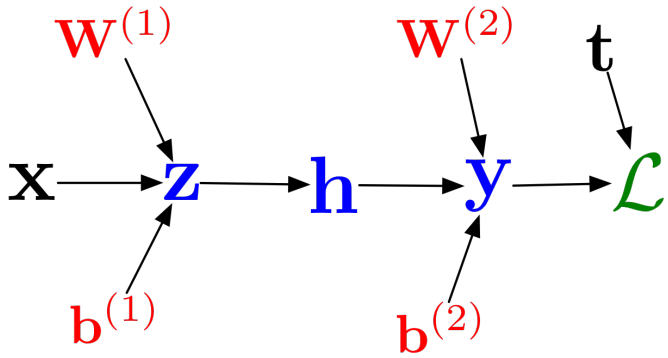
CS490/590 Lecture 12: Optimizing the Input

Eren Gultepe
Department of Computer Science SIUE Spring 2021

Adapted from Roger Grosse and Jimmy Ba

Overview

- Recall the computation graph:



- From this graph, you could compute $\partial \mathcal{L} / \partial \mathbf{x}$, but we never made use of this.
- This lecture: lots of fun things you can do by running gradient descent on the *input*!

Overview

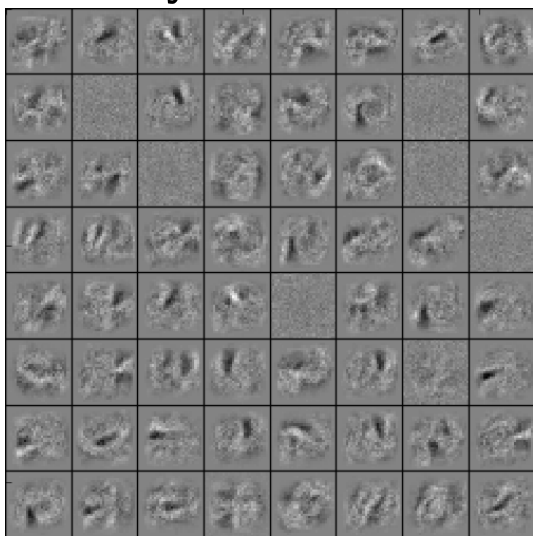
Use cases for input gradients:

- Visualizing what learned features represent
  - Visualizing image gradients
  - Optimizing an image to maximize activations
- Adversarial inputs
- "Deep Dream"

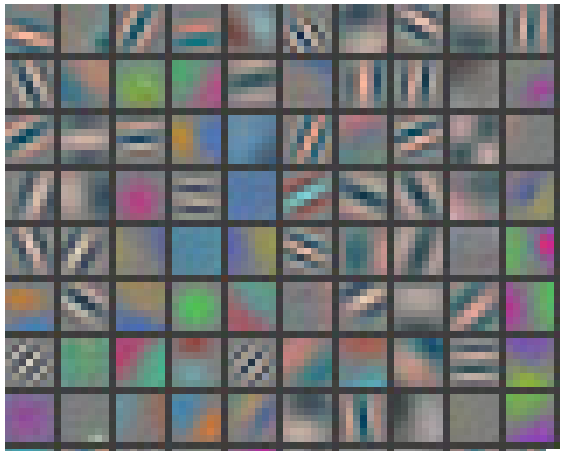
Feature Visualization

- Recall: we can understand what first-layer features are doing by visualizing the weight matrices.
- Higher-level weight matrices are hard to interpret.

