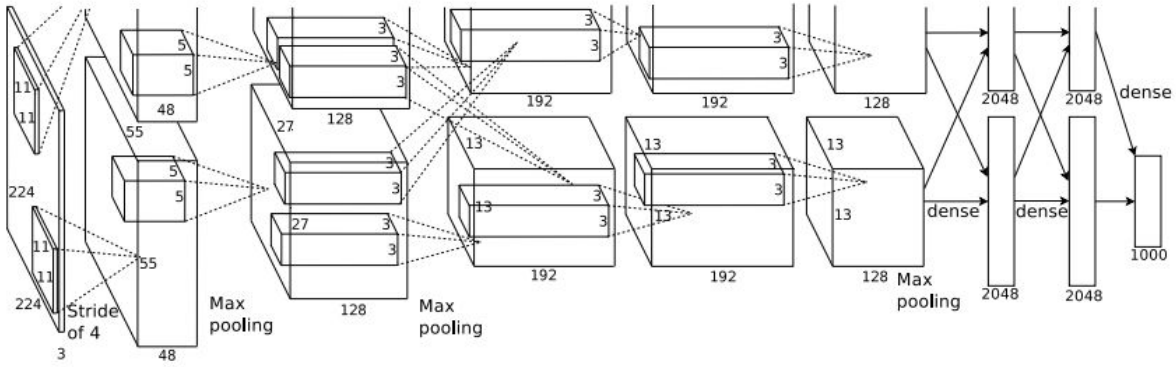
Visualization of CNN

Yuan YAO

HKUST

Visualizing Convolutional Networks

Understanding intermediate neurons?



Class Scores:
1000 numbers

Input Image:
3 x 224 x 224

What are the intermediate features looking for?

Visualizing CNN Features: Gradient Ascent

- **Gradient ascent**: Generate a synthetic image that maximally activates a neuron

$$I^* = \arg\max_I \boxed{f(I)} + \boxed{R(I)}$$

Neuron value

Natural image regularizer

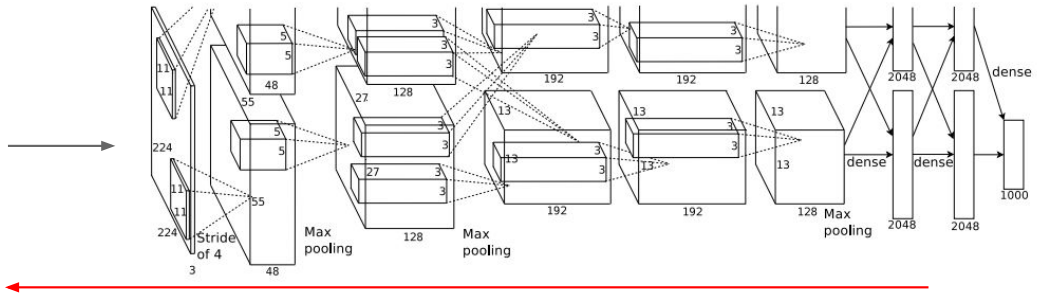
Visualizing CNN Features: Gradient Ascent

$$\arg\max_I S_c(I) - \lambda \|I\|_2^2$$

score for class c (before Softmax)

1. Initialize image to zeros

zero image



Repeat:

