Security and Fairness of Deep Learning

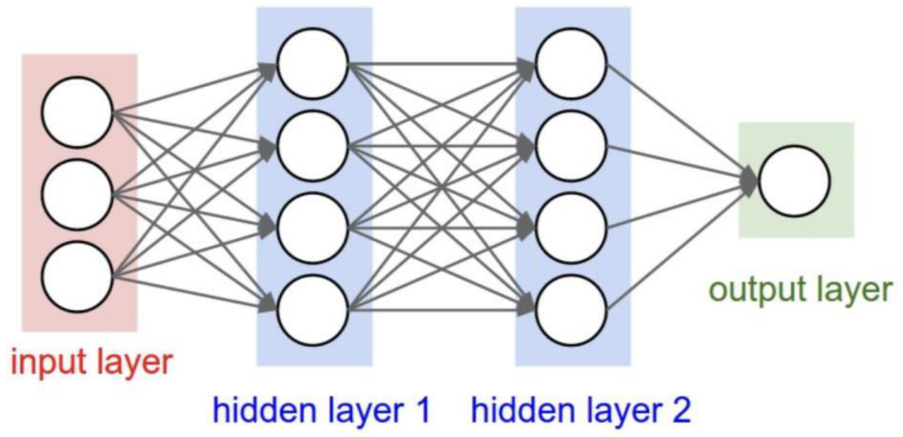
Convolutional Neural Networks I

Anupam Datta

CMU

Spring 2018

Neural network architectures



- Full connectivity is a problem for image inputs
  - Scalability: 200x200x3 images imply 120,000 weights per neuron in first hidden layer
  - Overfitting: Too many parameters would lead to overfitting

Convolutional Neural Networks [LeCun 1989]

- Specialized to the case where inputs are images (more generally, data with a grid-like topology)

- Sparse connections, parameter sharing
  - Efficient to train
  - Avoid overfitting

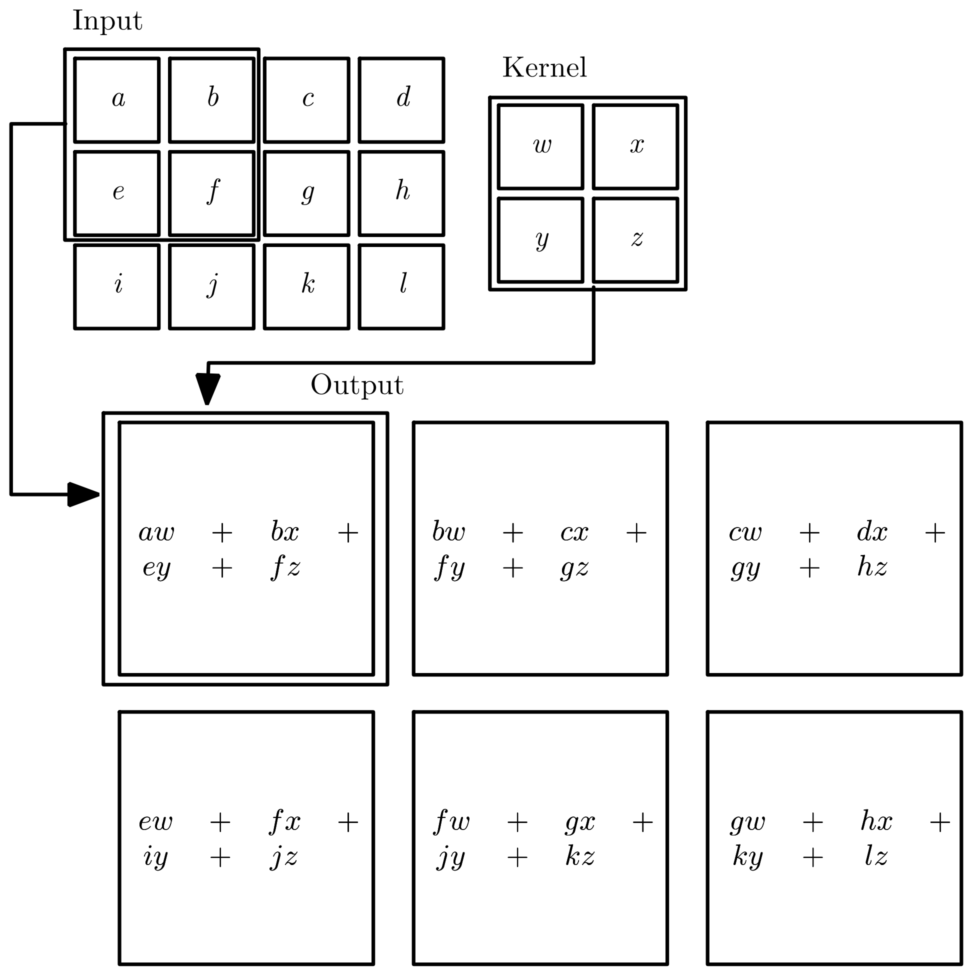
- Generalize across spatial translations of input
  - By sliding “filters” that learn distinct patterns (edges, blobs of color etc.)

Key idea

- Replace matrix multiplication in neural networks with <u>convolution</u>

- Everything else remains the same

2D Convolution



Sliding filters (kernels)

