Adversarial Machine Learning

Bo Li
University of Illinois at Urbana-Champaign

Course logistics

Class information & resources

- Course website: https://aisecure.github.io/TEACHING/2019.html
- Piazza: UIUC, CS 598, slack
- Office hours: after class each day (but not the first week). Please sign up in advance for a 10 -minute slot on the course doodle website
- My office: Siebel 4310
- TA: Boxin Wang: boxin.wang@outlook.com

Prerequisites & Enrollment

- All enrolled students must have taken machine learning classes
- Projects will require training neural networks with standard automatic differentiation packages (TensorFlow, Pytorch)
- Goal: Every group (max 2) in the class should have one top-tier conference paper for your project!

Grading Policy

Criteria	Percent of Grade
Project	60%
(Initial Proposal, Due 9.23)	(5%)
(Status Report, Due 10.28)	(15%)
(Final Report & Presentation, Due 12.14)	(40%)
Paper reading and presentation	30%
(Paper reviews)	(10%)
(Presentation)	(15%)
(Peer rating)	(5%)
Class participation	10%

Possible Hacking days:

- Attack/defense competition
- Privacy/defense competition
- Other ideas? Vote on slack ☺

What we will cover

- Syllabus on course website
- Different types of machine learning algorithms
- Different types of attacks (different perturbation bounds, different semantics)
- Different types of detection/defense methods
- Privacy problems in machine learning
- Fairness of machine learning
- Robustness of ML
- Secure learning vs. semantic based learning
- Open problems, research talks, invited lectures

What we will not cover

- NO how to train GANs
- NO which network is more accurate on ImageNet
- NO playing RL games

“Homework” today

- Sign up for slack
- Start to form your final project group (maximum 2). If you prefer to work alone, it is also good
- Check out which topic you would like to present papers about and do project for (don’t need to be the same)
- Each student can choose two topics for presentation, but one topic for project
- Each class will have two presenters
- Please confirm with TA for your presentation topics by the end of next week
- Future: Please sign up and we need to sync before your presentation

What is adversarial learning, and why should we care?

Perils of Stationary Assumption

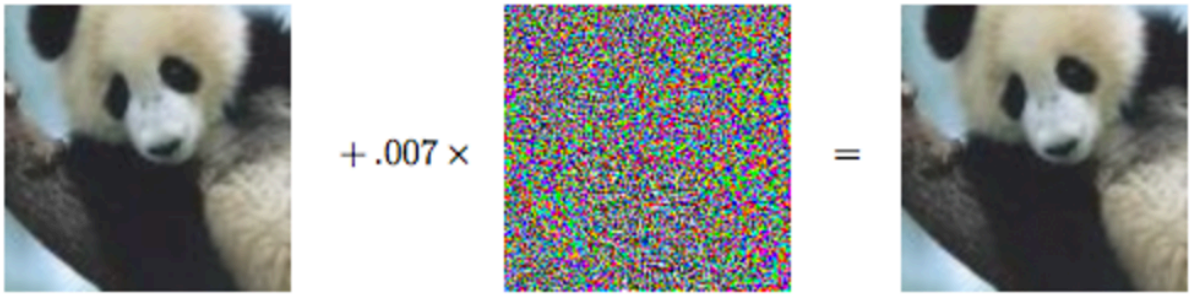
Traditional machine learning approaches assume

Training Data 

$\approx$

Testing Data 

Adversarial Examples



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

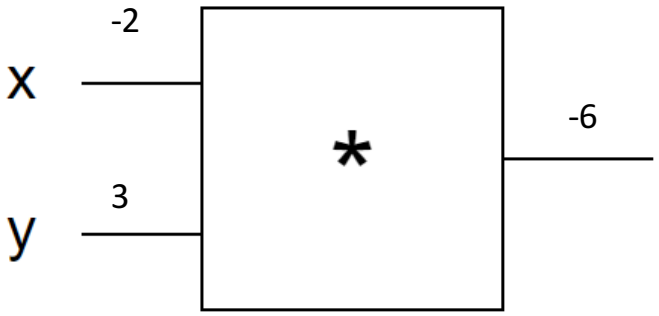
Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." *ICLR 2015.*

Li, Bo, Yevgeniy Vorobeychik, and Xinyun Chen. "A General Retraining Framework for Scalable Adversarial Classification." ICLR. (2016).

