

10417

Deep Learning: Fall 2020

Machine Learning Department

Guest Lecturer:
Elan Rosenfeld

Adversarial Examples

Training a neural network

By now, we've got a pretty decent handle on training a network.

Pass the input through the network, determine the loss.

Evaluate the gradient of the loss w.r.t. network parameters.

Take a step in the negative direction (to minimize loss).

At test time, just pass the test point through the network.

$$\mathcal{L}(\theta) = -\frac{1}{n} \sum_{i=1}^n y_i \log \hat{y}_i$$

$$\theta = \theta - \eta \nabla_{\theta} \mathcal{L}(\theta)$$

$$f(x_{test}) = f(x_{test}; \theta)$$

What if we differentiate wrt input?

[Szegedy et al. 2013]:

Pass a test input through the network, determine the loss.

Evaluate the gradient of the loss w.r.t. ~~network parameters~~ *the input*.

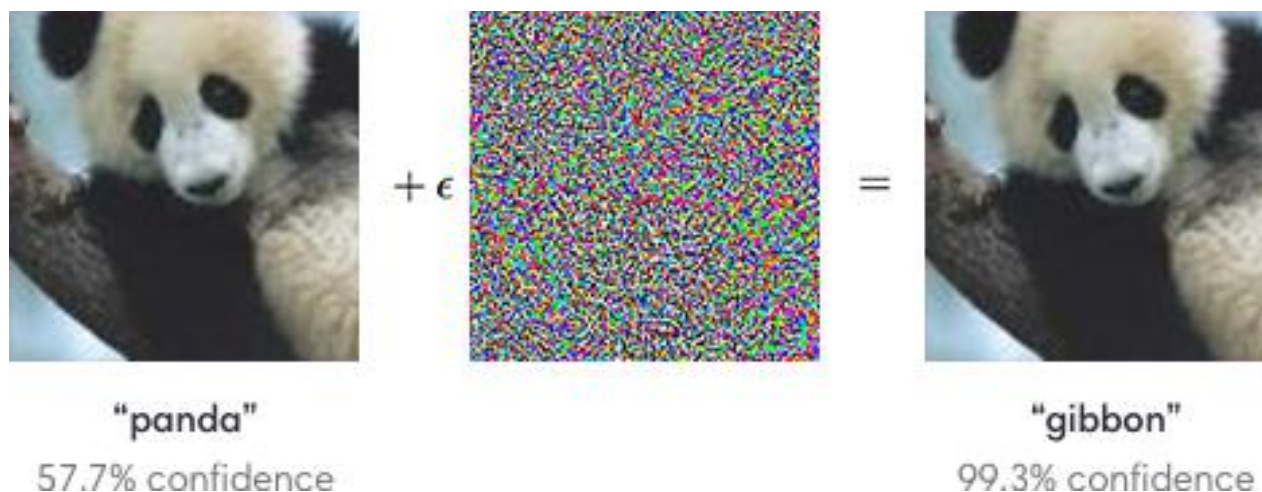
Take a step in the ~~negative~~ *positive* direction.

$$\mathcal{L}(x) = -y_i \log f_{\theta}(x)_i$$
$$x = x + \eta \nabla_x \mathcal{L}(x)$$

Turns out it's extremely easy to cause the network to make an error on this modified input. These modifications are known as ***Adversarial Examples***.

What if we differentiate wrt input?

Even worse, the perturbation that needs to be made to cause an error is *completely imperceptible* to humans.



Pattern repeats across all datasets and architectures.
Attacks *transfer* across all datasets and architectures.

Some important questions

Why is this happening? We don't know (exactly)! A couple theories:

- Deep neural networks have very non-linear decision boundaries. In high dimensions, there will be directions with large gradient. A tiny step in these dimensions means a large change in output.
 - This suggests that smoothing the output gradients could lead to better robustness. *[Ross et al. 2017]*
- Deep neural networks don't "see" like we do. They are machines optimized to extract statistical signal from the noise. So they pick up on patterns that exist but that we just don't notice. *[Ilyas et al. 2019]*
 - More on this later...

