# Neural Network Robustness

# About me

- My name is Radoslav Ivanov
  - Call me Rado
- Undergrad degree in CS and ECON from Colgate in 2011
- Got my PhD in CS from UPenn in 2017
- My research is on safe and secure autonomous systems
  - Verification of neural networks
  - Attack-resilient sensor fusion
  - Context-aware detection and estimation
- Started at RPI in Jan. 2022

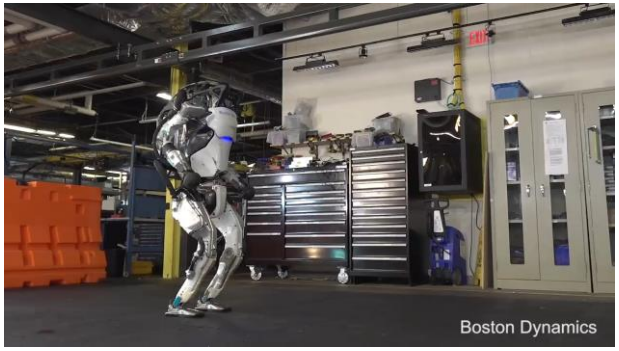
# Reading

- Papers (all available online)
  - Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. "Intriguing properties of neural networks." *arXiv preprint arXiv:1312.6199* (2013).
  - Madry, Aleksander, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards deep learning models resistant to adversarial attacks." *arXiv preprint arXiv:1706.06083* (2017).

# Impressive Progress in Autonomy

Rensselaer

## Control




**Boston Dynamics**

## Perception

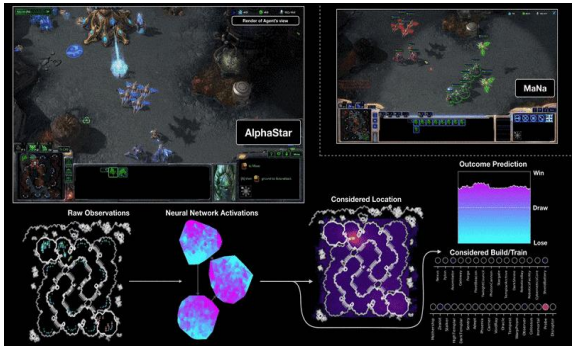

**YOLO v. 3**

## Learning


**Kormushev, Calinon, Caldwell, IROS'10**

**JPL-Caltech, DARPA Robotics Challenge**


**Zhu, Zhou, Daniilidis, ICCV'15**




**DeepMind**

# But we're not there yet…

Rensselaer

## Tesla Driver Was on Autopilot Eating a Bagel When He Smashed into a Fire Truck

A National Transportation Safety Board investigation found the driver had hands off the wheel and ignored warnings in the 2018 crash.

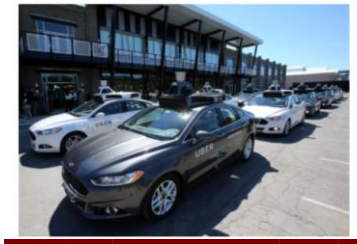
By Clifford Atiyeh SEP 4, 2019

- The driver of a 2014 Tesla Model S that ran into the back of a fire engine in California in 2018 was using Autopilot at the time, according to a National Transportation Safety Board (NTSB) report this week.
- The agency's investigators reported that the driver was having breakfast while he let Autopilot take over the driving; his hands were not on the steering wheel, and he did not brake prior to the crash.

## Uber self-driving car involved in fatal crash couldn't detect jaywalkers

The system had several serious software flaws, the NTSB said.

Steve Dent, @stevetcdent 11.06.19 in Transportation 32 Comments 2136 Shares



Sponsored Links

QuickBooks® - Get Pair QuickBooks® Online

Biotech Sector Action A Today

ум: Australian open 2020 - Т | Дискусия на живо: Аустралиън | Live Scores | Australian Open

320936/waymo-self-driving-car-crash-arizona

GOOGLE | TECH | TRANSPORTATION

## Waymo self-driving minivan involved in crash in Arizona

*Minor injuries reported*

By Andrew J. Hawkins | @andyjayhawk | May 4, 2018, 4:52pm EDT

SHARE

## Boeing 737 Max Lion Air crash caused by series of failures

25 October 2019 Share

Lion Air plane crash

The Washington Post | PostTV | Politics | Opinions | Local | Sports | Natio

## National Security

Home > Collections > Surveillance

### Iran says it downed U.S. stealth drone; I acknowledges aircraft downing

By Greg Jaffe and Thomas Erdbrink, December 04, 2011

A secret U.S. surveillance drone that went missing last week in western Afgha have crashed in Iran, in what may be the first case of such an aircraft ending u an adversary.

Iran's news agencies asserted that the nation's defense forces brought down t Iranian reports said was an RQ-170 stealth aircraft. It is designed to penetrate defenses that could see and possibly shoot down less-sophisticated Predator a

A stealthy RQ-170 drone played a critical role in surveilling the compound in Osama bin Laden was hiding in the months before the raid in which he was ki SEALs in May.

U.S. officials acknowledged Sunday that a drone had been lost near the Irania declined to say what kind of aircraft was missing.