Fig. Goodfellow et al. 2016

Edge detection by convolution

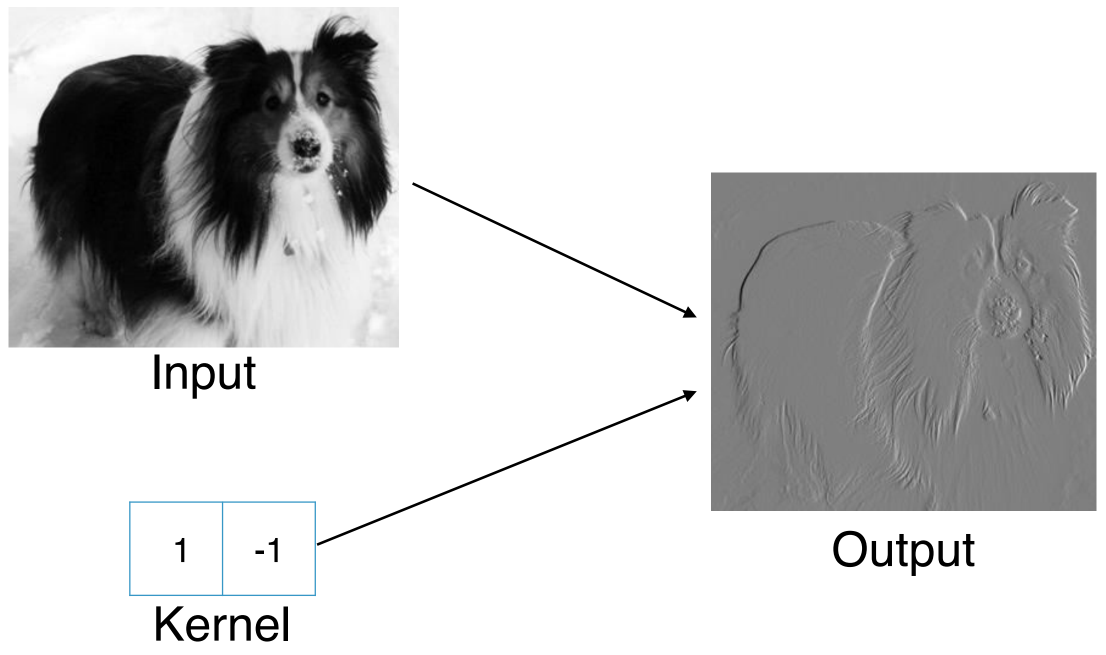
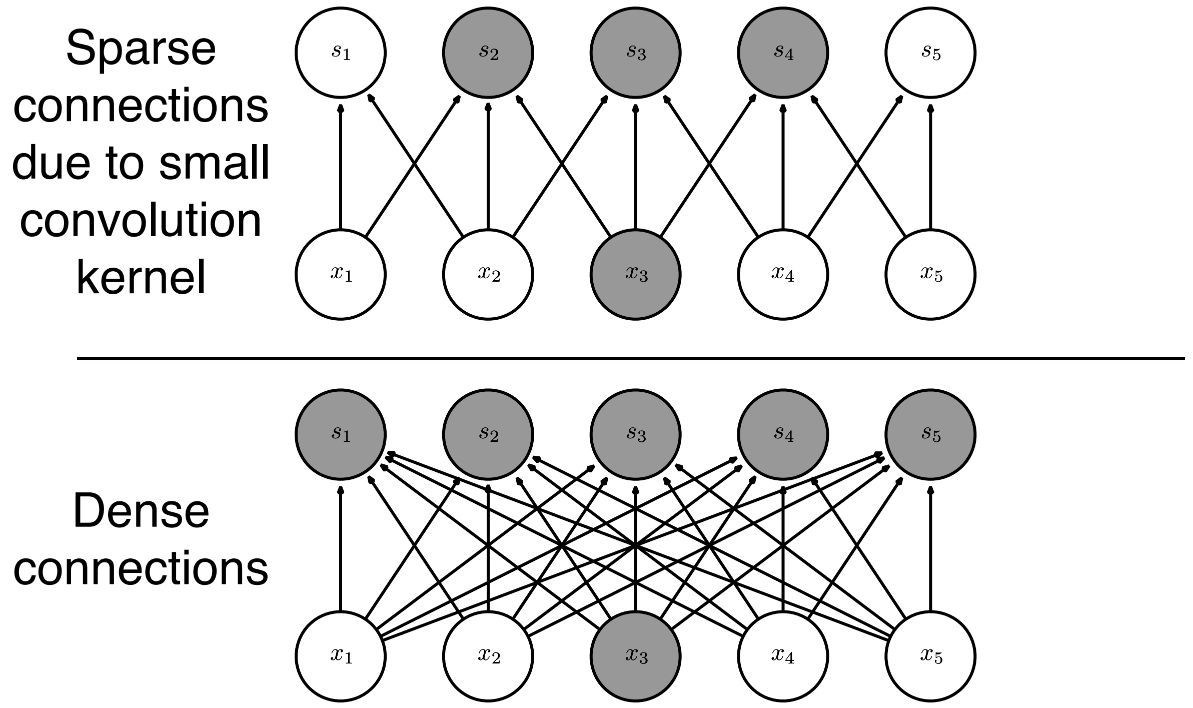


Figure 9.6

(Goodfellow 2016)

Sparse connectivity



Sparse connections due to small convolution kernel
Dense connections
s_1 s_2 s_3 s_4 s_5
x_1 x_2 x_3 x_4 x_5
s_1 s_2 s_3 s_4 s_5
x_1 x_2 x_3 x_4 x_5

Figure 9.2

(Goodfellow 2016)

Sparse connectivity

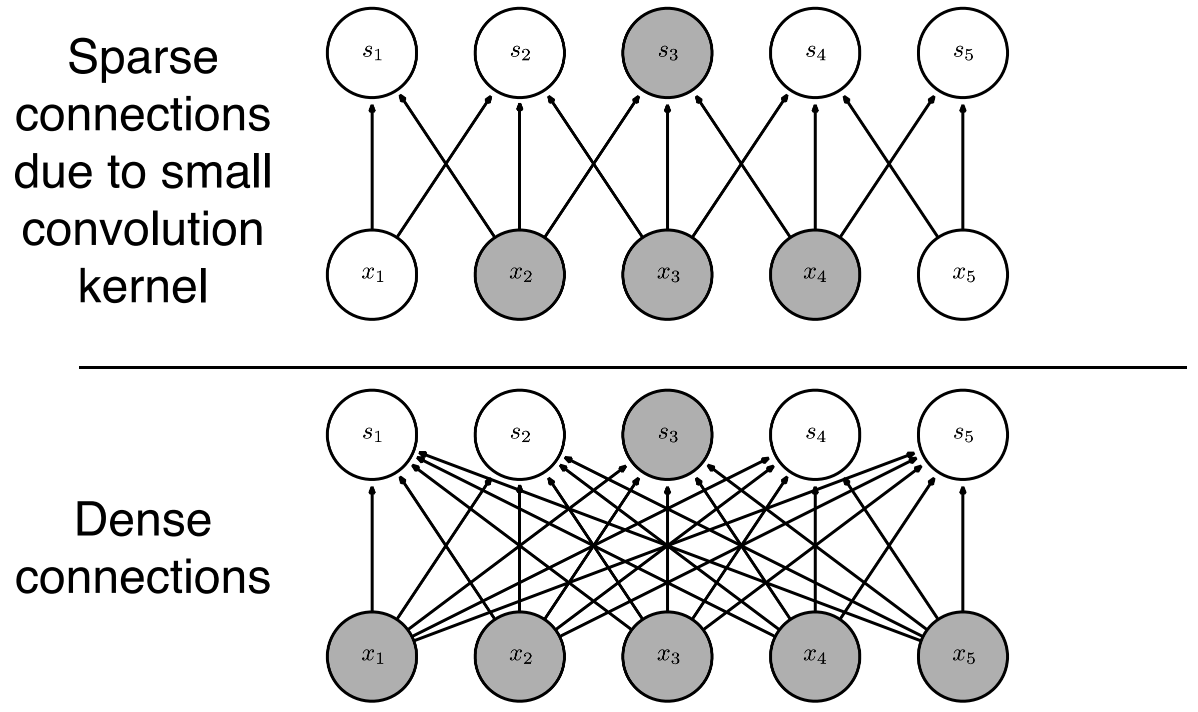


Figure 9.3

(Goodfellow 2016)