Some important questions

Why is this happening? We don't know (exactly)! A couple theories:

Regardless, learning a decision boundary that is robust to targeted input modifications is an inherently difficult problem.

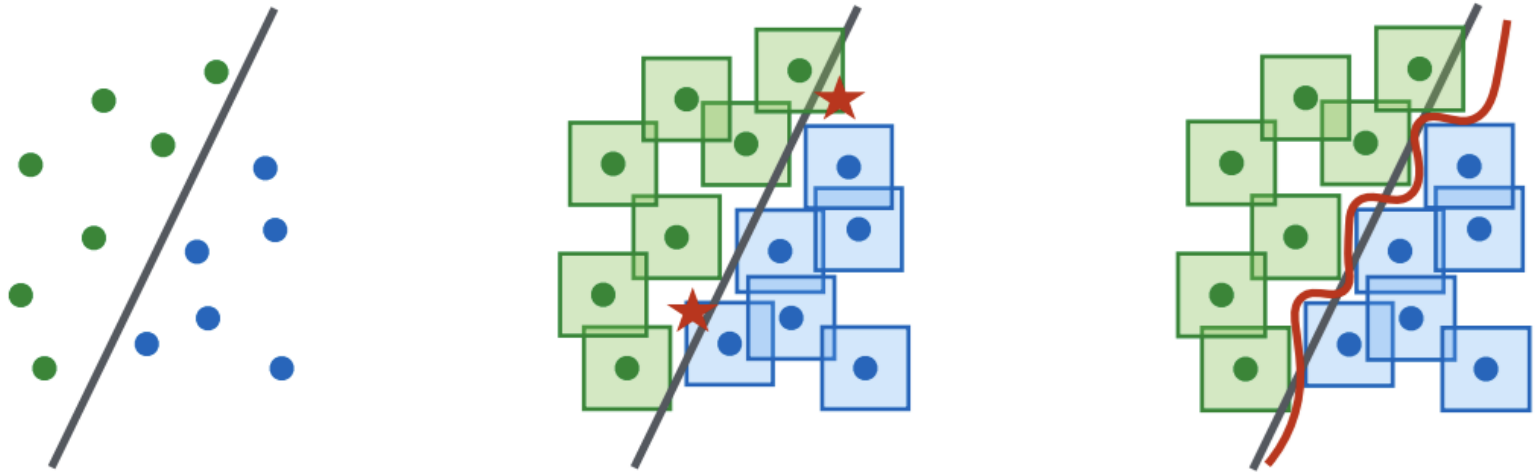


Image: Madry et al. 2017

Some important questions

Why should we care? It's not really a realistic threat model.

- Modifying inputs directly may not be realistic, but we can modify objects in the real world. Classic example is sticker on a stop sign.
[Kurakin et al. 2016]



Some important questions

Why should we care? It's not really a realistic threat model.

- Modifying inputs directly may not be realistic, but we can modify objects in the real world. Classic example is sticker on a stop sign.
[Kurakin et al. 2016]

(Adversarial examples can also be used for good, such as privacy!)



Some important questions

Why should we care? It's not really a realistic threat model.

- Studying why this happens and how to prevent it could lead to a better understanding of the strengths and weaknesses of neural networks, and statistical machine learning in general.
- More on this later...

Part I: Attacks

Threat Model

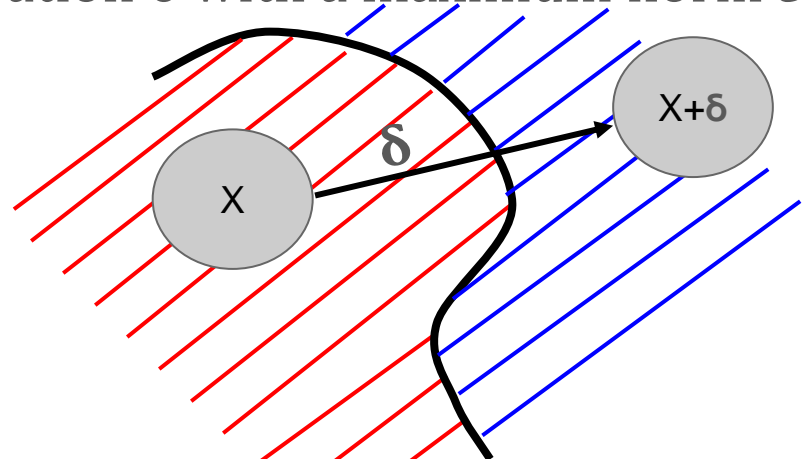
Defining the adversary's capabilities/limitations.

