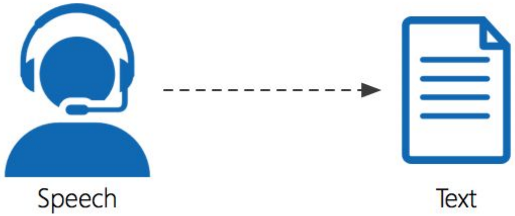
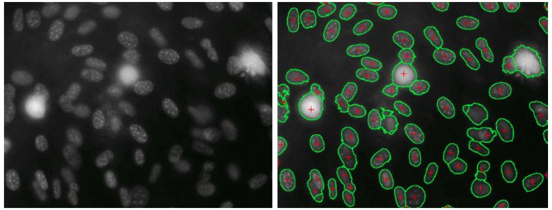
DeepFool: a simple and accurate method to fool deep neural networks

ZIAN WANG   BINGZHAO SHAN   SONGLIN LIU

Background



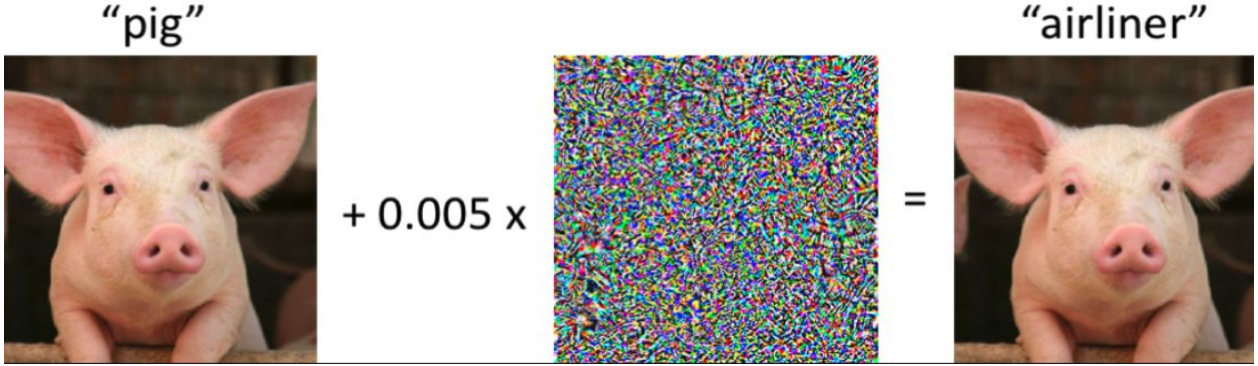
Background: Neural Networks



Deep neural networks are powerful learning models that achieve state-of-the-art pattern recognition performance in many research areas.

Background: Problems with Neural Networks

However, previous research (Szegedy, Zaremba et al., 2014) has shown that the high complexity of neural networks might be a reason explaining the presence of adversarial examples.



Terminology: Minimal adversarial perturbation

(1) The minimal adversarial perturbation

The minimal perturbation r that is sufficient to change the estimate label $\hat{k}(x)$:

$$\Delta(\boldsymbol{x}; \hat{k}) := \min_{\boldsymbol{r}} \|\boldsymbol{r}\|_2 \text{ subject to } \hat{k}(\boldsymbol{x} + \boldsymbol{r}) \neq \hat{k}(\boldsymbol{x}), \quad (1)$$

where x is an image and $\hat{k}(x)$ is the estimated label.

We call $\Delta\,(x; \hat{k})$ the robustness of $\hat{k}$ at point x.

Terminology: Robustness of a classifier

(2) The robustness of classifier $\hat{k}$ is then defined as

$$\rho_{\text{adv}}(\hat{k}) = \mathbb{E}_{\boldsymbol{x}} \frac{\Delta(\boldsymbol{x}; \hat{k})}{\|\boldsymbol{x}\|_2}, \qquad (2)$$

