNN, Transformer
What is the adversarial data?

Conventional Machine Learning Pipeline (classification):

Test Data → Input → Well-trained NN, Well-trained CNN, Well-trained Transformer → Predicted Labels
What is the adversarial data?

Adversarial attack happens:

Test Data + Adversarial Perturbations

Input

Well-trained NN,
Well-trained CNN
Well-trained Transformer

Predicted Labels

Untrusted
Why adversarial attack can be successful

Adversarial attacks


Different distributions → Significant Error
Basic Assumption in Machine Learning

Train a model
NN, CNN, …

Training Set

Test Set
Basic Assumption in Machine Learning

CIFAR10

CIFAR10.1 [ICML’19]

Basic Assumption in Machine Learning
Basic Assumption in Machine Learning

Different distributions break the assumption

Same Distribution

Significant Error

Basic Assumption in Machine Learning
Why adversarial attack can be successful

Adversarial attacks


How to prevent from adversarial attacks

Trustworthy Machine Learning Under Adversarial Data

- Actively Handle Adversarial Data
- Passively Handle Adversarial Data

Data Perspective
- Adversarial Purification

Model Perspective
- Adversarial Training
- Adversarial Detection
How to actively prevent from adversarial attacks

Adversarial attack happens:

Test Data + Adversarial Perturbations

Purify

Train a robust model

Well-trained NN, Well-trained CNN
Well-trained Transformer

Predicted Labels
How to passively prevent from adversarial attacks

Adversarial attack happens:

Test Data + Adversarial Perturbations

Discard the adversarial data

Input

Well-trained NN, Well-trained CNN, Well-trained Transformer

Predicted Labels
ACML 2023 Tutorial: Trustworthy Machine Learning on Adversarial Data

Part I: Adversarial Purification

Let’s try purify the upcoming test data!!
How to actively prevent from adversarial attacks

Adversarial attack happens:

Test Data + Adversarial Perturbations

Purify

Input

Well-trained NN, Well-trained CNN, Well-trained Transformer

Predicted Labels
Adversarial Purification: General Pipeline

Research trajectory: use the power of *generative models* to purify adversarial examples.

High-level idea: use a *pre-trained generative model on natural examples* to reconstruct adversarial examples into samples that are close to natural examples.

Intuition: a generative model reconstructs samples based on its learned knowledge (e.g., the manifold of natural examples). Thus, a *pre-trained generative model on natural examples* will align any samples to the manifold of natural examples during the reconstruction.
Adversarial Purification: General Pipeline

**Black curve**: manifold of natural examples in a 2-D sample space.

**Green dots**: natural examples.

**Red crosses**: adversarial examples.

Adversarial Purification: MagNet

MagNet: a Two-Pronged Defense against Adversarial Examples

Main idea: use the reconstructive power of an autoencoder to purify (i.e., reform) adversarial examples.

Limitation: the autoencoder is not strong enough, i.e., cannot purify examples that are too far away from the manifold of natural examples.

We can use detectors to filter out hard-to-purify adversarial examples first!

Adversarial Purification: MagNet

MagNet: a Two-Pronged Defense against Adversarial Examples

Solution: use detectors to filter out hard-to-purify adversarial examples first, and then use an autoencoder to purify the remaining adversarial examples.

Rationale: any adversarial examples that cannot be detected by detectors must be close to the manifold of natural examples, and thus can be purified by an autoencoder.
Adversarial Purification: MagNet

MagNet: a Two-Pronged Defense against Adversarial Examples

<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>

<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>$\epsilon = 0.025$</td>
<td>46.0%</td>
<td>99.9%</td>
</tr>
<tr>
<td>FGSM</td>
<td>$L^\infty$</td>
<td>$\epsilon = 0.050$</td>
<td>40.5%</td>
<td>100.0%</td>
</tr>
<tr>
<td>Iterative</td>
<td>$L^\infty$</td>
<td>$\epsilon = 0.010$</td>
<td>28.6%</td>
<td>96.0%</td>
</tr>
<tr>
<td>Iterative</td>
<td>$L^\infty$</td>
<td>$\epsilon = 0.025$</td>
<td>11.1%</td>
<td>99.9%</td>
</tr>
<tr>
<td>Iterative</td>
<td>$L^2$</td>
<td>$\epsilon = 0.25$</td>
<td>18.4%</td>
<td>76.3%</td>
</tr>
<tr>
<td>Iterative</td>
<td>$L^2$</td>
<td>$\epsilon = 0.50$</td>
<td>6.6%</td>
<td>83.3%</td>
</tr>
<tr>
<td>Deepfool</td>
<td>$L^\infty$</td>
<td></td>
<td>4.5%</td>
<td>93.4%</td>
</tr>
<tr>
<td>Carlini</td>
<td>$L^2$</td>
<td></td>
<td>0.0%</td>
<td>93.7%</td>
</tr>
<tr>
<td>Carlini</td>
<td>$L^\infty$</td>
<td></td>
<td>0.0%</td>
<td>83.0%</td>
</tr>
<tr>
<td>Carlini</td>
<td>$L^0$</td>
<td></td>
<td>0.0%</td>
<td>77.5%</td>
</tr>
</tbody>
</table>

Q: Can we purify adversarial examples without filtering out those hard-to-purify ones?
A: Yes! A natural idea is to use a stronger generative model.
Adversarial Purification: Defense GAN

Protecting Classifiers Against Adversarial Attacks Using Generative Models

Solution: use a pre-trained GAN to reconstruct adversarial examples.
Adversarial Purification: Defense GAN

Protecting Classifiers Against Adversarial Attacks Using Generative Models

Minimization: L steps of Gradient Descent are used to estimate the projection of the image onto the range of the generator.

\[ \mathcal{Z}_0 = \left\{ z_0^{(i)} \right\}_{i=1}^R \]

\[ x \]

\[ \mathcal{Z}_L = \left\{ z_L^{(i)} \right\}_{i=1}^R \]

\[ \eta_j \]

\[ j \leftarrow j + 1 \]

\[ L \text{ times} \]

\[ \arg \min_{z \in \mathcal{Z}_L} \| G(z) - x \|_2^2 \]

### Protecting Classifiers Against Adversarial Attacks Using Generative Models

**On MNIST:**

<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><strong>FGSM $\epsilon = 0.3$</strong></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><strong>RAND+FGSM $\epsilon = 0.3, \alpha = 0.05$</strong></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><strong>CW $\ell_2$ norm</strong></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>
Adversarial Purification: Defense GAN

Protecting Classifiers Against Adversarial Attacks Using Generative Models

On F-MNIST:

<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</td>
<td>A</td>
<td>0.934</td>
<td>0.102</td>
<td>0.879</td>
<td>0.089</td>
<td>0.797</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.747</td>
<td>0.102</td>
<td>0.629</td>
<td>0.168</td>
<td>0.136</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>0.933</td>
<td>0.139</td>
<td>0.896</td>
<td>0.110</td>
<td>0.804</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>0.892</td>
<td>0.082</td>
<td>0.875</td>
<td>0.099</td>
<td>0.698</td>
</tr>
<tr>
<td>RAND+FGSM</td>
<td>A</td>
<td>0.934</td>
<td>0.102</td>
<td>0.888</td>
<td>0.096</td>
<td>0.447</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.747</td>
<td>0.131</td>
<td>0.661</td>
<td>0.161</td>
<td>0.119</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>0.933</td>
<td>0.105</td>
<td>0.893</td>
<td>0.112</td>
<td>0.699</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>0.892</td>
<td>0.091</td>
<td>0.862</td>
<td>0.104</td>
<td>0.626</td>
</tr>
<tr>
<td>CW</td>
<td>A</td>
<td>0.934</td>
<td>0.076</td>
<td>0.896</td>
<td>0.060</td>
<td>0.157</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.747</td>
<td>0.172</td>
<td>0.656</td>
<td>0.131</td>
<td>0.118</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>0.933</td>
<td>0.063</td>
<td>0.896</td>
<td>0.084</td>
<td>0.107</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>0.892</td>
<td>0.090</td>
<td>0.875</td>
<td>0.069</td>
<td>0.149</td>
</tr>
</tbody>
</table>

Adversarial Purification: PixelDefend

Leveraging Generative Models to Understand and Defend against Adversarial Examples

**Solution:** use a pre-trained PixelCNN to reconstruct adversarial examples.

PixelCNN is a generative model with tractable likelihood especially designed for images. The model defines the joint distribution over all pixels by factorizing it into a product of conditional distributions.

