18AIC301J: DEEP LEARNING TECHNIQUES

B. Tech in ARTIFICIAL INTELLIGENCE, 5th semester

Faculty: **Dr. Athira Nambiar**

Section: A, slot:D

Venue: TP 804

Academic Year: 2022-22

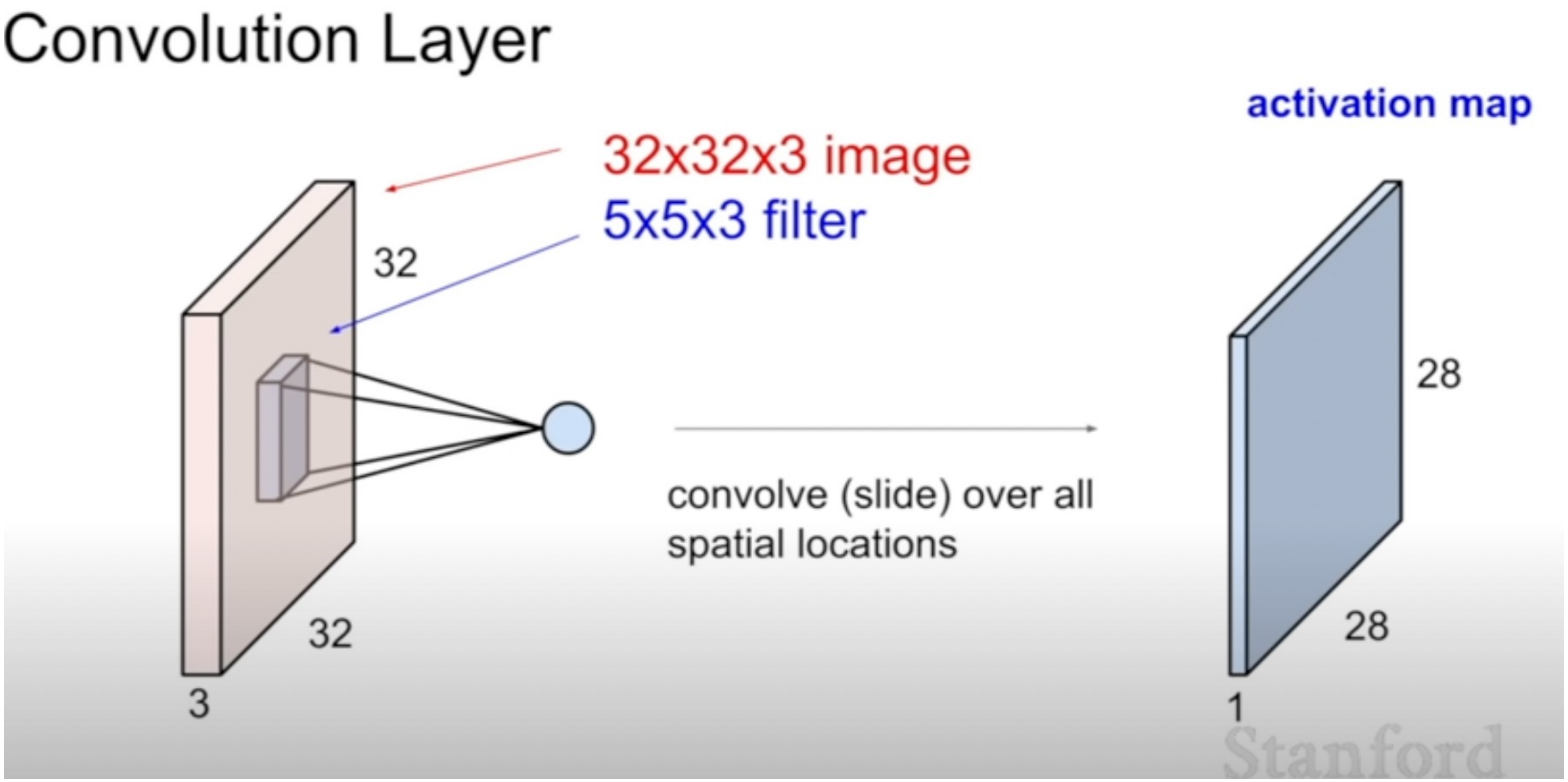
UNIT-3

One hot representation of words, Distributed representation of words
SVD for learning word Representations, Continuous bag of words model, Skip-gram model, Hierarchical Softmax
Implement skip gram model to predict words within a certain range before and after the current word
Introduction to Convolution Neural Networks, Kernel filters
The convolution operation with Filters, padding and stride, Multiple Filters, Max pooling and non-linearities
Implement LeNet for image classification
Classic CNNs architecture- The ImageNet challenge, Understanding Alex Net architecture
ZFNet, The intuition behind GoogleNet, Average pooling, Residual CNN-ResNet architecture
Implement ResNet for detecting Objects.

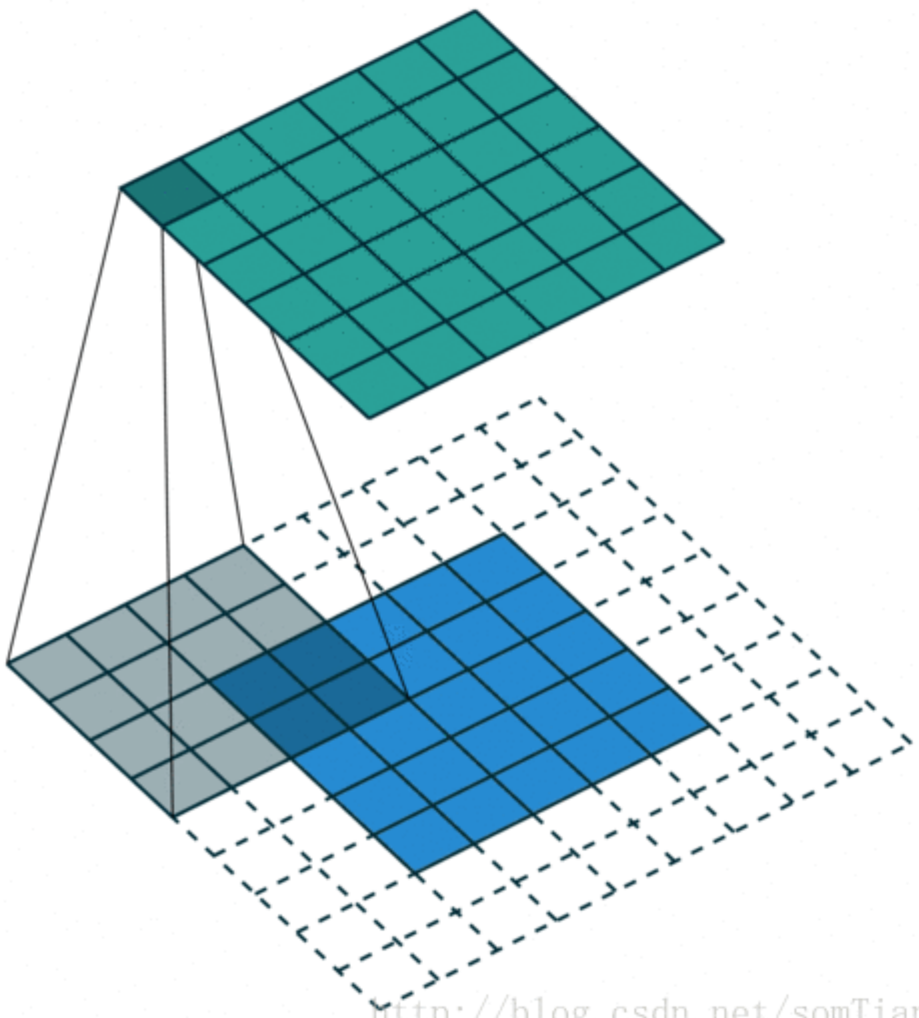
UNIT-3

One hot representation of words, Distributed representation of words
SVD for learning word Representations, Continuous bag of words model, Skip-gram model, Hierarchical Softmax
Implement skip gram model to predict words within a certain range before and after the current word
Introduction to Convolution Neural Networks, Kernel filters
The convolution operation with Filters, padding and stride, Multiple Filters, Max pooling and non-linearities
Implement LeNet for image classification
Classic CNNs architecture- The ImageNet challenge, Understanding Alex Net architecture
ZFNet, The intuition behind GoogleNet, Average pooling, Residual CNN-ResNet architecture
Implement ResNet for detecting Objects.