Two options:

1. **White box**: adversary has full access to model and gradients.
2. **Black box**: adversary only has query access to the model.

Most literature concerns **white box** attacks, so we'll focus on these.

1. Adversary receives input x , as well as model f and output $f(x)$.
2. Adversary produces perturbation δ with a maximum norm ϵ (typically ℓ_2 or ℓ_∞).
3. Goal is $f(x + \delta) \neq f(x)$.

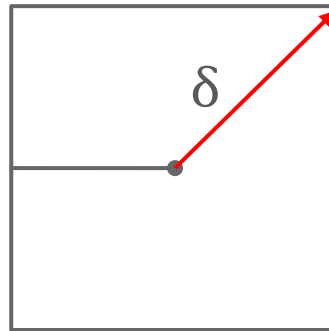


White Box Attack

[Goodfellow et al. 2014]: Fast Gradient Sign Method

The “most adversarial” perturbation typically has norm exactly ϵ . So let’s just step right to the corner of the ℓ_p norm ball, with the direction determined by the gradient.

$$\delta = \epsilon \cdot \text{sign}(\nabla_x L(x; \theta))$$



Simple to implement. Cheap to evaluate. Can we do better?

White Box Attack

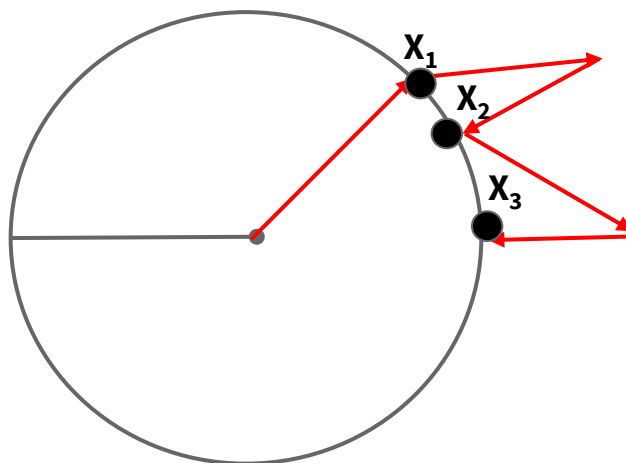
[Madry et al. 2017]: Projected Gradient Descent (PGD)

View this as an optimization problem.

Adversary's objective is: $\arg \max_{\delta} L(x + \delta; \theta) \quad \text{s.t.} \quad \|\delta\| \leq \epsilon$

This can be solved with projected gradient descent! Very simple:

1. Take a step in the direction of the gradient.
2. Project back onto the ℓ_p norm ball.
3. Repeat as many times as desired (more steps = stronger attack)



Part II: Defenses

Let's make a first attempt

What are the most obvious ways we could prevent this from happening?

The adversary uses gradient ascent to find a perturbation. So maybe we could hinder the adversary's ability to follow the gradient.

- We could add non-differentiability to the model after training (e.g., by outputting a one-hot vector rather than a softmax).
- We could add randomness to the test-time evaluation, meaning the adversary's estimate of the gradient has high variance.
- We could evaluate extremely “deep” networks with repeated computation, ensuring the gradient explodes or vanishes

None of these work, even a little bit. They're called **obfuscated gradients** and [Athalye et al. 2018] broke all of them.

So we've given up on empirical defenses, right?

Wrong! A whole slew of adaptive defenses have since been released.

k-Winners Take All, Generative Classifiers, ME-Net, Asymmetrical Adversarial Training, Sparse Fourier Transform...

