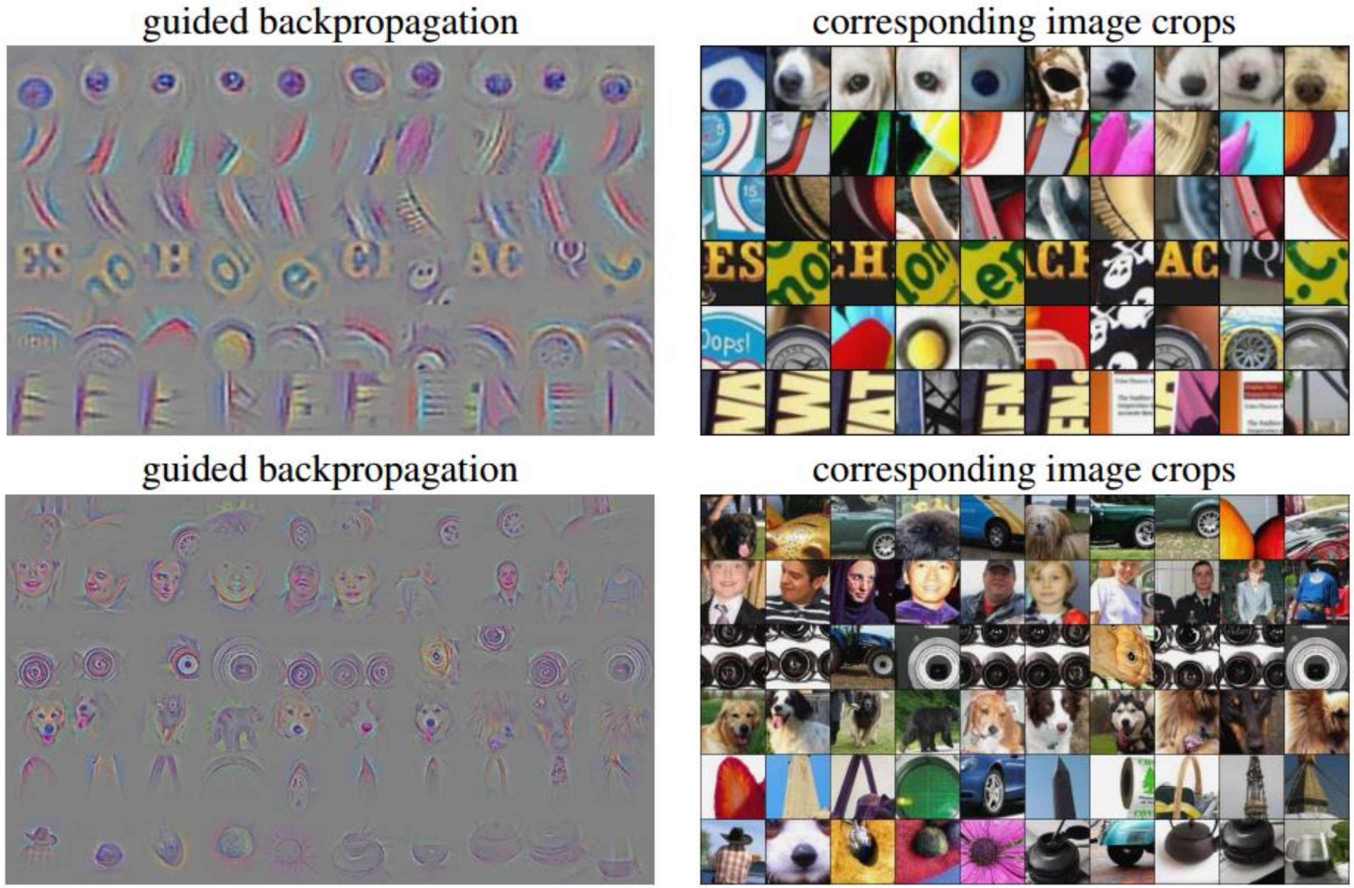
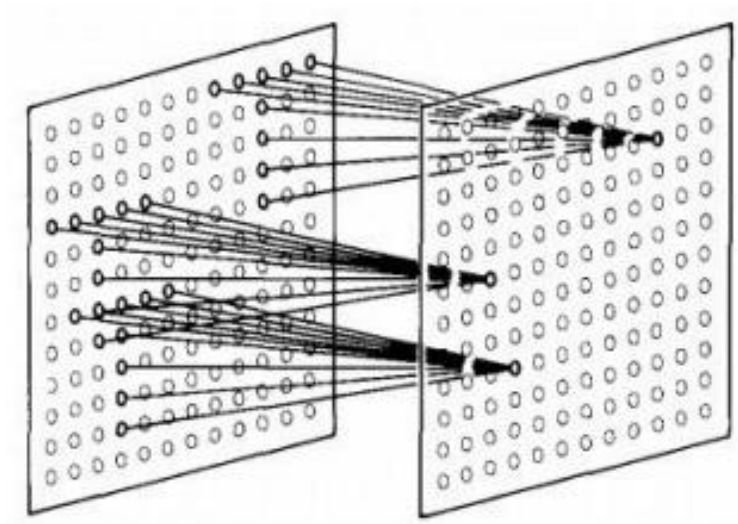
Understanding How ConvNets See