CONVOLUTION LAYER



CONVOLUTION LAYER



CONVOLUTION LAYER

The Convolution Operation

Suppose we want to compute the convolution of a 5x5 image and a 3x3 filter:

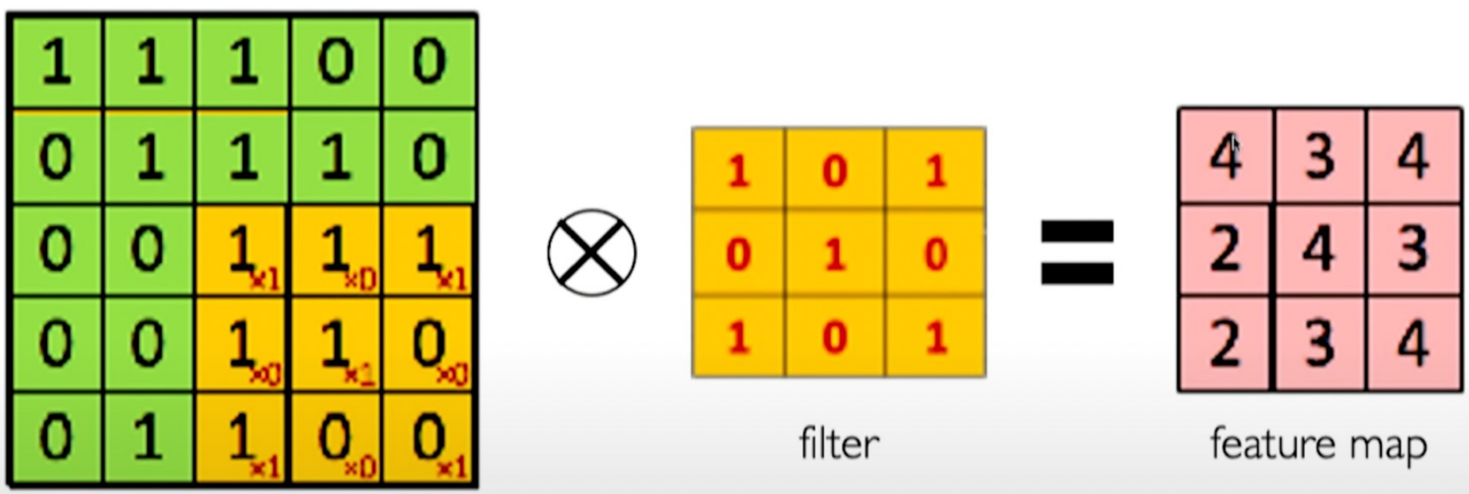
$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}}_{\text{image}} \otimes \underbrace{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}}_{\text{filter}}$$

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs…

CONVOLUTION LAYER

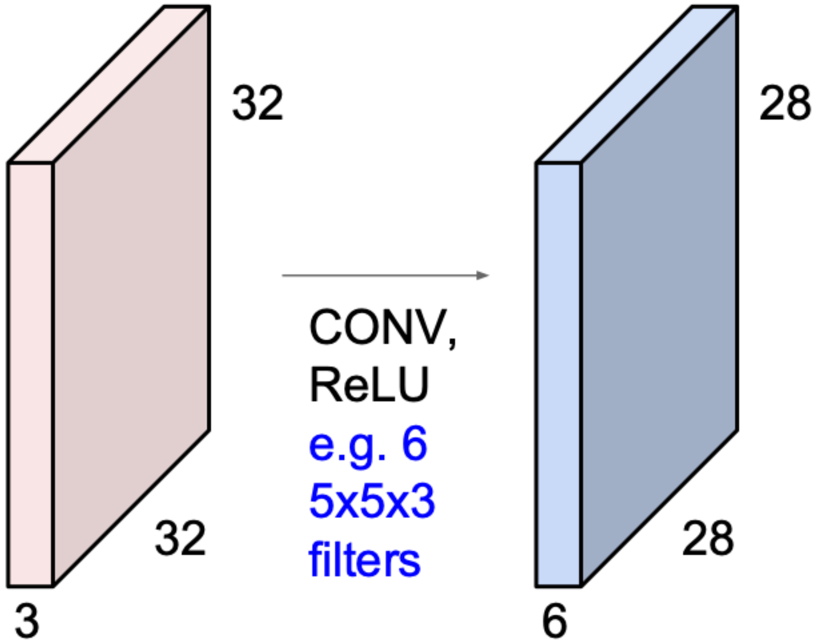
The Convolution Operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:



CONVOLUTION LAYER

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



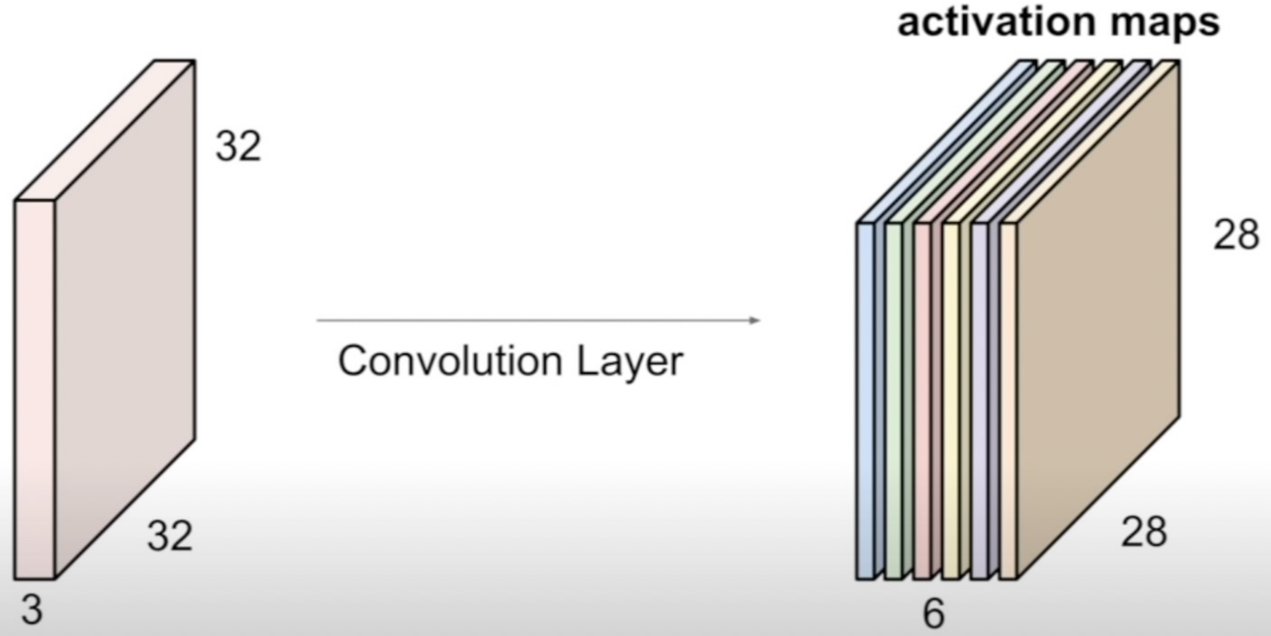
CONVOLUTION LAYER

Producing Feature Maps

Original | Sharpen | Edge Detect | "Strong" Edge Detect

CONVOLUTION LAYER

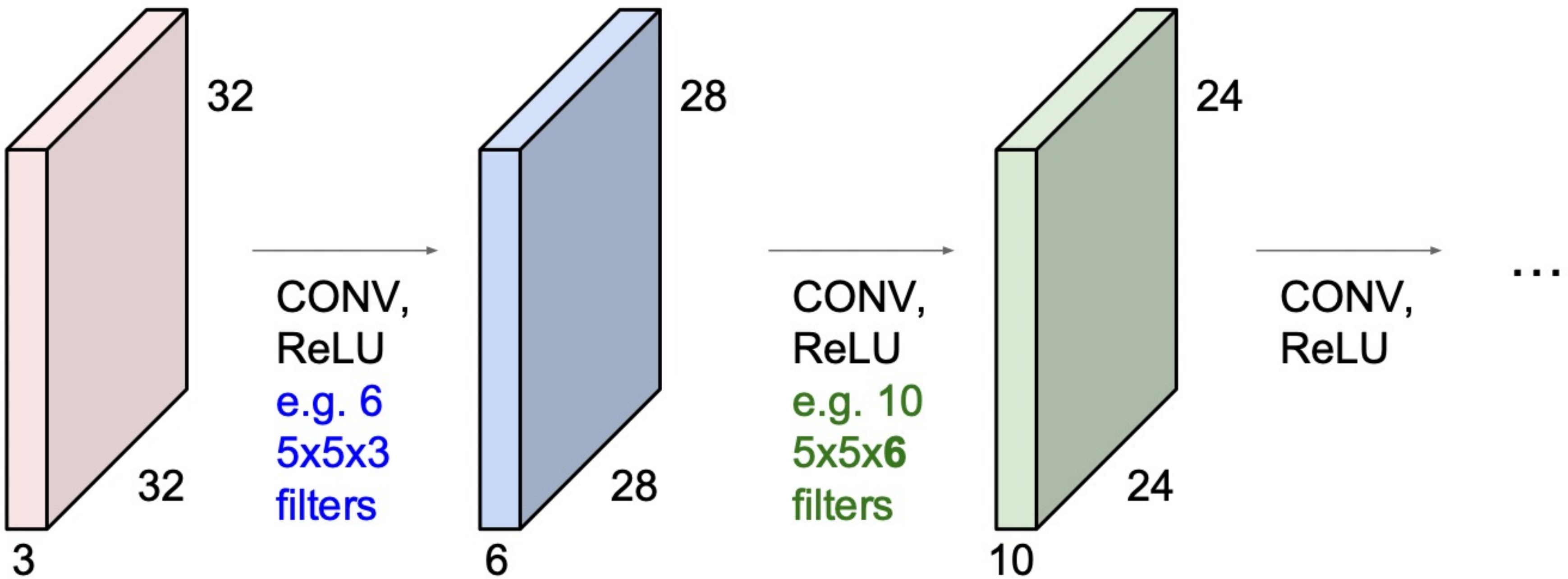
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



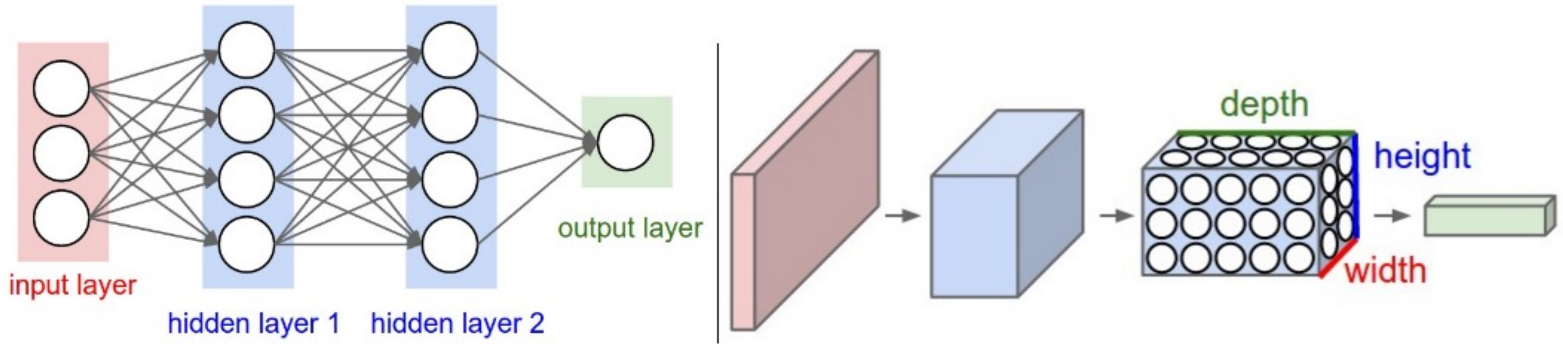
We stack these up to get a "new image" of size 28x28x6!

