CSC413/2516 Lecture 5: Interpretability

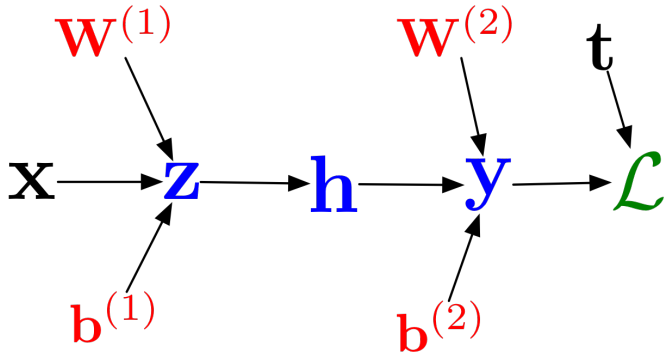
Jimmy Ba and Bo Wang

Overview

- We've seen convolutional neural networks are very successful computer vision models.
  - But, how do we know the network has learnt useful patterns from the training set?
- The interpretation of deep learning models is a challenge due to their size, complexity, and often opaque internal state.
- In this lecture, we discuss a few some tools to help understand the behavior of ML models.

Overview

- Recall the computation graph:



- From this graph, you could compute $\partial \mathcal{L}/\partial \mathrm{x}$, but we never made use of this.
- Basic idea: $\partial \mathcal{L}/\partial \mathrm{x}$ contains the model's sensitivity wrt changes of its input. It could be useful for interpreting or breaking the model!

Overview

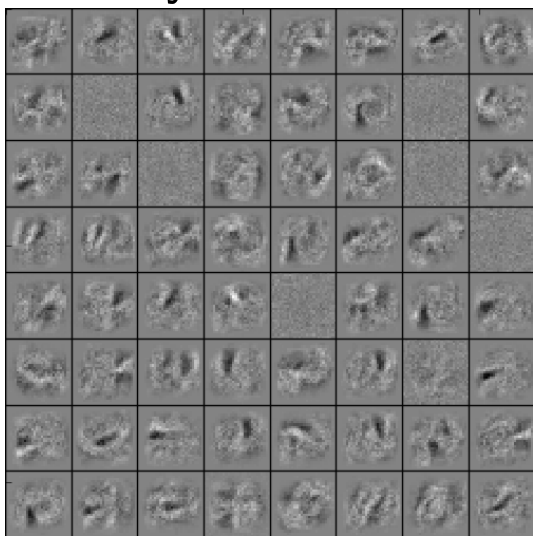
Use cases of input gradients:

- Visualizing what learned features represent
  - Visualizing image gradients to give us per-image feature visualization
  - Optimizing an image to maximize activations
- Adversarial inputs
- "Deep Dream"

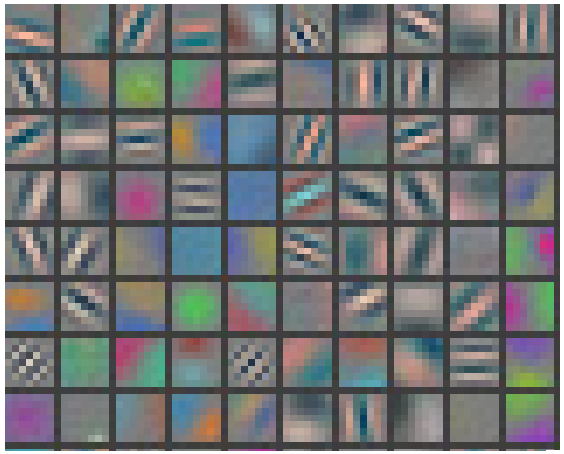
Feature Visualization

- Recall: we can understand what first-layer features are doing by visualizing the weight matrices.