Growing receptive fields

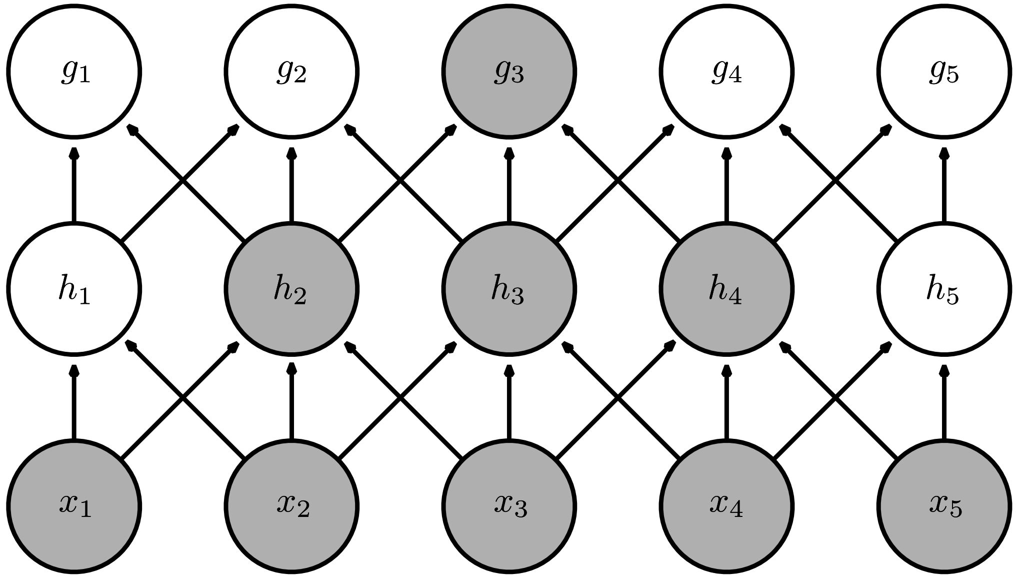


Figure 9.4

(Goodfellow 2016)

Parameter sharing

Convolution shares the same parameters across all spatial locations

Traditional matrix multiplication does not share any parameters

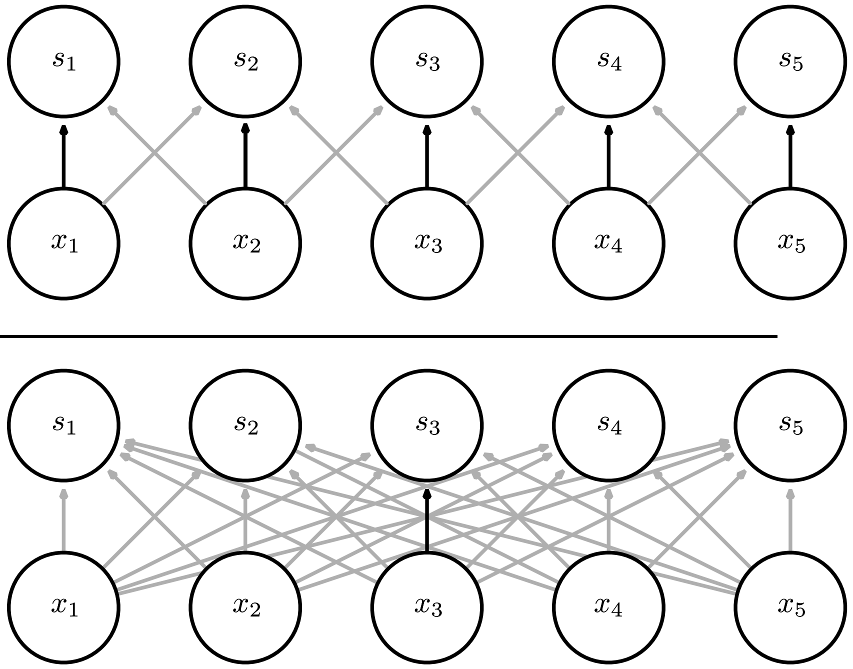


Figure 9.5

(Goodfellow 2016)

Edge detection by convolution

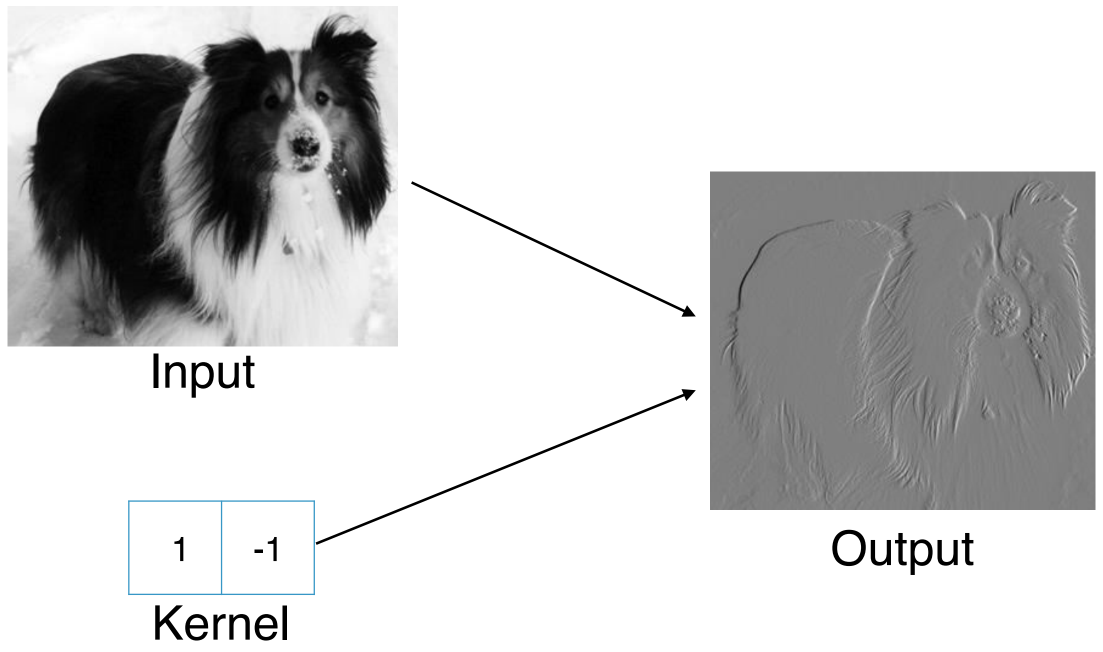


Figure 9.6

(Goodfellow 2016)

