AI Summer School 2024
Medical Imaging Informatics

University of Pittsburgh

Deep CNNs

Instructor: Ahmad P. Tafti, PhD, FAMIA

Learning Objectives

After completing this lecture, you should be able to:

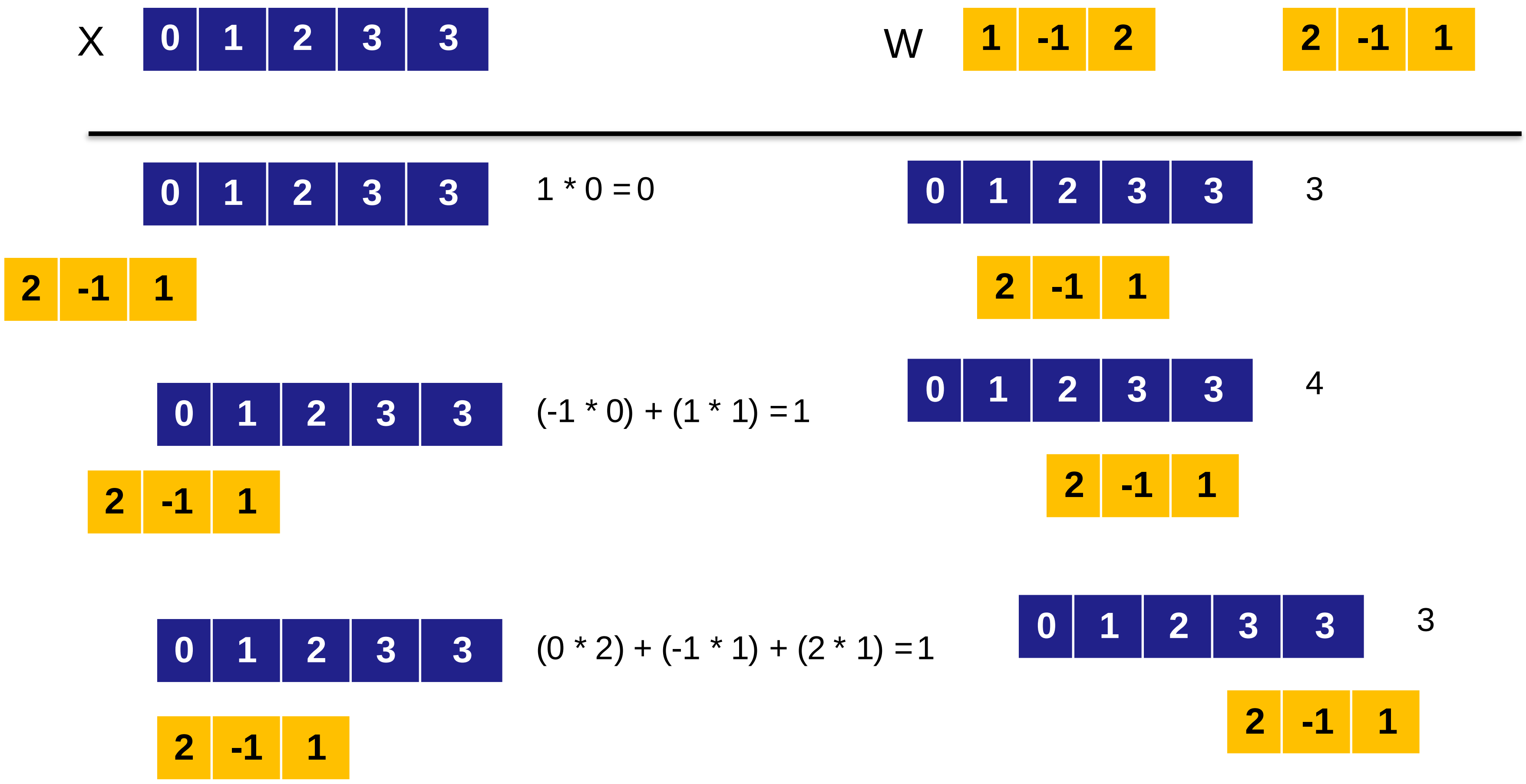
- Understand convolution and Convolution Neural Networks (CNNs)
- Explain convolution layer
- Discuss pooling layer and fully-connected layer

Outline

- CNNs

CONVOLUTION

- In mathematics, Convolution is an operation which does the integral of the product of 2 functions (e.g., 2 signals), with one of the signals flipped.



CONVOLUTION

0	1	2	3	3	6

2	-1	1

Output: X

0	1	1	3	4	3	6

Applications

- Filter signals
- See how much a signal is correlated to another
- **Pattern recognition in signals**

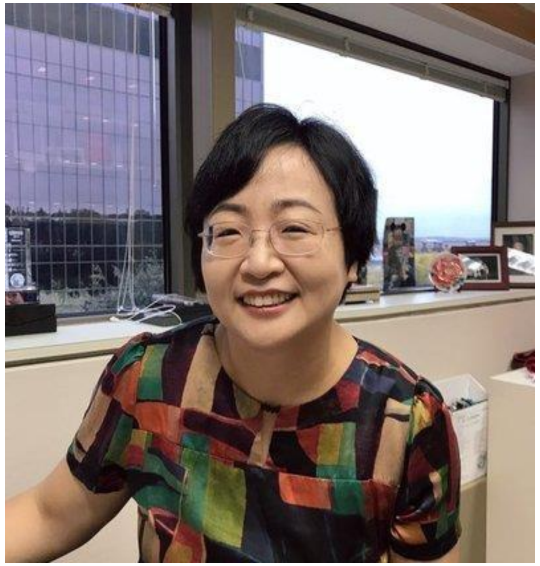
CONVOLUTIONAL NEURAL NETWORKS: WHY?

- Why do shallow fully connected neural networks not work when the input is an image?

- There are two main reasons:

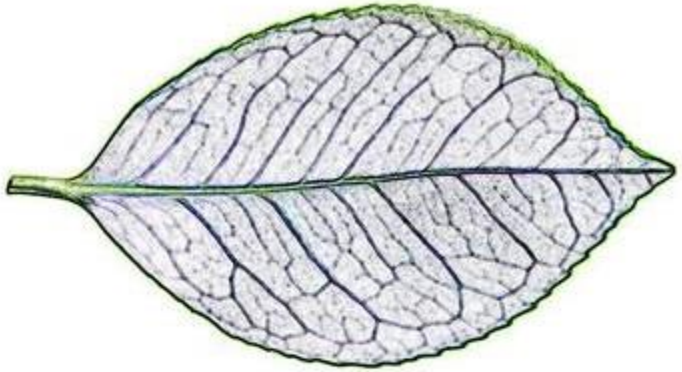
(1)The input consists of 3,000,000 numbers, therefore many weights are needed for each node in the hidden Layer. Saying 100 nodes in the first layer, this corresponds to 300,000,000 weight parameters required to define only this layer. More **parameters** mean **more training data** is needed to prevent **overfitting**. This leads to more time required to train the model.

(2) Processing by Fully Connected Deep Feed Forward Networks requires that the image data be transformed into a linear 1-D vector. This results in a **loss of structural information**, including correlation between pixel values in 2-D.



[1000 * 1000 * 3] = 3,000,000

CONVOLUTIONAL NEURAL NETWORKS: WHY?



