max planck institut informatik

UNIVERSITÄT DES SAARLANDES

High Level Computer Vision

Convolutional Neural Networks, Network Visualization, Feature Generalization @ May 8, 2019

Bernt Schiele - schiele@mpi-inf.mpg.de
Mario Fritz - mfritz@mpi-inf.mpg.de

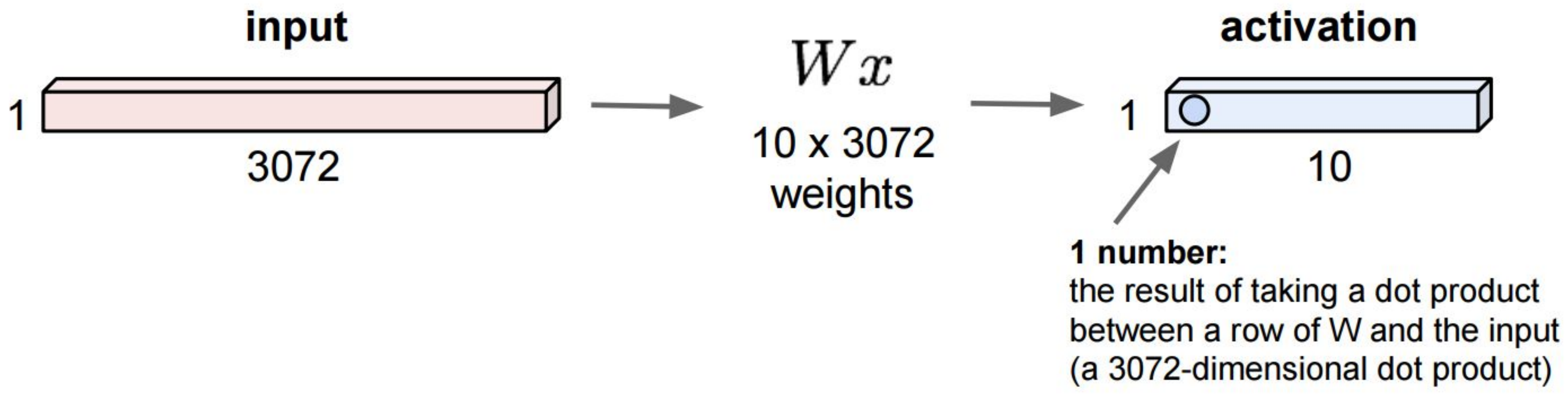
https://www.mpi-inf.mpg.de/hlcv

Overview Today

- Deep dive into convolutional networks
- Visualizing convolutional networks
- Feature Generalization
  - “pre-training” on large dataset,
    “fine-tuning” on target dataset

Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

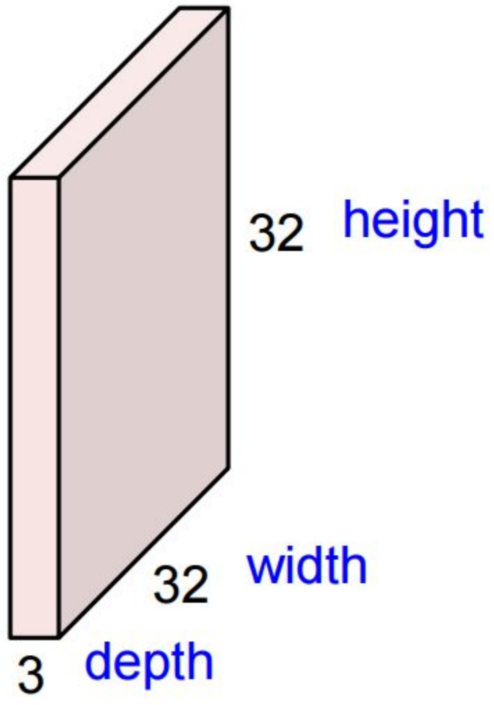


1 number:
the result of taking a dot product between a row of W and the input (a 3072-dimensional dot product)

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

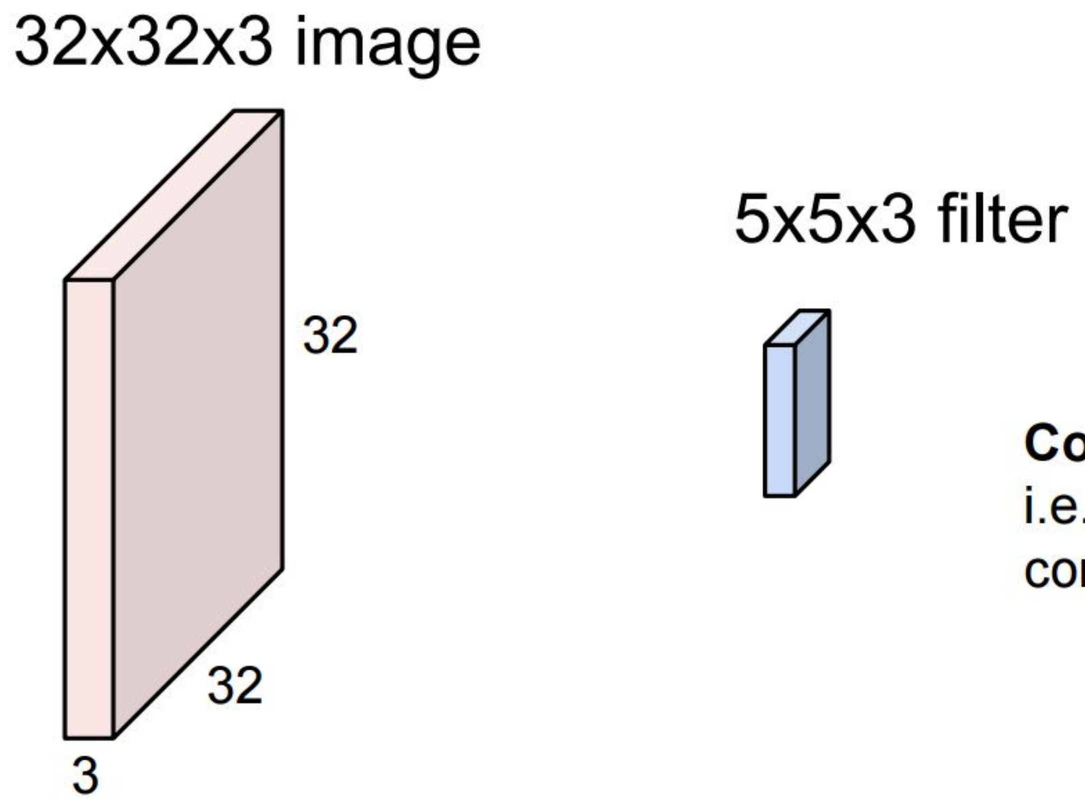
Convolutional Layer

32x32x3 image -> preserve spatial structure



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Convolutional Layer

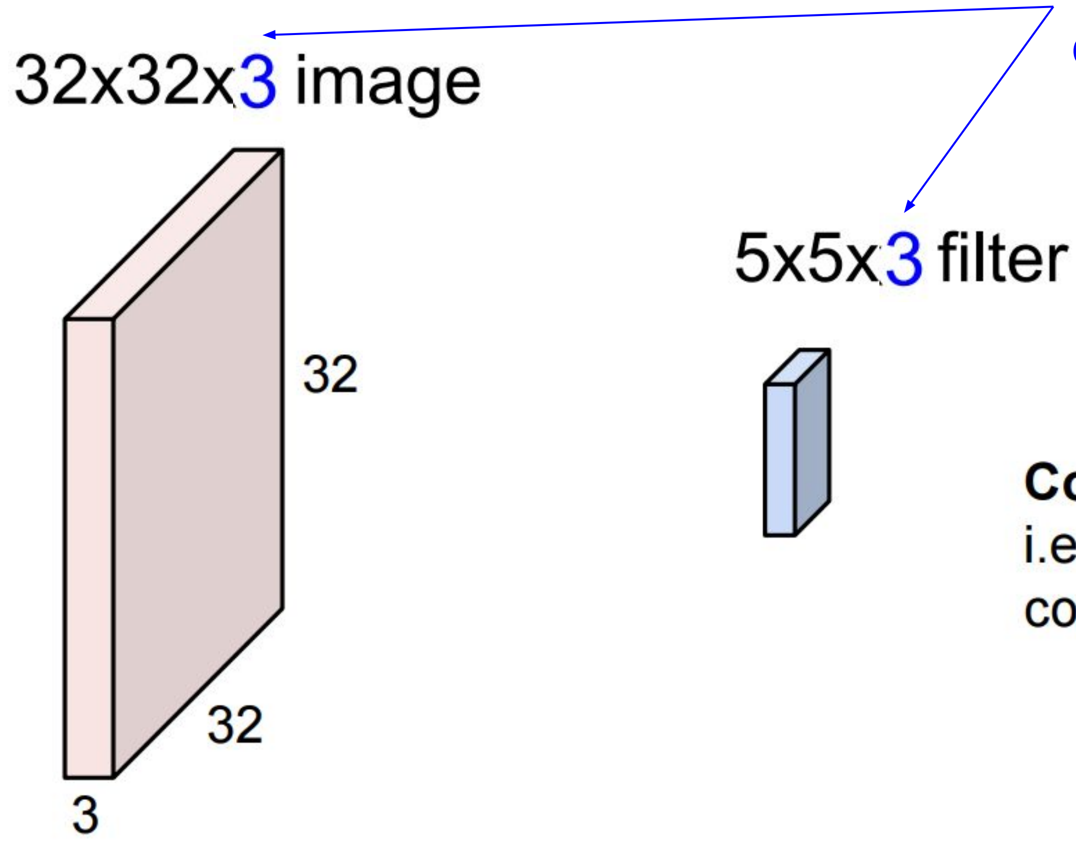


Convolve the filter with the image
i.e. “slide over the image spatially, computing dot products”

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Convolutional Layer

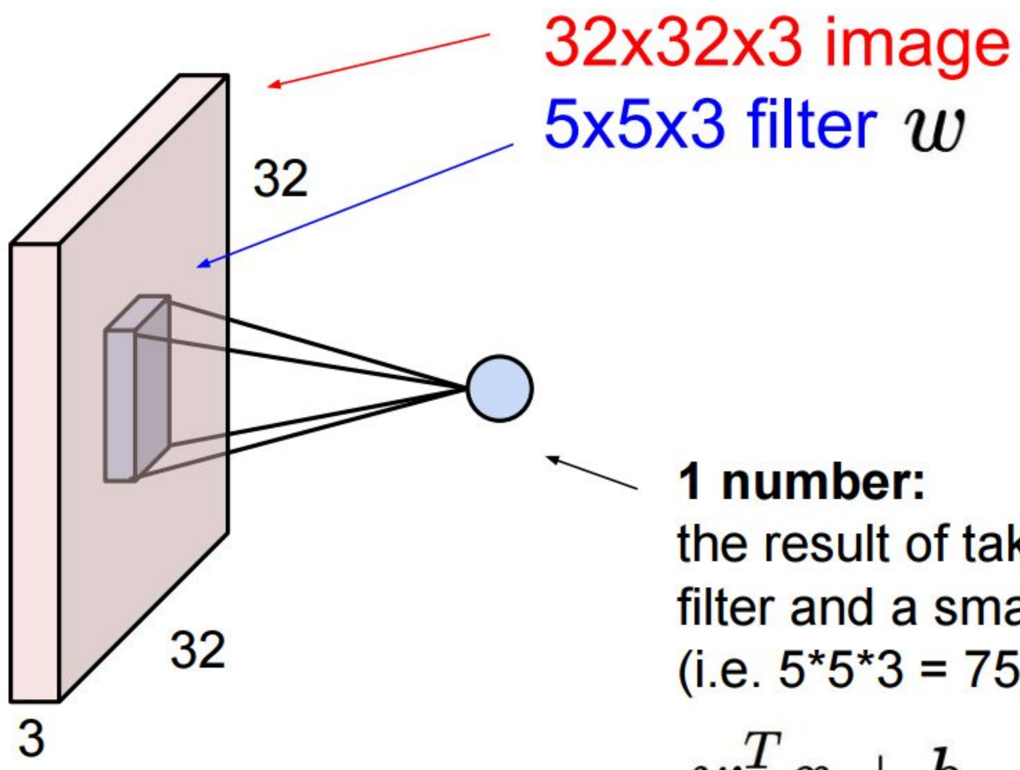
Filters always extend the full depth of the input volume



Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Convolutional Layer

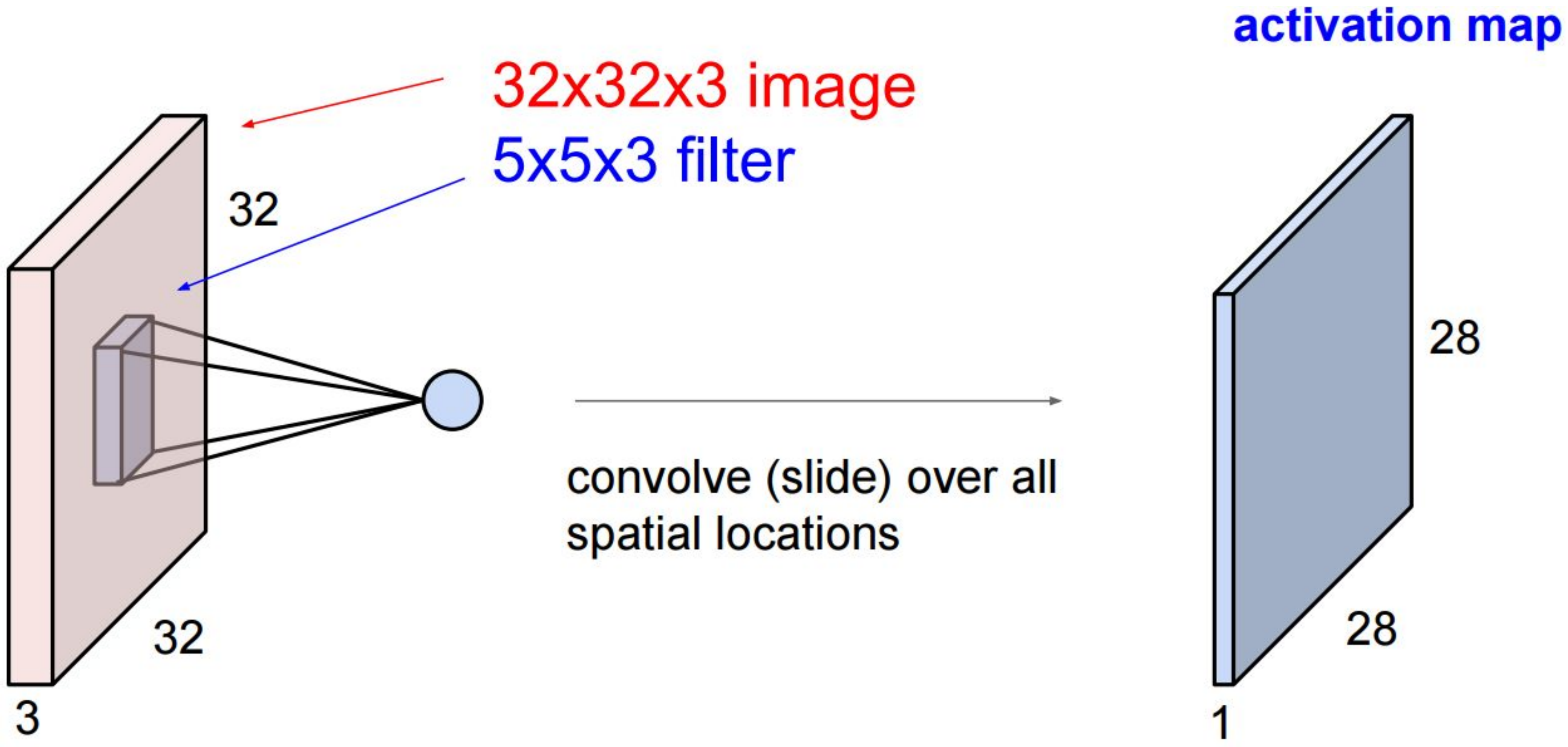


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

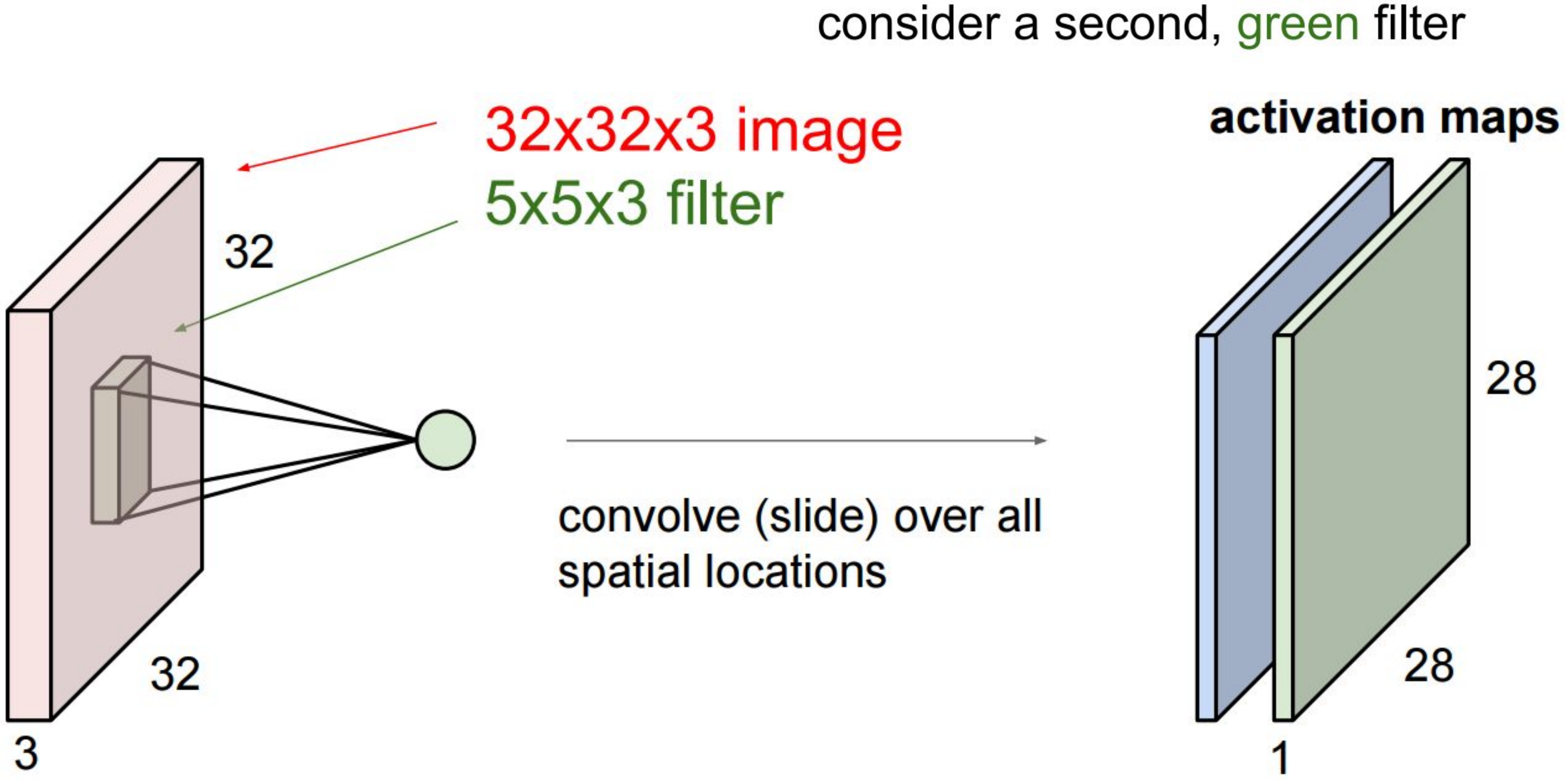
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Convolutional Layer



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

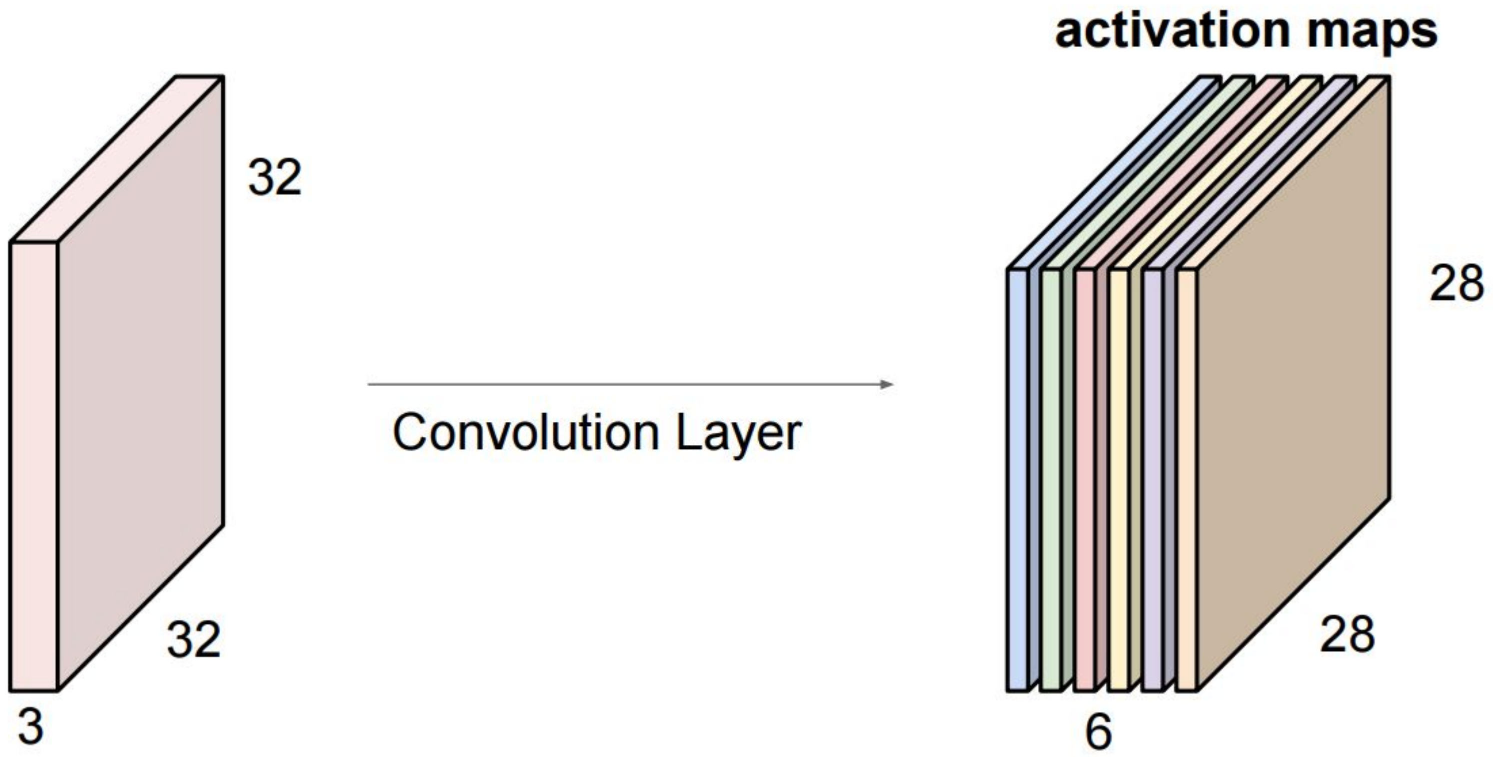
Convolutional Layer


consider a second, green filter
32x32x3 image
5x5x3 filter
32
32
3
activation maps
28
28
1
convolve (slide) over all spatial locations

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Convolutional Layer

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

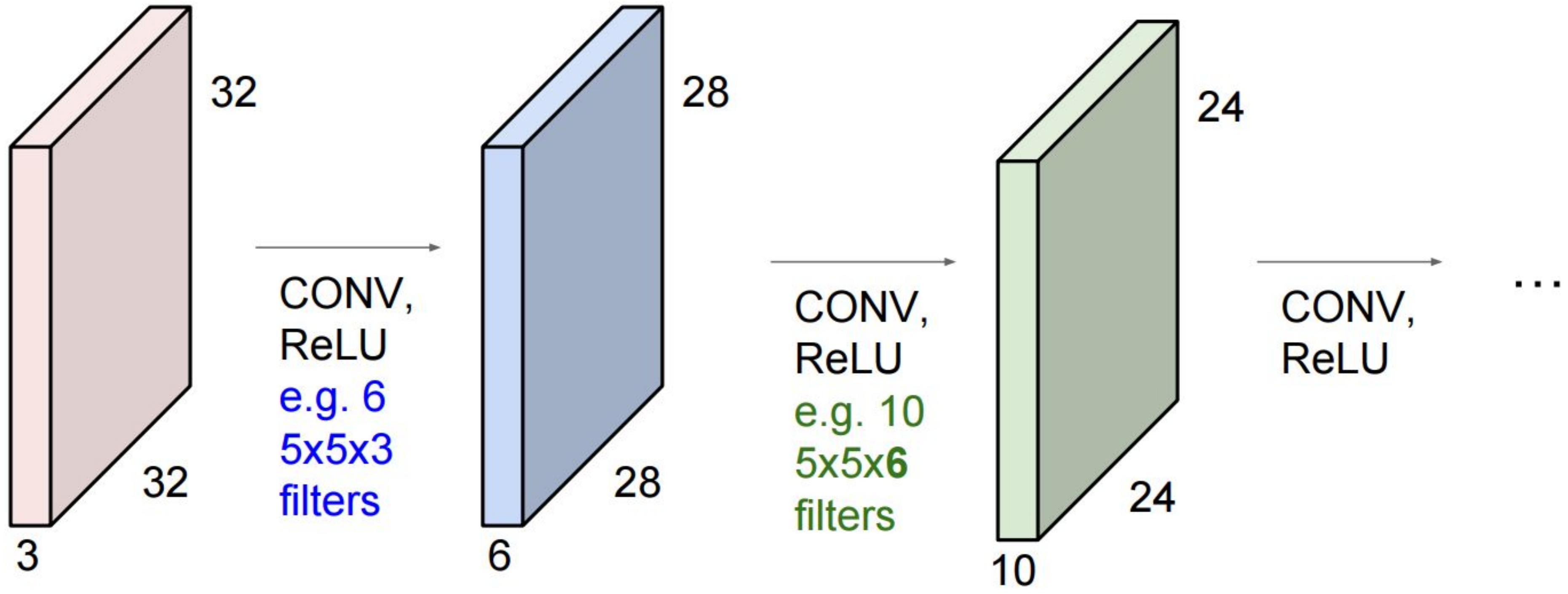


We stack these up to get a "new image" of size 28x28x6!

