Adversarial Examples and Adversarial Training

Ian Goodfellow, OpenAI Research Scientist
Uber, San Francisco 2016-10-27
In this presentation

- “Intriguing Properties of Neural Networks” Szegedy et al, 2013
- “Explaining and Harnessing Adversarial Examples” Goodfellow et al 2014
- “Adversarial Perturbations of Deep Neural Networks” Warde-Farley and Goodfellow, 2016
In this presentation

• “Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples” Papernot et al 2016

• “Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples” Papernot et al 2016

• “Adversarial Perturbations Against Deep Neural Networks for Malware Classification” Grosse et al 2016 (not my own work)
In this presentation

• “Distributional Smoothing with Virtual Adversarial Training” Miyato et al 2015 (not my own work)

• “Virtual Adversarial Training for Semi-Supervised Text Classification” Miyato et al 2016

Overview

• What are adversarial examples?

• Why do they happen?

• How can they be used to compromise machine learning systems?

• What are the defenses?

• How to use adversarial examples to improve machine learning, even when there is no adversary
Since 2013, deep neural networks have matched human performance at... 

...recognizing objects and faces... 

(Szegedy et al, 2014) 

(Taigmen et al, 2013)

...solving CAPTCHAS and reading addresses... 

(Goodfellow et al, 2013)

(Goodfellow et al, 2013)

and other tasks... 

(Goodfellow 2016)
Adversarial Examples

Timeline:
“Adversarial Classification” Dalvi et al 2004: fool spam filter
“Evasion Attacks Against Machine Learning at Test Time” Biggio 2013: fool neural nets
Szegedy et al 2013: fool ImageNet classifiers imperceptibly
Goodfellow et al 2014: cheap, closed form attack

(Goodfellow 2016)
Turning Objects into “Airplanes”
Attacking a Linear Model
Not just for neural nets

- Linear models
  - Logistic regression
  - Softmax regression
  - SVMs
- Decision trees
- Nearest neighbors
Adversarial Examples from Overfitting
Adversarial Examples from Excessive Linearity
Modern deep nets are very piecewise linear

Rectified linear unit

Maxout

Carefully tuned sigmoid

LSTM

(Goodfellow 2016)
Nearly Linear Responses in Practice

(Goodfellow 2016)
Small inter-class distances

All three perturbations have L2 norm 3.96
This is actually small. We typically use 7!

Perturbation changes the true class

Random perturbation does not change the class

Perturbation changes the input to “rubbish class”
The Fast Gradient Sign Method

\[ J(\tilde{x}, \theta) \approx J(x, \theta) + (\tilde{x} - x)^\top \nabla_x J(x). \]

Maximize

\[ J(x, \theta) + (\tilde{x} - x)^\top \nabla_x J(x) \]

subject to

\[ ||\tilde{x} - x||_\infty \leq \epsilon \]

\[ \Rightarrow \tilde{x} = x + \epsilon \text{sign} (\nabla_x J(x)) . \]
Maps of Adversarial and Random Cross-Sections

(collaboration with David Warde-Farley and Nicolas Papernot)

(Goodfellow 2016)
Maps of Adversarial Cross-Sections
Maps of Random Cross-Sections

Adversarial examples are not noise

(collaboration with David Warde-Farley and Nicolas Papernot)

(Goodfellow 2016)
Clever Hans

(“Clever Hans, Clever Algorithms,” Bob Sturm)
Wrong almost everywhere
High-Dimensional Linear Models

Weights

Signs of weights

Clean examples

Adversarial

(Goodfellow 2016)
Linear Models of ImageNet

(Andrej Karpathy, “Breaking Linear Classifiers on ImageNet”)
RBFs behave more intuitively
Cross-model, cross-dataset generalization
Cross-technique transferability

<table>
<thead>
<tr>
<th>Source Machine Learning Technique</th>
<th>DNN</th>
<th>LR</th>
<th>SVM</th>
<th>DT</th>
<th>kNN</th>
<th>Ens.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td>38.27</td>
<td>23.02</td>
<td>64.32</td>
<td>79.31</td>
<td>8.36</td>
<td>20.72</td>
</tr>
<tr>
<td>LR</td>
<td>6.31</td>
<td>91.64</td>
<td>91.43</td>
<td>87.42</td>
<td>11.29</td>
<td>44.14</td>
</tr>
<tr>
<td>SVM</td>
<td>2.51</td>
<td>36.56</td>
<td>100.0</td>
<td>80.03</td>
<td>5.19</td>
<td>15.67</td>
</tr>
<tr>
<td>DT</td>
<td>0.82</td>
<td>12.22</td>
<td>8.85</td>
<td>89.29</td>
<td>3.31</td>
<td>5.11</td>
</tr>
<tr>
<td>kNN</td>
<td>11.75</td>
<td>42.89</td>
<td>82.16</td>
<td>82.95</td>
<td>41.65</td>
<td>31.92</td>
</tr>
</tbody>
</table>

