Adversaries & Interpretability

SIDN: An IAP Practicum

Shibani Santurkar

Dimitris Tsipras

gradient-science.org

Outline for today

1. **Simple gradient explanations**
    - Exercise 1: Gradient saliency
    - Exercise 2: SmoothGrad
2. **Adversarial examples and interpretability**
    - Exercise 3: Adversarial attacks
3. **Interpreting robust models**
    - Exercise 4: Large adversarial attacks for robust models
    - Exercise 5: Robust gradients
    - Exercise 6: Robust feature visualization

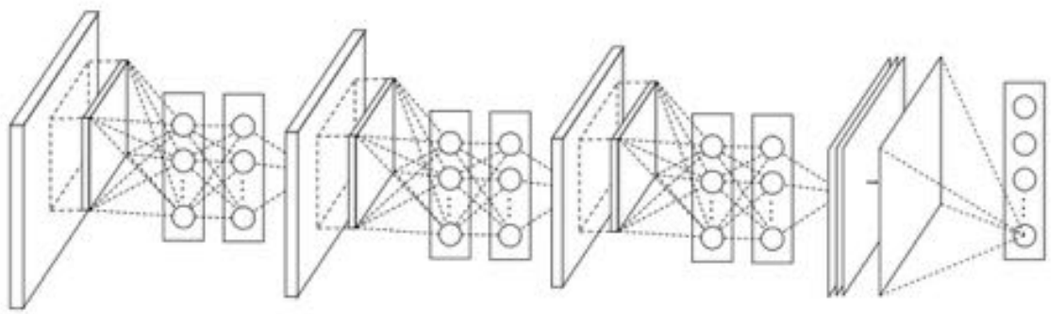
Lab notebook

github.com/SIDN-IAP/adversaries

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

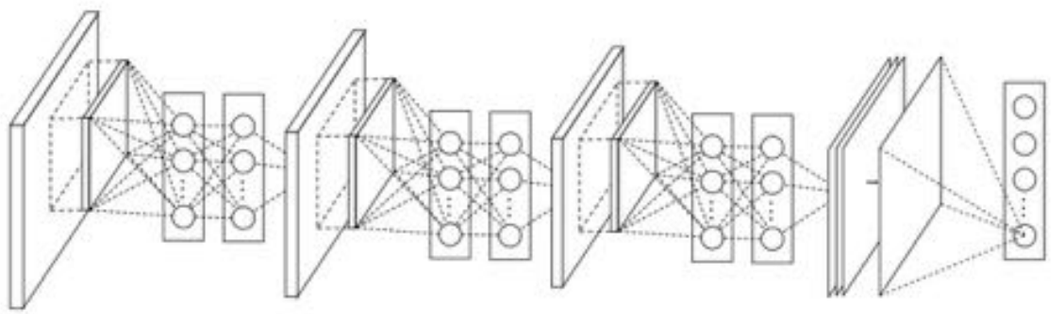
Why is **this image** classified as a dog?

Which pixels are important for this?

Local explanations

Sensitivity: How does each pixel affect predictions?

Input x



Pile of linear algebra

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

Predictions

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

→ **Conceptually:** Highlights important pixels

Exercise 1: Try it yourself (5m)

Explore model sensitivity via gradients

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

→ What is the dimension of the gradient?

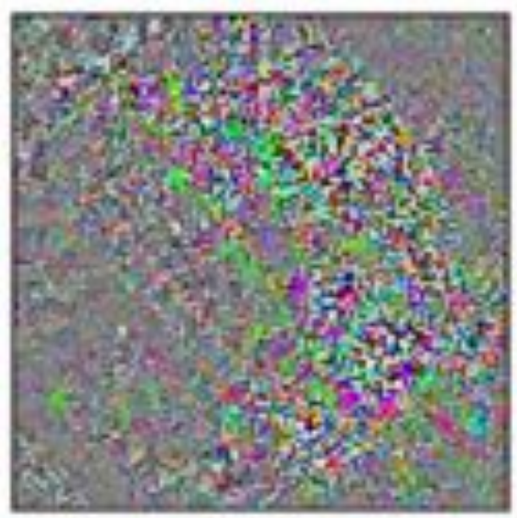
→ Optional: Does model architecture affect visualization?

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

Exercise 2: SmoothGrad (10m)

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?

