Adversarial Examples for Deep Neural Networks

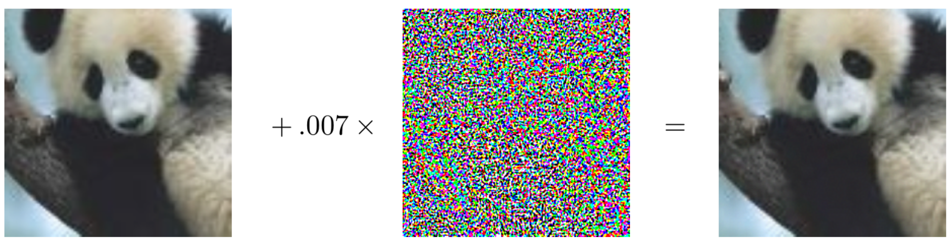
by Paul Hand
Northeastern University

Outline:

- Adversarial examples
- White box attacks
- Black box attacks
- Real-world attacks
- Adversarial Training

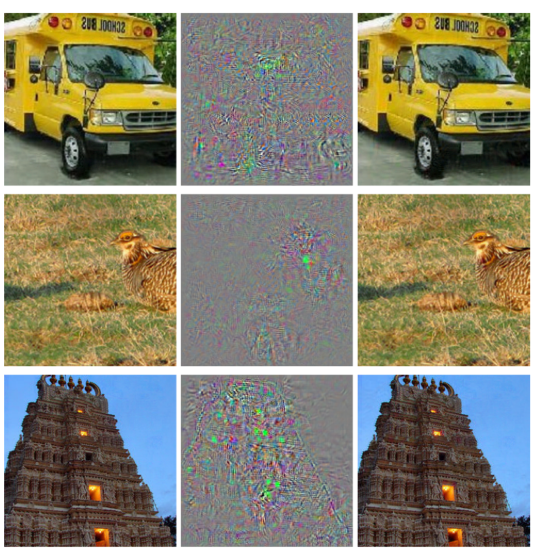
Adversarial examples

(Goodfellow et al. 2015)



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

GoogLeNet gets this image wrong, but a human gets it right



AlexNet classifies these as "ostrich".

(Szegedy et al. 2014)

Formulation

There is a trained neural net classifier

Net is fixed

$$f_\theta(x) = y$$

net · parameters · image · probability distribution over classes

This net assigns to x the class $C(x) = \arg\max_i y_i$

For some image x, find perturbation δ

such that $C(x+\delta) \neq C(x)$ untargeted

or

$C(x+\delta) = t$ targeted

Want: $x+\delta$ to appear to a human as class $C(x)$

Adversarial Examples:

- Targeted vs untargeted
- White box vs black box vs no box
- Imperceptible vs perceptible
- Digital vs physical
- Specific vs universal
- Attack vs defense

What is the meaning and an example of each of the following concepts:

Targeted vs Untargeted

In targeted attacks, we desire the system to output a specific erroneous class
- Build a pair of glasses to make systems think I am Brad Pitt

In an untargeted attack, we only desire the system to be wrong
- Simply make a point about DL methods

White box vs black box vs no box

White box - have access to classifiers, models, weights, can differentiate model
    If the model got leaked (self driving car company might have had a security breach)
Black box - have access to the classifiers (but not the parameters), can not differentiate
    Access to an API
No box - have no access to the classifier

Imperceptible vs perceptible

Imperceptible - a human can not determine that the image was modified
Perceptible - a human can determine that the image was modified
    Sticker on the stop sign
    T-shirt that fools a person detector

Digital vs physical

Physical attack - you are changing the real world
Digital attach - you are changing pixels in an image

Specific vs universal

Attack a single image or a signal classifier vs attacking set of images or set of classifiers

Attack vs defense

Attack - modify the image to get a misclassification
Defense - train a network so that someone can’t modify an image to get misclassification

White Box Attacks

Projected Gradient Descent

Given img x, model θ

Solve $X^* = \arg\max_{x' \in P_x} L(f(x'), y)$

maximization

Set of images w/ small perturbations from x

Cross entropy loss

true label of x

where $P_x = \{ x' \mid \|x' - x\|_\infty < \varepsilon \}$ for example

Could be other p-norms

Small constant

$\|x\|_\infty = \max_i |x_i|$

Why maximize loss with respect to the true label?

