

1



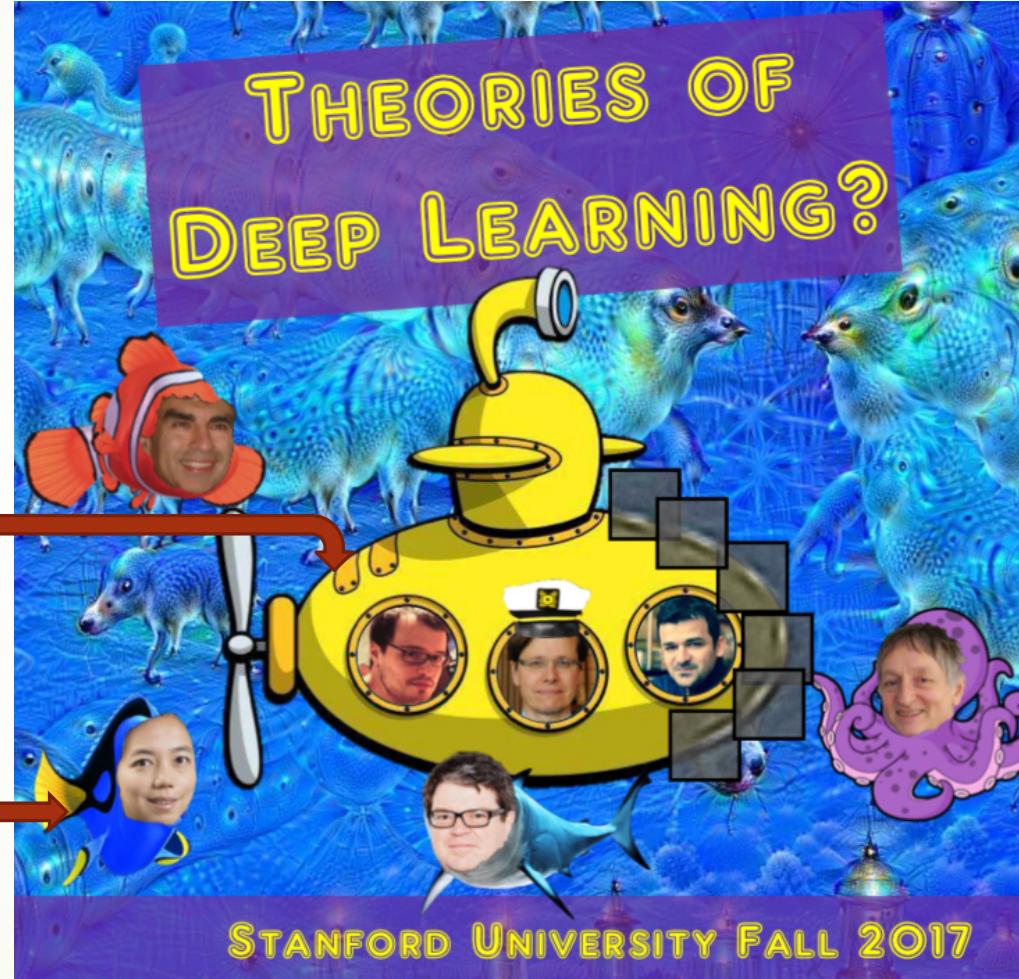
Topics on CNN: Visualization, Transfer Learning

Yuan YAO
HKUST

Acknowledgement

<https://stats385.github.io/>

<http://cs231n.github.io/>



A following-up course at HKUST: <https://deeplearning-math.github.io/>

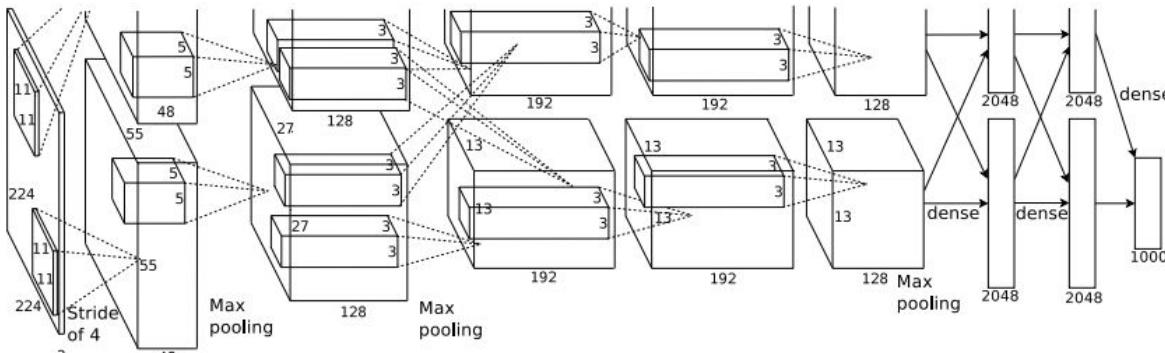


Visualizing Convolutional Networks

Understanding intermediate neurons?