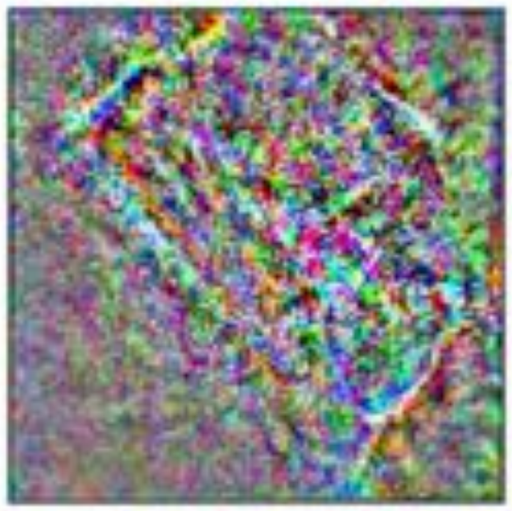
→ Play with number of samples (N) and variance (σ)

What did you see?

Interpretations look much cleaner

Original Image

SmoothGrad



But, did we change something fundamental?

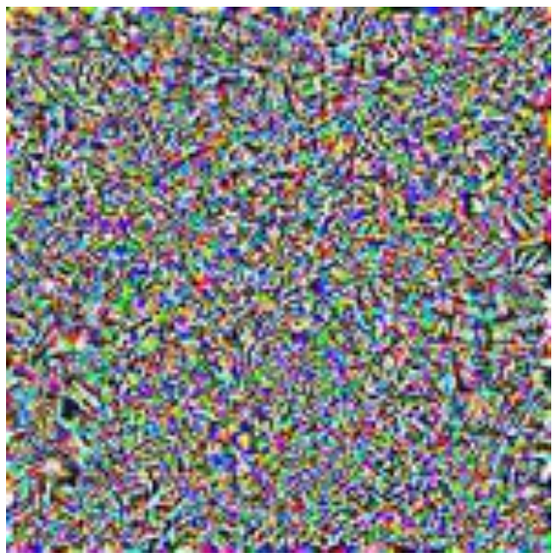
Did the “noise” we hide mean something?

Adversarial examples

"pig" (91%)

+0.005x

perturbation



=

"airliner" (99%)

[Biggio et al. 2013; Szegedy et al. 2013]

perturbation $= \arg\max_{\delta \in \Delta} \ell(\theta; x + \delta, y)$

Why is the model **so sensitive** to the perturbation?

Exercise 3: Adv. 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)

A conceptual model

Unreasonable sensitivity to meaningless features:
This has nothing to do with normal DNN behavior

Useful features (used to classify) **Useless features**

adv. examples flip these features to fool the model

Simple experiment

Training set
(cats vs. dogs)

dog

cat

Adv. ex. towards the other class

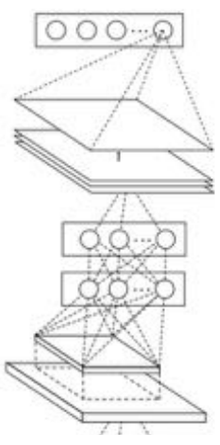
New training set
("mislabelled")

cat

dog

Train

Classifier



Evaluate on **original** test set

dog

cat

Simple experiment

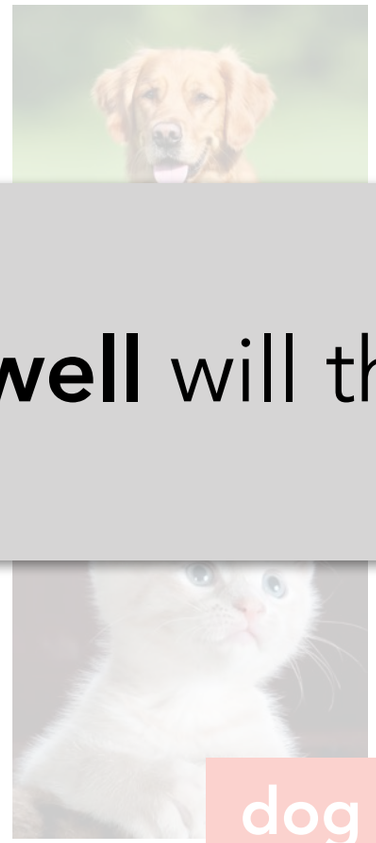
Training set
(cats vs. dogs)

dog

cat

Adv. ex.