Want system to misclassify the image. We trained the net to minimize loss (which did maximimum likelihood optimization). Instead, we will maximize loss (minimize the likelihood of a correct classification)

Why constrain the optimization?

If we desire an imperceptible perturbation, we need to enforce it.
Without any constraint, the image may simply output garbage (which would not fool a human)

What does constraining the optimization with P_x do?

Ensures that each pixel does not change by more than epsilon.

Is this formulation targeted or untargeted?

It is untargeted. It was never provided a target class as a parameter.

Write down a formulation that is targeted/~~untargeted~~.

Target class t

$$\min_{x' \in P_x} L(f(x'), t)$$

Projected Gradient ~~Descent~~ Ascent

To Solve:

$$x_{t+1} = \Pi\left(x_t + \eta_t \underbrace{\nabla_x}_{} L(f(x_t), y)\right)$$

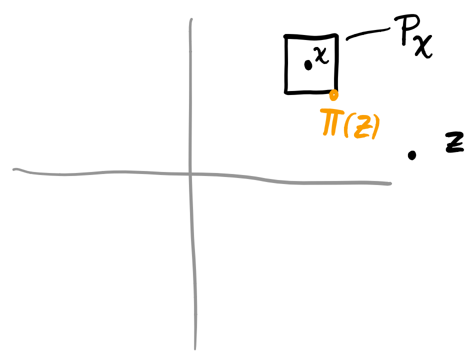
projector on to P_x

step size

gives direction of maximal ascent of L wrt x, evaluated at x_t

L^∞ ball around X

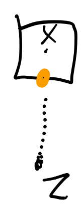
w/ $\Pi(z) = \underset{x' \in P_x}{\arg\min} \|x' - z\|_2$ — closest point in P_x to z

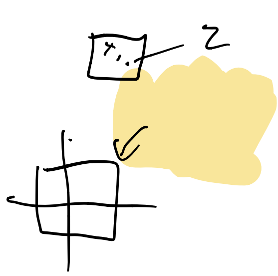


With Pi(z), do all points z map to a corner of the L^inf ball?

No

L^∞ ball $\{x' \mid \|x' - x\|_\infty < \varepsilon\}$





$\Pi(z) = z$

$\Pi(z) =$

$$\Pi(z)_i = \begin{cases} z_i & \text{if } |z_i| < \varepsilon \\ \varepsilon \, \text{sgn}(z_i) & \text{o/wise} \end{cases}$$

Fast gradient sign method (FGSM)

(Goodfellow et al. 2015)

$$X^* = X + \varepsilon \operatorname{sgn} \nabla_x L(x, y)$$

Special case of PGD

roughly projecting on ℓ_∞ ball

Noniterative $\Rightarrow$ fast

This method roughly performs projected gradient descent. Explain.

This is like one step of projected gradient descent method, but it is scaled in order to achieve a perturbation of L^infinity norm epsilon.

In what sense is this method non-iterative?

It is just a single formula for the adversarial example. It does not require sequential updates (like in PGD). Consequently, it is very fast.

Carlini-Wagner attack

(Carlini + Wagner 2017)

Want: $\min_\delta \|\delta\|_p$ st. $\underbrace{C(x+\delta) = t}_{\text{hard to work with this constraint}}$

$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$

Suppose we have access to classifier f

Notation: $Z = f(x)$ is class logits

take positive part

Solve: $\min_\delta \|\delta\|_p$ st $\left(\underbrace{\max_{i \neq t} Z(x+\delta)_i}_{\text{largest logit other than class } t} - Z(x+\delta)_t\right)^+ \leq 0$

Penalized form: $\min_\delta \|\delta\|_p + \lambda \left(\max_{i \neq t} Z(x+\delta)_i - Z(x+\delta)_t\right)^+$

Is this targeted or untargeted?