Fully connected



Convolutional



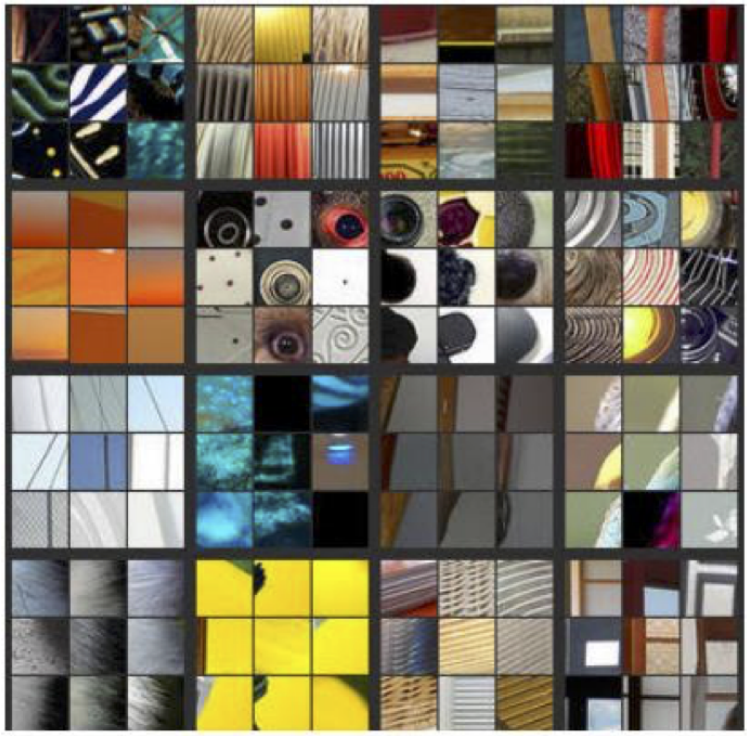
- The better the input matches these weights, the more the feature activates.
  - Obvious generalization: visualize higher-level features by seeing what inputs activate them.

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.
- Problems?

Feature Visualization

- Higher layers seem to pick up more abstract, high-level information.
- Problems?
  - Can't tell what the unit is actually responding to in the image.
  - We may read too much into the results, e.g. a unit may detect red, and the images that maximize its activation will all be stop signs.

Feature Visualization

- Higher layers seem to pick up more abstract, high-level information.
- Problems?
  - Can't tell what the unit is actually responding to in the image.
  - We may read too much into the results, e.g. a unit may detect red, and the images that maximize its activation will all be stop signs.
- Can use input gradients to diagnose what the unit is responding to. Two possibilities:
  - See how to change an image to increase a unit's activation
  - Optimize an image from scratch to increase a unit's activation

Overview

Use cases for input gradients:

- Visualizing what learned features represent
  - **Visualizing image gradients**
  - Optimizing an image to maximize activations
- Adversarial inputs
- "Deep Dream"

Feature Visualization

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

Original image

Gradient for “cat”

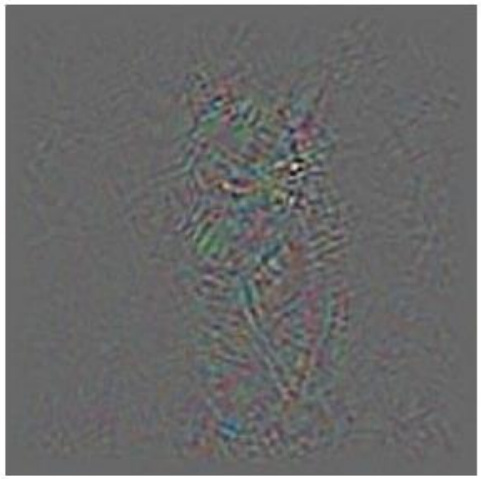
- The full explanation is beyond the scope of this course.
  - Part of it is that the network tries to detect cats everywhere; a pixel may be consistent with cats in one location, but inconsistent with cats in other locations.

Feature Visualization

- Guided backprop is a total hack to prevent this cancellation.
- Do the backward pass as normal, but apply the ReLU nonlinearity to all the activation error signals.

$$y = \mathrm{ReLU}(z) \qquad \bar{z} = \begin{cases} \bar{y} & \text{if } z > 0 \text{ and } \bar{y} > 0 \\ 0 & \text{otherwise} \end{cases}$$

- Note: this isn't really the gradient of anything!
- We want to visualize what excites a given unit, not what suppresses it.
- Results