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

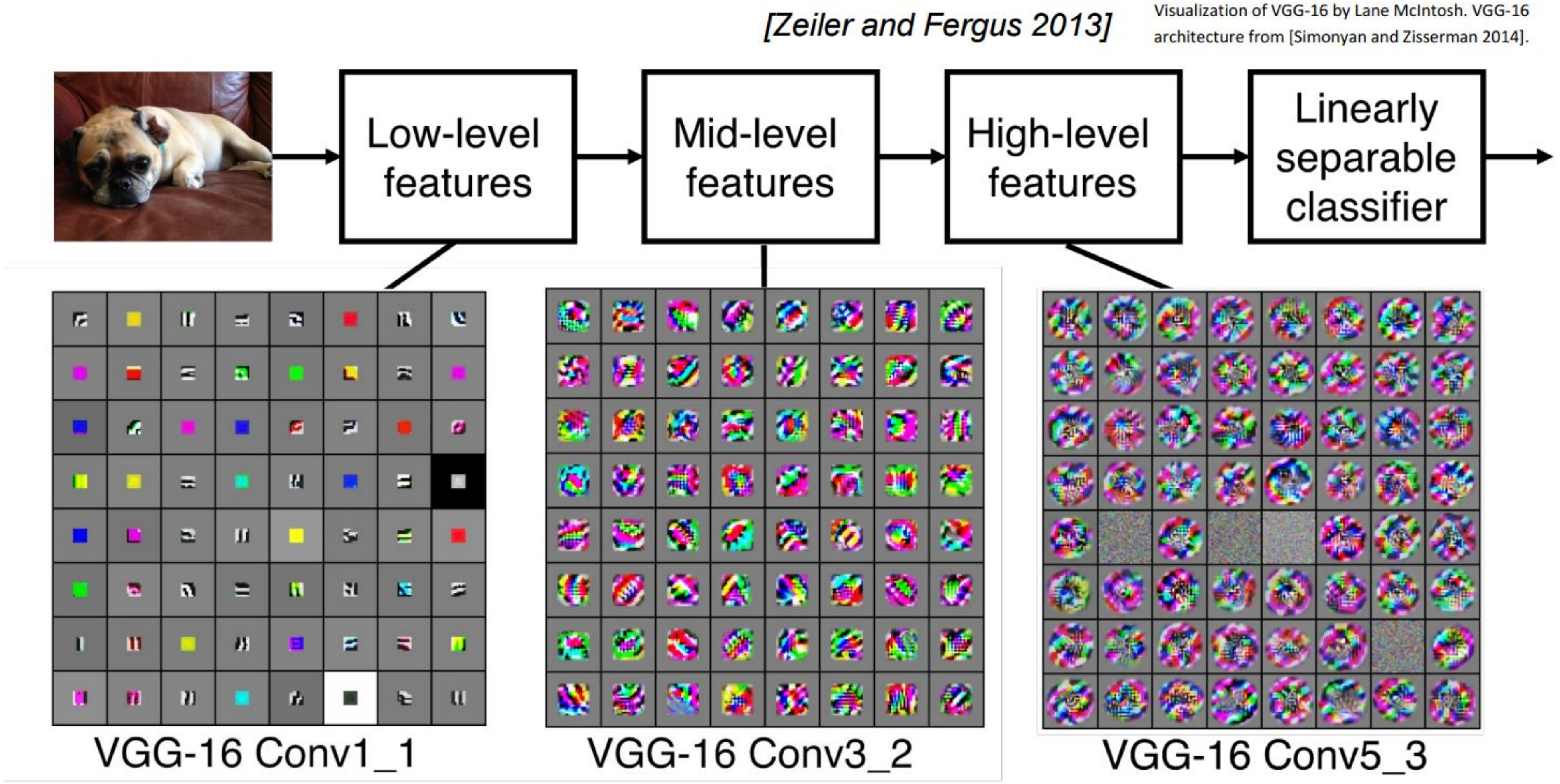
Convolutional Network

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



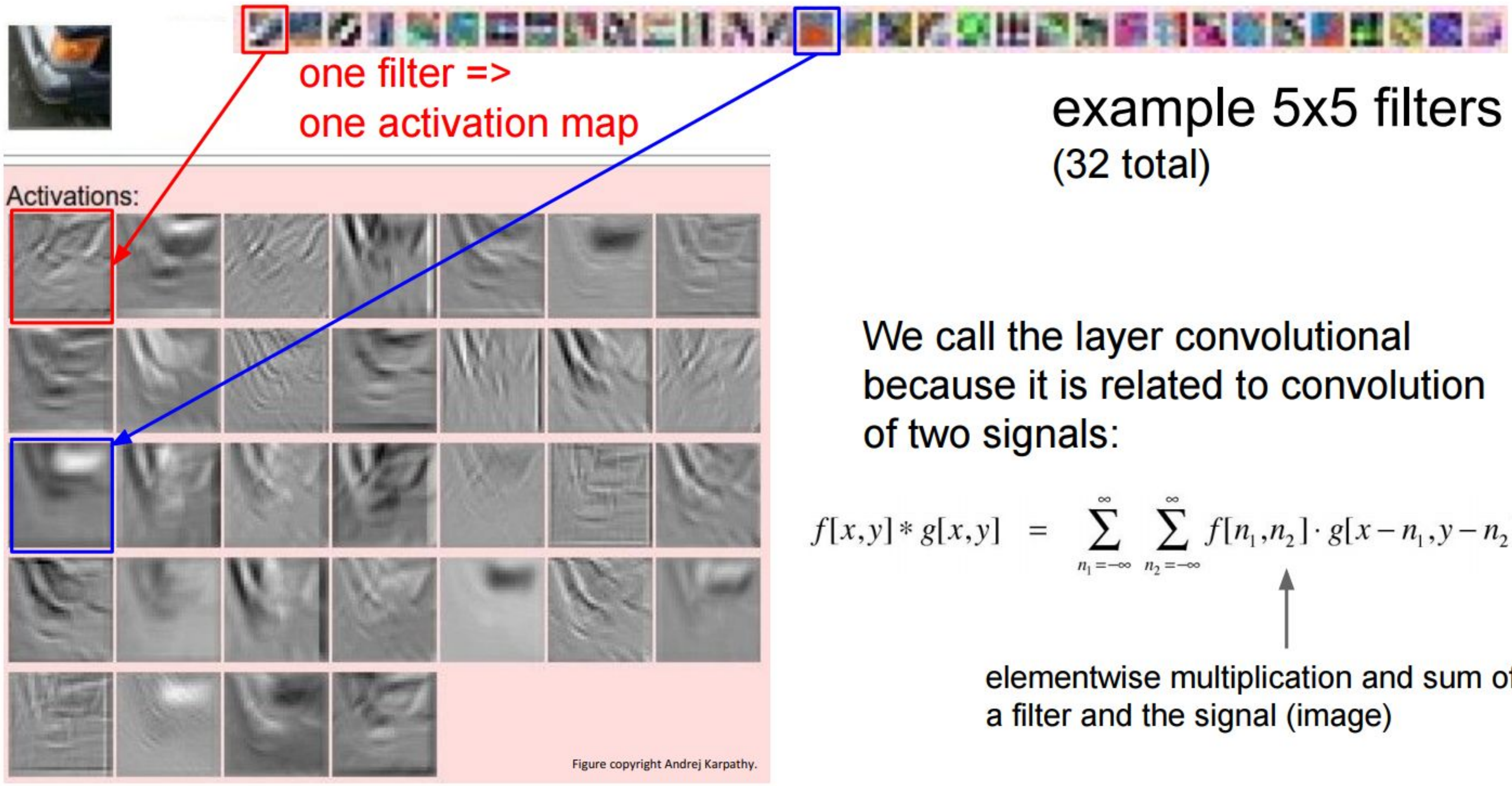
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Hierarchical Features


[Zeiler and Fergus 2013]
Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].
Low-level features
Mid-level features
High-level features
Linearly separable classifier
VGG-16 Conv1_1
VGG-16 Conv3_2
VGG-16 Conv5_3

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Activation Maps



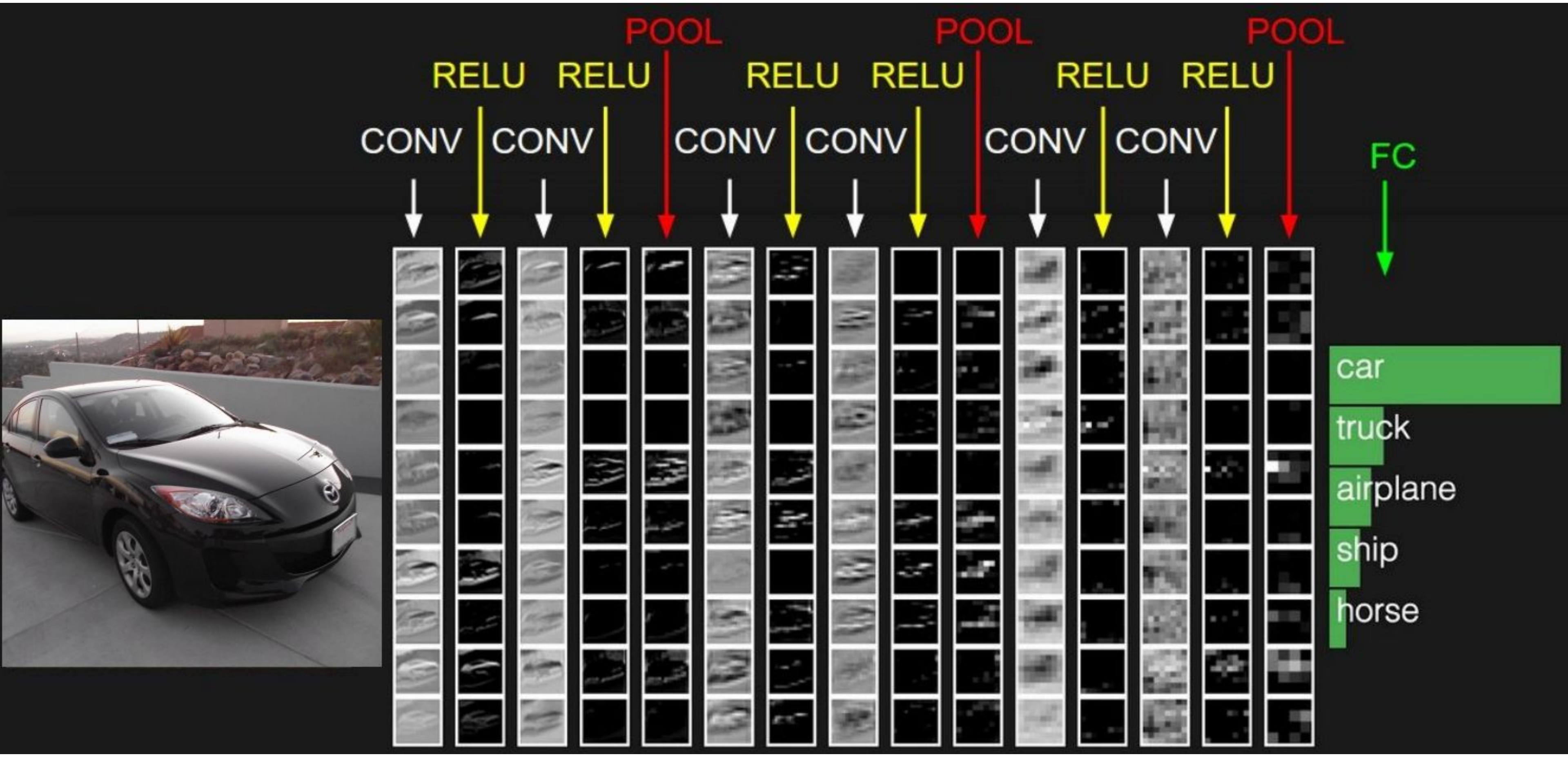
example 5x5 filters
(32 total)

We call the layer convolutional because it is related to convolution of two signals:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

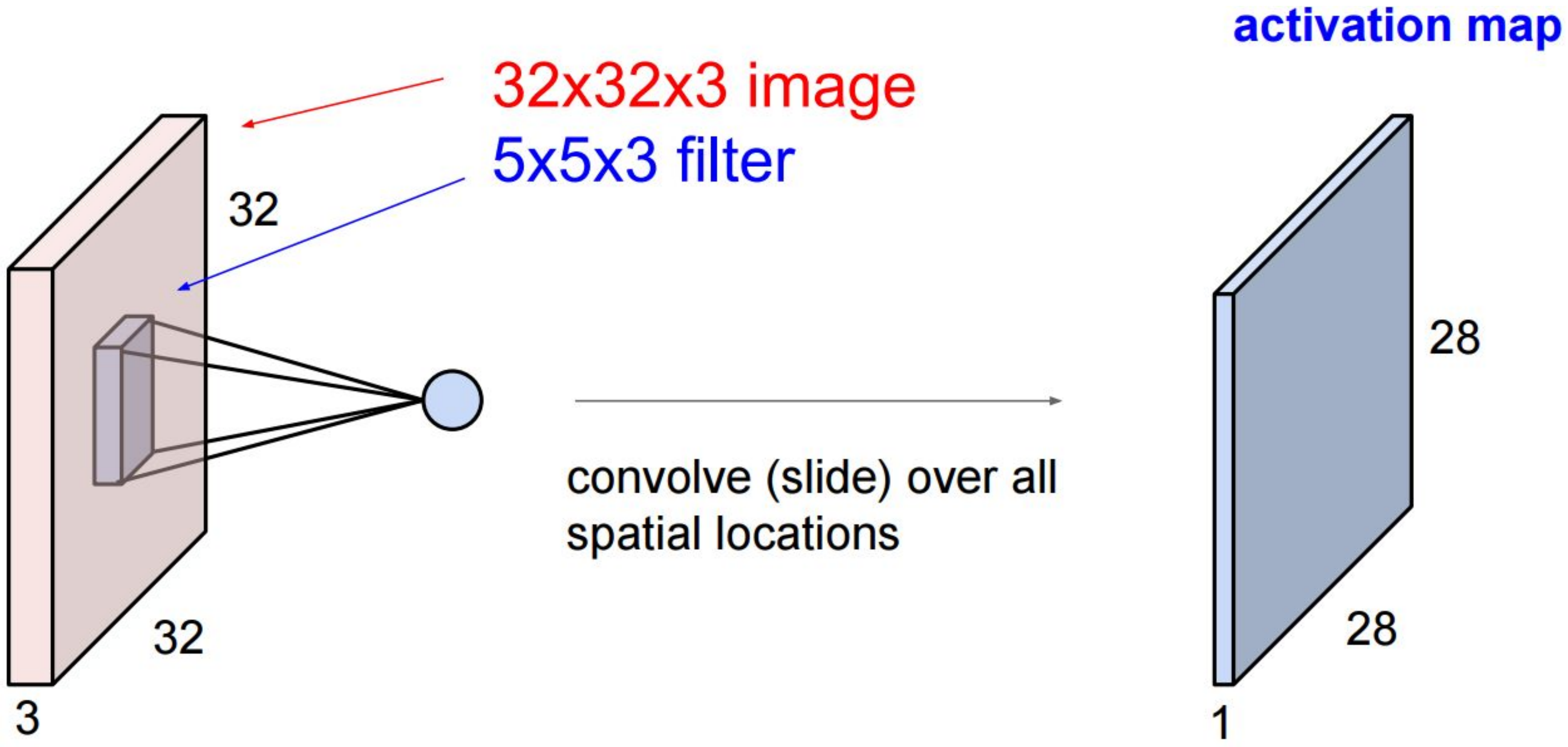
elementwise multiplication and sum of a filter and the signal (image)

Convolutional Network for classification



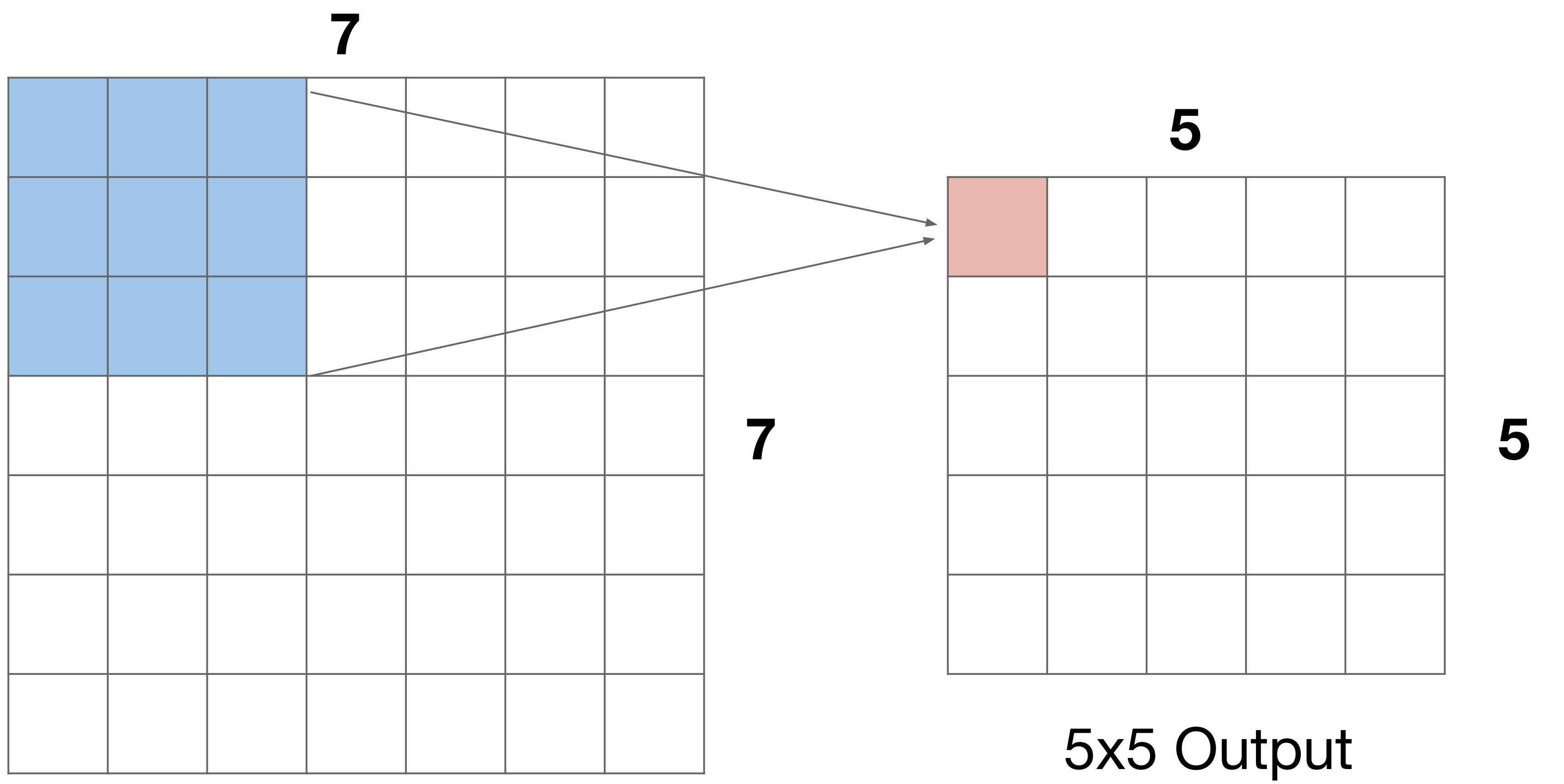
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Closer look at spatial dimensions



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

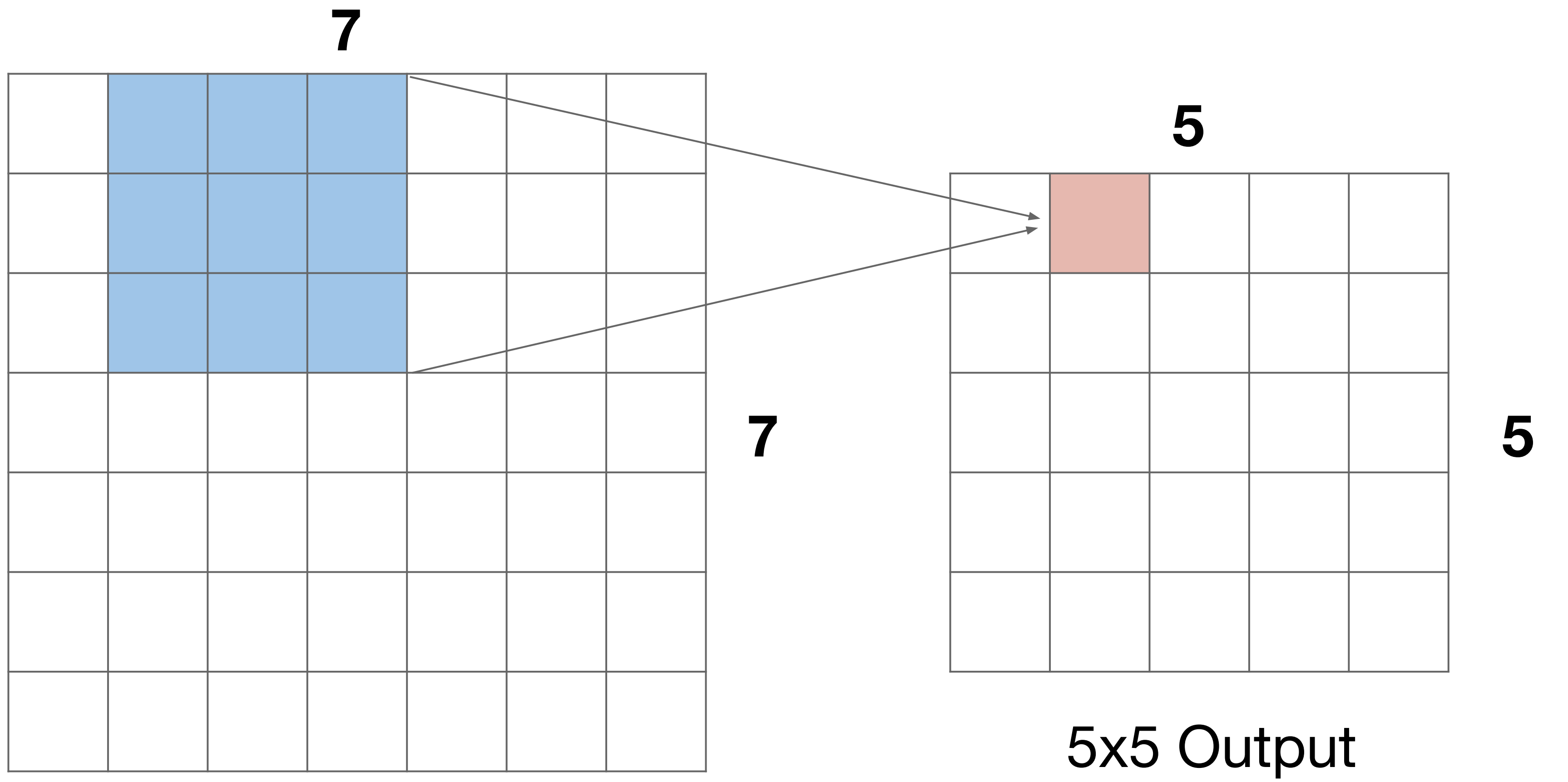
Closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter

5x5 Output

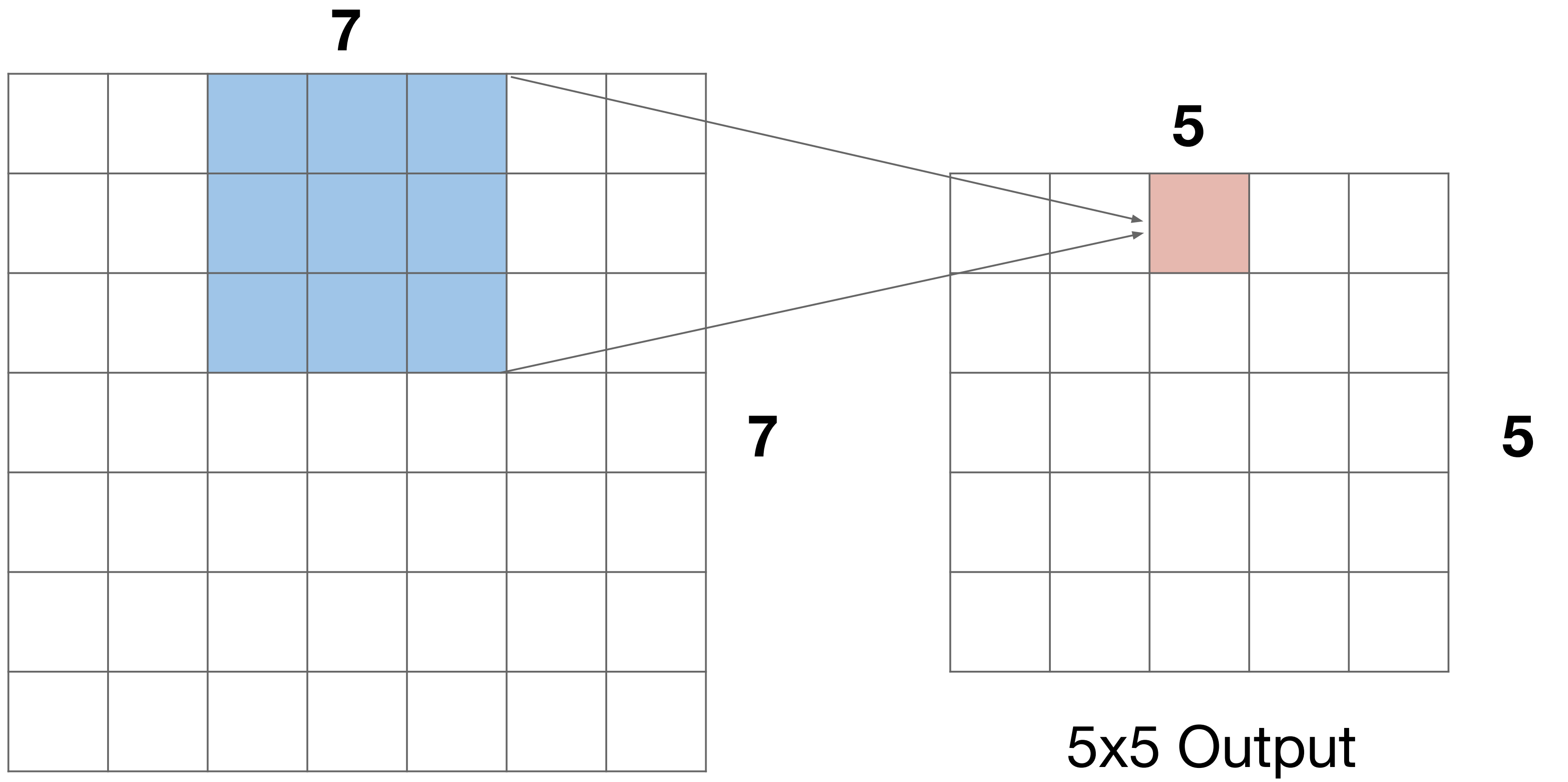
Closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter

5x5 Output

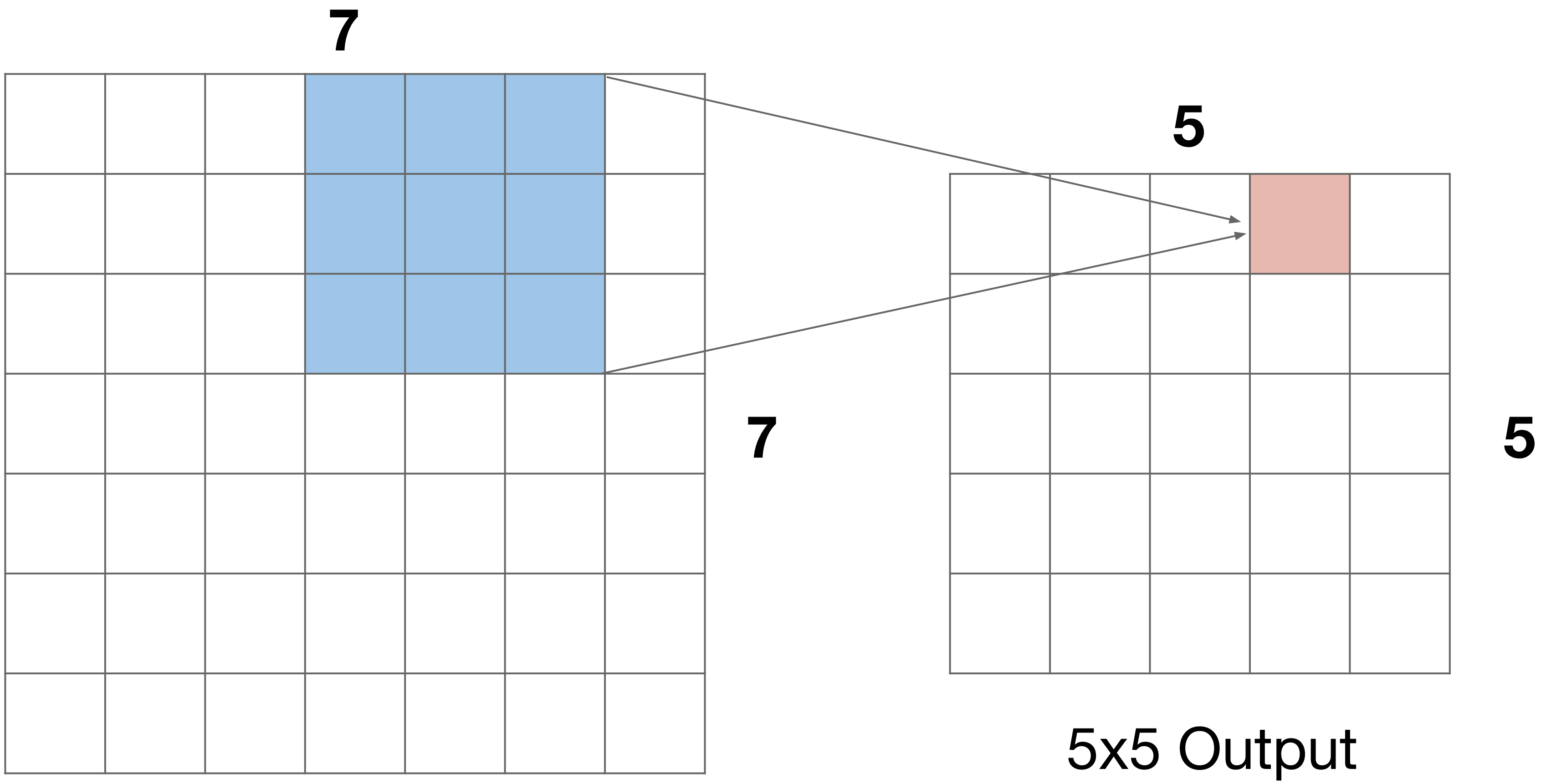
Closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter

5x5 Output

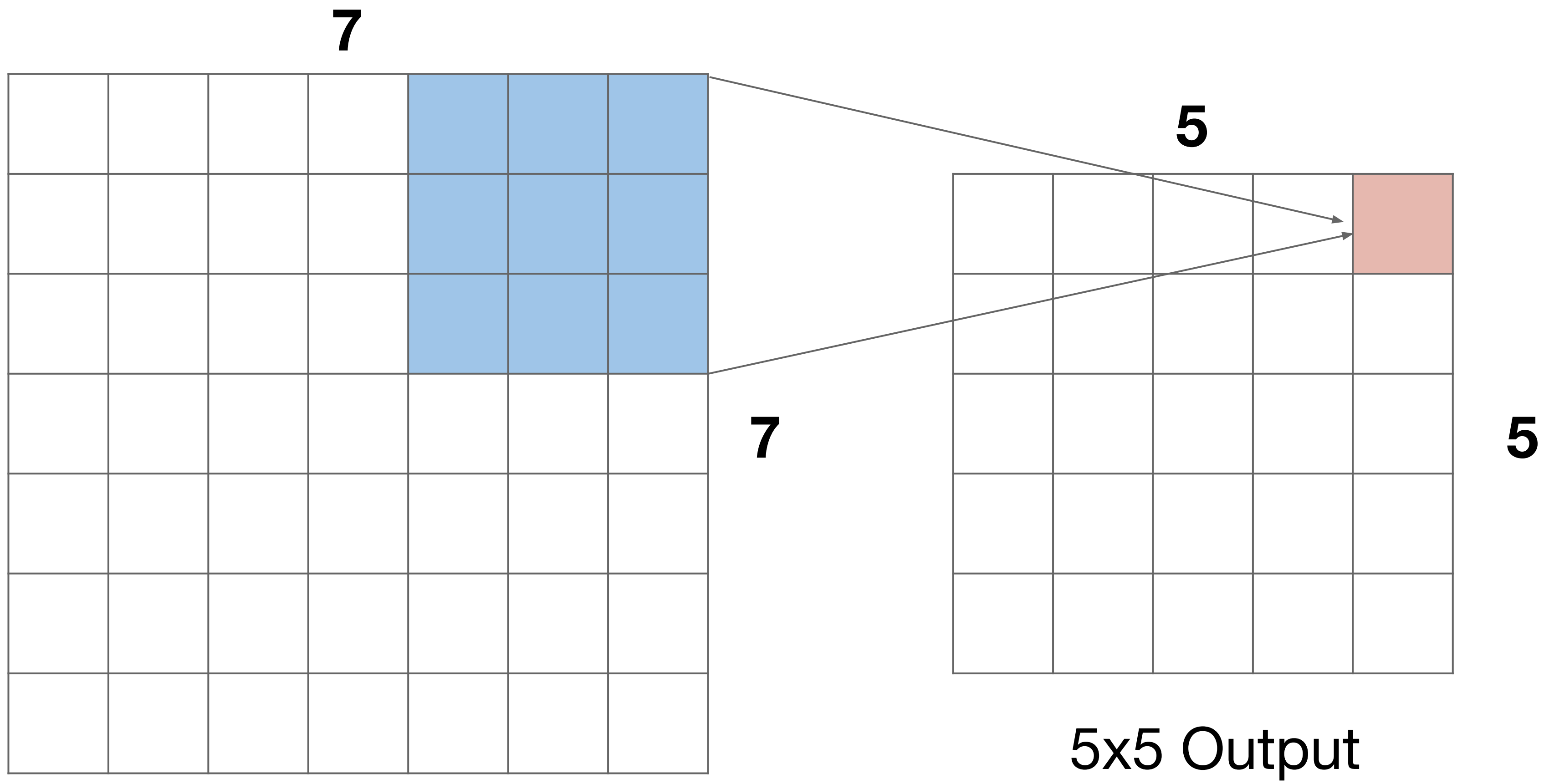
Closer look at spatial dimensions



5x5 Output

7x7 input (spatially)
assume 3x3 filter

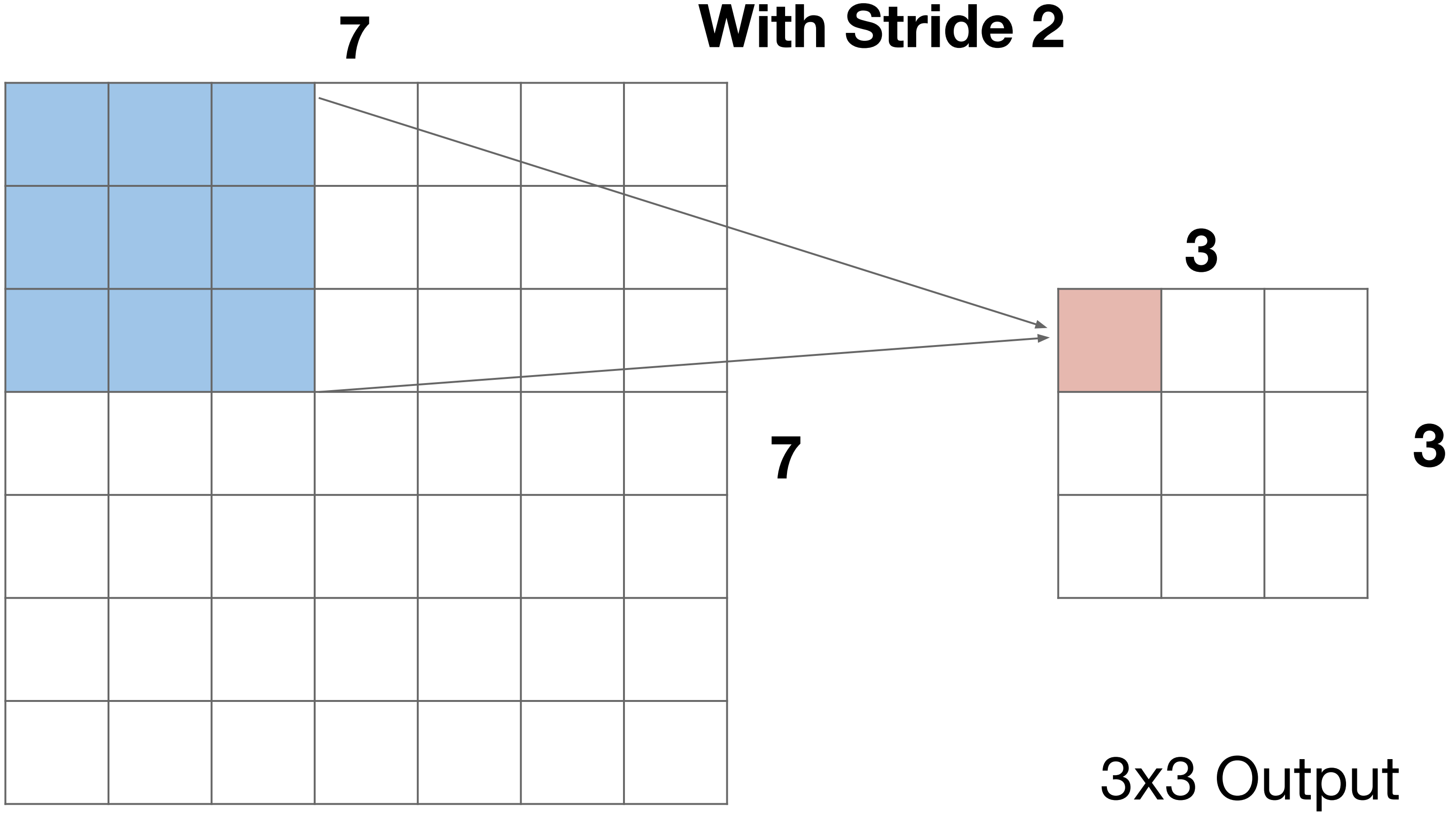
Closer look at spatial dimensions



7x7 input (spatially)
assume 3x3 filter

Closer look at spatial dimensions

With Stride 2

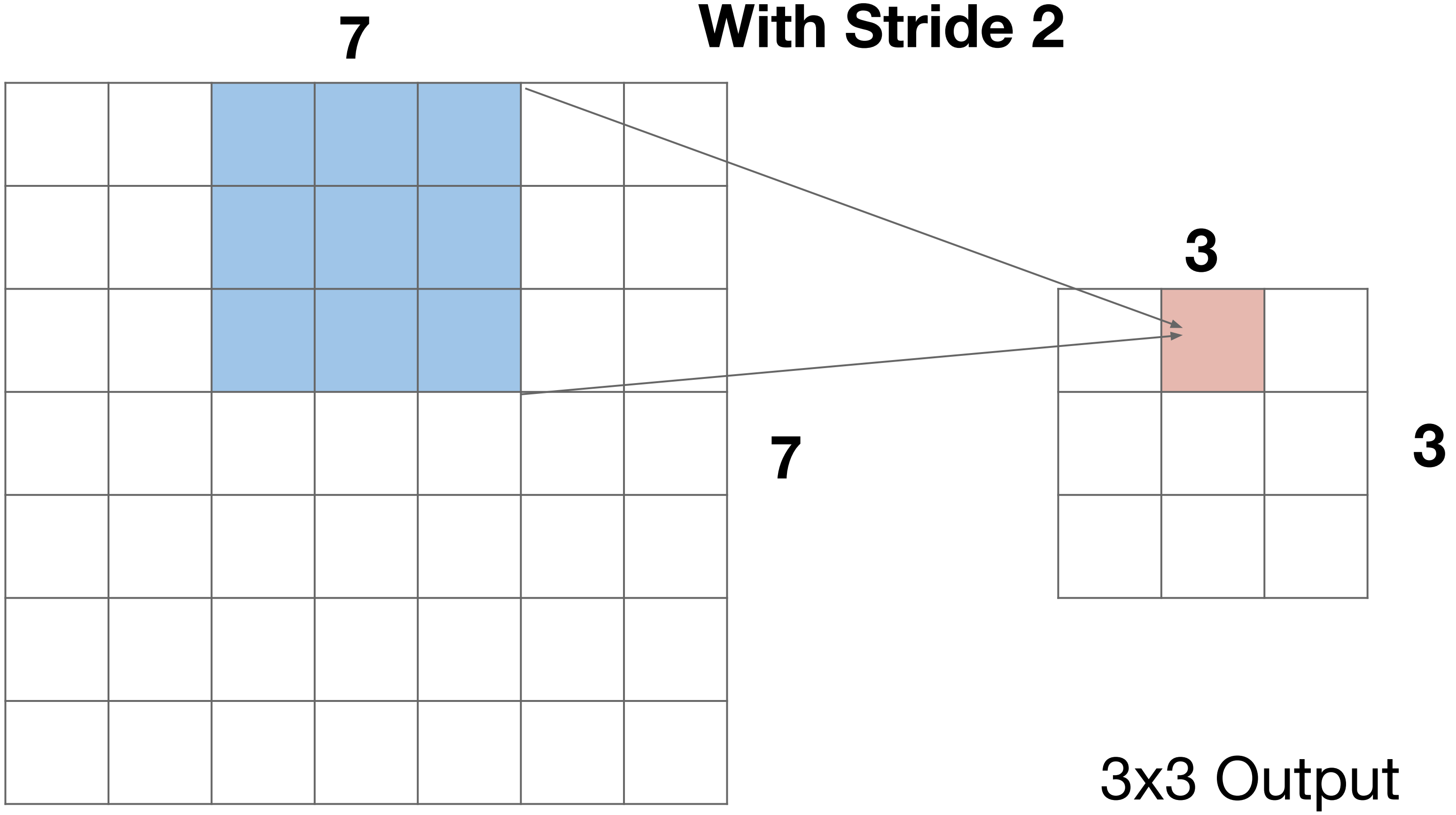


3x3 Output

7x7 input (spatially)
assume 3x3 filter

Closer look at spatial dimensions

With Stride 2

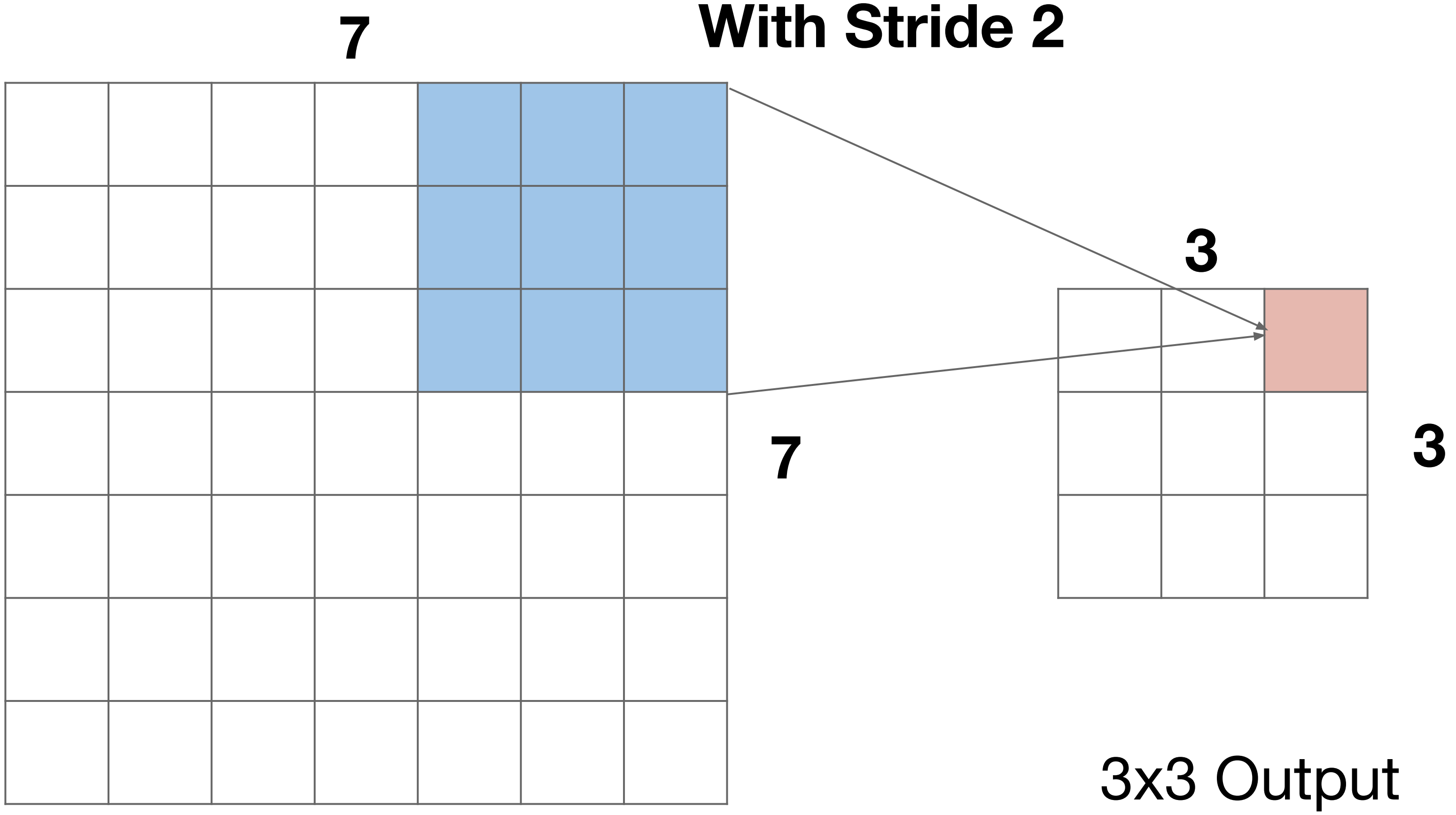


7x7 input (spatially)
assume 3x3 filter

3x3 Output

Closer look at spatial dimensions

With Stride 2

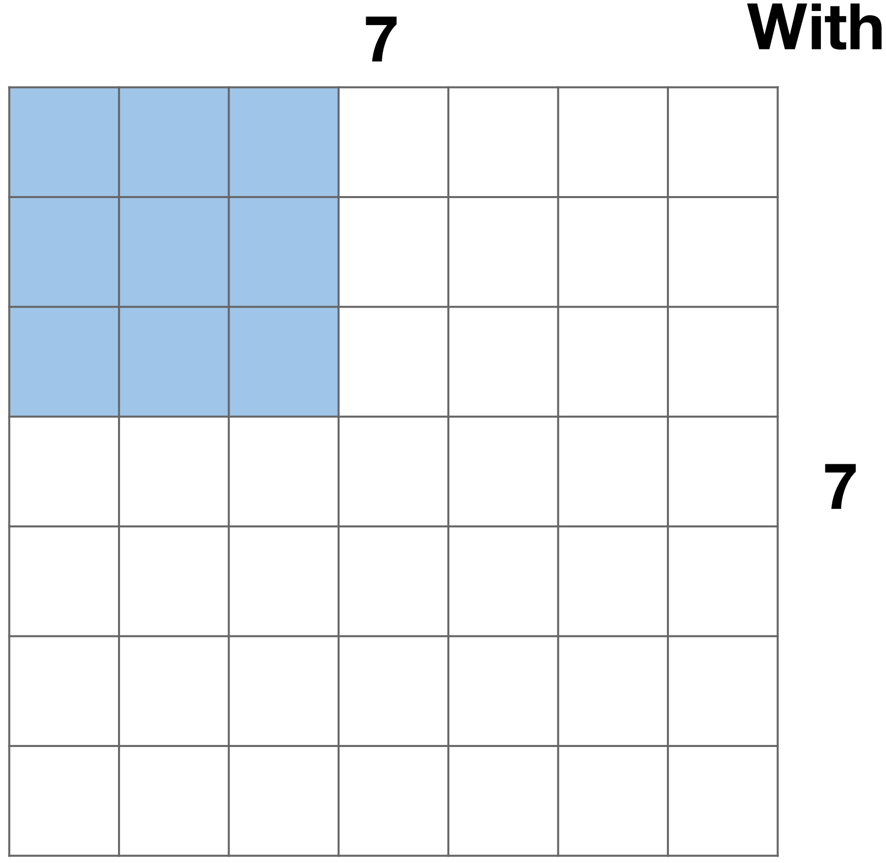


7x7 input (spatially)
assume 3x3 filter

3x3 Output

Closer look at spatial dimensions

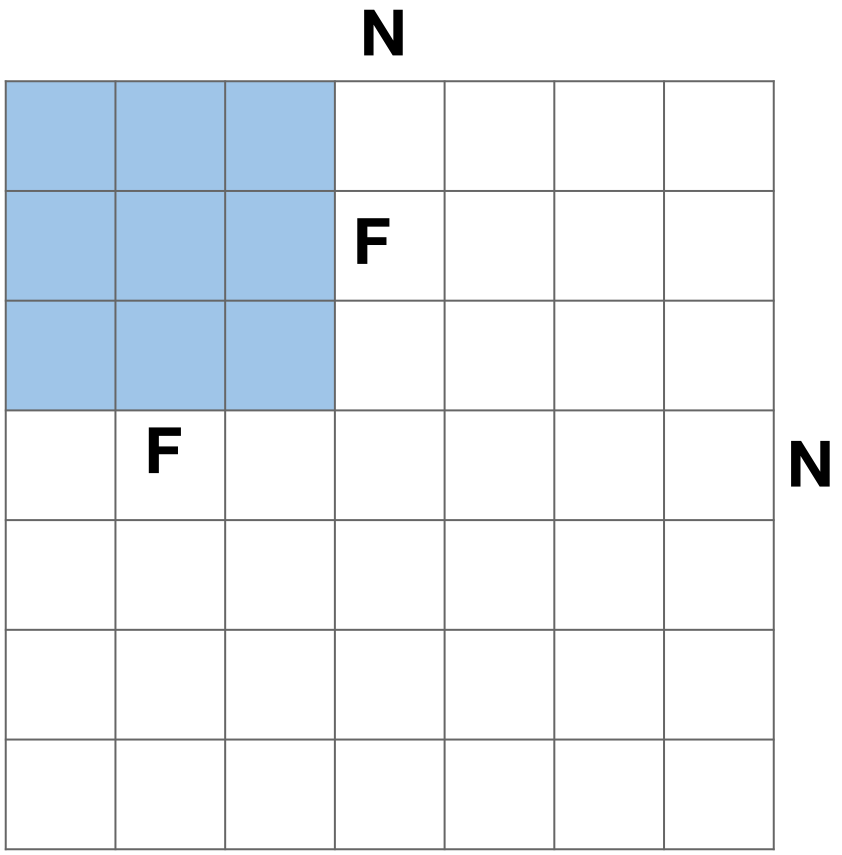
With Stride 3 ?



7x7 input (spatially)
assume 3x3 filter

doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

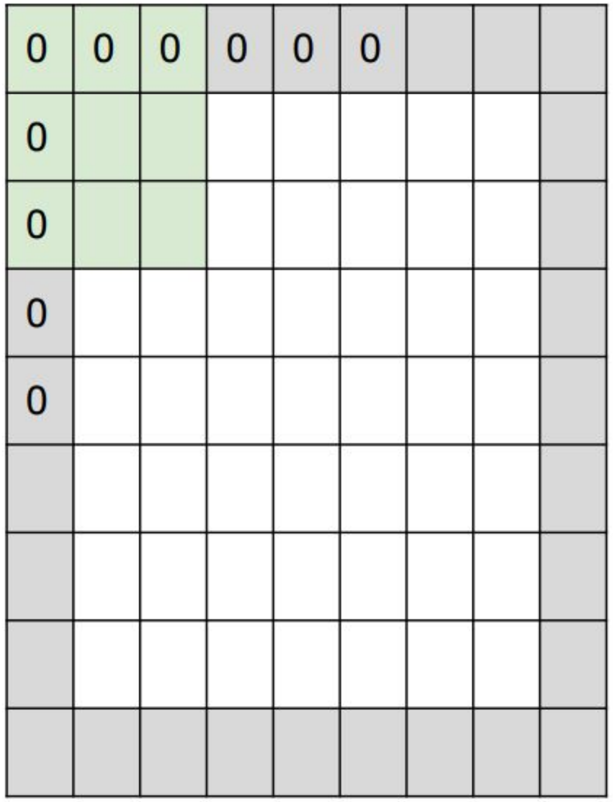
Output dimensions



Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

In Practice: Zero pad to preserve size

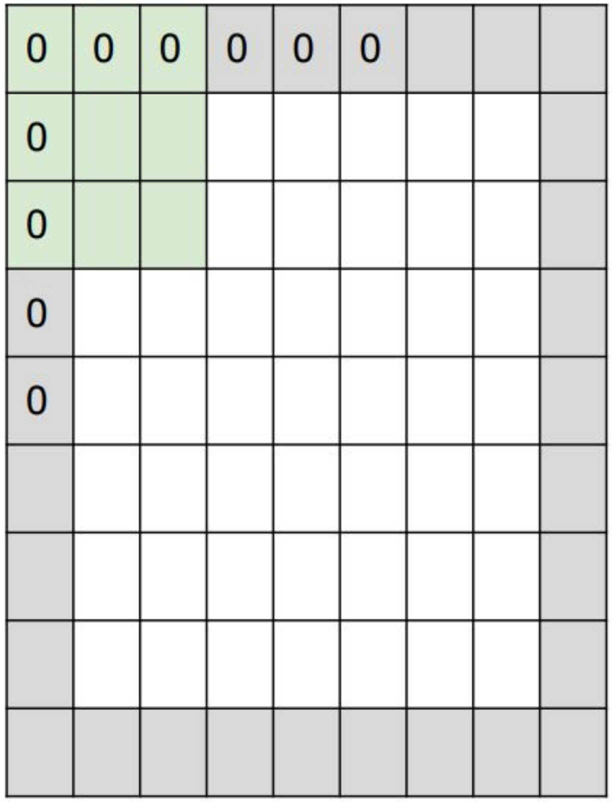


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

In Practice: Zero pad to preserve size

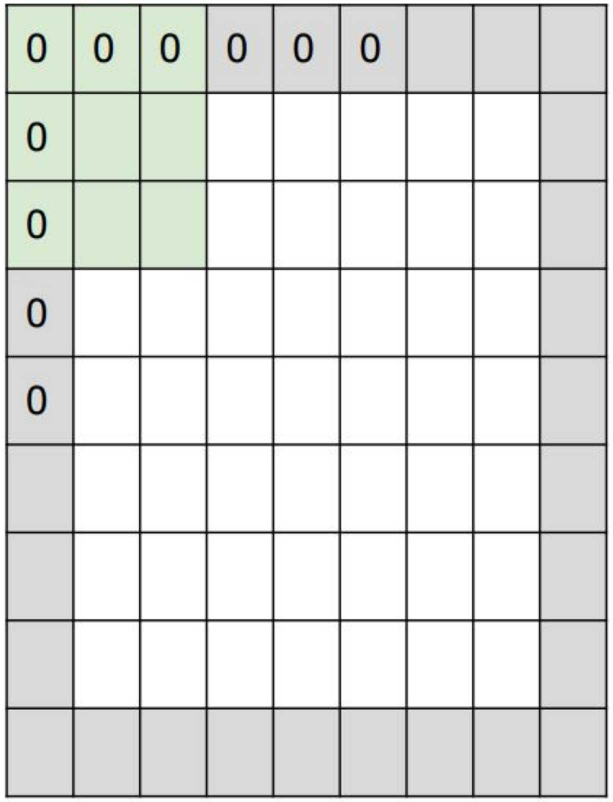


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

In Practice: Zero pad to preserve size



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)

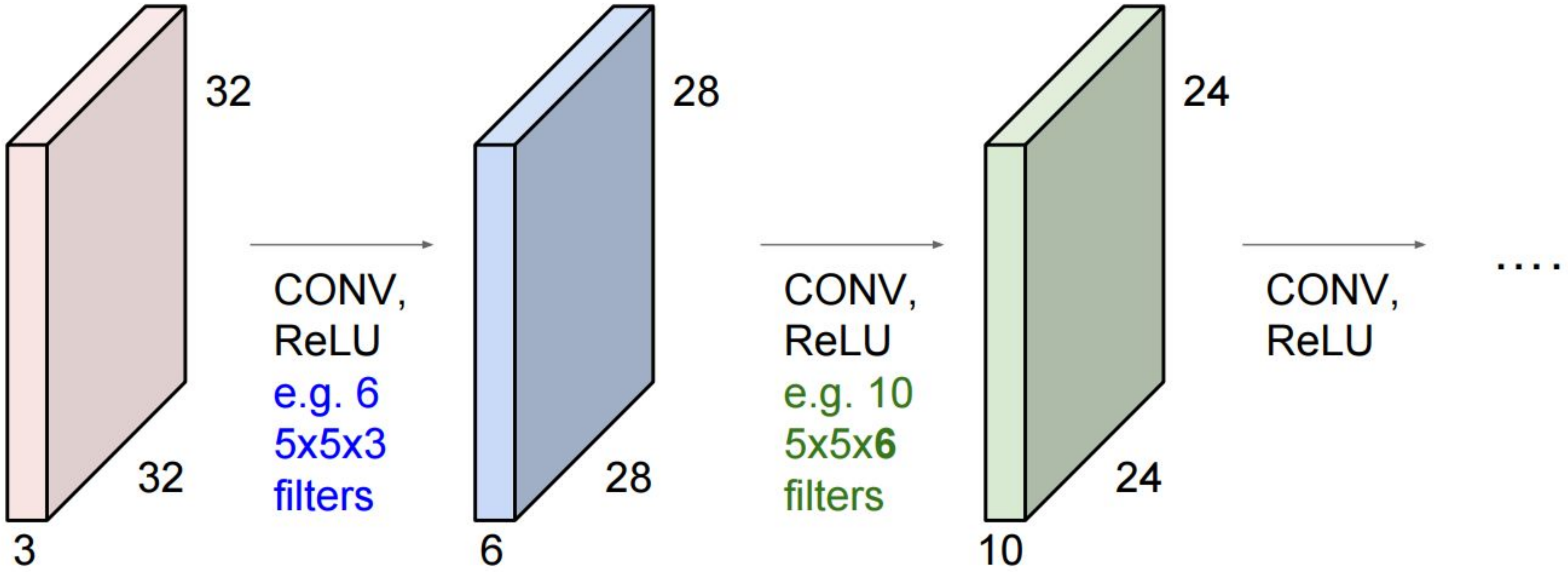
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Zero pad to preserve size

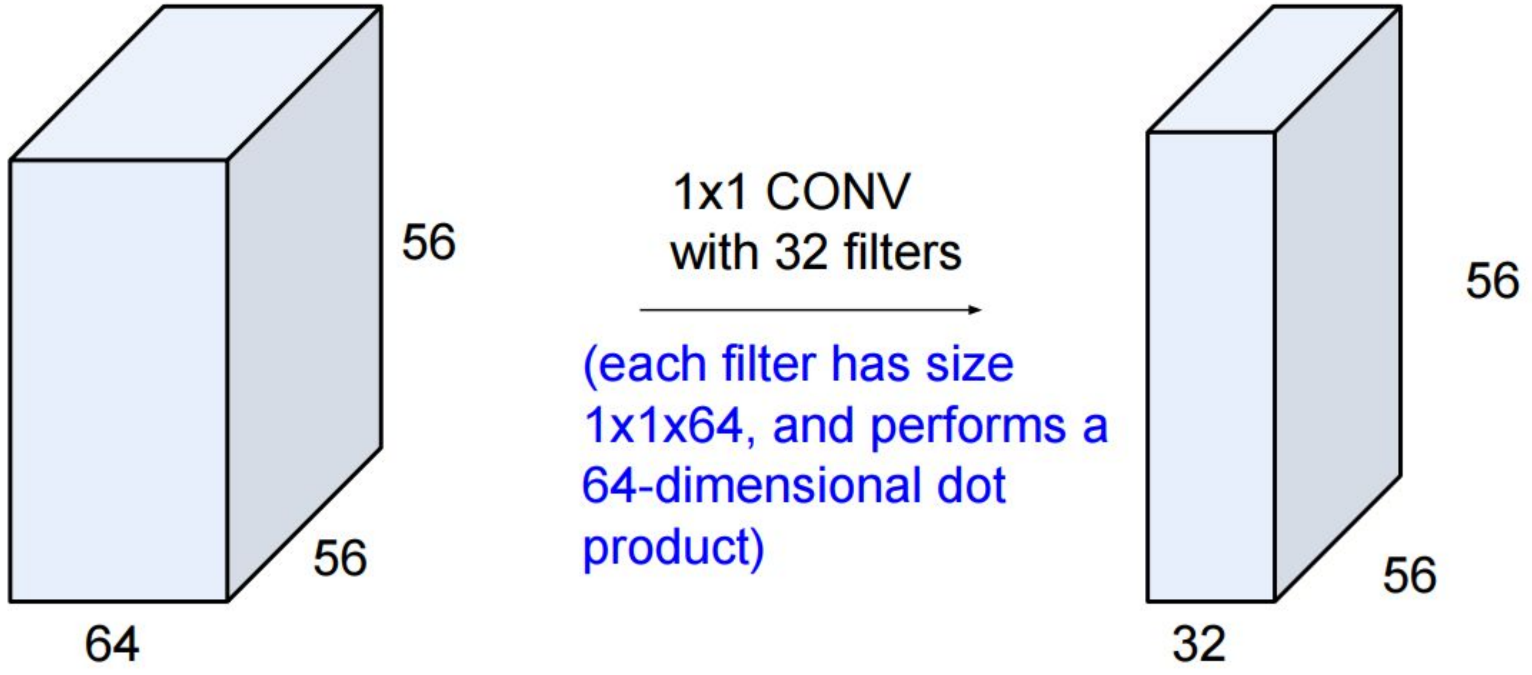
Remember back to…

E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn't work well.



