CS 4803 / 7643: Deep Learning

Topics:

- (Finish) Visualization
- Training NNs

Zsolt Kira

Georgia Tech

Administrative

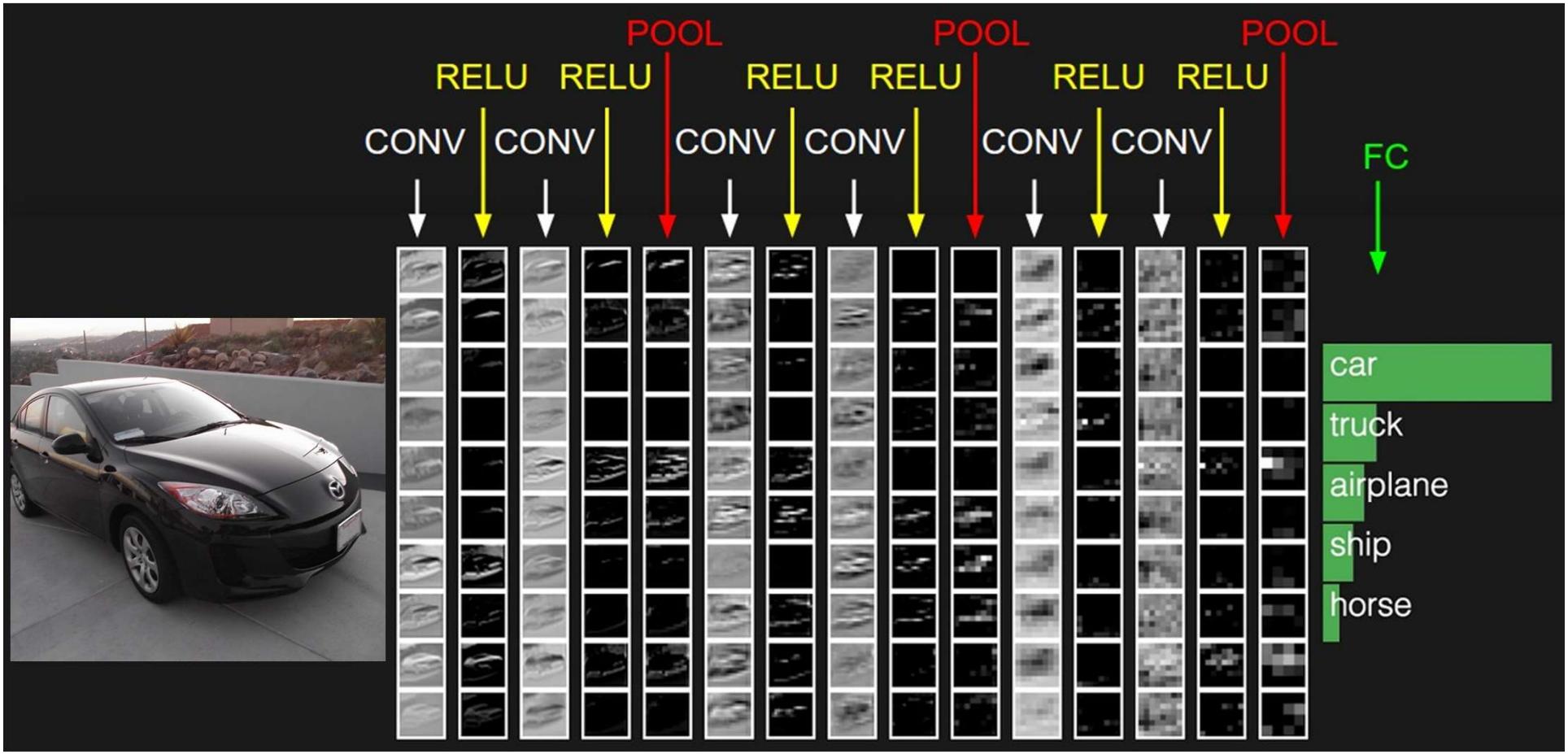
- PS2/HW2 Out, Due Feb 26th! (note 1-day extension)

- Added project ideas
  - Fill out spreadsheet! https://gtvault-my.sharepoint.com/:x:/g/personal/sdharur3_gatech_edu/EVXbNc4oxelMmj1T5WsEIRQBE4Hn532GeLQVcmOnWdG2Jg?e=dIGNfX
  - **Due March 12th** but you don't have to wait to get feedback!

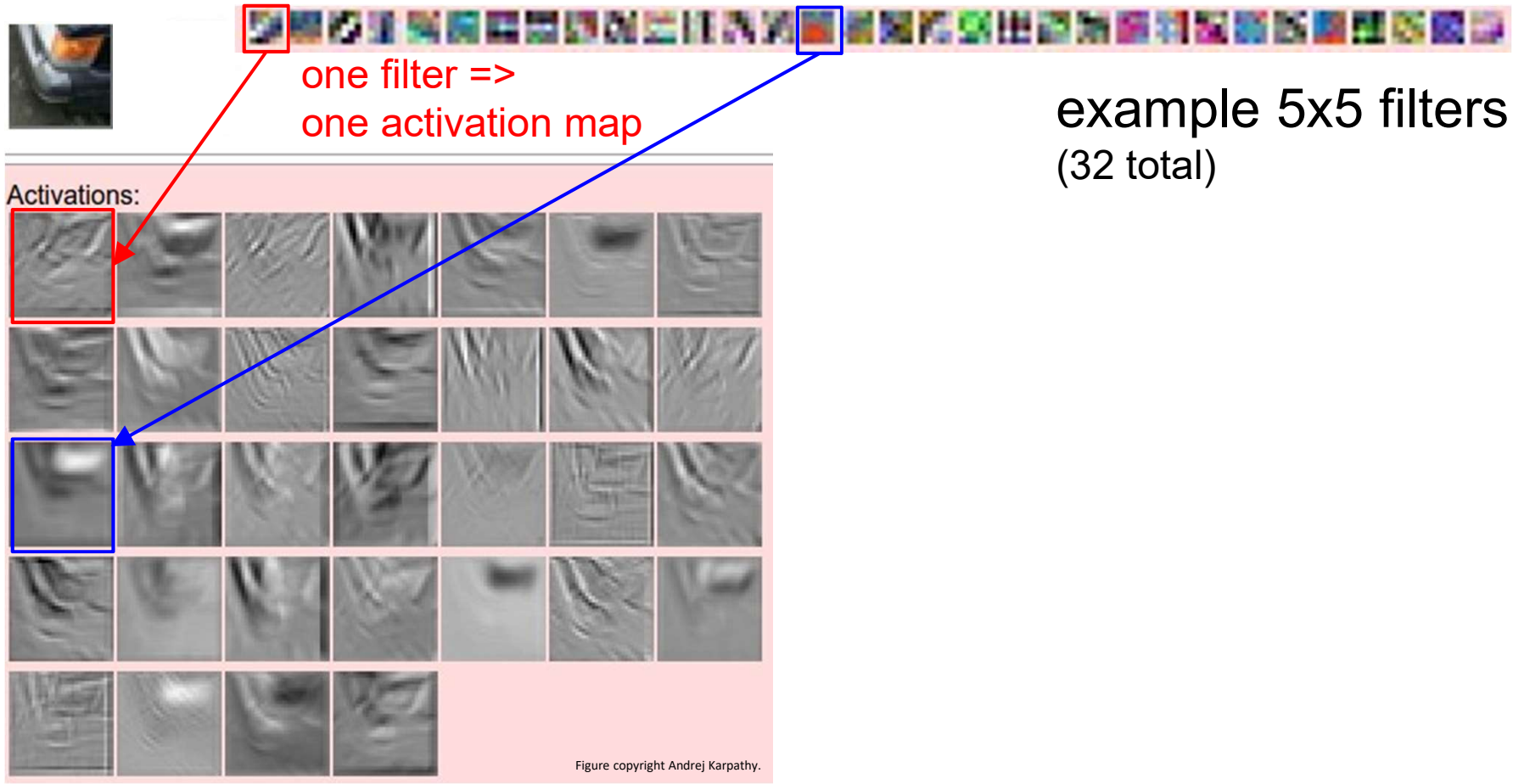
Recap

- CNN Architectures
- Visualization

preview:



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



example 5x5 filters
(32 total)

Visualizing Learned Filters

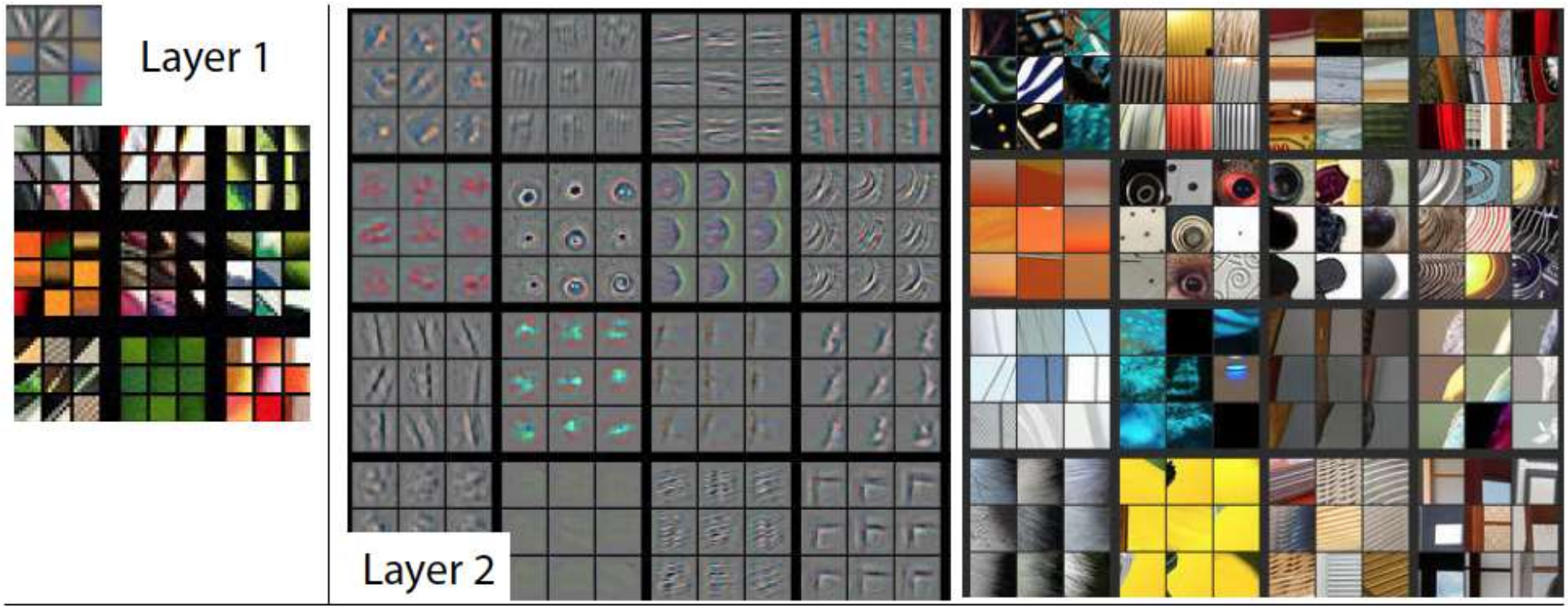


Figure Credit: [Zeiler & Fergus ECCV14]