\[
p_{\text{CNN}}(X) = \prod_i p_{\text{CNN}}(x_i | x_{1:(i-1)}).
\]
Algorithm 1 PixelDefend

Input: Image $X$, Defense parameter $\epsilon_{\text{defend}}$, Pre-trained PixelCNN model $p_{\text{CNN}}$

Output: Purified Image $X^*$

1: $X^* \leftarrow X$
2: for each row $i$ do
3:     for each column $j$ do
4:         for each channel $k$ do
5:             $x \leftarrow X[i, j, k]$
6:             Set feasible range $R \leftarrow [\max(x - \epsilon_{\text{defend}}, 0), \min(x + \epsilon_{\text{defend}}, 255)]$
7:             Compute the 256-way softmax $p_{\text{CNN}}(X^*)$.
8:             Update $X^*[i, j, k] \leftarrow \arg \max_{z \in R} p_{\text{CNN}[i, j, k, z]$
9:         end for
10:     end for
11: end for

Song et al., Leveraging Generative Models to Understand and Defend against Adversarial Examples. ICLR, 2018.
## Adversarial Purification: PixelDefend

Leveraging Generative Models to Understand and Defend against Adversarial Examples

### Table 1: Fashion MNIST \((\epsilon_{\text{attack}} = 8/25, \epsilon_{\text{defend}} = 32)\)

<table>
<thead>
<tr>
<th>NETWORK</th>
<th>TRAINING TECHNIQUE</th>
<th>CLEAN</th>
<th>RAND</th>
<th>FGSM</th>
<th>BIM</th>
<th>DEEP FOOL</th>
<th>CW</th>
<th>STRONGEST ATTACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet</td>
<td>Normal</td>
<td>93/93</td>
<td>89/71</td>
<td>38/24</td>
<td>00/00</td>
<td>06/06</td>
<td>20/01</td>
<td>00/00</td>
</tr>
<tr>
<td>VGG</td>
<td>Normal</td>
<td>92/92</td>
<td>91/87</td>
<td>73/58</td>
<td>36/08</td>
<td>49/14</td>
<td>43/23</td>
<td>36/08</td>
</tr>
<tr>
<td></td>
<td>Adversarial FGSM</td>
<td>93/93</td>
<td>92/89</td>
<td>85/85</td>
<td>51/00</td>
<td>63/07</td>
<td>67/21</td>
<td>51/00</td>
</tr>
<tr>
<td></td>
<td>Adversarial BIM</td>
<td>92/91</td>
<td>92/91</td>
<td>84/79</td>
<td>76/63</td>
<td>82/72</td>
<td>81/70</td>
<td>76/63</td>
</tr>
<tr>
<td>ResNet</td>
<td>Label Smoothing</td>
<td>93/93</td>
<td>91/76</td>
<td>73/45</td>
<td>16/00</td>
<td>29/06</td>
<td>33/14</td>
<td>16/00</td>
</tr>
<tr>
<td></td>
<td>Feature Squeezing</td>
<td>84/84</td>
<td>84/70</td>
<td>70/28</td>
<td>56/25</td>
<td>83/83</td>
<td>83/83</td>
<td>56/25</td>
</tr>
<tr>
<td></td>
<td>Adversarial FGSM + Feature Squeezing</td>
<td>88/88</td>
<td>87/82</td>
<td>80/77</td>
<td>70/46</td>
<td>86/82</td>
<td>84/85</td>
<td>70/46</td>
</tr>
<tr>
<td>ResNet</td>
<td>Normal + PixelDefend</td>
<td>88/88</td>
<td>88/89</td>
<td>85/74</td>
<td>83/76</td>
<td>87/87</td>
<td>87/87</td>
<td>83/74</td>
</tr>
<tr>
<td>VGG</td>
<td>Normal + PixelDefend</td>
<td>89/89</td>
<td>89/89</td>
<td>87/82</td>
<td>85/83</td>
<td>88/88</td>
<td>88/88</td>
<td>85/82</td>
</tr>
<tr>
<td>ResNet</td>
<td>Adversarial FGSM + PixelDefend</td>
<td>90/89</td>
<td>91/90</td>
<td>88/82</td>
<td>85/76</td>
<td>90/88</td>
<td>89/88</td>
<td>85/76</td>
</tr>
<tr>
<td></td>
<td>Adversarial FGSM + Adaptive PixelDefend</td>
<td>91/91</td>
<td>91/91</td>
<td>88/88</td>
<td>85/84</td>
<td>89/90</td>
<td>89/84</td>
<td>85/84</td>
</tr>
</tbody>
</table>
### Adversarial Purification: PixelDefend

Leveraging Generative Models to Understand and Defend against Adversarial Examples

Table 2: **CIFAR-10** ($\epsilon_{\text{attack}} = 2/8/16$, $\epsilon_{\text{defend}} = 16$)

<table>
<thead>
<tr>
<th>NETWORK</th>
<th>TRAINING TECHNIQUE</th>
<th>CLEAN</th>
<th>RAND</th>
<th>FGSM</th>
<th>BIM</th>
<th>DEEP FOOL</th>
<th>CW</th>
<th>STRONGEST ATTACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet</td>
<td>Normal</td>
<td>92/92/92</td>
<td>92/87/76</td>
<td>33/15/11</td>
<td>10/00/00</td>
<td>12/06/06</td>
<td>07/00/00</td>
<td>07/00/00</td>
</tr>
<tr>
<td>VGG</td>
<td>Normal</td>
<td>89/89/89</td>
<td>89/88/80</td>
<td>60/46/30</td>
<td>44/02/00</td>
<td>57/25/11</td>
<td>37/00/00</td>
<td>37/00/00</td>
</tr>
<tr>
<td>ResNet</td>
<td>Adversarial FGSM</td>
<td>91/91/91</td>
<td>90/88/84</td>
<td>88/91/91</td>
<td>24/07/00</td>
<td>45/00/00</td>
<td>20/00/07</td>
<td>20/00/00</td>
</tr>
<tr>
<td></td>
<td>Adversarial BIM</td>
<td>87/87/87</td>
<td>87/87/86</td>
<td>80/52/34</td>
<td>74/32/00</td>
<td>79/48/25</td>
<td>76/42/08</td>
<td>74/32/06</td>
</tr>
<tr>
<td></td>
<td>Label Smoothing</td>
<td>92/92/92</td>
<td>91/88/77</td>
<td>73/51/28</td>
<td>50/08/01</td>
<td>56/20/10</td>
<td>30/02/02</td>
<td>30/02/01</td>
</tr>
<tr>
<td></td>
<td>Feature Squeezing</td>
<td>84/84/84</td>
<td>83/82/76</td>
<td>31/20/18</td>
<td>13/00/00</td>
<td>75/75/75</td>
<td>78/78/78</td>
<td>78/78/78</td>
</tr>
<tr>
<td></td>
<td>Adversarial FGSM + Feature Squeezing</td>
<td>86/86/86</td>
<td>85/84/81</td>
<td>73/67/55</td>
<td>55/02/00</td>
<td>85/85/85</td>
<td>83/83/83</td>
<td>55/02/00</td>
</tr>
<tr>
<td>ResNet</td>
<td>Normal + <strong>PixelDefend</strong></td>
<td>85/85/88</td>
<td>82/83/84</td>
<td>73/46/24</td>
<td>71/16/25</td>
<td>80/80/80</td>
<td>78/78/78</td>
<td>71/46/24</td>
</tr>
<tr>
<td>ResNet</td>
<td>Adversarial FGSM + <strong>PixelDefend</strong></td>
<td>88/88/86</td>
<td>86/86/87</td>
<td>81/68/67</td>
<td>81/09/56</td>
<td>85/85/85</td>
<td>84/84/84</td>
<td>81/09/56</td>
</tr>
<tr>
<td></td>
<td>Adversarial FGSM + <strong>Adaptive PixelDefend</strong></td>
<td>90/90/90</td>
<td>86/87/87</td>
<td>81/70/67</td>
<td>81/70/56</td>
<td>82/81/82</td>
<td>81/80/81</td>
<td>81/70/56</td>
</tr>
</tbody>
</table>
Adversarial Purification based on Score

Adversarial Purification with Score-based Generative Models

**Motivation:** if we treat adversarial purification as a process of denoising adversarial attacks, why not try a generative model that is designed to denoise the perturbed samples?