The Iranian government has not released any pictures of the recovered aircraf was downed by defense forces after it flew across the border and into the cour unnamed Iranian defense official said in one report that a cyberattack caused

U.S. officials cast doubt on th Iranian assertions. "We have no indication that it was brought down by hostil senior Pentagon official, speaking on the condition of anonymity to discuss se activity.

BBC | News | Sport | Weather | Capital | Culture | Autos

NEWS TECHNOLOGY

Home | US & Canada | Latin America | UK | Africa | Asia | Europe | Mid-East | Business | Health | Sci/Environme

25 July 2013 Last updated at 19:04 ET

Share

## Car hackers use laptop to control standard car

By Zoe Kleinman

Technology reporter, BBC News

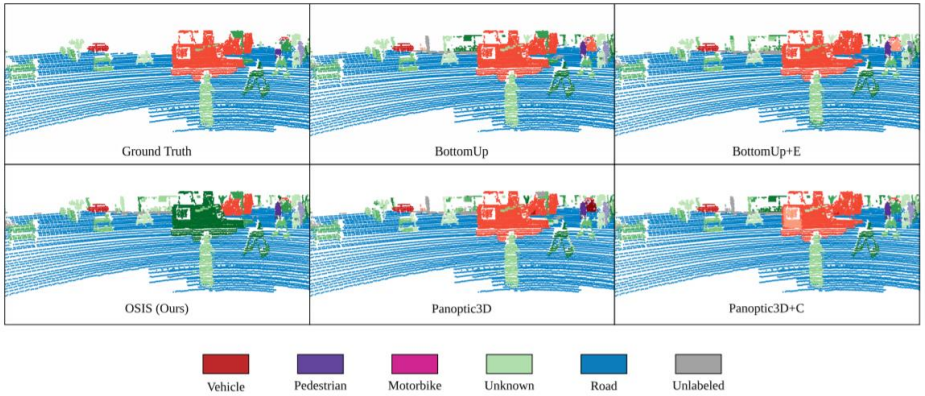

Brian Donohue Follow @TheBrianDonohue July 30, 2013 , 3:26 pm

A 213-foot luxury yacht veered off course while cruising in the Mediterranean Sea this

# Neural Network (NN) Vulnerabilities

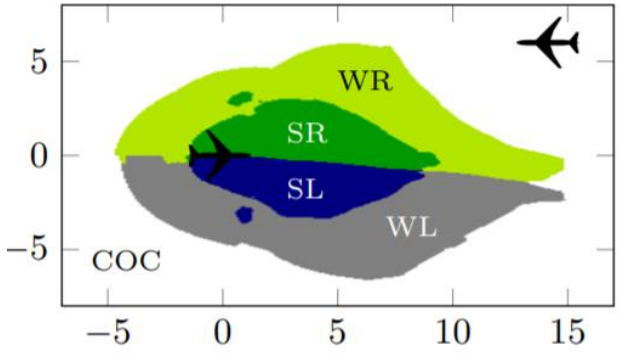
- Neural networks increasingly used in safety-critical systems

**Perception (autonomous cars)**



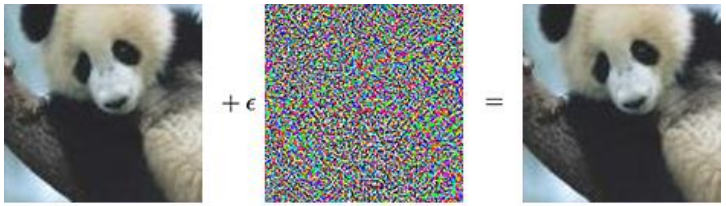

Wong et al., CoRL'19

**Control (air traffic avoidance)**




Katz et al., CAV '17

- Safety concerns discovered in both domains




Goofellow et al., ICRL'15

Table 2: Verifying properties of the ACAS Xu networks.

| | Networks | Result | Time | Stack | Splits |
|---|---|---|---|---|---|
| $\phi_1$ | 41 | UNSAT | 394517 | 47 | 1522384 |
| | 4 | TIMEOUT | | | |
| $\phi_2$ | 1 | UNSAT | 463 | 55 | 88388 |
| | 35 | SAT | 82419 | 44 | 284515 |
| $\phi_3$ | 42 | UNSAT | 28156 | 22 | 52080 |
| $\phi_4$ | 42 | UNSAT | 12475 | 21 | 23940 |

# A standard CPS design

F1/10 Autonomous Racing Competition, ES Week 2016

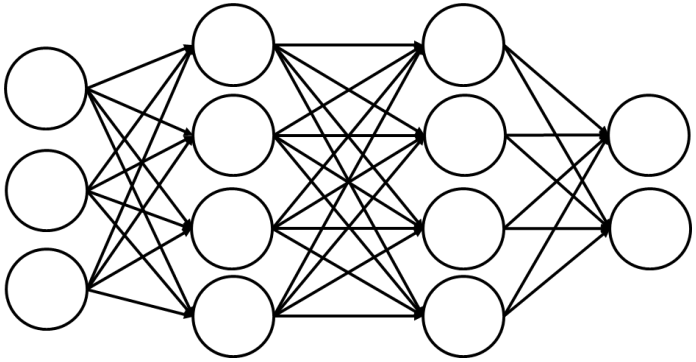
**Problem**: How do we know car won't crash?