Input Image:
3 x 224 x 224



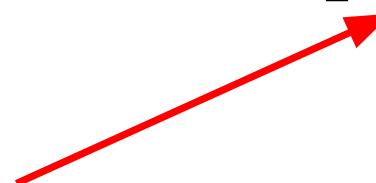
What are the intermediate features looking for?

Class Scores:
1000 numbers

Visualizing CNN Features: Gradient Ascent

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

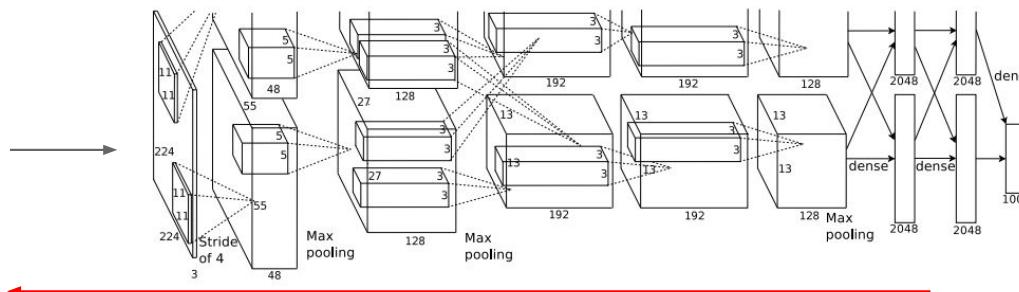
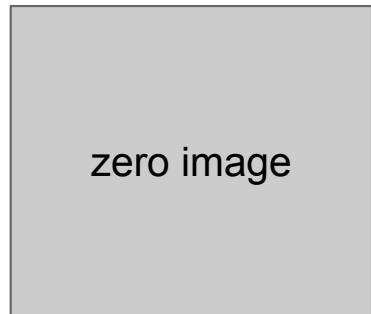
$$I^* = \arg \max_I f(I) + R(I)$$



Neuron value Natural image regularizer

Visualizing CNN Features: Gradient Ascent

1. Initialize image to zeros



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

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

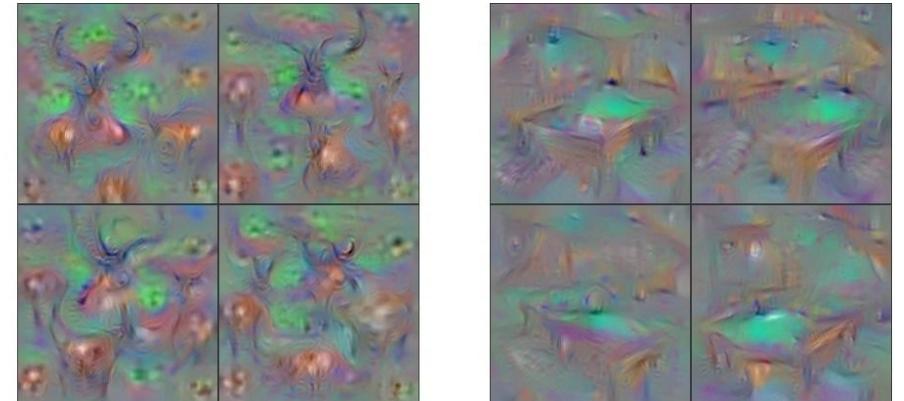
score for class c (before Softmax)

Visualizing CNN Features: Gradient Ascent

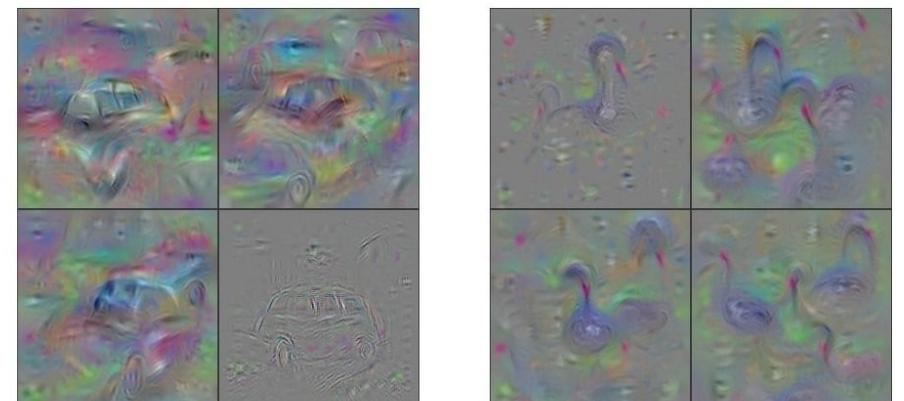
$$\arg \max_I S_c(I) - \lambda \|I\|_2^2$$

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

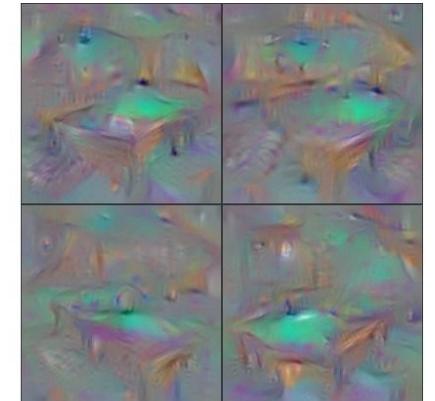
- (1) Gaussian blur image
- (2) Clip pixels with small values to 0
- (3) Clip pixels with small gradients to 0