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

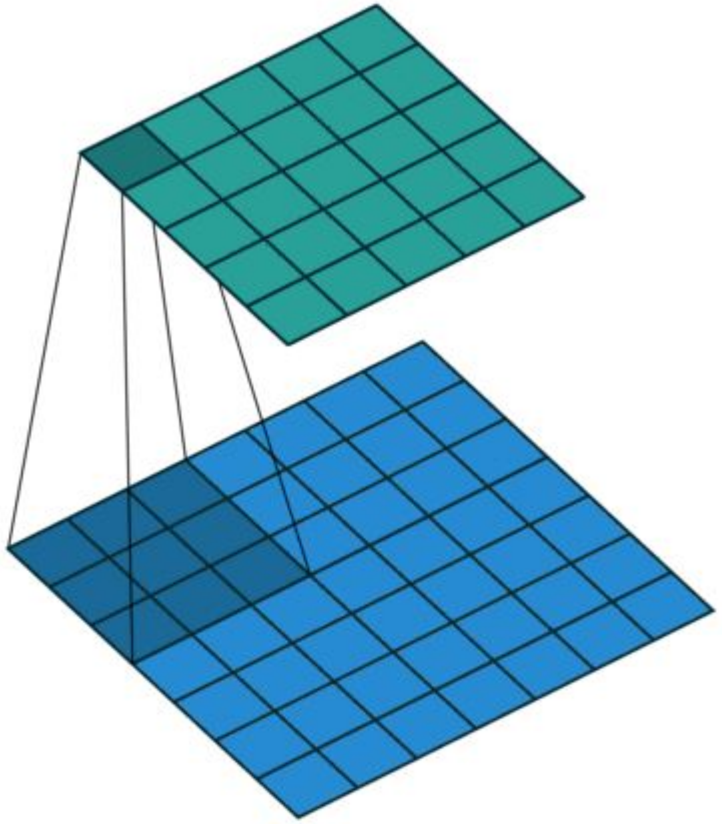
(btw, 1x1 convolution layers make perfect sense)

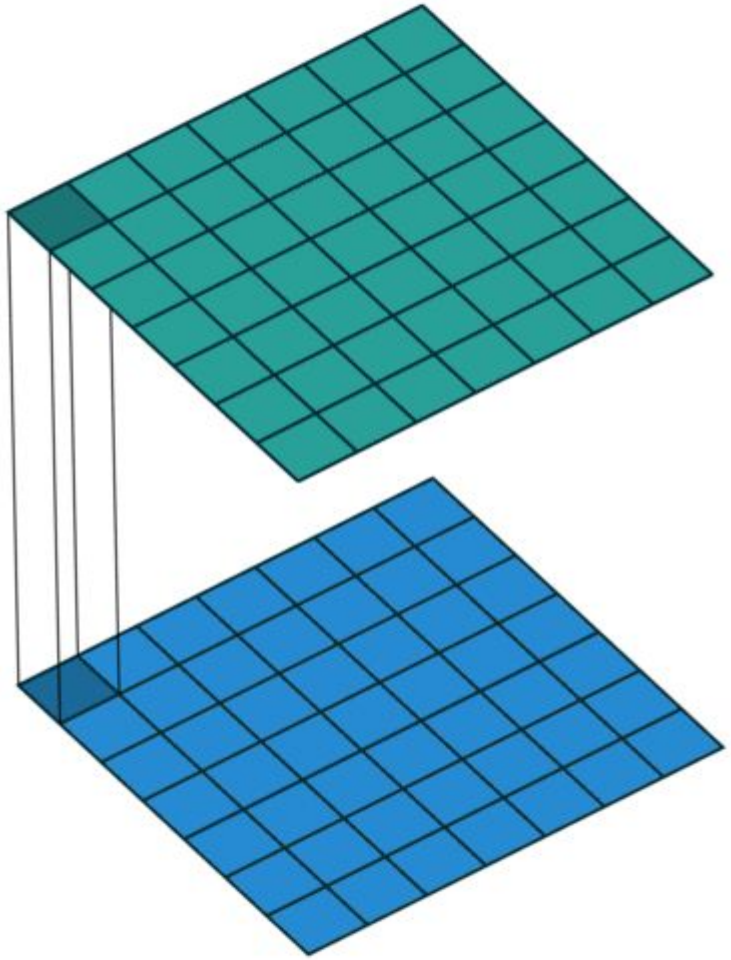


56
56
64
1x1 CONV
with 32 filters
(each filter has size 1x1x64, and performs a 64-dimensional dot product)
56
56
32

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

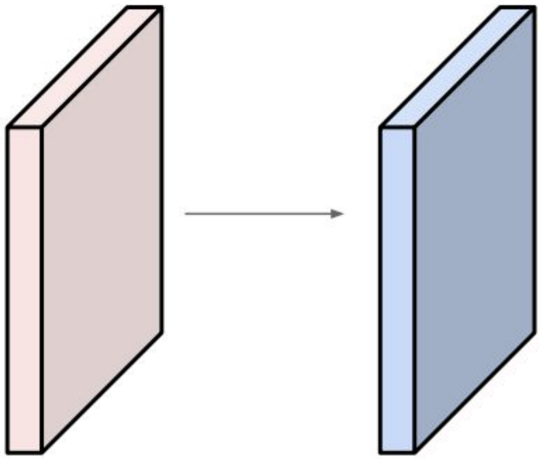
3x3 vs 1x1 convolutions





[1] Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning

Examples time



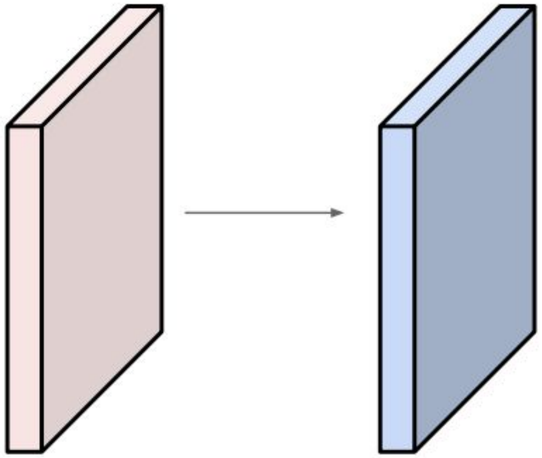
Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Output volume size: ?

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Examples time



Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

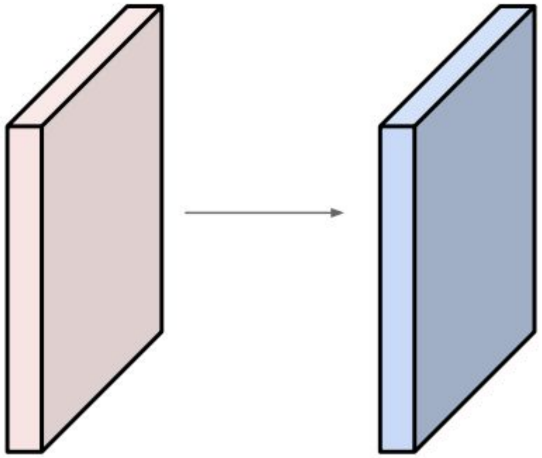
Output volume size:

(32+2*2-5)/1+1 = 32 spatially, so

32x32x10

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Examples time



Input volume: **32x32x3**

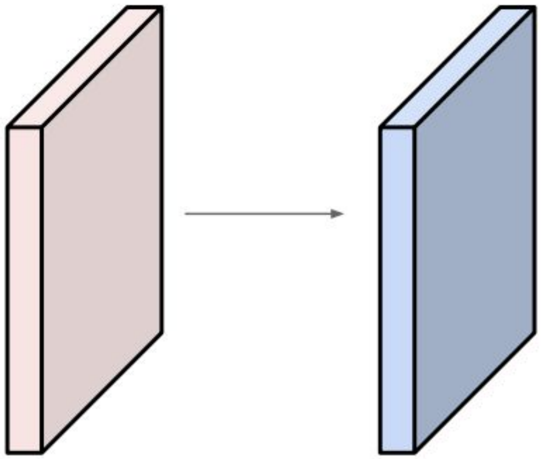
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Examples time

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

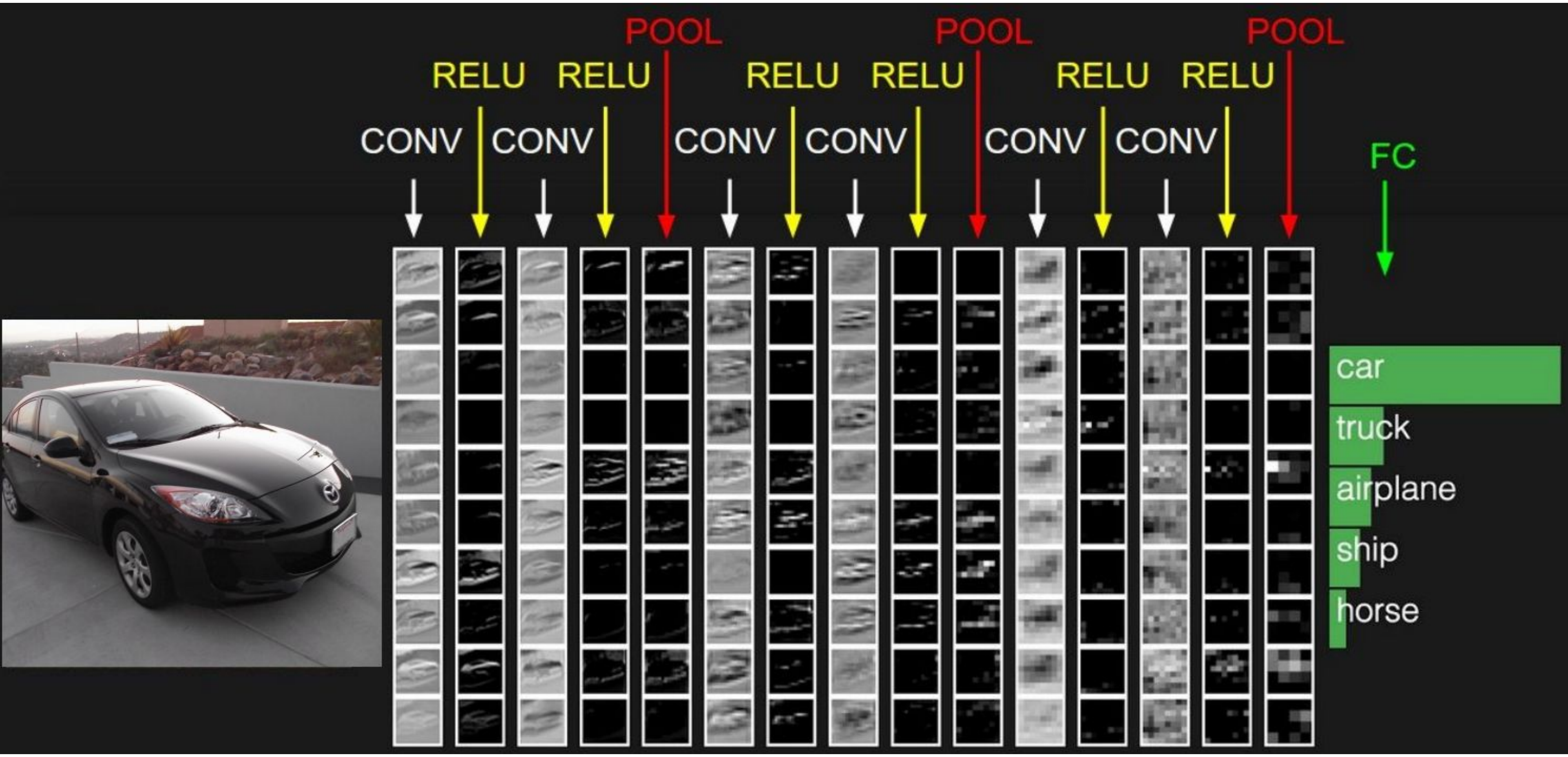


Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

FC layer of the same size = 32*32*3*10
= 30720 params

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

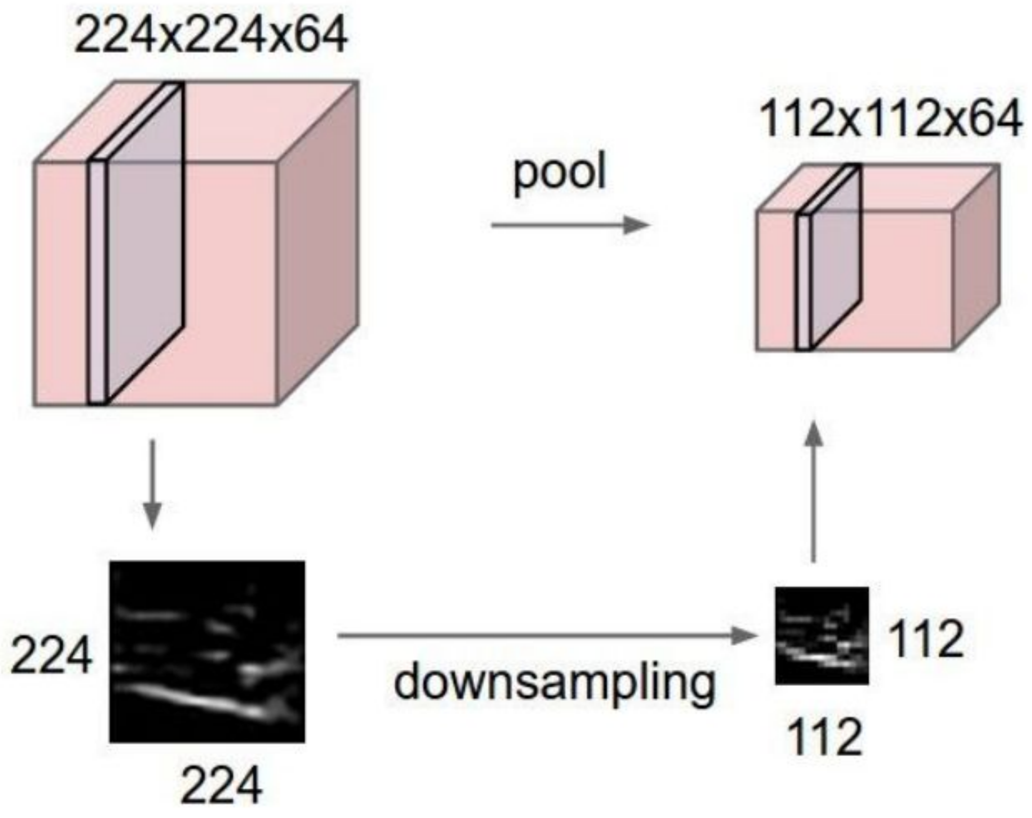
Pooling Layer



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

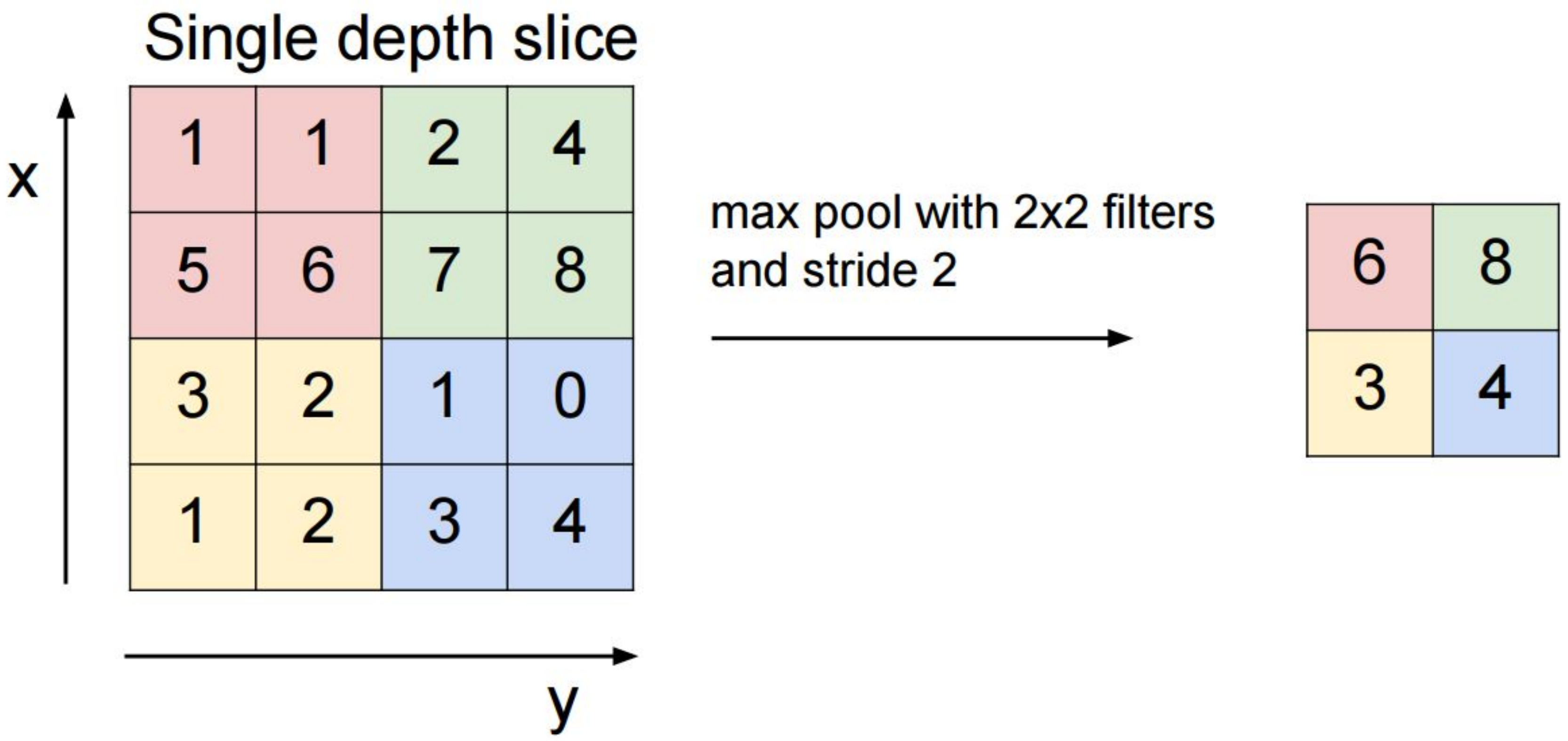
Pooling Layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Max Pooling



Zero parameters: output is a fixed function of input

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Large Convnets for Image classification

- Convnets stack layers of convolution, non-linearity, pooling layers
- Trend towards smaller filters and deeper architectures
  - Smaller filters tend to be easier to learn than large ones

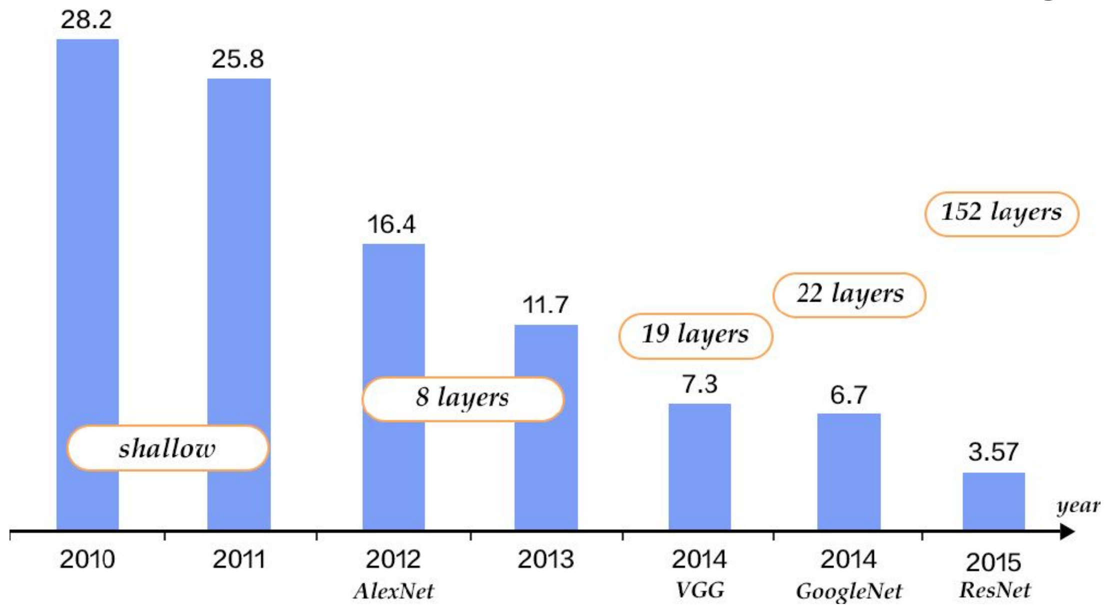


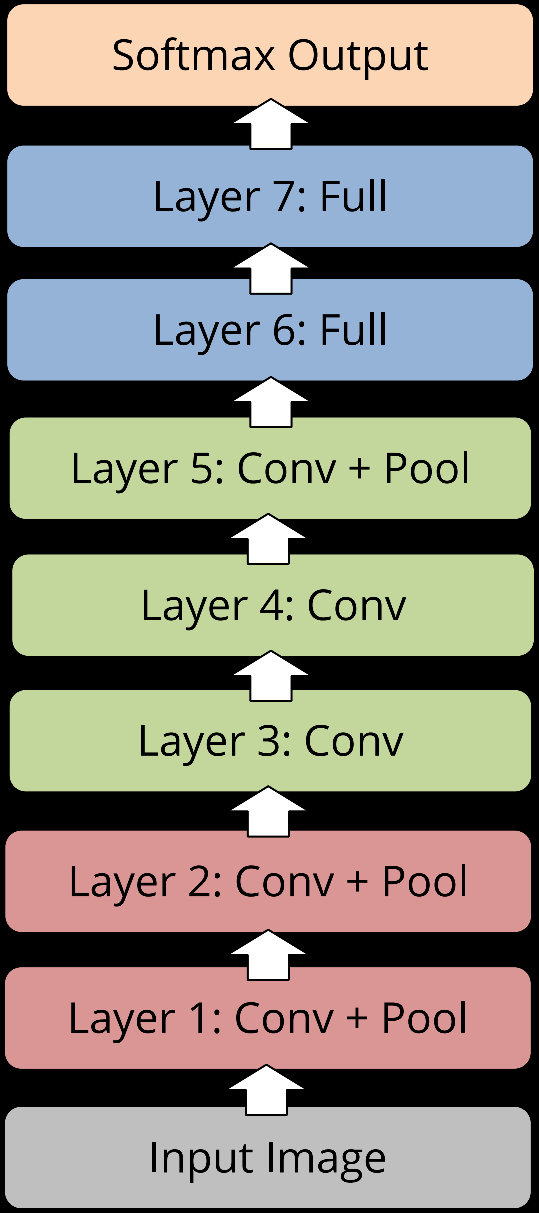
Figure Credit: Felsberg, Michael. “Five years after the Deep Learning revolution of computer vision : State of the art methods for online image and video analysis.” (2017).

ConvNet Architecture

Importance of Depth

Architecture of Krizhevsky et al.

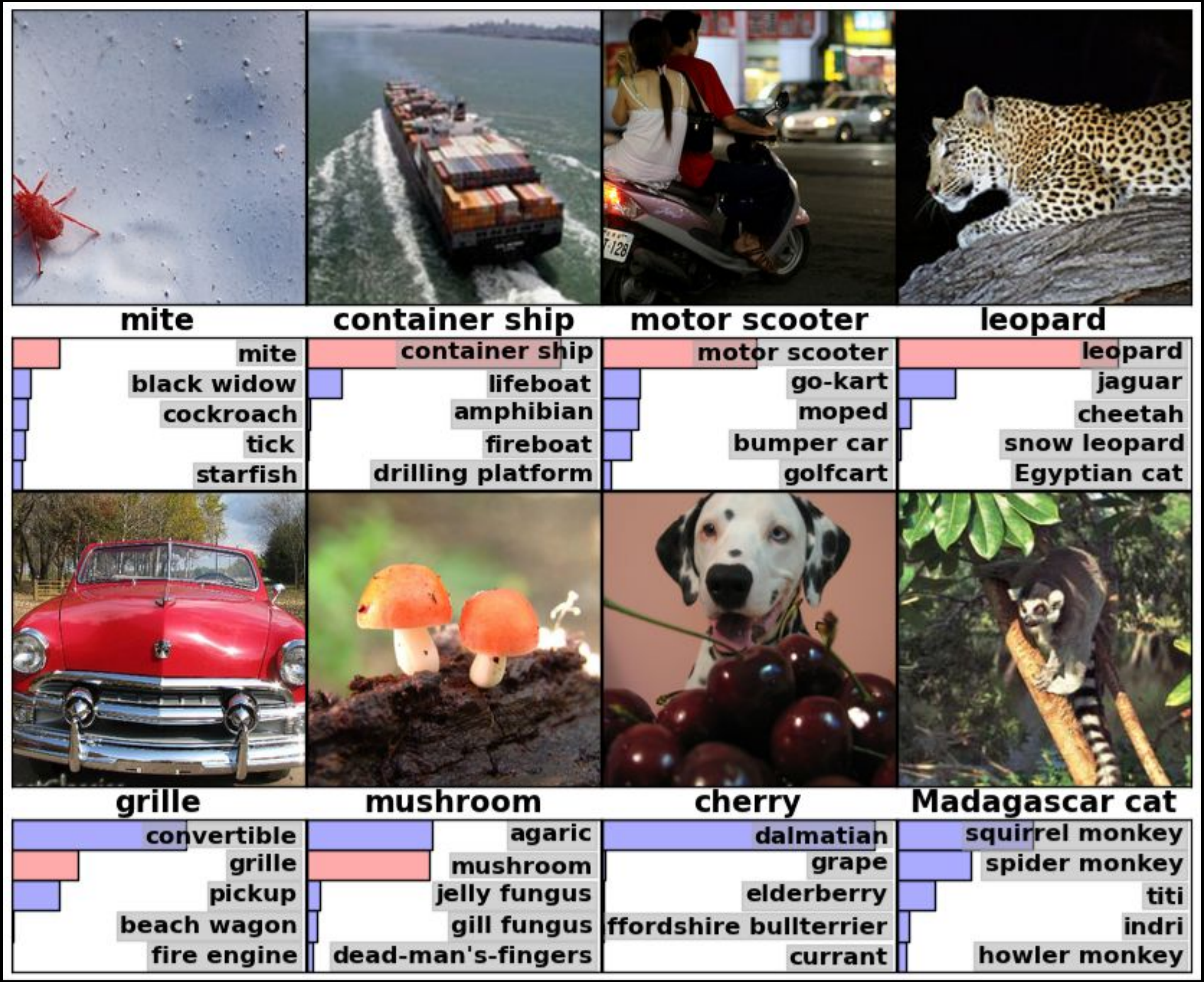
- 8 layers total
- Trained on Imagenet dataset [Deng et al. CVPR'09]
- 18.2% top-5 error
- Our reimplementation:
  18.1% top-5 error



slide credit: Rob Fergus

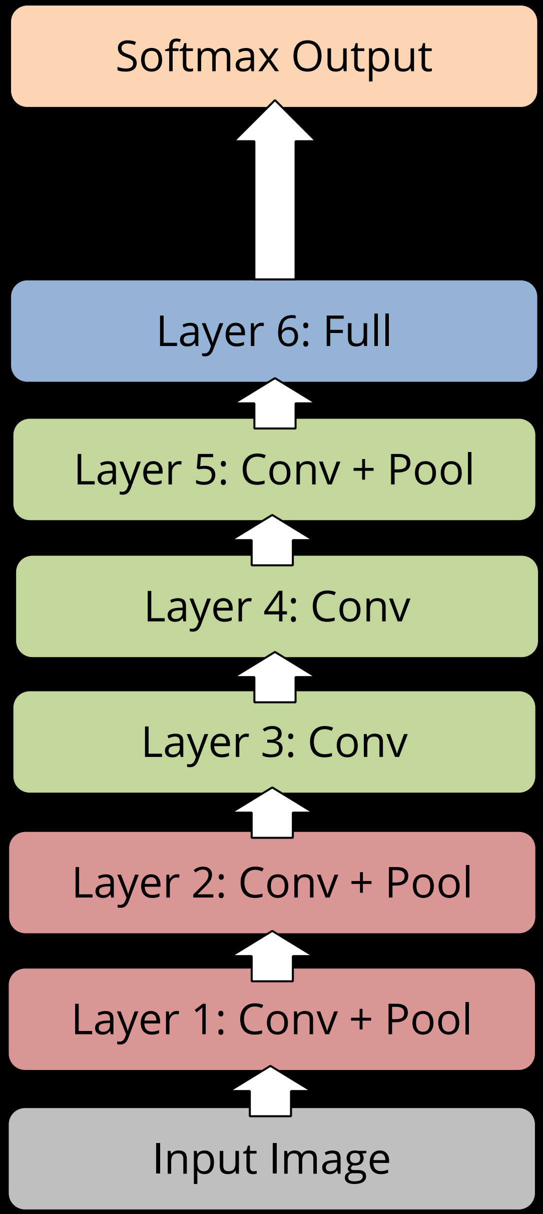
Sample Classification Results

[Krizhevsky et al. NIPS'12]



Architecture of Krizhevsky et al.

