Analyzing AI Model Internals for Debugging and Adversarial Sample Attack Detection

Xiangyu Zhang @ ETH, 2018
AI Driven Computing

- AI Models are becoming an integral part of modern computing
  - Autonomous vehicles, Apple Face ID, iRobots, Cotana, and computer games

- AI Models are shared/reused just like software components
  - Python face recognition package
AI Driven System Engineering

Tuning / Debugging / Optimization

Evaluation

Implementation
AI Models Are Prone to Bugs and Vulnerabilities Just Like Software Components

- Traditional engineering bugs
  - Coding bugs, data cleaning, mis-behaved data partitioning, improper data augmentation

- Model bugs – *misconducts in the AI model engineering process leading to undesirable consequences*
  - **Root causes**: biased training data, defective model structure, hyper-parameter(s), optimization algorithms, batch size, loss function, activation function(s)
  
  - **Symptoms**: low model accuracy, vulnerable to adversarial sample attacks

- E.g., State-of-the-art pre-trained models can only achieve 80% accuracy on an ImageNet classification challenge; 73% accuracy on Children’s Book Test challenge.
  - Numerous attacks on AI systems (Trojaning, perturbation, and patching attacks)
Debugging AI Models

• Debugging is hard

  • DNNs are not human understandable/interpretable
    • Each neuron denotes some abstract feature

  • Lack of scientific way of locating the root causes
    • Trial-and-error

• Unclear how to fix bugs
  • Cannot directly change weight values
  • Cannot train with failure inducing inputs
Theme of the Talk

• Leveraging what we have learned in program analysis and software engineering to open the box

• Outline
  • MODE: Automated Neural Network Model Debugging via State Differential Analysis and Input Selection (FSE’18)
  • Aml: Attacks Meet Interpretability, Attribute-steered Detection of Adversarial Samples (NIPS’18)
AI Model Bugs

- Input related bugs
  - Biased training inputs
    - Overfitting and underfitting

- Inclusion of problematic inputs in the training set leads to difficulty of convergence
  - Training a model to evaluate propositional logic expression

\[ T \land F \lor T \]
AI Model Bugs

• Input related bugs
  • Biased training inputs
    • Overfitting and underfitting
  • Inclusion of problematic inputs in the training set leads to difficulty of convergence
    • Training a model to evaluate propositional logic expression

• Problematic input embedding (for RNN models)
  • Similar embeddings do not entail similar semantics
    • “new” and “create”

• Structural bugs
  • Redundant/insufficient layers/neurons
  • In-effective structures
    • Forget gates in (LSTM) do not retain/throw-away certain contextual information
  • Suboptimal setting of reward values leading to extremely long training time in reinforcement learning
Drone with a suboptimal reward setting (two weeks training)
After fixing the reward setting (four hours training)
AI Model Bugs

• Input related bugs
  • Biased training inputs
    • Overfitting and underfitting
  • Inclusion of problematic inputs in the training set leads to difficulty of convergence
  • Problematic input embedding (for RNN models)
    • Embedding of training inputs does not provide good coverage
    • Similar embeddings do not entail similar semantics
      • General embeddings may not work well for domain-specific applications

• Structural bugs
  • Redundant/insufficient layers/neurons
  • In-effective structures
    • Forget gates in (LSTM) do not retain the appropriate contextual information
  • Suboptimal setting of reward values leading to extremely long training time in reinforcement learning
Overfitting and Underfitting Bugs

• We say a model has an underfitting bug if for some label, both training and test accuracies are lower than a threshold $t$
  • $t$ is domain specific

• We say a model has an overfitting bug if for some label, its training accuracy is higher than test accuracy by at least some threshold
Existing Works

- Applying pre-defined image operations on existing data points
  - Rotation, mirror, clip, brightness change etc.
- Using generative models to collect new data points
  - Variational Autoencoder (VAE) or Generative Adversarial Net (GAN)
  - Trend of using GAN
Using GAN is Not That Effective

