Module 2 :

Deep Networks

Convolutional Neural Networks

Géraldine Schaller-Conti, August 2024

Discussion Session

- Review of Notebook 5:
  - Vanishing/exploding gradients
    - Xavier/He initializations
    - Leaky ReLU, ELU, SELU
    - Batch normalization
    - Gradient clipping
  - Reusing pretrained layers
  - Faster optimizers
  - Learning rate scheduling
  - Regularization
    - L1, L2 regularization
    - Dropout
    - Max norm

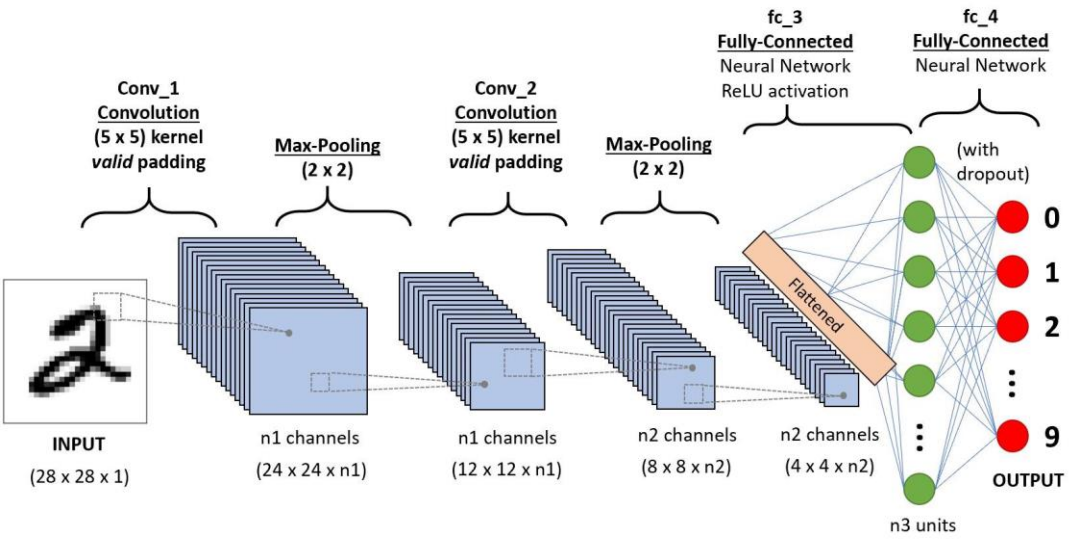
Bibliography

- Deep Learning book (Goodfellow, Bengio, Courville)
- Machine Learning @ Stanford (Prof Andrew Ng)
- Hands-On Machine Learning with Scikit-Learn & Tensorflow (Aurélien Géron)

Learning Objectives

- CNN components
- Most important architectures
- Object Detection
  - Self-driving cars example
- Face Detection

Convolutional Neural Networks



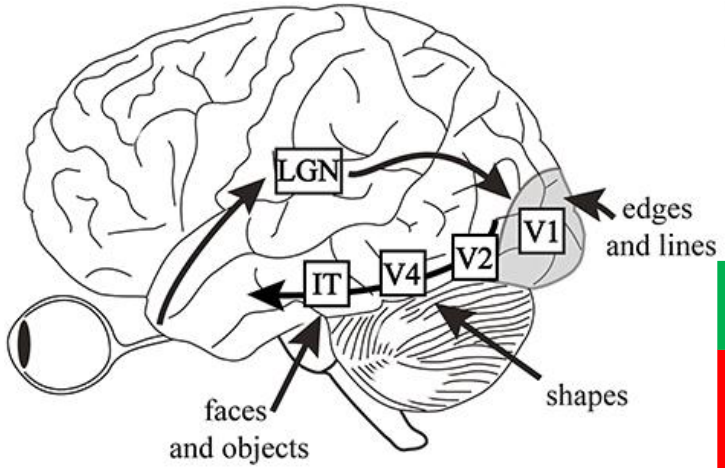
- Convolutional layer
- Padding, stride
- Filters
- Pooling layer

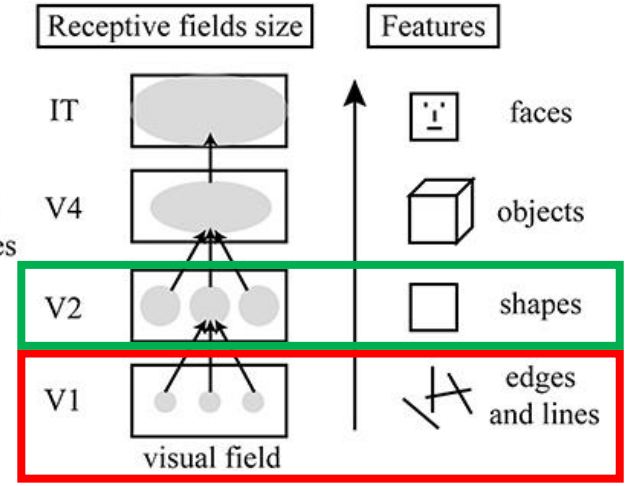
Introduction

- Convolutional Neural Networks (CNNs) emerged from the study for the brain's visual cortex
- Used in image recognition since the 1980s
  - Milestone (1998) with LeNet-5 architecture (LeCun)
- Huge improvements in the last few years due to :
  - Increased computational power
  - Amount of available training data
  - Tricks for training deep nets
- Everywhere: image search services, self-driving cars, automatic video classification,…

Visual Cortex

- Nobel prize in physiology (1981) : many neurons in the visual cortex react only to visual stimuli located in a limited region of the visual field (local receptive field)
  - Receptive fields overlap and tile the whole visual field

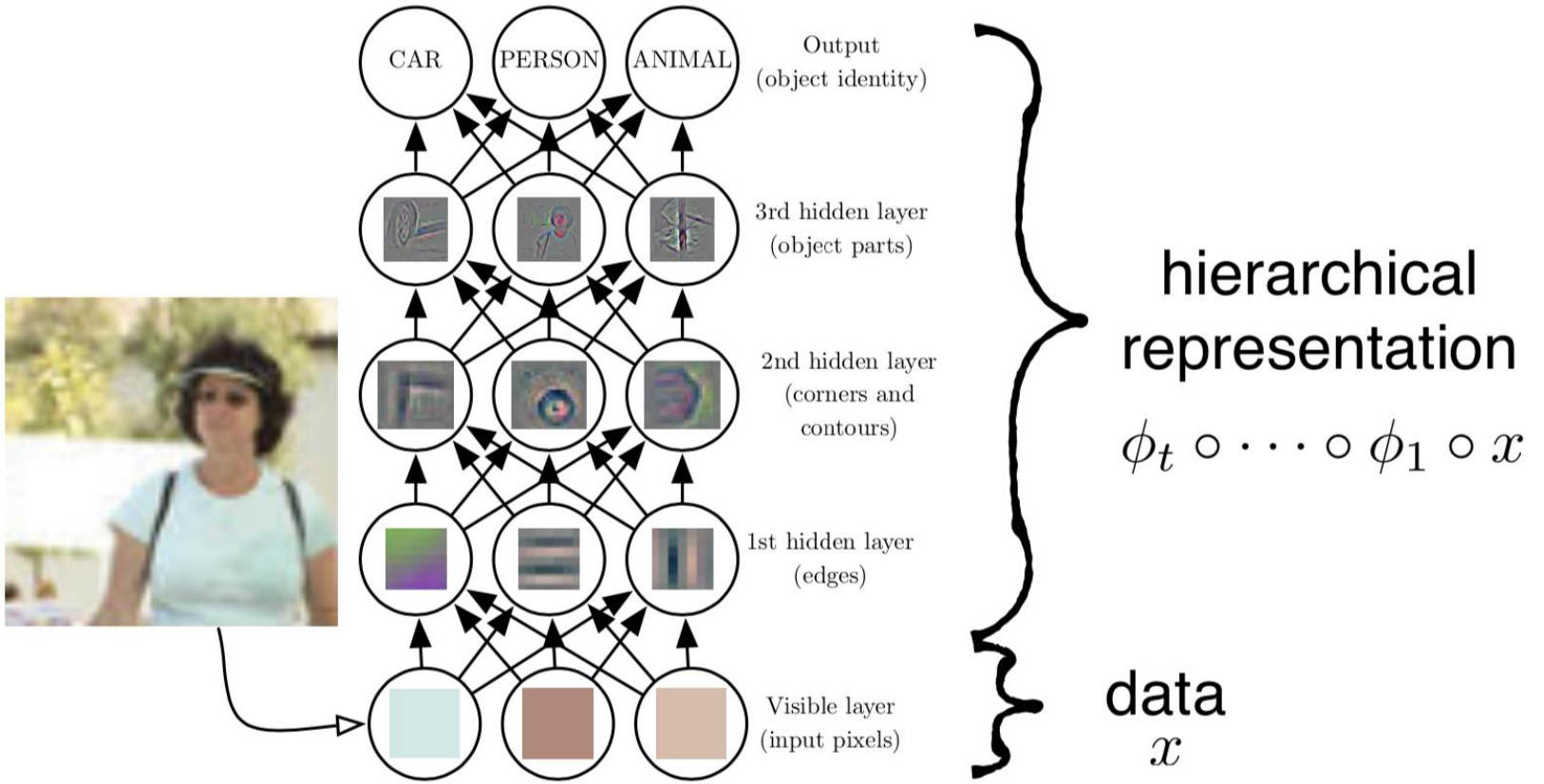




- Some neurons (with same receptive field) react to different line orientations (horizontal lines, lines with angle,…)

- Some neurons have larger receptive fields and react to more complex patterns : idea that higher-level neurons are based on the outputs of neighboring lower-level neurons

Convolutional Neural Network (CNN)



- Inspired by the organization of the visual cortex
- Its role is to reduce the images into a form that is easier to process without losing features that are critical for a good prediction