Targeted. We are given target class t. We constrain (in a soft way) the problem to output a perturbation that gets classified as t.

Design a variant that is ~~targeted~~/untargeted.

Let y be true class

$$\min_\delta \|\delta\|_p - \lambda\left(\max_{i \neq y} Z(x+\delta)_i - Z(x+\delta)_y\right)^-$$

Commonalities of methods so far:

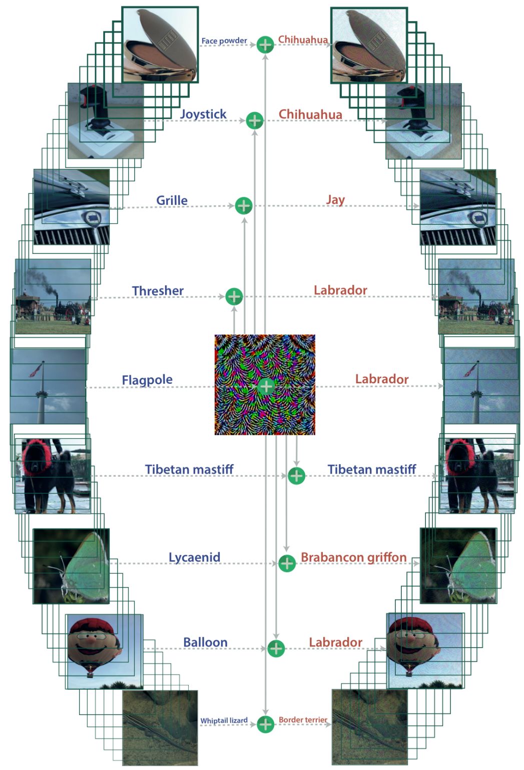
- Requires gradients of classifier (white box)
- Has variants for targeted and untargeted attacks
- Adversarial perturbation computed for a single image + classifier

Universal Adversarial Perturbations

(Moosavi-Dezfooli et al. 2017)

Find δ st $X+\delta$ is misclassified for most images X

δ is a universal or image agnostic perturbation



Algorithm 1 Computation of universal perturbations.

input: Data points X, classifier $\hat{k}$, desired ℓ_p norm of the perturbation ξ, desired accuracy on perturbed samples δ.
output: Universal perturbation vector v.
Initialize $v \leftarrow 0$.
while $\mathrm{Err}(X_v) \leq 1-\delta$ **do**
  for each datapoint $x_i \in X$ **do**
    if $\hat{k}(x_i+v) = \hat{k}(x_i)$ **then**
      Compute the *minimal* perturbation that sends x_i+v to the decision boundary:

$$\Delta v_i \leftarrow \arg\min_r \|r\|_2 \text{ s.t. } \hat{k}(x_i+v+r) \neq \hat{k}(x_i).$$

      Update the perturbation:

$$v \leftarrow \mathcal{P}_{p,\xi}(v+\Delta v_i).$$

    end if
  end for
end while

X_v - data set perturbed by v
$\{x_1+v, x_2+v, \cdots\}$

$\mathrm{Err}(X_v)$ - fraction of misclassified images in perturbed dataset

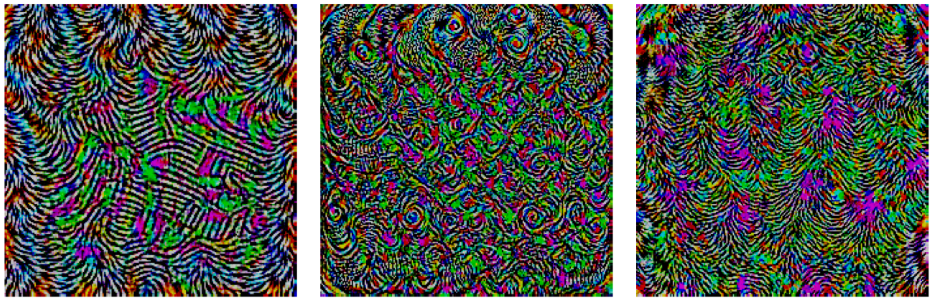
multiple ways to solve

project onto ℓ_p ball of radius ξ

What does it mean to project onto the l_p ball of radius xi?