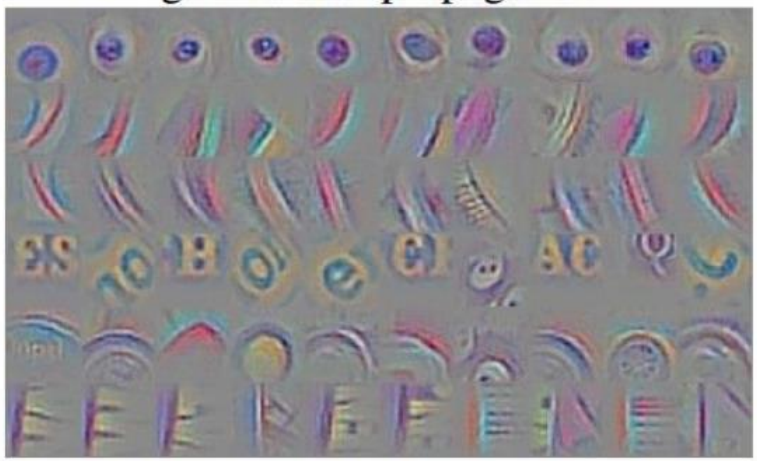


Backprop

Guided Backprop

Guided Backprop

guided backpropagation



corresponding image crops

guided backpropagation

corresponding image crops

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

Overview

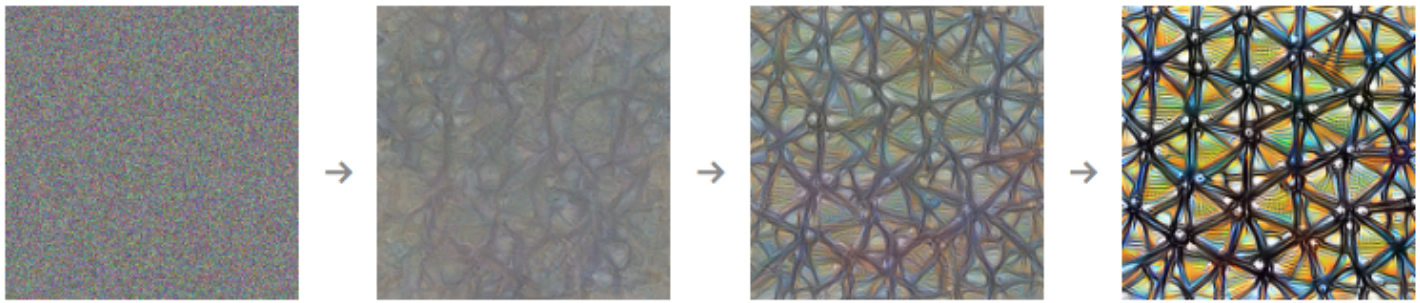
Use cases for input gradients:

- Visualizing what learned features represent
  - Visualizing image gradients
  - **Optimizing an image to maximize activations**
- Adversarial inputs
- "Deep Dream"

Gradient Ascent on Images

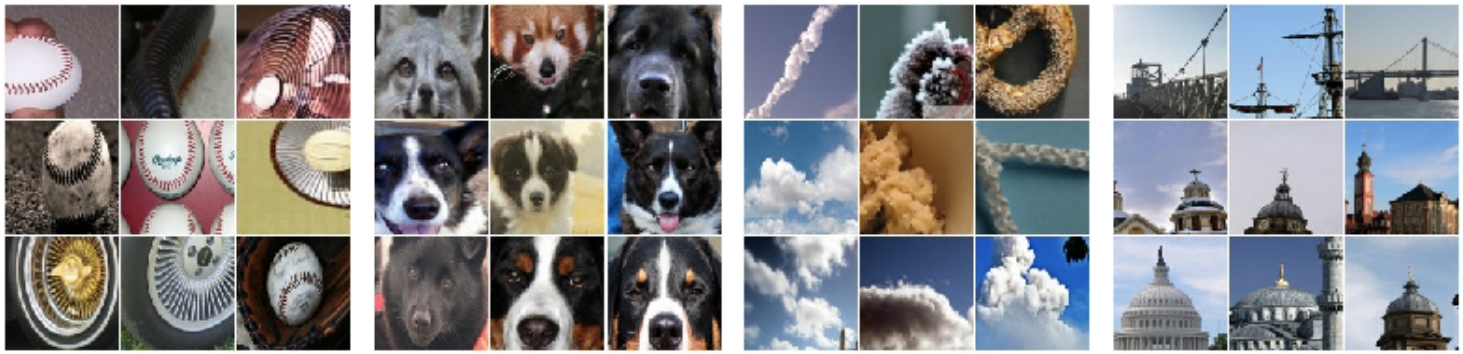
- Can do gradient ascent on an image to maximize the activation of a given neuron.
- Requires a few tricks to make this work; see `https://distill.pub/2017/feature-visualization/`

