Boulder

Computer Vision; Image Classification; Robustness

YouTube Playlist

Maziar Raissi

Assistant Professor

Department of Applied Mathematics

University of Colorado Boulder

maziar.raissi@colorado.edu

Intriguing properties of neural networks

YouTube Playlist

$x \in \mathbb{R}^m \rightarrow$ an input image

$\phi(x) \in \mathbb{R}^n \rightarrow$ activation values of some layer

Units of $\phi(x)$

$$x' = \arg\max_{x \in \mathcal{I}} \langle \phi(x), e_i \rangle$$

$e_i \rightarrow$ natural basis vector associated with the i-th hidden unit

$\mathcal{I} \rightarrow$ held-out set of images from the data distribution that the the network was not trained on

(a) Unit sensitive to lower round stroke.

(b) Unit sensitive to upper round stroke, or lower straight stroke.

(c) Unit senstive to left, upper round stroke.

(d) Unit senstive to diagonal straight stroke.

$$x' = \arg\max_{x \in \mathcal{I}} \langle \phi(x), v \rangle$$

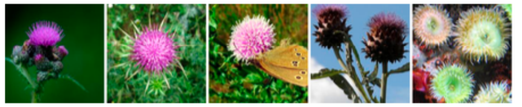
$v \in \mathbb{R}^n \rightarrow$ any random direction

(a) Direction sensitive to upper straight stroke, or lower round stroke.

(b) Direction sensitive to lower left loop.

(c) Direction senstive to round top stroke.

(d) Direction sensitive to right, upper round stroke.



Unit senstive to round, spiky flowers.

Direction sensitive to spread shapes.

Blind Spots

$f : \mathbb{R}^m \rightarrow \{1, \ldots, k\} \rightarrow$ a classifier

$$\min_r \quad \|r\|_2 \qquad \text{given } \ell \neq f(x)$$

$$\text{s.t.} \quad f(x+r) = \ell$$

$$x + r \in [0,1]^m$$

if $\ell = f(x)$ then $r = 0$ is a trivial solution

$\text{loss}_f : \mathbb{R}^m \times \{1, \ldots, k\} \rightarrow \mathbb{R}^+$

$$\min_r \quad c\|r\|_2 + \text{loss}_f(x+r, \ell)$$

$$\text{s.t.} \quad x + r \in [0,1]^m$$

Box-constrained L-BFGS

Cross-model generalization of adversarial examples

	FC10(10^{-4})	FC10(10^{-2})	FC10(1)	FC100-100-10	FC200-200-10	AE400-10	Av. distortion
FC10(10^{-4})	100%	11.7%	22.7%	2%	3.9%	2.7%	0.062
FC10(10^{-2})	87.1%	100%	35.2%	35.9%	27.3%	9.8%	0.1
FC10(1)	71.9%	76.2%	100%	48.1%	47%	34.4%	0.14
FC100-100-10	28.9%	13.7%	21.1%	100%	6.6%	2%	0.058
FC200-200-10	38.2%	14%	23.8%	20.3%	100%	2.7%	0.065
AE400-10	23.4%	16%	24.8%	9.4%	6.6%	100%	0.086
Gaussian noise, stddev=0.1	5.0%	10.1%	18.3%	0%	0%	0.8%	0.1
Gaussian noise, stddev=0.3	15.6%	11.3%	22.7%	5%	4.3%	3.1%	0.3

The general conclusion is that adversarial examples tend to stay hard even for models trained with different hyper-parameters.

Cross-training-set generalization error rate for the set of adversarial examples generated for different models

	FC100-100-10	FC123-456-10	FC100-100-10'
Distorted for FC100-100-10 (av. stddev=0.062)	100%	26.2%	5.9%
Distorted for FC123-456-10 (av. stddev=0.059)	6.25%	100%	5.1%
Distorted for FC100-100-10' (av. stddev=0.058)	8.2%	8.2%	100%
Gaussian noise with stddev=0.06	2.2%	2.6%	2.4%
Distorted for FC100-100-10 amplified to stddev=0.1	100%	98%	43%
Distorted for FC123-456-10 amplified to stddev=0.1	96%	100%	22%
Distorted for FC100-100-10' amplified to stddev=0.1	27%	50%	100%
Gaussian noise with stddev=0.1	2.6%	2.8%	2.7%

Adversarial examples remain hard for models trained even on a disjoint training set.

Szegedy, Christian, et al. "Intriguing properties of neural networks." *arXiv preprint arXiv:1312.6199* (2013).

Explaining and harnessing adversarial examples

YouTube Playlist

$\tilde{x} = x + \eta$
- x — input
- η — adversarial input

The classifier should assign the same class to $\tilde{x}$ and x, so long as $\|\eta\|_\infty < \epsilon$

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

activation grows by $w^T \eta$

$\eta = \text{sign}(w)\epsilon \implies \|\eta\|_\infty = \epsilon$

$$w^T \eta = \sum_{i=1}^{n} w_i \eta_i = \sum_{i=1}^{n} |w_i| \epsilon = \epsilon n m$$

$m \rightarrow$ average magnitude of an element of the weight vector

high-dimensional cases!

"fast gradient sign" method of generating adversarial examples

$\theta \rightarrow$ parameters of a model

$x \rightarrow$ input to the model

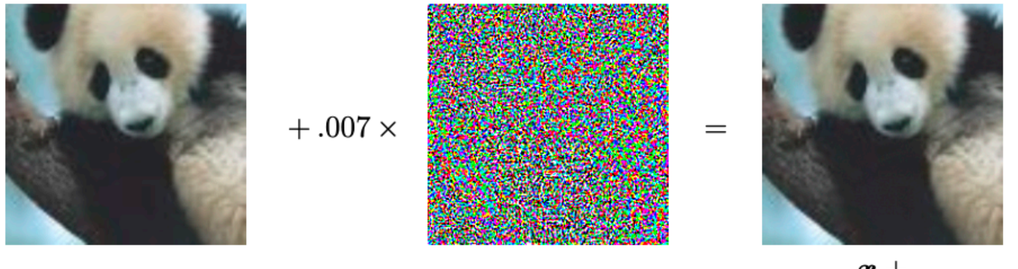
$y \rightarrow$ target associated with x

$J(\theta, x, y) \rightarrow$ cost

$\eta = \text{sign}(\nabla_x J(\theta, x, y))\epsilon$

Adversarial Training

$$\tilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1 - \alpha) J(\theta, x + \text{sign}(\nabla_x J(\theta, x, y))\epsilon, y)$$



$\boldsymbol{x}$
"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence

$\boldsymbol{x} + \epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

$$\tilde{J}(\theta, x, y) \approx J(\theta, x, y) + (1 - \alpha)\epsilon \text{sign}(\nabla_x J(\theta, x, y))^T \nabla_x J(\theta, x, y)$$
$$= J(\theta, x, y) + (1 - \alpha)\epsilon \|\nabla_x J(\theta, x, y)\|_1$$

Logistic Regression

$J = \log(1 + \exp(-y(w^T x + b)))$

$\nabla_x J = -y\sigma(-y(w^T x + b))w$

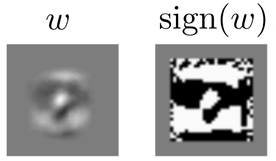
$\|\nabla_x J\|_1 = \|w\|_1$

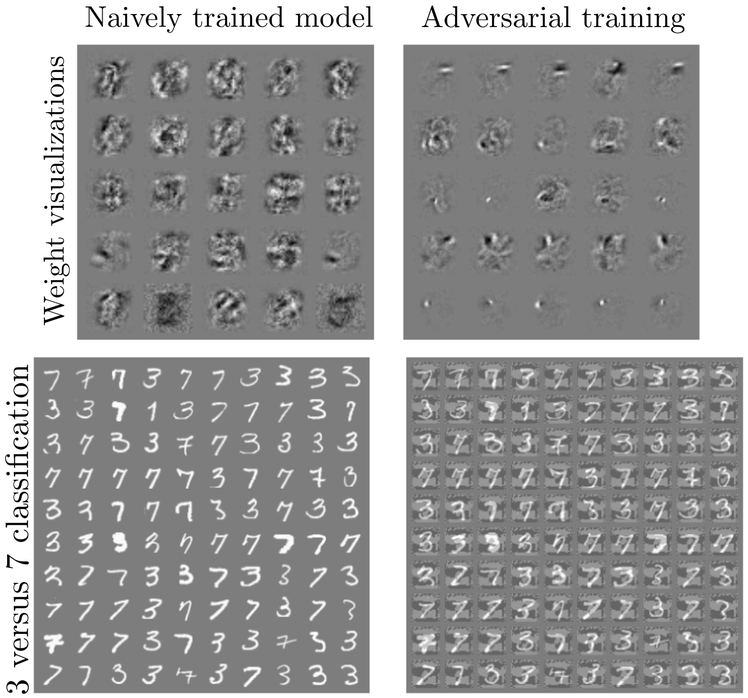
$y \in \{-1, 1\}$

$p(y = 1) = \sigma(w^T x + b)$

$$\sigma(\hat{y}) = \frac{1}{1 + \exp(-\hat{y})} = \frac{\exp(\hat{y})}{\exp(\hat{y}) + 1}$$

$$1 - \sigma(\hat{y}) = \frac{1 + \exp(-\hat{y}) - 1}{1 + \exp(-\hat{y})} = \frac{\exp(-\hat{y})}{\exp(-\hat{y}) + 1} = \sigma(-\hat{y})$$