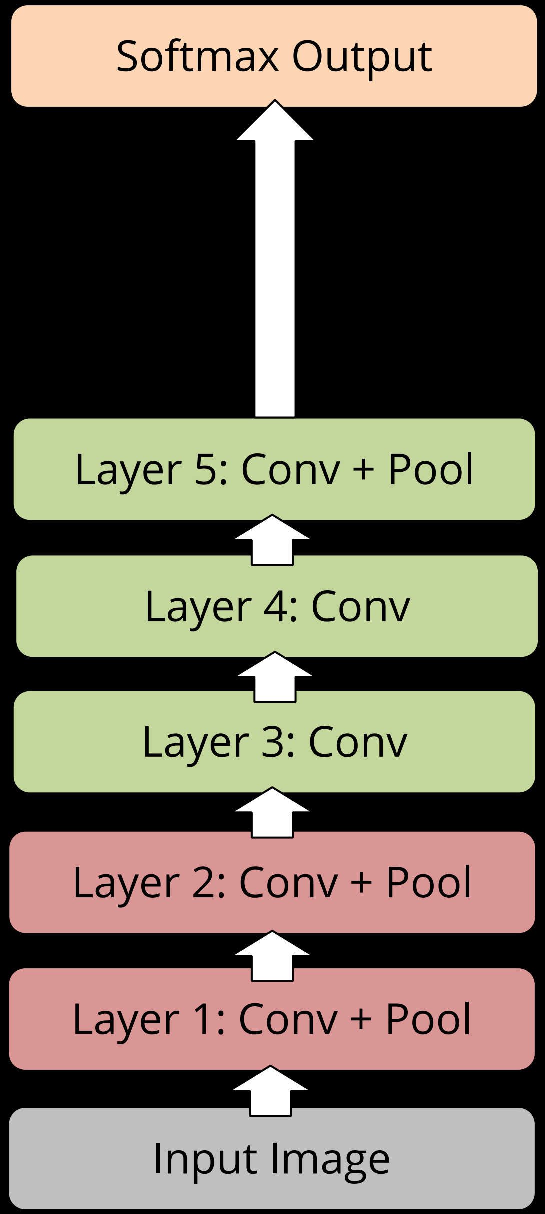
- Remove top fully connected layer
  - Layer 7
- Drop 16 million parameters
- Only 1.1% drop in performance!



slide credit: Rob Fergus

Architecture of Krizhevsky et al.

- Remove both fully connected layers
  - Layer 6 & 7
- Drop ~50 million parameters
- 5.7% drop in performance



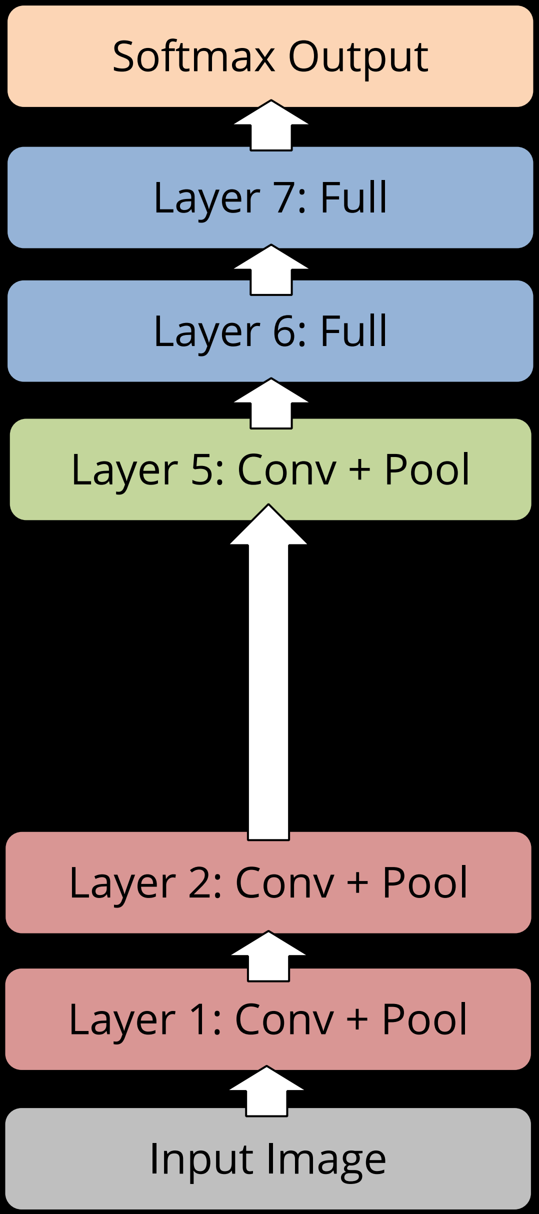
slide credit: Rob Fergus

Architecture of Krizhevsky et al.

- Now try removing upper feature extractor layers:
  - Layers 3 & 4

- Drop ~1 million parameters

- 3.0% drop in performance

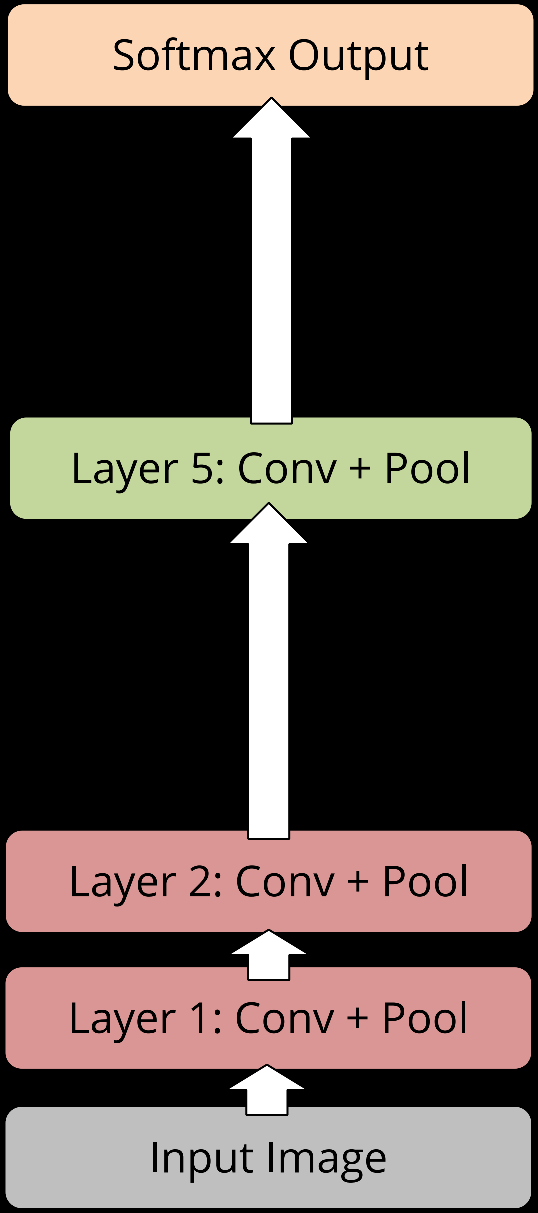


slide credit: Rob Fergus

Architecture of Krizhevsky et al.

- Now try removing upper feature extractor layers & fully connected:
  - Layers 3, 4, 6 ,7
- Now only 4 layers
- 33.5% drop in performance

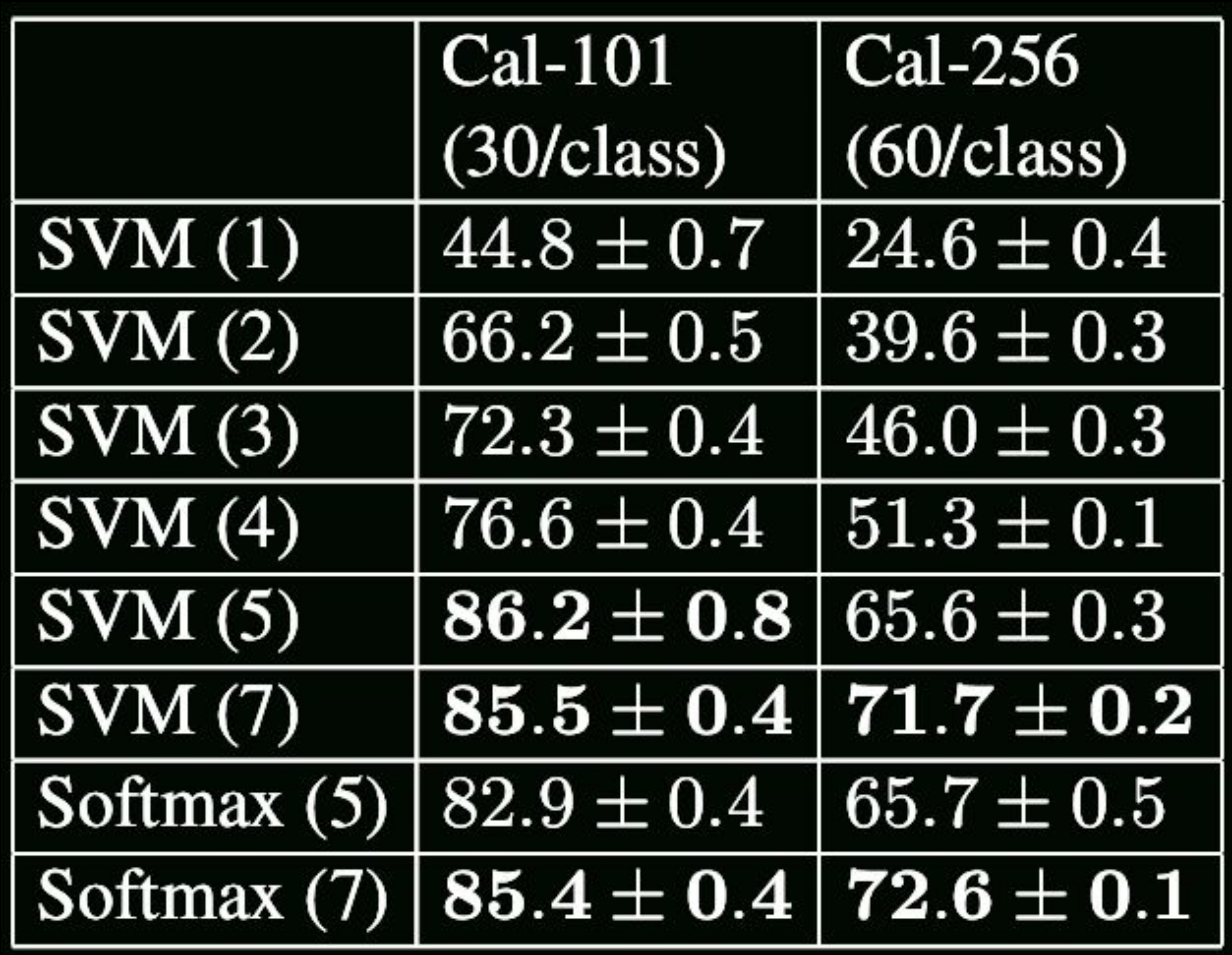
→ Depth of network is key



slide credit: Rob Fergus

Tapping off Features at each Layer

Plug features from each layer into linear SVM or soft-max



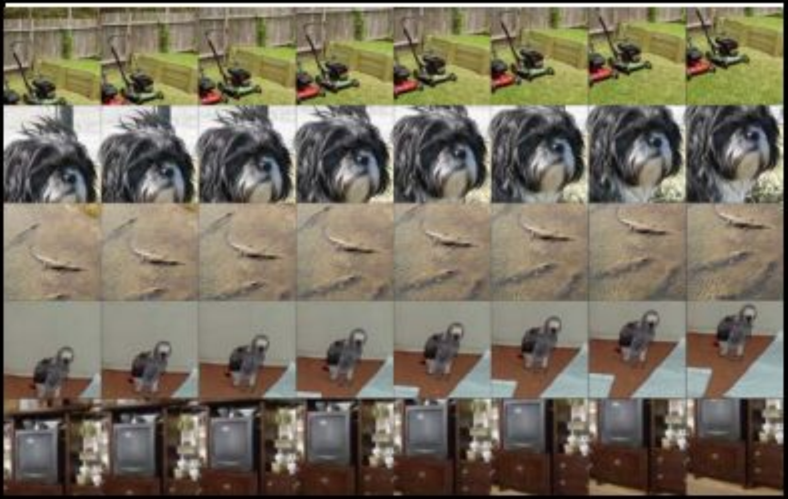
	Cal-101 (30/class)	Cal-256 (60/class)
SVM (1)	44.8 ± 0.7	24.6 ± 0.4
SVM (2)	66.2 ± 0.5	39.6 ± 0.3
SVM (3)	72.3 ± 0.4	46.0 ± 0.3
SVM (4)	76.6 ± 0.4	51.3 ± 0.1
SVM (5)	**86.2 ± 0.8**	65.6 ± 0.3
SVM (7)	**85.5 ± 0.4**	**71.7 ± 0.2**
Softmax (5)	82.9 ± 0.4	65.7 ± 0.5
Softmax (7)	**85.4 ± 0.4**	**72.6 ± 0.1**

slide credit: Rob Fergus

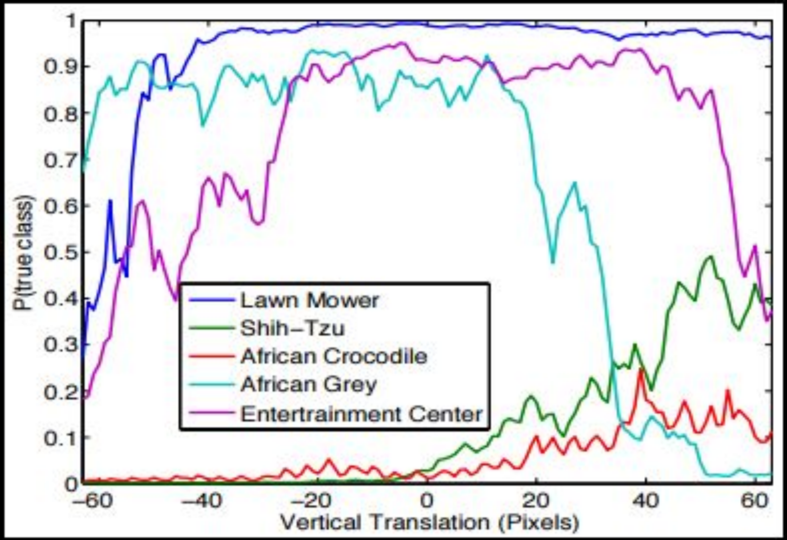
ConvNet Architecture

Invariance Properties

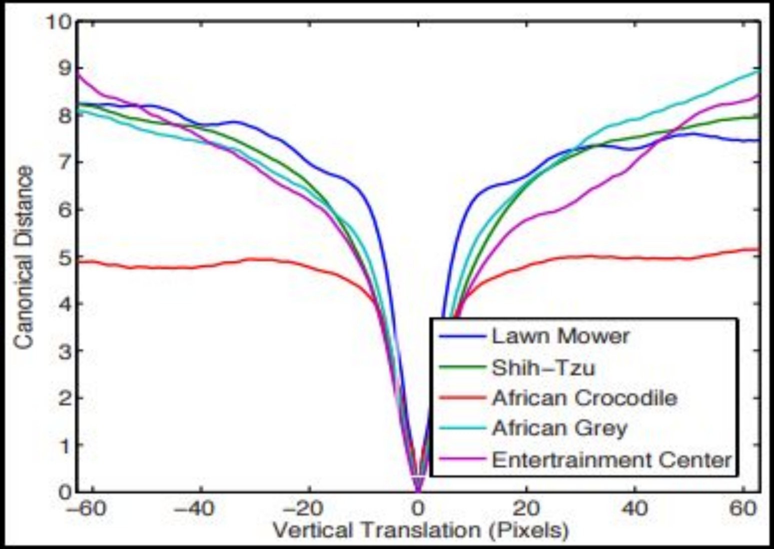
Translation (Vertical)



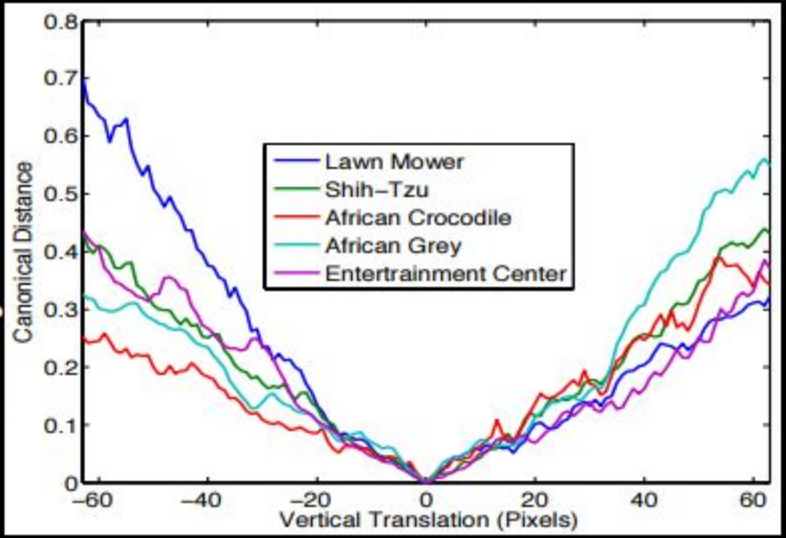
Output



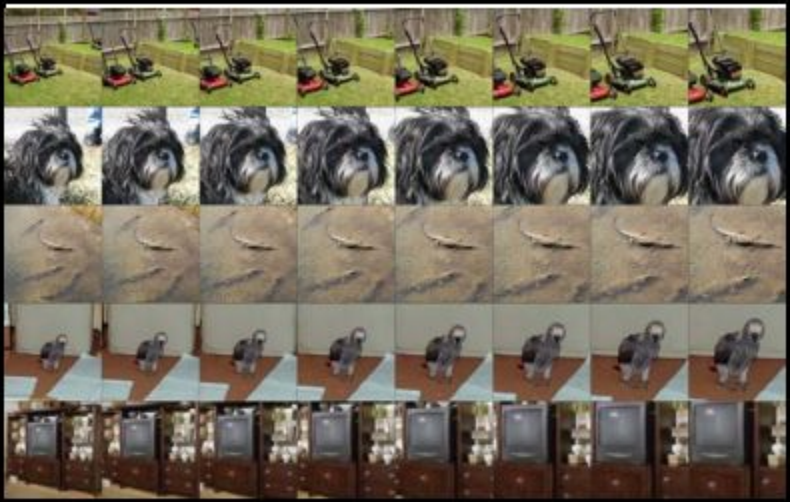
Layer 1



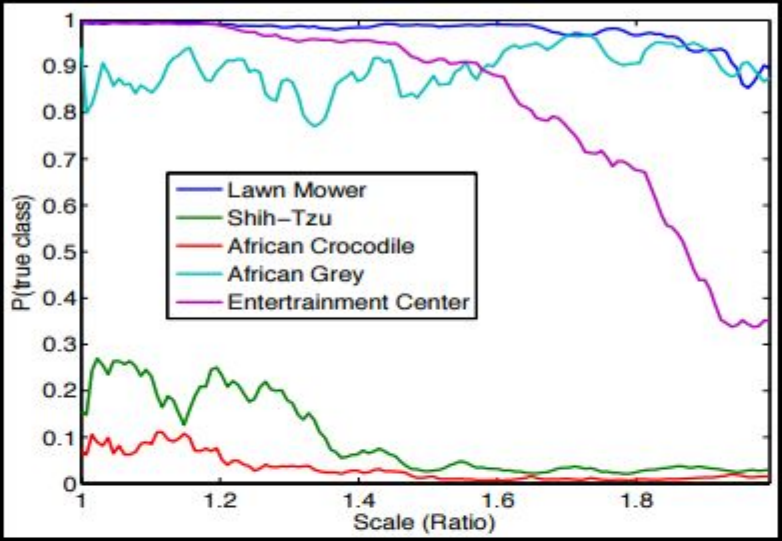
Layer 7



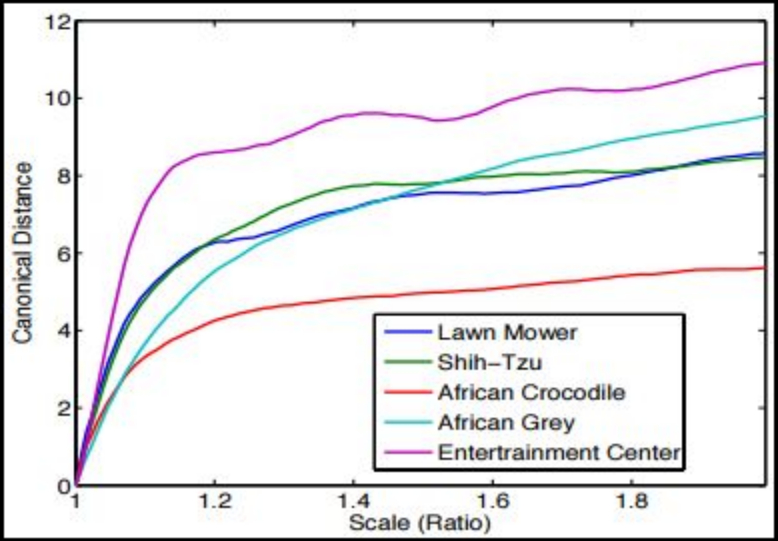
Scale Invariance



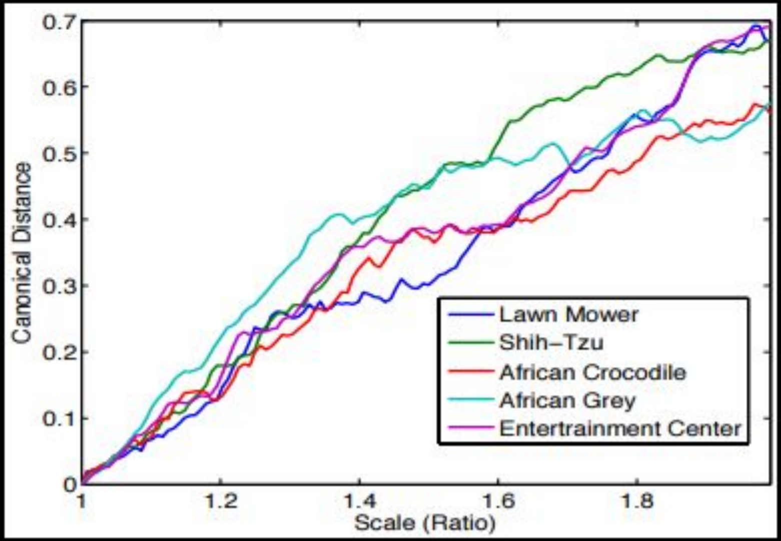
Output



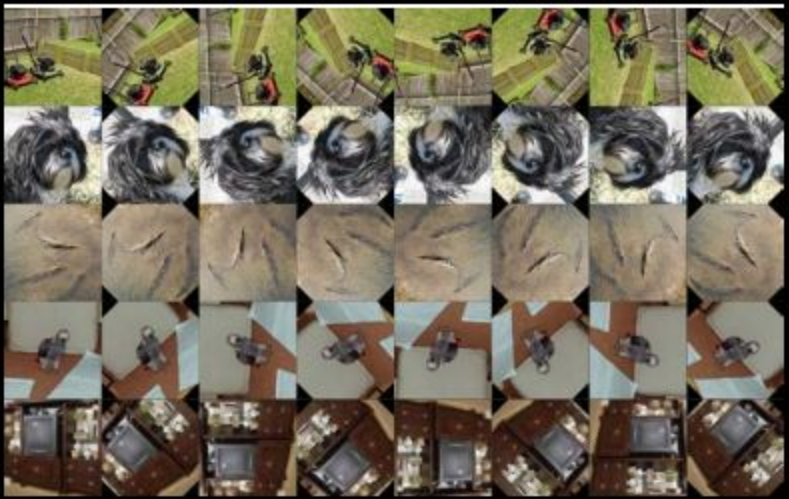
Layer 1

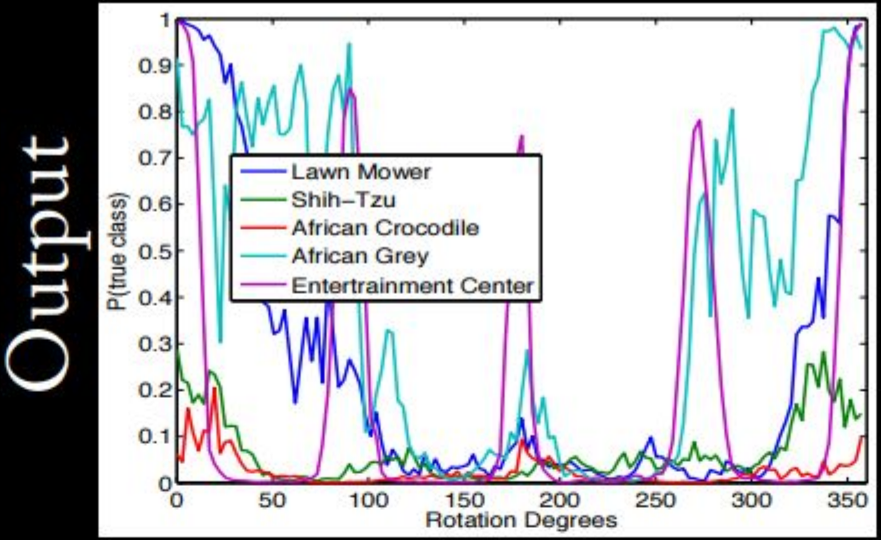


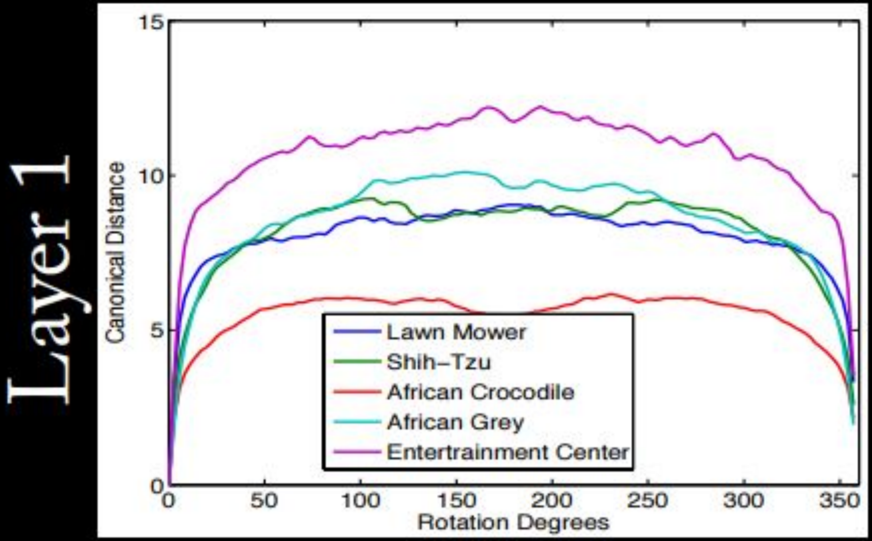
Layer 7

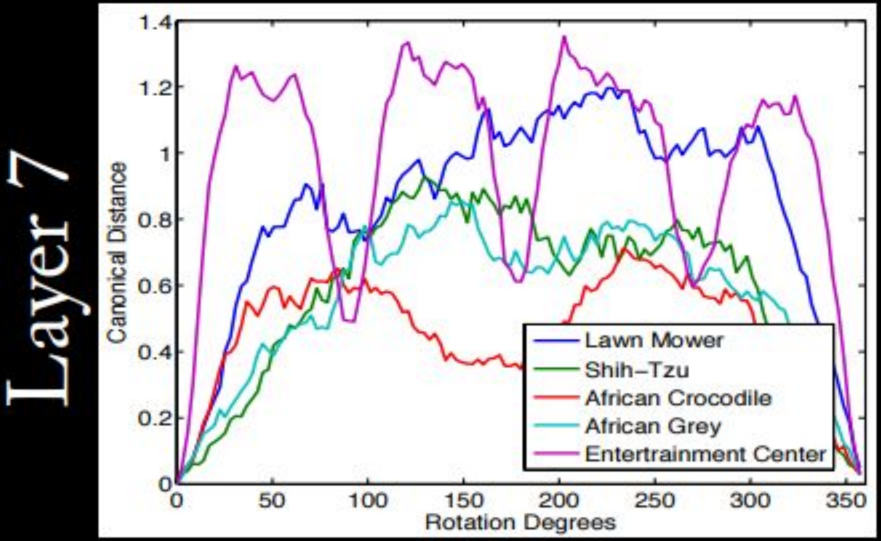


Rotation Invariance









Overview Today

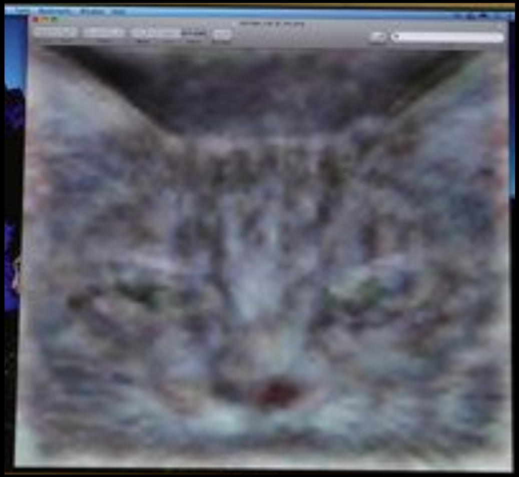
- Deep dive into convolutional networks
- **Visualizing convolutional networks**
- Feature Generalization
    - “pre-training” on large dataset,
      “fine-tuning” on target dataset

