

Adversarial attacks and their defences

Bhiksha Raj, Joseph Keshet and Raphael Olivier

Interspeech 2019

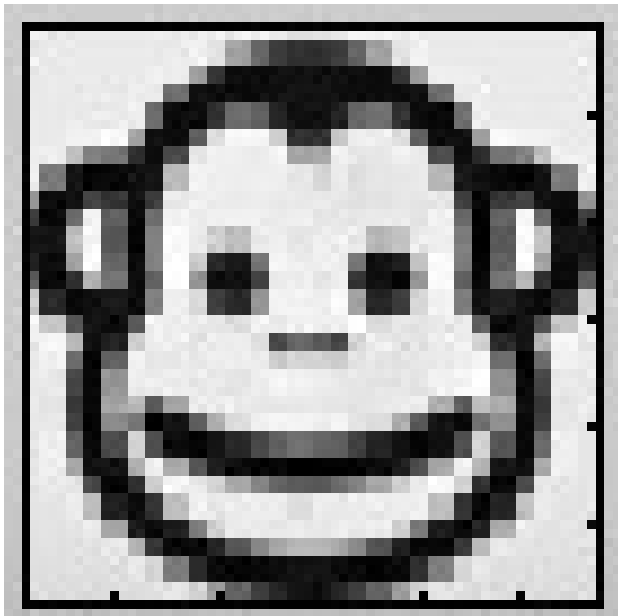
15 Sep 2019, Graz

Slides at: <https://tinyurl.com/ISP2019-Adversarial-Tutorial>

The story of “O”



One day in summer 2013



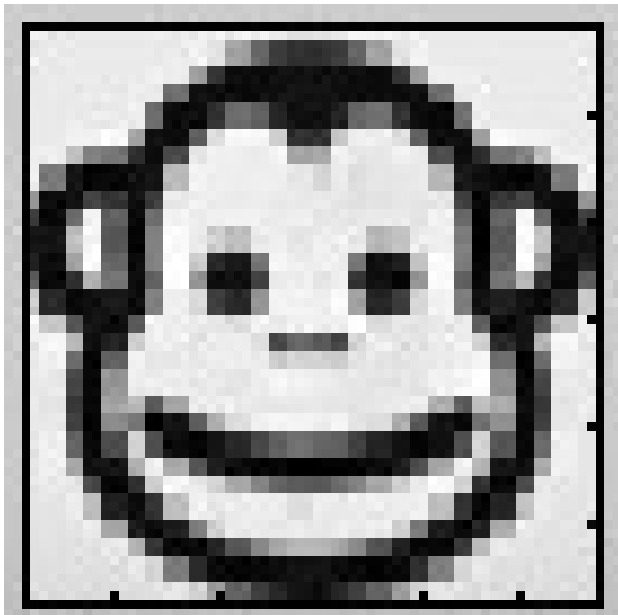
Boss, my MNIST recognizer thinks this monkey is the number 2!



The story of O:

"O" (a PhD student) has just written his own ultra-efficient distributed matlab-based deep learning toolkit

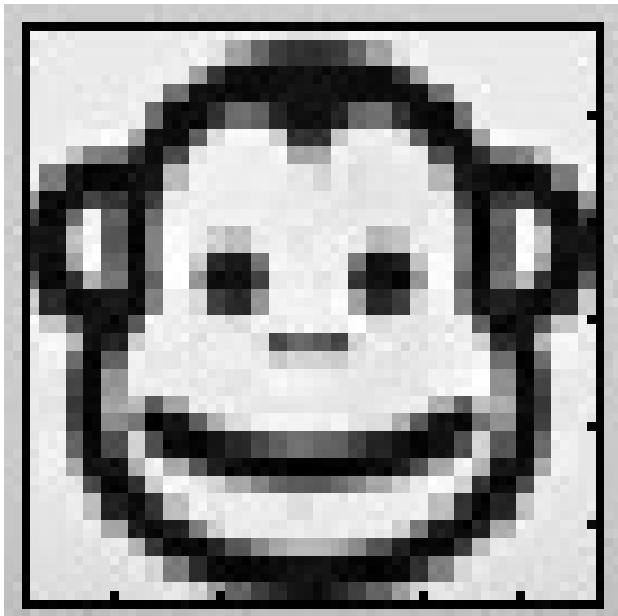
One day in summer 2013



2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2

??

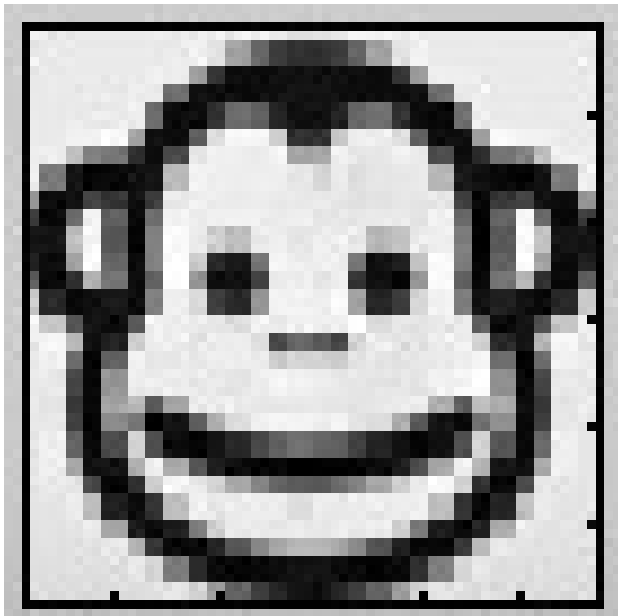
One day in summer 2013



No way! That
looks more like
an 8 or a 0



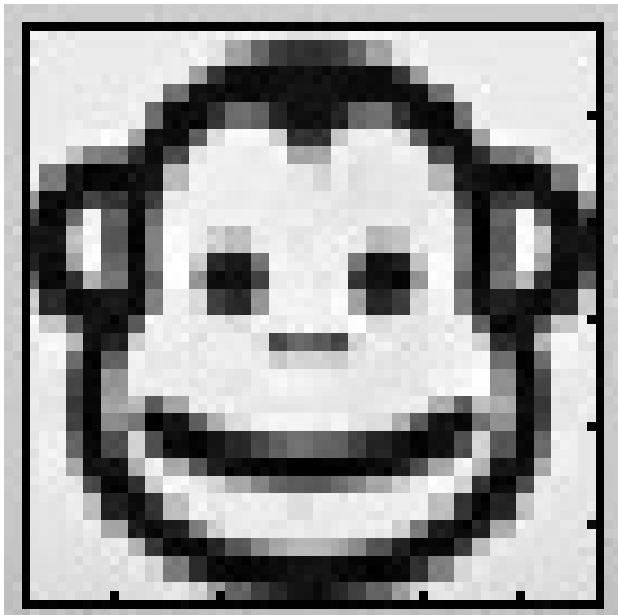
One day in summer 2013



Nope! It's the
number 2!



One day in summer 2013



Hm! I wonder
why. Try erasing
the smile.



One day in summer 2013



Now its an 8!



One day in summer 2013

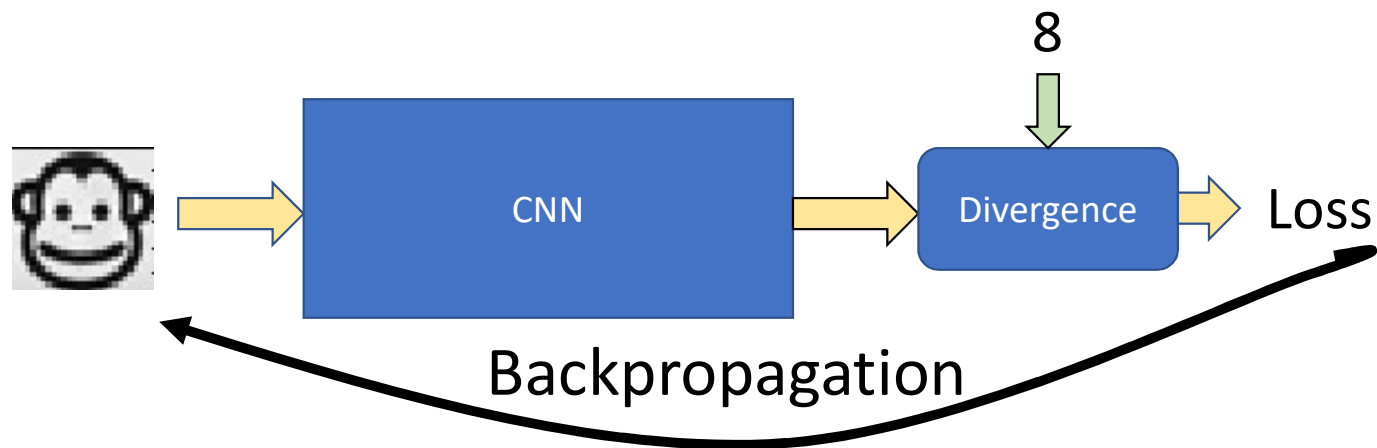
Can we
automatically figure
out how to edit it to
make it 8?



Sure! I know
how to do it.

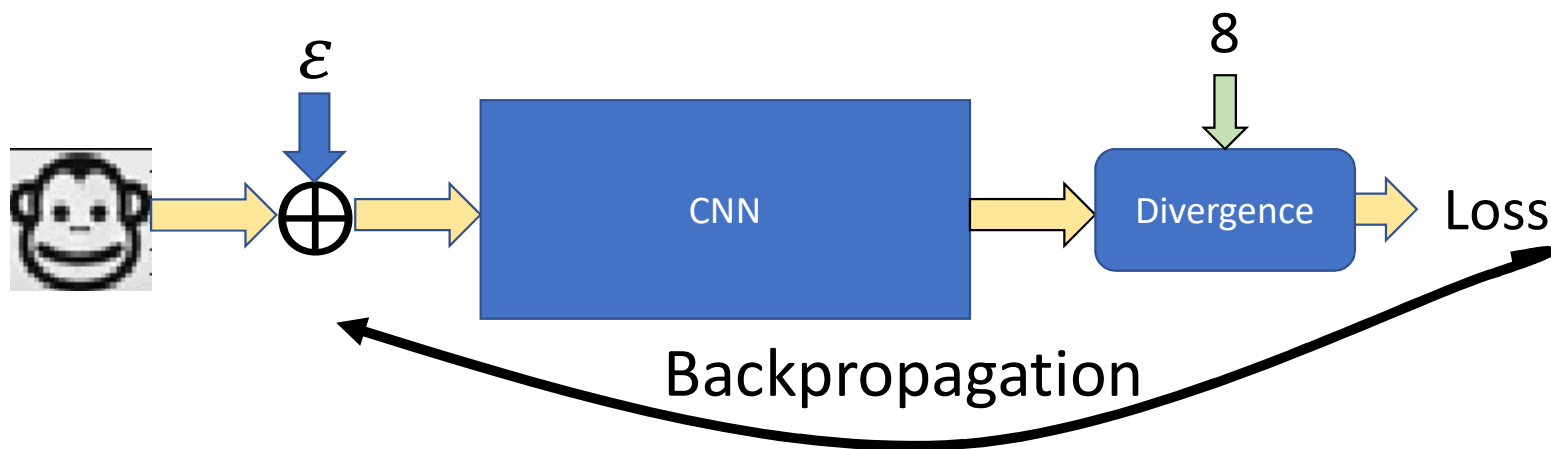


O makes a monkey an 8



- Backpropagate the error all the way back to the input to modify the *input*
 $\underset{x}{\operatorname{argmin}} \operatorname{Div}(\operatorname{CNN}(x), 8)$

O makes a monkey an 8



- Backpropagate the error all the way back to the input to modify the *input*, but keep the corrections small

$$\operatorname{argmin}_{\varepsilon} \operatorname{Div}(\operatorname{CNN}(x + \varepsilon), 8) + \lambda ||\varepsilon ||^2$$

0 makes a monkey 8

Boss, I made a
monkey 8!

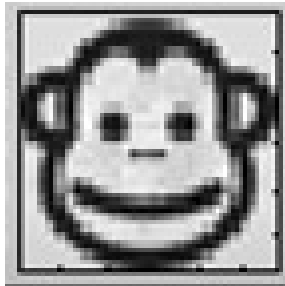


Neat! Perhaps
you can also
make it a 0?

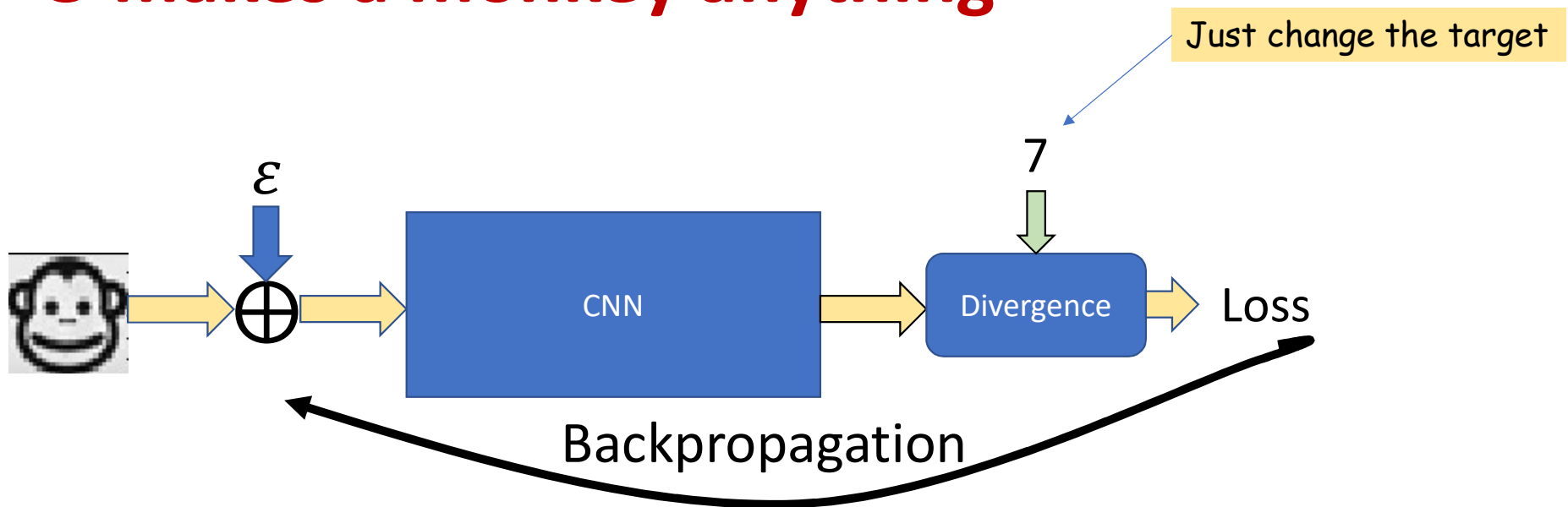


O can make a monkey anything

I can make it
anything!



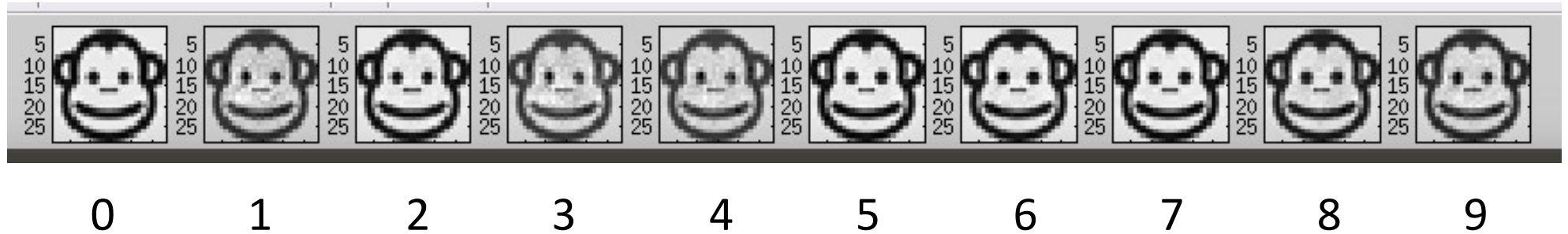
O makes a monkey anything



- Backpropagate the error all the way back to the input to modify the *input*, but keep the corrections small

$$\operatorname{argmin}_{\varepsilon} \operatorname{Div}(\operatorname{CNN}(x + \varepsilon), 7) + \lambda \|\varepsilon\|^2$$

The monkey digits



- Monkey figures can be minimally edited to make O's MNIST CNN recognize them as any digit of our choice!

Other figures

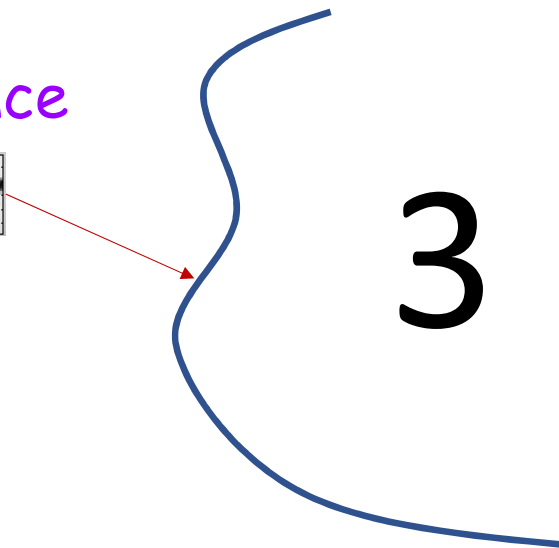
- In fact, you can do this with *any* figure



Fooling a classifier

- Any input can be minimally perturbed to fool the classifier into classifying it as any class!
 - Perturbations can be so small as to be imperceptible to a human observer

The monkey distance



People have been fooling classifiers for millennia!



- For as long as classifiers have existed

And we have had defences!