$p(y = -1) = 1 - \sigma(w^T x + b) = \sigma(-(w^T x + b))$

$p(y) = \sigma(y(w^T x + b))$

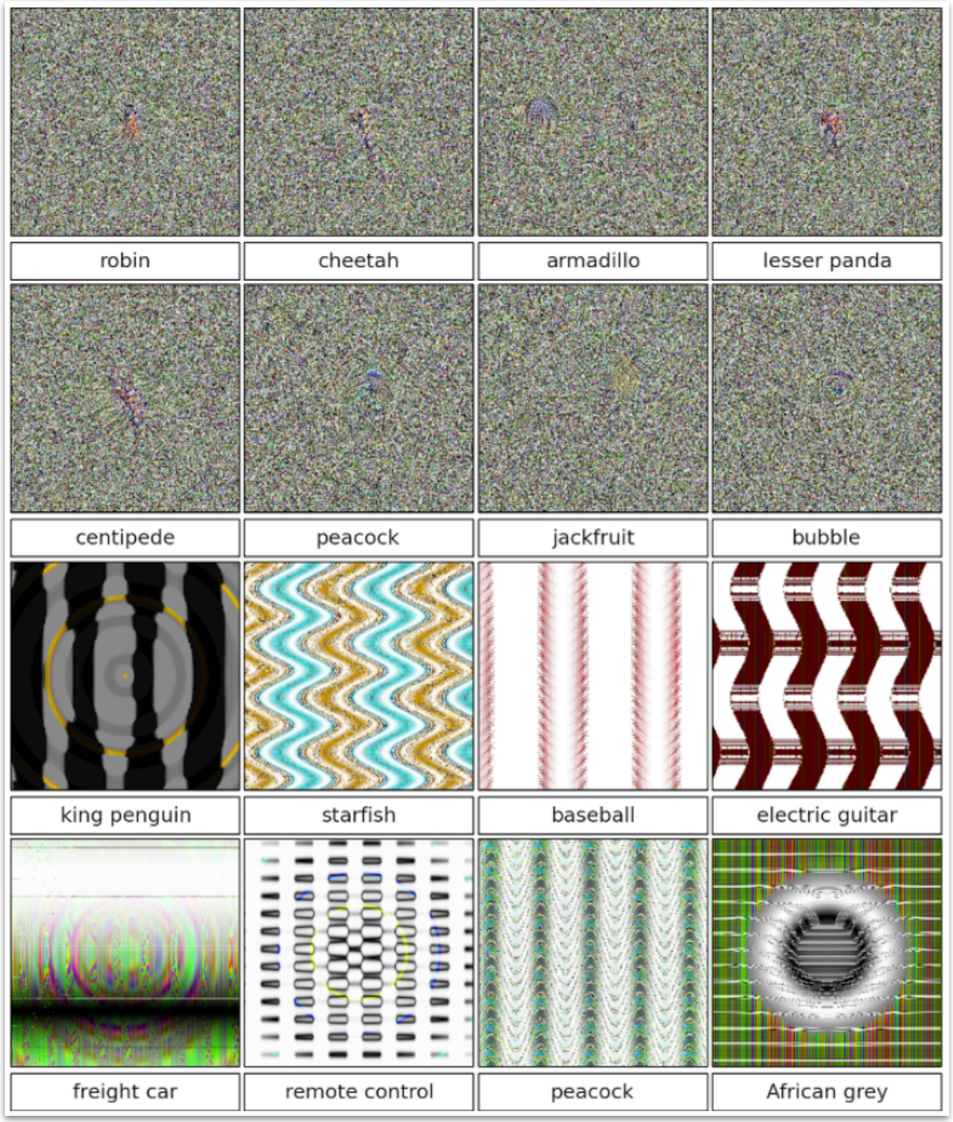
$$-\mathbb{E}_{x,y} \log p(y) = \mathbb{E}_{x,y} \log \frac{1}{p(y)} = \mathbb{E}_{x,y} \log \underbrace{\frac{1}{\sigma(y(w^T x + b))}}_{1 + \exp(-y(w^T x + b))}$$
$$= \mathbb{E}_{x,y} \zeta(-y(w^T x + b))$$

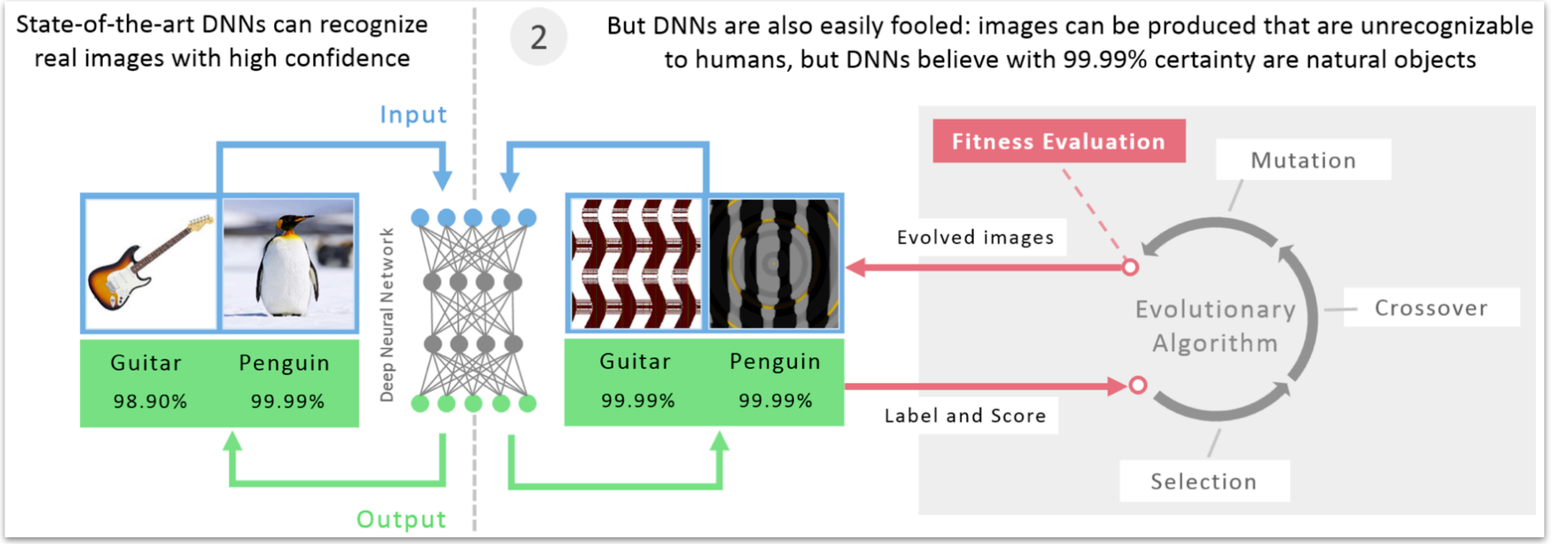
$\zeta(z) = \log(1 + \exp(z)) \rightarrow$ softplus function

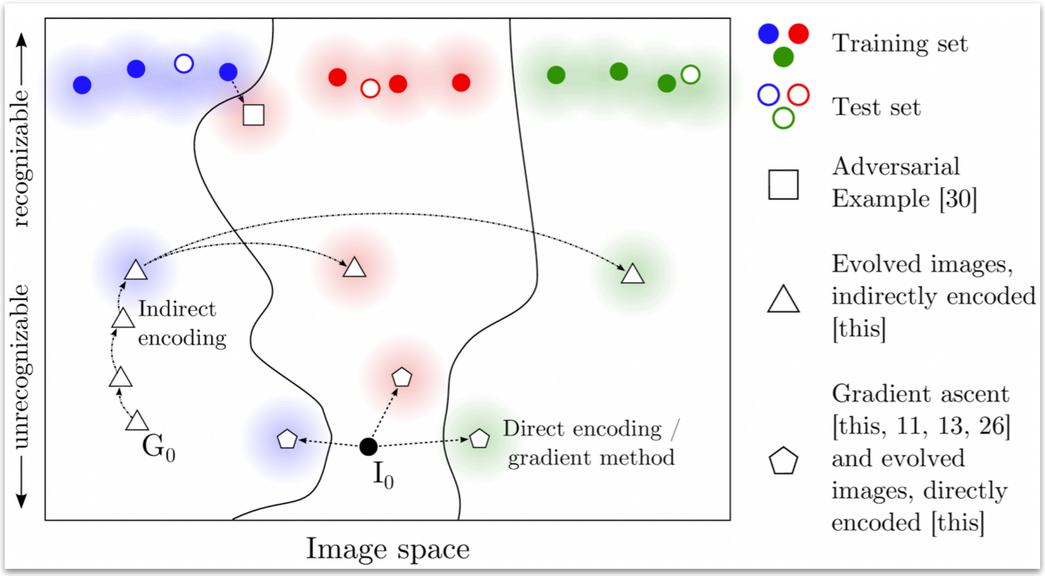
$\zeta'(z) = \sigma(z)$

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." *arXiv preprint arXiv:1412.6572* (2014).

Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images







Directly Encoded

Each pixel value is initialized with uniform random noise within the [0, 255] range. Those numbers are independently mutated.

Indirectly Encoded

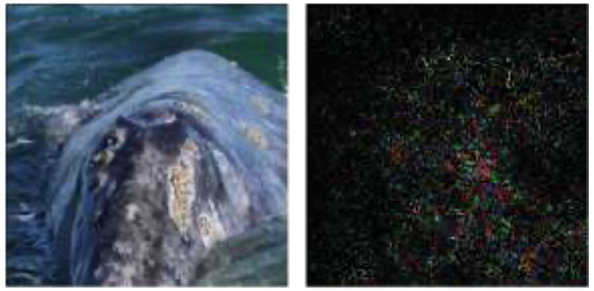
Compositional Pattern-Producing Network (CPPN)

A CPPN takes in the (x, y) position of a pixel as input, and outputs pixel values using sine, sigmoid, Gaussian and linear functions.

Nguyen, Anh, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

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

DeepFool: whale → turtle



Fast Gradient Sign: whale → turtle

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

$\Delta(\boldsymbol{x}; \hat{k})$ ← robustness of classifier $\hat{k}$ at point x

$$\rho_{\text{adv}}(\hat{k}) = \mathbb{E}_{\boldsymbol{x}} \frac{\Delta(\boldsymbol{x}; \hat{k})}{\|\boldsymbol{x}\|_2} \quad \rightarrow \text{robustness of classifier } \hat{k}$$

DeepFool for binary classifiers

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

$f \rightarrow$ an arbitrary scalar-valued image classification function

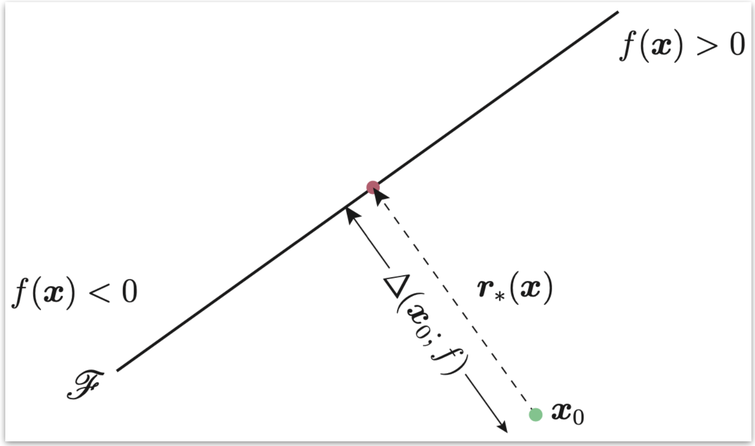
$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$$

$$\boldsymbol{r}_*(\boldsymbol{x}_0) := \arg\min \|\boldsymbol{r}\|_2$$

$$\text{subject to } \text{sign}(f(\boldsymbol{x}_0 + \boldsymbol{r})) \neq \text{sign}(f(\boldsymbol{x}_0))$$

$$= -\frac{f(\boldsymbol{x}_0)}{\|\boldsymbol{w}\|_2^2} \boldsymbol{w}.$$

$$w^T(x_0 - \frac{f(x_0)}{\|w\|_2^2} w) + b = f(x_0) - f(x_0) = 0$$



Algorithm 1 DeepFool for binary classifiers

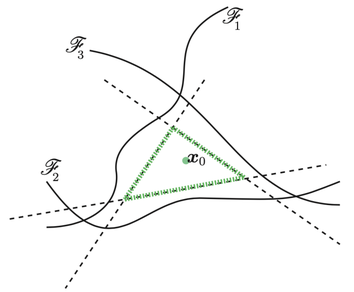
input: Image $\boldsymbol{x}$, classifier f.
output: Perturbation $\hat{\boldsymbol{r}}$.
Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\text{sign}(f(\boldsymbol{x}_i)) = \text{sign}(f(\boldsymbol{x}_0))$ **do**
  $\boldsymbol{r}_i \leftarrow -\frac{f(\boldsymbol{x}_i)}{\|\nabla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

DeepFool for multiclass classifiers

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 $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$



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*	-	-

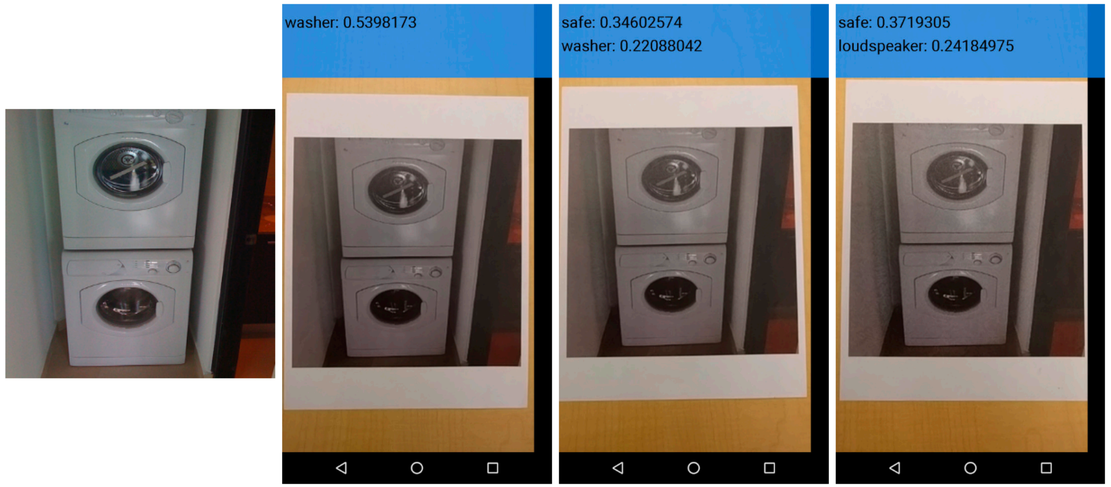
Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Adversarial Examples in the Physical World

YouTube Video

$M \rightarrow$ machine learning system
$C \rightarrow$ clean example
$M(C) = y_{\text{true}} \rightarrow$ correctly classified
$A \rightarrow$ adversarial example (perceptually indistinguishable from C)
$M(A) \neq y_{\text{true}}$
An adversarial example that was designed to be misclassified by a model M_1 is often also misclassified by a model M_2.
– robots perceiving world through cameras and other sensors
– video surveillance systems
– mobile applications for image or sound classification

Do adversarial examples survive transformations (e.g., printing the image and taking a photo of the result)?



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

Methods for generating adversarial examples

$\boldsymbol{X}$ - an image y_{true} - true class for the image $\boldsymbol{X}$

$J(\boldsymbol{X}, y)$ - cross-entropy cost function

$J(\boldsymbol{X}, y) = -\log p(y|\boldsymbol{X}) \rightarrow$ log probability of the class given the image

$$Clip_{X,\epsilon}\{\boldsymbol{X}'\}(x,y,z) = \min\Big\{255, \boldsymbol{X}(x,y,z)+\epsilon, \max\big\{0, \boldsymbol{X}(x,y,z)-\epsilon, \boldsymbol{X}'(x,y,z)\big\}\Big\}$$

will be in L_∞ ϵ-neighbourhood of the source image $\boldsymbol{X}$

Fast Method

$$\boldsymbol{X}^{adv} = \boldsymbol{X} + \epsilon \operatorname{sign}\big(\nabla_X J(\boldsymbol{X}, y_{true})\big) \rightarrow \text{decrease the probability of the true class}$$

Basic Iterative Method

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X}, \quad \boldsymbol{X}_{N+1}^{adv} = Clip_{X,\epsilon}\Big\{\boldsymbol{X}_N^{adv} + \alpha \operatorname{sign}\big(\nabla_X J(\boldsymbol{X}_N^{adv}, y_{true})\big)\Big\}$$