**Solution:** use a pre-trained generative model with Denoising Score Matching (DSM), i.e., the score network, which is particularly good at denoising perturbed images with noise.
Adversarial Purification based on Score

A breakthrough: injecting noise (e.g., Gaussian noise) into adversarial examples before feeding into the score network.

Benefits of injecting noise to adversarial examples:
1. by injecting relatively larger noise, we can make the adversarial perturbations negligible.
2. convert adversarial images to noisy images that are similar to the ones seen by the network.

Adversarial Purification based on Score

on CIFAR-10:

<table>
<thead>
<tr>
<th>Models</th>
<th>Accuracy (%)</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Standard</td>
<td>Robust</td>
</tr>
<tr>
<td>Raw WideResNet</td>
<td>95.80</td>
<td>0.00</td>
</tr>
<tr>
<td>ADP ($\sigma = 0.1$)</td>
<td>93.09</td>
<td>85.45</td>
</tr>
<tr>
<td>ADP ($\sigma = 0.25$)</td>
<td>86.14</td>
<td>80.24</td>
</tr>
</tbody>
</table>

Adversarial purification methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Accuracy</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Hill et al., 2021)</td>
<td>84.12</td>
<td>WRN-28-10</td>
</tr>
<tr>
<td>(Shi et al., 2021)</td>
<td>96.93</td>
<td>WRN-28-10</td>
</tr>
<tr>
<td>(Du &amp; Mordatch, 2019)*</td>
<td>48.7</td>
<td>WRN-28-10</td>
</tr>
<tr>
<td>(Grathwohl et al., 2020)*</td>
<td>75.5</td>
<td>WRN-28-10</td>
</tr>
<tr>
<td>(Yang et al., 2019)*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p = 0.8 \rightarrow 1.0$</td>
<td>94.9</td>
<td>ResNet-18</td>
</tr>
<tr>
<td>$p = 0.6 \rightarrow 0.8$</td>
<td>92.1</td>
<td>ResNet-18</td>
</tr>
<tr>
<td>$p = 0.4 \rightarrow 0.6$</td>
<td>89.2</td>
<td>ResNet-18</td>
</tr>
<tr>
<td>(Song et al., 2018)*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Natural + PixelCNN</td>
<td>82</td>
<td>ResNet-62</td>
</tr>
<tr>
<td>AT + PixelCNN</td>
<td>90</td>
<td>ResNet-62</td>
</tr>
</tbody>
</table>

Adversarial training methods, transfer-based

<table>
<thead>
<tr>
<th>Methods</th>
<th>Accuracy</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Madry et al., 2018)*</td>
<td>87.3</td>
<td>ResNet-56</td>
</tr>
<tr>
<td>(Zhang et al., 2019)*</td>
<td>84.9</td>
<td>ResNet-56</td>
</tr>
</tbody>
</table>

on CIFAR-10-C:

<table>
<thead>
<tr>
<th>Models</th>
<th>Accuracy</th>
<th>Noise</th>
<th>Blur</th>
<th>Weather</th>
<th>Digital</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw WideResNet</td>
<td>71.89</td>
<td>42.37</td>
<td>72.74</td>
<td>86.23</td>
<td>78.83</td>
</tr>
<tr>
<td>ADP ($\sigma = 0.25$)</td>
<td>77.45</td>
<td>84.68</td>
<td>75.52</td>
<td>77.98</td>
<td>73.42</td>
</tr>
<tr>
<td>ADP ($\sigma = 0.25$)+Detection</td>
<td>78.96</td>
<td>84.57</td>
<td>71.76</td>
<td>84.34</td>
<td>76.60</td>
</tr>
<tr>
<td>ADP ($\sigma = 0.1$)</td>
<td>76.25</td>
<td>86.88</td>
<td>68.55</td>
<td>78.88</td>
<td>73.36</td>
</tr>
<tr>
<td>ADP, ($\sigma = 0.0$)</td>
<td>80.49</td>
<td>84.58</td>
<td>73.12</td>
<td>86.39</td>
<td>78.89</td>
</tr>
<tr>
<td>ADP, ($\sigma = 0.0 + DCT$)</td>
<td>80.47</td>
<td>84.96</td>
<td>73.11</td>
<td>87.26</td>
<td>78.69</td>
</tr>
<tr>
<td>ADP, ($\sigma = 0.0 + AugMix$)</td>
<td>82.63</td>
<td>87.52</td>
<td>75.20</td>
<td>88.82</td>
<td>80.20</td>
</tr>
<tr>
<td>ADP, ($\sigma = 0.0 + DCT+AugMix$)</td>
<td>82.40</td>
<td>85.05</td>
<td>75.80</td>
<td>88.11</td>
<td>81.30</td>
</tr>
</tbody>
</table>

Adversarial training methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Accuracy</th>
<th>Noise</th>
<th>Blur</th>
<th>Weather</th>
<th>Digital</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Zhang et al., 2019)</td>
<td>75.63</td>
<td>77.83</td>
<td>78.37</td>
<td>74.98</td>
<td>71.88</td>
</tr>
<tr>
<td>(Carmon et al., 2019)</td>
<td>80.40</td>
<td>81.20</td>
<td>83.44</td>
<td>80.19</td>
<td>76.98</td>
</tr>
<tr>
<td>(Cohen et al., 2019)</td>
<td>73.70</td>
<td>81.48</td>
<td>72.60</td>
<td>72.19</td>
<td>70.46</td>
</tr>
</tbody>
</table>

Training classifiers with augmentations

<table>
<thead>
<tr>
<th>Methods</th>
<th>Accuracy</th>
<th>Noise</th>
<th>Blur</th>
<th>Weather</th>
<th>Digital</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Hendrycks et al., 2020)*</td>
<td>88.78</td>
<td>84.66</td>
<td>89.68</td>
<td>90.93</td>
<td>88.47</td>
</tr>
<tr>
<td>(Wang et al., 2021)*</td>
<td>89.52</td>
<td>85.61</td>
<td>89.68</td>
<td>91.88</td>
<td>89.95</td>
</tr>
<tr>
<td>(Hossain et al., 2020)*</td>
<td>89.17</td>
<td>86.80</td>
<td>88.75</td>
<td>91.25</td>
<td>89.28</td>
</tr>
</tbody>
</table>

A natural idea arises again! We can use stronger denoising generative models: diffusion models.
Adversarial Purification based on Diffusion Models

Diffusion Models for Adversarial Purification

Solution: use a pre-trained diffusion model to denoise adversarial examples.

Rationale: injecting Gaussian noise to adversarial examples makes the distribution of adversarial examples approach a Gaussian distribution.
Adversarial Purification based on Diffusion Models

Diffusion Models for Adversarial Purification

Rationale: ideally, the distribution of the noise-injected adversarial examples should approach the distribution of noise-injected natural examples, as they all approach a Gaussian distribution. Then denoising noise-injected adversarial examples is equivalent to denoising noise-injected natural examples.
A key challenge: the choice of $t$.

1. if $t$ is too small, then the distribution of noise-injected adversarial examples cannot align well with the distribution of noise-injected natural examples, which means the adversarial noise cannot be fully removed.

2. if $t$ is too large, then the image will become too noisy, which loses the semantic information. Without the guidance of the original semantic information, the purified image may have a different semantic meaning.

Thus, how to retain the original semantic information after purification becomes a problem.
### Table 1. Standard accuracy and robust accuracy against AutoAttack $\ell_\infty (\epsilon = 8/255)$ on CIFAR-10, obtained by different classifier architectures. In our method, the diffusion timestep is $t^* = 0.1$.