- Use 14 GANs downloaded from various sources for MNIST to generate inputs
- For each GAN, randomly select 40,000 generated inputs as additional training data to fix a MNIST model that has an underfitting bug for digit 5 (only 74% accuracy)
- 7 GANs fail to improve either digit 5 or the whole model, 4 improve the model but not digit 5, and only 3 can improve both (digit 5 to 83% after 1 hour of training)
  - MODE can improve to 94% in 5 mins
- Root Cause: *does not consider the reasons why a NN misbehaves*
What Have We Learned in Software Debugging
MODE: AI Model State Differential Analysis and Input Selection
Overview

Dataset → Model → Results

Selected New Dataset

Newly Generated Dataset

Input Selection

Differential Heatmap
Heat Map

- A matrix representing the importance of each neuron in a hidden layer
  - One heat map for each hidden layer
  - Each neuron denotes some abstract feature
- Visualization of heat-map
  - One pixel denotes the importance of one neuron/feature (to the output)
  - Red – *positive importance*
  - Blue – *negative importance*
A Motivating Example

• Assume a model that has an underfitting bug for label 1 (other numbers misclassified to 1)

\[
\begin{array}{c|c|c}
4 & 8 \\
\end{array}
\]

• Benign heat-map

\[
\begin{array}{c}
\text{Benign heat-map} \\
\end{array}
\]

• Faulty heat-map

\[
\begin{array}{c}
\text{Faulty heat-map} \\
\end{array}
\]

• Differential heat-map

\[
\begin{array}{c}
\text{Differential heat-map} \\
\end{array}
\]

• Selected samples

\[
\begin{array}{c|c}
1 & 1 \\
\end{array}
\]

\[
\begin{array}{c}
\text{Selected samples} \\
\end{array}
\]

• samples should not be selected

\[
\begin{array}{c|c}
\text{x} & 1 \\
\end{array}
\]

\[
\begin{array}{c}
\text{samples should not be selected} \\
\end{array}
\]
Heat-map Computation

• Cannot use gradient information -- *how much output changes can be induced by weight value changes*
  • Gradients are with-respect-to weight values, whereas importance is with-respect-to features (neurons)
  • Importance measures how much influence a feature has on the classification result of an output label
    • Important features may not have weight values of a large gradient
Heat-map Computation

- Feature model: part of the original model (including weights) + a newly trained SoftMax layer (used for prediction)

- The weights of the last layer measure the importance of individual features (for the prediction)

- The normalized weights for an output label of the newly trained SoftMax layer is the **Heatmap**

- Normalize the weights to [-1,1] with the absolute values denoting the importance and the signs denoting positive/negative importance
Differential Heat-maps for Under-fitting

• Two kinds of root causes
  (1) Extracted features cannot fully represent the uniqueness of the target label
    • Selecting cases that can emphasize the uniqueness
  
  (2) Cases mis-classified to the target label share common features with some cases of the target label
    • Not to select such cases
• Two corresponding kinds of differential heap-maps
  • For (1),
    • $DHM_L[f] = HM_L[f] - HM_k[f]$, when $|HM_L[f] - HM_k[f]|$ is minimal for $k! = L$
    • $DHM_L[f]$ represents the minimal similarity of feature $f$ regarding the target label $L$ and some other output label, larger values mean more uniqueness
Example

- **DHM** shows the importance features to differentiate two output labels, as shown in the example.

- Selecting cases with strong presence in the red zones and weak presence in the blue zones would help improve uniqueness.
Differential Heat-maps for Under-fitting

- Two kinds of root causes
  1. Extracted features cannot fully represent the uniqueness of the target label
     - Selecting cases that can emphasize the uniqueness
  2. Cases mis-classified to the target label share common features with some cases for the target label
     - Selecting samples that the model to disambiguate

- Two corresponding kinds of differential heap-maps
  - For (1), ...
  - For (2),
    - \( D\text{HM}_{L[f]} = H\text{M}_{\text{misclassified as } L[f]} - H\text{M}_{\text{correctly classified as } L[f]} \)
    - A large (red) value indicates the feature is critical for misclassification
Example

