Adversarial Robustness

M. Soleymani
Sharif University of Technology
Spring 2024

Most slides have been adapted from:

Zico Kolter, Aleksander Madry, Adversarial ML Tutorial

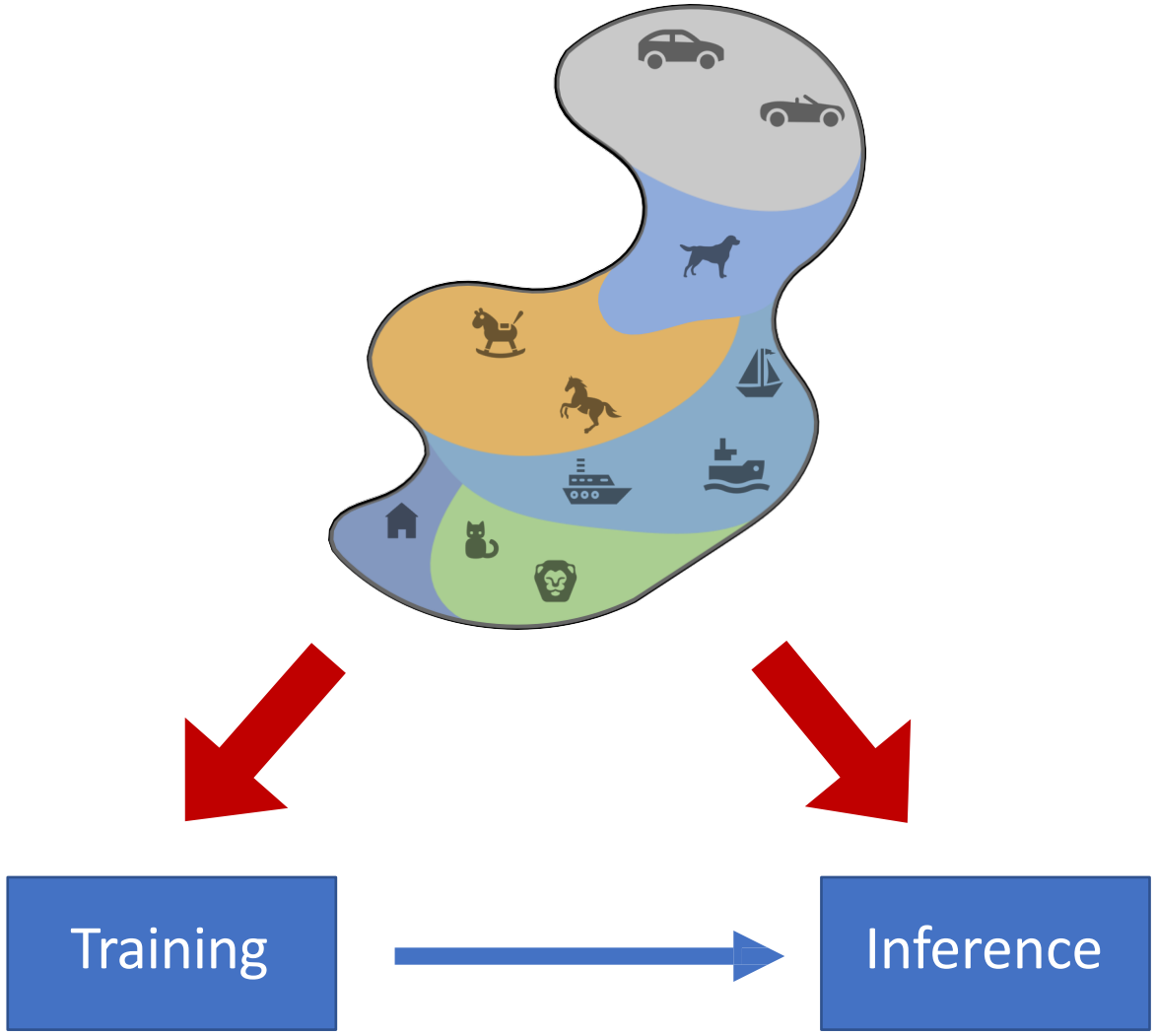
Anupam Datta, John Mitchell; Stanford CS329T

Outline

- Machine learning models can be very brittle
- What are adversarial attacks? How to generate adversarial examples?
- There are systematic methods to improve robustness
- There are limitations in the current state of the art/science
- Other types of threats for ML methods

Is ML **truly** ready for real-world deployment?

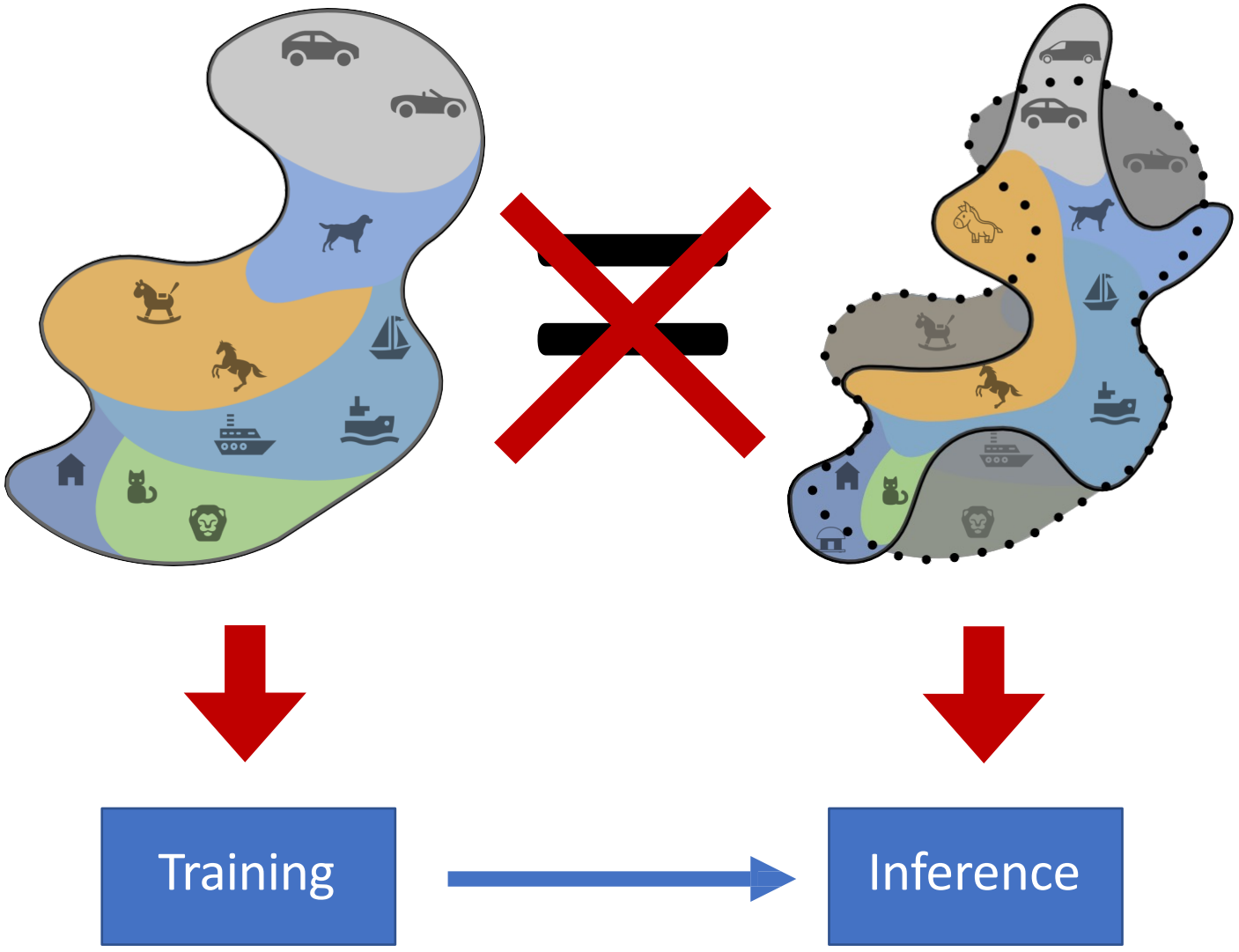
A Limitation of the (Supervised) ML Framework



Measure of performance:
Fraction of mistakes during testing

But: In reality, the distributions we **use** ML on are NOT the ones we **train** it on

A Limitation of the (Supervised) ML Framework



Measure of performance:
Fraction of mistakes during testing

But: In reality, the distributions we **use** ML on are NOT the ones we **train** it on

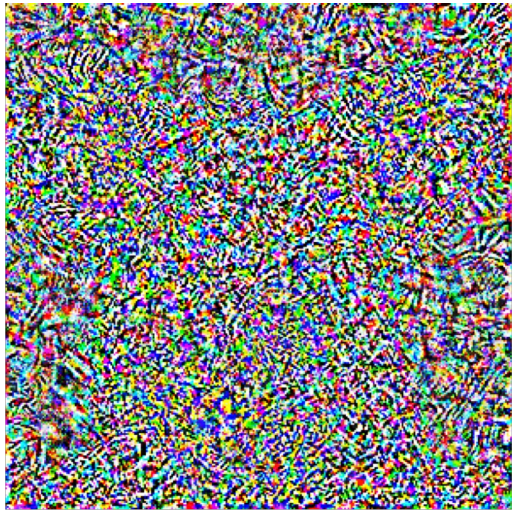
What can go wrong?

ML Predictions Are (Mostly) Accurate but Brittle

“pig” (91%)

+ 0.005 x

noise (NOT random)



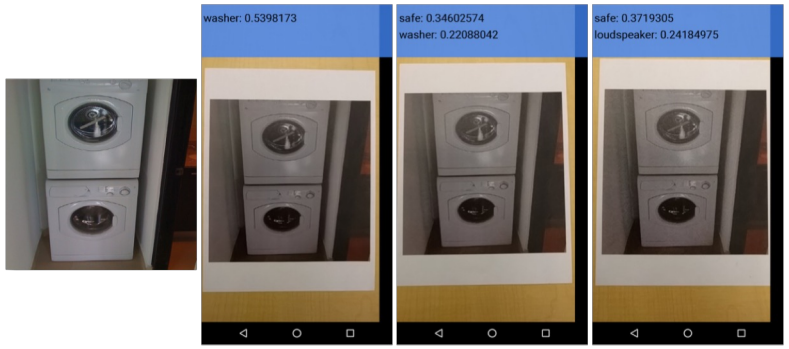
=

“airliner” (99%)

Adversarial examples are imperceptible to humans

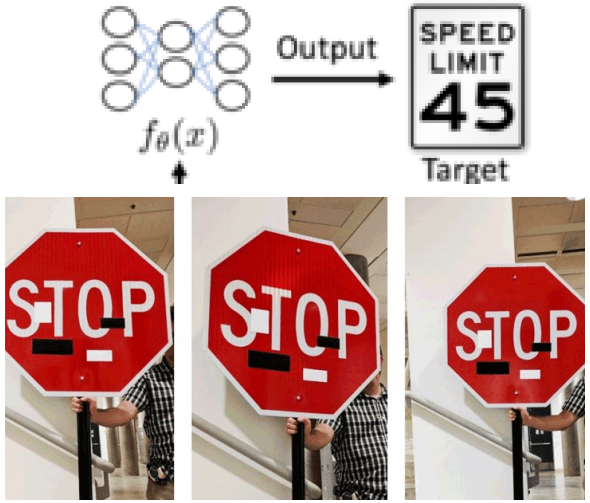
[Szegedy Zaremba Sutskever Bruna Erhan Goodfellow Fergus 2013]
[Biggio Corona Maiorca Nelson Srndic Laskov Giacinto Roli 2013]