where E_x is the expectation over the distribution of data

Introduction

Introduction: DeepFool

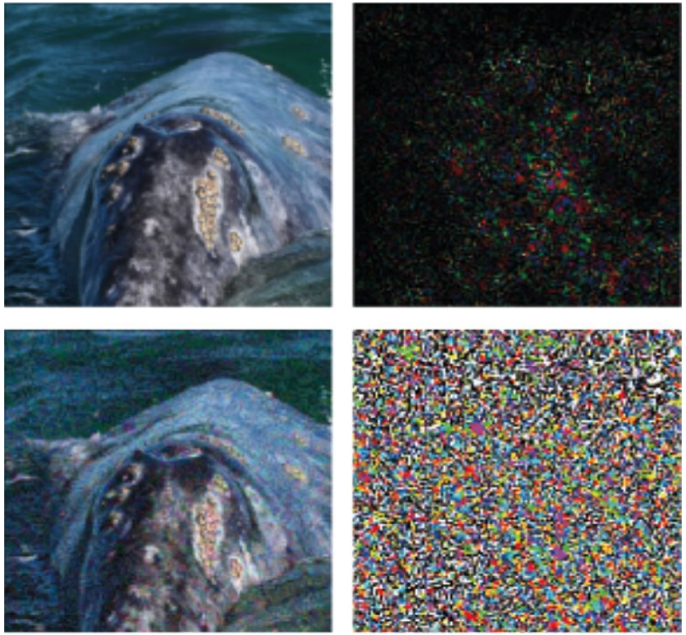
DeepFool: a simple and accurate method to fool deep neural networks

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Pascal Frossard
École Polytechnique Fédérale de Lausanne
{seyed.moosavi,alhussein.fawzi,pascal.frossard} at epfl.ch

Introduction: DeepFool

DeepFool: a simple and accurate method to fool deep neural networks

Original image (whale)



Attacked by DeepFool (turtle)

Fast Gradient Sign (turtle)

Main contributions

- Proposed a simple yet accurate method for **computing and comparing the robustness** of different classifiers to adversarial perturbations.
- Provided a more efficient approach to obtain a **coarse approximation of the minimal perturbation.**
- Provided a **guidance for data augmentation**
- Showed that **imprecise approaches** could lead to different and sometimes misleading conclusions about the robustness. Proposed a better understanding of this intriguing phenomenon and of its influence factors.

Methods

Affine Binary Classifiers

- Minimal Perturbation for Affine Binary Classifiers: $f(x) = w^T x + b$

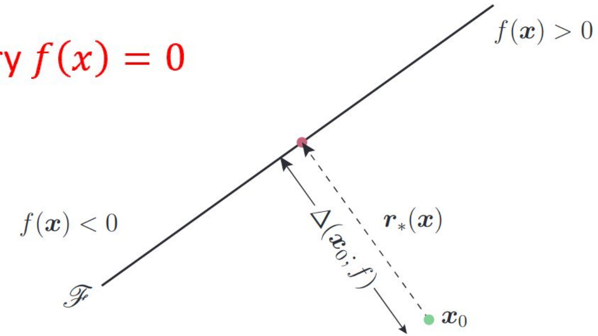
$$\boldsymbol{r}_*(\boldsymbol{x}_0) := \arg\min \|\boldsymbol{r}\|_2 \quad \text{subject to } \operatorname{sign}(f(\boldsymbol{x}_0 + \boldsymbol{r})) \neq \operatorname{sign}(f(\boldsymbol{x}_0))$$

$$\hat{k}(\boldsymbol{x}) = \operatorname{sign}(f(\boldsymbol{x})),$$

- Closed-form solution:

  $r_*(x_0)$ equals to the projection of x_0 onto decision boundary $f(x) = 0$

$$\mathbf{r}_*(\mathbf{x}_0) := -\frac{f(\mathbf{x}_0)}{\|\mathbf{w}\|_2^2}\mathbf{w}$$



Binary Differential Classifiers

Iteratively "linearizes" model output at intermediate points to obtain an optimal update direction, and applies perturbation until change classes.

- At each iteration
  - Linearize classifier

$$f_{\text{approx}} = f(x_i) + \nabla f(x_i)^T (x - x_i)$$

  - Compute minimal perturbation for zero level set $f_{approx} = 0$

$$\arg\min_{\boldsymbol{r}_i} \|\boldsymbol{r}_i\|_2 \text{ subject to } f(\boldsymbol{x}_i) + \nabla f(\boldsymbol{x}_i)^T \boldsymbol{r}_i = 0$$

$$r_i = -\frac{f(x_i)}{||\nabla f(x_i)||_2^2} \nabla f(x_i)$$

Binary Classifier Pseudocode

Algorithm 1 DeepFool for binary classifiers

input: Image $\boldsymbol{x}$, classifier f.
output: Perturbation $\hat{\boldsymbol{r}}$.
Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\text{sign}(f(\boldsymbol{x}_i)) = \text{sign}(f(\boldsymbol{x}_0))$ **do**
    $\boldsymbol{r}_i \leftarrow -\frac{f(\boldsymbol{x}_i)}{\|\nabla f(\boldsymbol{x}_i)\|_2^2} \nabla f(\boldsymbol{x}_i)$,
    $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$,
    $i \leftarrow i + 1$.
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$.

