Adversarial Examples

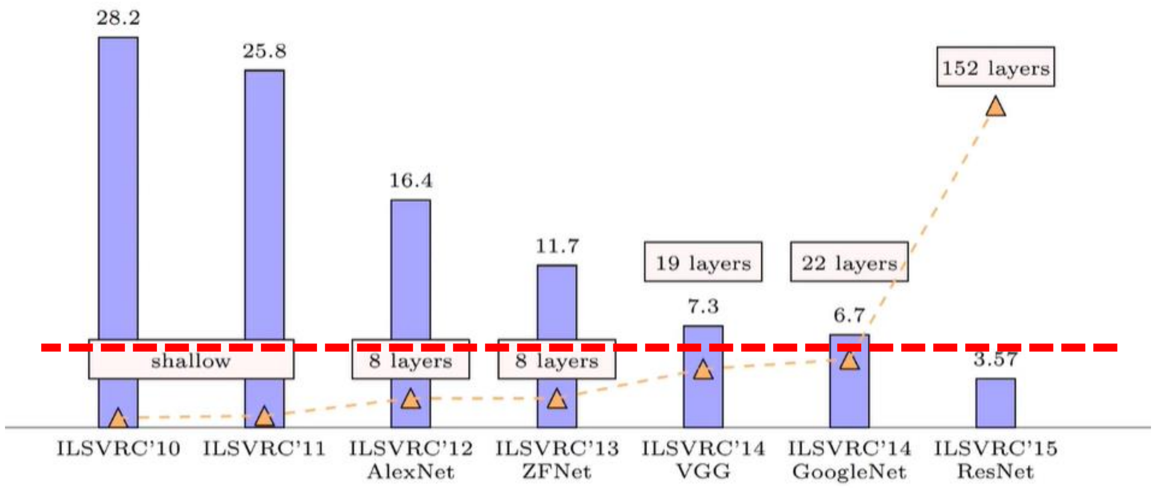
Designing, Visualizing and Understanding Deep Neural Networks

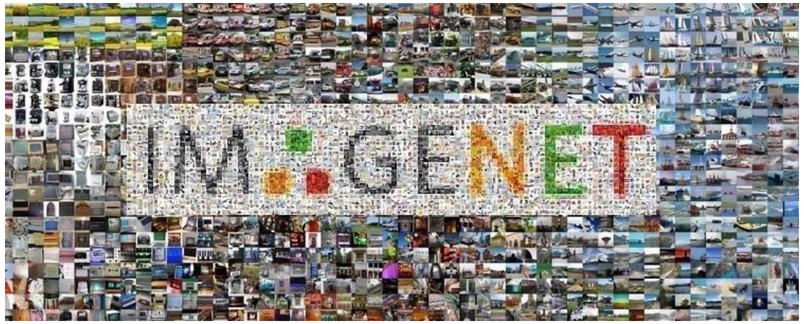
CS W182/282A

Instructor: Sergey Levine
UC Berkeley

Do deep nets generalize?

What a strange question!





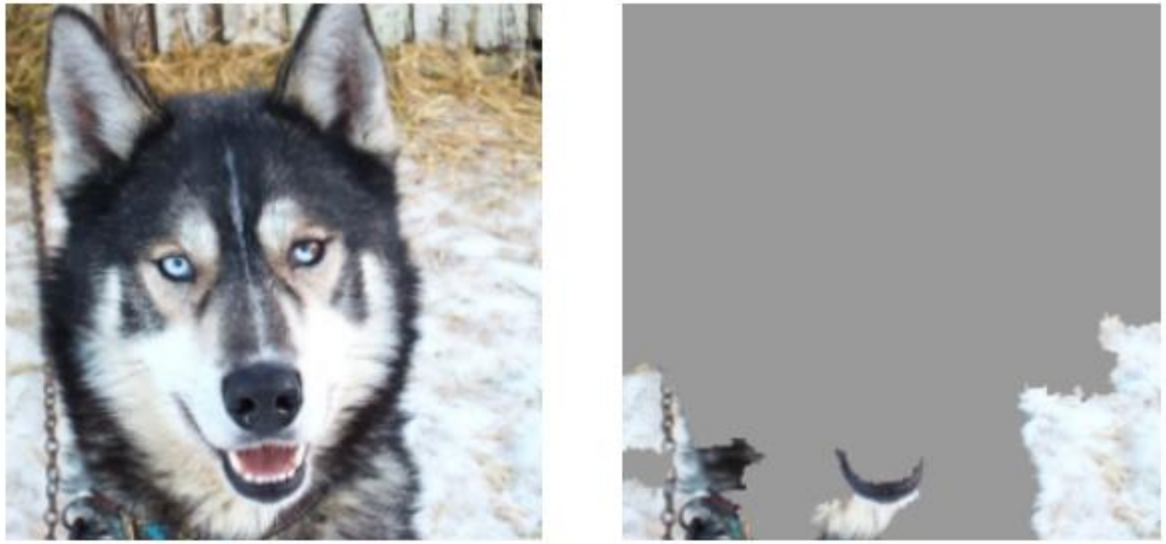
human performance:
about 5% error

but what about the **mistakes**? What kinds of mistakes are they?

Do deep nets generalize?

Even the mistakes make sense (sometimes)!

Do deep nets generalize?



(a) Husky classified as wolf (b) Explanation

Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

Ribeiro, Singh, Guestrin. **"Why Should I Trust You?" Explaining the Predictions of Any Classifier**. 2016.

A metaphor for machine learning…

Clever Hans

or: when the training/test paradigm goes wrong

Everything might be “working as intended”, but we might still not get what we want!

Example borrowed from Ian Goodfellow (“Adversarial Examples and Adversarial Training”)

Distribution shift

One source of trouble: the test inputs might come from a different distribution than training inputs

often especially problematic if the training data has spurious correlations

traffic sign classification dataset

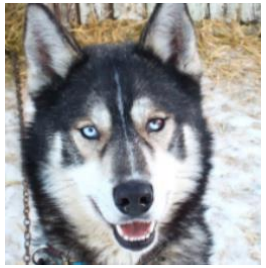
traffic sign in reality

Some more realistic examples:

- Medical imaging: different hospitals have different machines
- Even worse, different hospitals have different positive rates (e.g., some hospitals get more sick patients)
- Induces machine ⇔ label correlation
- Selection biases: center crop, canonical pose, etc.
- Feedback: the use of the ML system causes users to change their behavior, thus changing the input distribution
    - Classic example: spam classification

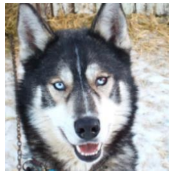
Calibration

Definition: the predicted probabilities reflect the actual frequencies of the predicted events

 →  → husky: 0.3
wolf: 0.7

"7 times out of 10 times, a person would say this is a wolf"

how is the data generated?