Fully connected



Convolutional



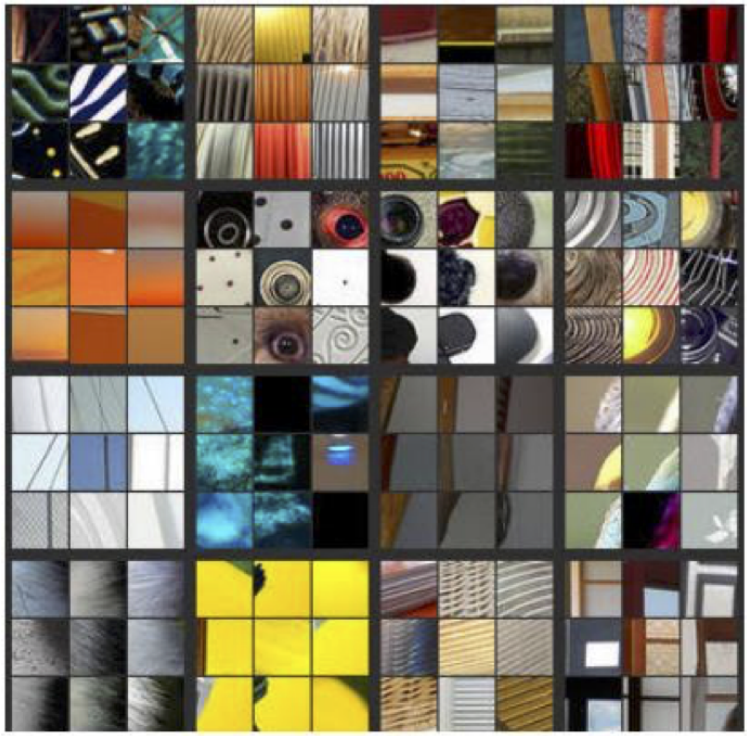
- What to do with the higher-level features?

Feature Visualization

- One way to formalize: pick the images in the training set which activate a unit most strongly.
- Here's the visualization for layer 1:

Feature Visualization

- Layer 3:



Feature Visualization

- Layer 4:

Feature Visualization

- Layer 5:

Feature Visualization

- Higher layers seem to pick up more abstract, high-level information.
  - Problem: can't tell what the unit is actually responding to in the image!

Overview

Use cases of input gradients:

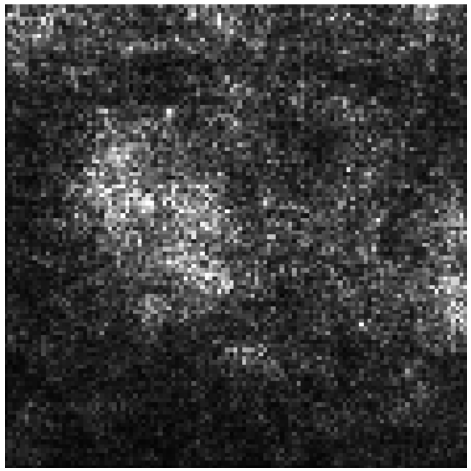
- Visualizing what learned features represent
  - **Visualizing image gradients to give us per-image feature visualization**
  - Optimizing an image to maximize activations
- Adversarial inputs
- "Deep Dream"

Feature Visualization

- Input gradients can be noisy and hard to interpret.
- Take a good object recognition conv net (Alex Net) and compute the gradient of $\log p(y = \text{“deer”}|\text{x})$:

Original image

Gradient for “deer”



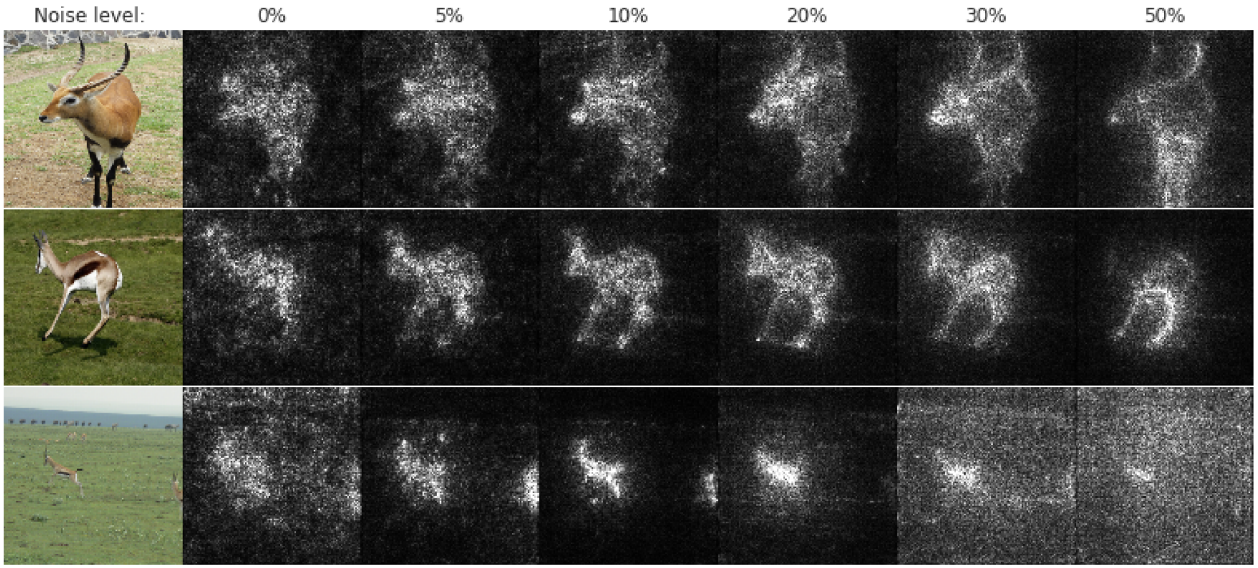
- This is partially due to the steepest directions are local sensitivity wrt the current input pixels. It is difficult to pick out important global features from one instance of the local changes.