Roughly speaking, how is a universal perturbation built?

Examples of universal perturbations



(d) VGG-19 (e) GoogLeNet (f) ResNet-152

Universal Perturbations generalize across architectures

	VGG-F	CaffeNet	GoogLeNet	VGG-16	VGG-19	ResNet-152
VGG-F	**93.7%**	71.8%	48.4%	42.1%	42.1%	47.4 %
CaffeNet	74.0%	**93.3%**	47.7%	39.9%	39.9%	48.0%
GoogLeNet	46.2%	43.8%	**78.9%**	39.2%	39.8%	45.5%
VGG-16	63.4%	55.8%	56.5%	**78.3%**	73.1%	63.4%
VGG-19	64.0%	57.2%	53.6%	73.5%	**77.8%**	58.0%
ResNet-152	46.3%	46.3%	50.5%	47.0%	45.5%	**84.0%**

Table 2: Generalizability of the universal perturbations across different networks. The percentages indicate the fooling rates. The rows indicate the architecture for which the universal perturbations is computed, and the columns indicate the architecture for which the fooling rate is reported.

You can attack an unknown classifier by training your own (with a different architecture) and running a white box method

Why do you suspect that adversarial examples can generalize across different` architectures?

Black box attacks

Cant backprop / differentiate the classifier you are attacking

You may have access to logit output or perhaps only to predictions or nothing

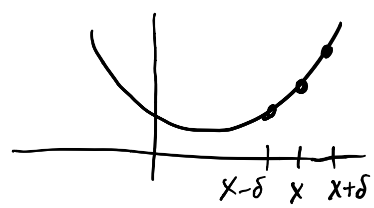
Approaches:

Zeroth order Optimization (ZOO)
Transferability Attacks

ZOO in general:

To compute $\min_{X \in \mathbb{R}^n} g(X)$ w/o derivatives:

1d case:



with the values of g at these 3 points, can estimate $f'(X)$ & $g''(X)$.

$$g'(x) \simeq \frac{g(X+\delta) - g(X-\delta)}{2\delta}$$

$$g''(X) \simeq \frac{g(X+\delta) - 2g(X) + g(X-\delta)}{\delta^2}$$

Model as a parabola and find it's minimum.

Higher dim case: Stochastic Coordinate Descent

Choose a random coordinate e_i

Compute $\min_{s} g(x + s e_i)$ as in 1d

ZOO for adversarial examples: (w/ logits)

Use stochastic coordinate descent on CW formulation (Chen et al. 2017)

ZOO for adversarial examples: (w/ only class labels)

Randomized gradient free (RGF) method

(Cheng et al. 2018)

In adversarial examples, accurate gradients are not needed. (eg FGSM)

Transferability Attacks (Liu et al. 2017)