But some of these work, right?

Nope! [Tramèr et al. 2020] broke these.

Ok, so now we're *really* done with empirical defenses, right??

Just one more: “Adversarial Training”

This one actually works (so far, no one has broken it). It’s very simple.

[Goodfellow et al. 2014, Madry et al. 2017]

We can view adversarial examples as a game:

1. Defender learns parameters θ to minimize $\mathbb{E}_{x \sim D} L(x; \theta)$ on the test set.
2. Adversary receives input $x \sim D$, wants to produce δ within an ϵ norm ball to maximize $L(x + \delta; \theta)$.

Why not modify defender’s objective to allow for this perturbation?

Defender’s objective now becomes:

$$\min_{\theta} \mathbb{E}_{x \sim D} \left[\max_{\delta: \|\delta\| \leq \epsilon} \mathcal{L}(x + \delta; \theta) \right]$$

Minimax optimization is tough to solve.

Done by attacking *every training point for every batch* during training.

Very slow! A couple follow-up works address this. *[Shafahi et al. 2019]*

Provable Defenses

Clearly, the ML community has a lot to learn from the security community.

Rule #1: If you can't prove it can't be broken, it can be broken.

*Rule #2: Even if you can prove it can't be broken, **it can probably be broken.***

Obviously we can't guarantee that the network is correct (that's the whole point of machine learning!).

Instead, we can guarantee that the network's decision boundary is not contained within an ϵ -norm ball.

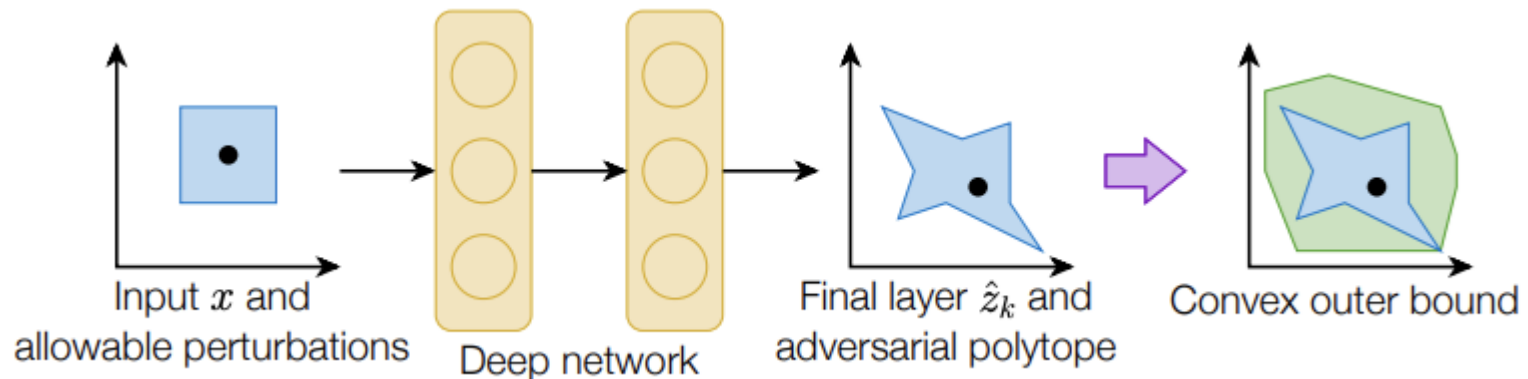
Why is this meaningful?

If we assume adversary is limited to an ϵ -norm perturbation, and decision boundary is not that close, then the input's classification cannot have been modified with a perturbation.

If our classifier is wrong, *it would have been wrong anyways.*

Provable Defenses

[Wong & Kolter, 2017]: As we pass the input through the network, maintain a convex envelope of possible activations caused by a perturbation. Then train the network to minimize this envelope.

