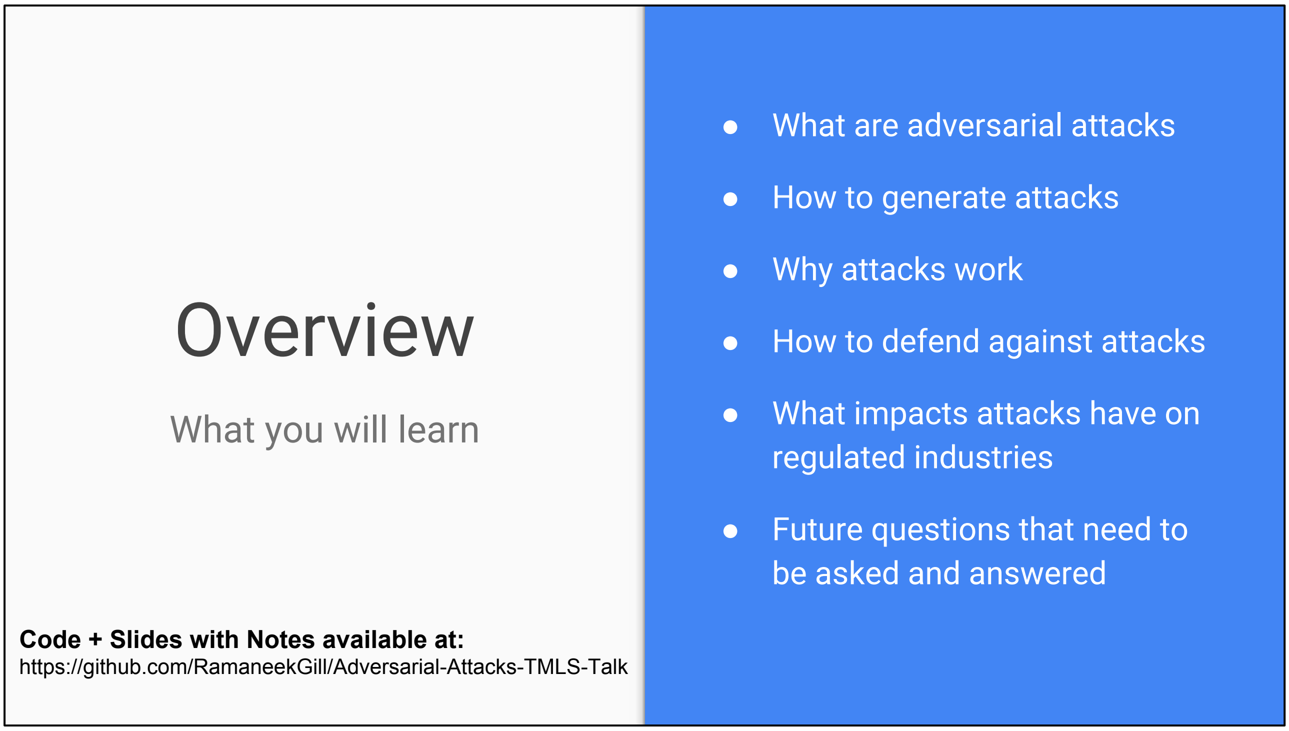
Adversarial Attacks and Their Impact on Insurance

Ramaneek Gill
CTO @ Yolk

Charles Huang
ML Engineer @ Yolk

Yilun Zhang
Data Scientist @ LOFT, Manulife



Overview

What you will learn

Code + Slides with Notes available at:
https://github.com/RamaneekGill/Adversarial-Attacks-TMLS-Talk

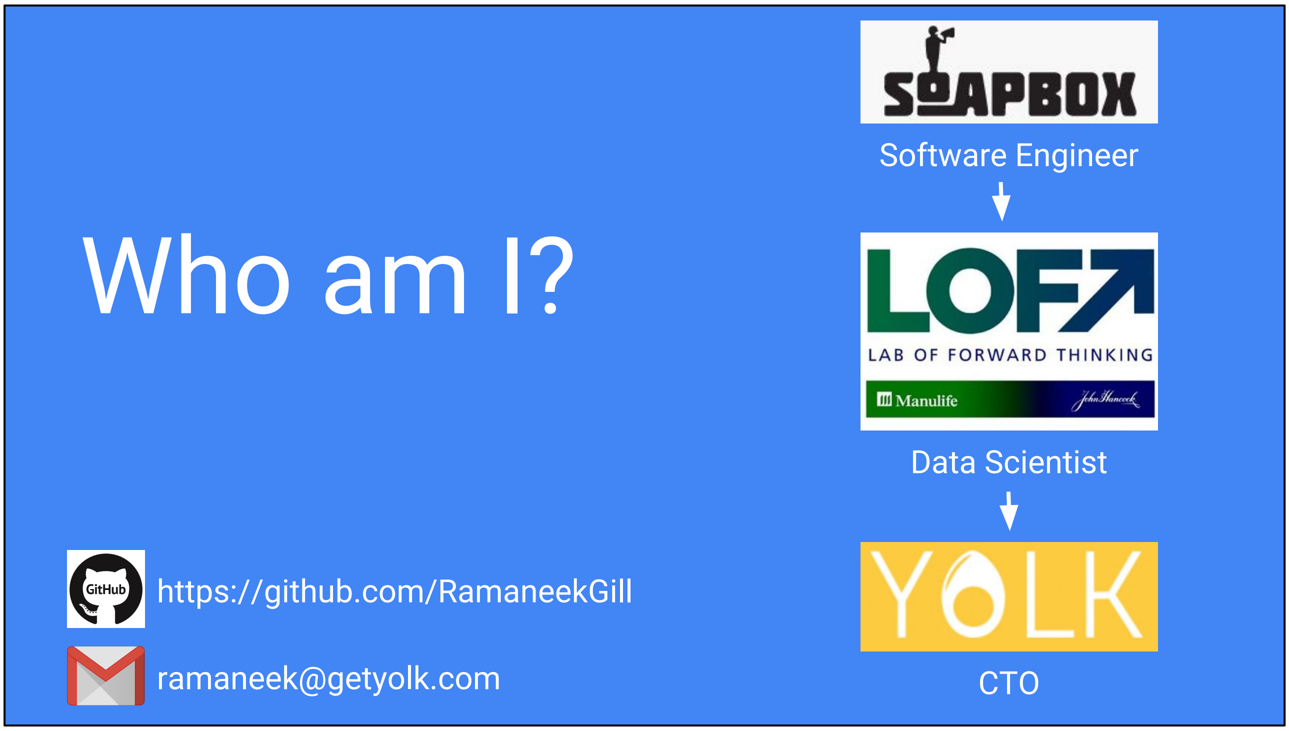
- What are adversarial attacks
- How to generate attacks
- Why attacks work
- How to defend against attacks
- What impacts attacks have on regulated industries
- Future questions that need to be asked and answered

I'll be talking about

What exactly are adversarial attacks
How to generate them
Why they work so well
How to defend against these attacks

What impacts do these attacks have on highly regulated industries such as insurance
And
Future questions that need to be asked and answered about adversarial attacks going forward

There's a lot to cover in a short time so my intention is to convey the intuitions of these attacks instead of a deep dive into the mathematics.



Who am I?