CNN: The layers

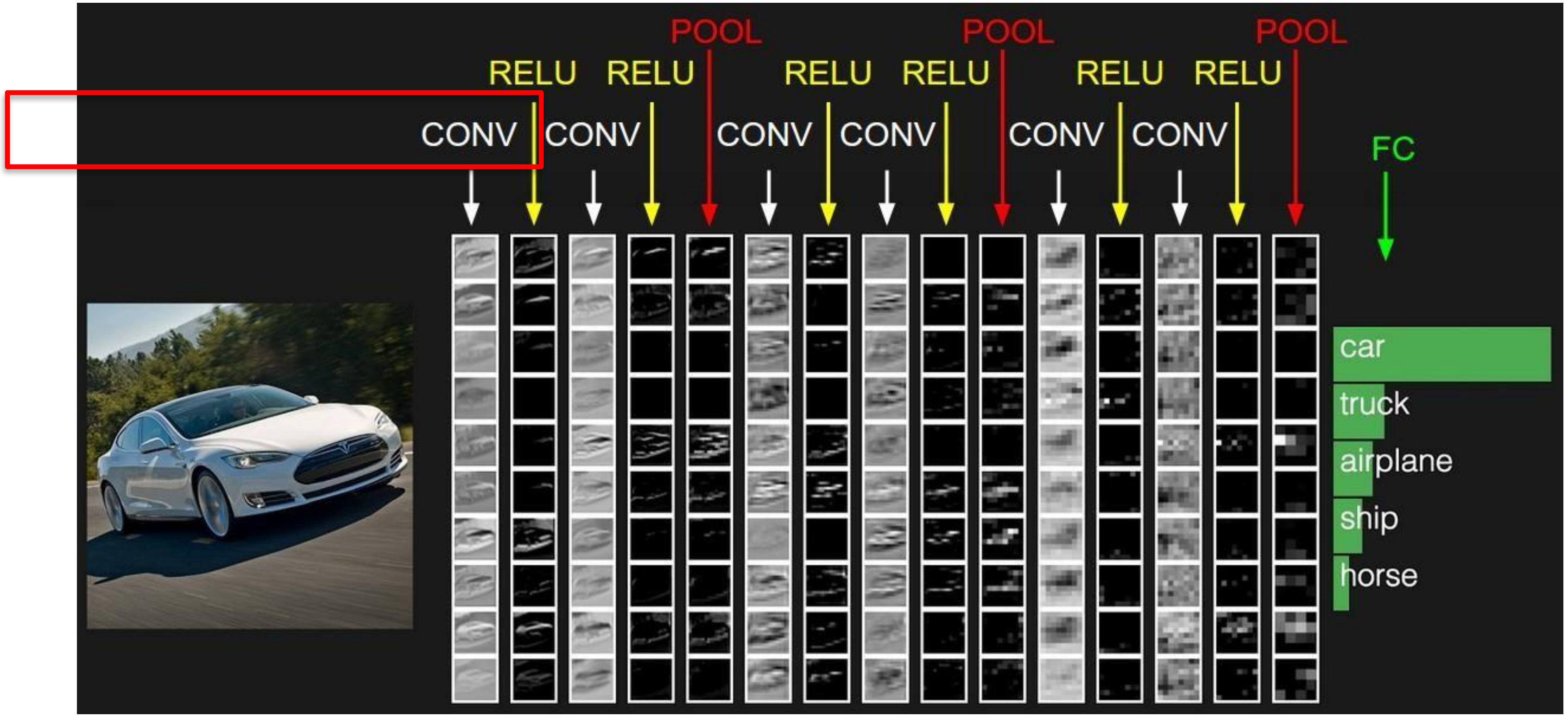
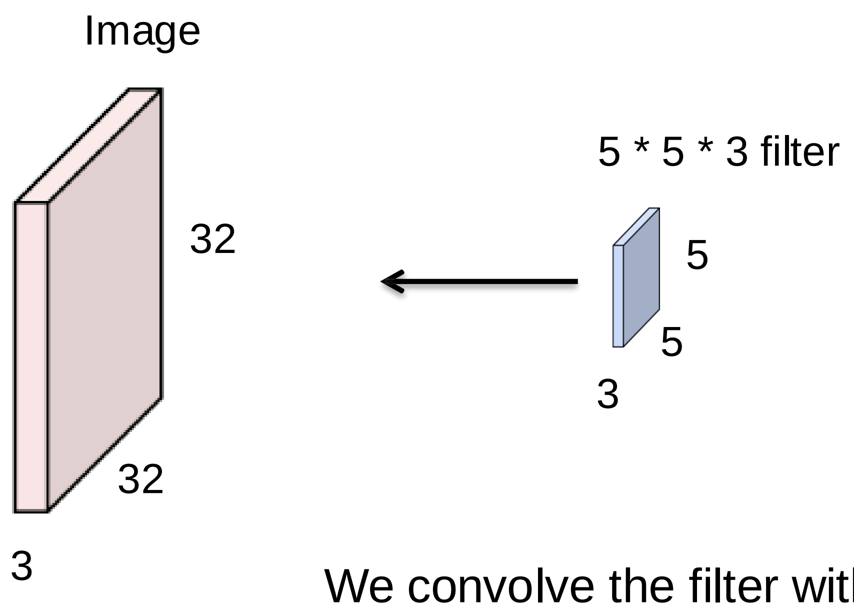


Image from: http://cs231n.github.io/convolutional-networks/

CONVOLUTIONAL LAYER



We convolve the filter with the image:
Slide over the image and compute dot products

Filter should have a same depth of the input image

CONVOLUTIONAL LAYER

Input Image: 5 * 5
Filter: 3 * 3

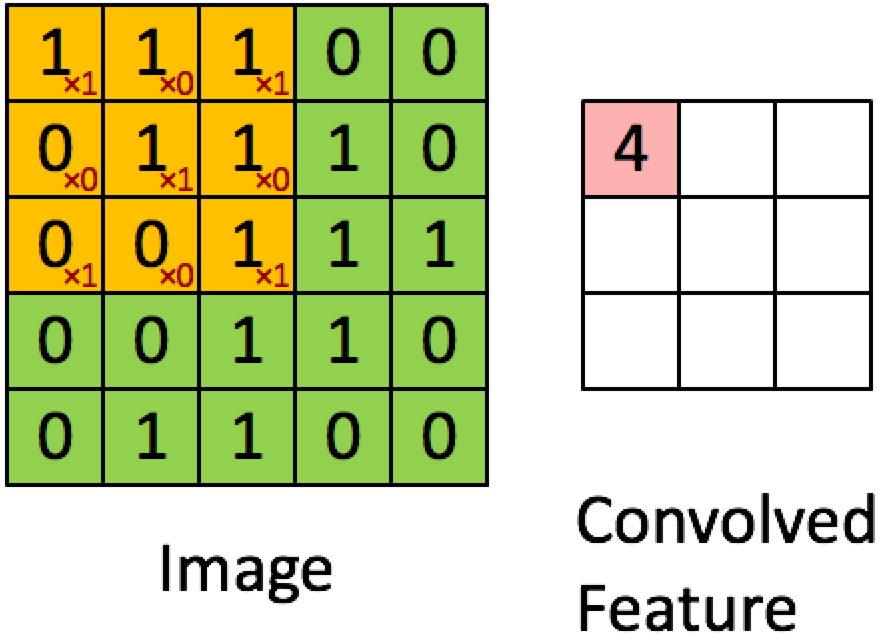
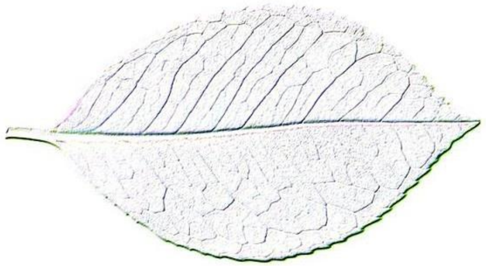


Image from: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/convolution.html