Deep Learning Mini Crash Course

- Neural Networks Background
- Convolutional Neural Networks (CNNs)

Real-Valued Circuits



$$f(x, y) = xy$$

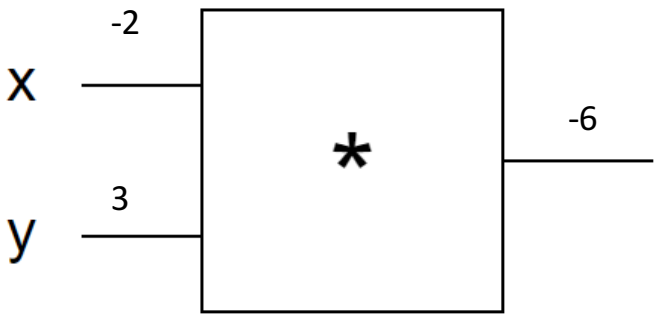
Goal: How do I increase the output of the circuit?

- Tweak the inputs. But how?

- Option 1. Random Search?

x = x + step_size * random_value
y = y + step_size * random_value

Real-Valued Circuits



$$f(x, y) = xy$$

Goal: How do I increase the output of the circuit?

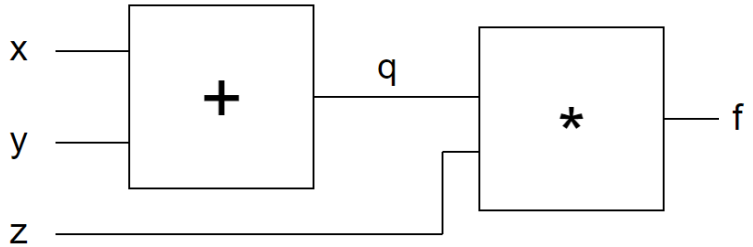
- Option 2. Analytic Gradient

$$\frac{\partial f(x,y)}{\partial x} = \frac{f(x+h,y) - f(x,y)}{h}$$

Limit as h -> 0

x = x + step_size * x_gradient
y = y + step_size * y_gradient

Composable Real-Valued Circuits



$$f(x, y, z) = (x + y)z$$

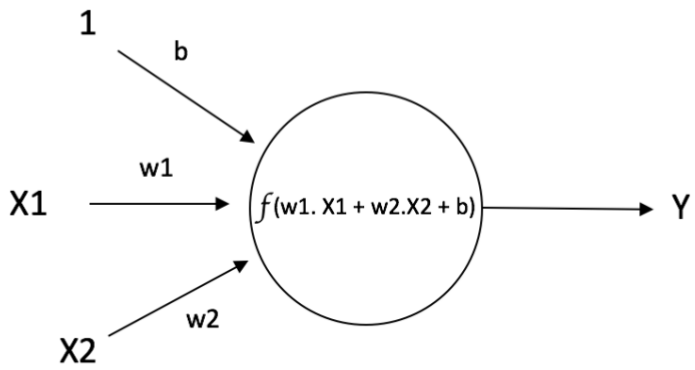
$$f(q, z) = qz \quad \Longrightarrow \quad \frac{\partial f(q,z)}{\partial q} = z, \qquad \frac{\partial f(q,z)}{\partial z} = q$$

$$q(x, y) = x + y \quad \Longrightarrow \quad \frac{\partial q(x,y)}{\partial x} = 1, \qquad \frac{\partial q(x,y)}{\partial y} = 1$$

Chain Rule $$\frac{\partial f(q,z)}{\partial x} = \frac{\partial q(x,y)}{\partial x}\frac{\partial f(q,z)}{\partial q}$$

Backpropagation!

