Trustworthy AI
Spring 2024

Yuan Tian

#2: Adversarial Machine Learning and Defenses

Pointing the need for broader view on AI

Human-Level Intelligence or Animal-Like Abilities?
Communications of the ACM, Oct 2018
Adnan Darwiche

https://cacm.acm.org/magazines/2018/10/231373-human-level-intelligence-or-animal-like-abilities/fulltext

"…We need a new generation of AI researchers who are well versed in and appreciate **classical AI**, **machine learning**, and computer science more broadly while also being informed about AI history…"

Today: Adversarial Examples

What are these? More examples in various domains

Why do they exist?

How to generate adversarial examples?

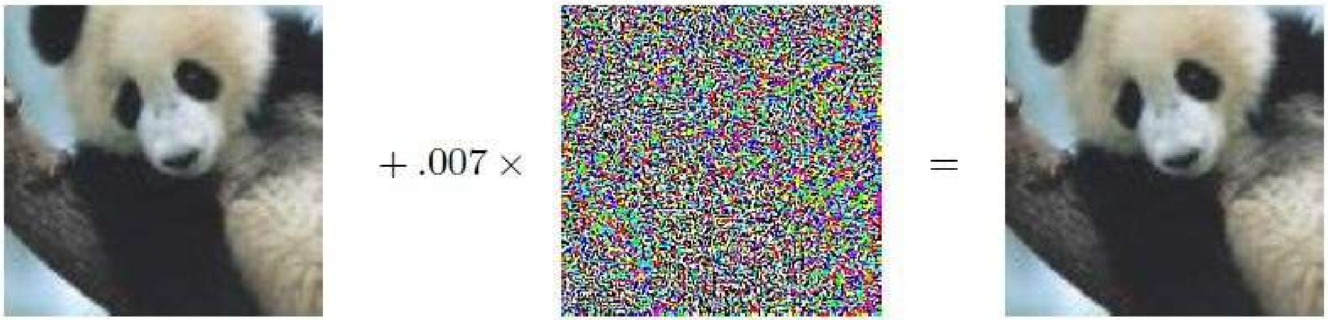
(techniques also used for adversarial training, logic, etc...discussed in later lectures)

“Adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake”

(Goodfellow et al 2017)

Adversarial Examples

Noisy attack: vision system thinks we now have a gibbon…



x — "panda" 57.7% confidence

$+ .007 \times$ $\text{sign}(\nabla_{x} J(\theta, x, y))$ — "nematode" 8.2% confidence

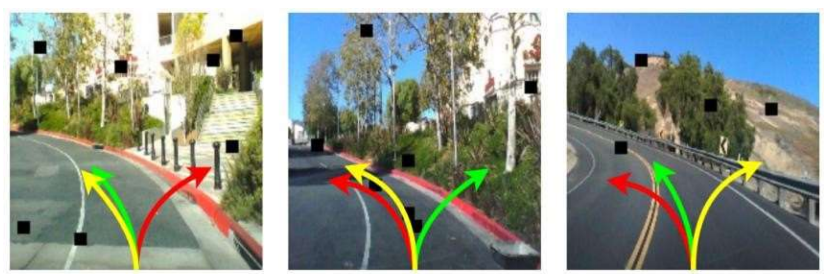
$=$ $x + \epsilon\text{sign}(\nabla_{x} J(\theta, x, y))$ — "gibbon" 99.3 % confidence

Explaining and Harnessing Adversarial Examples, ICLR '15

Tape pieces make network predict a 45mph sign

Robust Physical-World Attacks on Deep Learning Visual Classification, CVPR'18

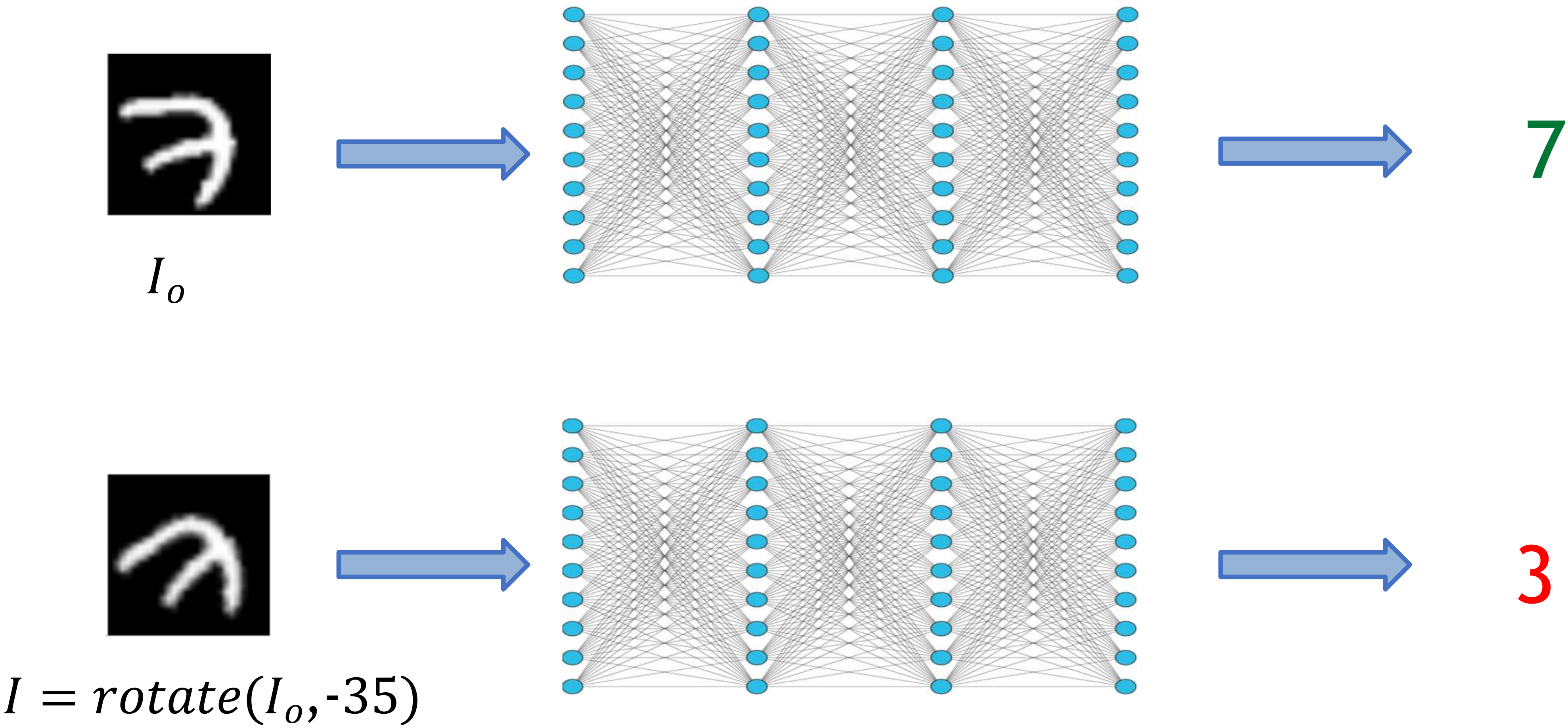
Self-driving car: in each picture one of the 3 networks makes a mistake…



DRV_C1: right DRV_C2: right DRV_C3: right

DeepXplore: Automated Whitebox Testing of Deep Learning Systems, SOSP'17

Adversarial Geometric Perturbations



Adversarial Examples (more)