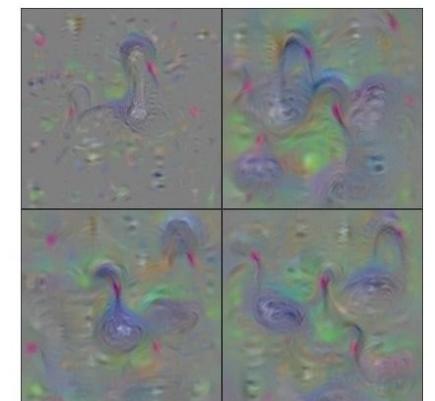
Hartebeest



Station Wagon



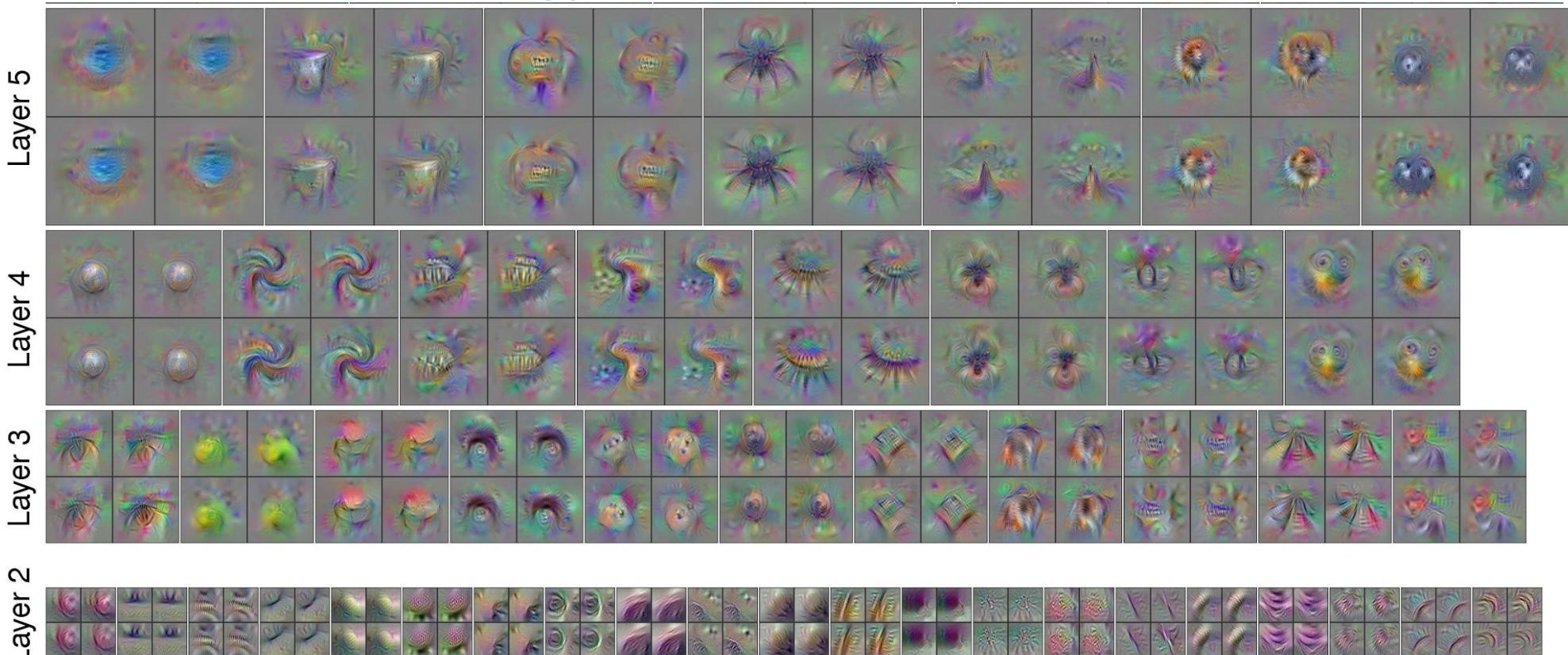
Billiard Table



Black Swan

Visualizing CNN Features: Gradient Ascent

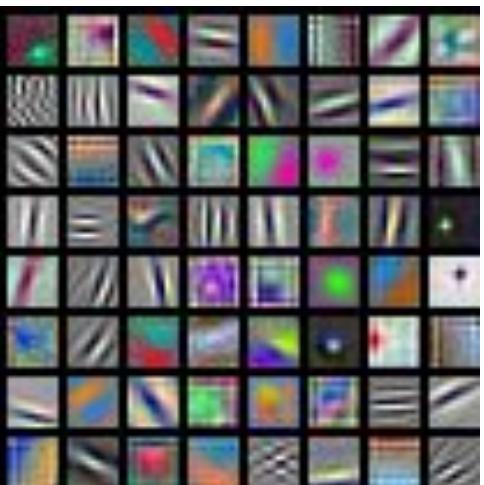
Use the same approach to visualize intermediate features



Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.
Figure copyright Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, 2014. Reproduced with permission.