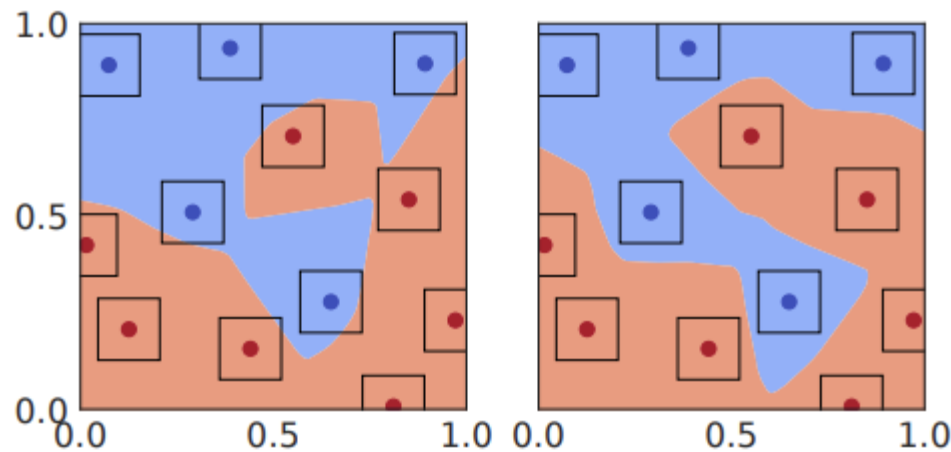


Called the **Convex Outer Adversarial Polytope**.

How well does this work?

Provable Defenses

[Wong & Kolter, 2017]: As we pass the input through the network, maintain a convex envelope of possible activations caused by a perturbation. Then train the network to minimize this envelope.



Pretty well! What's the catch?

Solving this optimization problem is very expensive, and gets even more so as the network grows. Not really scalable.



Provable Defenses

[Mirman et al., 2018, Zhang et al., 2019]: As we pass the input through the network, maintain an axis-aligned polytope of possible activations caused by a perturbation. Use this to certify the worst possible case

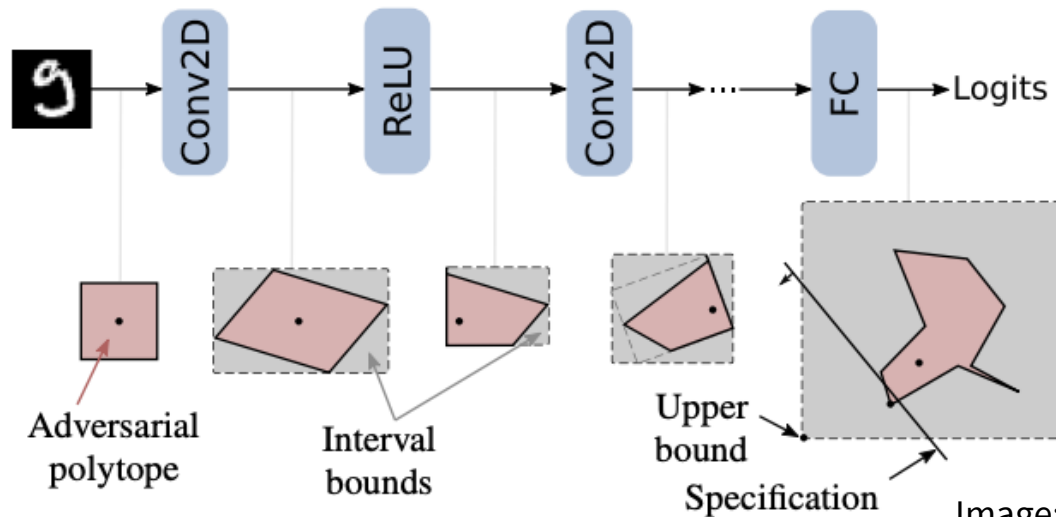


Image: Gowal et al. 2019

This is known as **Interval Bound Propagation**.
Significantly faster, but still can't scale to ImageNet.