Visualizing Learned Filters

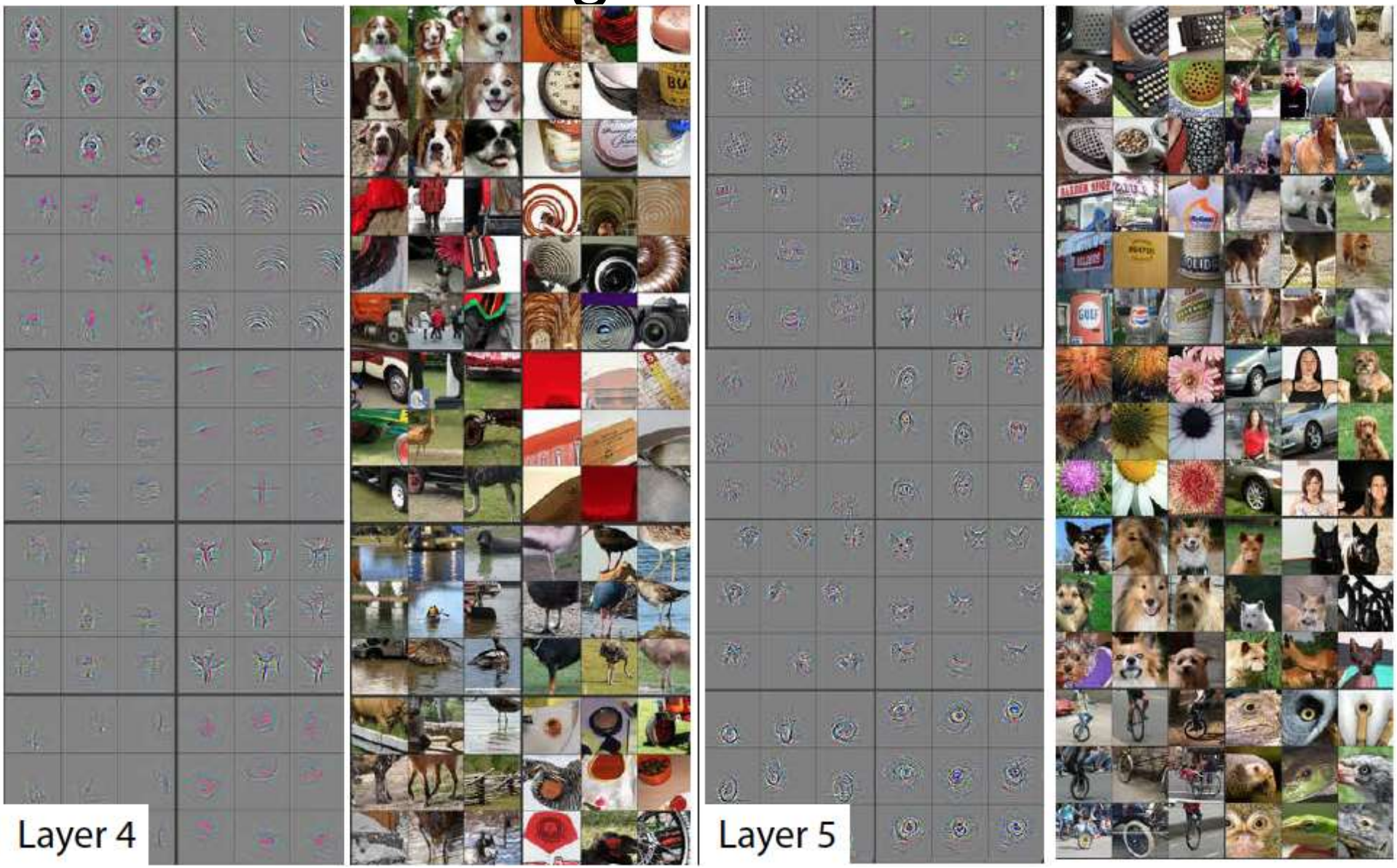
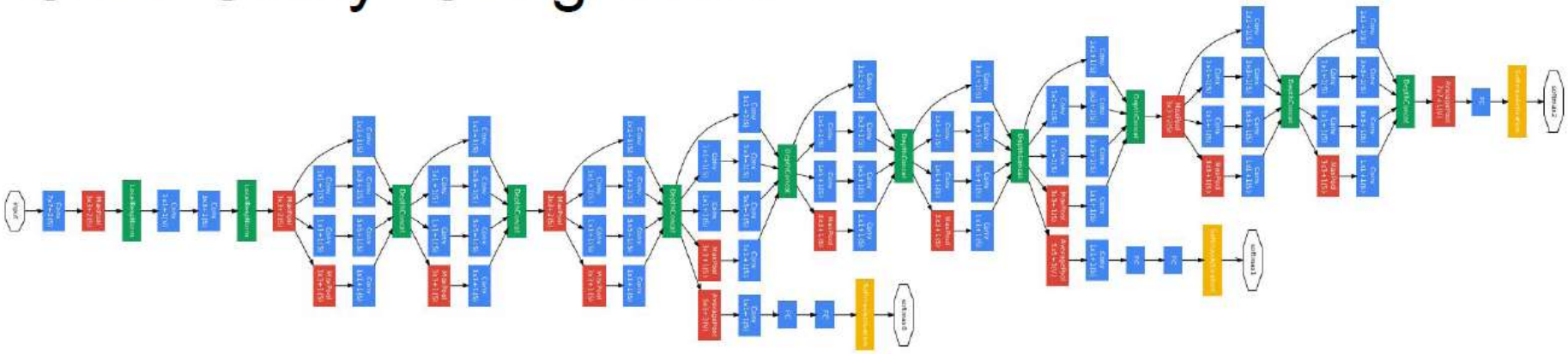
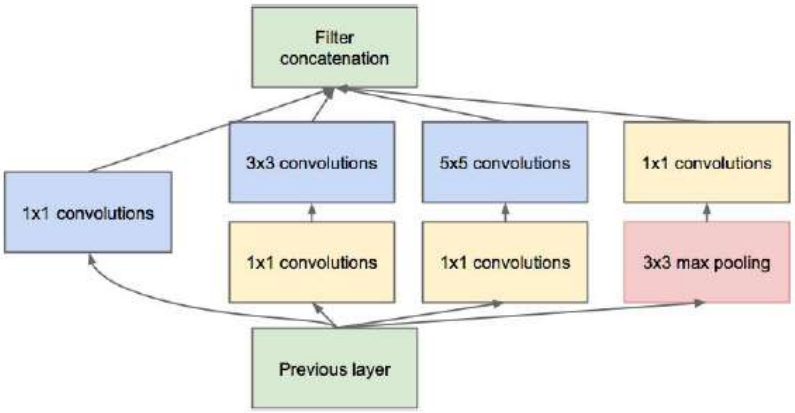


Figure Credit: [Zeiler & Fergus ECCV14]

Case Study: GoogLeNet

[Szegedy et al., 2014]



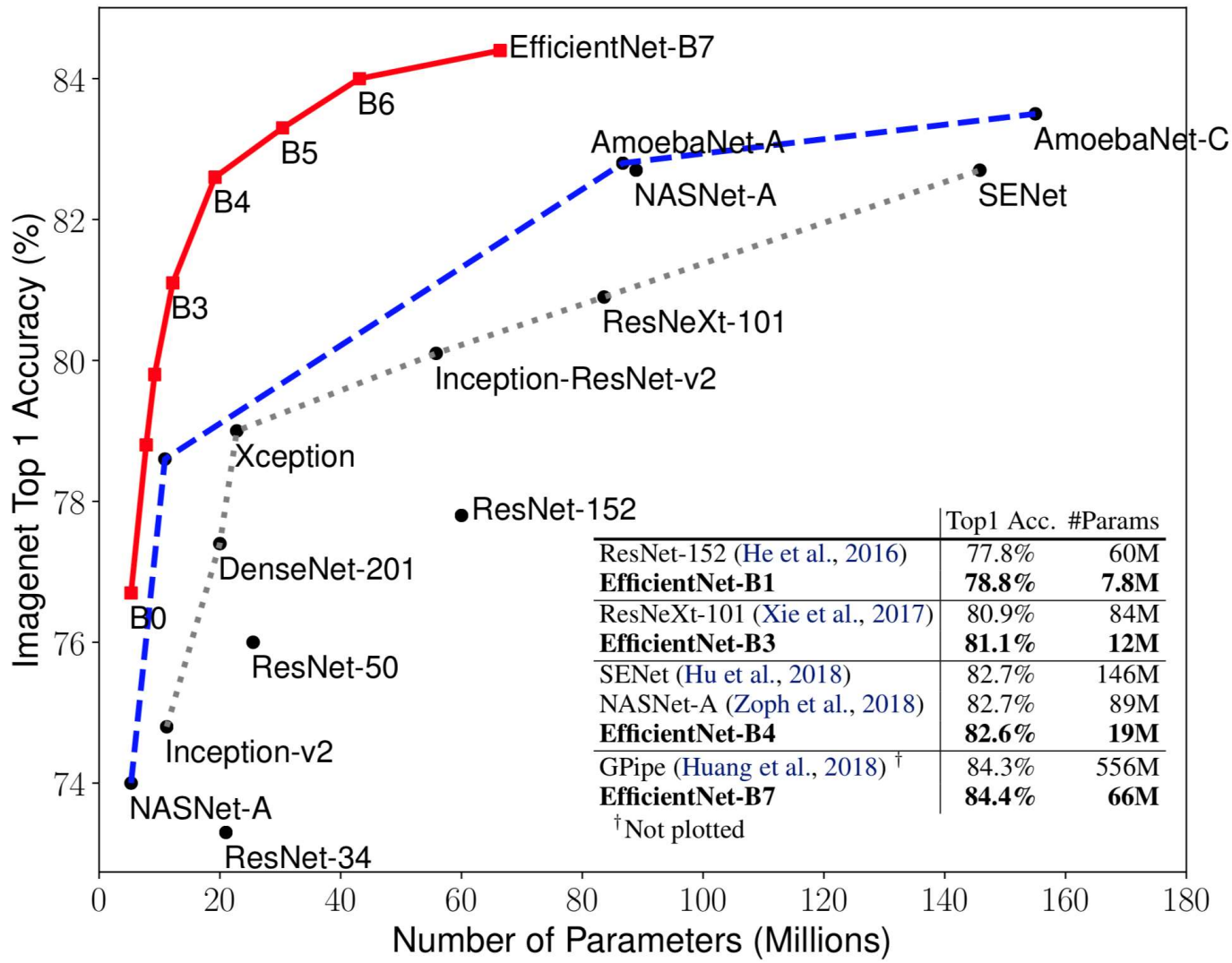


Inception module

ILSVRC 2014 winner (6.7% top 5 error)