 →  →  → "husky"

$x \sim p(x)$ $y \sim p(y|x)$

Does this happen?

Usually not, such models typically give **confident** but **wrong** predictions on OOD inputs (but not always!)

why?

Are **in-distribution** predictions calibrated?

Usually not, but there are many methods for improving calibration

out of distribution:

 →  → husky: 0.5
wolf: 0.5

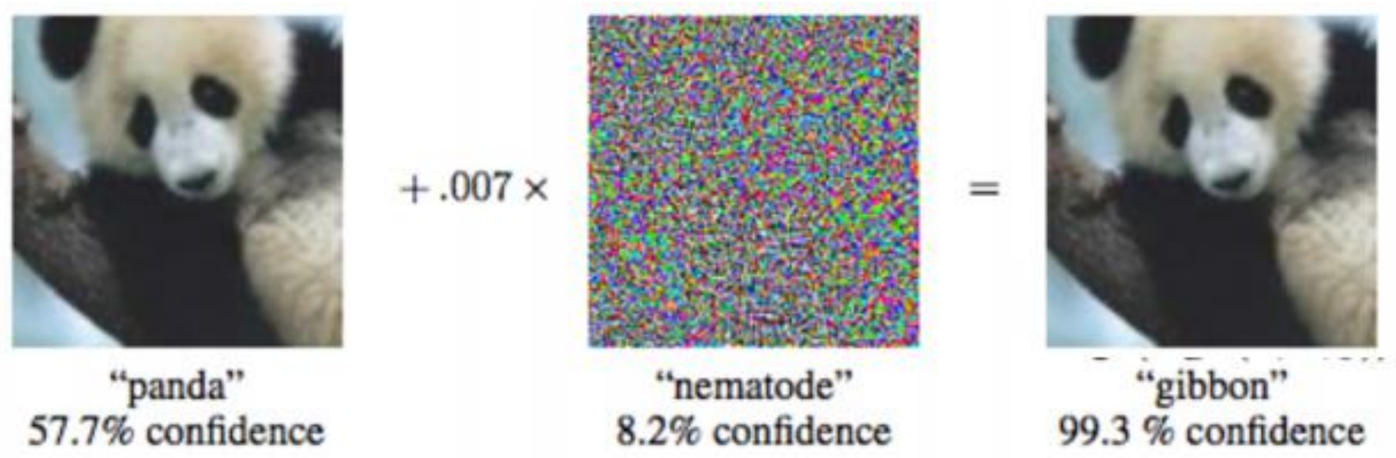
"I don't know what this is"

Adversarial examples

Adversarial examples

A particularly vivid illustration of how learned models may or may not generalize correctly

this is **not** random noise –
special pattern design to “fool”
the model



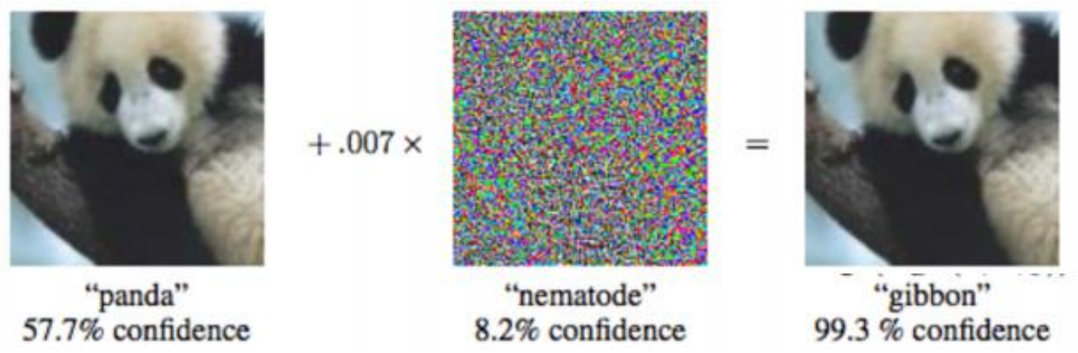
What’s going on here? very special patterns, almost imperceptible to people, can change a model’s classification drastically

Why do we care? **The direct issue:** this is a potential way to “attack” learned classifiers
The bigger issue: this implies some strange things about generalization

Some facts about adversarial examples

We'll discuss many of these facts in detail, but let's get the full picture first:

- It's not just for gibbons. Can turn basically **anything** into **anything else** with enough effort
- It is **not** easy to defend against, obvious fixes can help, but nothing provides a bulletproof defense (that we know of)
- Adversarial examples can transfer across different networks (e.g., the same adversarial example can fool both AlexNet and ResNet)
- Adversarial examples can work in the real world, not just special and very precise pixel patterns
- Adversarial examples are **not** specific to (artificial) neural networks, virtually all learned models are susceptible to them