Feature Visualization

- SmoothGrad is a method to estimate a global "saliency" map.
- Do the backward pass on a few noisy version of the input images, then average their input gradients.

$$S_{deer} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial \mathcal{L}_{deer}}{\partial \mathrm{x}}(\mathrm{x} + \epsilon_i), \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I)$$

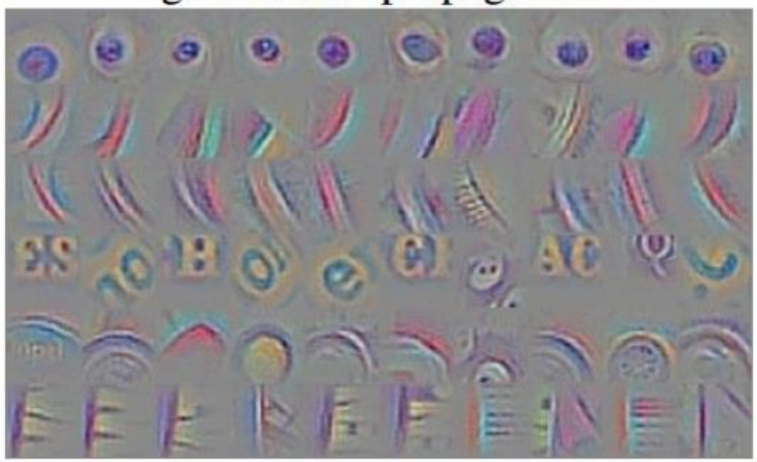
- We want to average out the local sensitity effect from slightly perturbed input images.
- Results



Cautionary Tales of Image Gradients

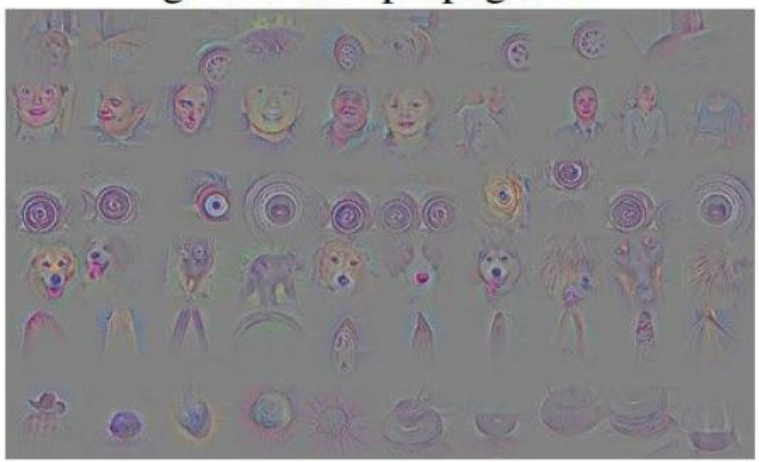
This looks very convincing!

