On Detecting Adversarial Perturbations

Presenter: Shijia Wang

Jan Hendrik Metzen Tim Genewein Volker Fischer Bastian Bischoff

Bosch Center for Artificial Intelligence, Robert Bosch GmbH

ICLR 2017

Outline

1. Introduction
   - Adversarial Examples
2. Methods
   - Fast Method
   - Basic Iterative Method
   - DeepFool Method
   - Adversary Detection Network
   - Dynamic Adversaries and Detectors
3. Conclusion
   - Experiments

Outline

1. Introduction
   - Adversarial Examples

2. Methods
   - Fast Method
   - Basic Iterative Method
   - DeepFool Method
   - Adversary Detection Network
   - Dynamic Adversaries and Detectors

3. Conclusion
   - Experiments

Definition

- Small perturbations almost imperceptible by humans but lead to incorrect classifications
- Want network to be more robust against adversarial examples
- Propose a binary detection network that detects adversarial examples

Past Solutions

- Augmenting the training input (Goodfellow et al., 2015)
- Append a stability term to the objective function (Zheng et al. 2016)
- Distilling a hardened network from the original classifier network (Papernot et al., 2016b)

Why Exist Theories

- High non-linearity of deep networks cause the existence of pockets of low-probability adversarial examples (Szegedy et al. 2014)
- Linear explanation: for some input x and adversarial noise η, the adversarial example $x^{adv} = x + \eta$ multiplied by the weight vector w makes $w^T x^{adv} = w^T x + x^T \eta$. Many small changes in η causes neuron changes. (Goodfellow et al., 2015)
- Class boundary lies close to a data manifold. (Tanay & Griffin 2016)

Math

- x input
- $y_{true}(x)$ one-hot encoding of true class of image x
- $\mathbf{J}_{cls}(x, y(x))$ the cost function of the classifier

Outline

1. Introduction
   - Adversarial Examples

2. Methods
   - Fast Method
   - Basic Iterative Method
   - DeepFool Method
   - Adversary Detection Network
   - Dynamic Adversaries and Detectors

3. Conclusion
   - Experiments

Fast Method

- $$x^{adv} = x + \epsilon sgn(\nabla_x \mathbf{J}_{cls}(x, y_{true}(x)))$$
- Applied perturbation is in the direction of the in image space which yields the highest increase of the linearized cost function under l_∞-norm
- One step in the direction of the gradient's sign with step ϵ

Goodfellow et al. (2015)

Fast Method

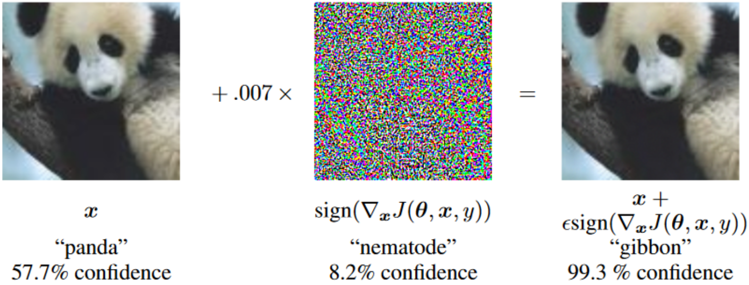

x
"panda"
57.7% confidence
$+ .007 \times$
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$=$
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Figure 1: A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014a) on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet's classification of the image. Here our ϵ of .007 corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet's conversion to real numbers.

Outline

1 Introduction
- Adversarial Examples

2 Methods
- Fast Method
- Basic Iterative Method
- DeepFool Method
- Adversary Detection Network
- Dynamic Adversaries and Detectors

3 Conclusion
- Experiments

Basic Iterative Method l_∞

- l_∞-norm

$$x_0^{adv} = x, x_{n+1}^{adv} = Clip_{x,\epsilon}\{x_n^{adv} + \alpha sgn(\nabla_x \mathbf{J}_{cls}(x_n^{adv}, y_{true}(x)))$$

- $Clip_{X,\epsilon}\{X'\}(x,y,z)$
  $= min\{255, X(x,y,z) + \epsilon, max\{0, X(x,y,z) - \epsilon, X'(x,y,z)\}\}$
  X source image, x, y coordinates, z channel
- Step size α=1, iterations = 10