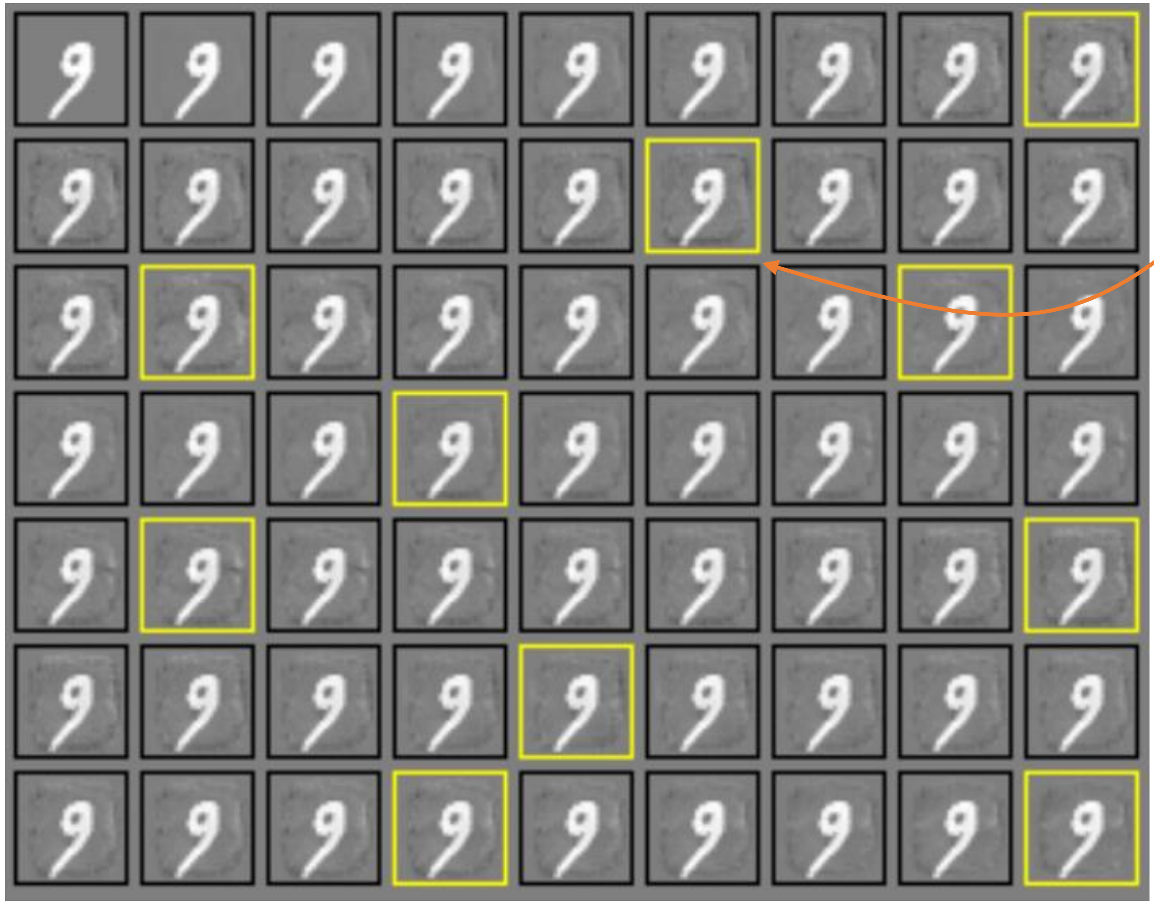
speed limit: 45

photo is not altered in any way!

but the sign is

including your brain, which is a type of learned model

A problem with deep nets?



classified as “0” (90%)

classified as “1” (90%)

linear model (logistic regression)

adversarial examples appear to be a general phenomenon for most learned models (and all high-capacity models that we know of)

Example from: Ian Goodfellow, 2016

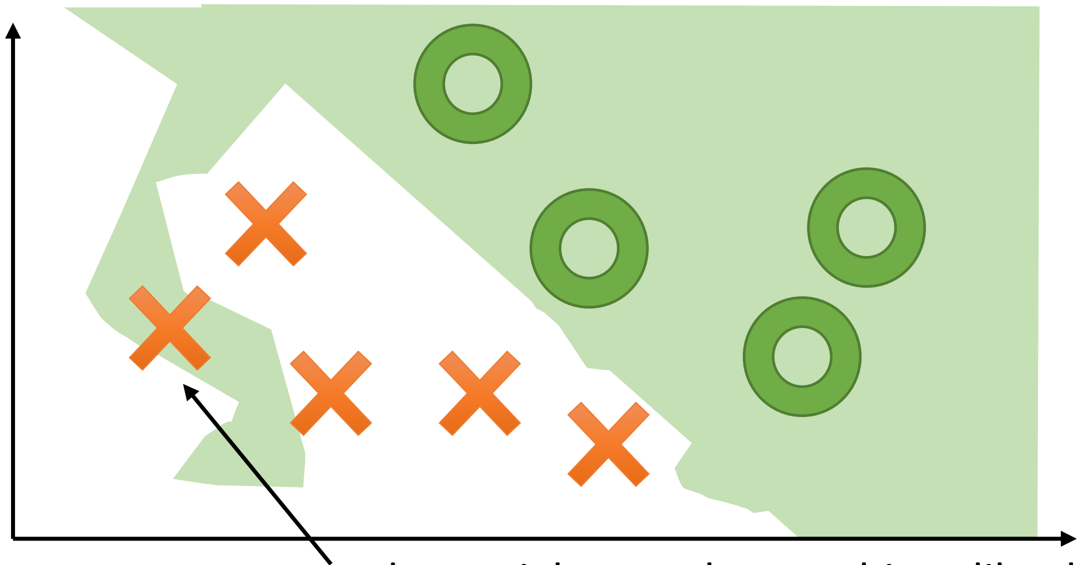
Is it due to **overfitting**?

Overfitting hypothesis: because neural nets have a huge number of parameters, they tend to overfit, making it easy to find inputs that produce crazy outputs

Implication: to fix adversarial examples, stop using neural nets

most evidence suggests that this hypothesis is false

The mental model:



adversarial examples are things like this

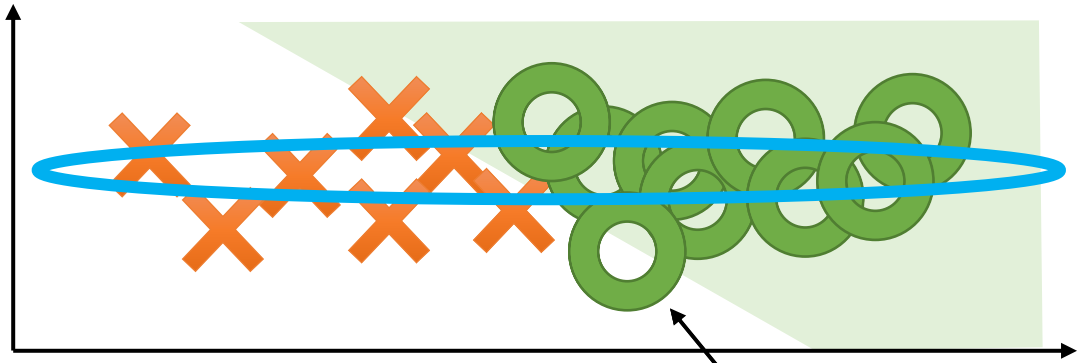
- If this were true, we would expect different models to have very different adversarial examples (high variance)
  - This is conclusively not the case
- If this were true, we would expect low capacity models (e.g., linear models) not to have this issue
  - Low capacity models also have this
- If this were true, we would expect highly nonlinear decision boundaries around adversarial examples
  - This appears to not be true

Slide based on material from Ian Goodfellow (2017)

Linear models hypothesis

Linear models hypothesis: because neural networks (and many other models!) tend to be locally linear, they extrapolate in somewhat counterintuitive ways when moving away from the data

this has a somewhat counterintuitive meaning in high dimensions

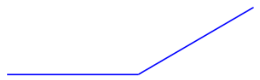


“realistic images” manifold

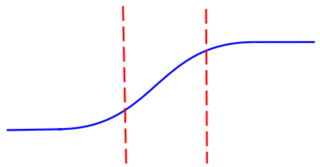
adversarial examples are things like this

why so linear?