One vs All MultiClass Classifiers

DeepFool targets one vs all multiclass classifiers

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^c \qquad \hat{k}(\boldsymbol{x}) = \arg\max_k f_k(\boldsymbol{x})$$

Affine Multiclass Classifiers Hyperplanes

- Affine Multiclass Classifiers

$$f(x) = W^T x + b$$

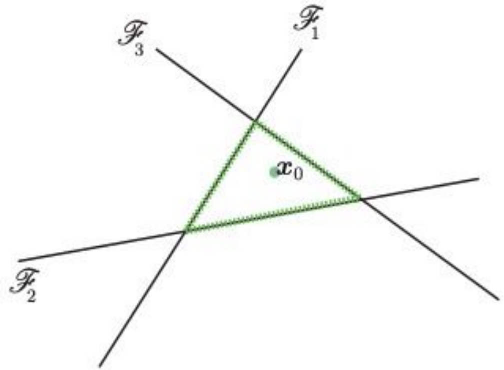
- Decision boundaries are hyperplanes

$$\mathscr{F}_k = \{x : f_k(x) - f_{k(x_0)}(x) = 0\}$$

where $k(x_0)$ is the true class of x_0

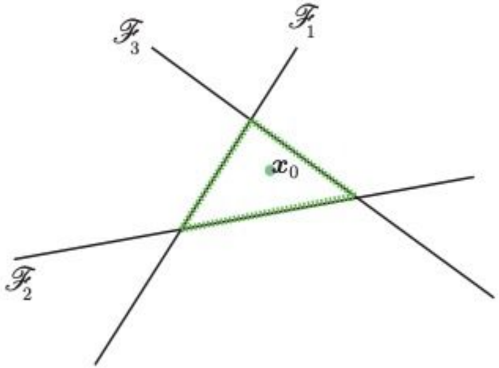
- Hyperplanes form a polyhedron P

$$P = \bigcap_{k=1}^{c} \{\boldsymbol{x} : f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}) \geq f_k(\boldsymbol{x})\}$$



Question

In the direction of which class (hyperplane) should we move x_0 to obtain minimum perturbation?



A. Second most likely class
B. Least most likely class

Affine Multiclass Classifiers

Minimum perturbation $r_*(x_0)$ is the minimum vector that projects onto the closest hyperplane.

- Find closest hyperplane

$$\hat{l}(\boldsymbol{x}_0) = \underset{k \neq \hat{k}(\boldsymbol{x}_0)}{\arg\min} \frac{\left| f_k(\boldsymbol{x}_0) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_0) \right|}{\| \boldsymbol{w}_k - \boldsymbol{w}_{\hat{k}(\boldsymbol{x}_0)} \|_2}.$$

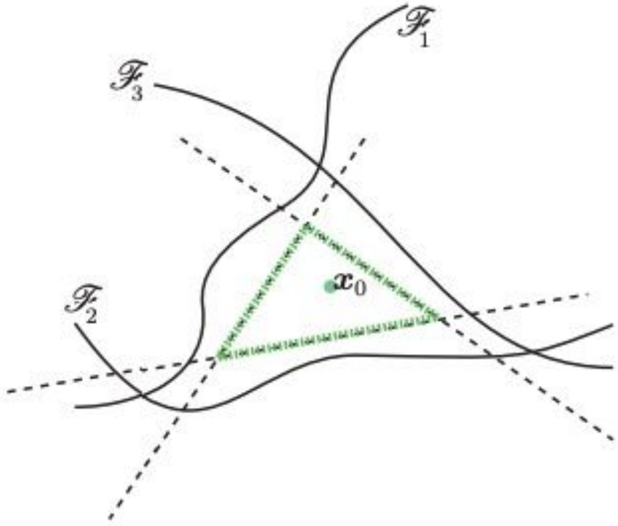
- Closed-form minimum perturbation for closest hyperplane zero level set