And more recently



- Spammers had been fooling spam filters for *decades* already

The History of Email

- The first “E-mail” :1965
 - MIT’s “Compatible Time-Sharing System” (CTSS)

The History of Email Spam

- The first “E-mail” :1965
 - MIT’s “Compatible Time-Sharing System” (CTSS)
- The first email spam: 1 May 1978
 - By Digital Equipment Corporation
 - Although it wasn’t *called* “spam” until April 1993

DIGITAL WILL BE GIVING A PRODUCT PRESENTATION OF THE NEWEST MEMBERS OF THE DECSYSTEM-20 FAMILY; THE DECSYSTEM-2020, 2020T, 2060, AND 2060T. THE DECSYSTEM-20 FAMILY OF COMPUTERS HAS EVOLVED FROM THE TENEX OPERATING SYSTEM AND THE DECSYSTEM-10 <PDP-10> COMPUTER ARCHITECTURE. BOTH THE DECSYSTEM-2060T AND 2020T OFFER FULL ARPANET SUPPORT UNDER THE TOPS-20 OPERATING SYSTEM. THE DECSYSTEM-2060 IS AN UPWARD EXTENSION OF THE CURRENT DECSYSTEM 2040 AND 2050 FAMILY. THE DECSYSTEM-2020 IS A NEW LOW END MEMBER OF THE DECSYSTEM-20 FAMILY AND FULLY SOFTWARE COMPATIBLE WITH ALL OF THE OTHER DECSYSTEM-20 MODELS.

WE INVITE YOU TO COME SEE THE 2020 AND HEAR ABOUT THE DECSYSTEM-20 FAMILY AT THE TWO PRODUCT PRESENTATIONS WE WILL BE GIVING IN CALIFORNIA THIS MONTH. THE LOCATIONS WILL BE:

TUESDAY, MAY 9, 1978 – 2 PM
HYATT HOUSE (NEAR THE L.A. AIRPORT)
LOS ANGELES, CA

THURSDAY, MAY 11, 1978 – 2 PM
DUNFEY’S ROYAL COACH
SAN MATEO, CA
(4 MILES SOUTH OF S.F. AIRPORT AT BAYSHORE, RT 101 AND RT 92)

A 2020 WILL BE THERE FOR YOU TO VIEW. ALSO TERMINALS ON-LINE TO OTHER DECSYSTEM-20 SYSTEMS THROUGH THE ARPANET. IF YOU ARE UNABLE TO ATTEND, PLEASE FEEL FREE TO CONTACT THE NEAREST DEC OFFICE FOR MORE INFORMATION ABOUT THE EXCITING DECSYSTEM-20 FAMILY.

The earliest spam filter

- The first “E-mail” :1965
 - MIT’s “Compatible Time-Sharing System” (CTSS)
- The first email spam: 1 May 1978
 - By Digital Equipment Corporation
 - Although it wasn’t *called* “spam” until April 1993
- Earliest attempts at prevention of SPAM: 1996
 - “Mail abuse prevention system” (MAPS, opposite of SPAM)

And the first adversarial attack

- The first “E-mail” :1965
 - MIT’s “Compatible Time-Sharing System” (CTSS)
- The first email spam: 1 May 1978
 - By Digital Equipment Corporation
 - Although it wasn’t *called* “spam” until April 1993
- Earliest attempts at prevention of SPAM: 1996
 - “Mail abuse prevention system” (MAPS, opposite of SPAM)
- Earliest address spoofing attack: 1997

History of ML-based spam filtering

- The first “E-mail” :1965
 - MIT’s “Compatible Time-Sharing System” (CTSS)
- The first email spam: 1 May 1978
 - By Digital Equipment Corporation
 - Although it wasn’t *called* “spam” until April 1993
- Earliest attempts at prevention of SPAM: 1996
 - “Mail abuse prevention system” (MAPS, opposite of SPAM)
- Earliest address spoofing attack: 1997
- Earliest ML based spam filter: 20 April 2001
 - Spam Assassin

... and adversarial attacks on ML systems

- The first “E-mail” :1965
 - MIT’s “Compatible Time-Sharing System” (CTSS)
- The first email spam: 1 May 1978
 - By Digital Equipment Corporation
 - Although it wasn’t *called* “spam” until April 1993
- Earliest attempts at prevention of SPAM: 1996
 - “Mail abuse prevention system” (MAPS, opposite of SPAM)
- Earliest address spoofing attack: 1997
- Earliest ML based spam filter: 20 April 2001
 - Spam Assassin
- Earliest adversarial attack on an ML Spam filter: 21 April 2001

Spam becomes a thing of the past

- The first “E-mail” :1965
 - MIT’s “Compatible Time-Sharing System” (CTSS)
- The first email spam: 1 May 1978
 - By Digital Equipment Corporation
 - Although it wasn’t *called* “spam” until April 1993
- Earliest attempts at prevention of SPAM: 1996
 - “Mail abuse prevention system” (MAPS, opposite of SPAM)
- Earliest address spoofing attack: 1997
- Earliest ML based spam filter: 20 April 2001
 - Spam Assassin
- Earliest adversarial attack on an ML Spam filter: 21 April 2001
- **Bill Gates declares Spam “soon to be a thing of the past” : January 2004**

More seriously..

- AI systems have taken over the world
 - Largely powered by powerful machine-learned pattern recognition systems
 - In turn largely powered by artificial neural networks

ASR

www.technewsworld.com/story/84013.html

top 40 maps that explain Amazon Web Services Primers | Math n Pro: deeplearning.net/tuto Deep Learning Tutori deep learning PHILIPS - Golden Ears Language Technology MyIDCare - Dashboa Other bookmarks

TECHNEWSWORLD EMERGING TECH

Computing Internet IT Mobile Tech Reviews Security Technology Tech Blog Reader Services

Microsoft AI Beats Humans at Speech Recognition

By Richard Adhikari
Oct 20, 2016 11:40 AM PT

Print Email




Image: Adobe Stock

Microsoft's Artificial Intelligence and Research Unit earlier this week reported that its speech recognition technology had surpassed the performance of human transcriptionists.

Google+ 5
Tweet 25
Facebook Share 45
LinkedIn Share 11
Reddit Share 0
StumbleUpon share 104

Most Popular Newsletters News Alerts

How do you feel about Black Friday and Cyber Monday?

- ☐ They're great -- I get a lot of bargains!
- ☐ The deals are too spread out -- I'd prefer just one day.
- ☐ They're a fun way to kick off the holiday season.
- ☐ I don't like the commercialization of Thanksgiving Day.
- ☐ They're crucial for the retail industry and the economy.
- ☐ The deals typically aren't that good.

Vote to See Results

E-Commerce Times

- Black Friday Shoppers Hungry for New Experiences, New Tech
- Pay TV's Newest Innovation: Giving Users Control
- Apple Celebrates Itself in \$300 Coffee Table Tome
- AWS Enjoys Top Perch in IaaS, PaaS Markets
- US Comptroller Gears Up for Blockchain and

MT

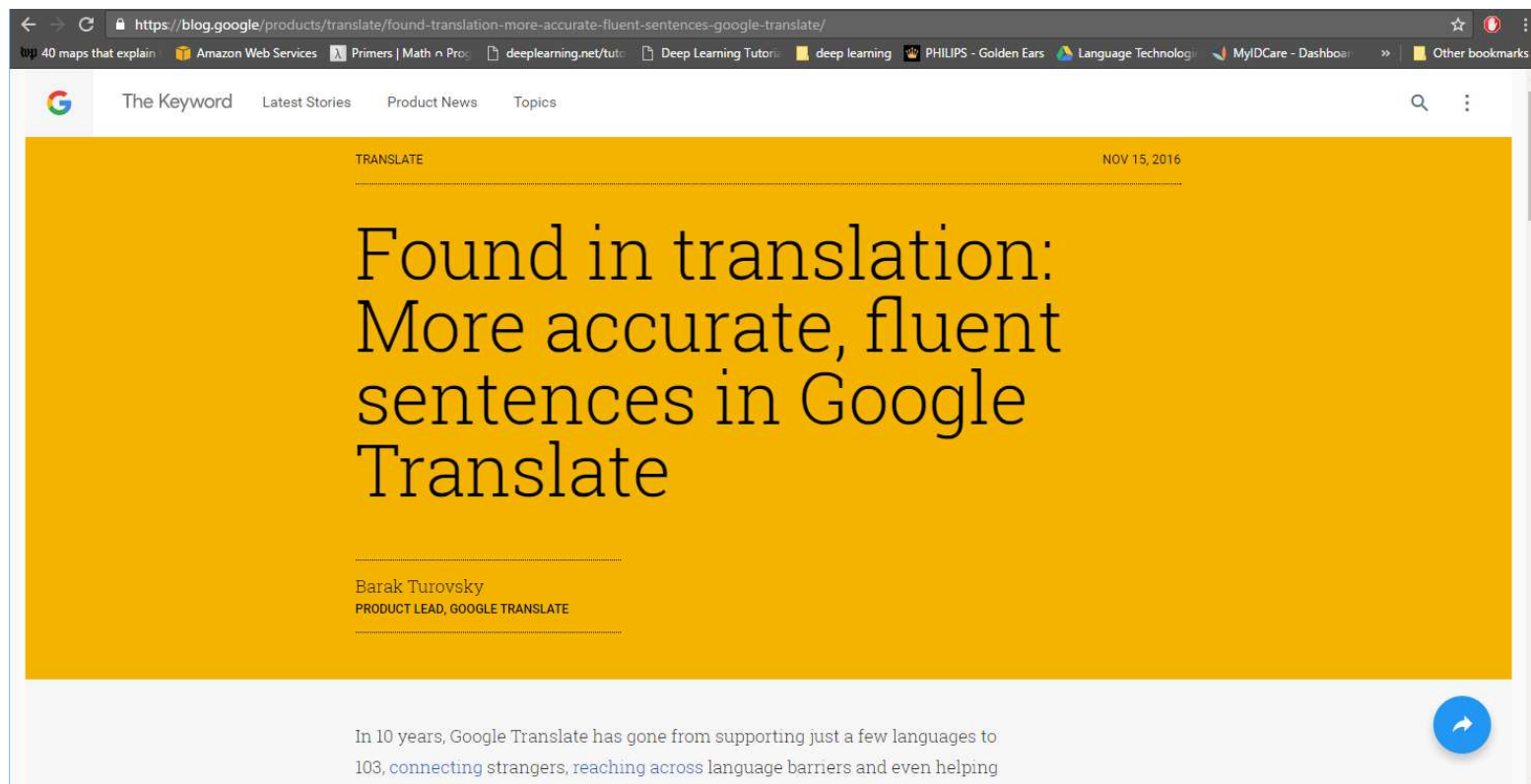
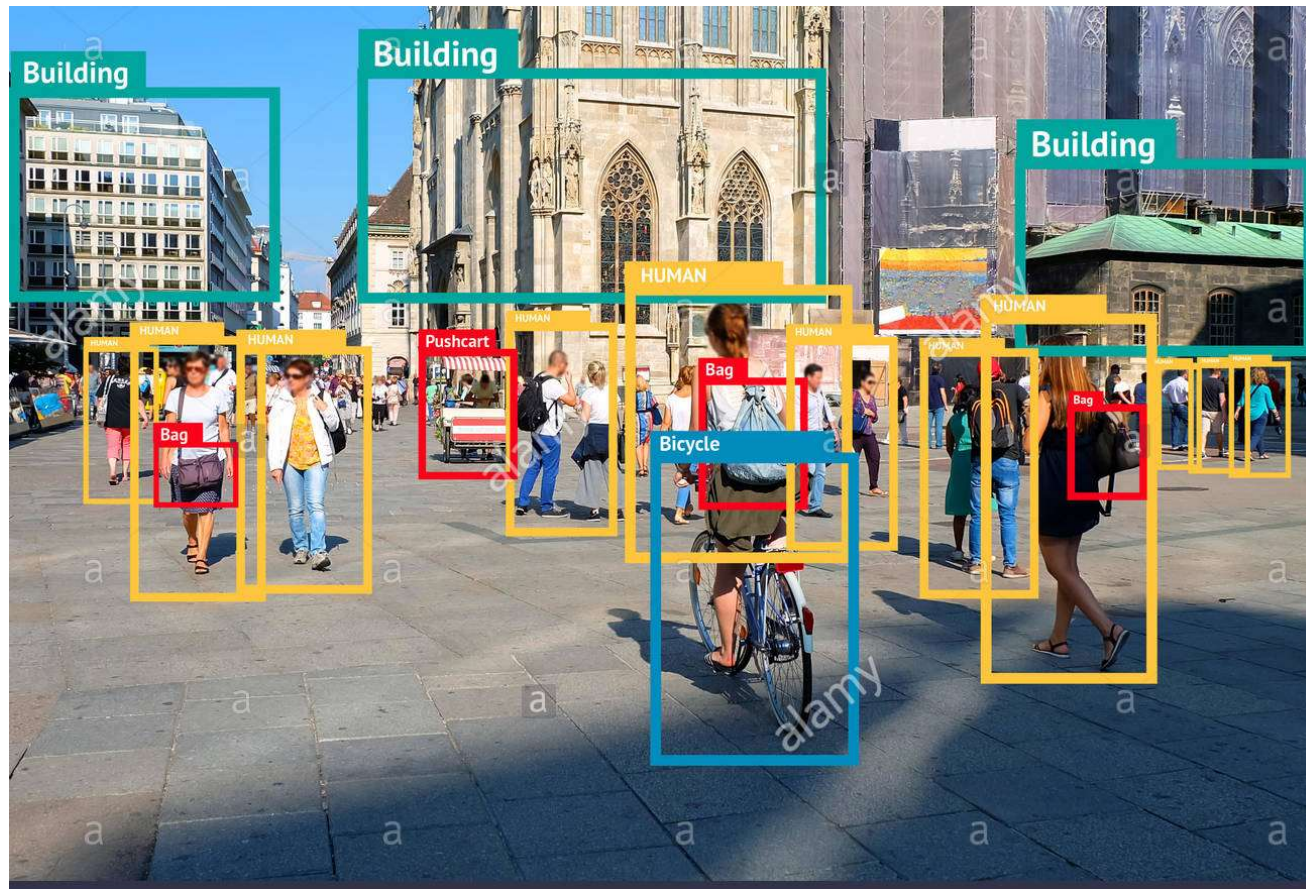


Image segmentation and recognition



Games

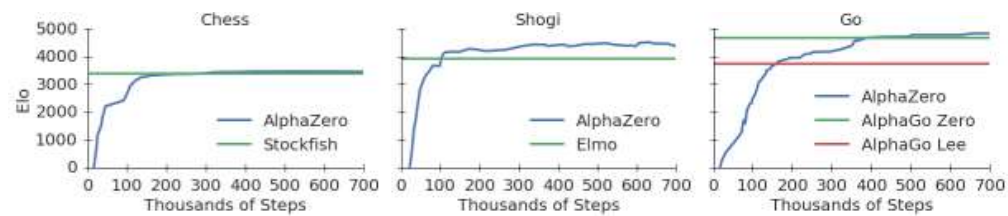
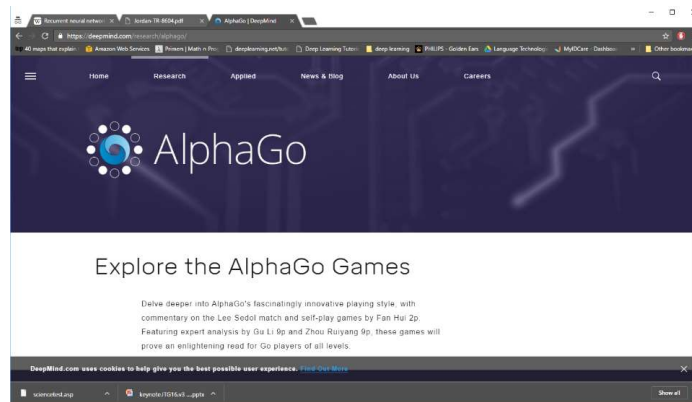


Figure 1: Training *AlphaZero* for 700,000 steps. Elo ratings were computed from evaluation games between different players when given one second per move. **a** Performance of *AlphaZero* in chess, compared to 2016 TCEC world-champion program *Stockfish*. **b** Performance of *AlphaZero* in shogi, compared to 2017 CSA world-champion program *Elmo*. **c** Performance of *AlphaZero* in Go, compared to *AlphaGo Lee* and *AlphaGo Zero* (20 block / 3 day) (29).

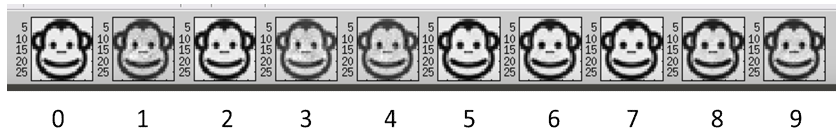
The future is bright!



- AI can solve everything!!!

But..

- Unfortunately, they can be fooled
 - By “**adversarial**” instances that have been subtly modified for misclassification
 - Raising security concerns
 - Hackers can make systems that we increasingly trust misbehave



Free Xanax, low cost HGH

Sound is drop. Line whether soft oxygen. Cross burn make suggest, minute. Cover part reason. Why fresh wire. Notice, are fact find hold. Move such light city, feet. Near hot, pick other busy, book.

This tutorial

- Defining adversarial inputs
- How to generate adversarial inputs
 - And intuitions on why they happen
- How to generate them specifically for speech
 - With a generalization to general, non-differentiable losses
 - With a brief segue into misappropriating the methods for audio watermarking and audio steganography
- Explaining why adversarial inputs happen
- And how to defend against them

Presenters



- Bhiksha Raj
- Carnegie Mellon University
- Pittsburgh, PA, USA
- bhiksha@cs.cmu.edu



- Joseph Keshet
- Bar Ilan University
- Ramat Gan (Tel Aviv), Israel
- jkeshet@cs.biu.ac.il



- Raphael Franck Olivier
- Carnegie Mellon University
- Pittsburgh, PA, USA
- rolivier@cs.cmu.edu

This tutorial

- Defining adversarial inputs
- How to generate adversarial inputs
 - And intuitions on why they happen
- How to generate them specifically for speech
 - With a generalization to general, non-differentiable losses
 - With a brief segue into misappropriating the methods for audio watermarking and audio steganography
- Explaining why adversarial inputs happen
- And how to defend against them



Spam filtering in 2000

- Spam filters are mostly Naïve Bayes classifiers

$$\frac{\prod_i P(X_i|spam)}{\prod_i P(X_i|notspam)} > \theta? \quad \begin{array}{l} \text{Yes} \Rightarrow \text{Spam} \\ \text{No} \Rightarrow \text{not Spam} \end{array}$$

- X_i s are “features” derived from the message
- Typically words or word patterns
 - E.g. CRM114
- Alternately, maximum-entropy classifiers

$$P(spam|X) = \frac{1}{Z} e^{c + \sum_i \lambda_i f_i(X)}$$
$$P(spam|X) > \theta? \quad \begin{array}{l} \text{Yes} \Rightarrow \text{Spam} \\ \text{No} \Rightarrow \text{not Spam} \end{array}$$

- Bayesian classifier based on max-ent models for the distributions

The “goodword” attack

Free Xanax, Low cost HGH

Sound is drop. Line whether soft oxygen. Cross burn make suggest, minute. Cover part reason. Why fresh wire. Notice, are fact find hold. Move such light city, feet. Near hot, pick other busy, book.

