University of Nottingham
UK | CHINA | MALAYSIA

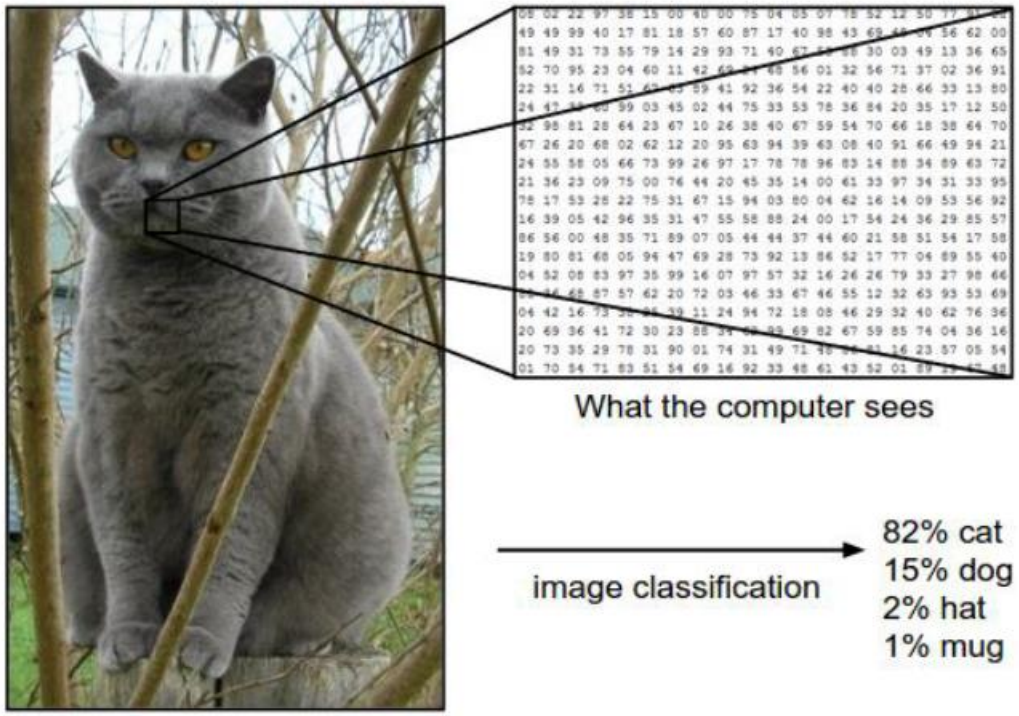
COMP3055 Machine Learning

Topic 15 – Convolutional Neural Network

Zheng LU

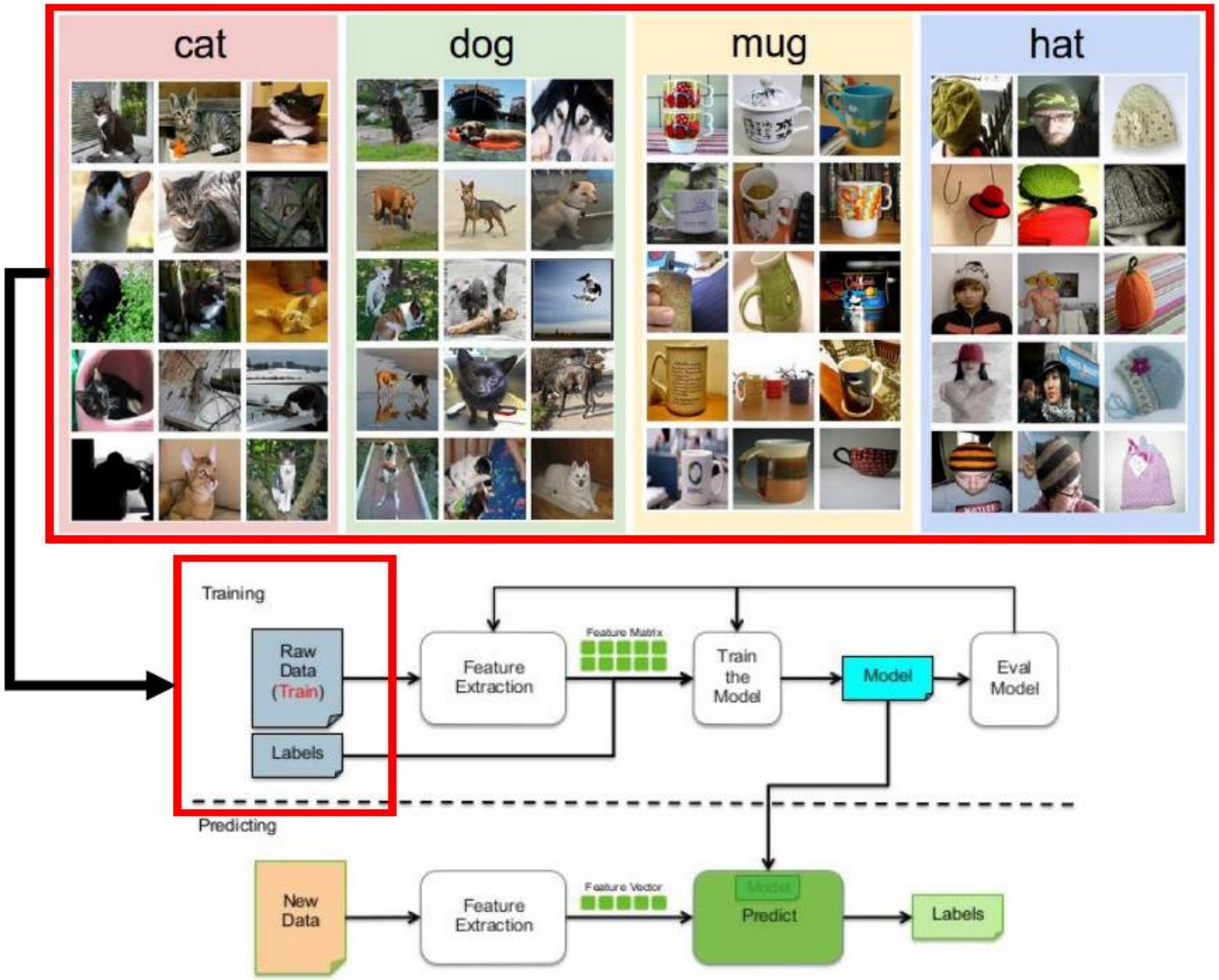
2024 Autumn

Image are Numbers



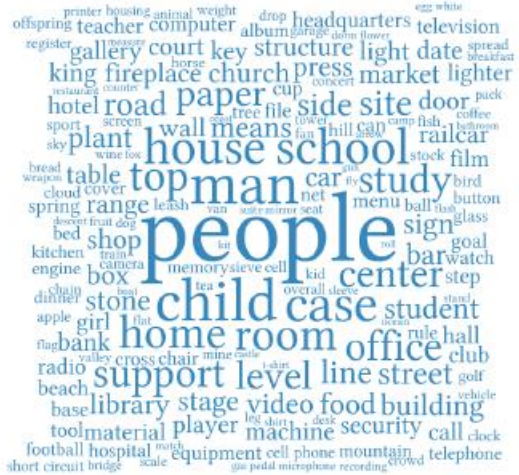
- **Regression:** The output variable takes continuous values
- **Classification:** The output variable takes class labels
  - Underneath it may still produce continuous values such as probability of belonging to a particular class.

Image Classification Pipeline



Famous CV Datasets

MNIST: handwritten digits



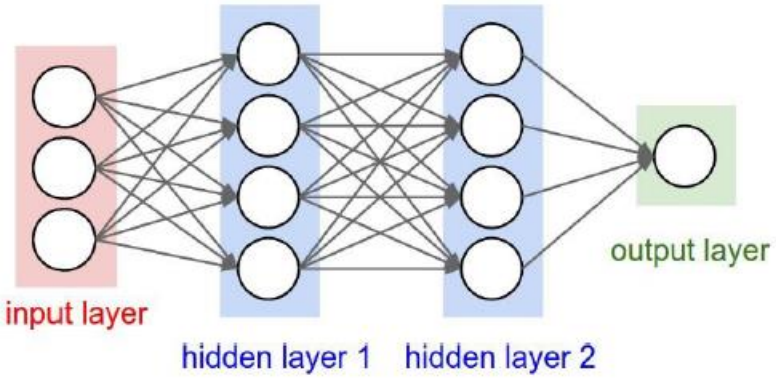
ImageNet: WordNet hierarchy

CIFAR-10(0): tiny images

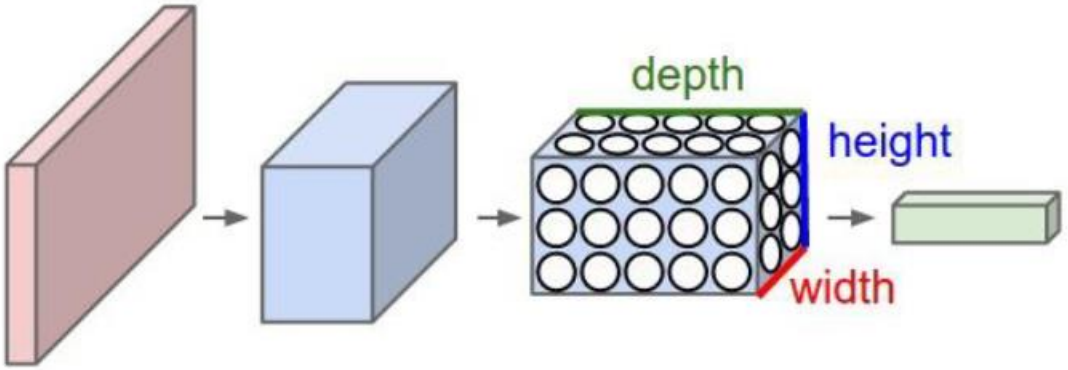
Places: natural scenes

Convolutional Neural Network

Regular neural network (fully connected):



Convolutional neural network:



Each layer takes a 3d volume, produces 3d volume with some smooth function that may or may not have parameters.

Why CNN for Image

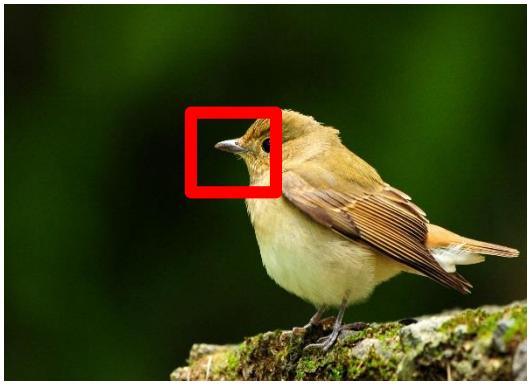
- Some patterns are much smaller than the whole image

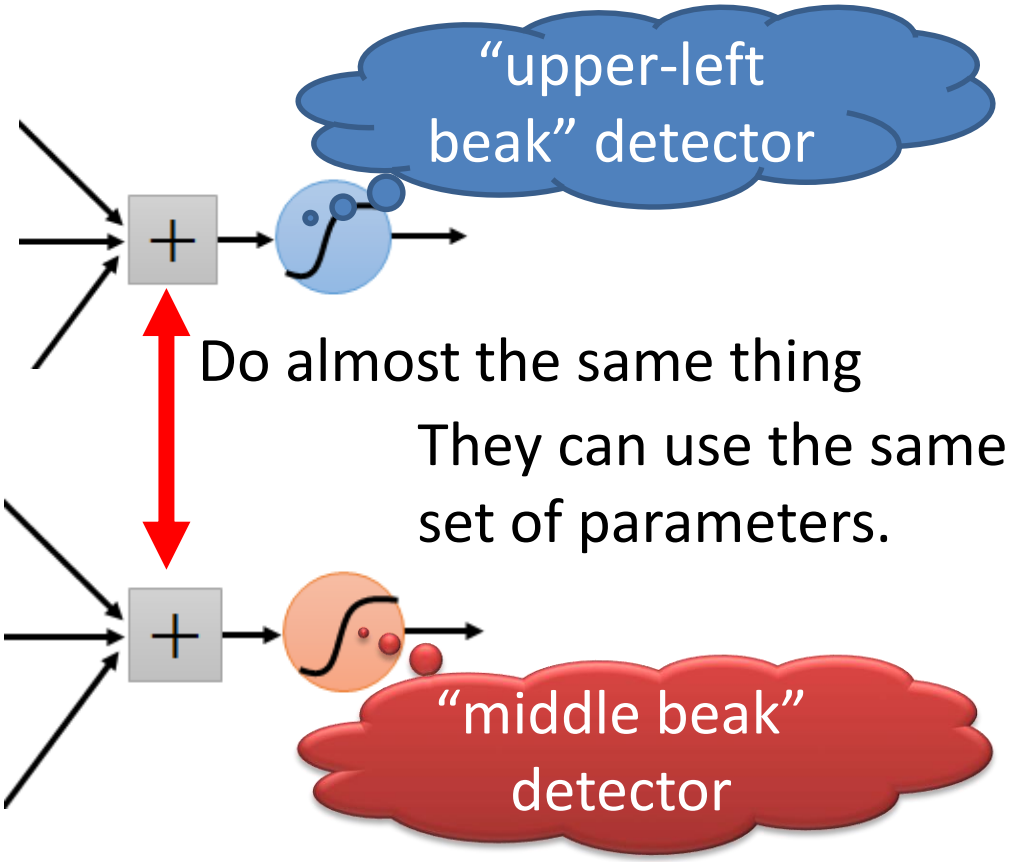
A neuron does not have to see the whole image to discover the pattern.

Connecting to small region with less parameters

Why CNN for Image

- The same patterns appear in different regions.





Why CNN for Image

- Downsampling the pixels will not change the object

We can downsample the pixels to make image smaller

→ Less parameters for the network to process the image

CNN Layers

- **INPUT**: will hold the raw data (e.g., pixel values of the image)
- **CONV**: will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume
- **RELU**: will apply an elementwise activation function
- **POOL**: will perform a downsampling operation along the spatial dimensions (width, height)
- **FC**: will compute the class scores

Convolution Layer

Those are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

Matrix

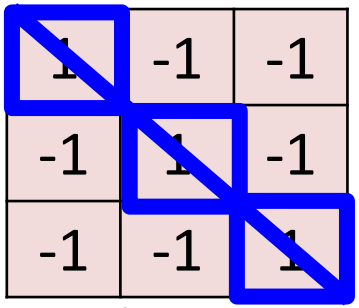
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

Matrix

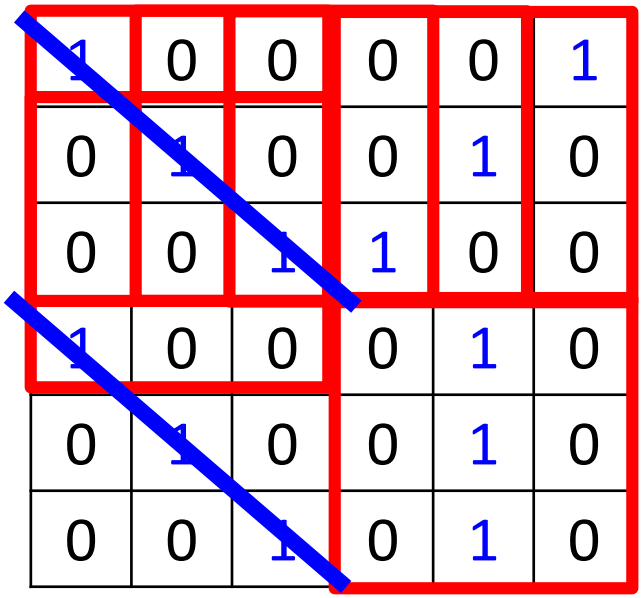
Each filter detects a small pattern (3 x 3).

CNN – Convolution

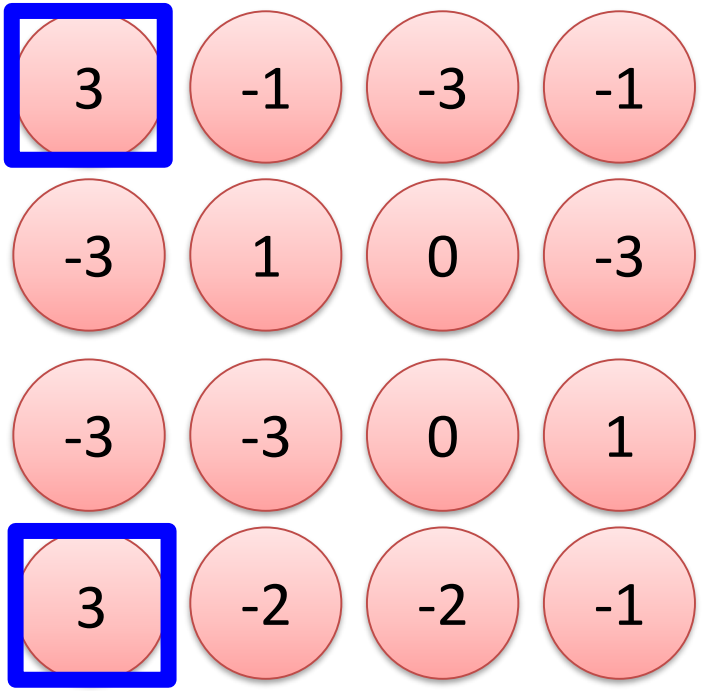


Filter 1

stride=1



6 x 6 image



CNN – Convolution

-1	1	-1
-1	1	-1
-1	1	-1

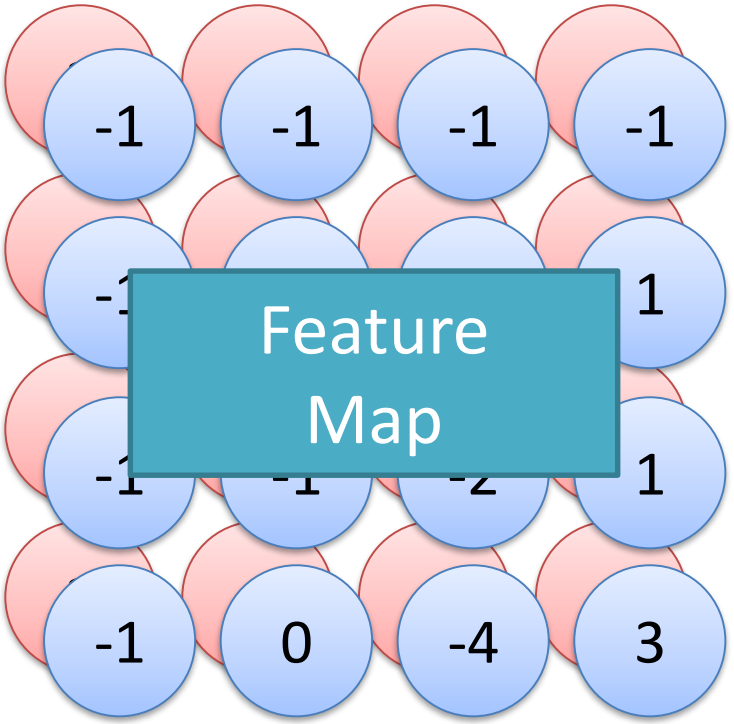
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