Single Neuron



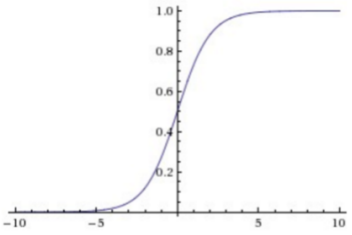
(Input 1)

(Input 2)

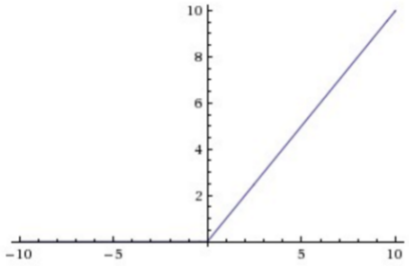
(Output)

Output of neuron = Y= f(w1. X1 + w2.X2 + b)

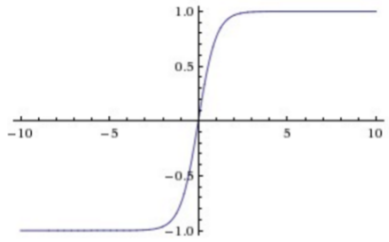
Activation function



Sigmoid

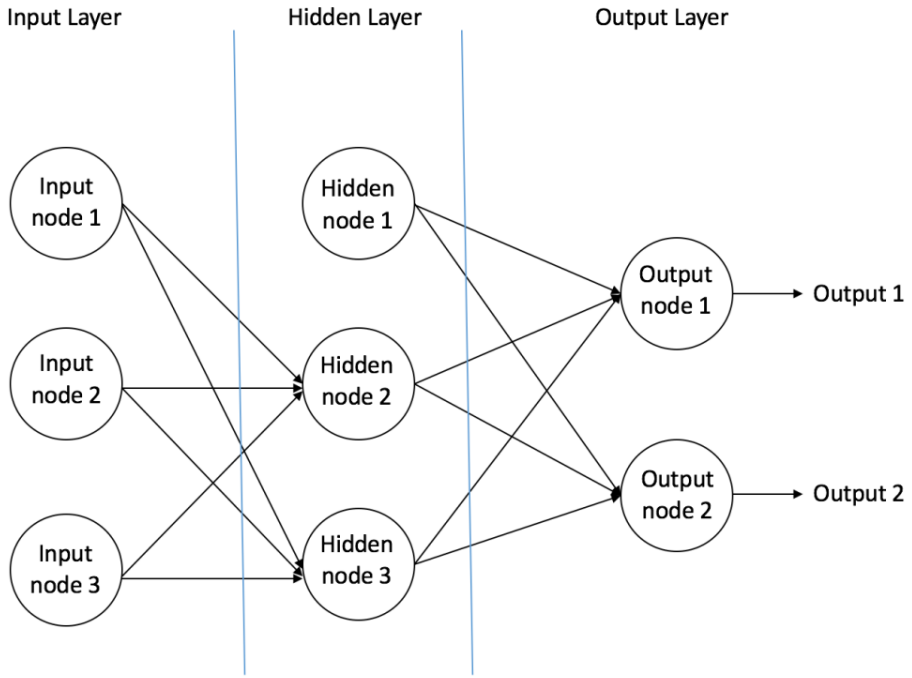


ReLU



tanh

(Deep) Neural Networks!



Organize neurons into a structure

Train (Optimize) using backpropagation

Convolutional Neural Networks (CNNs)