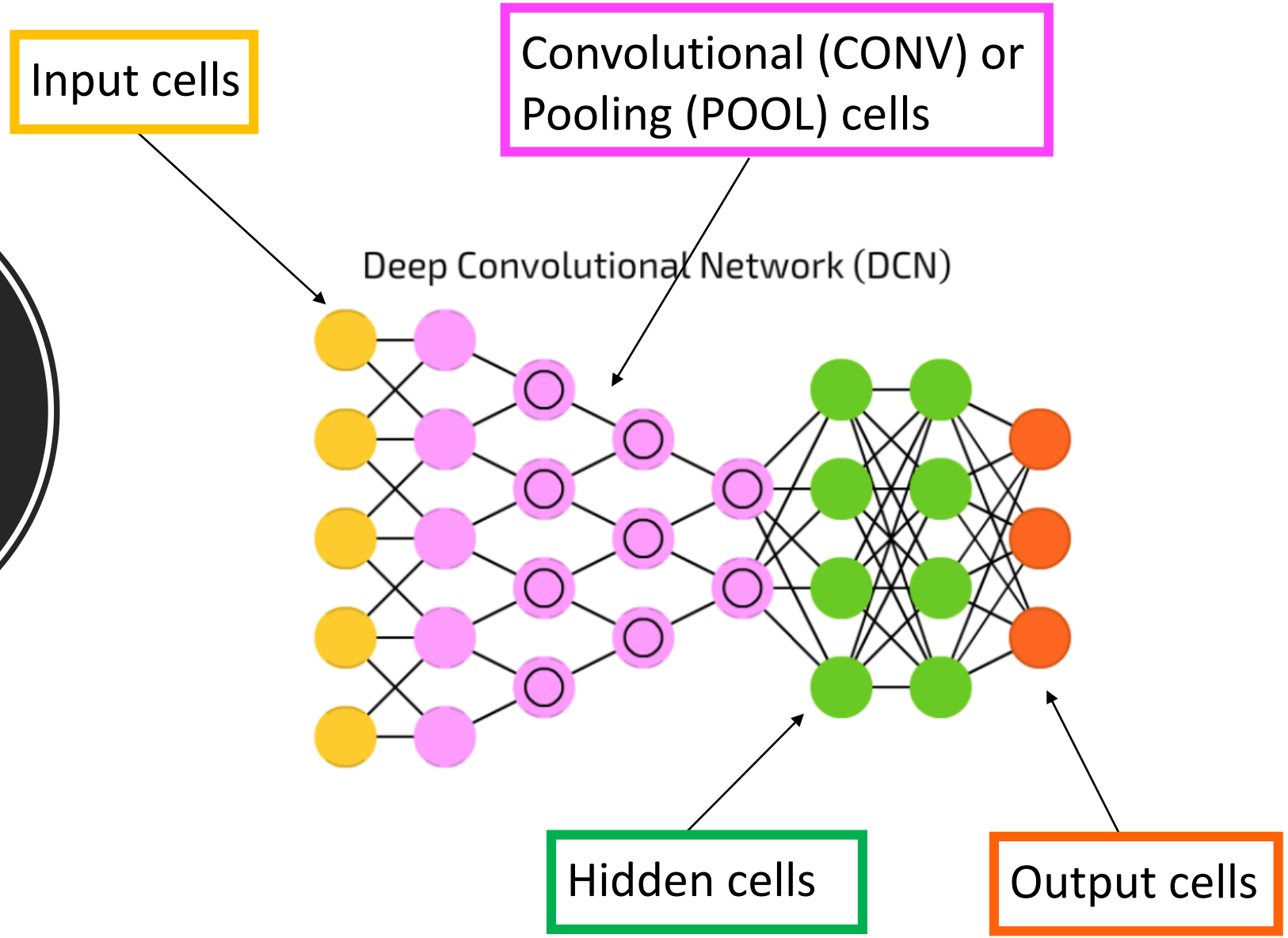
CNN use cases

- well suited for computer vision tasks
  - Image classification
  - Object detection

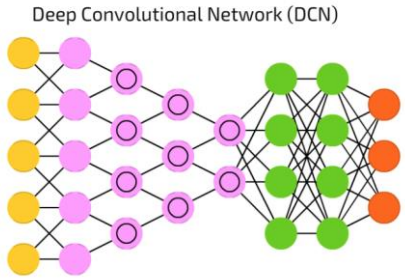
- More generally, specialized Neural Network for data arranged on a grid
  - Images
  - DNA sequences
  - …

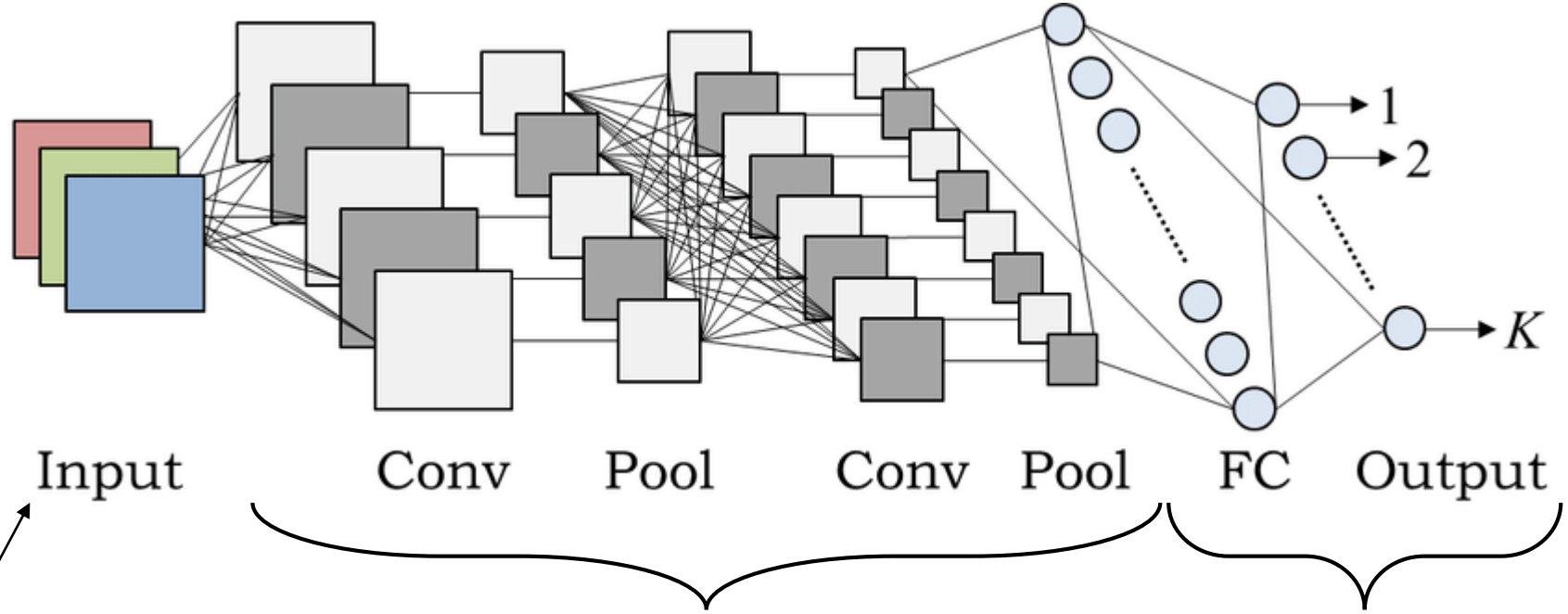
	A	C	G	T	W	S	M	K	R	Y	B	D	H	V	N	Z
A	1	0	0	0	1/2	0	1/2	0	1/2	0	0	1/3	1/3	1/3	1/4	0
C	0	1	0	0	0	1/2	1/2	0	0	1/2	1/3	0	1/3	1/3	1/4	0
G	0	0	1	0	0	1/2	0	1/2	1/2	0	1/3	1/3	0	1/3	1/4	0
T	0	0	0	1	1/2	0	0	1/2	0	1/2	1/3	1/3	1/3	0	1/4	0

CNN ingredients (1)



CNN ingredients (2)





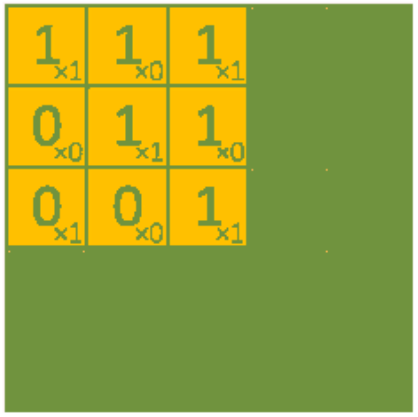
Feature learning

Classification

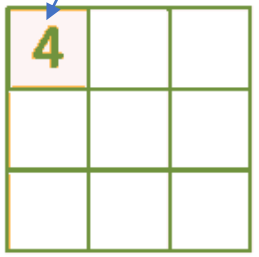
RGB image separated by its 3 color planes

Convolutional (CONV) layer : the Kernel

- Used to detect features (vertical/horizontal filter,…)



how well the pattern in the kernel matches the content in that part of the image



= Feature map

- Different kernels to create different feature maps → learn to see various patterns and details in images

1	0	-1
1	0	-1
1	0	-1

Vertical

1	1	1
0	0	0
-1	-1	-1

Horizontal

Vertical edges

Horizontal edges

Convolution on RGB image

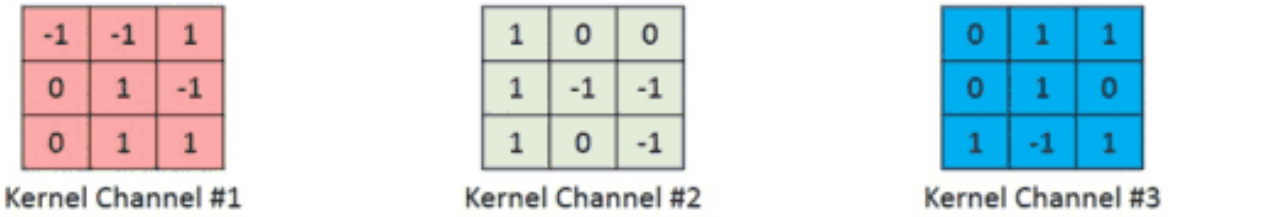
- In this example, use of a 3x3x3 filter

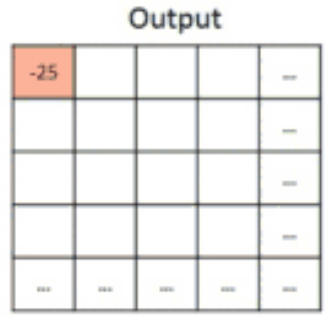
Width, height, #input channels

Padding ?

Stride ?