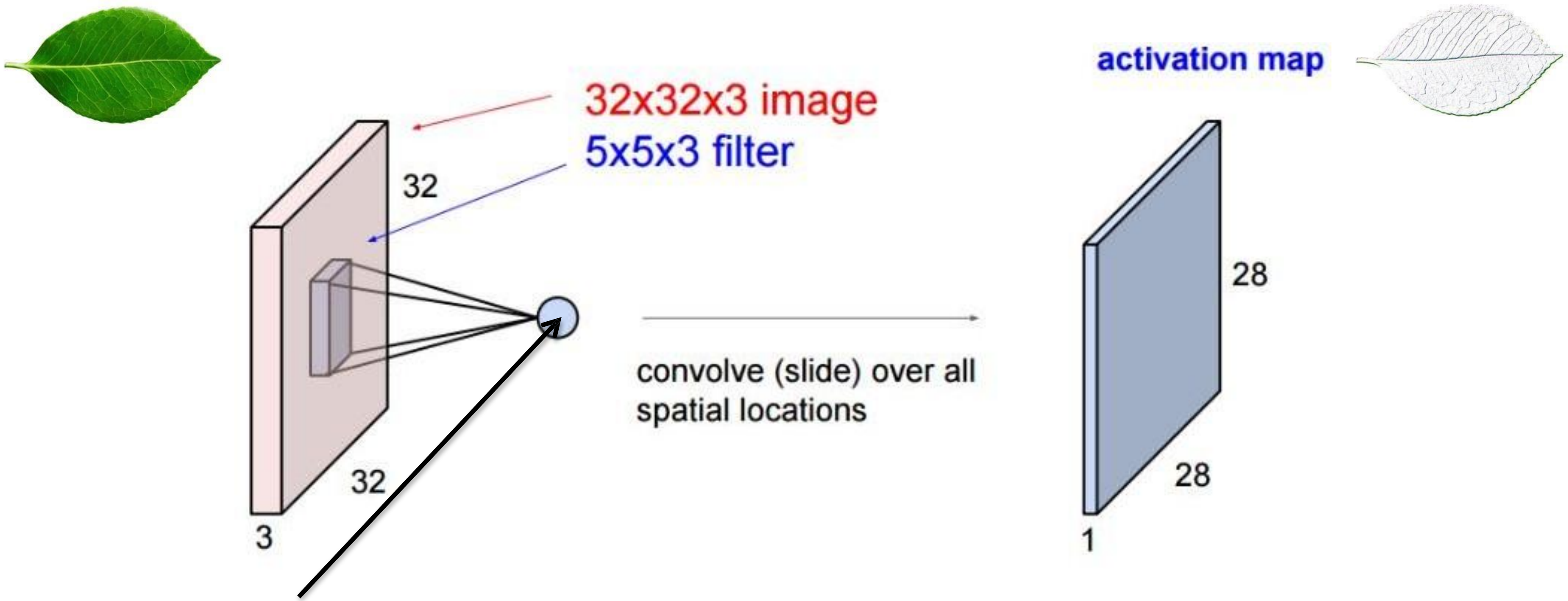
CONVOLUTIONAL LAYER

-1	-1	-1
-1	8	-1
-1	-1	-1



Convolution of an image (left) with an edge detector convolution kernel (middle). Right is the output.

Convolutional Layer



Just <u>one</u> number:
The results of taking a dot product (filter and 5 * 5 * 3 chunk of the image)

Slide over the image and compute dot products

Image from: https://legacy.gitbook.com/book/leonardoaraujosantos/artificial-inteligence

CONVOLUTIONAL LAYER

- In CNN, we are working with **multiple filters**. Each filter looks for a specific kind of **feature/pattern/concept** in the input image. For example, we want our convolution layer to look for 6 different patterns. So, our convolution layer will have 6 number of 5x5x3 filters, each one looks for a specific pattern on the image.

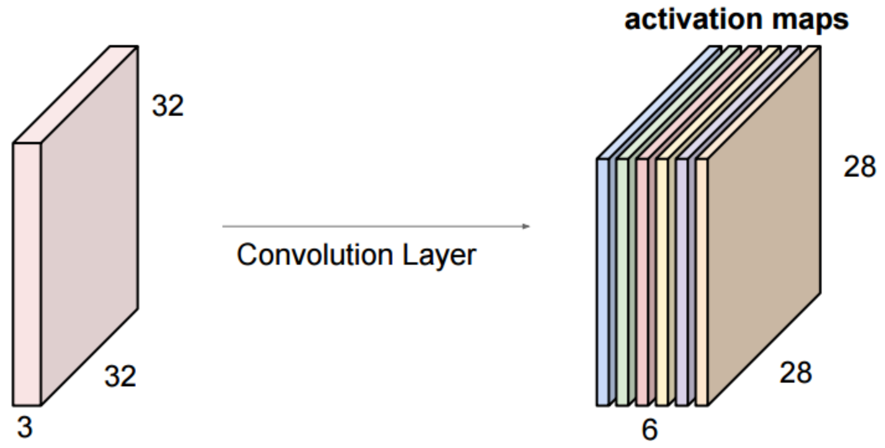
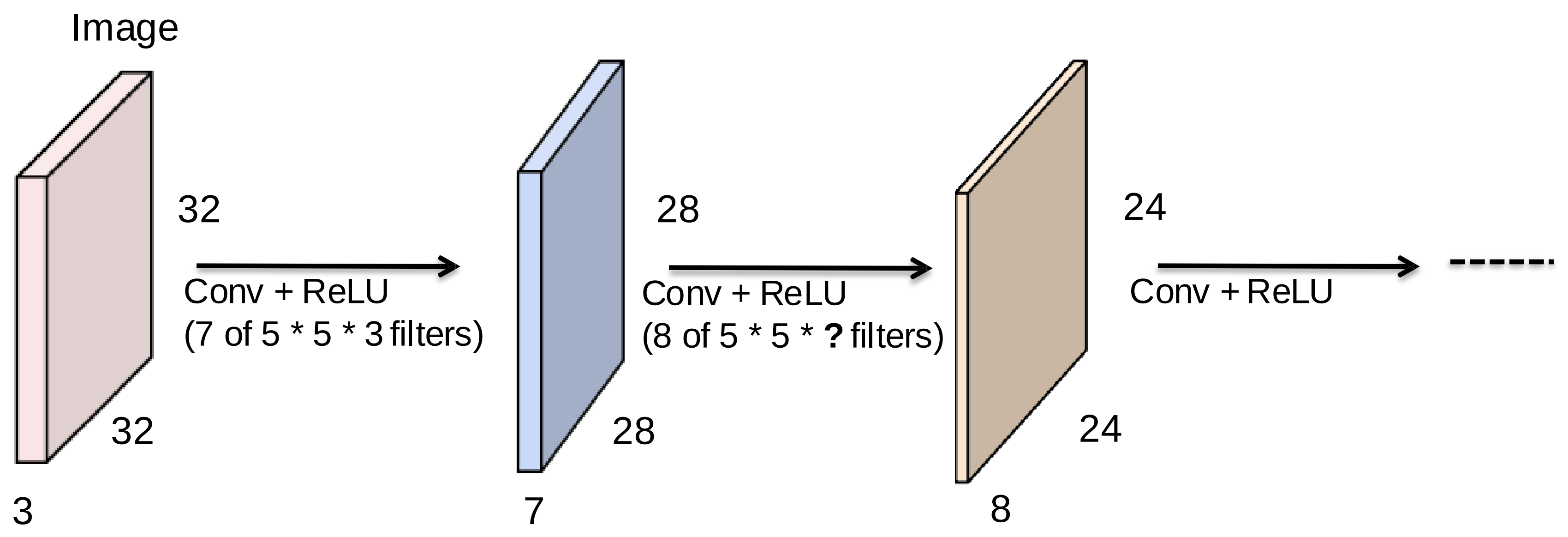


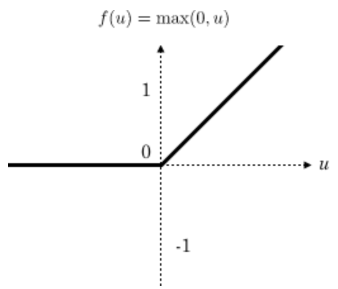
Image from: https://legacy.gitbook.com/book/leonardoaraujosantos/artificial-inteligence

- Stacking these up to make a new image of size 28 * 28 *6

CONVOLUTIONAL LAYER

- Convolution itself is a linear kind of operation. There is a need to add at the end of the convolution layer a non-linear layer, called ReLU activation. ReLU is the max function(x,0) with input x matrix from a convolved image. ReLU then sets all negative values in the matrix x to zero and all other values are kept constant.