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

What Does a Neuron Do in a ConvNet? (1)

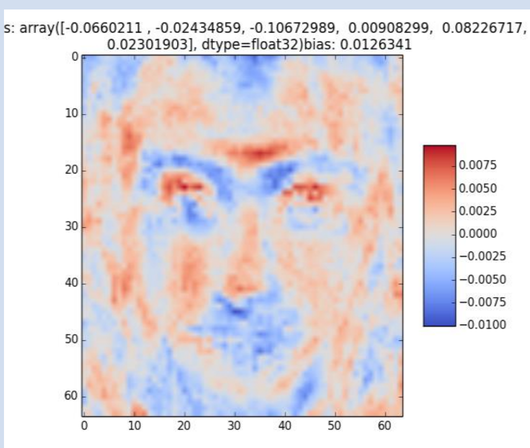
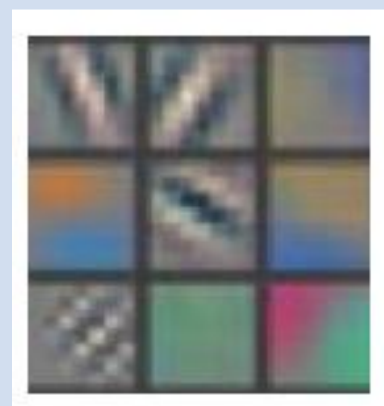
- A neuron in the first hidden layer computes a weighted sum of pixels in a patch of the image for which it is responsible



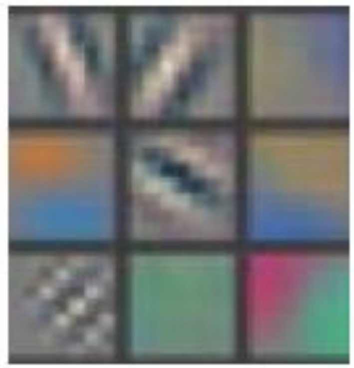
K. Fukushima, “Neurocognitron: A self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position” (Biol. Cybernetics 1980)

What Does a Neuron Do in a ConvNet? (2)

- For Neurons in the first hidden layer, we can visualize the weights.

Example weights for fully-connected single-hidden layer network for faces, for one neuron	Weights for 9 features in the first convolutional layer of a layer for classifying ImageNet images
	 Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”

What Does a Neuron Do in a ConvNet? (3)



- **The neuron would be activated the most if the input looks like the weight matrix**
- These are called "Gabor-like filters"
- The colour is due to the input being 3D. We visualize the strength of the weight going from each of the R, G, and B components

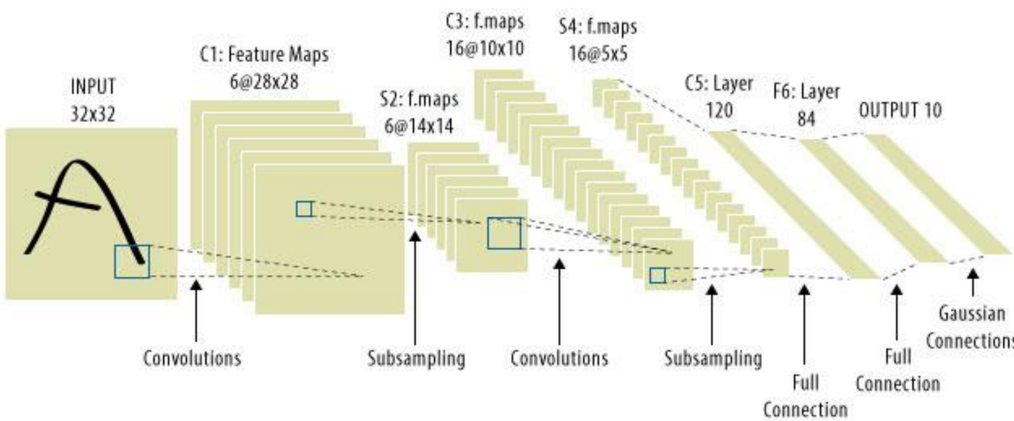
What Does a Neuron Do in a ConvNet (4)