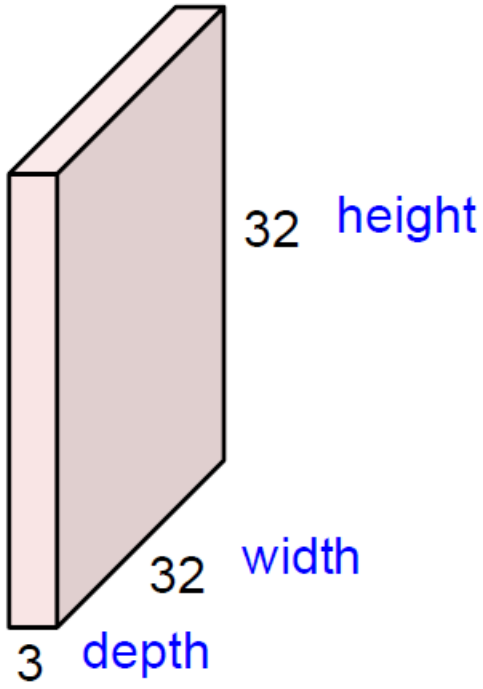
Do the same process for every filter



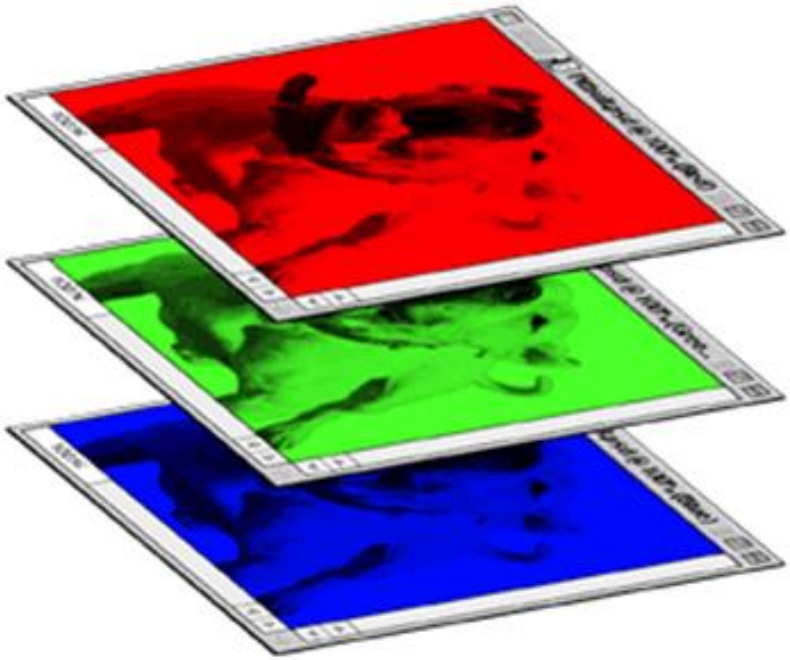
4 x 4 image

Convolution Layer

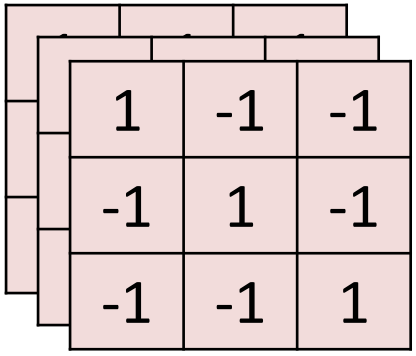
32x32x3 image -> preserve spatial structure



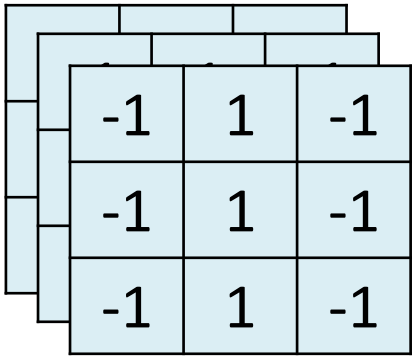
Colorful image



Convolution Layer

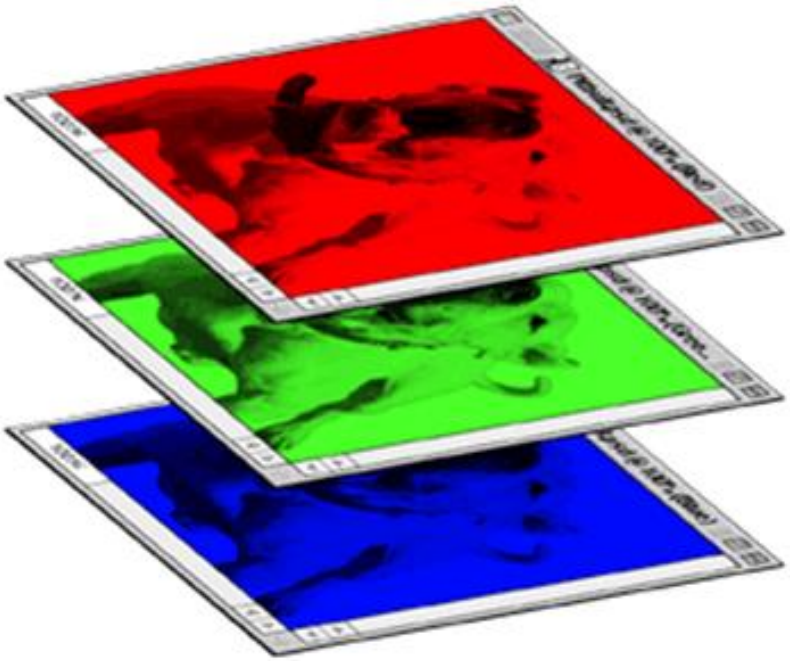


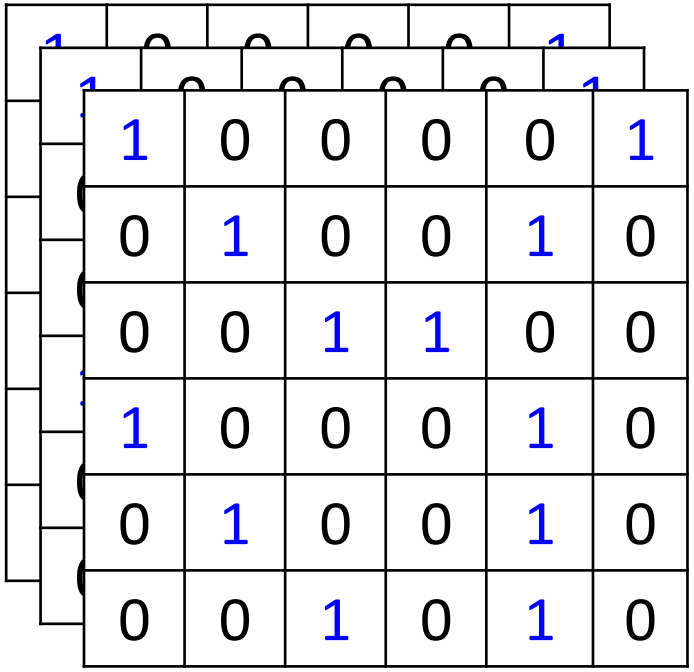
Filter 1



Filter 2

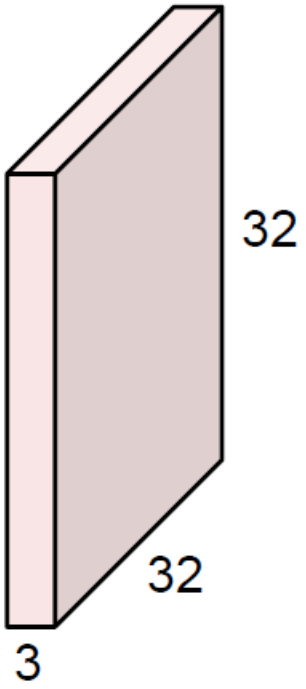
Colorful image



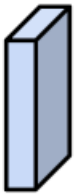


Convolution Layer

32x32x3 image

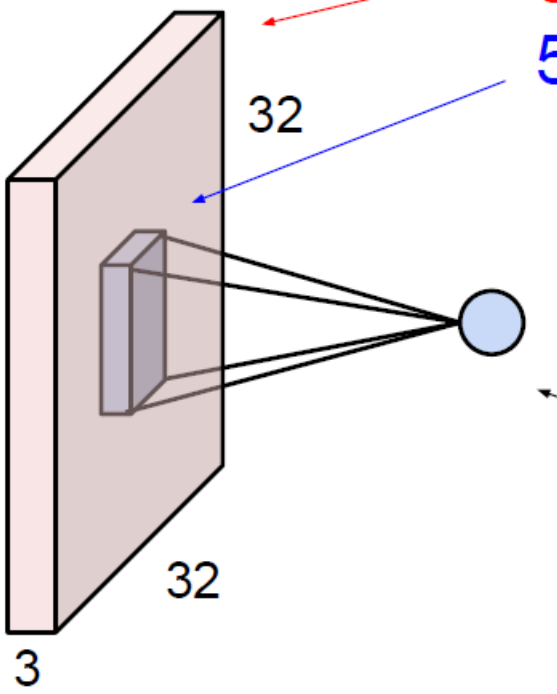


5x5x3 filter



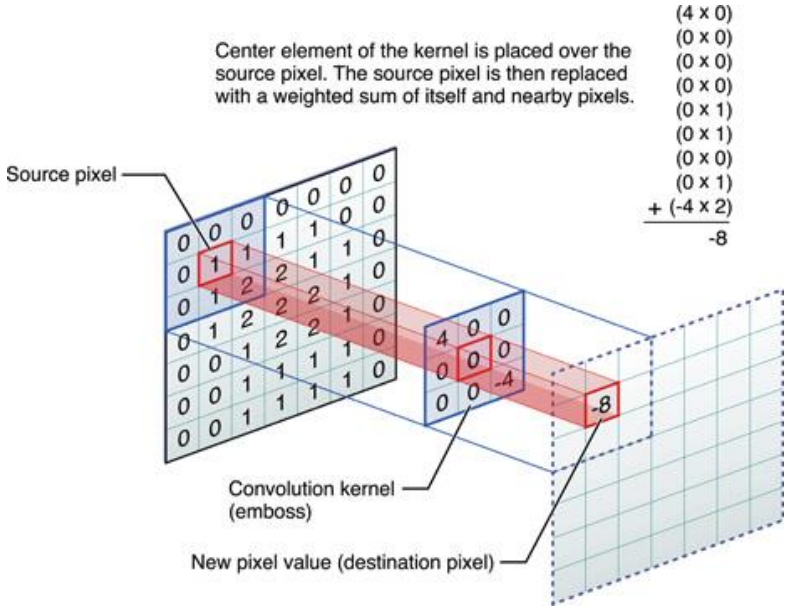
Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Convolution Layer



32x32x3 image

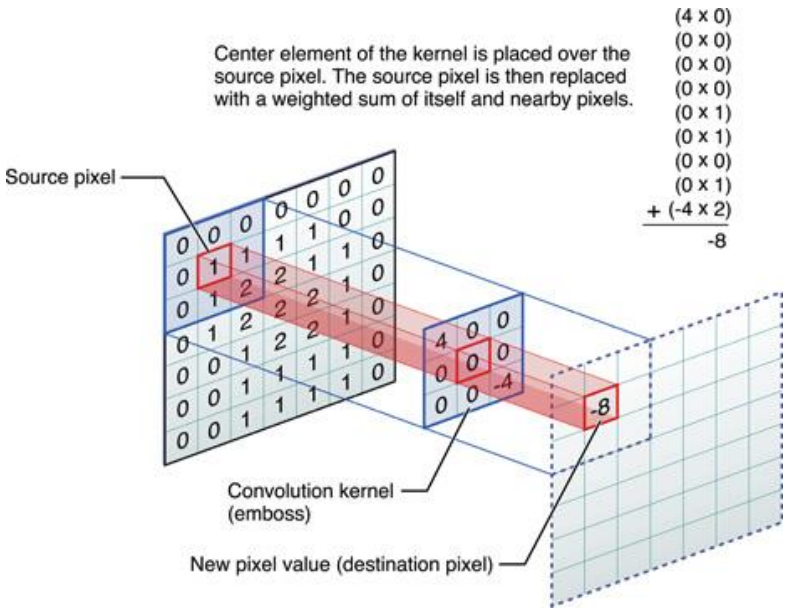
5x5x3 filter w

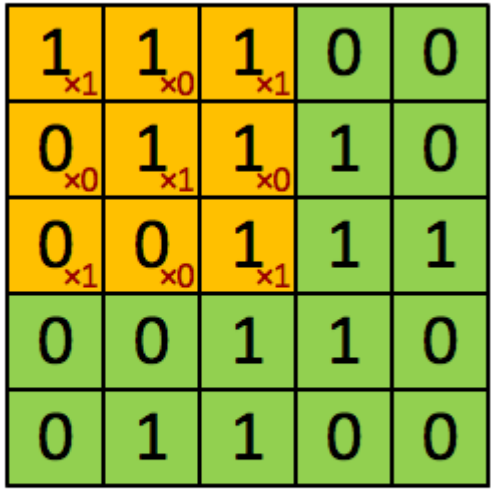


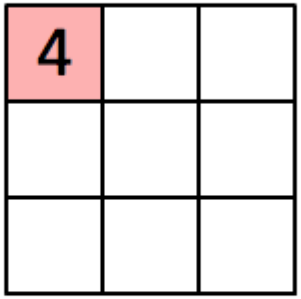
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