It's easy to visualize early layers

First Layer: Visualize Filters



AlexNet: 64 x 3 x 11 x 11



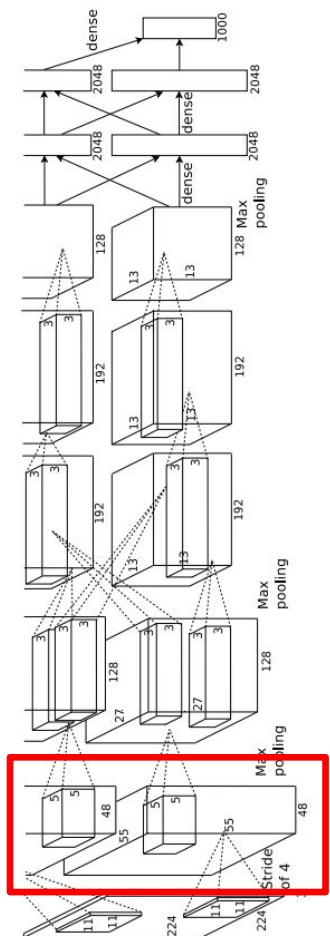
ResNet-18: 64 x 3 x 7 x 7



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

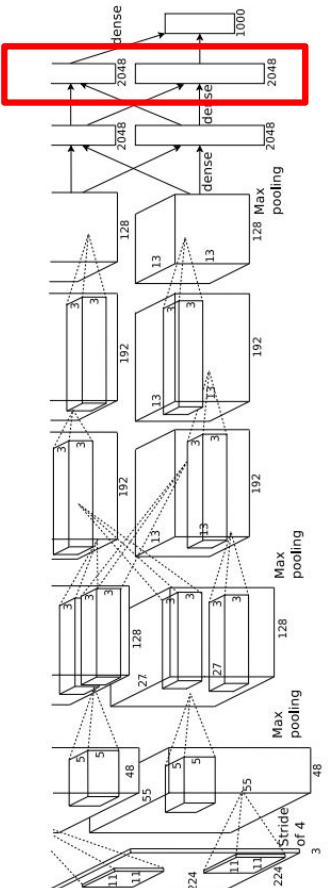
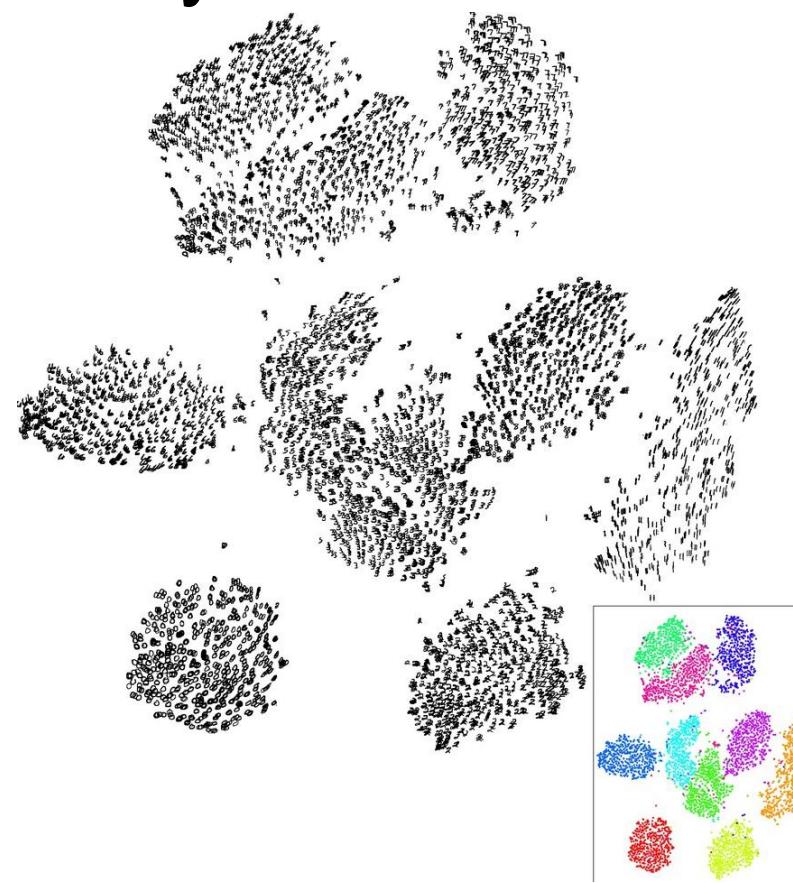
Last layers are hard to visualize

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)

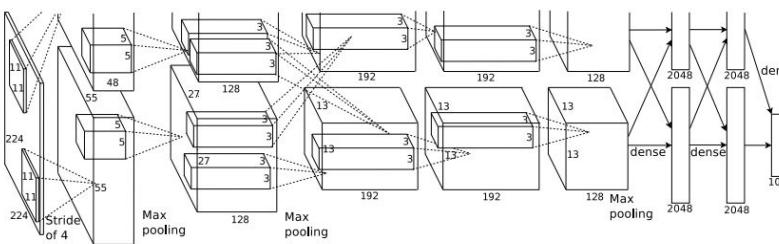
More complex: t-SNE



Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008
Figure copyright Laurens van der Maaten and Geoff Hinton, 2008. Reproduced with permission.