ML Predictions Are (Mostly) Accurate but Brittle

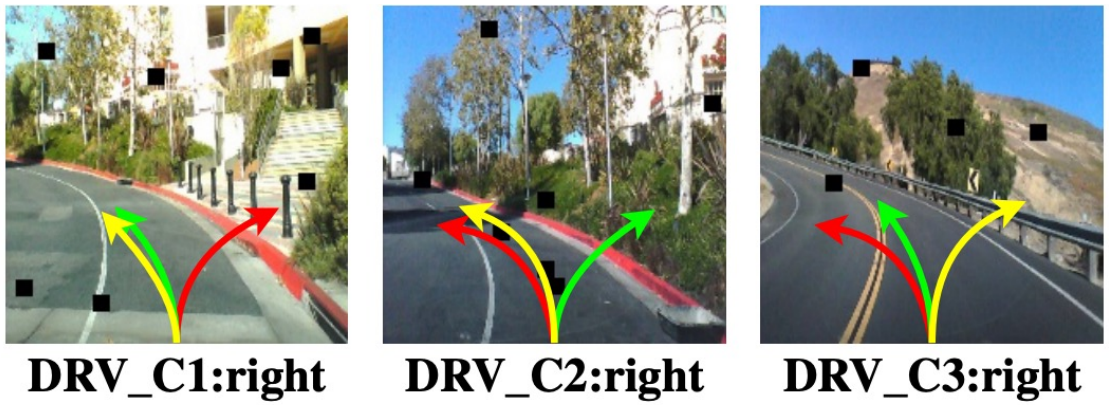


[Kurakin, Goodfellow, Bengio, 2017]

[Athalye et al., 2017]

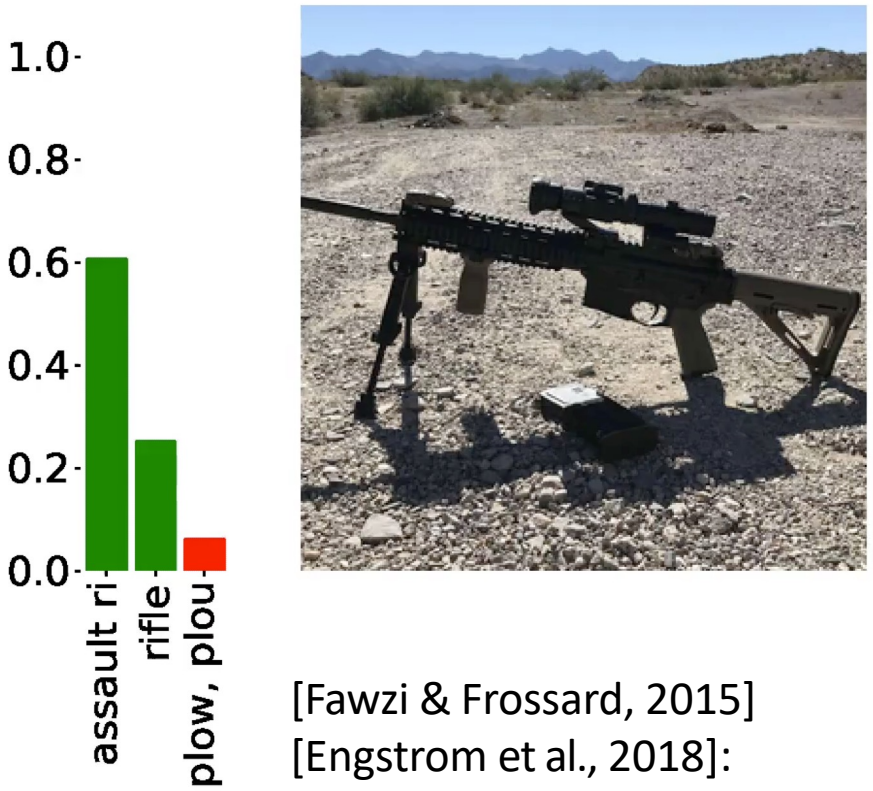


[Eykholt et al., 2017]



[Pei et al., DeepXplore, 2017]

ML Predictions Are (Mostly) Accurate but Brittle



[Fawzi & Frossard, 2015]
[Engstrom et al., 2018]:

Rotation + Translation suffices to fool state-of-the-art vision models

→ Data augmentation does not seem to help here either

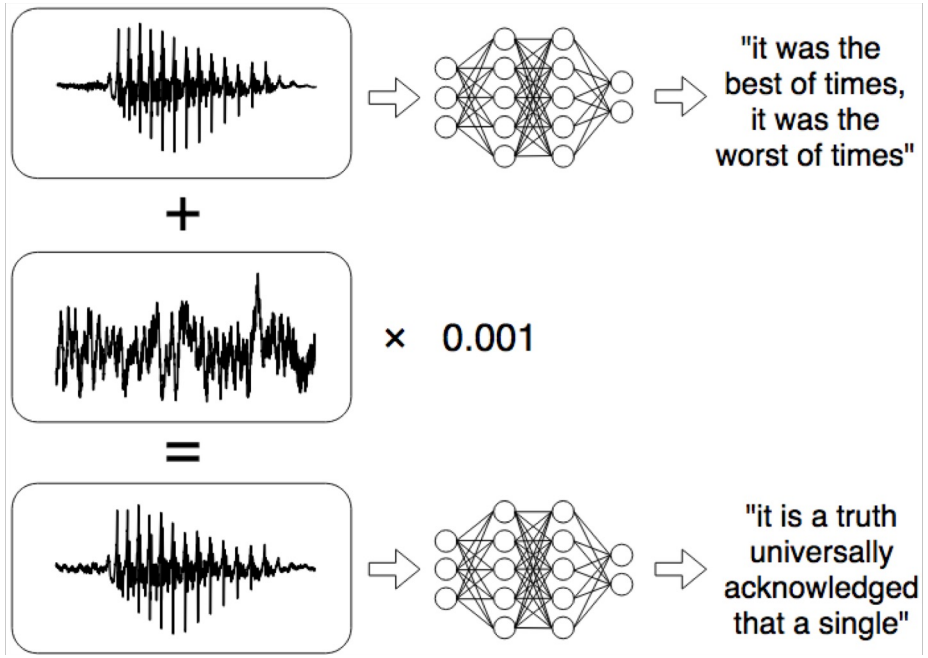
So: Brittleness of ML is a thing

Should we be worried?

Why Is This Brittleness of ML a Problem?

→ Security

[Carlini Wagner 2018]:
Voice commands that are unintelligible to humans

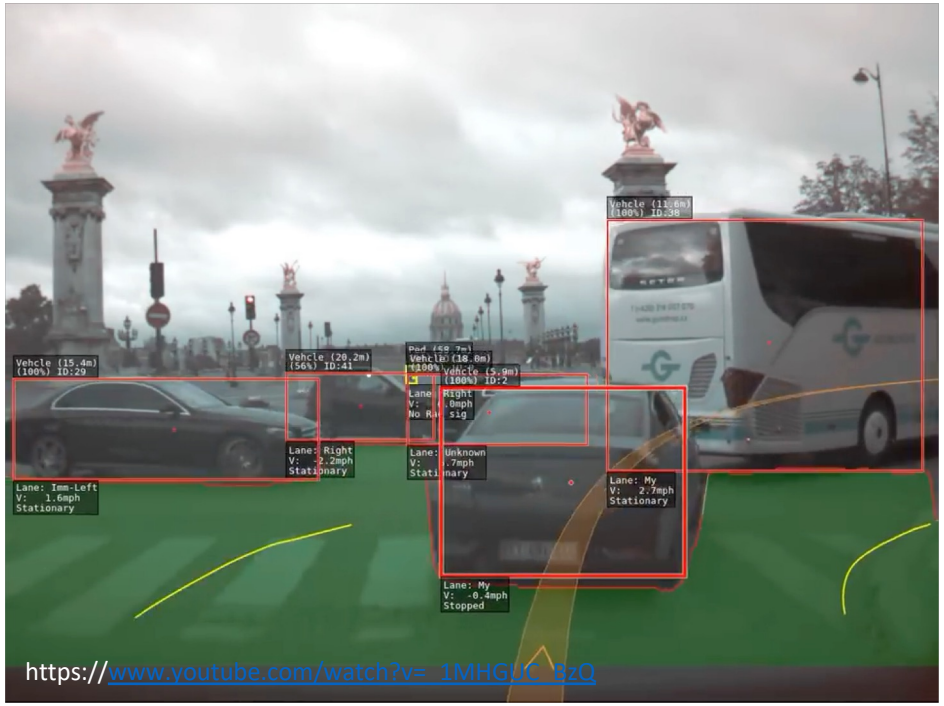


[Sharif et al., 2016]:
Glasses that fool face recognition

Why Is This Brittleness of ML a Problem?

→ Security

→ Safety

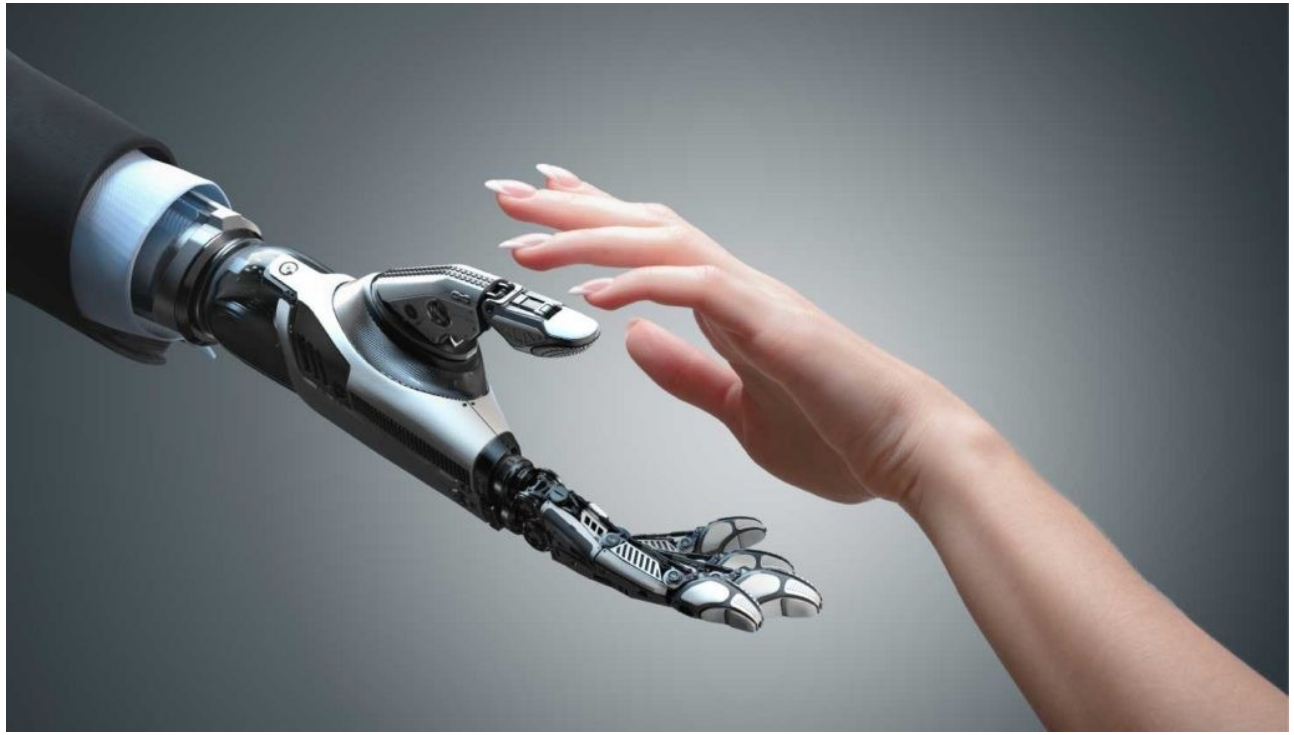


Why Is This Brittleness of ML a Problem?

→ Security

→ Safety

→ ML Alignment



Need to understand the
“failure modes” of ML

Adversarial attack

- Given an input image X and a label t, find $X' \approx X$ with $f(X') = t$

Dog

Hummingbird

This is a strong form of attack: for *any* X and *any* t, find X' with predicted class t

Targeted Attack vs. untargeted Attack

- Targeted Attack aims to misclassify the input to a specific target label
  - **Input**: a deep model $C_{\boldsymbol{\theta}}(.)$, an input $\boldsymbol{x}$, and a target class t
  - **Output**: A perturbation $\boldsymbol{\delta}$ such that $C_{\theta}(\boldsymbol{x}+\boldsymbol{\delta}) = t$

- Untargeted Attack aims to misclassify the input to any wrong label
  - **Input**: a deep model $C_{\boldsymbol{\theta}}(.)$ and an input $\boldsymbol{x}$
  - **Output**: A perturbation $\boldsymbol{\delta}$ such that $C_{\theta}(\boldsymbol{x}+\boldsymbol{\delta}) \neq C_{\theta}(\boldsymbol{x})$

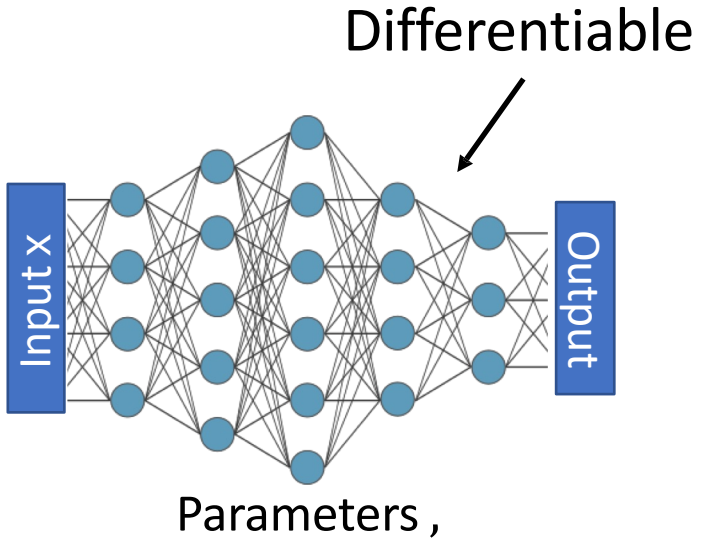
Where Do Adversarial Examples Come From?