Very widely used, and very useful

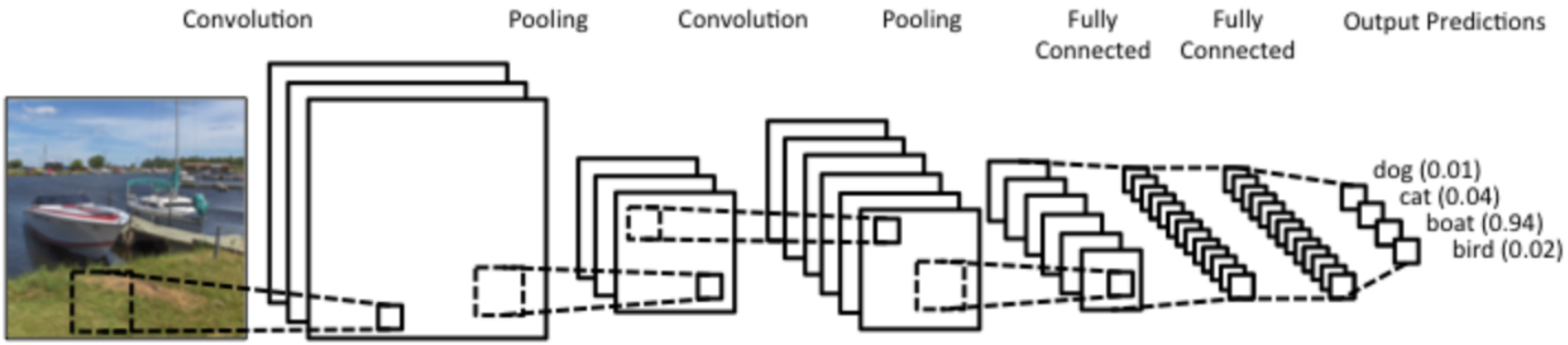
a plate with a sandwich and a salad

a group of motorcycles parked in front of a building

a man riding a wave on top of a surfboard

http://cs.stanford.edu/people/karpathy/neuraltalk2/demo.html

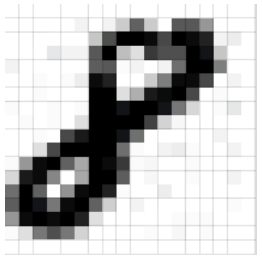
Convolutional Neural Networks (CNNs)



A CNN generally consists of 4 types of architectural units

Convolution
Non Linearity (RELU)
Pooling or Subsampling
Classification (Fully Connected Layers)

How is an image represented for NNs?



- Matrix of numbers, where each number represents pixel intensity
- If image is colored, then there are three channels per pixel, each channel representing (R, G, B) values

Convolution Operator

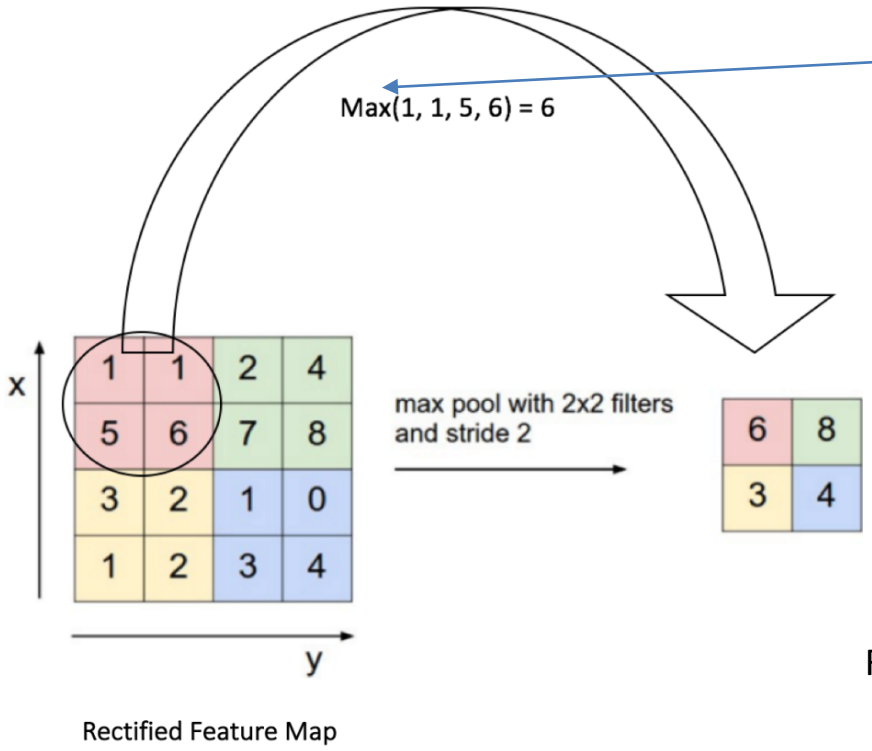
- Slide the kernel over the input matrix
- Compute element wise multiplication (Hadamard/schur product), add results to get a single value
- Output is a feature map

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Many types of filters

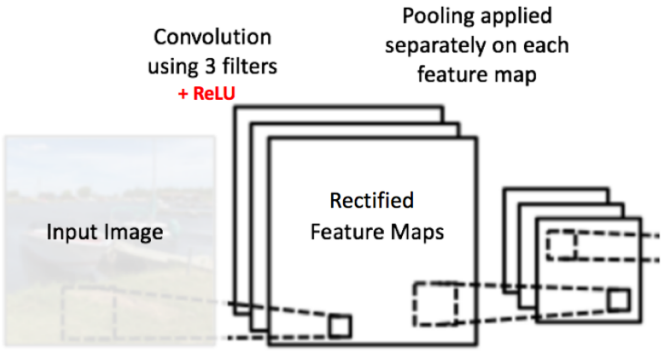
A CNN learns these filters during training