- Introducing a set of words and word patterns that are much more frequent in good email than spam will fool the naïve Bayes filter
- E.g. D. Lowd, C. Meek, *Good word attacks on statistical spam filters*, 2nd Conf. Email and Anti-Spam (CEAS), Mountain View, CA, USA, 2005

Goodword attack: White box

Free Xanax, Low cost HGH

Sound is drop. Line whether soft oxygen. Cross burn make suggest, minute. Cover part reason. Why fresh wire. Notice, are fact find hold. Move such light city, feet. Near hot, pick other busy, book.

$$\frac{\prod_i P(X_i|spam)}{\prod_i P(X_i|notspam)} > \theta?$$

- When both features X_i and probabilities $P(X_i|spam)$ and $P(X_i|notspam)$ are known
- Select features for which $\frac{P(X_i|spam)}{P(X_i|notspam)}$ is high

Goodword attack: Grey box

Free Xanax, Low cost HGH

Sound is drop. Line whether soft oxygen. Cross burn make suggest, minute. Cover part reason. Why fresh wire. Notice, are fact find hold. Move such light city, feet. Near hot, pick other busy, book.

$$\frac{\prod_i P(X_i|spam)}{\prod_i P(X_i|notspam)} > \theta?$$

- When only features X_i are known and probabilities are not known
- Select features X_i which are much more common in benign email than in spam

Goodword attack: Black box

Free Xanax, Low cost HGH

Sound is drop. Line whether soft oxygen. Cross burn make suggest, minute. Cover part reason. Why fresh wire. Notice, are fact find hold. Move such light city, feet. Near hot, pick other busy, book.

- When neither features X_i nor the probabilities are known
- Find word patterns that are much more common in benign email than in spam
 - And hope they are the ones in the classifier
 - Needs guesswork
 - But list of “useful” features can be refined based on the behavior of the filter

A more formal approach

- Naïve Bayes and max-entropy classifiers are linear classifiers

$$\frac{\prod_i P(X_i|spam)}{\prod_i P(X_i|notspam)} > \theta?$$

$$\Rightarrow \sum_i (\log P(X_i|spam) - \log P(X_i|notspam)) - \log \theta > 0?$$

- Letting $\log P(X_i|spam) - \log P(X_i|notspam) = f(X_i)$, this can be rewritten as

$$\sum_i f(X_i) - b > 0?$$

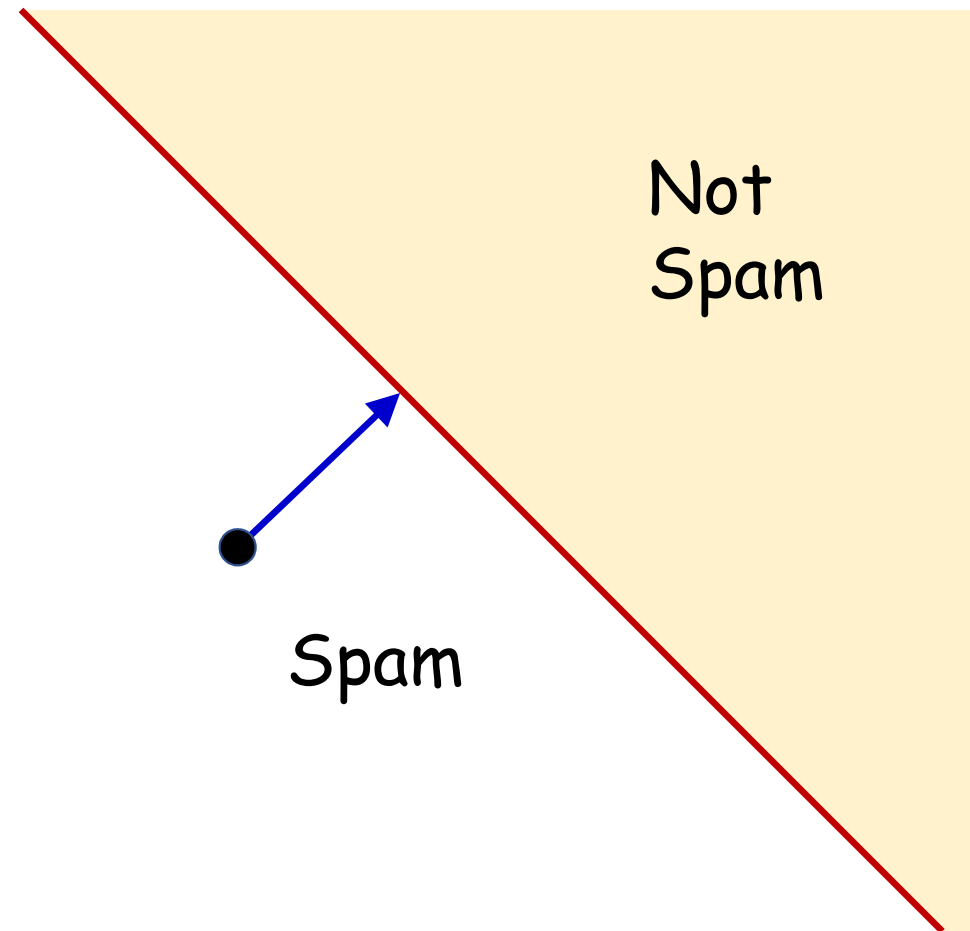
- Or more generally as

$$\sum_i \lambda_i f(X_i) - b > 0?$$

- Which is the *maximum entropy* classifier
- These are simple linear classifiers

“Adversifying” an instance for a linear classifier

- To minimally modify an instance to change the classifier output, must only move it to the closest location on the boundary (and ϵ beyond)
 - This will change the classifier output
 - Not restricted to Spam filters, holds for *any* linear classifier

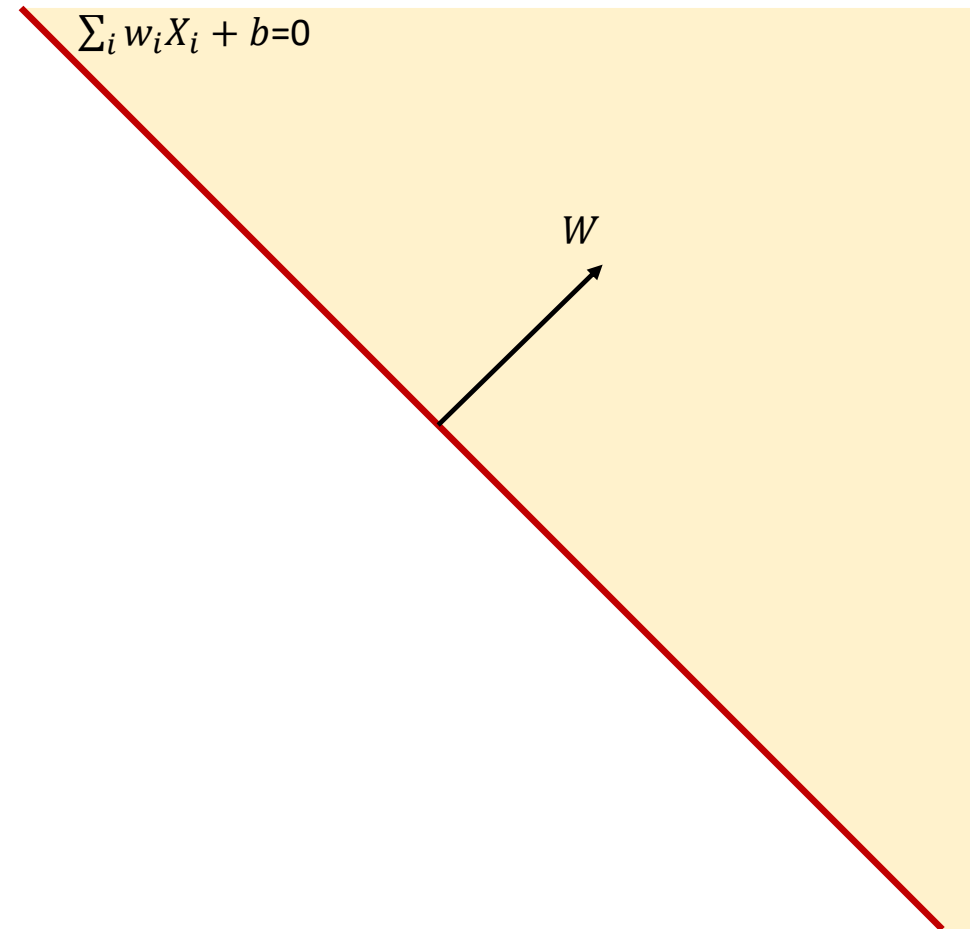


“Adversifying” an instance for a linear classifier

- The discriminant function for a linear classifier

$$f(X) = \sum_i w_i X_i + b$$

- The decision boundary is $f(X) = 0$
 - The vector $W = [w_1 \ w_2 \ \dots]$ is normal to it



“Adversifying” an instance for a linear classifier

- The discriminant function for a linear classifier

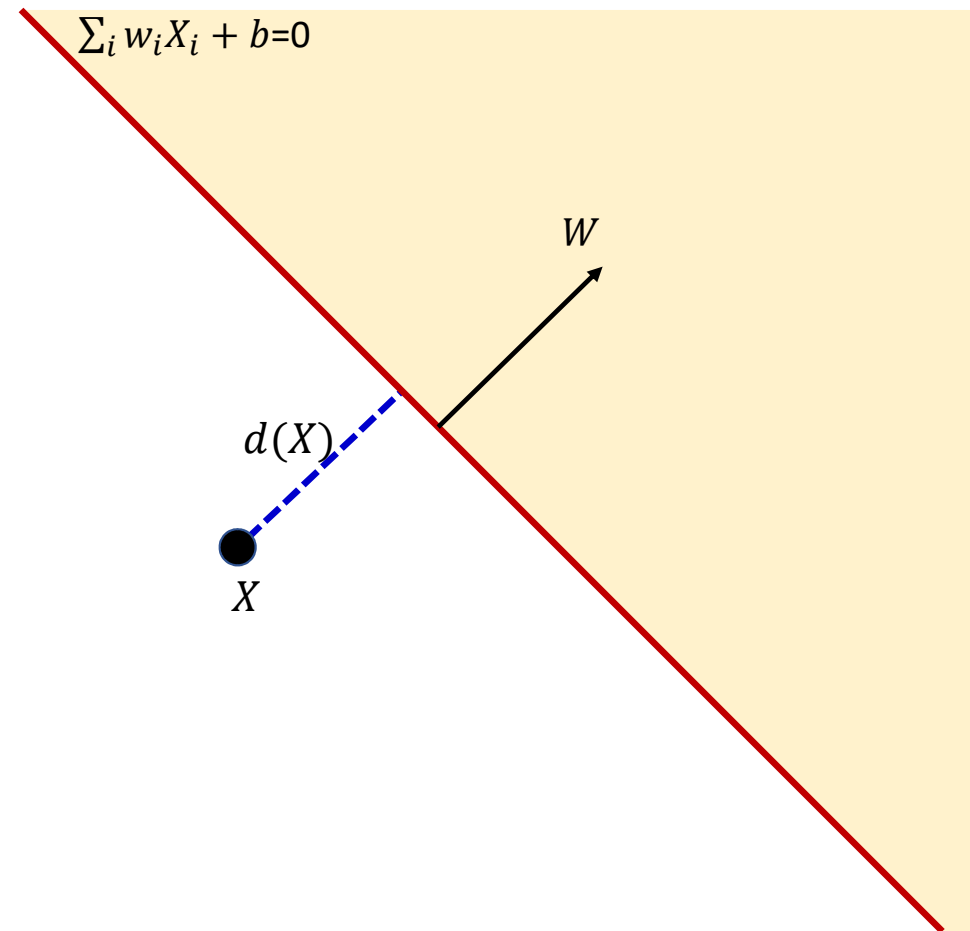
$$f(X) = \sum_i w_i X_i + b$$

- The decision boundary is $f(X) = 0$
 - The vector $W = [w_1 \ w_2 \ \dots]$ is normal to it

- The distance of any X from the boundary:

$$d(X) = \frac{f(X)}{\|W\|}$$

- This is a signed number



“Adversifying” an instance for a linear classifier

- The discriminant function for a linear classifier

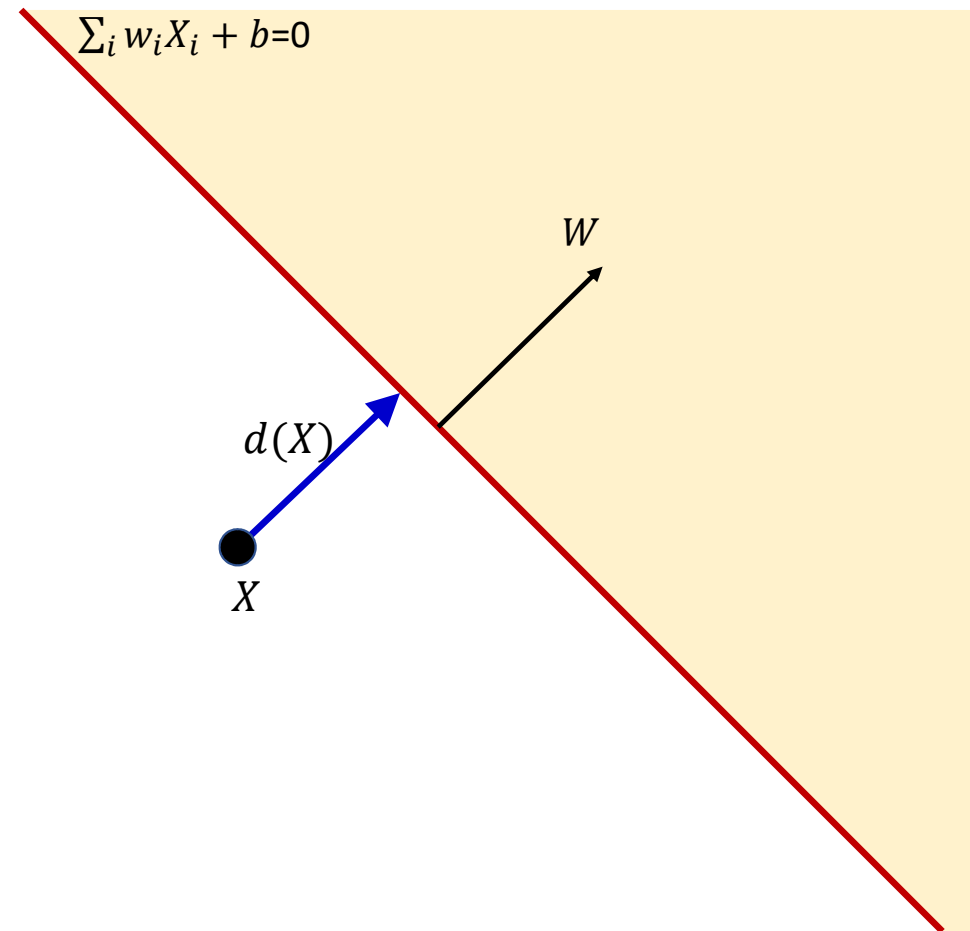
$$f(X) = \sum_i w_i X_i + b$$

- The decision boundary is $f(X) = 0$
 - The vector $W = [w_1 \ w_2 \ \dots]$ is normal to it

- The distance of any X from the boundary:

$$d(X) = \frac{f(X)}{\|W\|}$$

- This is a signed number
- To shift X to the boundary
$$X \rightarrow X - d(X) \frac{W}{\|W\|}$$
 - Move it by distance $-d(X)$ in the direction of W



“Adversifying” an instance for a linear classifier

- The discriminant function for a linear classifier

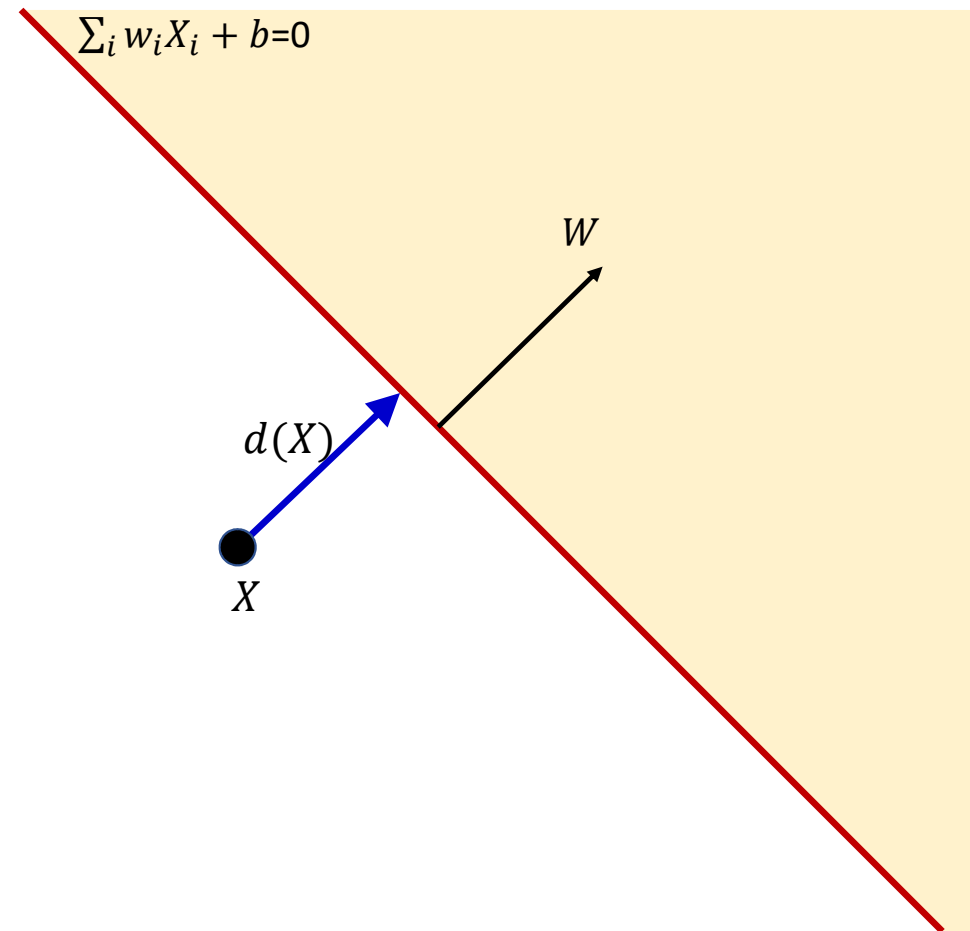
$$f(X) = \sum_i w_i X_i + b$$

- The decision boundary is $f(X) = 0$
 - The vector $W = [w_1 \ w_2 \ \dots]$ is normal to it