In a standard color image, the number of input channels is 3 (for red, green, and blue)





Kernels (Filters)

- Values inside a kernel are learned during the training process of the CNN

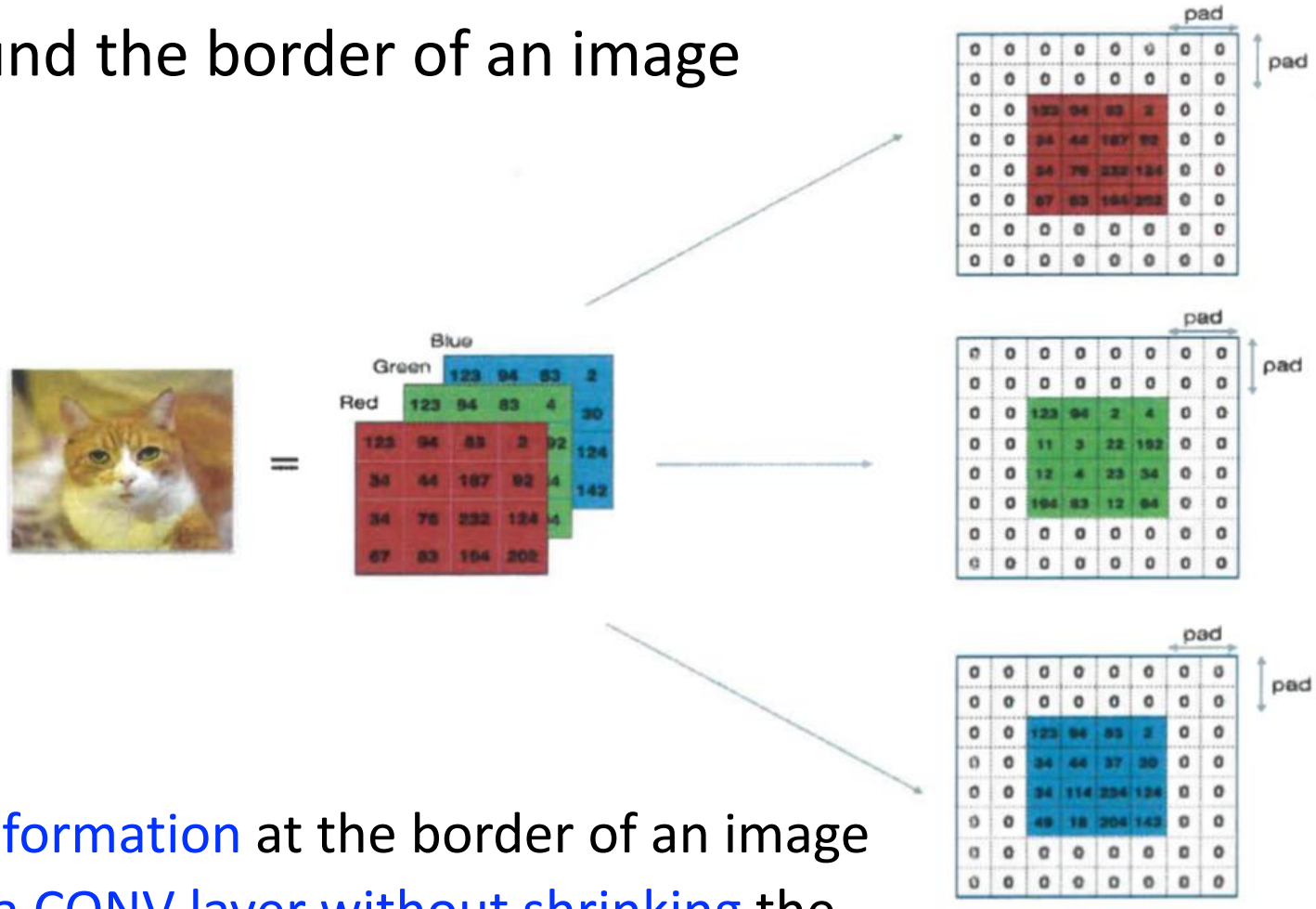
- Two aspects to consider for kernels :
  - Architecture :
    - how large they should be
    - How many input channels the kernel operates on
    - how many of them are applied to the input data

parameters = (Filter Width * Filter Height * Input Channels * Number of Filters / Stride) + Number of Filters

  - Initialization : how they should look at the start of training
    - Random initialization (Xavier/He initialization)
    - Transfer learning

Padding

- Adds zeros around the border of an image

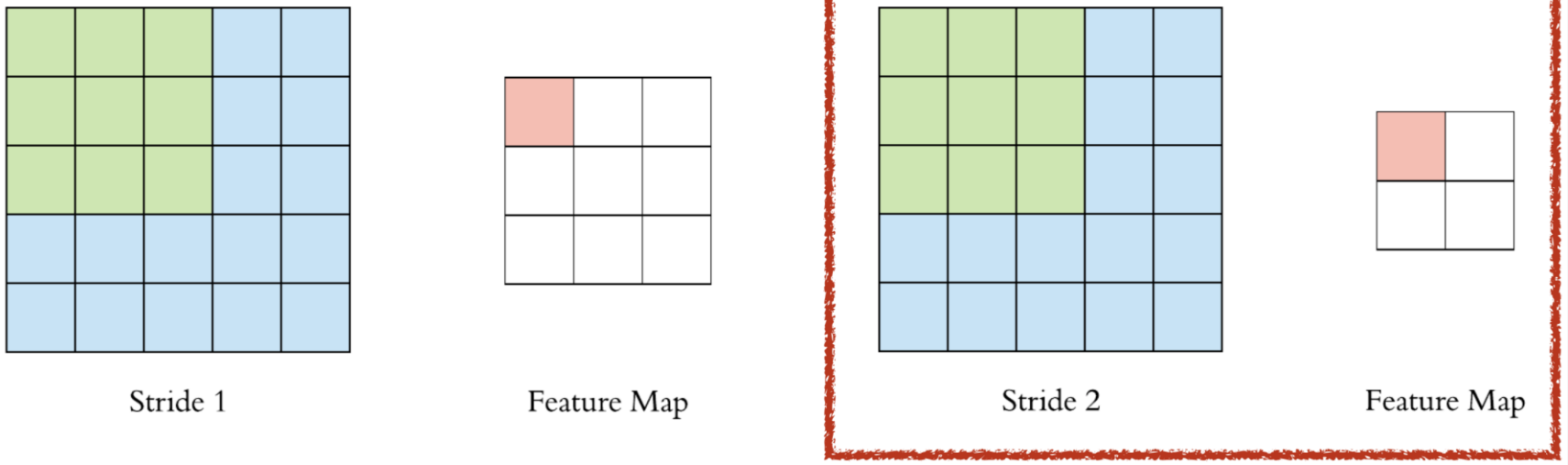


- Use cases :
  - Keeps more information at the border of an image
  - Allows to use a CONV layer without shrinking the height and width of the volumes
    - *important for deeper networks*

Stride

- By how much you move the filter

- Use cases :
  - Connect a large input layer to a much smaller layer by spacing out the receptive fields (reduce dimensionality)



increasing stride from 1 to 2

Exercise

- 300x300 RGB image, how many parameters does a hidden layer have (incl. bias) in the following cases:

a) Hidden layer with 100 neurons, each fully connected to the input (*no convolution*)

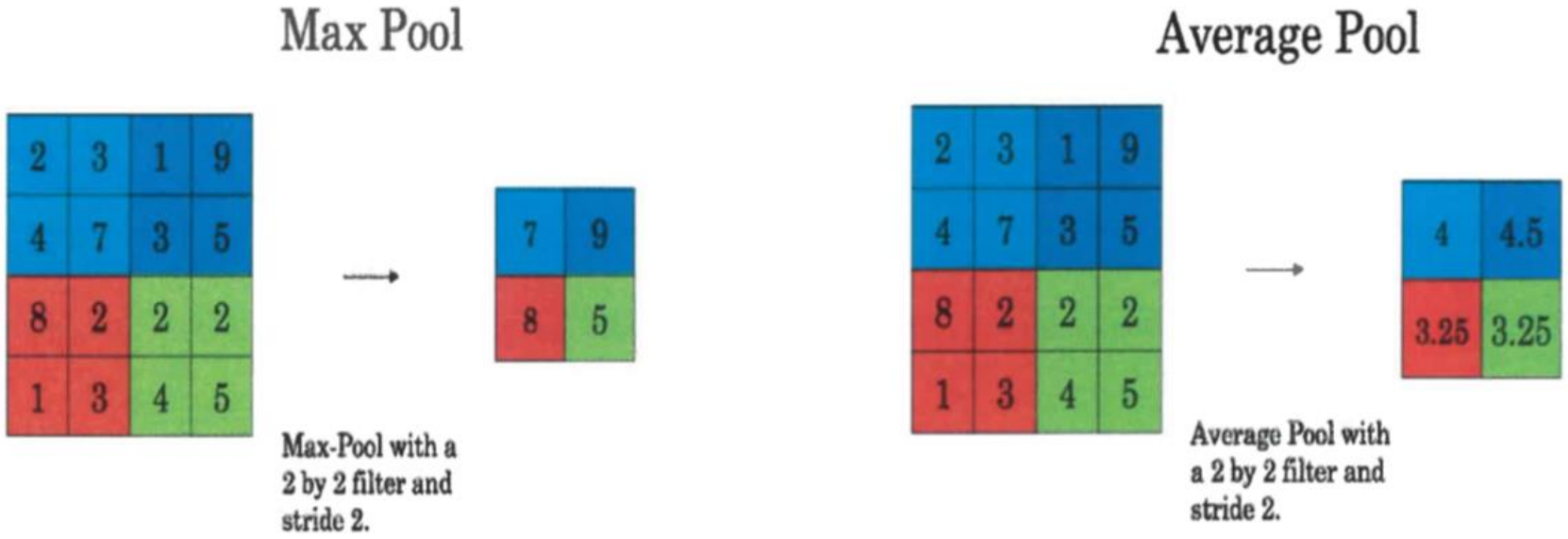
params = (number of input connections + 1) * number of neurons

a) Hidden *convolution* layer with 100 filters that are each 5x5

params = (Filter Width * Filter Height * Number of Input Channels / Stride^2) * Number of Filters + Number of Filters