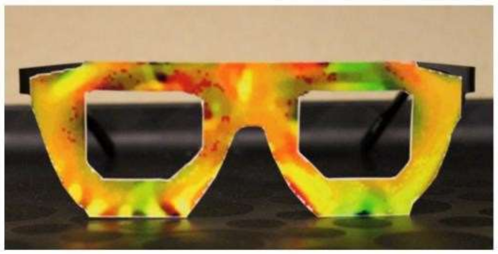
Reese Witherspoon

Russel Crowe

Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition, CCS '16

Real World Impersonation/Dodging Attacks

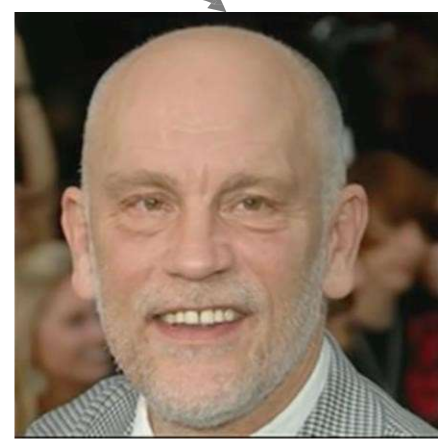
Real glasses



100% success

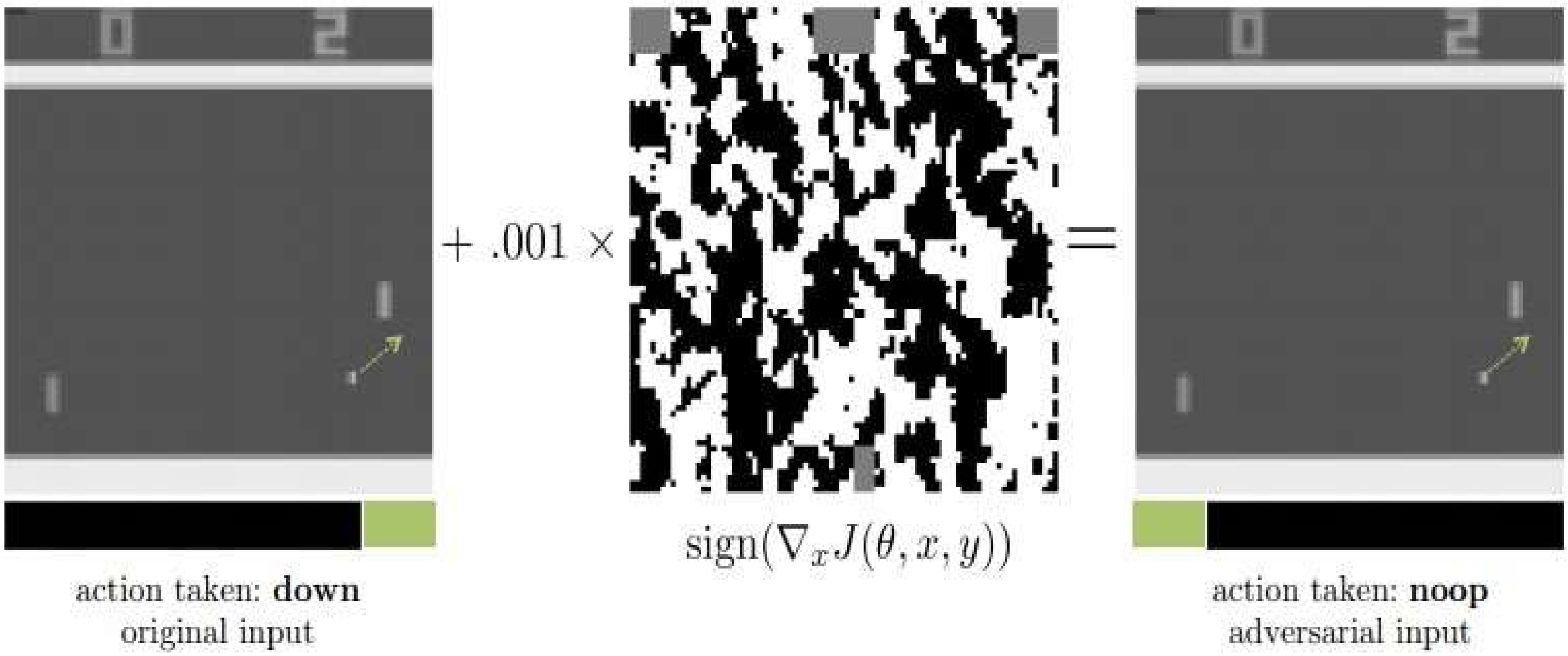
Lujo Bauer

=



John Malkovich

Adversarial Examples in Reinforcement Learning



An agent (Deep Q Network) plays the game by selecting actions from a given state (image) that the game produces.

An attacker can perturb the image slightly so that the DQN agent chooses the wrong action: here, it wrongly picks noop (do nothing) in the right image, instead of moving the paddle down (left image).

Adversarial Attacks on Neural Network Policies, Huang et.al, 2017

Adversarial Examples in NLP

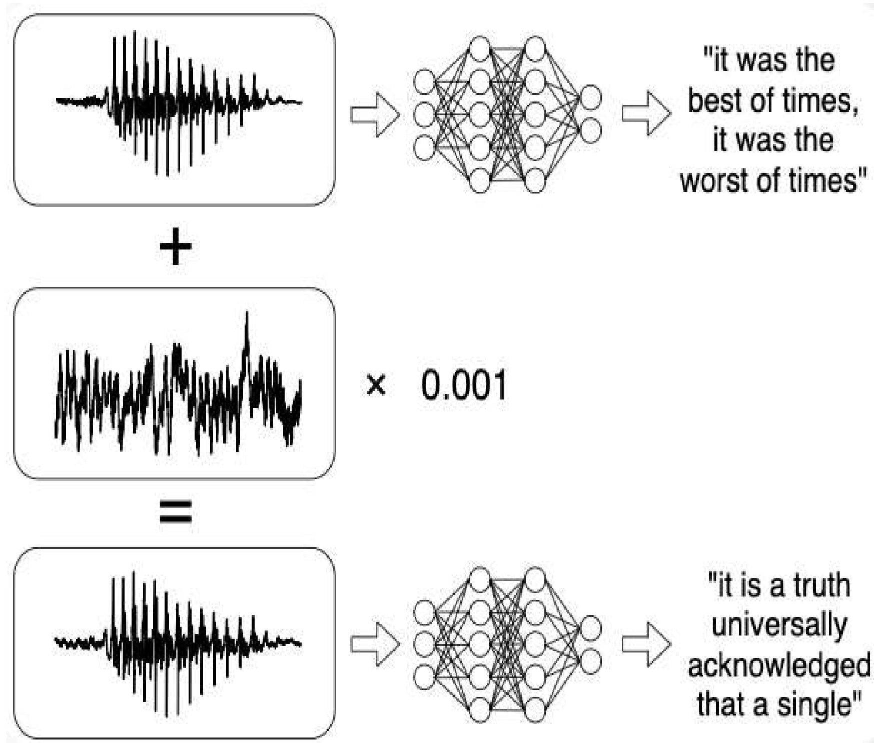
Article: Super Bowl 50
Paragraph: "*Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver's Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.*"
Question: "*What is the name of the quarterback who was 38 in Super Bowl XXXIII?*"
Original Prediction: John Elway
Prediction under adversary: Jeff Dean

The Ensemble model is fooled by the addition of an adversarial distracting sentence in blue.

Adversarial Examples for Evaluating Reading Comprehension Systems, EMNLP'17

Adversarial Examples in Audio Processing: Speech to Text

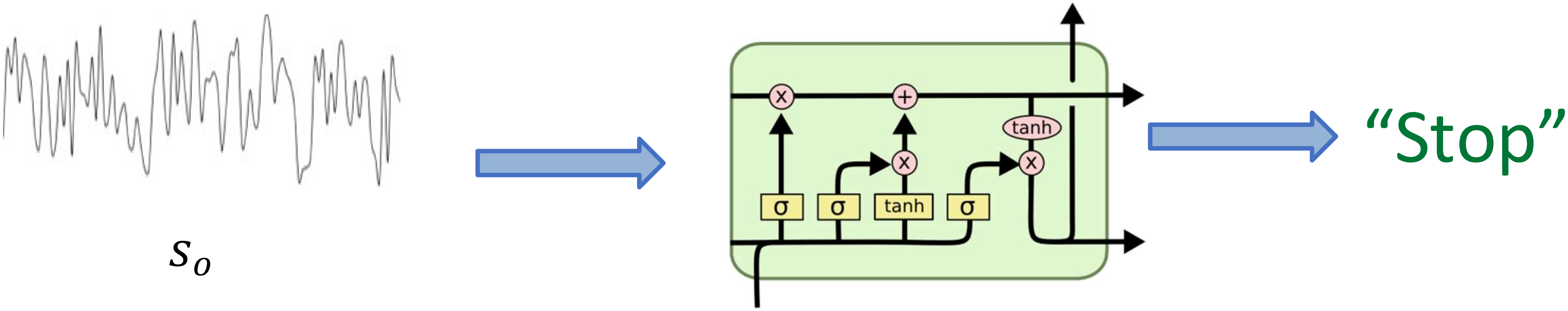
An attack on DeepSpeech:

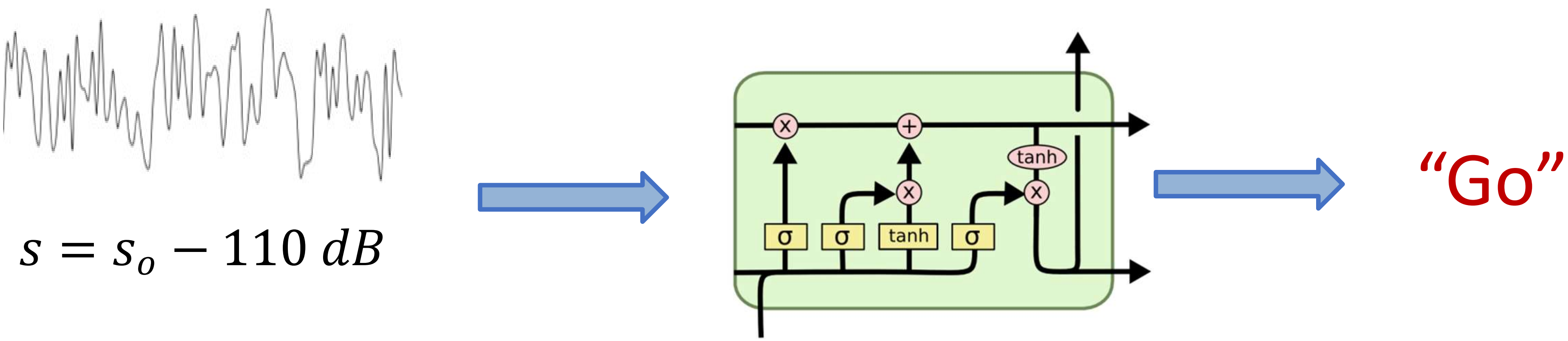


Adding small noise to the input audio makes the network transcribe any arbitrary phrase

Audio Adversarial Examples: Targeted Attacks on Speech-to-Text, ICML 2018 workshop

Adversarial Examples in Audio Processing: Text classification





Adding small noise to the input audio changes the classification of the output

Adversarial Examples: Some History

2006: Deep learning models gain renewed interest

2012: Multiple works showed that deep networks can achieve near-human performance (sometimes even better)

2013: Research in understanding neural networks behavior becomes critical with society implications beyond computer science

2014: While trying to understand decision making in neural networks, Szegedy et al. discovered adversarial examples

2015-on: Finding adversarial examples and proving their absence becomes an active research area…

2023-on : Jail break LLMs, the alignment problem…

Robustness

Robustness: A network is **robust** if it returns correct output on all inputs

Impractical: the input space is too large to be covered

Local Robustness (informal): A learning model is **locally-robust** if it returns the correct output on inputs ***similar*** to inputs in the training set

This was believed to be evident by having high accuracy on the test set

Why is High Accuracy Not Enough?