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

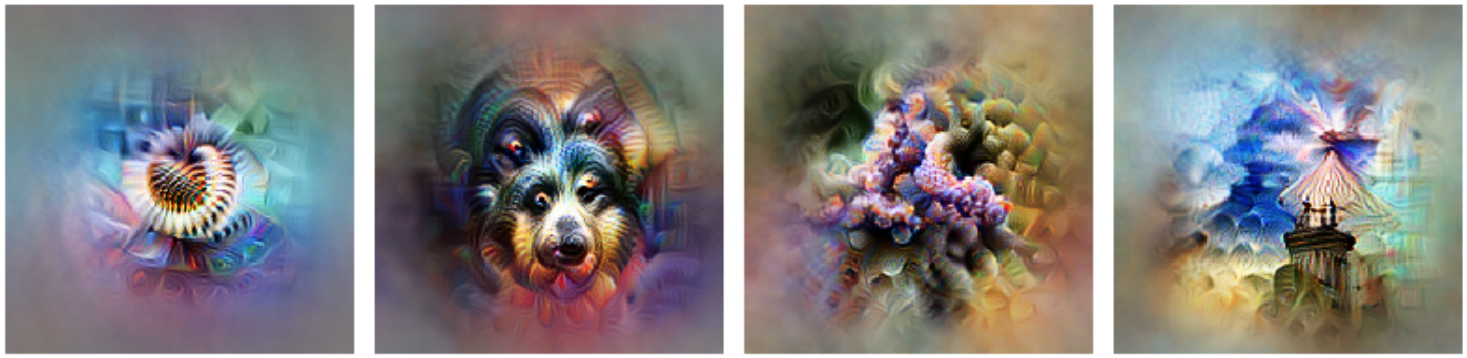


Gradient Ascent on Images

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/

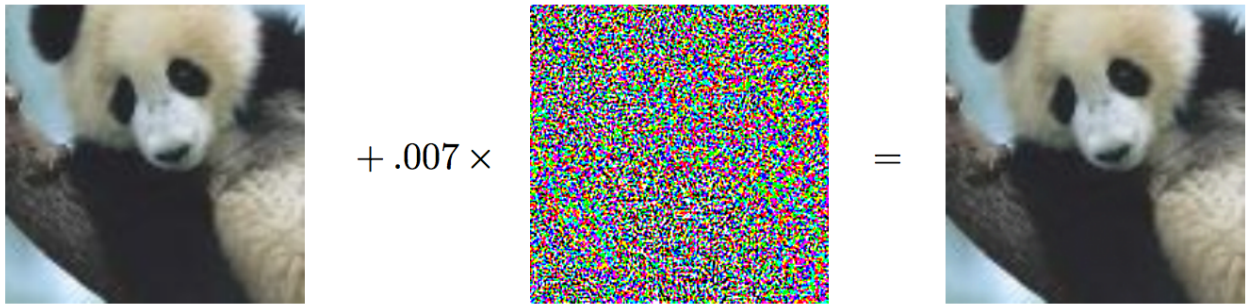
Overview

Use cases for input gradients:

- Visualizing what learned features represent
  - Visualizing image gradients
  - 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.