https://github.com/RamaneekGill

ramaneek@getyolk.com

Software Engineer

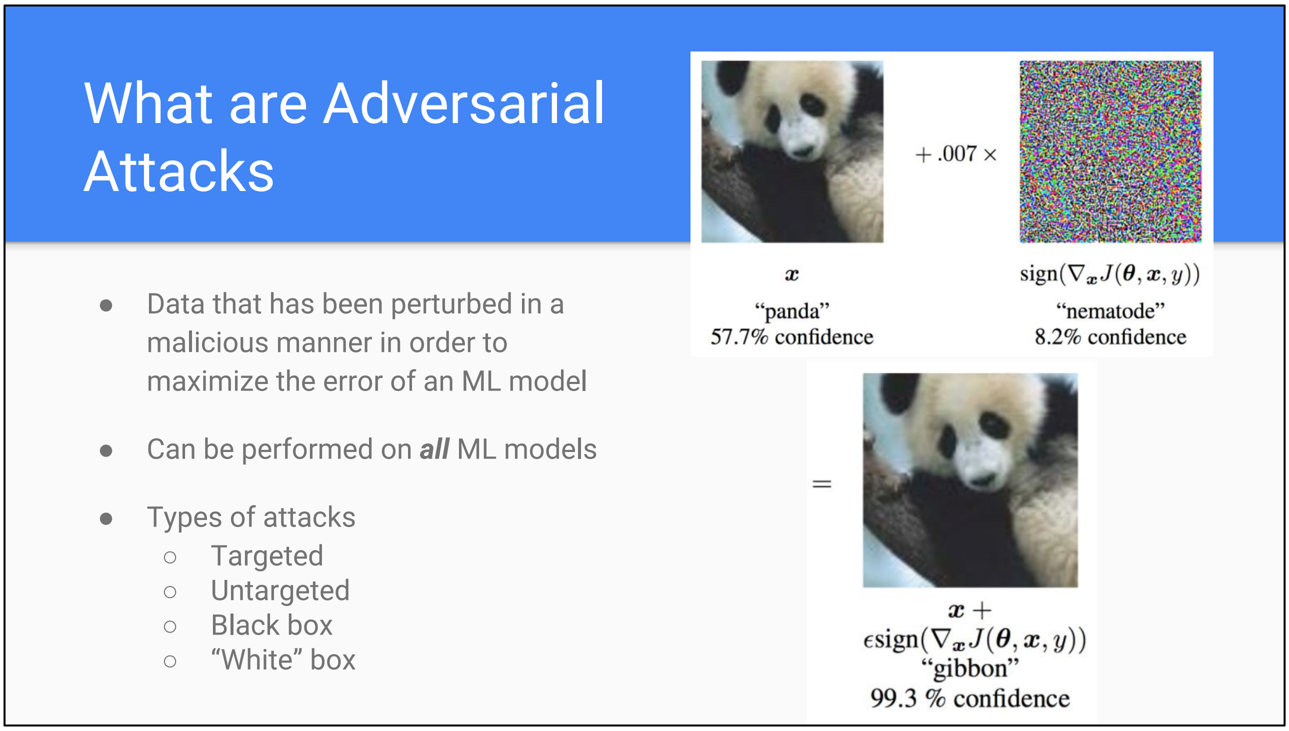
Data Scientist

CTO

So… Who am I?

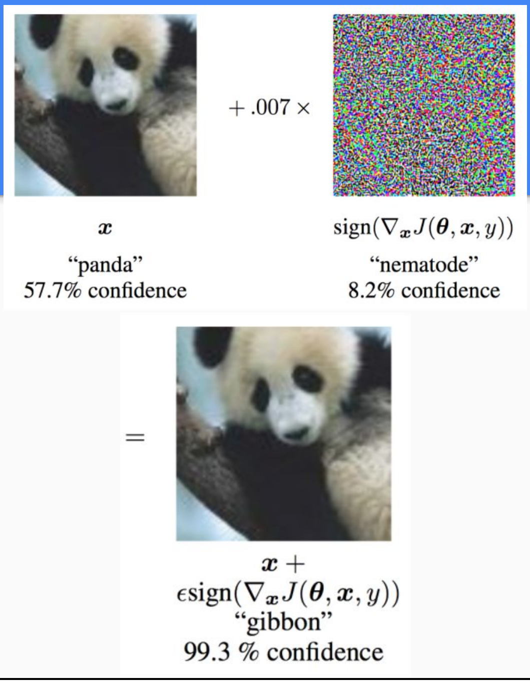
I studied computer science at university of Toronto.
Part of this research was done while I was a data scientist at LOFT which was manulife's innovation Lab

Now I'm a CTO at Yolk, we use machine learning to reduce the costs of running enterprise service desks by increasing throughput and decreasing ticket volumes.



What are Adversarial Attacks

- Data that has been perturbed in a malicious manner in order to maximize the error of an ML model
- Can be performed on ***all*** ML models
- Types of attacks
  - Targeted
  - Untargeted
  - Black box
  - “White” box


$+ .007 \times$
$\boldsymbol{x}$
“panda”
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
“nematode”
8.2% confidence
$=$
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
“gibbon”
99.3 % confidence

An adversarial attack perturbs the original data in a malicious manner in order to maximize the error of an ML model.

A good attack preserves the original structure of the data and is incredibly difficult for humans to detect.

Adversarial attacks don't necessarily have to have a sensible structure, random noise is often an example of an attack.

Can be performed on all ML models

- Simple models such as logistic regression also have the ability to be attacked
- Adversarial attacks aren't exclusive to the vision domain. They exist for text, audio, and any type of data

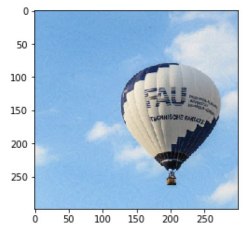
There are different types of attacks

- Targeted, Untargeted both of which can take place in a Black box or “White” box environment

Targeted vs Untargeted Attacks

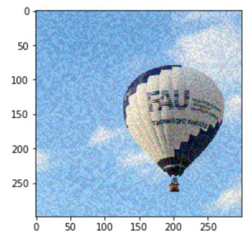
- **Targeted**
  - Specify a target label you wish for the ML model to predict
  - More difficult to do
- **Untargeted**
  - No control over which label the ML model outputs
  - Easy to do

```
Adversarial Top-5 Predictions
1: goose 0.077361
2: hand-held computer, hand-held microcomputer 0.0529477
3: ballpoint, ballpoint pen, ballpen, Biro 0.0482919
4: combination lock 0.0135026
5: bath towel 0.00898165
```



Targeted

```
Adversarial Top-5 Predictions
1: ballpoint, ballpoint pen, ballpen, Biro 0.958054
2: parallel bars, bars 0.0143989
3: CD player 0.000246849
4: whiskey jug 0.000245998
5: altar 0.000184889
```

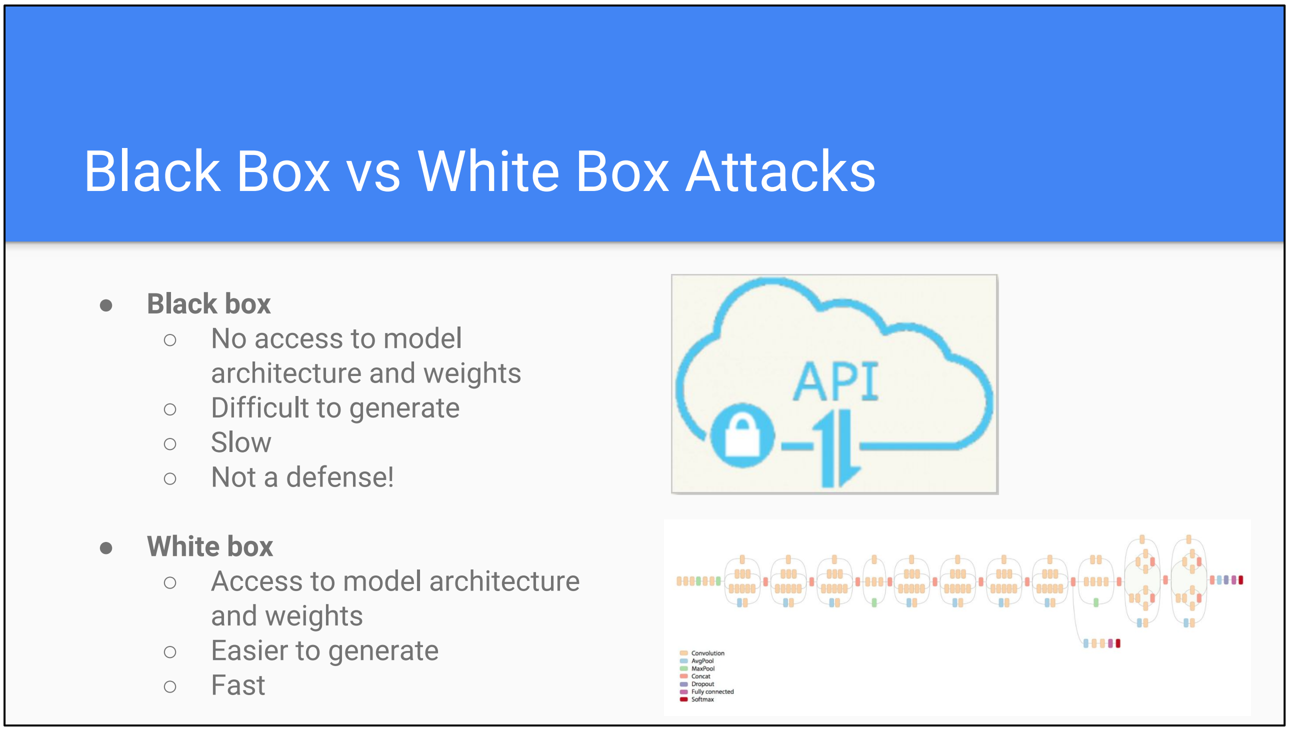


Untargeted

Targeted attacks allow you to specify a label you wish the ML model to output.

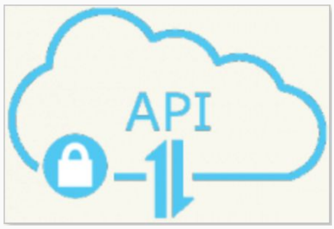
As you can imagine with malicious intent and in certain settings this can result in quite disturbing scenarios.