Inputs in the training and test set are taken from a given distribution

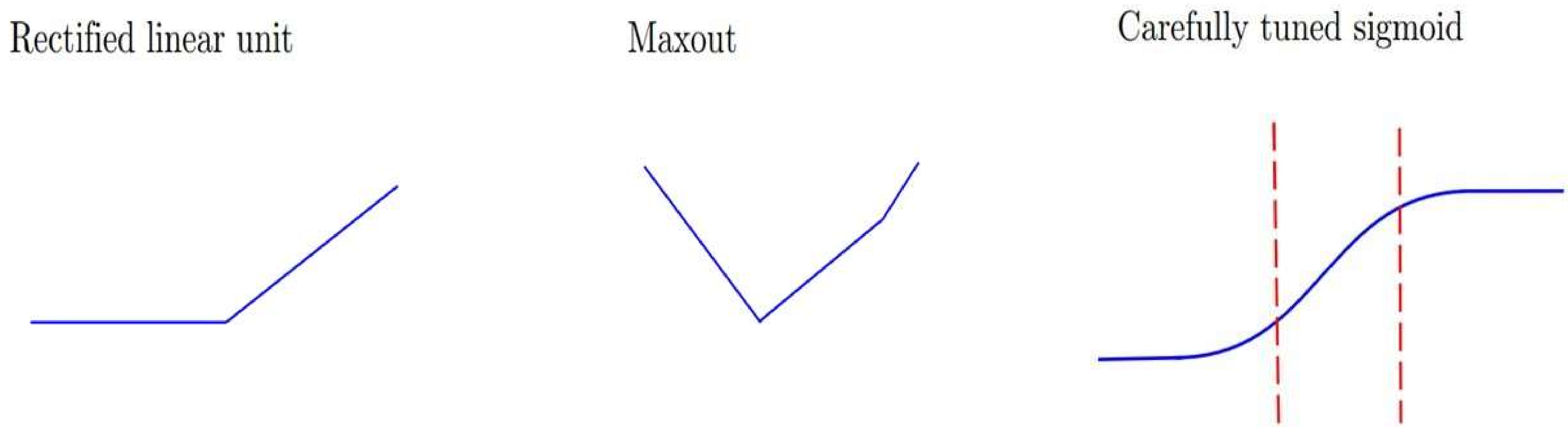
Neural networks aim to achieve **high accuracy** on test sets drawn **from the given distribution**

There are still many similar inputs that are never tested (and have low-probability for the given distribution)

The Story of Clever Hans

Why Do Adversarial Examples Exist?

Neural Networks are too linear

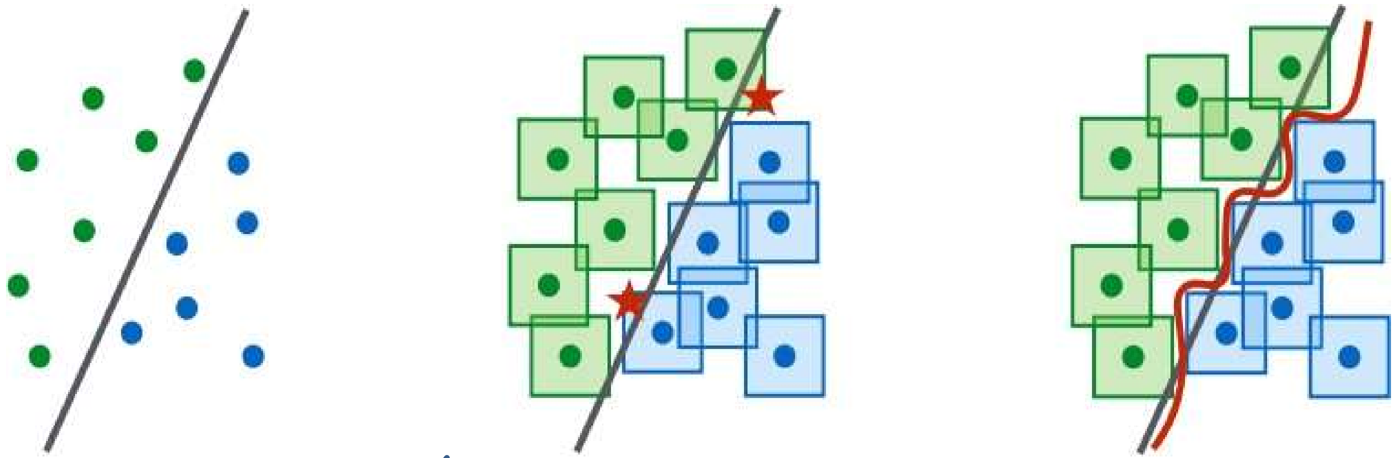


Why are they designed to be linear?

Linear functions are easy to optimize!

(Goodfellow 2016)

Neural Networks are too linear

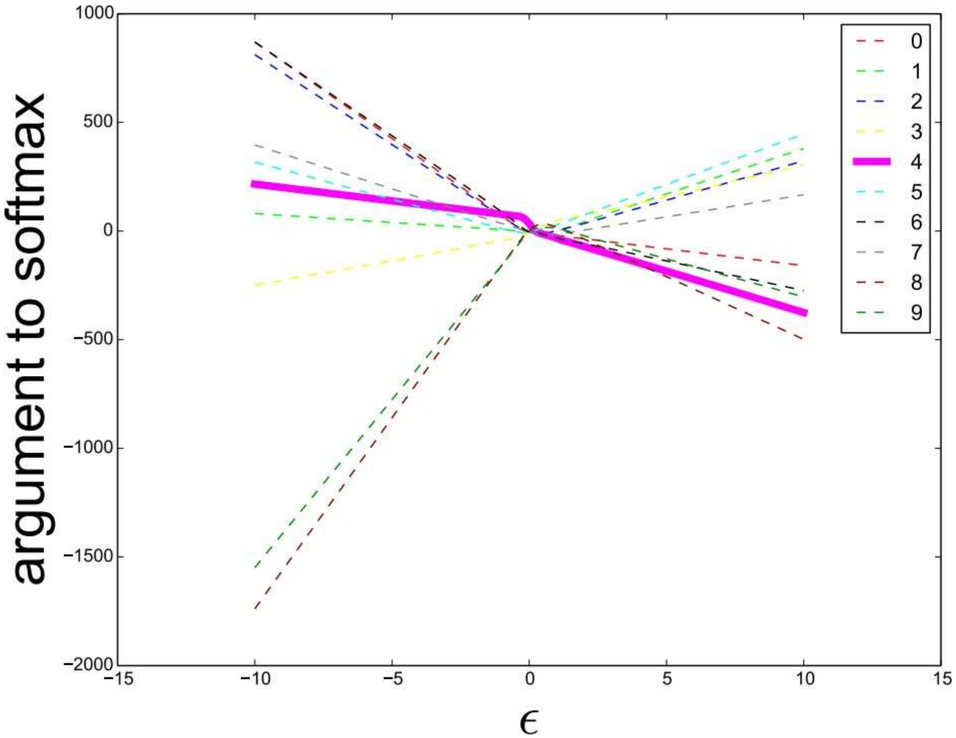


Learned Model not powerful enough to fit data properly

Learned Model trained on adversarial examples now more powerful (less linear)

(Madry 2017)

Experimental Linearity of Perturbations



- We have 1 image and 10 classes (0 to 9). The correct classification is 4.
- x-axis is perturbation (chose direction and perturb the image by that direction).
- y-axis are logits (values before calling softmax), unnormalized probabilities.

We see that the function which computes the particular logit, is basically almost (piece wise) linear in the perturbation.

Only around 0 (no perturbation) does the function behave in a non-linear manner and where classification is correct (i.e. 4).