<table>
<thead>
<tr>
<th>Method</th>
<th>Extra Data</th>
<th>Standard Acc</th>
<th>Robust Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>WideResNet-28-10</td>
<td>✓</td>
<td>89.36</td>
<td>59.96</td>
</tr>
<tr>
<td>(Zhang et al., 2020)</td>
<td>✓</td>
<td>88.25</td>
<td>62.11</td>
</tr>
<tr>
<td>(Wu et al., 2020)</td>
<td>✓</td>
<td>89.48</td>
<td>62.70</td>
</tr>
<tr>
<td>(Gowal et al., 2020)</td>
<td>✓</td>
<td>85.36</td>
<td>59.18</td>
</tr>
<tr>
<td>(Wu et al., 2020)</td>
<td>X</td>
<td>87.33</td>
<td>61.72</td>
</tr>
<tr>
<td>(Rebuffi et al., 2021)</td>
<td>X</td>
<td>87.50</td>
<td>65.24</td>
</tr>
<tr>
<td>Ours</td>
<td>X</td>
<td>89.02±0.21</td>
<td>70.64±0.39</td>
</tr>
<tr>
<td>WideResNet-70-16</td>
<td>✓</td>
<td>91.10</td>
<td>66.02</td>
</tr>
<tr>
<td>(Gowal et al., 2020)</td>
<td>✓</td>
<td>92.23</td>
<td>68.56</td>
</tr>
<tr>
<td>(Rebuffi et al., 2021)</td>
<td>X</td>
<td>85.29</td>
<td>59.57</td>
</tr>
<tr>
<td>(Gowal et al., 2020)</td>
<td>X</td>
<td>88.54</td>
<td>64.46</td>
</tr>
<tr>
<td>(Rebuffi et al., 2021)</td>
<td>X</td>
<td>88.74</td>
<td>66.60</td>
</tr>
<tr>
<td>Ours</td>
<td>X</td>
<td>90.07±0.97</td>
<td>71.29±0.55</td>
</tr>
</tbody>
</table>

### Table 3. Standard accuracy and robust accuracy against AutoAttack $\ell_\infty (\epsilon = 4/255)$ on ImageNet, obtained by different classifier architectures. In our method, the diffusion timestep is $t^* = 0.15$. (*Robust accuracy is directly reported from the respective paper.*)

<table>
<thead>
<tr>
<th>Method</th>
<th>Extra Data</th>
<th>Standard Acc</th>
<th>Robust Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>✓</td>
<td>62.56</td>
<td>31.06</td>
</tr>
<tr>
<td>(Engstrom et al., 2019)</td>
<td>X</td>
<td>55.62</td>
<td>26.95</td>
</tr>
<tr>
<td>(Wong et al., 2020)</td>
<td>X</td>
<td>64.02</td>
<td>37.89</td>
</tr>
<tr>
<td>(Salman et al., 2020)</td>
<td>X</td>
<td>67.38</td>
<td>35.51</td>
</tr>
<tr>
<td>Ours</td>
<td>X</td>
<td>67.79±0.43</td>
<td>40.93±1.96</td>
</tr>
<tr>
<td>WideResNet-50-2</td>
<td>✓</td>
<td>68.46</td>
<td>39.25</td>
</tr>
<tr>
<td>(Salman et al., 2020)</td>
<td>X</td>
<td>71.16±0.75</td>
<td>44.39±0.95</td>
</tr>
<tr>
<td>Ours</td>
<td>X</td>
<td>73.63±0.62</td>
<td>43.18±1.27</td>
</tr>
<tr>
<td>DeiT-S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Bai et al., 2021)†</td>
<td>X</td>
<td>66.50</td>
<td>35.50</td>
</tr>
<tr>
<td>Ours</td>
<td>X</td>
<td>73.63±0.62</td>
<td>43.18±1.27</td>
</tr>
</tbody>
</table>

Nie et al., Diffusion Models for Adversarial Purification. ICML, 2022.
Adversarial Purification based on Diffusion Models

DensePure: Understanding Diffusion Models for Adversarial Robustness

**Theoretical proof:** when the data density of natural samples is sufficiently high, the conditional density of the purified samples will also be high. This means that samples drawn from this conditional distribution have a high probability of recovering the original semantic meaning.

**Solution:** purify an input multiple times to get multiple purified samples, then feed them into the classifier and use majority voting to select the final prediction.
Adversarial Purification based on Diffusion Models

DensePure: Understanding Diffusion Models for Adversarial Robustness

DensePure: Understanding Diffusion Models for Adversarial Robustness

Figure 2: An illustration of the robust region $\mathcal{D}(x_0; t) = \bigcup_{i=1}^{3} \mathcal{D}_{sub}(x_i; t)$, where $x_0, x_1, x_2$ are samples with ground-truth label and $x_3$ is a sample with another label. $x_a = x_0 + \epsilon_a$ is an adversarial sample such that $P(x_a; t) = x_1 \neq x_0$ and thus the classification is correct but $x_a$ is not reversed back to $x_0$. $r_{sub}(x_0) < r(x_0)$ shows our claim that the union leads to a larger robust radius.
# Adversarial Purification based on Diffusion Models

## DensePure: Understanding Diffusion Models for Adversarial Robustness

<table>
<thead>
<tr>
<th>Method</th>
<th>Off-the-shelf</th>
<th>CIFAR-10</th>
<th>Certified Accuracy at $\epsilon(%)$</th>
<th>ImageNet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.25</td>
<td>0.5</td>
<td>0.75</td>
</tr>
<tr>
<td>PixelDP (Lecun et al., 2015)</td>
<td>X</td>
<td>(71.0) 22.0</td>
<td>(44.0) 2.0</td>
<td>-</td>
</tr>
<tr>
<td>RS (Cohen et al., 2019)</td>
<td>X</td>
<td>(75.0) 61.0</td>
<td>(75.0) 43.0</td>
<td>(65.0) 32.0</td>
</tr>
<tr>
<td>SmoothAdv (Salman et al., 2019)</td>
<td>X</td>
<td>(82.0) 68.0</td>
<td>(76.0) 54.0</td>
<td>(68.0) 41.0</td>
</tr>
<tr>
<td>Consistency (Jeong &amp; Shit, 2020)</td>
<td>X</td>
<td>(77.8) 68.8</td>
<td>(75.8) 58.1</td>
<td>(72.9) 48.5</td>
</tr>
<tr>
<td>MACER (Zhai et al., 2020)</td>
<td>X</td>
<td>(81.0) 71.0</td>
<td>(81.0) 59.0</td>
<td>(66.0) 46.0</td>
</tr>
<tr>
<td>Boosting (Horváth et al., 2021)</td>
<td>X</td>
<td>(83.4) 70.6</td>
<td>(76.8) 60.4</td>
<td>(71.6) 52.4</td>
</tr>
<tr>
<td>SmoothMix (Jeong et al., 2021)</td>
<td>✓</td>
<td>(77.1) 67.9</td>
<td>(77.1) 57.9</td>
<td>(74.2) 47.7</td>
</tr>
<tr>
<td>Denoised (Salman et al., 2020)</td>
<td>✓</td>
<td>(72.0) 56.0</td>
<td>(62.0) 41.0</td>
<td>(62.0) 28.0</td>
</tr>
<tr>
<td>Leo (Lg., 2021)</td>
<td>✓</td>
<td>60.0</td>
<td>42.0</td>
<td>28.0</td>
</tr>
<tr>
<td>Carlini (Carlini et al., 2021)</td>
<td>✓</td>
<td>(88.0) 73.8</td>
<td>(88.0) 56.2</td>
<td>(88.0) 41.6</td>
</tr>
<tr>
<td>Ours</td>
<td>✓</td>
<td>(87.6) 76.6</td>
<td>(87.6) 64.6</td>
<td>(87.6) 50.4</td>
</tr>
</tbody>
</table>

---

Conclusion

Use weak generative model + weak detector:

Use stronger generative models:

Use denoising generative models:

Use stronger denoising generative models:
5. Nie et al., Diffusion Models for Adversarial Purification. ICML, 2022
Part II: Adversarial Training

Let’s try train a robust model!!
How to actively prevent from adversarial attacks

Adversarial attack happens:

Test Data + Adversarial Perturbations

Train a robust model

Well-trained NN, Well-trained CNN, Well-trained Transformer

Predicted Labels
Adversarial Training: Two Main Directions

Adversarial Training

(From: Madry, et al., 2018)

- generate adversarial examples
  \[ \delta^{(t)} \leftarrow \text{Proj} \left[ \delta^{(t-1)} + \alpha \text{sign} \left( \nabla_{\theta} \ell \left( x_i + \delta_i^{(t-1)}, y_i; \theta \right) \right) \right] \] (1)

- train with adversarial examples
  \[ \text{argmin}_{\theta} \sum_i \ell \left( x_i + \delta_i^{(T)}, y_i; \theta \right) \] (2)

Instance-reweighted Adversarial Training

\[ \text{argmin}_{\theta} \sum_i \omega_i \ell \left( x_i + \delta_i^{(T)}, y_i; \theta \right) \] (3)

s.t. \( \omega_i \geq 0 \) and \( \sum_i \omega_i = 1 \)
Figure 3: A conceptual illustration of standard vs. adversarial decision boundaries. Left: A set of points that can be easily separated with a simple (in this case, linear) decision boundary. Middle: The simple decision boundary does not separate the $\ell_\infty$-balls (here, squares) around the data points. Hence there are adversarial examples (the red stars) that will be misclassified. Right: Separating the $\ell_\infty$-balls requires a significantly more complicated decision boundary. The resulting classifier is robust to adversarial examples with bounded $\ell_\infty$-norm perturbations.
Adversarial Training: Instance Equivalence