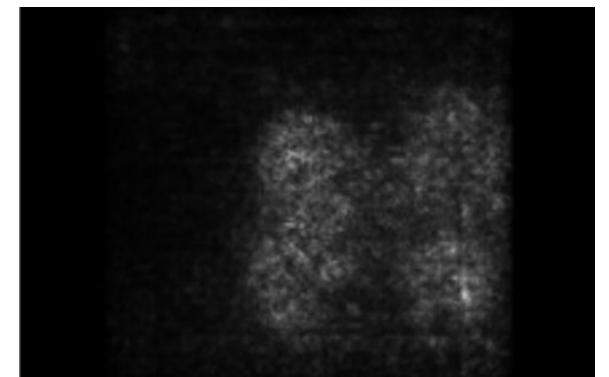
Saliency Maps

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

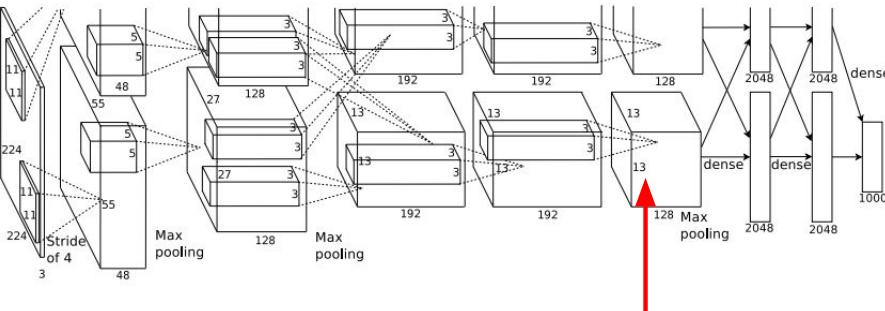


Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

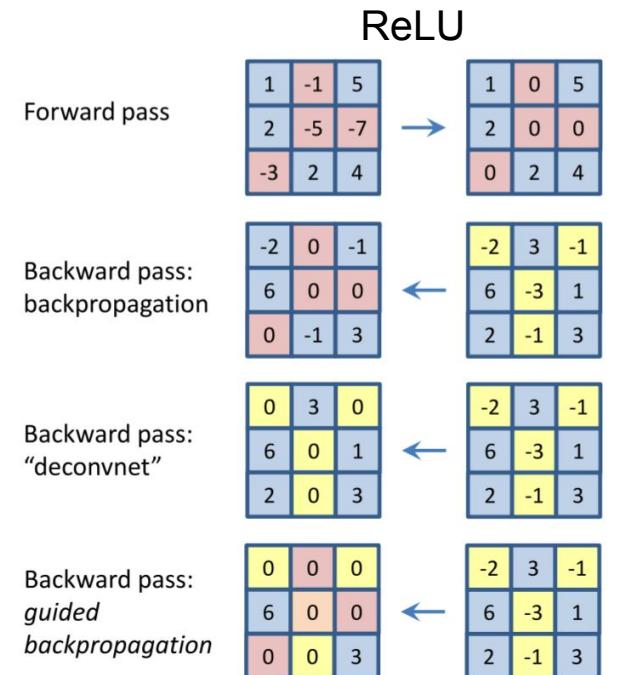
Guided BP

Intermediate features via (guided) backprop



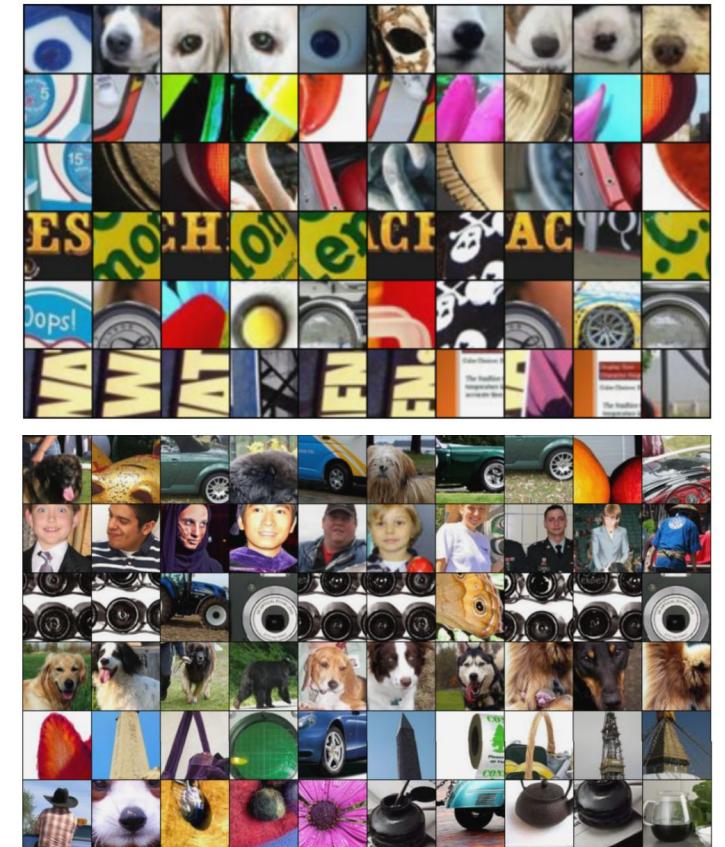
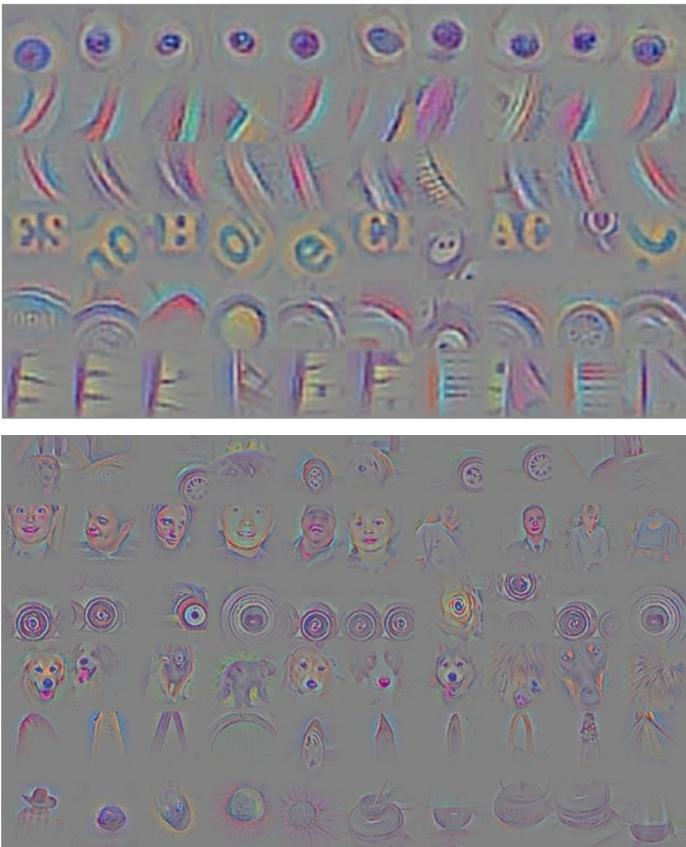
Pick a single intermediate neuron, e.g. one value in $128 \times 13 \times 13$ conv5 feature map