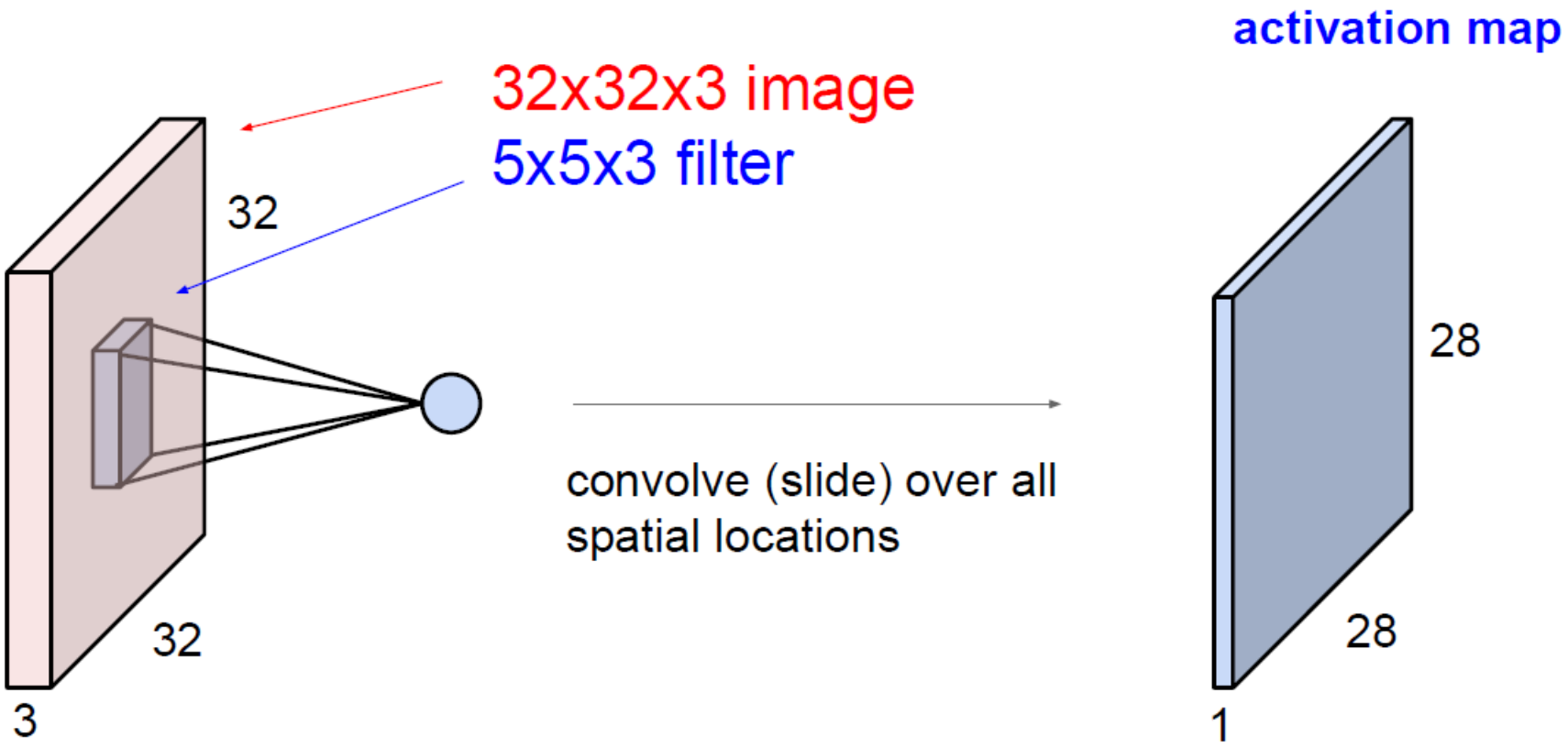
Convolution Layer



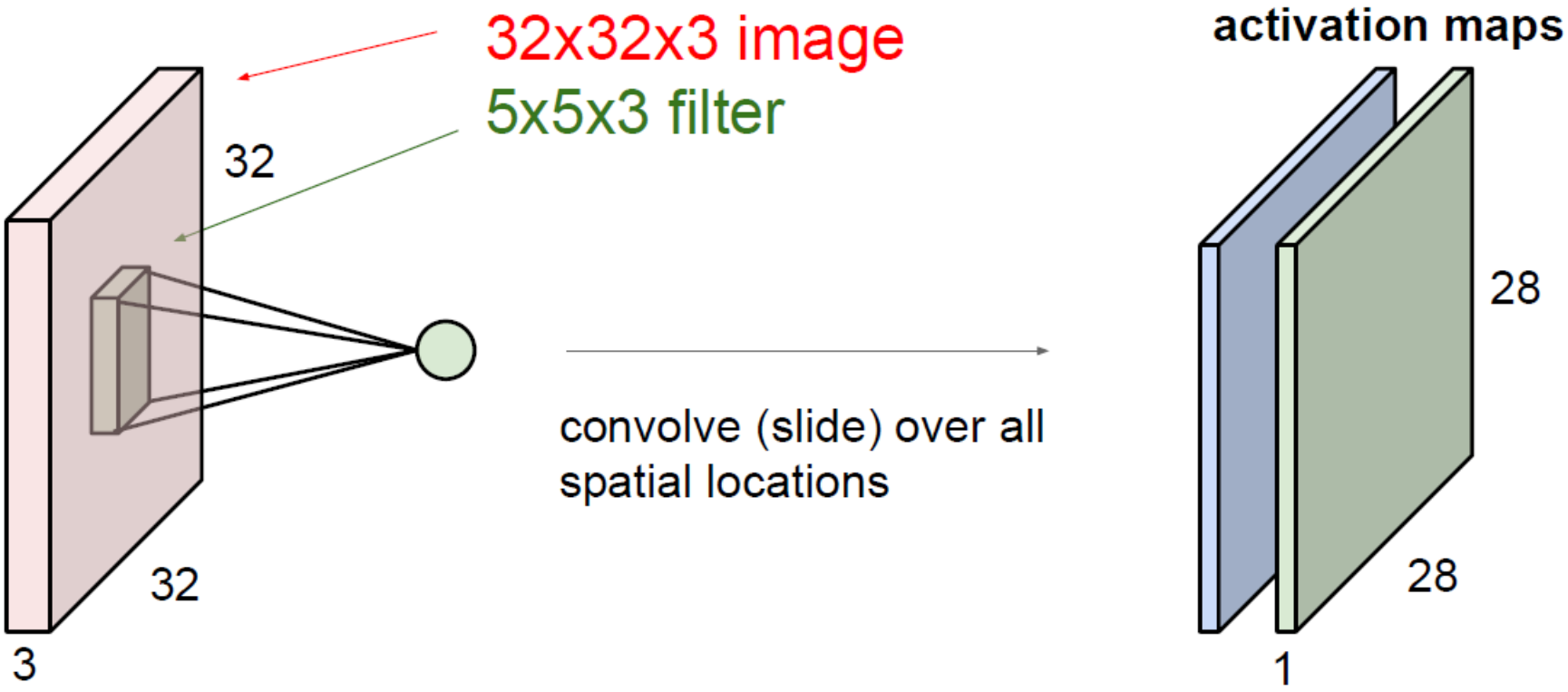




Convolution Layer

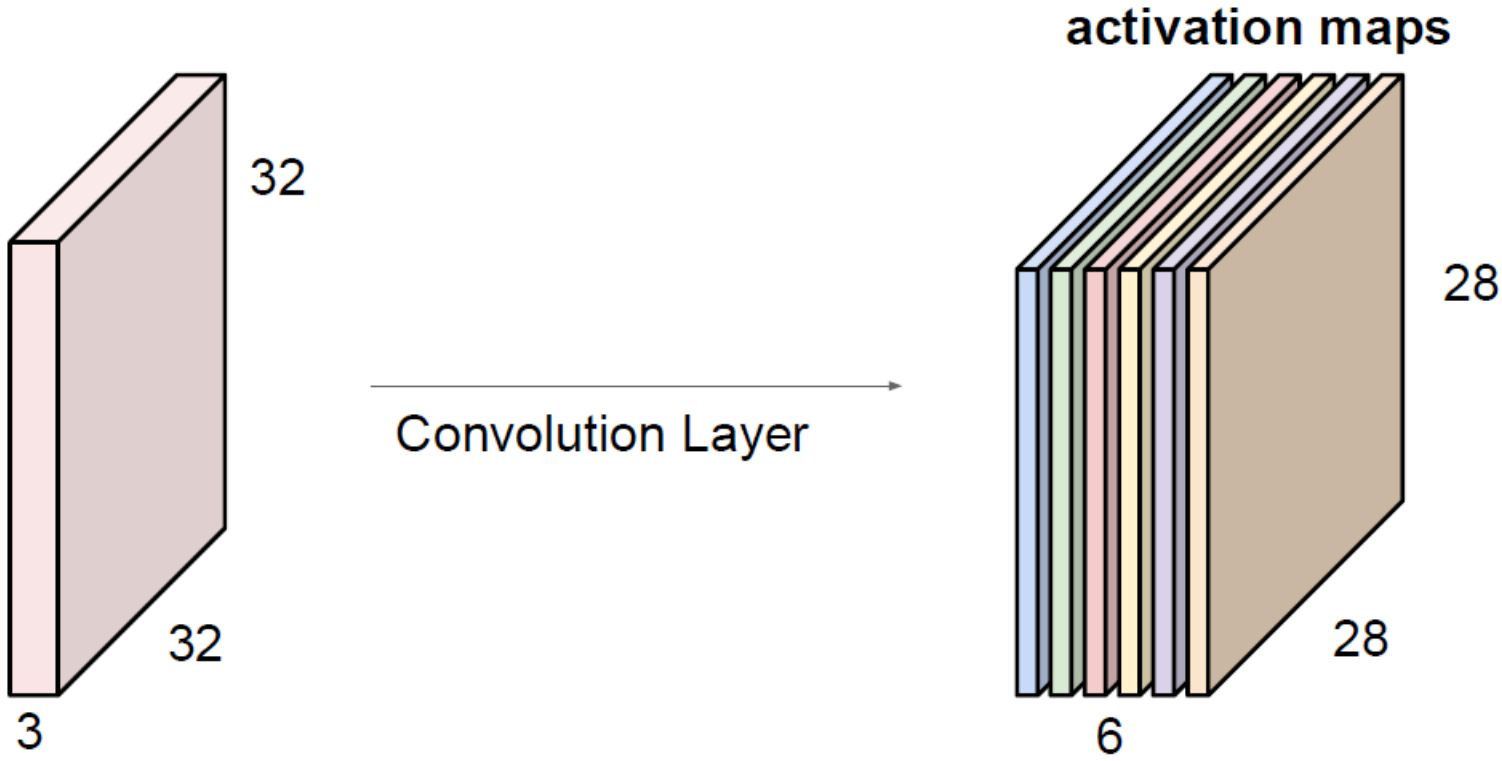


Convolution Layer



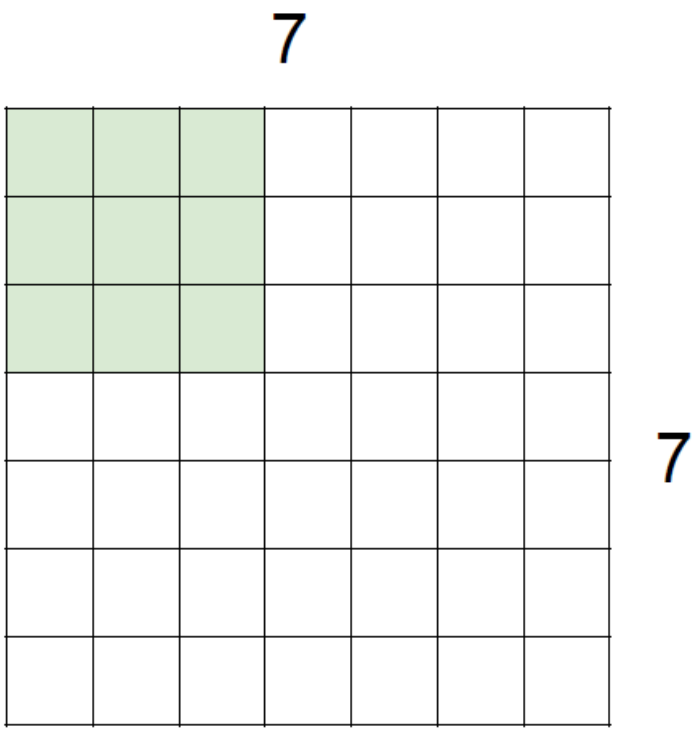
Convolution Layer

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



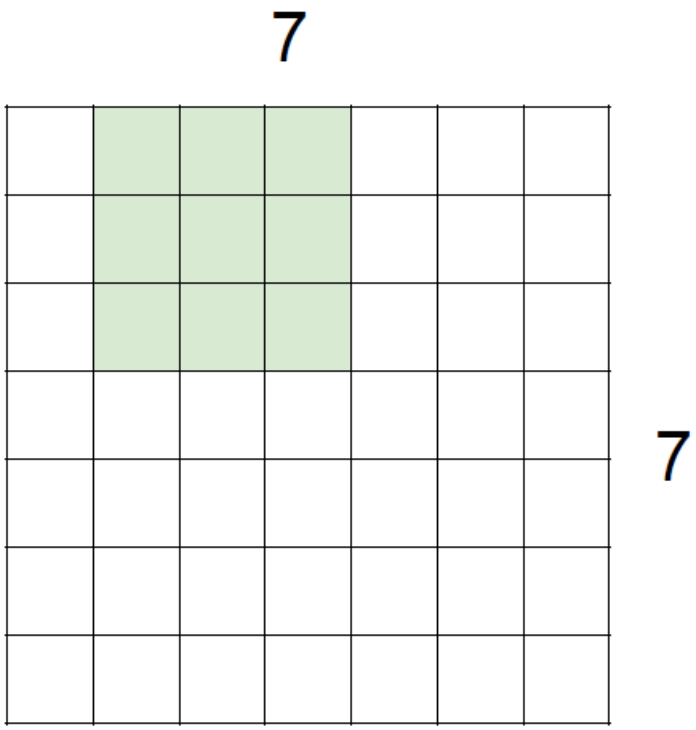
We stack these up to get a "new image" of size 28x28x6!

Convolution Layer: Spatial Dimensions



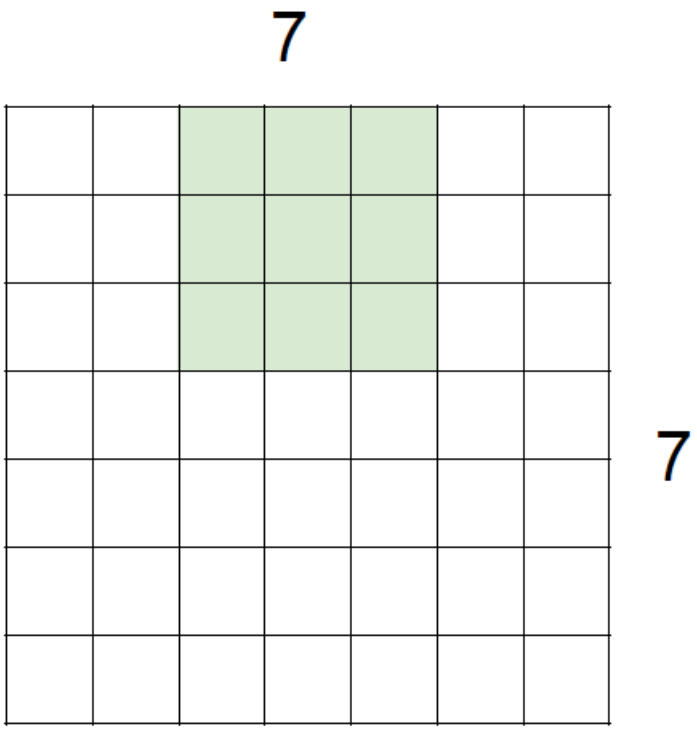
7x7 input (spatially)
assume 3x3 filter

Convolution Layer: Spatial Dimensions



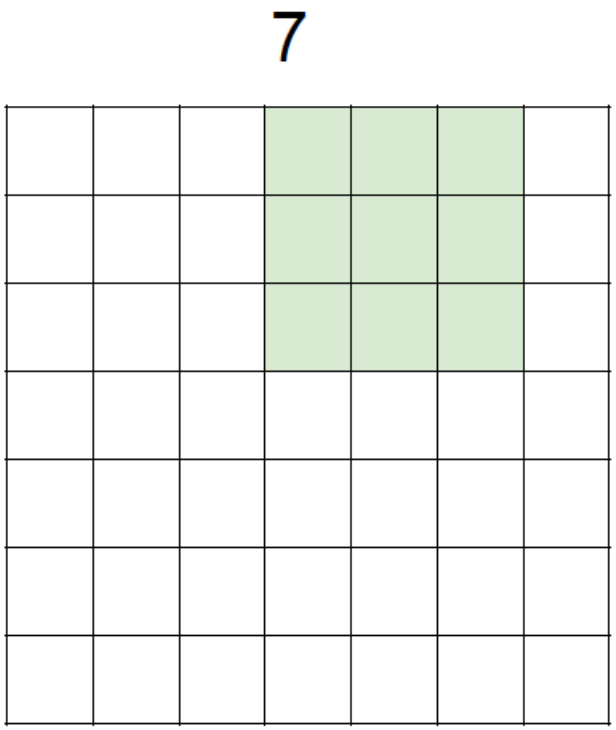
7x7 input (spatially)
assume 3x3 filter

Convolution Layer: Spatial Dimensions



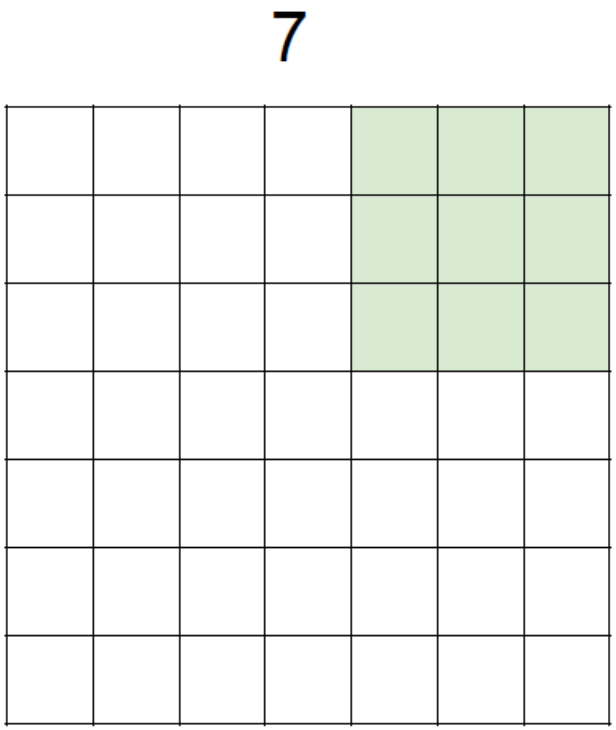
7x7 input (spatially)
assume 3x3 filter

Convolution Layer: Spatial Dimensions



7x7 input (spatially)
assume 3x3 filter

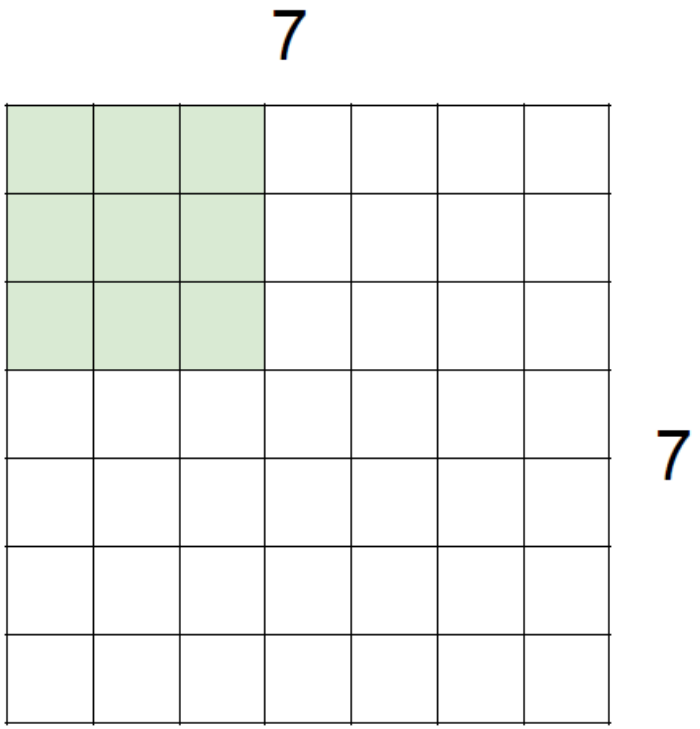
Convolution Layer: Spatial Dimensions



7x7 input (spatially)
assume 3x3 filter

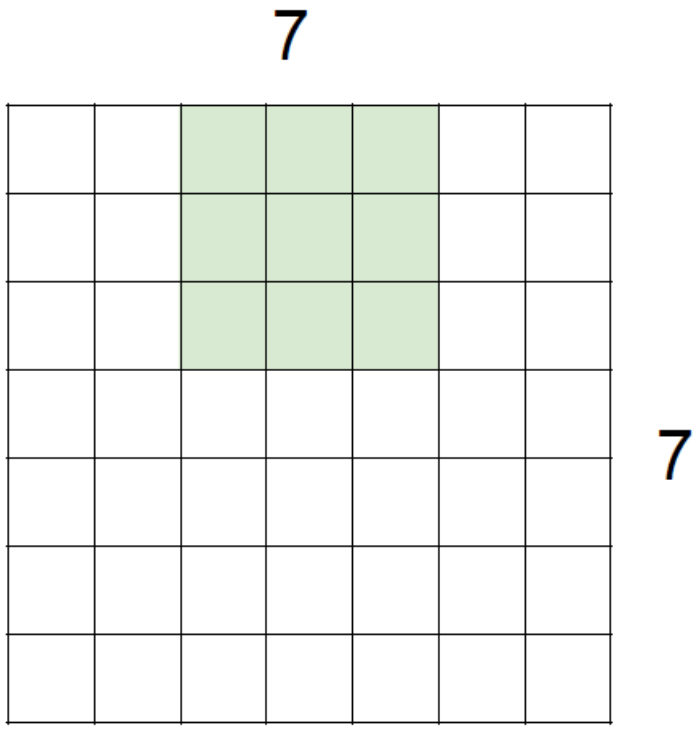
=> 5x5 output

Convolution Layer: Spatial Dimensions



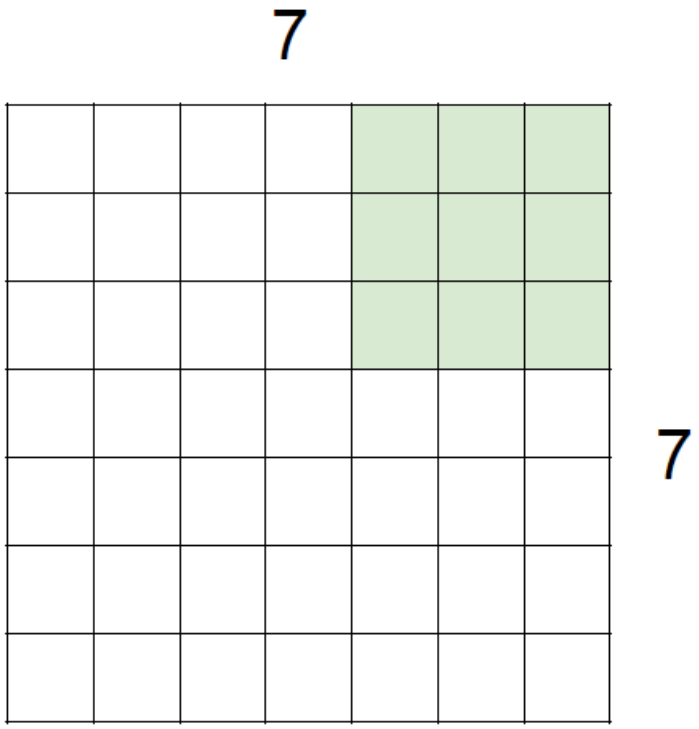
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Convolution Layer: Spatial Dimensions



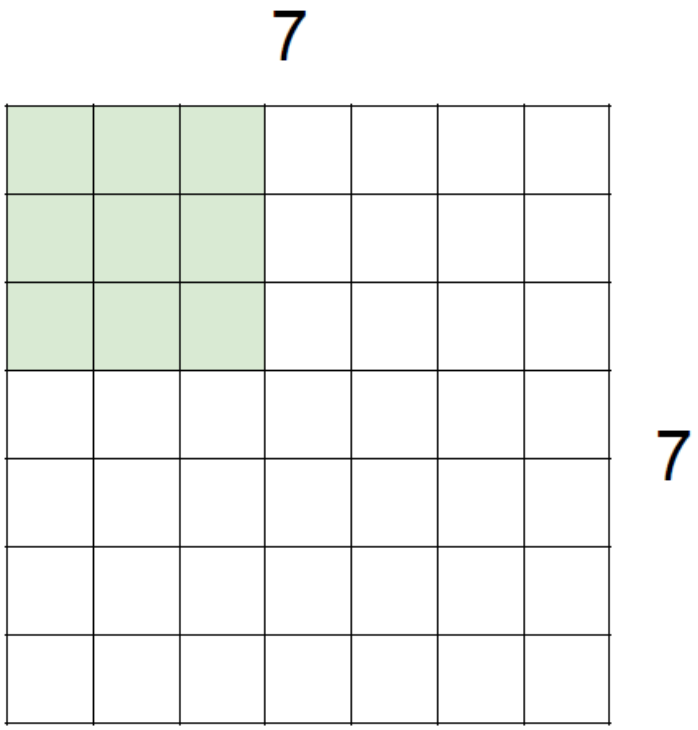
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Convolution Layer: Spatial Dimensions



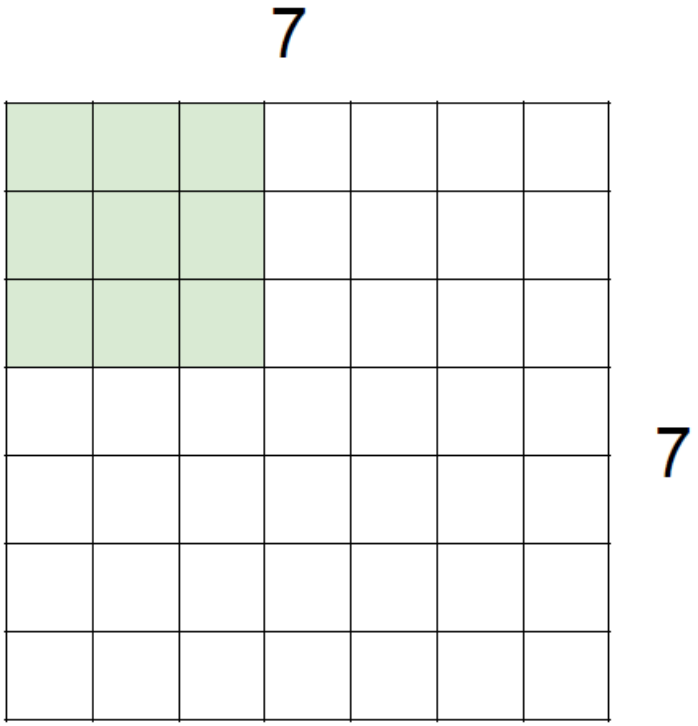
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