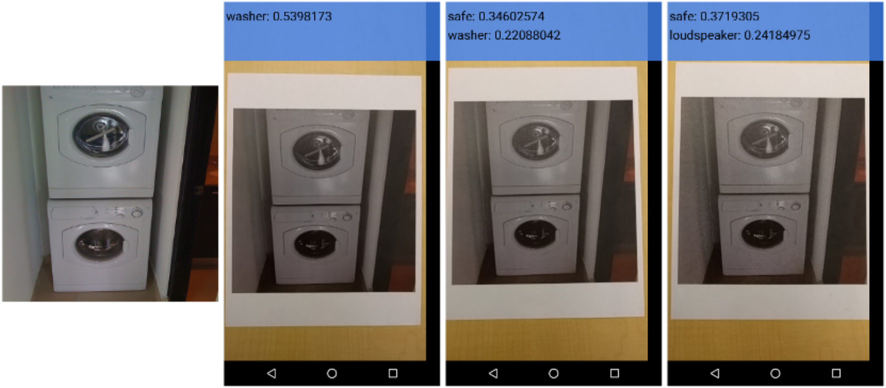
Kurakin et al. (2016)

Basic Iterative Method l_2

- Move toward the gradient but inside the ϵ neighborhood
- if the l_2 distance exceeds ϵ, project back on the ϵ ball

$$x_0^{adv} = x, x_{n+1}^{adv} = Project_{x,\epsilon}\{x_n^{adv} + \alpha \frac{\nabla_x \mathbf{J}_{cls}(x_n^{adv}, y_{true}(x))}{||\nabla_x \mathbf{J}_{cls}(x_n^{adv}, y_{true}(x))||_2}\}$$

Basic Iterative Method



washer: 0.5398173

safe: 0.34602574
washer: 0.22088042

safe: 0.3719305
loudspeaker: 0.24184975

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

Figure 1: Demonstration of a black box attack (in which the attack is constructed without access to the model) on a phone app for image classification using physical adversarial examples. We took a clean image from the dataset (a) and used it to generate adversarial images with various sizes of adversarial perturbation ϵ. Then we printed clean and adversarial images and used the TensorFlow Camera Demo app to classify them. A clean image (b) is recognized correctly as a "washer" when

Outline

1 Introduction

- Adversarial Examples

2 Methods

- Fast Method
- Basic Iterative Method
- DeepFool Method
- Adversary Detection Network
- Dynamic Adversaries and Detectors

3 Conclusion

- Experiments

DeepFool

- Iteratively step to cross class boundary.
- $min_r ||r||_2$ s.t. $y_{true}(x + r) \neq y_{true}(x)$
- Used l_2 and l_∞ norms

Moosavi-Dezfooli et al. (2016b)

Basic Iterative Method

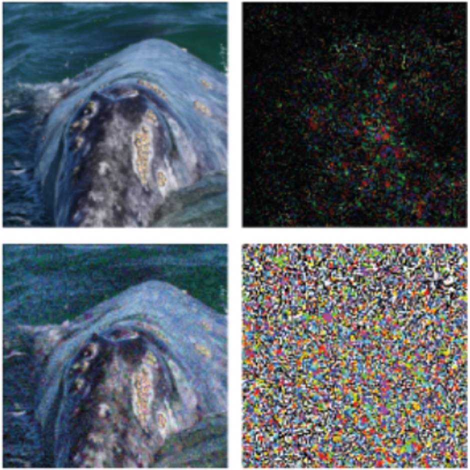


Figure 1: An example of adversarial perturbations computed by DeepFool and the fast gradient sign method [4]. First row: the original image x which is classified as "whale" ($\hat{k}(x)$). Second row: the image classified as "turtle" ($\hat{k}(x + r)$) and the corresponding perturbation r com-

Outline

1. Introduction
   - Adversarial Examples

2. Methods
   - Fast Method
   - Basic Iterative Method
   - DeepFool Method
   - Adversary Detection Network
   - Dynamic Adversaries and Detectors