Visualizing ConvNets

slide credit: Rob Fergus

Visualizing Convnets

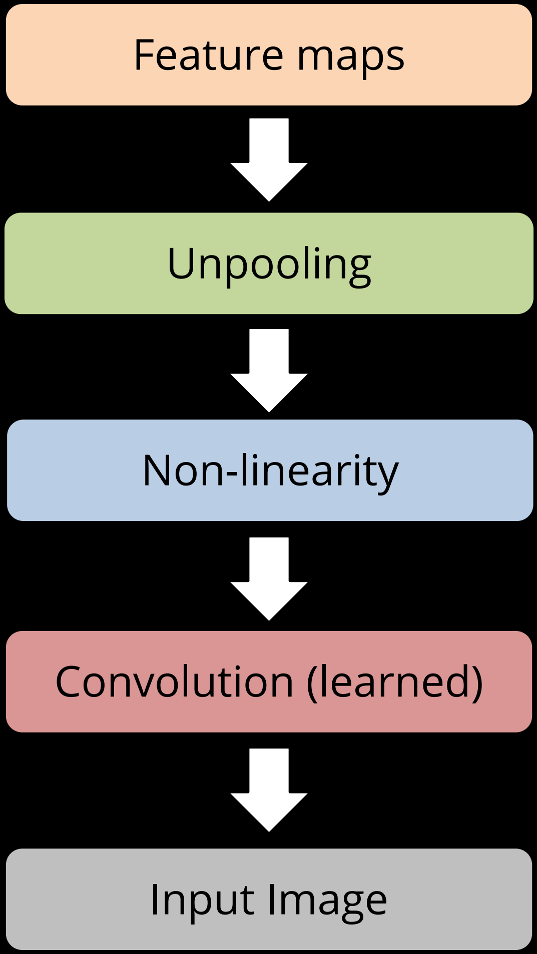
- Raw coefficients of learned filters in higher layers difficult to interpret
- Several approaches look to optimize input to maximize activity in a high-level feature
  - Erhan et al. [Tech Report 2009]
  - Le et al. [NIPS 2010]
  - Depend on initialization
  - Model invariance with Hessian about (locally) optimal stimulus



Visualization using Deconvolutional Networks

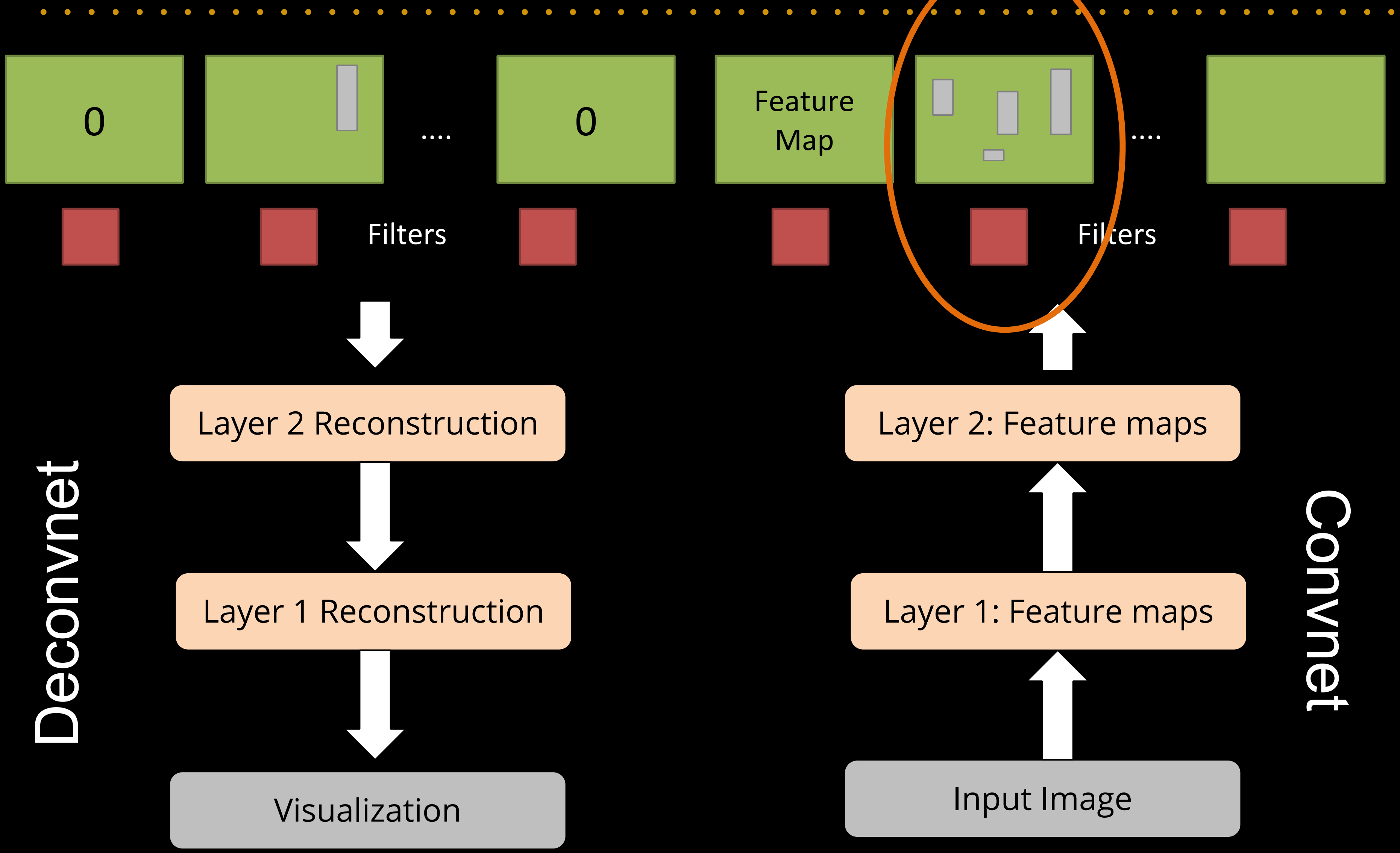
[Zeiler et al. CVPR’10, ICCV’11, arXiv’13]

- Provide way to map activations at high layers back to the input
- Same operations as Convnet, but in reverse:
  - Unpool feature maps
  - Convolve unpooled maps
    - Filters copied from Convnet
- Used here purely as a probe
  - Originally proposed as unsupervised learning method
  - No inference, no learning



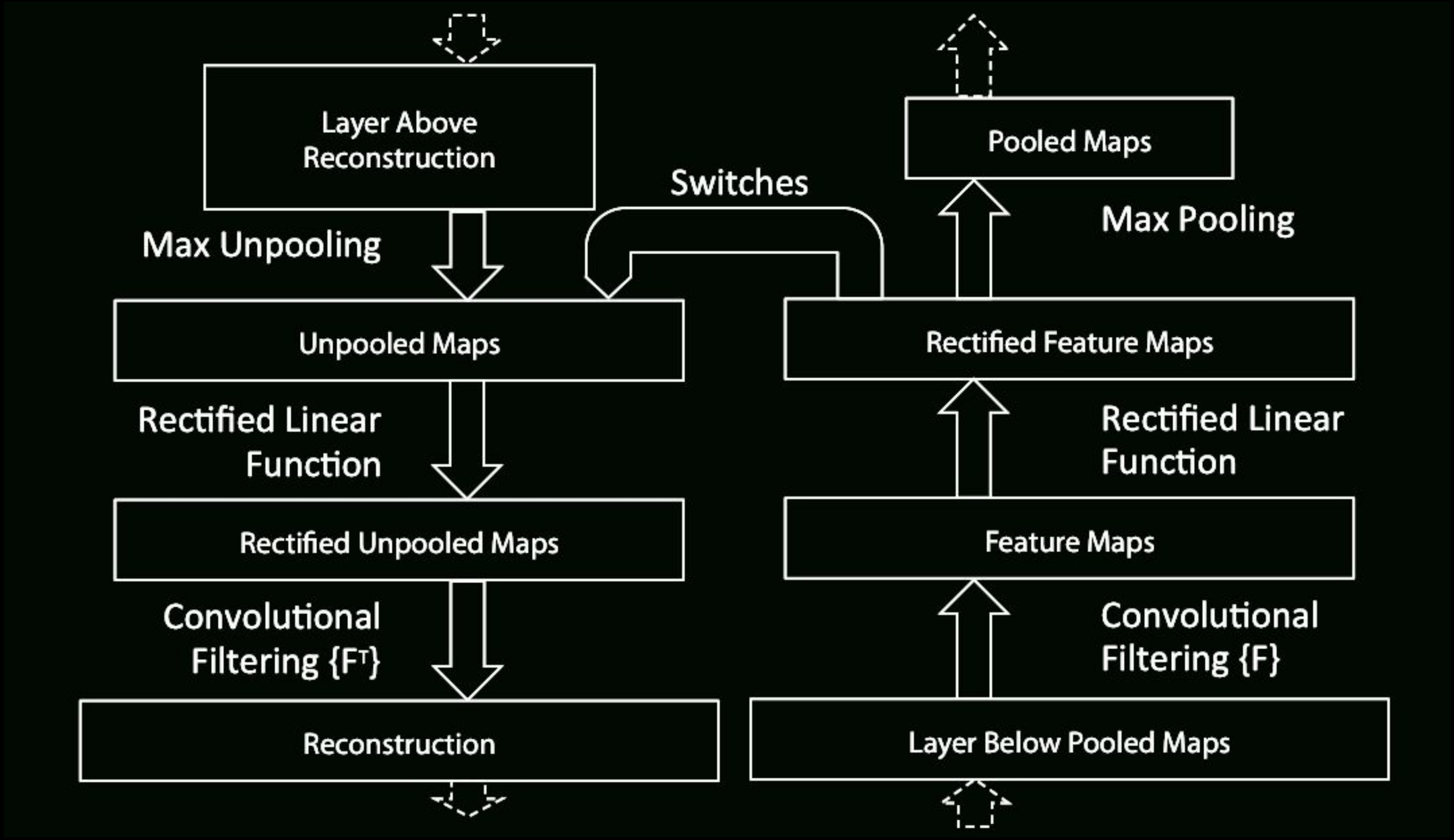
Deconvnet Projection from Higher Layers

[Zeiler and Fergus. arXiv'13]

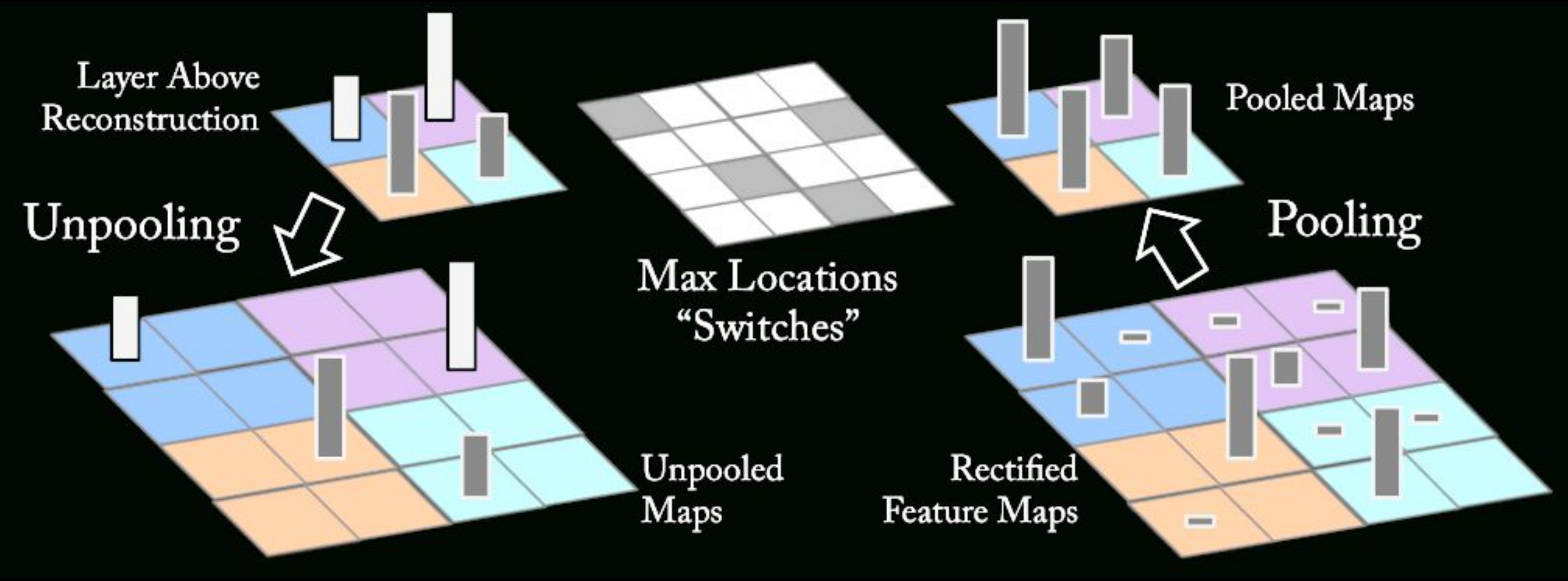


Details of Operation

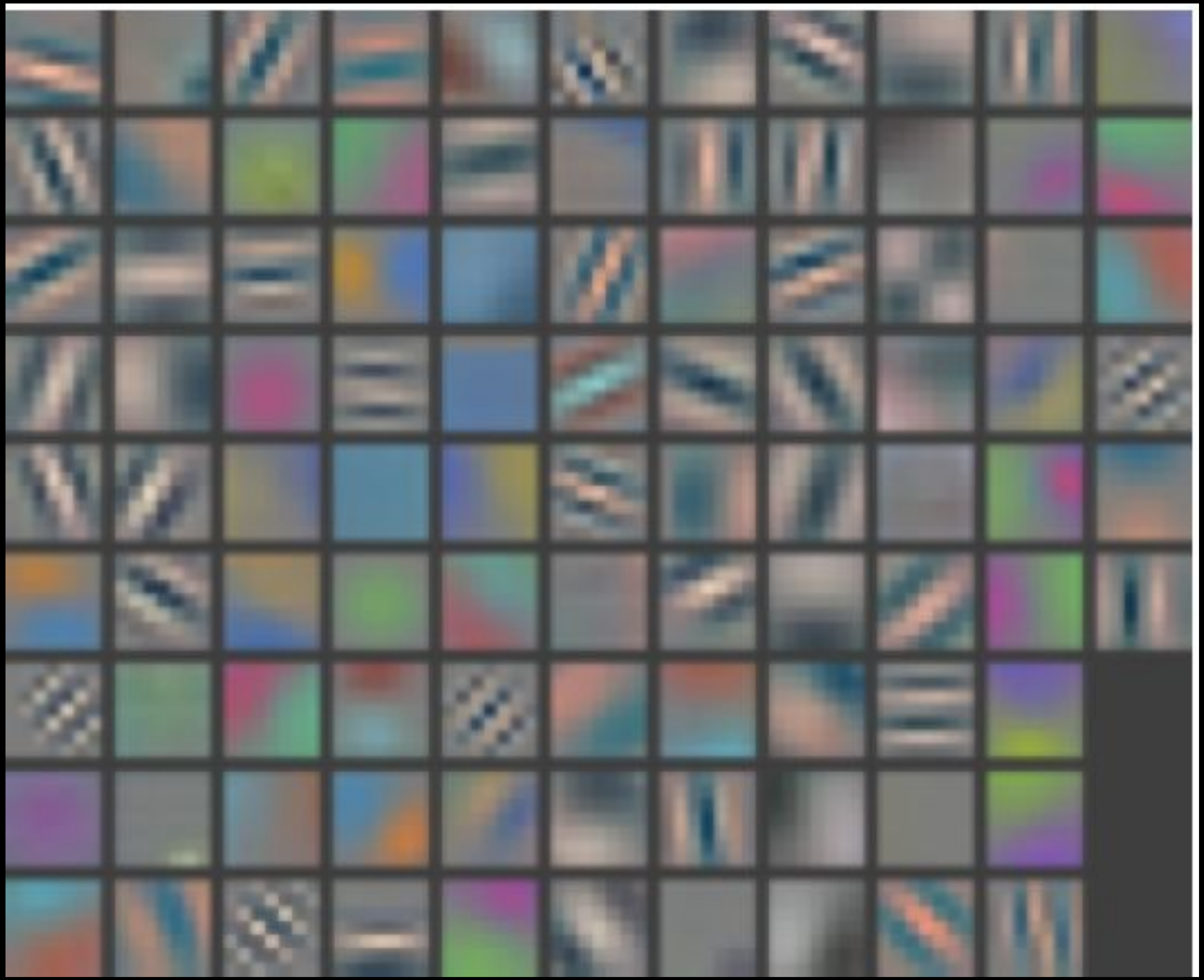
Deconvnet layer   Convnet layer



Unpooling Operation



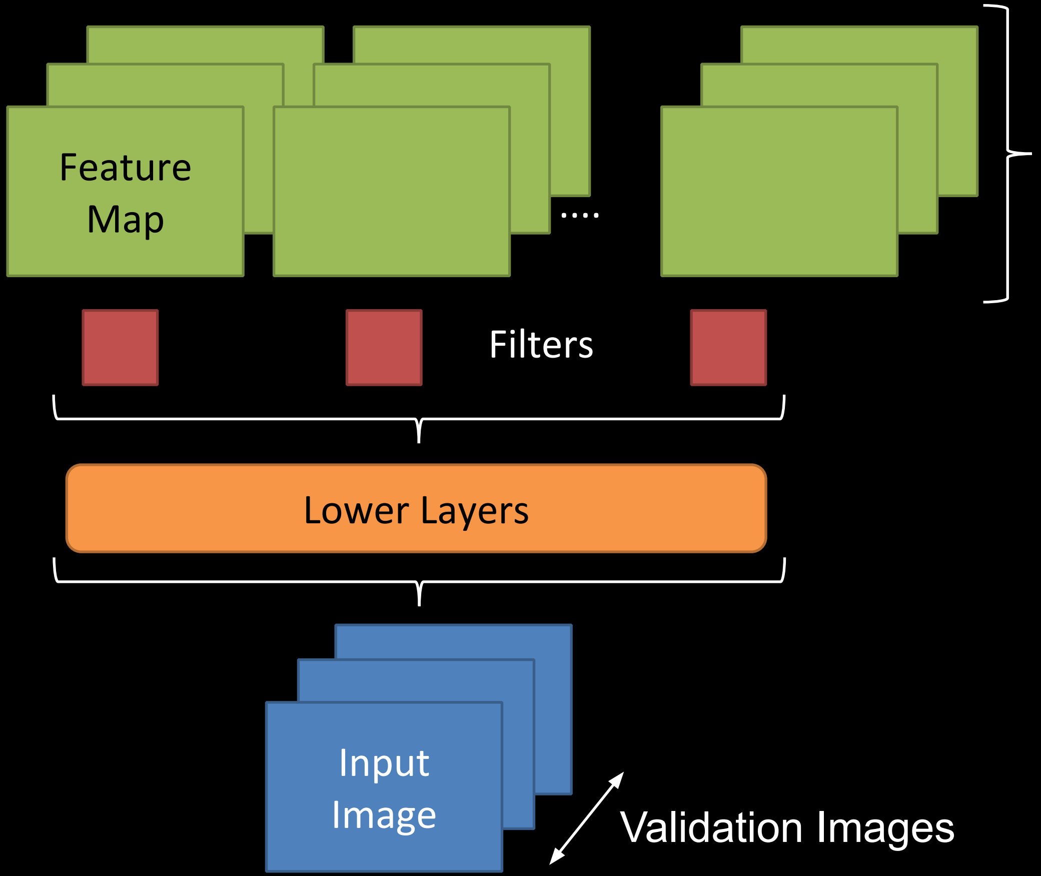
Layer 1 Filters



Visualizations of Higher Layers

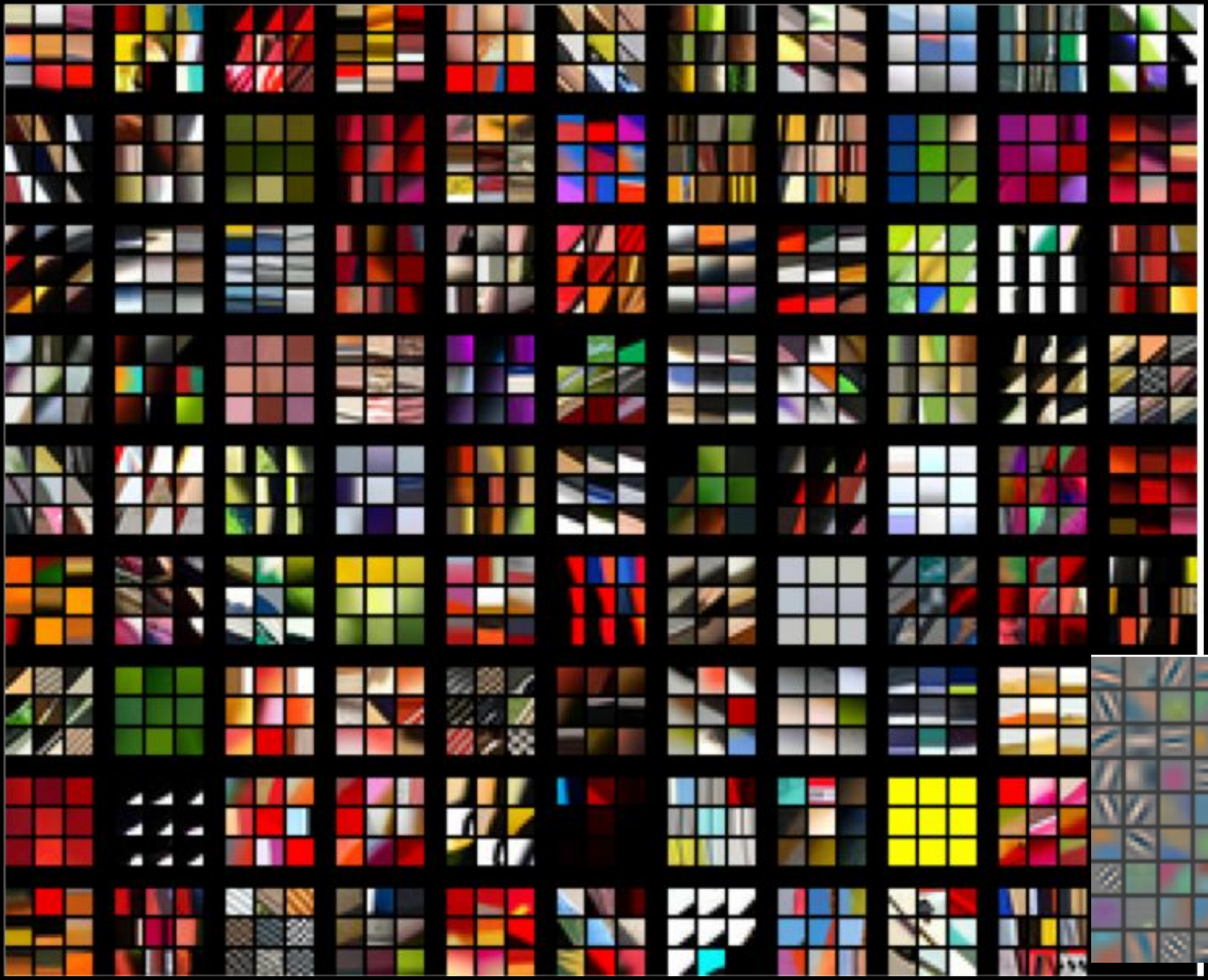
[Zeiler and Fergus. arXiv'13]

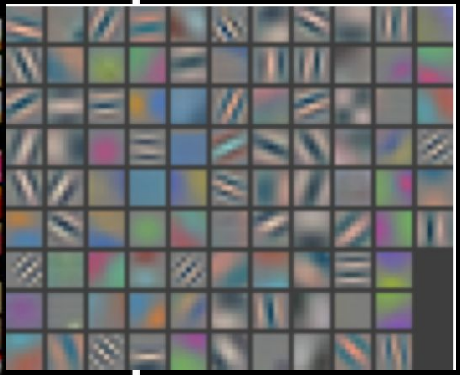
- Use ImageNet 2012 validation set
- Push each image through network



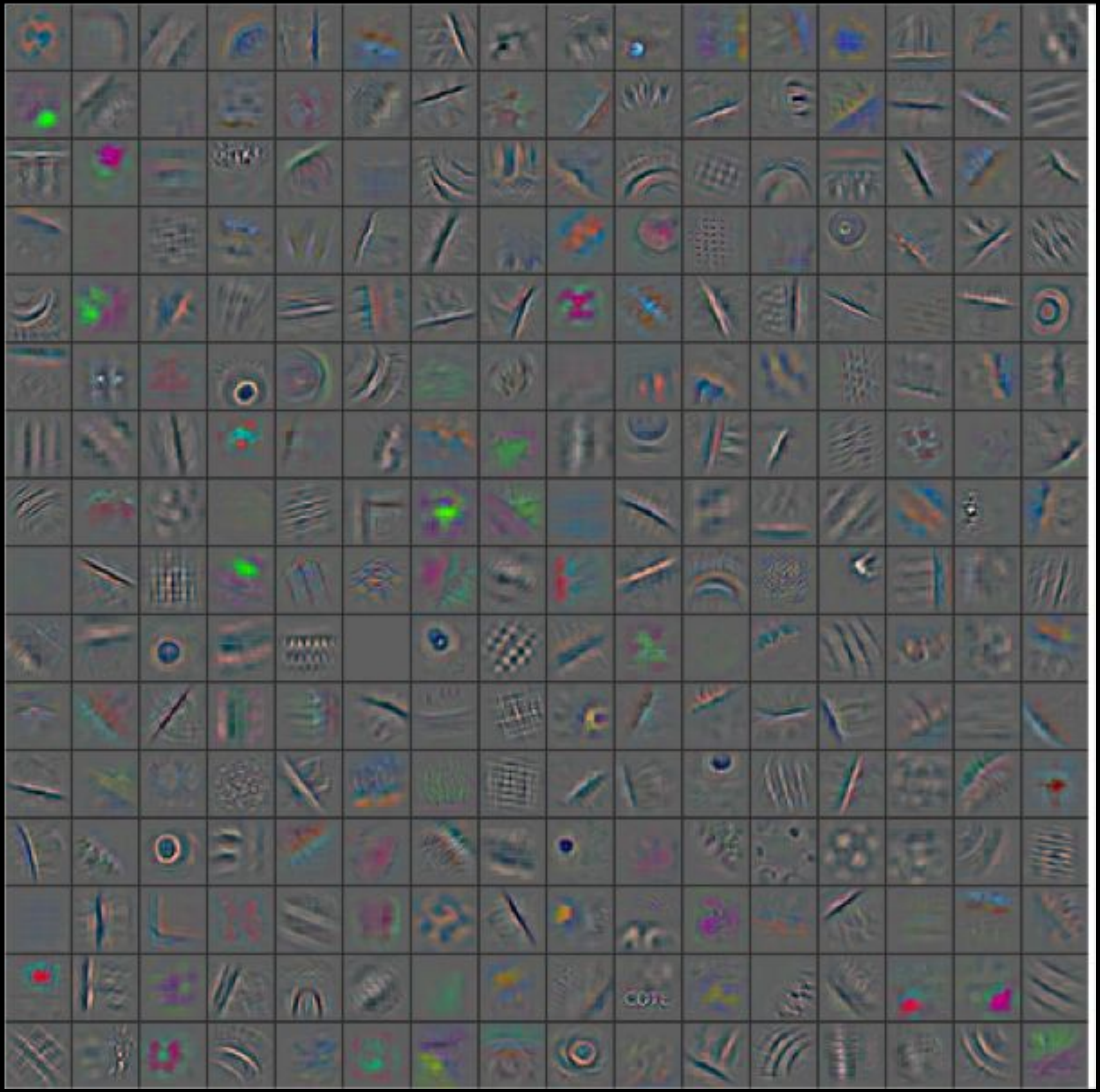
- Take max activation from feature map associated with each filter
- Use Deconvnet to project back to pixel space
- Use pooling "switches" peculiar to that activation