Pooling



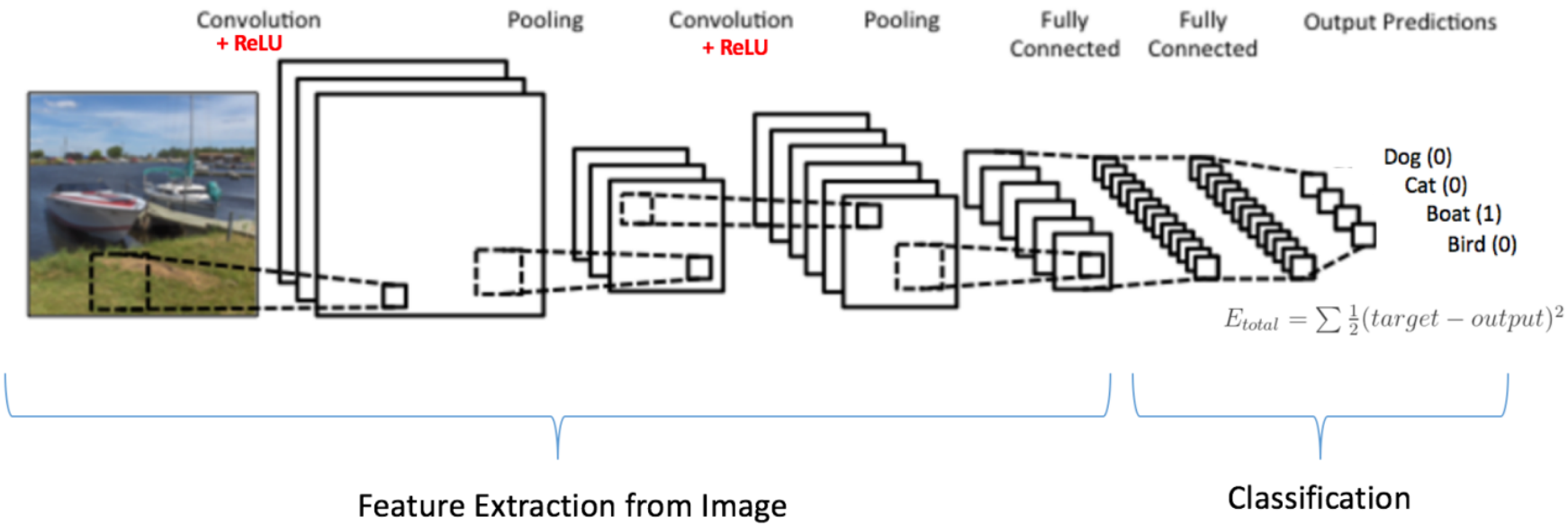
Rectified Feature Map

Can be Avg, sum, min, …



Reduce dimensionality, but retain important features

Putting Everything Together



Digital Adversarial Example

Introduction

- Szegedy et al. (2014b) : Vulnerability of machine learning models to adversarial examples
- A wide variety of models with different architectures trained on different subsets of the training data misclassify the same adversarial example – fundamental blind spots in training algorithms?
- Speculative explanations:
  - Extreme non linearity
  - Insufficient model averaging and insufficient regularization

Linear explanation of adversarial examples

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\eta}$$

$$||\boldsymbol{\eta}||_\infty < \epsilon$$

$$\boldsymbol{w}^\top \tilde{\boldsymbol{x}} = \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{w}^\top \boldsymbol{\eta}$$

$$\eta = \text{sign}(\boldsymbol{w})$$

Linear explanation of adversarial examples

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\eta}$$

$$||\boldsymbol{\eta}||_\infty < \epsilon$$

$$\boldsymbol{w}^\top \tilde{\boldsymbol{x}} = \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{w}^\top \boldsymbol{\eta}$$

Activations grow linearly!

$$\eta = \text{sign}(\boldsymbol{w})$$

Linear perturbation of non-linear models

- ReLUs, maxout networks etc. - easier to optimize linear networks
- “Fast gradient sign method”