Recently: Neural Architecture Search

- Can we better search the space of architectures?
- Several approaches
  - Evolutionary algorithms (typically of repeated blocks)
  - Training overparameterized networks & pruning



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

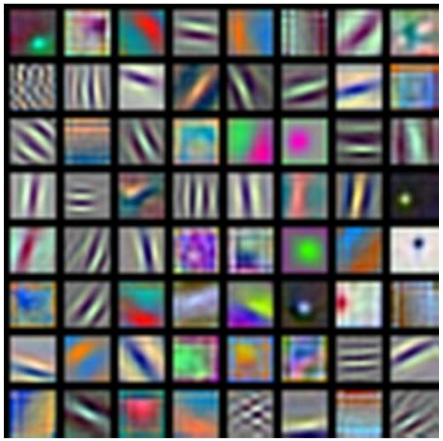
Visualization

- What do individual neurons look for in images?
  - Visualizing filters
  - Last layer embeddings
  - Visualizing activations
  - Maximally activating patches
- How pixels affect model decisions?
  - Occlusion maps
  - Salient or “important” pixels
    - Gradient-based visualizations

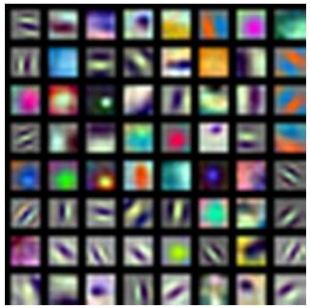
Visualization

- **What do individual neurons look for in images?**
  - Visualizing filters
  - Last layer embeddings
  - Visualizing activations
  - Maximally activating patches
- How pixels affect decisions?
  - Occlusion maps
  - Salient or “important” pixels
    - Gradient-based visualizations

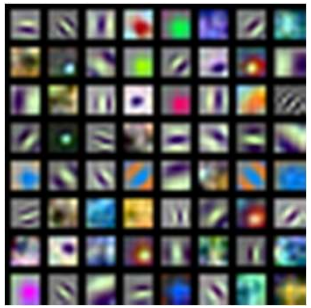
Visualizing filters in first layer



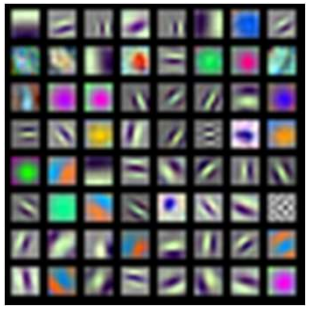
AlexNet:
64 x 3 x 11 x 11



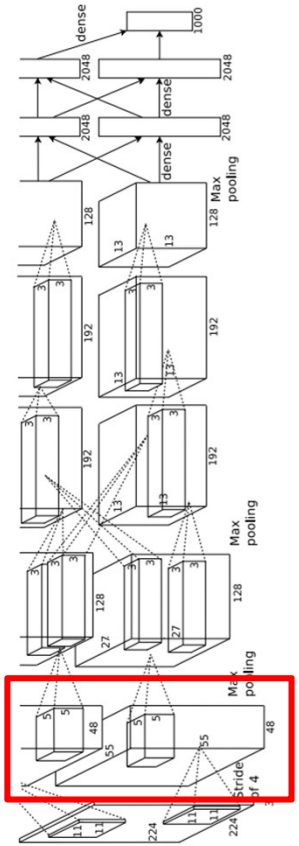
ResNet-18:
64 x 3 x 7 x 7



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

Visualizing filters in intermediate layers

Visualize the filters/kernels (raw weights)

We can visualize filters at higher layers, but not that interesting

layer 1 weights

16 x 3 x 7 x 7

layer 2 weights

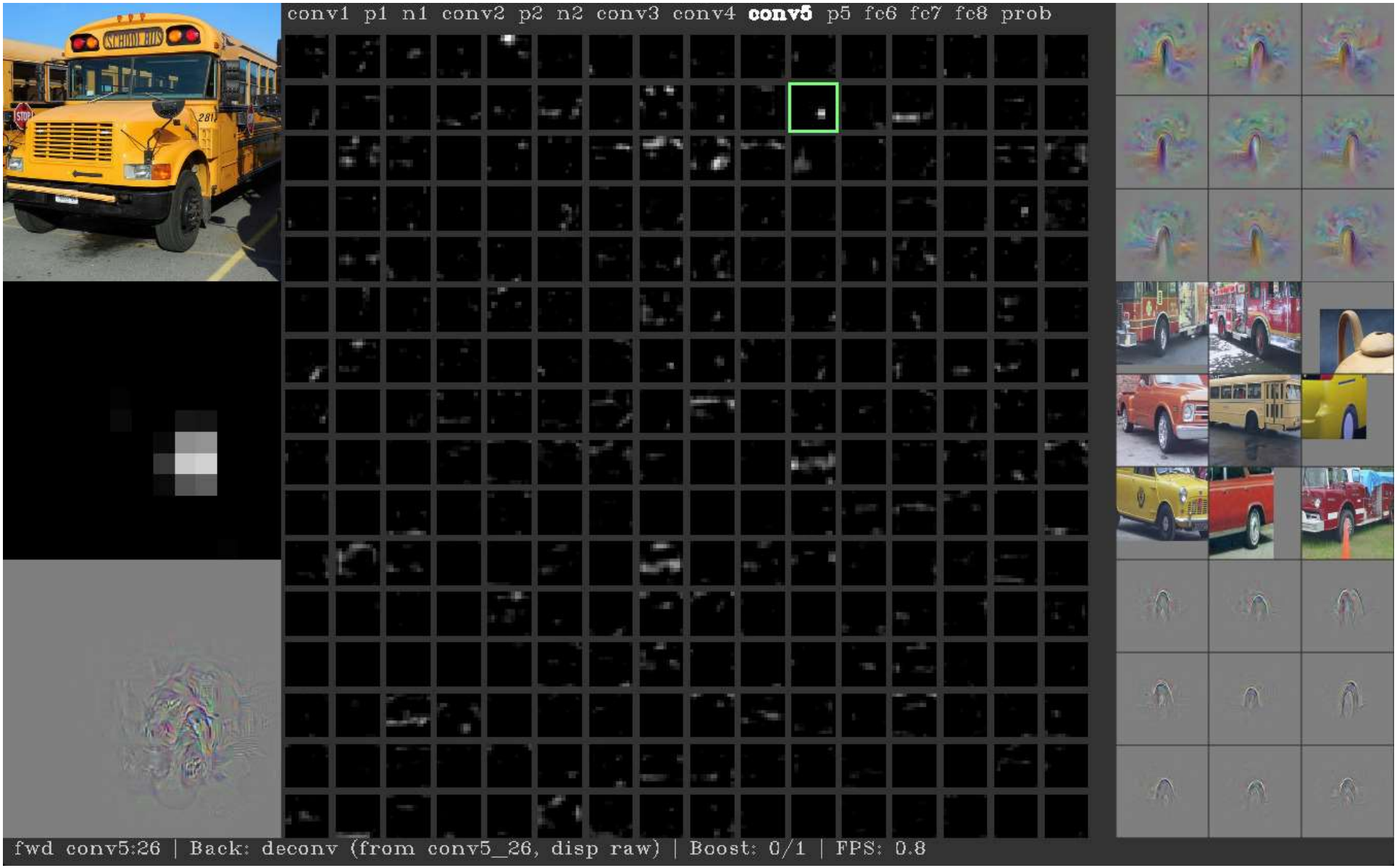
20 x 16 x 7 x 7

layer 3 weights

20 x 20 x 7 x 7

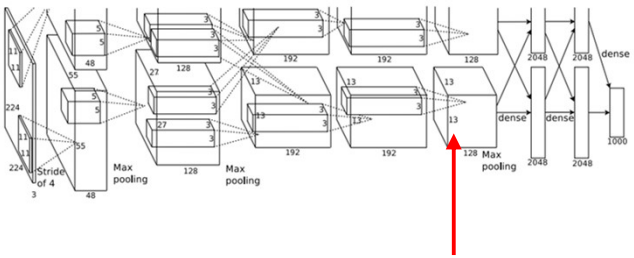
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Visualizing activations in intermediate layers



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

Maximally Activating Patches



Pick a layer and a channel; e.g. conv5 is 128 x 13 x 13, pick channel 17/128

Run many images through the network, record values of chosen channel

Visualize image patches that correspond to maximal activations

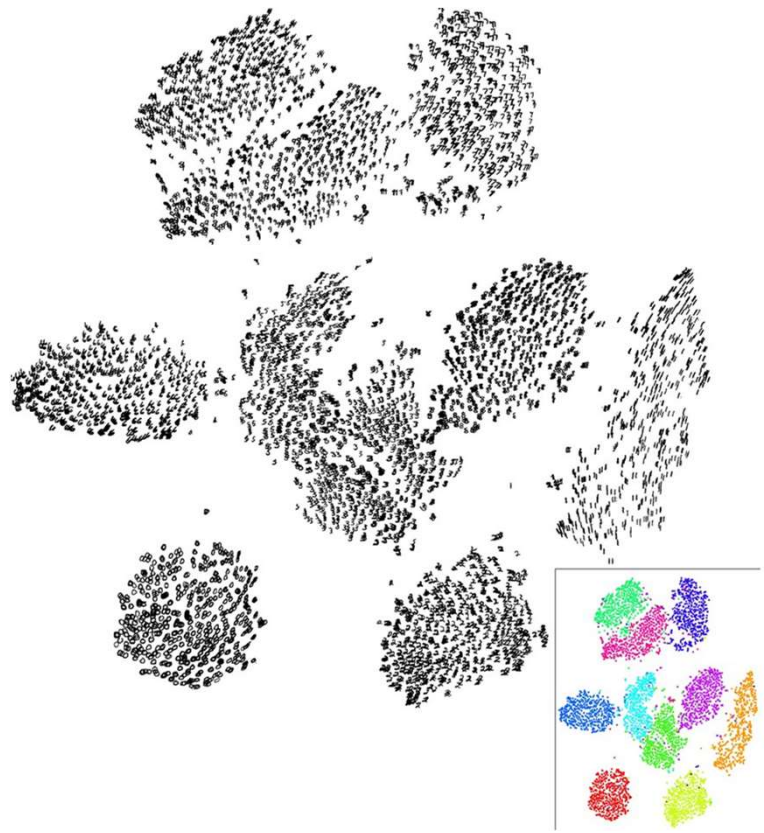
Springenberg et al, “Striving for Simplicity: The All Convolutional Net”, ICLR Workshop 2015
Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