New training set
("mislabelled")



dog

Classifier

Evaluate on
original test set

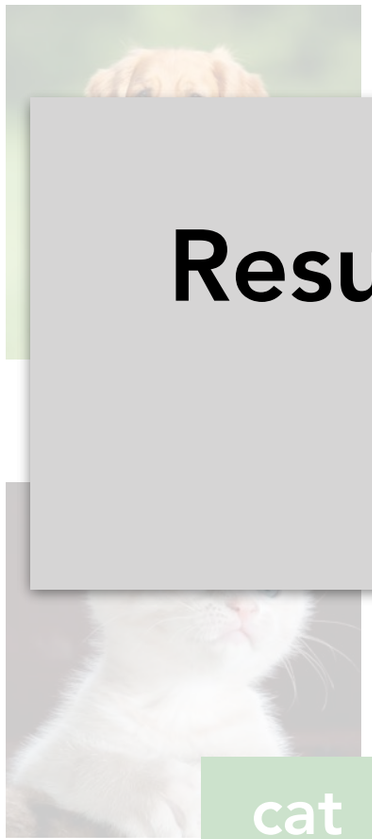
dog

cat

How well will this model do?

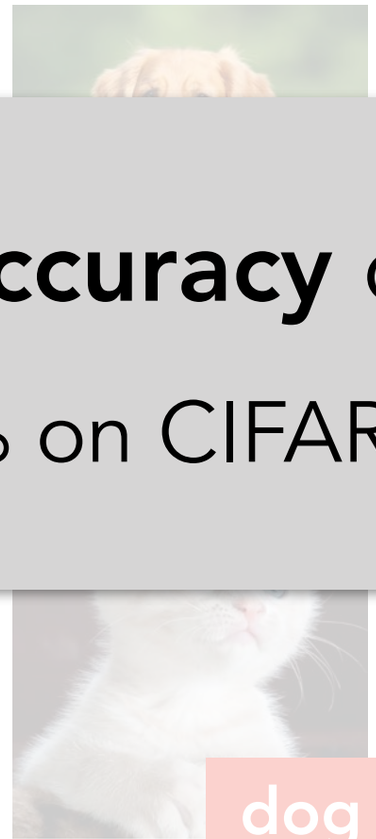
Simple experiment

Training set
(cats vs. dogs)



cat

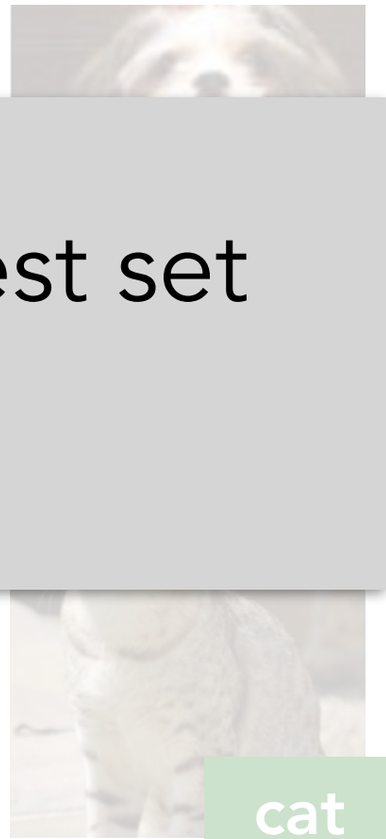
New training set
("mislabelled")