To get an adv. example

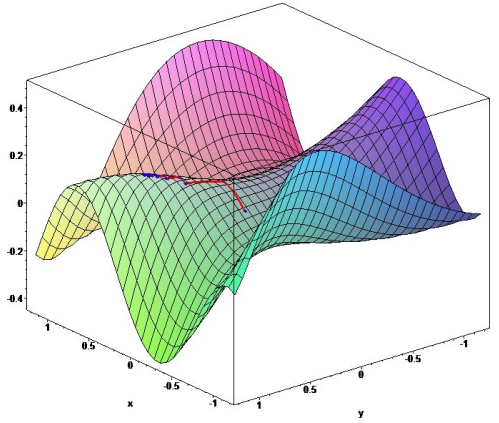
~~**Goal of training:**~~

Input Parameters Correct Label

$$min_\theta \; loss(f(x;\theta), y)$$



Can use gradient descent method to find good θ

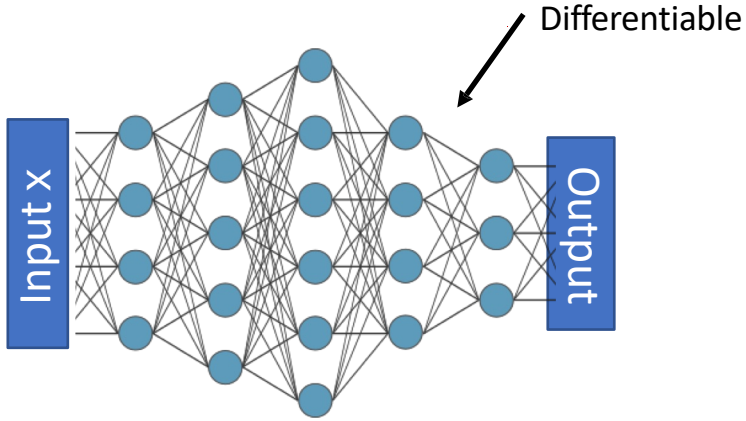


Where Do Adversarial Examples Come From?

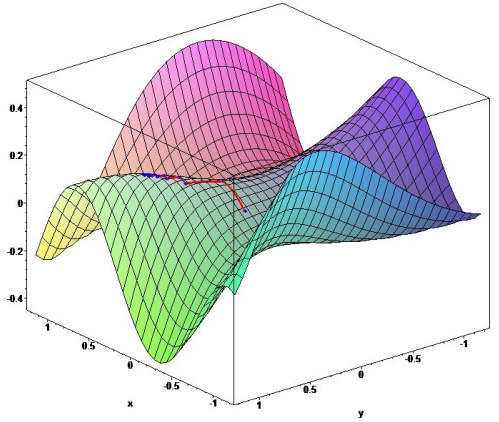
To get an adv. example

~~**Goal of training:**~~

$$max_{\delta \in \Delta} loss(f(x+\delta;\theta), y)$$



Can use gradient descent method to find good δ

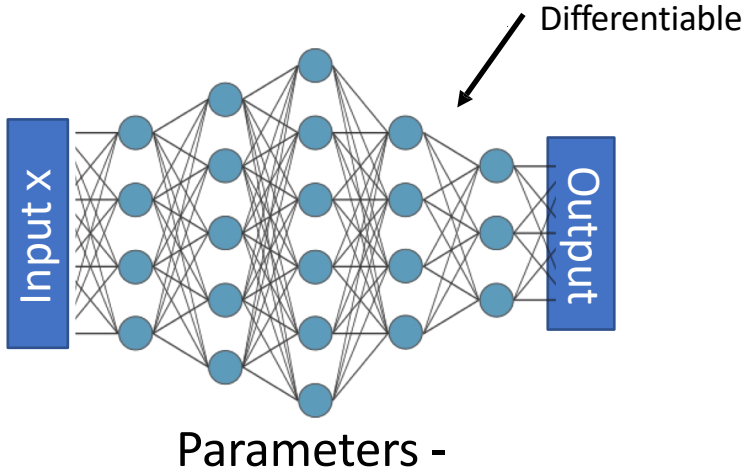


Where Do Adversarial Examples Come From?

To get an adv. example

~~Goal of training:~~

$$max_{\delta \in \Delta}\, loss(f(x+\delta;\theta), y)$$



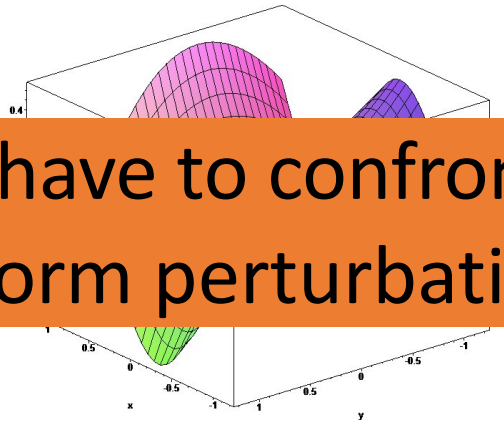
Which δ are allowed?

Examples: that is small wrt:

- L_P-norm
- Rotation and/or translation
- VGG feature perturbation

(add the perturbation you need here)

Can use gradient descent method to find good δ



Still: We have to confront (small) L_P norm perturbations

Fast Gradient Sign Method (FGSM)

- After training the network, for each (x,y), do one gradient step to increase $loss(\theta; x + \delta, y)$ (untargeted attack):

$$x_{adv} = x + \epsilon \, \text{sign}(\nabla_x loss(\theta; x + \delta, y))$$

or specifically consider the target class t (targeted attack):

$$x_{adv} = x - \epsilon \, \text{sign}(\nabla_x loss(\theta; x + \delta, t))$$

Attacks can be targeted or untargeted (the class of x_{adv} is given or not)

- L_∞ bound on the perturbation magnitude

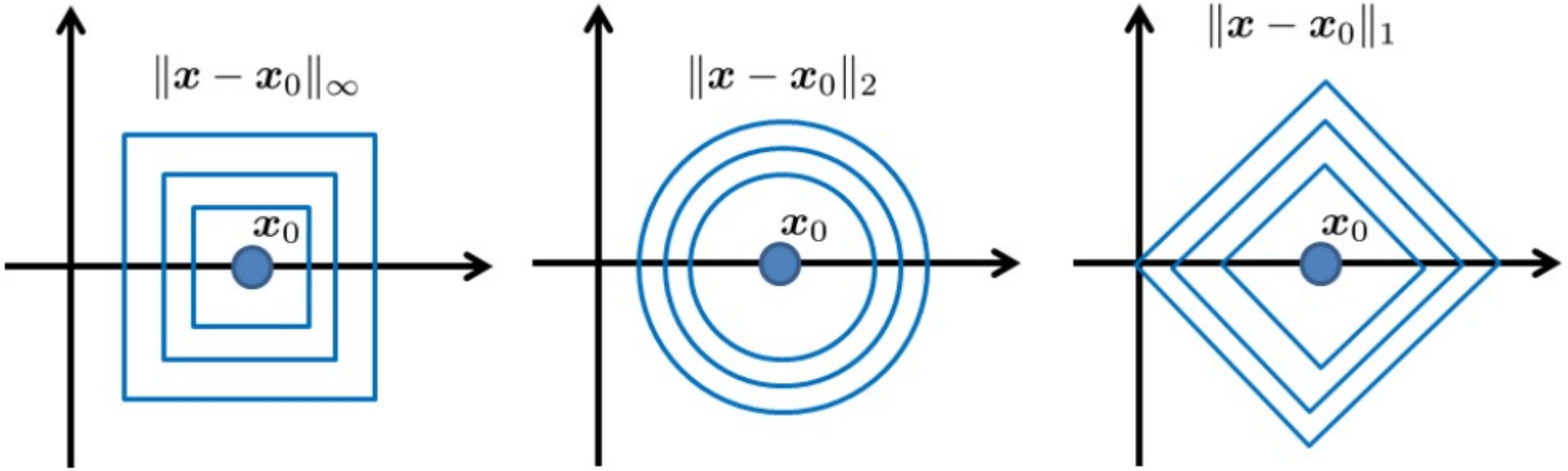
- It is 100% successful on IMAGENET even for small ϵ.

Goodfellow et al., Explaining and Harnessing adversarial examples, 2014

Human Alignment: Metrics used for robustness

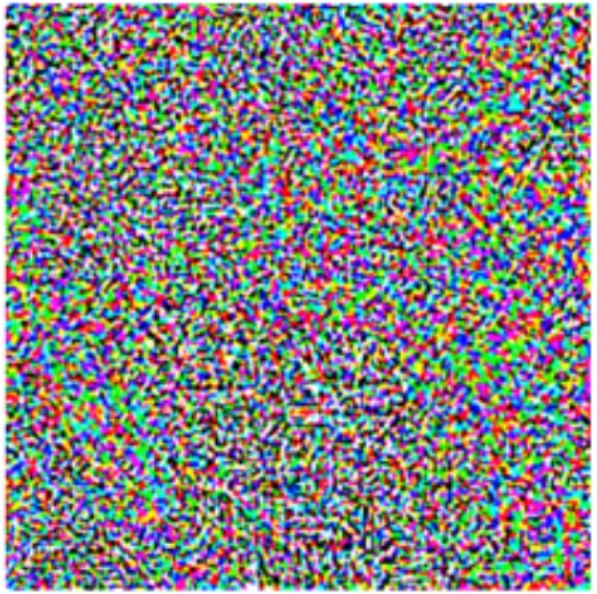
- Do metrics match human perceptibility?
  - L_0: how many pixels changed at all
  - L_1: total absolute pixel change
  - L_2: Euclidean distance
  - L_∞: max change of any single pixel

$$\|\boldsymbol{x} - \boldsymbol{x}'\|_p = \sqrt[p]{\sum_i (x_i - x_i')^p}$$



FGSM attack: Example

$+ .007 \times$



$=$

$\boldsymbol{x}$

"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

Goodfellow et al., Explaining and Harnessing adversarial examples, 2014

FGSM

- FGSM is a 1-step attack
- $x + \delta$ is guaranteed to stay inside the box $[x - \epsilon, x + \epsilon]$
- FGSM was designed to be fast, not optimal
  - may not compute minimal perturbation
  - is not a strong attack as shown later

Carlini & Wagner attack

- Basic idea: Given input x and desired label t, find adversarial input $x + \delta$

$$\min_{\boldsymbol{\delta}} D(\boldsymbol{x} + \boldsymbol{\delta}, \boldsymbol{x})$$
$$s.t.\ C(\boldsymbol{x} + \boldsymbol{\delta}) = t$$
$$\boldsymbol{x} + \boldsymbol{\delta} \in [0,1]^n$$

- Problem: This constraints in the optimization problem does not work well

- Solution: Define g such that $C(\boldsymbol{x} + \boldsymbol{\delta}) = t$ if and only if $g_t(\boldsymbol{x} + \boldsymbol{\delta}) \leq 0$