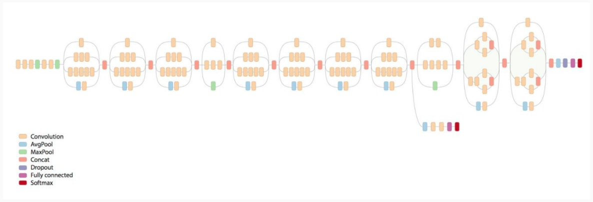
Untargeted attacks are easier to perform. In this case your goal is only to maximize errors regardless of the label being predicted.



Black Box vs White Box Attacks

- **Black box**
  - No access to model architecture and weights
  - Difficult to generate
  - Slow
  - Not a defense!
- **White box**
  - Access to model architecture and weights
  - Easier to generate
  - Fast





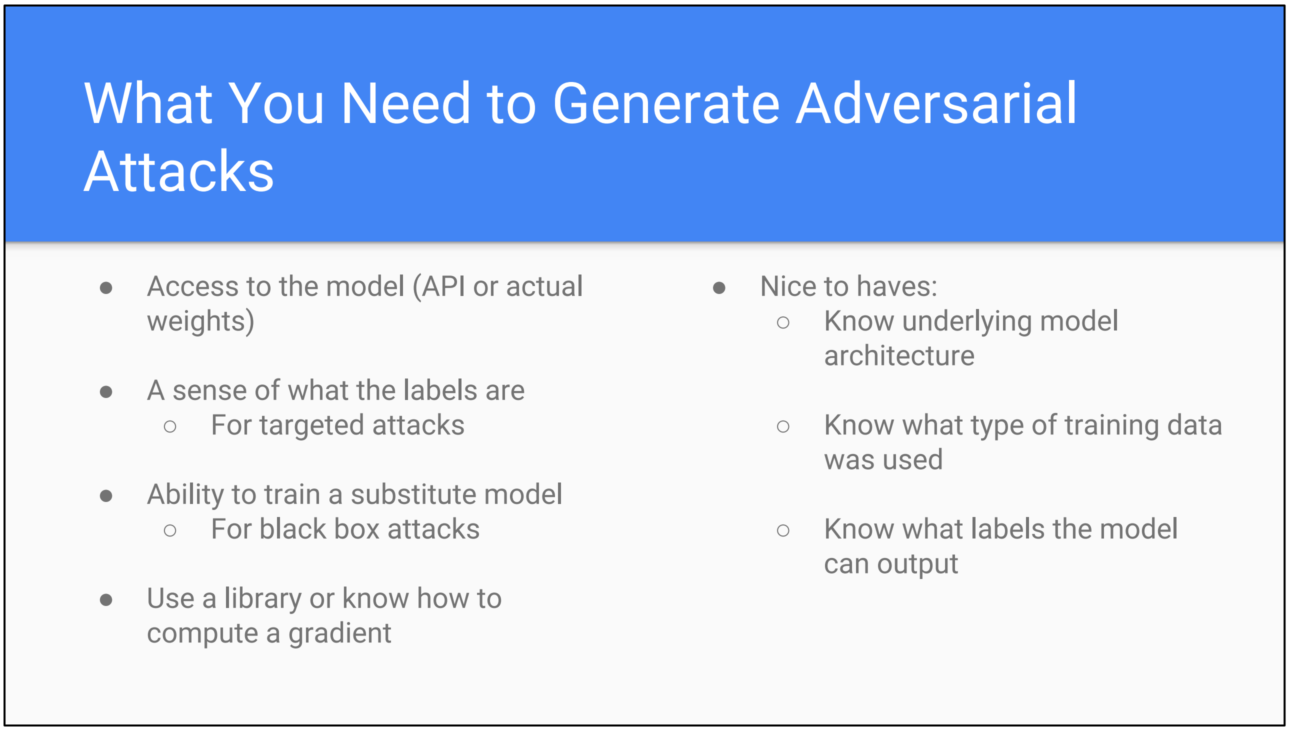
Black box attacks are situations where the adversary does not have access to the models weights

In this environment you generally have access to some API which spits out predictions when fed data.

For example IBM Watson or a similar service

In this case you have to train a substitute model that learns a similar weight space of the model being attacked. Then using the substituted model you can generate adversarial attacks to feed the API.

White box attacks are much simpler, you have access to the model and can do as you please.



What You Need to Generate Adversarial Attacks

- Access to the model (API or actual weights)
- A sense of what the labels are
  - For targeted attacks
- Ability to train a substitute model
  - For black box attacks
- Use a library or know how to compute a gradient
- Nice to haves:
  - Know underlying model architecture
  - Know what type of training data was used
  - Know what labels the model can output

To mount an attack all you need is access to the model in some way and some labelled data along with some coding or mathematical understanding. 0

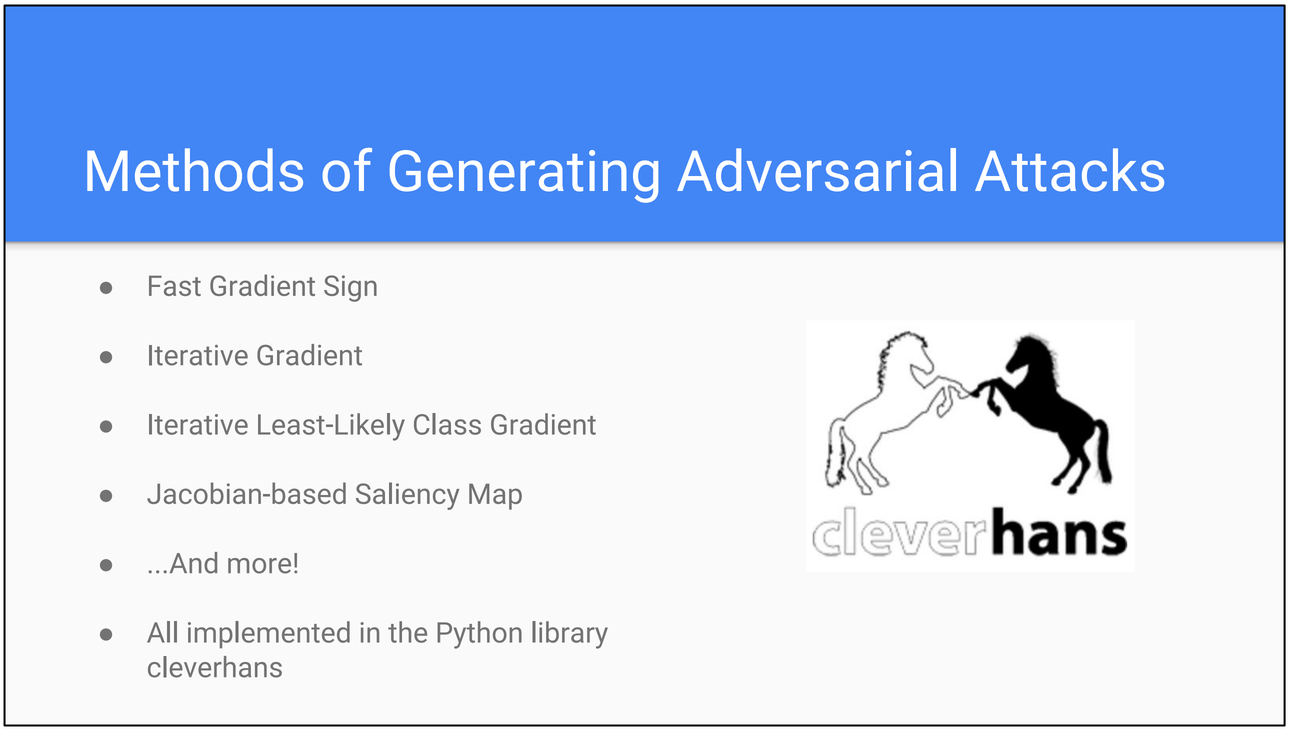
The barrier to entry is very low

Having access to the underlying model architecture helps!
 - Adversarial examples that fool one CNN often fool another CNN

Knowing the type training data and the labels really helps

 - For example fooling a model to output dog instead of human is harder for a model that has been trained primarily on those two types of data

Knowing the labels allows you to do targetted attacks,
Also allows you to perform very good untargetted attacks