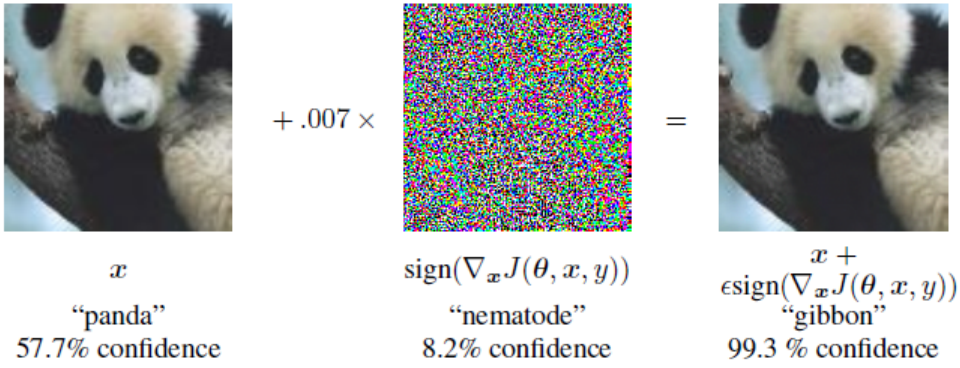
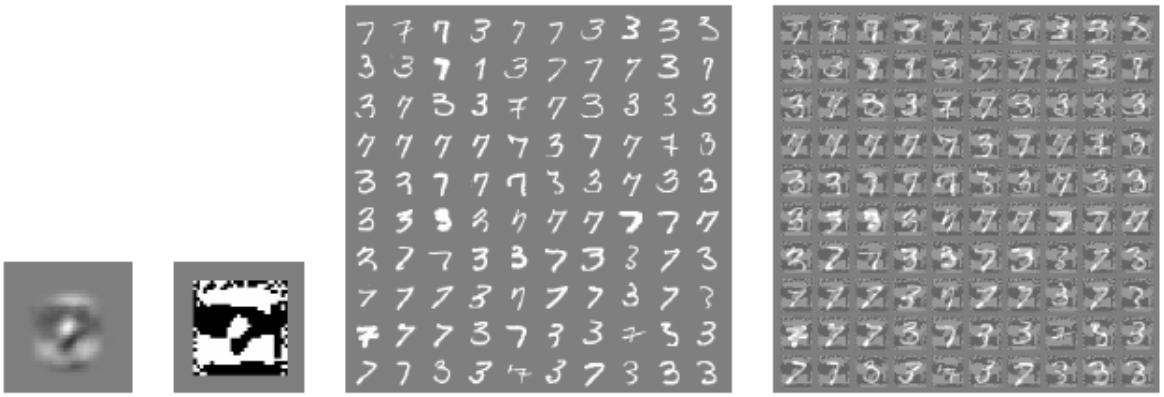


Image from reference paper

Fast gradient sign – logistic regression



1.6% error rate

99% error rate

Image from reference paper

Adversarial training of deep networks

- Deep networks are vulnerable to adversarial examples - Misguided assumption

- How to overcome this?
  - Training with an adversarial objective function based on the fast gradient sign method
  - Error rate reduced from 94% to 84%

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

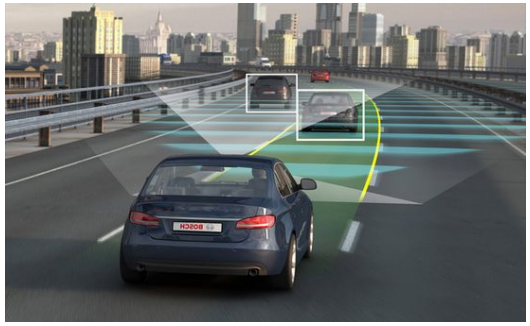
Alternate Hypothesis

- Generative training
  - MP-DBM: ϵ of 0.25, error rate of 97.5% on adversarial examples generated from the MNIST
  - Being generative alone is not sufficient
- Ensemble training
  - Ensemble of 12 maxout networks on MNIST: ϵ of 0.25, 91.1% error on adversarial examples on MNIST
  - One member of the ensemble: 87.9% error

Summary

- Adversarial examples are a result of models being too linear
- Generalization of adversarial examples across different models occurs as a result of adversarial perturbations being highly aligned with the weight vector
- The direction of perturbation rather than space matters the most
- Introduces fast methods of generating adversarial examples
- Adversarial training can result in regularization
- Models easy to optimize are easy to perturb

Autonomous Driving is the Trend…



However, What We Can See Everyday…

The Physical World Is… Messy

Varying Physical Conditions (Angle, Distance, Lighting, …)

Physical Limits on Imperceptibility

Fabrication/Perception Error (Color Reproduction, etc.)