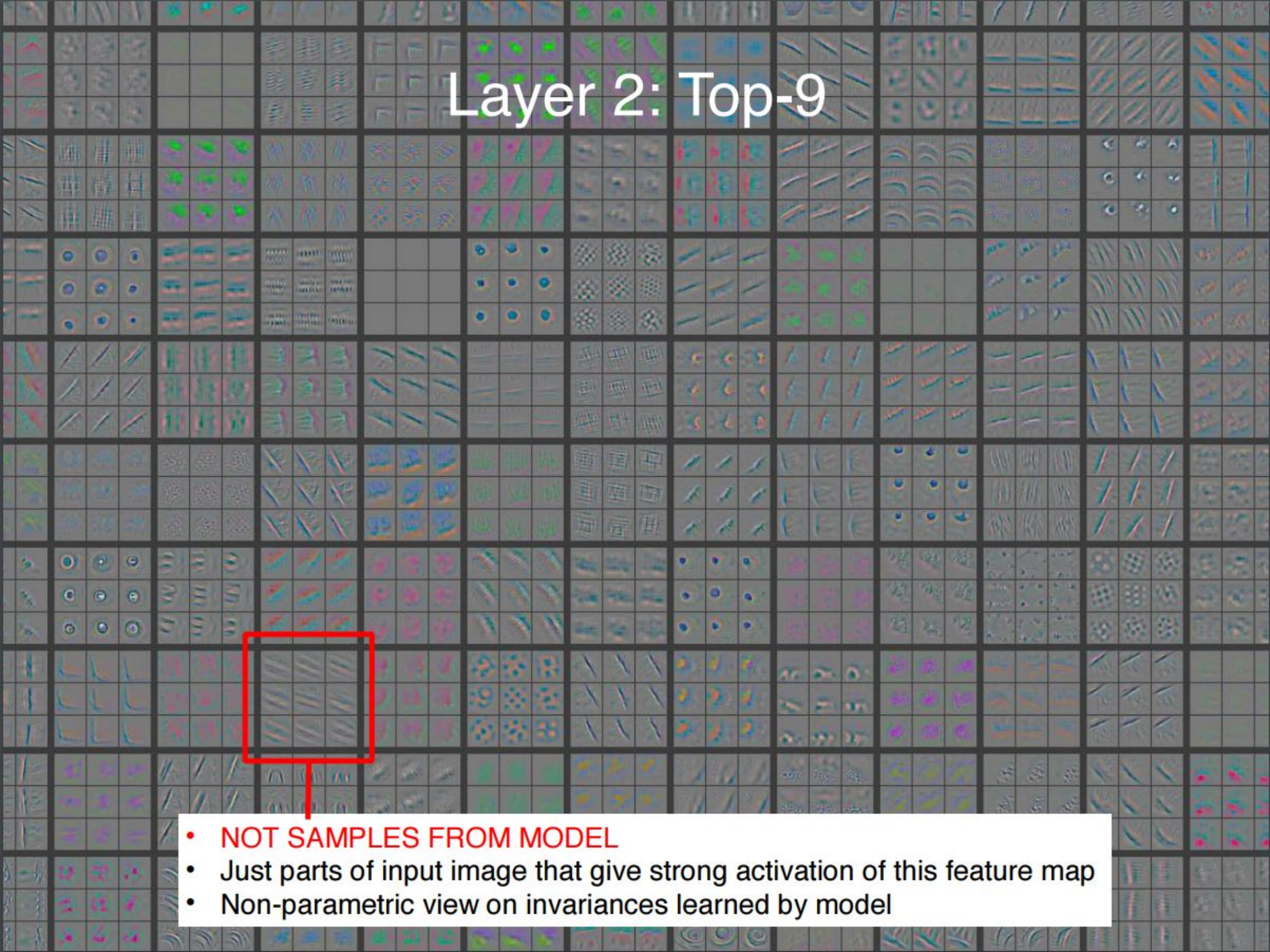
Layer 1: Top-9 Patches





Layer 2: Top-1

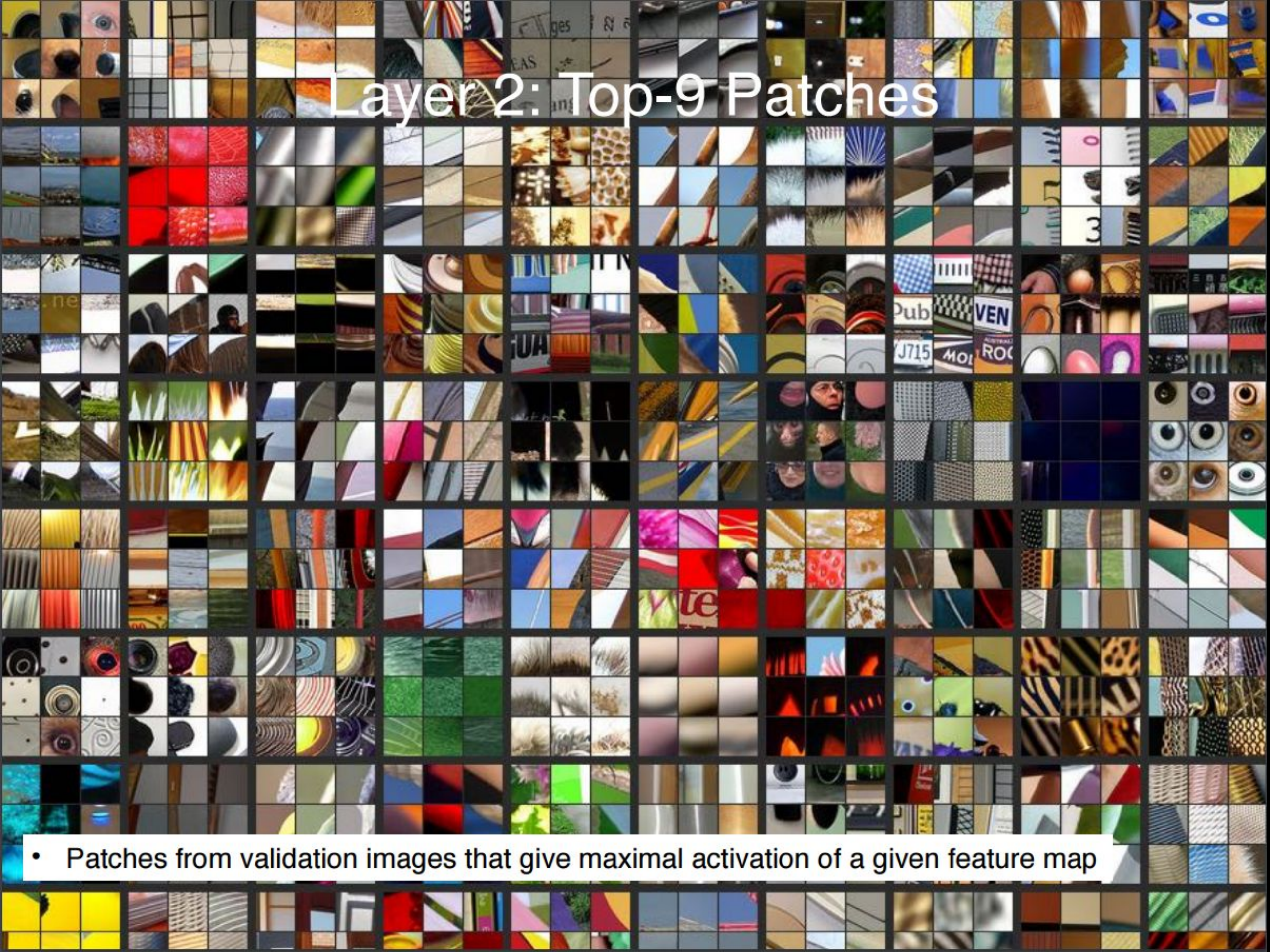




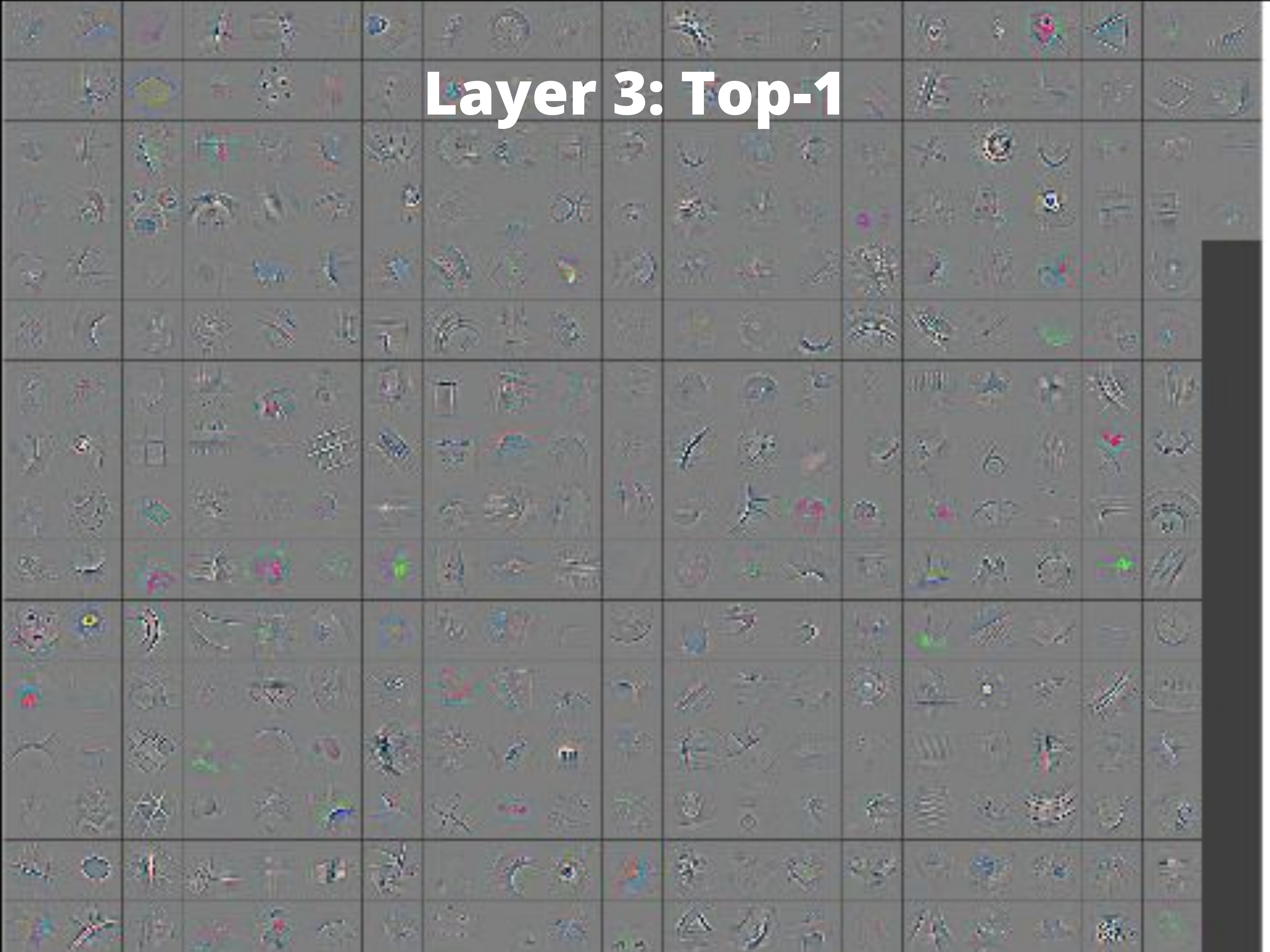
Layer 2: Top-9

- NOT SAMPLES FROM MODEL
- Just parts of input image that give strong activation of this feature map
- Non-parametric view on invariances learned by model

Layer 2: Top-9 Patches

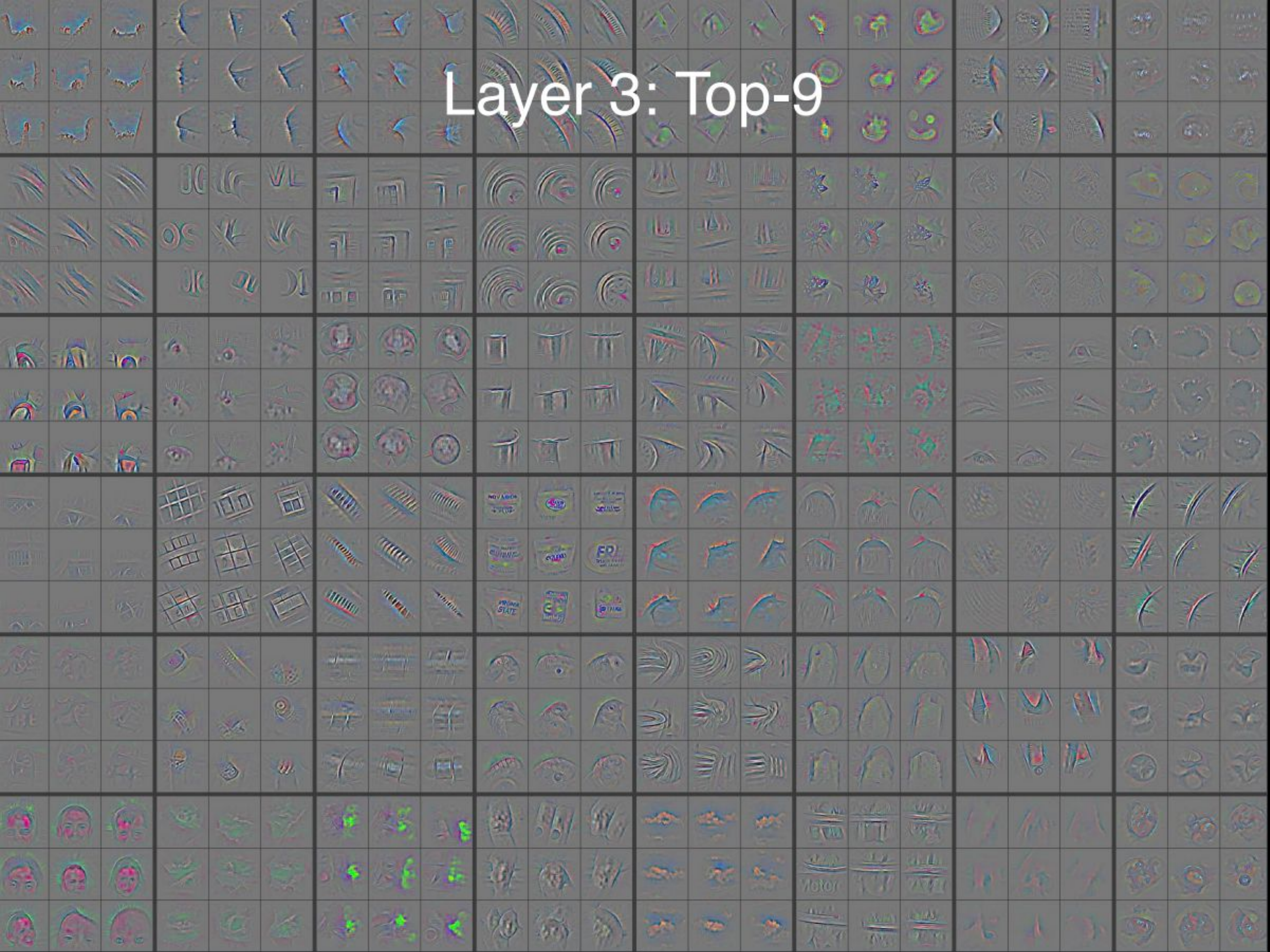


- Patches from validation images that give maximal activation of a given feature map