Methods of Generating Adversarial Attacks

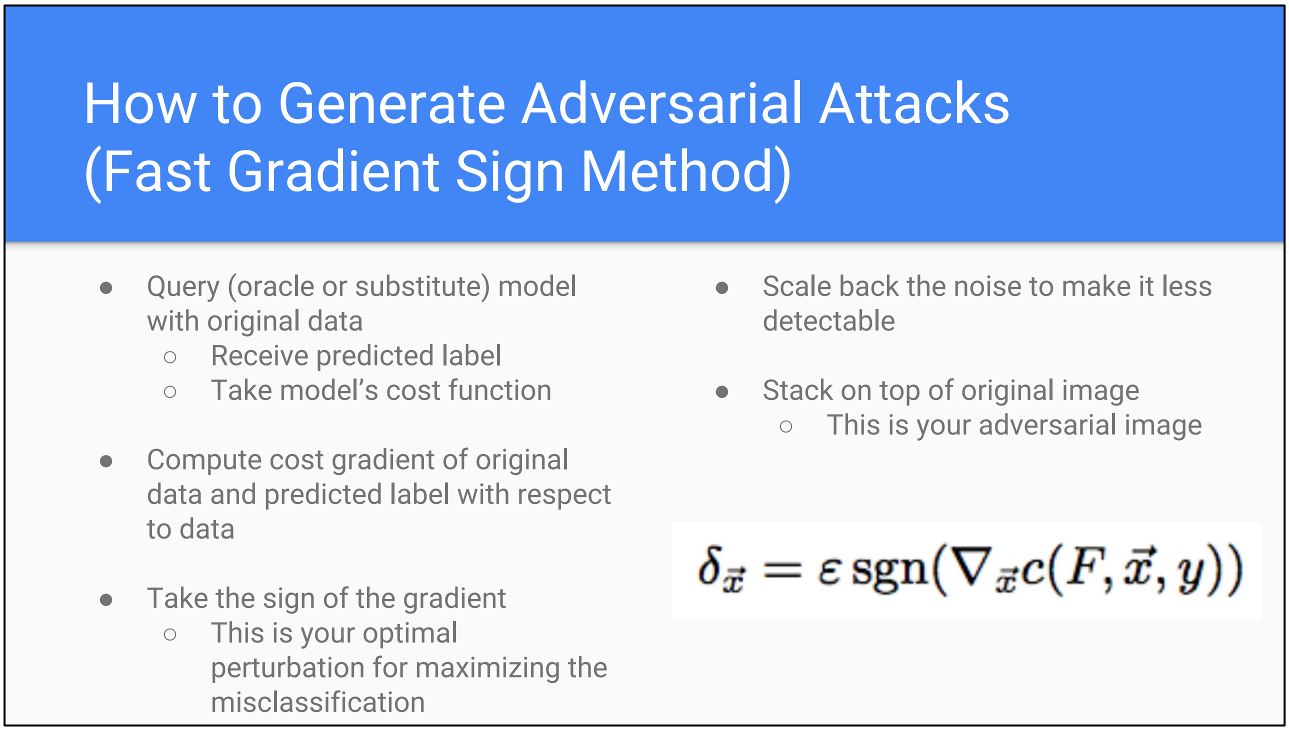
- Fast Gradient Sign
- Iterative Gradient
- Iterative Least-Likely Class Gradient
- Jacobian-based Saliency Map
- ...And more!
- All implemented in the Python library cleverhans

cleverhans

There are many methods to generate an attack.

The most popular ones are Fast Gradient Sign Method and the Basic Iterative Gradient approach

These are all implemented in the open source repository cleverhans.

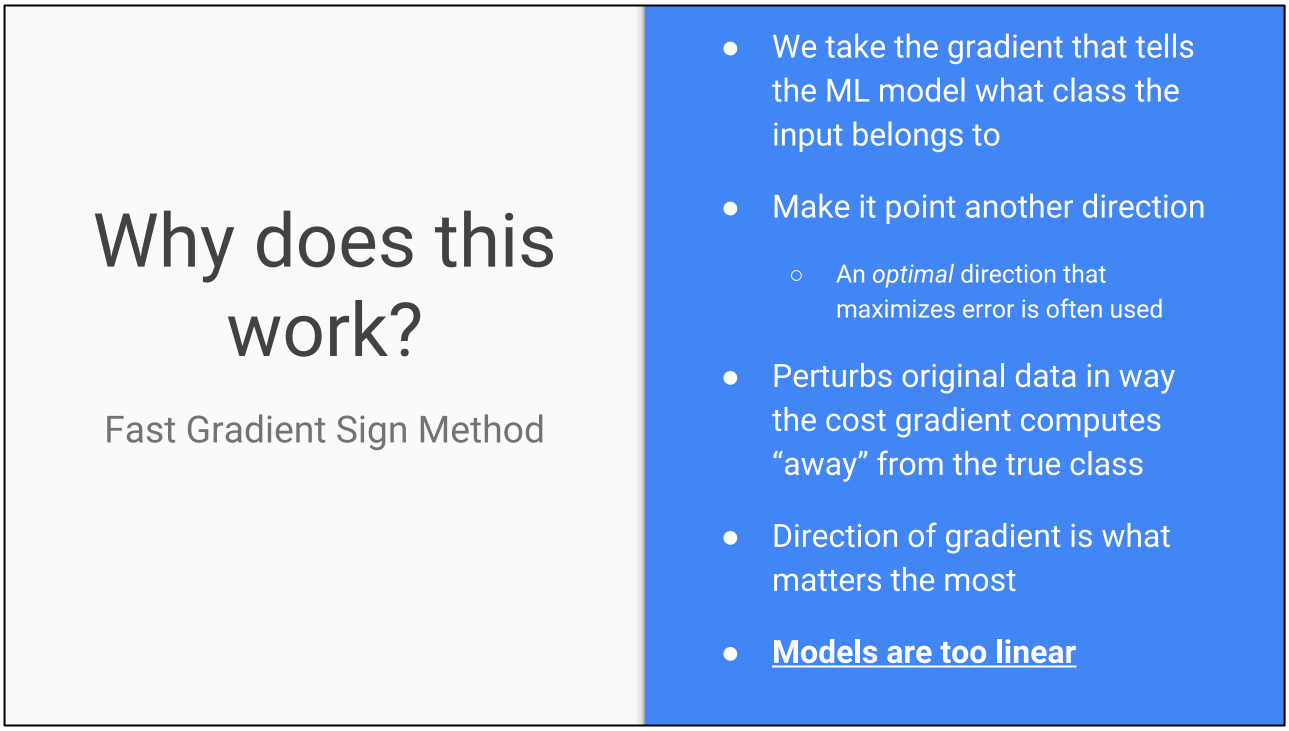


How to Generate Adversarial Attacks (Fast Gradient Sign Method)

- Query (oracle or substitute) model with original data
  - Receive predicted label
  - Take model's cost function
- Compute cost gradient of original data and predicted label with respect to data
- Take the sign of the gradient
  - This is your optimal perturbation for maximizing the misclassification
- Scale back the noise to make it less detectable
- Stack on top of original image
  - This is your adversarial image

$$\delta_{\vec{x}} = \varepsilon \, \mathrm{sgn}(\nabla_{\vec{x}} c(F, \vec{x}, y))$$

(((Read the slide)))

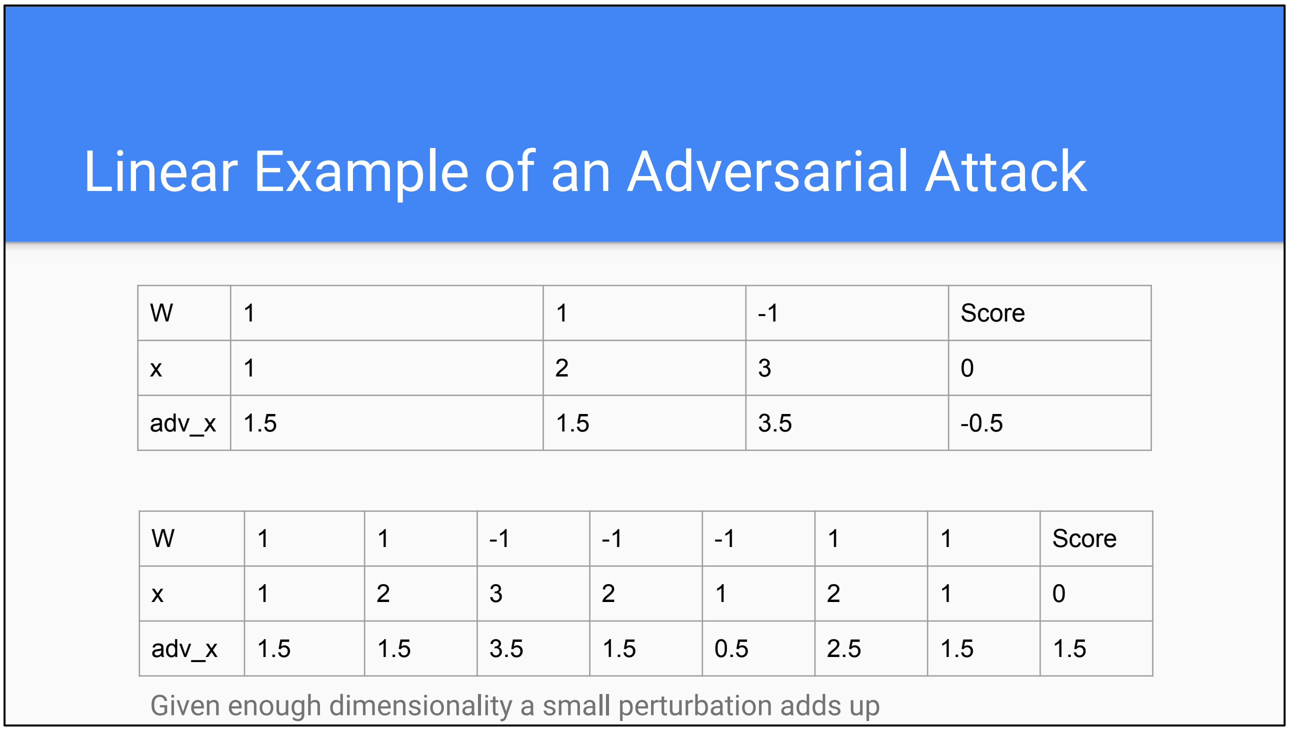


Why does this work?