Efficiency of edge detection by convolution

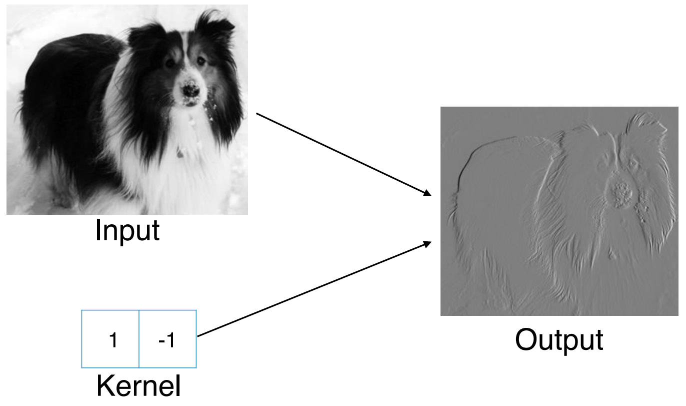


Figure 9.6 (Goodfellow 2016)

- Input: 320 x 280

- With convolution
  - 319 x 280 x 3 floating point ops
  - 267960

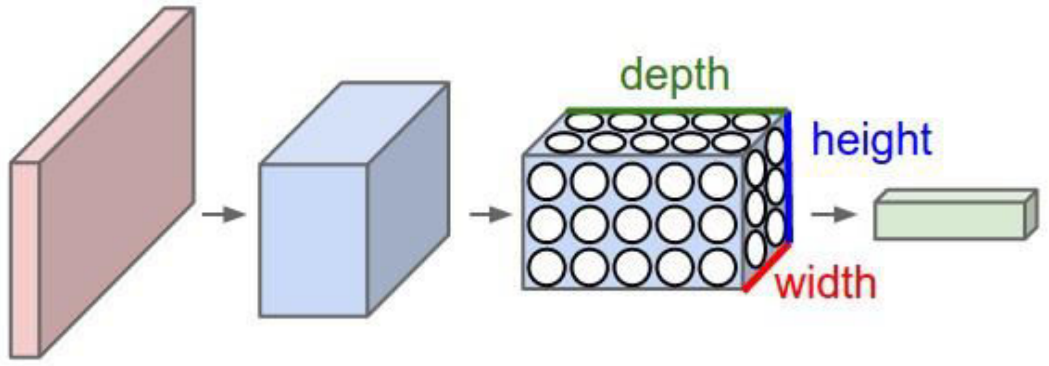
- With simple matrix multiplication
  - 320 x 280 x 319 x 280 elements in matrix
  - 16 billion + floating point ops

Limitations of convolution

- Captures spatial translation of input features but not changes in scale or rotation

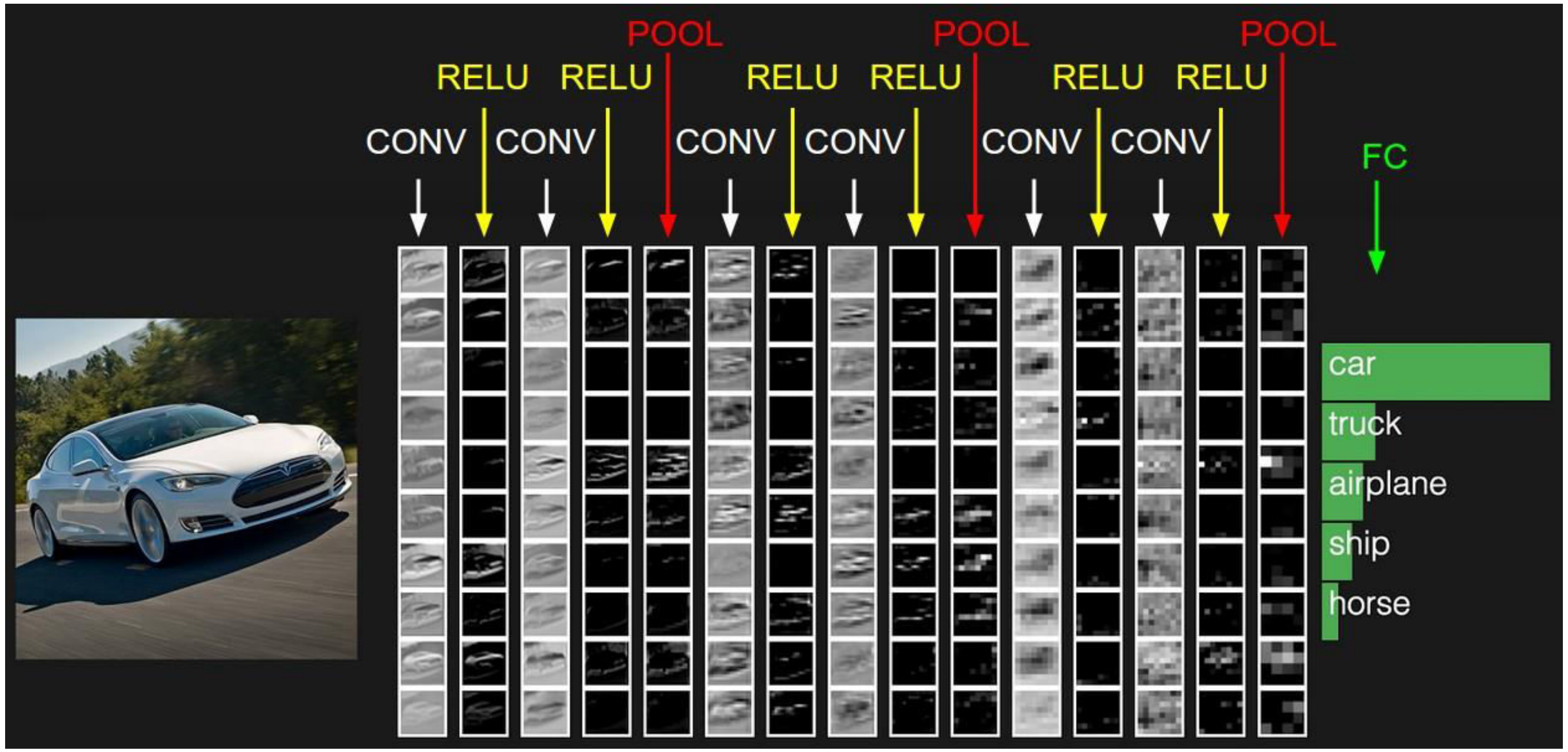
Dynamic Routing Between Capsules
(paper reading suggestion)