- $HM_1$
- $HM_{\text{mis-classified as 1}}$
- $DHM_1$

- **$DHM$** shows the confusing features
- Selecting cases that avoid the red areas and cover the blue areas will benefit
Differential Heat-map for Over-fitting

• Root cause – *narrowly scoped training data, model too large, or training with too many epochs*
  • More diverse training data for the target label is needed

• $DHM_L[f] = \max_k (HM_{L \text{ misclassified as } k}[f] - HM_{correctly classified as } L[f])$
  • A large value denotes that the feature is responsible for misclassifying $L$ to $k$
  • We need more samples that has this feature
Example

- The red regions in the **DHM** denote the features helpful for generalization, the blue regions denote the overfitted features.

- Larger-sized 0 are needed
Input Selection

• For each new input $i$, we feed it to the feature model (without running through the output layer) to acquire a feature value vector $V$.

  \[ \text{score} = V \cdot DHM \]

• $DHM$ is a vector pointing to the most promising direction
Evaluation

• RQ1: How effective and efficient is MODE in fixing model bugs?

• RQ2: How does MODE compare to using random samples or faulty samples to fix model bugs?

• RQ3: What is the impact of different parameters?
Experiment One

• Three data sets
  • Digit recognition (MNIST), Fashion-icon recognition (FM), Object recognition (CIFAR)
• For each data set, we have downloaded multiple models
  • Total 20 models
  • 20k-20M weight values
• Training with batches of 2000 samples, capped at 20,000 samples and 4 hours for small models, and 40,000 samples and 24 hours for large models
• Partition an original data set 30% training, 10% validation, 10% test, 50% bug fixing
• Select one UF and one OF for each model
  • UF: the output label with the lowest training and test accuracy
  • OF: the label with good training accuracy but the lowest test accuracy
Effectiveness: MNIST

<table>
<thead>
<tr>
<th>Model (Size)</th>
<th>Bug Type</th>
<th>Accuracy</th>
<th>Accuracy Mode</th>
<th>Randomly Selecting GAN Samples</th>
<th>Failing Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Model</td>
<td>Label</td>
<td>#Samples</td>
<td>Time</td>
</tr>
<tr>
<td>MNIST-1</td>
<td>Under-fitting</td>
<td>88%</td>
<td>74%</td>
<td>500+1500</td>
<td>5m</td>
</tr>
<tr>
<td>7k</td>
<td>Over-fitting</td>
<td>84%</td>
<td>81%</td>
<td>500+1500</td>
<td>6m</td>
</tr>
<tr>
<td>MNIST-2</td>
<td>Under-fitting</td>
<td>89%</td>
<td>72%</td>
<td>500+1500</td>
<td>5m</td>
</tr>
<tr>
<td>185k</td>
<td>Over-fitting</td>
<td>84%</td>
<td>76%</td>
<td>500+1500</td>
<td>5m</td>
</tr>
<tr>
<td>MNIST-3</td>
<td>Under-fitting</td>
<td>86%</td>
<td>76%</td>
<td>500+1500</td>
<td>5m</td>
</tr>
<tr>
<td>185k</td>
<td>Over-fitting</td>
<td>84%</td>
<td>78%</td>
<td>500+1500</td>
<td>5m</td>
</tr>
<tr>
<td>MNIST-4</td>
<td>Under-fitting</td>
<td>86%</td>
<td>78%</td>
<td>500+1500</td>
<td>5m</td>
</tr>
<tr>
<td>185k</td>
<td>Over-fitting</td>
<td>84%</td>
<td>74%</td>
<td>500+1500</td>
<td>5m</td>
</tr>
<tr>
<td>MNIST-5</td>
<td>Under-fitting</td>
<td>82%</td>
<td>77%</td>
<td>500+1500</td>
<td>5m</td>
</tr>
<tr>
<td>122k</td>
<td>Over-fitting</td>
<td>85%</td>
<td>77%</td>
<td>500+1500</td>
<td>5m</td>
</tr>
<tr>
<td>MNIST-6</td>
<td>Under-fitting</td>
<td>84%</td>
<td>72%</td>
<td>1000+3000</td>
<td>10m</td>
</tr>
<tr>
<td>244k</td>
<td>Over-fitting</td>
<td>84%</td>
<td>74%</td>
<td>1000+3000</td>
<td>9m</td>
</tr>
<tr>
<td>MNIST-7</td>
<td>Under-fitting</td>
<td>87%</td>
<td>77%</td>
<td>1000+3000</td>
<td>9m</td>
</tr>
<tr>
<td>185k</td>
<td>Over-fitting</td>
<td>85%</td>
<td>72%</td>
<td>1000+3000</td>
<td>9m</td>
</tr>
<tr>
<td>MNIST-8</td>
<td>Under-fitting</td>
<td>86%</td>
<td>73%</td>
<td>1000+3000</td>
<td>10m</td>
</tr>
<tr>
<td>185k</td>
<td>Over-fitting</td>
<td>84%</td>
<td>73%</td>
<td>1000+3000</td>
<td>12m</td>
</tr>
<tr>
<td>MNIST-9</td>
<td>Under-fitting</td>
<td>84%</td>
<td>73%</td>
<td>1000+3000</td>
<td>9m</td>
</tr>
<tr>
<td>257k</td>
<td>Over-fitting</td>
<td>84%</td>
<td>72%</td>
<td>1000+3000</td>
<td>9m</td>
</tr>
</tbody>
</table>
Effectiveness: Fashion-MNIST