CONVOLUTION LAYER

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

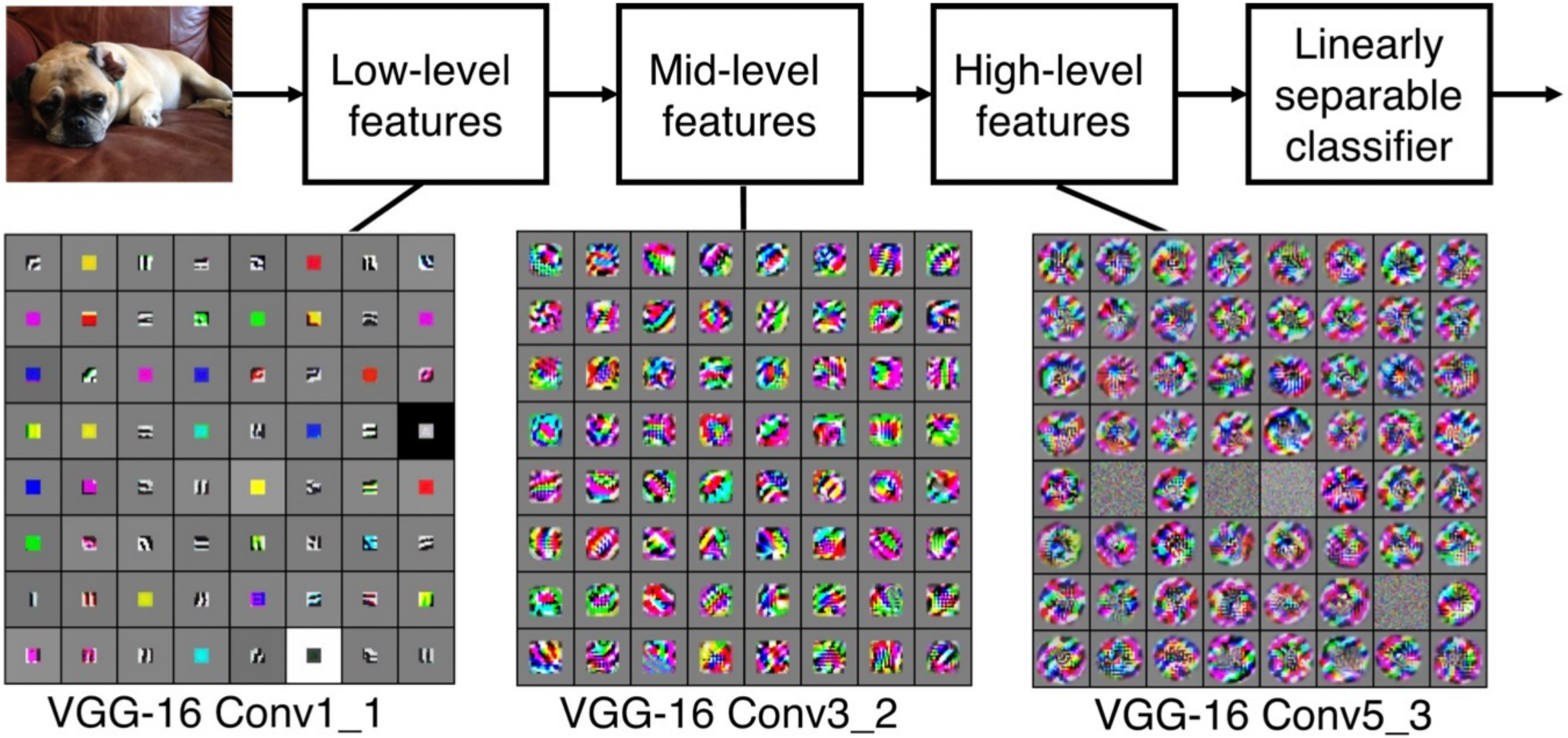


CONVOLUTION LAYER



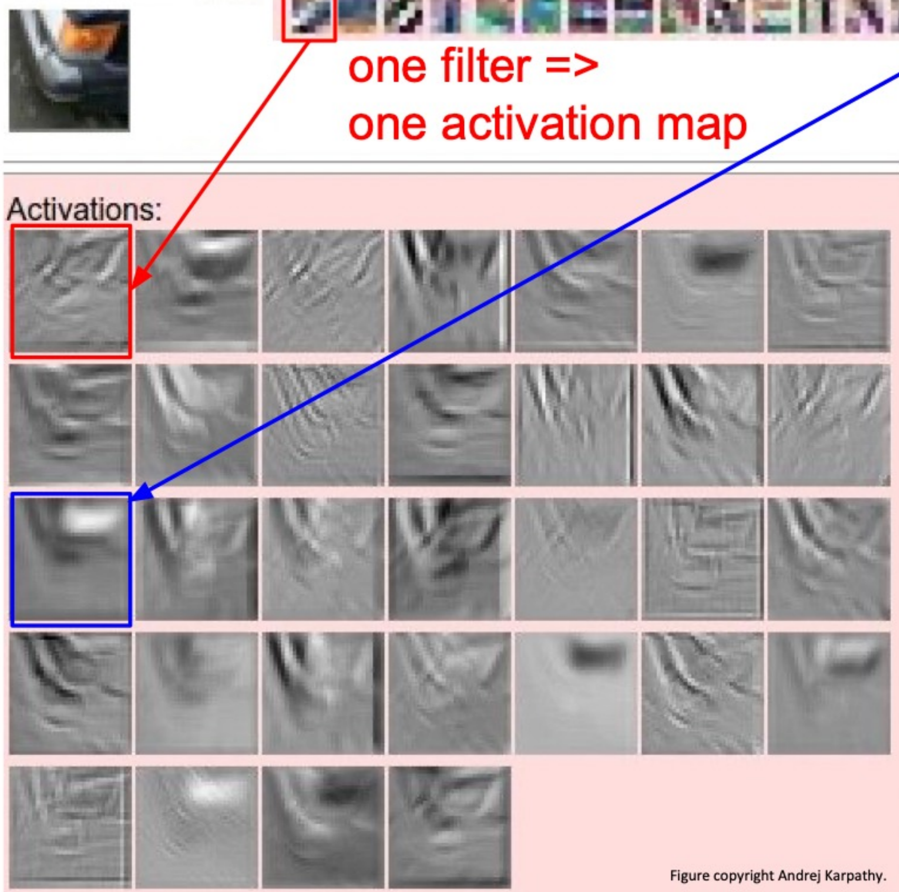
Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

CONVOLUTION LAYER



*[Zeiler and Fergus 2013], V*isualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014]

CONVOLUTION LAYER



example 5x5 filters
(32 total)

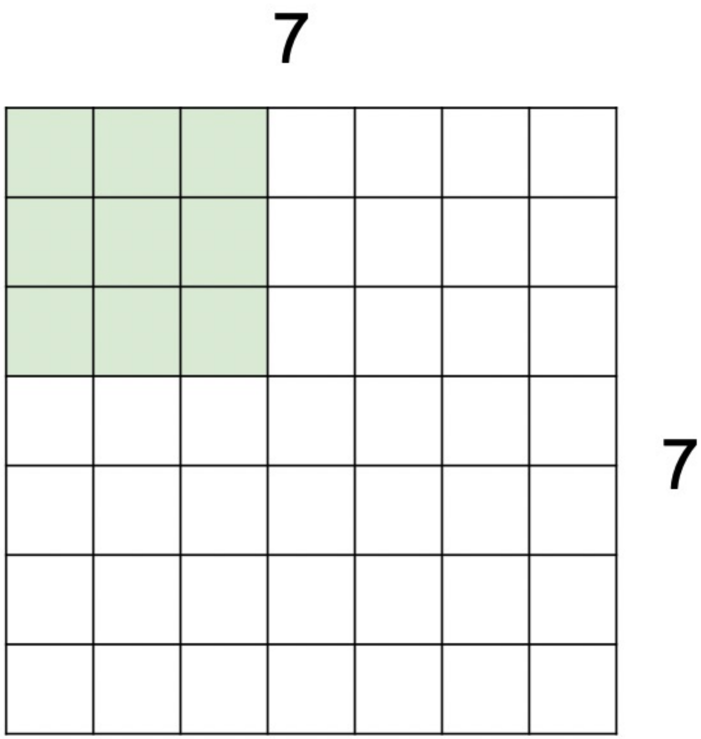
We call the layer convolutional because it is related to convolution of two signals:

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

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

CONVOLUTION LAYER

A closer look at spatial dimensions:

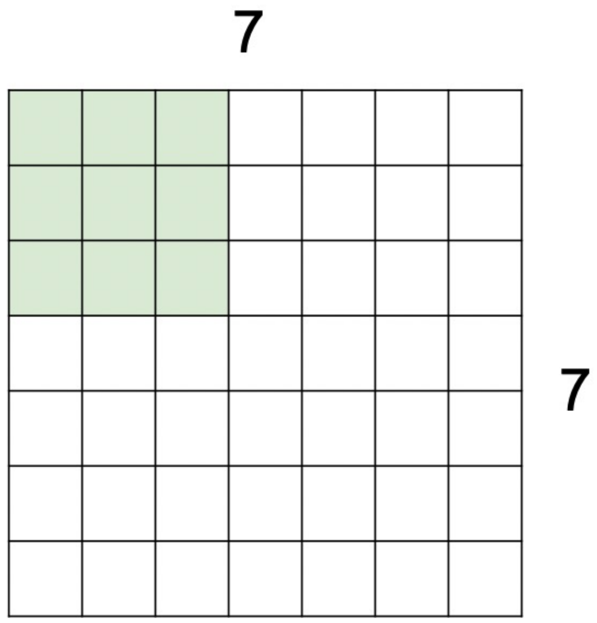


7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

CONVOLUTION LAYER-STRIDE

A closer look at spatial dimensions:

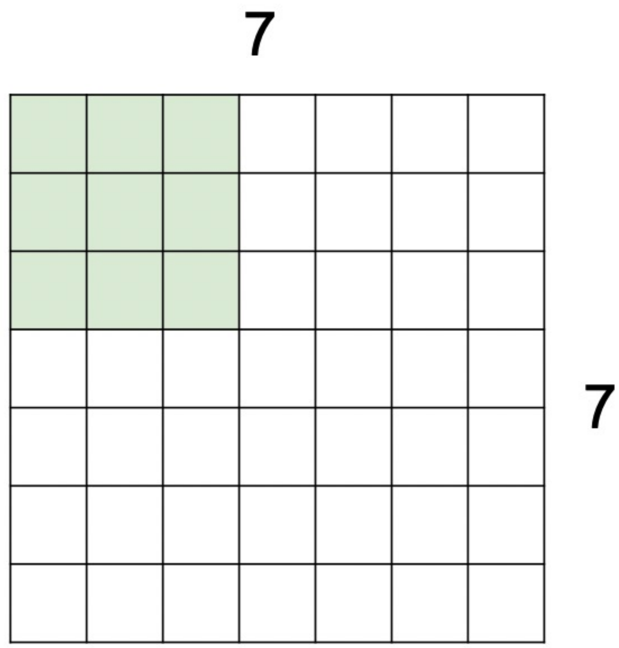


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

=> 3x3 output!

CONVOLUTION LAYER- STRIDE

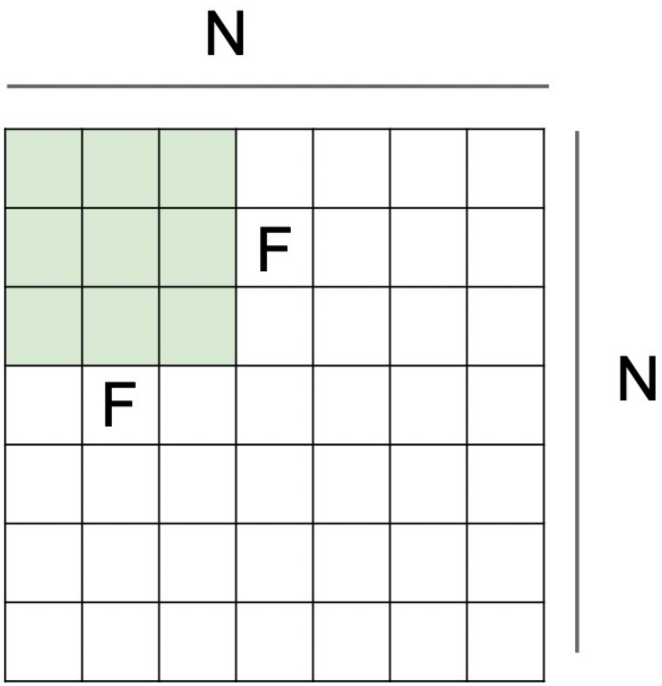
A closer look at spatial dimensions:



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

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

CONVOLUTION LAYER- STRIDE

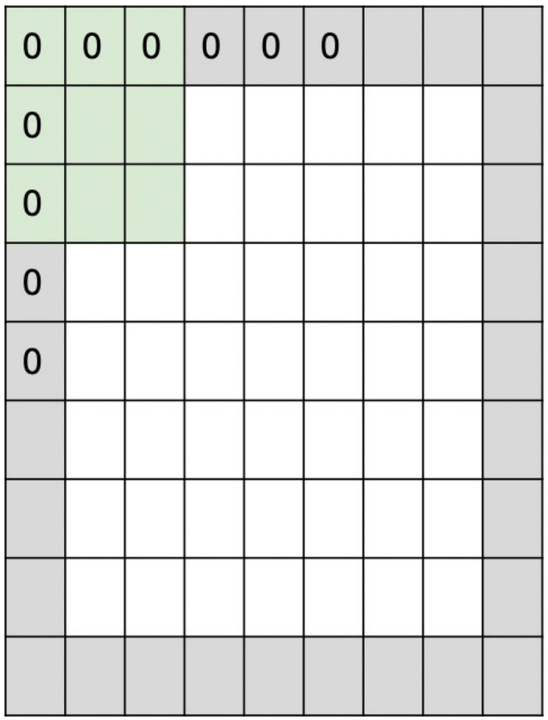


Output size:
(N - F) / stride + 1

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 :\

CONVOLUTION LAYER- STRIDE, PADDING

In practice: Common to zero pad the border



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

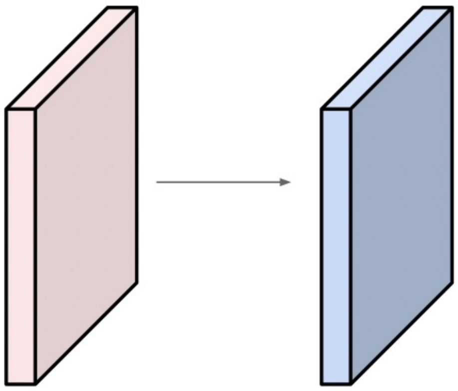
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

CONVOLUTION LAYER- STRIDE, PADDING

Examples time:



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

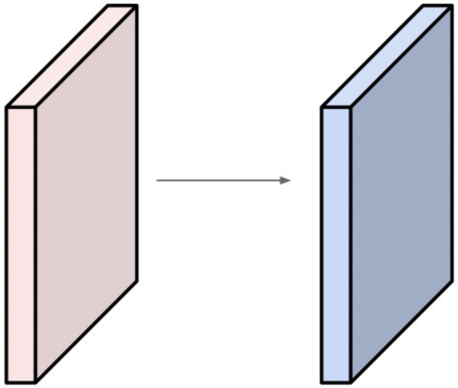
Output volume size: ?

CONVOLUTION LAYER- STRIDE, PADDING

Examples time:

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

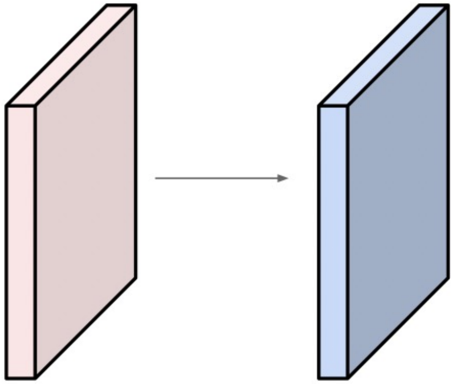
Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10



CONVOLUTION LAYER- STRIDE, PADDING

Examples time:

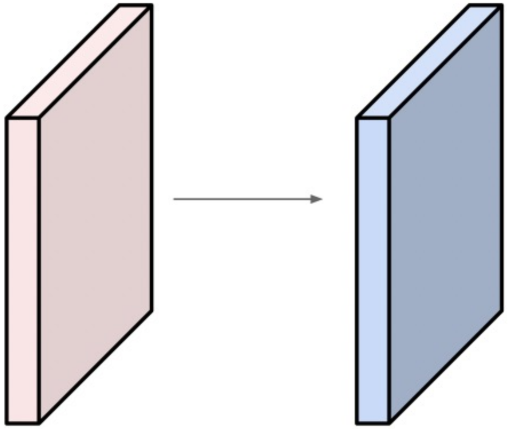
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2



Number of parameters in this layer?