Explaining and harnessing adversarial examples,
Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy, ICLR'15

Generating Adversarial Examples

(somewhat possible due to these mostly linear properties)

Targeted vs. Untargeted Attacks

Targeted Attack – aims to misclassify the input (e.g., image) to a specific label (e.g. panda to gibbon)

Untargeted Attack – aims to misclassify the input to any wrong label (e.g. panda to any other animal)

Formulated as a slightly different optimization problem

Targeted Attack: Problem Statement

Input:

- neural network $\boldsymbol{f}: \boldsymbol{X} \rightarrow \boldsymbol{C}$
- input $\boldsymbol{x} \in \boldsymbol{X}$
- target label $\boldsymbol{t} \in \boldsymbol{C}$, such that $\boldsymbol{f}(\boldsymbol{x}) \neq \boldsymbol{t}$

Output:

- A perturbation $\boldsymbol{\eta}$ such that $\boldsymbol{f}(\boldsymbol{x} + \boldsymbol{\eta}) = \boldsymbol{t}$

Adversarial example
$x' = x + \eta$

Untargeted Attack: Problem Statement

Input:

- neural network $\boldsymbol{f}: \boldsymbol{X} \rightarrow \boldsymbol{C}$
- input $\boldsymbol{x} \in \boldsymbol{X}$

Output:

- A perturbation $\boldsymbol{\eta}$ such that $\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{\eta}) \neq \boldsymbol{f}(\boldsymbol{x})$

Adversarial example

$x' = x + \eta$

Types of Attacks

White box attacker: the attacker knows the model, the parameters, and the architecture

Black box attacker: the attacker knows the architecture (e.g., the layers) but not its parameters (e.g., weights)

Note: it was found adversarial examples are **transferrable**, hence given the same training data as the original network, an attacker can train their own **mirror network** of the black box original network and then attack the mirror network with white-box techniques. If attack on mirror network succeeds, it will likely succeed on the original.

We will look at **white box attacks first**

Targeted Fast Gradient Sign Method

1. Compute perturbation:

$\eta = \epsilon \cdot \mathrm{sign}(\nabla_x \mathrm{loss}_t\ (x))$, where

$$\nabla_x \mathrm{loss}_t = \left(\frac{\partial \mathrm{loss}_t}{\partial x_1}, \dots, \frac{\partial \mathrm{loss}_t}{\partial x_n}\right) \qquad \mathrm{sign}(g) = \begin{cases} -1, & \text{if } g < 0 \\ 0, & \text{if } g = 0 \\ 1, & \text{if } g > 0 \end{cases}$$

2. Perturb the input:

$$x' = x - \eta$$

3. Check if:

$$f(x') = t$$

- Here, each x_i is a pixel
- ϵ is a very small constant (e.g., 0.007)
- As FGSM is 1-step, x' is guaranteed to stay inside the box [x - ϵ, x + ϵ], so no need to project.
- t is the target, bad label
- loss_t is the loss w.r.t target label
- FGSM was designed to be fast, not optimal (may not compute minimal perturbation)

Untargeted version of FGSM

1. **Compute perturbation:**

$$\eta = \epsilon \cdot \text{sign}(\nabla_x \text{loss}_s \ (x))$$

2. **Perturb the input:**

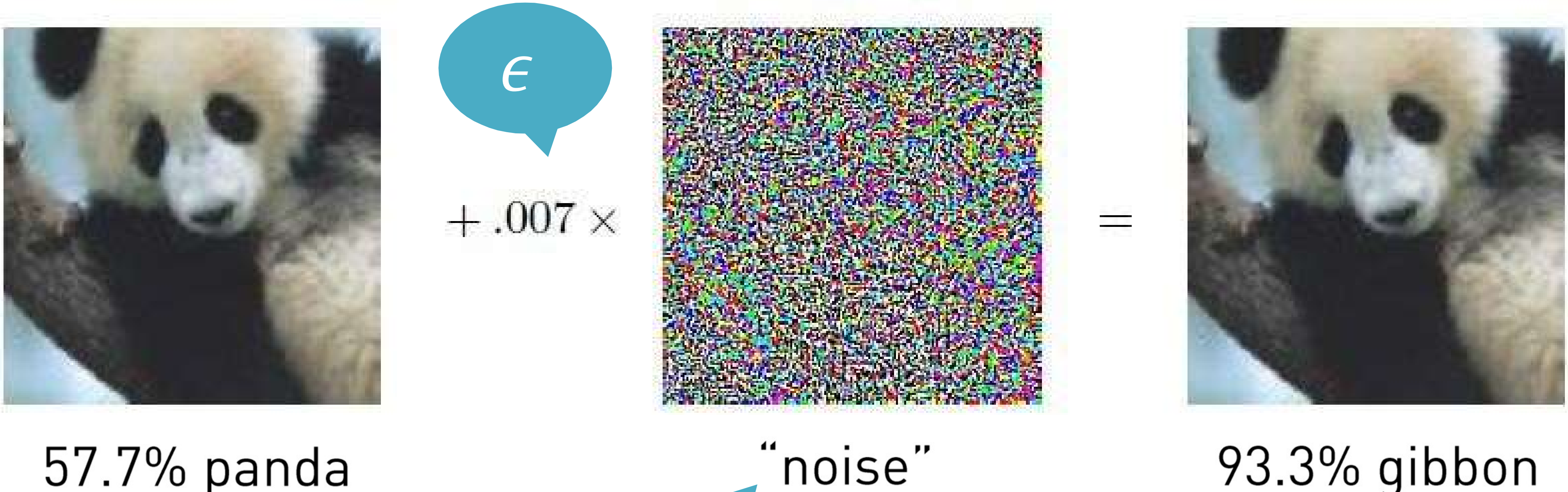
$$x' = x + \eta$$

3. **Check if:**

$$f(x') \neq S$$

- With untargeted FGSM, we do not know what the target (bad) label is that we want.
- We just want some label different than the correct label s.
- So we try to “get away” from the correct label by maximizing the value of the loss

FGSM



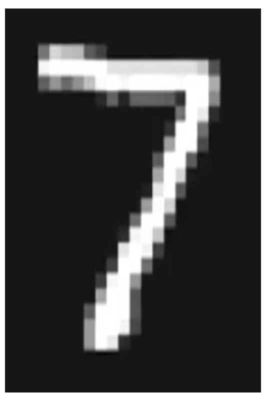
ϵ

$\mathrm{sign}(\nabla_x \mathrm{loss}_t(x))$

Explaining and Harnessing Adversarial Examples, ICLR’15

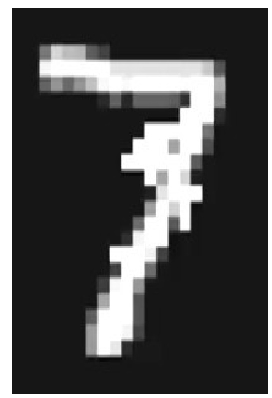
Importance of Small Perturbations

Original image



Too Perturbed Image

Slightly Perturbed Image



We need some notion of distance….

Norm: Notion of Distance

Similarity of $\boldsymbol{x} \sim \boldsymbol{x'}$ is usually captured by an l_p norm:

$$\boldsymbol{x} \sim \boldsymbol{x'} \text{ iff } \|\boldsymbol{x} - \boldsymbol{x'}\|_{\boldsymbol{p}} < \boldsymbol{\epsilon},$$

$$\text{where } \|\boldsymbol{x} - \boldsymbol{x'}\|_{\boldsymbol{p}} = \left((|\boldsymbol{x_1} - \boldsymbol{x'_1}|)^{\boldsymbol{p}} + \cdots + (|\boldsymbol{x_n} - \boldsymbol{x'_n}|)^{\boldsymbol{p}}\right)^{\frac{1}{p}}$$

l_0 (when $0^0 = 0$ and we get rid of 1/p root) captures the number of changed pixels.

l_2 captures the Euclidian distance between x and x'. It can remain small if there are many small changes to many pixels.

l_∞ captures **maximum noise (change)** added to any coordinate. It is the maximum of the absolute values of the entries:

$$\|\boldsymbol{x} - \boldsymbol{x'}\|_{\infty} = \boldsymbol{max}(|\boldsymbol{x_1} - \boldsymbol{x'_1}|, \ldots, |\boldsymbol{x_n} - \boldsymbol{x'_n}|)$$

This is the most common norm used for adversarial example generation and it is argued that it most naturally captures human vision.