- The distance of any X from the boundary:

$$d(X) = \frac{f(X)}{\|W\|}$$

- This is a signed number
- To shift X to the boundary
$$X \rightarrow X - d(X) \frac{W}{\|W\|^2} = X - \frac{f(X)}{\|W\|^2} W$$
 - Move it by distance $-d(X)$ in the direction of W



“Adversifying” an instance for a linear classifier

- The discriminant function for a linear classifier

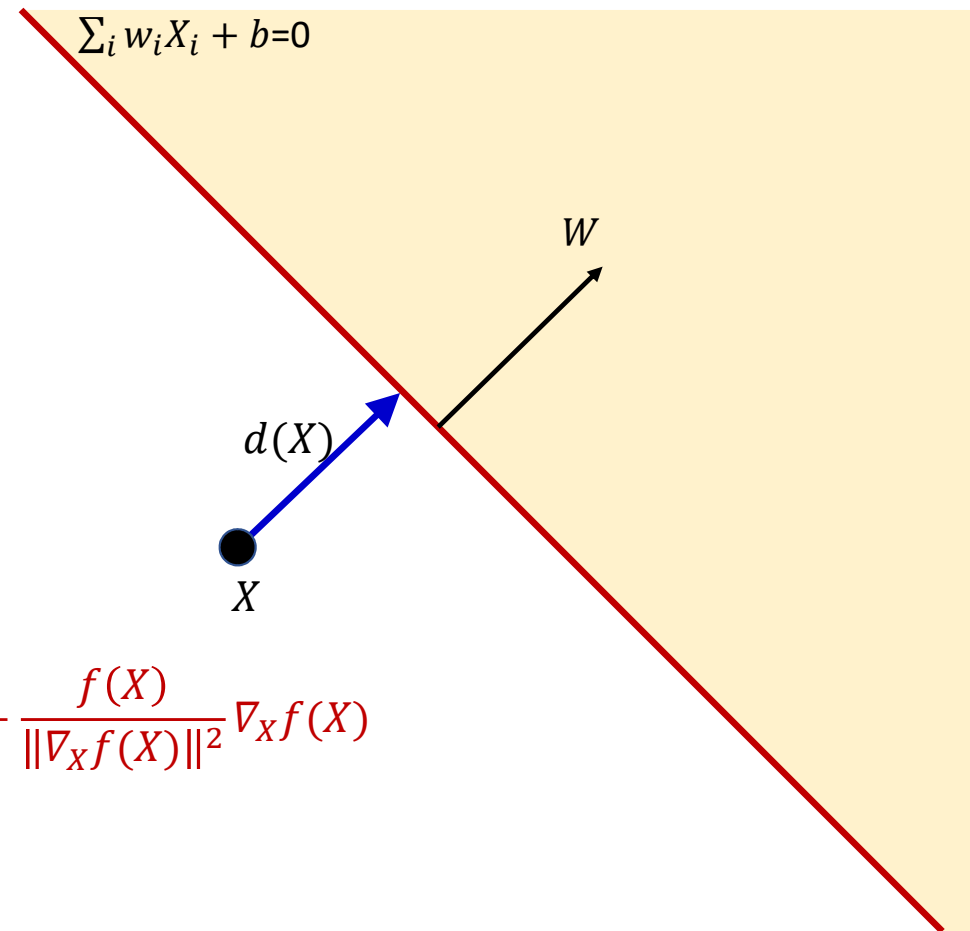
$$f(X) = \sum_i w_i X_i + b$$

- The decision boundary is $f(X) = 0$
 - The vector $W = [w_1 \ w_2 \ \dots]$ is normal to it

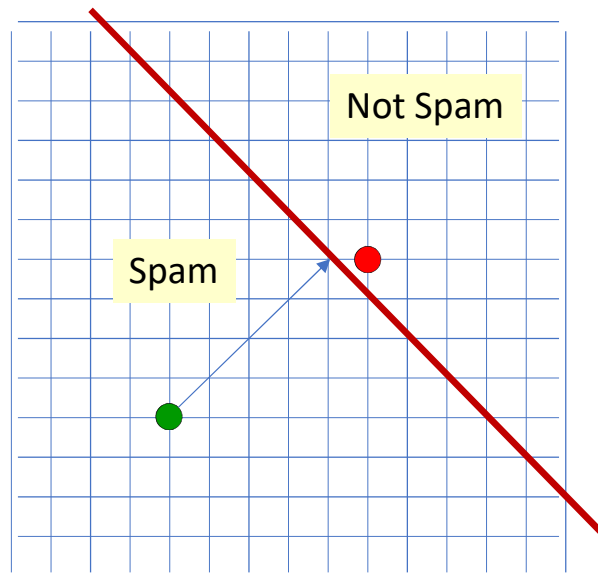
- The distance of any X from the boundary:

$$d(X) = \frac{f(X)}{\|W\|}$$

- This is a signed number
- To shift X to the boundary
$$X \rightarrow X - d(X) \frac{W}{\|W\|} = X - \frac{f(X)}{\|W\|^2} W = X - \frac{f(X)}{\|\nabla_X f(X)\|^2} \nabla_X f(X)$$
 - Move it by distance $-d(X)$ in the direction of W

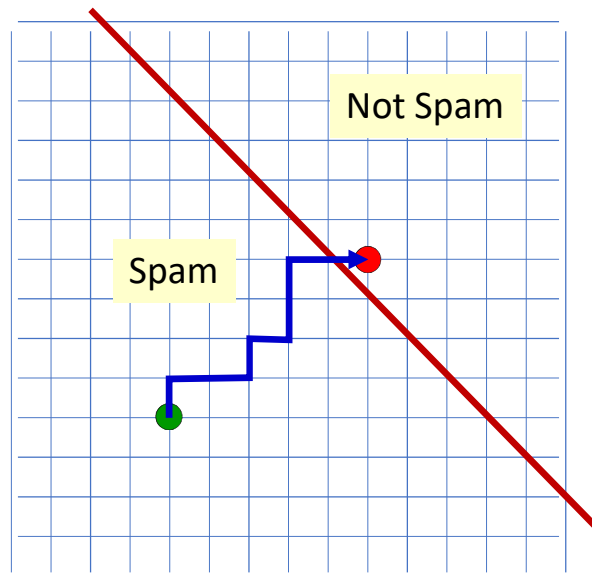


Beating linear classifier spam filters



- To fool a spam filter, a spam message must only be moved to the closest point on the boundary
- However, we cannot make continuous-valued edits to spam: word edits are discrete phenomena
 - The actual modifications can only lie on the corners of a grid
 - The closest valid non-spam point will be on a grid location close to the boundary
 - Need to consider this while finding the point

Beating linear classifier spam filters



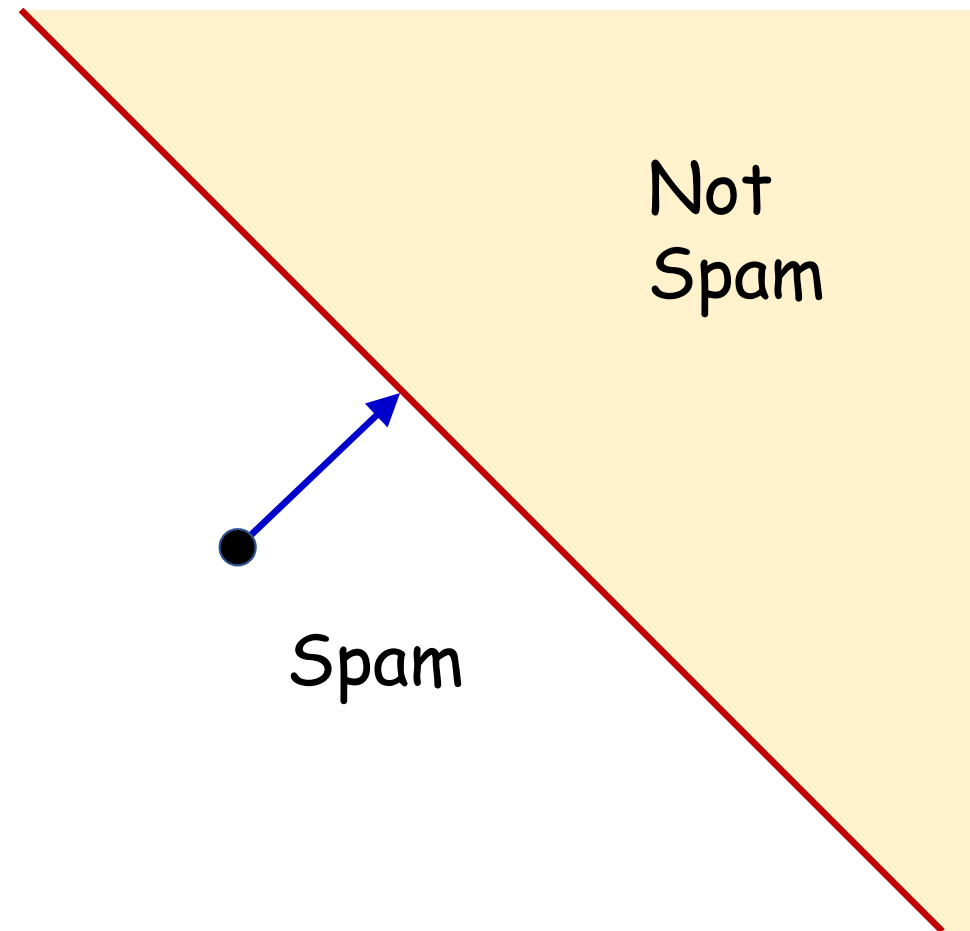
- N. Dalvi, P. Domingos, Mausam, S. Sanghai, D. Verma, *Adversarial classification*. International Conference on Knowledge Discovery and Data Mining, 2004
- Integer programming algorithm to determine minimal edits to “convert” a spam to not spam

“Adversifying” an instance for a linear classifier

- The principle of moving the instance to the closest point on the boundary applies to any linear classifier

$$X \rightarrow X - \frac{f(X)}{||W||^2} W$$

- Perceptrons
- Logistic regression
- SVM



And even to kernel SVMs

- Which are non-linear classifiers, but are actually linear in the Kernel space

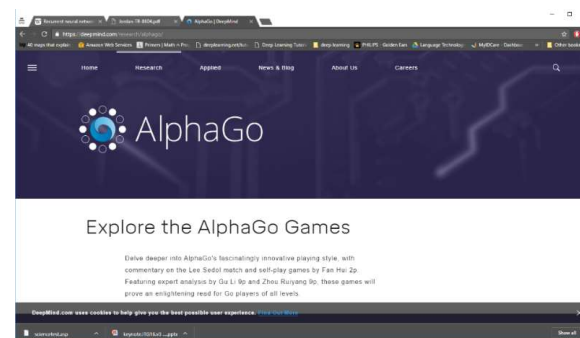
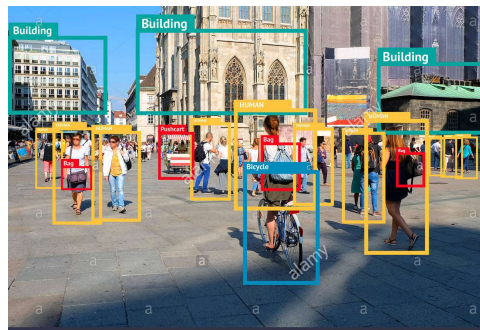
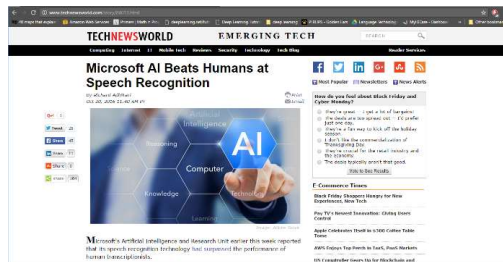
$$f(X) = \sum_i c_i y_i K(X, S_i) - b$$

- S_i are the support vectors
- The adversifying correction
$$X \rightarrow X + \hat{\varepsilon}; \quad \hat{\varepsilon} = \underset{\varepsilon}{\operatorname{argmin}} ||\varepsilon||^2 \quad s.t. \operatorname{sign}(f(X))f(X + \varepsilon) < 0$$
 - Note: we're minimizing the correction length in the original space, not the Kernel space
 - The shortest distance to the boundary in Kernel space may map to large changes to input data
- Must be solved with iterative methods

2004-present: adversarial attacks on simple linear and non-linear classifiers

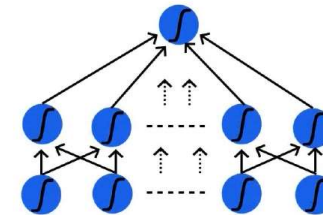
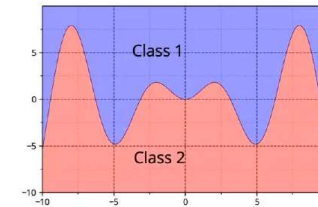
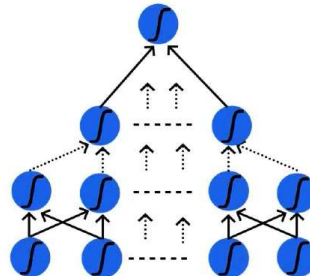
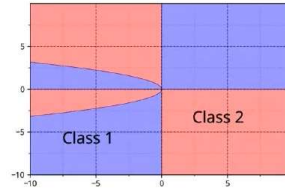
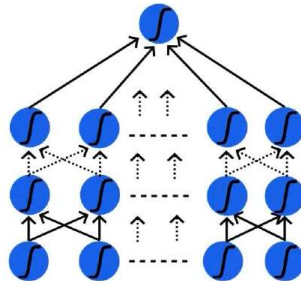
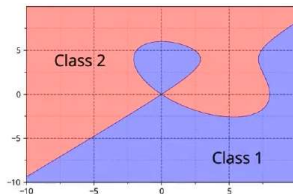
- On perceptrons
- On logistic regressions and softmax
- On SVMs
 - And SVMs with non-linear kernels
- Even on how to “poison” training data to make learned SVMs misbehave
 - And how to defend
- Biggio, Battista, and Fabio Roli. *Wild patterns: Ten years after the rise of adversarial machine learning*. Pattern Recognition 84 (2018)

2010s.. Neural networks rule..



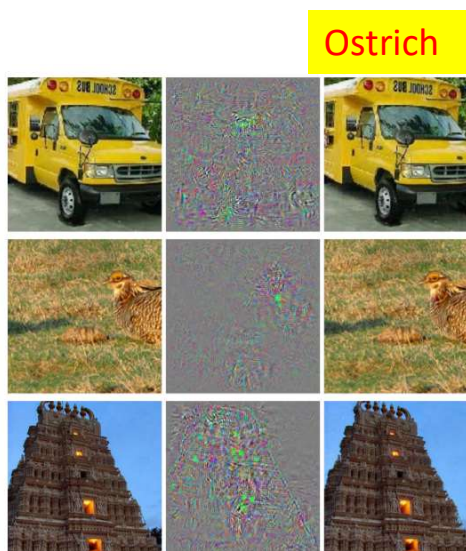
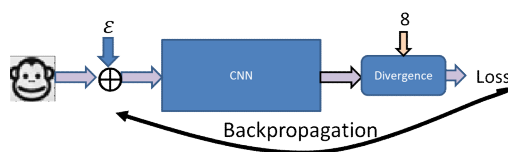
- Neural network systems have established the state of the art in many many tasks...

Nnets are universal approximators



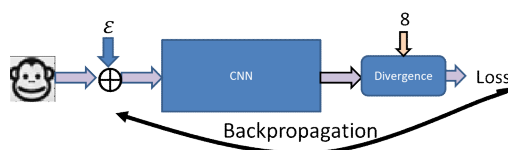
- Can approximate anything!
 - Any Boolean function
 - Any classification boundary
 - Any real-valued function
 - To arbitrary precision
- Surely they're more robust than simple naïve classifiers?