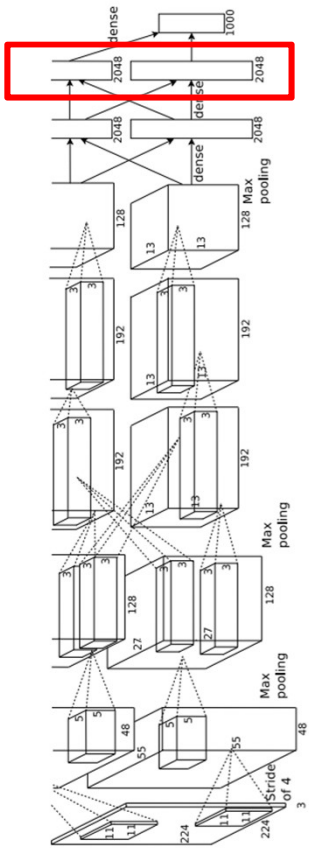
Last Layer: Dimensionality Reduction

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

Simple algorithm: Principal Component Analysis (PCA)

More complex: **t-SNE**

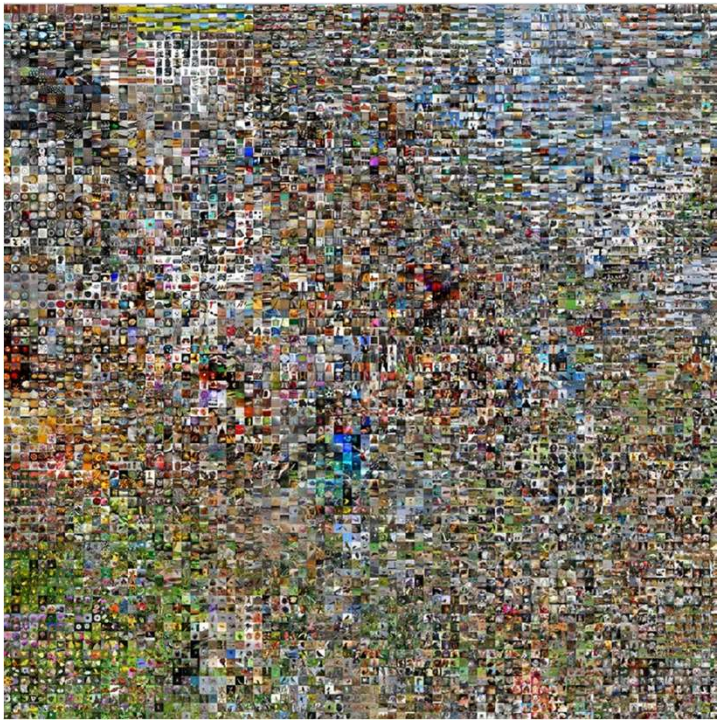




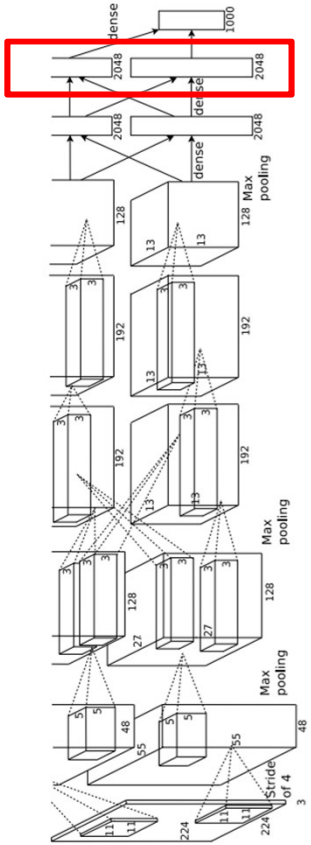
Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008

Last Layer: Dimensionality Reduction

Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008
Krizhevsky et al, “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012.
Figure reproduced with permission.



See high-resolution versions at
http://cs.stanford.edu/people/karpathy/cnnembed/



Visualization

- What do individual neurons look for in images?
  - Visualizing filters
  - Last layer embeddings
  - Visualizing activations
  - Maximally activating patches
- **How pixels affect decisions from CNNs?**
  - Occlusion maps
  - Salient or “important” pixels
    - Gradient-based visualizations

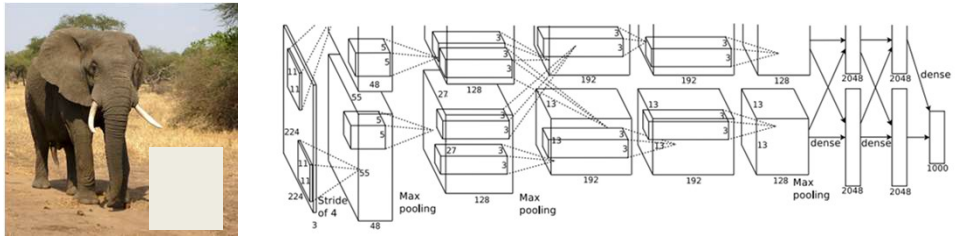
How pixels affect decisions?

Visual Explanations

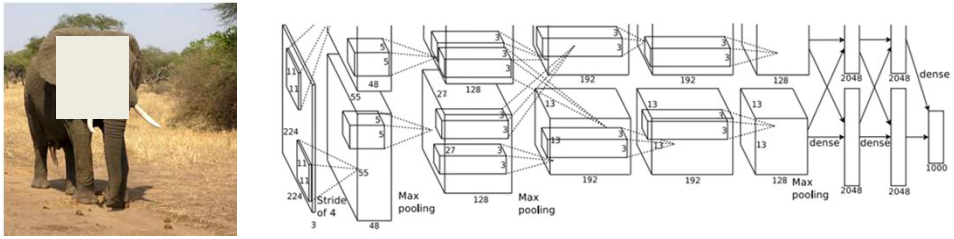
Where does an intelligent system "look" to make its predictions?

Which pixels matter: Occlusion Maps

Idea: Mask part of the image before feeding to CNN, check how much predicted probabilities change



P(elephant) = 0.95

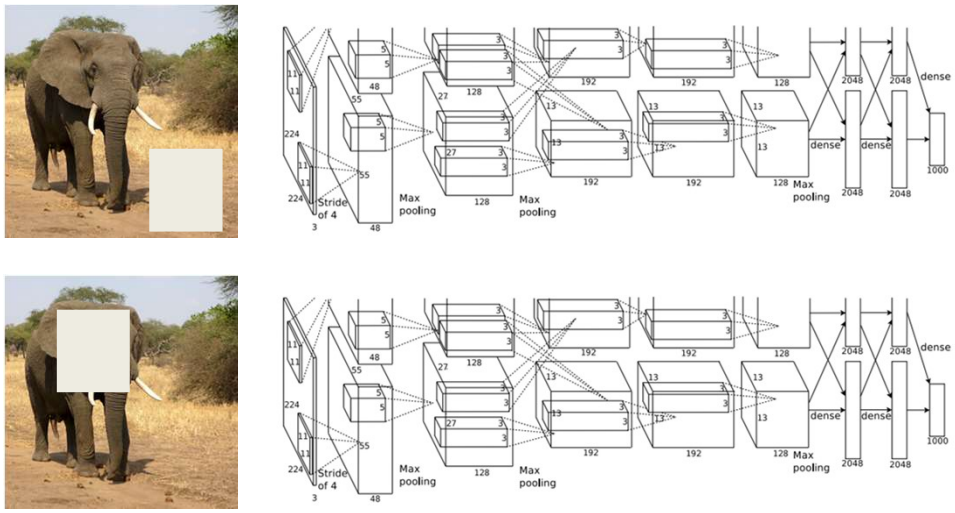


P(elephant) = 0.75

Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

Which pixels matter: Occlusion Maps

Mask part of the image before feeding to CNN,
check how much predicted probabilities change



Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

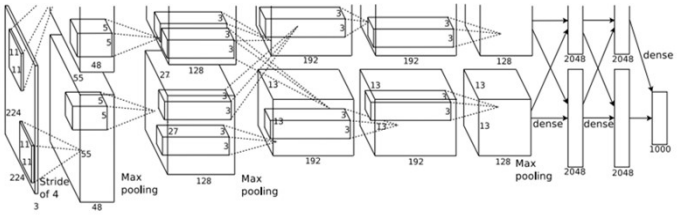
Faithful ☺

Very expensive ☹

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

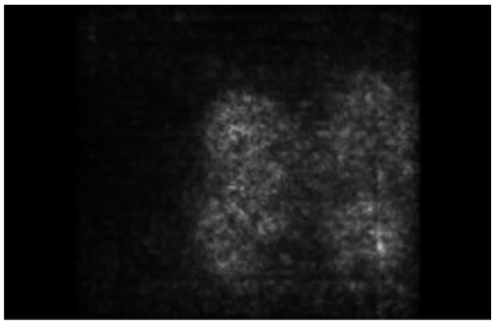
Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities



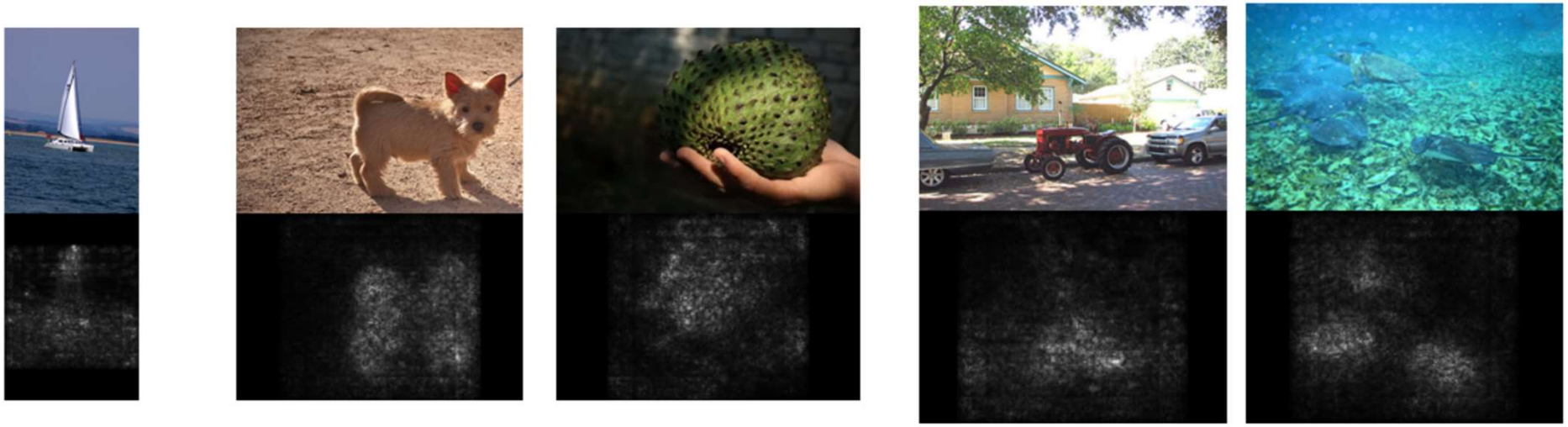
Dog

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.