Fast Gradient Sign Method

- We take the gradient that tells the ML model what class the input belongs to
- Make it point another direction
  - An *optimal* direction that maximizes error is often used
- Perturbs original data in way the cost gradient computes “away” from the true class
- Direction of gradient is what matters the most
- **Models are too linear**

By models being too linear what I … and Ian Goodfellow et al mean is that our models are easy to optimize so they are easy to fool. For example many CNNs use ReLU activations which are just linear transformations.
Given enough dimensionality, a small perturbation across each dimension sums up to a large change in overall score



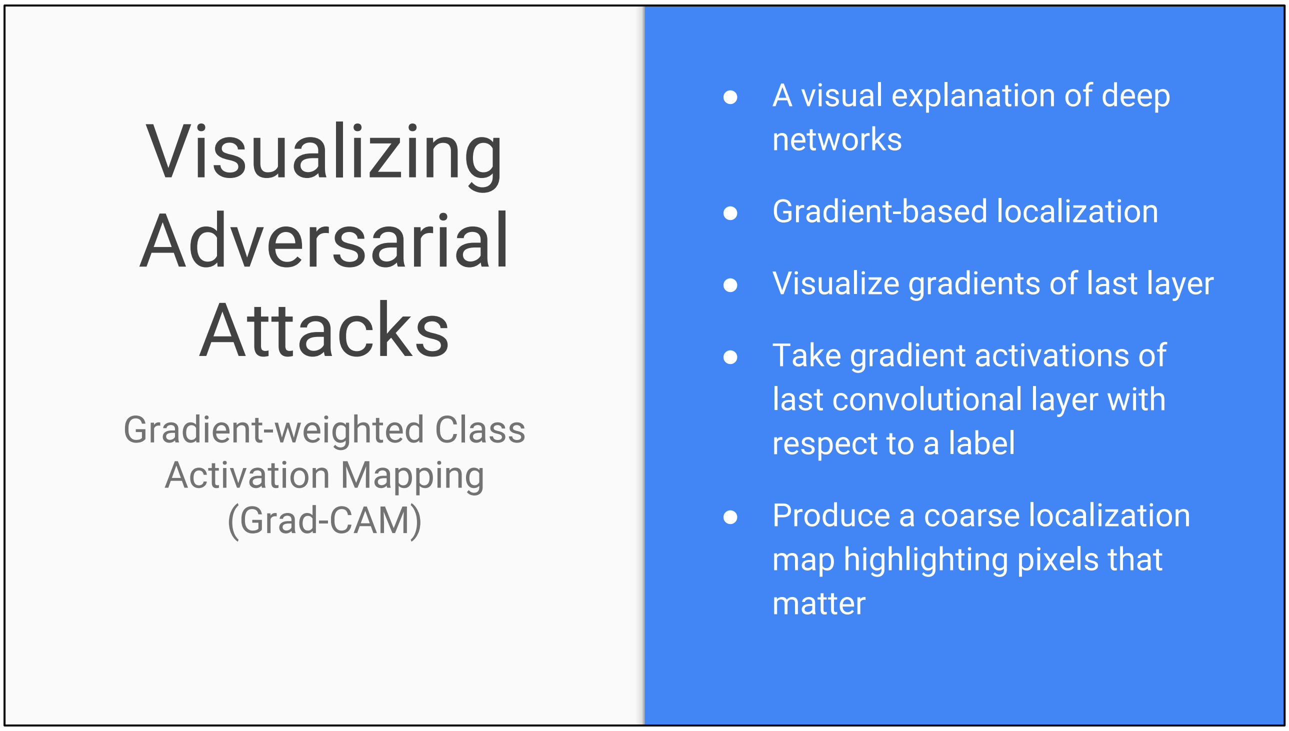
Linear Example of an Adversarial Attack

W	1	1	-1	Score
x	1	2	3	0
adv_x	1.5	1.5	3.5	-0.5

W	1	1	-1	-1	-1	1	1	Score
x	1	2	3	2	1	2	1	0
adv_x	1.5	1.5	3.5	1.5	0.5	2.5	1.5	1.5

Given enough dimensionality a small perturbation adds up

Here's a slide that shows that higher dimensions imply that perturbations add up



Visualizing Adversarial Attacks

Gradient-weighted Class Activation Mapping (Grad-CAM)

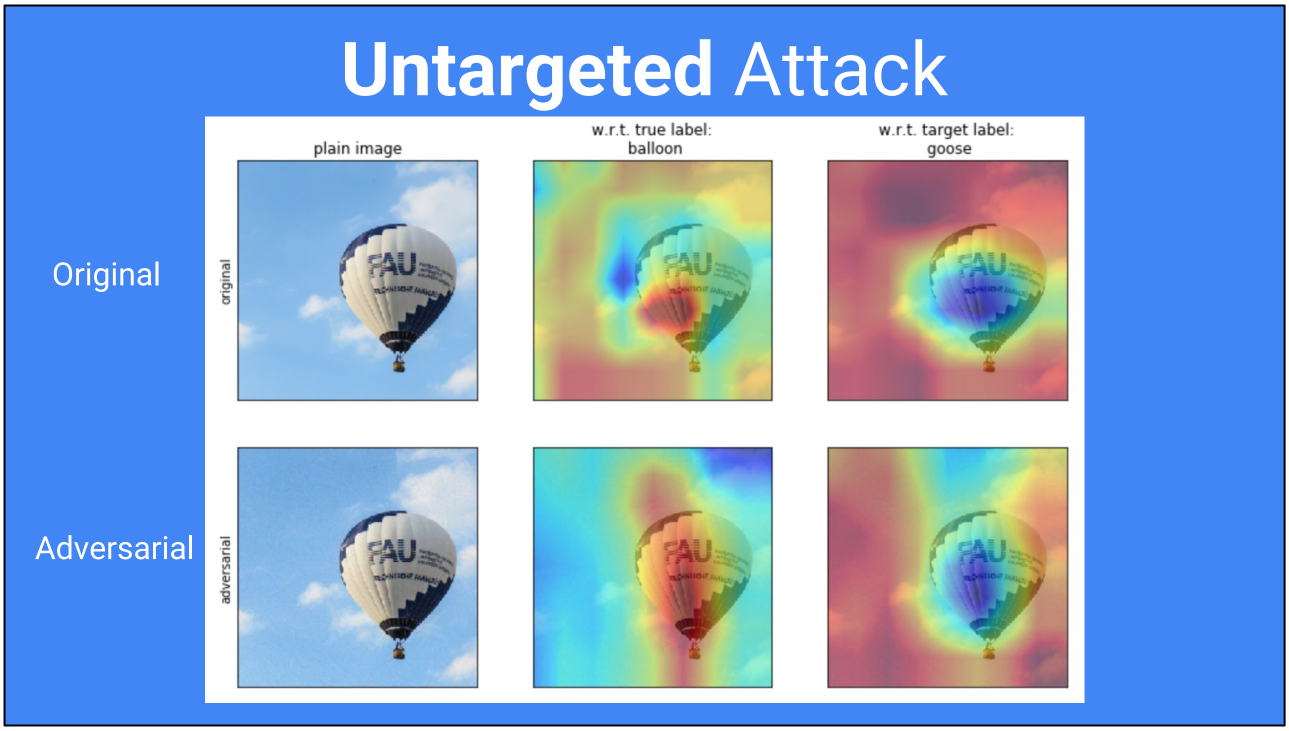
- A visual explanation of deep networks
- Gradient-based localization
- Visualize gradients of last layer
- Take gradient activations of last convolutional layer with respect to a label
- Produce a coarse localization map highlighting pixels that matter

To visually explain why adversarial attacks performed so well we visualized the gradients of inception net

We used Grad Cam because it produces a coarse localization map.

Grad-CAM uses the gradients with respect to a target label and visualized the last convolutional layer

Untargeted Attack



An output from our Grad-CAM visualizations.
A picture is worth a thousand words so there are many things to note!