3. Conclusion
   - Experiments

Subnetwork

- Subnetwork at some intermediate layer assigns probability if input is adversarial.
- Train by first training classification network, make adversarial examples, freeze classification network weights, then training subnetwork
- Adversarial data get 1, good get 0

Structure

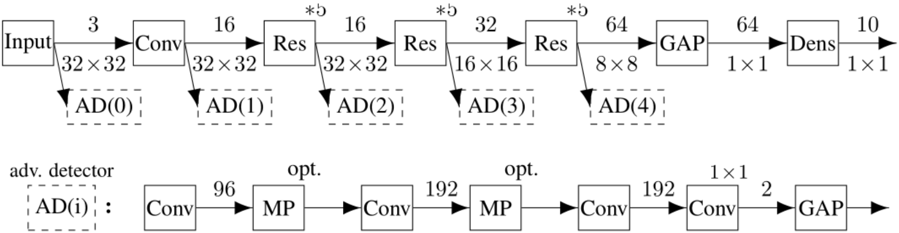


Figure 1: (Top) ResNet used for classification. Numbers on top of arrows denote the number of feature maps and numbers below arrows denote spatial resolutions. Conv denotes a convolutional layer, Res^{*5} denotes a sequence of 5 residual blocks as introduced by He et al. (2016), GAP denotes a global-average pooling layer and Dens a fully-connected layer. Spatial resolutions are decreased by strided convolution and the number of feature maps on the residual's shortcut is increased by 1x1 convolutions. All convolutional layers have 3x3 receptive fields and are followed by batch normalization and rectified linear units. (Bottom) Topology of detector network, which is attached to one of the AD(i) positions. MP denotes max-pooling and is optional: for AD(3), the second pooling layer is skipped, and for AD(4), both pooling layers are skipped.

Outline

1. Introduction
   - Adversarial Examples

2. Methods
   - Fast Method
   - Basic Iterative Method
   - DeepFool Method
   - Adversary Detection Network
   - Dynamic Adversaries and Detectors

3. Conclusion
   - Experiments

Dynamic Adversary

- Fool both the classifier and the detector.
- For $\sigma \in [0, 1]$
  $x_0^{adv} = x;$
  $x_{n+1}^{adv} = Clip_{x,\epsilon}\{x_n^{adv} + \alpha[(1-\sigma)sgn(\nabla_x \mathbf{J}_{cls}(x_n^{adv}, y_{true}(x))) + \sigma sgn(\mathbf{J}_{det}(x_n^{adv}, 1))]\}$
- Trade off between costs

Dynamic Adversary Training

- Compute adversarial on the fly with changing σ
- Adversary modify each data point with probability 0.5
- Train detector to resist for various values of σ

Outline

1. Introduction
   - Adversarial Examples
2. Methods
   - Fast Method
   - Basic Iterative Method
   - DeepFool Method
   - Adversary Detection Network
   - Dynamic Adversaries and Detectors
3. Conclusion
   - Experiments

CIFAR10

- Use 32-layer Residual Network
- CIFAR10 data

Static Adversaries

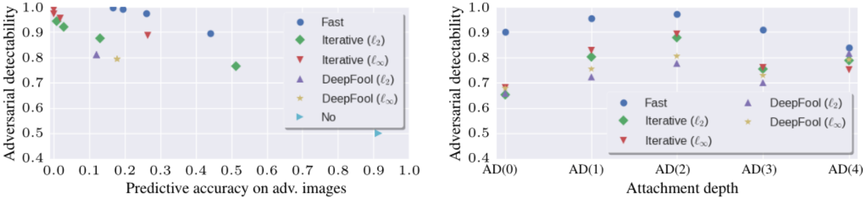
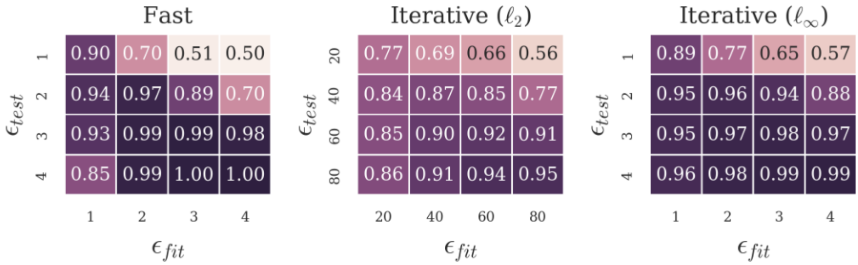