$$\boldsymbol{r}_*(\boldsymbol{x}_0) = \frac{\left| f_{\hat{l}(\boldsymbol{x}_0)}(\boldsymbol{x}_0) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_0) \right|}{\| \boldsymbol{w}_{\hat{l}(\boldsymbol{x}_0)} - \boldsymbol{w}_{\hat{k}(\boldsymbol{x}_0)} \|_2^2} (\boldsymbol{w}_{\hat{l}(\boldsymbol{x}_0)} - \boldsymbol{w}_{\hat{k}(\boldsymbol{x}_0)}).$$

General Multiclass Classifiers

For general differentiable classifiers, follow the same iterative linearization procedure in binary case

$$\tilde{P}_i = \bigcap_{k=1}^{c} \Big\{ \boldsymbol{x} : f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i) + \nabla f_k(\boldsymbol{x}_i)^\top \boldsymbol{x} - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)^\top \boldsymbol{x} \le 0 \Big\}$$

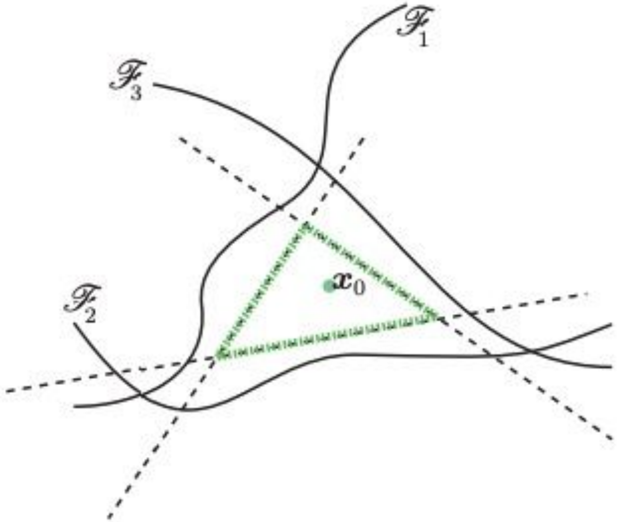


Multiclass DeepFool Pseudocode

Algorithm 2 DeepFool: multi-class case

input: Image $\boldsymbol{x}$, classifier f.
output: Perturbation $\hat{\boldsymbol{r}}$.

Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\hat{k}(\boldsymbol{x}_i) = \hat{k}(\boldsymbol{x}_0)$ **do**
  for $k \neq \hat{k}(\boldsymbol{x}_0)$ **do**
    $\boldsymbol{w}'_k \leftarrow \nabla f_k(\boldsymbol{x}_i) - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
    $f'_k \leftarrow f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
  end for
  $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(\boldsymbol{x}_0)} \frac{|f'_k|}{\|\boldsymbol{w}'_k\|_2}$
  $\boldsymbol{r}_i \leftarrow \frac{|f'_{\hat{l}}|}{\|\boldsymbol{w}'_{\hat{l}}\|_2^2} \boldsymbol{w}'_{\hat{l}}$
  $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$
  $i \leftarrow i + 1$
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$



DeepFool for Various Norms

For any ℓ_p norm ($p \in [1, \infty)$)

$$\hat{l}(x_0) = \underset{k \neq \hat{(k(x_0))}}{\operatorname{argmin}} \frac{\left|f_k(x_0 - f_{\hat{k}(x_0)}(x_0)\right|}{||w_k - w_{\hat{k}(x_0)}||_q}$$

$$r_*(x_0) = \frac{\left|f_{\hat{l}(x_0)}(x_0) - f_{\hat{k}(x_0)}(x_0)\right|}{||w_{\hat{l}(x_0)} - w_{\hat{k}(x_0)}||_q^q} \left|w_{\hat{l}(x_0)} - w_{\hat{k}(x_0)}\right|^{q-1} \odot sign(w_{\hat{l}(x_0)} - w_{\hat{k}(x_0)})$$

Experiments Results

Experiment Setup

Dataset + Architecture combinations:

- MNIST:
  - 2 FC layers
  - 2-layer LeNet
- CIFAR-10
  - 3-layer LeNet
  - Network in Network(NIN)
- ILSVRC2012 (1.2 million training, 15,000 test)
  - CaffeNet
  - GoogLeNet