Iterative Least-Likely Class Method

$$y_{LL} = \arg\min_y \big\{p(y|\boldsymbol{X})\big\}$$

least-likely class according to the prediction of the trained network

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X}, \quad \boldsymbol{X}_{N+1}^{adv} = Clip_{X,\epsilon}\big\{\boldsymbol{X}_N^{adv} - \alpha \operatorname{sign}\big(\nabla_X J(\boldsymbol{X}_N^{adv}, y_{LL})\big)\big\}$$

increase the probability of the least likely class

Destruction Rate

$$d = \frac{\sum_{k=1}^{n} C(\boldsymbol{X}^k, y_{true}^k)\overline{C(\boldsymbol{X}_{adv}^k, y_{true}^k)}C(T(\boldsymbol{X}_{adv}^k), y_{true}^k)}{\sum_{k=1}^{n} C(\boldsymbol{X}^k, y_{true}^k)\overline{C(\boldsymbol{X}_{adv}^k, y_{true}^k)}}$$

Fraction of adversarial images which are no longer misclassified after the transformations!

$$C(\boldsymbol{X}, y) = \begin{cases} 1, & \text{if image } \boldsymbol{X} \text{ is classified as } y; \\ 0, & \text{otherwise.} \end{cases}$$

$\overline{C(\boldsymbol{X}, y)} = 1 - C(\boldsymbol{X}, y) \rightarrow$ binary negation

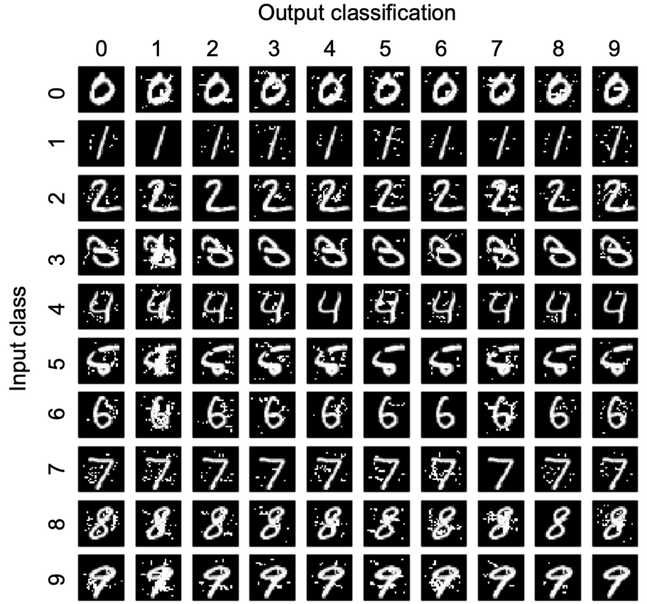
$n \rightarrow$ number of images used to compute the destruction rate

Adversarial method	Average case		Prefiltered case	
	top-1	top-5	top-1	top-5
fast $\epsilon = 16$	12.5%	40.0%	5.1%	39.4%
fast $\epsilon = 8$	33.3%	40.0%	14.6%	70.8%
fast $\epsilon = 4$	46.7%	65.9%	32.4%	91.2%
fast $\epsilon = 2$	61.1%	63.2%	49.5%	91.9%
iter. basic $\epsilon = 16$	40.4%	69.4%	60.0%	87.8%
iter. basic $\epsilon = 8$	52.1%	90.5%	64.7%	91.2%
iter. basic $\epsilon = 4$	52.4%	82.6%	77.5%	94.1%
iter. basic $\epsilon = 2$	71.7%	81.5%	80.8%	96.9%
l.l. class $\epsilon = 16$	72.2%	85.1%	87.5%	97.9%
l.l. class $\epsilon = 8$	86.3%	94.6%	88.9%	97.0%
l.l. class $\epsilon = 4$	90.3%	93.9%	91.9%	98.0%
l.l. class $\epsilon = 2$	82.1%	93.9%	93.1%	98.0%

Kurakin, Alexey, Ian Goodfellow, and Samy Bengio. "Adversarial examples in the physical world." (2016).

The Limitations of Deep Learning in Adversarial Settings

YouTube Video



$f : x \mapsto y$

↳ Deep Neural Network

$x \rightarrow$ raw feature vector

$y \rightarrow$ output vector

$x^* = x + \delta_x \rightarrow$ adversarial sample

$\delta_x \rightarrow$ perturbation vector

$\arg\min_{\delta_x} \|\delta_x\|$ s.t. $f(x^*) = y^* \neq y$

Adversarial Saliency Maps

$\text{label}(x) = \arg\max_j f_j(x)$

↳ score of class j

Modify x s.t. it is misclassified as $t \neq \text{label}(x)$

Increase $f_t(x)$ and decrease $f_j(x), \forall j \neq t$

Finding features that the adversary should increase to achieve misclassification!

$$J_{ij}(x) := \frac{\partial f_j(x)}{\partial x_i} \qquad i \rightarrow \text{an input feature}$$

$$s(i; x, t) := 0 \text{ if } J_{it}(x) < 0 \text{ or } \sum_{j \neq t} J_{ij}(x) > 0$$

$$:= J_{it}(x) | \sum_{j \neq t} J_{ij}(x) | \text{ otherwise}$$

Algorithm 1 Crafting adversarial samples

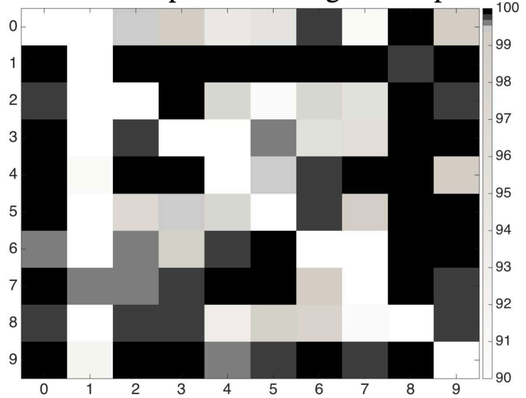
$\mathbf{X}$ is the benign sample, $\mathbf{Y}^*$ is the target network output, $\mathbf{F}$ is the function learned by the network during training, Υ is the maximum distortion, and θ is the change made to features. This algorithm is applied to a specific DNN architecture and dataset in Algorithm 2.

Input: $\mathbf{X}$, $\mathbf{Y}^*$, $\mathbf{F}$, Υ, θ

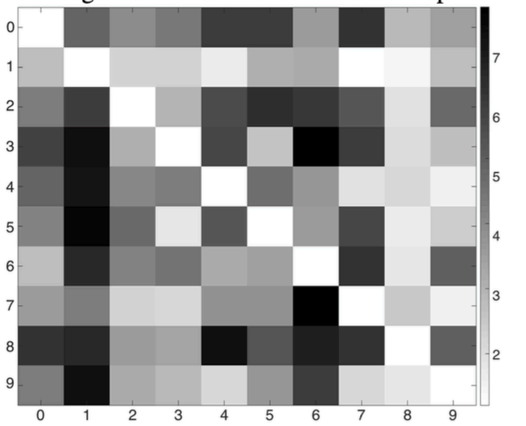
1: $\mathbf{X}^* \leftarrow \mathbf{X}$
2: $\delta_{\mathbf{X}} \leftarrow \vec{0}$
3: $\Gamma = \{1 \dots |\mathbf{X}|\}$
4: **while** $\mathbf{F}(\mathbf{X}^*) \neq \mathbf{Y}^*$ and $||\delta_{\mathbf{X}}|| < \Upsilon$ **do**
5: Compute forward derivative $J_{\mathbf{F}}(\mathbf{X}^*)$
6: $S(\mathbf{X}, \mathbf{Y}^*) = \texttt{saliency\_map}\,(J_{\mathbf{F}}(\mathbf{X}^*), \Gamma, \mathbf{Y}^*)$
7: $i_{max} \leftarrow \arg\max_i S(\mathbf{X}, \mathbf{Y}^*)[i]$
8: Modify $\mathbf{X}^*_{i_{max}}$ by θ
9: $\delta_{\mathbf{X}} \leftarrow \mathbf{X}^* - \mathbf{X}$
10: **end while**
11: **return** $\mathbf{X}^*$

$$H(s,t) \approx \sum_{k=1}^{K-1} (\tau_{k+1} - \tau_k) \frac{\varepsilon(s,t,\tau_{k+1}) + \varepsilon(s,t,\tau_k)}{2}$$

Success rate per source-target class pair.