Trade-off between Adversarial Robustness and Accuracy (TRADES)

\[
\arg\min_{\theta} \sum_i CE(p(x_i; \theta), y) + \lambda \text{KL}(p(x_i; \theta) \parallel p(x_i + \delta_i^{(T)}; \theta))
\]

**Natural Error** encourages the natural error to be optimized by minimizing the risk w.r.t. the ground truth.

**Robust Error** encourages the output to be smooth in combating adversarial attack.
Adversarial Training: Instance Equivalence

Trade-off between Adversarial Robustness and Accuracy (TRADES)

Figure 1: **Left figure**: decision boundary learned by natural training method. **Right figure**: decision boundary learned by our adversarial training method, where the orange dotted line represents the decision boundary in the left figure. It shows that both methods achieve zero natural training error, while our adversarial training method achieves better robust training error than the natural training method.
Adversarial Training: Instance Equivalence

Trade-off between Adversarial Robustness and Accuracy (TRADES)

**Algorithm 1** Adversarial training by TRADES

1. **Input:** Step sizes $\eta_1$ and $\eta_2$, batch size $m$, number of iterations $K$ in inner optimization, network architecture parametrized by $\theta$
2. **Output:** Robust network $f_{\theta}$
3. Randomly initialize network $f_{\theta}$, or initialize network with pre-trained configuration
4. **repeat**
5. Read mini-batch $B = \{x_1, \ldots, x_m\}$ from training set
6. **for** $i = 1, \ldots, m$ (in parallel) **do**
7. $x_i' \leftarrow x_i + 0.001 \cdot N(0, I)$, where $N(0, I)$ is the Gaussian distribution with zero mean and identity variance
8. **for** $k = 1, \ldots, K$ **do**
9. $x_i' \leftarrow \Pi_{E(x_i, \varepsilon)}(\eta_1 \text{sign}(\nabla_{x_i'} L(f_{\theta}(x_i), f_{\theta}(x_i'))) + x_i')$, where $\Pi$ is the projection operator
10. **end for**
11. **end for**
12. $\theta \leftarrow \theta - \eta_2 \sum_{i=1}^{m} \nabla_{\theta}[L(f_{\theta}(x_i), y_i) + L(f_{\theta}(x_i), f_{\theta}(x_i'))/\lambda]/m$
13. **until** training converged

## Adversarial Training: Instance Equivalence

<table>
<thead>
<tr>
<th>Defense</th>
<th>Defense type</th>
<th>Under which attack</th>
<th>Dataset</th>
<th>Distance</th>
<th>$A_{\text{max}}(f)$</th>
<th>$A_{\text{rob}}(f)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[BRRG18]</td>
<td>gradient mask</td>
<td>[ACW18]</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>-</td>
<td>0%</td>
</tr>
<tr>
<td>[MLW$^+$18]</td>
<td>gradient mask</td>
<td>[ACW18]</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>-</td>
<td>5%</td>
</tr>
<tr>
<td>[DAL$^+$18]</td>
<td>gradient mask</td>
<td>[ACW18]</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>-</td>
<td>0%</td>
</tr>
<tr>
<td>[SKN$^+$18]</td>
<td>gradient mask</td>
<td>[ACW18]</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>-</td>
<td>9%</td>
</tr>
<tr>
<td>[NKMI7]</td>
<td>gradient mask</td>
<td>[ACW18]</td>
<td>CIFAR10</td>
<td>0.015 ($\ell_\infty$)</td>
<td>-</td>
<td>15%</td>
</tr>
<tr>
<td>[WSMK18]</td>
<td>robust opt.</td>
<td>FGSM$^{20}$ (PGD)</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>27.07%</td>
<td>23.54%</td>
</tr>
<tr>
<td>[MMS$^+$18]</td>
<td>robust opt.</td>
<td>FGSM$^{20}$ (PGD)</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>87.30%</td>
<td>47.04%</td>
</tr>
<tr>
<td>[ZSLG16]</td>
<td>regularization</td>
<td>FGSM$^{20}$ (PGD)</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>94.64%</td>
<td>0.15%</td>
</tr>
<tr>
<td>[KGB17]</td>
<td>regularization</td>
<td>FGSM$^{20}$ (PGD)</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>85.25%</td>
<td>45.89%</td>
</tr>
<tr>
<td>[RDV17]</td>
<td>regularization</td>
<td>FGSM$^{20}$ (PGD)</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>95.34%</td>
<td>0%</td>
</tr>
<tr>
<td>TRADES (1/$\lambda$ = 1)</td>
<td>regularization</td>
<td>FGSM$^{1,000}$ (PGD)</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>88.64%</td>
<td>48.90%</td>
</tr>
<tr>
<td>TRADES (1/$\lambda$ = 6)</td>
<td>regularization</td>
<td>FGSM$^{1,000}$ (PGD)</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>84.92%</td>
<td>56.43%</td>
</tr>
<tr>
<td>TRADES (1/$\lambda$ = 1)</td>
<td>regularization</td>
<td>FGSM$^{20}$ (PGD)</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>88.64%</td>
<td>49.14%</td>
</tr>
<tr>
<td>TRADES (1/$\lambda$ = 6)</td>
<td>regularization</td>
<td>FGSM$^{20}$ (PGD)</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>84.92%</td>
<td>56.61%</td>
</tr>
<tr>
<td>TRADES (1/$\lambda$ = 1)</td>
<td>regularization</td>
<td>DeepPool ($\ell_\infty$)</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>88.64%</td>
<td>59.10%</td>
</tr>
<tr>
<td>TRADES (1/$\lambda$ = 6)</td>
<td>regularization</td>
<td>DeepPool ($\ell_\infty$)</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>84.92%</td>
<td>61.38%</td>
</tr>
<tr>
<td>TRADES (1/$\lambda$ = 1)</td>
<td>regularization</td>
<td>LBFGSAttack</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>88.64%</td>
<td>84.41%</td>
</tr>
<tr>
<td>TRADES (1/$\lambda$ = 6)</td>
<td>regularization</td>
<td>LBFGSAttack</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>84.92%</td>
<td>81.58%</td>
</tr>
<tr>
<td>TRADES (1/$\lambda$ = 1)</td>
<td>regularization</td>
<td>MI-FGSM</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>88.64%</td>
<td>51.26%</td>
</tr>
<tr>
<td>TRADES (1/$\lambda$ = 6)</td>
<td>regularization</td>
<td>MI-FGSM</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>84.92%</td>
<td>57.95%</td>
</tr>
<tr>
<td>TRADES (1/$\lambda$ = 1)</td>
<td>regularization</td>
<td>C&amp;W</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>88.64%</td>
<td>84.03%</td>
</tr>
<tr>
<td>TRADES (1/$\lambda$ = 6)</td>
<td>regularization</td>
<td>C&amp;W</td>
<td>CIFAR10</td>
<td>0.031 ($\ell_\infty$)</td>
<td>84.92%</td>
<td>81.24%</td>
</tr>
</tbody>
</table>

Misclassification Aware Adversarial Training (MART)

Figure 1: The distinctive influence of misclassified examples ($S^-$) versus correctly classified ones ($S^+$) on the robustness of adversarial training. We test the white-box robustness of different strategies on either subset of examples: (a) using them directly for training ("not perturb"); (b) using weak attack (FGSM) in the inner maximization; and (c) using "regularized CE" loss in the outer minimization.
Adversarial Training: Instance Equivalence

Misclassification Aware Adversarial Training (MART)

$$\arg\min_\theta \sum_i \text{BCE}\left(p\left(x_i + \delta_i^{(T)}\right), y_i\right) + \lambda \text{KL}\left(p\left(x_i + \delta_i^{(T)}; \theta\right) || p(x_i; \theta)\right) \cdot \left(1 - p_{y_i}(x_i; \theta)\right)$$

**Term 1.** Commonly used CE loss with the margin term to improve the decision margin.

**Term 2.** Measuring if model predictions are consistent before and after adversarial perturbing.

**Term 3.** Emphasizing learning on misclassified examples.

Adversarial Training: Instance Equivalence

Misclassification Aware Adversarial Training (MART)