Convolution Layer: Spatial Dimensions



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

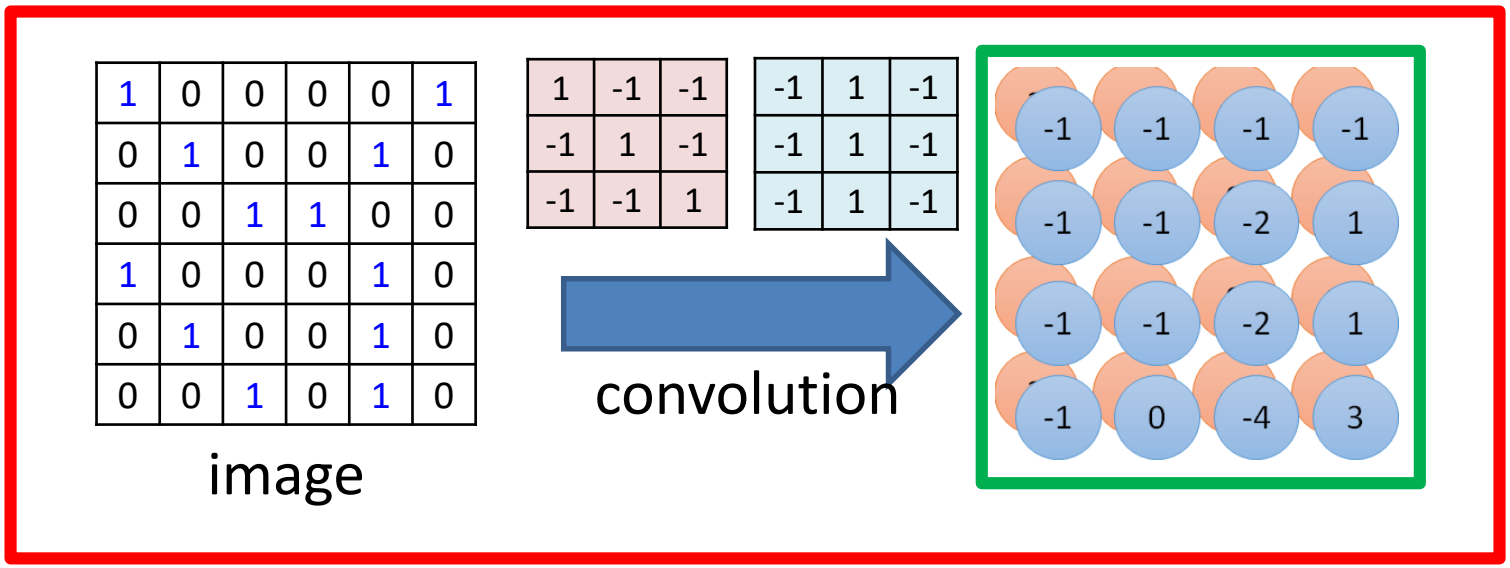
Convolution Layer: Spatial Dimensions



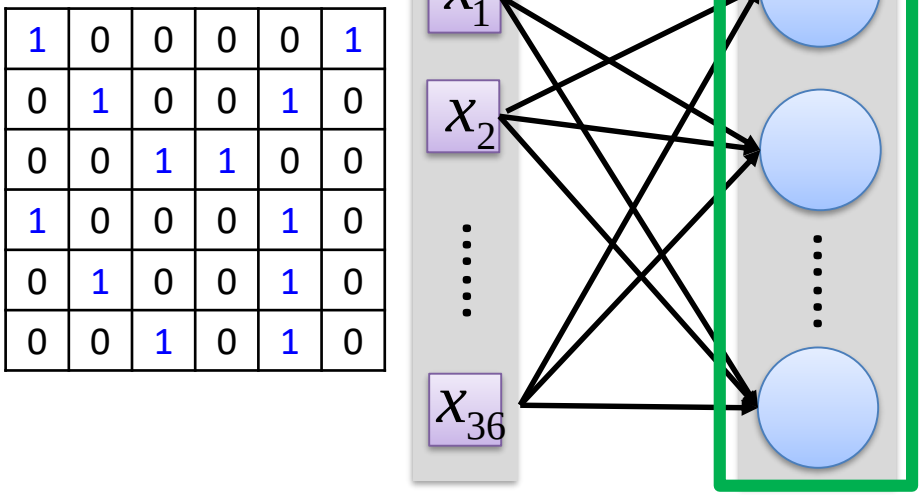
7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

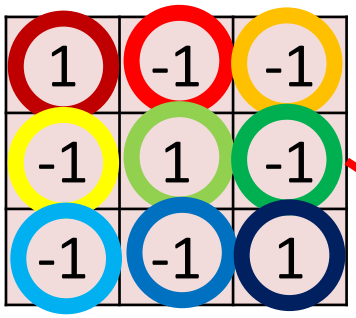
doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

Convolution v.s. Fully Connected



Fully-connected

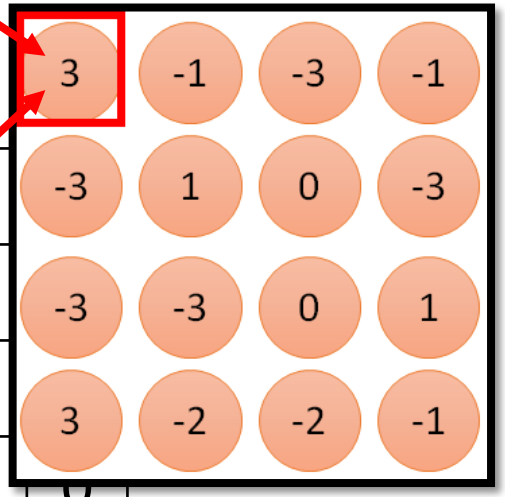




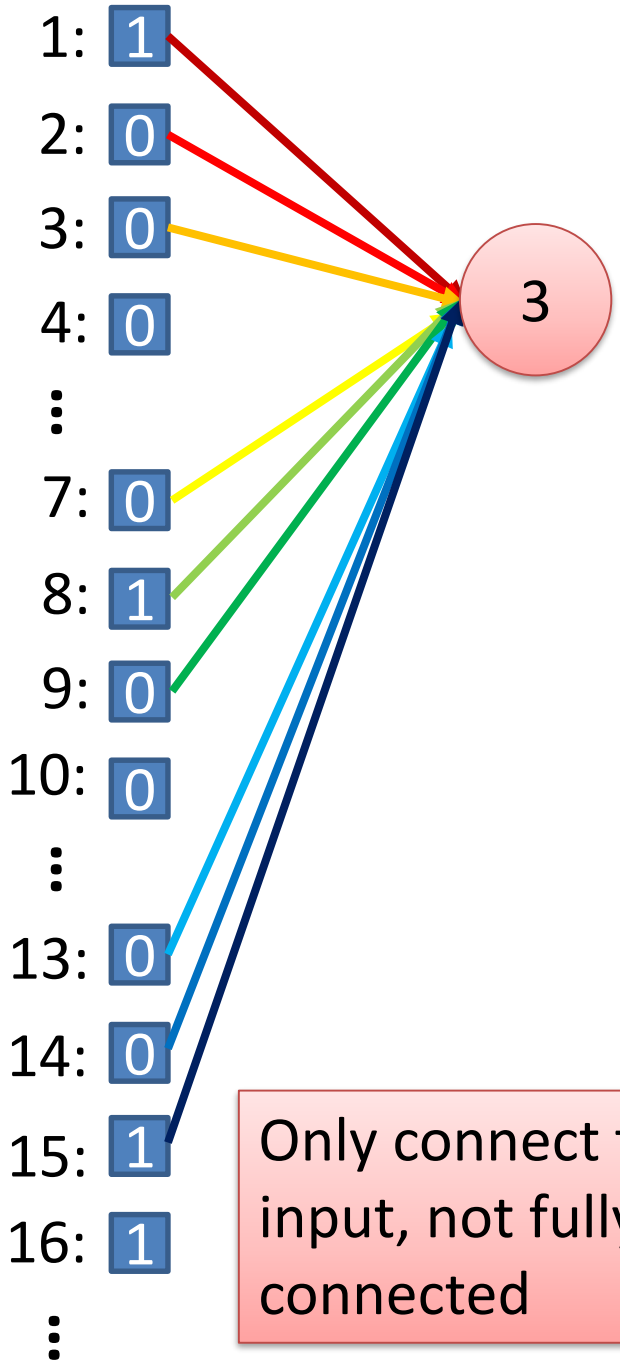
Filter 1

1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

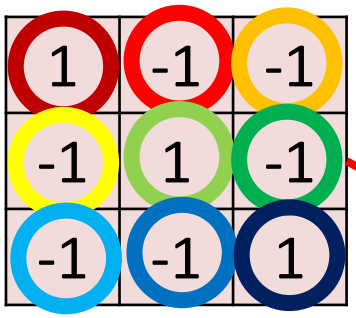
6 x 6 image



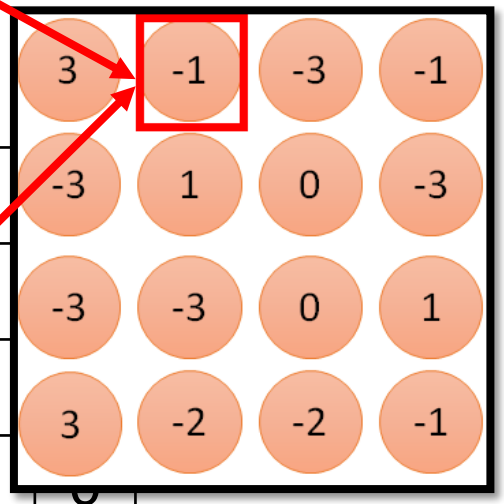
Less parameters!



Only connect to 9 input, not fully connected



Filter 1

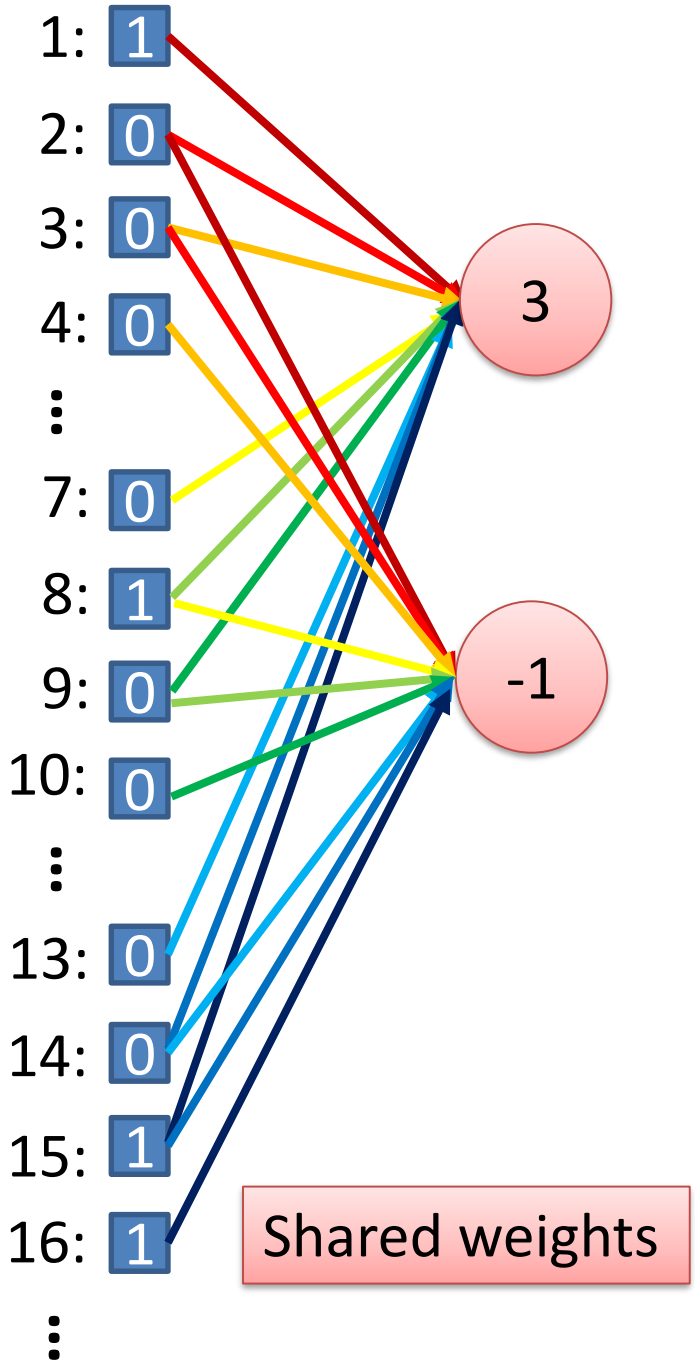


1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

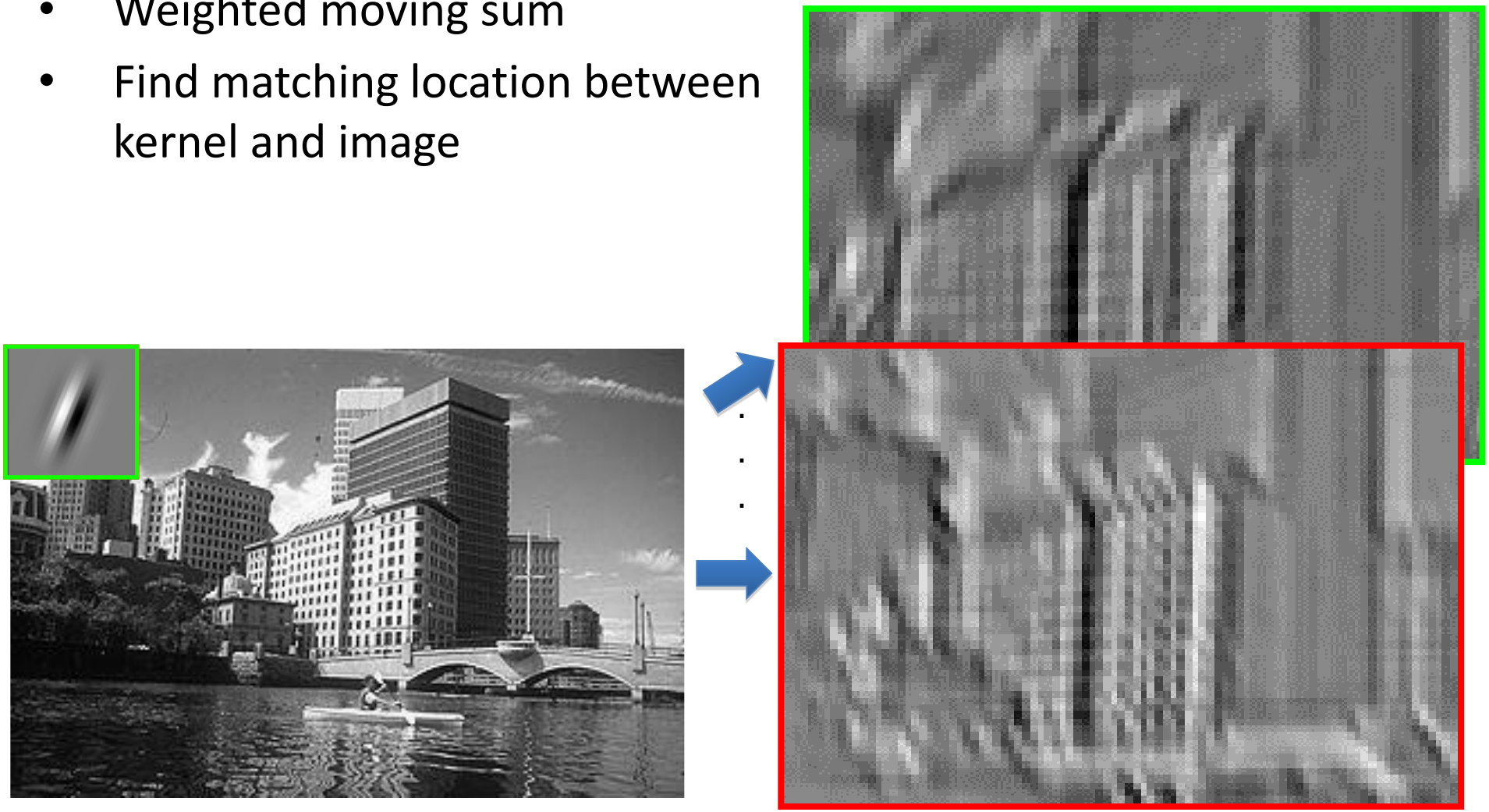
Less parameters!

Even less parameters!



What Does Convolution Do?

- Weighted moving sum
- Find matching location between kernel and image



Input

Feature Activation Map

slide credit: S. Lazebnik

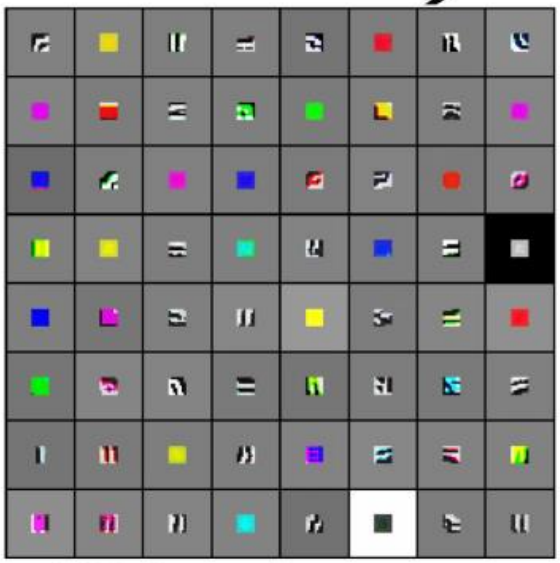
What Does Convolution Do?

