Adversarial Examples & Human-ML Alignment

Shibani Santurkar

Based on joint works with:

Logan Engstrom

Andrew Ilyas

Brandon Tran

Dimitris Tsipras

Alexander Turner

Aleksander Mądry

gradient-science.org

Lab notebook

github.com/MadryLab/AdvEx_Tutorial

Outline for Demos

1. **Adversarial examples**
    - Exercise 1: Adversarial attacks
    - Exercise 2: Are adversarial examples meaningless?
2. **Adversarial examples and interpretability**
    - Exercise 3: Gradient saliency
    - Exercise 4: SmoothGrad
3. **A closer look at robust models**
    - Exercise 5: (Large) adversarial attacks for robust models
    - Exercise 6: Robust gradients
    - Exercise 7: Robust feature visualization

I. Adversarial examples (5m)

Fool std. models with imperceptible changes to inputs

Perturbation: $\delta' = \arg\max_{||\delta||_2 \in \epsilon} \ell(\theta; x + \delta, y)$

→ **Method:** Gradient descent to increase loss w.r.t. true label

(Pick an incorrect class, and make model predict it)

→ How far do we need to go from original input?

→ Play with attack parameters (steps, step size, epsilon)

II. Are adv. examples meaningless? (5m)

→ **Method:**

1. Train a binary classifier on “cats’” vs “airplanes”.
2. Use model in 1. to find adv. examples for entire training set.
3. Use adv. examples from 2. and **(incorrect) labels** predicted by model to construct a new dataset.
4. Use dataset in 3. to train a new model and then evaluate it on the **standard test set (w/ correct labels).**

→ How well does the model trained on the “mislabeled” dataset in perform?

→ How well would you expect humans to perform given the same training data?

Try at home

Pre-generated Datasets

github.com/MadryLab/constructed-datasets

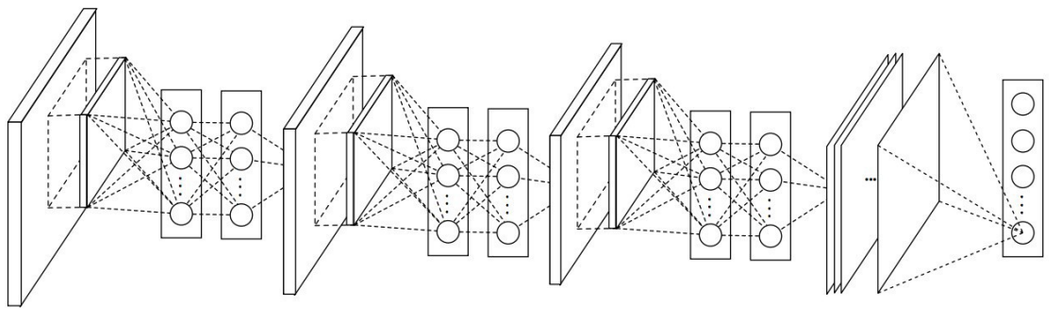
Adversarial examples & training library

github.com/MadryLab/robustness

Local explanations

How can we understand **per-image** model behavior?

Input x



Pile of linear algebra

Dog	**95%**
Bird	2%
...	
Primate	4%
Truck	0%

Predictions

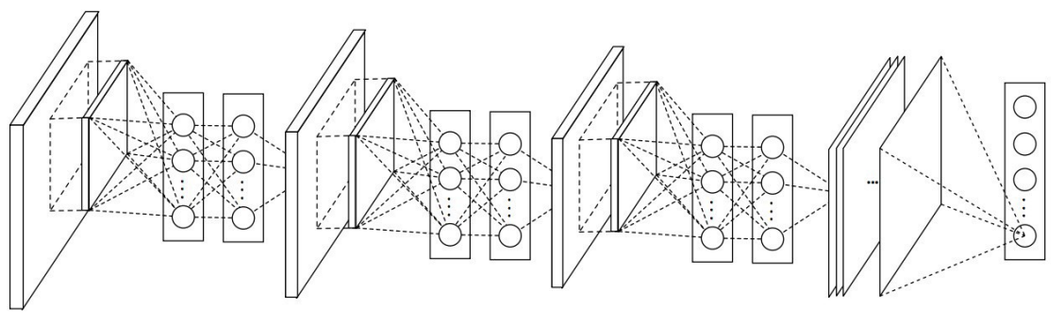
Why is **this image** classified as a dog?

Which pixels are important for this?