- Another to figuring out what kind of images active the neuron: just try lots of images in a dataset, and see which ones active the neuron the most

For each feature, fine the 9 images that produce the highest activations for the neuron, and crop out the relevant patch

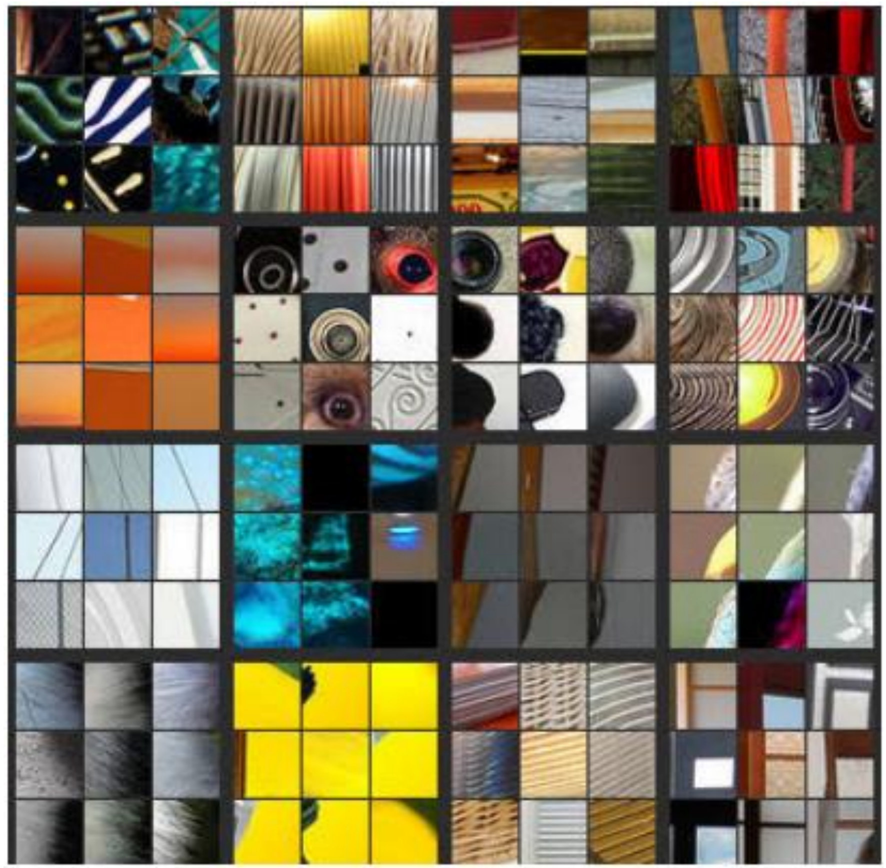
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks"

Aside: Relevant Patch?



- Each neuron is affected by some small patch in the layer below
- Can recursively figure out what patch in the input layer each neuron is affected
- Neurons in the top layers are affected by (almost) the entire image

This allows us to look at layers besides the first one: layer 3



Layer 4

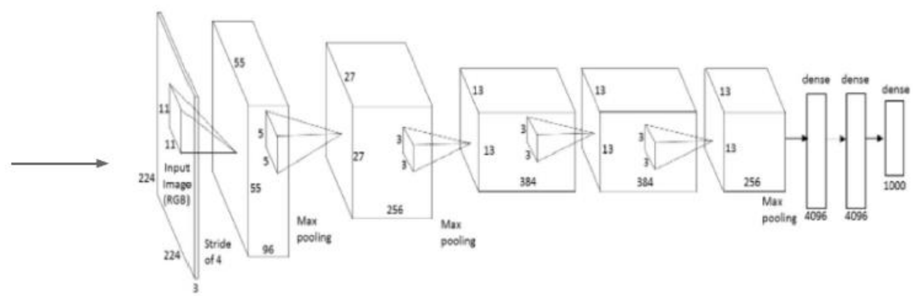
Layer 5

Which Pixels in the Input Affect the Neuron the Most?