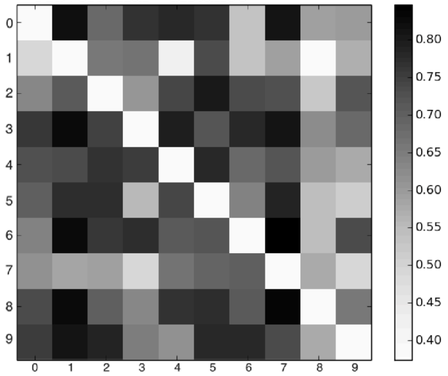
Average distortion ε of successful samples



Adversarial Distance

$$A(\mathbf{X}, t) = 1 - \frac{1}{M} \sum_{i \in 1..M} 1_{S(\mathbf{X},t)[i]>0}$$

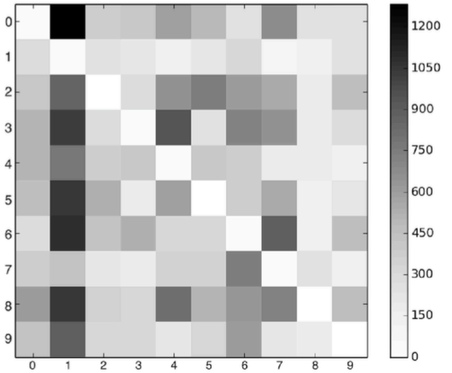
Adversarial distance



Robustness

$$R(\mathbf{F}) = \min_{(\mathbf{X},t)} A(\mathbf{X}, t)$$

Hardness matrix



Hardness measure

$$H(s,t) = \int_\tau \varepsilon(s,t,\tau) \mathrm{d}\tau$$

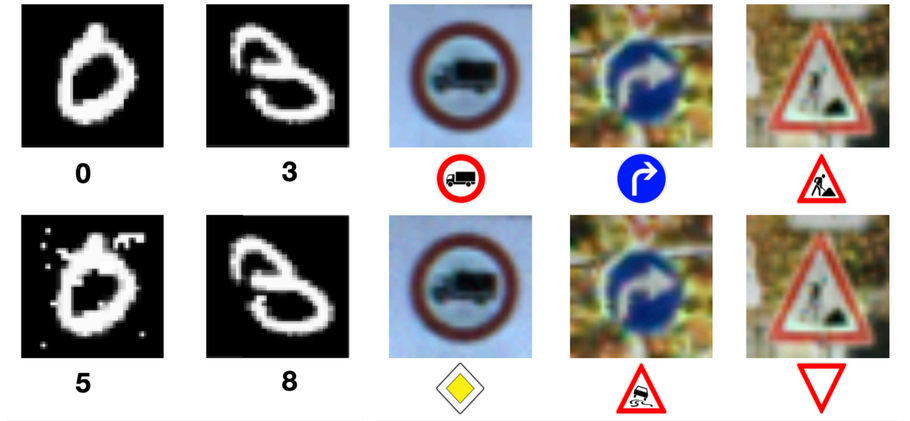
↳ average distortion of a set of samples for the corresponding success rate τ

$(\varepsilon_k, \tau_k), k = 1, \dots, K \rightarrow$ correspond to different Υ_k

Papernot, Nicolas, et al. "The limitations of deep learning in adversarial settings." *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2016.

Practical Black-Box Attacks against Machine Learning

YouTube Playlist



Algorithm 1 - Substitute DNN Training: for oracle $\tilde{O}$, a maximum number max_ρ of substitute training epochs, a substitute architecture F, and an initial training set S_0.

Input: $\tilde{O}$, max_ρ, S_0, λ

Define architecture F
for $\rho \in 0 \,..\, max_\rho - 1$ **do**
  // Label the substitute training set
  $D \leftarrow \left\{(\vec{x}, \tilde{O}(\vec{x})) : \vec{x} \in S_\rho\right\}$
  // Train F on D to evaluate parameters θ_F
  $\theta_F \leftarrow \text{train}(F, D)$
  // Perform Jacobian-based dataset augmentation
  $S_{\rho+1} \leftarrow \{\vec{x} + \lambda \cdot \text{sgn}(J_F[\tilde{O}(\vec{x})]) : \vec{x} \in S_\rho\} \cup S_\rho$
end for
return θ_F

Black-Box Attacks: The adversary (a) has no information about the structure or parameters of the DNN, and (b) does not have access to any large training dataset.

The adversary is only capable of observing labels assigned by the DNN for chosen inputs.

Attack Strategy: Train a local substitute DNN with a "synthetic" dataset.

$O \rightarrow$ Oracle (targeted DNN)

$x \rightarrow$ input

$\tilde{O}(x) \rightarrow$ label (the class assigned the largest probability by O)

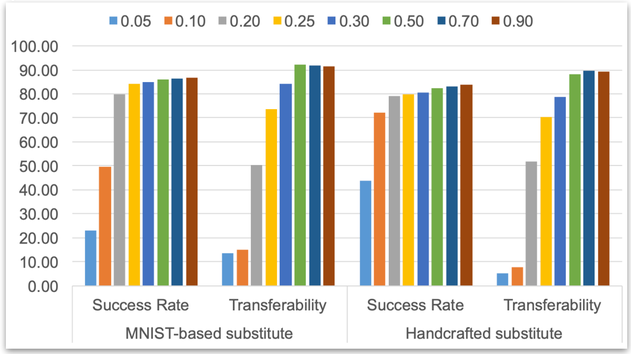
$\tilde{O}(x) = \arg\max_j O_j(x)$

$O(x) \rightarrow$ probability vector

$x^* = x + \delta_x \rightarrow$ adversarial example

$\delta_x = \arg\min\{z : \tilde{O}(x+z) \neq \tilde{O}(x)\}$

$F \rightarrow$ substitute network (learn similar decision boundary to O)

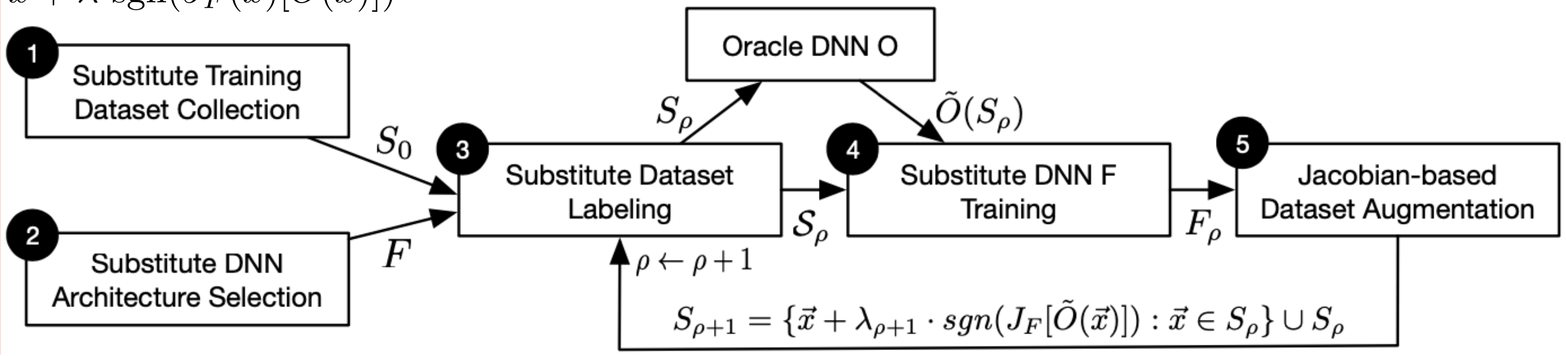


Jacobian-based Dataset Augmentation

$J_F(x) = \left[\frac{\partial F_j(x)}{\partial x_i}\right]_{i,j} \rightarrow$ model's Jacobian matrix

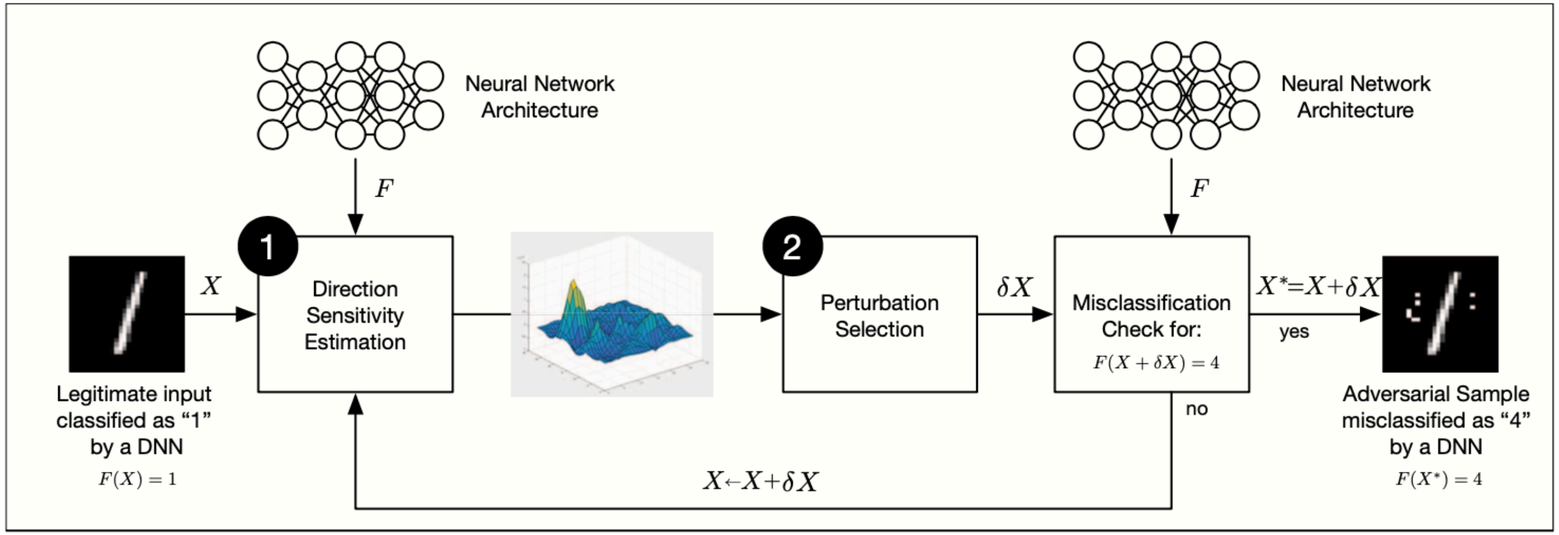
$x + \lambda\ \text{sgn}(J_F(x)[\tilde{O}(x)])$