Rectified linear unit



Carefully tuned sigmoid

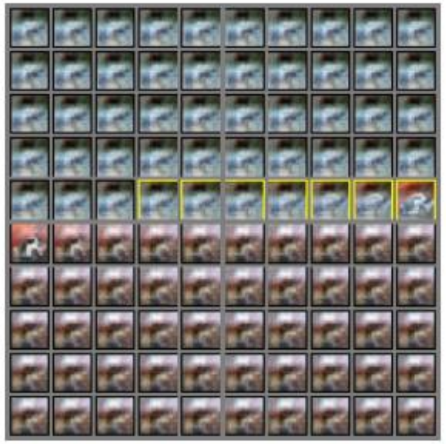


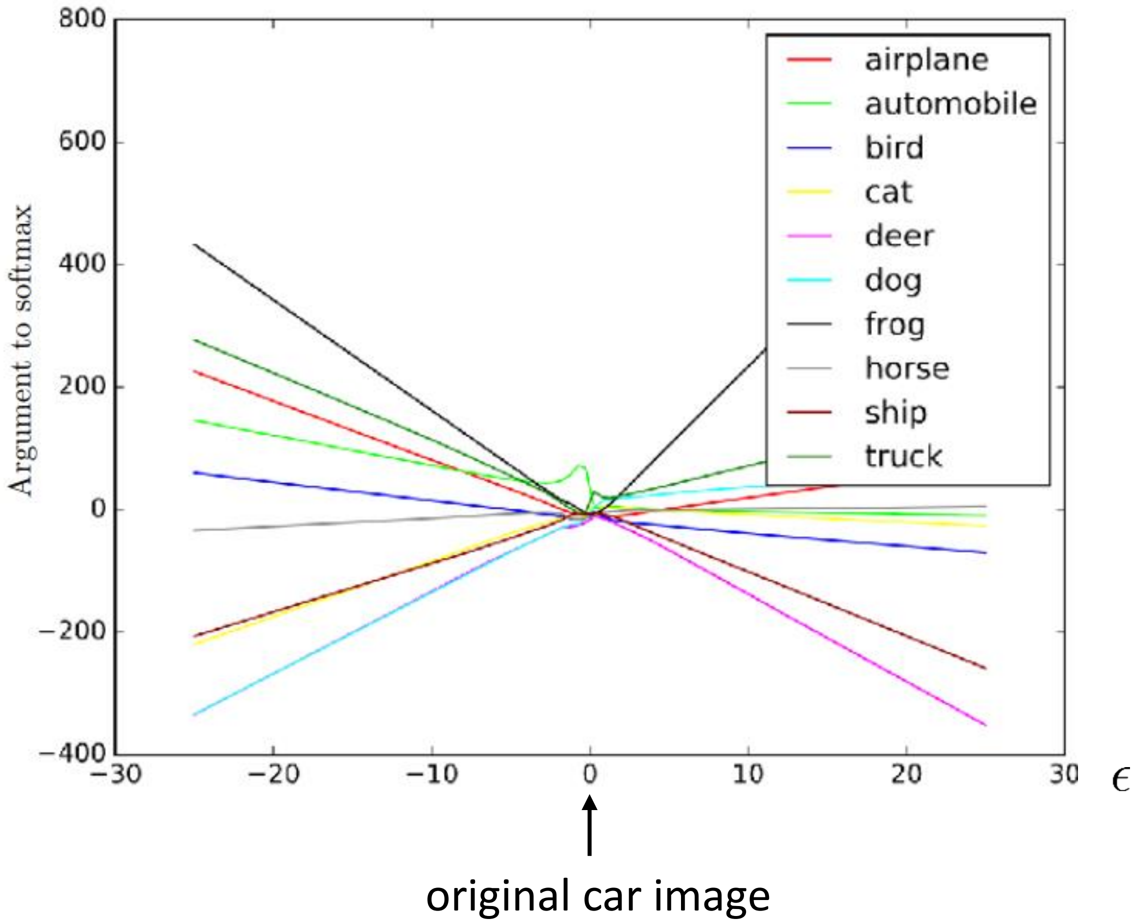
- Consistent with transferability of adversarial examples
- Reducing “overfitting” doesn’t fix the problem

Slide based on material from Ian Goodfellow (2017)