Szegedy et al. *Intriguing properties of neural networks.* ICLR 2014

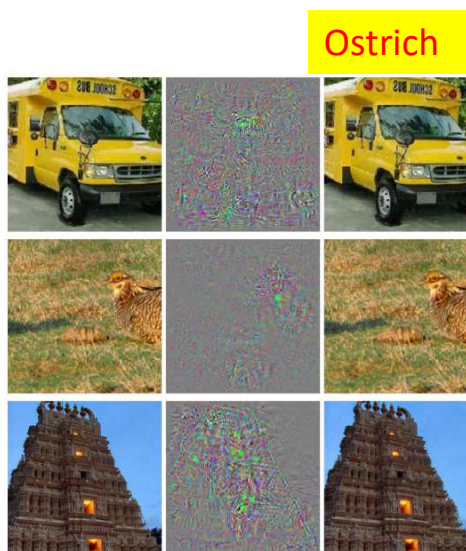


- Adding often imperceptible noise to images can result in targeted misclassification
- Noise that is computed as follows will cause images to be misclassified:
$$\hat{n} = \underset{n}{\operatorname{argmin}} \lambda |n| + L(f(x + n; \theta), y_{false})$$
 - Subject to $(x + n) \in [0,1]^m$ (noisified images stay in valid range of pixel values)
- Basically “O”s method

Szegedy et al. *Intriguing properties of neural networks.* ICLR 2014



Keep the noise small



Ostrich

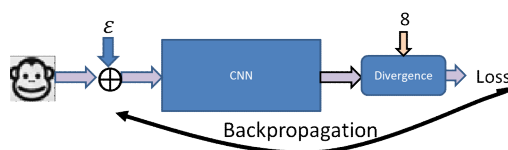
Minimize the error (loss) between the actual output of the net $f(x + n; \theta)$ and the desired bogus output y_{false} (ostrich in this case)

- Adding often imperceptible noise to images can result in targeted misclassification
- Noise that is computed as follows will cause images to be misclassified:

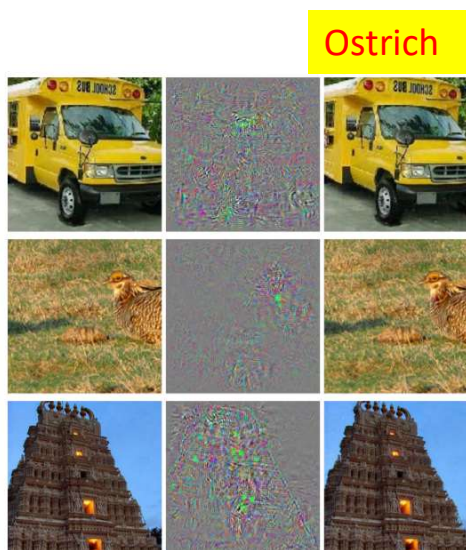
$$\hat{n} = \underset{n}{\operatorname{argmin}} \lambda |n| + L(f(x + n; \theta), y_{false})$$

- Subject to $(x + n) \in [0,1]^m$ (noisified images stay in valid range of pixel values)
- Basically “O”s method

Szegedy et al. *Intriguing properties of neural networks.* ICLR 2014



Keep the noise small



Ostrich

Minimize the error (loss) between the actual output of the net $f(x + n; \theta)$ and the desired bogus output y_{false} (ostrich in this case)

- Adding often imperceptible noise to images can result in targeted misclassification
- Noise that is computed as follows will cause images to be misclassified:

$$\hat{n} = \underset{n}{\operatorname{argmin}} \lambda |n| + L(f(x + n; \theta), y_{false})$$

- Subject to $(x + n) \in [0,1]^m$ (noisified images stay in valid range of pixel values)
- Basically “O”s method

In no case should the sum of the image and the noise exceed valid pixel values

The optimization

$$\hat{n} = \operatorname{argmin}_n \lambda |n| + L(f(x + n; \theta), y_{false})$$

- Subject to $(x + n) \in [0,1]^m$ (noisified images stay in valid range of pixel values)
- Actual algorithm used in paper: L-BGFS, but any iterative constrained optimization algorithm should work
- Typical optimization:

$$n \leftarrow n - \delta \nabla_x L(f(x + n; \theta), y_{false})$$

- Note, this is gradient *descent*
- For initialization $n = 0$ the first iteration is

$$n = -\delta \nabla_x L(f(x; \theta), y_{false})$$

Untargeted adversarial modification

$$\hat{n} = \operatorname{argmin}_n \lambda |n| - L(f(x + n; \theta), y_{true})$$

- Subject to $(x + n) \in [0,1]^m$ (noisified images stay in valid range of pixel values)
- Aim: Just make the model misclassify the input
 - Modify a cat image, so it calls it *not* a cat
 - *Maximize* the error between classifier output and “cat”
- Typical optimization:

$$n \leftarrow n + \delta \nabla_x L(f(x + n; \theta), y_{true})$$

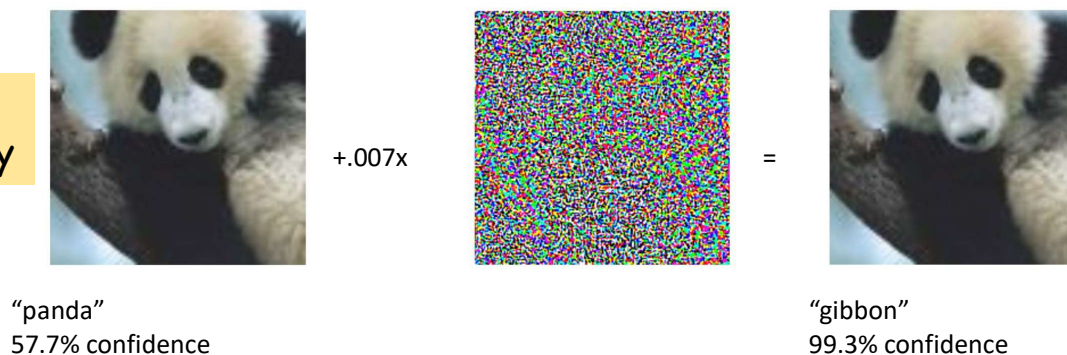
- Note, this is gradient *descent*
- For initialization $n = 0$ the first iteration is

$$n = \delta \nabla_x L(f(x; \theta), y_{true})$$

Goodfellow et al. *Explaining and harnessing adversarial examples*. arXiv:1412.6572 (2014)

Attempts to force classifier to misclassify

Untargeted; has no choice on *what* it is misclassified as



$$\hat{x} = x + \delta \operatorname{sign}(\nabla_x L(f(x; \theta), y_{\text{true}}))$$

- *Fast signed gradient method FSGM*
- Attempt to force misclassification by *maximizing* loss between output of network and *true* output
 - One-step process, without iteration
 - This is the first step of gradient *ascent* when we initialize $n = 0$
- The gradient $\nabla_x L(f(x; \theta), y_{\text{true}})$ is computable using backpropagation

Targeted misclassification: Kurakin et al. *Adversarial machine learning at scale*. *arXiv:1611.01236* (2016)

- *Targeted* FSGM: Have a specific target in mind

$$\hat{x} = x - \delta \operatorname{sign} \left(\nabla_x L(f(x; \theta), y_{\text{target}}) \right)$$

- First step of gradient *descent*
- Full iterative algorithm
 - Uses projected gradients

$$n \leftarrow n - \delta \nabla_x L(f(x + n; \theta), y_{\text{target}})$$

$$n \leftarrow \operatorname{Clip}(n, \varepsilon)$$

- If any component of n is larger than ε , it is clipped to ε
- Ensures *maximum* noise in any pixel is constrained, rather than the overall length of noise vector
- Original paper uses $\varepsilon = 1$ (least significant bit flip in each pixel in each channel)
- Can also be used for *untargeted* attacks: $n \leftarrow \operatorname{Clip}(n + \delta \nabla_x L(f(x + n; \theta), y_{\text{true}}), \varepsilon)$

Many variants to the solution

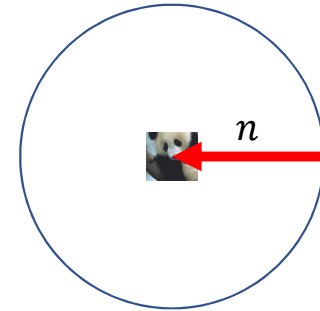
- Generally based on two approaches
- Norm minimization: norm of noise as regularizer

$$\hat{n} = \operatorname{argmin}_n \lambda |n|_p + L(f(x + n; \theta), y)$$

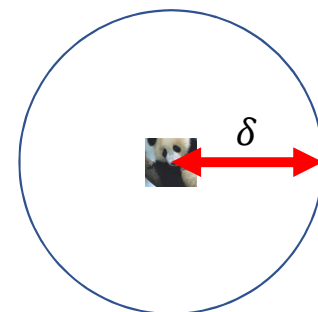
- Norm-constrained minimization: Impose hard constraints on noise while minimizing loss

$$\hat{n} = \operatorname{arg} \min_{n: |n|_p < \delta} L(f(x + n; \theta), y)$$

Shrink noise ball while minimizing loss



Fix radius of noise ball and look for noise within the ball

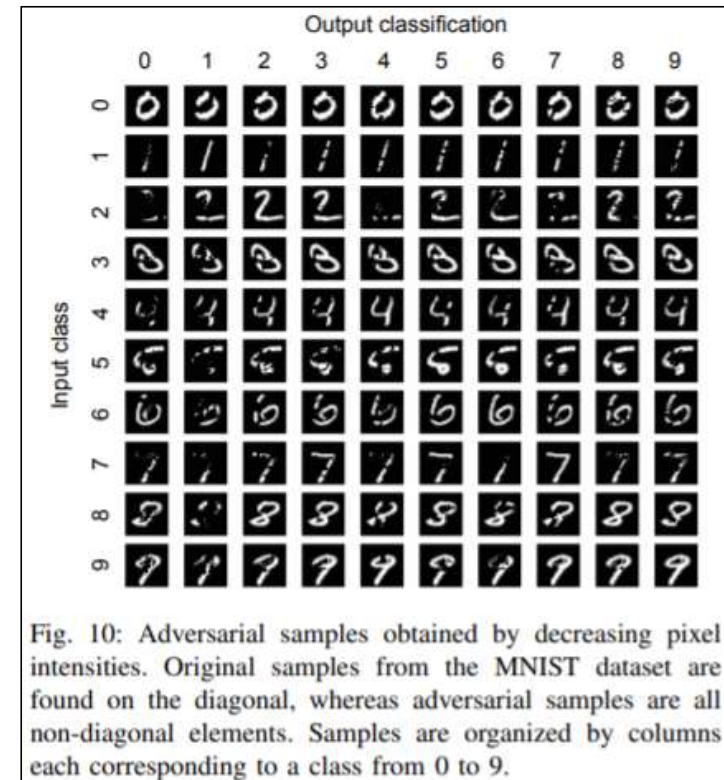


An alternate approach

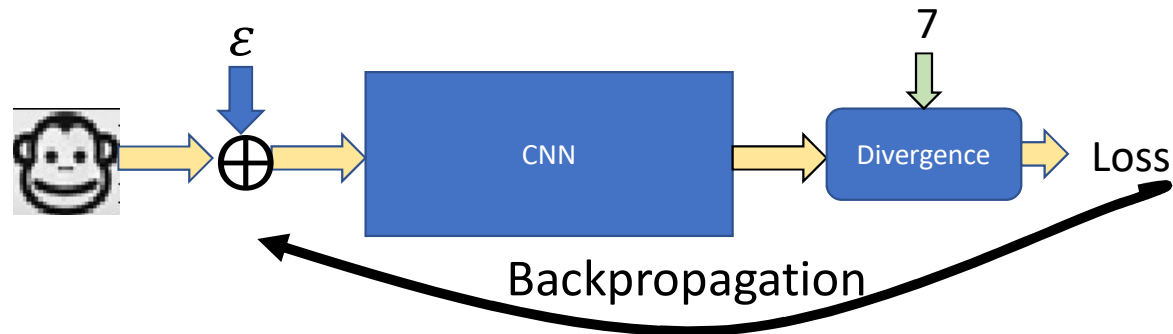
- In the previous algorithms the noise is constrained or minimized while minimizing loss
 - (or maximizing it for untargeted adversariality)
- No guarantee that the generated instance will *actually* be adversarial
 - Due to constraints on noise
- Alternate approach : Ensure that an erroneously classified instance is generated, even if it means noise becomes large

$$\operatorname{argmin}_n |n|_p \quad \text{such that } f(x + n) = y_{\text{target}}$$

- The generated instances may have too much noise
 - Not really “adversarial” – wont fool the human eye
- Papernot et al. *The limitations of deep learning in adversarial settings*. IEEE European Symposium on Security and Privacy. 2016

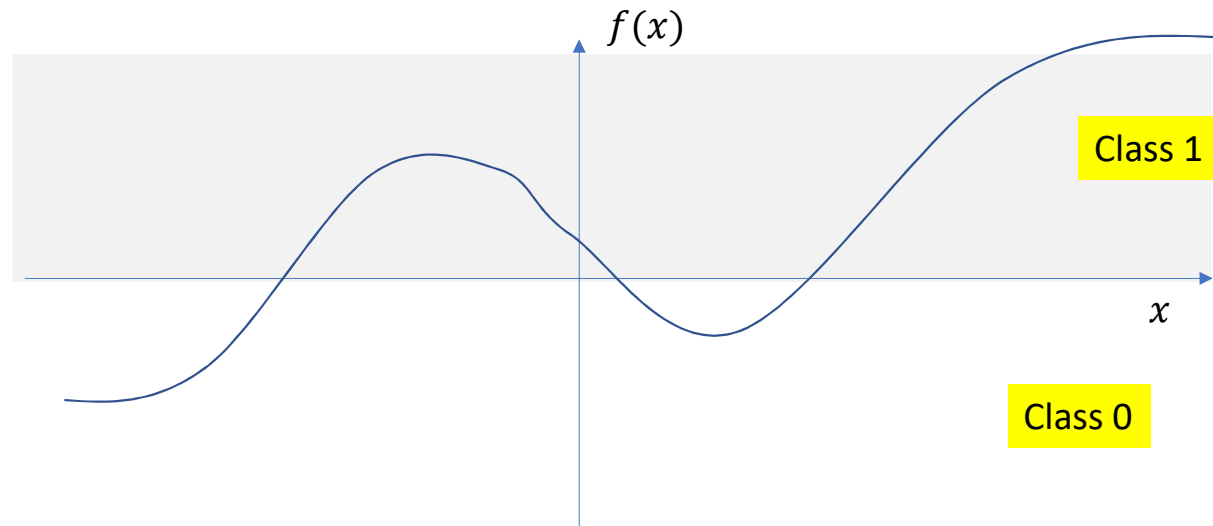


Working directly on the discriminant



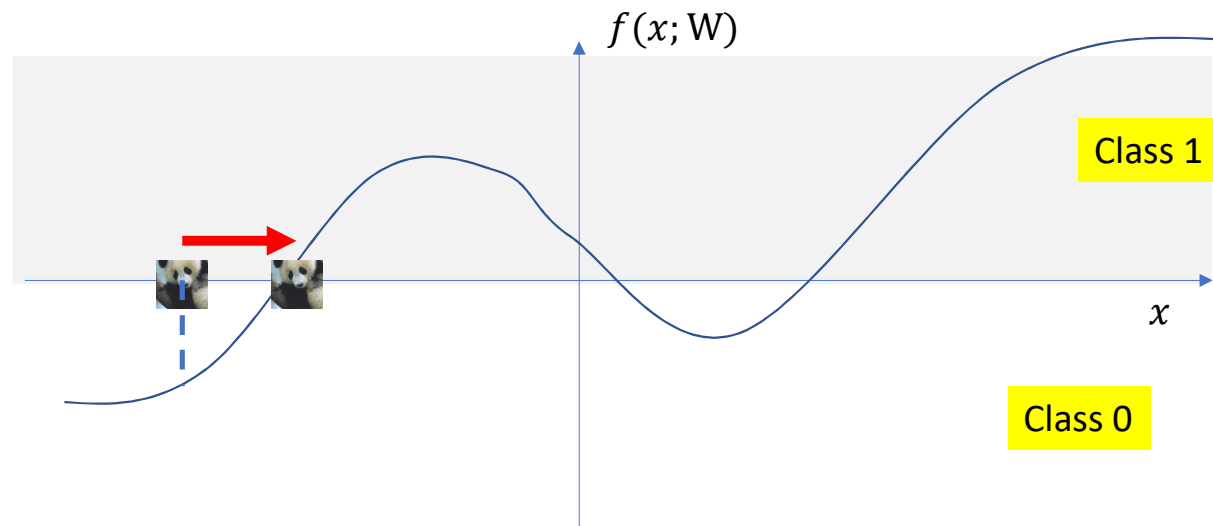
- All previous methods attempted to minimize a loss between a target class and the actual network output
 - Maximize the loss for untargeted attacks
- A much more efficient method can be obtained by directly operating on the discriminant function itself
 - As we did for the linear classifier case
 - 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

Deepfool



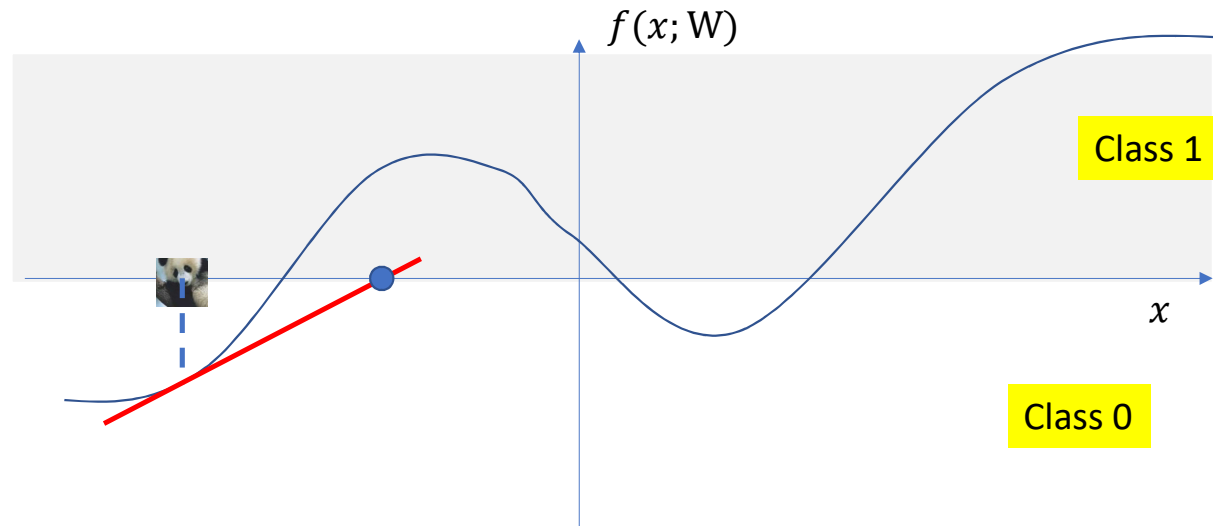
- The network is actually a discriminator $f(X)$
 - For binary classification, when $f(X) \geq 0$, the input X is classified as one class, when $f(X) < 0$ it is a different class
 - Multi-class classifiers can be viewed as a collection of such discriminators