Untargeted basic iterative attack which was moderately successful

Explanation of what you're seeing:

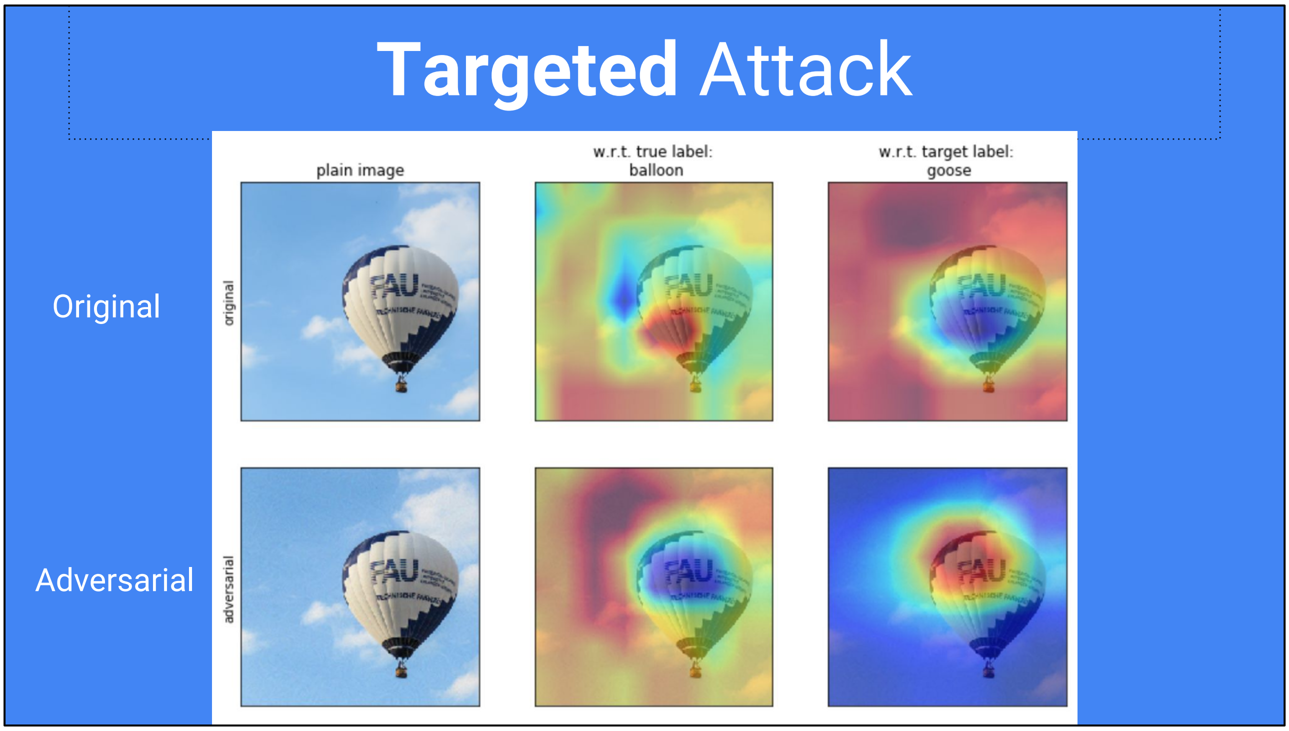
- Top row is the original image
- Bottom row is adversarial image
- First column is the image without heat map visualizations
- Red is high intensity, blue is low intensity of the gradient activations from the last convolutional later
- Second column is visualizing Inception net activations when differentiating with respect to the target label `balloon`.
- Third is visualization with respect to the label `goose`

Note:

- Original and adversarial image look almost the same! (They're not!)
- The original image doesn't have "goose" present on the balloon, that makes sense
- The adversarial image doesn't have "goose" present on the balloon, that makes sense
    - This was an untargeted attack, so we didn't try and make the model

    - output `goose` just something else
- The heatmap for adversarial image and original image differ but hold a similar structure

Targeted Attack



Now we adversarially perturbed the original image to output `goose`. This is a targeted attack using the Basic Iterative Gradient Method.

In this attack the model predicted goose as its first choice, balloon was not present in the top 5 predictions

Note:

- Original and adversarial image look almost the same! (They're not!)
- The original image doesn't have "goose" present on the balloon, that makes sense

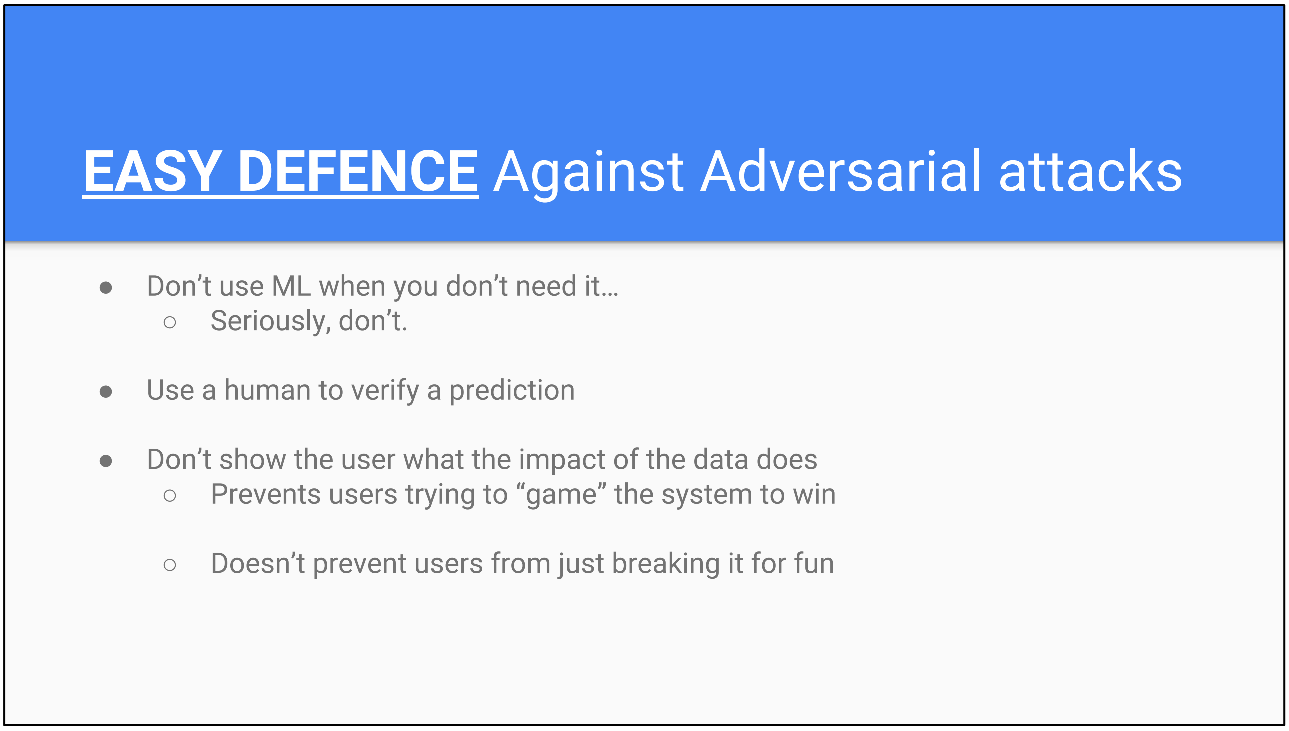
- Adversarial image when visualizing `balloon` is very similar to original image visualizing goose
    - This is very very interesting
    - Adversarial image made our model say "There's no `balloon` but I really see a goose"
    - Original image made our model say "Theres no `goose` but I really see a balloon"

- **Adversarial image when visualizing `goose` is almost an inverse of the original image visualizing goose**

- The heatmap for adversarial image and original image differ but hold a similar structure