guided backpropagation



corresponding image crops

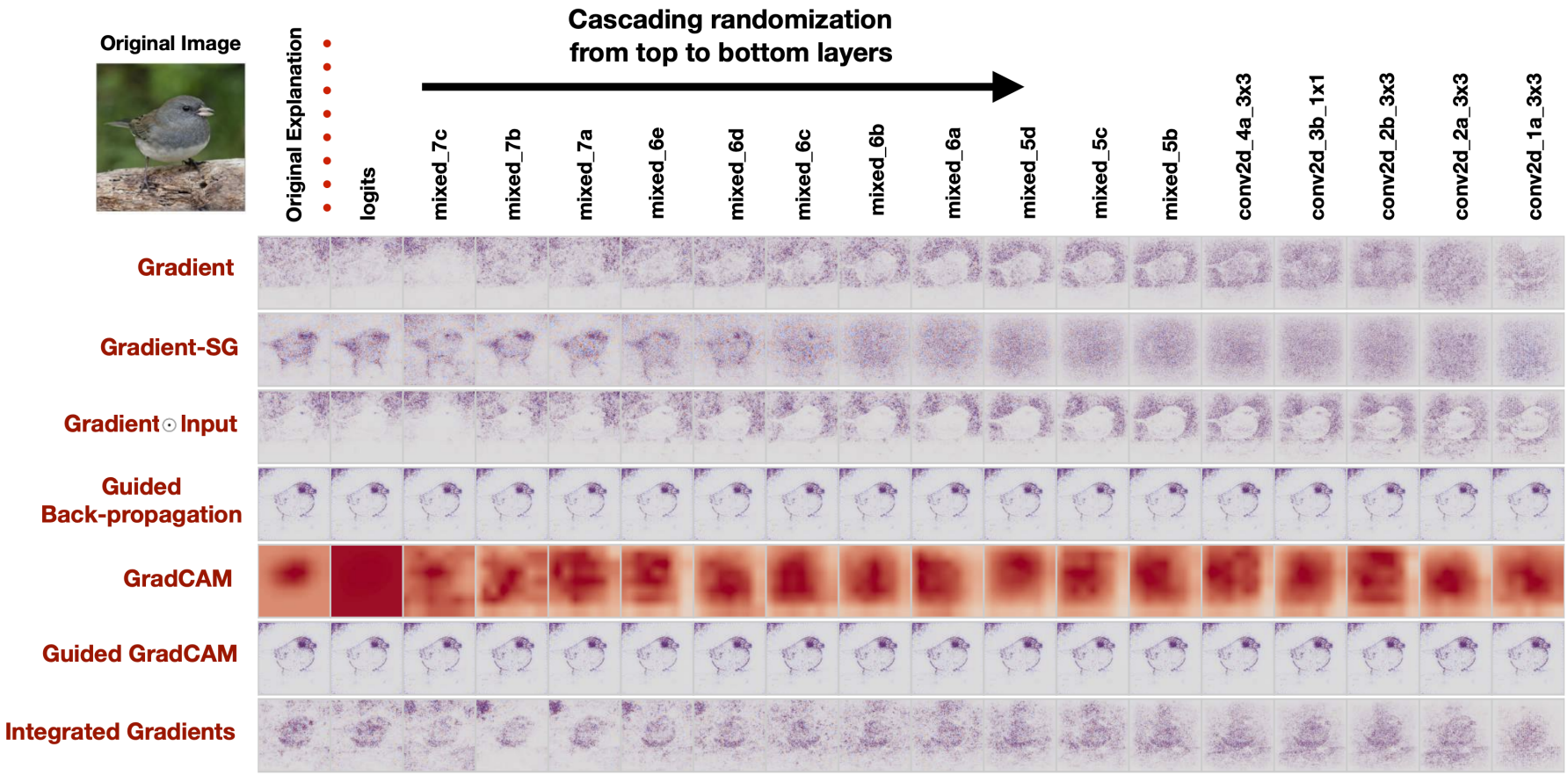
guided backpropagation



corresponding image crops

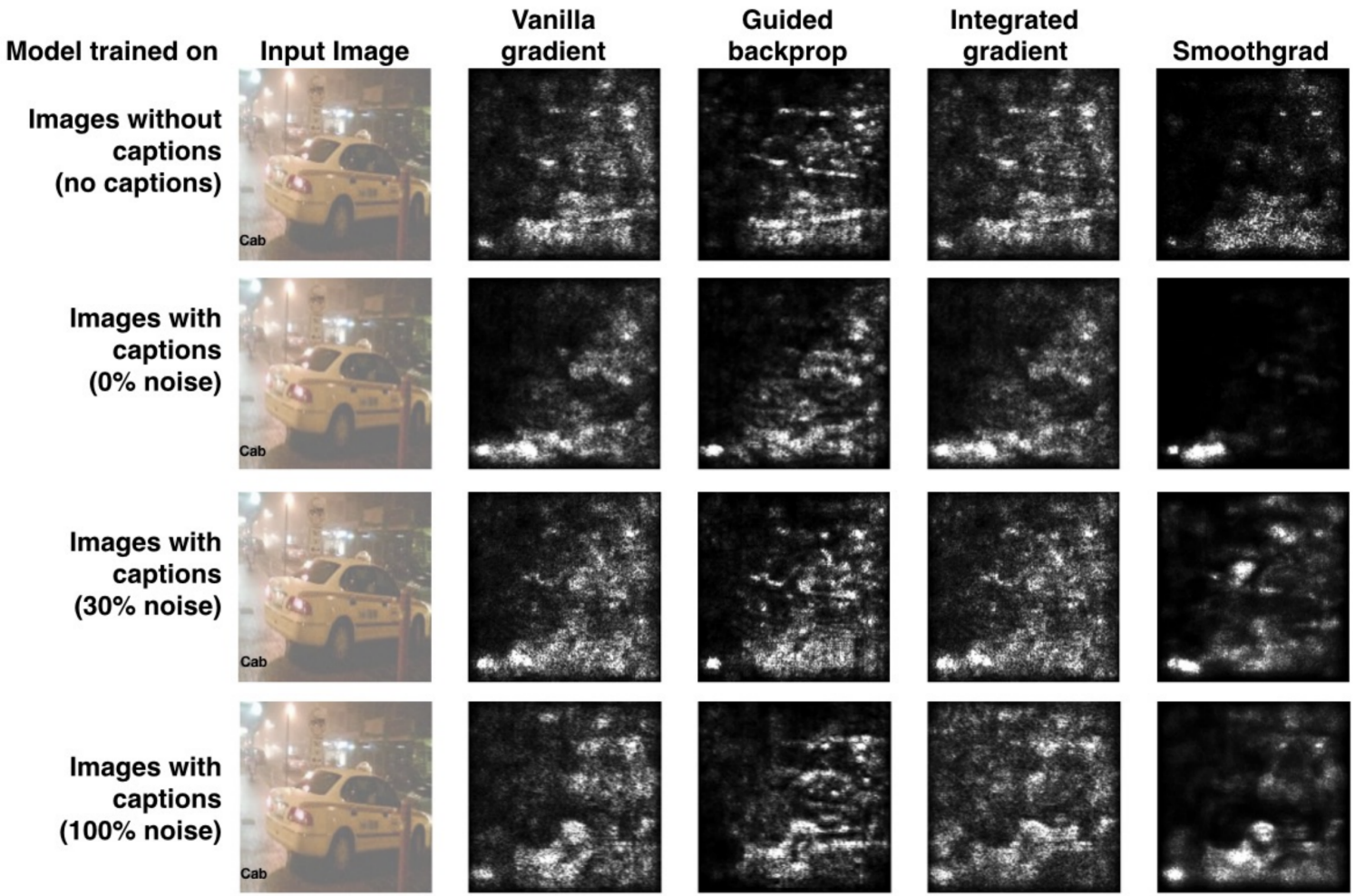
Springerberg et al, Striving for Simplicity: The All Convolutional Net (ICLR 2015 workshops)

Cautionary Tales of Image Gradients



Sanity check for saliency maps, `http://papers.nips.cc/paper/8160-sanity-checks-for-saliency-maps.pdf`