Deepfool



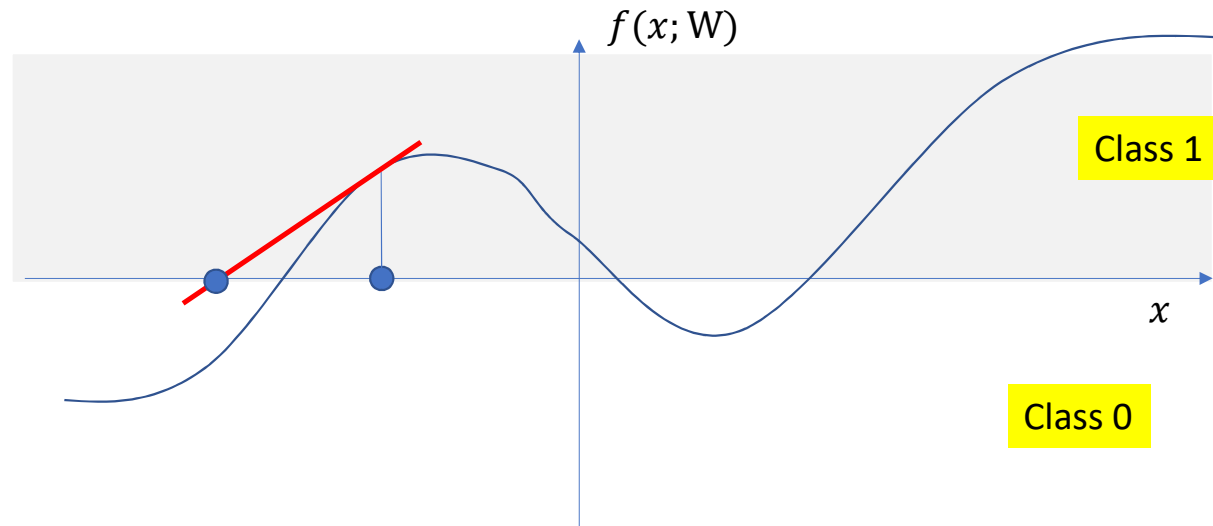
- The network is actually a discriminator $f(X)$
 - For binary classification, when $f(X) \geq 0$, the input X is classified as one class, when $f(X) < 0$ it is a different class
 - Multi-class classifiers can be viewed as a collection of such discriminators
- To change the classification output for an input, shift it by the minimum amount so that $f(X)$ changes sign

Deepfool



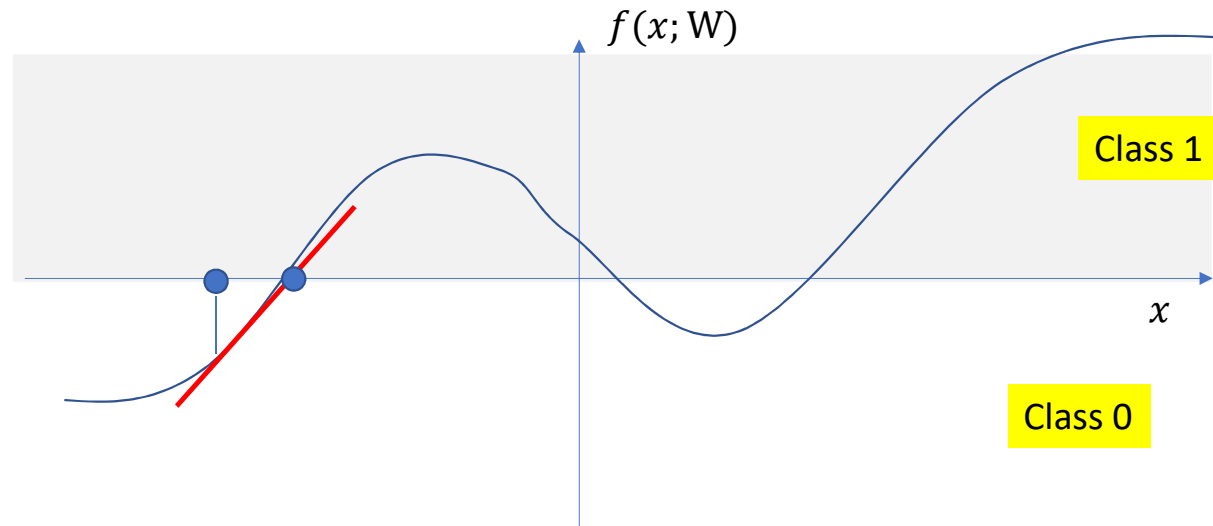
- Iteratively linearize the function and find location of 0
 - Until a location where the actual $f(X) = 0$ is found

Deepfool



- Iteratively linearize the function and find location of 0
 - Until a location where the actual $f(X) = 0$ is found

Deepfool



- Iteratively linearize the function and find location of 0
 - Until a location where the actual $f(X) = 0$ is found

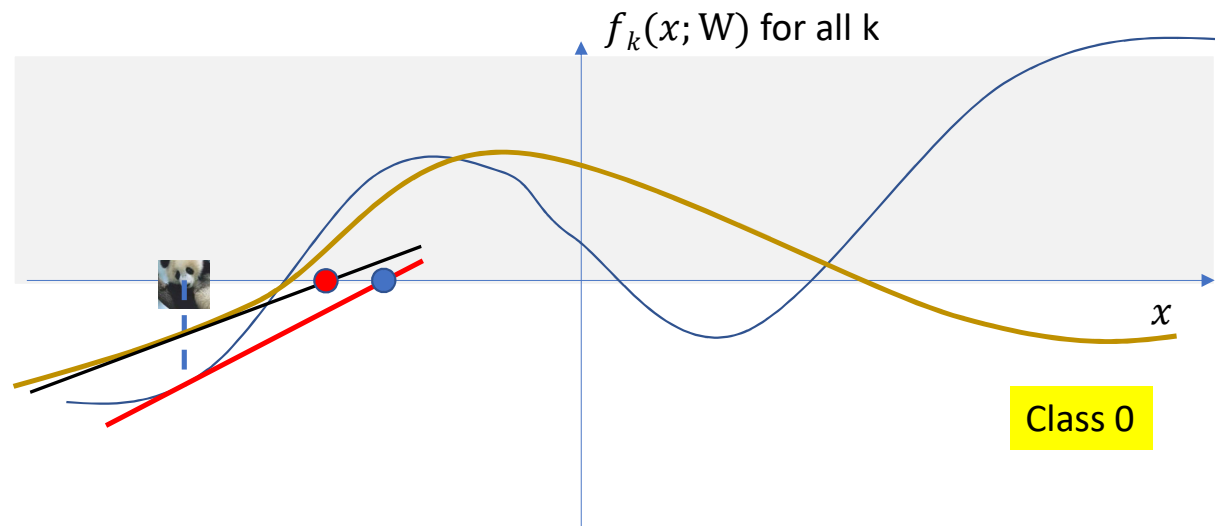
Deepfool, binary case

Algorithm 1 DeepFool for binary classifiers

```
1: input: Image  $\mathbf{x}$ , classifier  $f$ .  
2: output: Perturbation  $\hat{\mathbf{r}}$ .  
3: Initialize  $\mathbf{x}_0 \leftarrow \mathbf{x}$ ,  $i \leftarrow 0$ .  
4: while  $\text{sign}(f(\mathbf{x}_i)) = \text{sign}(f(\mathbf{x}_0))$  do  
5:    $\mathbf{r}_i \leftarrow -\frac{f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|_2^2} \nabla f(\mathbf{x}_i)$ ,  
6:    $\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \mathbf{r}_i$ ,  
7:    $i \leftarrow i + 1$ .  
8: end while  
9: return  $\hat{\mathbf{r}} = \sum_i \mathbf{r}_i$ .
```

From Moosavi-Dezfooli et al.

Deepfool: Multi class



- Comparison of the current class to each of the other $K-1$ classes results in $K-1$ discriminators
- At each step simultaneously optimize for each of the $K-1$ discriminators
 - Choose the smallest of the corrections

Deepfool, multi-class case

Algorithm 2 DeepFool: multi-class case

```
1: input: Image  $\mathbf{x}$ , classifier  $f$ .
2: output: Perturbation  $\hat{\mathbf{r}}$ .
3:
4: Initialize  $\mathbf{x}_0 \leftarrow \mathbf{x}, i \leftarrow 0$ .
5: while  $\hat{k}(\mathbf{x}_i) = \hat{k}(\mathbf{x}_0)$  do
6:   for  $k \neq \hat{k}(\mathbf{x}_0)$  do
7:      $\mathbf{w}'_k \leftarrow \nabla f_k(\mathbf{x}_i) - \nabla f_{\hat{k}(\mathbf{x}_0)}(\mathbf{x}_i)$ 
8:      $f'_k \leftarrow f_k(\mathbf{x}_i) - f_{\hat{k}(\mathbf{x}_0)}(\mathbf{x}_i)$ 
9:   end for
10:   $\hat{l} \leftarrow \arg \min_{k \neq \hat{k}(\mathbf{x}_0)} \frac{|f'_k|}{\|\mathbf{w}'_k\|_2}$ 
11:   $\mathbf{r}_i \leftarrow \frac{|f'_l|}{\|\mathbf{w}'_l\|_2^2} \mathbf{w}'_l$ 
12:   $\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \mathbf{r}_i$ 
13:   $i \leftarrow i + 1$ 
14: end while
15: return  $\hat{\mathbf{r}} = \sum_i \mathbf{r}_i$ 
```

From Moosavi-Dezfooli et al.

Techniques are increasingly sophisticated

- And increasingly efficient!
- And versatile

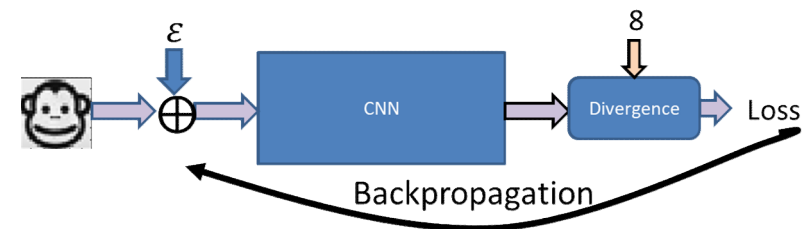
You can simply have an instance misclassified...

Aim: Modify cat image so that its *not* classified as cat

$$\hat{n} = \arg \max_{n: |n|_p < \delta} L(f(x + n; \theta), y_{true})$$



Cat

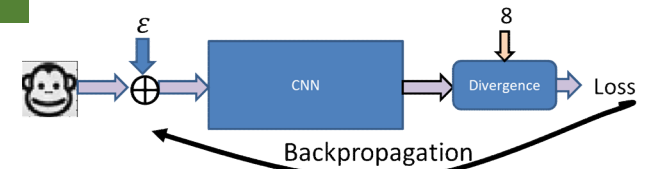


- To just misclassify an input, find noise to *maximize* the error between the network output and the *true* label

Or even choose what it is misclassified as

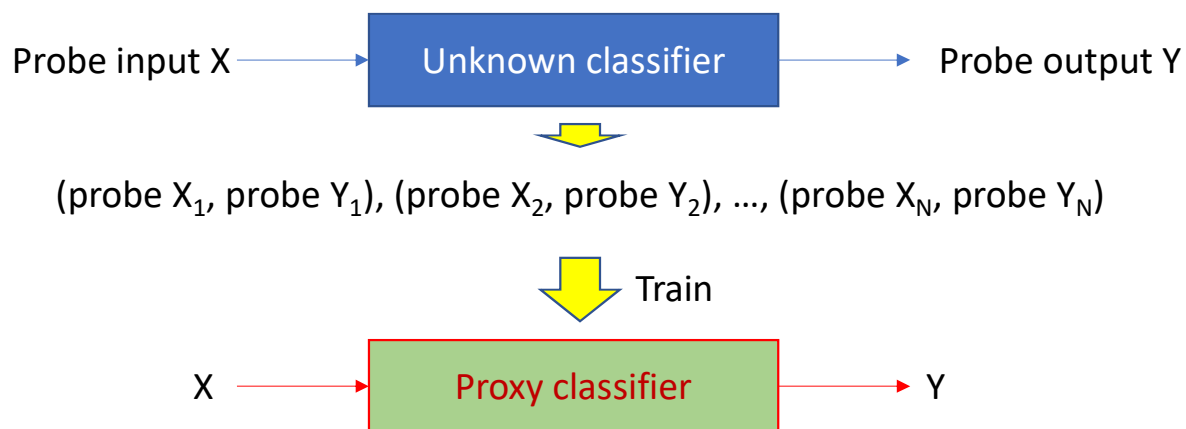
Aim: Specifically modify cat image so that it is classified as a bottle

$$\hat{n} = \arg \min_{n: |n|_p < \delta} L(f(x + n; \theta), \text{bottle})$$



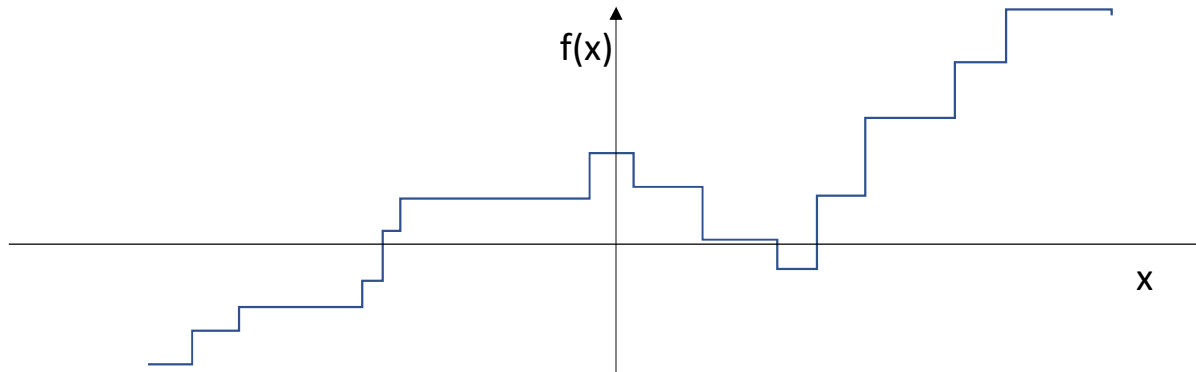
- Find noise to *minimize* the error between the network output and the *desired bogus label*

You don't even need to know the classifier



- Just probe the unknown classifier to obtain input-output pairs
- Train a proxy classifier with the probe data
- Use the proxy classifier to build your adversarial inputs
 - They will transfer to the original classifier!
 - More on this later

This provides an unexpected benefit to adversaries

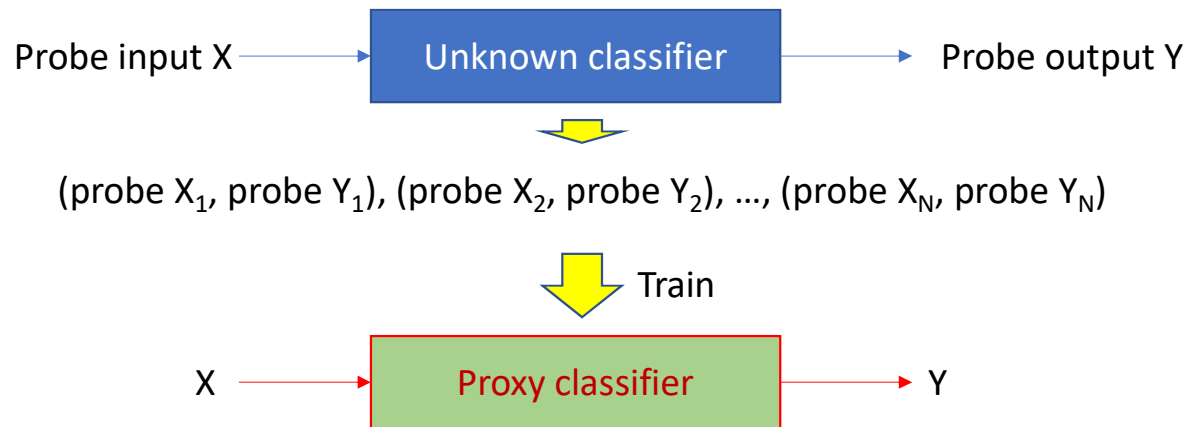


- Training adversarial noise requires differentiation

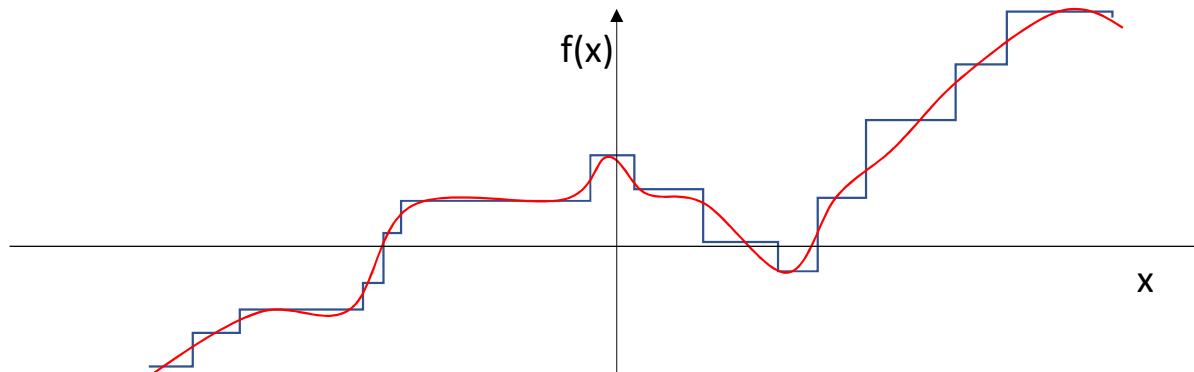
$$\hat{n} = \operatorname{argmin}_n \lambda |n| + L(f(x + n; \theta), y_{target})$$

- Gradient descent estimators require the differentiation of $L(\cdot)$ and $f(\cdot)$
- $f(\cdot)$ is the network
- What if it is designed to not be differentiable?
 - A separate question: What if $L(\cdot)$ is not differentiable? Yossi will deal with this

This provides an unexpected benefit to adversaries



- The adversary can simply train a differentiable proxy and use that instead



But these are only artificial, right?