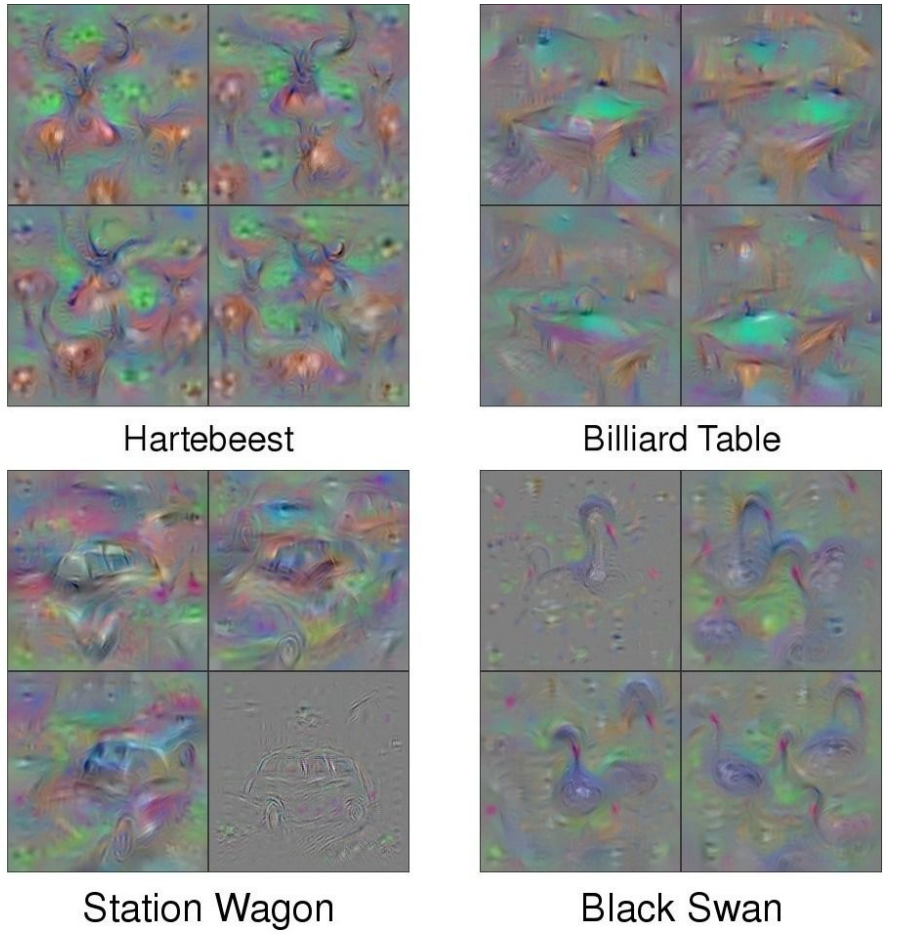
2. Forward image to compute current scores
3. Backprop to get gradient of neuron value with respect to image pixels
4. Make a small update to the image