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)

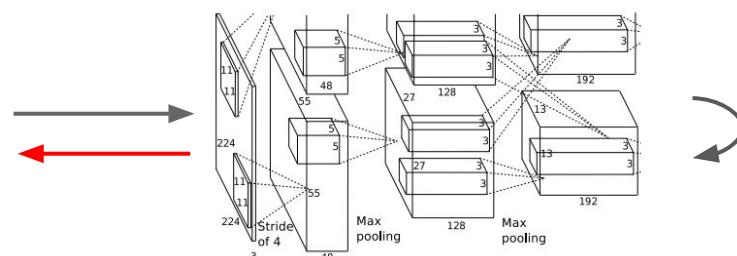
Intermediate features via Guided BP



Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015
Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

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

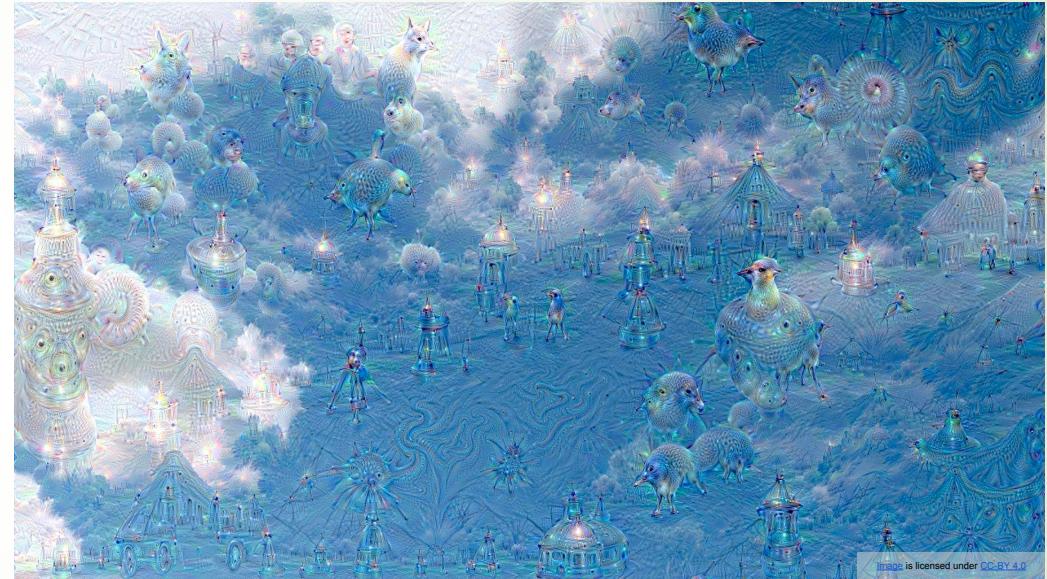


Choose an image and a layer in a CNN; repeat:

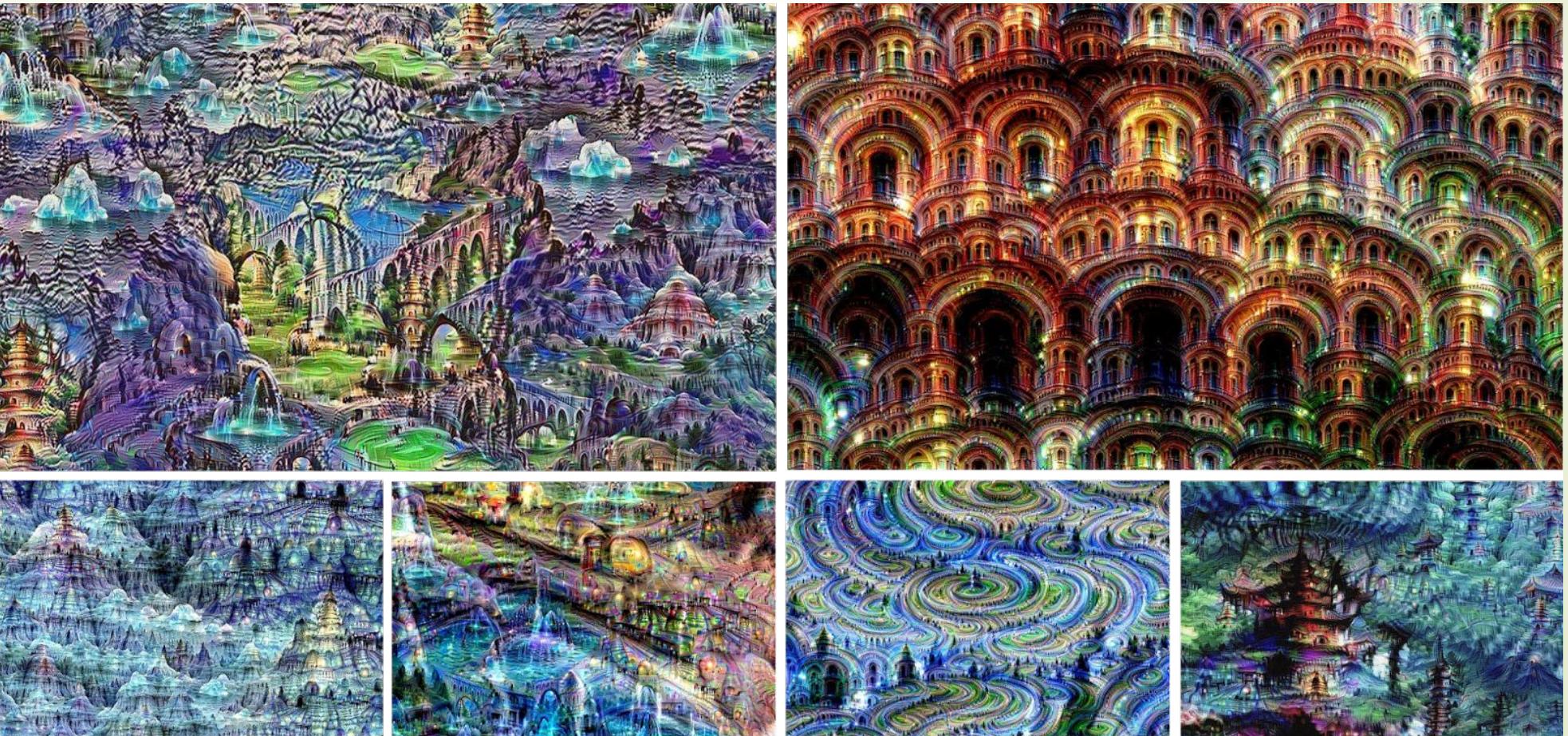
1. Forward: compute activations at chosen layer
2. Set gradient of chosen layer *equal to its activation*
3. Backward: Compute gradient on image
4. Update image

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

Example: DeepDream of Sky



More Examples



[Image](#) is licensed under CC-BY 4.0

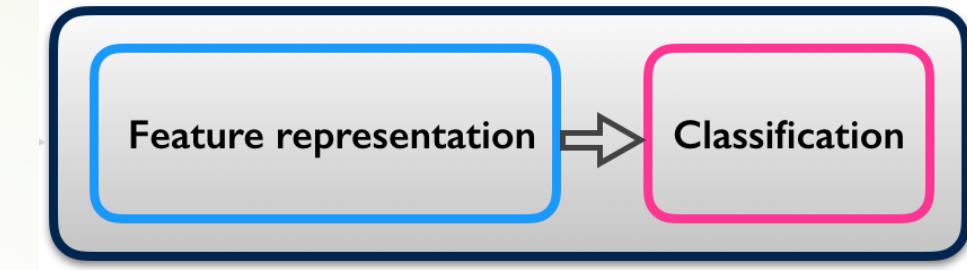
Python Notebooks

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



Transfer Learning: Fine Tuning

Transfer Learning?