Convolutional Neural Networks



- A ConvNet is made up of Layers
- Every Layer transforms an input 3D volume to an output 3D volume with some differentiable function that may or may not have parameters
- Neurons in a layer will only be connected to a small region of the layer before it

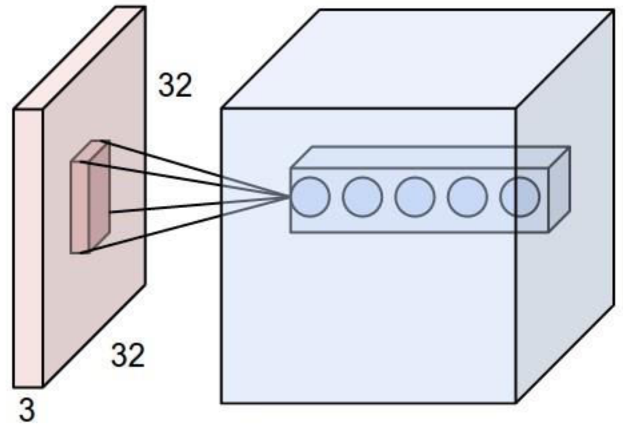
Example ConvNet architecture



Layers: CONV, RELU, POOL, FC

Convolutional layer

Connectivity

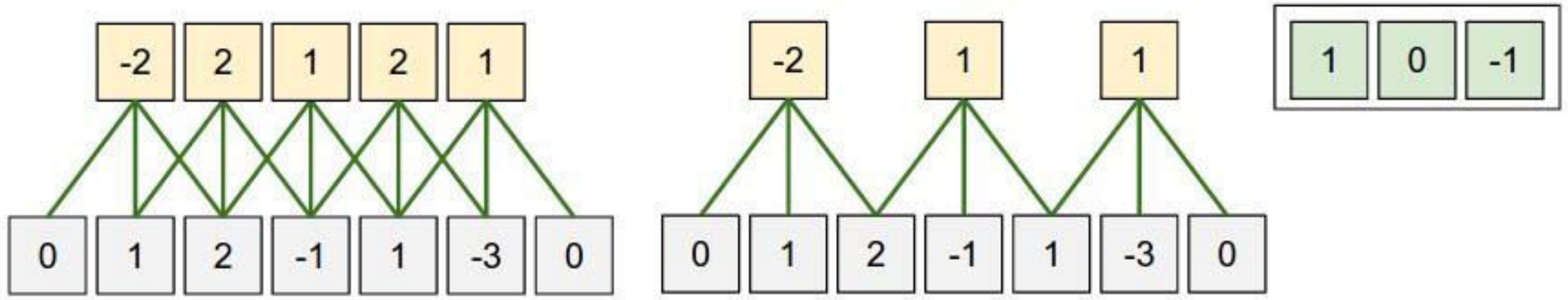


- An example input volume in red (e.g. a 32x32x3 CIFAR-10 image), and an example volume of neurons in the first Convolutional layer.
- Each neuron in the convolutional layer is connected only to a local region in the input volume spatially, but to the full depth (i.e. all color channels).
- If the receptive field (or the filter size) is 5x5, then each neuron in the Conv Layer will have weights to a [5x5x3] region in the input volume, for a total of 5*5*3 = 75 weights (and +1 bias parameter).
- There are multiple neurons (5 in this example) along the depth, all looking at the same region in the input; these are part of different filters.

Spatial arrangement

- Output volume depends on
  - Depth (Number of filters) K
  - Spatial extent of filters (receptive field) F
  - Stride S
  - Amount of zero-padding P

Spatial arrangement



- One spatial dimension (x-axis), one neuron with a receptive field size of F = 3, the input size is W = 5, and there is zero padding of P = 1
- Left: stride = 1; center: stride =2
- Right: neuron weights shared across all yellow neurons in the same depth slice (parameter sharing)
- Number of output neurons = (W−F+2P)/S+1
- Often P=(F−1)/2 when S=1; ensures number of output neurons = W

Spatial arrangement

- Depth
  - Number of filters
  - Each filter learns to look for a pattern in the input (e.g., first CONV layer filters may activate in the presence of differently oriented edges or blobs of color)

Spatial arrangement

- Stride
  - With which we slide the filters
  - When the stride is 1 then we move the filters one pixel at a time. When the stride is 2 (or uncommonly 3 or more) then the filters jump 2 pixels at a time as we slide them around

Spatial arrangement

- Zero-padding
  - Pad the input volume with zeros around the border
  - Allows us to control the spatial size of the output volumes