Visualizing CNN Features: Gradient Ascent

$$\arg\max_I S_c(I) - \lambda \|I\|_2^2$$

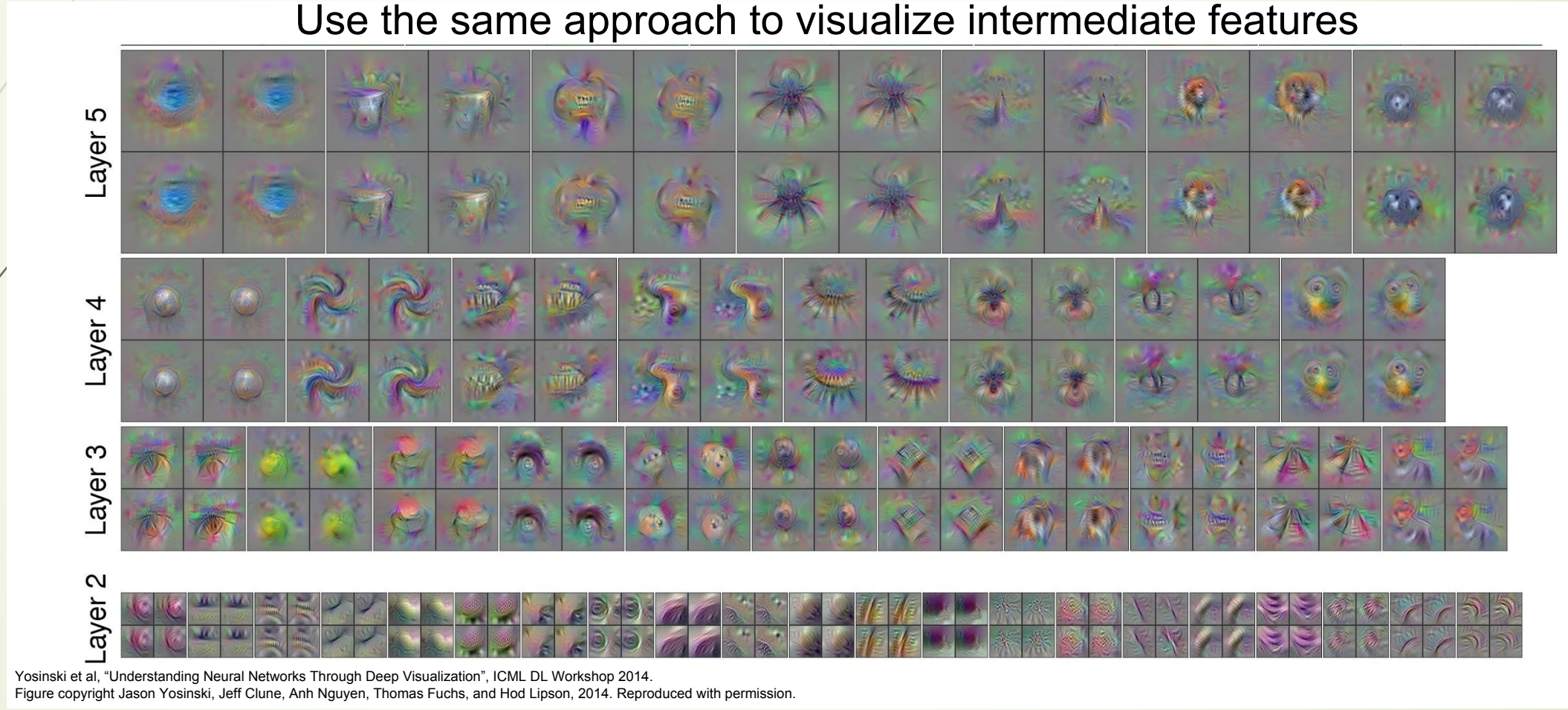
Better regularizer: Penalize L2 norm of image; also during optimization periodically

(1) Gaussian blur image
(2) Clip pixels with small values to 0
(3) Clip pixels with small gradients to 0



Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

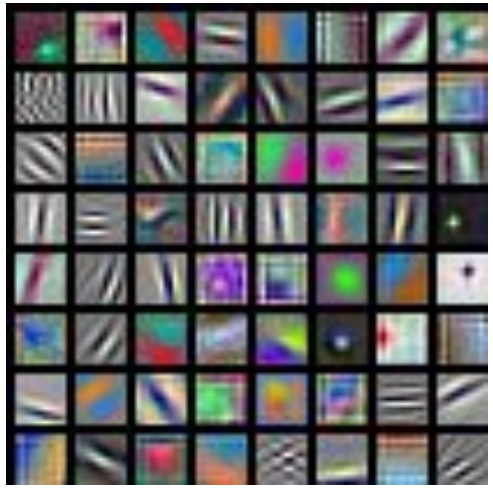
Visualizing CNN Features: Gradient Ascent



Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

It's easy to visualize early layers

First Layer: Visualize Filters

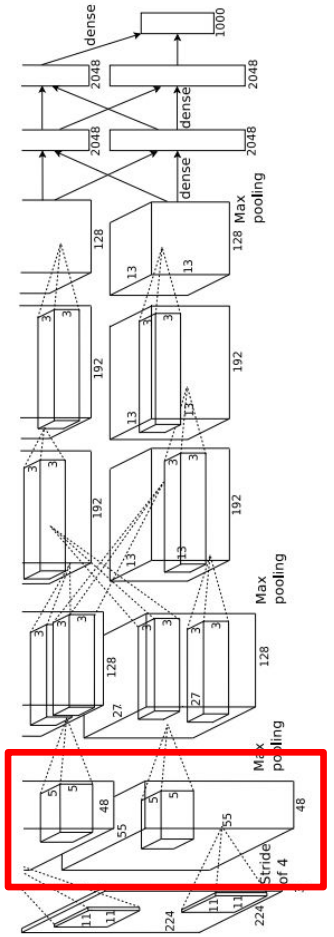


AlexNet:
64 x 3 x 11 x 11

ResNet-18:
64 x 3 x 7 x 7

ResNet-101:
64 x 3 x 7 x 7

DenseNet-121:
64 x 3 x 7 x 7



Krizhevsky, "One weird trick for parallelizing convolutional neural networks", arXiv 2014
He et al, "Deep Residual Learning for Image Recognition", CVPR 2016
Huang et al, "Densely Connected Convolutional Networks", CVPR 2017