Cautionary Tales of Image Gradients



Testing with Concept Activation Vectors, `https://arxiv.org/pdf/1711.11279.pdf`

Overview

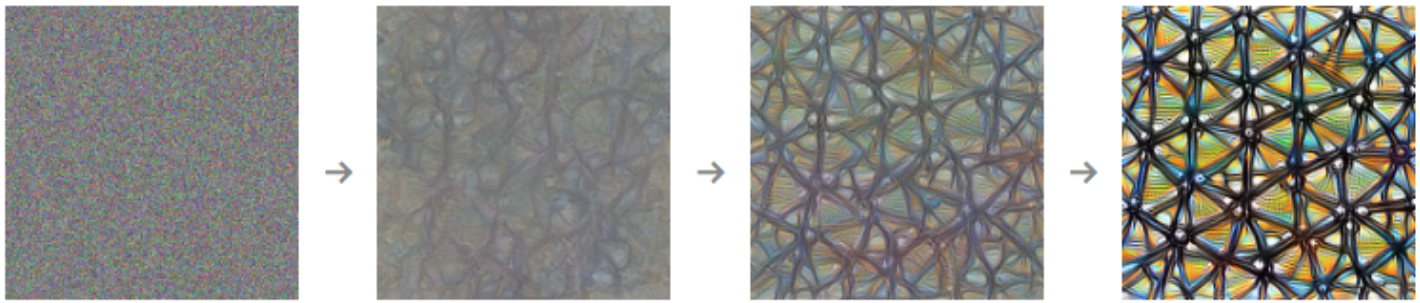
Use cases of input gradients:

- Visualizing what learned features represent
  - Visualizing image gradients to give us per-image feature visualization
  - **Optimizing an image to maximize activations**
- Adversarial inputs
- "Deep Dream"