Ensemble approach

Train multiple classifiers w/ different architectures

Try to fool average (logit) output over the ensemble

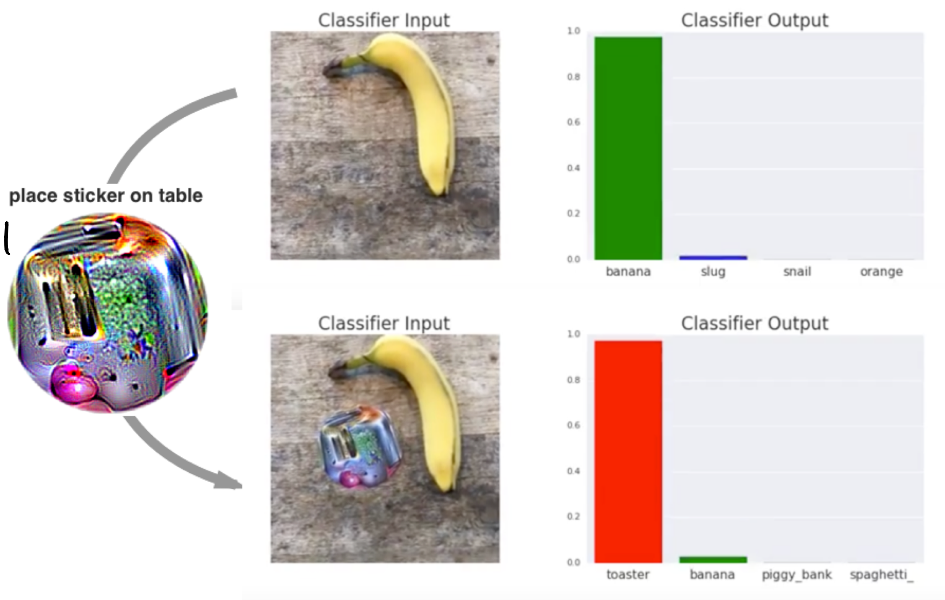
	RMSD	ResNet-152	ResNet-101	ResNet-50	VGG-16	GoogLeNet
-ResNet-152	17.17	0%	0%	0%	0%	0%
-ResNet-101	17.25	0%	1%	0%	0%	0%
-ResNet-50	17.25	0%	0%	2%	0%	0%
-VGG-16	17.80	0%	0%	0%	6%	0%
-GoogLeNet	17.41	0%	0%	0%	0%	5%

Table 4: Accuracy of non-targeted adversarial images generated using the optimization-based approach. The first column indicates the average RMSD of the generated adversarial images. Cell (i, j) corresponds to the accuracy of the attack generated using four models except model i (row) when evaluated over model j (column). In each row, the minus sign "$-$" indicates that the model of the row is not used when generating the attacks. Results of top-5 accuracy can be found in the appendix (Table 14).

Real World Attacks

(Brown et al. 2018)

Adversarial Patch



Patch is so visually salient, a classifier ignores the rest of the image

Challenges:

- Can't change all pixels
- Needs to work for all backgrounds
- Must be robust to physical transformation

Build a model for transformations:

A( ,  , location, $\underbrace{\text{rotation, scale,...}}_{T}$) = 

δ $\quad x \quad$ ℓ

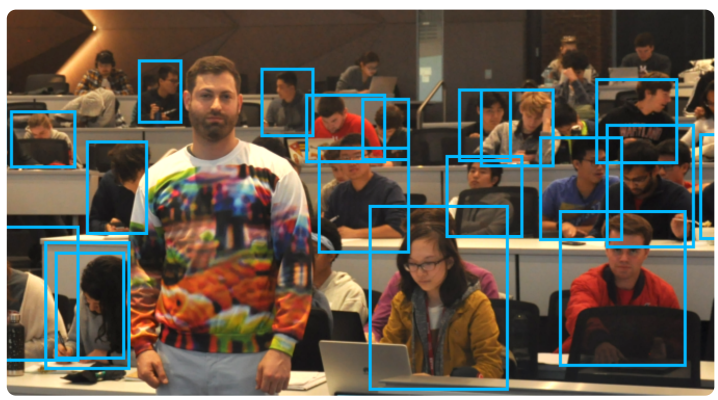
Find adversorial patch δ by choosing target class t:

$$\underset{\delta}{\arg\max}\ \mathbb{E}_{x,\ell,T} \log P(\text{ img } A(\delta, x, \ell, T) \text{ is class } t)$$

Can make it further robust by using ensembles

Other examples:

(Sharif et al. 2016)



(Wu et al. 2019)

(Eykholt et al. 2018)

Adversarial Training

(Goodfellow et al. 2015)

Train a classifier and try to ensure adversarial perturbations get correctly classified

Example for FGSM:

Instead of optimizing $L(\theta, x, y)$ at train time,

Optimize

$$\underbrace{\alpha L(\theta, x, y) + (1-\alpha)}_{\text{weighted combination}} L(\theta, \underbrace{x + \varepsilon \operatorname{sgn} \nabla_x L(\theta, x, y)}_{\text{FGSM adversarial example}}, y)$$

Challenge:

Wont be robust to other methods

Game of Cat and mouse