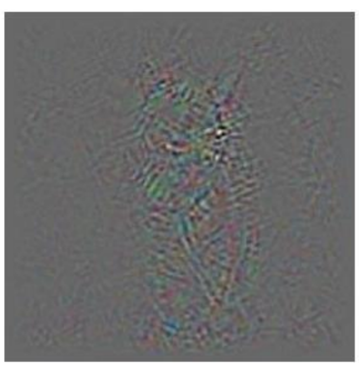
- Rephrased: which pixels would make the neuron not turn on if they had been different?
- In other words, for which inputs is $\frac{\partial neuron}{\partial x_i}$ large?

Typical Gradient of a Neuron

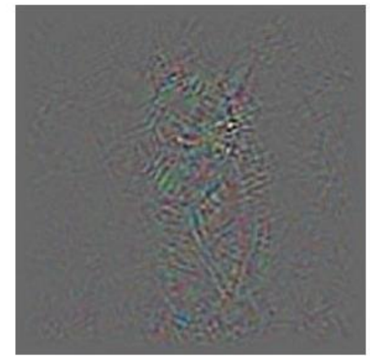
- Visualize the gradient of a particular neuron with respect to the input *x*
- Do a forward pass:



- Compute the gradient of a particular neuron using backprop:



Typical Gradient of a Neuron



- Mostly zero away from the object, but the results are not very satisfying
- Every pixel influences the neuron via multiple hidden neurons.
  The network is trying to detect kittens everywhere, and the same pixel could fit a kitten in one location but not another, leading to its overall effect on the kitten neuron to be 0
  (Explanation on the board)

“Guided Backpropagation”

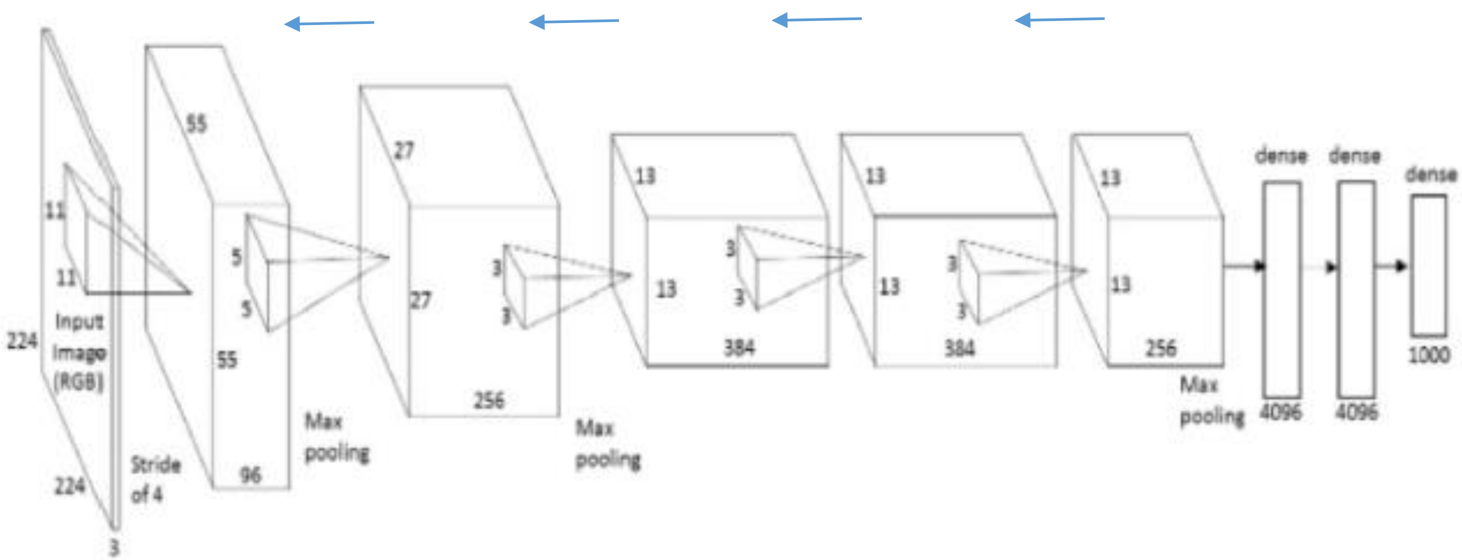
- Idea: neurons act like detectors of particular image features
- We are only interested in what image features the neuron detects, not in what kind of stuff it *doesn’t* detect
- So when propagating the gradient, we set all the negative gradients to 0
  - We don’t care if a pixel “suppresses” a neuron somewhere along the part to our neuron