Gradient Ascent on Images

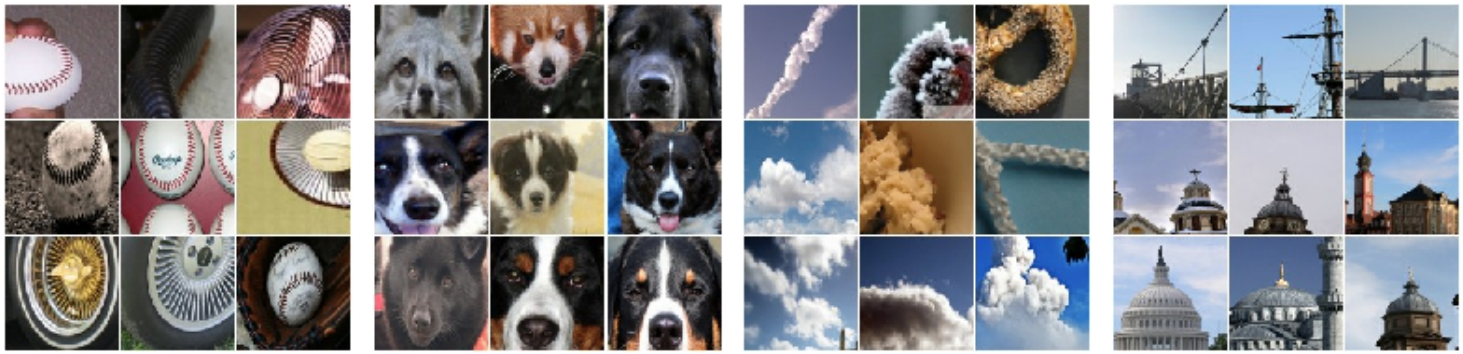
- Can do gradient ascent on an image to maximize the activation of a given neuron.
- Requires a few tricks to make this work; see `https://distill.pub/2017/feature-visualization/`

Starting from random noise, we optimize an image to activate a particular neuron (layer mixed4a, unit 11).

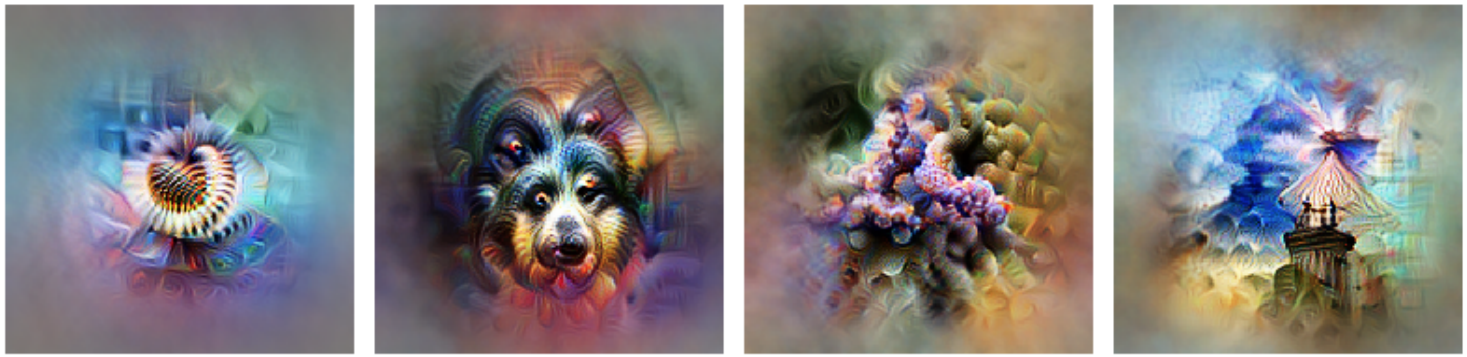


Gradient Ascent on Images

Dataset Examples show us what neurons respond to in practice



Optimization isolates the causes of behavior from mere correlations. A neuron may not be detecting what you initially thought.