- How do we build safe algorithms?
- How do we analyze algorithms?
- What about "black-box" components such as neural networks?
- How do we convince other people car is safe (assurance argument)?

# Feedforward Neural Networks

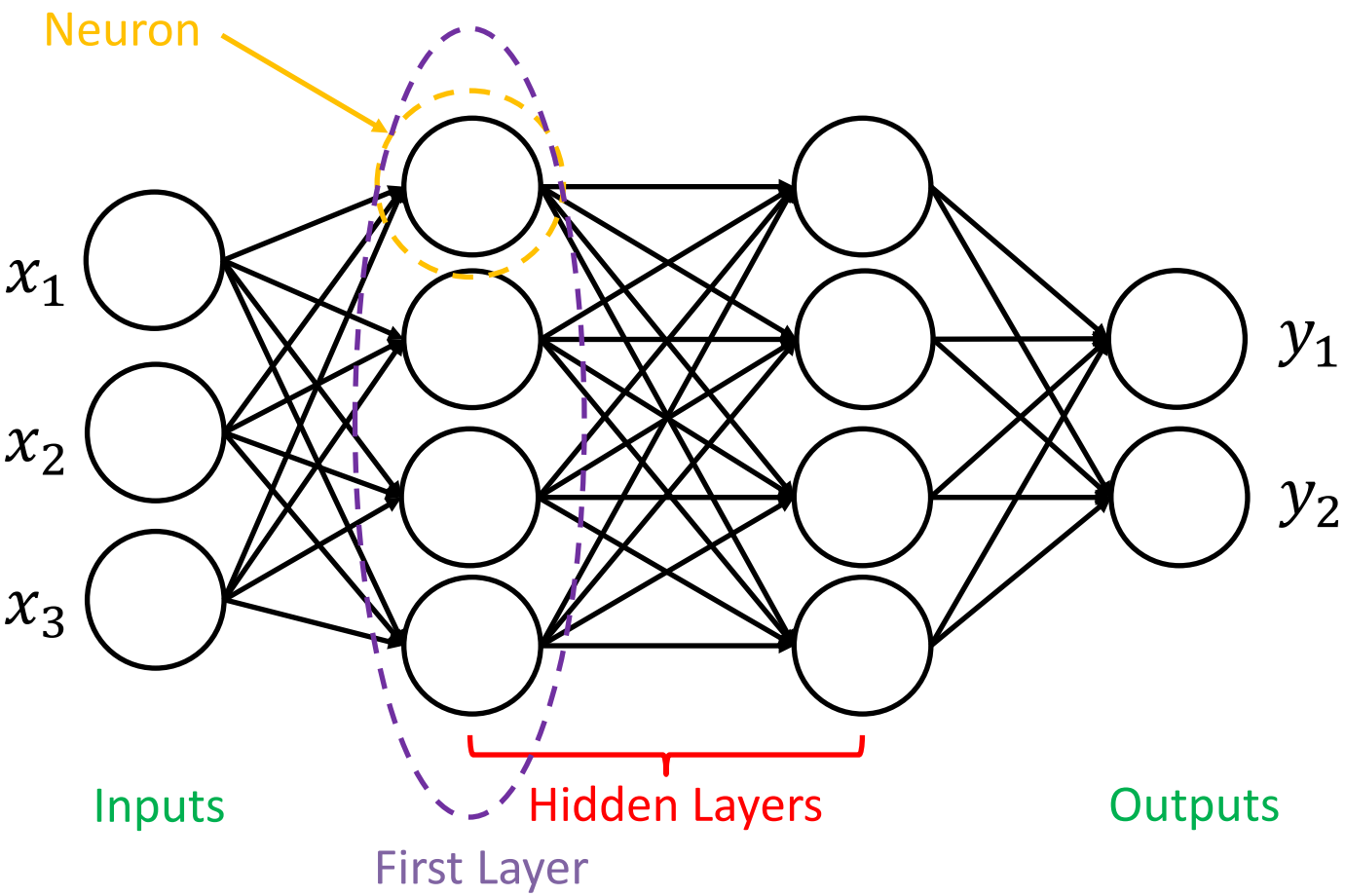
Rensselaer

- Also known as multi-layer perceptrons
  - Old name, at least from the 1960's

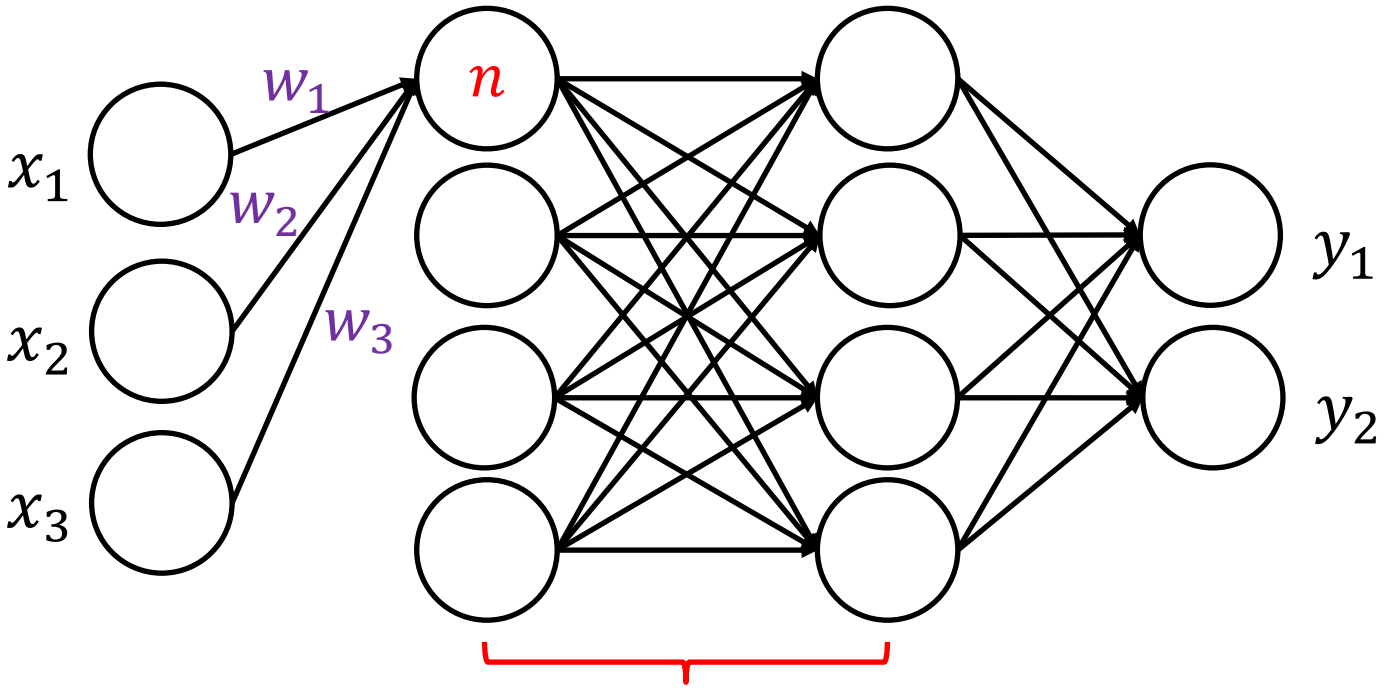


- The term "deep neural networks" is essentially rebranding
  - Modern networks are deeper than ever, however
  - Term "neural" is (very) loosely inspired by neuroscience
- The term "feedforward" means that computation happens from left to right in network, without any feedback

# NN terminology



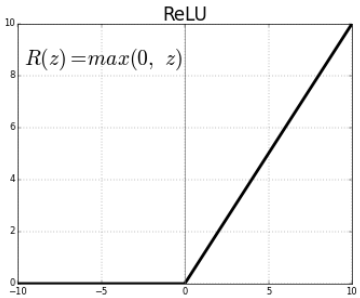
# NN Functionality

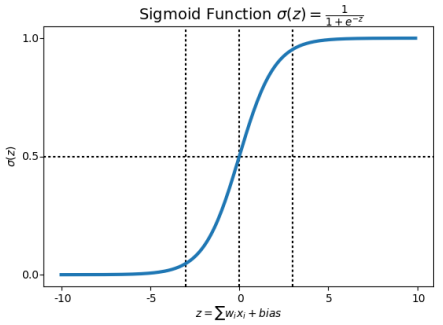



$$n = a(w_1 y_1 + w_2 y_2 + w_3 y_3)$$

Common activations include:

- Relu: $a(x) = \max(0, x)$
- sigmoid: $a(x) = \frac{1}{1+e^{-x}}$



## NNs as functions

- Standard ML model

$$y = f(\boldsymbol{x}; \boldsymbol{\theta})$$

  - $\boldsymbol{x}$ are the inputs (e.g., pixels), $y$ are the outputs (e.g., labels), $\boldsymbol{\theta}$ are the parameters to be optimized

- Can be written as a composition of its $L$ hidden layers

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = f_L \circ f_{L-1} \circ \cdots \circ f_1(\boldsymbol{x})$$

- Typically trained by minimizing a loss function on a labeled training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$
  - E.g., least squares

$$\min_{\boldsymbol{\theta}} \sum_{i=0}^{N} (y_i - f(\boldsymbol{x}_i; \boldsymbol{\theta}))^2$$

# Intriguing properties of neural networks

## Overview

Rensselaer

- This paper started a frantic search for improving neural network robustness
  - Even the authors were probably surprised by the paper's huge impact
- Essentially, they found that even though NNs perform well on data from the "distribution", they are not robust to even tiny perturbations
  - This means that NNs are highly unpredictable and cannot be used in safety-critical systems (e.g., autonomous cars) without understanding them better

# Setup

Rensselaer

- Pretrained NNs on several benchmarks using different algorithms
  - MNIST
  - ImageNet
- Test how sensitive these NNs are to small input perturbations