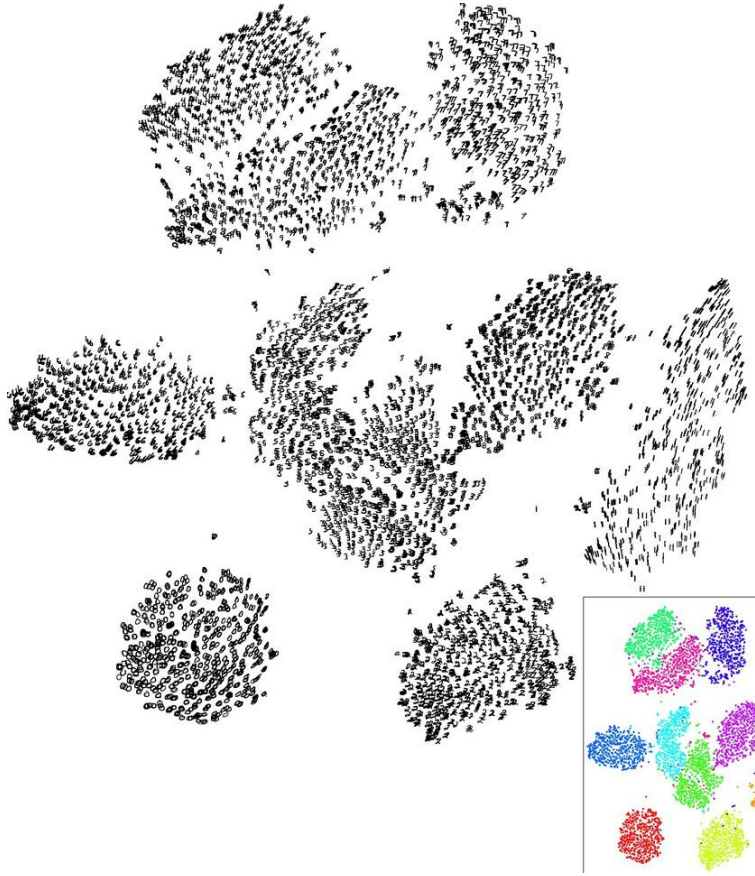
Last layers are hard to visualize

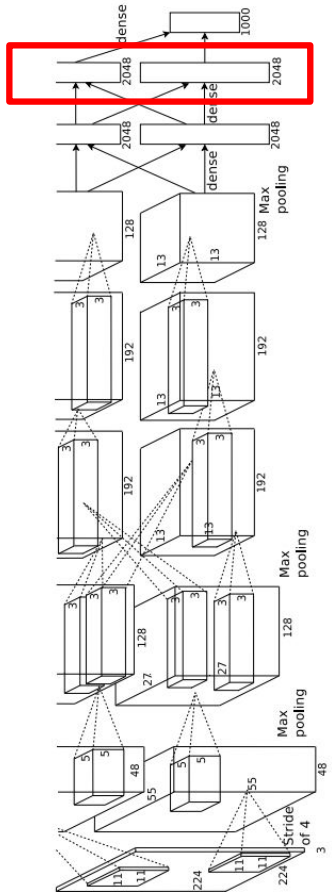
Last Layer: Dimensionality Reduction

Visualize the “space” of FC7 feature vectors by reducing dimensionality of vectors from 4096 to 2 dimensions

Simple algorithm: Principle Component Analysis (PCA)