Properties of Adversarial Attacks

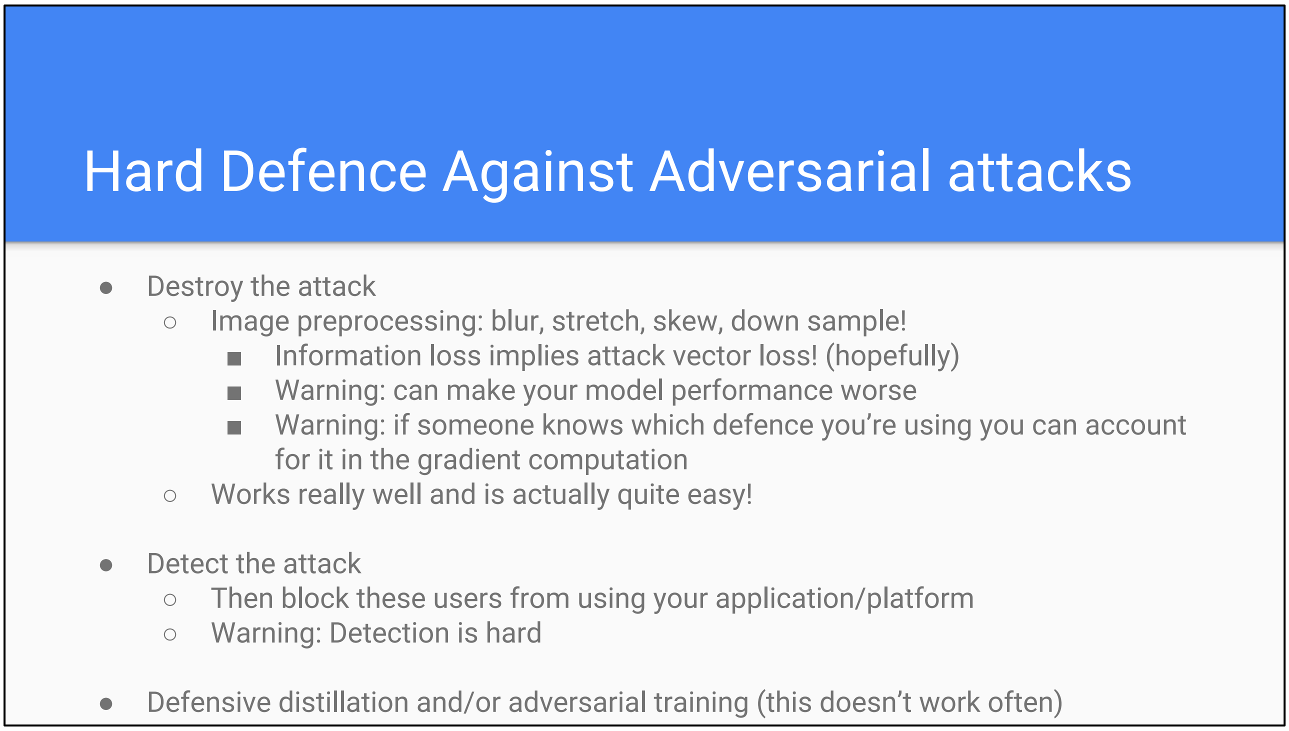
- Adversarial attacks are transferable
  - If they work on one CNN model they will likely work on another CNN model
  - Even with disjoint training sets!
  - Taking a photo of an adversarial attack still retains it somewhat
- Adversarial examples are hard to detect
- Hard to defend against
  - Very hard...



EASY DEFENCE Against Adversarial attacks

- Don't use ML when you don't need it…
    - Seriously, don't.
- Use a human to verify a prediction
- Don't show the user what the impact of the data does
    - Prevents users trying to "game" the system to win
    - Doesn't prevent users from just breaking it for fun

- Don't use ML when you don't need it… Seriously… Don't
    - Your developers and your wallet will be much happier
- Use a human to verify a prediction
- Don't show the user what the impact of the data does
- Prevents users trying to "game" the system to win
- Doesn't prevent users from just breaking it for fun
    - If you're application on the internet not the best idea...



Hard Defence Against Adversarial attacks

- Destroy the attack
  - Image preprocessing: blur, stretch, skew, down sample!
    - Information loss implies attack vector loss! (hopefully)
    - Warning: can make your model performance worse
    - Warning: if someone knows which defence you're using you can account for it in the gradient computation
  - Works really well and is actually quite easy!

- Detect the attack
  - Then block these users from using your application/platform
  - Warning: Detection is hard

- Defensive distillation and/or adversarial training (this doesn't work often)

These are the methods of defence in order of difficulty

Destroying the attack

Detecting the attack

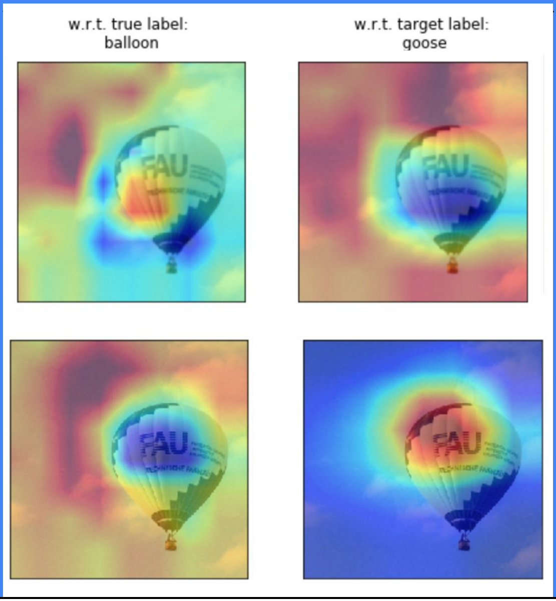
Resisting the attack

As you can probably see… the hardest way to defend is the best way from a machine learning perspective
It is easier to defend from a systems/process perspective.

Targeted Attack + Blur Defence

Adversarial Image With Gaussian Blur

Adversarial Image Targeting Goose

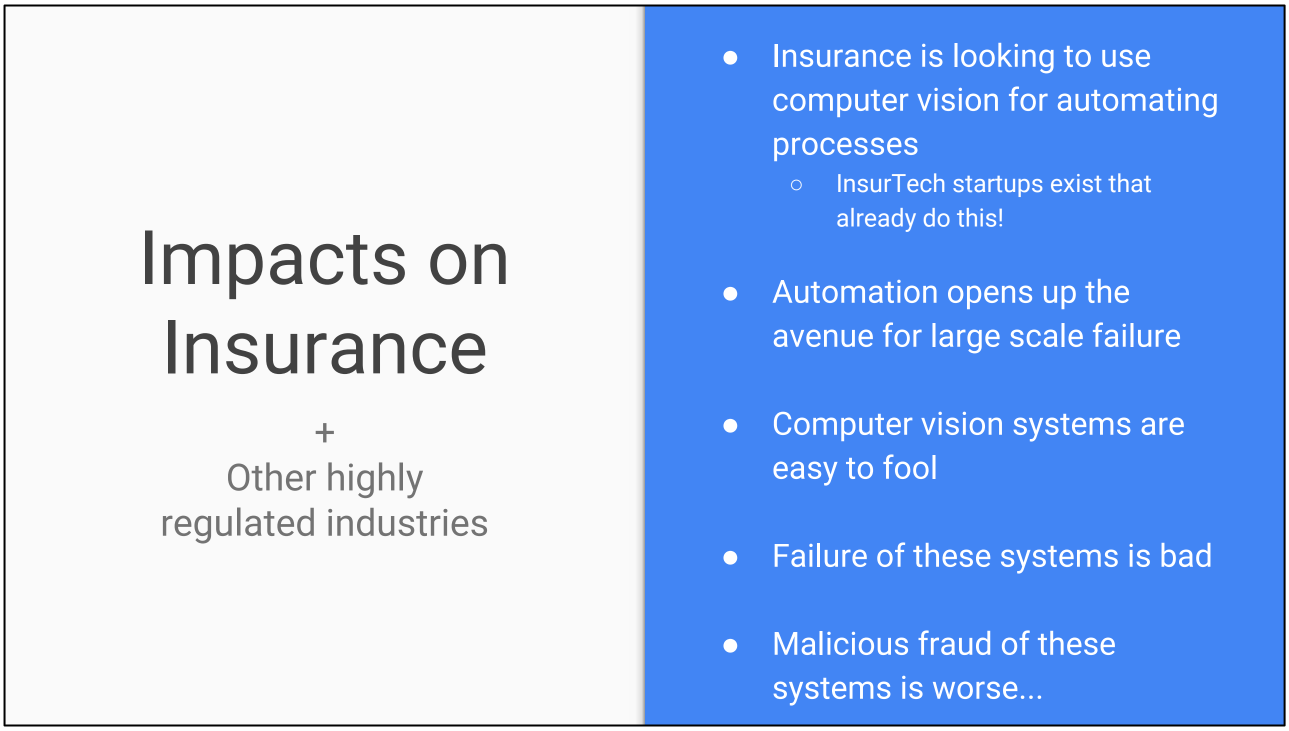


- Adversarial Top-5 Predictions after Gaussian Filter
    - 1: balloon 0.969152
    - 2: parachute, chute 0.00150866
    - 3: bathing cap, swimming cap 0.00069065
    - 4: water tower 0.000344483

Blur defences work really well!
A very good targeted attack fell apart after blurring

The blur resulted in the structure of the perturbation to be lost.
At the same time it reduced the quality of the image

Remember the attack used the Basic Iterative Gradient method, this is method that performs very fine perturbations that are hard to see.
Small perturbations are easily lost with blur or other operations that result in information loss



Impacts on Insurance

+

Other highly regulated industries

- Insurance is looking to use computer vision for automating processes
  - InsurTech startups exist that already do this!
- Automation opens up the avenue for large scale failure
- Computer vision systems are easy to fool
- Failure of these systems is bad
- Malicious fraud of these systems is worse...