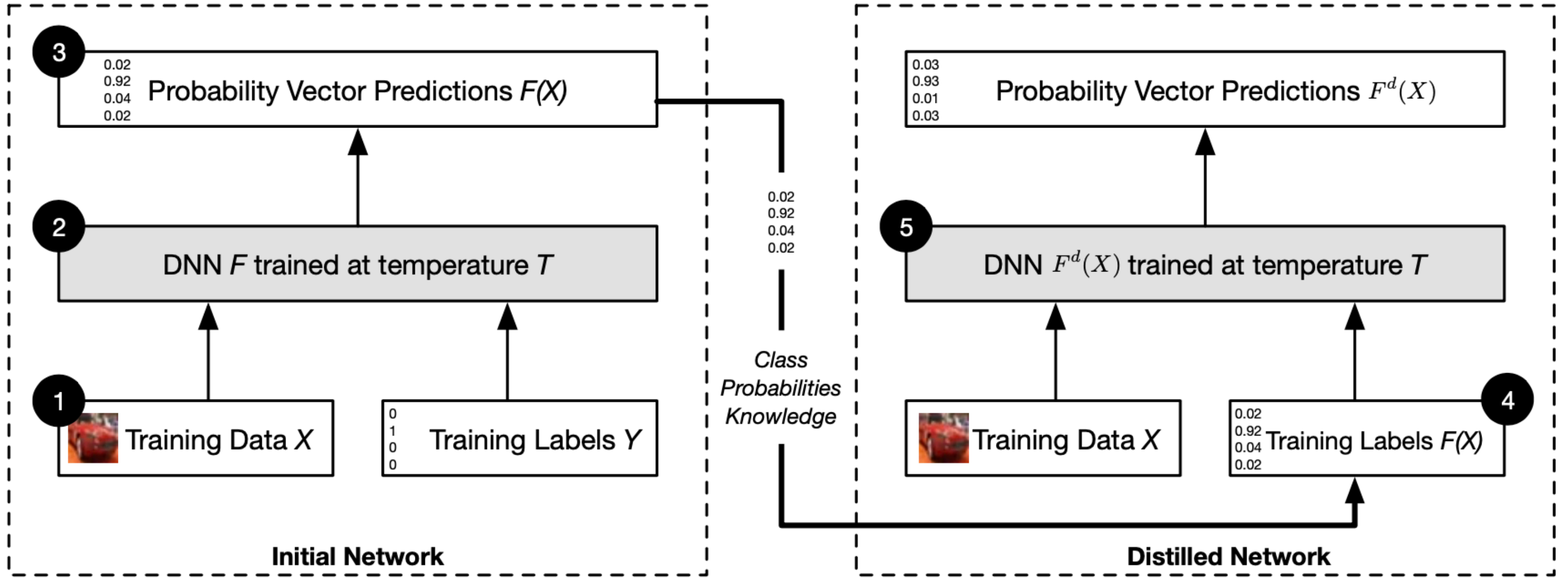
Adversarial Sample Crafting
– Fast Gradient Sign Method
– Adversarial Saliency Maps



Papernot, Nicolas, et al. "Practical black-box attacks against machine learning." Proceedings of the 2017 ACM on Asia conference on computer and communications security. 2017.

Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks





Robustness

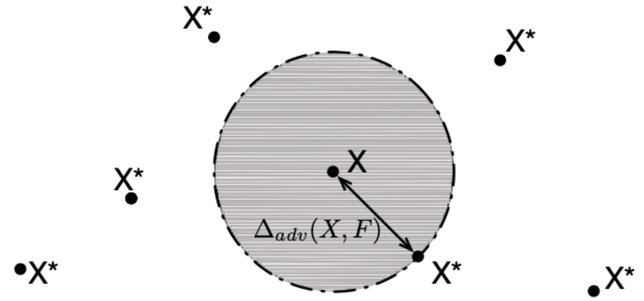
Robustness is the average minimal perturbation required to produce an adversarial sample from the distribution modeled by F.

$$\rho_{adv}(F) = E_\mu[\Delta_{adv}(X, F)]$$

$$\Delta_{adv}(X, F) = \arg\min_{\delta X}\{\|\delta X\| : F(X + \delta X) \neq F(X)\}$$

$X \rightarrow$ inputs drawn from distribution μ that DNN architecture is modeling with F

$\Delta_{adv}(X, F) \rightarrow$ minimum perturbation required to misclassify sample X in each of the other classes



$$\rho_{adv}(F) \simeq \frac{1}{|\mathcal{X}|}\sum_{X\in\mathcal{X}} \min_{\delta X} \|\delta X\|$$

$\delta X \rightarrow$ evaluated by considering each of the 9 possible adversarial targets corresponding to sample $X \in \mathcal{X}$

$\mathcal{X} \rightarrow$ test set

Distillation increases robustness!

Papernot, Nicolas, et al. "Distillation as a defense to adversarial perturbations against deep neural networks." *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016.

Towards Evaluating the Robustness of Neural Networks

Boulder

YouTube Playlist

Targeted Adversarial Examples

$x \to$ input

$c^*(x) \to$ correct label for x

$t \to$ target

$t \neq c^*(x)$

$x' \to$ adversarial input

$c(x') \to$ predicted label for x'

$c(x') = t$

x & x' are close according to some distance metric

Untargetted Adversarial Examples

$c(x') \neq c^*(x)$

Attack Algorithm

$$\min_\delta \quad \mathcal{D}(x, x+\delta)$$
$$\text{s.t.} \quad c(x+\delta) = t$$
$$x + \delta \in [0,1]^n$$

$\mathcal{D} \to$ some distance metric (e.g., L_0, L_2, L_∞)

L_0 distance: number of coordinates i such that $x_i \neq x'_i$

number of pixels that are different

Define f s.t. $c(x+\delta) = t$ iff $f(x+\delta) \leq 0$

$$f(x') = \left(\max_{i \neq t} z(x')_i - z(x')_t\right)^+$$

$x' = x + \delta$

$$c(x') = \arg\max_i \text{softmax}(z(x'))_i$$

$$(\cdot)^+ = \max(\cdot, 0)$$

$$f(x') \leq 0 \iff \max_{i \neq t} z(x')_i \leq z(x')_t$$

$$\min_\delta \quad \mathcal{D}(x, x+\delta)$$
$$\text{s.t.} \quad f(x+\delta) \leq 0$$
$$x + \delta \in [0,1]^n$$

$$\min_\delta \quad \mathcal{D}(x, x+\delta) + \lambda f(x+\delta)$$
$$\text{s.t.} \quad x + \delta \in [0,1]^n$$

$$\min_\delta \quad \|\delta\|_p + \lambda f(x+\delta)$$
$$\text{s.t.} \quad x + \delta \in [0,1]^n$$

$$\delta = \frac{1}{2}(\tanh(w) + 1) - x$$

$$\implies 0 \leq x + \delta \leq 1$$

Defensive Distillation

$$\text{softmax}(x, T)_i = \frac{\exp(x_i/T)}{\sum_j \exp(x_j/T)}$$

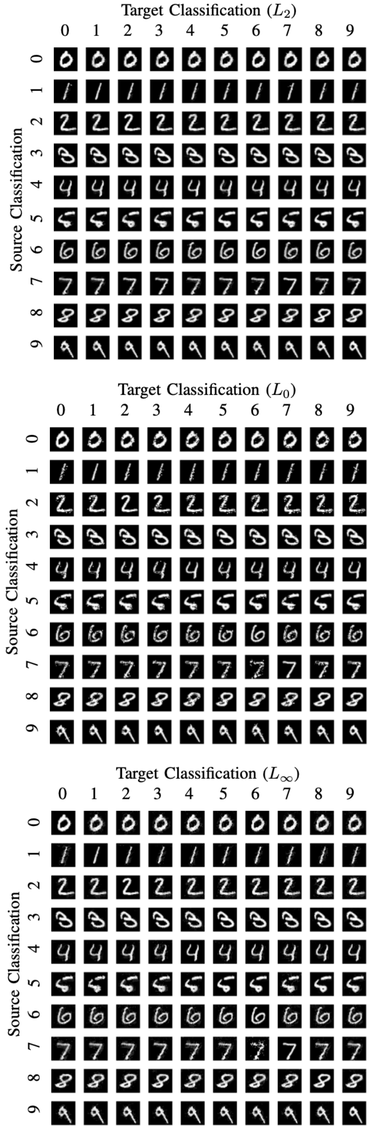
$T \to$ temperature

$$\lim_{T \to 0} \text{softmax}(x, T) = \max(x)$$

$$\lim_{T \to \infty} \text{softmax}(x, T) = \text{uniform}(x)$$

1. Train the teacher network by setting T as the temperature
2. Compute soft labels (temperature T)
3. Train the distilled network on the soft labels (temperature T)
4. Use the distilled network during testing (Temperature 1)

Teacher & distilled networks have the same architecture



Carlini, Nicholas, and David Wagner. "Towards evaluating the robustness of neural networks." *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017.