- Synthetic examples, where you add noise to pre-recorded images
 - Using significant computation in each case
- Doesn't carry over to real-life where you will generally not have the ability to carefully manipulate an image with iterative algorithms

Even if you print them out and show them to a camera

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%

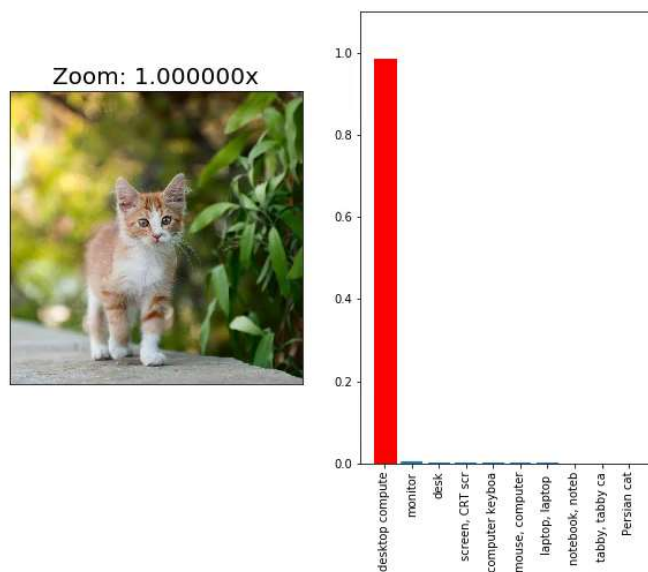
Fraction of adversarial instances that fail to be adversarial to a camera

Prefiltered: instances that are originally classified with high confidence, whose adversarial variants are also misclassified with high confidence

- Doesn't really work that well unless the noise ϵ is very high
 - Kurakin, Alexey, Ian Goodfellow, and Samy Bengio. "Adversarial examples in the physical world." *arXiv preprint arXiv:1607.02533* (2016).

The problem

- Adversarial noise is very specific to the input image
- Even minor zoom (as little as 1.002), minor changes in angle or perspective, or any other transformation destroys the adversariality of the noise



From: <https://openai.com/blog/robust-adversarial-inputs/>
(by Anish Athalye)

The solution

- Train an ensemble of classifiers, that operate on transformed and zoomed images
 - A random selection of transforms and zooms
 - The more, the merrier
- Actual objective optimized (maximized):
$$\mathbb{E}_{t \sim T} \log P(y_{target} | t(x + n)) - \lambda \mathbb{E}_{t \sim T} d(t(x + n), t(x))$$
 - T is the set of Transforms
 - Expectation is simply averaging over the sampled transforms
 - $d(t(x + n), t(x))$ is a divergence measure, typically L2
- Athalye, Anish, et al. *Synthesizing robust adversarial examples*. *arXiv:1707.07397* (2017).

The solution

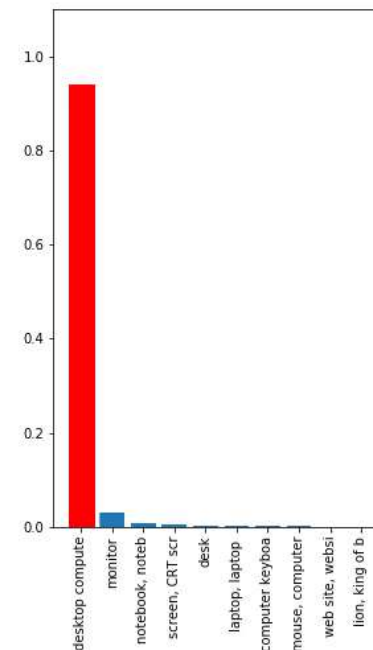
- Train an ensemble of classifiers on transformed images
 - A random selection of transforms
 - The more, the merrier

- Actual objective optimized (maximize)

$$\mathbb{E}_{t \sim T} \log P(y_{\text{target}} | t(x))$$

- T is the set of Transforms
- Expectation is simply averaging
- $d(t(x + n), t(x))$ is a divergence

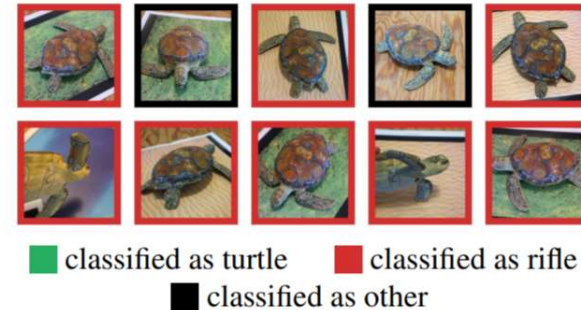
- Athalye, Anish, et al. *Synthesizing robust adversarial examples*. *arXiv:1707.07397* (2017).



ed

Real-world adversaries

- Approach also extends to 3D objects
 - Pictures of actual 3D objects
 - Color patterns of 3D models of objects are optimized such that images taken from various perspectives and zooms are all adversarially classified
 - Finally, 3D print the object



- Athalye, Anish, et al. (2017).

Figure 1. Randomly sampled poses of a 3D-printed turtle adversarially perturbed to classify as a rifle at every viewpoint². An unperturbed model is classified correctly as a turtle nearly 100% of the time.

Still, real-life scenarios are not really affected

- One must either generate robust pictures
 - Time consuming
 - Wont affect real-life applications like computer vision in a self-driving car
- Or construct entire 3-D objects
 - Which must be optimized
- What about simple perturbation of existing objects?

Real world attacks



Sharif, Mahmood, et al. *Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition*. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*.

- Noise is restricted to eyeglass-frame region of face image
 - Eyeglasses are superimposed on face image
 - Different positions and adjustments are included
- Loss includes “printability” and “smoothness” regularizers
 - To ensure pleasant, printable patterns
- The learned noise is printed on actual eyeglasses

Can we do this in less constrained situations?

- Face recognition systems require fairly standardized presentation of the input
 - Constrained positioning and registration of image
 - Makes the generation of adversarial instances simpler
- Can we do this in less constrained situations?

Real-world attacks



Eykholt et al., *Robust Physical-World Attacks on Deep Learning Visual Classification*, CVPR 2018

- Apply a mask to actual real world objects and take many pictures from various angles etc
- Learn to simultaneously misclassify all instances, such that only the masked regions are affected by (identical) noise in each case
 - Noise is a color pattern that is uniform within each sub-region of the mask, to avoid having to learn to geometrically transform the noise patterns
- Add printability regularization and regularization term to account for natural variation in how cameras record color
- Print actual masks of the learned colors and apply them to the physical object

These only work on images though..



- Are attacks limited to images?
 - Attacking simple binary or multi-class classifiers
- Will it work on harder tasks like speech recognition
 - With effectively infinite classes
- Yossi will address this in his portion of this tutorial

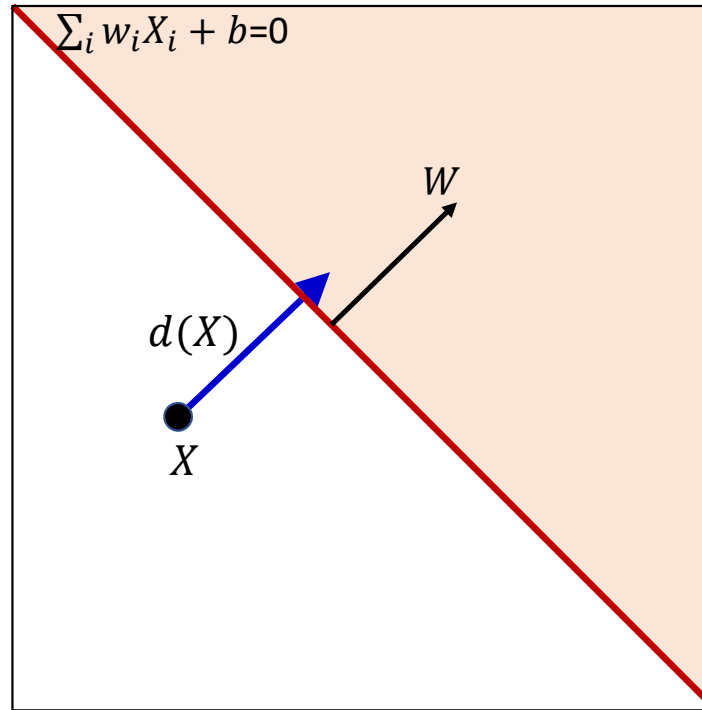
Why does this happen

- So why are classifiers so fragile
 - Get them to do any odd thing
- And why are attacks transferrable?
 - Learn to fool one classifier and have it carry to another

Why does this happen

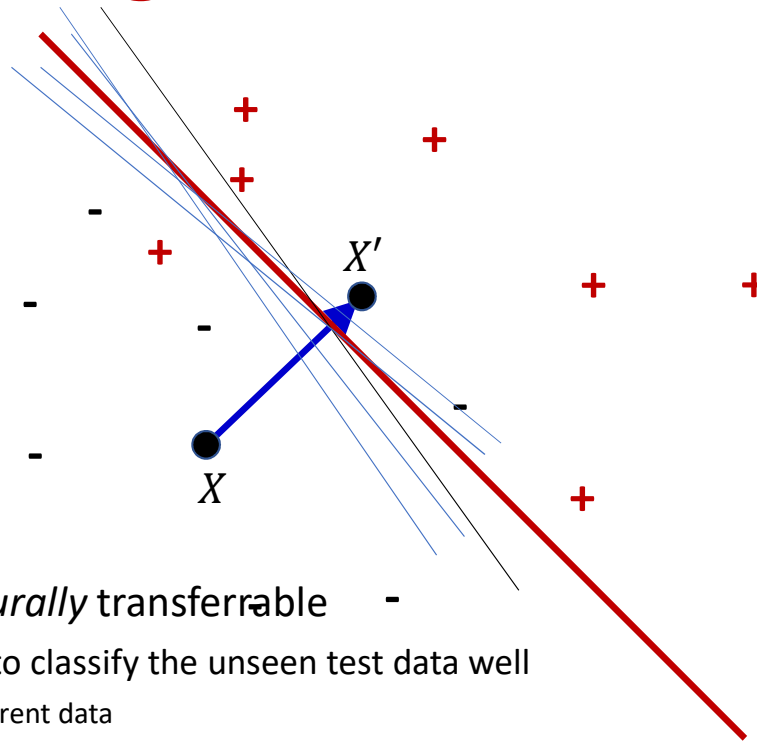
- So why are classifiers so fragile
 - Get them to do any odd thing
- And why are attacks transferrable?
 - Learn to fool one classifier and have it carry to another

Why are attacks against linear classifiers transferrable



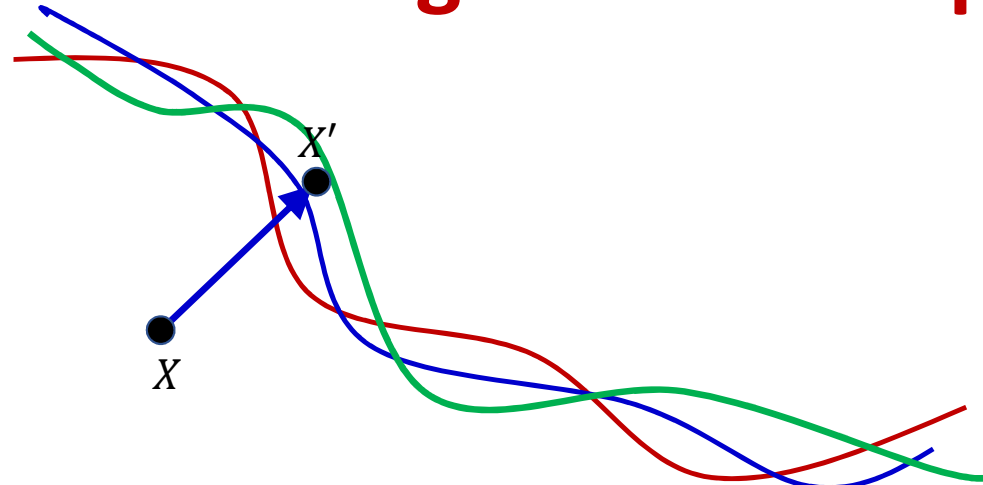
- An adversarial attack moves an instance past the closest location on the decision boundary

Why are attacks against linear classifiers transferrable



- Well trained classifiers are *naturally* transferrable
 - After all, they are all expected to classify the unseen test data well
 - Even if they are trained on different data
- Near-optimal classifiers for the model class with similar performances have very similar decision boundaries *in dense regions*
 - They will classify any instance in this region similarly with high probability
 - Including adversarial instances
- Crossing one boundary near a data dense region will cross most boundaries

Well-trained classifiers have similar boundaries in dense regions of the space

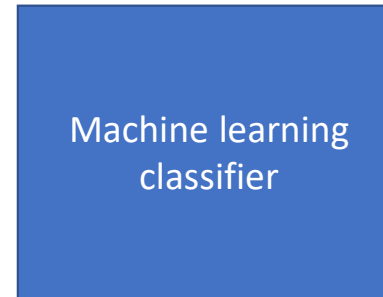
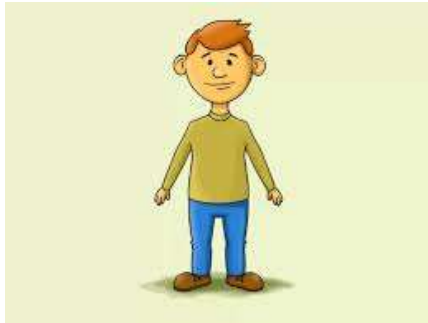


- Well trained classifiers are *naturally* transferrable
 - After all, they are all expected to classify the unseen test data well
 - Even if they are trained on different data
- Near-optimal classifiers for the model class with similar performances have very similar decision boundaries *in dense regions*
 - They will classify any instance in this region similarly with high probability
 - Including adversarial instances
- Crossing one boundary near a data dense region will cross most boundaries

So why are classifiers so fragile

- Perceptual reasoning
- Statistical reasoning

Perceptual reasoning



- We're actually working with *two* classifiers
 - Human perception (typically)
 - The ML classifier
- We want to modify the data such that the two classify the data differently

Human perception is very forgiving

On átkíns or the soùth beach diet, try our diet páтч. A new cutting edge, advanced áppétite sùpprèssant, mètabólism bóòster, and ènérgy ènháncer...all in one. The perfect supplémènt to ássist you in lôsíng those extrá pôúnds just in time for sùmmèr

Lèarn the trùth about losing wèight.

All ordérs backéd by our nó risk, monéy back Guarántée!

Shipped Discrèetly.

[Why wait, the solution is now](#)

No further €máils plèáse
<http://thesedealzwontlast.com>

- We *want* to find patterns

Human perception is very forgiving



(a)



(b)



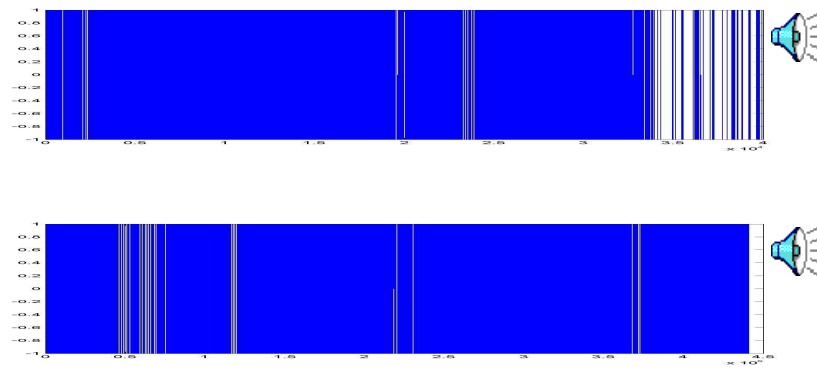
(c)



(d)

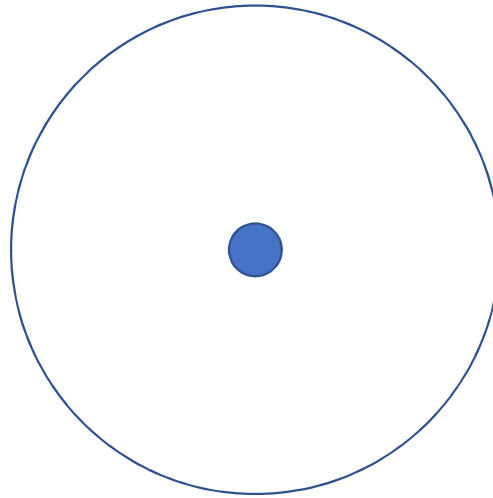
- *We want* to find patterns

Human perception is very forgiving



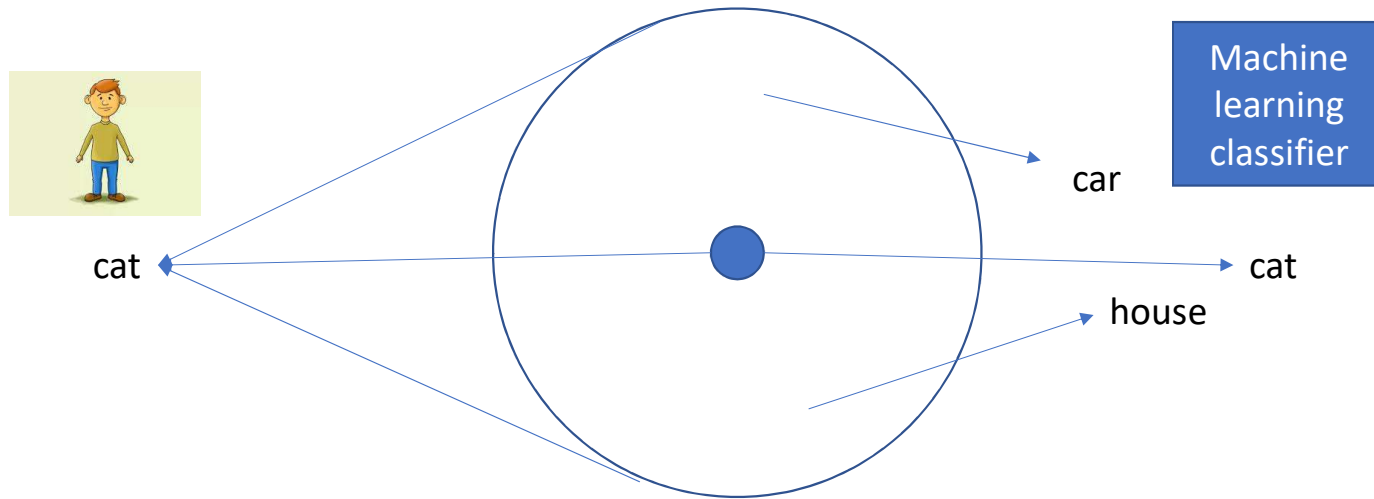
- We *want* to find patterns
 - Tom Sullivan and Schubert, for the curious