Guided Backpropagation

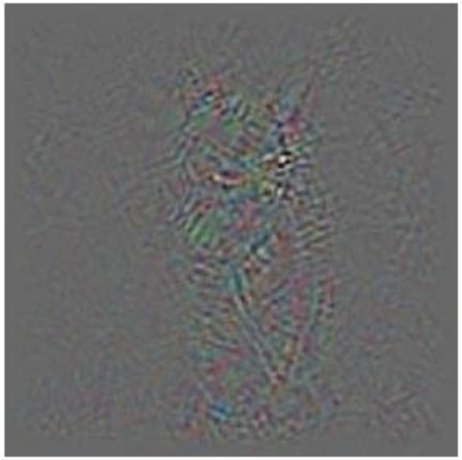
Compute gradient, zero out negatives, backpropagate

Compute gradient, zero out negatives, backpropagate

Compute gradient, zero out negatives, backpropagate



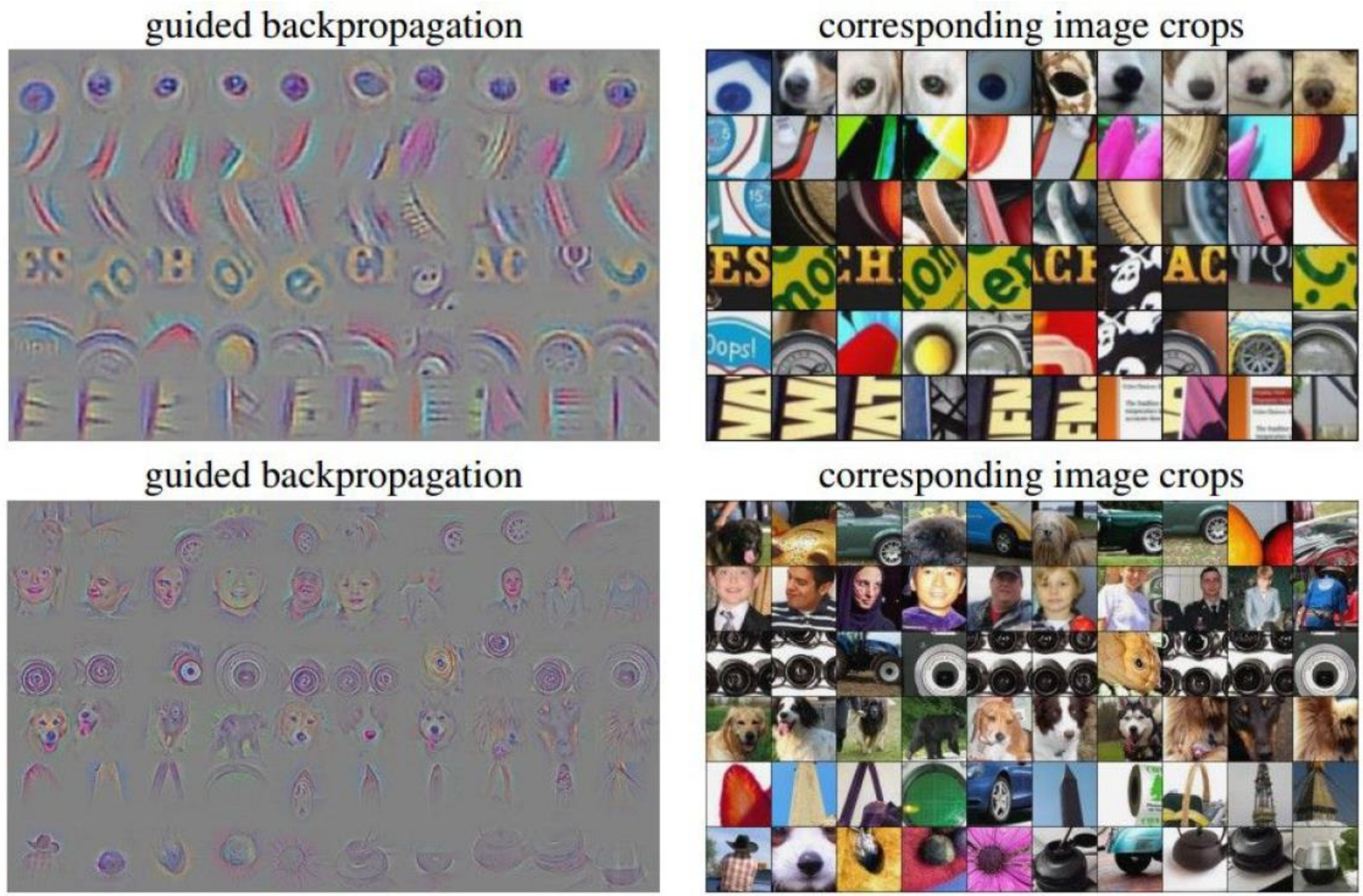