Parameter sharing

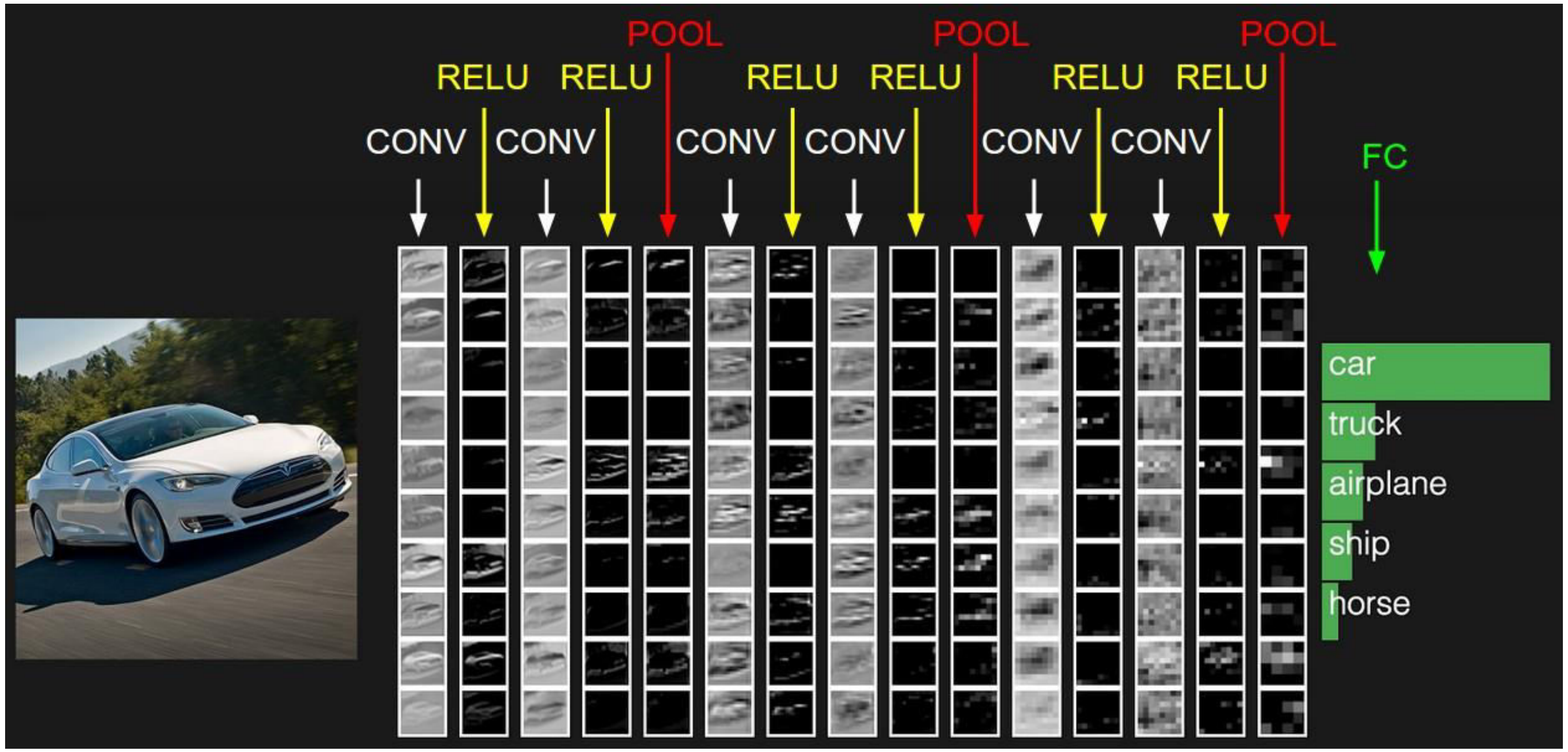
- Assumption
  - If one feature is useful to compute at some spatial position (x,y), then it should also be useful to compute at a different position (x2,y2)

- All neurons in the same depth slice use the same weights and bias

Convolution Demo

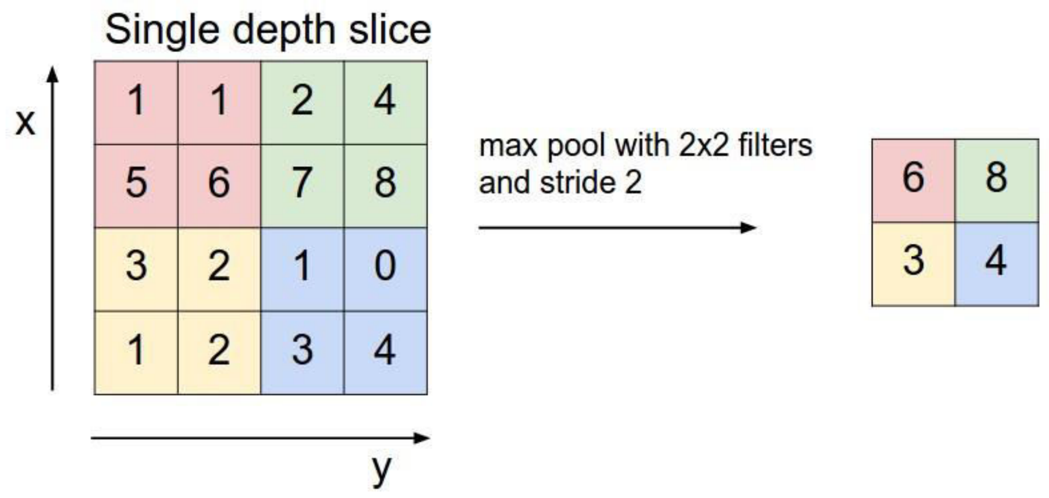
- http://cs231n.github.io/convolutional-networks/

Example ConvNet architecture



Layers: CONV, RELU, POOL, FC

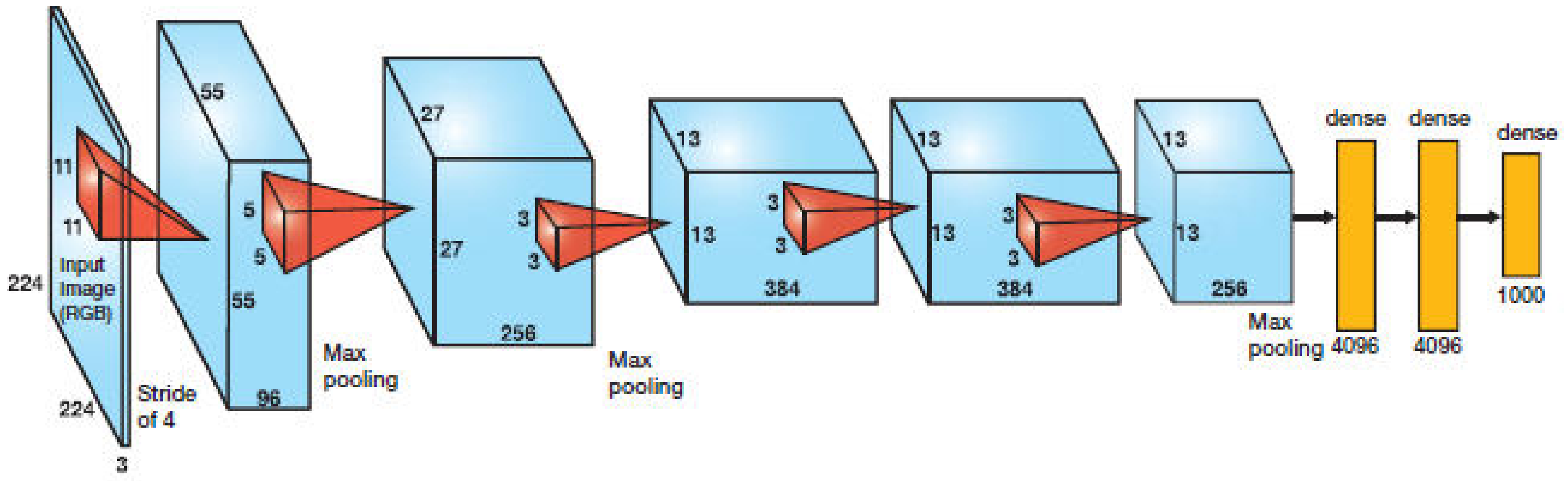
Pooling layer



- Goal: Reduce amount of parameters in network
  - Efficiency, address overfitting
- Common settings: 2x2 filter with stride 2 (fig above), 3x3 filter with stride 2