Provable Defenses

[Li et al. 2019, Cohen et al., 2019]: Don't try to certify the network f !
Instead, define a *new* network g whose output is f convolved with Gaussian noise:

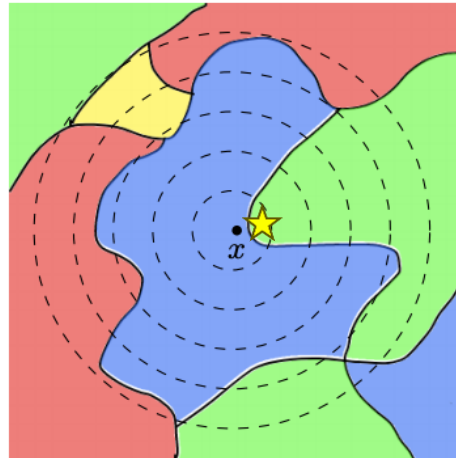
$$g(x) = \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)}[f(x + \epsilon)]$$


This new network g can be certified as a function of the margin with which the largest softmax class wins.

Known as **Randomized Smoothing**. The decision boundary of g is “smoothed” compared to f .


Randomized Smoothing

Here are the decision regions of an arbitrary classifier f :



The point x is classified as **blue**, but a small perturbation to the  will move it into the decision region for **green**.

Instead, integrate the decision regions with a Gaussian distribution centered at x .

Even though the  itself is in **green**, the “majority vote” will still be for **blue**.

Randomized Smoothing

How well does this work? Outperforms others by a large margin.
Even better, it's relatively cheap and applies to *any classifier f* .
Easily scales to larger networks capable of classifying ImageNet.



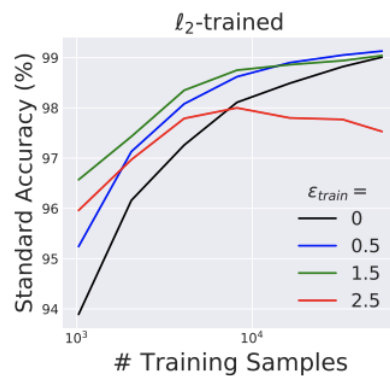
A few caveats:

1. You can't actually integrate a neural network's decision regions!
Instead, this is approximated via Monte Carlo sampling.
1. Gaussian noise ideal for ℓ_2 norm, but needs modifications to work for other norms. Recent work shows noise must have variance $\Omega(d)$ for ℓ_∞ [Blum et al. 2020].

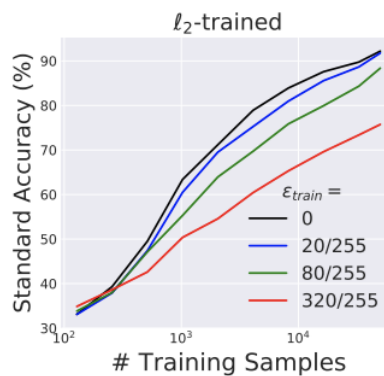
Part III: Moving Forward

The Cost of Adversarial Robustness

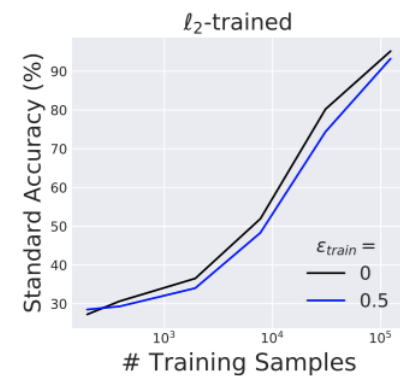
Adversarial training can be seen as a form of “data augmentation”. We might expect such training to lead to improved generalization.



(a) MNIST



(b) CIFAR-10



(c) Restricted ImageNet

Image: Tsipras et al. 2019

It does, for very small sample size. But for larger sample size/stronger adversary we see a *large* drop in standard accuracy. SOTA robustness on CIFAR10 reduces standard accuracy to $\sim 75\%$. Why is this happening?

The Cost of Adversarial Robustness

[Schmidt et al. 2018] Consider the following (informal) model:

Ground-truth mean vector $\theta^* \in \mathbb{R}^d$ with norm \sqrt{d}

Two classes: $\{-1, 1\}$

For each sample i :

1. Draw a label y_i uniformly
2. Sample $x_i \sim N(\theta^* \cdot y_i, \sigma^2 I)$

It's easy to see that each dimension of x_i is *weakly* correlated with y .