Guided Backpropagation



Backprop

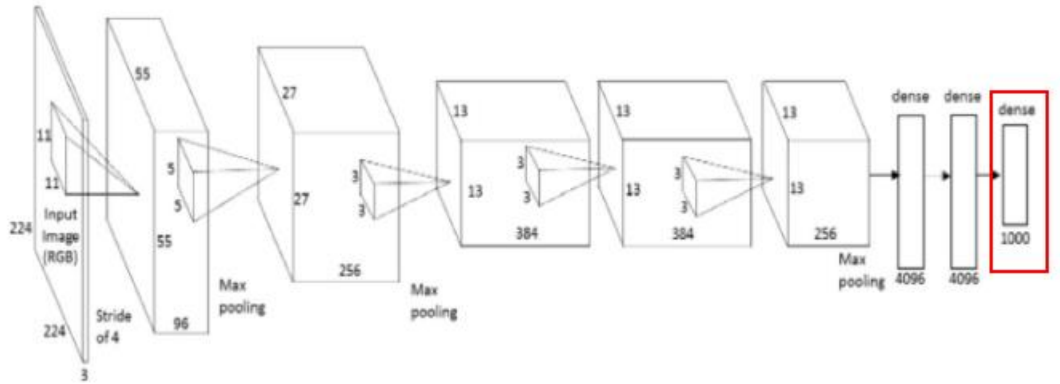
Guided Backprop

Guided Backpropagation



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

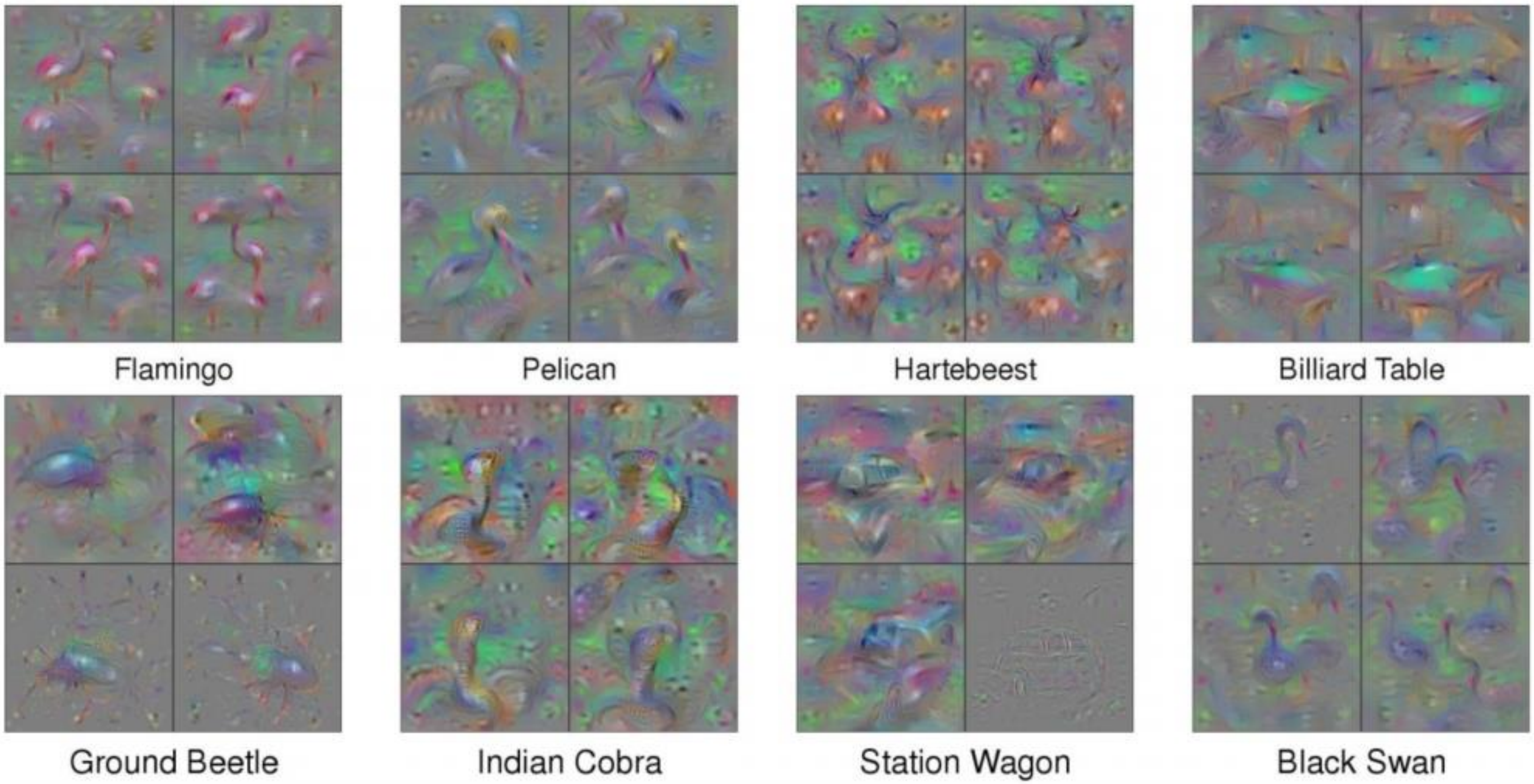
What About Doing Gradient Descent?