The perceptual radius



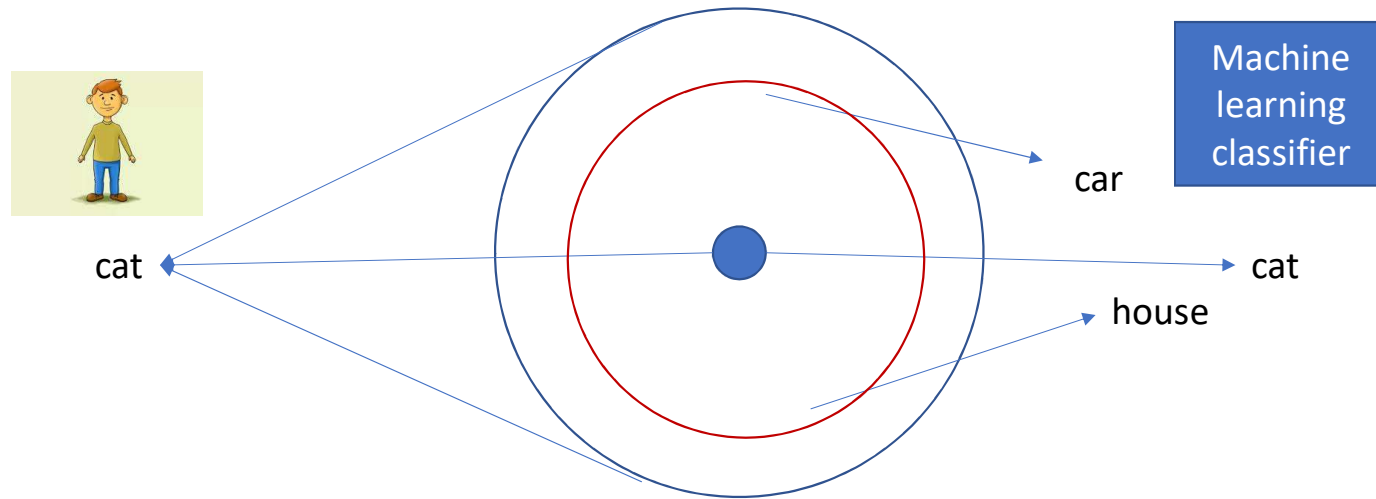
- There is a “ball” of modifications around any valid pattern that we are tolerant to
 - ML algorithms, on the other hand, are sensitive these variations
 - The “perceptual ball” may not even be connected

The perceptual rationale



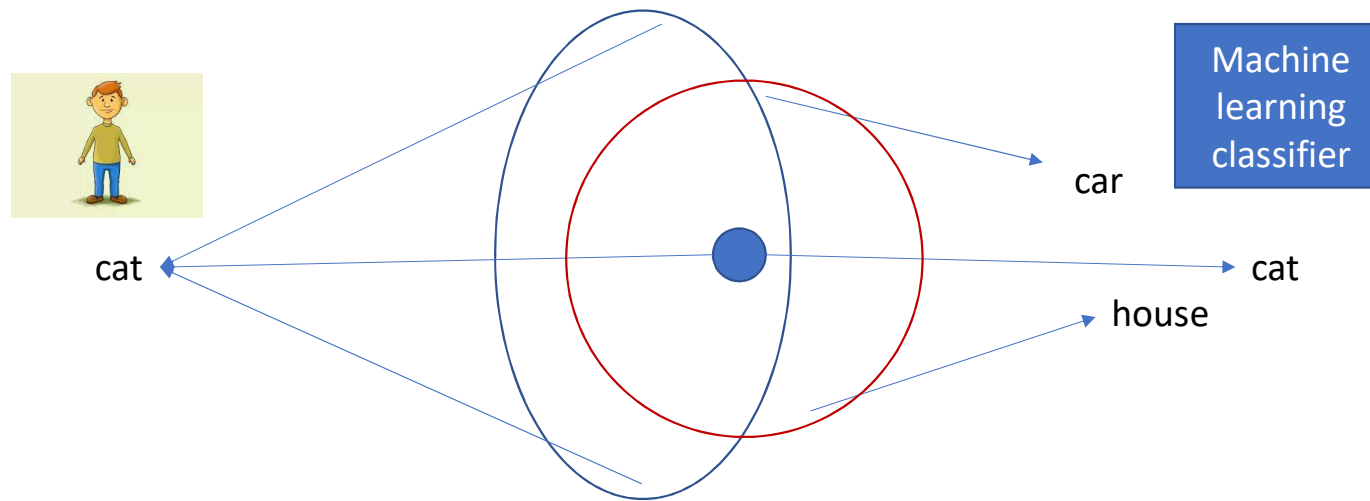
- Machine learning algorithms that are provided training samples, only learn the function at the sample, but not the entire perceptual circle around it
 - Which cannot even be characterized in most cases

The perceptual rationale



- Adversarial attacks search for points within this ball for which the ML algorithm responds differently than we do
 - Since we don't really know the perceptual ball, they model it instead as a metric ball of small radius
 - E.g. $x + n$, $\|n\|_p < \varepsilon$

The perceptual rationale



- When perceptual and metric balls do not match, adversarial attacks can, in fact, fail



Statistical reasoning

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Target function:
 $Y = X_1 \text{ XOR } X_2$

- Consider an ML algorithm that has been provided this training data
 - Trivial to learn
 - Simple XOR

Spurious inputs

X_1	X_2	X_3	Y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	0

Target function:
 $Y = X_1 \text{ XOR } X_2$

- Now the algorithm has been provided this new table instead
 - The target function is still $X_1 \text{ XOR } X_2$
 - X_3 is a *spurious input*

What will the algorithm learn

X_1	X_2	X_3	Y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0
0	0	1	0
1	1	1	0

X_1	X_2	X_3	Y
0	1	1	0
1	0	1	0

X_1	X_2	X_3	Y
0	1	1	0
1	0	1	1

X_1	X_2	X_3	Y
0	1	1	1
1	0	1	0

X_1	X_2	X_3	Y
0	1	1	1
1	0	1	1

- The algorithm can learn any of these patterns for the unseen input combinations
 - Only one is right for our target function
 - If it learns any of the others, the output for some combinations of X_1 and X_2 can be made erroneous by choosing the right X_3

What will the algorithm learn

For $(X_1, X_2) = (0, 1)$
 setting $X_3 = 1$ results in
 adversarial output $Y = 0$

X_1	X_2	X_3	Y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0
0	0	1	0
1	1	1	0

X_1	X_2	X_3	Y
0	1	1	0
1	0	1	0

X_1	X_2	X_3	Y
0	1	1	0
1	0	1	1

X_1	X_2	X_3	Y
0	1	1	1
1	0	1	0

X_1	X_2	X_3	Y
0	1	1	1
1	0	1	1

- The algorithm can learn any of these patterns for the unseen input combinations
 - Only one is right for our target function
 - If it learns any of the others, the output for some combinations of X_1 and X_2 can be made erroneous by choosing the right X_3

What will the algorithm learn

Each additional spurious bit of input adds a superexponential number of ways for the algorithm to learn the wrong thing

This makes it foolable

X_1	X_2	X_3	Y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0
0	0	1	0
1	1	1	0

Num tables = 2^{2^k}

X_1	X_2	X_3	Y
0	1	1	0
1	0	1	0

X_1	X_2	X_3	Y
0	1	1	0
1	0	1	1

X_1	X_2	X_3	Y
0	1	1	1
1	0	1	0

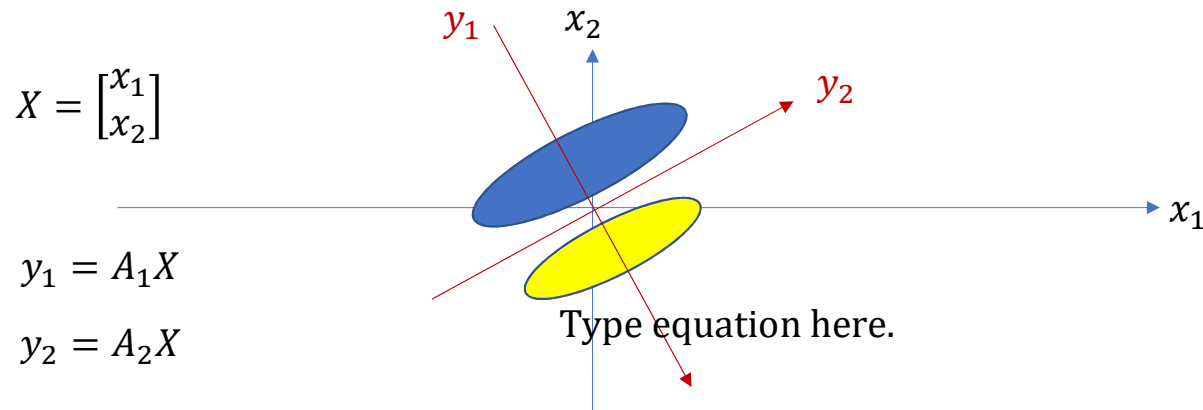
X_1	X_2	X_3	Y
0	1	1	1
1	0	1	1

- The number of missing patterns is exponential in the number of spurious bits
- The number of possible extensions to the table is exponential in the number of missing patterns
- The number of ways of adversarially modifying inputs increases superexponentially with the number of spurious bits

Sufficient statistic

- A sufficient statistic is the *minimal* function of the input that is sufficient to compute the output
- For the previous example (x_1, x_2) is a sufficient statistic
- (x_1, x_2, x_3) is *not* a sufficient statistic
 - It is overspecified

Sufficient statistic: Linear example

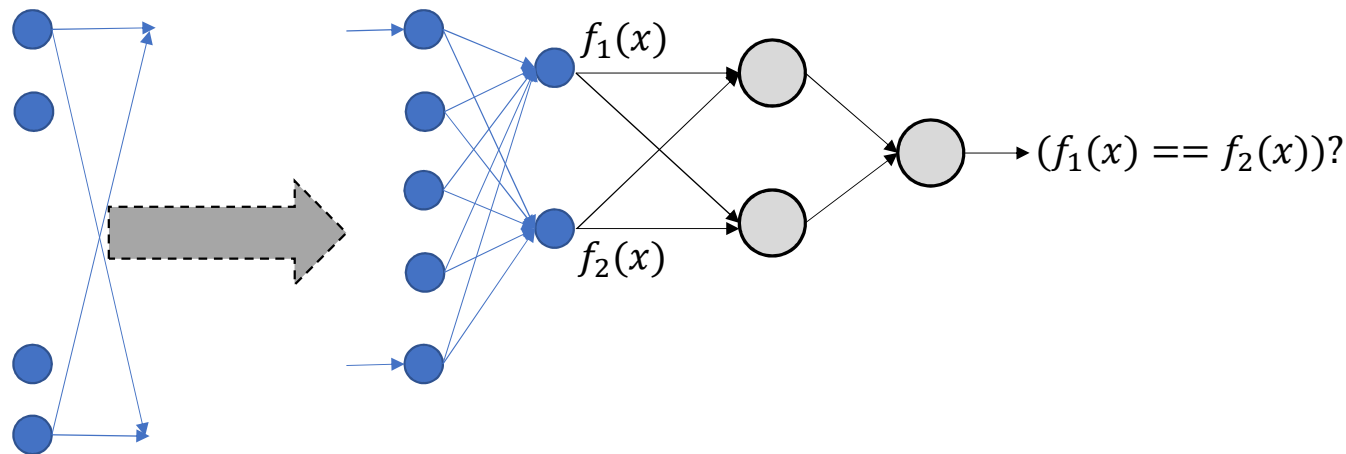


- Binary classification problem
 - Blue class vs. yellow class
- $y_1 = A_1 X$ is a sufficient statistic
 - (y_1, y_2) is not a sufficient statistic
 - (x_1, x_2) is not a sufficient statistic

Sufficient statistic

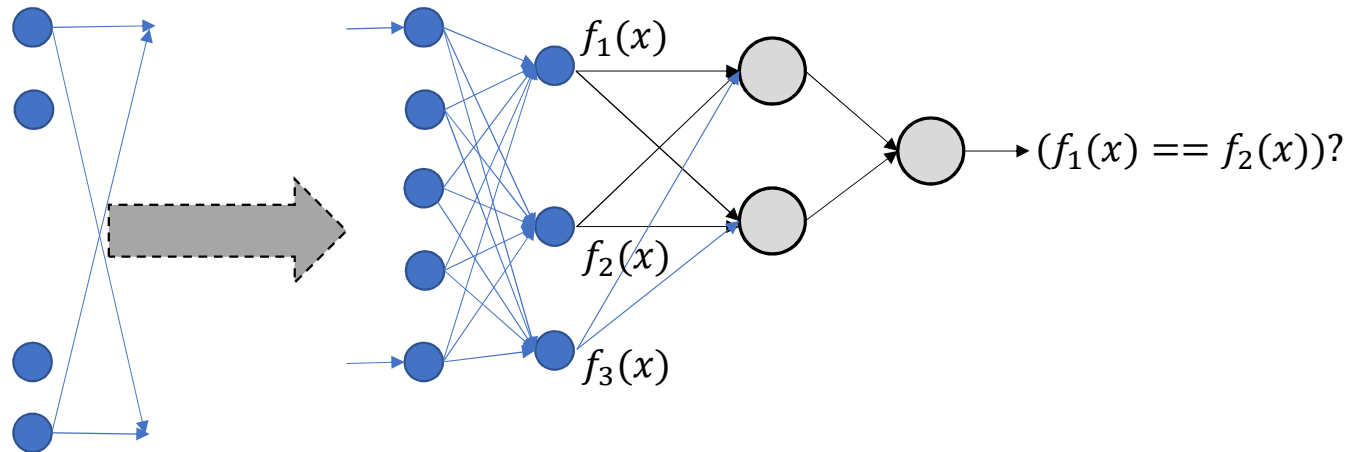
- Any classifier that operates on a non-sufficient statistic of the input is exponentially hard to learn and can be fooled by adversarial examples
- The input to any linear classifier that can be fooled by adversarial examples is *not* a sufficient statistic
 - B. Li, G. Friedland, J. Wang, R. Jia, C. Spanos, D. Song, *One Bit Matters: Understanding Adversarial Examples as the Abuse of Data Redundancies*, arXiv preprint arXiv:1810.09650, 2018
- Summary: If you provide redundant input to the classifier, it can be fooled by an adversarial example

The susceptibility of networks



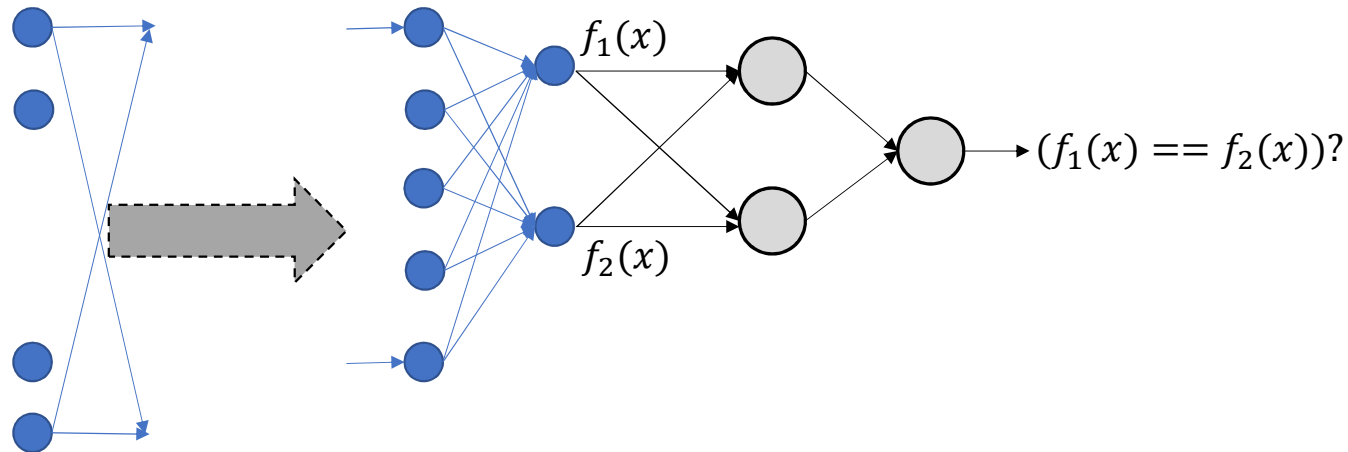
- Consider the example of $y = (f_1(x) == f_2(x))?$
 - $f_1(x)$ and $f_2(x)$ are two outputs at k th layer

The susceptibility of networks



- Consider the example of $y = (f_1(x) == f_2(x))?$
 - $f_1(x)$ and $f_1(x)$ are two outputs at k th layer
- If the network produces *three* features at the k th layer, this opens up the possibility of adversarial attack

Susceptibility of networks



- Adversarial attacks can only be prevented by having a “perfect” network
 - At least one layer that produces exactly sufficient statistic
- It is impossible to know what the minimal network architecture is for any given problem
- Any practical solutions will *always* be exploitable
 - By a more motivated attacker

Over to Yossi..

- Questions?