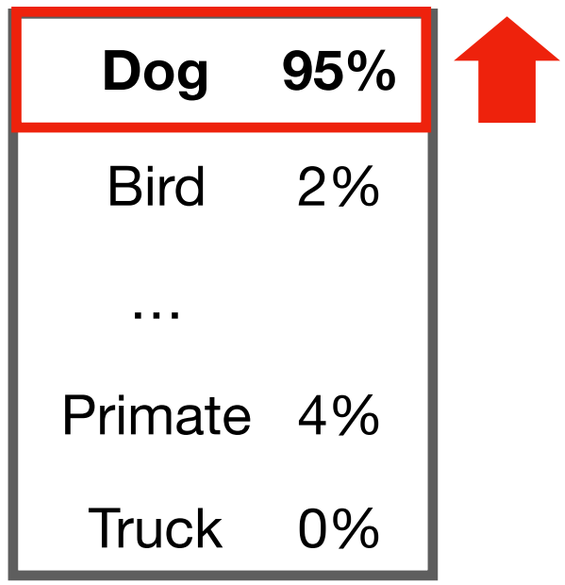
Local explanations

Sensitivity: How does each pixel affect predictions?

Input x



Pile of linear algebra



Predictions

Gradient saliency: $g_i(x) = \nabla_x C_i(x; \theta)$

→ **Conceptually:** Highlights important pixels

III. Gradient saliency (5m)

Explore model sensitivity via gradients

→ **Basic method:** Visualize gradients for different inputs

→ What is the dimension of the gradient?

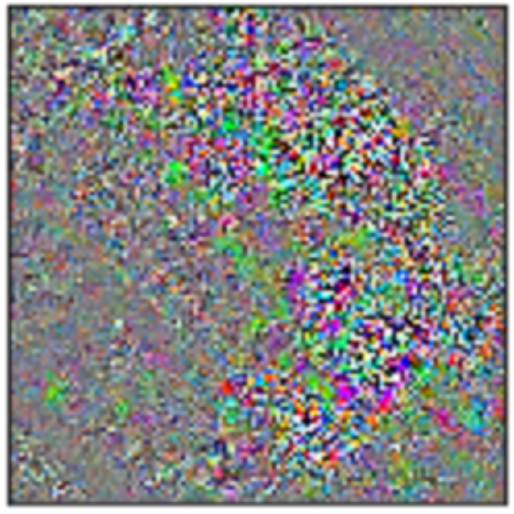
→ Optional: Does model architecture affect visualization?

What did you see?

Gradient explanations do not look amazing

Original Image

Gradient



How can we get rid of all this noise?

Better Gradients

SmoothGrad: average gradients from multiple (nearby) inputs

[Smilkov et al. 2017]

$$sg(x) = \frac{1}{N}\sum^{N} g(x + N(0,\sigma))$$

average

add noise

Intuition: “noisy” part of the gradient will cancel out

IV. SmoothGrad (5m)

Implement SmoothGrad

$$sg(x) = \frac{1}{N}\sum^{N} g(x + N(0,\sigma))$$

→ **Basic method:** Visualize SmoothGrad for different inputs

→ Does visual quality improve over vanilla gradient?

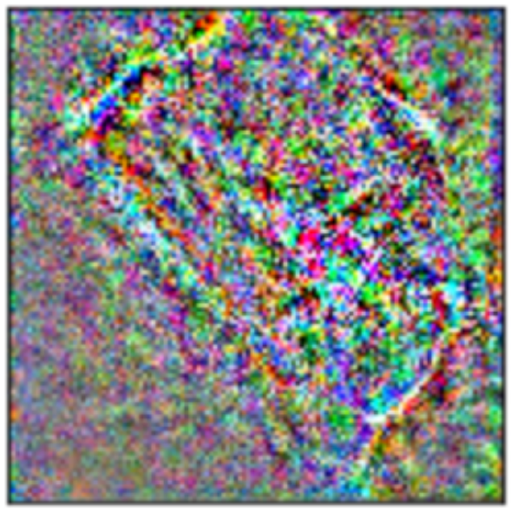
→ Play with number of samples (N) and variance (σ)

What did you see?

Interpretations look much cleaner

Original Image

SmoothGrad



But, did we change something fundamental?

Did the “noise” we hide mean something?

V. Adv. Examples for Robust Models (5m)

Imperceptible change images to fool **robust** models

Perturbation: $\delta' = \arg\max_{||\delta||_2 \in \epsilon} \ell(\theta; x + \delta, y)$

→ **Once again:** Gradient descent to increase loss

(Pick an incorrect class, and make model predict it)

→ How easy is it to change the model prediction?