CNN: The layers

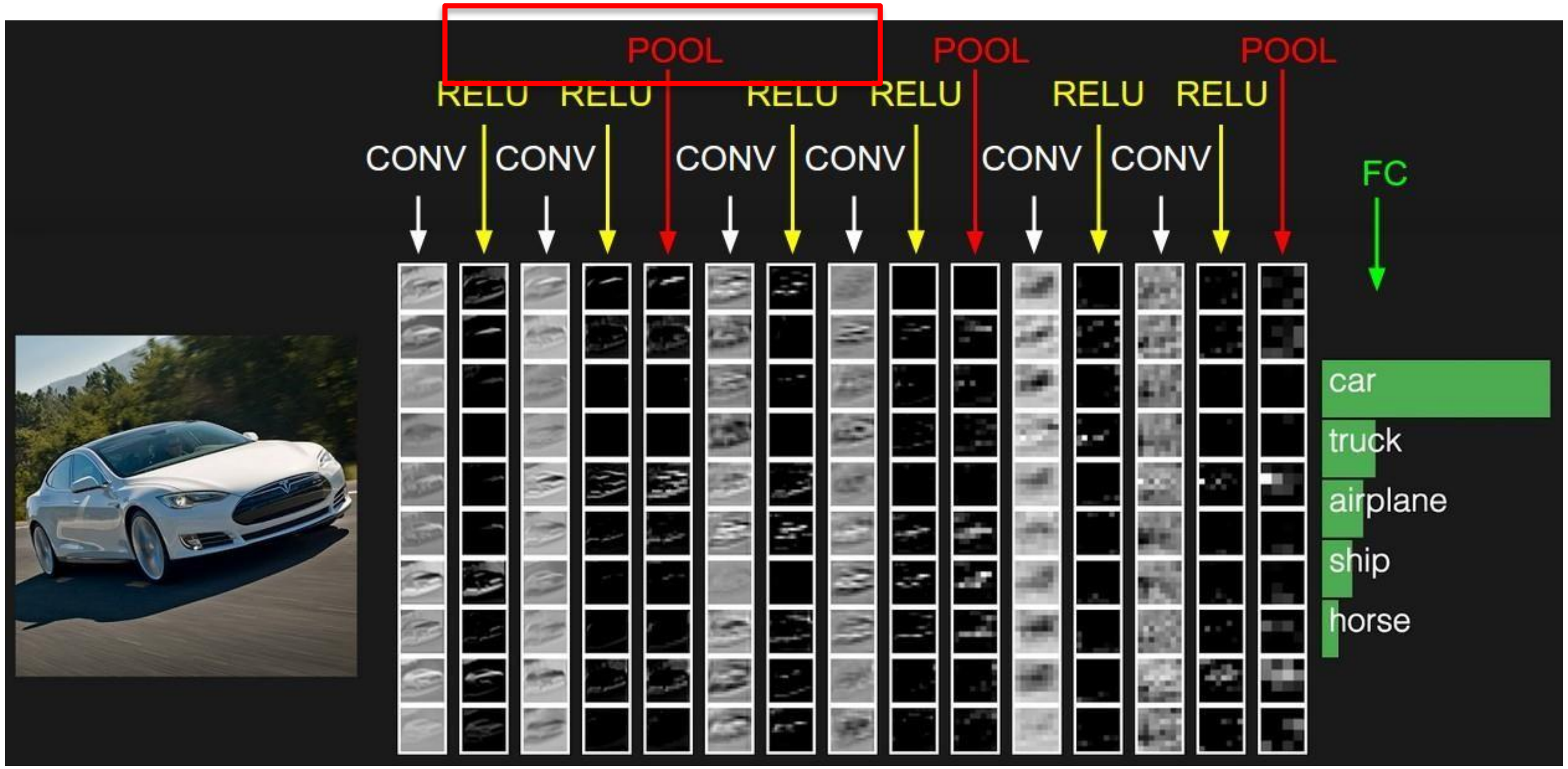
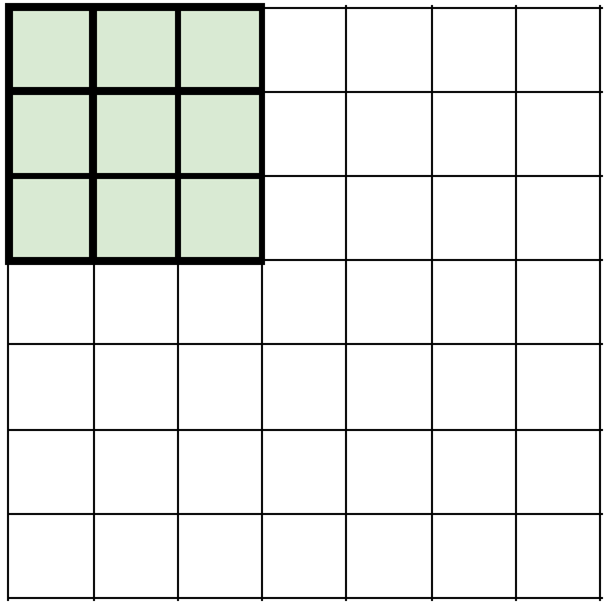


Image from: http://cs231n.github.io/convolutional-networks/

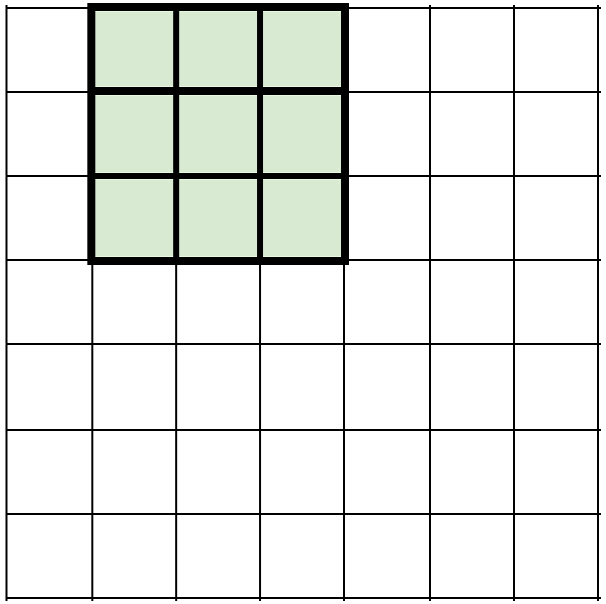
CONVOLUTIONAL NEURAL NETWORKS: A CLOSER LOOK

Input image: 7 * 7
Filter size: 3 * 3
Stride: 1



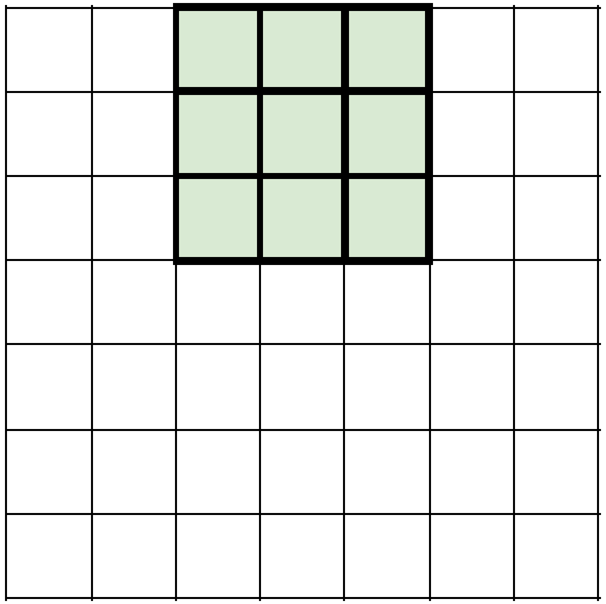
CONVOLUTIONAL NEURAL NETWORKS: A CLOSER LOOK

Input image: 7 * 7
Filter size: 3 * 3
Stride: 1



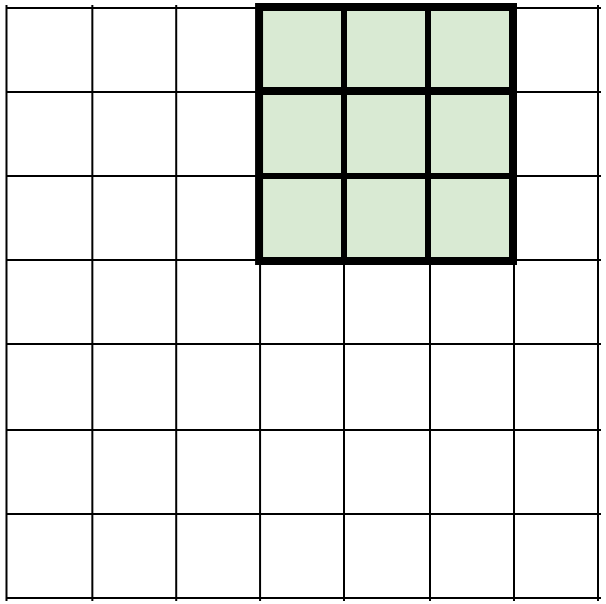
CONVOLUTIONAL NEURAL NETWORKS: A CLOSER LOOK

Input image: 7 * 7
Filter size: 3 * 3
Stride: 1



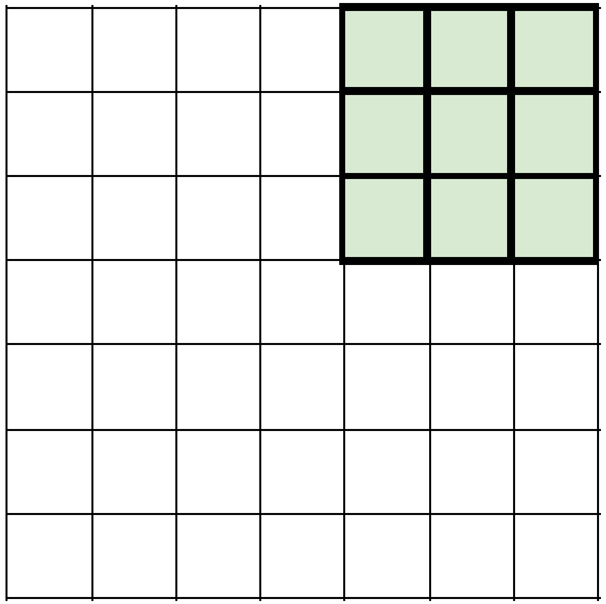
CONVOLUTIONAL NEURAL NETWORKS: A CLOSER LOOK