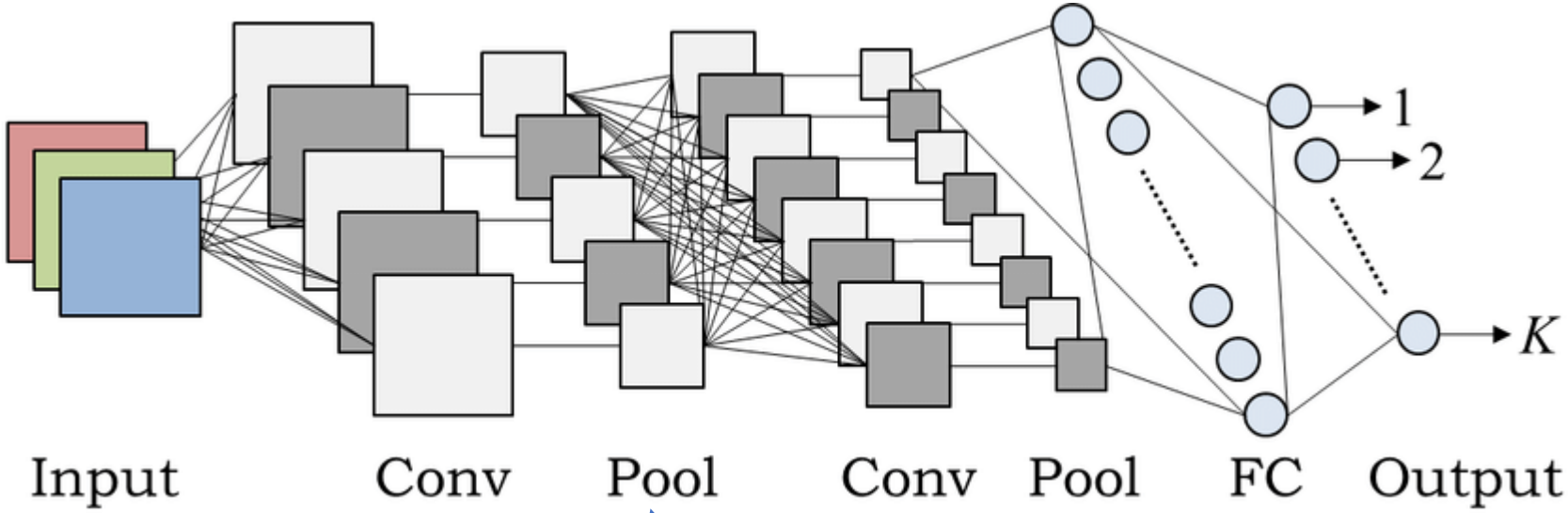
Pooling Layer

- Goal is to subsample the input image : the exact location of a feature is less important than its rough location relative to other features.
  - Reduce the computational load, memory usage, number of parameters (hence overfitting)



- Pooling neuron has *no weights*, it aggregates the inputs with an aggregation function (max, mean)

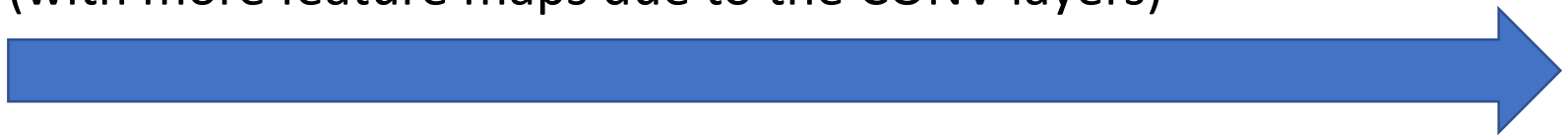
CNN Architecture : Summary

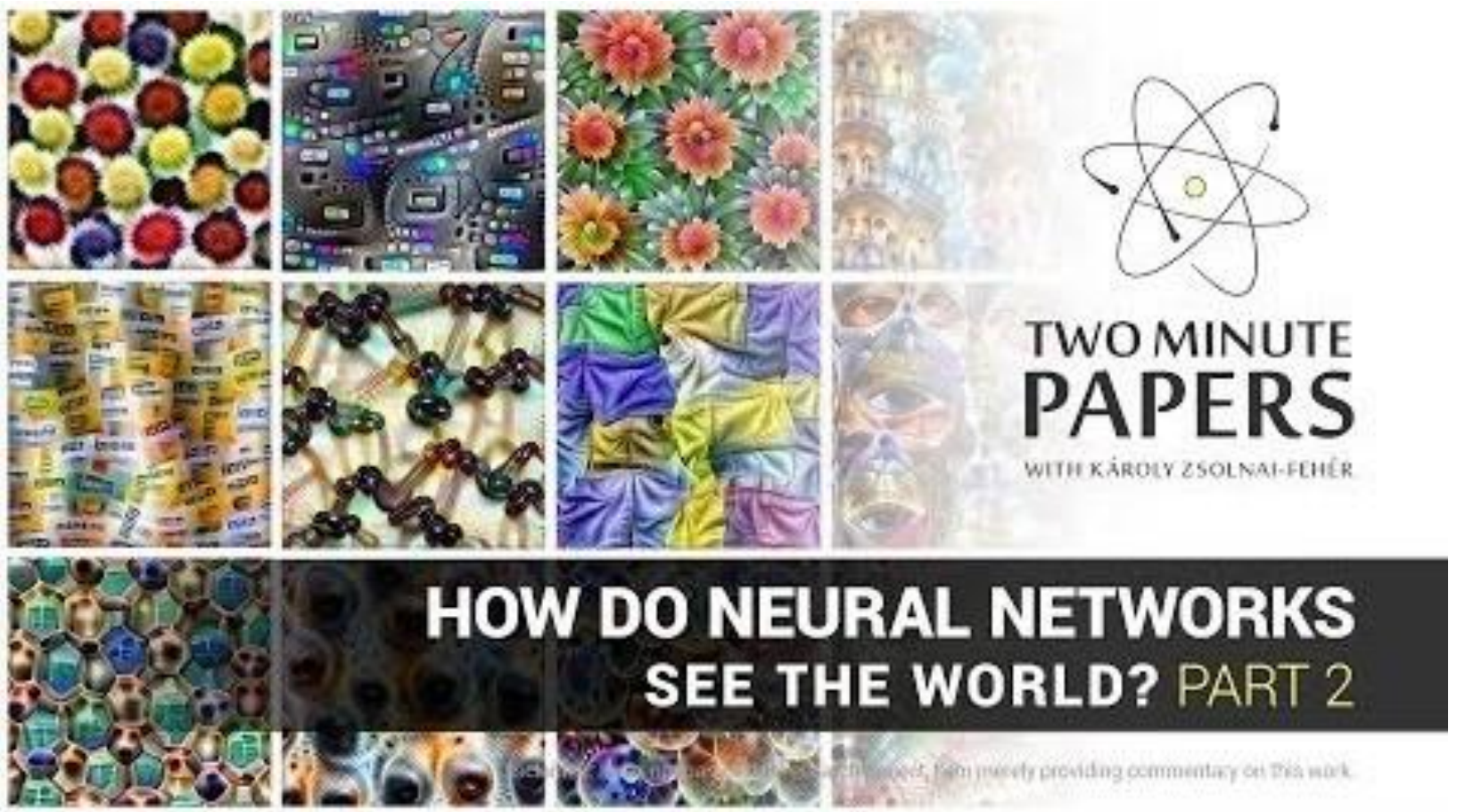


detects patterns in multiple sub-regions in the input field using receptive fields

progressively reduces the spatial size of the representation (fewer parameters, less overfitting)

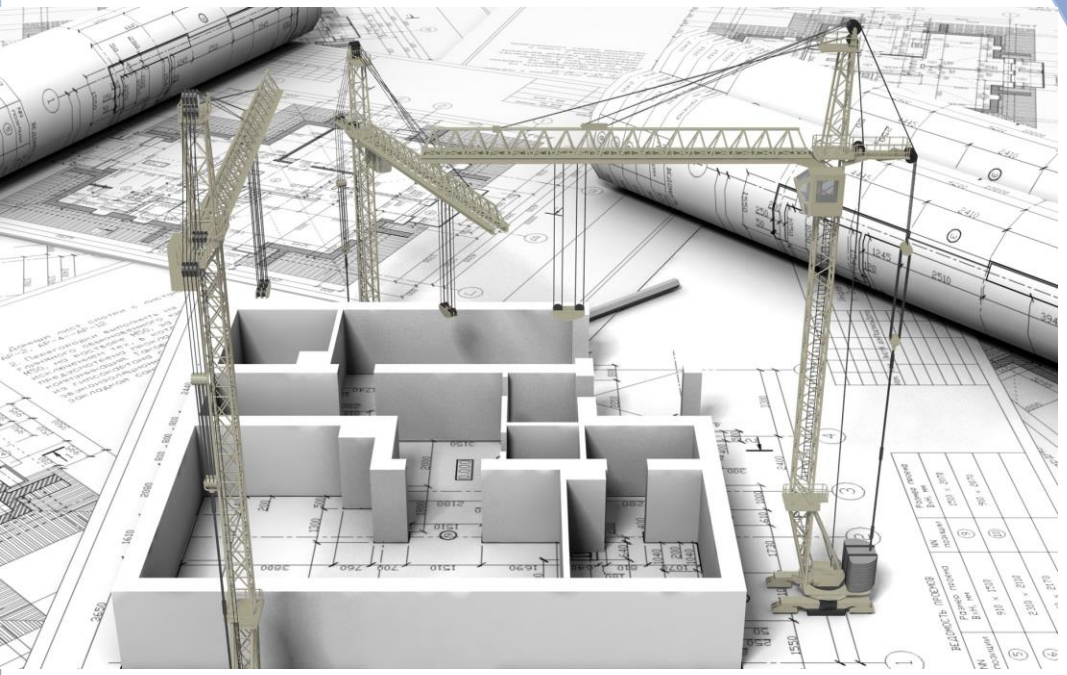
Image gets smaller and smaller, but also deeper and deeper (with more feature maps due to the CONV layers)