More complex: **t-SNE**

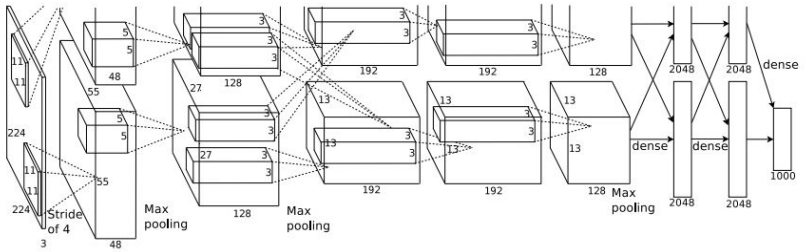




Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008

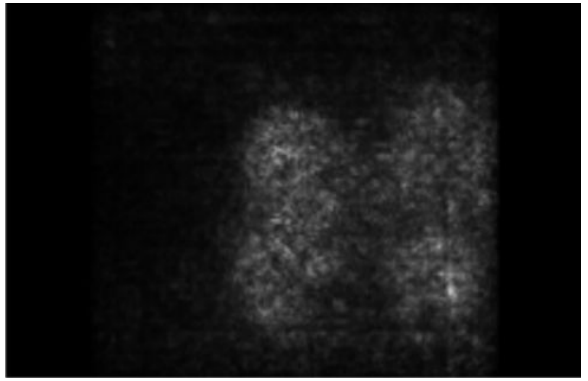
Saliency Maps

How to tell which pixels matter for classification?



Dog

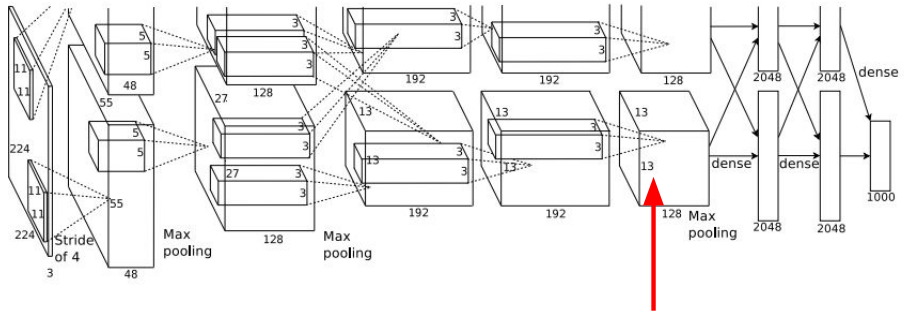
Compute gradient of (unnormalized) class score with respect to image pixels, take absolute value and max over RGB channels



Simonyan, Vedaldi, and Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014.

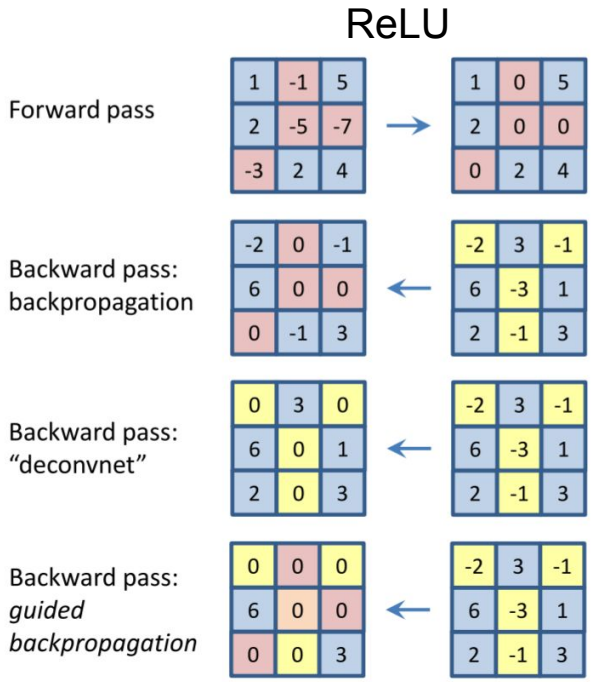
Guided BP

Intermediate features via (guided) backprop



Pick a single intermediate neuron, e.g. one value in 128 x 13 x 13 conv5 feature map