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FM-1</td>
<td>Under-fitting</td>
<td>88%</td>
<td>80%</td>
<td>500+1500</td>
<td>5m</td>
<td>93%</td>
<td>90%</td>
<td>2(10)</td>
<td>84%</td>
<td>88%</td>
<td>20000</td>
</tr>
<tr>
<td>FM-2</td>
<td>Under-fitting</td>
<td>85%</td>
<td>77%</td>
<td>500+1500</td>
<td>5m</td>
<td>95%</td>
<td>95%</td>
<td>2(10)</td>
<td>89%</td>
<td>88%</td>
<td>20000</td>
</tr>
<tr>
<td>1.2M</td>
<td>Over-fitting</td>
<td>87%</td>
<td>74%</td>
<td>500+1500</td>
<td>5m</td>
<td>94%</td>
<td>94%</td>
<td>2(10)</td>
<td>90%</td>
<td>84%</td>
<td>20000</td>
</tr>
<tr>
<td>FM-3</td>
<td>Under-fitting</td>
<td>87%</td>
<td>72%</td>
<td>500+1500</td>
<td>10m</td>
<td>93%</td>
<td>91%</td>
<td>2(10)</td>
<td>89%</td>
<td>78%</td>
<td>20000</td>
</tr>
<tr>
<td>3.2M</td>
<td>Over-fitting</td>
<td>85%</td>
<td>69%</td>
<td>500+1500</td>
<td>9m</td>
<td>93%</td>
<td>93%</td>
<td>2(10)</td>
<td>88%</td>
<td>88%</td>
<td>20000</td>
</tr>
<tr>
<td>FM-4</td>
<td>Under-fitting</td>
<td>86%</td>
<td>73%</td>
<td>500+1500</td>
<td>5m</td>
<td>92%</td>
<td>94%</td>
<td>3(10)</td>
<td>83%</td>
<td>80%</td>
<td>20000</td>
</tr>
<tr>
<td>765k</td>
<td>Over-fitting</td>
<td>85%</td>
<td>74%</td>
<td>500+1500</td>
<td>5m</td>
<td>91%</td>
<td>92%</td>
<td>1(10)</td>
<td>88%</td>
<td>80%</td>
<td>20000</td>
</tr>
<tr>
<td>FM-5</td>
<td>Under-fitting</td>
<td>87%</td>
<td>80%</td>
<td>500+1500</td>
<td>5m</td>
<td>92%</td>
<td>92%</td>
<td>2(10)</td>
<td>87%</td>
<td>86%</td>
<td>20000</td>
</tr>
<tr>
<td>113k</td>
<td>Over-fitting</td>
<td>83%</td>
<td>73%</td>
<td>500+1500</td>
<td>5m</td>
<td>92%</td>
<td>93%</td>
<td>2(10)</td>
<td>83%</td>
<td>82%</td>
<td>20000</td>
</tr>
<tr>
<td>FM-6</td>
<td>Under-fitting</td>
<td>89%</td>
<td>81%</td>
<td>500+1500</td>
<td>5m</td>
<td>93%</td>
<td>95%</td>
<td>2(10)</td>
<td>91%</td>
<td>91%</td>
<td>20000</td>
</tr>
<tr>
<td>26M</td>
<td>Over-fitting</td>
<td>82%</td>
<td>74%</td>
<td>500+1500</td>
<td>9m</td>
<td>92%</td>
<td>94%</td>
<td>3(10)</td>
<td>85%</td>
<td>83%</td>
<td>20000</td>
</tr>
</tbody>
</table>
Effectiveness: CIFAR