**Algorithm 1** Misclassification Aware adverSarial Training (MART)

1. **Input:** Training data $\{x_i, y_i\}_{i=1,\ldots,n}$, outer iteration number $T_O$, inner iteration number $T_I$, maximum perturbation $\epsilon$, step size for inner optimization $\eta_I$, step size for outer optimization $\eta_O$
2. **Initialization:** Standard random initialization of $h_\theta$
3. **for** $t = 1, \ldots, T_O$ **do**
4. **Uniformly** sample a minibatch of training data $B^{(t)}$
5. **for** $x_i \in B^{(t)}$ **do**
6. $x_i' = x_i + \epsilon \cdot \xi$, with $\xi \sim \mathcal{U}(-1, 1)$ # $\mathcal{U}$ is a uniform distribution
7. **for** $s = 1, \ldots, T_I$ **do**
8. $x_i' \leftarrow \Pi_{B_\epsilon(x_i)} \left( x_i' + \eta_I \cdot \text{sign}(\nabla_{x_i} CE(p(x_i', \theta), y_i)) \right)$ # $\Pi(\cdot)$ is the projection operator
9. **end for**
10. $\tilde{x}_i' \leftarrow x_i'$
11. **end for**
12. $\theta \leftarrow \theta - \eta_O \sum_{x_i \in B^{(t)}} \nabla_\theta \mathcal{L}(x_i, y_i, \tilde{x}_i'; \theta)$
13. **end for**
14. **Output:** Robust classifier $h_\theta$

**Adversarial Training: Instance Equivalence**

Table 2: White-box robustness (accuracy (%) on white-box test attacks) on MNIST and CIFAR-10.

<table>
<thead>
<tr>
<th>Defense</th>
<th>MNIST</th>
<th>CIFAR-10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Natural</td>
<td>FGSM</td>
</tr>
<tr>
<td>Standard</td>
<td>99.11</td>
<td>97.17</td>
</tr>
<tr>
<td>MMA</td>
<td>98.92</td>
<td>97.25</td>
</tr>
<tr>
<td>Dynamic</td>
<td>98.96</td>
<td>97.34</td>
</tr>
<tr>
<td>TRADES</td>
<td>99.25</td>
<td>96.67</td>
</tr>
<tr>
<td>MART</td>
<td>98.74</td>
<td>97.87</td>
</tr>
</tbody>
</table>

Table 3: Black-box robustness (accuracy (%) on black-box test attacks) on MNIST and CIFAR-10.

<table>
<thead>
<tr>
<th>Defense</th>
<th>MNIST</th>
<th>CIFAR-10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FGSM</td>
<td>PGD$^{10}$</td>
</tr>
<tr>
<td>Standard</td>
<td>96.12</td>
<td>95.73</td>
</tr>
<tr>
<td>MMA</td>
<td>96.11</td>
<td>95.94</td>
</tr>
<tr>
<td>Dynamic</td>
<td>97.60</td>
<td>96.25</td>
</tr>
<tr>
<td>TRADES</td>
<td>97.49</td>
<td>96.03</td>
</tr>
<tr>
<td>MART</td>
<td>97.77</td>
<td>96.96</td>
</tr>
</tbody>
</table>
Adversarial Training: Instance Equivalence

Friendly Adversarial Training (FAT)

- $X''$: Most adversarial data
- $X'$: Friendendy adversarial data (ours)
- $X$: Natural data

When adversarial data are wrongly predicted:

When adversarial data are correctly predicted:

Adversarial Training: Instance Equivalence

Friendly Adversarial Training (FAT)

\[
\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(\tilde{x}_i), y_i),
\]
\[
\tilde{x}_i = \arg \min_{\tilde{x} \in \mathcal{B}(x_i)} \ell(f(\tilde{x}), y_i) \quad \text{s.t.} \quad \ell(f(\tilde{x}), y_i) - \min_{y \in \mathcal{Y}} \ell(f(\tilde{x}), y) \geq \rho
\]

**Objective.** Minimize the adversarial loss given that some confident adversarial data has been found. This \( \tilde{x}_i \) could be regarded as a ‘friend’ among the adversaries.

**Constraint.** Ensures \( y_i \neq \arg \min_{y \in \mathcal{Y}} \ell(f(\tilde{x}), y) \) or \( \tilde{x} \) is misclassified.

**Constraint.** Ensures for \( \tilde{x} \), the wrong prediction is better than the desired prediction \( y_i \) by at least \( \rho \) in terms of the loss value.
Adversarial Training: Instance Equivalence

Friendly Adversarial Training (FAT)

Algorithm 1 PGD-$K$-$\tau$

```
Input: data $x \in \mathcal{X}$, label $y \in \mathcal{Y}$, model $f$, loss function $\ell$, maximum PGD step $K$, step $\tau$, perturbation bound $\epsilon$, step size $\alpha$
Output: $\tilde{x}$

$\tilde{x} \leftarrow x$
while $K > 0$ do
    if $\text{arg max}_i f(\tilde{x}) \neq y$ and $\tau = 0$ then
        break
    else if $\text{arg max}_i f(\tilde{x}) \neq y$ then
        $\tau \leftarrow \tau - 1$
    end if
    $\tilde{x} \leftarrow \Pi_{B[x, \epsilon]}(\alpha \text{sign}(\nabla_{\tilde{x}} \ell(f(\tilde{x}), y)) + \tilde{x})$
    $K \leftarrow K - 1$
end while
```

Algorithm 2 Friendly Adversarial Training (FAT)

```
Input: network $f_\theta$, training dataset $S = \{(x_i, y_i)\}_{i=1}^n$, learning rate $\eta$, number of epochs $T$, batch size $m$, number of batches $M$
Output: adversarially robust network $f_\theta$
for epoch = 1, . . . , $T$ do
    for mini-batch = 1, . . . , $M$ do
        Sample a mini-batch $\{(x_i, y_i)\}_{i=1}^m$ from $S$
        for $i = 1, . . . , m$ (in parallel) do
            Obtain adversarial data $\tilde{x}_i$ of $x_i$ by Algorithm 1
        end for
        $\theta \leftarrow \theta - \eta \frac{1}{m} \sum_{i=1}^m \nabla_{\theta} \ell(f_\theta(\tilde{x}_i), y_i)$
    end for
end for
```
## Friendly Adversarial Training (FAT)

### Table 1. Evaluations (test accuracy) of deep models (WRN-32-10) on CIFAR-10 dataset

<table>
<thead>
<tr>
<th>Defense</th>
<th>Natural</th>
<th>FGSM</th>
<th>PGD-20</th>
<th>C&amp;W∞</th>
<th>PGD-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Madry</td>
<td>87.30</td>
<td>56.10</td>
<td>45.80</td>
<td>46.80</td>
<td>-</td>
</tr>
<tr>
<td>CAT</td>
<td>77.43</td>
<td>57.17</td>
<td>46.06</td>
<td>42.28</td>
<td>-</td>
</tr>
<tr>
<td>DAT</td>
<td>85.03</td>
<td>63.53</td>
<td>48.70</td>
<td>47.27</td>
<td>-</td>
</tr>
<tr>
<td>FAT (ε\text{train} = 8/255)</td>
<td><strong>89.34 ± 0.221</strong></td>
<td>65.52 ± 0.355</td>
<td>46.13 ± 0.409</td>
<td>46.82 ± 0.517</td>
<td>45.31 ± 0.531</td>
</tr>
<tr>
<td>FAT (ε\text{train} = 16/255)</td>
<td>87.00 ± 0.203</td>
<td><strong>65.94 ± 0.244</strong></td>
<td><strong>49.86 ± 0.328</strong></td>
<td><strong>48.65 ± 0.176</strong></td>
<td><strong>49.56 ± 0.255</strong></td>
</tr>
</tbody>
</table>

Results of Madry, CAT and DAT are reported in (Wang et al., 2019). FAT has the same evaluations.