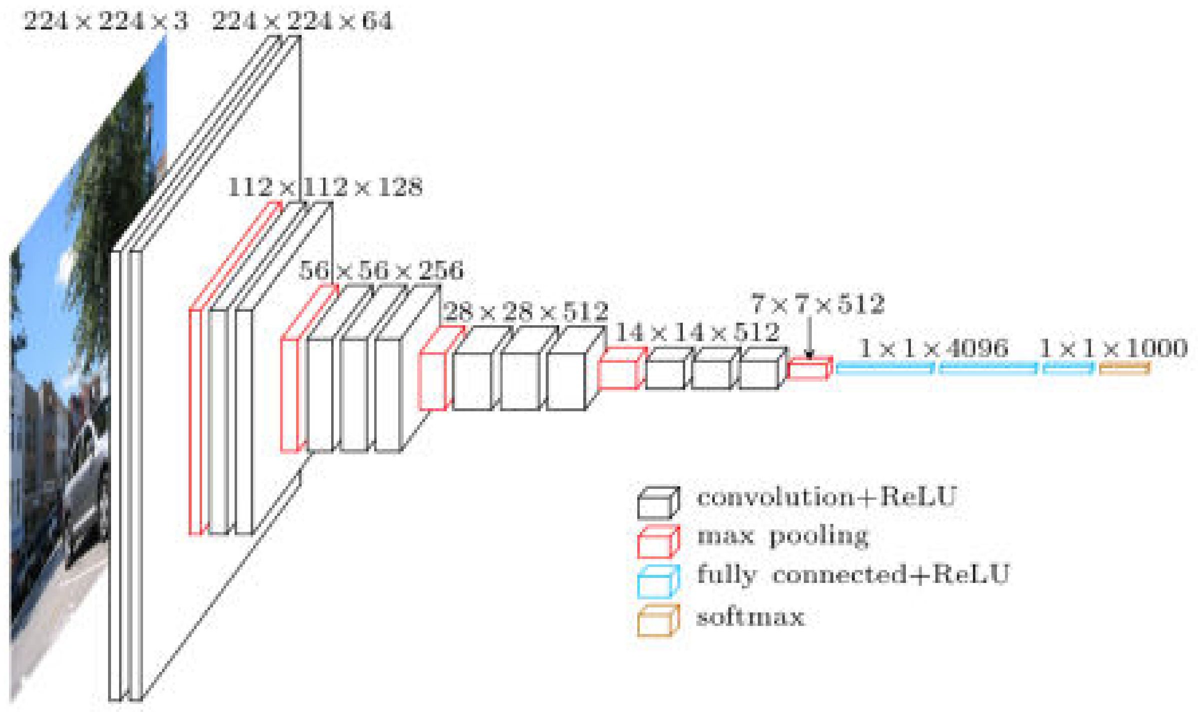
CNN architectures

AlexNet (2012 ImageNet ILSVRC challenge winner)



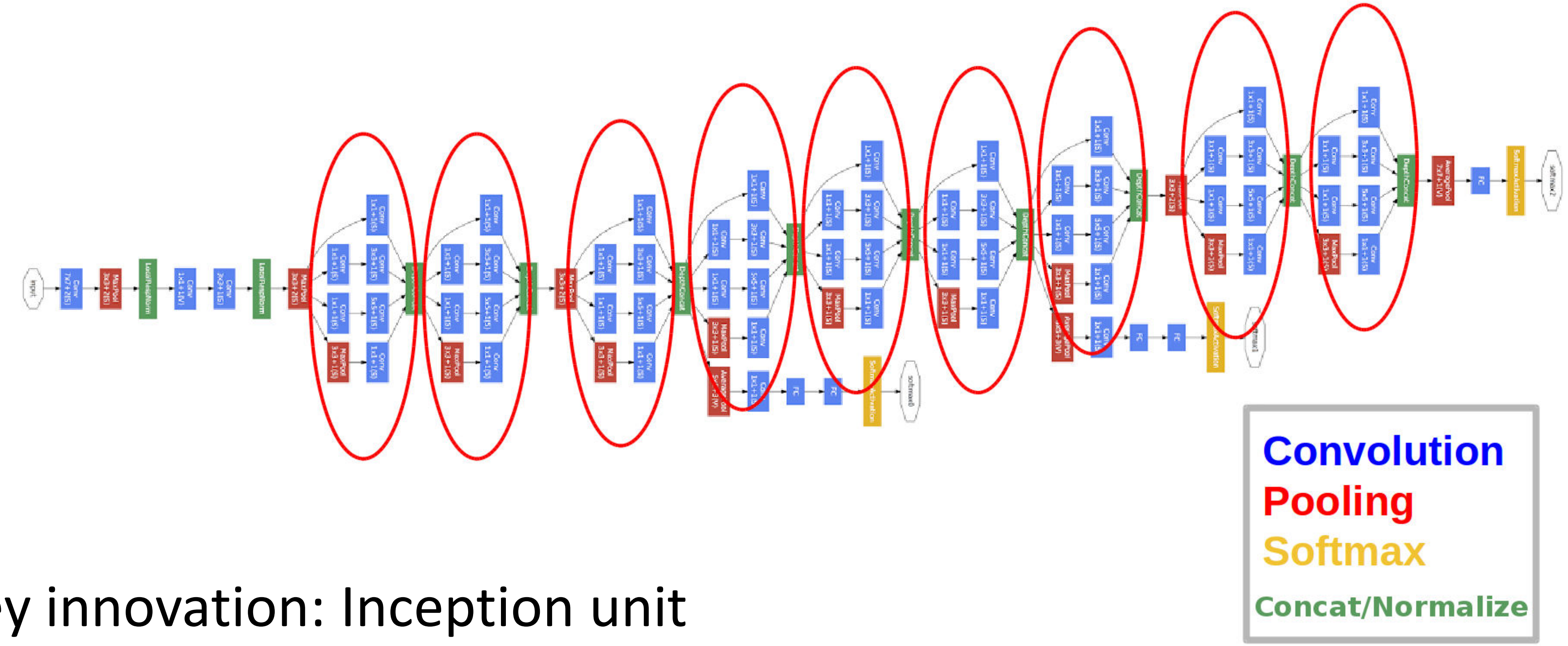
- Popularized CNNs in computer vision (top 5 error of 16% compared to runner-up 26%)
- Similar to LeNet but deeper, bigger
- Convolution layers stacked on top of each other without pooling in between