Input image: 7 * 7
Filter size: 3 * 3
Stride: 1



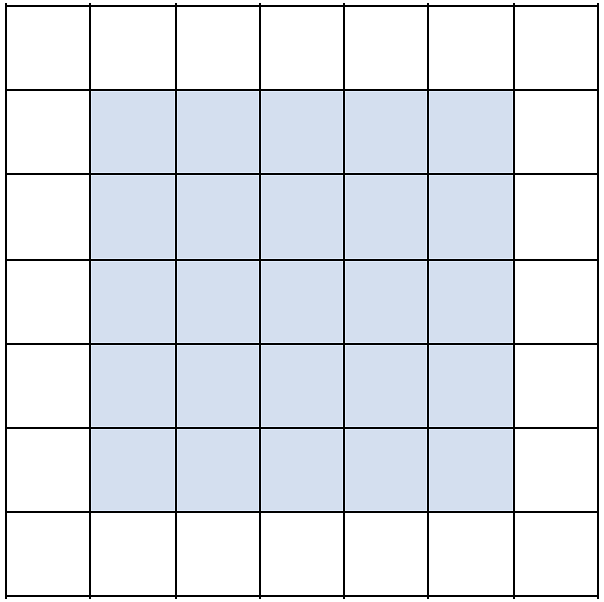
CONVOLUTIONAL NEURAL NETWORKS: A CLOSER LOOK

Input image: 7 * 7
Filter size: 3 * 3
Stride: 1



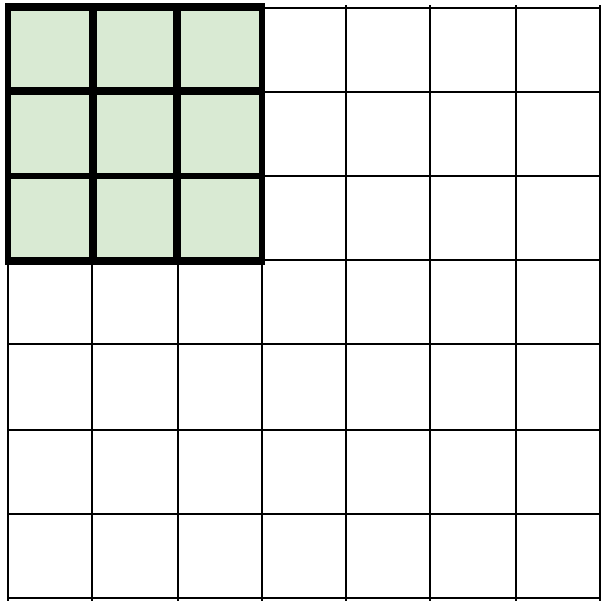
CONVOLUTIONAL NEURAL NETWORKS: A CLOSER LOOK

Output: 5 * 5



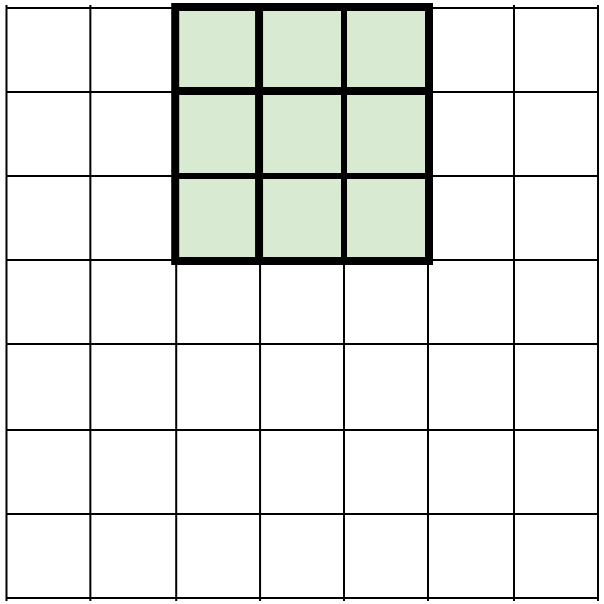
CONVOLUTIONAL NEURAL NETWORKS: A CLOSER LOOK

Input image: 7 * 7
Filter size: 3 * 3
Stride: 2



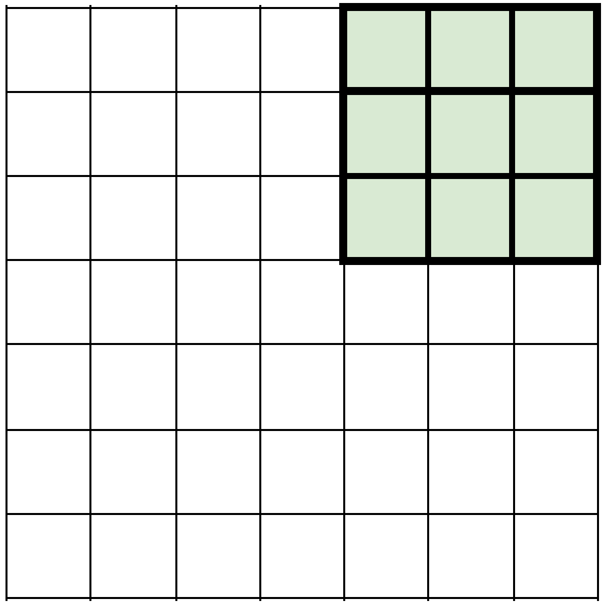
CONVOLUTIONAL NEURAL NETWORKS: A CLOSER LOOK

Input image: 7 * 7
Filter size: 3 * 3
Stride: 2



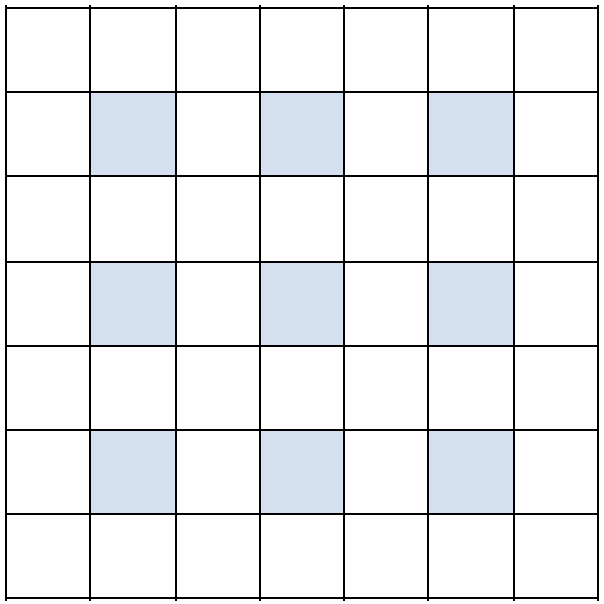
CONVOLUTIONAL NEURAL NETWORKS: A CLOSER LOOK