A bunch of weakly correlated dimensions can be combined to produce a strong statistical signal (à la boosting).

Given a single training point (x_1, y_1) , optimal linear classifier is

$f_w(x) = \text{sign } \langle w, x \rangle$ with $w = x \cdot y$.

With small enough noise, this classifier can achieve $<1\%$ test error.

The Cost of Adversarial Robustness

[Schmidt et al. 2018] Consider the following (informal) model:

Ground-truth mean vector $\theta^* \in \mathbb{R}^d$ with norm \sqrt{d}

Two classes: $\{-1, 1\}$

For each sample i :

1. Draw a label y_i uniformly
2. Sample $x_i \sim N(\theta^* \cdot y_i, \sigma^2 I)$

But what if we allow for ℓ_∞ perturbations of size ϵ ?

Each dimension x_{id} is drawn from $N(\theta_d^* \cdot y, \sigma^2)$. With high probability, many dimensions will be within $\epsilon/2$ of 0. So we can *flip* the weak correlation for those dimensions!

Can provably show that test error is at least $(1 - 1/d) / 2$ in expectation if training set is not large enough.

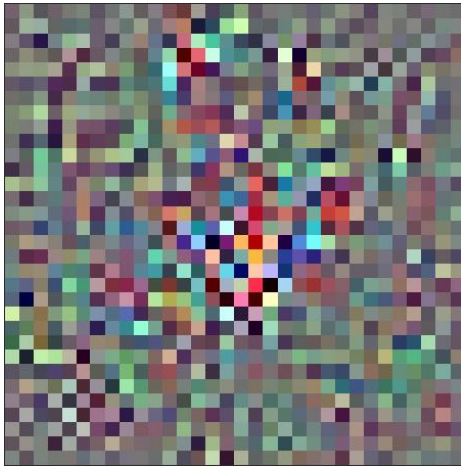
Adversarially robust generalization requires more data (for this model)

One Last Theory

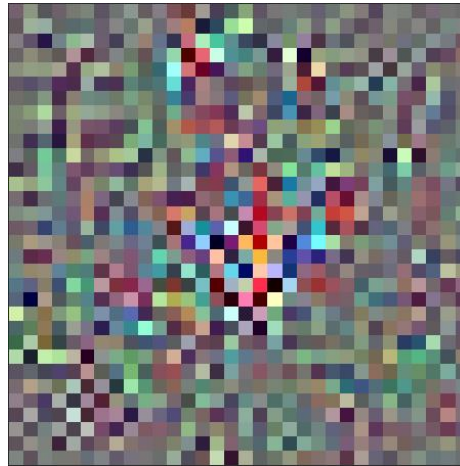
One of the most compelling recent theories for this phenomenon is the following:

Adversarial examples are not bugs, they are features! [*Ilyas et al. 2019*]

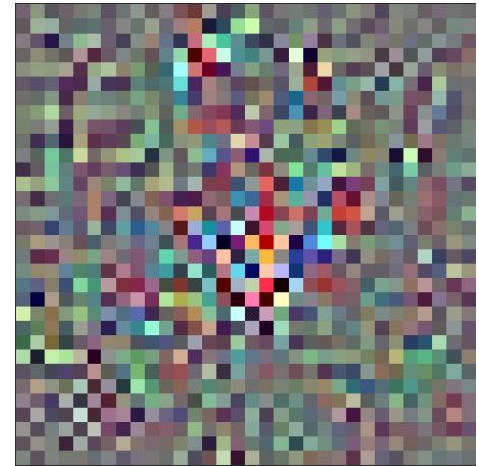
Suppose you were given a collection of images, and asked to learn to classify them:



Dog



Dog



Cat

One Last Theory

These images have no “features”, right? Well, maybe they do, but they are features like “the pixel in location (4, 5) is white”.

This is how neural networks view images.