# Input-Output Sensitivity

Rensselaer

- NN input-output sensitivity is obviously important
  - E.g., don't want the output to change if you move the camera by 0.1 inch
- Some spuriousness introduced by the fact that NNs typically trained with cross-entropy
  - Tries to approximate the conditional distribution of the labels given an example
  - Bound to assign non-zero probability to spurious classes
  - Spuriousness did not exist in classical computer vision
    - Heavy use of filtering and smoothening improves robustness
    - We implicitly assumed that such spuriousness is low for NNs also

# Adversarial Examples

- Turns out that NNs are not at all robust to small perturbations
  - For **any** NN and **any** correctly classified image, one can find an imperceptibly small perturbation that causes the NN to misclassify the image
  - Especially true for more complex datasets
  - Perturbed image is called an **adversarial example**
  - Some techniques have been developed to alleviate this issue but it is still very much an open problem

## Formal Setup

- Let $f: \mathbb{R}^m \rightarrow \{1, \dots, k\}$ be a pre-trained classifier
- We are given an image $\boldsymbol{x}$ and a target label $l$
  - Note that $l$ may or may not be the ground-truth label
  - Turns out this works for any $l$!
- Goal: find the minimum perturbation $\boldsymbol{r}$ such that $f(\boldsymbol{x} + \boldsymbol{r}) = l$
- Formally,

$$\begin{aligned} \min_{\boldsymbol{r}} \quad & ||\boldsymbol{r}||_2 \\ subject\ to \quad & f(\boldsymbol{x} + \boldsymbol{r}) = l \end{aligned}$$

- Of course, $\boldsymbol{x} + \boldsymbol{r}$ has to be an image, i.e., $\boldsymbol{x} + \boldsymbol{r} \in [0,255]^m$
  - Or normalized to $[0,1]^m$

## Solving the minimization problem

Rensselaer

- Of course, solving the minimization problem is hard because NNs
- Instead use gradient descent, but with $\boldsymbol{r}$ being the optimization variable
  - minimize original loss: $loss_f(\boldsymbol{x} + \boldsymbol{r}, l)$
- To find a *small* $\boldsymbol{r}$, also add an extra term $c|\boldsymbol{r}|$, with $c$ fixed
  - Find minimum $c$ for which you can find a good $\boldsymbol{r}$
- Final loss is

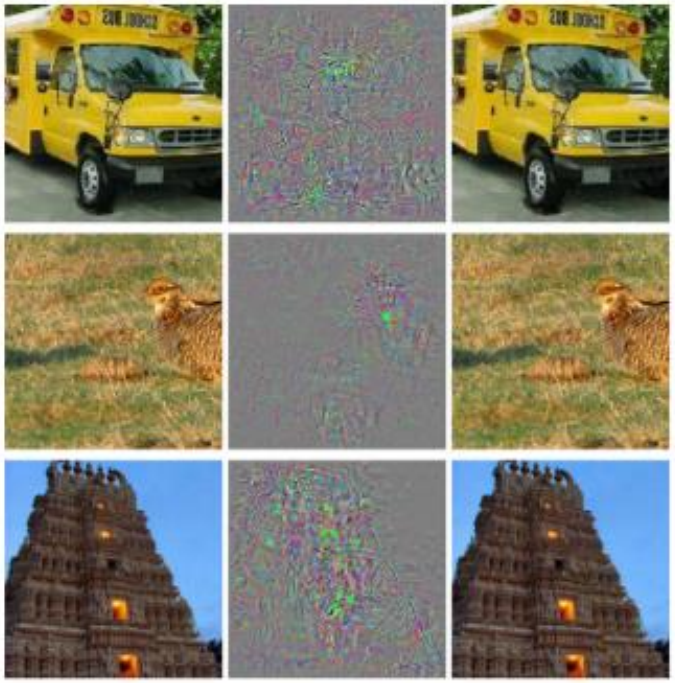
$$c|\boldsymbol{r}| + loss_f(\boldsymbol{x} + \boldsymbol{r}, l)$$

- How do you choose $c$?
  - Linear search is a good start

# Results Overview

- For almost every NN and every example, an almost imperceptible adversarial example was found
  - The *almost* part removed in later papers that came up with better ways of solving the optimization problem
- Adversarial examples are transferable
  - An adversarial example for one NN is also misclassified by another
  - Also true even if second NN trained on a disjoint training set
  - Suggests that adversarial examples are a fundamental artifact of the distribution/training method

# Examples, ImageNet



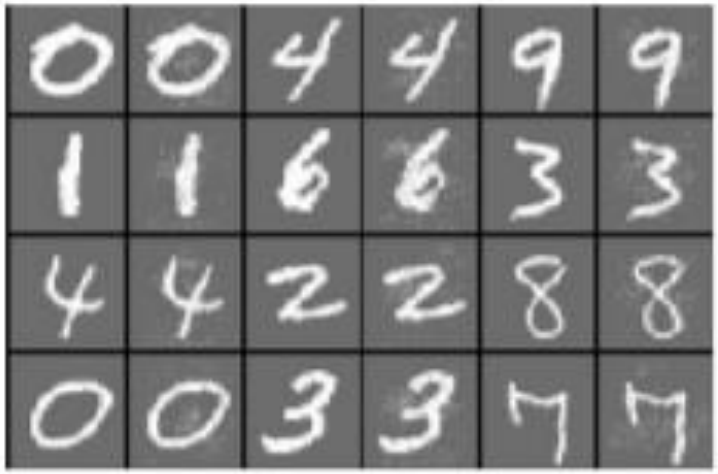
(a)