Low-level features

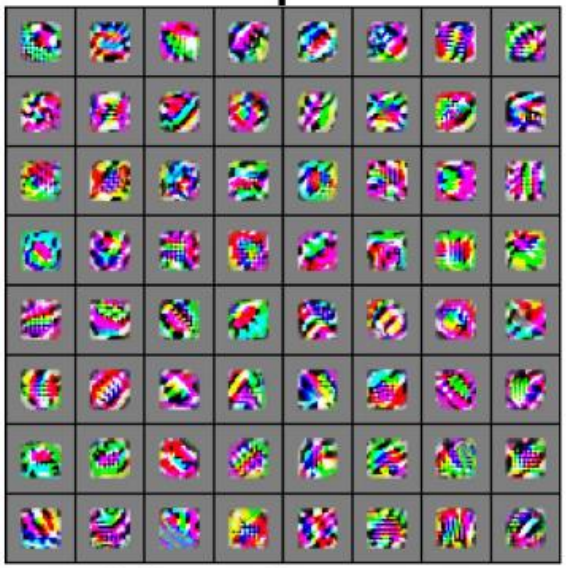
Mid-level features

High-level features

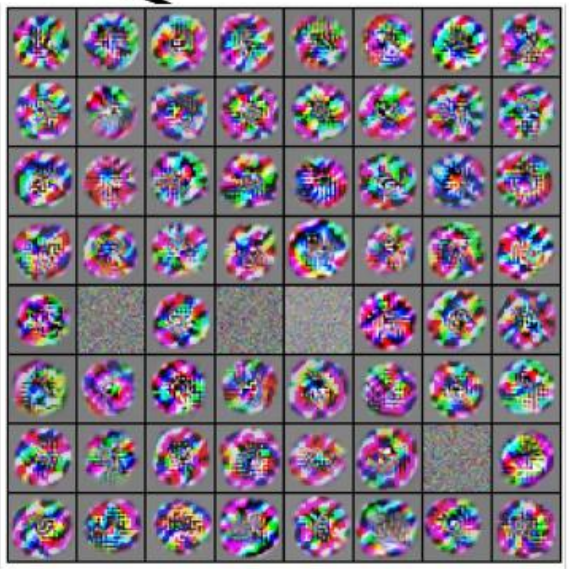
Linearly separable classifier



VGG-16 Conv1_1



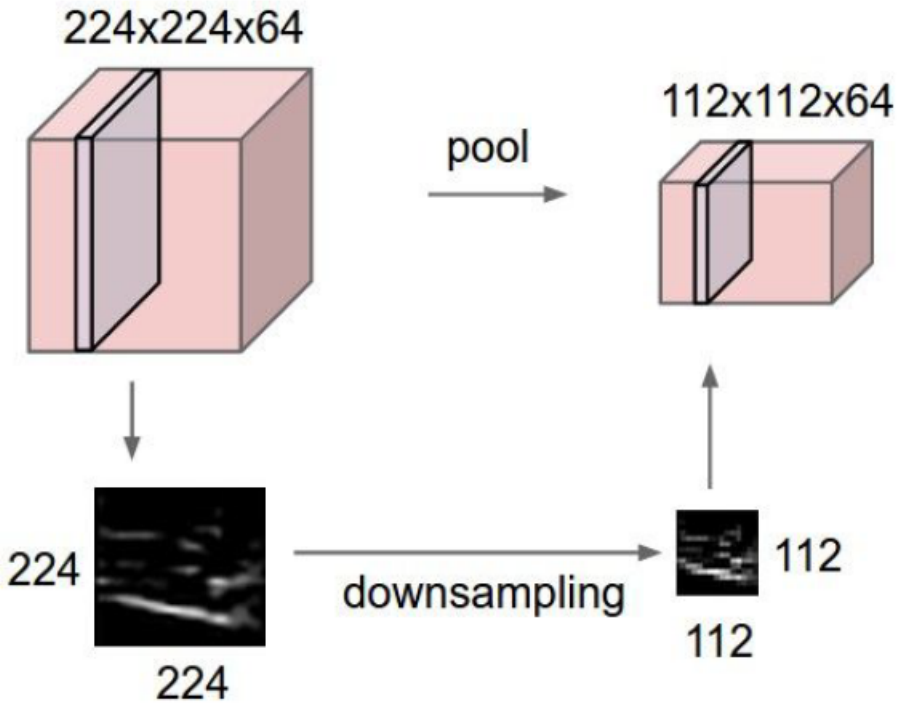
VGG-16 Conv3_2



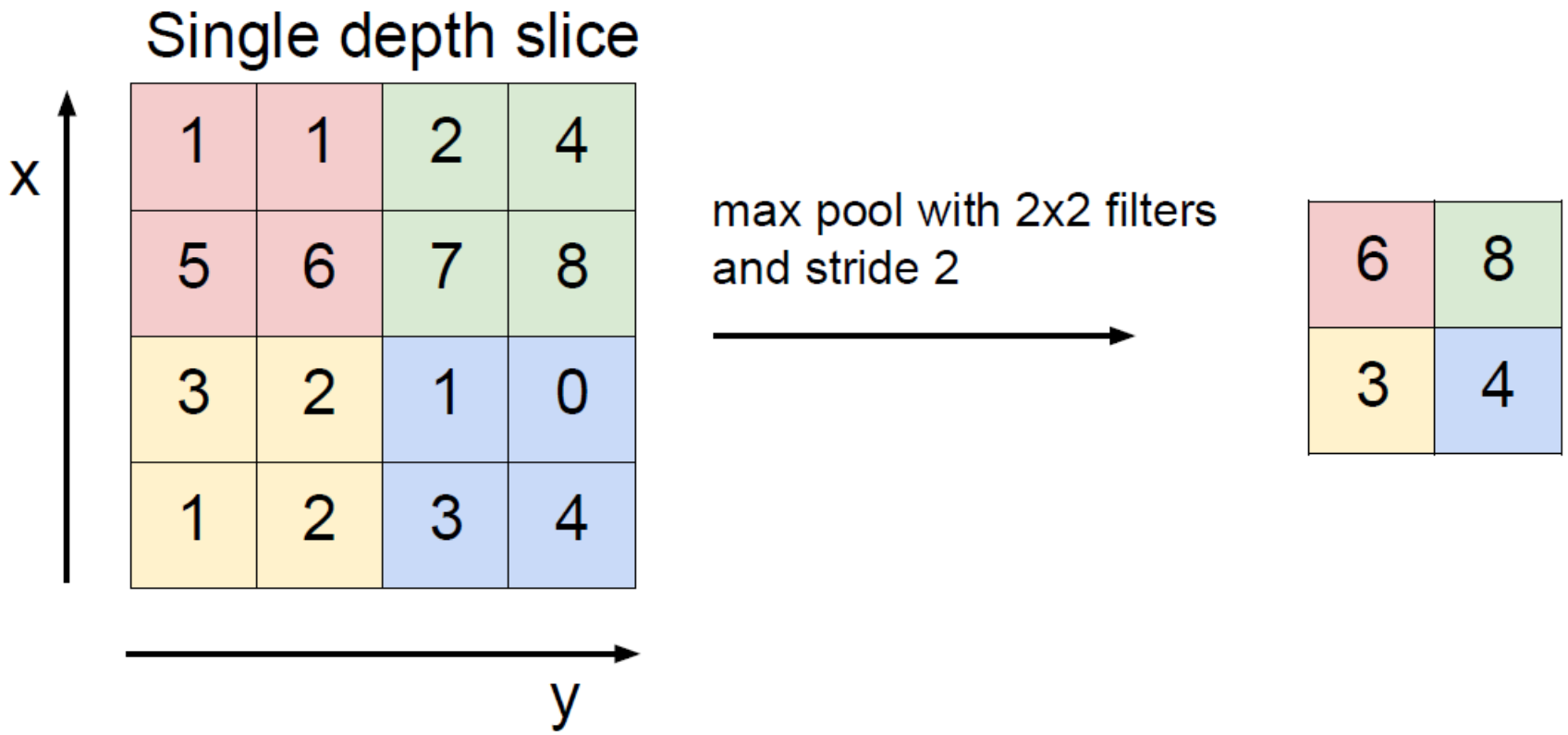
VGG-16 Conv5_3

Pooling Layer

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



Pooling Layer



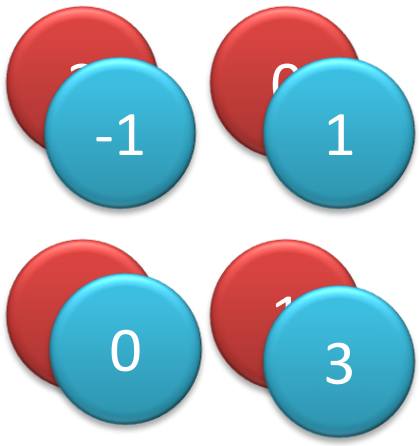
Pooling Layer

- The result of using a pooling layer and creating down sampled or pooled feature maps is a **summarized version** of the **features detected** in the input.
- They are useful as small changes in the location of the feature in the input detected by the convolutional layer will result in a pooled feature map with the feature in the same location.
- This capability added by pooling is called the model's **invariance to local translation**.

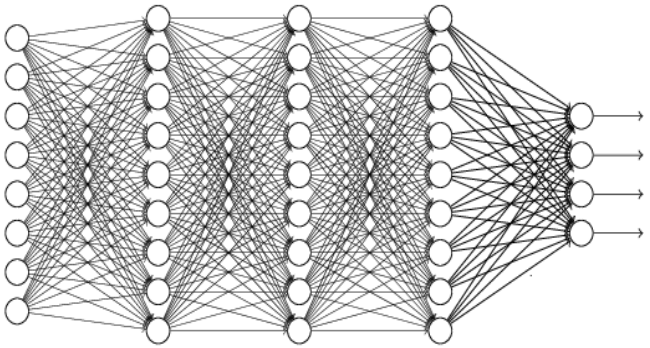
Pooling Layer

- *"In all cases, pooling helps to make the representation become approximately invariant to small translations of the input. Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change"*