Linear models hypothesis

Experiment 1: vary images along one vector, and see how predictions change

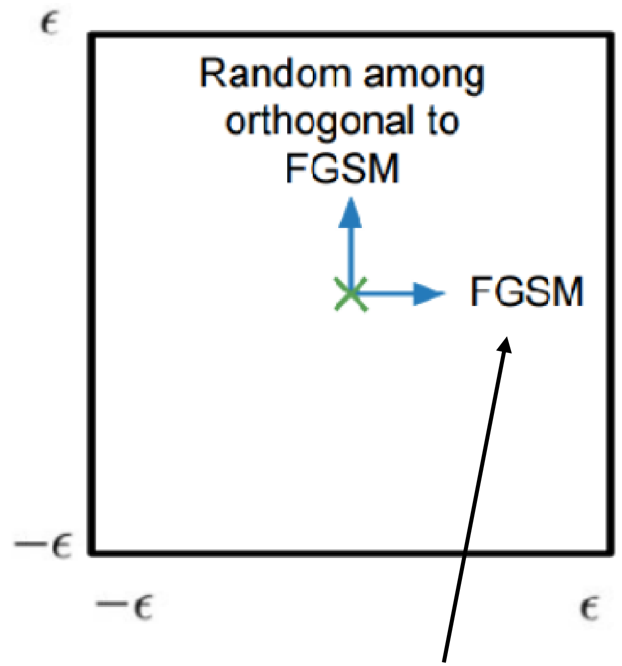




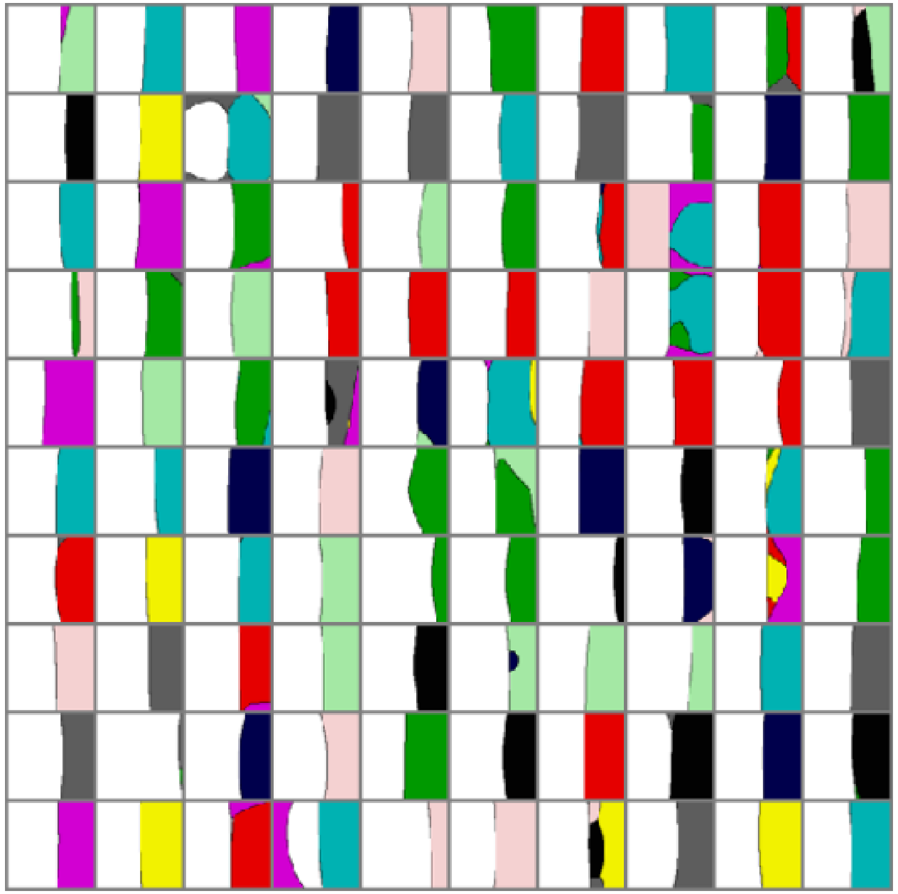
Slide based on material from Ian Goodfellow (2017)

Linear models hypothesis

Experiment 2: vary images along two directions: an adversarial one, and a random one



adversarial direction

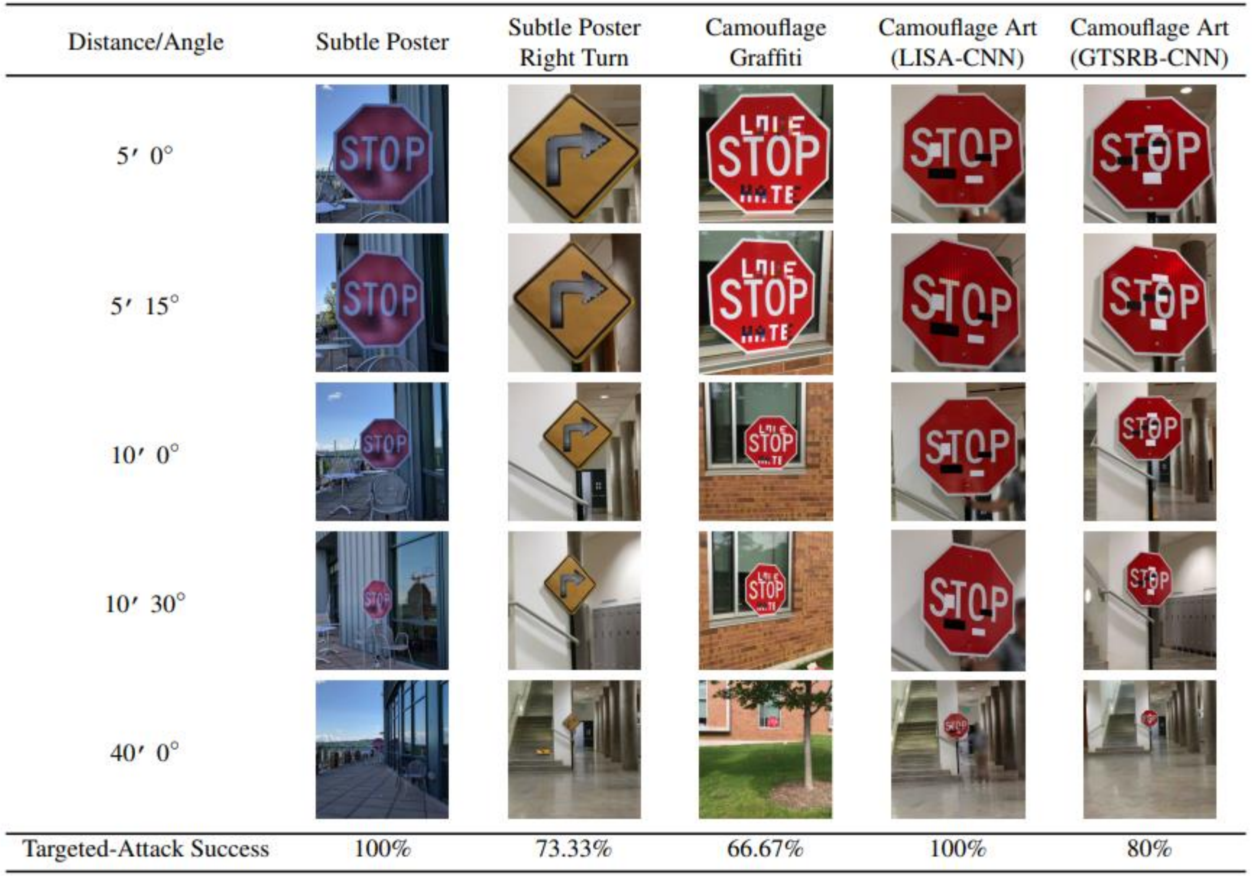


not much variation **orthogonal** to adversarial direction

clean “shift” on **one side** for adversarial direction, suggesting a mostly linear decision boundary

Slide based on material from Ian Goodfellow (2017), with Warde-Farley and Papernot