Baseball—or stripes?
mixed4a, Unit 6

Animal faces—or snouts?
mixed4a, Unit 240

Clouds—or fluffiness?
mixed4a, Unit 453

Buildings—or sky?
mixed4a, Unit 492

Gradient Ascent on Images

- Higher layers in the network often learn higher-level, more interpretable representations

Edges (layer conv2d0) **Textures** (layer mixed3a) **Patterns** (layer mixed4a)

https://distill.pub/2017/feature-visualization/

Gradient Ascent on Images

- Higher layers in the network often learn higher-level, more interpretable representations

Parts (layers mixed4b & mixed4c) **Objects** (layers mixed4d & mixed4e)

https://distill.pub/2017/feature-visualization/

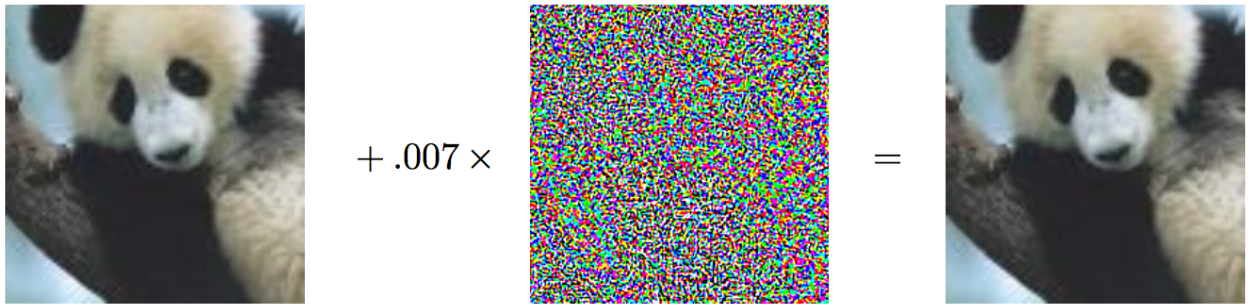
Overview

Use cases of input gradients:

- Visualizing what learned features represent
  - Visualizing image gradients to give us per-image feature visualization
  - Optimizing an image to maximize activations
- **Adversarial inputs**
- "Deep Dream"

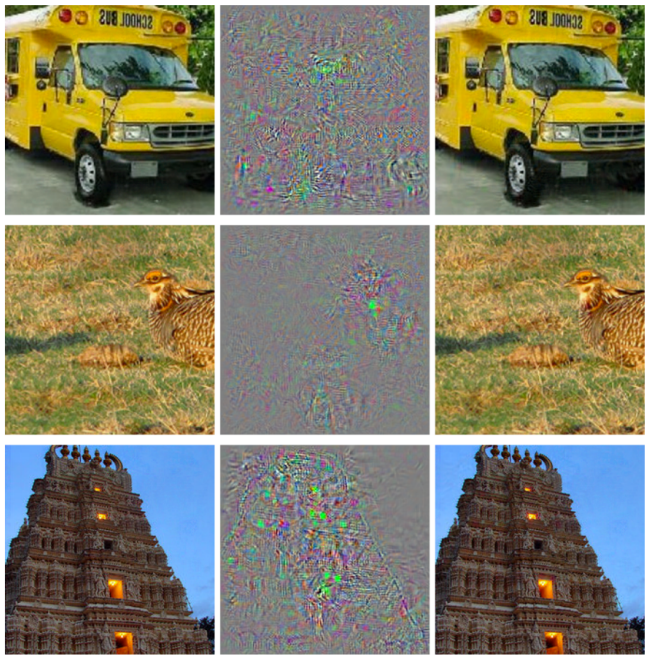
Adversarial Examples

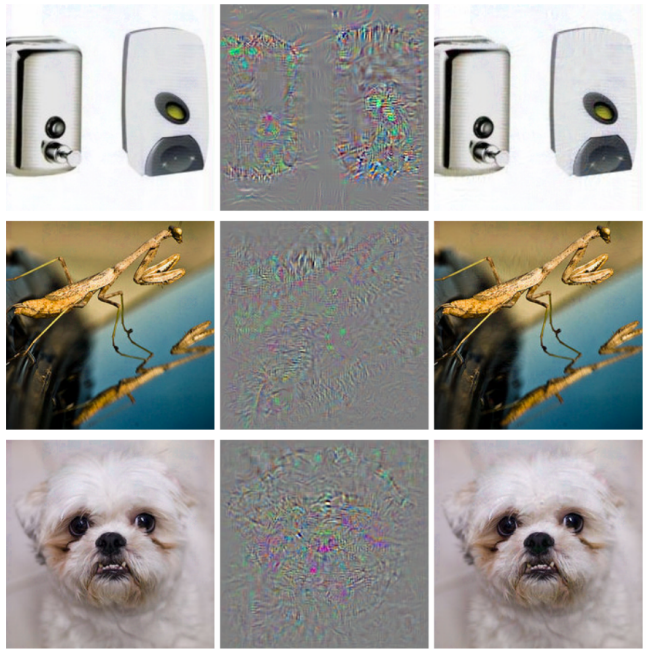
- One of the most surprising findings about neural nets has been the existence of adversarial inputs, i.e. inputs optimized to fool an algorithm.
- Given an image for one category (e.g. "cat"), compute the image gradient to maximize the network's output unit for a different category (e.g. "dog")
  - Perturb the image very slightly in this direction, and chances are, the network will think it's a dog!
  - Works slightly better if you take the sign of the entries in the gradient; this is called the fast gradient sign method.