dog

Classifier

Evaluate on
original test set

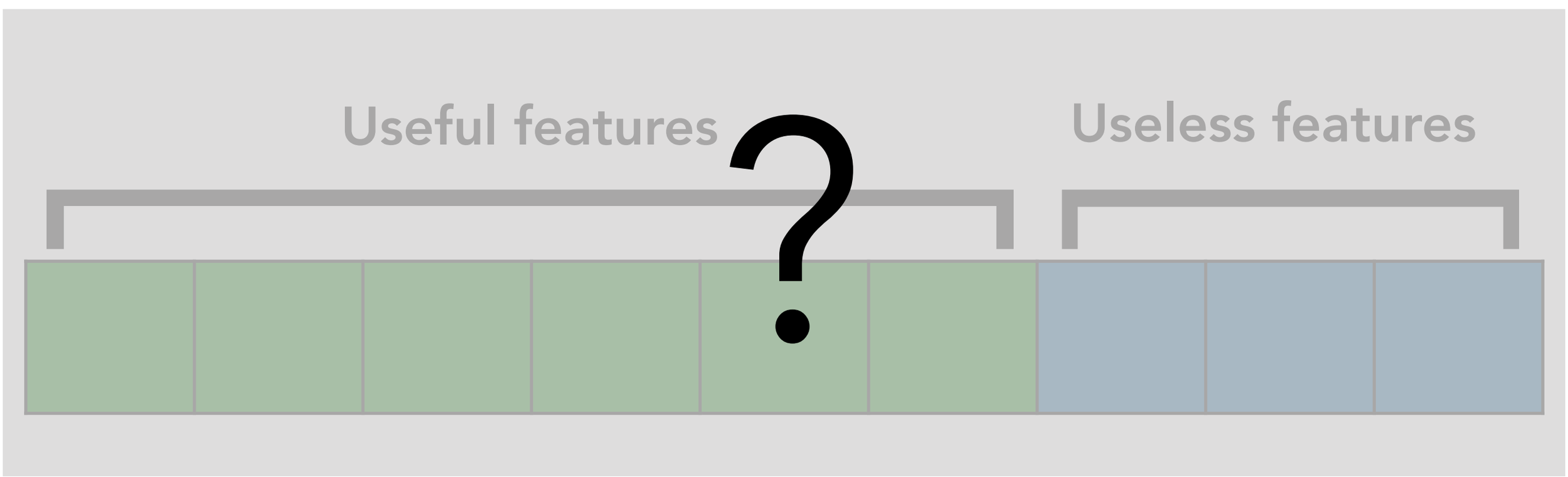


cat

Result: Good accuracy on the **original** test set

(e.g., 78% on CIFAR-10 cats vs. dogs)

What is our model missing?



Fixing our conceptual model

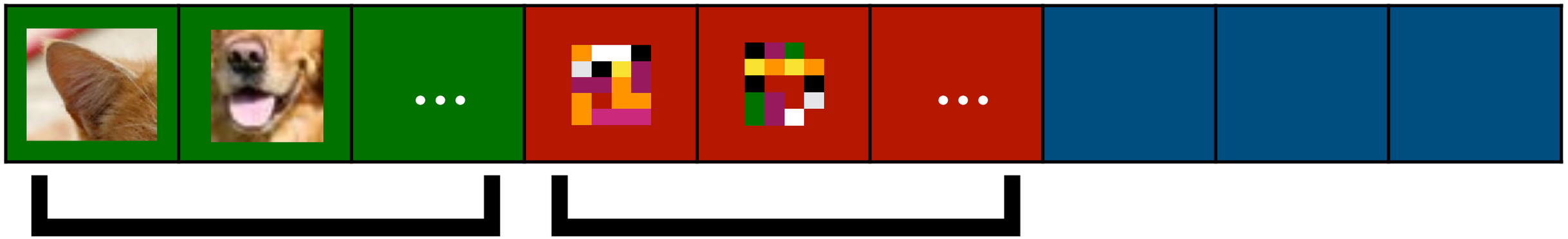
Useful features (used to classify)

Useless features

Fixing our conceptual model

Useful features (used to classify)