Real-world adversarial examples

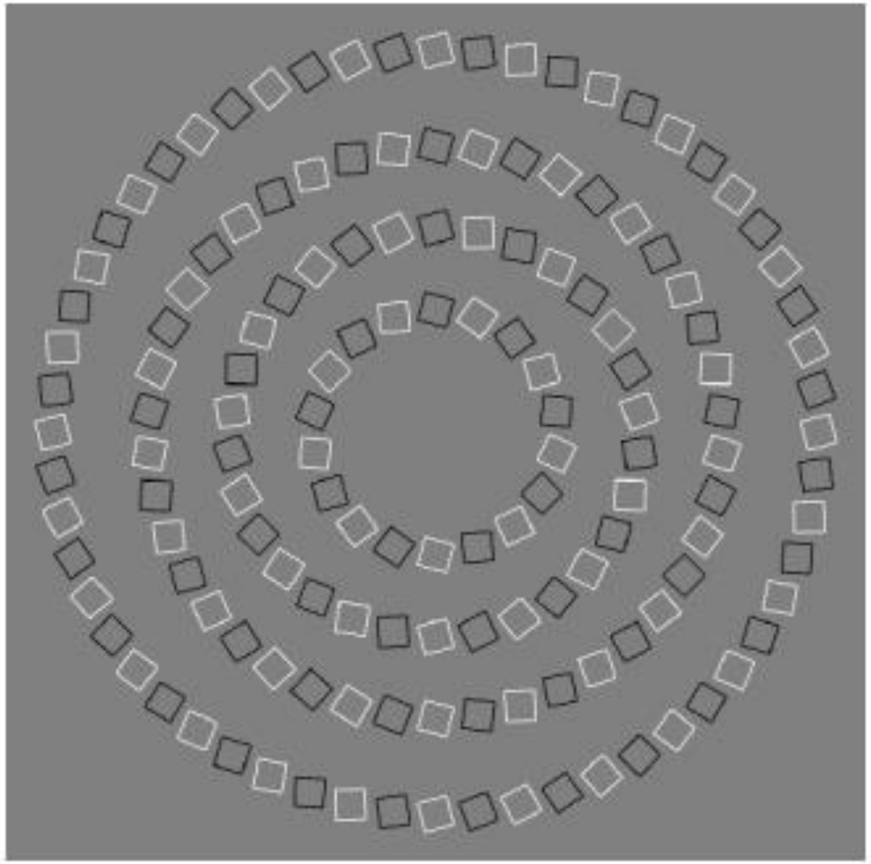


all of these turn into 45 mph speed limit signs

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

Athalye et al. **Synthesizing Robust Adversarial Examples**. 2017.

Human adversarial examples?



(Pinna and Gregory, 2002)

These are concentric circles, not intertwined spirals.

(Goodfellow 2016)

Human adversarial examples?

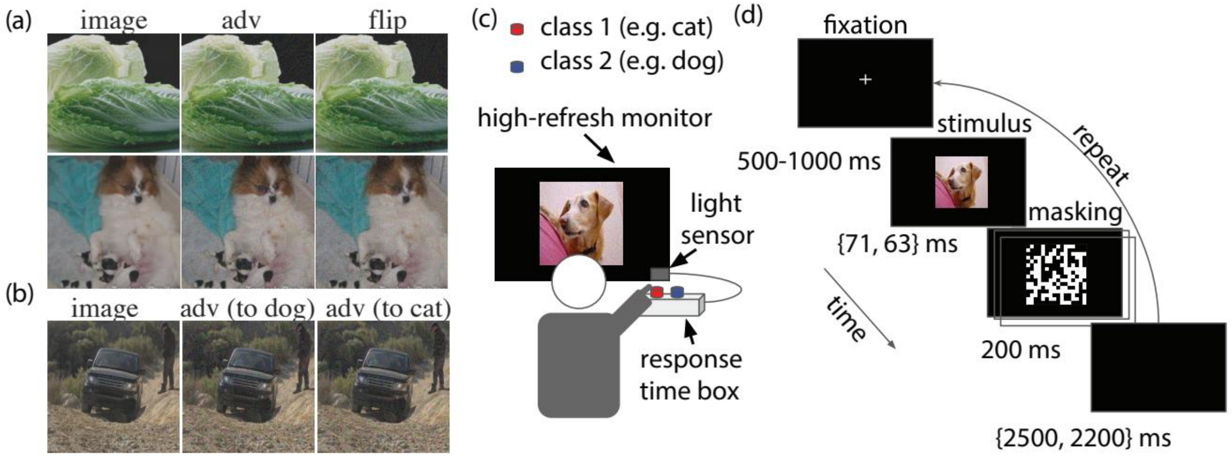


Figure 1: **Experiment setup and task.** (a) examples images from the conditions (`image`, `adv`, and `flip`). Top: `adv` targeting broccoli class. bottom: `adv` targeting cat class. See definition of conditions at Section 3.2.2. (b) example images from the `false` experiment condition. (c) Experiment setup and recording apparatus. (d) Task structure and timings. The subject is asked to repeatedly identify which of two classes (e.g. dog vs. cat) a briefly presented image belongs to. The image is either adversarial, or belongs to one of several control conditions. See Section 3.2 for details.

Elsayed et al. **Adversarial Examples that Fool both Computer Vision and Time-Limited Humans.** 2018.

What does this have to do with generalization?

Linear hypothesis is relevant not just for adversarial examples, but for understanding how neural nets do (and don't) generalize

When you train a model to classify cats vs. dogs, it is not actually learning what cats and dogs look like, it is learning about the patterns **in your dataset**

From there, it will extrapolate in potentially weird ways

Put another way, adversarial examples are not **bugs** they are **features** of your learning algorithm

Literally: Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, Aleksander Madry. **Adversarial Examples Are Not Bugs, They Are Features**. 2019.

Basic idea: neural nets pay attention to "adversarial directions" because it **helps** them to get the right answer on the training data!