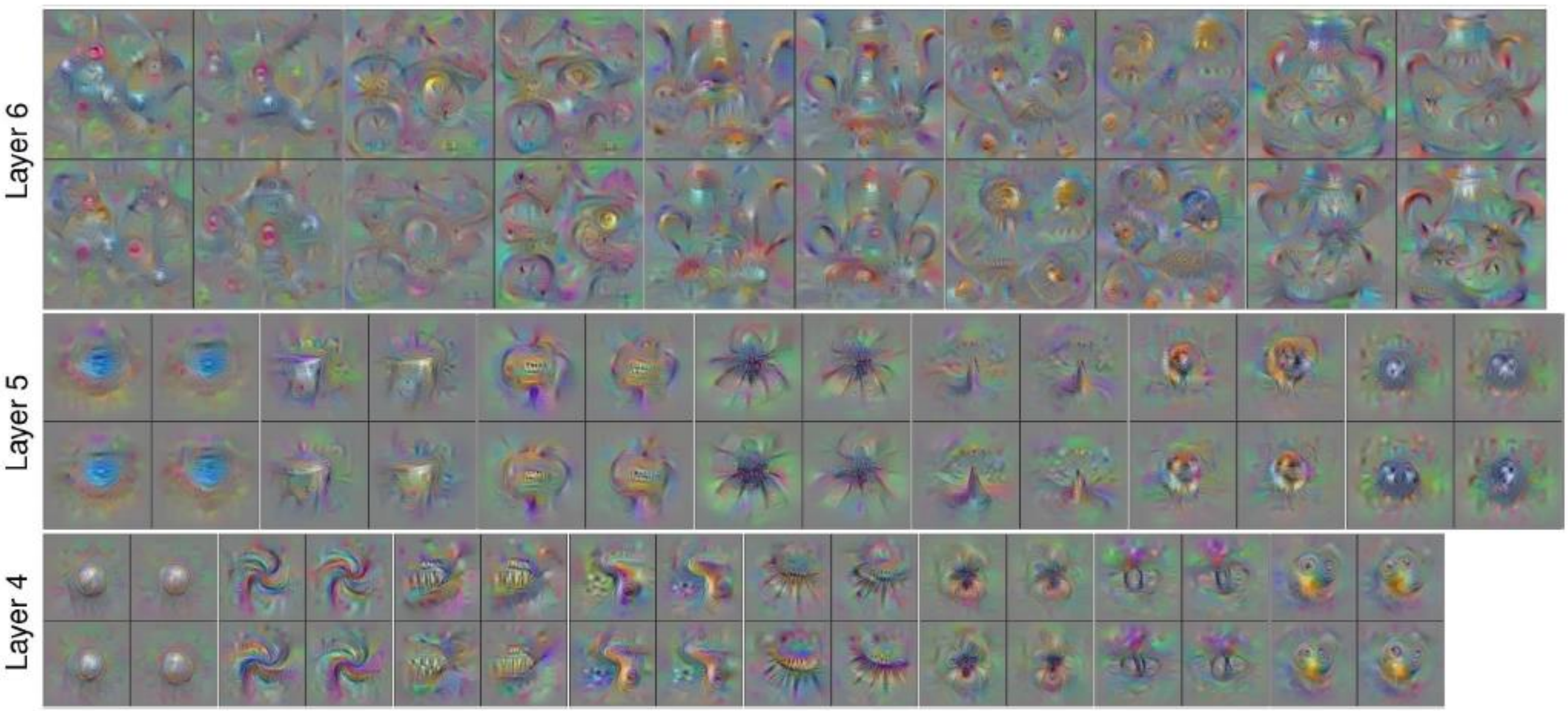
- What to maximize the i-th output of the softmax
- Can compute the gradient of the i-th output of the softmax with respect to the *input x* (the W's and b's are fixed to make classification as good as possible)
- Perform gradient descent on the *input*