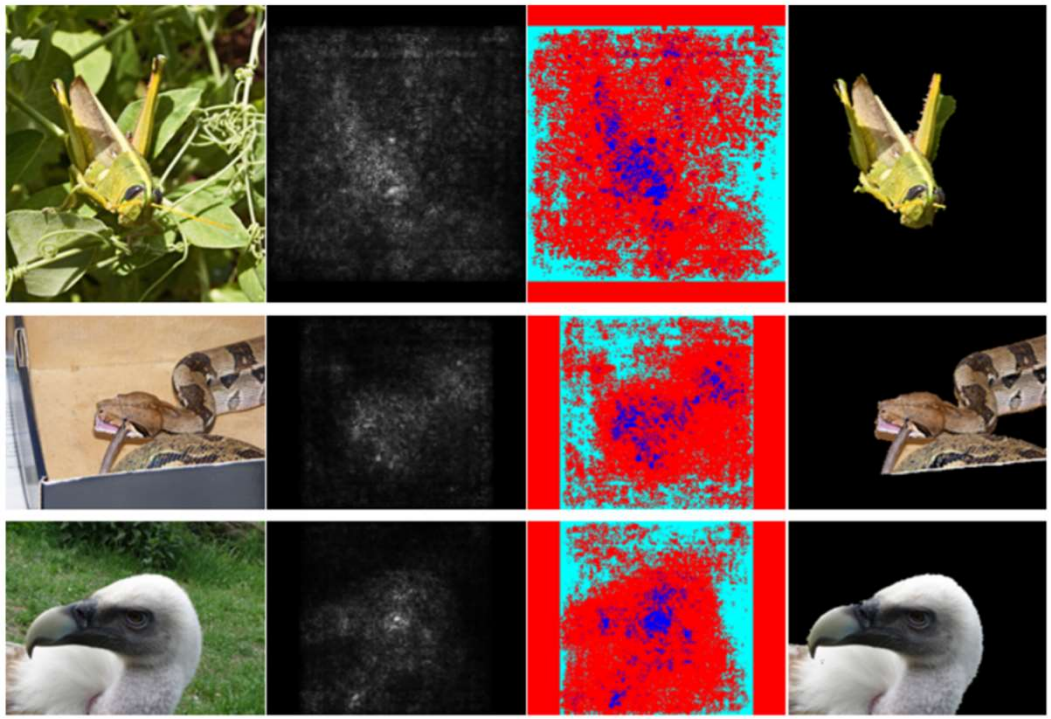
Saliency Maps



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

Saliency Maps: Segmentation without supervision

Use GrabCut on saliency map

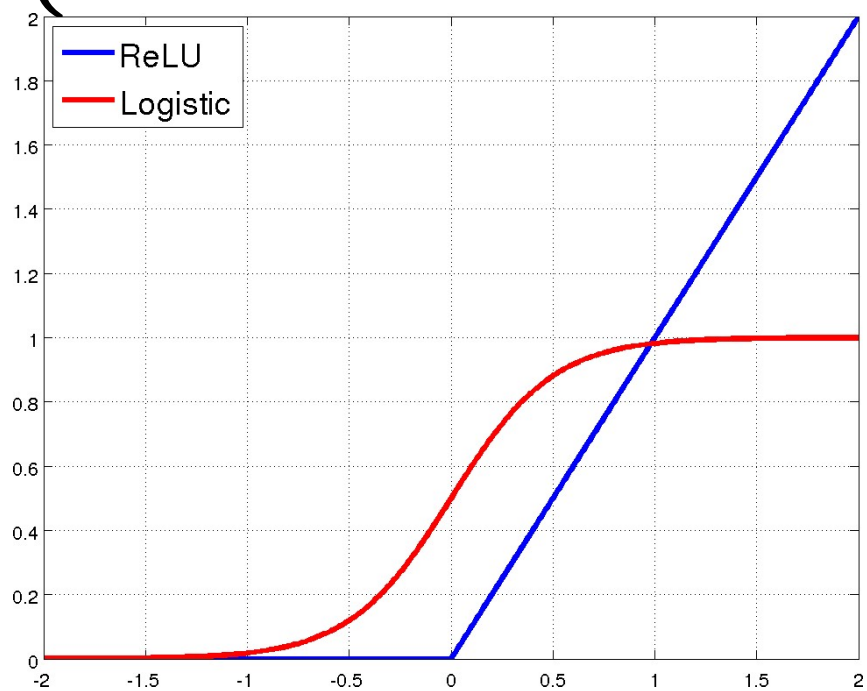


Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.
Rother et al, "Grabcut: Interactive foreground extraction using iterated graph cuts", ACM TOG 2004

Remember ReLUs?

$$h^{l+1} = \mathrm{ReLU}(h^l) = \max\{0, h^l\}$$

$$\frac{\partial h^{l+1}}{\partial h^l} = \begin{cases} 0 & \text{if } h^l < 0 \\ 1 & \text{if } h^l > 0 \end{cases} = [\![h^l > 0]\!]$$



$$h^{l+1} = \max\{0, h^l\}$$

Forward pass

h^l

1	-1	5
2	-5	-7
-3	2	4

→

1	0	5
2	0	0
0	2	4

h^{l+1}

$$\frac{\partial L}{\partial h^l} = [[h^l > 0]]\frac{\partial L}{\partial h^{l+1}}$$

Backward pass: backpropagation

-2	0	-1
6	0	0
0	-1	3

←

-2	3	-1
6	-3	1
2	-1	3

$\frac{\partial L}{\partial h^{l+1}}$

$$\frac{\partial L}{\partial h^l} = [[\frac{\partial L}{\partial h^{l+1}} > 0]]\frac{\partial L}{\partial h^{l+1}}$$

Backward pass: "deconvnet"

0	3	0
6	0	1
2	0	3

←

-2	3	-1
6	-3	1
2	-1	3

$\frac{\partial L}{\partial h^{l+1}}$

$$\frac{\partial L}{\partial h^l} = [[h^l > 0 \&\& \frac{\partial L}{\partial h^{l+1}} > 0]]\frac{\partial L}{\partial h^{l+1}}$$

Backward pass: *guided backpropagation*

0	0	0
6	0	0
0	0	3

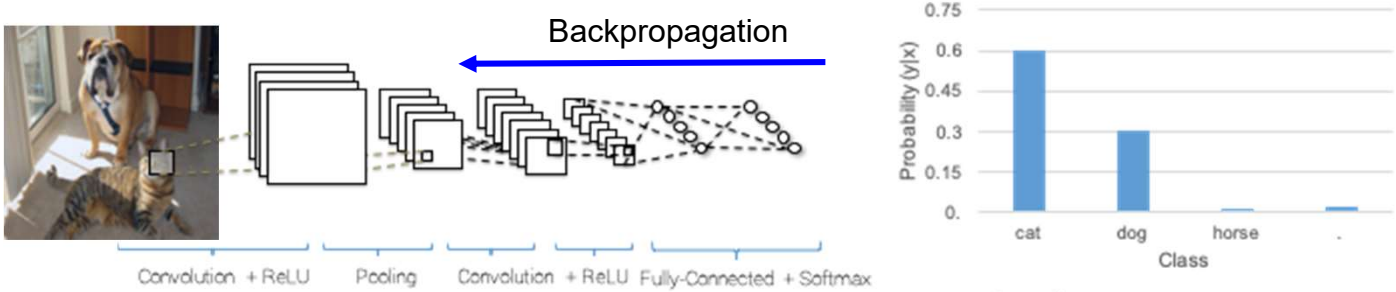
←

-2	3	-1
6	-3	1
2	-1	3

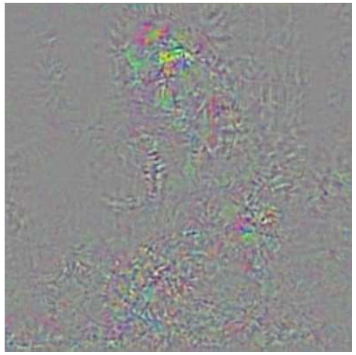
$\frac{\partial L}{\partial h^{l+1}}$

Gradient-based visualizations

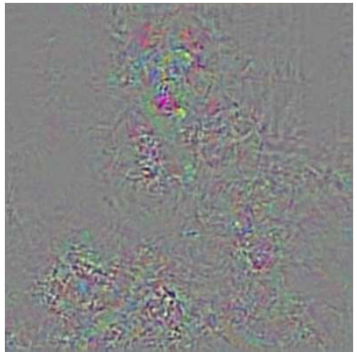
x $\qquad f(x) = y \qquad$ y



$$< w_c, x > + b \approx f(x)$$



Backprop for `cat'



Backprop for `dog'

$$w_c = \frac{\partial y_c}{\partial x}\bigg|_{x=x_0}$$

Guided Backprop for `cat'

Guided Backprop for `dog'

Noisy

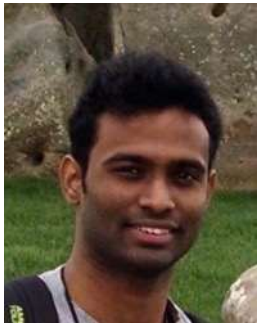
Not Class-Discriminative

Grad-CAM

Visual Explanations from Deep Networks via Gradient-based Localization

[ICCV '17]