Summary

- Neural nets generalize very well on tests sets drawn from the **same distribution** as the training set
- They sometimes do this by being a smart horse
  - This is not their fault! It’s your fault for asking the wrong question
- They are often not **well-calibrated**, especially on out-of-distribution inputs
- A related (but not the same!) problem is that we can almost always synthesize **adversarial examples** by modifying normal images to “fool” a neural network into producing an incorrect label
- Adversarial examples are most likely **not** a symptom of overfitting
  - They are conserved across different models, and affect low-capacity models
- There is reason to believe they are actually due to excessively linear (simple) models attempting to extrapolate + distribution shift

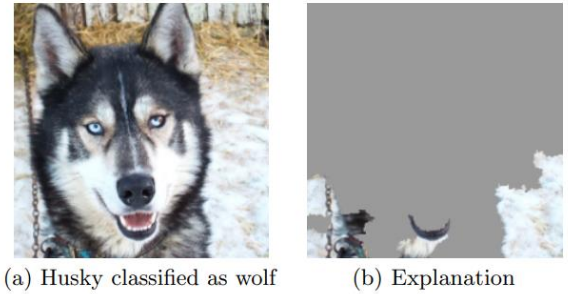
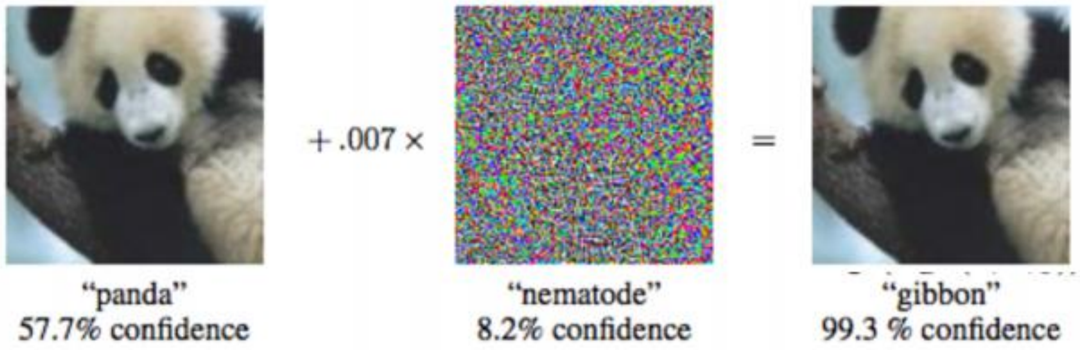


Figure 11: Raw data and explanation of a bad model’s prediction in the “Husky vs Wolf” task.



Adversarial attacks

A formal definition

Caveat: formally defining an adversarial attack is helpful for mathematicians, but can hide some important real-world considerations

Real attackers don't care about your definitions

relation: $R(x, x')$

original image

altered image

speed limit: 45

what is ϵ?

example: $R_\infty(x, x') = ||x - x'||_\infty$

each pixel changed by at most ϵ

attack: $x^\star \leftarrow \arg\max_{x' : R(x,x') \leq \epsilon} \mathcal{L}_\theta(x', y)$

e.g., $\mathcal{L}_\theta(x, y) = -\log p_\theta(y|x)$

pick image x' close to x

maximize the loss of your choice

defense: $\theta^\star \leftarrow \arg\min_\theta \underbrace{\sum_{(x,y)\in D} \max_{x' : R(x,x') \leq \epsilon} \mathcal{L}_\theta(x', y)}_{\text{robust loss}}$

Fast gradient sign method (FGSM)

A very simple approximate method for an infinity norm relation

$$R(x, x') = ||x - x'||_\infty$$

$$\text{attack: } x^\star \leftarrow \arg\max_{x' : R(x,x') \leq \epsilon} \mathcal{L}_\theta(x', y)$$

$$\text{“first order” assumption: } \mathcal{L}(x', y) \approx \mathcal{L}(x, y) + (x' - x)^T \nabla_x \mathcal{L}$$

ordinarily, we might think that this would make for a very **weak** attack, but we saw before how neural nets seem to behave locally linearly!

$$\text{attack: } x^\star \leftarrow \arg\max_{x' : ||x - x'||_\infty \leq \epsilon} (x' - x)^T \nabla_x \mathcal{L}$$

optional solution: move each dimension of x in direction of $\nabla_x \mathcal{L}$ by ϵ

$$x^\star = x + \epsilon \text{sign}(\nabla_x \mathcal{L})$$

this works **very well** against standard (naïve) neural nets

it can be defeated with simple defenses, but more advanced attacks can be more resilient

A more general formulation

attack: $x^\star \leftarrow \arg\max_{x':R(x,x')\leq\epsilon} \mathcal{L}_\theta(x',y)$

Lagrange multiplier
could be chosen heuristically
or optimized with e.g. dual gradient descent

$$x^\star \leftarrow \arg\max_{x'} \mathcal{L}_\theta(x',y) - \lambda R(x,x')$$

optimize to convergence, for example with ADAM

$$\delta = x' - x$$

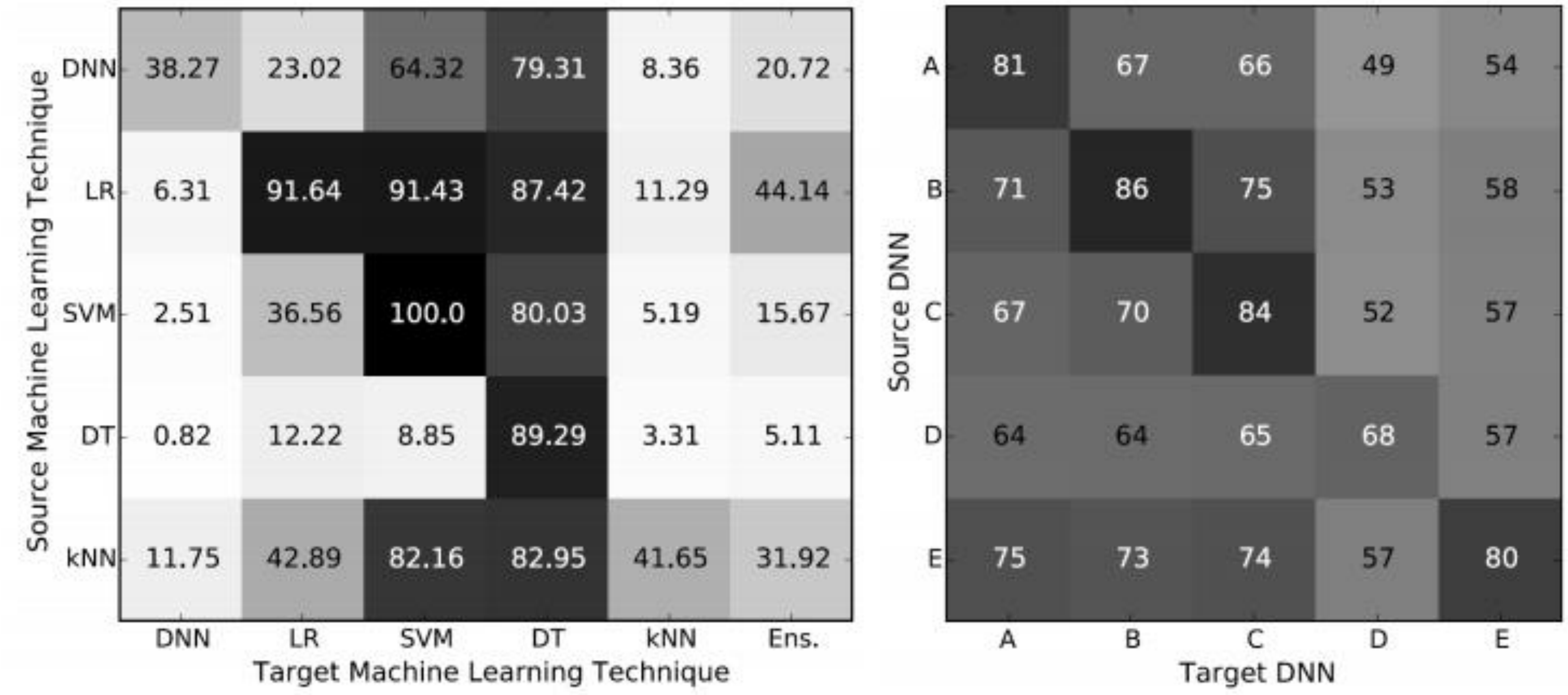
$$\delta^\star \leftarrow \arg\max_{\delta} \mathcal{L}_\theta(x+\delta,y) - \lambda||\delta||_\infty$$