Figure 2: (Left) Illustration of detectability of different adversaries and values for ε on CIFAR10. The x-axis shows the predictive accuracy of the CIFAR10 classifier on adversarial examples of the test data for different adversaries. The y-axis shows the corresponding detectability of the adversarial examples, with 0.5 corresponding to chance level. "No" corresponds to an "adversary" that leaves the input unchanged. (Right) Analysis of the detectability of adversarial examples of different adversaries for different attachment depths of the detector.

Static Adversaries



Fast

ϵ_{test} \ ϵ_{fit}	1	2	3	4
1	0.90	0.70	0.51	0.50
2	0.94	0.97	0.89	0.70
3	0.93	0.99	0.99	0.98
4	0.85	0.99	1.00	1.00

Iterative (ℓ_2)

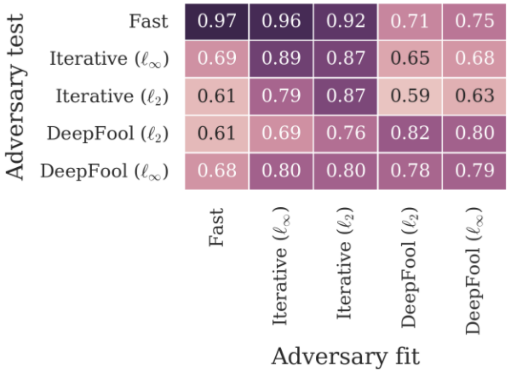
ϵ_{test} \ ϵ_{fit}	20	40	60	80
20	0.77	0.69	0.66	0.56
40	0.84	0.87	0.85	0.77
60	0.85	0.90	0.92	0.91
80	0.86	0.91	0.94	0.95

Iterative (ℓ_∞)

ϵ_{test} \ ϵ_{fit}	1	2	3	4
1	0.89	0.77	0.65	0.57
2	0.95	0.96	0.94	0.88
3	0.95	0.97	0.98	0.97
4	0.96	0.98	0.99	0.99

Figure 3: Transferability on CIFAR10 of detector trained for adversary with maximal distortion ϵ_{fit} when tested on the same adversary with distortion ϵ_{test}. Different plots show different adversaries. Numbers correspond to the accuracy of detector on unseen test data.

Static Adversaries



Adversary test \ Adversary fit	Fast	Iterative (ℓ_∞)	Iterative (ℓ_2)	DeepFool (ℓ_2)	DeepFool (ℓ_∞)
Fast	0.97	0.96	0.92	0.71	0.75
Iterative (ℓ_∞)	0.69	0.89	0.87	0.65	0.68
Iterative (ℓ_2)	0.61	0.79	0.87	0.59	0.63
DeepFool (ℓ_2)	0.61	0.69	0.76	0.82	0.80
DeepFool (ℓ_∞)	0.68	0.80	0.80	0.78	0.79

Figure 4: Transferability on CIFAR10 of detector trained for one adversary when tested on other adversaries. The maximal distortion ϵ of the adversary (when applicable) has been chosen minimally such that the predictive accuracy of the classifier is below 30%. Numbers correspond to the accuracy of the detector on unseen test data.

Dynamic Adversaries

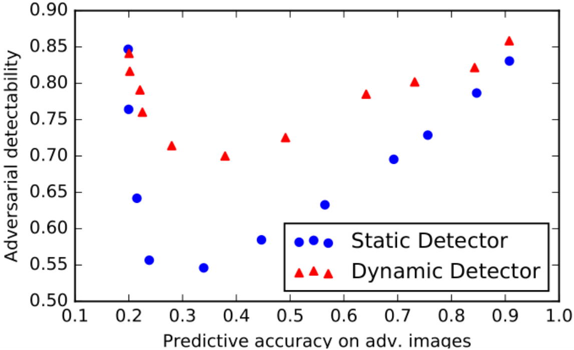


Figure 5: Illustration of detectability versus classification accuracy of a dynamic adversary for different values of σ against a static and dynamic detector. The parameter σ has been chosen as $\sigma \in \{0.0, 0.1, \ldots, 1.0\}$, with smaller values of σ corresponding to lower predictive accuracy, i.e., being further on the left.