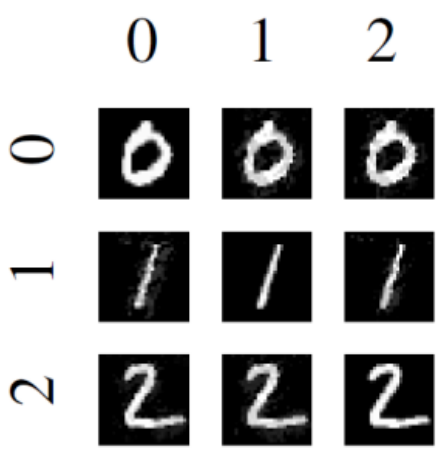
$$\min_{\boldsymbol{\delta}} D(\boldsymbol{x} + \boldsymbol{\delta}, \boldsymbol{x}) + \gamma g_t(\boldsymbol{x} + \boldsymbol{\delta})$$
$$s.t.\ \boldsymbol{x} + \boldsymbol{\delta} \in [0,1]^n$$

$$g_t(\boldsymbol{x}) = \left(\max_{i \neq t} f_i(\boldsymbol{x}) - f_t(\boldsymbol{x})\right)^+$$

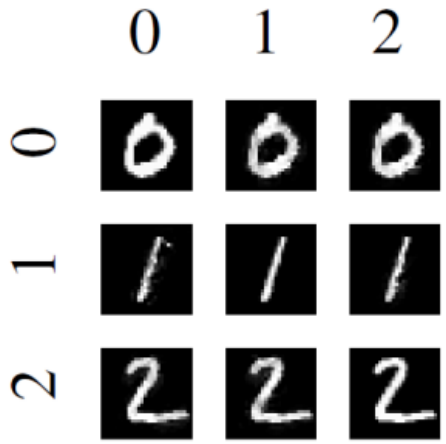
Carlini et al., Towards Evaluating the Robustness of Neural Networks, August 2016

Carlini & Wagner attack: Example

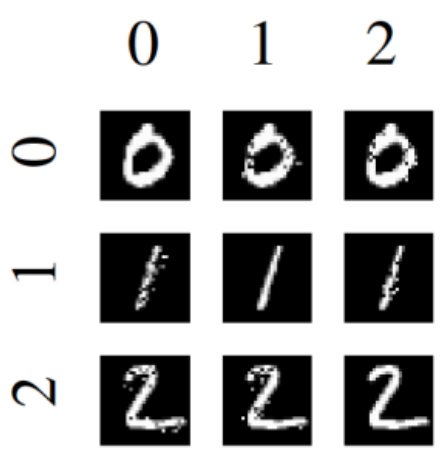
L_∞ attack



L_2 attack

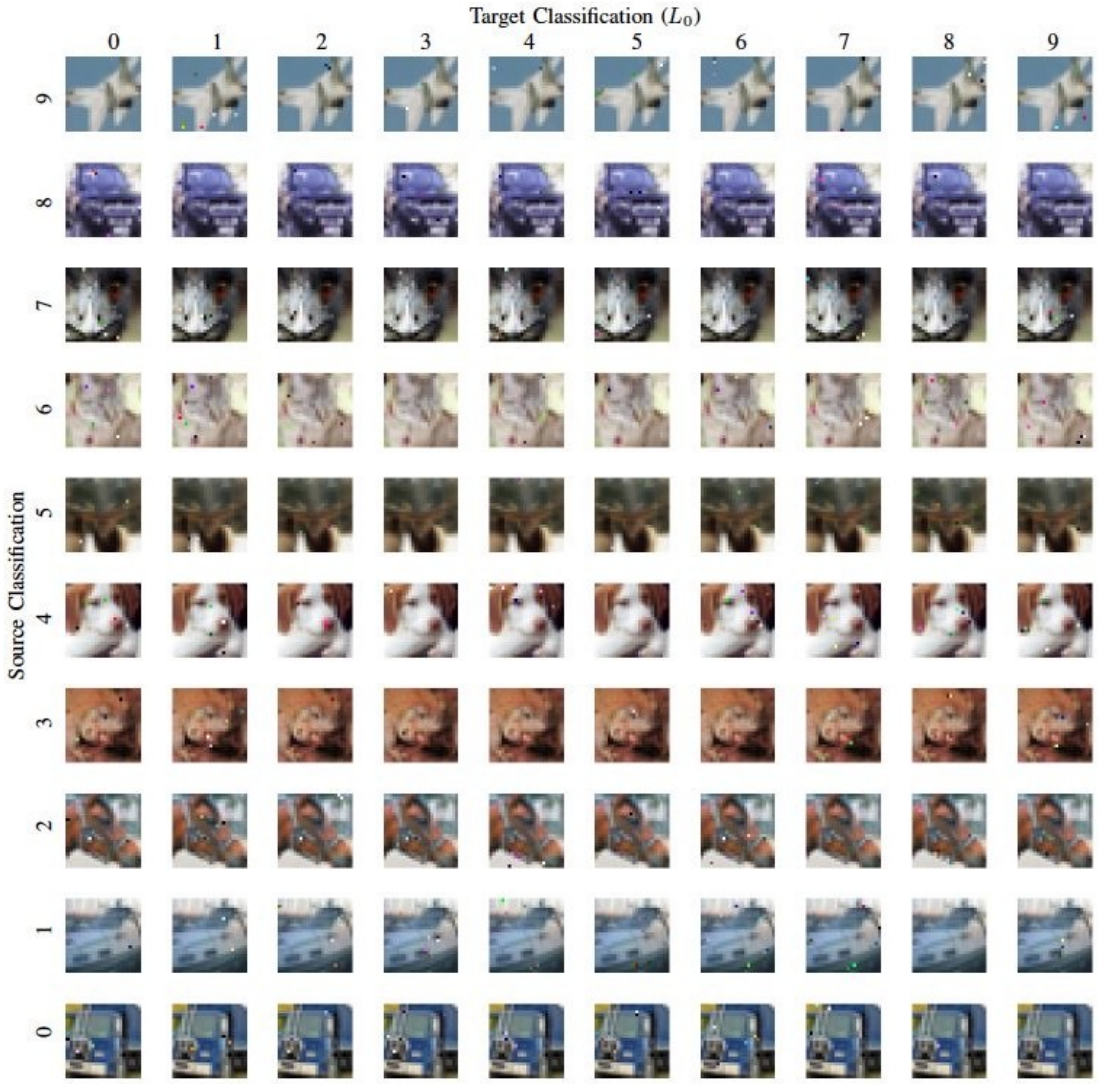


L_0 attack



Carlini et al., Towards Evaluating the Robustness of Neural Networks, August 2016

Carlini & Wagner attack: Example



Carlini et al., Towards Evaluating the Robustness of Neural Networks, August 2016

Projected gradient descent (PGD) attack

- It is an extension of FGSM

- After each step of perturbation, the adversarial example is projected back onto the ϵ-ball of x using a projection function Π:

$$x_{adv}^{t} = \Pi_{\epsilon}\left(\boldsymbol{x}_{adv}^{t-1} + \alpha \operatorname{sign} \nabla_{\boldsymbol{x}} loss\left(\boldsymbol{\theta}; \boldsymbol{x}_{adv}^{t-1} + \boldsymbol{\delta}, y\right)\right)$$

is smaller than ϵ

- PGD is regarded as the strongest L_∞ attack

Madry et al., Towards Deep Learning Models Resistant to Adversarial Attacks, 2017

Comparing attacks

- **Strongness:** Attack 1 is stronger than Attack 2 in the L_p distance if it has more ability to generate successful $x_{adv} \in B_p(x, \epsilon)$

- CW, PGD are the most powerful attacks.
  - There has been very slight improvements since then.

- L_0, L_2, L_∞ are generally imperceptible.

Attacks: Summary

Attack	Region	Optimization	Outcome
FGSM	Fixed to $\delta \in \{-\epsilon, \epsilon\}^n$	Take one ϵ-sized step	On the region boundary
PGD	Any region one can project to	Take many steps	Inside region
C&W	No restriction (only $x + \delta \in [0,1]^n$)	Want to produce δ with small L_∞. Takes many steps using LBFGS-B to ensure $x + \delta$ stays in $[0,1]^n$	Hopefully small $L_\infty(\delta)$

White-box vs. Black-box Attacks

- White box attacks: 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)

- Adversarial examples are transferrable:
  - Given the same training data as the original network
    - an attacker can train their own mirror network of the black box original network
    - can attack the mirror network with white-box techniques.
  - If attack on mirror network succeeds, it will likely succeed on the original

Robustness

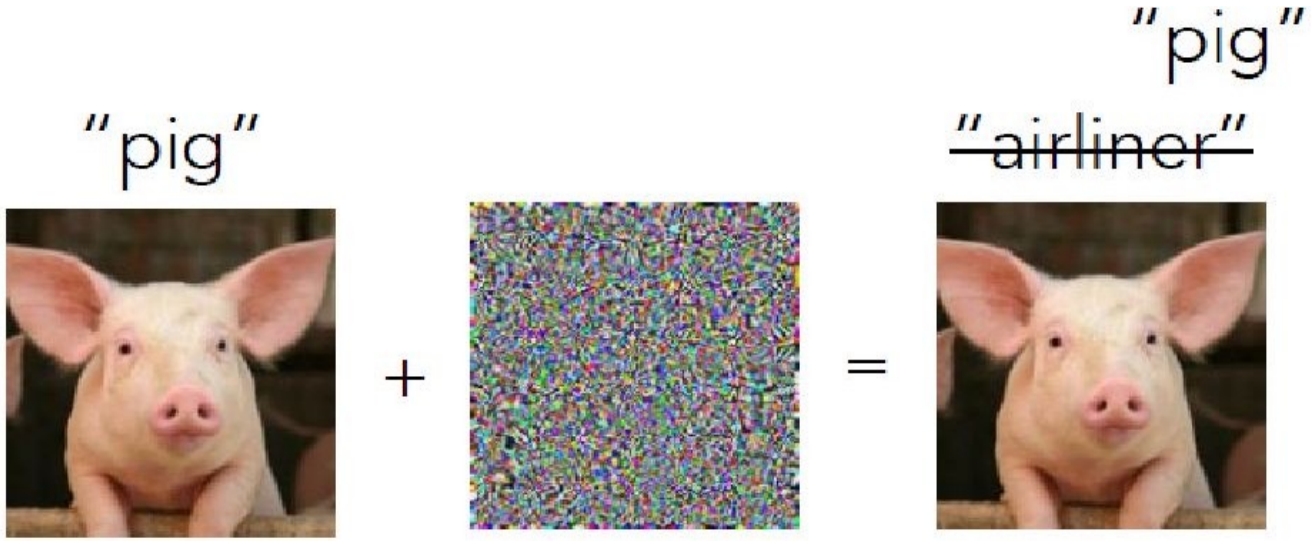
- Robustness: A network is robust if it returns correct output on all inputs
- Impractical: the input space is too large to be covered
- Local Robustness (informal): A learning model is locally-robust if it returns the correct output on inputs similar to inputs in training data

Accuracy vs. robustness tradeoff

- Accuracy: fraction of inputs which are correctly classified
- ϵ-robustness score: faction of inputs x such that for all $x' \in B_p(x, \epsilon)$, $class(x) = class(x')$
- Highly accurate models may have very low robustness scores

Systematic methods can increase robustness

- Robust training
  - Train so that small changes in input do not change the classifier output



Towards ML Models that Are Adv. Robust

Key observation: Lack of adv. robustness is **NOT** at odds with what we currently want our ML models to achieve

Standard generalization: $E_{(x,y)\sim D}[loss(\theta; x, y)]$

But: Adversarial noise is a "needle in a haystack"

Makelov, Schmidt, Tsipras, Vladu, 2018

Towards ML Models that Are Adv. Robust

Key observation: Lack of adv. robustness is **NOT** at odds with what we currently want our ML models to achieve

~~Standard~~ generalization: $E_{(x,y)\sim D}[\max_{\delta\in\Delta} loss(\theta; x+\delta, y)]$

Adversarially robust

But: Adversarial noise is a "needle in a haystack"

Makelov, Schmidt, Tsipras, Vladu, 2018

Use similar framework to train robustly

$$\min_{\theta} \mathbb{E}_{x,y\sim D} \left[\max_{\delta \in \Delta} loss(\theta, x + \delta, y) \right]$$

finding a robust model

finding a worst-case perturbation

Improve robustness: Train on perturbed inputs

(aka "adversarial training" [Goodfellow et al. 2015])

Actually leads to **robust models** (with some care)

Questions for discussion

- Do you think this goal captures “robustness” correctly?
  - A small change in input should not produce arbitrary large changes in output
- Are there other goals you associate with “robustness”?
- Does you think it is always better to build modules by perturbing input slightly, as shown in last few slides?
- What kind of input changes qualify as “slight perturbation”?