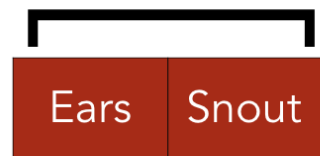
They are simply trained to extract the maximum signal available.

What if, when we make these imperceptible changes to the images, we are actually *changing the statistical properties of an image*, but in a way humans can't detect?

This would imply there are two types of features: **robust** and **non-robust**.

Robust features

Correlated with label
even with adversary

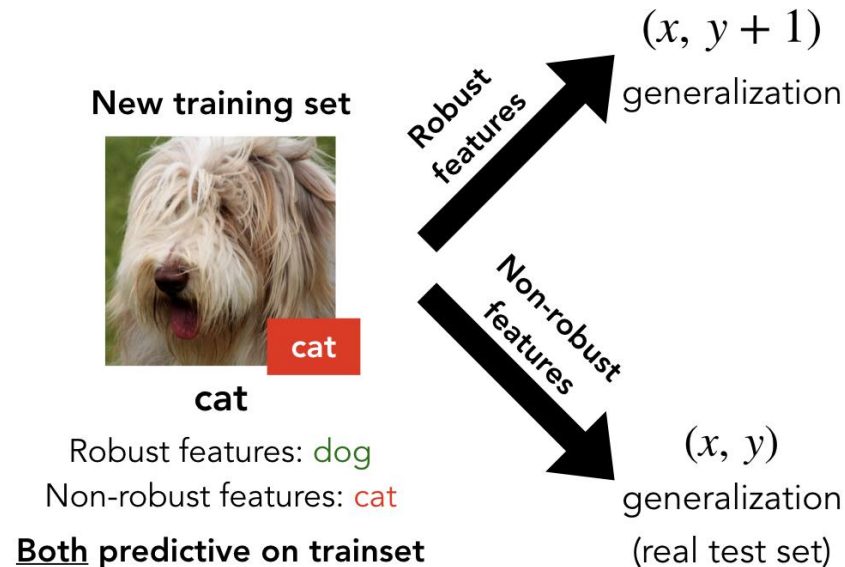


Input

One Last Theory

We can test this!

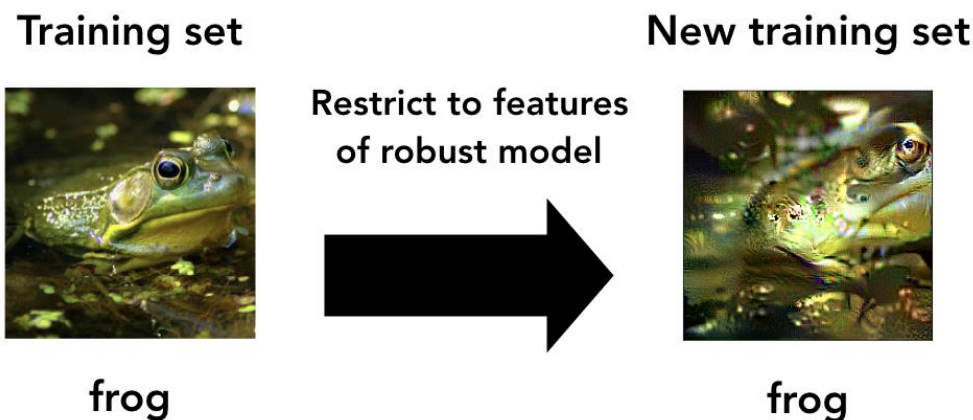
Construct a dataset of adversarially perturbed images, with a permutation on labels. E.g., all labels y are perturbed to label $y+1$.



Standard training on this *clearly mislabeled* dataset leads to generalization on the original test set!

One Last Theory

Another experiment: perturb each image in the training set so that only robust features contain a signal (by perturbing non-robust features randomly).



Standard training on this dataset leads to *adversarially robust* generalization.

So there is some signal in the “non-robust” features, and by removing them we are forcing the model to learn “robust” features!

This is another possible explanation for the cost of adversarial robustness: we are forcing the classifier to *ignore* this signal.