<table>
<thead>
<tr>
<th>Model</th>
<th>Type</th>
<th>Test Set</th>
<th>Train Time</th>
<th>Overfit</th>
<th>Underfit</th>
<th>Training Time</th>
<th>Val Time</th>
<th>Test Acc</th>
<th>Val Acc</th>
<th>Overfit</th>
<th>Underfit</th>
<th>Training Time</th>
<th>Val Time</th>
<th>Test Acc</th>
<th>Val Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-1</td>
<td>Under-fitting</td>
<td>79%</td>
<td>64%</td>
<td>500+1500</td>
<td>6m</td>
<td>88%</td>
<td>89%</td>
<td>0(3)</td>
<td>74%</td>
<td>64%</td>
<td>40000</td>
<td>1h14m</td>
<td>79%</td>
<td>66%</td>
<td></td>
</tr>
<tr>
<td>62k</td>
<td>Over-fitting</td>
<td>79%</td>
<td>65%</td>
<td>500+1500</td>
<td>7m</td>
<td>92%</td>
<td>91%</td>
<td>0(3)</td>
<td>82%</td>
<td>63%</td>
<td>40000</td>
<td>1h21m</td>
<td>80%</td>
<td>68%</td>
<td></td>
</tr>
<tr>
<td>CIFAR-2</td>
<td>Under-fitting</td>
<td>84%</td>
<td>76%</td>
<td>500+1500</td>
<td>15m</td>
<td>91%</td>
<td>90%</td>
<td>0(3)</td>
<td>80%</td>
<td>74%</td>
<td>40000</td>
<td>2h40m</td>
<td>81%</td>
<td>82%</td>
<td></td>
</tr>
<tr>
<td>0.97M</td>
<td>Over-fitting</td>
<td>83%</td>
<td>72%</td>
<td>500+1500</td>
<td>21m</td>
<td>88%</td>
<td>89%</td>
<td>0(3)</td>
<td>88%</td>
<td>79%</td>
<td>40000</td>
<td>2h50m</td>
<td>85%</td>
<td>78%</td>
<td></td>
</tr>
<tr>
<td>CIFAR-3</td>
<td>Under-fitting</td>
<td>82%</td>
<td>78%</td>
<td>500+1500</td>
<td>30m</td>
<td>91%</td>
<td>90%</td>
<td>0(3)</td>
<td>81%</td>
<td>78%</td>
<td>40000</td>
<td>4h10m</td>
<td>83%</td>
<td>74%</td>
<td></td>
</tr>
<tr>
<td>1.7M</td>
<td>Over-fitting</td>
<td>86%</td>
<td>83%</td>
<td>500+1500</td>
<td>24m</td>
<td>93%</td>
<td>92%</td>
<td>0(3)</td>
<td>84%</td>
<td>74%</td>
<td>40000</td>
<td>4h9m</td>
<td>80%</td>
<td>72%</td>
<td></td>
</tr>
<tr>
<td>CIFAR-4</td>
<td>Under-fitting</td>
<td>84%</td>
<td>74%</td>
<td>1000+3000</td>
<td>12h40m</td>
<td>92%</td>
<td>93%</td>
<td>0(3)</td>
<td>87%</td>
<td>75%</td>
<td>38000</td>
<td>24h</td>
<td>86%</td>
<td>74%</td>
<td></td>
</tr>
<tr>
<td>20M</td>
<td>Over-fitting</td>
<td>87%</td>
<td>78%</td>
<td>1000+3000</td>
<td>12h9m</td>
<td>91%</td>
<td>92%</td>
<td>0(3)</td>
<td>90%</td>
<td>77%</td>
<td>38000</td>
<td>24h</td>
<td>87%</td>
<td>77%</td>
<td></td>
</tr>
<tr>
<td>CIFAR-5</td>
<td>Under-fitting</td>
<td>88%</td>
<td>79%</td>
<td>1000+3000</td>
<td>10h</td>
<td>92%</td>
<td>94%</td>
<td>0(3)</td>
<td>85%</td>
<td>78%</td>
<td>40000</td>
<td>24h</td>
<td>88%</td>
<td>78%</td>
<td></td>
</tr>
<tr>
<td>20M</td>
<td>Over-fitting</td>
<td>86%</td>
<td>79%</td>
<td>1000+3000</td>
<td>9h40m</td>
<td>93%</td>
<td>94%</td>
<td>0(3)</td>
<td>88%</td>
<td>78%</td>
<td>40000</td>
<td>24h</td>
<td>86%</td>
<td>73%</td>
<td></td>
</tr>
</tbody>
</table>
Effectiveness: Experiment Two