10-Class ImageNet

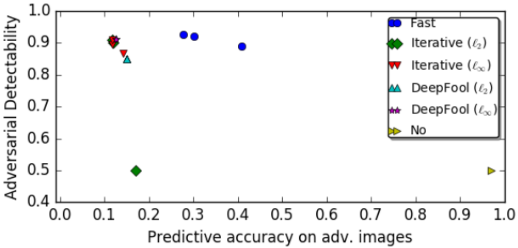


Figure 6: Illustration of detectability of different adversaries and values for ε on 10-class ImageNet. The x-axis shows the predictive accuracy of the ImageNet classifier on adversarial examples of the test data for different adversaries. The y-axis shows the corresponding detectability of the adversarial examples, with 0.5 corresponding to chance level.

10-Class ImageNet

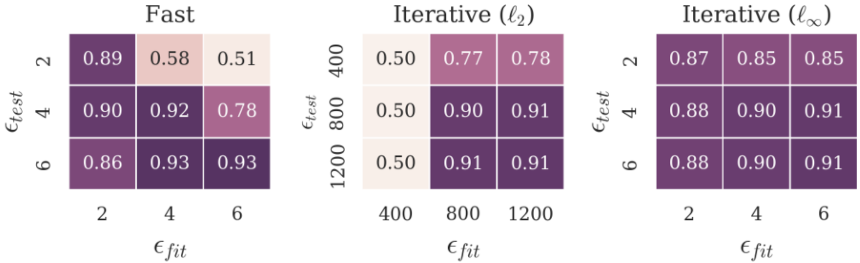
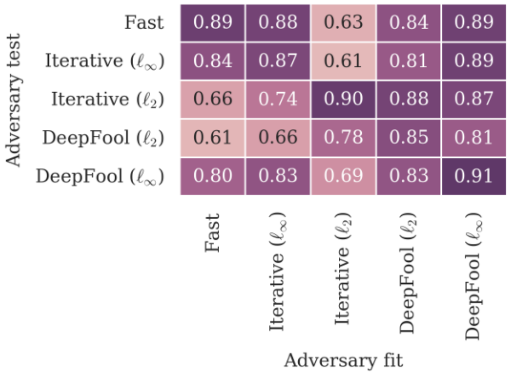


Figure 7: Transferability on 10-class ImageNet of detector trained for adversary with maximal distortion ϵ_{fit} when tested on the same adversary with distortion ϵ_{test}. Different plots show different adversaries. Numbers correspond to the accuracy of the detector on unseen test data.

10-Class ImageNet



Adversary test \ Adversary fit	Fast	Iterative (ℓ_∞)	Iterative (ℓ_2)	DeepFool (ℓ_2)	DeepFool (ℓ_∞)
Fast	0.89	0.88	0.63	0.84	0.89
Iterative (ℓ_∞)	0.84	0.87	0.61	0.81	0.89
Iterative (ℓ_2)	0.66	0.74	0.90	0.88	0.87
DeepFool (ℓ_2)	0.61	0.66	0.78	0.85	0.81
DeepFool (ℓ_∞)	0.80	0.83	0.69	0.83	0.91

Figure 8: Transferability on 10-class ImageNet of detector trained for one adversary when tested on other adversaries. The maximal distortion of the ℓ_∞-based Iterative adversary has been chosen as $\varepsilon = 2$ and as $\varepsilon = 800$ for the ℓ_2-based adversary. Numbers correspond to the accuracy of detector on unseen test data.

Summary

- Pretty high rate of identifying adversarial input
- Image-based perturbations are sufficiently regular to be detectable
- Dynamic detector much harder to fool
  - Reduce the area adversarial to both the classifier and detector
  - Area might become more irregular and harder to find with gradient descent
- Further work: Use the gradient as a source of regularization