Towards Deep Learning Models Resistant to Adversarial Attacks

YouTube Playlist

– autonomous cars
– face recognition
– malware detection

$\mathcal{D} \rightarrow$ data distribution
$x \in \mathbb{R}^d \rightarrow$ examples
$y \in \{1, \ldots, k\} \rightarrow$ labels
$L(\theta, x, y) \rightarrow$ loss function
$\mathbb{E}_{(x,y)\sim\mathcal{D}}[L(\theta, x, y)] \rightarrow$ empirical risk

Take an example x belonging to class c_1 as input and find examples x^{adv} such that x^{adv} is very close to x but the model incorrectly classifies x^{adv} as belonging to class $c_2 \neq c_1$.

saddle point problem

$$\min_\theta \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{S}} L(\theta, x+\delta, y)\right]$$

$\mathcal{S} \subset \mathbb{R}^d \rightarrow$ set of allowed perturbations

– robust optimization

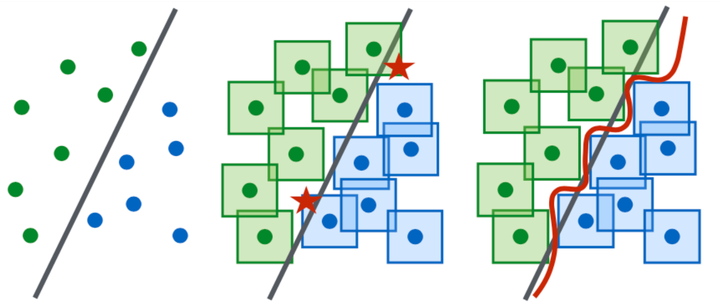
Fast Gradient Sign Method (FGSM)

L_∞ bounded adversary

$x + \varepsilon\,\text{sgn}(\nabla_x L(\theta, x, y))$

Projected Gradient Ascent (PGA)

$x^{t+1} = \Pi_{x+\mathcal{S}}\left(x^t + \alpha\,\text{sgn}(\nabla_x L(\theta, x, y))\right)$



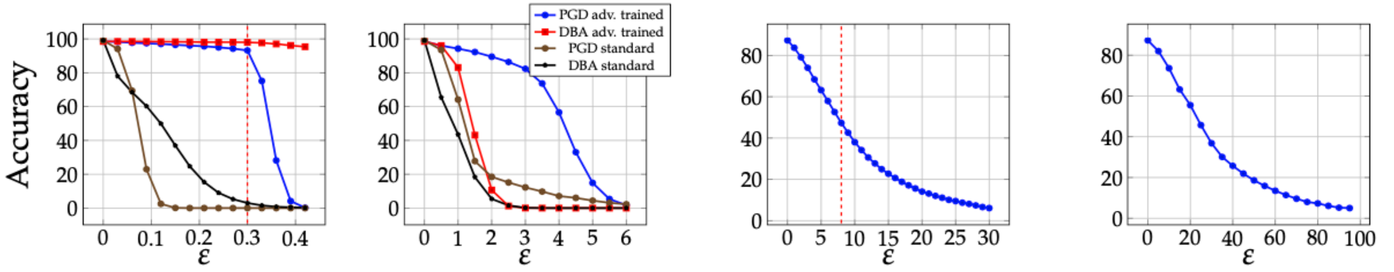
Method	Steps	Source	Accuracy
Natural	-	-	87.3%
FGSM	-	A	56.1%
PGD	7	A	50.0%
PGD	20	A	**45.8%**
CW	30	A	46.8%
FGSM	-	A′	67.0%
PGD	7	A′	**64.2%**
CW	30	A′	78.7%
FGSM	-	A_{nat}	85.6%
PGD	7	A_{nat}	86.0%

black-box attack
A_{nat}: copy of the network trained on natural examples

white-box attack
A: The network itself

black-box attack
A′: independently trained copy of the network
B: different architecture

Method	Steps	Restarts	Source	Accuracy
Natural	-	-	-	98.8%
FGSM	-	-	A	95.6%
PGD	40	1	A	93.2%
PGD	100	1	A	91.8%
PGD	40	20	A	90.4%
PGD	100	20	A	**89.3%**
Targeted	40	1	A	92.7%
CW	40	1	A	94.0%
CW+	40	1	A	93.9%
FGSM	-	-	A′	96.8%
PGD	40	1	A′	96.0%
PGD	100	20	A′	**95.7%**
CW	40	1	A′	97.0%
CW+	40	1	A′	96.4%
FGSM	-	-	B	**95.4%**
PGD	40	1	B	96.4%
CW+	-	-	B	95.7%



(a) MNIST, ℓ_∞-norm (b) MNIST, ℓ_2-norm (c) CIFAR10, ℓ_∞-norm (d) CIFAR10, ℓ_2-norm

Madry, Aleksander, et al. "Towards deep learning models resistant to adversarial attacks." *arXiv preprint arXiv:1706.06083* (2017).

Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples

Boulder

YouTube Video

International Conference on Learning Representations (ICLR) 2018 Defenses

1. Thermometer Encoding

The purpose of thermometer encoding is to break linearity.

$x \rightarrow$ image

$x_{i,j,c} \rightarrow$ pixel color

$$\hat{\tau}(x_{i,j,c})_k = \min(\max(x_{i,j,c} - k/l, 0), 1)$$

$\tau(x_{i,j,c}) \in \mathbb{R}^l \rightarrow l$-level thermometer encoding

$\tau(x_{i,j,c})_k = 1$ if $x_{i,j,c} > \frac{k}{l}$

$= 0$ otherwise

$$\tau(x_{i,j,c})_k = \text{floor}(\hat{\tau}(x_{i,j,c})_k) \quad \text{let } g(x) = \hat{\tau}(x)$$

$\tau(0.66) = 1111110000 \rightarrow$ 10-level thermometer encoding

2. Input Transformations

(a) randomly drop pixels and restore them by performing total variance minimization; and

(b) image quilting: reconstruct images by replacing small patches with patches from "clean" images.

3. Local Intrinsic Dimensionality (LID)

$\overrightarrow{\text{LID}}(x) = \{\text{LID}_{d_j}(x)\}_{j=1}^n$ $\quad f^{1..j} \rightarrow$ composition of layers 1 through j

$d_j(x, y) = \left\| f^{1..j}(x) - f^{1..j}(y) \right\|_2$

$$\text{LID}_d(x) = -\left(\frac{1}{k} \sum_{i=1}^{k} \log \frac{r_i(x)}{r_k(x)} \right)^{-1}$$

$r_i(x) \rightarrow$ distance between sample x and its i-th nearest neighbor

Train a logistic regression classifier to detect adversarial examples!

4. Stochastic Activation Pruning (SAP)

```
SAP randomly drops some neurons of each layer with probability proportional to
their absolute value. Values which are retained are scaled up (as is done in
dropout) to retain accuracy.
```

5. Mitigating through Randomization

For a classifier that takes a 299×299 input, the defense first randomly rescales the image to a $r \times r$ image, with $r \in [299, 331)$, and then randomly zero-pads the image so that the result is 331×331.

6. PixelDefend

PixelDefend proposes using a PixelCNN generative model to project a potential adversarial example back onto the data manifold before feeding it into a classifier.

7. Defense-GAN

Defense-GAN uses a Generative Adversarial Network to project samples onto the manifold of the generator before classifying them.

Obfuscated Gradients

– shattered gradients: non-existent or incorrect

– stochastic gradients: randomized network or input

– vanishing/exploding gradients

Backward Pass Differentiable Approximation (BPDA)

$f^i \rightarrow$ non-differentiable (or not usefully-differentiable) layer

Find a differentiable approximation g to f^i, and perform backward pass by replacing f^i by g.

Expectation over Transformation (EoT)

$$\nabla \mathbb{E}_{t \sim T} f(t(x)) = \mathbb{E}_{t \sim T} \nabla f(t(x))$$

Reparameterization

vanishing/exploding gradients

Athalye, Anish, Nicholas Carlini, and David Wagner. "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples." *arXiv preprint arXiv:1802.00420* (2018).