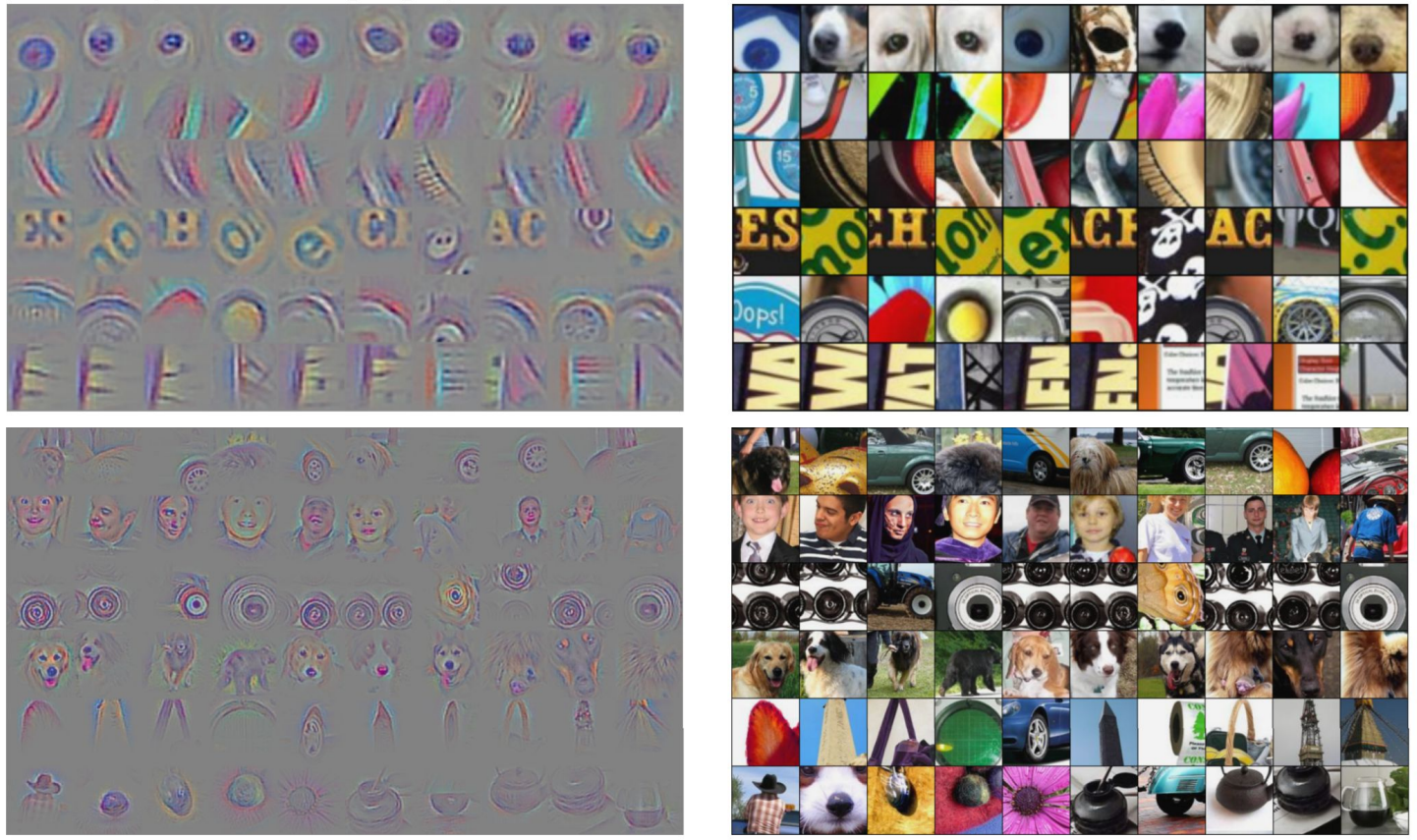
Compute gradient of neuron value with respect to image pixels



Images come out nicer if you only backprop positive gradients through each ReLU (guided backprop)