To derive the max equation, see:
https://math.stackexchange.com/questions/3099179/proving-the-infinity-norm-is-equal-to-the-maximum-value-of-the-vector&sa=D&source=hangouts&ust=1600861806197000&usg=AFQjCNFLvo4hpAfiNBK06EwAZVFkuRiDNw

Logistics

- ***30 seconds self-introduction today!***

https://docs.google.com/presentation/d/1Hmn7cl7aj-dWzVotkYH-N0HriHJznDZ-d5evrCz3qMA/edit#slide=id.g26cef34236d_0_288

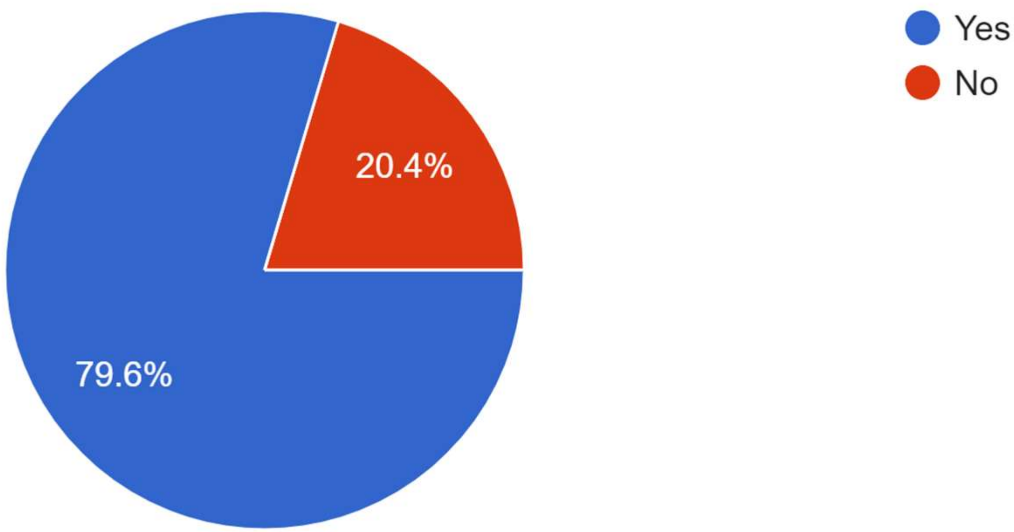
- Lit. Review paper selection and team form up due on April 8, sign up here:

https://docs.google.com/spreadsheets/d/1IWN_taP0FCrk4qtkF_PfhJaA8zBRv35XXEaT4uCTcp8/edit#gid=0

- If you want to propose your own project ideas, please email Yuan one-page slide about your idea by 04/06.

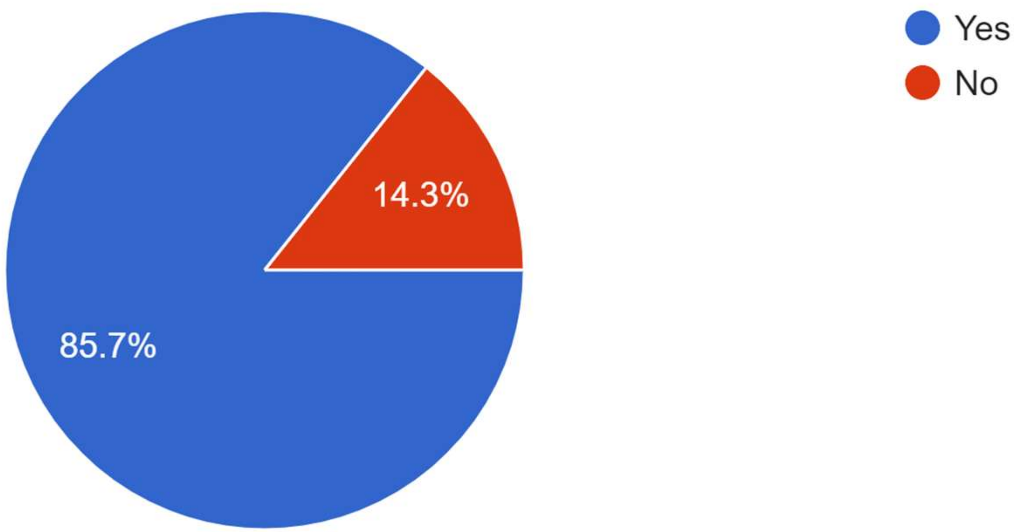
Pre-class survey

Are you familiar with Pytorch (i.e., training deep neural network with PyTorch)?
49 responses



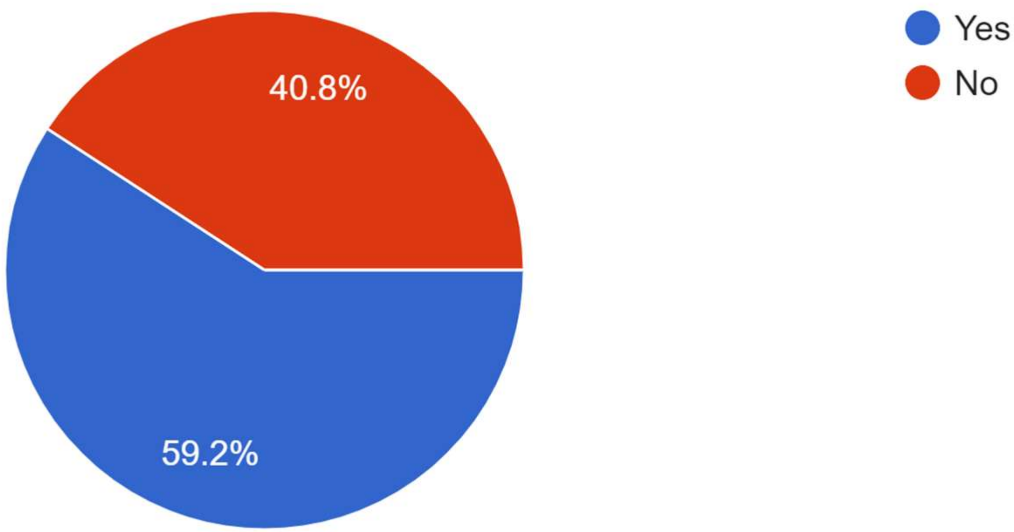
Have you ever used Google Colab before?

49 responses



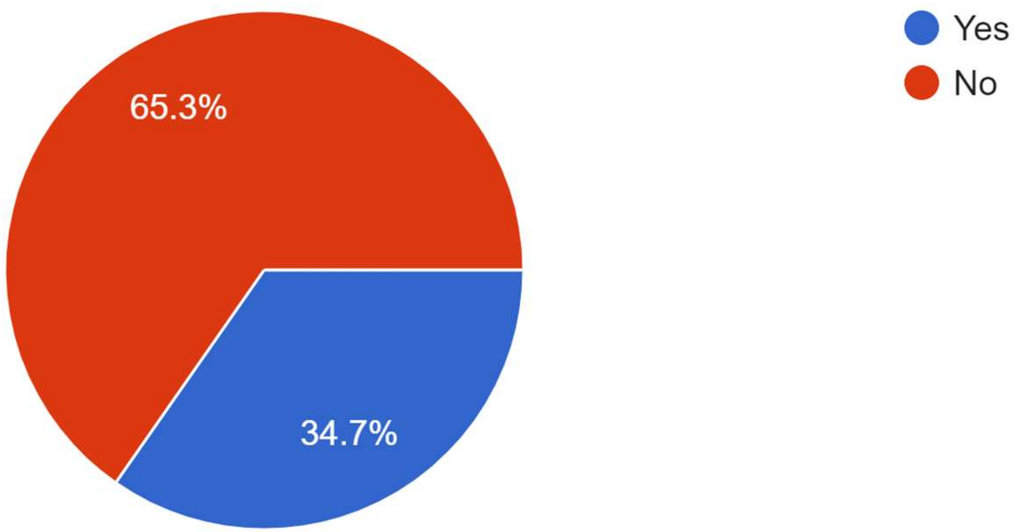
Do you know the concept of adversarial example?

49 responses



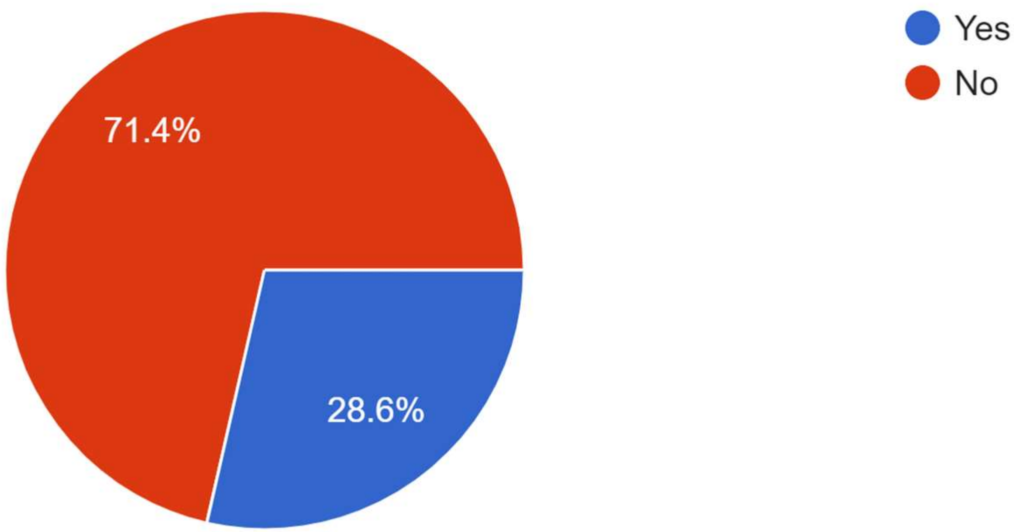
Do you know the concept of backdoor attack?