Input image: 7 * 7
Filter size: 3 * 3
Stride: 2



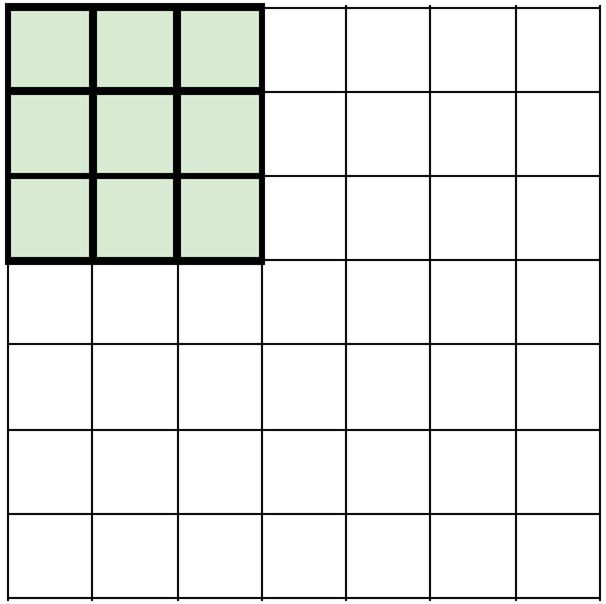
CONVOLUTIONAL NEURAL NETWORKS: A CLOSER LOOK

Output: 3 * 3



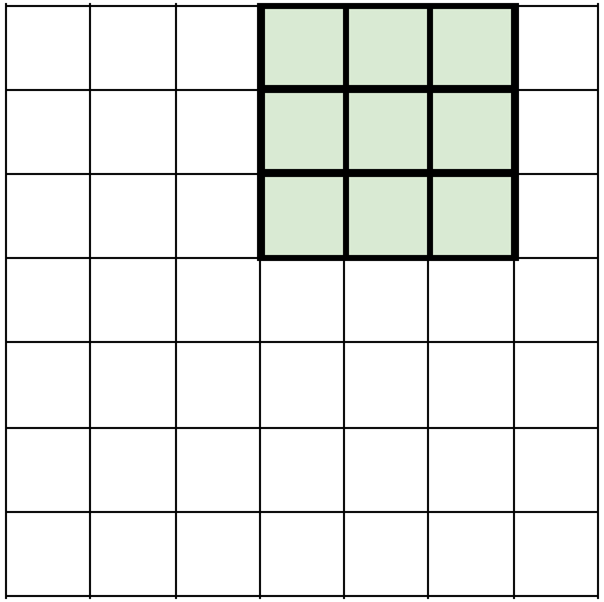
CONVOLUTIONAL NEURAL NETWORKS: A CLOSER LOOK

Input image: 7 * 7
Filter size: 3 * 3
Stride: 3



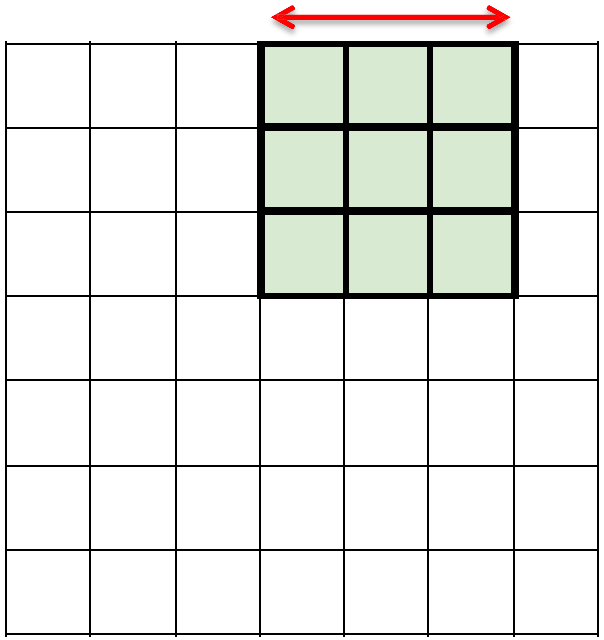
CONVOLUTIONAL NEURAL NETWORKS: A CLOSER LOOK

Input image: 7 * 7
Filter size: 3 * 3
Stride: 3



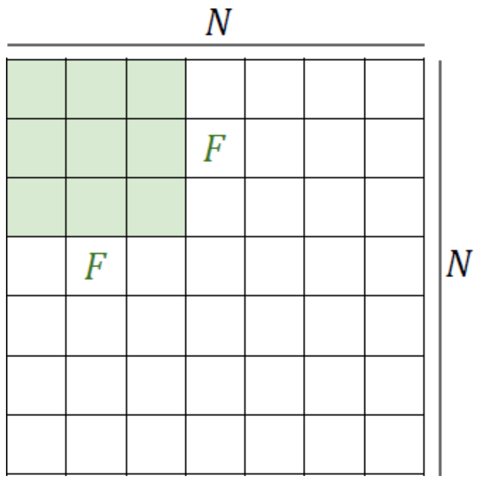
Convolutional Neural Networks: A Closer look

Input image: 7 * 7
Filter size: 3 * 3
Stride: 3



CONVOLUTIONAL NEURAL NETWORKS: A CLOSER LOOK

Output Size = (N – F) / Stride + 1



N = 7 F = 3

Stride = 1 Output Size = (7 – 3) / 1 +1 = 5

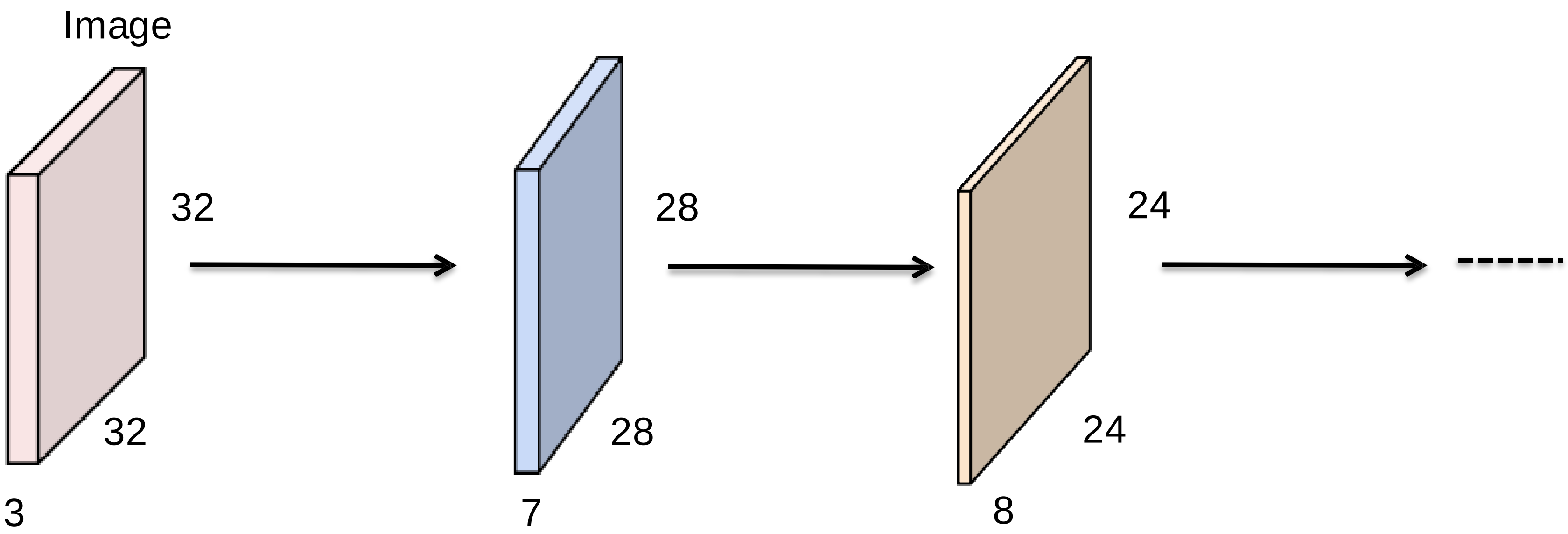
Stride = 2 Output Size = (7 – 3) / 2 + 1 = 3

Stride = 3 Output Size = (7 – 3) / 3 + 1 = 2.33

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

CONVOLUTIONAL NEURAL NETWORKS

- We are condensing the data spatially! Too fast! What does that mean?

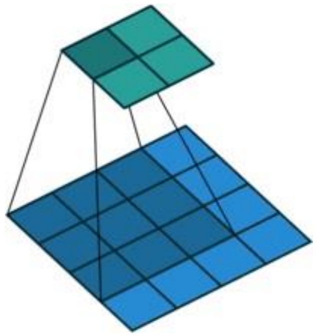


- Solution?

Zero Padding (pad): Add zeros on the image border to let the convolution output size be the same as the input image size.