Useless features



Robust features

Non-robust features

Adversarial examples **flip some useful features**

Try at home

Pre-generated Datasets

github.com/MadryLab/constructed-datasets

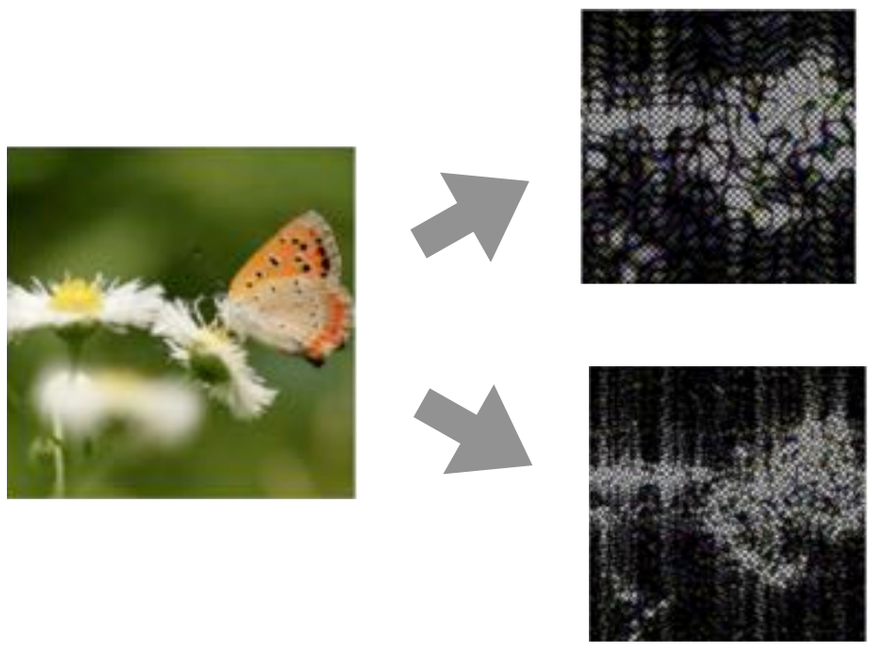
Adversarial examples & training library

github.com/MadryLab/robustness

Similar findings

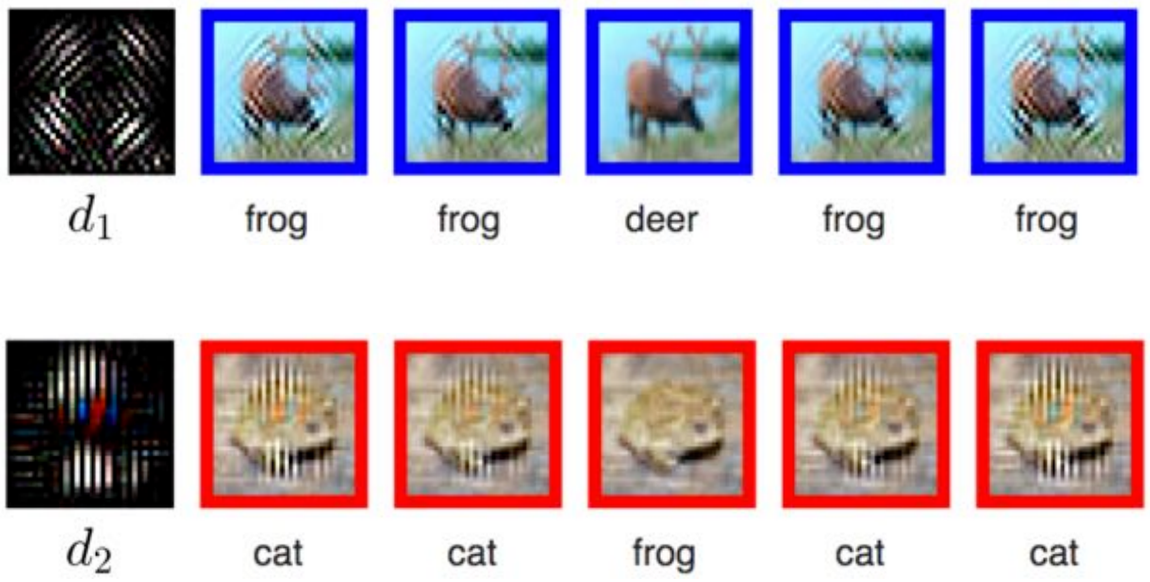
High-frequency components

[Yin et al. 2019]



Predictive linear directions

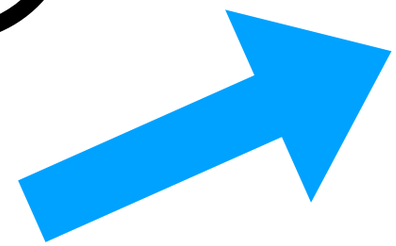
[Jetley et al. 2018]

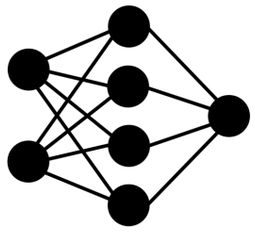


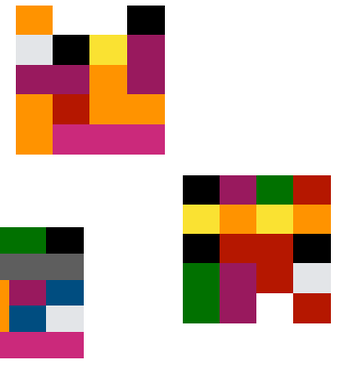
Take away: Models rely on **unintuitive features**

Back to interpretations

dog







Equally valid classification methods

Model faithful explanations

Interpretability methods might be
hiding relevant information

→ **Human-meaningless** does **not** mean **useless**

→ Are we **improving explanations** or **hiding things?**

→ Better visual quality might have nothing to do with model

[Adebayo et al. 2018]

How do we get better saliency?