49 responses



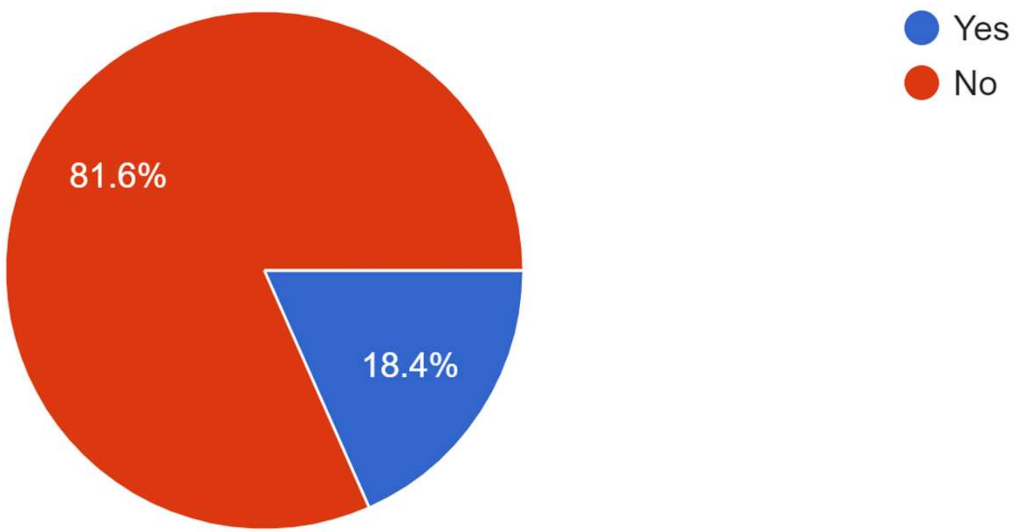
Do you know the concept of watermarking for neural network?

49 responses



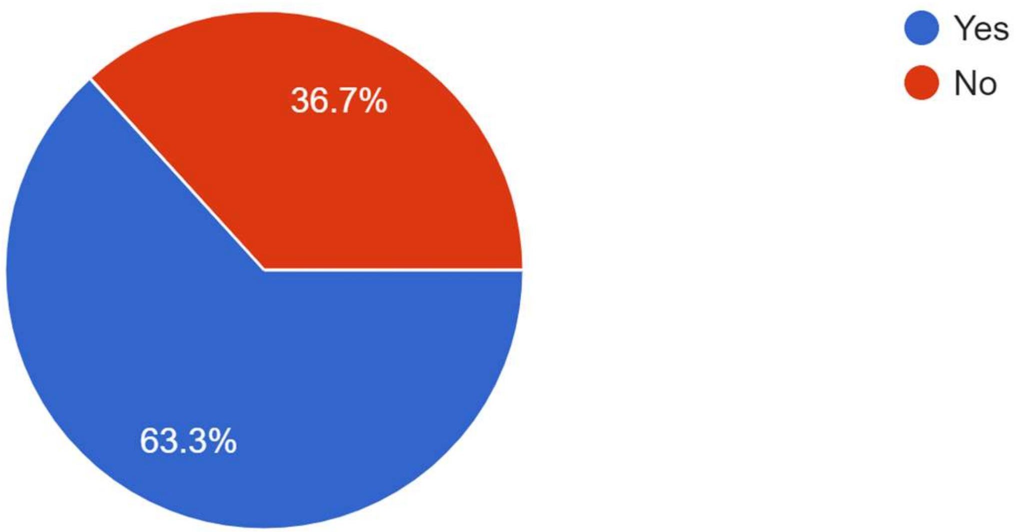
Do you know the concept of model stealing attacks?

49 responses



Have you ever participated in a research project on machine learning?

49 responses



Security classes

ECE 117
None
N/A
None, Control Theory & Systems Background
ECE 209AS: Secure and Trustworthy Edge Computing Systems
ECE 188/ECE 117: Computer Systems Security, ECE 188: Secure Computing Systems
ECE 209 AS Cyber Security
cryptography
computer security
Cybersecurity
209AS - Secure and Trustworthy Edge Computing Systems
Secure and Trustworthy Edge Computing Systems, Computer System Security
209AS by Nader
EC ENGR 209AS
Secure and Advanced Computer Architecture
network security
ECE 209AS - Secure and Trustworthy Edge Computing Systems
None - I am very new to security but excited to learn!
none
ECE 209: Mobile security
I do not have much experience with security, but am excited to learn. I had only taken an assembly language class before where they covered security.
ECE 209AS Secure and Trustworthy Edge Computing Systems
Secure and Advanced Computer Architecture (current), Secure and Trustworthy edge computing systems, Cryptographic Protocols, Introduction to Cyber Security (CSULB)

Machine learning classes

None, Control Theory & Systems Background
CS260, CS249 - several ML classes
ECE 219
ECE 188
Artificial Intelligence(undergrad), 239AS(computational imaging), 149(computer vision), 219(data mining)
CS245, CS264A, ECE219
ECE 147: Deep Learning & Neural Networks
N/A
Courses: CS M146, CS 161, CS 162; Certifications/Online Courses: DeepLearning.AI TensorFlow course by Andrew Ng and Laurence Moroney
into to machine learning
large scale data mining
None at UCLA. Have taken online courses on machine learning and neural networks, and trained and used models in projects and in internships I've completed.
CS146 (Intro to Machine Learning), CS188 (Deep Learning and Computer Vision), ECE147 (Deep Learning and Neural Networks), CS162 (Natural Language Processing)
247
Non-ucla: Deep Learning, ML for BME, regression analysis. UCLA: computational imaging (kadambi), large scale data mining (219)
Undergraduate level introduction to machine learning
CS 247, ECE 247, CS 146

Machine learning classes -cont

Large Scale Data Mining, Large Scale Networks,
Digital Speech Processing, natural language generation, machine learning algorithms (ECE247, CS260)
219
Intro to Machine Learning, Advanced Data Mining, Advanced Information Retrieval, Reinforcement Learning
Neural Networks
Large-Scale Data Mining: Models and Algorithms
neural networks and deep learning
Human-Centered Artificial Intelligence Systems
CS M148 - Intro to Data Science, CS 260B - Algorithmic Machine Learning, CS 260R - Reinforcement Learning
CS 146 Intro to Machine Learning and several NLP-specific classes.
C247
ECE247
machine learning, deep learning and neural network
Artificial Intelligence and Machine Learning, Neural Networks and Deep Learning
none
ECE C247: Neural Networks and Deep Learning
CS M146 - Intro to ML, ECE 149 - Computer Vision Fundamentals, ECE 239AS -- Special Topics in Imaging (we covered Gen AI), ECE 219 - Laege-Scaled Data Models
ECE C247 neural network & deep learning, ECE 219 large-scale data mining
CS 188 (Deep Learning for Computer Vision)
CS M146
EECS 195 ML for Engineers (UCI); EECS 118 Artificial Intelligence (UCI)
Introduction to deep learning (CSULB)
ECE C247, ECE 149

What's your expectation in taking this course?

More exposure to current issues related to building trustworthy AI
Learning something about AI Security
I learned adjusting and applying AI algorithms but never learned how to secure it, which I think is very important. I expect to learn about how the attacks happen and how to secure the model.
Learning more about how to defend and build trustworthy AI models rather than attack them from by background.
I want to learn more about machine learning and AI related stuff
Learning more about AI fairness and ethics, but, hopefully, mechanistic AI safety research areas.
To learn a lot about vulnerabilities in neural networks, and how to safeguard against them; concepts that would be useful in any potential internship or career that I pursue.

Learning and becoming familiar with any concept necessary to become a proficient researcher in security and privacy in machine learning
Learn more about AI
learn about the security aspects of production-level ML products
Expand knowledge basis and overall understanding of these topics.
Learning more about AI and security, with an emphasis on application and execution
Understanding the necessities of trustworthy AI generally and usefulness in medical domains
Learn the concept of trustworthy AI and how it works
My research in related to the field. I think this course would be helpful to gain insights about it.
Learn more about Trustworthiness in AI

I am interested in data transparency and security of machine learning models
Learning more about security concepts
To learn more about ways in which AI can be exploited, and hopefully how we can mitigate these.
I'm hoping to get a broad overview of what Trustworthy AI entails. My research focus is on fairness in NLP, so I would love to delve deeper into this subject in this class. I am also part of UCLA's Institute for Technology, Law, and Policy, so I am very interested in the papers we're planning to cover on AI regulation.
Get to learn about the security, which may help me in doing job with cloud computing and network security
have a good grade and learn sth
Learn how to defend cyber systems for industrial applications
To gain familiarity with AI security, and do some hands on work.

Preliminary understanding of A.I. an emerging technology that can be integrated into all industries.
To learn the basic knowledge of security. It may be helpful for my future career as a software engineer.
Learn more about trustworthness in AI
I think it will a really fun course to learn about all these different topics :)
learn about various fields of trustworthy ai
Explore embedded/edge AI security via the final project
I hope to learn some interesting cyber security aspects of AI/ML. My background is mostly in cyber so I am a little worried about implemnting and using ML algoithms/libraries for this class.

Targeted Attack with Small Changes

Input:

- neural network $\boldsymbol{f}: \boldsymbol{X} \rightarrow \boldsymbol{C}$
- input $\boldsymbol{x} \in \boldsymbol{X}$
- target label $\boldsymbol{t} \in \boldsymbol{C}$, such that $\boldsymbol{f}(\boldsymbol{x}) \neq \boldsymbol{t}$

Output:

- A perturbation $\boldsymbol{\eta}$ such that $\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{\eta}) = \boldsymbol{t}$
- $||\boldsymbol{\eta}||_{\boldsymbol{p}}$ **is minimized**

Optimization Problem

The problem of generating small perturbations can be phrased as an optimization problem:

$p \in \{0, 2, \infty\}$

find	$\boldsymbol{\eta}$
minimize	$\lVert\boldsymbol{\eta}\rVert_p$
such that	$f(\boldsymbol{x}+\boldsymbol{\eta}) = t$
	$\boldsymbol{x}+\boldsymbol{\eta} \in [0,1]^n$

This is a hard discrete constraint which is difficult to optimize for with gradient methods.