(compare to standard models)

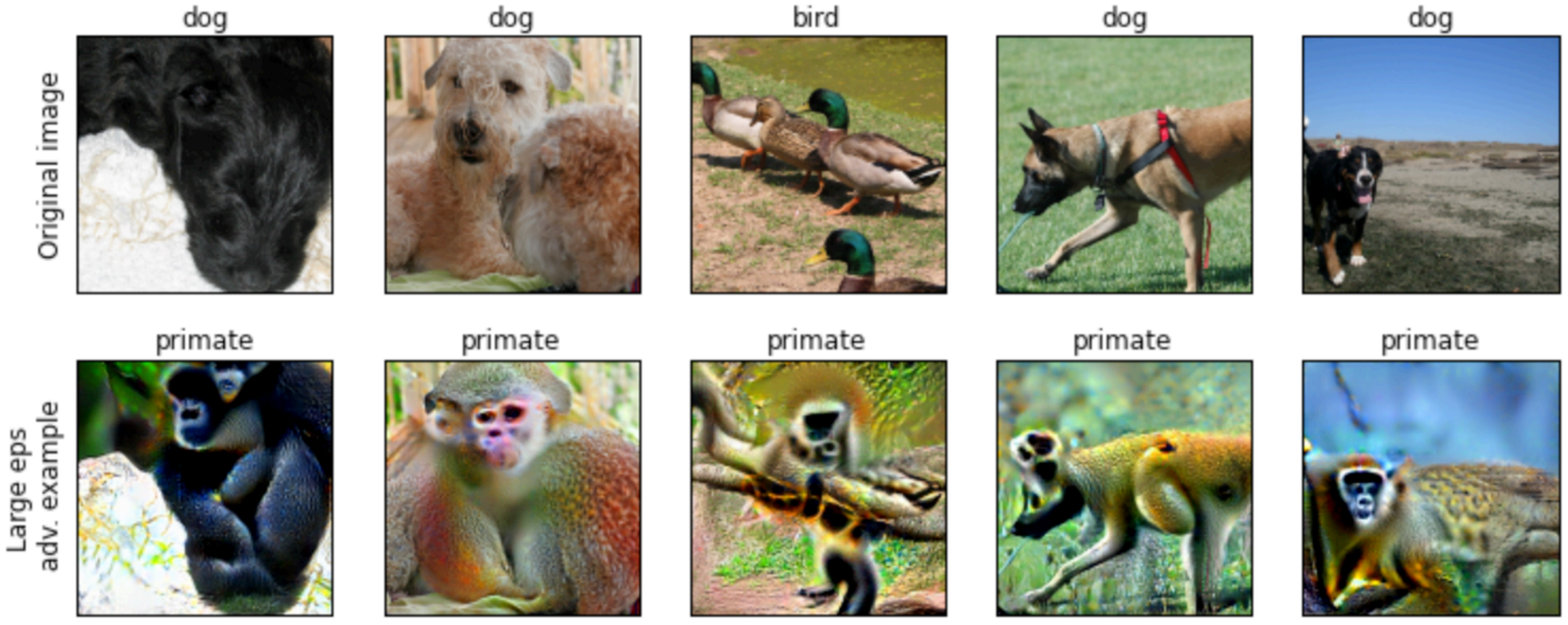
→ Again play with attack parameters (steps, step size, epsilon)

Small ε: What did we see?

For robust model: Harder to change prediction with imperceptible (small ε) perturbation

Large ε: What did we see?

Target class: ``Primate``



Large-ε adv. examples for robust models actually modify **semantically meaningful** features in input