Digital Noise (What you want) | What is printed | What a camera may see

Background Modifications*

Image Courtesy, OpenAI

[Evtimov, Eykholt, Fernandes, Kohno, Li, Prakash, Rahmati, and Song, *2017*]

Optimization Based Attack

$$\begin{aligned} \text{minimize} \quad & \mathcal{D}(x, x+\delta) \\ \text{such that} \quad & C(x+\delta) = t \\ & x+\delta \in [0,1]^n \end{aligned} \qquad \begin{aligned} \text{minimize} \quad & \mathcal{D}(x, x+\delta) + c \cdot f(x+\delta) \\ \text{such that} \quad & x+\delta \in [0,1]^n \end{aligned}$$

	Best Case				Average Case				Worst Case			
	MNIST		CIFAR		MNIST		CIFAR		MNIST		CIFAR	
	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob
Our L_0	8.5	100%	5.9	100%	16	100%	13	100%	33	100%	24	100%
JSMA-Z	20	100%	20	100%	56	100%	58	100%	180	98%	150	100%
JSMA-F	17	100%	25	100%	45	100%	110	100%	100	100%	240	100%
Our L_2	1.36	100%	0.17	100%	1.76	100%	0.33	100%	2.60	100%	0.51	100%
Deepfool	2.11	100%	0.85	100%	–	-	–	-	–	-	–	-
Our L_∞	0.13	100%	0.0092	100%	0.16	100%	0.013	100%	0.23	100%	0.019	100%
Fast Gradient Sign	0.22	100%	0.015	99%	0.26	42%	0.029	51%	–	0%	0.34	1%
Iterative Gradient Sign	0.14	100%	0.0078	100%	0.19	100%	0.014	100%	0.26	100%	0.023	100%

[Carlini, Wagner, Towards robustness of neural networks. 2017]

An Optimization Approach To Creating Robust Physical Adversarial Examples

$$\underset{\delta}{\operatorname{argmin}}\ \lambda||\delta||_p + J(f_\theta(x+\delta), y^*)$$

Perturbation/Noise Matrix

Lp norm (L-0, L-1, L-2, ...)

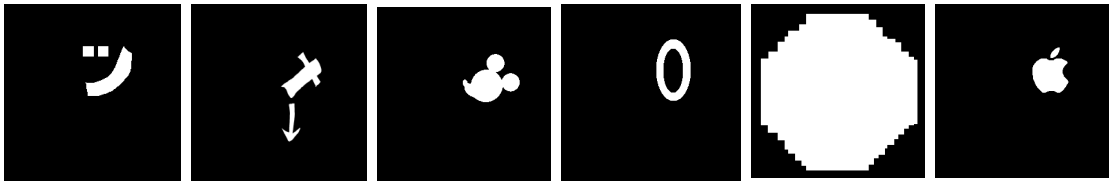
Loss Function

Adversarial Target Label

$$\underset{\delta}{\operatorname{argmin}}\ \lambda||\delta||_p + \frac{1}{k}\sum_{i=1}^{k} J(f_\theta(x_i+\delta), y^*)$$

Optimizing Spatial Constraints (Handling Limits on Imperceptibility)

$$\underset{\delta}{\operatorname{argmin}}\ \lambda||M_x \cdot \delta||_p + \frac{1}{k}\sum_{i=1}^{k} J(f_\theta(x_i + M_x \cdot \delta), y^*)$$