Note: $\boldsymbol{\eta}$ can have negative components.

Key insight: Relaxation of the hard constraint

Carlini et al., Towards Evaluating the Robustness of Neural Networks, 2017

Optimization Problem

Two steps:

Step 1: Define an objective function $\boldsymbol{obj_t}$ such that:

$$\text{if } \boldsymbol{obj_t(x+\eta) \leq 0} \quad \text{then } \boldsymbol{f\,(x+\eta) = t}$$

Step 2: Solve the following optimization problem:

find	$\boldsymbol{\eta}$
minimize	$\|\|\boldsymbol{\eta}\|\|_p + c \cdot \boldsymbol{obj_t(x+\eta)}$
such that	$\boldsymbol{x+\eta} \in [\boldsymbol{0}, \boldsymbol{1}]^{\boldsymbol{n}}$

Optimization Problem

Two steps:

Step 1: Define an objective function $\boldsymbol{obj_t}$ such that:

$$\text{if } \boldsymbol{obj_t(x+\eta) \leq 0} \text{ then } \boldsymbol{f}\,(\boldsymbol{x}+\boldsymbol{\eta}) = \boldsymbol{t}$$

What are examples of functions for $\boldsymbol{obj}$ with the property of Step 1?

Choice I:

$$\boldsymbol{obj_t}(\boldsymbol{x}') = \mathrm{loss_t}(\boldsymbol{x}') - 1$$

Lets take cross entropy loss for loss_t

Choice II:

$$\boldsymbol{obj_t}(\boldsymbol{x}') = \max(0, 0.5 - \mathbf{p_f}(\boldsymbol{x}')_{\mathrm{t}})$$

$\mathbf{p_f}(\boldsymbol{x}')_{\mathrm{t}}$ returns the probability of class $\boldsymbol{t}$ for input $\boldsymbol{x}'$ on network **f**

Choice I: $\boldsymbol{obj}(\boldsymbol{x}) = \mathrm{loss}_{\mathrm{t}}(\boldsymbol{x}) \; - 1$

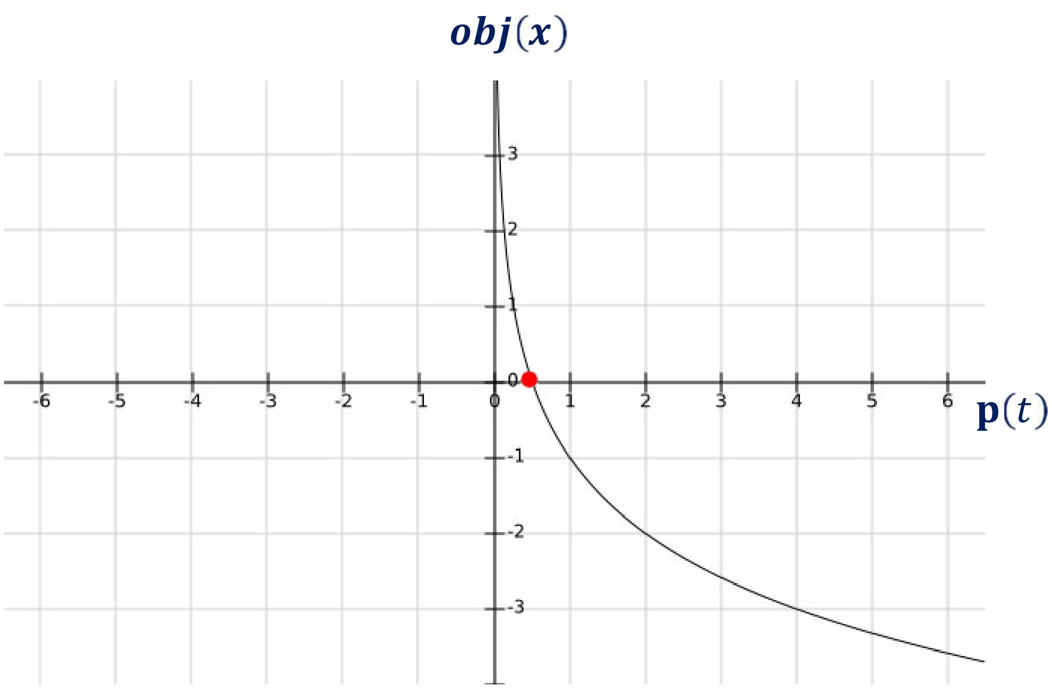
Choice I:

$$\boldsymbol{obj}_{\boldsymbol{t}}(\boldsymbol{x}) = \mathrm{loss}_{\mathrm{t}}(\boldsymbol{x}) \; - 1$$

$$= -\mathbf{log}_2(\mathbf{p}(t)) - 1$$

Plug in cross entropy loss for loss_t with logarithm base 2

Here, we use $\mathbf{p}(t)$ as a shortcut for $\mathbf{p}_{\mathbf{f}}(\boldsymbol{x})_{\mathrm{t}}$ so to avoid clutter



What we see here is that if the $\boldsymbol{obj}_{\boldsymbol{t}}$ function is 0 or negative, then the probability $\mathbf{p}(t)$ is ■0.5 (50%) .

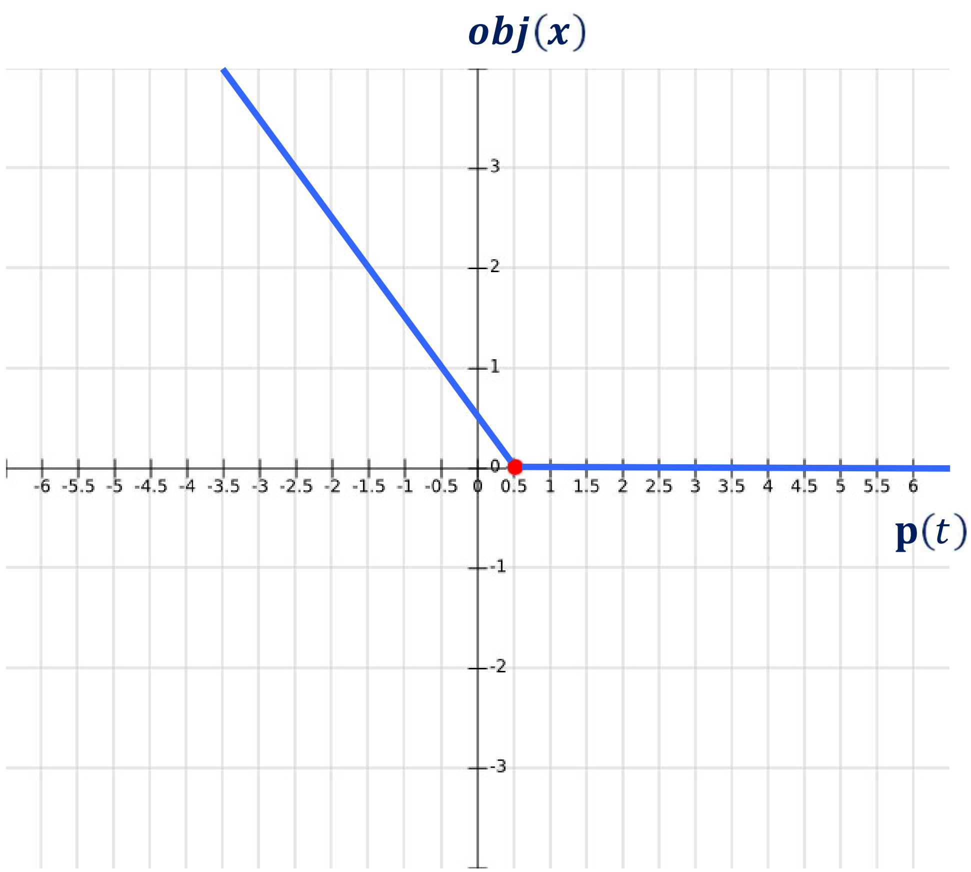
But if $\mathbf{p}(t)$ is ■0.5 for the input $\boldsymbol{x}$, then $\boldsymbol{f}$ will return t as a classification for $\boldsymbol{x}$ because this is the highest probability class. Hence, the desired property of Step 1 holds.

Choice II: $\max(0, 0.5 - \mathbf{p_f}(\boldsymbol{x})_t)$

Choice II:

$$\boldsymbol{obj_t}(\boldsymbol{x}) = \max(0, 0.5 - \mathbf{p}(t))$$

Again we use $\mathbf{p}(t)$ as a shortcut for $\mathbf{p_f}(\boldsymbol{x})_t$ so to avoid clutter



What we see here is that the $\boldsymbol{obj_t}$ function is always 0 or greater.

It is only 0 when $\mathbf{p}(t)$ is ▪0.5 for the input $\boldsymbol{x}$.