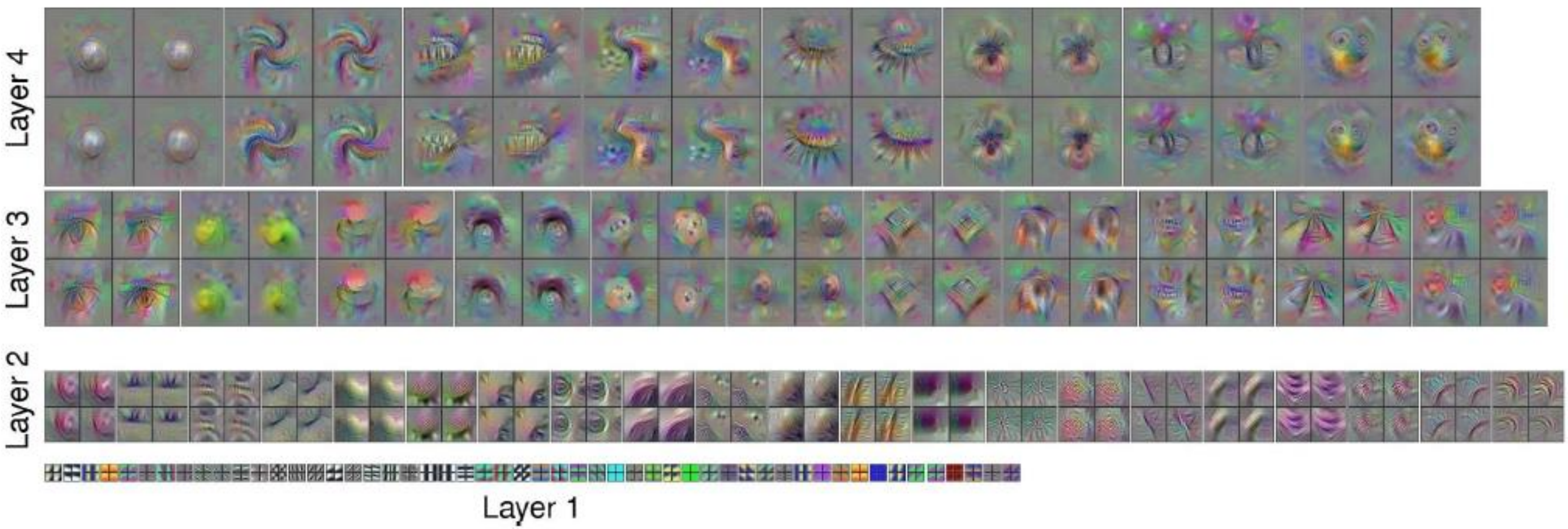
(A Small Tweak For the Gradient Descent Algorithm)

- Doing gradient descent can lead to things that don’t look like images at all, and yet maximize the output
- To keep images from looking like white noise, do the following:
  - Update the image x using a gradient descent step
  - Blur the image x



Yosinski et al, Understanding Neural Networks Through Deep Visualization (ICML 2015)


Layer 6
Layer 5
Layer 4



Layer 4
Layer 3
Layer 2
Layer 1