(b)

All images go from correctly classified to “ostrich”

# Examples, MNIST

(a) Even columns: adversarial examples for a linear (FC) classifier (stddev=0.06)



(b) Even columns: adversarial examples for a 200-200-10 sigmoid network (stddev=0.063)

# Adversarial Example Transferability

| | 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 |

# Towards deep learning models resistant to adversarial attacks

## Overview

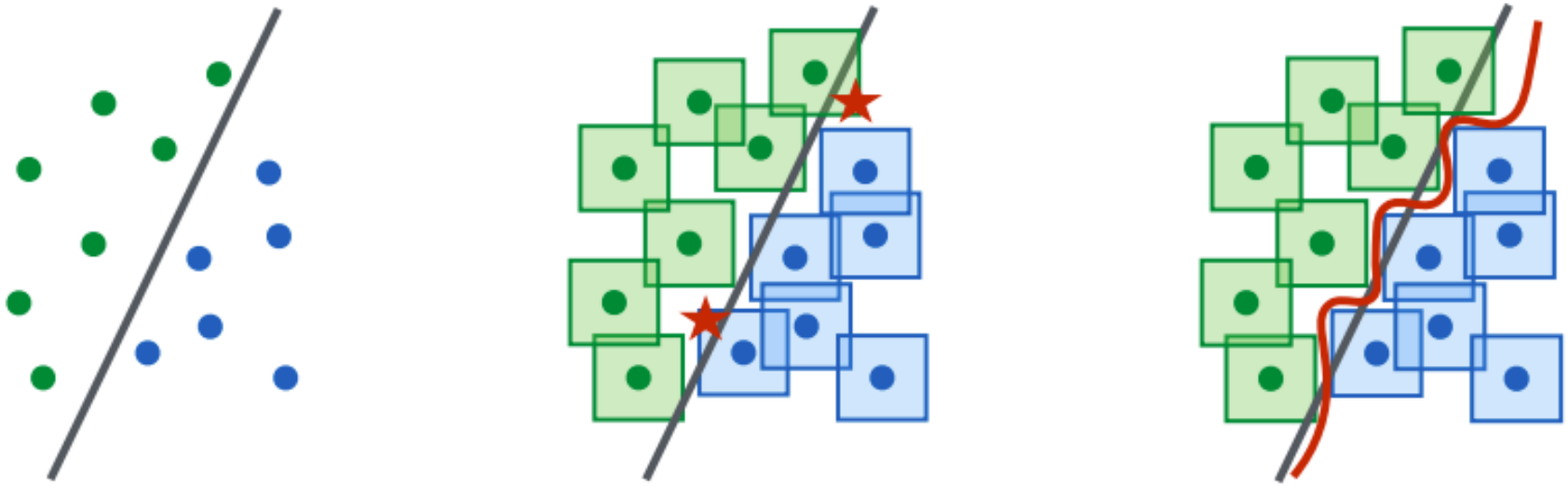
Rensselaer

- After all the robustness issues were found, researchers got busy trying to find solutions
  - Can we train adversarially robust NNs?
- Adversarial training was one of the first major methods and remains the building block for most methods today
  - However, the benefits are only significant in the case of MNIST
  - Problem is still very much open in general

# Adversarial Robustness High-Level Goal

Rensselaer

- In standard training (left), we find any decision boundary that maximally separates the training points
  - Boundary typically does not enforce robustness (middle)
- The goal is train in a way that produces a decision boundary that is robust around each training point (right)



# Adversarial Training Setup

Rensselaer

- Reminder: in standard training, we try to minimize the expected loss over the training set

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}} \left[ L\big(\boldsymbol{\theta}, (\boldsymbol{x}, y)\big) \right]$$

- In adversarial training, the goal is to minimize the loss not only over the training set, but also over a box around each point from the training set
- In particular, we want to minimize the worst-case loss over any perturbation within that box

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}} \left[ \max_{\boldsymbol{r}\in\mathcal{R}} L\big(\boldsymbol{\theta}, (\boldsymbol{x}+\boldsymbol{r}, y)\big) \right]$$

- The set $\mathcal{R}$ can in theory be any set
  - Usually we take it to be an $L_\infty$ ball

# Inner Optimization

- The inner optimization problem is essentially what the adversarial attacks do

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}} \left[ \max_{\boldsymbol{r}\in\mathcal{R}} L\big(\boldsymbol{\theta}, (\boldsymbol{x}+\boldsymbol{r}, y)\big) \right]$$

- It makes sense to train NNs that are robust to these attacks by minimizing the quantity the attacks try to maximize
- Note that this problem is quite narrow in a sense
  - Unclear if we want *robust* NNs or NNs that work well on all examples in our distribution
  - However, the distinction between a *distribution* and a perturbation-induced-distribution is blurry