$\boldsymbol{x}$

"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

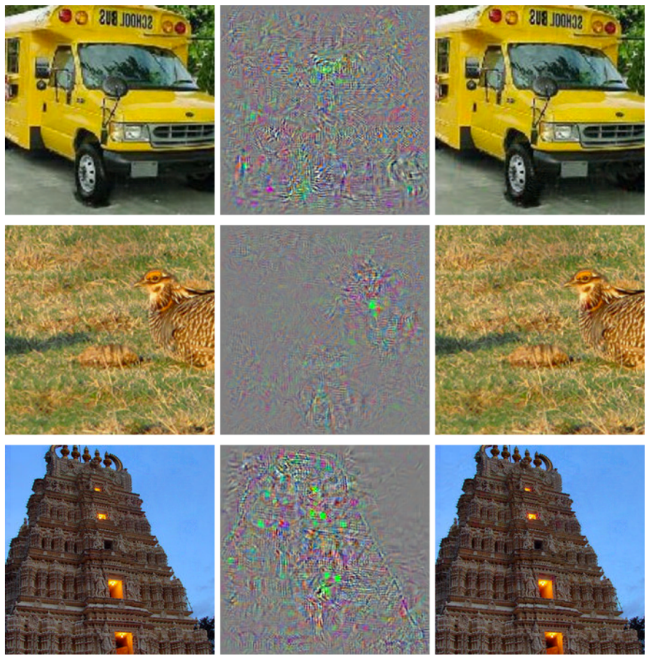
"nematode"
8.2% confidence

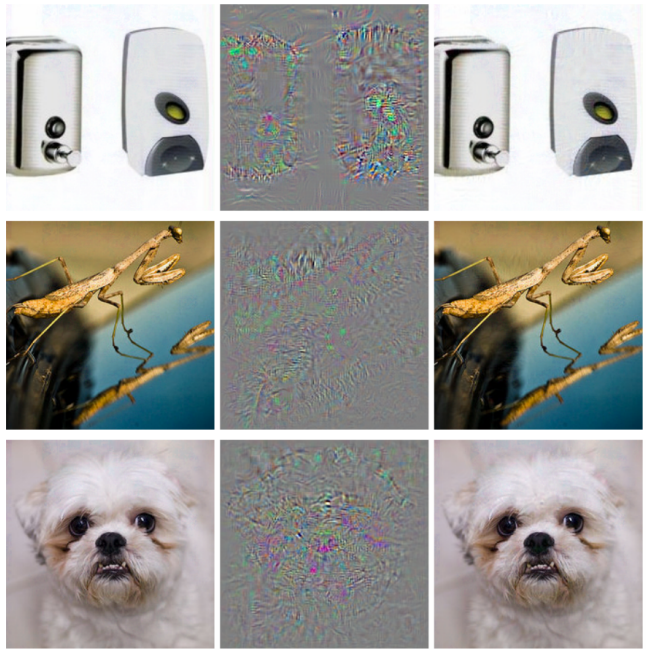
$\boldsymbol{x} + \epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

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

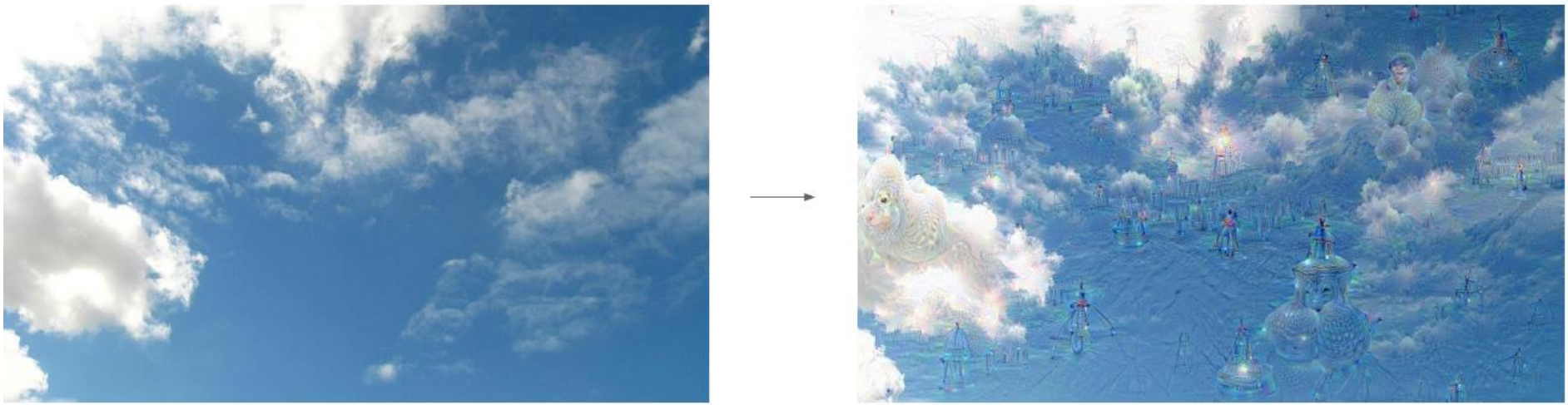
Use cases for input gradients:

- Visualizing what learned features represent
  - Visualizing image gradients
  - 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 $\overline{\mathbf{h}} = \mathbf{h}$, 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