CONVOLUTION LAYER- STRIDE, PADDING

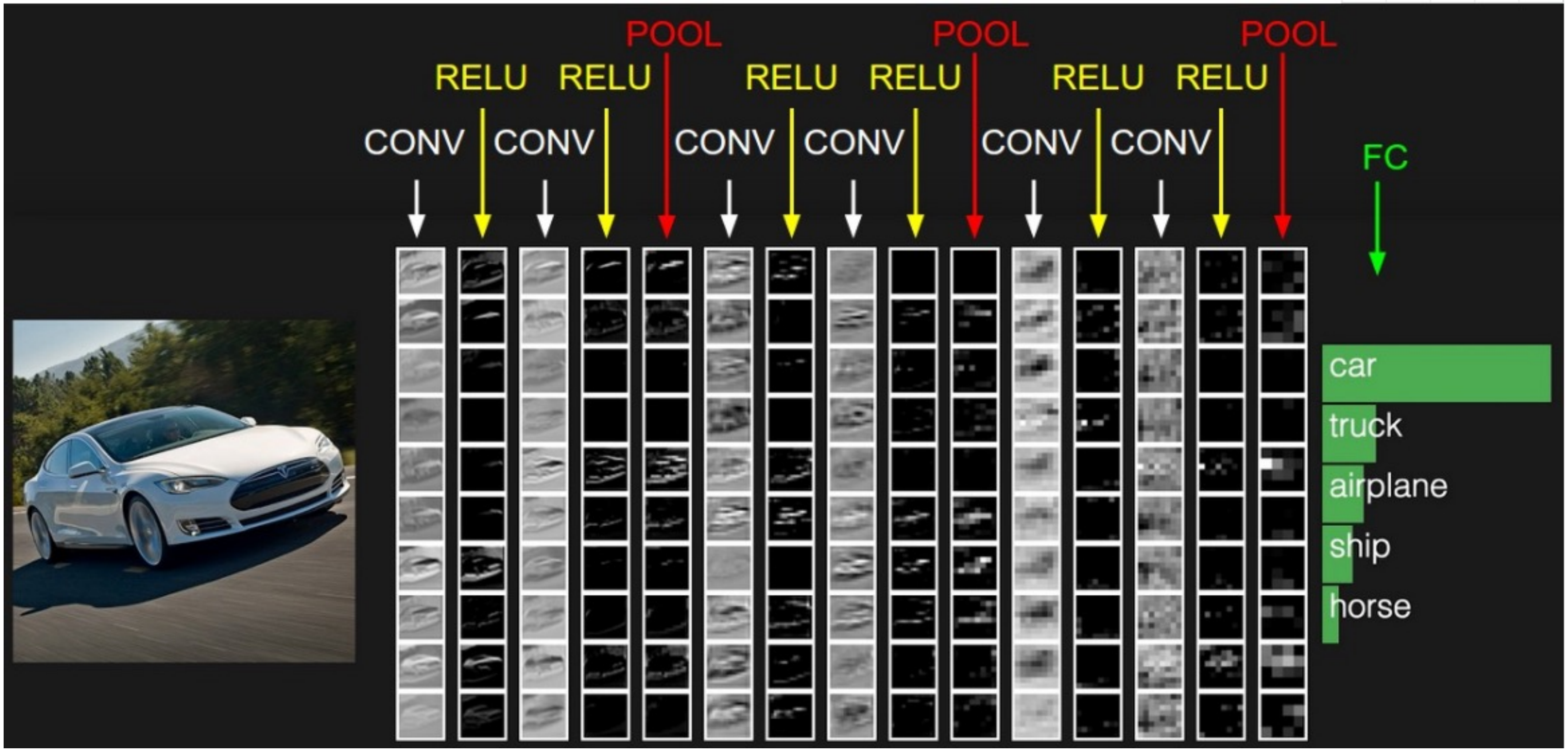
Examples time:



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

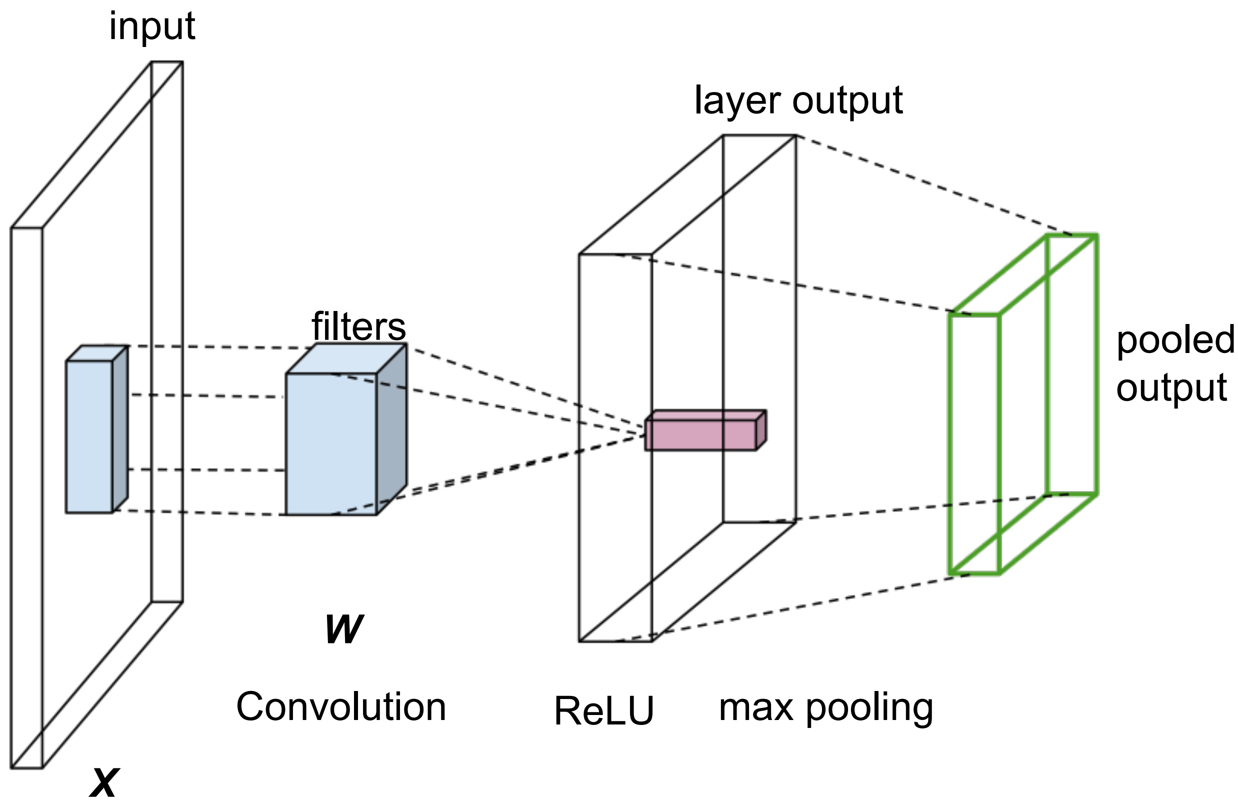
Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