$$\max_{\delta \in \Delta} loss(\theta, x + \delta, y)$$

Allowed perturbations: pixel-wise, rotations, …

Adversarial Robustness Beyond Security

ML via Adversarial Robustness Lens

Overarching question:
How does adv. robust ML differ from "standard" ML?

$$E_{(x,y)\sim D}[loss(\theta; x, y)]$$

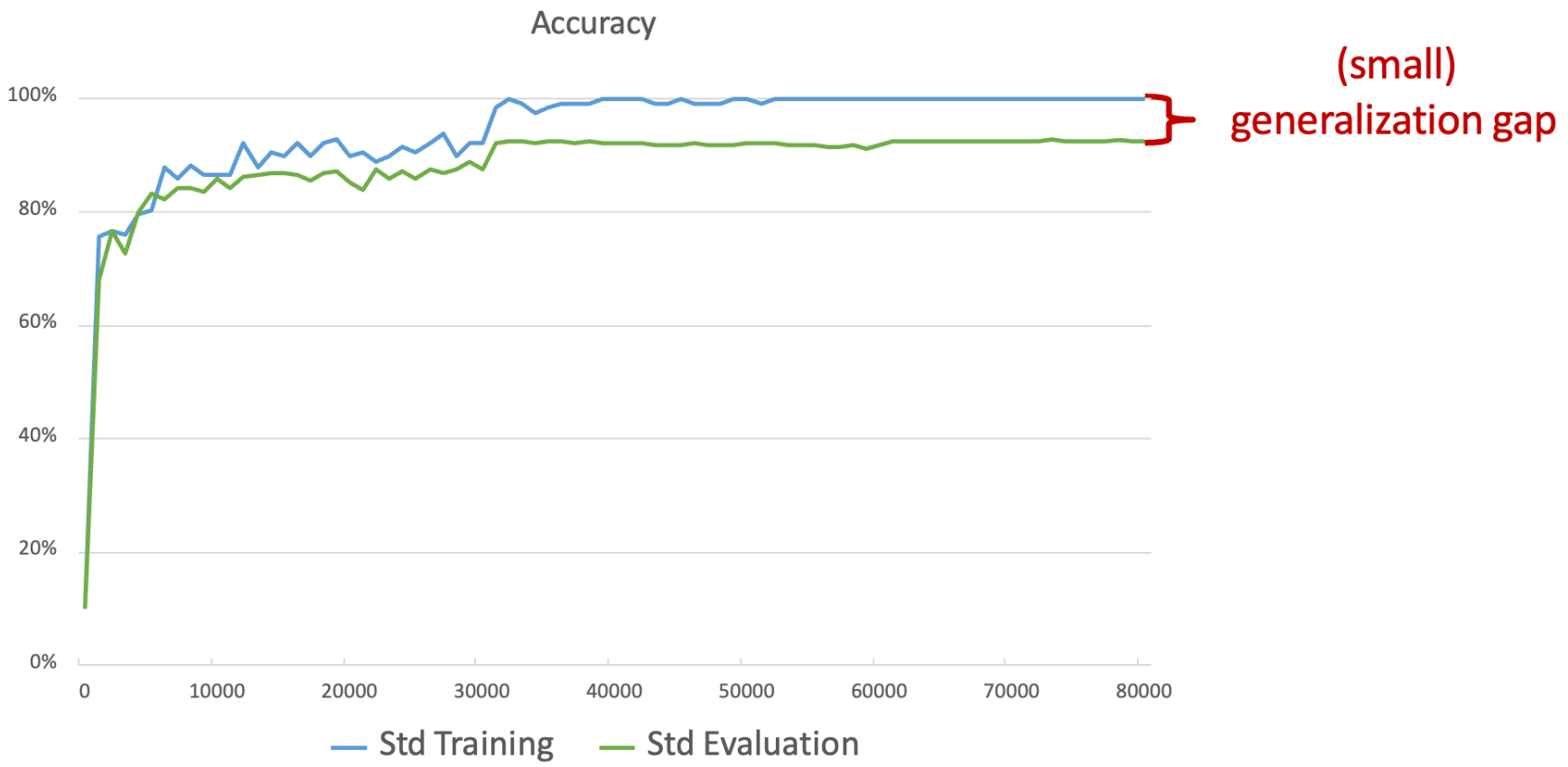
vs

$$E_{(x,y)\sim D}[\max_{\delta\in\Delta} loss(\theta; x+\delta, y)]$$

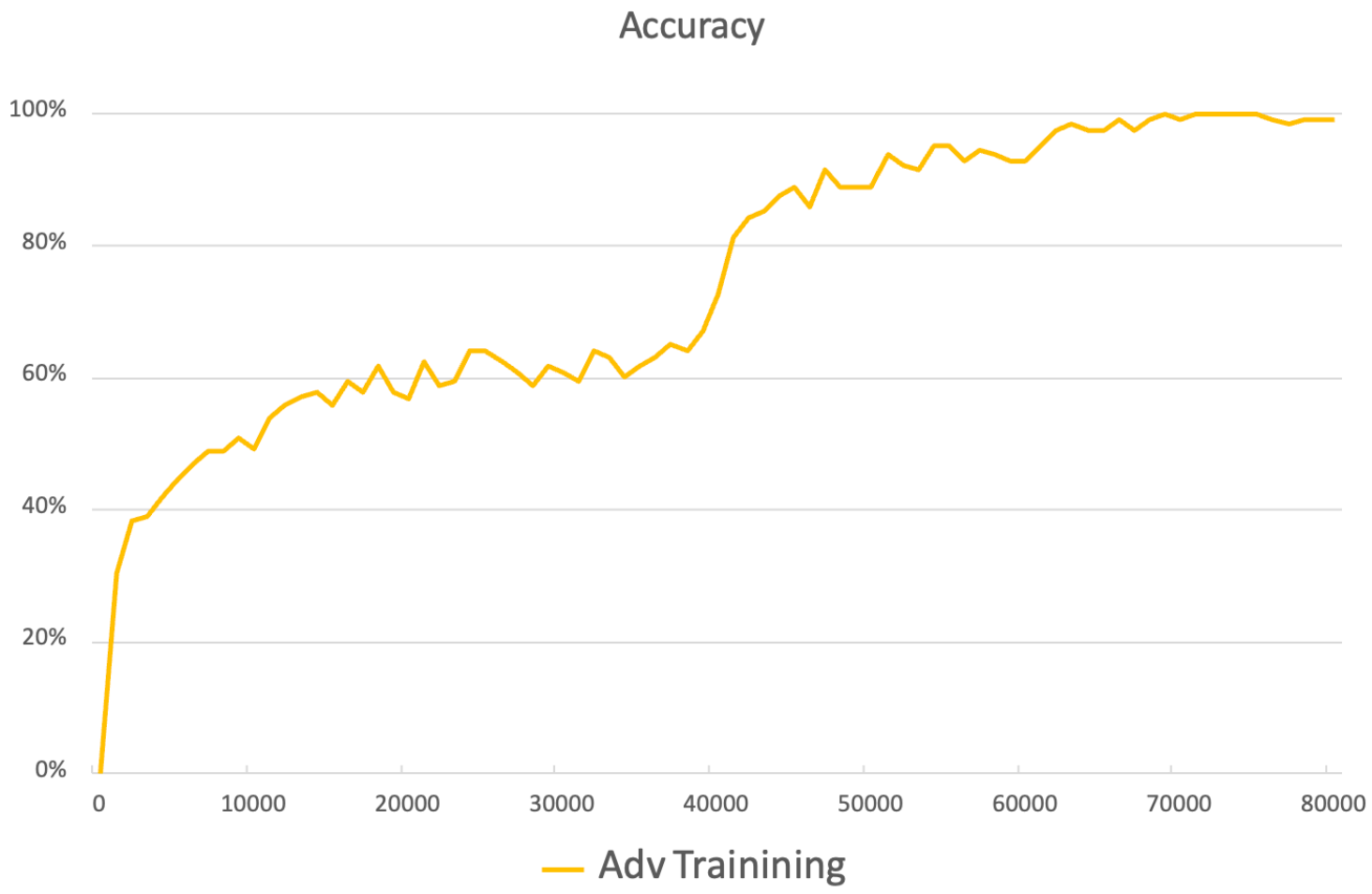
(This goes **beyond** deep learning)

Do Robust Deep Networks Overfit?

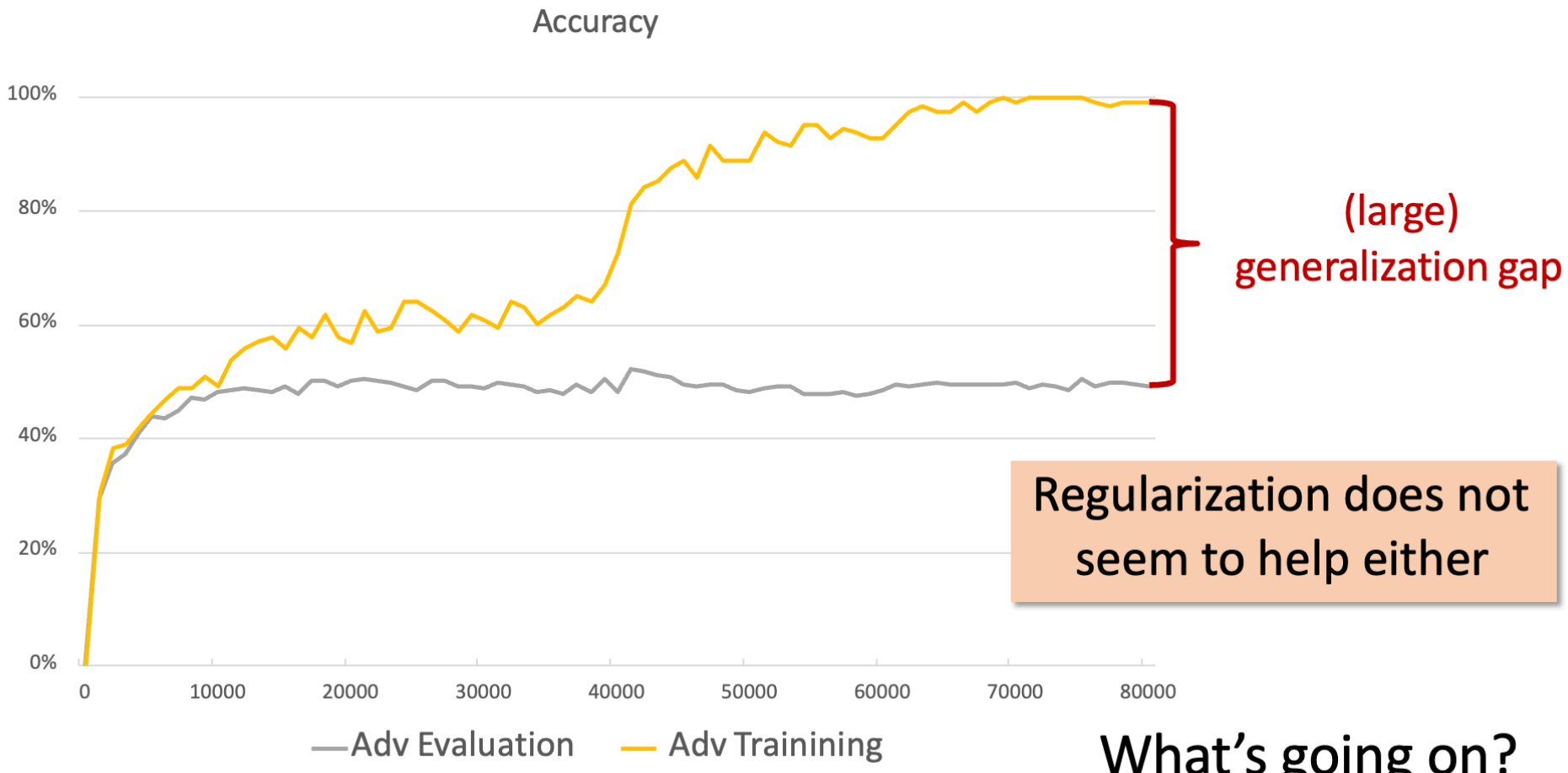
Do Robust Deep Networks Overfit?



Do Robust Deep Networks Overfit?



Do Robust Deep Networks Overfit?