# First, investigate adversarial examples

Rensselaer

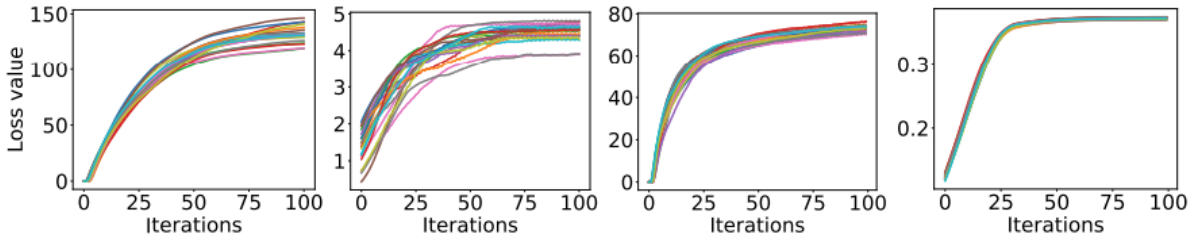
- Use projected gradient descent (PGD) to find adversarial examples and record their loss

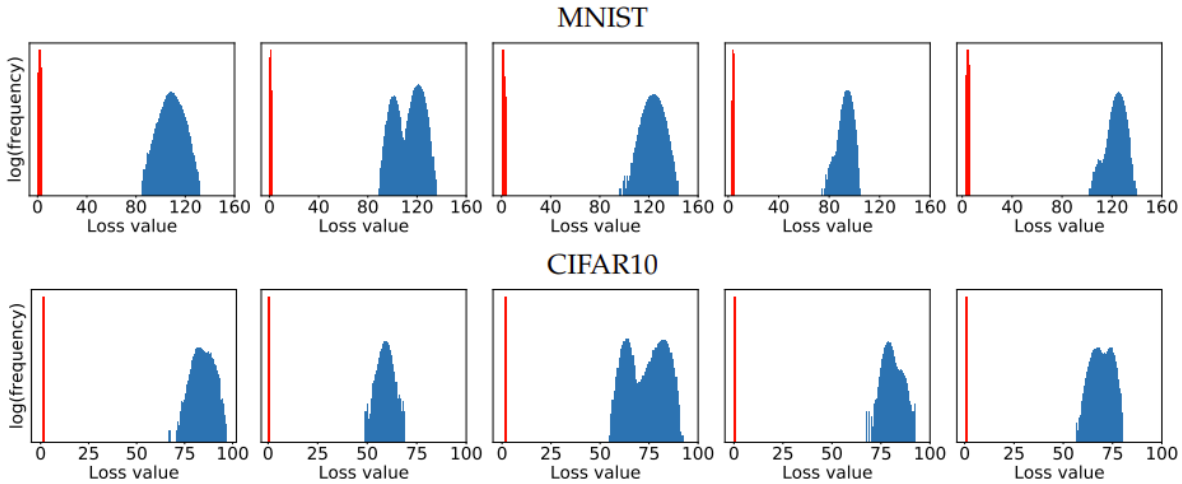
$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^{t} + \alpha \nabla_{\boldsymbol{x}} L\big(\boldsymbol{\theta}, (\boldsymbol{x}, y)\big)$$
$$\boldsymbol{x}^{t+1} = clip_{x+\epsilon}\left[\boldsymbol{x}^{t+1}\right]$$

  - Can also use sign of gradient
  - Initialize $\boldsymbol{x}^{0}$ randomly around a training example $\boldsymbol{x}$

# Zoo of Adversarial Examples

- It seems that most adversarial examples lead to the same loss
  - Authors claims this means inner optimization problem is tractable, which is a questionable claim



# Adversarial Training Algorithm

Rensselaer

- Use gradient descent as before (combined with PGD)

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}} \left[ \max_{\boldsymbol{r}\in\mathcal{R}} L\big(\boldsymbol{\theta}, (\boldsymbol{x}+\boldsymbol{r}, y)\big) \right]$$

- For a given example $\boldsymbol{x}$
  1. Perform $k$ iterations of PGD
     a) Here we are optimizing $\boldsymbol{r}$, keeping $\boldsymbol{\theta}$ constant
  2. Perform gradient descent using $\boldsymbol{x}+\boldsymbol{r}$
- Of course, this is done for a mini-batch as before
- Hyperparameters are $k$, $\epsilon$ (robustness ball) and $\alpha$ (PGD step size)

# Experimental Setup

- MNIST
  - $k = 40$
  - $\alpha = 0.01$ (used with the gradient sign)
  - $\epsilon = 0.3$ (out of 1)
  - 3-layer CNN (2 convolutional + 1 fully-connected)
- CIFAR10
  - $k = 20$
  - $\alpha = 2$ (out of 255)
  - $\epsilon = 8$ (out of 255)
  - ResNet architecture (quite large…)

# Results, MNIST

- Very high adversarial accuracy even for $\epsilon = 0.3$
  - Natural accuracy is 98.8%, down from 99.2% with no adversarial training, so a very minor difference
  - FGSM is PGD with 1 step

| 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% |

# Results, CIFAR10

- Adversarial robustness goes up from ~3% to ~45%
  - However, natural accuracy drops from 95% to 87%
  - Note that wide model (i.e., more neurons) performs a bit better
    - More on this next