Adversarial Examples

- The following adversarial examples are misclassified as ostriches. (Middle = perturbation $\times 10$.)





Adversarial Examples

- 2013: ha ha, how cute!
  - The paper which introduced adversarial examples was titled "Intriguing Properties of Neural Networks."

Adversarial Examples

- 2013: ha ha, how cute!
  - The paper which introduced adversarial examples was titled "Intriguing Properties of Neural Networks."
- 2018: serious security threat
  - Nobody has found a reliable method yet to defend against them.
    - 7 of 8 proposed defenses accepted to ICLR 2018 were cracked within days.
  - Adversarial examples transfer to different networks trained on a totally separate training set!
  - You don't need access to the original network; you can train up a new network to match its predictions, and then construct adversarial examples for that.
    - Attack carried out against proprietary classification networks accessed using prediction APIs (MetaMind, Amazon, Google)

Adversarial Examples

- You can print out an adversarial image and take a picture of it, and it still works!

(a) Printout (b) Photo of printout (c) Cropped image

- Can someone paint over a stop sign to fool a self-driving car?

Adversarial Examples

- An adversarial example in the physical world (network thinks it's a gun, from a variety of viewing angles!)

Overview

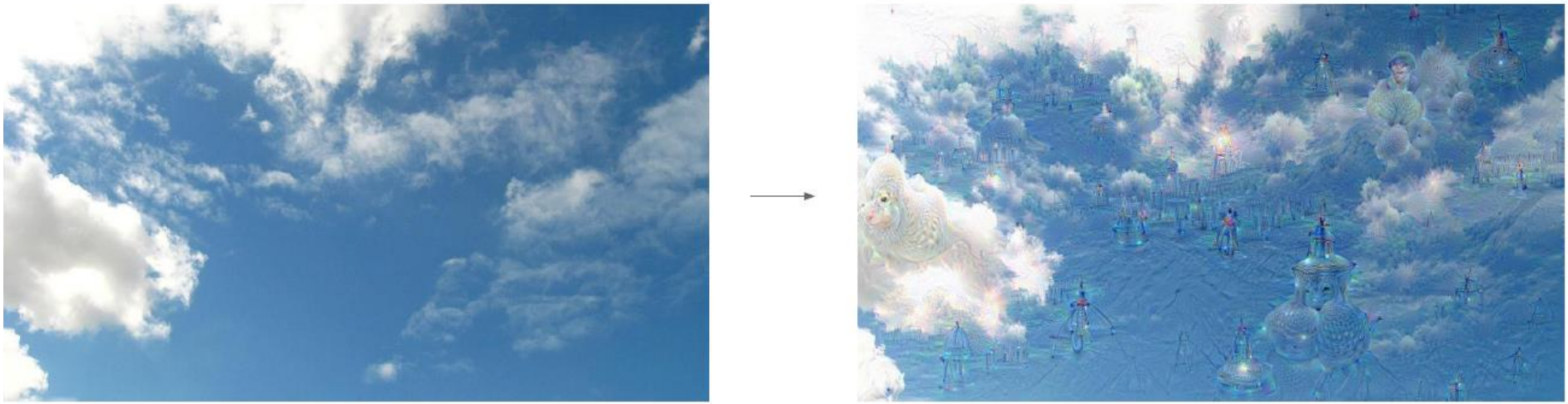
Use cases of input gradients:

- Visualizing what learned features represent
  - Visualizing image gradients to give us per-image feature visualization
  - Optimizing an image to maximize activations
- Adversarial inputs
- **"Deep Dream"**

Deep Dream

- Start with an image, and run a conv net on it.
- Pick a layer in the network.
- Change the image such that units which were already highly activated get activated even more strongly. "Rich get richer."
  - I.e., set $\overline{h} = h$, and then do backprop.
  - Aside: this is a situation where you'd pass in something other than 1 to `backward_pass` in autograd.
- Repeat.
- This will accentuate whatever features of an image already kind of resemble the object.

Deep Dream



Deep Dream


"Admiral Dog!"


"The Pig-Snail"


"The Camel-Bird"


"The Dog-Fish"

Deep Dream