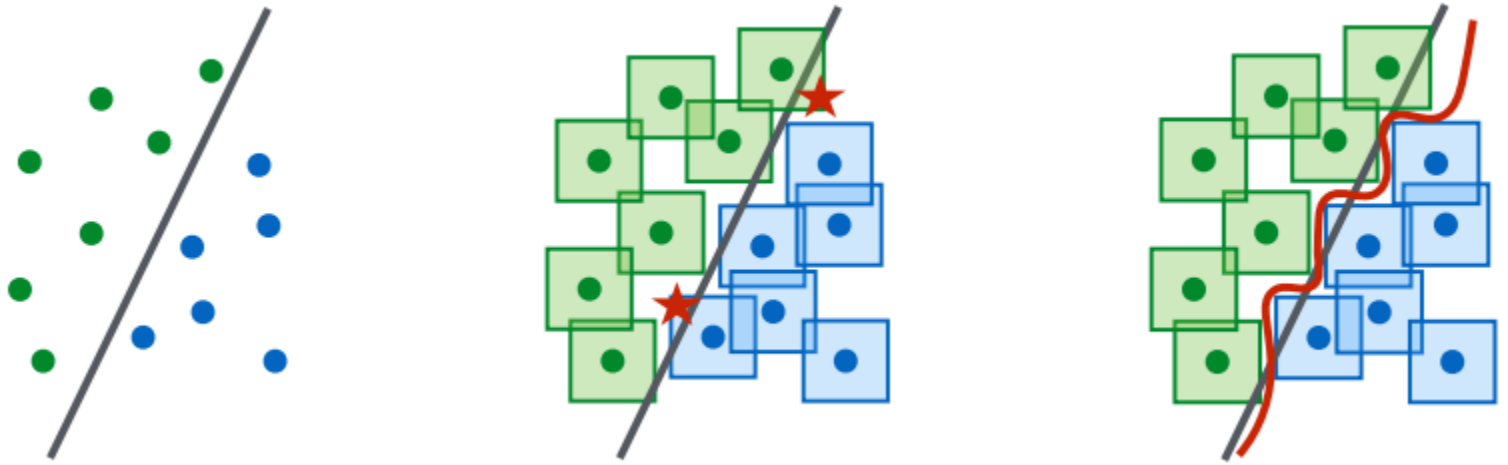


Regularization does not seem to help either

What's going on?

Adversarial robustness

- More capacity is required



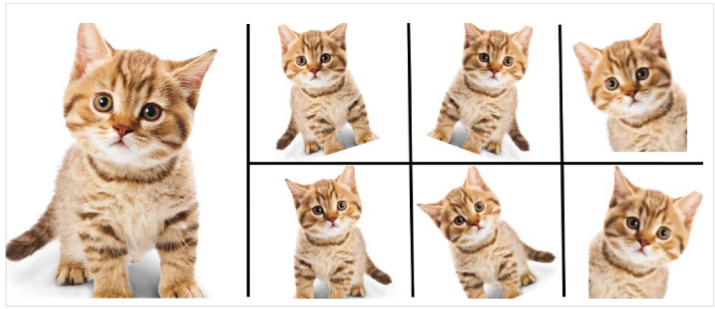
- More training data might be required

- Optimization during training may be more difficult

Madry et al., Towards Deep Learning Models Resistant to Adversarial Attacks, 2018

Does Being Robust Help "Standard" Generalization?

Data augmentation: An effective technique to improve "standard" generalization

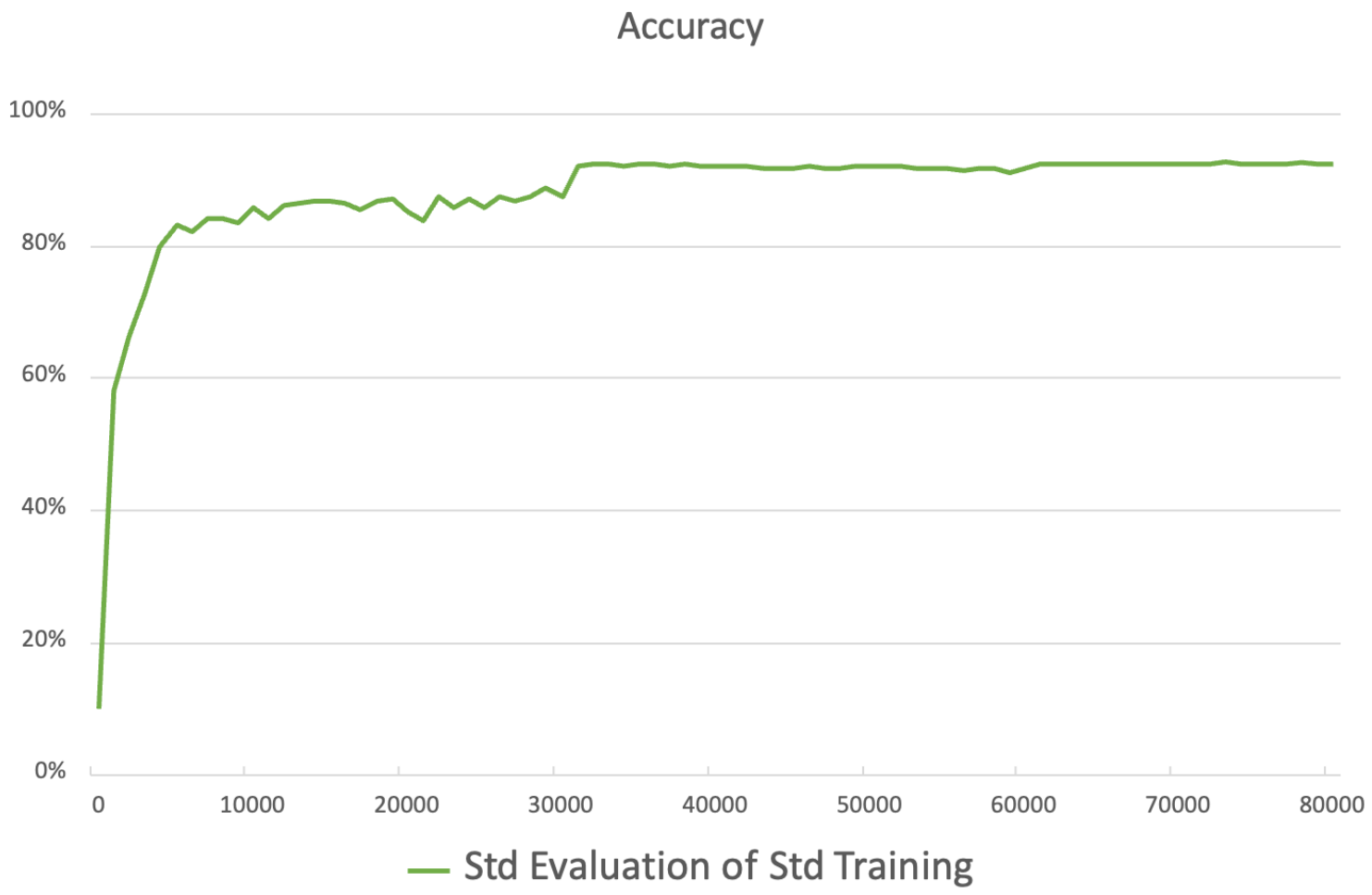


Adversarial training
=
An "ultimate" version of data augmentation?

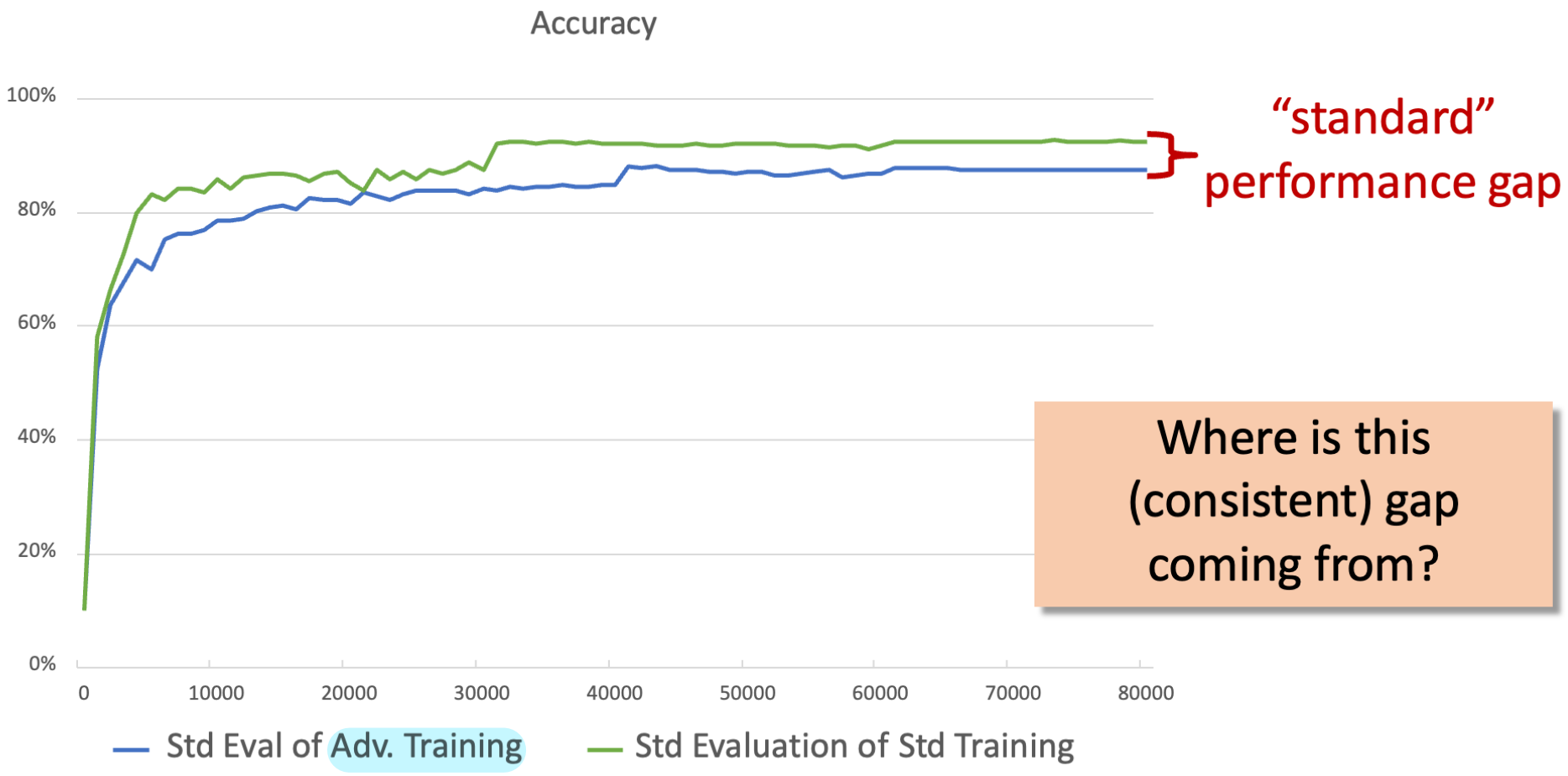
(since we train on the "most confusing" version of the training set)

Does adversarial training always improve "standard" generalization?

Does Being Robust Help “Standard” Generalization?



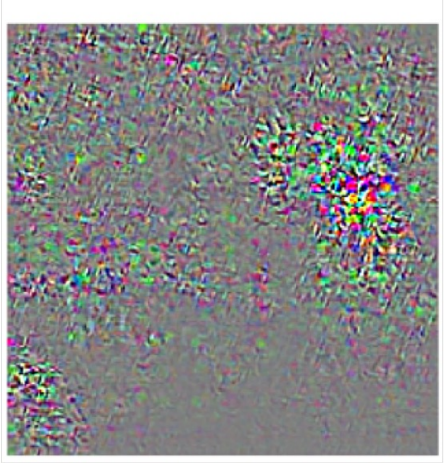
Does Being Robust Help “Standard” Generalization?