Ramprasaath Selvaraju



Michael Cogswell

Abhishek Das

Ramakrishna Vedantam

Devi Parikh

Dhruv Batra

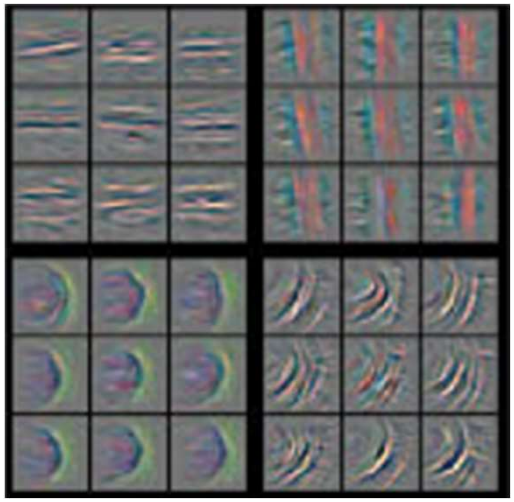
VirginiaTech

facebook research

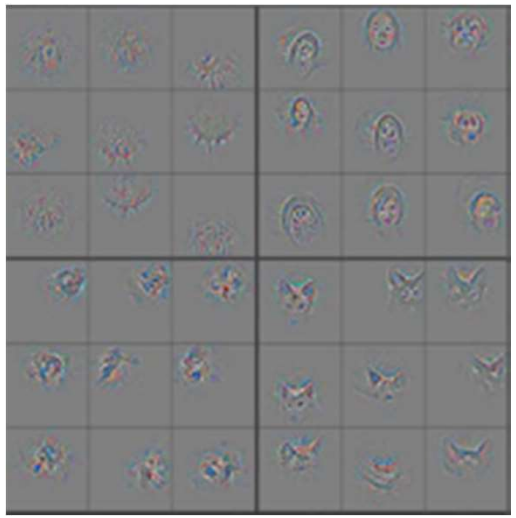
Georgia Institute of Technology

Grad-CAM Motivation

- Perturb semantic neurons in the image and see how it affects the decision



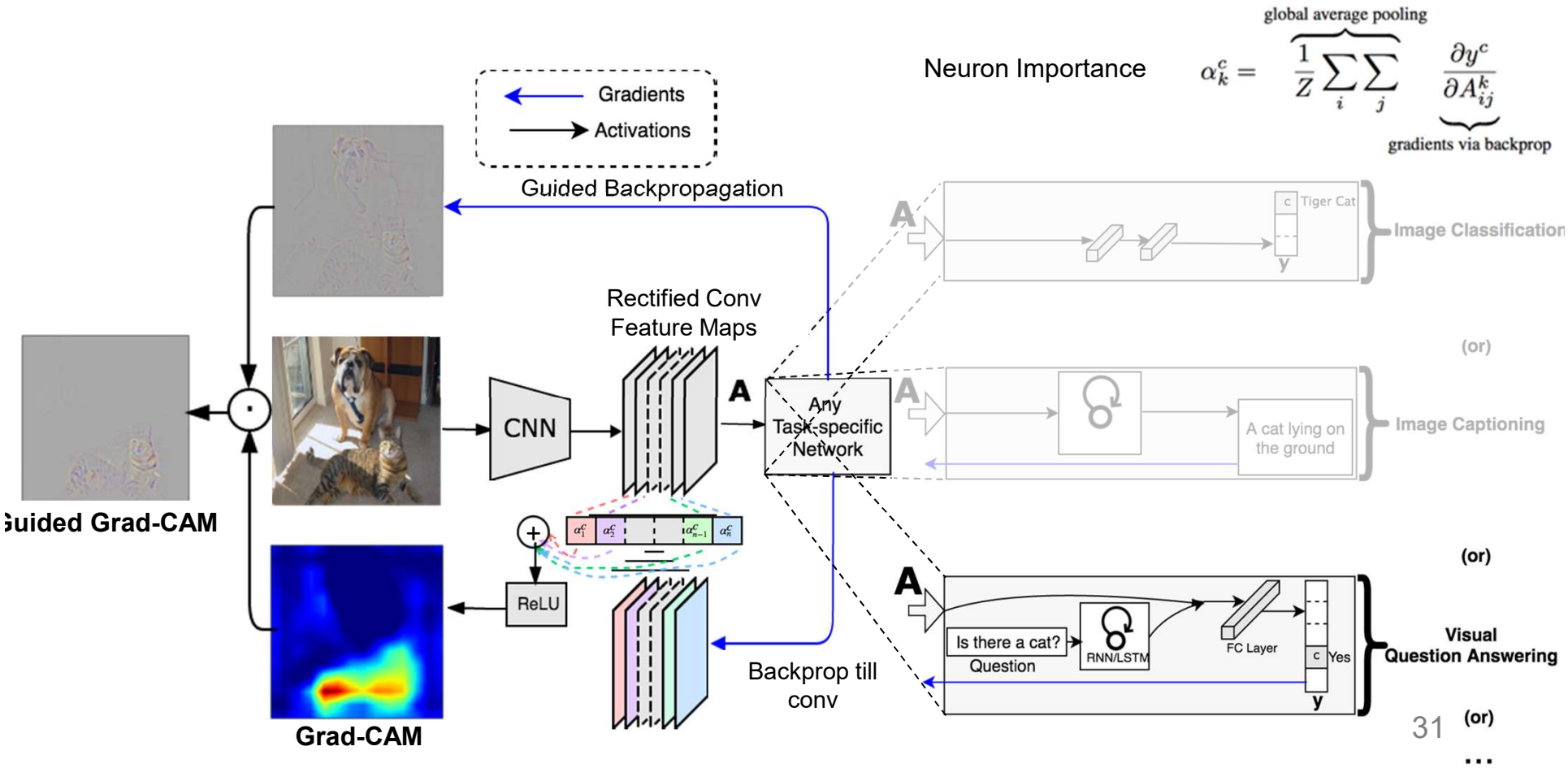
Lower



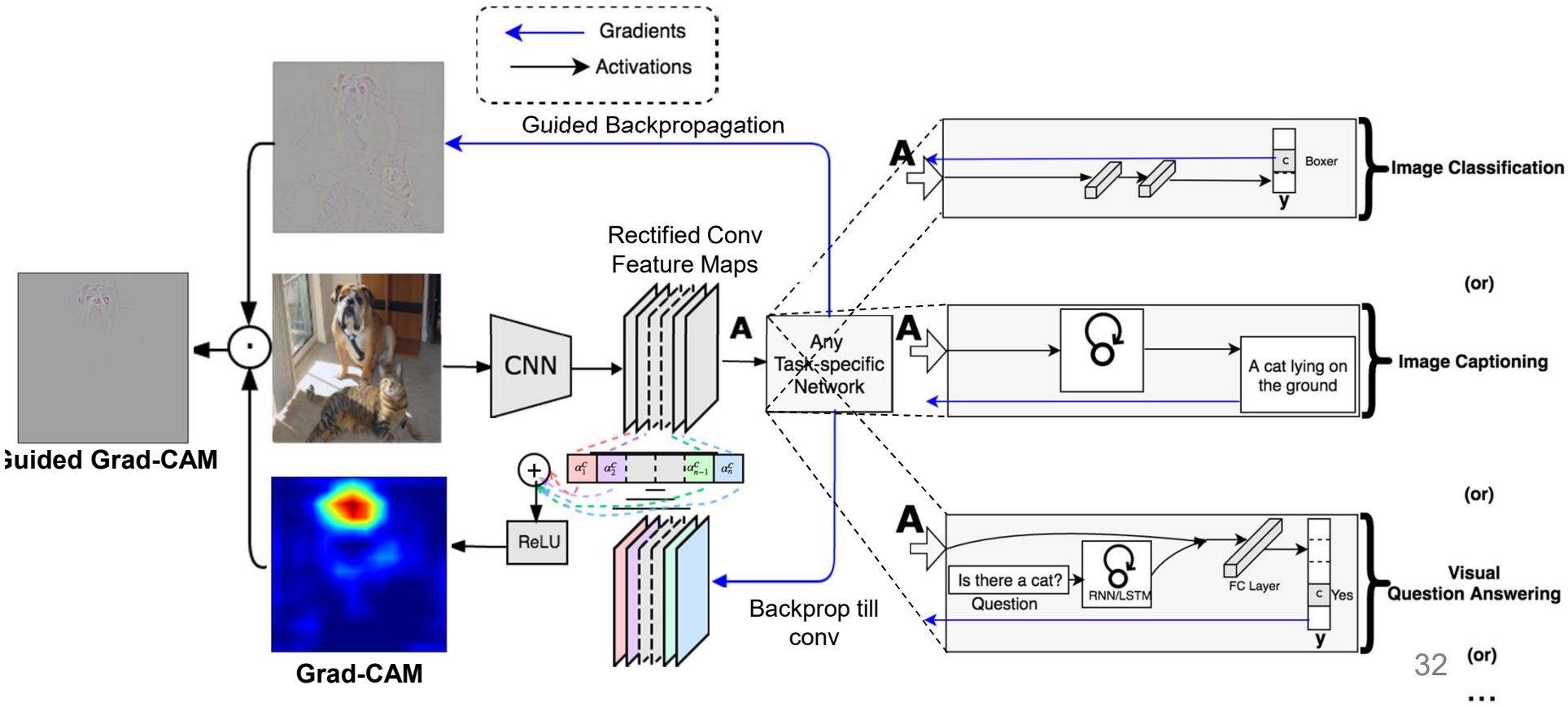
Higher

- Last convolutional layer forms a best compromise between high-level semantics and detailed spatial resolution