A simple CNN structure



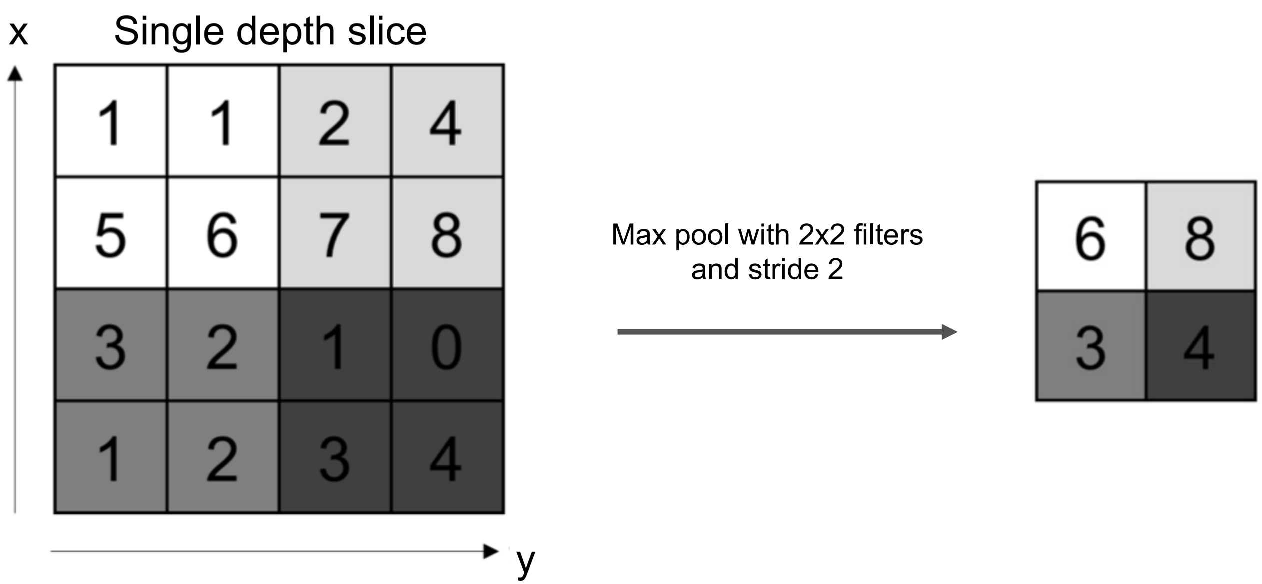
CONV: Convolutional kernel layer
RELU: Activation function
POOL: Dimension reduction layer
FC: Fully connection layer

Convolutional Neural Networks



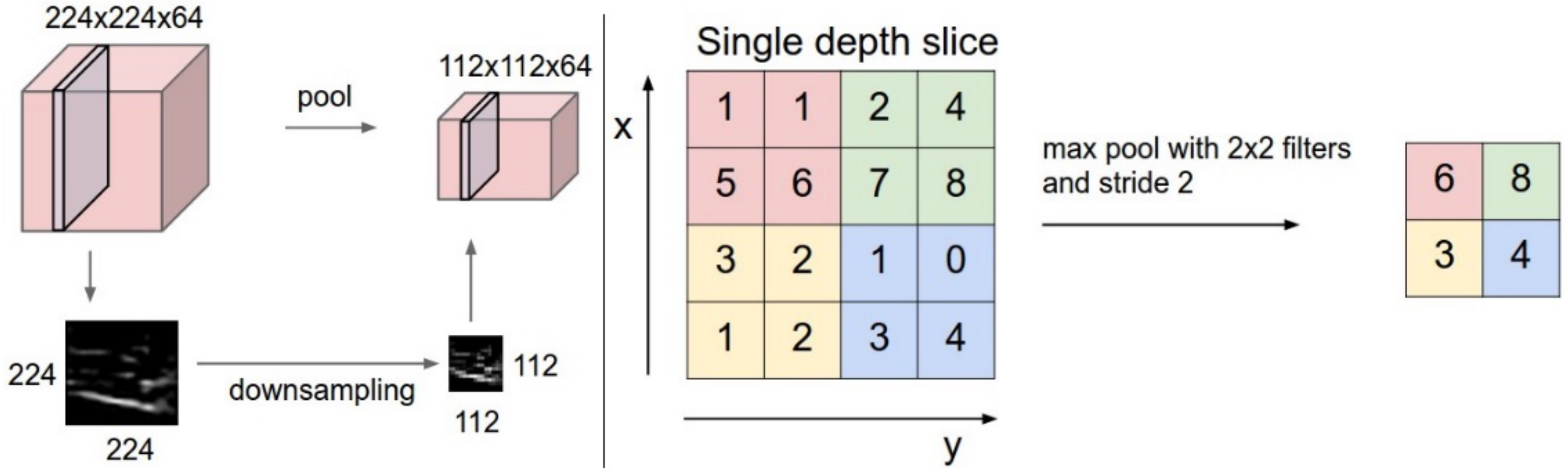
- Max pooling for **dimensionality reduction**
- Apply activation function (non-linearity) after every convolution operation
- **Conv+ ReLU+ Max**

Max pooling operation



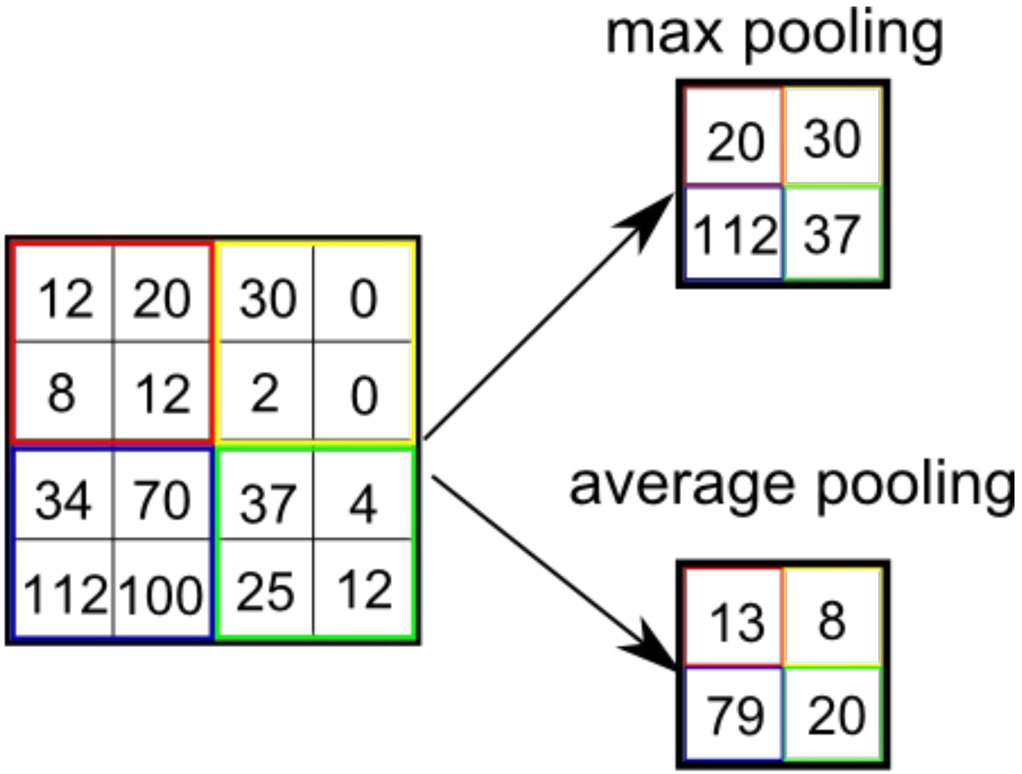
- Reduce dimensionality and preserve spatial invariance with **pooling**