(Papernot 2016)
Transferability Attack

Target model with unknown weights, machine learning algorithm, training set; maybe non-differentiable

Train your own model

Substitute model mimicking target model with known, differentiable function

Deploy adversarial examples against the target; transferability property results in them succeeding

Adversarial examples

Adversarial crafting against substitute

(Goodfellow 2016)
Cross-Training Data Transferability

(Papernot 2016)
Adversarial Examples in the Human Brain

These are concentric circles, not intertwined spirals.

(Pinna and Gregory, 2002)
Practical Attacks

- Fool real classifiers trained by remotely hosted API (MetaMind, Amazon, Google)
- Fool malware detector networks
- Display adversarial examples in the physical world and fool machine learning systems that perceive them through a camera
Adversarial Examples in the Physical World

(a) Printout

(b) Photo of printout

(c) Cropped image
Hypothetical Attacks on Autonomous Vehicles

Denial of service

Confusing object

Harm others

Adversarial input recognized as “open space on the road”

Harm self / passengers

Adversarial input recognized as “navigable road”
Failed defenses

Generative pretraining
Adding noise at test time
Confidence-reducing perturbation at test time
Weight decay
Various non-linear units
Removing perturbation with an autoencoder
Ensembles
Multiple glimpses
Double backprop
Dropout
Adding noise at train time

Error correcting codes

(Goodfellow 2016)
Generative Modeling is not Sufficient to Solve the Problem

Both these two class mixture models implement roughly the same marginal over $x$, with very different posteriors over the classes. The likelihood criterion cannot strongly prefer one to the other, and in many cases will prefer the bad one.
Universal approximator theorem

Neural nets can represent either function:

Maximum likelihood doesn’t cause them to learn the right function. But we can fix that...
Training on Adversarial Examples

![Graphs showing validation set error over training time for clean and adversarial examples](image)

- **Clean examples**
  - Validation set error decreases significantly with training time.
- **Adversarial examples**
  - Validation set error remains almost constant, indicating limited improvement.

Bachelor of Science (Statistics and Mathematics)  
University of California, Berkeley  
February 2017

(Original Source: Goodfellow 2016)
Adversarial Training of other Models

- Linear models: SVM / linear regression cannot learn a step function, so adversarial training is less useful, very similar to weight decay

- $k$-NN: adversarial training is prone to overfitting.

- Takeway: neural nets can actually become more secure than other models. *Adversarially trained neural nets have the best empirical success rate on adversarial examples of any machine learning model.*
Weaknesses Persist
Adversarial Training

Labeled as bird

Decrease probability of bird class

Still has same label (bird)
Virtual Adversarial Training

Unlabeled; model guesses it’s probably a bird, maybe a plane

New guess should match old guess (probably bird, maybe plane)

Adversarial perturbation intended to change the guess
Text Classification with VAT

RCV1 Misclassification Rate

<table>
<thead>
<tr>
<th>Method</th>
<th>Misclassification Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Earlier SOTA</td>
<td>7.70</td>
</tr>
<tr>
<td>SOTA</td>
<td>7.20</td>
</tr>
<tr>
<td>Our baseline</td>
<td>7.40</td>
</tr>
<tr>
<td>Adversarial</td>
<td>7.12</td>
</tr>
<tr>
<td>Virtual Adversarial</td>
<td>7.05</td>
</tr>
<tr>
<td>Both</td>
<td>6.97</td>
</tr>
<tr>
<td>Both + bidirectional model</td>
<td>6.68</td>
</tr>
</tbody>
</table>

(Zoomed in for legibility)
Universal engineering machine (model-based optimization)

Make new inventions by finding input that maximizes model’s predicted performance

Training data Extrapolation

(Goodfellow 2016)
Conclusion

• Attacking is easy

• Defending is difficult

• Benchmarking vulnerability is training

• Adversarial training provides regularization and semi-supervised learning

• The out-of-domain input problem is a bottleneck for model-based optimization generally
cleverhans

Open-source library available at:
https://github.com/openai/cleverhans
Built on top of TensorFlow (Theano support anticipated)
Standard implementation of attacks, for adversarial training
and reproducible benchmarks