Experiment Setup

Approaches compared:

- DeepFool
- Fast Gradient Sign (min epsilon to generate 90% error on data)
- Optimization based method in *Intriguing properties of neural networks*

Metrics:

- Time to compute one adversarial example
- Average robustness computed as $$\hat{\rho}_{\mathrm{adv}}(f) = \frac{1}{|\mathscr{D}|} \sum_{\boldsymbol{x} \in \mathscr{D}} \frac{\|\hat{\boldsymbol{r}}(\boldsymbol{x})\|_2}{\|\boldsymbol{x}\|_2}$$

Result analysis

Classifier	**Test error**	$\hat{\rho}_{\text{adv}}$ [DeepFool]	time	$\hat{\rho}_{\text{adv}}$ [4]	time	$\hat{\rho}_{\text{adv}}$ [18]	time
LeNet (MNIST)	1%	2.0×10^{-1}	110 ms	1.0	20 ms	2.5×10^{-1}	> 4 s
FC500-150-10 (MNIST)	1.7%	1.1×10^{-1}	50 ms	3.9×10^{-1}	10 ms	1.2×10^{-1}	> 2 s
NIN (CIFAR-10)	11.5%	2.3×10^{-2}	1100 ms	1.2×10^{-1}	180 ms	2.4×10^{-2}	>50 s
LeNet (CIFAR-10)	22.6%	3.0×10^{-2}	220 ms	1.3×10^{-1}	50 ms	3.9×10^{-2}	>7 s
CaffeNet (ILSVRC2012)	42.6%	2.7×10^{-3}	510 ms*	3.5×10^{-2}	50 ms*	-	-
GoogLeNet (ILSVRC2012)	31.3%	1.9×10^{-3}	800 ms*	4.7×10^{-2}	80 ms*	-	-

[4] = fast gradient sign

[18] = Optimization based method (*Intriguing properties of neural networks*)

The Same Conclusion on l_∞

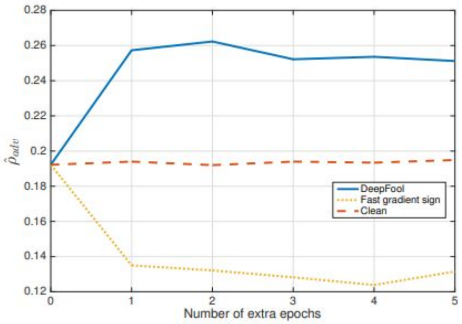
Classifier	DeepFool	Fast gradient sign
LeNet (MNIST)	0.10	0.26
FC500-150-10 (MNIST)	0.04	0.11
NIN (CIFAR-10)	0.008	0.024
LeNet (CIFAR-10)	0.015	0.028

Table 2: Values of $\hat{\rho}^{\infty}_{\text{adv}}$ for four different networks based on DeepFool (smallest l_∞ perturbation) and fast gradient sign method with 90% of misclassification.

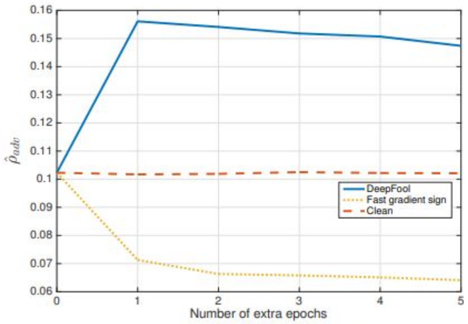
Fine-tuning with Adversarial Examples

- MNIST and CIFAR-10
- 5 extra epochs with adversarial data generated separately by DeepFool and fast gradient sign
- Half original learning rate
- Measures average robustness on every epoch with attack from DeepFool

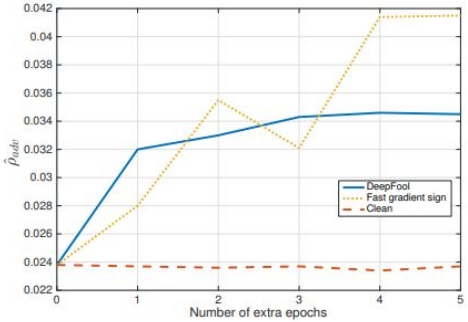
Result



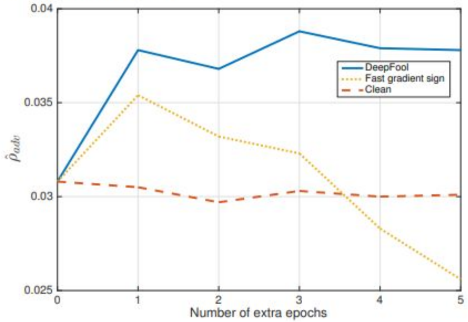
(a) Effect of fine-tuning on adversarial examples computed by two different methods for LeNet on MNIST.