CIFAR10

| | Simple | Wide |
|---|---|---|
| Natural | 92.7% | 95.2% |
| FGSM | 27.5% | 32.7% |
| PGD | 0.8% | 3.5% |

(a) Standard training

| | Simple | Wide |
|---|---|---|
| Natural | 87.4% | 90.3% |
| FGSM | 90.9% | 95.1% |
| PGD | 0.0% | 0.0% |

(b) FGSM training

| | Simple | Wide |
|---|---|---|
| Natural | 79.4% | 87.3% |
| FGSM | 51.7% | 56.1% |
| PGD | 43.7% | 45.8% |

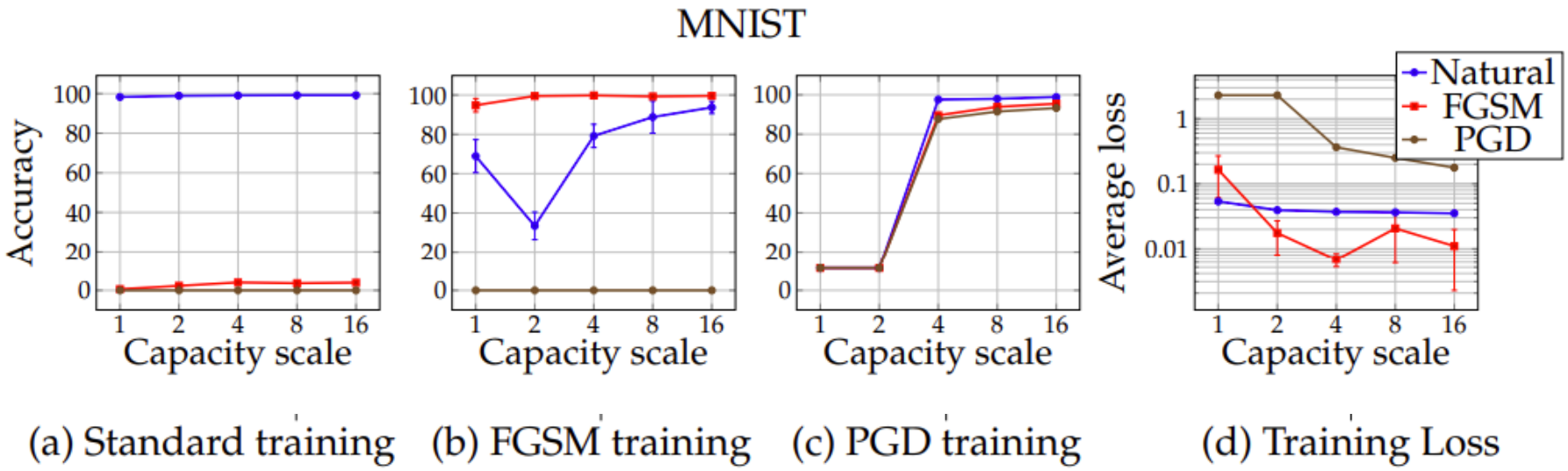
(c) PGD training

| | Simple | Wide |
|---|---|---|
| Natural | 0.00357 | 0.00371 |
| FGSM | 0.0115 | 0.00557 |
| PGD | 1.11 | 0.0218 |

(d) Training Loss

# Capacity vs Robustness

- Higher-capacity models appear to be more robust also
  - Unclear if this is because they are easier to train or because smaller models cannot be made robust
  - Higher-capacity = double # of filters and FC neurons



# Conclusion

- Adversarial training helps but it does not solve the problem
  - CIFAR10 is not even the hardest dataset
- Not only is robustness not very high but there is also a hit in standard accuracy
  - Cannot talk about robustness without accuracy: a coin flip is the most robust classifier (why?)
- Many, many papers since the Madry paper
  - There's a leaderboard here: https://robustbench.github.io/

## So what?

- Maybe we don't care about robustness since those examples are carefully crafted and will never appear in real life
- Sure, but…

1. There's something unsettling about the fact that the NN doesn't work on images that look exactly the same as normal ones (to humans)
2. Neural networks are not robust to natural perturbations (day vs. night, etc.), which suggests the problem is bigger than contrived adversarial examples

# Robust physical-world attacks on deep learning visual classification

Rensselaer

- Modify real-world objects in small ways that wouldn't fool a human but fool classifiers used in autonomous cars

All misclassified as Speed Limit 45 signs

| Distance/Angle | Subtle Poster | Subtle Poster Right Turn | Camouflage Graffiti | Camouflage Art (LISA-CNN) | Camouflage Art (GTSRB-CNN) |
|---|---|---|---|---|---|
| 5′ 0° |  |  |  |  |  |
| 5′ 15° |  |  |  |  |  |
| 10′ 0° |  |  |  |  |  |
| 10′ 30° |  |  |  |  |  |
| 40′ 0° |  |  |  |  |  |
| Targeted-Attack Success | 100% | 73.33% | 66.67% | 100% | 80% |