Subtle Poster

Camouflage Sticker

Mimic vandalism

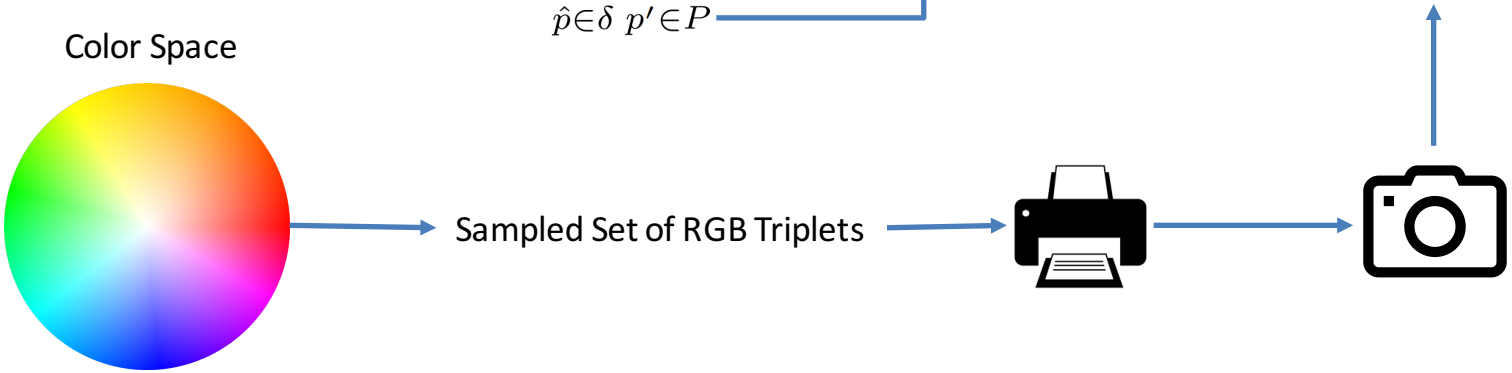
"Hide in the human psyche"

Handling Fabrication/Perception Errors

$$\underset{\delta}{\operatorname{argmin}}\ \lambda||M_x \cdot \delta||_p + \frac{1}{k}\sum_{i=1}^{k} J(f_\theta(x_i + M_x \cdot \delta), y^*) + NPS(M_x \cdot \delta)$$

$$NPS(\delta) = \sum_{\hat{p} \in \delta} \prod_{p' \in P} |\hat{p} - p'|$$

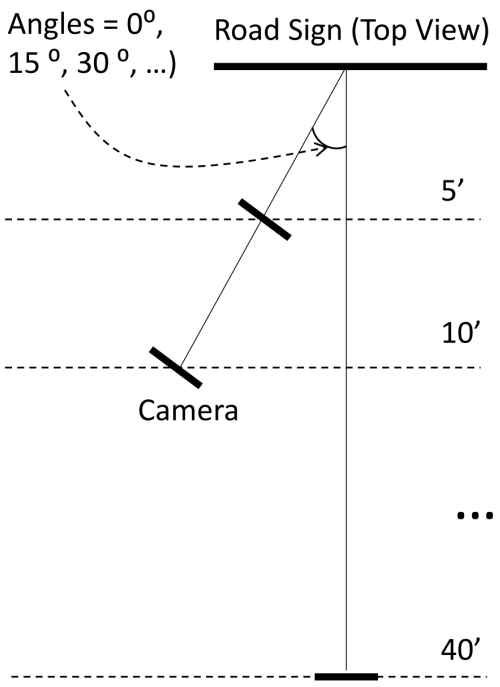
P is a set of printable RGB triplets



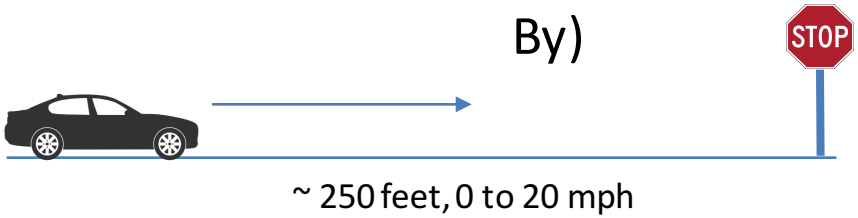
NPS based on Sharif et al., "Accessorize to a crime," CCS 2016

How Can We Realistically Evaluate Attacks?

Lab Test (Stationary)



Field Test (Drive-By)



Record video

Sample frames every k frames

Run sampled frames through DNN

Subtle Poster

Lab Test Summary (Stationary)

Target Class: Speed Limit 45

GTSRB*-CNN

Art Perturbation

Subtle Perturbation

Adversarial Examples in Physical World

Adversarial perturbations are possible in physical world under different conditions and viewpoints, including the distances and angles.