### Table 2. Evaluations (test accuracy) of deep models (WRN-34-10) on CIFAR-10 dataset

<table>
<thead>
<tr>
<th>Defense</th>
<th>Natural</th>
<th>FGSM</th>
<th>PGD-20</th>
<th>C&amp;W∞</th>
<th>PGD-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRADES (β = 1.0)</td>
<td>88.64</td>
<td>56.38</td>
<td>49.14</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FAT for TRADES (ε\text{train} = 8/255)</td>
<td><strong>89.94 ± 0.303</strong></td>
<td><strong>61.00 ± 0.418</strong></td>
<td><strong>49.70 ± 0.653</strong></td>
<td>49.35 ± 0.363</td>
<td>48.35 ± 0.240</td>
</tr>
<tr>
<td>TRADES (β = 6.0)</td>
<td>84.92</td>
<td>61.06</td>
<td>56.61</td>
<td></td>
<td>54.47</td>
</tr>
<tr>
<td>FAT for TRADES (ε\text{train} = 8/255)</td>
<td><strong>86.60 ± 0.548</strong></td>
<td><strong>61.97 ± 0.570</strong></td>
<td>55.98 ± 0.209</td>
<td>54.29 ± 0.173</td>
<td>55.34 ± 0.291</td>
</tr>
<tr>
<td>FAT for TRADES (ε\text{train} = 16/255)</td>
<td>84.39 ± 0.030</td>
<td>61.73 ± 0.131</td>
<td><strong>57.12 ± 0.233</strong></td>
<td>54.36 ± 0.177</td>
<td><strong>56.07 ± 0.155</strong></td>
</tr>
</tbody>
</table>

Results of TRADES (β = 1.0 and 6.0) are reported in (Zhang et al., 2019b). FAT for TRADES has the same evaluations.
Adversarial Training

(From: Madry, et al., 2018)

- **generate adversarial examples**
  \[
  \delta^{(t)} \leftarrow \text{Proj} \left[ \delta^{(t-1)} + \alpha \text{sign} \left( \nabla_{\theta} \ell \left( x_i + \delta^{(t-1)}_i, y_i; \theta \right) \right) \right]
  \] (1)

- **train with adversarial examples**
  \[
  \arg\min_{\theta} \sum_i \ell \left( x_i + \delta^{(t)}_i, y_i; \theta \right)
  \] (2)

Instance-reweighted Adversarial Training

\[
\arg\min_{\theta} \sum_i \omega_i \ell \left( x_i + \delta^{(T)}_i, y_i; \theta \right)
\] (3)

s.t. \( \omega_i \geq 0 \) and \( \sum_i \omega_i = 1 \)
Adversarial Training: Instance Reweighted

**Geometric Aware Instance-Reweighted Adversarial Training (GAIRAT)**

(From: Zhang, et al., 2021)

data point closer to the class boundary is less robust, requiring larger weight in training.

**The Least PGD Steps (LPS) $\kappa(x, y)$**

$$\omega(x, y) = \frac{1 + \tanh(\lambda + 5 \times (1 - 2 \times \kappa(x, y) / K))}{2}$$

(4)
Adversarial Training: Instance Reweighted

The least PGD steps (LPS)

Path-Dependency

- May change given the same start and end point
- Make the computation unstable

Discreteness

- Take only a few values
- Make the measurement ambiguous
The least PGD steps (LPS)

Path-Dependency
- May change given the same start and end point
- Make the computation unstable

Discreteness
- Take only a few values
- Make the measurement ambiguous
Adversarial Training: Instance Reweighted

Existing methods measuring the closeness are not very reliable: they are discrete and path-dependent. Here, we adopt the least PGD steps (LPS) as an example.
Adversarial Training: Instance Reweighted

Aiming at giving a geometric metric which are continuous and path-dependent.

- Bring the idea for the multiclass margin in the margin theory.

- Compute in a low dimensional embedding space with normalization.

Adversarial Training: Instance Reweighted

Multi-class Margin

\[ h_y(x; \theta) - \max_{j, j \neq y} h_j(x; \theta) \quad (4) \]

Probabilistic Margin

\[ p_y(x; \theta) - \max_{j, j \neq y} p_j(x; \theta) \quad (5) \]

- **Positive Value**: safe data with correct prediction.
- **Zero Value**: safe data on the decision boundary.
- **Negative Value**: critical data with wrong prediction.

Adversarial Training: Instance Reweighted

Multi-class Margin

\[ h_y(x; \theta) - \max_{j \neq y} h_j(x; \theta) \]  

(4)

Probabilistic Margin

\[ p_y(x; \theta) - \max_{j \neq y} p_j(x; \theta) \]  

(5)

- **Positive Value**: safe data with correct prediction.
- **Zero Value**: safe data on the decision boundary.
- **Negative Value**: critical data with wrong prediction.

Three Specifications

- **PM_adv**
  \[ \text{PM}_{\text{adv}} = p_y(x + \delta^{(T)}; \theta) - \max_{j \neq y} p_j(x + \delta^{(T)}; \theta) \]  

(4)

- **PM_nat**
  \[ \text{PM}_{\text{nat}} = p_y(x; \theta) - \max_{j \neq y} p_j(x; \theta) \]  

(5)

- **PM_dif**
  \[ \text{PM}_{\text{dif}} = p_y(x; \theta) - \max_{j \neq y} p_j(x + \delta^{(T)}; \theta) \]  

(6)

Adversarial Training: Instance Reweighted

Algorithm 1 MAIL: The Overall Algorithm.

Input: a network model with the parameters $\theta$; and a training dataset $S$ of size $n$.
Output: a robust model with parameters $\theta^*$.
1: for $e = 1$ to num_epoch do
2:    for $b = 1$ to num_batch do
3:        sample a mini-batch $\{(x_i, y_i)\}_{i=1}^m$ from $S$; \hspace{1cm} \triangleright \text{mini-batch of size } m.$
4:    for $i = 1$ to batch_size do
5:        $\delta_{i}^{(0)} = \xi$, with $\xi \sim U(0, 1)$;
6:    for $t = 1$ to $T$ do
7:        $\delta_{i}^{(t)} \leftarrow \text{Proj} \left[ \delta_{i}^{(t-1)} + \alpha \text{sign} \left( \nabla_\theta \ell(x_i + \delta_{i}^{(t-1)}, y_i; \theta) \right) \right]$;
8:    end for
9:        $w_{i}^{\text{ann}} = \text{sigmoid}(-\gamma(P_{M_{i}} - \beta))$;
10: end for
11: $\omega_i = M \times w_{i}^{\text{ann}} / \sum_{j} w_{j}^{\text{ann}}, \forall i \in [m]$; \hspace{1cm} $\triangleright \omega_i = 1$ during burn-in period.
12: $\theta \leftarrow \theta - \eta \nabla_\theta \sum_{i=1}^{m} \omega_i \ell(x_i + \delta_{i}^{(T)}, y_i; \theta) + \mathcal{R}(x_i, y_i; \theta)$;
13: end for
14: end for

- Either PM_nat, PM_adv, or PM_dif can be adopted in measuring the geometric distances.
- The sigmoid function is adopted in weight assignment.
- The learning objective of the form $\sum_{i} \omega_i \log P_{y_i}(x_i + \delta_{i}^{(T)}, \theta) + \mathcal{R}(x_i, y_i; \theta)$ is adopted.
Adversarial Training: Instance Reweighted

Table 2: Average accuracy (%) and standard deviation on CIFAR-10 dataset with ResNet-18.

<table>
<thead>
<tr>
<th></th>
<th>NAT</th>
<th>PGD</th>
<th>APGD</th>
<th>CW</th>
<th>AA</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT [29]</td>
<td>84.86</td>
<td>48.91</td>
<td>47.70</td>
<td>51.61</td>
<td>44.90</td>
</tr>
<tr>
<td></td>
<td>± 0.17</td>
<td>± 0.14</td>
<td>± 0.06</td>
<td>± 0.15</td>
<td>± 0.53</td>
</tr>
<tr>
<td>TRADES [45]</td>
<td>84.00</td>
<td>52.66</td>
<td>52.37</td>
<td>52.30</td>
<td>48.10</td>
</tr>
<tr>
<td></td>
<td>± 0.23</td>
<td>± 0.16</td>
<td>± 0.24</td>
<td>± 0.06</td>
<td>± 0.26</td>
</tr>
<tr>
<td>MART [40]</td>
<td>82.28</td>
<td>53.50</td>
<td>52.73</td>
<td>51.59</td>
<td>48.40</td>
</tr>
<tr>
<td></td>
<td>± 0.14</td>
<td>± 0.46</td>
<td>± 0.57</td>
<td>± 0.16</td>
<td>± 0.14</td>
</tr>
<tr>
<td>FAT [46]</td>
<td>87.97</td>
<td>46.78</td>
<td>46.68</td>
<td>49.92</td>
<td>43.90</td>
</tr>
<tr>
<td></td>
<td>± 0.15</td>
<td>± 0.12</td>
<td>± 0.16</td>
<td>± 0.26</td>
<td>± 0.82</td>
</tr>
<tr>
<td>AWP [42]</td>
<td>85.17</td>
<td>52.63</td>
<td>50.40</td>
<td>51.39</td>
<td>47.00</td>
</tr>
<tr>
<td></td>
<td>± 0.40</td>
<td>± 0.17</td>
<td>± 0.26</td>
<td>± 0.18</td>
<td>± 0.25</td>
</tr>
<tr>
<td>GAIRAT [47]</td>
<td>83.22</td>
<td>54.81</td>
<td>50.95</td>
<td>39.86</td>
<td>33.35</td>
</tr>
<tr>
<td></td>
<td>± 0.06</td>
<td>± 0.15</td>
<td>± 0.49</td>
<td>± 0.08</td>
<td>± 0.57</td>
</tr>
<tr>
<td>MAIL-AT</td>
<td>84.52</td>
<td>55.25</td>
<td>53.20</td>
<td>48.88</td>
<td>44.22</td>
</tr>
<tr>
<td></td>
<td>± 0.46</td>
<td>± 0.23</td>
<td>± 0.38</td>
<td>± 0.11</td>
<td>± 0.21</td>
</tr>
<tr>
<td>MAIL-TRADES</td>
<td>81.84</td>
<td>53.68</td>
<td>52.92</td>
<td>52.89</td>
<td>50.60</td>
</tr>
<tr>
<td></td>
<td>± 0.18</td>
<td>± 0.14</td>
<td>± 0.62</td>
<td>± 0.31</td>
<td>± 0.22</td>
</tr>
</tbody>
</table>
Conclusion