- Three new large data sets and models: face recognition (FR), objection detection CelebA (OD), age classification (AC)
- No available GANs

### Table 2: Accuracy Improvement without GANs

<table>
<thead>
<tr>
<th>Model</th>
<th>Bug</th>
<th>Original</th>
<th>MODE</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MAcc</td>
<td>LAcc</td>
<td>MAcc</td>
</tr>
<tr>
<td>FR</td>
<td>OF</td>
<td>76%</td>
<td>65%</td>
<td>88%</td>
</tr>
<tr>
<td>2.1M</td>
<td>UF</td>
<td>72%</td>
<td>64%</td>
<td>85%</td>
</tr>
<tr>
<td>OD</td>
<td>OF</td>
<td>83%</td>
<td>74%</td>
<td>89%</td>
</tr>
<tr>
<td>3.2M</td>
<td>UF</td>
<td>82%</td>
<td>75%</td>
<td>88%</td>
</tr>
<tr>
<td>AC</td>
<td>OF</td>
<td>33%/44%</td>
<td>13%/22%</td>
<td>46%/60%</td>
</tr>
<tr>
<td>30M</td>
<td>UF</td>
<td>25%/36%</td>
<td>11%/20%</td>
<td>42%/52%</td>
</tr>
</tbody>
</table>
Experiment Three: Improving Pre-trained Models

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Original Acc.</th>
<th># Samples</th>
<th>MODE Acc.</th>
<th>Random Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>MNIST-10 [23]</td>
<td>95.2%</td>
<td>2000</td>
<td>97.4%</td>
<td>94.8%</td>
</tr>
<tr>
<td></td>
<td>MNIST-11 [23]</td>
<td>93.4%</td>
<td>2000</td>
<td>96.8%</td>
<td>94.3%</td>
</tr>
<tr>
<td>Fashion</td>
<td>FM-7 [15]</td>
<td>87.6%</td>
<td>2000</td>
<td>92.3%</td>
<td>88.9%</td>
</tr>
<tr>
<td>MNIST</td>
<td>FM-8 [15]</td>
<td>91.6%</td>
<td>2000</td>
<td>92.6%</td>
<td>88.5%</td>
</tr>
<tr>
<td>CIFAR</td>
<td>CIFAR-6 [5]</td>
<td>87.3%</td>
<td>4000</td>
<td>93.2%</td>
<td>87.3%</td>
</tr>
<tr>
<td>CIFAR</td>
<td>CIFAR-7 [5]</td>
<td>88.4%</td>
<td>4000</td>
<td>92.8%</td>
<td>88.2%</td>
</tr>
</tbody>
</table>
Sample Ratios
Theme of the Talk