Flatten



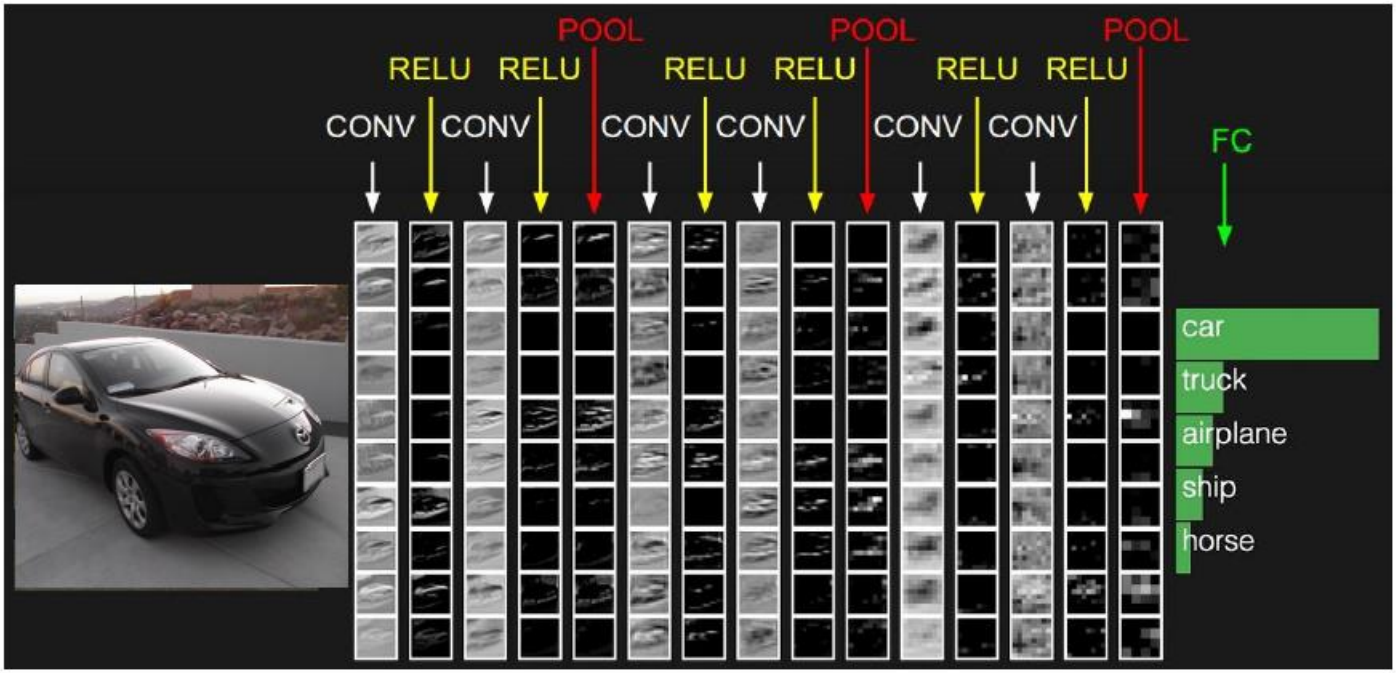
Flatten



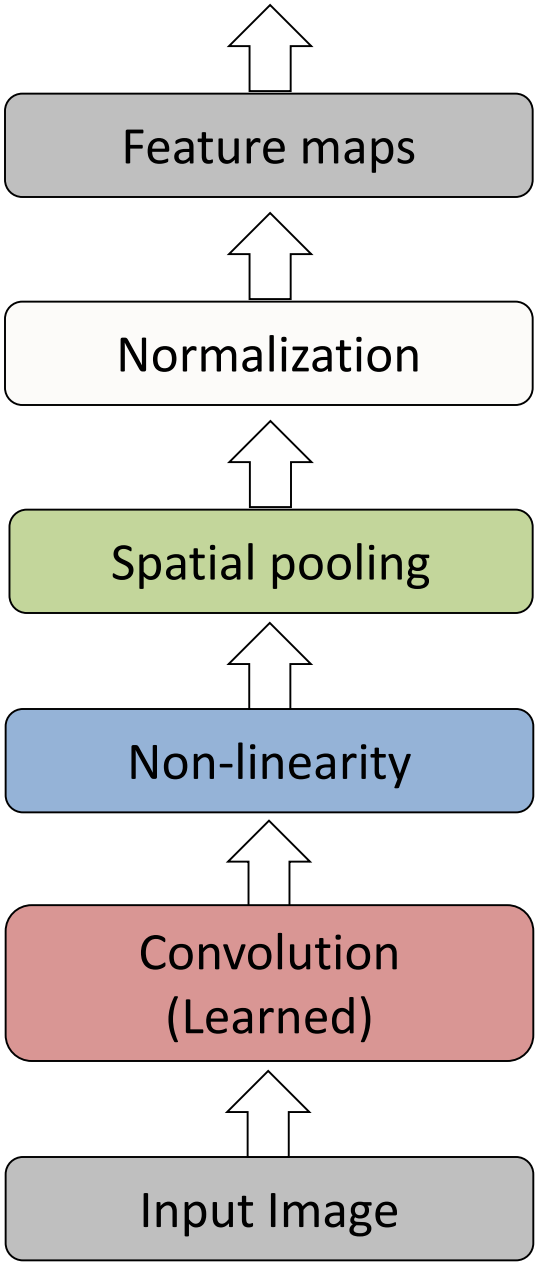
Fully Connected Feedforward network

Fully Connected Layer

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks

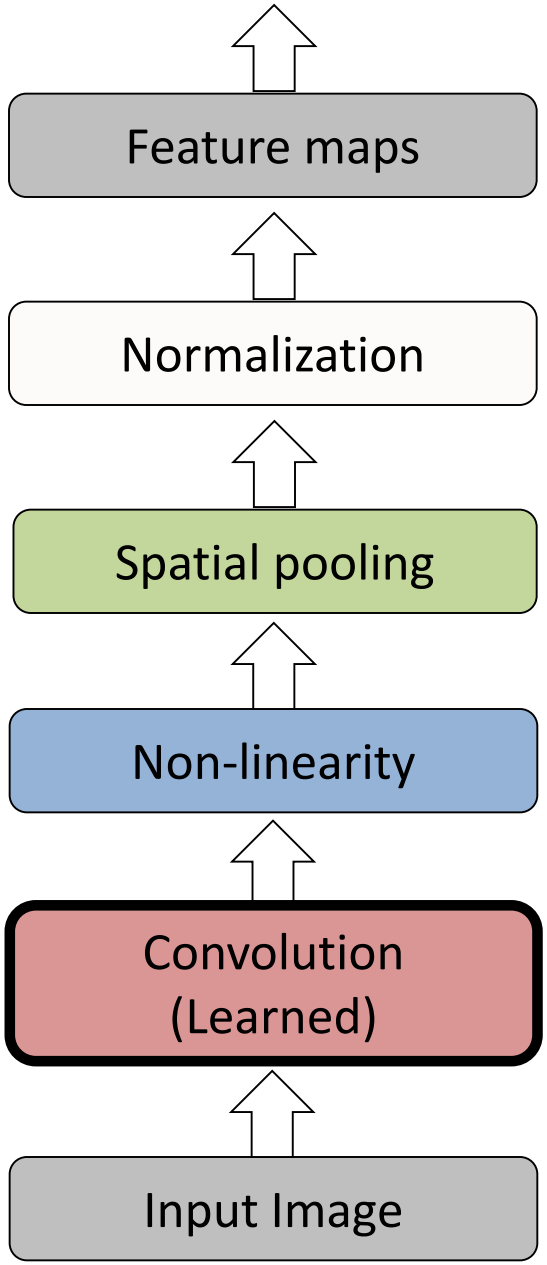


Convolutional Neural Networks



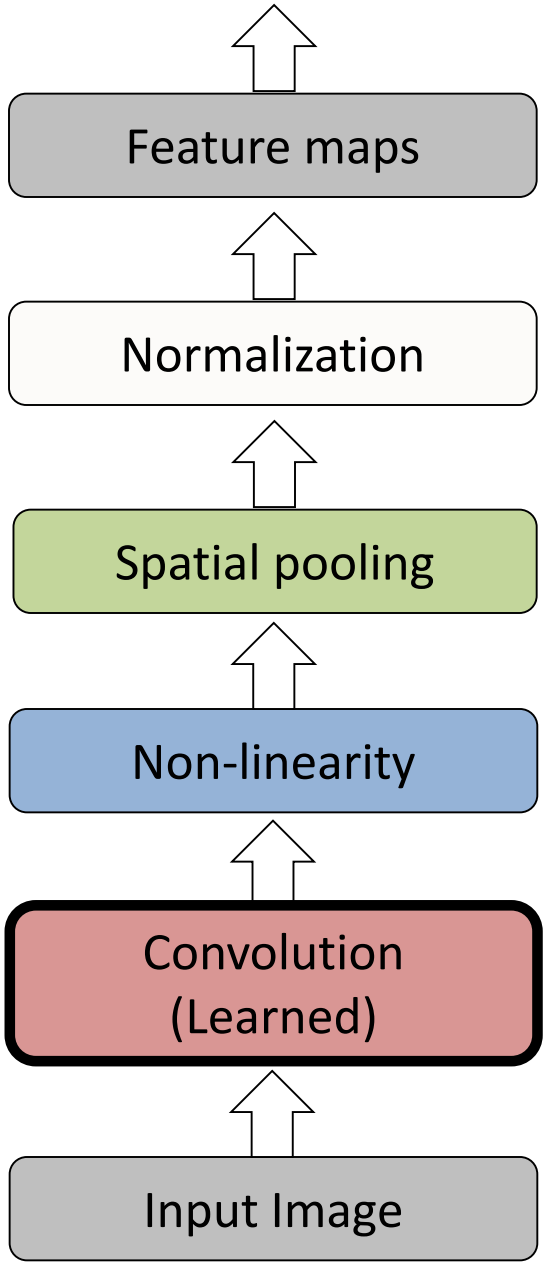
slide credit: S. Lazebnik

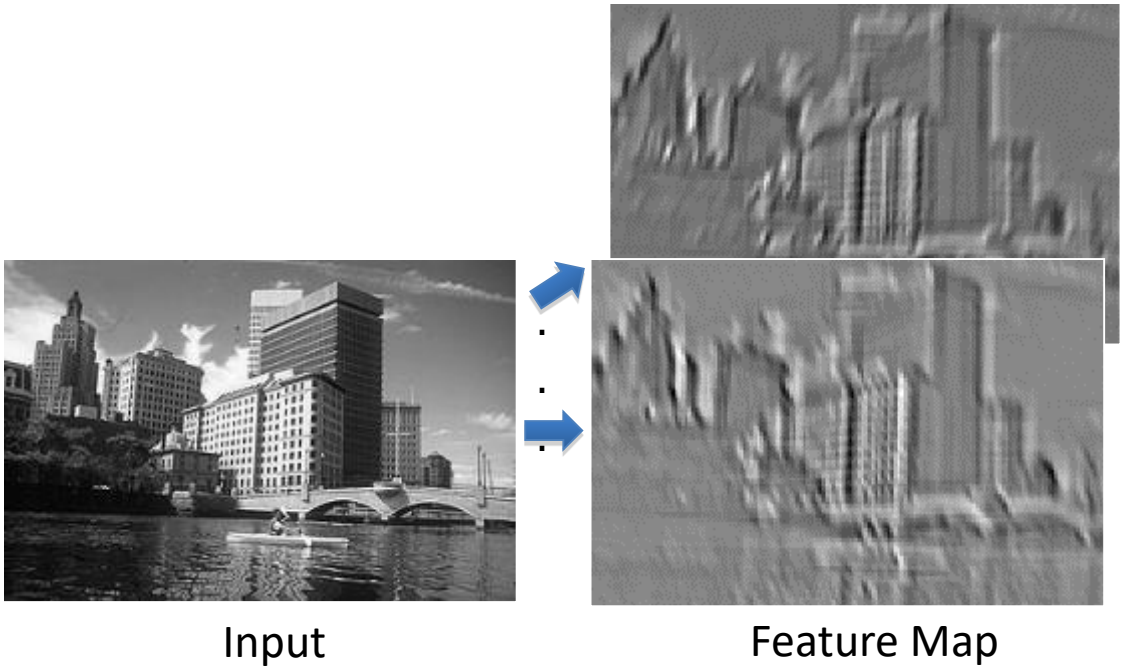
Convolutional Neural Networks



slide credit: S. Lazebnik

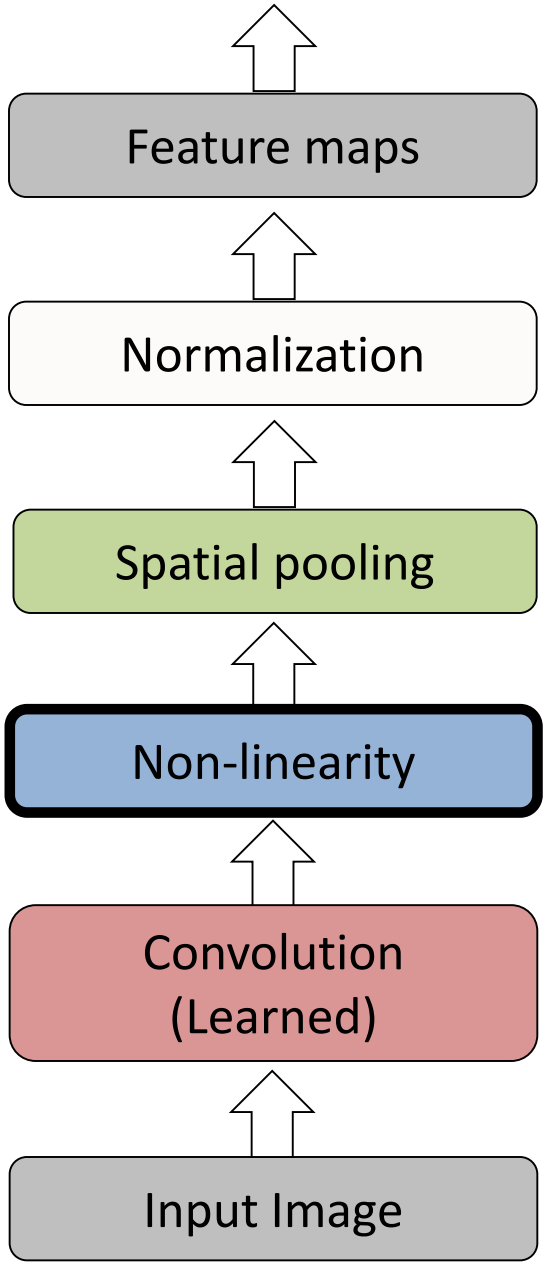
Convolutional Neural Networks





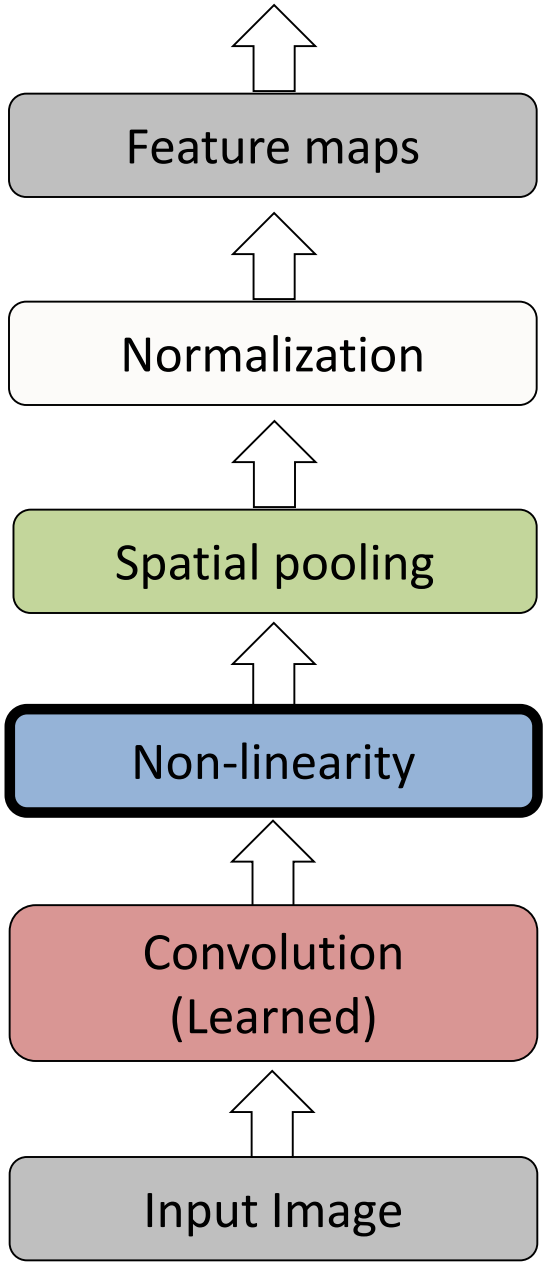
slide credit: S. Lazebnik

Convolutional Neural Networks

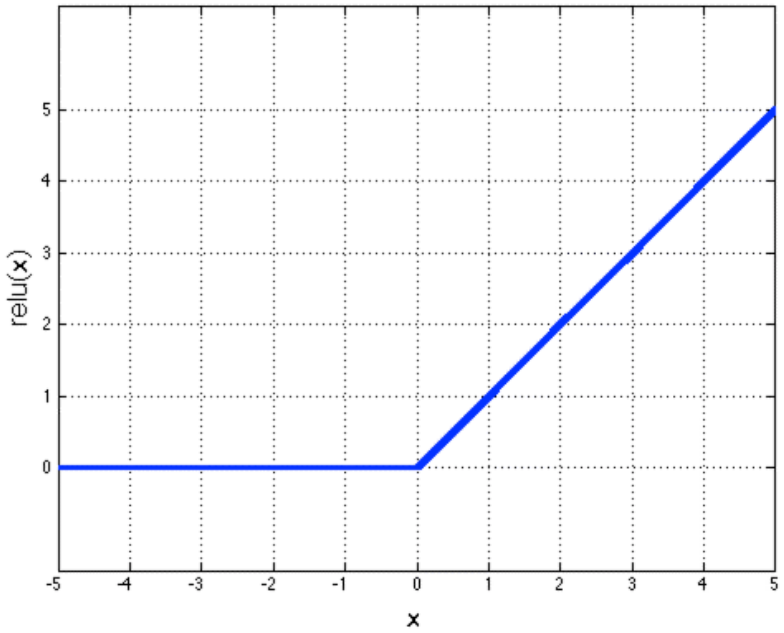


slide credit: S. Lazebnik

Convolutional Neural Networks

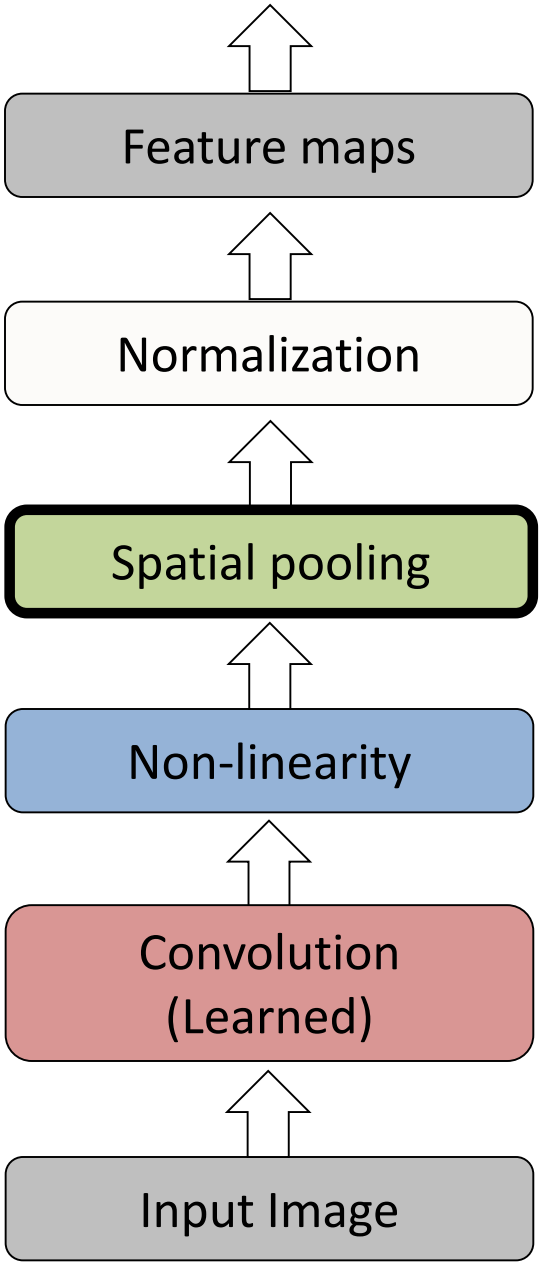


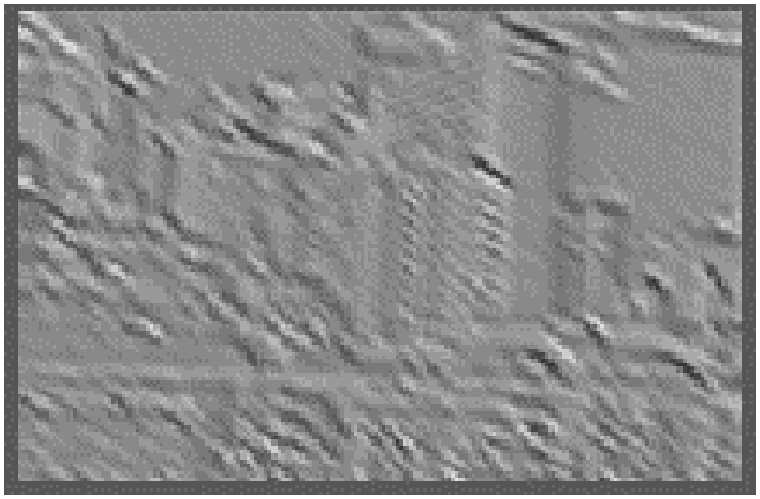
Rectified Linear Unit (ReLU)



slide credit: S. Lazebnik

Convolutional Neural Networks

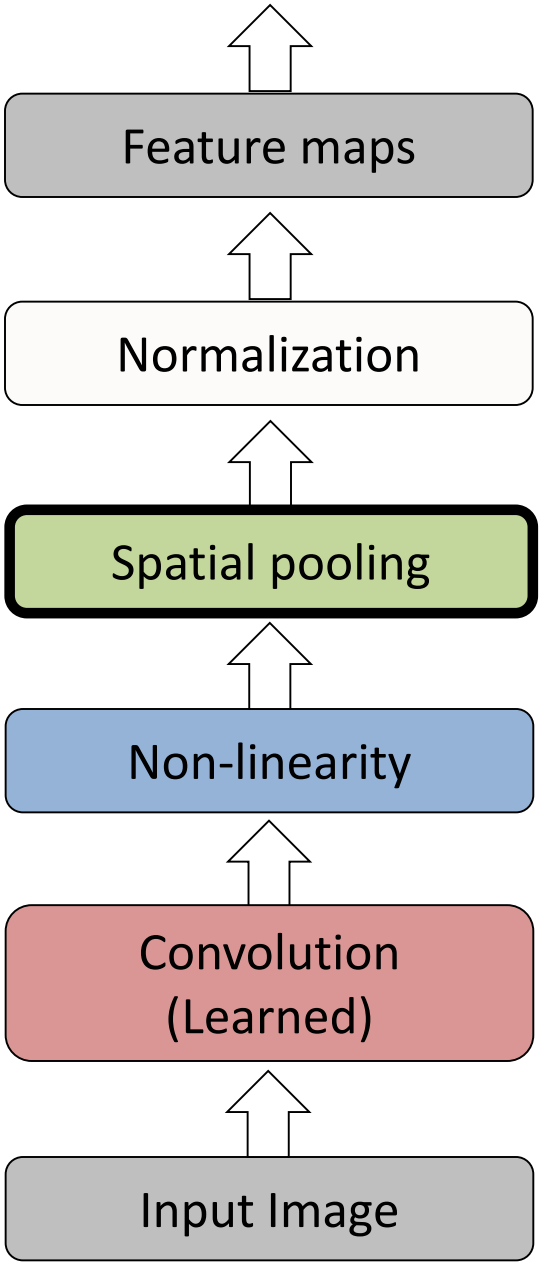


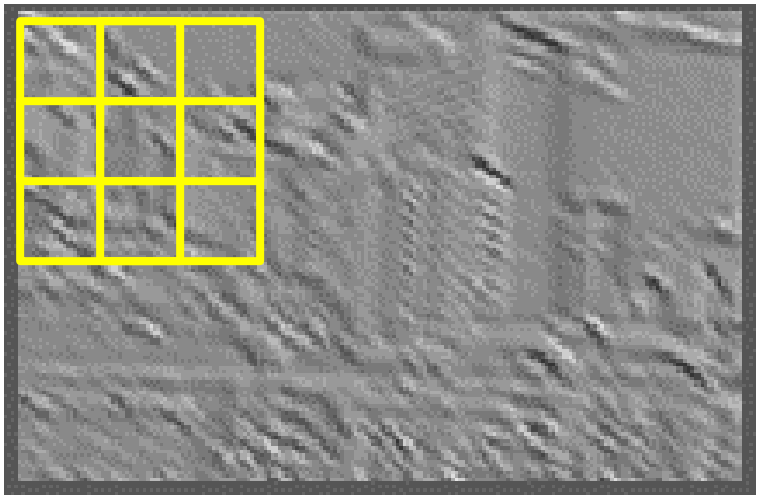


Max-pooling: a non-linear down-sampling

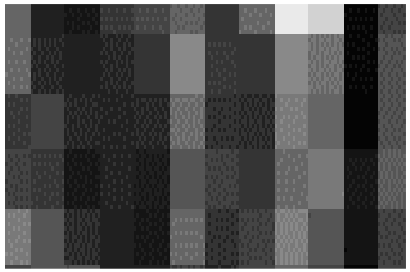
slide credit: S. Lazebnik

Convolutional Neural Networks





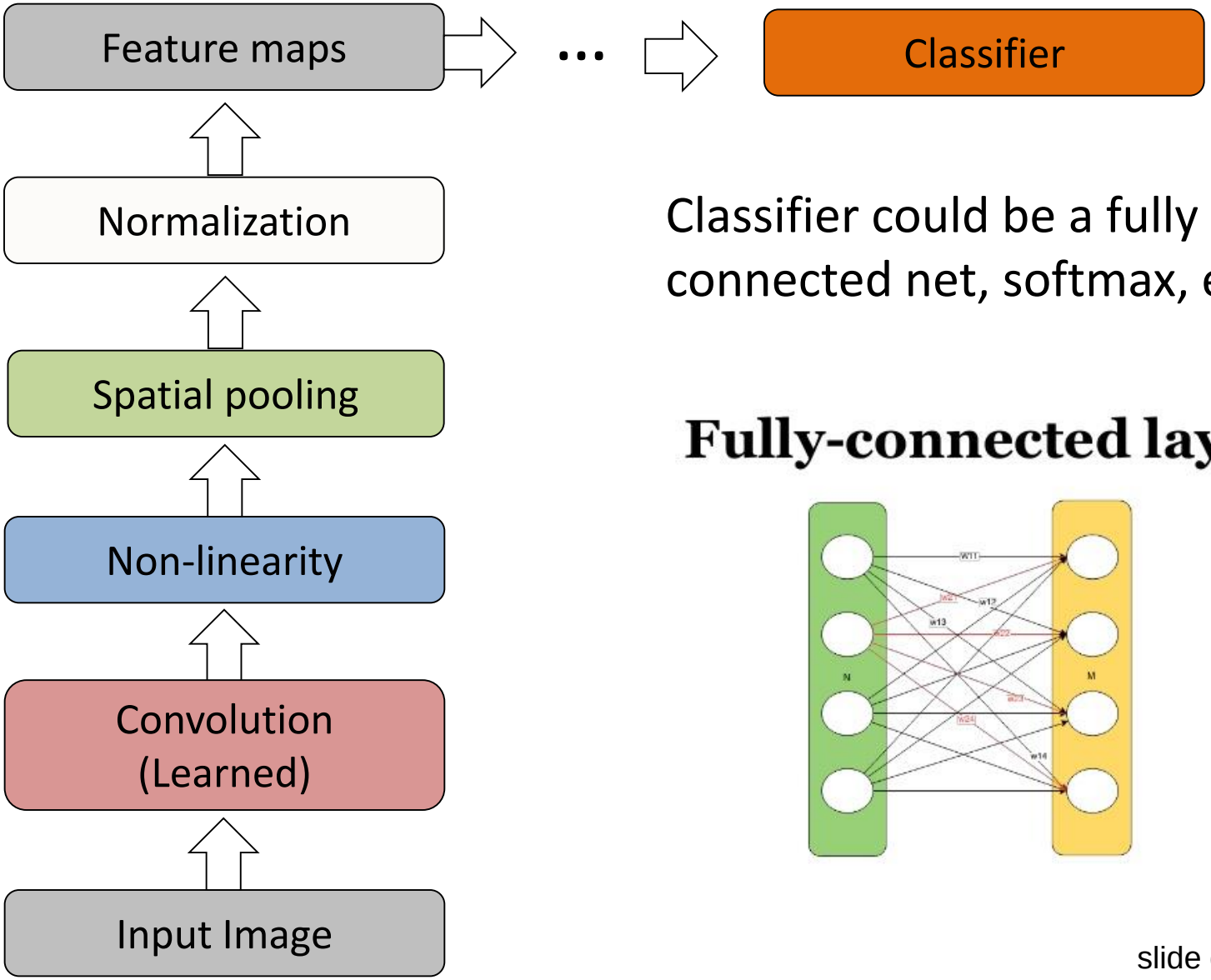
Max pooling



Max-pooling: a non-linear down-sampling

slide credit: S. Lazebnik

Convolutional Neural Networks

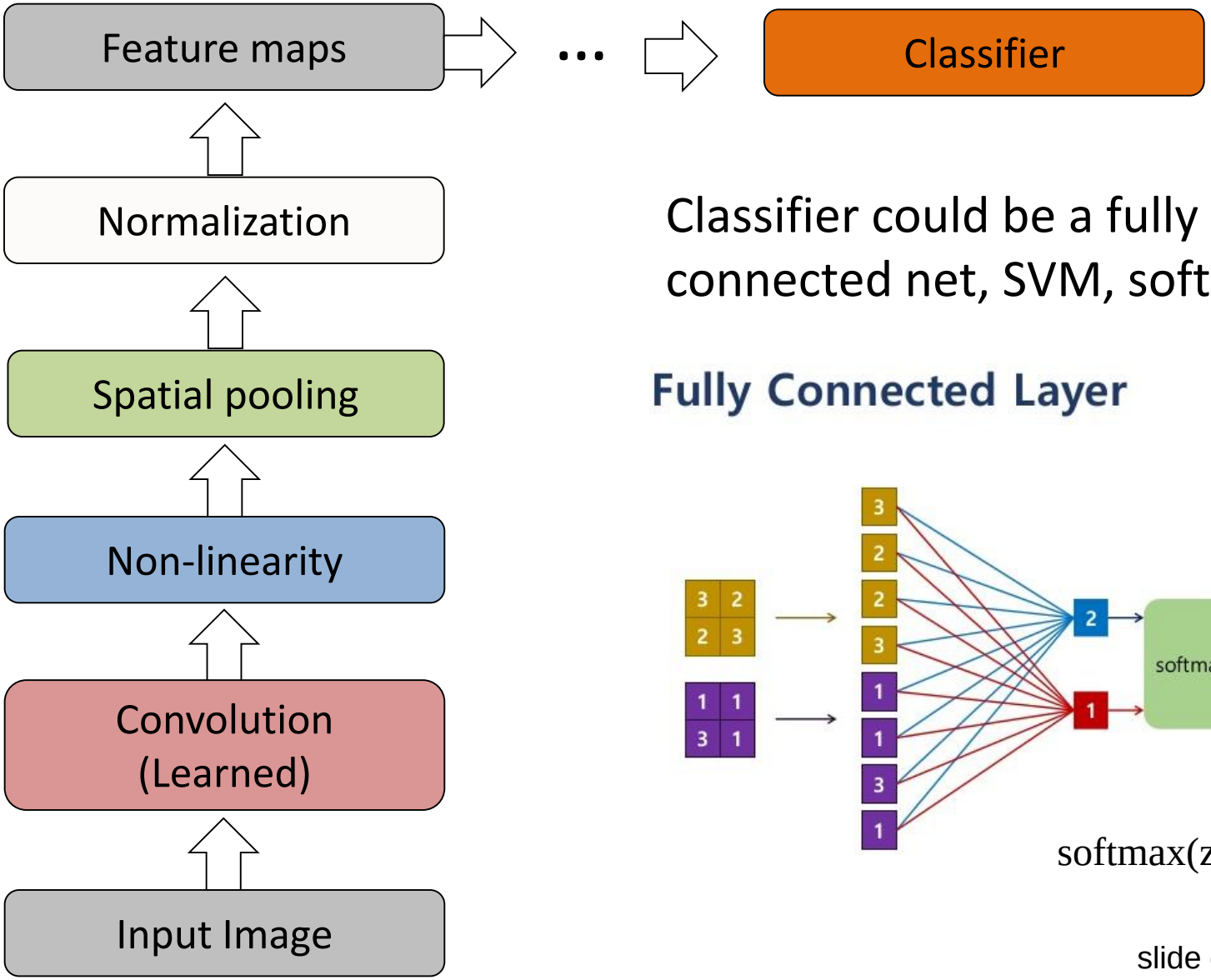


Classifier could be a fully connected net, softmax, etc.