CONVOLUTIONAL NEURAL NETWORKS

Input Size: 4 * 4
Filter Size: 3 * 3
Stride: 1
Padding: 0 (No Padding)



Input Size: 5 * 5
Filter Size: 3 * 3
Stride: 1
Padding: 1

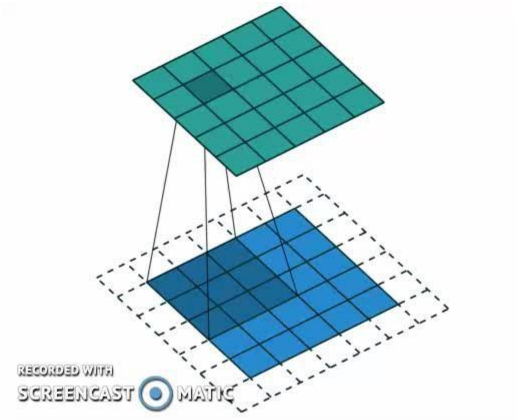


Image from: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/convolutional_neural_networks.html

Convolutional Neural Networks

Convolutional Layer:

It takes a data volume of size $W_1 * H_1 * D_1$

Hyper Parameters:

- Number of Filters (K)
- Filter Size (F)
- Stride (S)
- Zero Padding (P)

Common Configurations:

- K = 32, 64, 128, …
- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = 2

CONVOLUTIONAL NEURAL NETWORKS

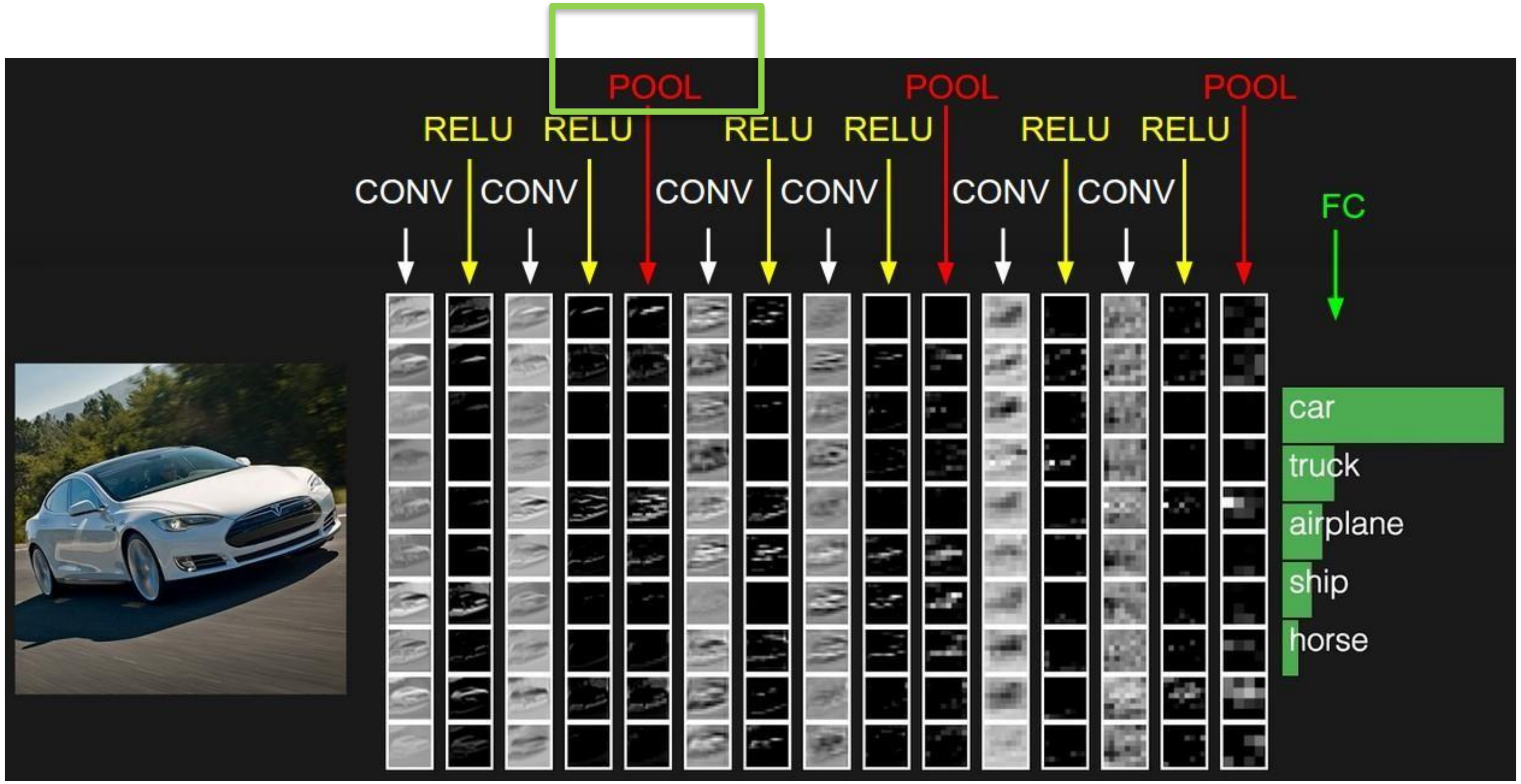
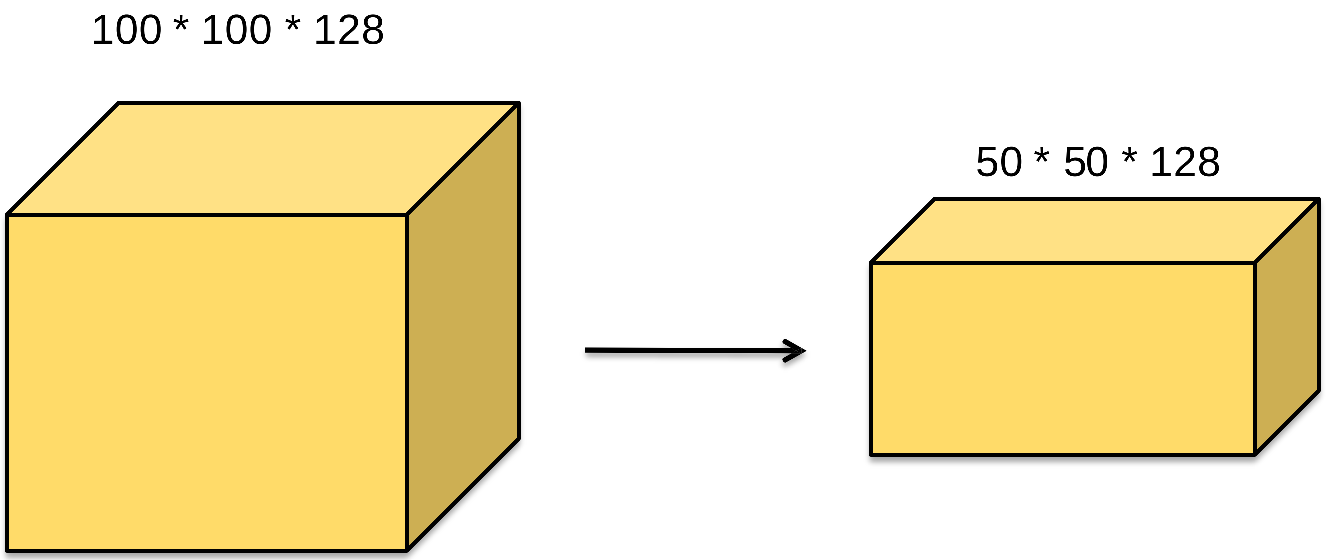


Image from: http://cs231n.github.io/convolutional-networks/

MAX POOL (POOLING)

It performs downsampling across the spatial dimensions (width, height). The representation would be smaller and more manageable.



Max Pool (Pooling): How it works?

Filter Size: 2 * 2
Stride: 2

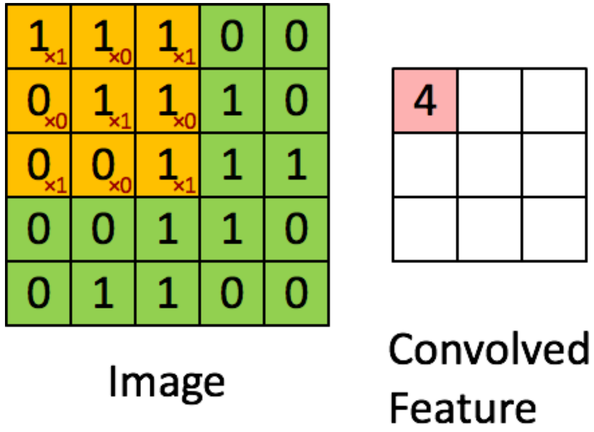
5	6	7	8
2	10	4	11
7	9	3	5
8	6	7	1

5	6	7	8
2	10	4	11
7	9	3	5
8	6	7	1

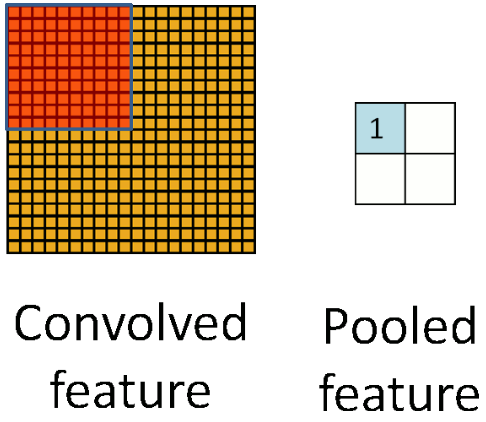
→

10	11
9	7

CONVOLUTION AND MAX POOL: A REVIEW



Convolution



Pooling

Image from: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/convolution.html