https://www.youtube.com/watch?v=1zvohULpe_0

Two-Minute Papers



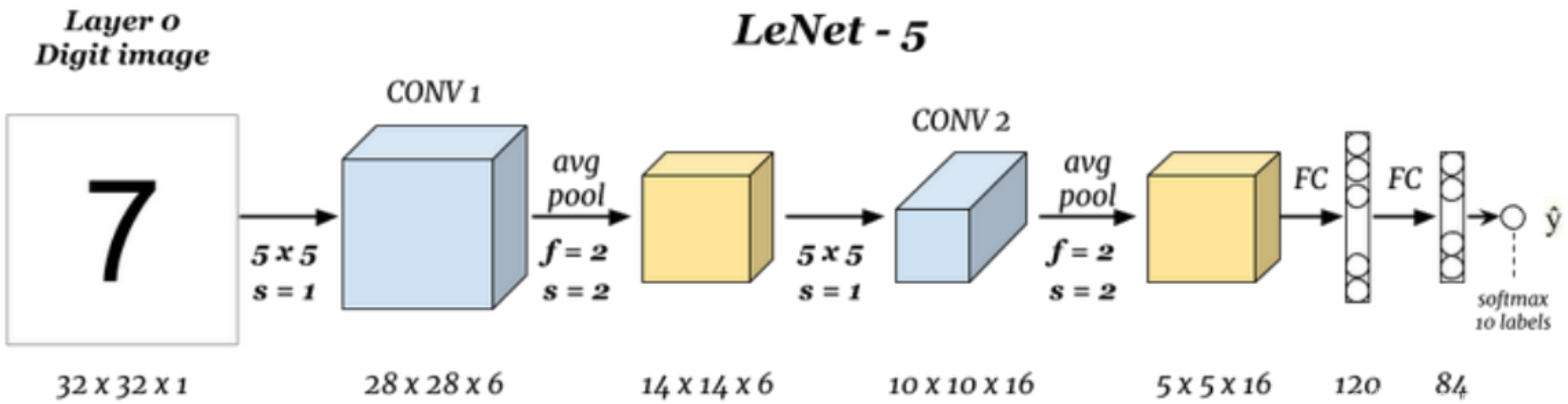
CNN Famous Architectures

1. LeNet-5
2. AlexNet
3. GoogLeNet
4. ResNet

MNIST Dataset

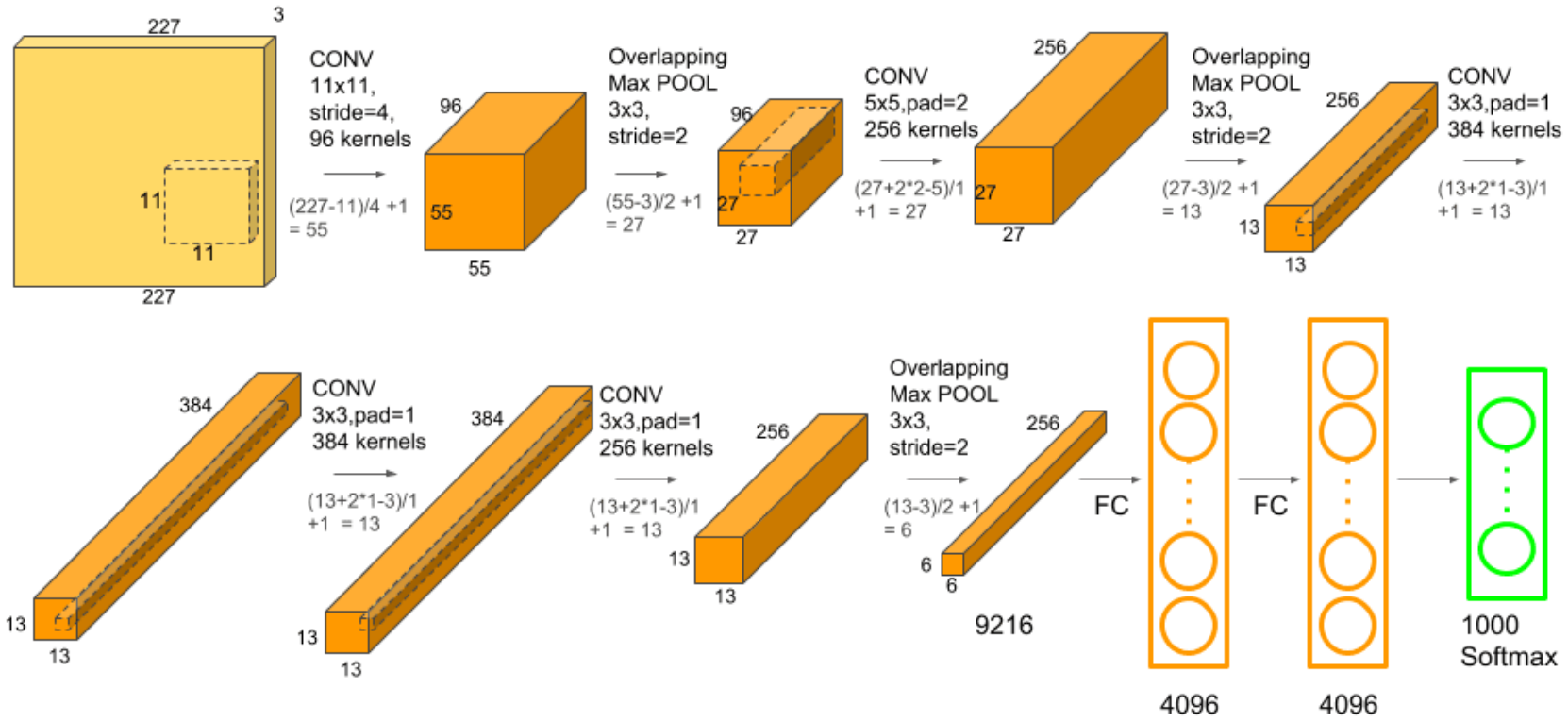
- hand-written digit recognition

1. LeNet-5 (1998)



- Size of image keeps shrinking, as padding not used within the network
- Activation functions : sigmoid/tanh
- 60K parameters

2. AlexNet (2012)

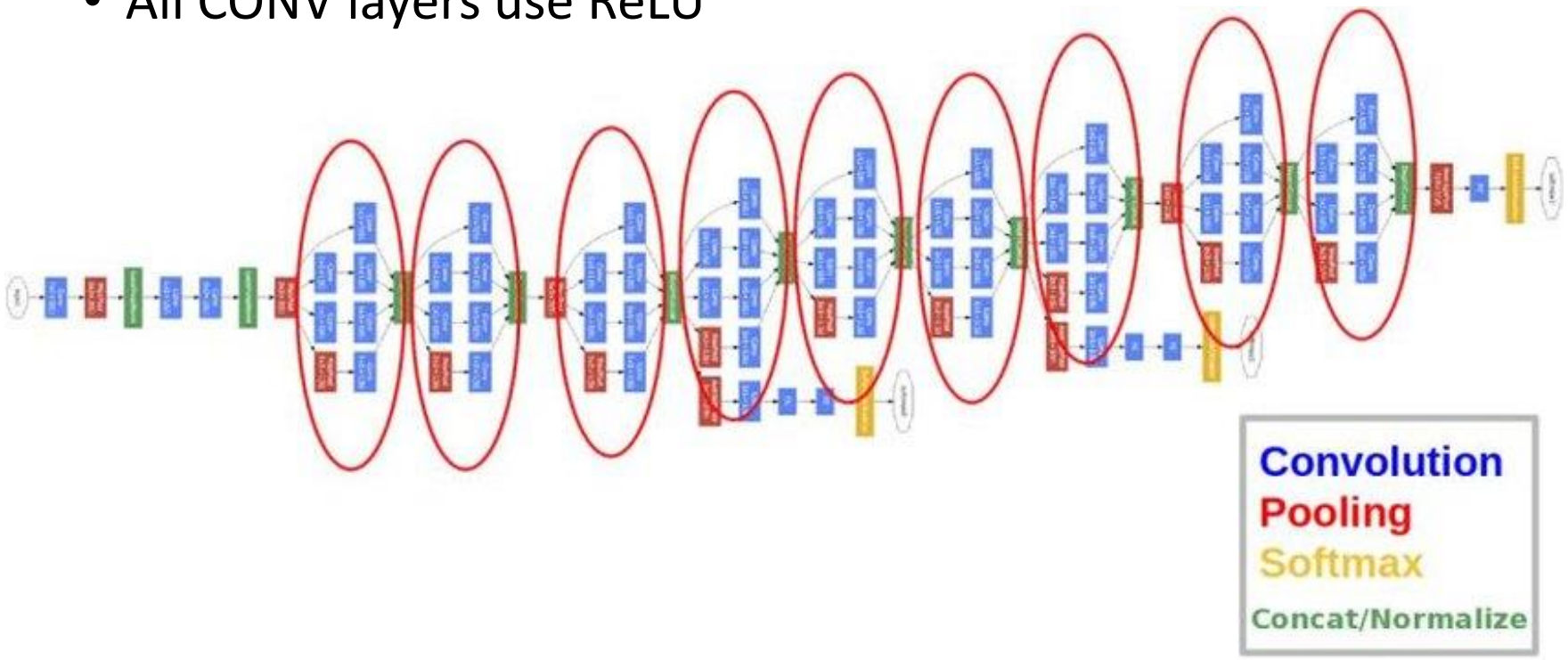


- First one to stack CONV layers directly on top of each other instead of stacking a POOL layer on top of each CONV layer
- Activation function : ReLU
- 60mio parameters (much larger and deeper than LeNet-5)

3. GoogLeNet (2014)

- Much deeper than previous CNNs thanks to sub-networks called inception modules
  - These use parameters much more efficiently : 6 mio of parameters instead of 60mio for AlexNet

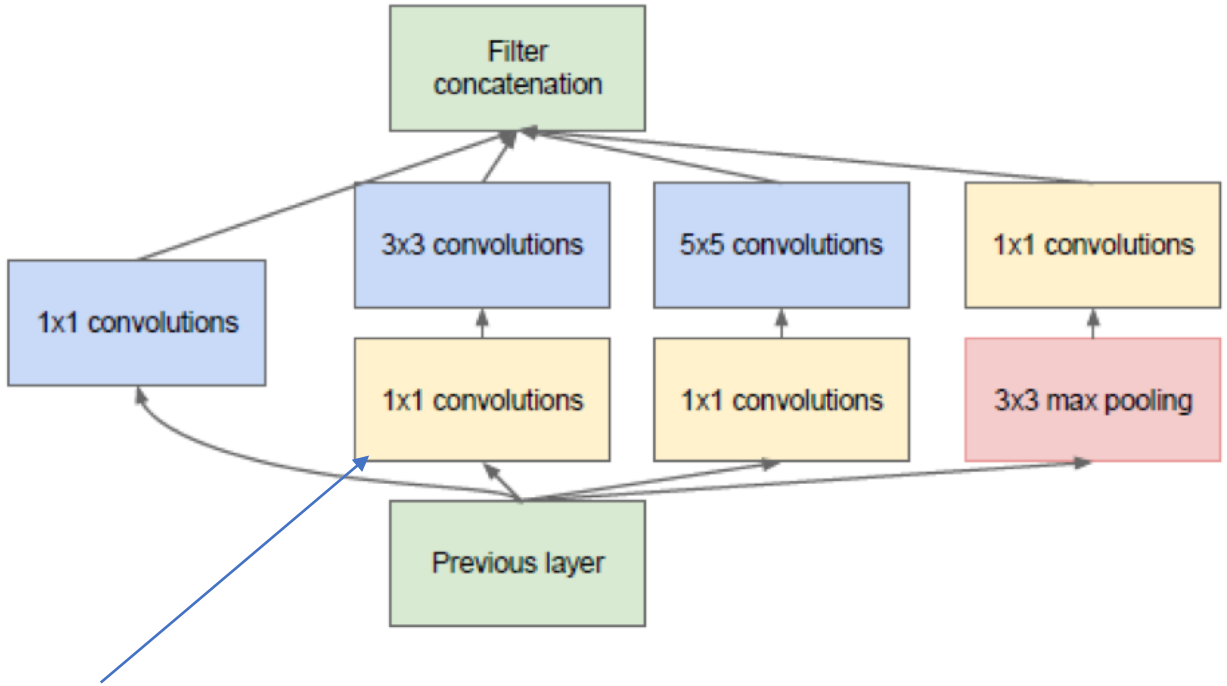
- Architecture :
  - 9 inception modules included
  - All CONV layers use ReLU