Gradient of standard models are faithful but don't look great

Better interpretability (human priors)

Can hide too much!

Better models

How do we get better saliency?

Gradient of standard models are faithful but don't look great

Better interpretability (human priors)

Can hide too much!

Better models

One idea: Robustness as prior

Key idea: Force models to ignore non-robust features

Standard Training:

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

Robust Training:

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

Set of invariances

Exercise 4: Adv. Examples II (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)

What did we see?

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

Exercise 5: Robust models (5m)

Changing model predictions: larger perturbations

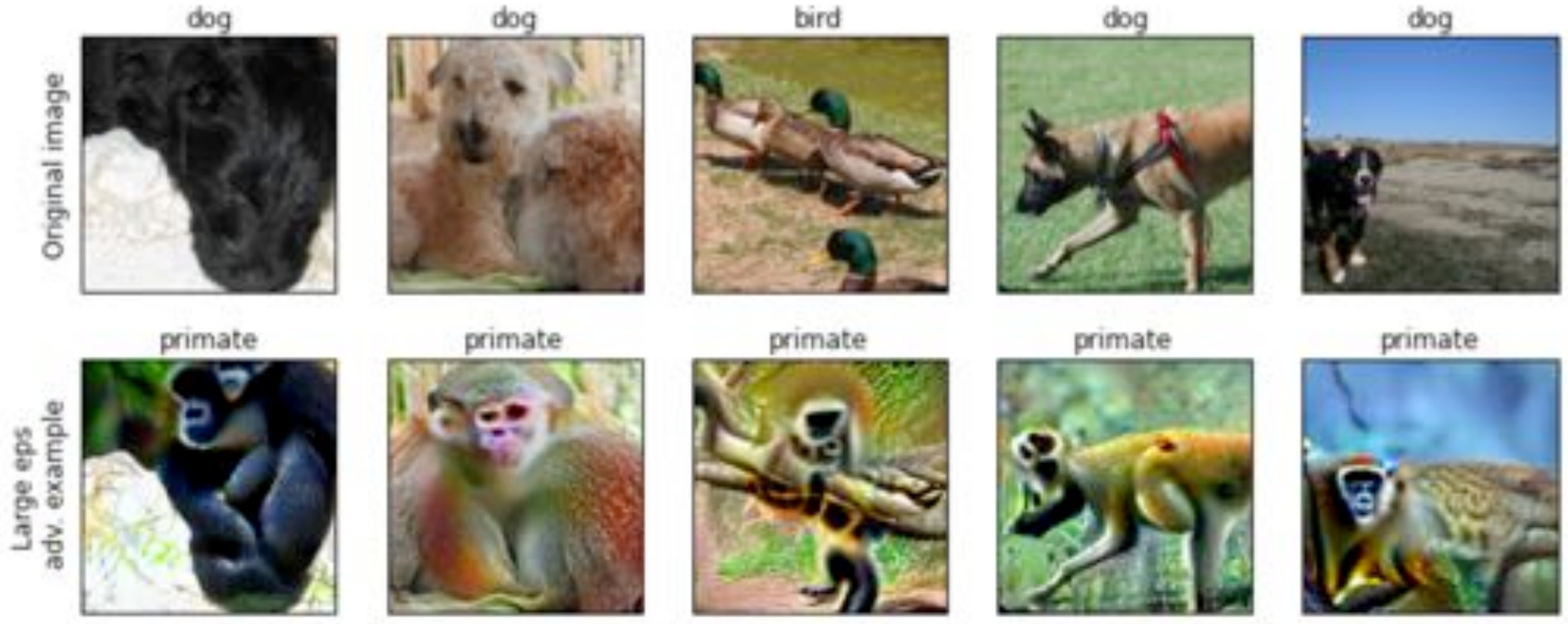
→ Goal: modify input so that model prediction changes

- Again, gradient descent to make prediction target class
- Since small epsilons don’t work, try **larger ones**

→ What does the modified input look like?