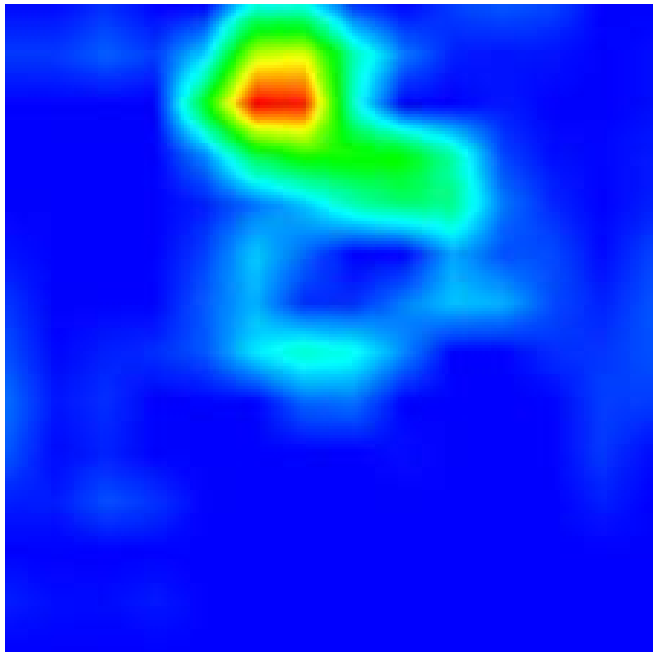
Guided Grad-CAM



Guided Grad-CAM



Grad-CAM Visual Explanations for VQA

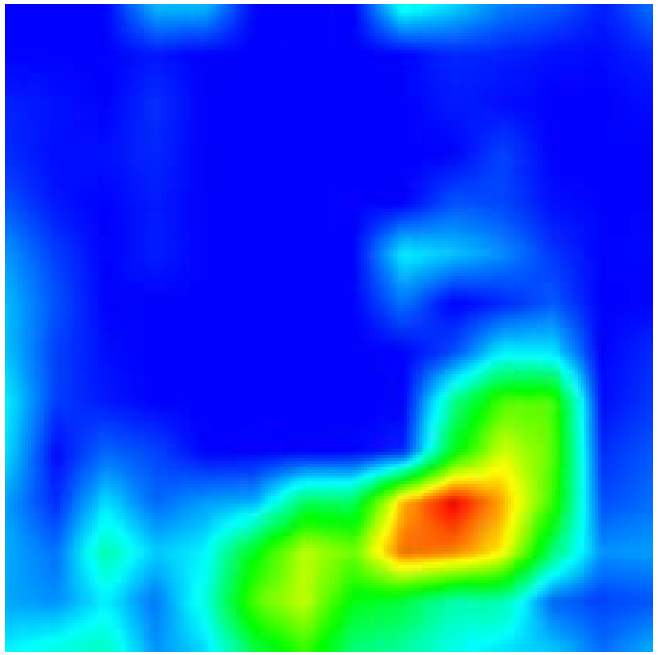


What animal is in this picture? Dog

- Even simple non-attention based CNN + LSTM models learn to look at appropriate regions

Slide Credit: Abhishek Das

Grad-CAM Visual Explanations for VQA



What animal is in this picture? Cat

- Even simple non-attention based CNN + LSTM models learn to look at appropriate regions

Slide Credit: Abhishek Das

Generating prototypical images for a class

Visualizing CNN features: Gradient Ascent on Pixels

(Guided) backprop:
Find the part of an image that a neuron responds to?

Gradient ascent on pixels:
Generate a synthetic image that maximally activates a neuron

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

Neuron value

Natural image regularizer

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

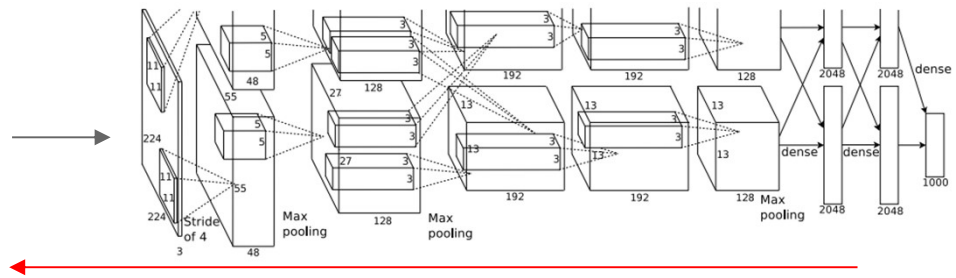
Visualizing CNN features: Gradient Ascent on Pixels

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

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 on Pixels

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

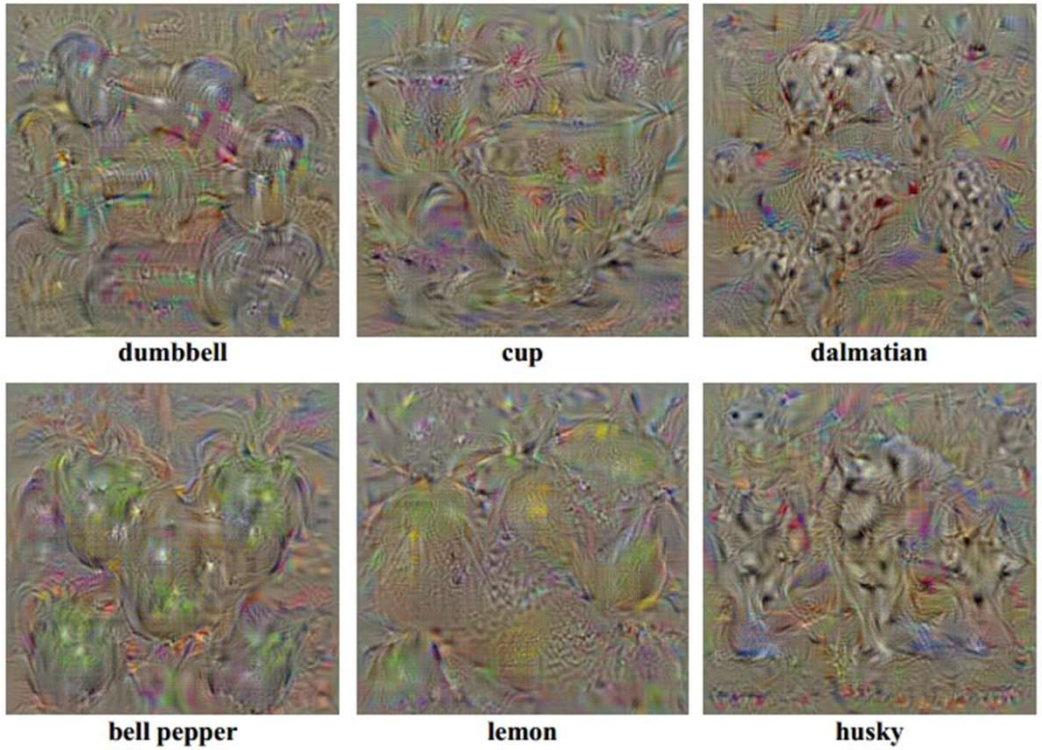
Simple regularizer: Penalize L2 norm of generated image

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

Visualizing CNN features: Gradient Ascent on Pixels

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

Simple regularizer: Penalize L2 norm of generated image



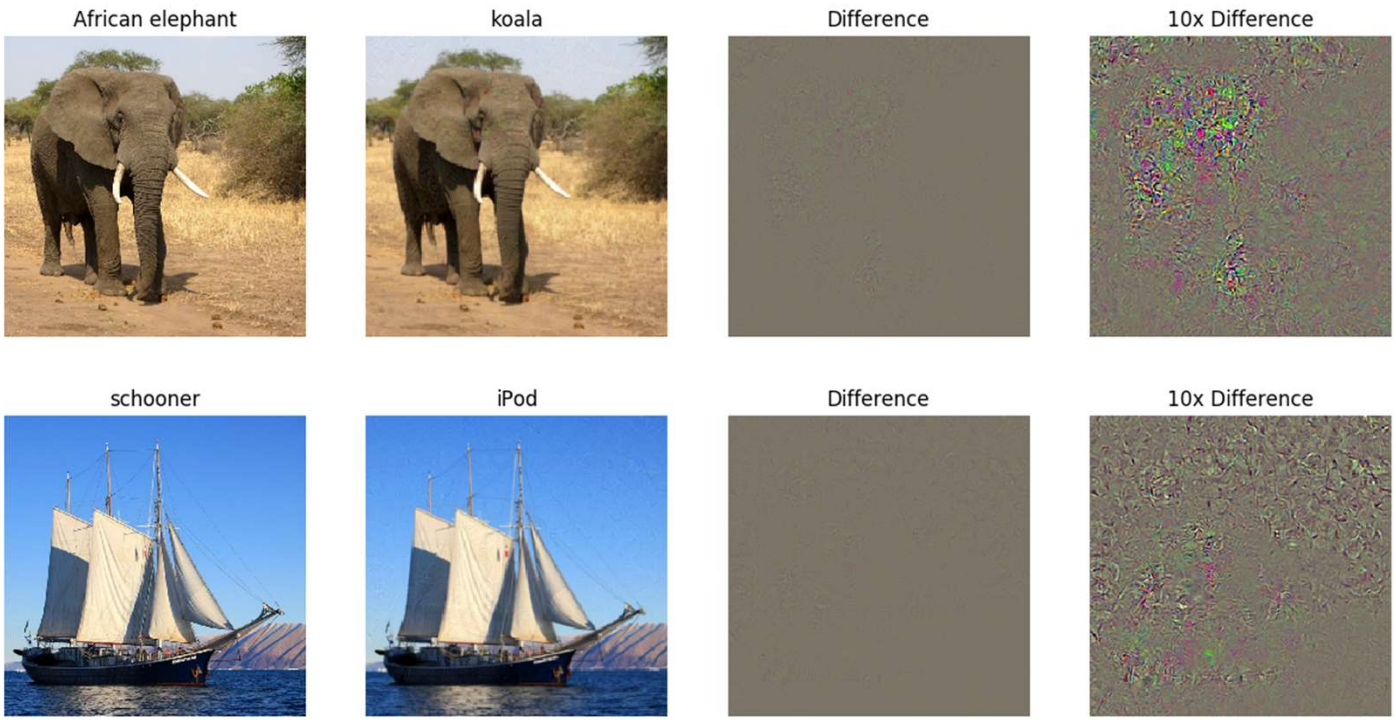
Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

Fooling Images / Adversarial Examples

(1)Start from an arbitrary image
(2)Pick an arbitrary class
(3)Modify the image to maximize the class
(4)Repeat until network is fooled

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Fooling Images / Adversarial Examples



Given the feature vector can you reconstruct the image?