Inception Modules

- Allow to use multiple types of filter sizes instead of one

- Output : concatenate the outputs
  - The network gets progressively wider, not deeper



Reduces dimensionality across channels : number of input channels is limited

4. ResNet (2015)

- Extremely deep CNN composed of 152 layers

- Very deep NNs suffer from vanishing gradients : skip connections

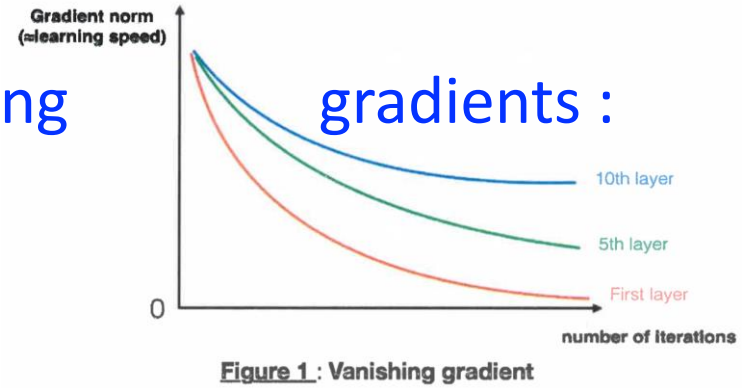
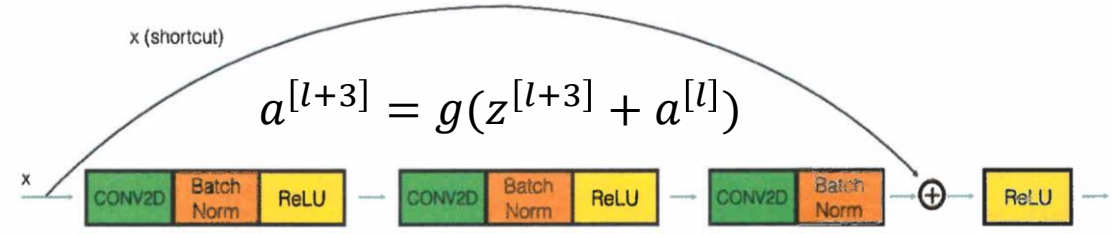


Figure 1 : Vanishing gradient

- Residual training : the signal feeding into a layer is also added to the output of a layer located a bit higher up the stack
  - Network forced to model : f(x) = h(x) – x rather than h(x)

- Architecture : stack of residual units, where each residual unit is a small NN with a skip connection



Some reading

10 Architectures

https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d#676b

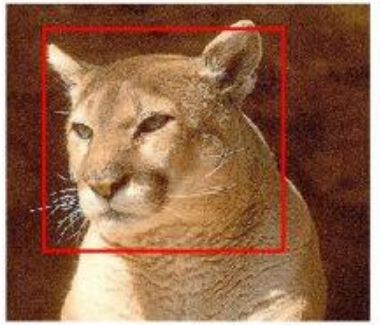
Object Detection

Introduction

Classification

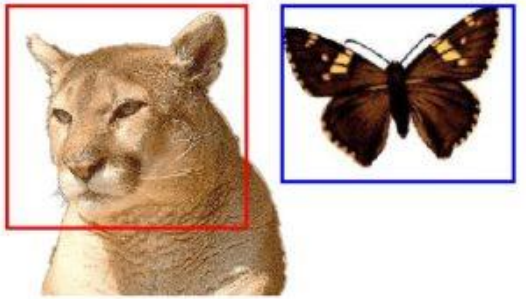
Cougar

Classification + Localization



Cougar

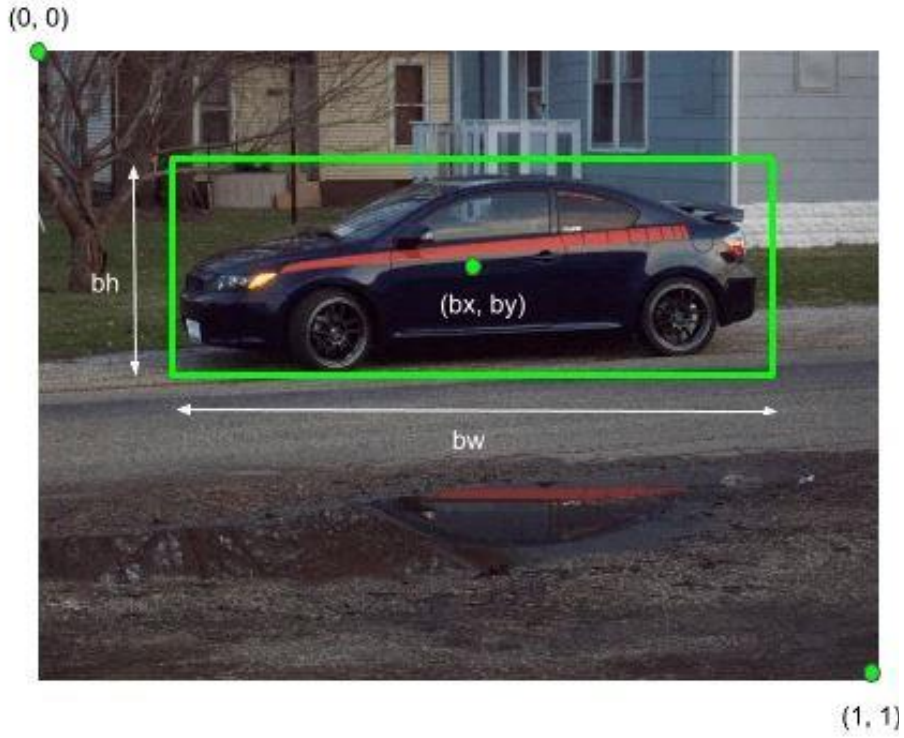
Object Detection



Cougar, Butterfly

Multiple Objects

Object Localization



Target variable :

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

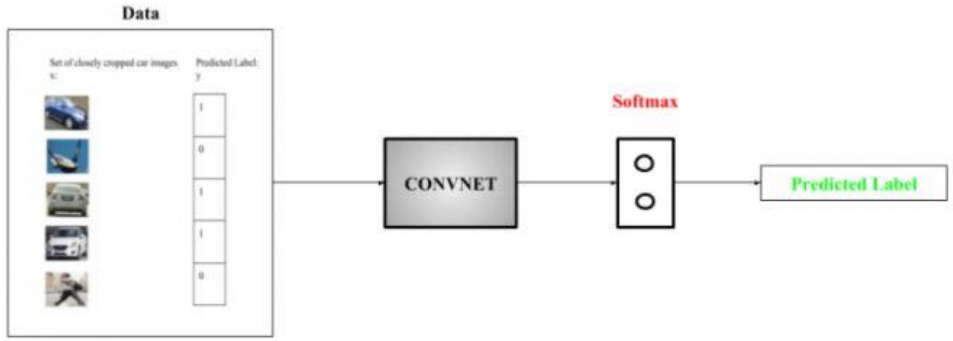
pc = Probability/confidence of an object (i.e the four classes) being present in the bounding box

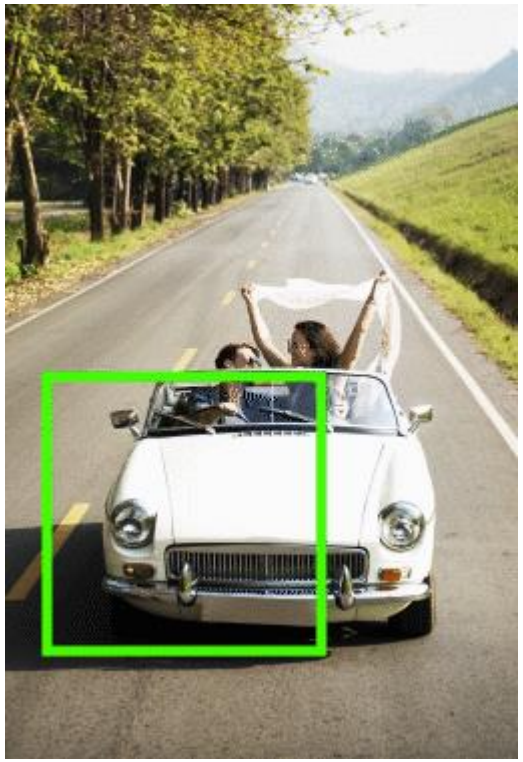
ci= Probability of the ith class the object belongs to

- Bounding box parameters :
  - bx, by : coordinates of the center of the bounding box
  - bw : width of the bounding box w.r.t the image width
  - bh : height of the bounding box w.r.t the image height

Object Detection : Principle

1) build a classifier that can classify closely cropped images of an object





2) Use a sliding window mechanism and crop a part of the image in each slide

3) Each cropped image is then passed to a ConvNet model, which in turn predicts the probability of the cropped image is a car