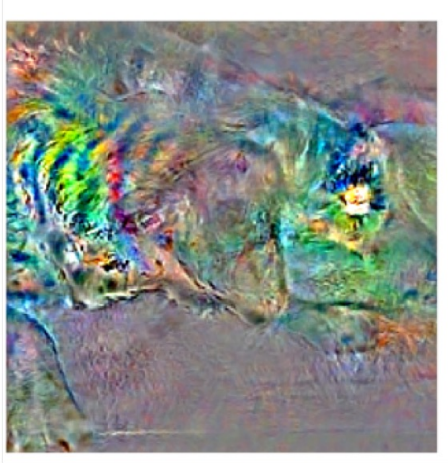
But There Are (Unexpected?) Benefits Too

Models become more **semantically meaningful**

Input



Gradient of
standard model

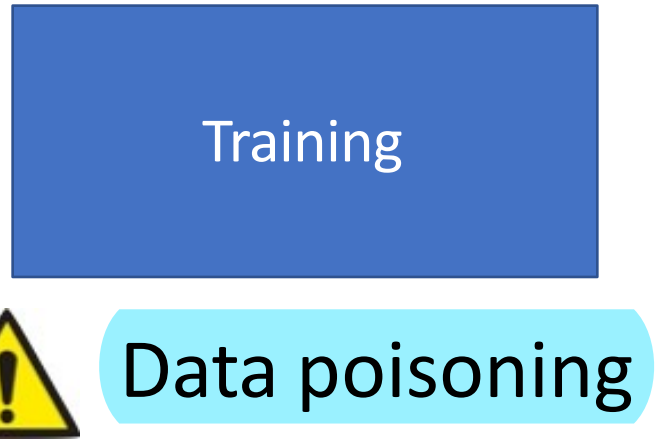


Gradient of
adv. robust model

[Tsipras et al., 2018]

What are other types of attacks?

Training time attack



Data poisoning

Adversarial Examples

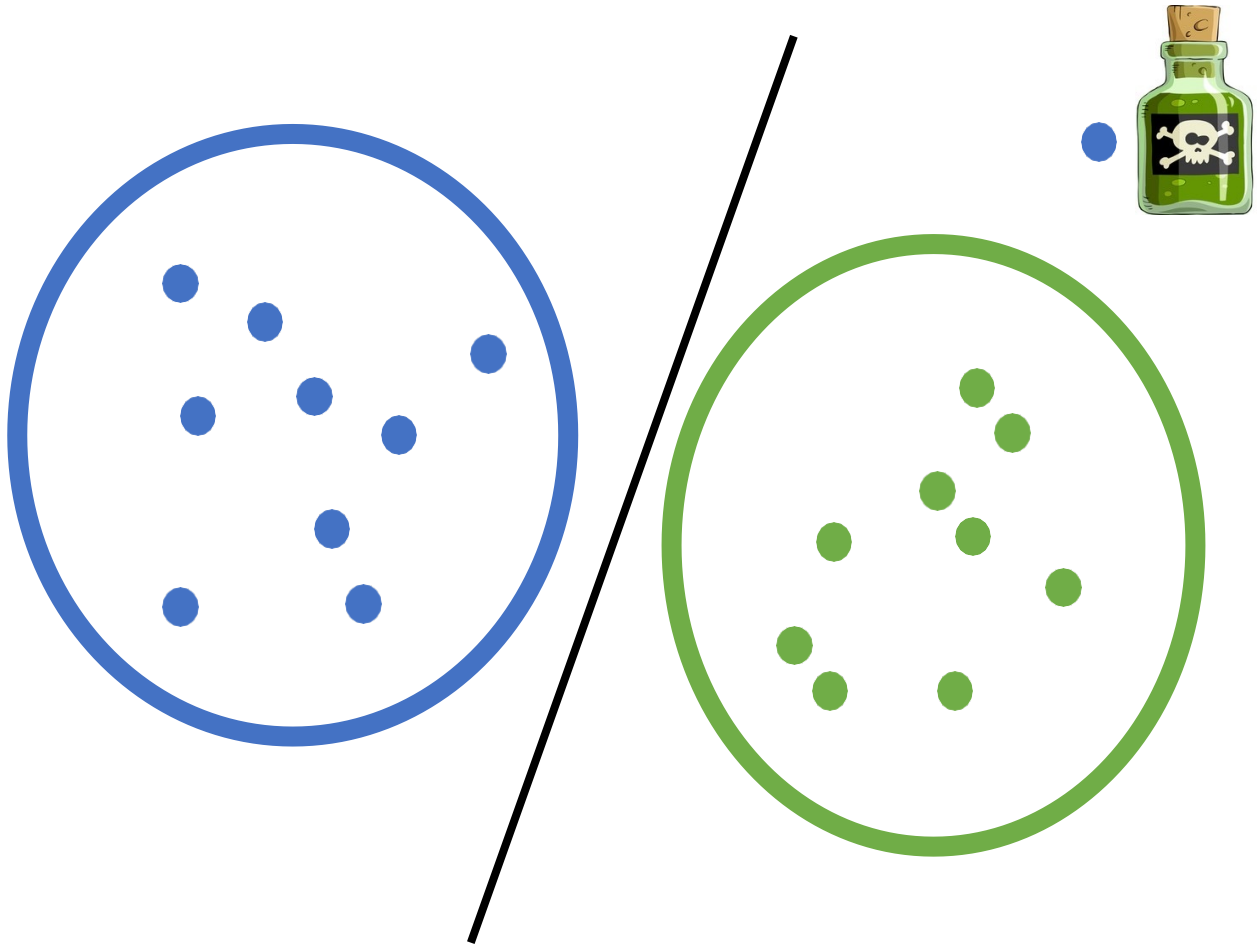
(Deep) ML is ”data hungry”

→ Can’t afford to be too picky about where we get the training data from

What can go wrong?

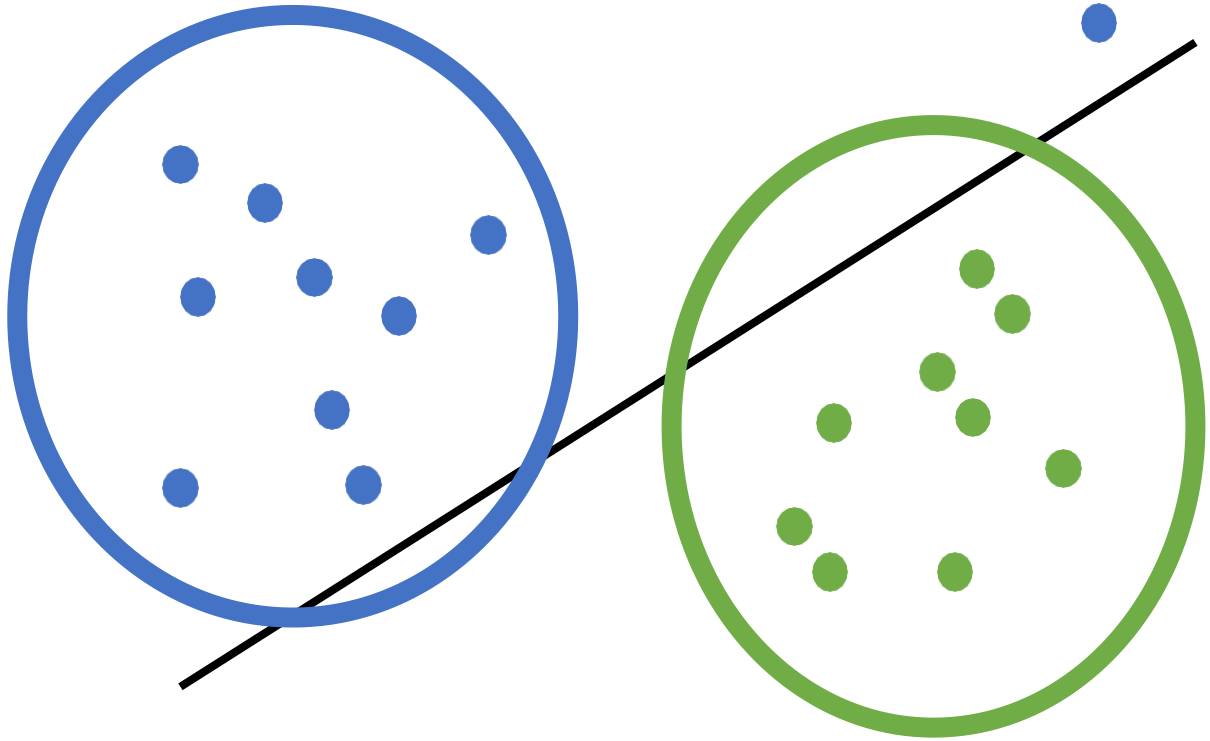
Data Poisoning

Goal: Maintain training accuracy but hamper generalization



Data Poisoning

Goal: Maintain training accuracy but hamper generalization



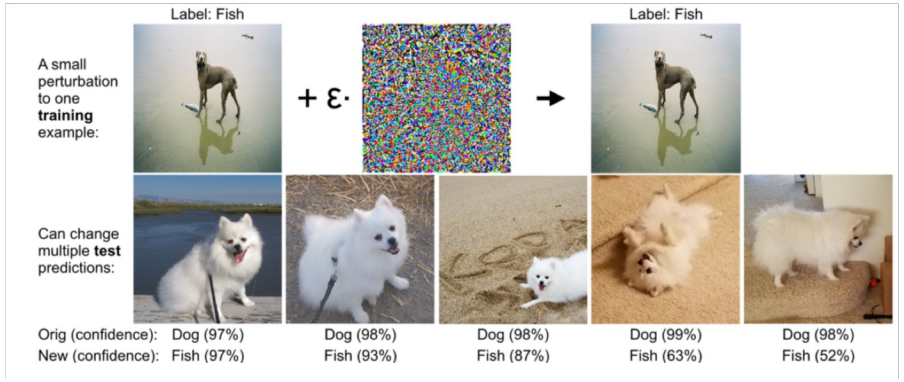
→ Fundamental problem
in “classic” ML (robust statistics)

→ **But:** seems less so in deep learning

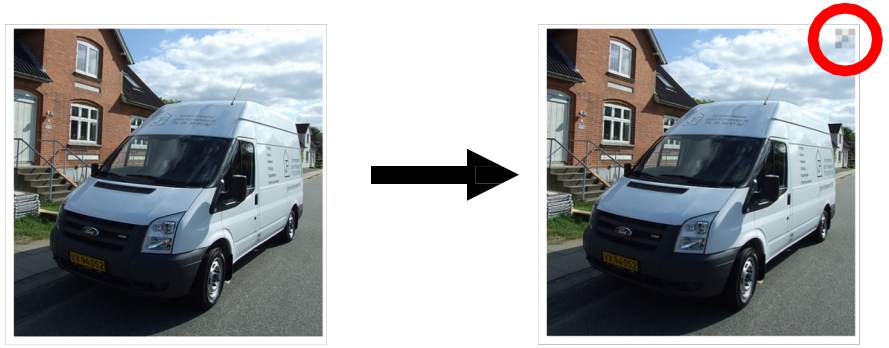
→ **Reason:** Memorization?

Data Poisoning

Goal: Maintain training accuracy but hamper generalization



[Koh Liang 2017]: Can manipulate many predictions with a single "poisoned" input



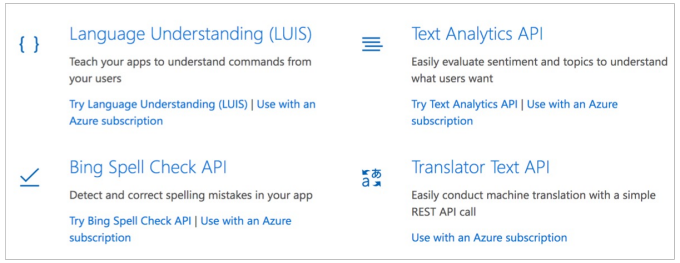
"van" "dog"

[Gu Dolan-Gavitt Garg 2017][Turner Tsipras M 2018]: Can plant an undetectable backdoor that gives an almost total control over the model