What did we see?

Target class: ``Primate``



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

Exercise 6.1: Robust gradients (5m)

Explore **robust** model sensitivity via gradients

→ Visualize gradients for different inputs

→ Compare to grad (and SmoothGrad) for standard models

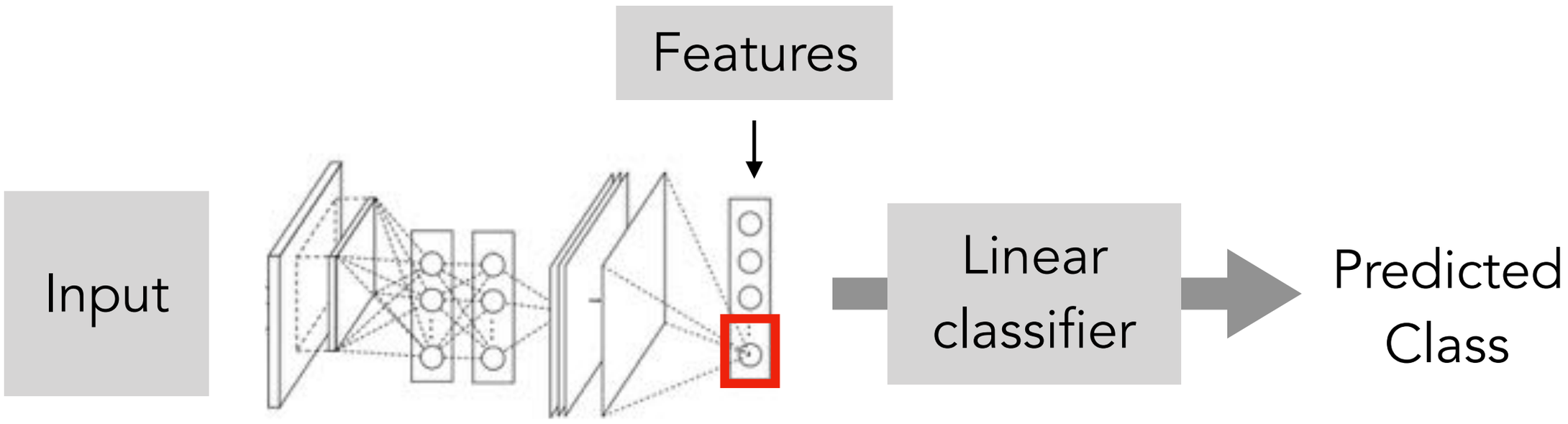
What did we see?

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 6.2: Visualize Features (10m)

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

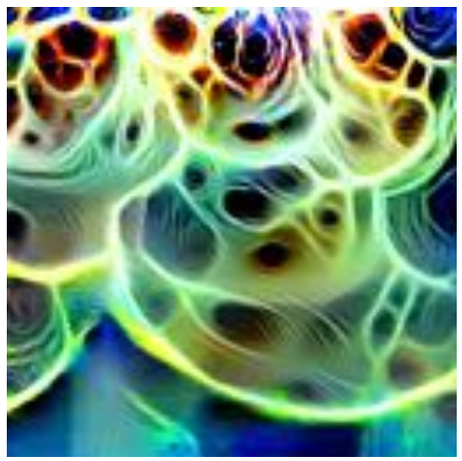
→ Optional: Repeat for standard models

→ Optional: Start optimization from noise instead

What did we see?

Maximizing inputs

Neuron 200



Neuron 500

Neuron 1444

Top-activating test images

High-level concepts

Takeaways

Nice-looking explanations might hide things

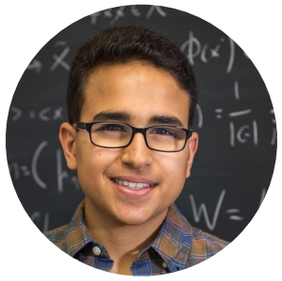
Models can rely on weird features

Robustness can be a powerful feature prior

“Robust Features”

Based on joint work with

Logan Engstrom



Andrew Ilyas

Brandon Tran

Alexander Turner

Aleksander Mądry

`robustness`

`gradsci.org`