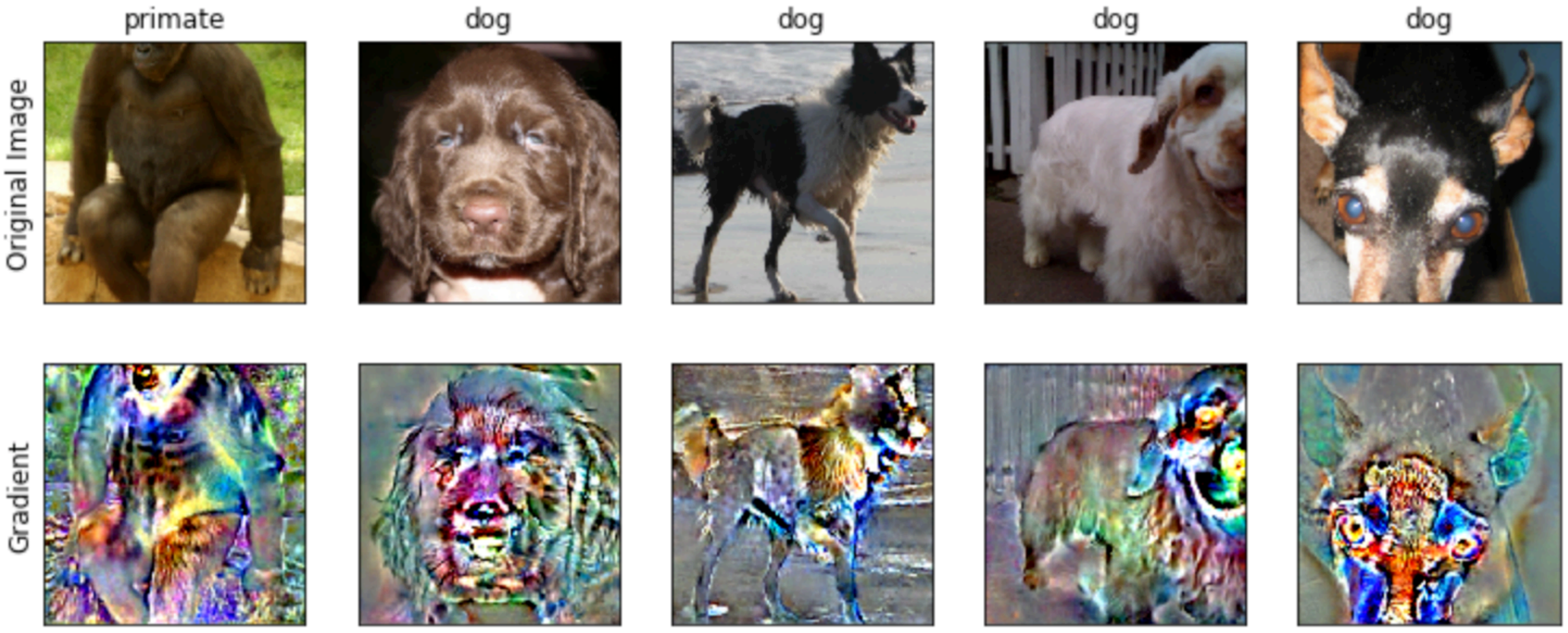
VI.I Robust model gradients (5m)

Explore **robust** model sensitivity via gradients

→ Visualize gradients for different inputs

→ Compare to grad (and SmoothGrad) for standard models

What did we see?

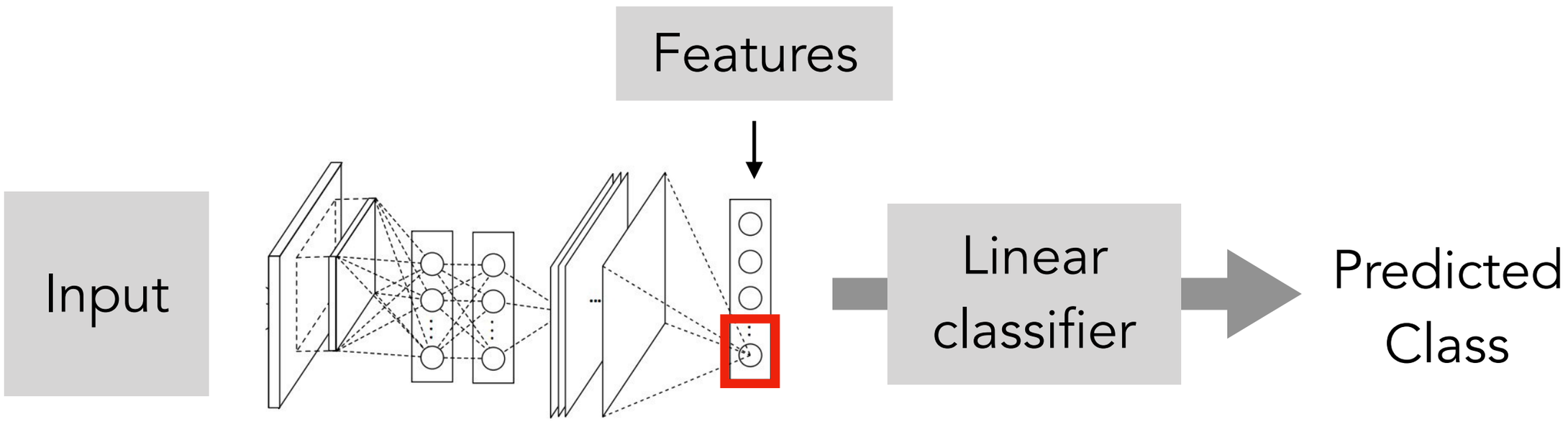


Vanilla gradients look nice, without **post-processing**

Maybe robust models rely on ``better`` features

Dig deeper

Visualize learned representations



Use gradient descent to maximize neurons

Exercise VI.II: Visualize features (5m)

Finding inputs that maximize specific features

→ Extract feature representation from model

(What are its dimensions?)

→ Write loss to **max. individual neurons** in feature rep.

→ **As before:** Use gradient descent to find inputs that max. loss

→ Optional: Repeat for standard models

→ Optional: Start optimization from noise instead