Pooling layer

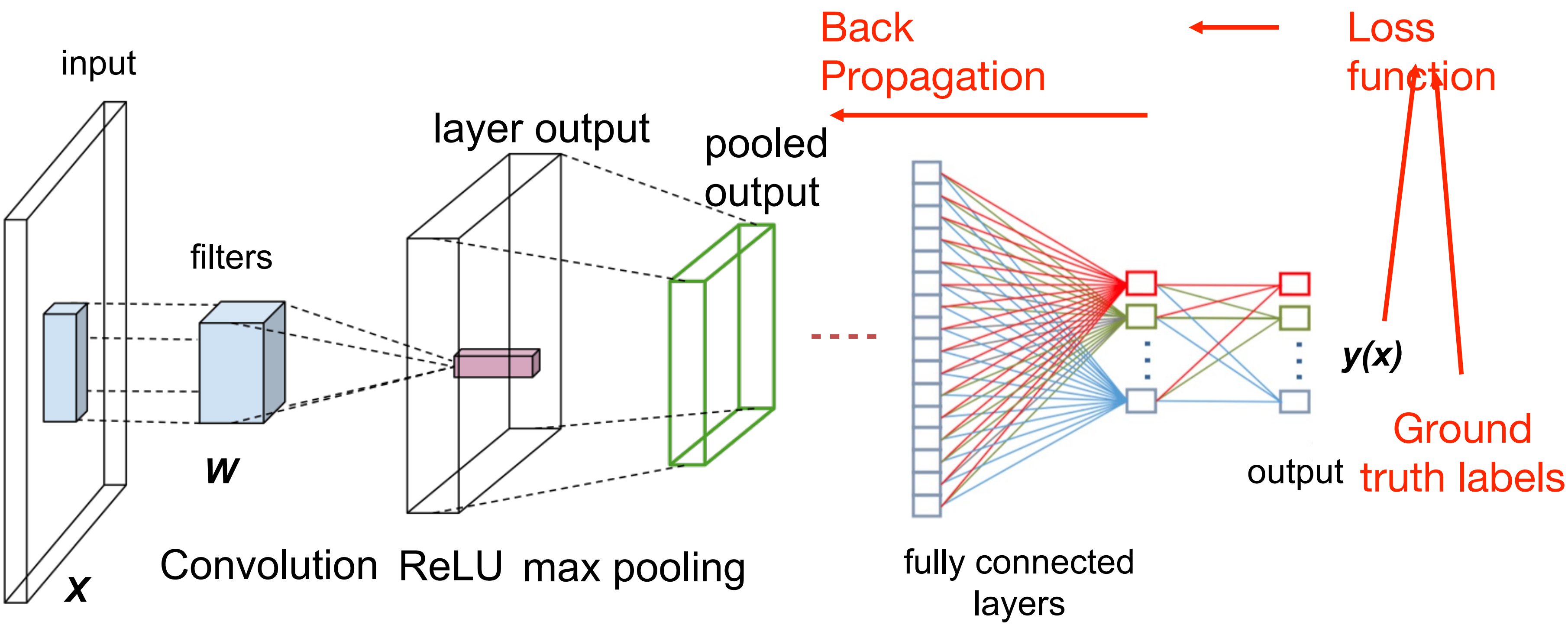


Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. **Left:** In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. **Right:** The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).

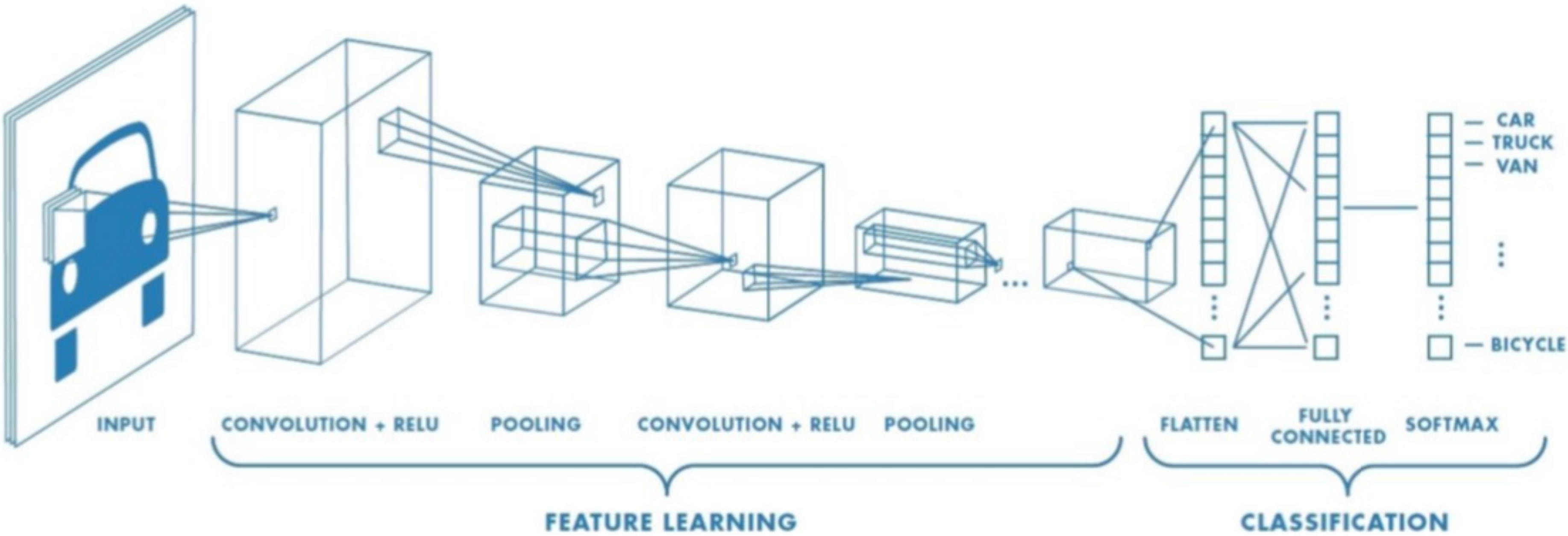
Pooling


max pooling
average pooling
12
20
30
0
8
12
2
0
34
70
37
4
112
100
25
12
20
30
112
37
13
8
79
20

Convolutional Neural Networks



CNN for classification



1. Feature Learning: Conv + ReLU + pooling

2. Classification:

- Fully connected layer uses these features for classifying input image
- Express output as **probability** of image belonging to a particular class

$$\mathrm{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

Learning Resources

- Charu C. Aggarwal, Neural Networks and Deep Learning, Springer, 2018.
- Eugene Charniak, Introduction to Deep Learning, MIT Press, 2018.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016.
- Michael Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.
- Deng & Yu, Deep Learning: Methods and Applications, Now Publishers, 2013.

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

- https://www.youtube.com/watch?v=uapdILWYTzE&t=2172s
- https://www.youtube.com/watch?v=bNb2fEVKeEo&t=2949s

Thank you