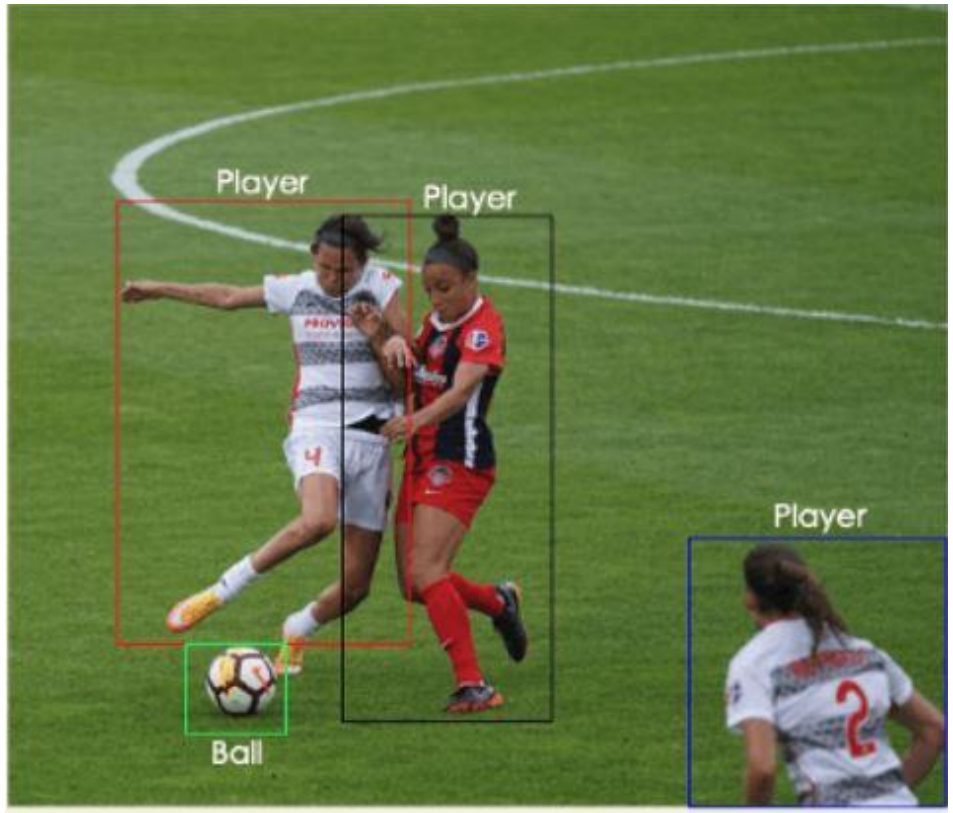
Self-driving cars

Object Detection Example

YOLO Algorithm

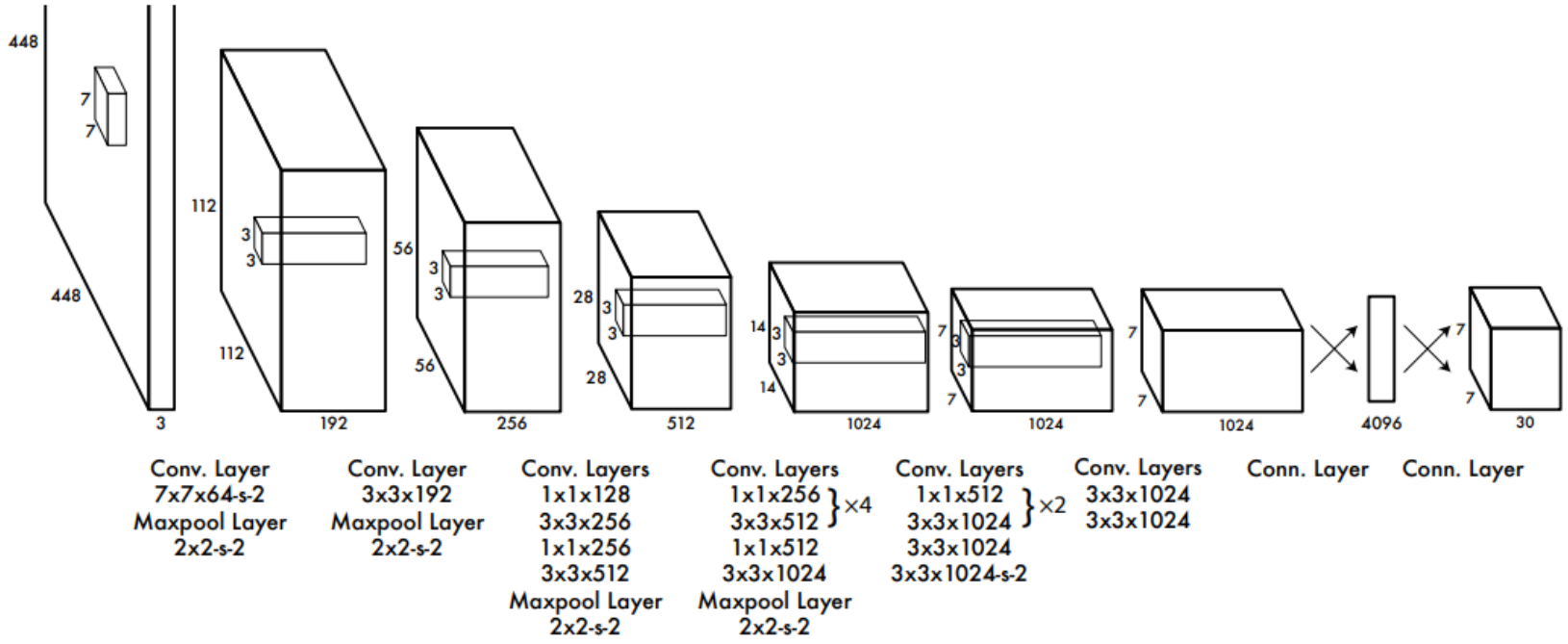
- You Only Look Once (YOLO) : real-time object detection algorithm

- object detection problem framed as a regression problem instead of a classification task
  - spatially separate bounding boxes and associate probabilities to each of the detected images using a CNN



YOLO Architecture

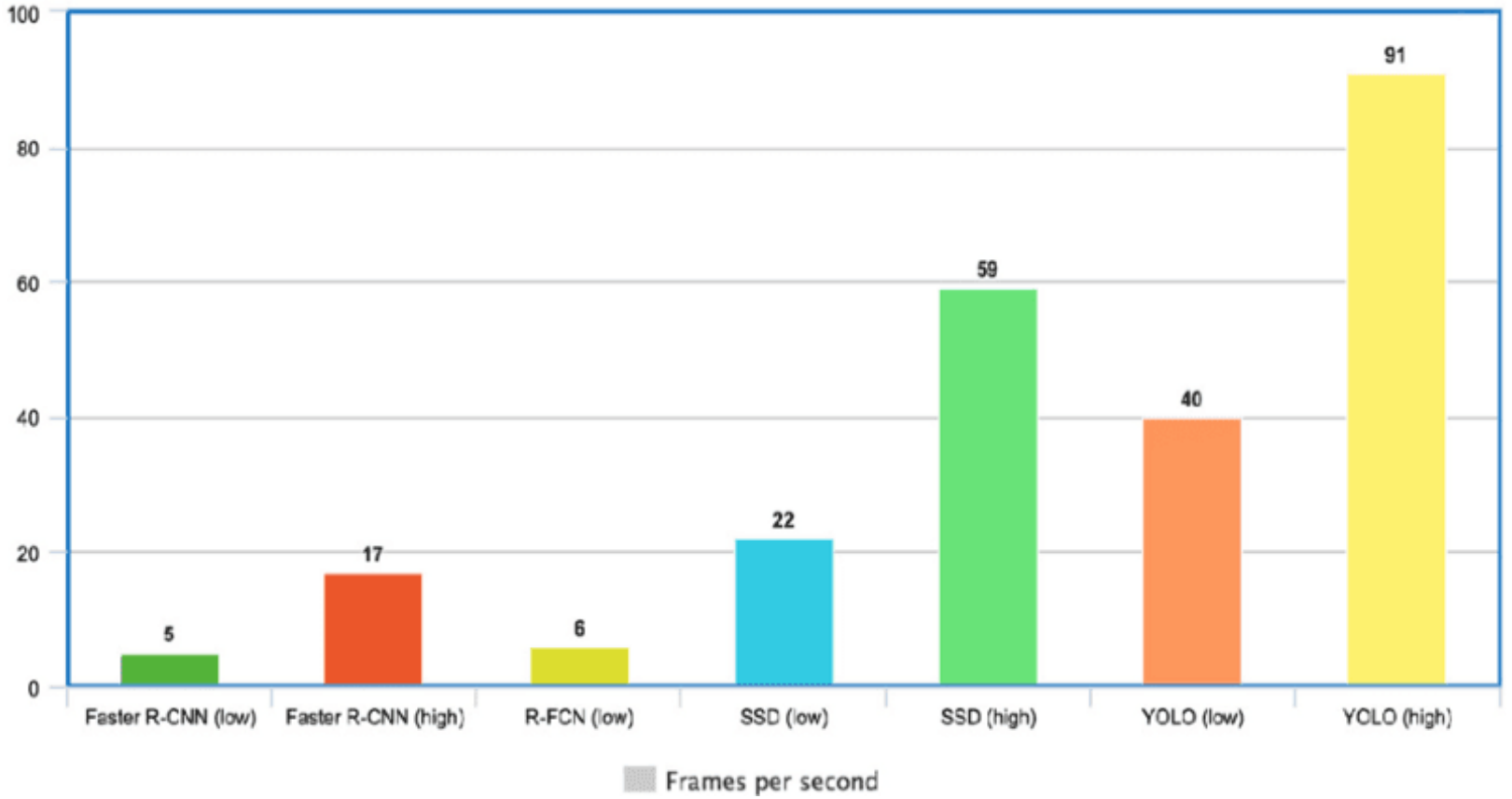
- YOLO architecture is similar to GoogleNet
  - 24 convolutional layers
  - 4 max-pooling layers
  - 2 fully connected layers



- Pretrained YOLOv2 model, trained on the COCO image dataset with classes similar to the Berkeley Driving Dataset
  - load pretrained weights
  - freeze all the weights except for the last layer during training

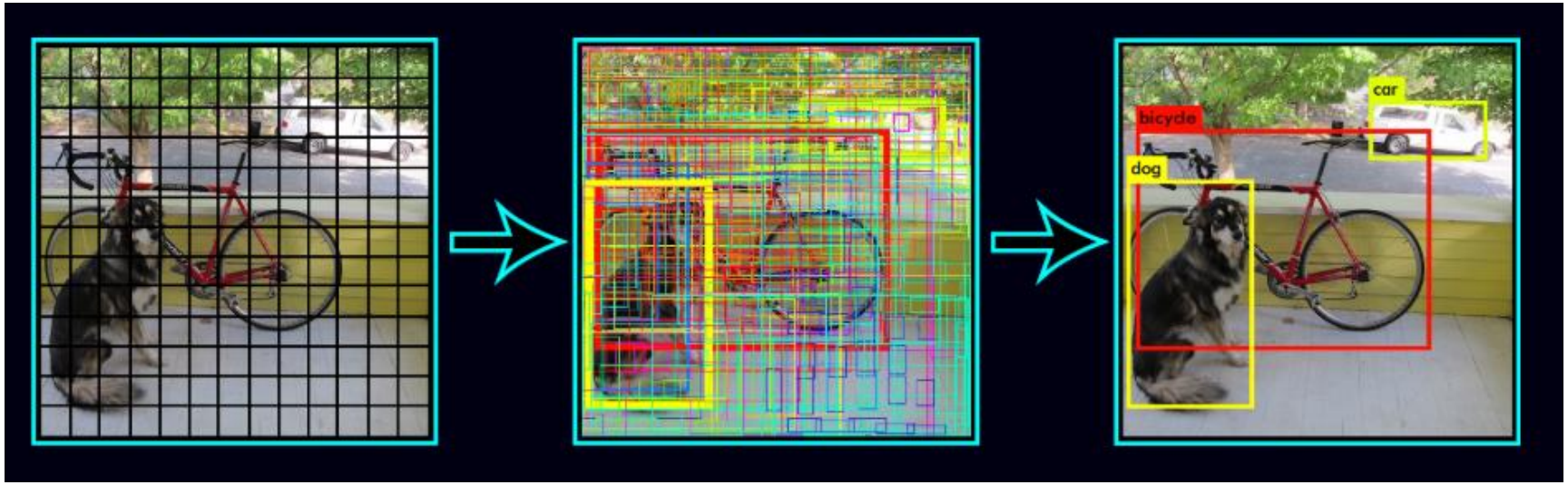
YOLO Popularity

- Speed
- Detection accuracy
- Good generalization
- Open-source



How does it work

- The algorithm works based on the four ingredients :
  - Residual blocks : divide the image into NxN grid cells of equal shape
  - Bounding box regression
  - Intersection Over Unions (IoU)
  - Non-Max Suppression : keep only the boxes with the highest probability score of detection.