Again, then $\boldsymbol{f}$ will return t as a classification for $\boldsymbol{x}$ because this is the highest probability class.

Hence, the desired property holds for Step 1.

Optimization Problem

Two steps:

Step 1: Define an objective function $\boldsymbol{obj_t}$ such that:

$$\text{if } \boldsymbol{obj_t(x+\eta) \le 0} \quad \text{then } \boldsymbol{f\ (x+\eta) = t}$$

Step 2: Solve the following optimization problem:

find	$\boldsymbol{\eta}$
minimize	$\lVert\boldsymbol{\eta}\rVert_p + c \cdot obj_{\boldsymbol{t}}(\boldsymbol{x}+\boldsymbol{\eta})$
such that	$\boldsymbol{x}+\boldsymbol{\eta} \in [\mathbf{0},\mathbf{1}]^{\boldsymbol{n}}$

Optimization Problem

Two steps:

Step 1: Define an objective function $\boldsymbol{obj_t}$ such that:

$$\text{if } \boldsymbol{obj_t}(\boldsymbol{x}+\boldsymbol{\eta}) \le \mathbf{0} \quad \text{then } \boldsymbol{f}\,(\boldsymbol{x}+\boldsymbol{\eta}) = \boldsymbol{t}$$

Step 2: Solve the following optimization problem:

find	$\boldsymbol{\eta}$
minimize	$\|\boldsymbol{\eta}\|_\infty + c \cdot obj_{\boldsymbol{t}}(\boldsymbol{x}+\boldsymbol{\eta})$
such that	$\boldsymbol{x}+\boldsymbol{\eta} \in [\mathbf{0},\mathbf{1}]^n$

This is a problem for optimization

Lets take a closer look at $||\boldsymbol{\eta}||_{\infty}$

$||\boldsymbol{\eta}||_{\infty}$ computes the maximum change: it takes the absolute value of every coordinate in $\boldsymbol{\eta}$ and returns the maximum value.

$\boldsymbol{\eta}$ = (0.5, 0.49, 0.48)

$$\frac{\partial||\boldsymbol{\eta}||_{\infty}}{\partial\boldsymbol{\eta}_1} = 1 \qquad \frac{\partial||\boldsymbol{\eta}||_{\infty}}{\partial\boldsymbol{\eta}_2} = 0 \qquad \frac{\partial||\boldsymbol{\eta}||_{\infty}}{\partial\boldsymbol{\eta}_3} = 0$$

After one step we get: $\boldsymbol{\eta} = \boldsymbol{\eta} - \gamma \cdot (1,0,0) = \boldsymbol{\eta} - 0.03 \cdot (1,0,0)$ = (0.47, 0.49, 0.48)

After two steps we get: $\boldsymbol{\eta} = \boldsymbol{\eta} - \gamma \cdot (0,1,0) = \boldsymbol{\eta} - 0.03 \cdot (0,1,0)$ = (0.47, 0.46, 0.48)

After three steps we get: $\boldsymbol{\eta} = \boldsymbol{\eta} - \gamma \cdot (0,0,1) = \boldsymbol{\eta} - 0.03 \cdot (0,0,1)$ = (0.47, 0.46, 0.45)

What we see is that because the gradient is 0 at all non-max locations, the gradient does not impose a penalty on the optimizer increasing a little bit those locations (due to the $obj_t(\boldsymbol{x} + \boldsymbol{\eta})$ term in the optimization). Also, only one entry is changed at a time.

Lets take a closer look at $||\boldsymbol{\eta}||_{\infty}$

Going back to the full optimization problem which also includes $obj(\boldsymbol{x} + \boldsymbol{\eta})$

After one step we get: $\boldsymbol{\eta} = \boldsymbol{\eta} - \gamma \cdot (1,0,0) = \boldsymbol{\eta} - 0.03 \cdot (1,0,0)$ = (0.47, 0.49, 0.48)

Now the optimizer can slightly bump up the second location:

After one full step of optimizer, it may also bump up the 2nd location: (0.47, 0.5, 0.48)

After second full step of optimizer, we may get: (0.5, 0.47, 0.48)

Well, we are just oscillating now and bouncing around...turns out SGD may not be a good way to optimize $||\boldsymbol{\eta}||_{\infty}$ especially with other terms

One approach to solving the issue

Replace $||\boldsymbol{\eta}||_\infty$ with other proxy functions that reflect the distance

One idea is to penalize large values in $\boldsymbol{\eta}$ via a term τ:

Replace $||\boldsymbol{\eta}||_\infty$ with $\sum i \max(0, (|\boldsymbol{\eta}_i| - \tau))$

- τ is basically intended to capture the $||\boldsymbol{\eta}||_\infty$ bound when optimization finishes.
- τ will be continuosly minimized. Initially, τ starts at 1
- τ is decreased with some factor (say 0.9) at every iteration if all $|\boldsymbol{\eta}_i|$ are less than τ (then, entire expression will be 0).

Note: an iteration consist of K small steps.

Note: when τ is large, gradient of $\sum i \max(0, (|\boldsymbol{\eta}_i| - \tau))$ is similar to gradient of $||\boldsymbol{\eta}||_\infty$.

Example & Notes on Optimization

Let $L(\boldsymbol{\eta}) = \sigma_i \max(0, (|\boldsymbol{\eta}_i| - \tau))$ and $\boldsymbol{\eta}$ = (0.47, 0.49, 0.48)

Start with τ = 1, then $L(\boldsymbol{\eta})$ = 0

Next iteration: τ = 0.9, then $L(\boldsymbol{\eta})$ = 0

Next iteration: τ = 0.81, then $L(\boldsymbol{\eta})$ = 0

…

At some iteration: τ = 0.478, now via one step, we get:

$$L(\boldsymbol{\eta}) = \sigma_i \max\left(0, (|\boldsymbol{\eta}_i| - 0.478)\right) = \sigma_i (0, 0.012, 0.002) = \mathbf{0.014}$$

$$\nabla L(\boldsymbol{\eta}) = (\mathbf{0}, \mathbf{1}, \mathbf{1})$$

We can then update $\boldsymbol{\eta}$ as usual, complete this step, and continue with the next step

Here we only show the optimization of one term, namely $L(\boldsymbol{\eta})$, to illustrate the relationship with optimizing $||\boldsymbol{\eta}||_\infty$

Notes on optimization:

- There are K steps within an iteration, each updating $\boldsymbol{\eta}$.
- Entire optimization stops if after K steps $L(\boldsymbol{\eta}) \neq 0$. Otherwise, if $L(\boldsymbol{\eta}) = 0$, optimization continues with a new τ = 0.9 * previous τ.
- Entire optimization stops if it also reaches some pre-defined value of τ (1/256 for Carlini & Wagner).
- When optimization stops, we return $\boldsymbol{\eta}$ at the iteration before the last one. This means $||\boldsymbol{\eta}||_\infty \leq \tau$ where τ is the one used at iteration before last.
- If $\boldsymbol{\eta}_{top2} < r < \boldsymbol{\eta}_{top1}$ where $\boldsymbol{\eta}_{top1}$ is the largest element and $\boldsymbol{\eta}_{top2}$ is second largest element in $\boldsymbol{\eta}$, then the gradient $\nabla_{\boldsymbol{\eta}} L(\boldsymbol{\eta})$ will be the same as $\nabla_{\boldsymbol{\eta}} ||\boldsymbol{\eta}||_\infty$

Summary: Optimization Problem

Two steps:

Step 1: Define an objective function $\boldsymbol{obj_t}$ such that:

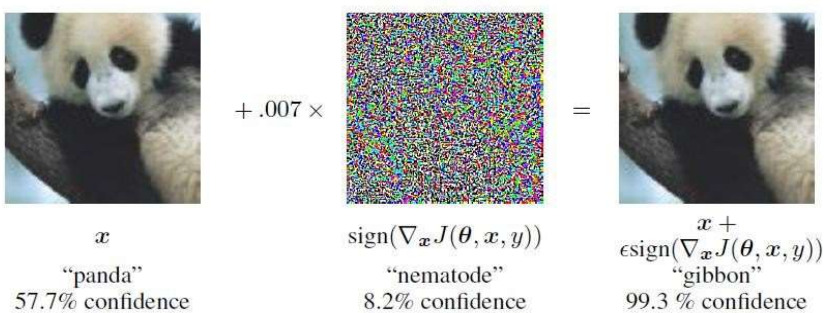
if $\boldsymbol{obj_t(x+\eta) \leq 0}$ then $\boldsymbol{f\ (x+\eta) = t}$

Step 2: Solve the following optimization problem:

find	$\boldsymbol{\eta}$
minimize	$\lVert\boldsymbol{\eta}\rVert_\infty + c \cdot obj_t(\boldsymbol{x}+\boldsymbol{\eta})$
such that	$\boldsymbol{x}+\boldsymbol{\eta} \in [\mathbf{0}, \mathbf{1}]^n$

Lecture Summary

Deep Learning is susceptible to adversarial examples in various domains



Generating Adversarial examples (basically, an optimization problem)

- FGSM: targeted and untargeted
- Small perturbation attacks
- Need suitable optimization problem

Next lecture: Dealing with Constraints and Adversarial defenses