• Leveraging what we have learned in program analysis and software engineering to open the box

• Outline
  • MODE: Automated Neural Network Model Debugging via State Differential Analysis and Input Selection (FSE’18)
  • Aml: Attacks Meet Interpretability, Attribute-steered Detection of Adversarial Samples (NIPS’18)
Adversarial Samples

- Adversarial samples are model inputs generated by adversaries to fool neural networks (i.e., unexpected prediction results).
Existing Adversarial Attacks

- Patching
  - Restricted area to manipulate pixels
  - Utilize semantics of input space

- Pervasive perturbations
  - Full access to pixel alteration
  - Different distance metrics: $L_0$, $L_2$, $L_\infty$

$$\Delta(x, x') = ||x - x'||_p = \left( \sum_{i=1}^{n} |x_i - x'_i|^p \right)^{\frac{1}{p}}$$
Different Attacks

a. Original
b. Patch
c. Glasses
d. C&W₀
e. C&W₂
f. C&W∞
g. FGSM
h. BIM

Targeted
Untargeted
Understanding Adversarial Samples

- Idea: is the classification result of a model mainly based on human perceptible attributes?
Architecture of AmI

*Attribute witness: learned features that correspond to human perceptible attributes
Challenges

• Are there correspondences between attributes and neurons?
• If yes, how to extract the correspondence?
• Propose: Bi-directional reasoning
  • Forward: attribute changes —> neuron activation changes
  • Backward: neuron activation changes —> attribute changes
  • Backward: no attribute changes —> no neuron activation changes
Attribute Witness Extraction

- Attribute substitution
- Attribute preservation
- Feature variants
- Feature invariants
- Attribute witnesses
Attribute-steered Model

- Constructed by transforming the original model (without additional training)
  - Neuron weakening (non-witness)
    \[ v' = e^{-\frac{v-\mu}{\alpha \cdot \sigma}} \cdot v \]
  - Neuron strengthening (witness)
    \[ v' = \epsilon \cdot v + \left(1 - e^{-\frac{v-\min}{\beta \cdot \sigma}}\right) \cdot v \]

\( v \): activation of a neuron
\( \mu \): mean of witness neurons
\( \sigma \): deviation of witness neurons
\( \alpha \): weakening factor
\( \epsilon, \beta \): strengthening factor
\( \min \): minimum of witness neurons
Evaluation

• Model
  • VGG-Face: 16 layers, 97.27% on LFW

• Datasets
  • VGG Face dataset (VF)
  • Labeled Faces in the Wild (LFW)
  • CelebFaces Attributes dataset (CelebA)

• Attacks
  • Patch, Glasses, C&W₀, C&W₂, C&W∞, FGSM, BIM
Extracted Attribute Witnesses