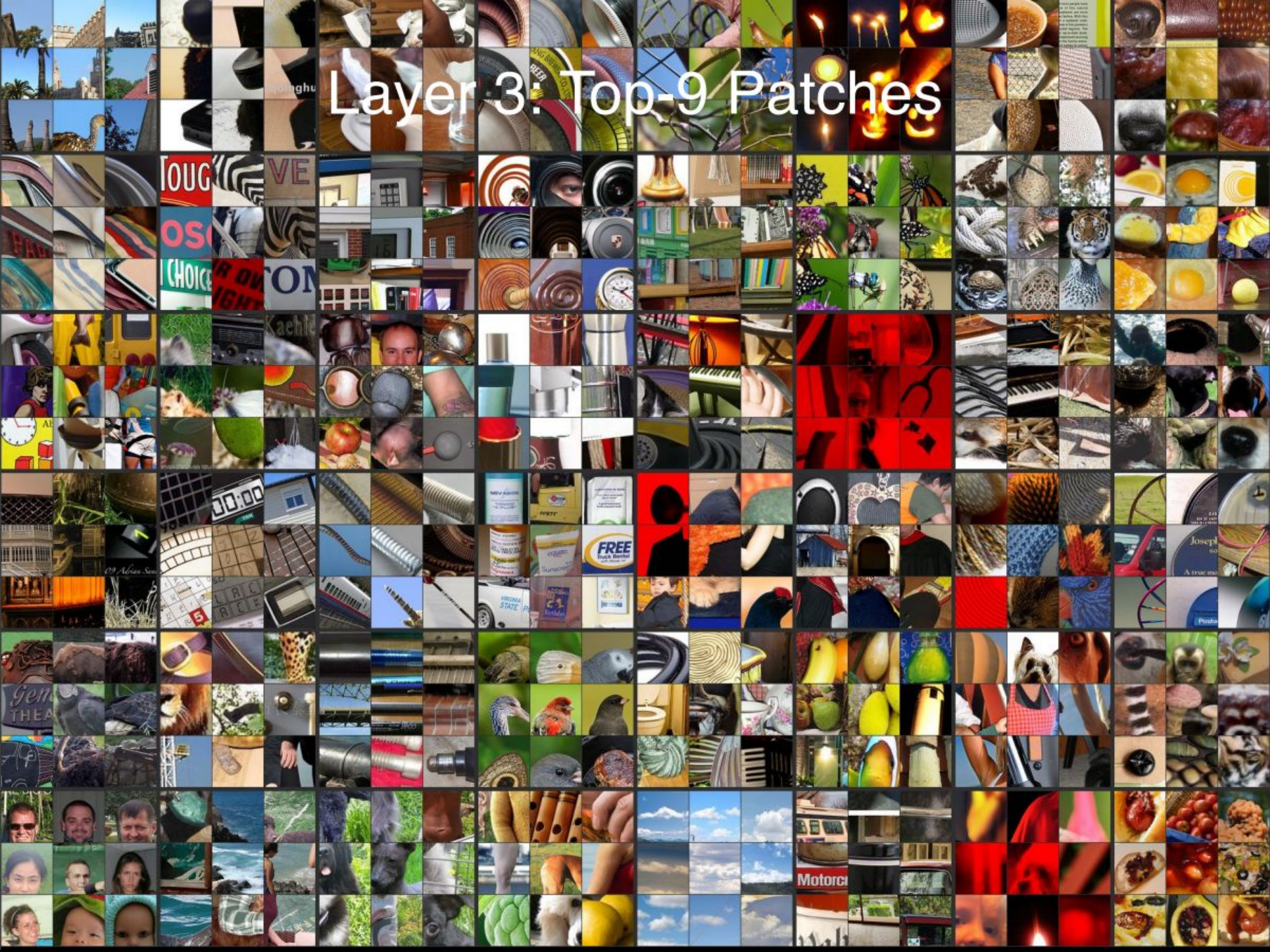


Layer 3: Top-1

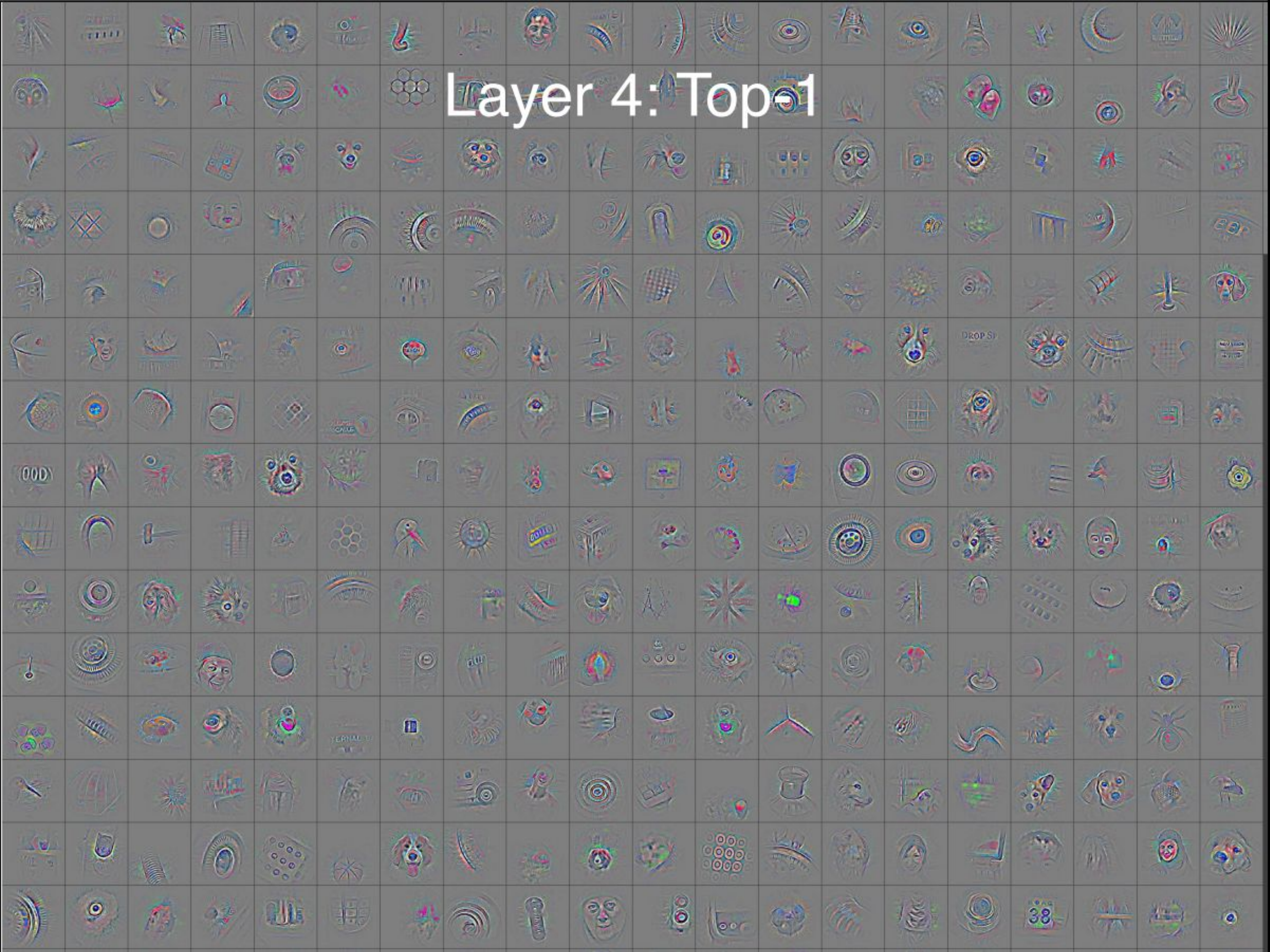
Layer 3: Top-9



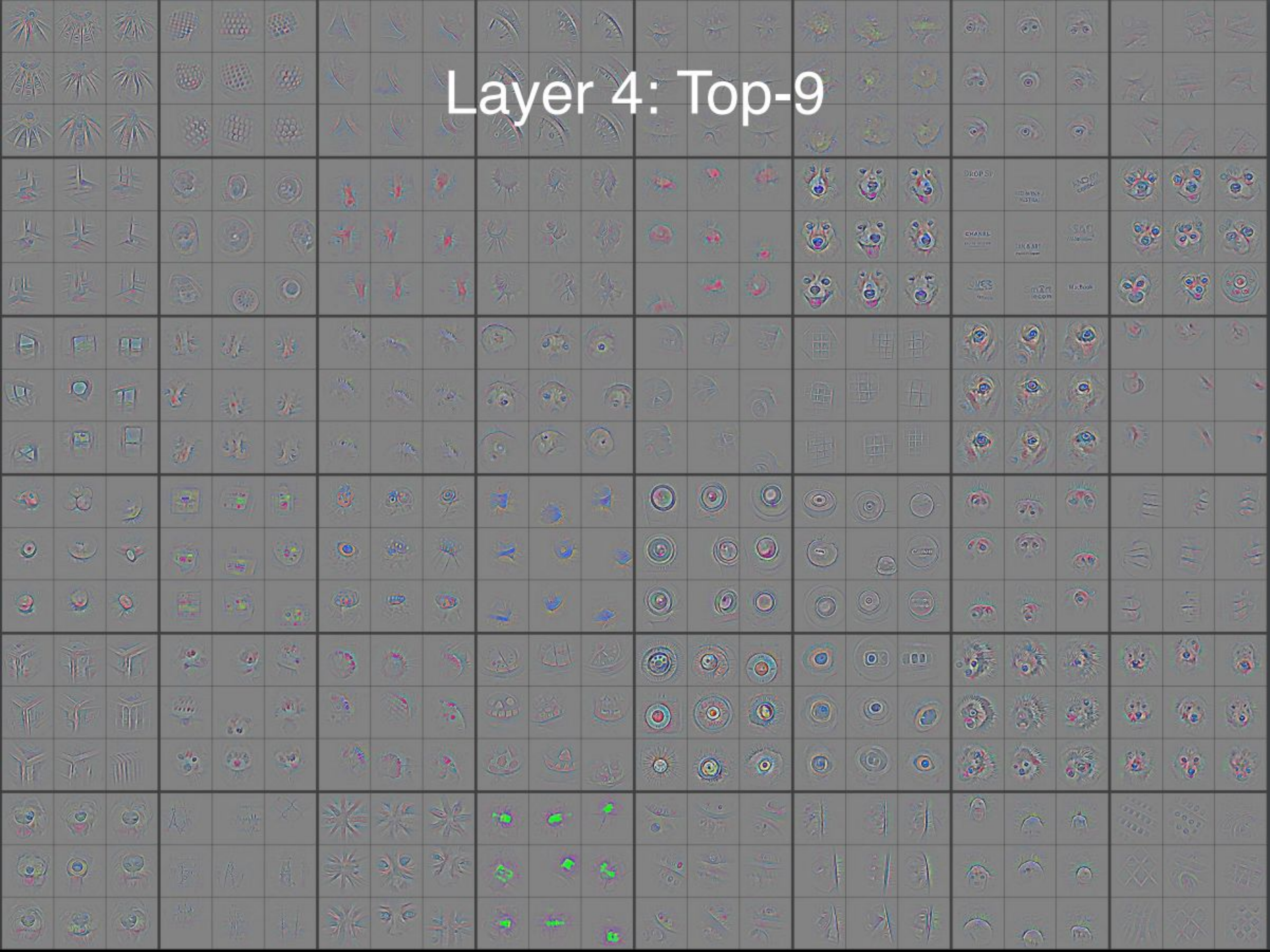
Layer 3: Top-9 Patches



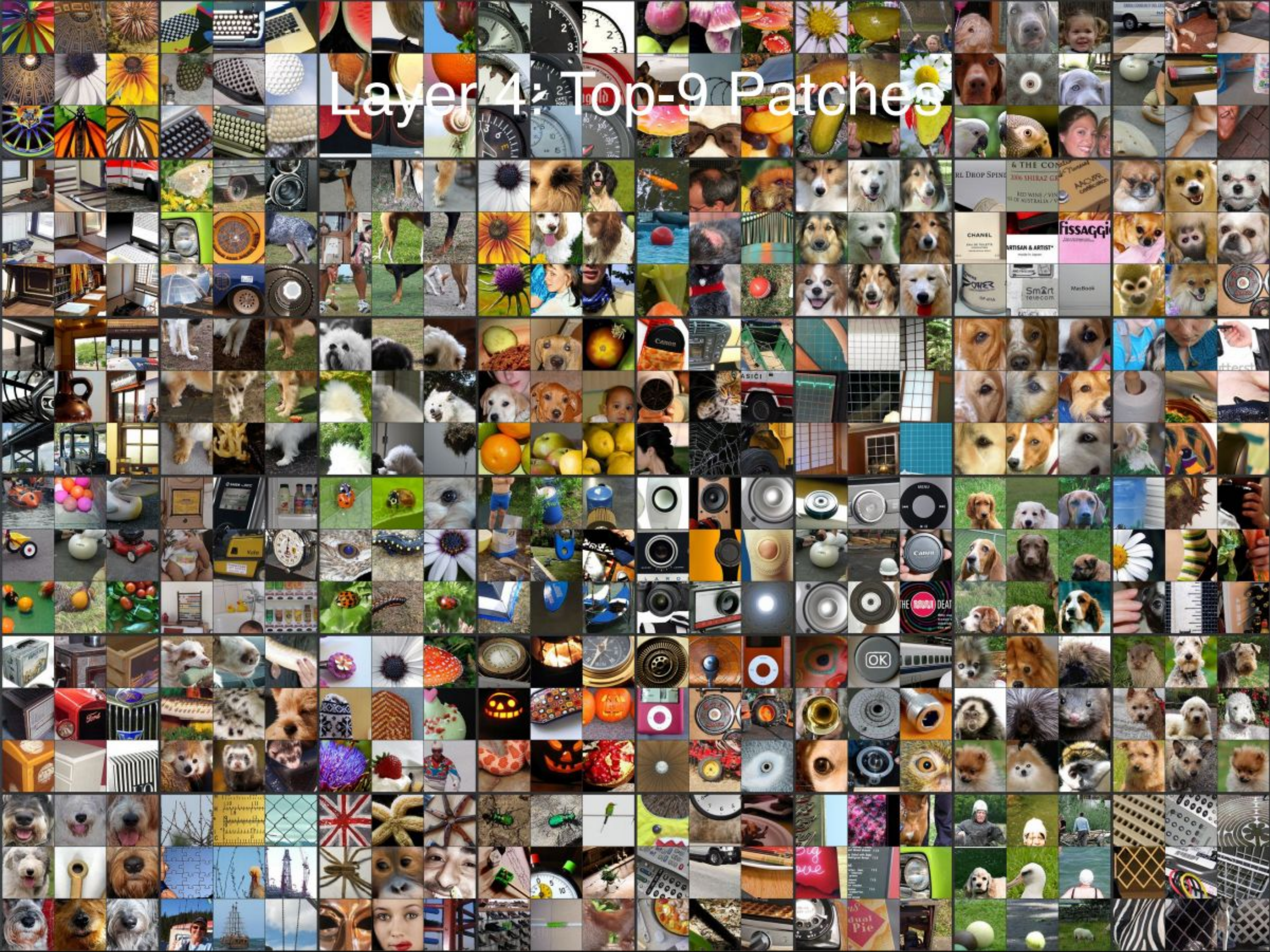
Layer 4: Top-1



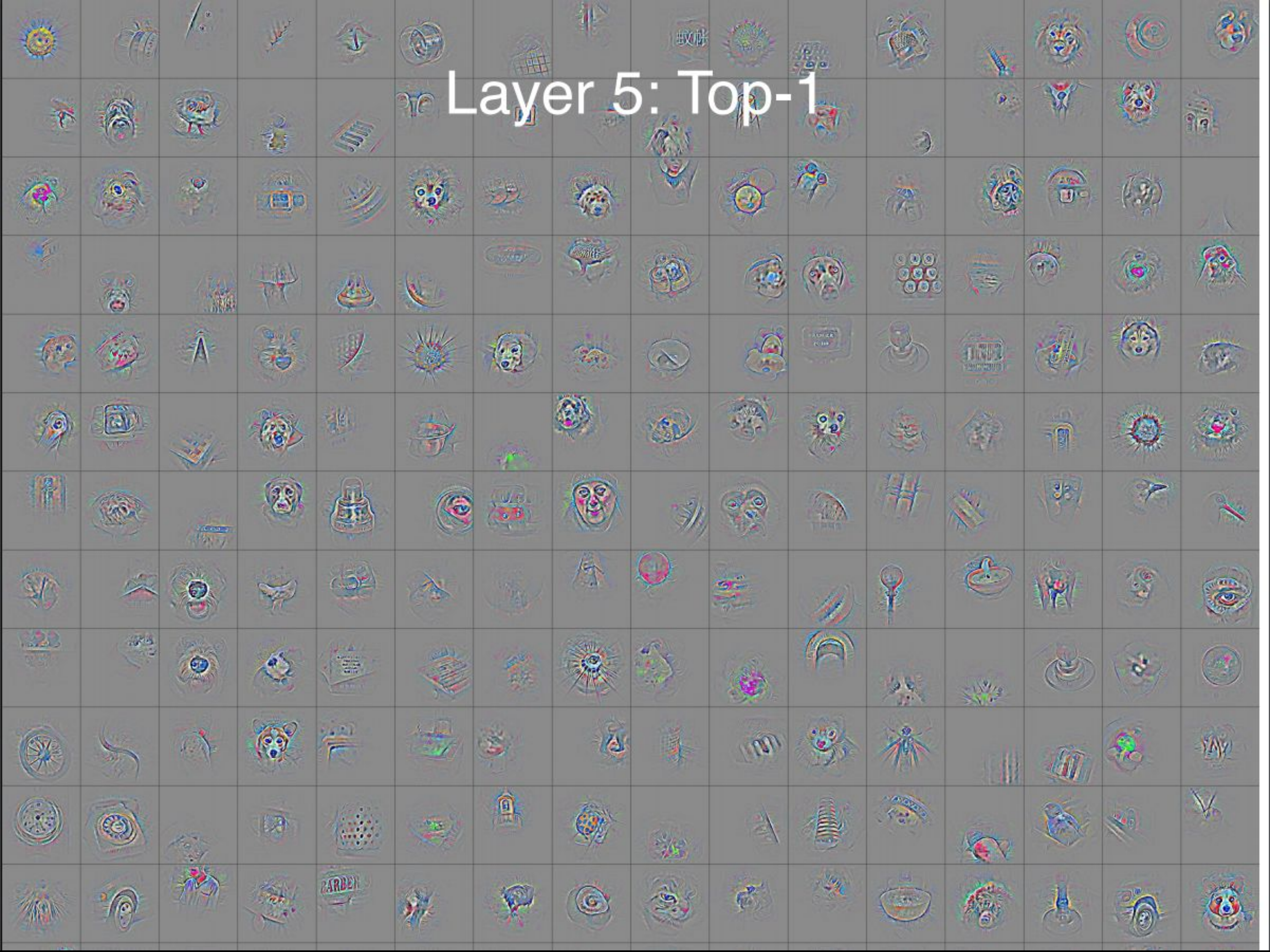
Layer 4: Top-9

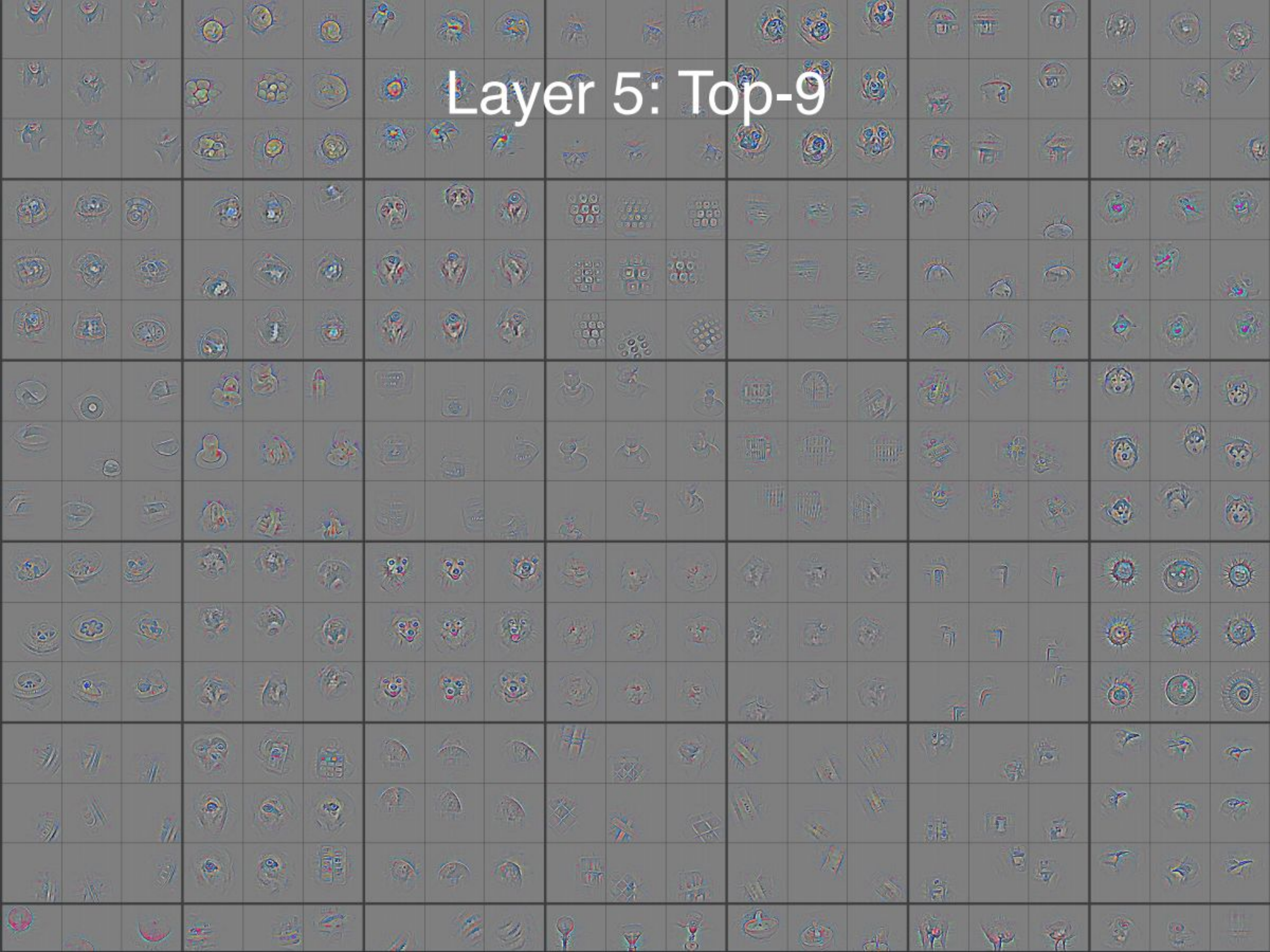


Layer 4: Top-9 Patches



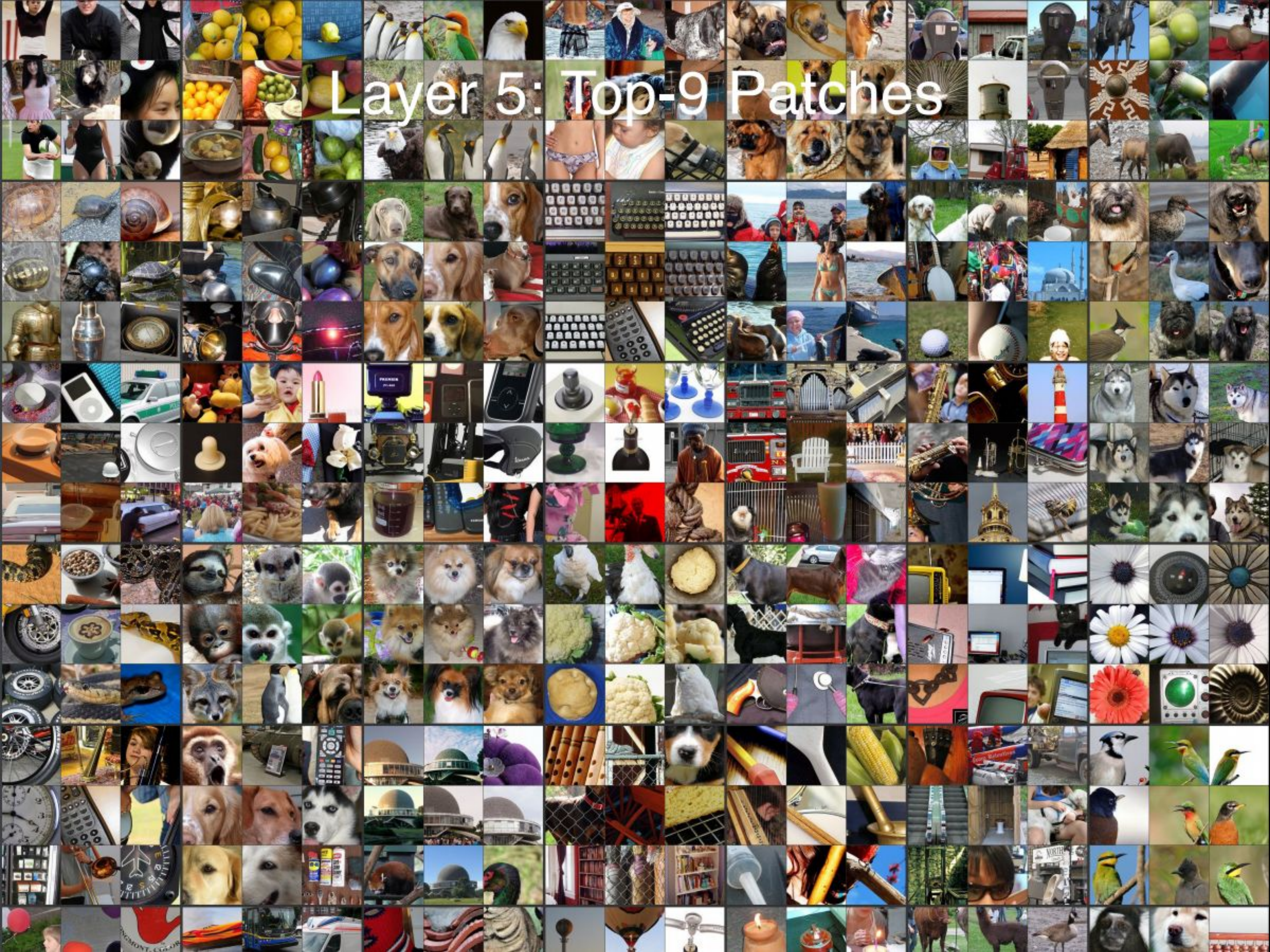
Layer 5: Top-1





Layer 5: Top-9

Layer 5: Top-9 Patches



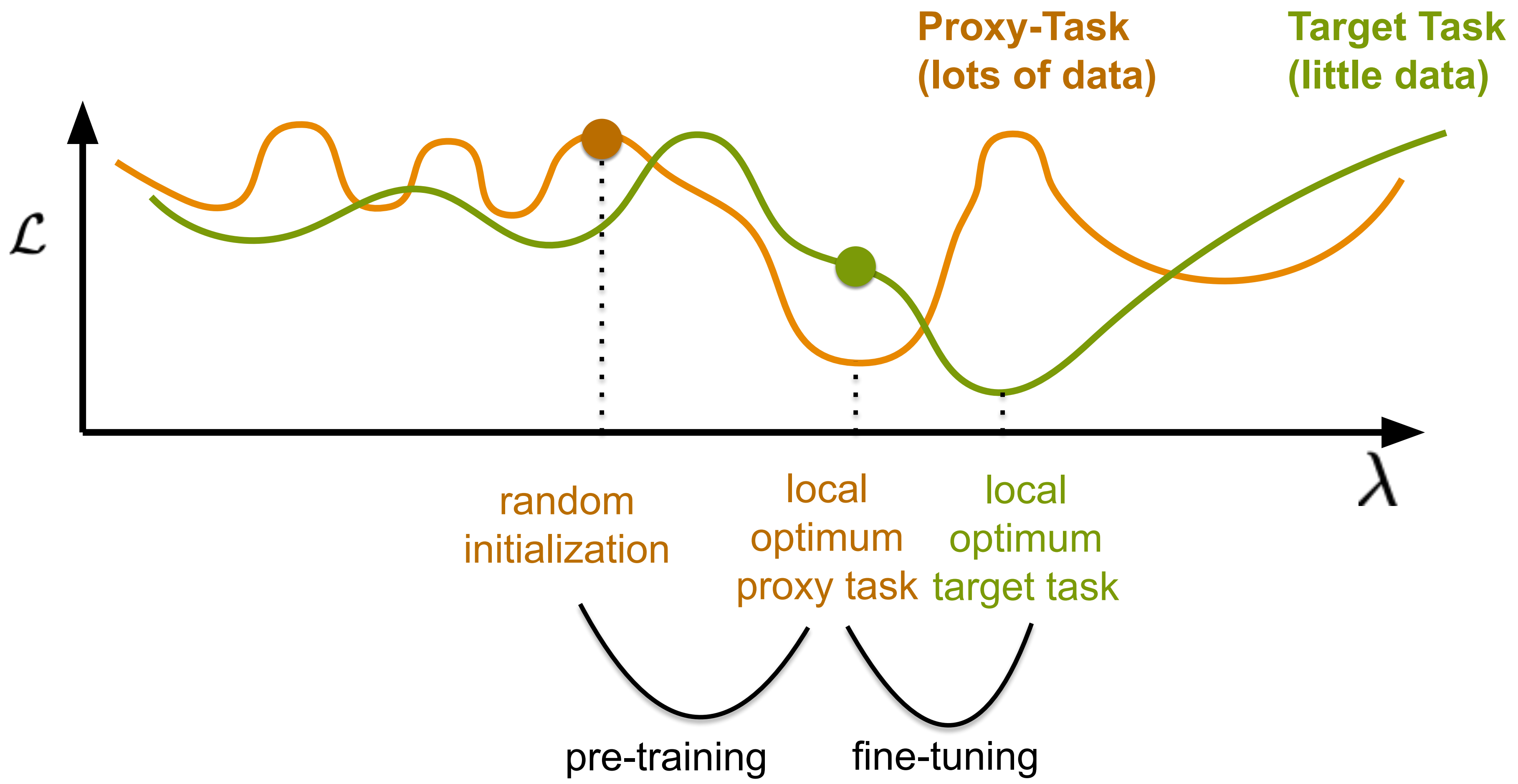
Overview Today

- Deep dive into convolutional networks
- Visualizing convolutional networks
- **Feature Generalization**
  - “pre-training” on large dataset, “fine-tuning” on target dataset

Feature Generalization and Pretraining: Overview

- Typically we are lacking data
- But there are large datasets for some tasks
- Idea:
  - Can we use learnt features from other trasks?
  - How can we transfer learnt features from other tasks?
  - Can we still do end-to-end learning?

Feature Generalization and Pretraining: Overview



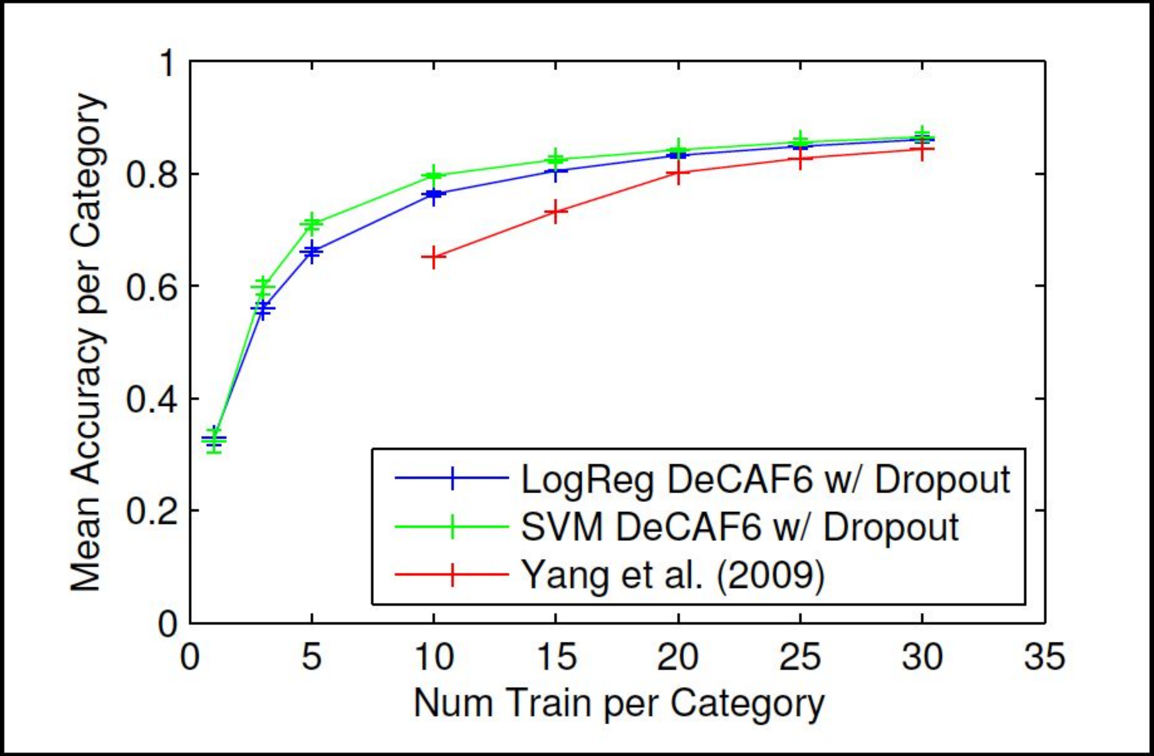
Training Features on Other Datasets

- Train model on ImageNet 2012 training set
- Re-train classifier on new dataset
  - Just the softmax layer
- Classify test set of new dataset

slide credit: Rob Fergus, NIPS’13 tutorial

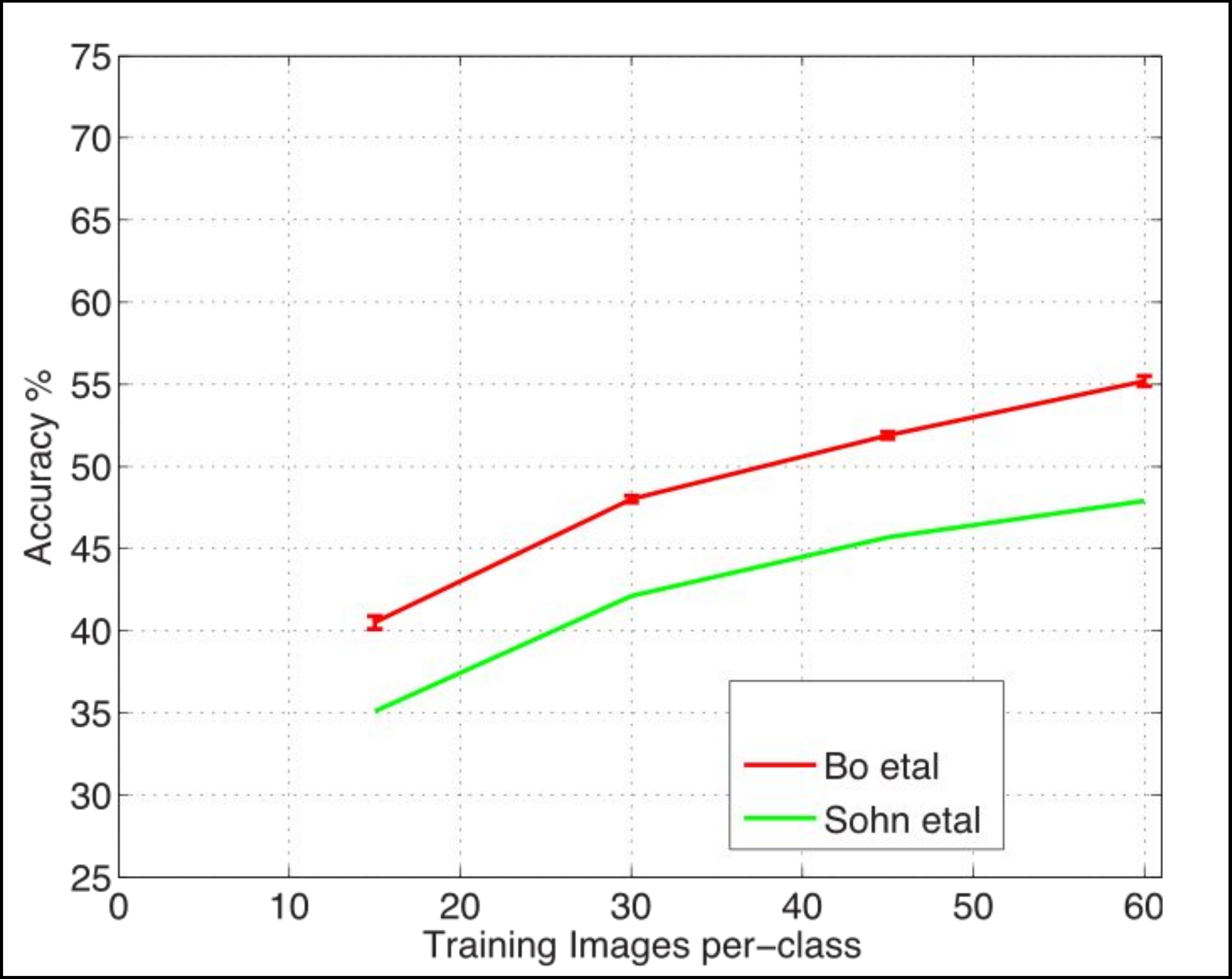
Caltech-101

Donahue et al., *DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition*, arXiv 1310.1531, 2013



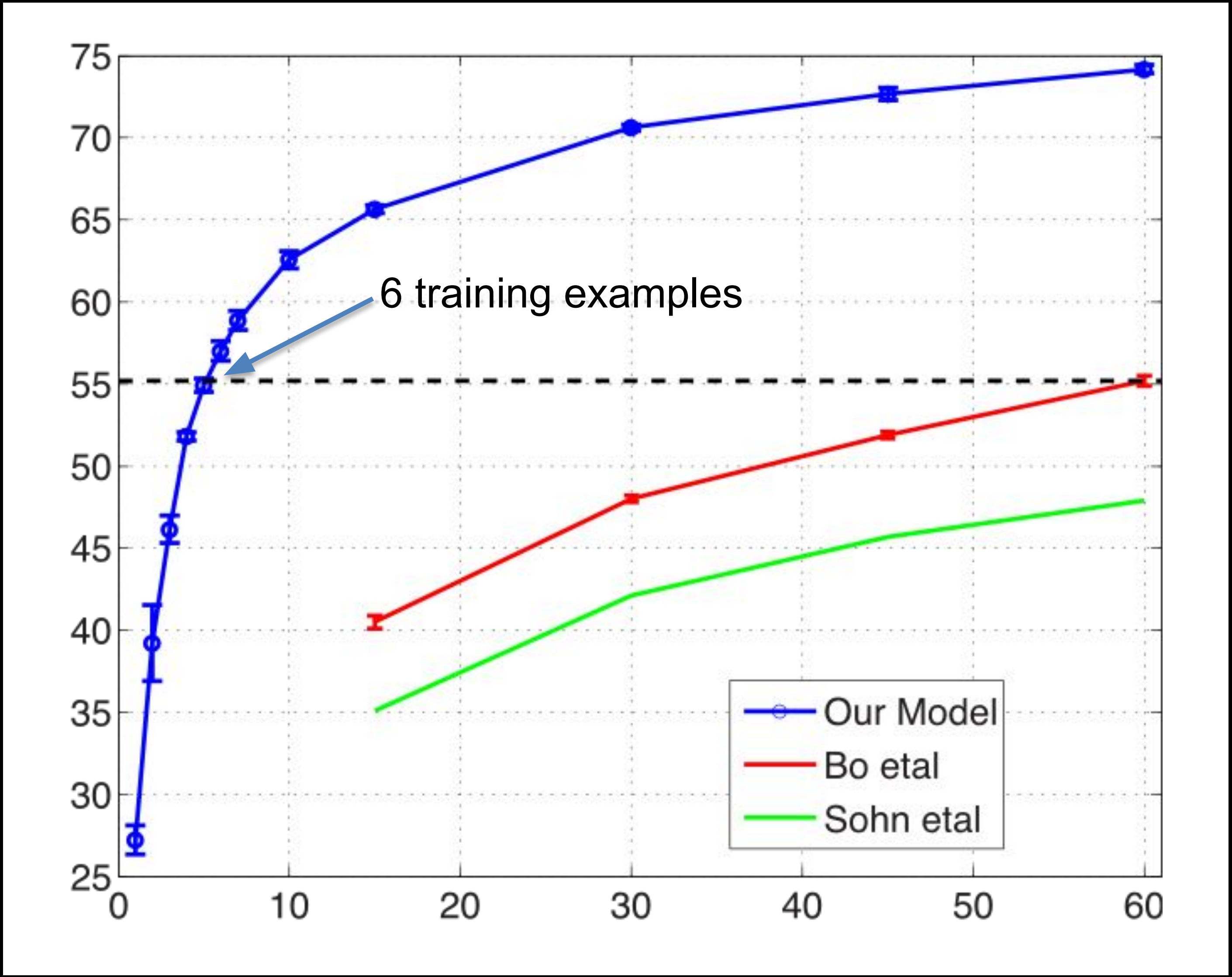
Caltech 256

Zeiler & Fergus, *Visualizing and Understanding Convolutional Networks*, arXiv 1311.2901, 2013



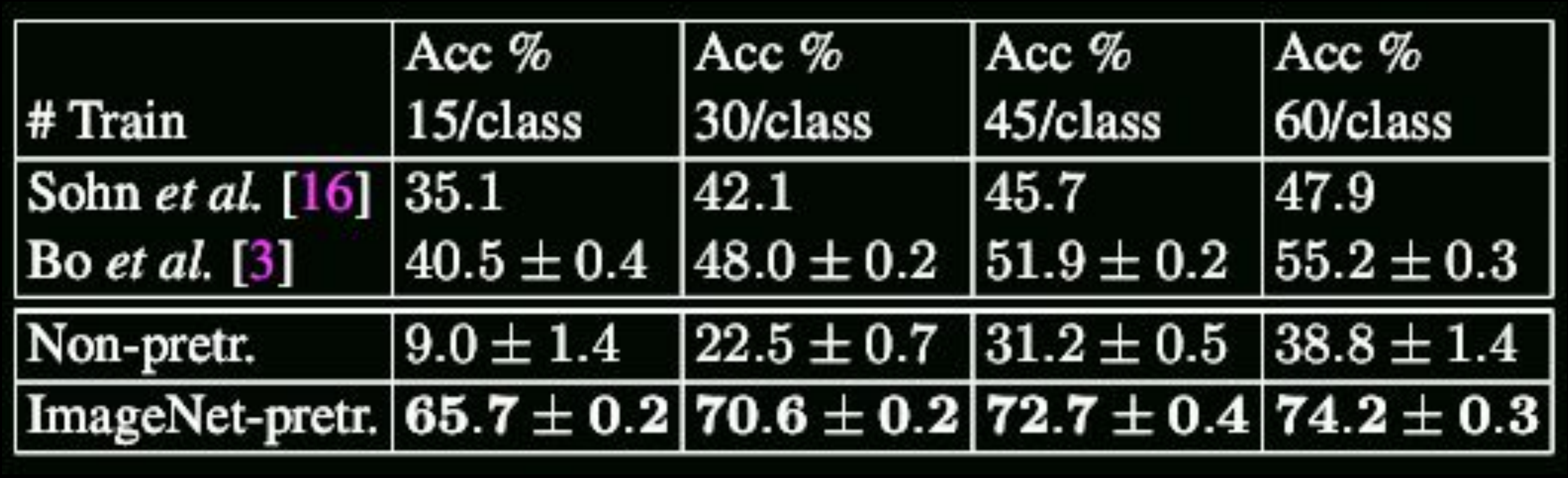
Caltech 256

Zeiler & Fergus, *Visualizing and Understanding Convolutional Networks*, arXiv 1311.2901, 2013



Caltech 256

Zeiler & Fergus, *Visualizing and Understanding Convolutional Networks*, arXiv 1311.2901, 2013



# Train	Acc % 15/class	Acc % 30/class	Acc % 45/class	Acc % 60/class
Sohn *et al.* [16]	35.1	42.1	45.7	47.9
Bo *et al.* [3]	40.5 ± 0.4	48.0 ± 0.2	51.9 ± 0.2	55.2 ± 0.3
Non-pretr.	9.0 ± 1.4	22.5 ± 0.7	31.2 ± 0.5	38.8 ± 1.4
ImageNet-pretr.	**65.7 ± 0.2**	**70.6 ± 0.2**	**72.7 ± 0.4**	**74.2 ± 0.3**

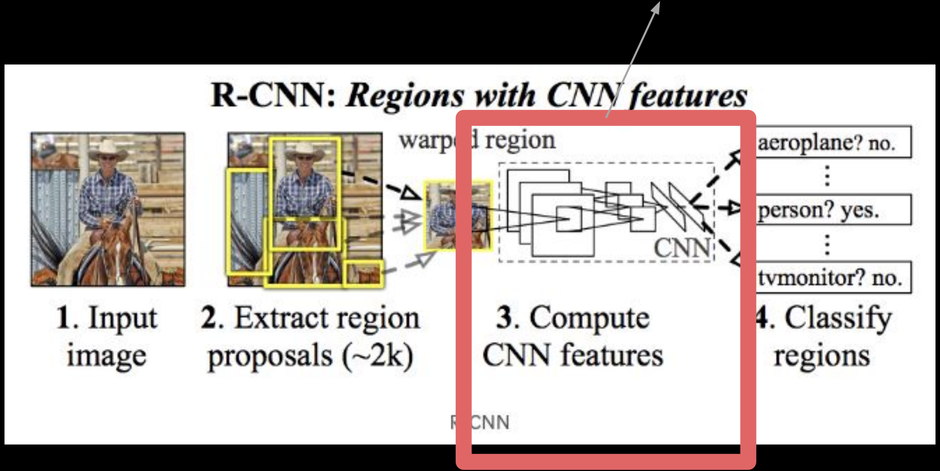
[3] L. Bo, X. Ren, and D. Fox. Multipath sparse coding using hierarchical matching pursuit. In CVPR, 2013.

[16] K. Sohn, D. Jung, H. Lee, and A. Hero III. Efficient learning of sparse, distributed, convolutional feature representations for object recognition. In ICCV, 2011.

slide credit: Rob Fergus, NIPS’13 tutorial

Standard Practice in many tasks

- Object detection and Segmentation
  - Feature extraction layers are **pre-trained** on Imagenet



- Image Captioning and question answering
  - Image embeddings are obtained with pretrained network