Convolutional Neural Networks

Max Pool Layer:

It takes a data volume of size $W_1 * H_1 * D_1$

Hyper Parameters:

- Filter Size (F)
- Stride (S)

Common Configurations:

- F = 2, S = 2
- F = 3, S = 2

FULLY CONNECTED LAYER

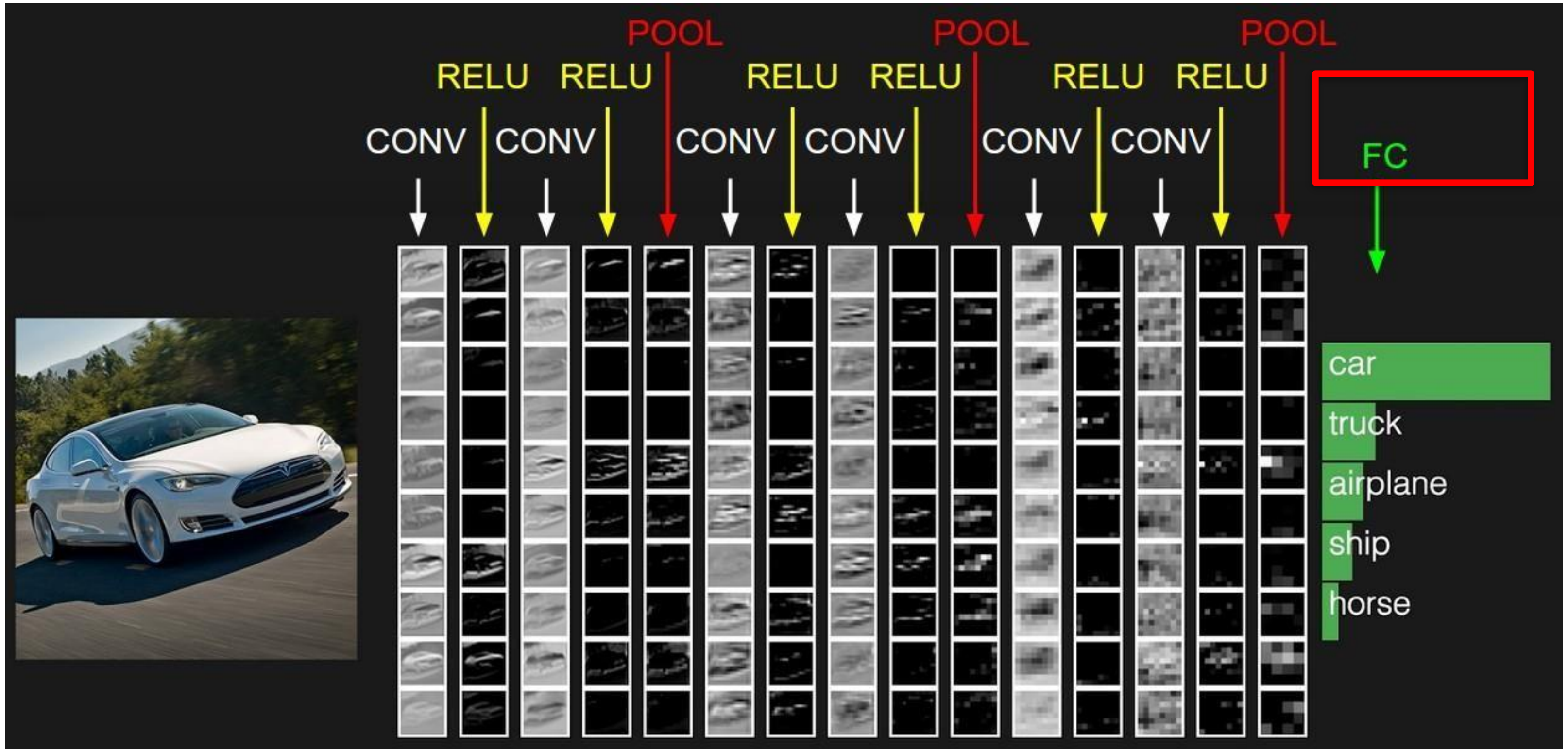


Image from: http://cs231n.github.io/convolutional-networks/

Fully connected layer

- It computes the class scores.

- This layer takes an input volume (the output of the Conv + ReLU + Pooling layer preceding it) and outputs an **N** dimensional vector, where **N** is the number of classes that we want to choose. For example, if we want to develop an object detection for Doors, Stairs, and Signs, then **N** would be 3.

- Each number in this **N** dimensional vector shows the probability of a class. For example, if the resulting vector for is [.1 .1 .80] for [Doors, Stair, Sign], then this represents a 10% probability that the image is a door, 10% probability that the image is a stair, and 80% probability that the image is sign.

CNN ARCHITECTURE: REVIEW

- **Input:** In our scenario, it holds the raw pixel values of an image (e.g., an image of width 32, height 32, and with three color channels R,G,B).

- **Convolutional Layer:** This layer filters (convolve) the inputs to provide very useful information appropriate for object modeling. These convolutional layers help to automatically extract the most valuable information for the task at hand without human designed feature selection. This layer will result in data volume such as [32 * 32 * 16] if we used for example 16 filters.

- **ReLU Layer:** will apply a pixelwise activation function, such as the $max(0,x)$ thresholding at zero. This layer keeps the size of the data volume unchanged (e.g., [32 * 32 * 16]).

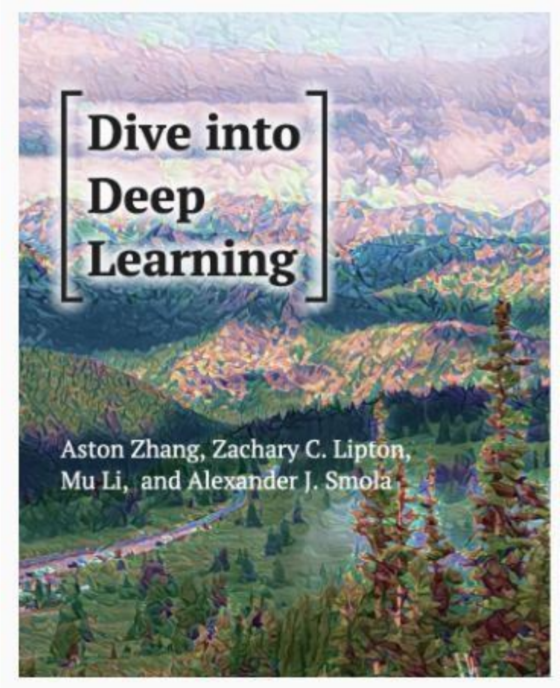
- **Pooling Layer**: It does a downsampling operation across the spatial dimensions (width, height), and will result in data volume such as [16 * 16 * 16].

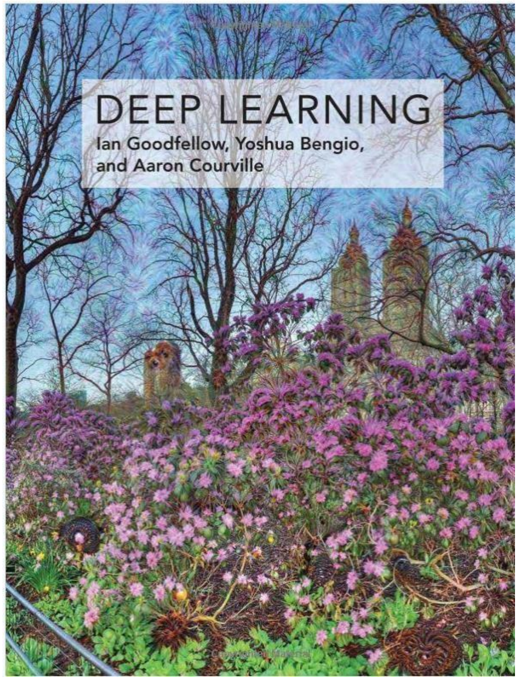
- **Fully Connected Layer:** This layer computes the class scores, and it will result in volume of size [1 * 1 * 3], where each of those 3 numbers correspond to a class score, such as among the 3 categories (doors, stairs, signs).

CNN ARCHITECTURE: REVIEW

CONV → **ReLU** → **Pooling** → **CONV** → **ReLU** → **Pool** → **FC** and **Softmax** (during training stage)

REFERENCES


Dive into
Deep
Learning
Aston Zhang, Zachary C. Lipton,
Mu Li, and Alexander J. Smola


DEEP LEARNING
Ian Goodfellow, Yoshua Bengio,
and Aaron Courville

Thank you!

Questions!