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015

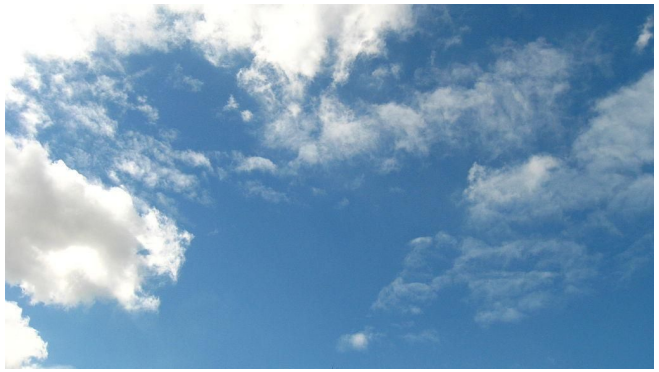
Intermediate features via Guided BP

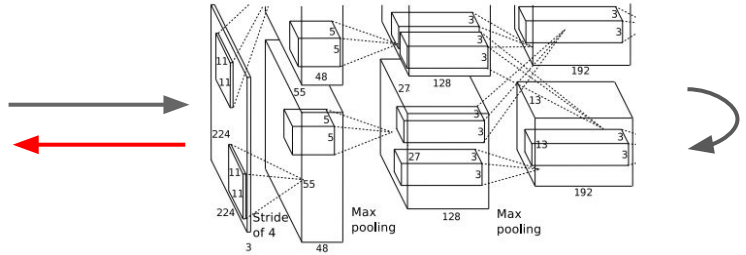


Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014
Springenberg et al, “Striving for Simplicity: The All Convolutional Net”, ICLR Workshop 2015

DeepDream: amplifying features

Rather than synthesizing an image to maximize a specific neuron, instead try to **amplify** the neuron activations at some layer in the network





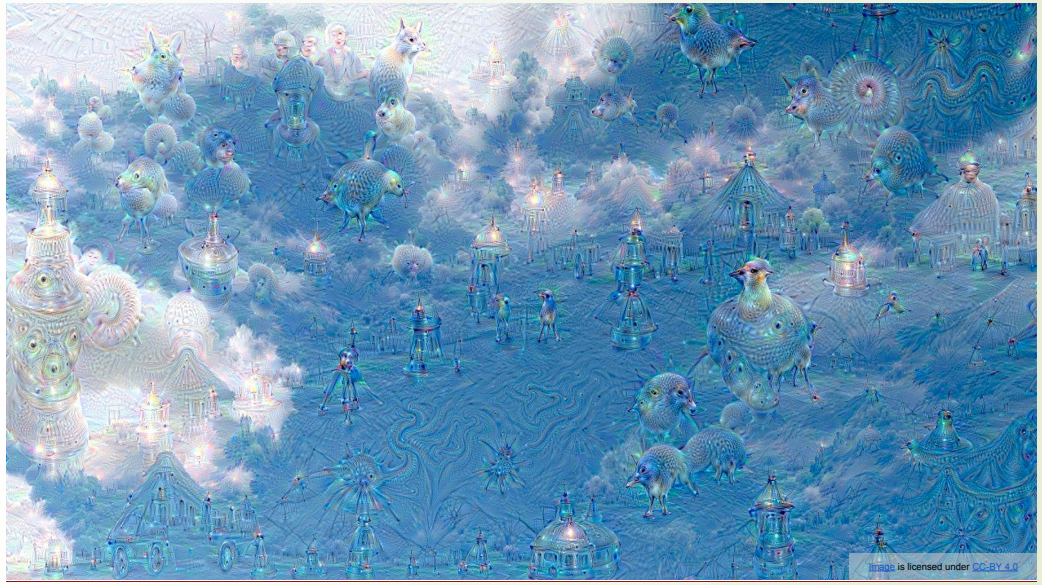
Choose an image and a layer in a CNN; repeat:

1. Forward: compute activations at chosen layer
2. Set gradient of chosen layer *equal to its activation*
3. Backward: Compute gradient on image
4. Update image

Equivalent to:

$$I^* = \arg\max_I \sum_i f_i(I)^2$$

Example: DeepDream of Sky



"Admiral Dog!"

"The Pig-Snail"

"The Camel-Bird"

"The Dog-Fish"

More Examples

Python Notebooks

- An interesting Pytorch Implementation of these visualizatoin methods
  - https://github.com/utkuozbulak/pytorch-cnn-visualizations
- Some examples demo

Thank you!