Fully-connected layer

slide credit: S. Lazebnik

Convolutional Neural Networks



Classifier could be a fully connected net, SVM, softmax, etc.

slide credit: S. Lazebnik

LeNet (1989 - 1998)

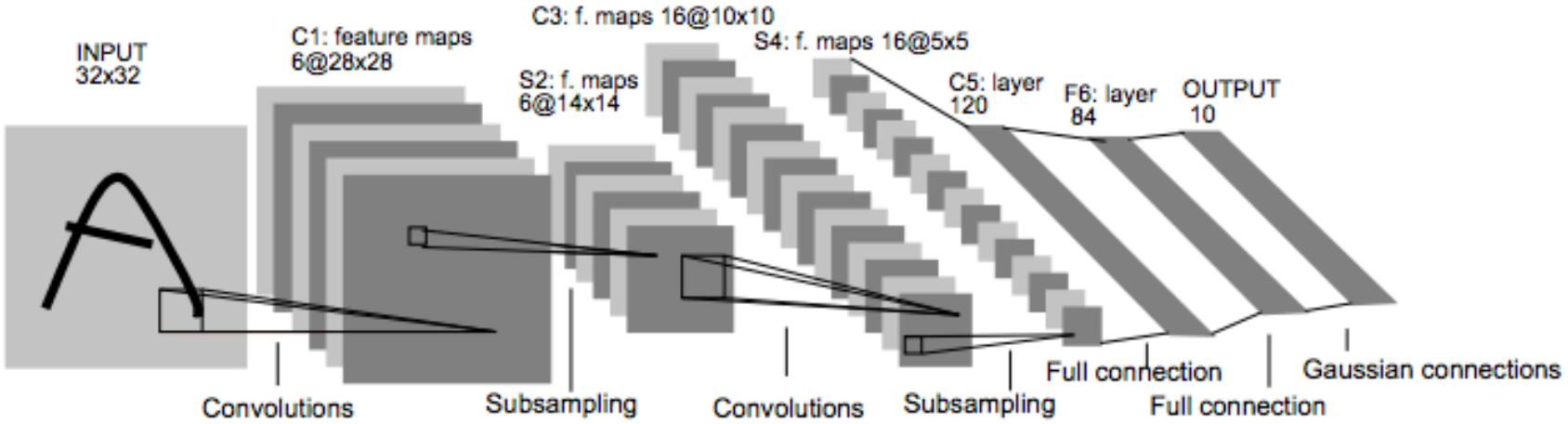
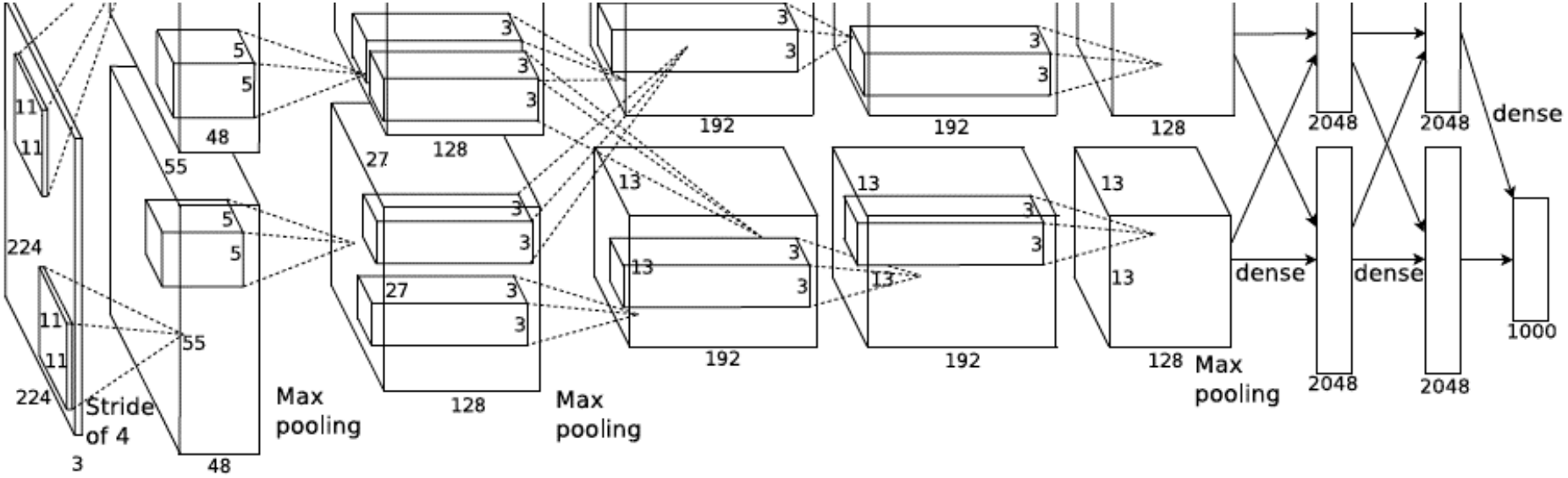


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

http://yann.lecun.com/exdb/lenet/

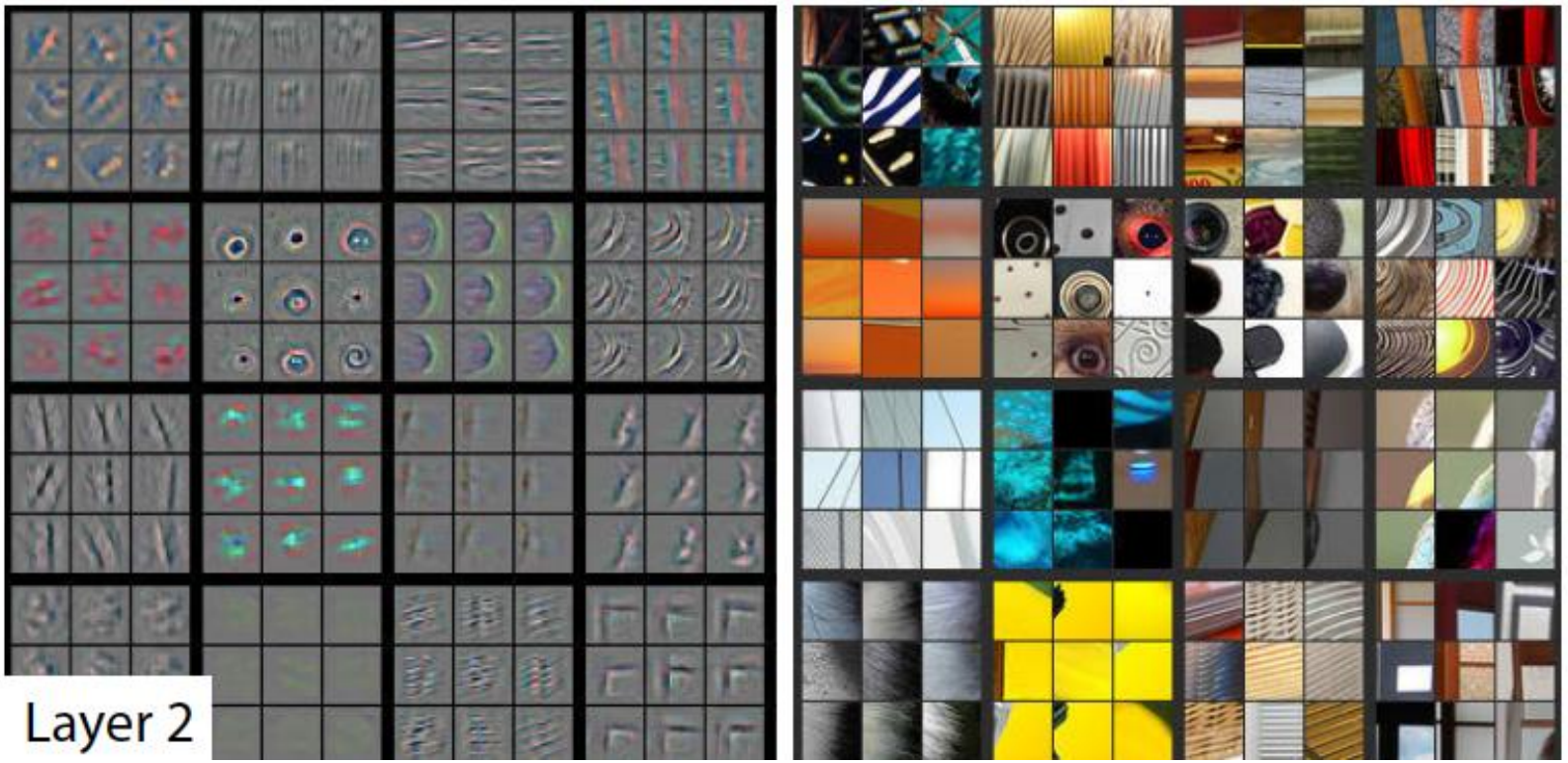
AlexNet

- Similar framework to LeCun'98 but:
  - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
  - More data (10^6 vs. 10^3 images)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week



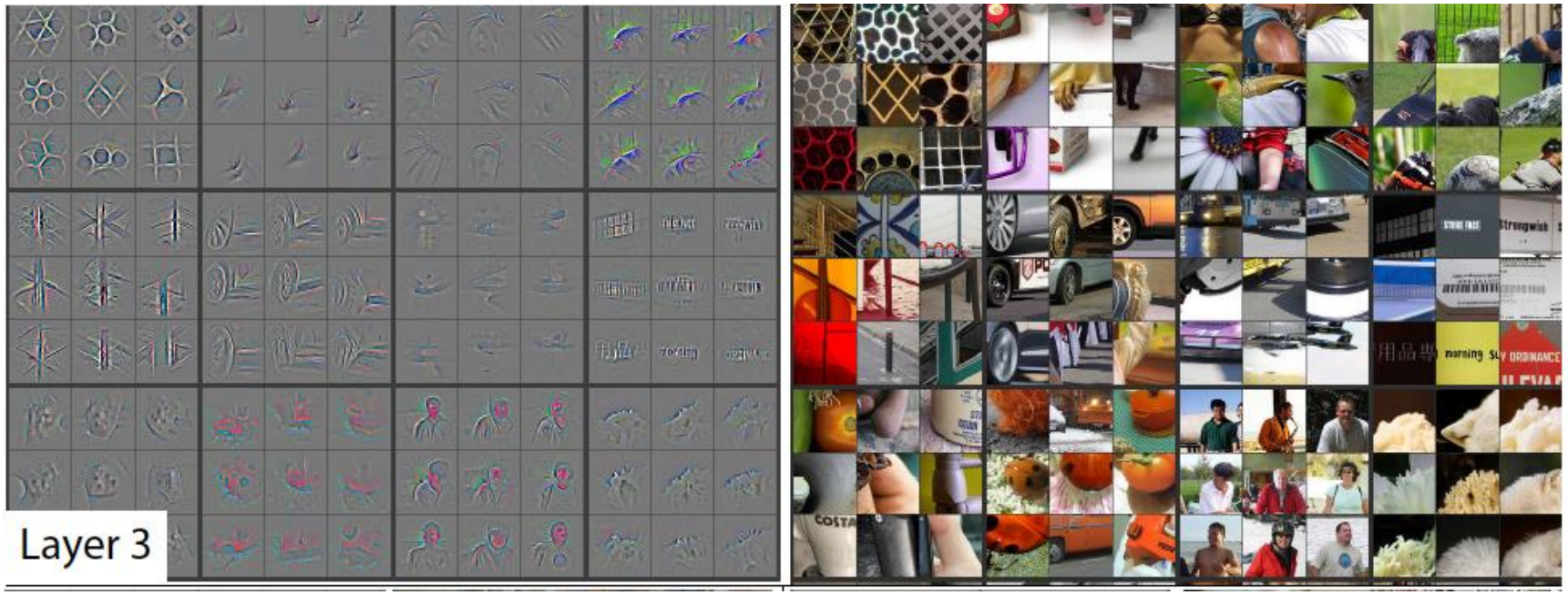
A. Krizhevsky, I. Sutskever, and G. Hinton,
ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Visualising Learned Filters

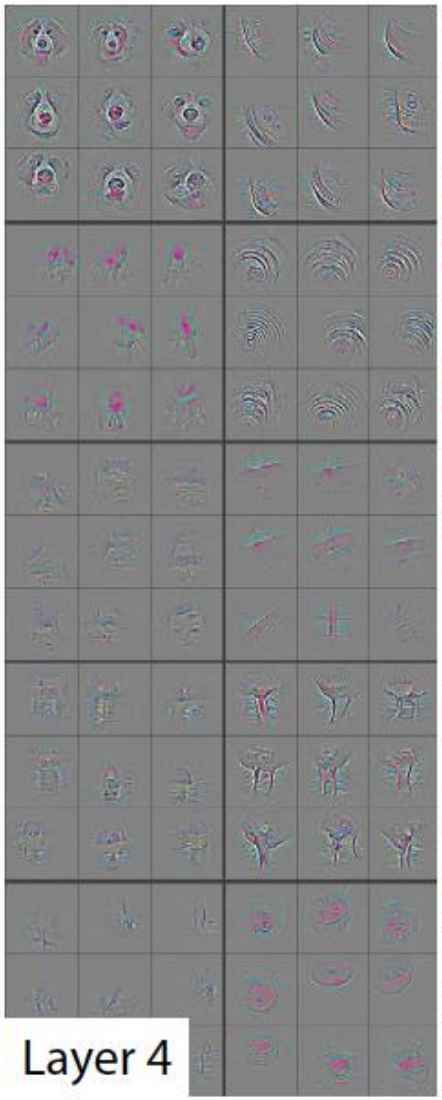


- Top 9 activations in a randomly selected set of (16) feature maps, projected down to previous level
- The image patches that generated those activations

Visualising Learned Features



Visualising Learned Features



Applications

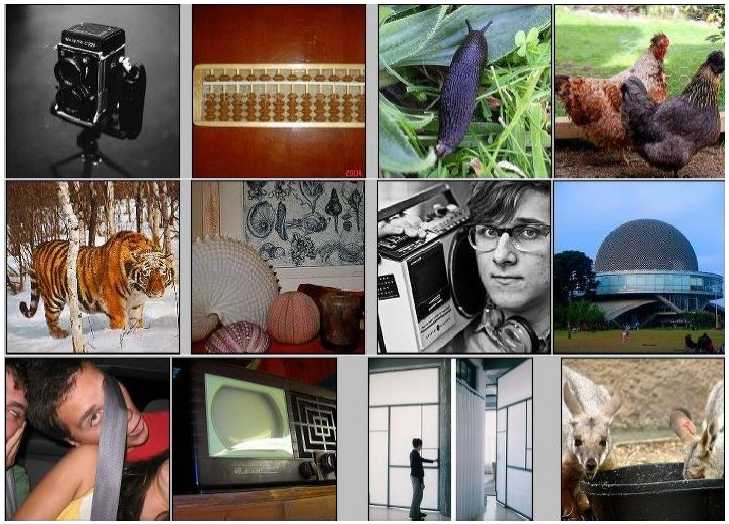
- Handwritten text/digits
  - MNIST (0.17% error [Ciresan et al. 2011])
  - Arabic & Chinese [Ciresan et al. 2012]

- Simpler recognition benchmarks
  - CIFAR-10 (9.3% error [Wan et al. 2013])
  - Traffic sign recognition
    - 0.56% error vs 1.16% for humans [Ciresan et al. 2011]

Slide: R. Fergus

Application: ImageNet

IMAGENET



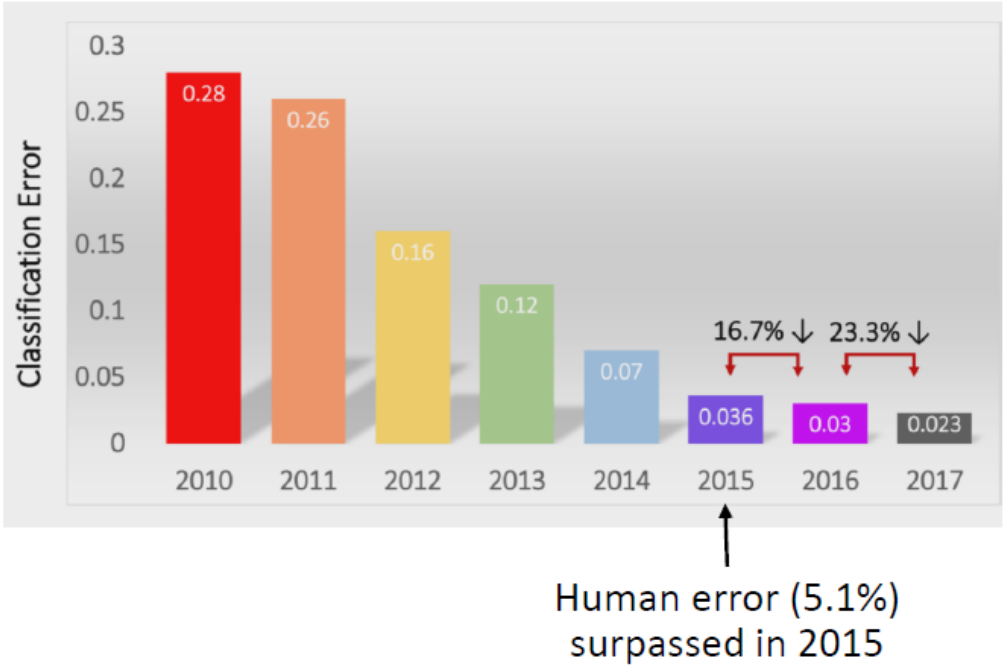
[Deng et al. CVPR 2009]

- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon Turk

https://sites.google.com/site/deeplearningcvpr2014

Slide: R. Fergus

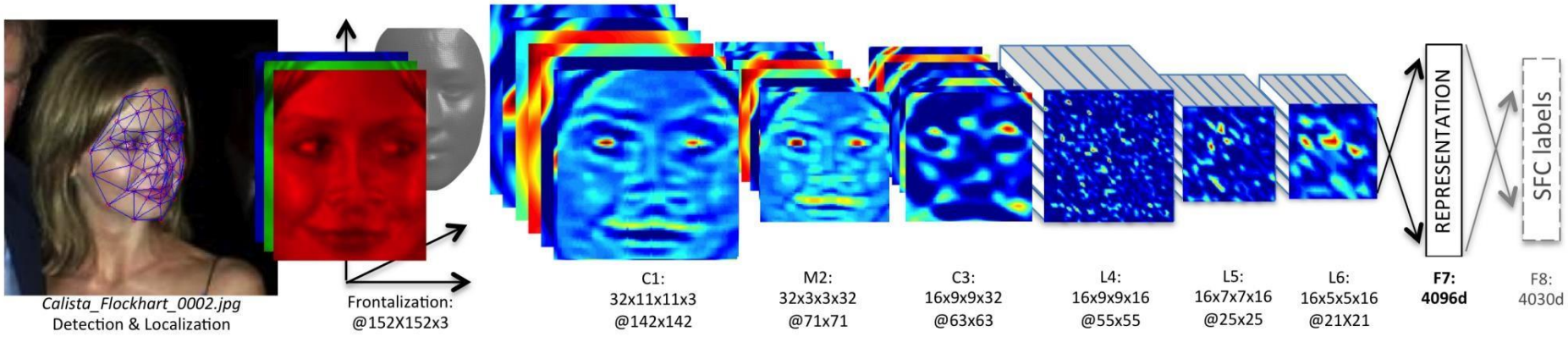
ImageNet Classification Challenge



- **AlexNet (2012): First CNN (15.4%)**
  - 8 layers
  - 61 million parameters
- **ZFNet (2013): 15.4% to 11.2%**
  - 8 layers
  - More filters. Denser stride.
- **VGGNet (2014): 11.2% to 7.3%**
  - Beautifully uniform:
    3x3 conv, stride 1, pad 1, 2x2 max pool
  - 16 layers
  - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
  - Inception modules
  - 22 layers
  - 5 million parameters
    (throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
  - More layers = better performance
  - 152 layers
- **CUImage (2016): 3.57% to 2.99%**
  - Ensemble of 6 models
- **SENet (2017): 2.99% to 2.251%**
  - Squeeze and excitation block: network is allowed to adaptively adjust the weighting of each feature map in the convolutional block.