Insurance (and other industries) are looking to use computer vision for automating underwriting and other processes

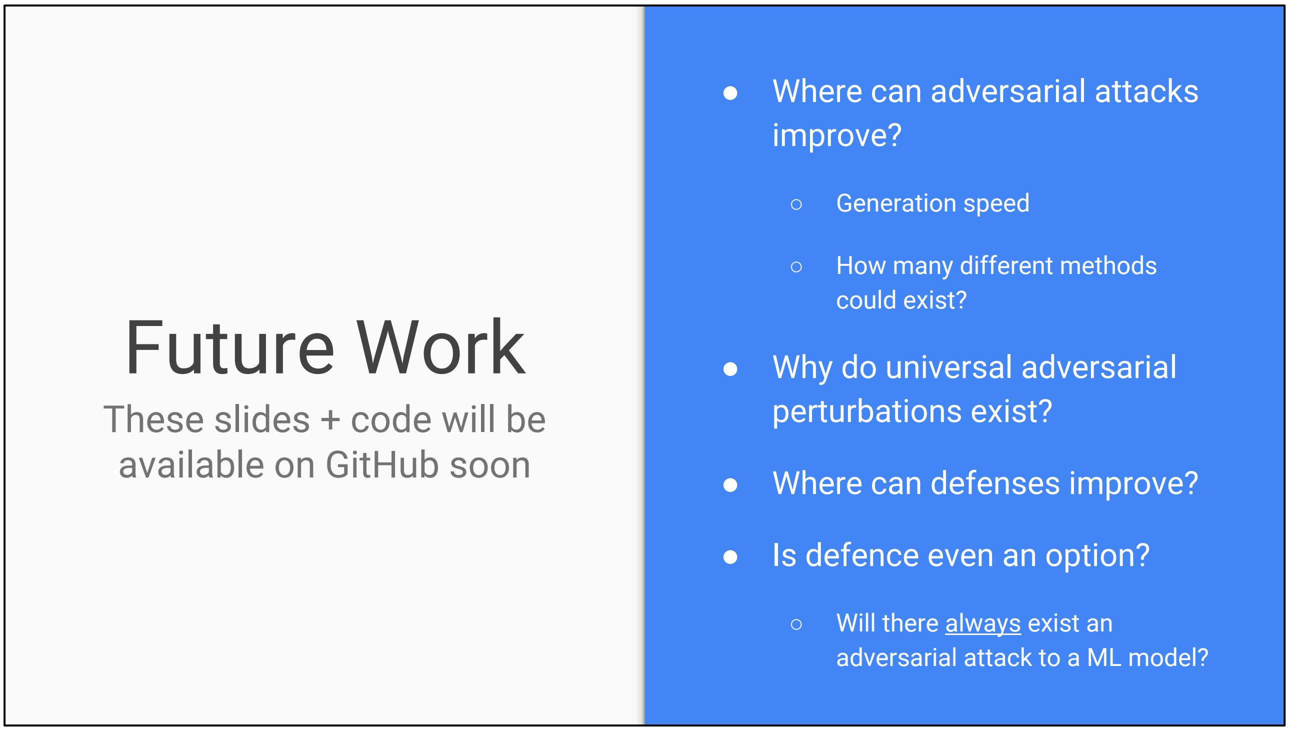
There are startups that exist in the InsurTech space that already do this

Automation opens up the avenue for large scale failure when things go wrong. Concentrated risk is increasing.

Malicious fraud is the worst scenario for this situation. Today anyone can download a script to crack weak wireless networks,

Maybe tomorrow there will be a web app to increase your auto or home insurance payout in non ethical and fraudulent ways.

And in this scenario who takes the blame?

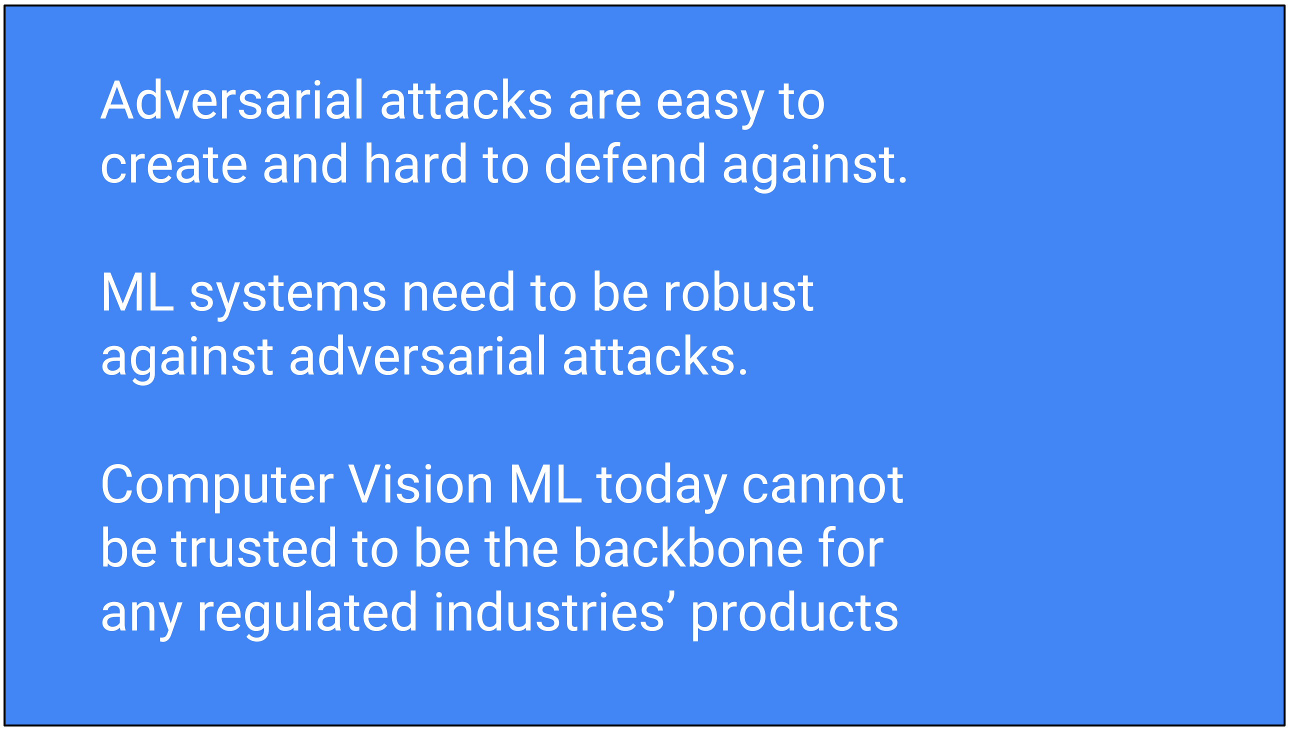


Future Work

These slides + code will be available on GitHub soon

- Where can adversarial attacks improve?
  - Generation speed
  - How many different methods could exist?
- Why do universal adversarial perturbations exist?
- Where can defenses improve?
- Is defence even an option?
  - Will there always exist an adversarial attack to a ML model?

- We already have methods to generate adversarial attacks that are hard to detect for humans and models
  - Generating attacks are slow, this can be improved in the future
- There is recent research showing a perturbation that universally destroys many deep learning classifiers when added on to images
  - Why do these perturbations exist and how can we combat them
- Whereever there is an ML model, there is a way to generate and adversarial attack
  - Will we ever come to a point where these attacks stop existing
  - At what threshold is the difference between an adversarial perturbation vs bad/noisy data?



Adversarial attacks are easy to create and hard to defend against.

ML systems need to be robust against adversarial attacks.

Computer Vision ML today cannot be trusted to be the backbone for any regulated industries’ products

3 take aways from this talk

Thank you

End of talk

References

- https://blog.xix.ai/how-adversarial-attacks-work-87495b81da2d
- https://blog.openai.com/adversarial-example-research/
- https://github.com/tensorflow/cleverhans/
- Breaking Things Is Easy
- Is attacking machine learning harder than defending it?
- The challenge of verification and testing of machine learning
- Adversarial Introductory Paper (Intriguing properties of neural networks)
- Fast Gradient Sign Method (Explaining and Harnessing Adversarial Examples)
- Black-Box Attack (Practical Black-Box Attacks against Machine Learning)
- Basic Iterative Method (Adversarial Examples In The Physical World)
- Jacobian-based Saliency Map Approach (The Limitations of Deep Learning in Adversarial Settings)
- Virtual Adversarial Training (Distributional Smoothing with Virtual Adversarial Training)
- Delving into Transferable Adversarial Examples and Black-Box Attacks
- Deep Adversarial Examples (Adversarial Manipulation of Deep Representations)
- Ensemble Adversarial Training: Attacks and Defenses
- Synthesizing Robust Adversarial Examples
- Towards Deep Learning Models Resistant to Adversarial Attacks
- Adversarial Machine Learning at Scale