In general can use a variety of losses here, including perceptual losses

In general, such attacks are very hard to defeat

Transferability of adversarial attacks

Oftentimes it just works

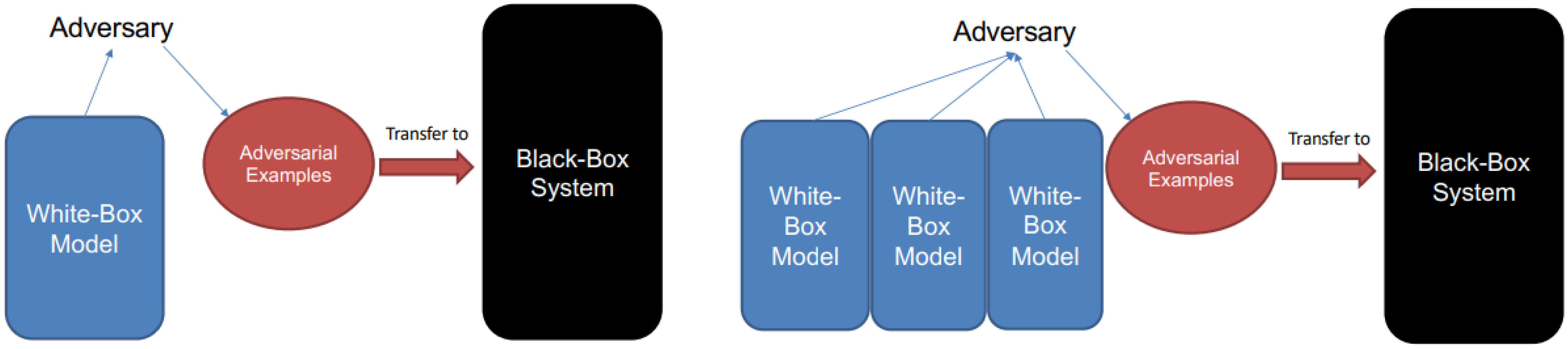


In particular, this means that we often don't need direct gradient access to a neural net we are actually attacking – we can just use **another** neural net to construct our adversarial example!

% success rate at fooling one model when trained on another

Papernot et al. **Transferability in Machine Learning: from Phenomena to BlackBox Attacks using Adversarial Samples**. 2016.

Zero-shot black-box attack



Liu, Chen, Liu, Song. **Delving into Transferable Adversarial Examples and Black-box Attacks**, ICLR 2017

Finite differences gradient estimation

It's possible to estimate the gradient with a moderate number of queries to a model (e.g., on a web server) without being able to actually directly access its gradient

$$x^\star = x + \epsilon \text{sign}(\nabla_x \mathcal{L})$$

all we need is the sign of the gradient

for each dimension i of x:

$$\text{get } v_i \leftarrow \mathcal{L}(x + 10^{-3} e_i, y)$$

a small number

i^{th} canonical vector

$$\nabla_x \mathcal{L} \approx (v - \mathcal{L}(x, y))/(10^{-3})$$

If you really want to do this, there are fancy tricks to even further reduce how many queries are needed to estimate the gradient

Defending against adversarial attacks?

There are **many** different methods in the literature for "robustifying" models against adversarial examples

$$\text{defense: } \theta^\star \leftarrow \arg\min_\theta \underbrace{\textstyle\sum_{(x,y)\in D} \max_{x':R(x,x')\leq\epsilon} \mathcal{L}_\theta(x',y)}_{\text{robust loss}}$$

Simple recipe: adversarial training

1. sample minibatch $\{(x_i, y_i)\}$ from dataset $\mathcal{D}$
2. for each x_i, compute adversarial x'_i e.g., FGSM: $x'_i \leftarrow x_i + \epsilon\text{sign}(\nabla_{x_i}\mathcal{L}_\theta(x_i, y_i))$
3. take SGD step: $\theta \leftarrow \theta - \alpha\sum_i \nabla_\theta\mathcal{L}_\theta(x'_i, y_i)$ usually also add original loss grad $\nabla_\theta\mathcal{L}_\theta(x_i, y_i)$

Usually doesn't come for free:
increases robustness to adversarial attacks (lower % fooling rate)
decreases overall accuracy on the test set (compared to naïve network)

Summary

- **Fast gradient sign method:** a simple and convenient way to construct an attack with an infinity-norm constraint (i.e., each pixel can change by at most a small amount)
- **Better attack methods:** use many steps of gradient descent to optimize an image subject to a constraint
- **Black box attack** without access to a model's gradients
  - **Construct** your own model (or an ensemble), attack those, and then transfer the adversarial example in zero shot
  - **Estimate** the gradient using queries (e.g., finite differences)
- **Defenses:** heavily studied topic! But very hard

$$x^\star = x + \epsilon \text{sign}(\nabla_x \mathcal{L})$$

$$\delta^\star \leftarrow \arg\max_\delta \mathcal{L}_\theta(x+\delta, y) - \lambda ||\delta||_\infty$$