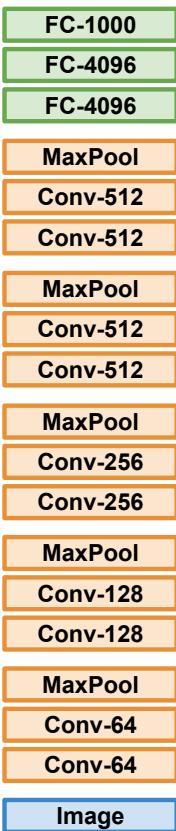


- Filters learned in first layers of a network are transferable from one task to another
- When solving another problem, no need to retrain the lower layers, just fine tune upper ones
- Is this simply due to the large amount of images in ImageNet?
- Does solving many classification problems simultaneously result in features that are more easily transferable?
- Does this imply filters can be learned in unsupervised manner?
- Can we characterize filters mathematically?

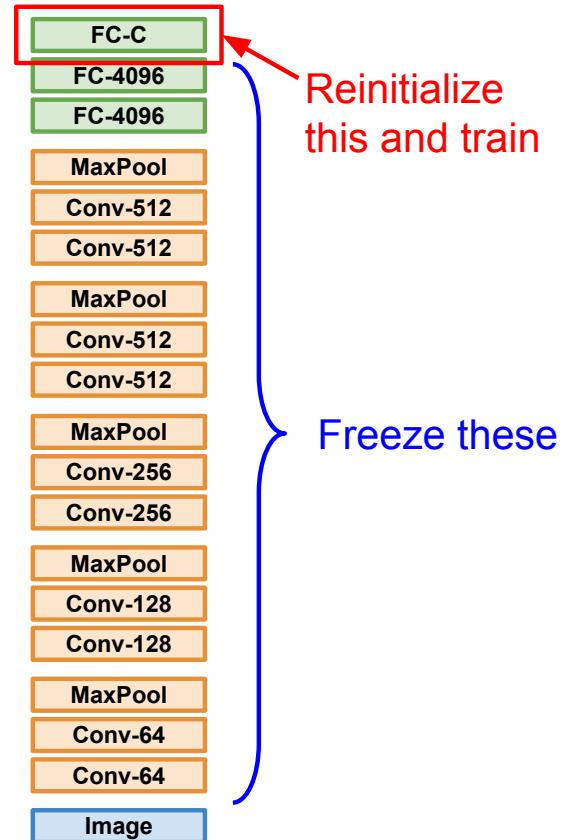
Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

Transfer Learning with CNNs

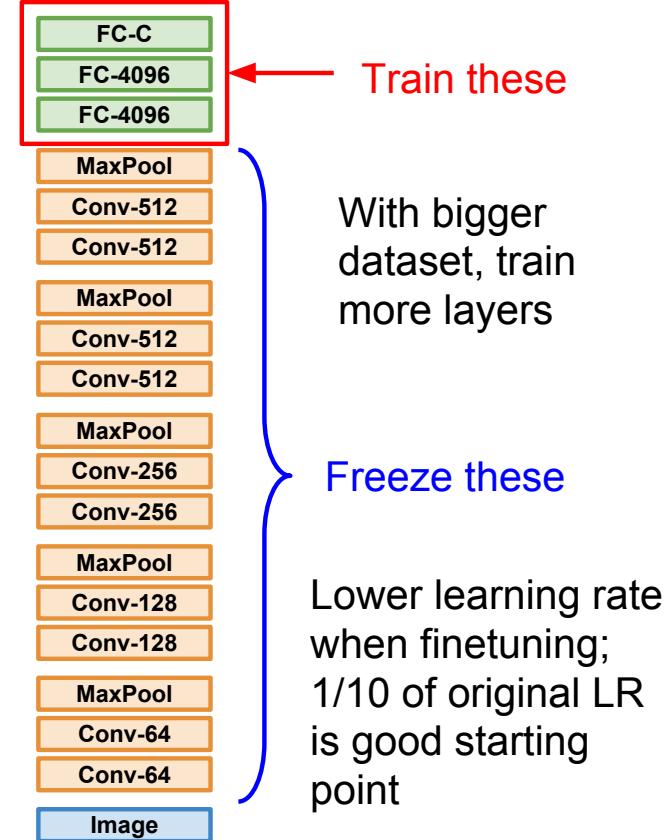
1. Train on Imagenet



2. Small Dataset (C classes)



3. Bigger dataset





More specific

More generic

	very similar dataset	very different dataset
very little data	Use Linear Classifier on top layer	You're in trouble... Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

Thank you!