(b) Effect of fine-tuning on adversarial examples computed by two different methods for a fully-connected network on MNIST.



(c) Effect of fine-tuning on adversarial examples computed by two different methods for NIN on CIFAR-10.



(d) Effect of fine-tuning on adversarial examples computed by two different methods for LeNet on CIFAR-10.

Figure 6

Reasoning on Robustness Drop

- Perturbation computed by fast gradient sign is not minimal
- Overly perturbed examples can go across the boundary of classes

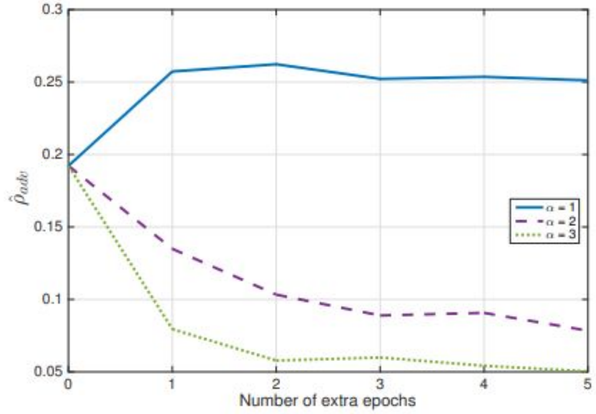


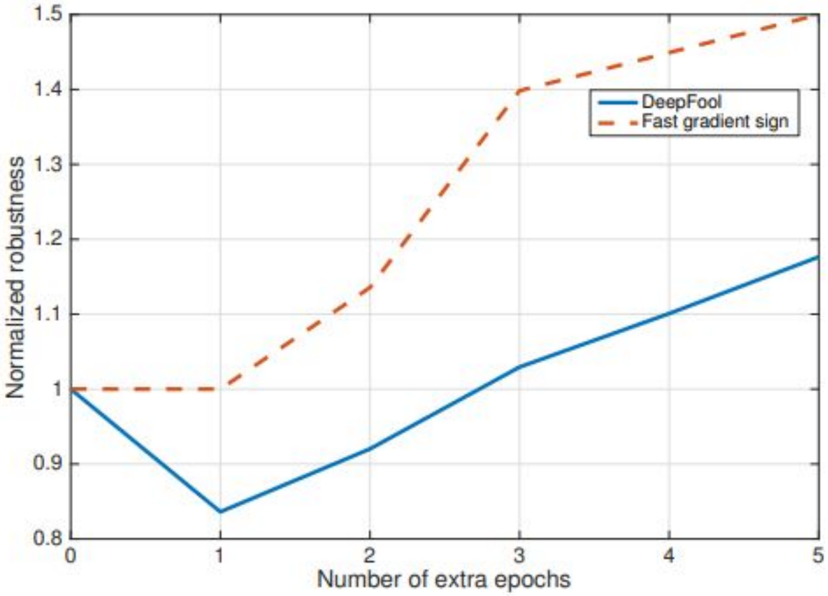
Figure 7: Fine-tuning based on magnified DeepFool's adversarial perturbations.

Fine-tuning with DeepFool Improves Robustness

Classifier	DeepFool	Fast gradient sign	Clean
LeNet (MNIST)	0.8%	4.4%	1%
FC500-150-10 (MNIST)	1.5%	4.9%	1.7%
NIN (CIFAR-10)	11.2%	21.2%	11.5%
LeNet (CIFAR-10)	20.0%	28.6%	22.6%

Table 3: The test error of networks after the fine-tuning on adversarial examples (after five epochs). Each columns correspond to a different type of augmented perturbation.

Importance of Accurate Measurement



Conclusion & Contribution

Demo

Q&A