Feature Inversion

Given a CNN feature vector for an image, find a new image that:

- Matches the given feature vector
- "looks natural" (image prior regularization)

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^{H \times W \times C}}{\operatorname{argmin}} \ \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda \mathcal{R}(\mathbf{x})$$

Given feature vector

Features of new image

$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

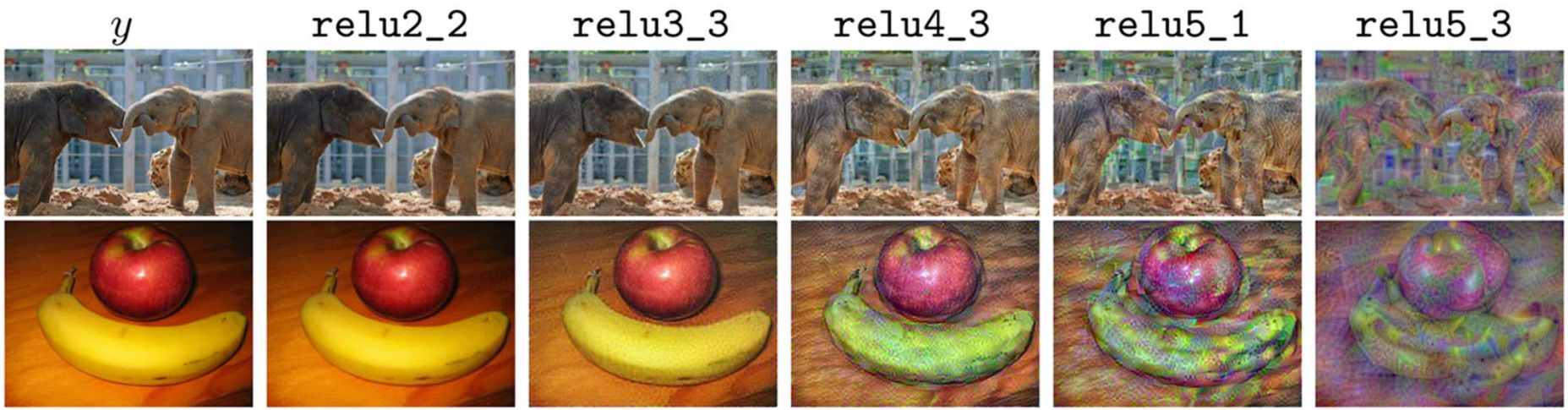
$$\mathcal{R}_{V^\beta}(\mathbf{x}) = \sum_{i,j} \left((x_{i,j+1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2 \right)^{\frac{\beta}{2}}$$

Total Variation regularizer
(encourages spatial smoothness)

Mahendran and Vedaldi, "Understanding Deep Image Representations by Inverting Them", CVPR 2015

Feature Inversion

Reconstructing from different layers of VGG-16



Mahendran and Vedaldi, “Understanding Deep Image Representations by Inverting Them”, CVPR 2015
Figure from Johnson, Alahi, and Fei-Fei, “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”, ECCV 2016. Copyright Springer, 2016.
Reproduced for educational purposes.

Side-effect - style transfer

- *Content representation:* feature map at each layer
- *Style representation:* Covariance matrix at each layer
  - Spatially invariant
  - Average second-order statistics

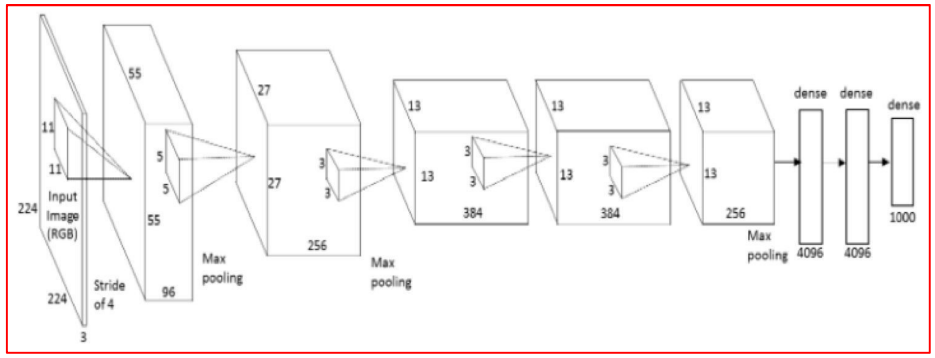
- Idea: Optimize x to match content of one image and style of another

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}.$$

Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." *arXiv preprint arXiv:1508.06576* (2015).

Neural Style

Step 1: Extract **content targets** (ConvNet activations of all layers for the given content image)

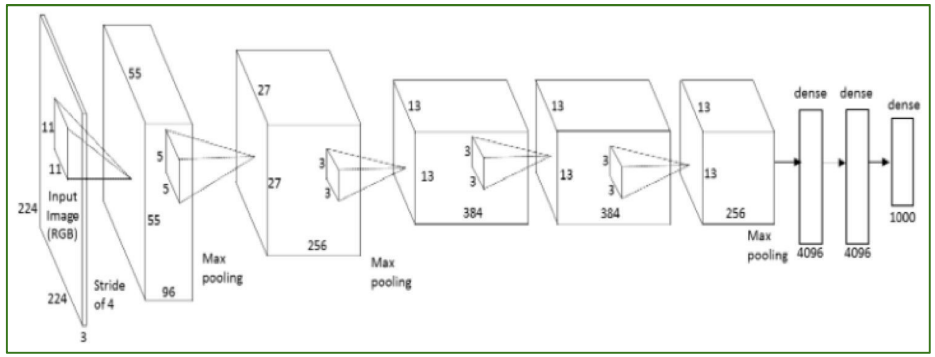


content activations

e.g.
at CONV5_1 layer we would have a [14x14x512] array of target activations

Neural Style

Step 2: Extract **style targets** (Gram matrices of ConvNet activations of all layers for the given style image)



style gram matrices

$$G = V^{\mathrm{T}} V$$

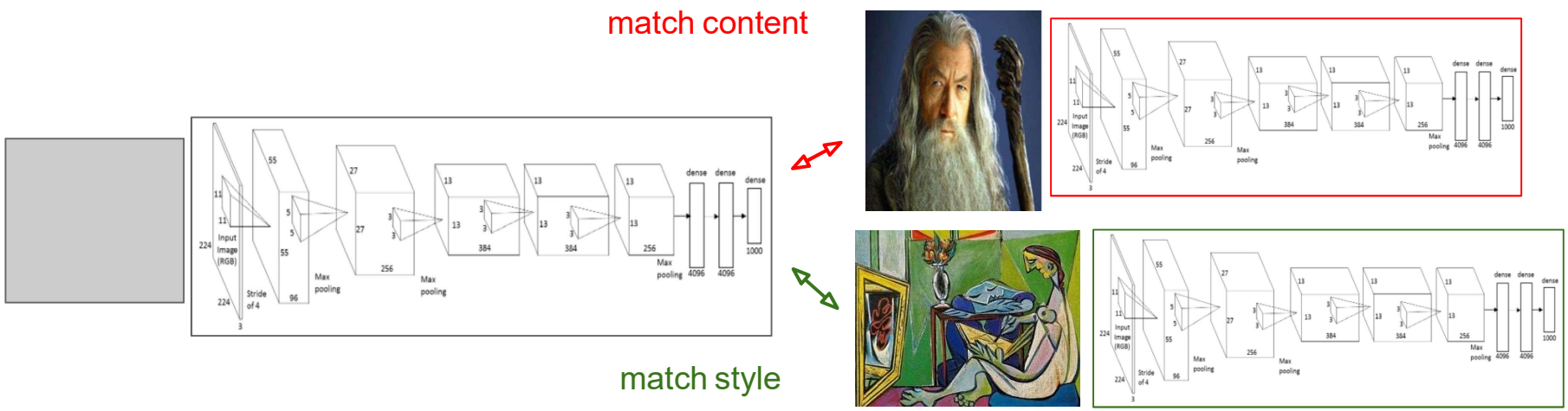
e.g.
at CONV1 layer (with [224x224x64] activations) would give a [64x64] Gram matrix of all pairwise activation covariances (summed across spatial locations)

Neural Style

Step 3: Optimize over image to have:

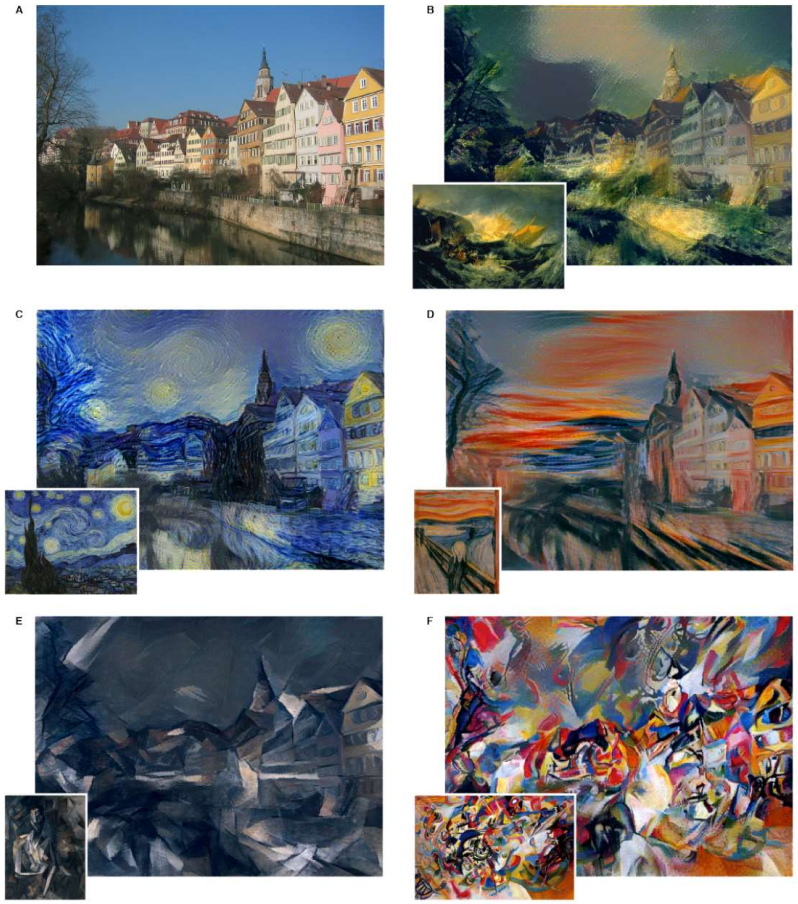
- The **content** of the content image (activations match content)
- The **style** of the style image (Gram matrices of activations match style)

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$



Adapted from Andrej Karpathy

Style transfer



Summary of Methods

- We will use ***gradients*** to understand and visualize what the network has captured.
  - We will also sometimes use **activations** as a representation of the image (just as the last hidden feature vector can be a low-dimensional representation of the image content, we can do this across all layers)

- Each method consists of:
  - **What the target category & layer/filter is:** sometimes a target ***category***, sometimes a specific ***layer*** or slice of the output map (corresponding to a ***filter/kernel)*** is also chosen
  - **An input:** sometimes an image, sometimes a zero image
  - **Optimization criteria:** What is being optimized, sometimes the class score is maximized (gradient *ascent*), sometime a loss is *minimized*