Use instance equivalent adversarial training:


Use instance-reweighted adversarial training:

Part III: Adversarial Detection

Let’s try detect adversarial data!!
How to passively prevent from adversarial attacks

Adversarial attack happens:

Test Data + Adversarial Perturbations

Discard the adversarial data

Input

Well-trained NN,
Well-trained CNN
Well-trained Transformer

Predicted Labels
Adversarial Detection

Detecting Adversarial Samples from Artifacts

Three possible situations of the adversarial sample $x^*$ lies in the data manifold:

(a) Two simple 2D submanifolds. (b) One submanifold has a 'pocket'. (c) Nearby 2D submanifolds.

Given the intuition that adversarial samples lie off the true data manifold, Feinman et al. devise two measures that can be used to detect adversarial samples:

- **Kernel Density:**
  \[
  \hat{K}(x, X_t) = \frac{1}{|X_t|} \sum_{x_i \in X_t} k_\sigma(\phi(x), \phi(x_i)),
  \]
  where $k_\sigma(x, y) \sim \exp(-\|x - y\|^2/\sigma^2)$.

- **Bayesian Uncertainty:**
  \[
  U(x^*) = \frac{1}{T} \sum_{i=1}^{T} \tilde{y}_i^* \tilde{y}_i^* - \left( \frac{1}{T} \sum_{i=1}^{T} \tilde{y}_i^* \right)^T \left( \frac{1}{T} \sum_{i=1}^{T} \tilde{y}_i^* \right)
  \]

Feinman et al., Detecting Adversarial Samples from Artifacts. Arxiv, 2017.
Adversarial Detection

Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality

Ma et al. consider expansion-based measures of intrinsic dimensionality as an alternative density measure.

- **Local Intrinsic Dimensionality (LID):**
  \[
  \text{LID}_F(r) = \lim_{\epsilon \to 0} \frac{\ln(F((1 + \epsilon) \cdot r)/F(r))}{\ln(1 + \epsilon)} = \frac{r \cdot F'(r)}{F(r)},
  \]
  where \( r \in \mathbb{R} \) denotes the distance from \( x \) to other data samples.

- **Maximum Likelihood Estimation:**
  \[
  \text{LID}(x) = -\left( \frac{1}{k} \sum_{i=1}^{k} \log \frac{r_i(x)}{r_k(x)} \right)^{-1}
  \]

Kernel density is not effective for the detection when attack falls in local adversarial regions!
Lee et al. propose to detect abnormal test samples including OOD and adversarial ones using a pre-trained softmax neural classifier without re-training.

- **Obtain the Mahalanobis distance-based score** based on the classifier:
  \[ M(x) = \max_c - (f(x) - \mu_c)^T \Sigma^{-1} (f(x) - \mu_c) \]

- Additional techniques to improve detection performance:
  - **Input pre-processing:** Add a small controlled noise to a test sample to make in- and out-of-distribution samples more separable:
    \[ \tilde{x} = x + \epsilon \text{sign}(\nabla_x M(x)) = x - \epsilon \text{sign} \left( \nabla_x (f(x) - \mu_c)^T \Sigma^{-1} (f(x) - \mu_c) \right) \]
  - **Feature ensemble:** Measure and combine the confidence scores from not only the final features but also the other low-level features in DNNs to further improve the performance.
LiBRe: A Practical Bayesian Approach to Adversarial Detection

- Deng et al. leverage Bayesian neural networks (BNNs) and propose **Lightweight Bayesian Refinement (LiBRe)** for detecting adversarial test samples.

- Given a pre-trained DNN, LiBRe converts its last few layers to be Bayesian and uses the uncertainty as a metric:

\[
U(x) = \frac{1}{T - 1} \left[ \sum_{t=1}^{T} \|z^{(t)}\|_2^2 - T \left( \frac{1}{T} \sum_{t=1}^{T} \|z^{(t)}\|_2^2 \right)^2 \right]
\]

where \(z^{(t)}\) is the \(t\)-th logits of the BNN.
Adversarial Detection: Distributional Difference?

The consensus in related works: Two-sample test is unaware of adversarial attacks.

A natural question: are natural and adversarial data really from different distributions? Answer: affirmative.

Our investigation: the previous use of MMD test on the purpose missed three key factors.

Factor 1. Limited representation power of the Gaussian kernel.

Factor 2. The overlook of the optimization for the used kernel parameters.

Factor 3. Non-IID adversarial data break a basic assumption of the MMD test.
Different semantic meanings between natural and adversarial examples

Visualization of the difference using t-SNE.

Kernel Architecture (designed for testing):

\[ k_\omega(x, y) = [(1 - \varepsilon_0)s_f(x, y) + \varepsilon_0]q(x, y) \]

- \( s_f(x, y) = \kappa(\phi_p(x), \phi_p(y)) \) is a deep kernel function;
- \( \phi_p \) is the second to the last fully connected layer in \( \hat{f} \); (we use \( \phi_p \) to extract semantic features.)
- \( \kappa \) is the Gaussian kernel with the optimized bandwidth \( \sigma_{\phi_p} \);
- \( q(x, y) \) is the Gaussian kernel with bandwidth \( \sigma_q \);
- \( \varepsilon_0 \in (0,1) \);
- \( q(x, y) \) and \( \varepsilon_0 \) ensure that \( k_\omega(x, y) \) is a characteristic kernel.
- The set of parameters of \( k_\omega \) is \( \omega = \{ \varepsilon_0, \sigma_{\phi_p}, \sigma_q \} \).

Discrepancy of MMD value between different layers’ outputs in \( \hat{f} \).
Adversarial Detection from the score

Adversarial Detection with Expected Perturbation Score

- Intuition from score function $\nabla_x \log p(x)$
  - Score $\nabla_x \log p(x)$ represents the momentum of the sample towards high density areas of natural data (Song et al., 2019)
  - A lower score norm indicates the sample is closer to the high-density areas of natural data

- Consider the score of data perturbed by a diffusion process due to the score of original natural/adversarial data is not distinguished
  - Most natural samples have lower score norms than adversarial samples, but they are very sensitive to the timesteps due to the significant overlap across all timesteps

One score is useful but not effective enough!
Adversarial Detection from the score

**Adversarial Detection with Expected Perturbation Score**

Computing expected perturbation score (EPS) using a pre-trained score model

- Add perturbations to a set of **natural images** and a **test image** following a diffusion process with time step $T^*$ and obtain their EPSs via the score model.

Zhang et al., Detecting Adversarial Data by Probing Multiple Perturbations Using Expected Perturbation Score. ICML, 2023.
Can we detect the single-instance-based difference?

In the second work, we detect adversarial data using batches instead of instances (like many papers did). However, we have a question:

Can we do the single-instance-based adversarial data detection perfectly?

In our recent work [1], we find that the answer is NO. Because adversarial data and natural data are too close to be perfectly distinguished. We provide an impossibility theorem in [1] to support this answer.

Thank you

fengliu.ml@gmail.com
https://fengliu90.github.io/