VGGNet (2014 ImageNet ILSVRC challenge runner-up)



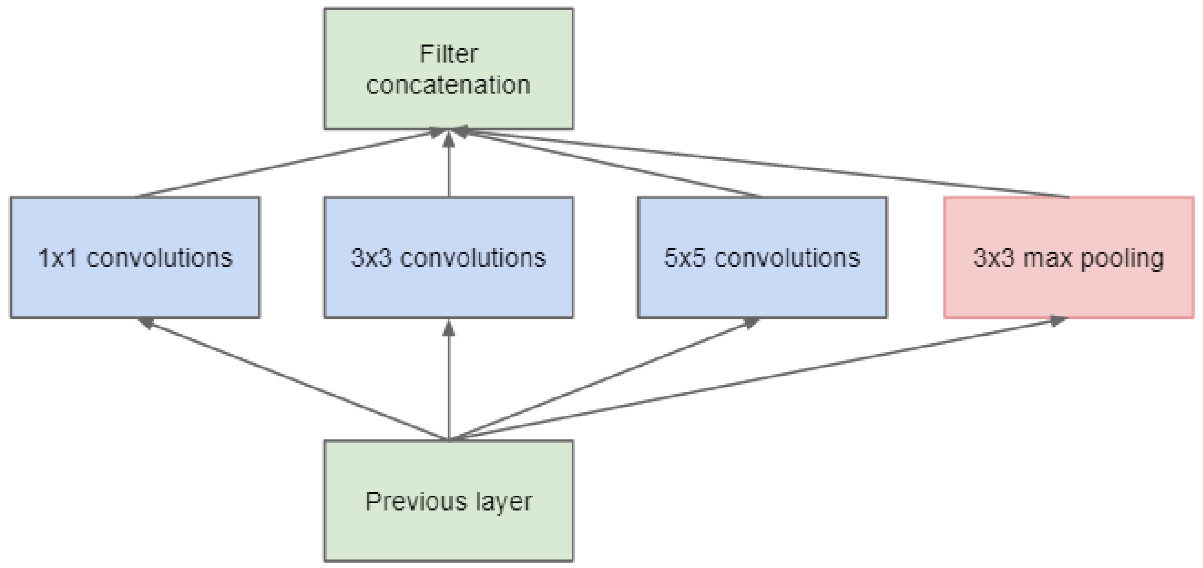
- Depth of network critical for good performance (16 CONV/FC layers)
- More expensive to evaluate; many parameters (140M)

GoogleNet (2014 ImageNet ILSVRC challenge winner)



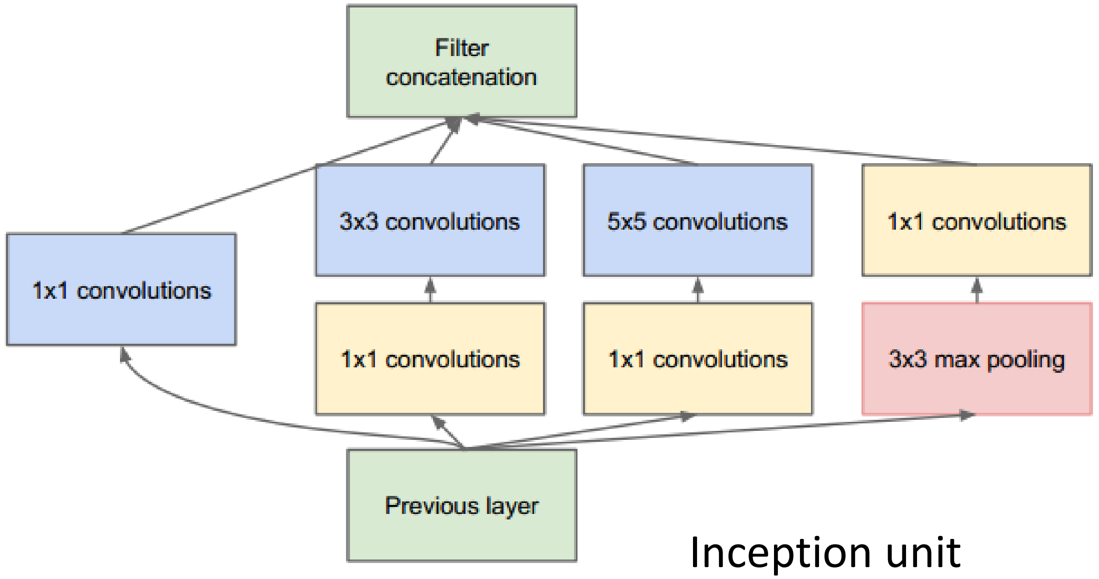
- Key innovation: Inception unit

GoogleNet (2014 ImageNet ILSVRC challenge winner)



- Filters with different sizes better handle multiple objects scales
- Max pooling layer is added to summarize the content of the previous layer

GoogleNet (2014 ImageNet ILSVRC challenge winner)



Inception unit

- Why the 1x1 convolutions?
  - Summarize information along depth slice of previous layer
  - Greatly reduce number of parameters: efficient, avoids overfitting

Acknowledgment

Based in part on material from Stanford CS231n
http://cs231n.github.io/ and Deep Learning book by Goodfellow et al.

Example ConvNet for CIFAR-10

- INPUT [32x32x3] will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.
- CONV layer 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. This may result in volume such as [32x32x12] if we decided to use 12 filters.
- RELU layer will apply an elementwise activation function, such as the max(0,x). This leaves the size of the volume unchanged ([32x32x12]).
- POOL layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].
- FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score.

AlexNet

- The Krizhevsky et al. architecture that won the ImageNet challenge in 2012 accepted images of size [227x227x3].
- On the first Convolutional Layer, it used neurons with receptive field size F=11, stride S=4 and no zero padding P=0.
- Since (227 - 11)/4 + 1 = 55, and since the Conv layer had a depth of K=96, the Conv layer output volume had size [55x55x96].
- Each of the 55*55*96 neurons in this volume was connected to a region of size [11x11x3] in the input volume.
- Moreover, all 96 neurons in each depth column are connected to the same [11x11x3] region of the input, but of course with different weights.

AlexNet

- Number of parameters
  - Without parameter sharing
    - 55*55*96 = 290,400 neurons in the first Conv Layer, and each has 11*11*3 = 363 weights and 1 bias. Together, this adds up to 290400 * 364 = 105,705,600 parameters on the first layer of the ConvNet
  - With parameter sharing
    - The first Conv Layer in our example would now have only 96 unique set of weights (one for each depth slice), for a total of 96*11*11*3 = 34,848 unique weights, or 34,944 parameters (+96 biases).