Explaining and Harnessing Adversarial Examples

Ian J. Goodfellow, Jonathan Schlens, and Christian Szegedy
Google Inc.
Presented by Jonathan Dingess
Neural network used is GoogLeNet, a 22-layer deep convolutional neural network that claimed state-of-the-art on the ImageNet database at the time of its release. Left image is from ImageNet; Right image is adversarial input generated from left
A quick note

• Adversarial examples are, by and large, **not** naturally occurring
• Neural networks are not the only algorithms prone to adversarial examples, even linear classifiers have this issue
• This isn’t meant as a presentation of a flaw, but as an opportunity to improve neural networks
Objectives

• Given a neural network and a dataset, efficiently generate adversarial examples
• Use adversarial examples as training data to regularize a neural network
• Explain and demystify adversarial examples
• Analyze some interesting properties of adversarial examples
What causes this?

• First, consider a linear model
• With 8 bits for each pixel, a pixel can have values of 0-255
• Our model can still process non-integer values
• Consider input $\tilde{x} = x + \eta$, $\|\eta\|_\infty < \varepsilon$

$$w^T \tilde{x} = w^T x + w^T \eta$$

$$\eta = \varepsilon \cdot \text{sign}(w)$$
Non-linear models

• Does our linear perturbation work just as well on non-linear models?
• Some neural networks are intentionally designed to behave in very linear ways to be easier to optimize
• Define the “Fast gradient sign method”:
  \[ \eta = \varepsilon \times \text{sign}(\nabla_x \mathcal{L}(\theta, x, y)) \]
\[ x + .007 \times \text{sign}(\nabla_x J(\theta, x, y)) = x + \epsilon \text{sign}(\nabla_x J(\theta, x, y)) \]

"panda" 57.7% confidence

"nematode" 8.2% confidence

"gibbon" 99.3% confidence
Adversarial Training

• Unlike shallow linear models, deep neural networks can become resistant to adversarial examples

\[
\tilde{\mathcal{L}}(\theta, x, y) = \alpha \mathcal{L}(\theta, x, y) + (1 - \alpha) \mathcal{L}(\theta, x + \varepsilon \text{sign}(\nabla_x \mathcal{L}(\theta, x, y)), y)
\]
Thoughts

• Adversarial training can be thought of as minimizing the worst-case error when the input is perturbed maliciously.

• Think of it as active learning, where new points are generated dimensionally far away but perceptually close, and we heuristically just give the new point the same label as the data it is close to.
Results

• From 0.94% error down to 0.84% on the same architecture
• Down to 0.77% with higher dimensionality
• More resistant to adversarial examples
  – Without training, the model misclassified adversarial examples 89.4% of the time
  – With training, this fell to 17.9%
Other Considerations I

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

- Gradient can’t predict the adversary’s changes because of the sign function
Other Considerations II

• Should we perturb the input or the hidden layers or both?
  – The results are inconsistent
  – No sense in perturbing the final layer

• What about other network types?
  – Some low capacity networks, such as RBF networks, are resistant to adversarial examples
Interesting Properties

• Adversarial Examples generalize
  – An adversarial example generated for one neural network will often be an adversarial example for others
Summary and Discussion I

• Adversarial networks can be explained as a property of high-dimensional dot products; they are a result of models being too linear rather than too nonlinear

• The generalization of adversarial examples across different models can be explained as a result of adversarial perturbations being highly aligned with the weight vectors of a model, and different models learning similar functions when trained to do the same task
Summary and Discussion II

• The direction of perturbation, not the specific point in space, is what matters most
• Because it is the direction that matters most, adversarial examples generalize across different clean examples
• We have a family of fast methods to generate adversarial examples
• Adversarial training can result in regularization further than dropout
Summary and Discussion III

- Models that are easy to optimize are easy to perturb
- Linear models lack the capacity to resist adversarial perturbations; only structures with a hidden layer should be trained to resist adversarial perturbation
Questions?