Industry Deployment

- Used in Facebook, Google, Microsoft
- Image Recognition, Speech Recognition, ….
- Fast at test time

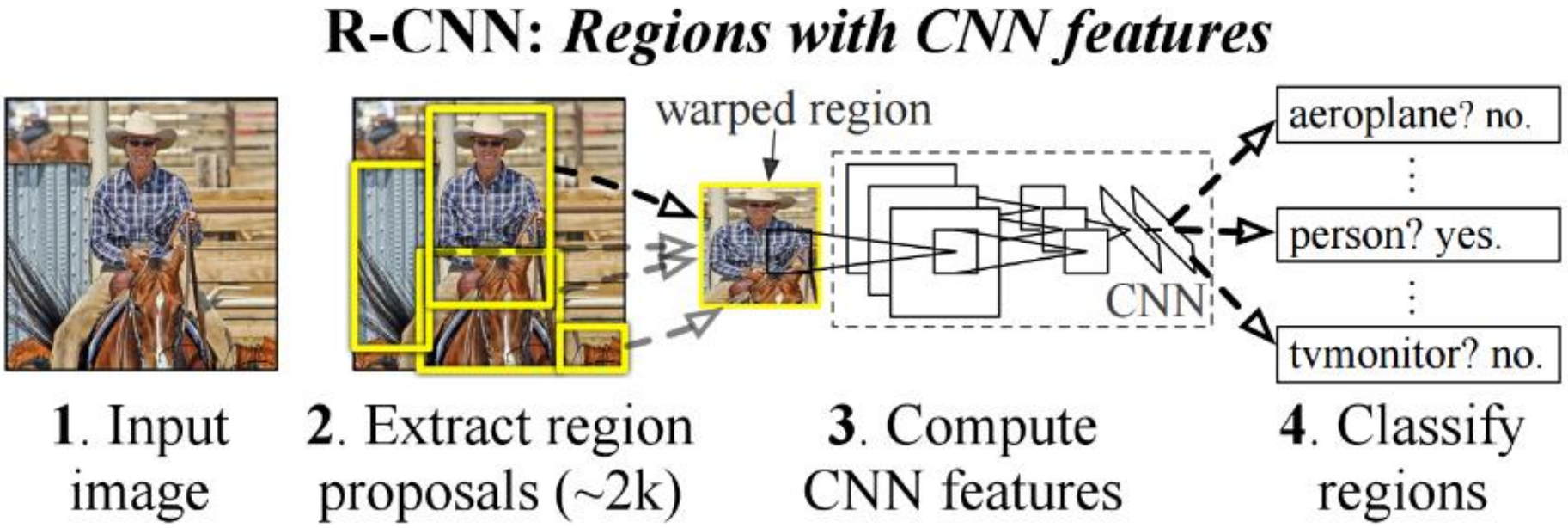


Taigman et al. DeepFace: Closing the Gap to Human-Level Performance in Face Verification, CVPR'14

Slide: R. Fergus

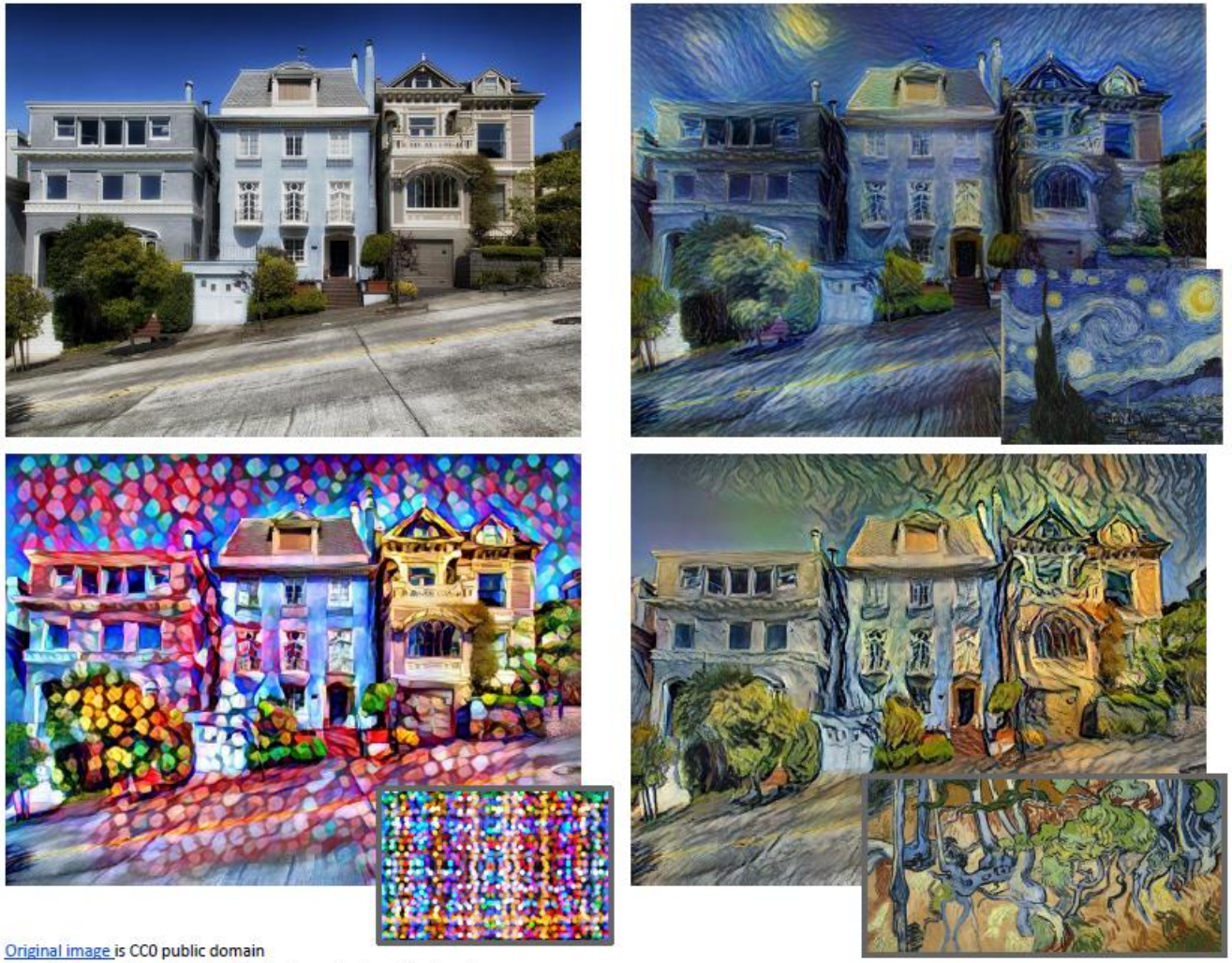
R-CNN: Regions with CNN features

- Trained on ImageNet classification
- Finetune CNN on PASCAL



RCNN [Girshick et al. CVPR 2014]

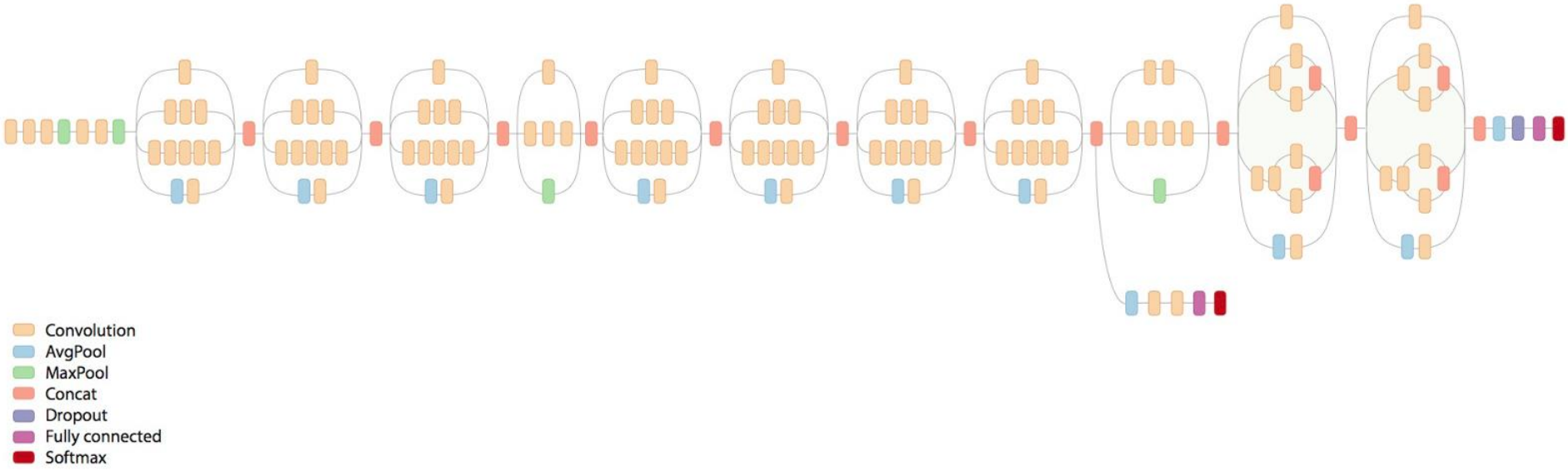
Style Transfer



Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain
Bokeh image is in the public domain
Stylized images copyright Justin Johnson, 2017;
reproduced with permission

Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

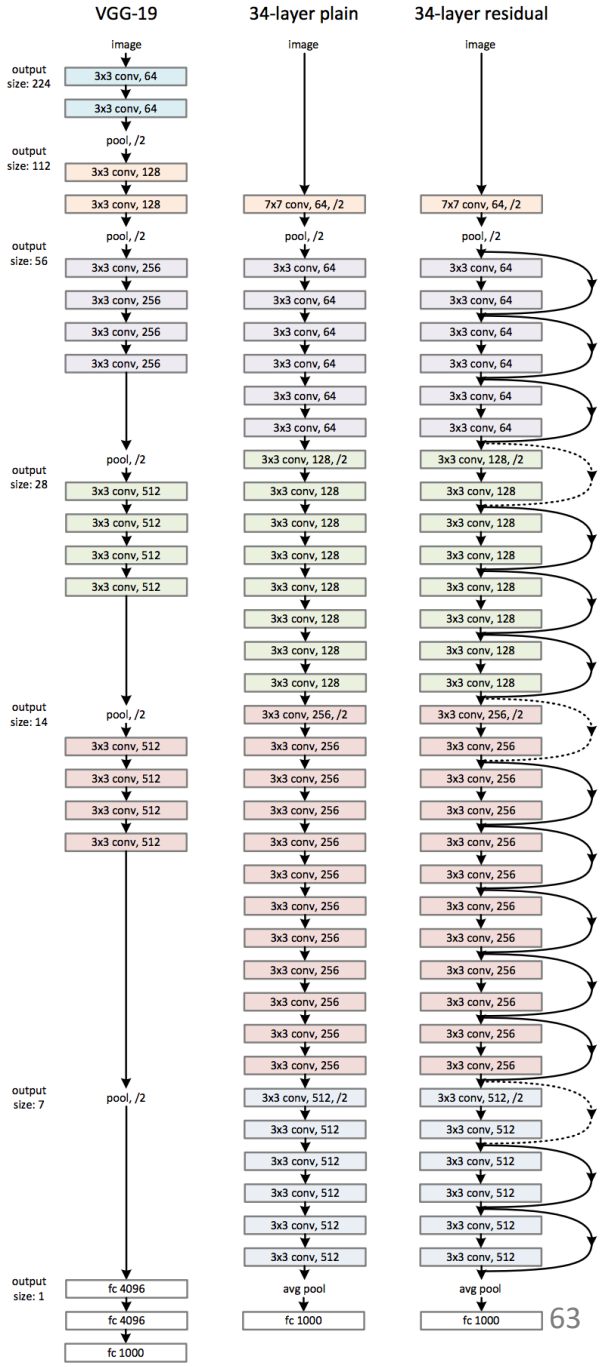
Inception V3 (2015)



Rethinking the Inception Architecture for Computer Vision, 2015
Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, 2016

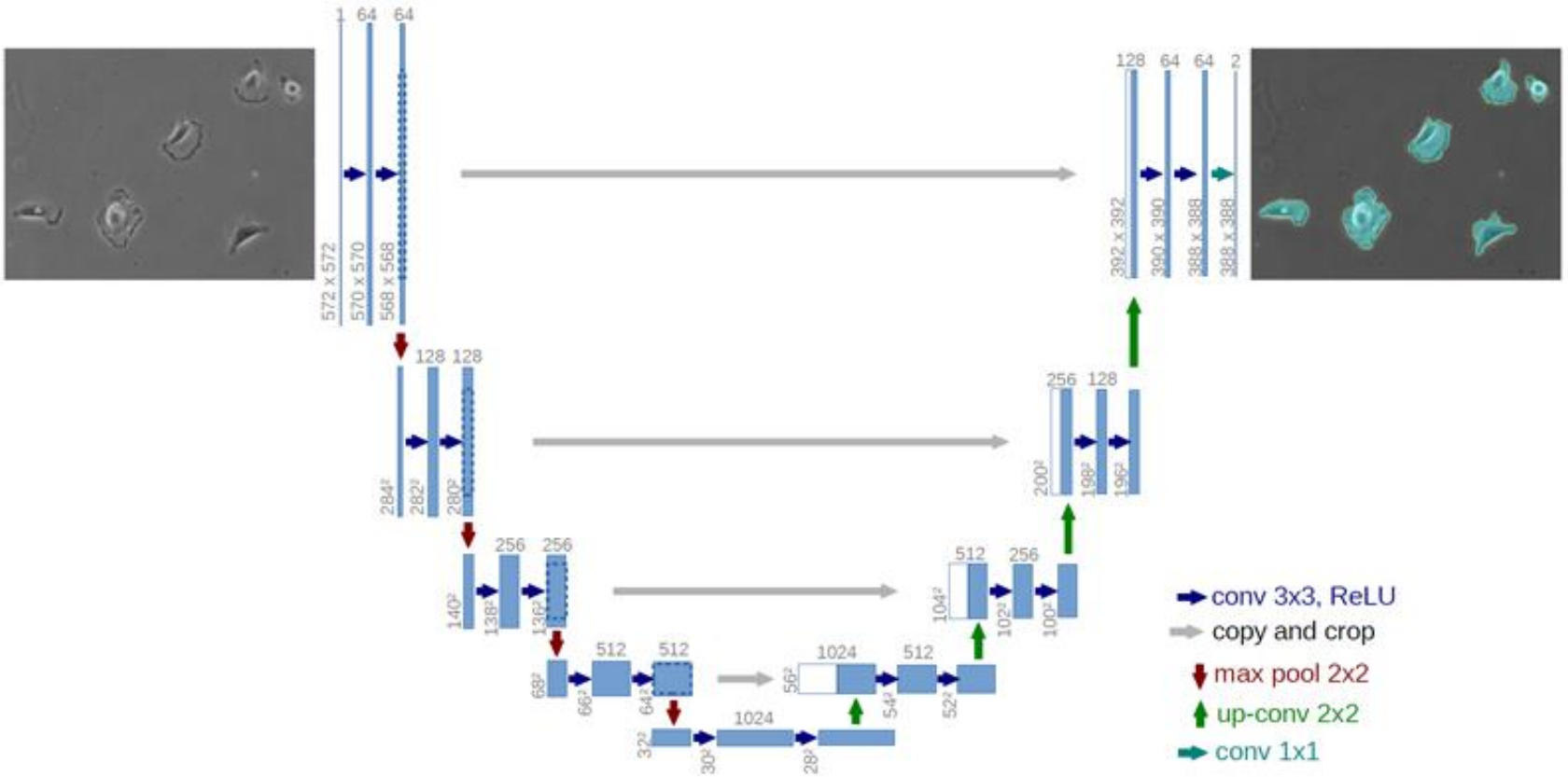
ResNets (2015)

- Gradients highways



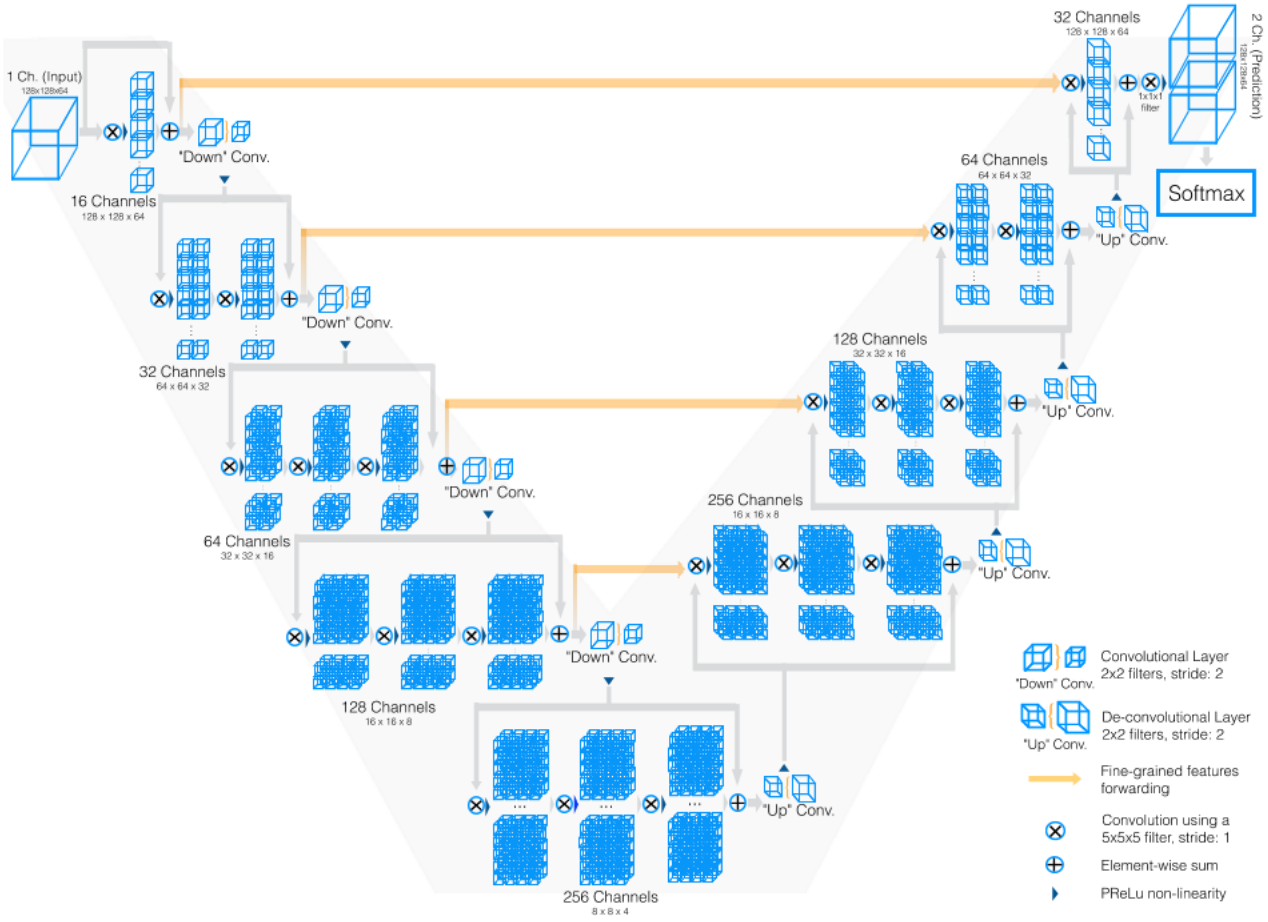
Deep Residual Learning for Image Recognition, 2015

U-net (2015)



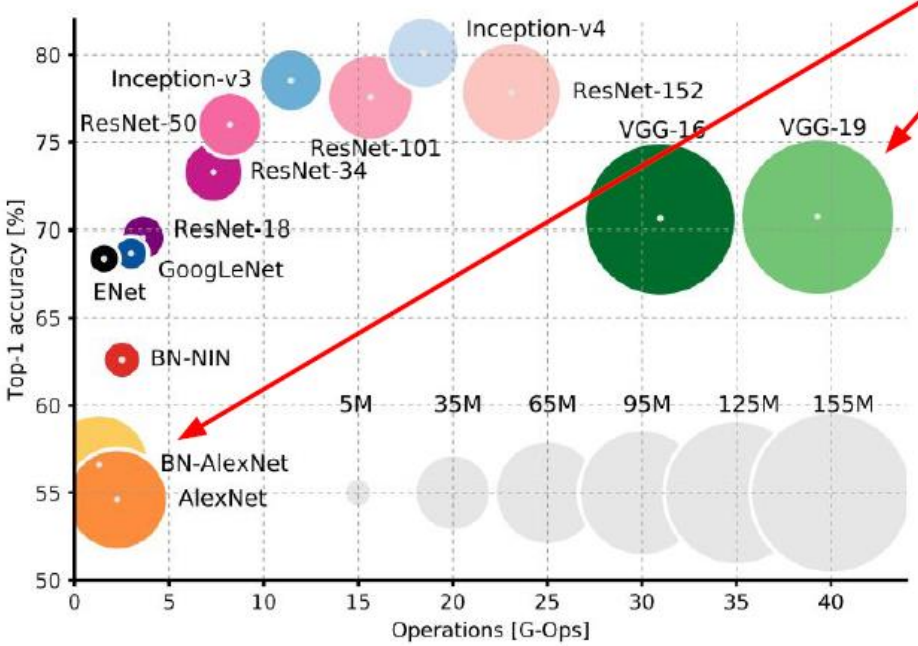
U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015

V-Net (2016)



V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation, 2016

CNN Architectures



AlexNet and VGG have tons of parameters in the fully connected layers

AlexNet: ~62M parameters

FC6: 256x6x6 -> 4096: 38M params
FC7: 4096 -> 4096: 17M params
FC8: 4096 -> 1000: 4M params
~59M params in FC layers!

Summary

- CNN stack Conv, Pool, FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of Pool/ FC layers