Ensemble Adversarial Training: Attacks and Defenses

$x \in [0,1]^d \rightarrow$ input data

$y_{\text{true}} \in \mathbb{Z}_k \rightarrow$ label

$\mathcal{D} \rightarrow$ data distribution

$h \in \mathcal{H} \rightarrow$ hypothesis (model)

$h(x) \in \mathbb{R}^k \rightarrow$ class scores

$L(h(x), y) \rightarrow$ loss function (e.g., cross-entropy)

Threat Model

For some target model $h \in \mathcal{H}$ and inputs (x, y_{true}) the adversary's goal is to find an adversarial example x^{adv} such that x^{adv} and x are "close" yet the model misclassifies x^{adv}.

$$\text{s.t. } \|x^{\text{adv}} - x\|_\infty \leq \epsilon$$

White-box adversaries: Access to the target model's parameters (i.e., h)

Black-box adversaries: Access to only partial information about the model's inner workings.

Although security against white-box attacks is the stronger notion (and the one we ideally want ML models to achieve), black-box security is a reasonable and more tractable goal for deployed ML models.

Adversarial Training

$$h^* = \arg\min_{h \in \mathcal{H}} \mathbb{E}_{(x, y_{\text{true}}) \sim \mathcal{D}} \left[\max_{\|x^{\text{adv}} - x\|_\infty \leq \epsilon} L(h(x^{\text{adv}}), y_{\text{true}}) \right]$$

Adversarial training with single-step methods (e.g., Fast Gradient Sign Method, Single-Step Least-Likely Class Method, etc.) admits a degenerate global minimum, wherein the model's loss can not be reliably approximated by a linear function.

Ensemble Adversarial Training

Decoupling the attacks from the model being trained!

Augment a model's training data with adversarial examples crafted on other static pre-trained models.

	Top 1			Top 5		
Model	**Clean**	**Step-LL**	**Max. Black-Box**	**Clean**	**Step-LL**	**Max. Black-Box**
v3	**22.0**	69.6	51.2	**6.1**	42.7	24.5
v3_{adv}	**22.0**	**26.6**	40.8	**6.1**	**9.0**	17.4
$\text{v3}_{\text{adv-ens3}}$	23.6	30.0	**34.0**	7.6	**10.1**	**11.2**
$\text{v3}_{\text{adv-ens4}}$	24.2	43.3	**33.4**	7.8	19.4	**10.7**
IRv2	**19.6**	50.7	44.4	**4.8**	24.0	17.8
IRv2_{adv}	**19.8**	**21.4**	34.5	**4.9**	**5.8**	11.7
$\text{IRv2}_{\text{adv-ens}}$	**20.2**	26.0	**27.0**	**5.1**	7.6	**7.9**

Error rates (in %) for Ensemble Adversarial Training on ImageNet.

v3 $\rightarrow$ Inception v3

IRv2 $\rightarrow$ Inception ResNet v2

Max. Black-Box: worst-case error over a series of black-box attacks (Step-LL, R+Step-LL, FGSM, I-FGSM, PGD) transferred from some holdout models

Randomized single-step attack (R+FGSM & R+Step-LL)

$$x' = x + \alpha \cdot \texttt{sign}(\mathcal{N}(\mathbf{0}^d, \mathbf{I}^d))$$

$$x^{\text{adv}} = x' + (\varepsilon - \alpha) \cdot \texttt{sign}\left(\nabla_{x'} J(x', y_{\text{true}})\right) \qquad \alpha < \varepsilon$$

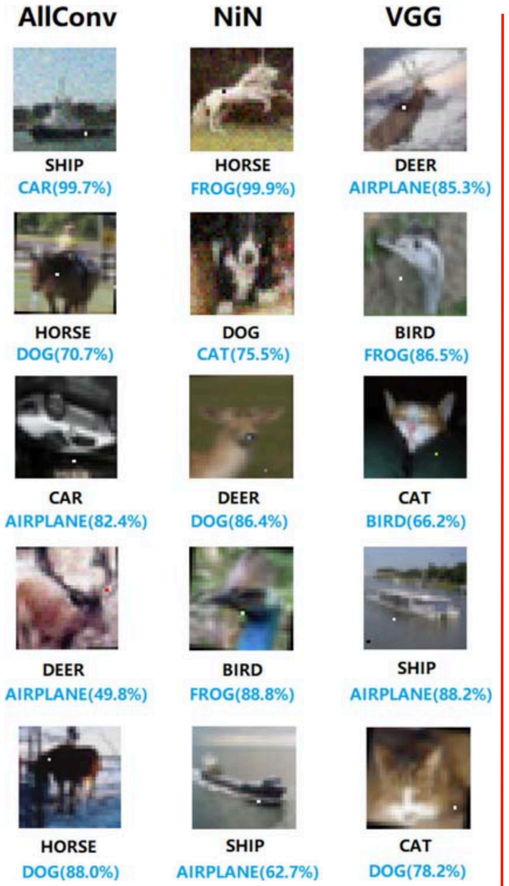
Subsequent work found that more elaborate black-box attacks could significantly enhance transferability and reduce effectiveness of ensemble adversarial training.

Adversarial training with **multi-step attacks** is currently regarded as the state-of-the-art approach for attaining robustness to a fixed type of perturbations, whether in a white-box or black-box setting.

Tramèr, Florian, et al. "Ensemble adversarial training: Attacks and defenses." *arXiv preprint arXiv:1705.07204* (2017).

One Pixel Attack for Fooling Deep Neural Networks

YouTube Playlist



$f \rightarrow$ target image classifier
$x = (x_1, \ldots, x_n) \rightarrow$ original image correctly classified as class t
$f_t(x) \rightarrow$ probability of x belonging to class t

$e(x) = (e_1, \ldots, e_n) \rightarrow$ additive adversarial perturbation

$$\max_{e^*(x)} \quad f_{\text{adv}}(x + e(x))$$

$$\text{s.t.} \quad \|e(x)\| \leq L$$ ← limitation of maximum modification

adv $\rightarrow$ target class
– black-box attack

$$\max_{e^*(x)} \quad f_{\text{adv}}(x + e(x))$$

$$\text{s.t.} \quad \|e(x)\|_0 \leq d \text{ (e.g., d = 1)}$$

Differential Evolution (DE)

```
DE is a population-based optimization algorithm for
solving complex multi-modal optimization problems.

DE belongs to the general class of evolutionary
algorithms (EAs).
```

$$\{\underbrace{(x_1, y_1, r_1, g_1, b_1)}_{\mathcal{X}_1}, \ldots, (x_d, y_d, r_d, g_d, b_d)\}$$ ← candidate solution

Initially $x_i, y_i \sim \underbrace{U(1, 32)}_{\text{CIFAR-10}}$ or $\underbrace{U(1, 227)}_{\text{ImageNet}}$

$r_i, g_i, b_i \sim \mathcal{N}(\mu = 128, \sigma = 127)$

parents: current population (400)

children: candidate solutions (400)

$\mathcal{X}_i(g+1) = \mathcal{X}_j(g) + \lambda(\mathcal{X}_k(g) - \mathcal{X}_\ell(g))$
$j \neq k \neq \ell \rightarrow$ random numbers
$\mathcal{X}_i \rightarrow$ an element of the candidate solution
$g \rightarrow$ current index of generation
$\lambda = 0.5 \rightarrow$ scale parameter
fitness function = prob. label of the target class (CIFAR-10)
= prob. label of the true class (ImageNet)
fitness function $> 90\% \implies$ stop (CIFAR-10, targeted attack)
fitness function $< 5\% \implies$ stop (ImageNet, non-targeted attack)
max # iter = 100


Cup(16.48%)
Soup Bowl(16.74%)


Bassinet(16.59%)
Paper Towel(16.21%)


Teapot(24.99%)
Joystick(37.39%)


Hamster(35.79%)
Nipple(42.36%)


Original image (dog)

CIFAR-10, One Pixel Attack

	AllConv	NiN	VGG16	BVLC
OriginAcc	85.6%	87.2%	83.3%	57.3%
Targeted	19.82%	23.15%	16.48%	–
Non-targeted	68.71%	71.66%	63.53%	16.04%
Confidence	79.40%	75.02%	67.67%	22.91%

CIFAR-10

	3 pixels	5 pixels
Success rate(tar)	40.57%	44.00%
Success rate(non-tar)	86.53%	86.34%
Rate/Labels	79.17%	77.09%

Airplane	Automobile	Bird
Cat	Deer	Frog
Horse	Ship	Truck

Target classes

Su, Jiawei, Danilo Vasconcellos Vargas, and Kouichi Sakurai. "One pixel attack for fooling deep neural networks." *IEEE Transactions on Evolutionary Computation* 23.5 (2019): 828-841.

Boulder

Questions?

YouTube Playlist