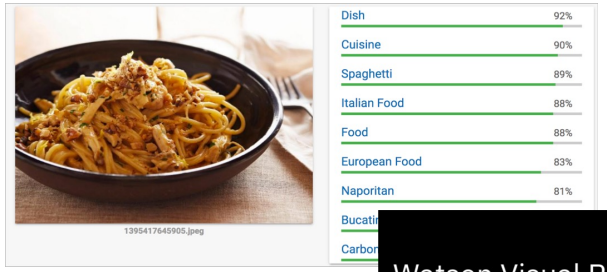
But: This gets (much) worse

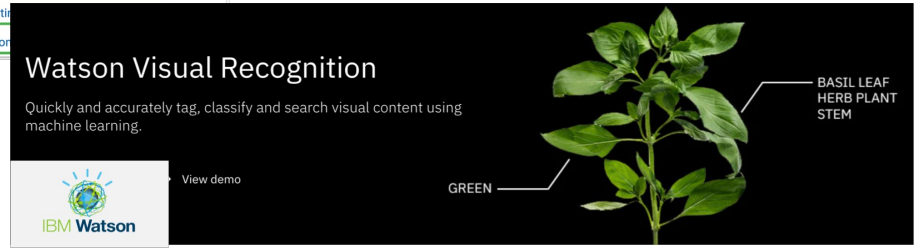
Model steeling

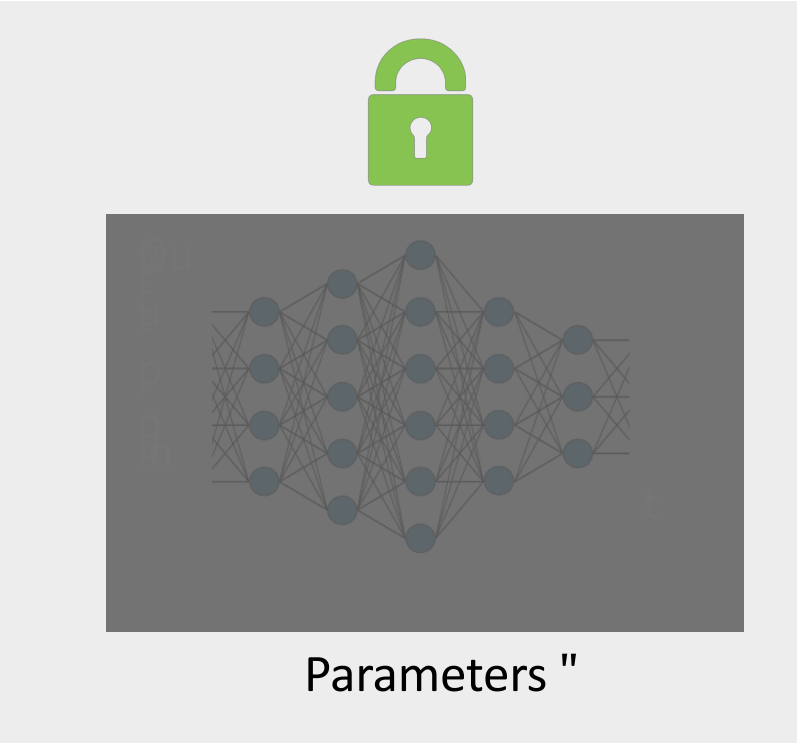
Microsoft Azure (Language Services)



Google Cloud Vision API







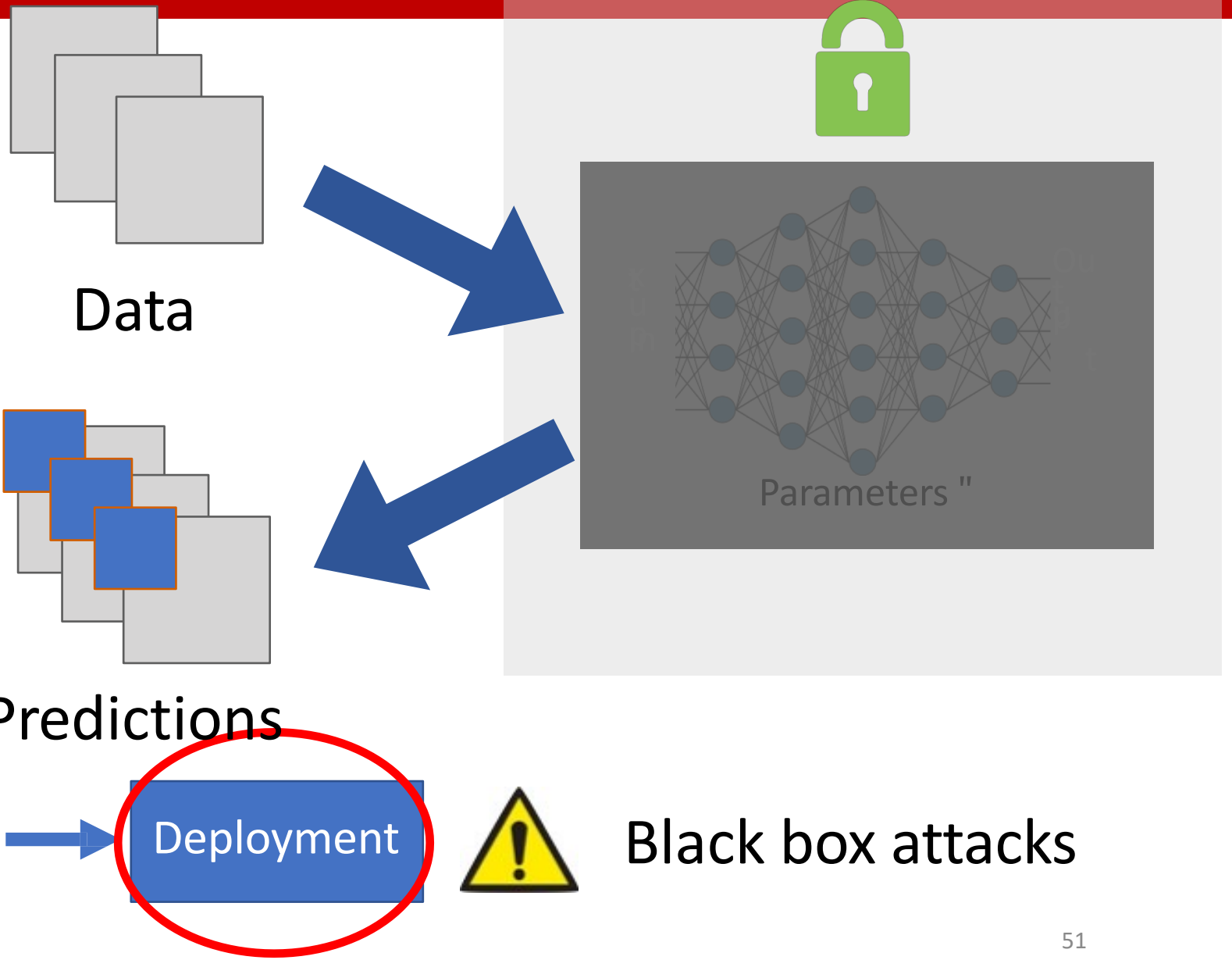
Parameters "

Training → Inference → Deployment

Model steeling

Does limited access give security?

In short: No

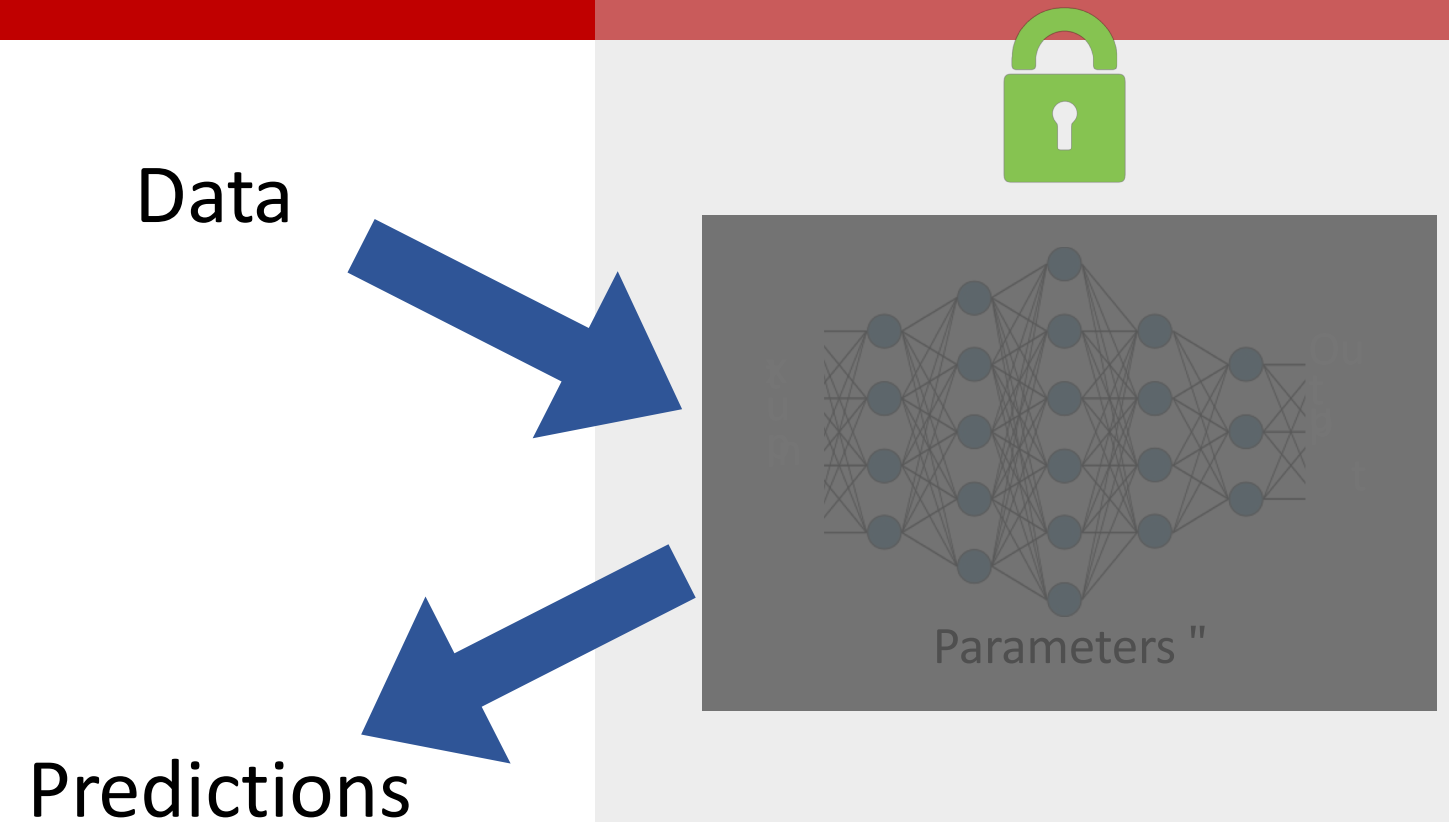


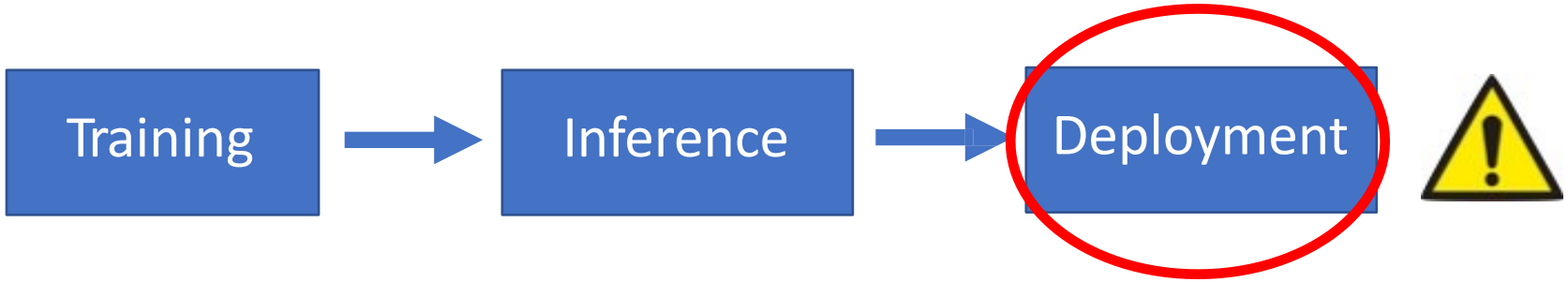
Black-box attacks

Does limited access give security?

Model stealing: "Reverse engineer" the model
[Tramer et al., 2016]

Black box attacks: Construct adv. examples from queries





Black box attacks

Three commandments of Secure/Safe ML

- shall not train on data you don't fully trust

  (because of data poisoning)

- shall not let anyone use your model (or observe its outputs) unless you completely trust them

  (because of model stealing and black box attacks)

- shall not fully trust the predictions of your model

  (because of adversarial examples)

Summary

- Machine learning models can be very brittle
  - A small change in the input can produce a very different model output
- Adversarial examples can be produced systematically by well-known attacks like FGSM, PGD, and C&W
- There are systematic methods to improve robustness
- There are limitations in the current state of the art/science
  - More robust models may be less accurate, depending on circumstances
  - Intuitive measures of “small change in input” are tricky - no clear metric for human perception
- Other types of threats for ML methods