- Extracted witnesses of VGG-Face model

<table>
<thead>
<tr>
<th>Layer Name</th>
<th>conv1_1</th>
<th>conv1_2</th>
<th>pool1</th>
<th>conv2_1</th>
<th>conv2_2</th>
<th>pool2</th>
<th>conv3_1</th>
<th>conv3_2</th>
<th>conv3_3</th>
<th>pool3</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Neuron</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>#Left Eye</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>#Right Eye</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>#Nose</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>#Mouth</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>15</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>#Shared</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Layer Name</th>
<th>conv4_1</th>
<th>conv4_2</th>
<th>conv4_3</th>
<th>pool4</th>
<th>conv5_1</th>
<th>conv5_2</th>
<th>conv5_3</th>
<th>pool5</th>
<th>fc6</th>
<th>fc7</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Neuron</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>4096</td>
<td>4096</td>
</tr>
<tr>
<td>#Left Eye</td>
<td>9</td>
<td>5</td>
<td>15</td>
<td>7</td>
<td>12</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>#Right Eye</td>
<td>7</td>
<td>3</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>#Nose</td>
<td>10</td>
<td>8</td>
<td>17</td>
<td>13</td>
<td>7</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>#Mouth</td>
<td>19</td>
<td>12</td>
<td>12</td>
<td>11</td>
<td>8</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>#Shared</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Attribute Detection

- Predict the presence of attributes
- Train only on VF dataset, test on VF (disjoint set) and LFW
- Face descriptor: fc7 layer of VGG-Face model

<table>
<thead>
<tr>
<th>Dataset</th>
<th>VF [19]</th>
<th>LFW [33]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>Left Eye</td>
<td>Right Eye</td>
</tr>
<tr>
<td>Face Descriptor</td>
<td>0.830</td>
<td>0.830</td>
</tr>
<tr>
<td>Attribute Witness</td>
<td>0.940</td>
<td>0.935</td>
</tr>
</tbody>
</table>
# Accuracy of Adversary Detection

<table>
<thead>
<tr>
<th>Detector</th>
<th>FP</th>
<th>Targeted</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Untargeted</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Patch</td>
<td>Glasses</td>
<td>C&amp;W₀</td>
<td>C&amp;W₂</td>
<td>C&amp;Wₘ₀</td>
<td>FGSM</td>
<td>BIM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>First</td>
<td>Next</td>
<td>First</td>
<td>Next</td>
<td>First</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS [18]</td>
<td>23.32%</td>
<td>0.77</td>
<td>0.71</td>
<td>0.73</td>
<td>0.58</td>
<td>0.68</td>
<td>0.65</td>
<td>0.60</td>
<td>0.50</td>
</tr>
<tr>
<td>AS</td>
<td>20.41%</td>
<td>0.96</td>
<td>0.98</td>
<td>0.97</td>
<td>0.97</td>
<td>0.93</td>
<td>0.99</td>
<td>0.99</td>
<td>1.00</td>
</tr>
<tr>
<td>AP</td>
<td>30.61%</td>
<td>0.89</td>
<td>0.96</td>
<td>0.69</td>
<td>0.75</td>
<td>0.96</td>
<td>0.94</td>
<td>0.99</td>
<td>0.97</td>
</tr>
<tr>
<td>WKN</td>
<td>7.87%</td>
<td>0.94</td>
<td>0.97</td>
<td>0.71</td>
<td>0.76</td>
<td>0.83</td>
<td>0.89</td>
<td>0.99</td>
<td>0.97</td>
</tr>
<tr>
<td>STN</td>
<td>2.33%</td>
<td>0.08</td>
<td>0.19</td>
<td>0.16</td>
<td>0.19</td>
<td>0.90</td>
<td>0.94</td>
<td>0.97</td>
<td>1.00</td>
</tr>
<tr>
<td>AmI</td>
<td>9.91%</td>
<td>0.97</td>
<td>0.98</td>
<td>0.85</td>
<td>0.85</td>
<td>0.91</td>
<td>0.95</td>
<td>0.99</td>
<td>0.99</td>
</tr>
</tbody>
</table>

FP: false positive  
First: the first label of classes  
Next: the next label of the correct prediction  
FS: feature squeezing (NDSS ’18)  
AS/AP: attribute substitution/preservation  
WKN/STN: neuron weakening/strengthening
Conclusion

• Looking into the internals of AI models to provide important hints to address debugging problems and adversarial sample attack problems

• Both projects open-sourced on github

• On-going works: develop tools to fix a wide range of AI model bugs
Thank you!