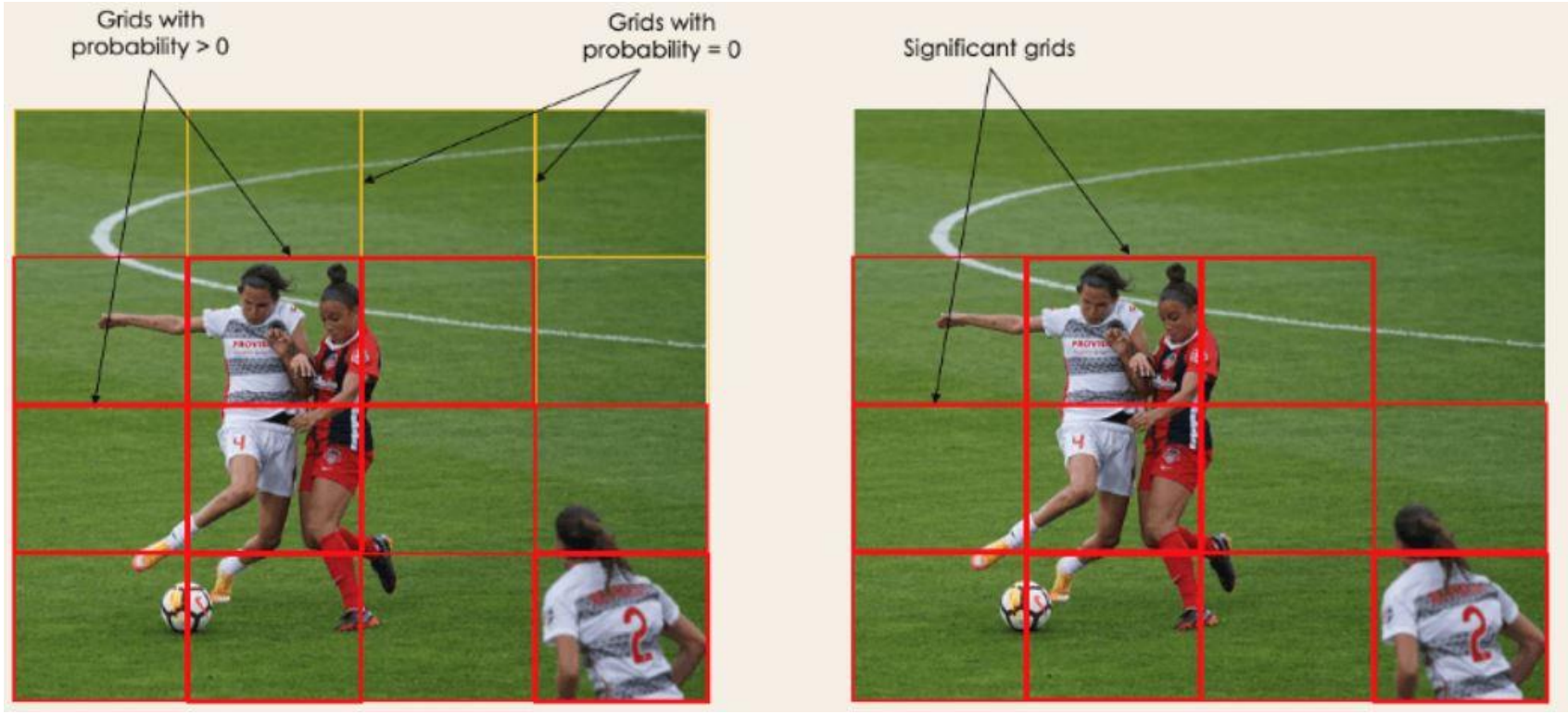


Bounding Box Regression

- YOLO determines the attributes of these bounding boxes using a single regression module
  - Y is the final vector representation for each bounding box :
  -

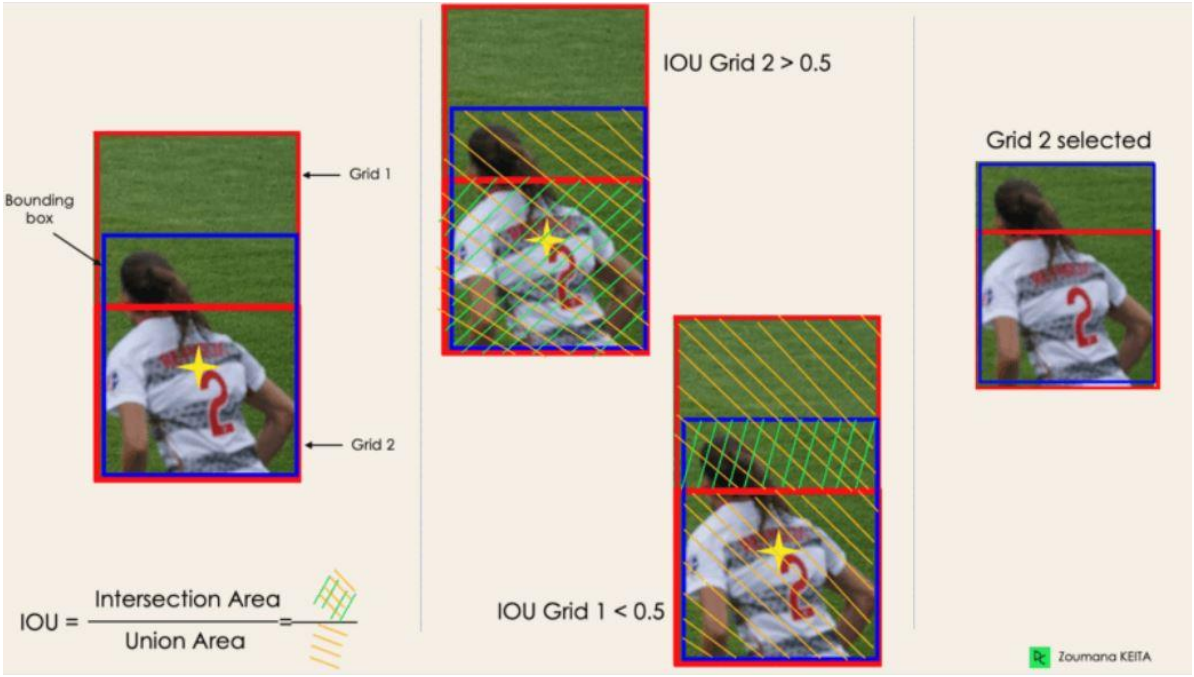
$$Y = [pc, bx, by, bh, bw, c1, c2]$$

- pc corresponds to the probability score of the grid containing an object



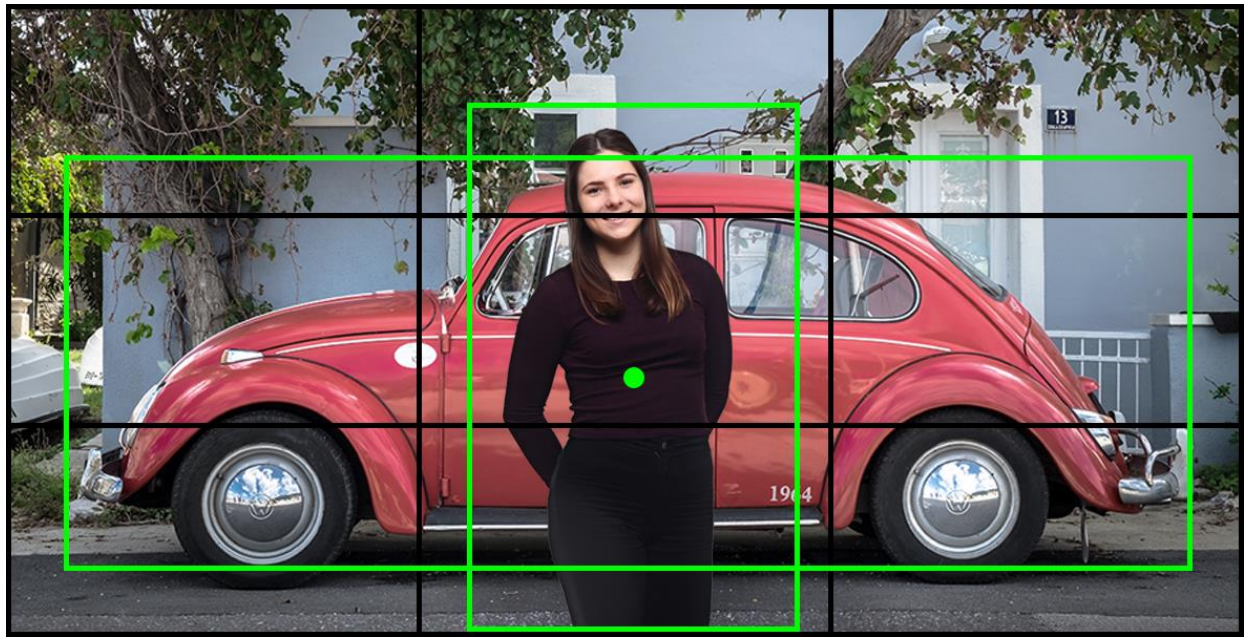
Intersection over Union (IoU)

- a single object can have multiple grid box candidates for prediction
- IoU (between 0 and 1) helps discarding grid boxes to only keep the relevant ones
  - IoU computed for each grid cell = Intersection area divided by the Union Area.
  - IoU selection threshold to defined (0.5)
  - ignore the prediction of the grid cells having an IOU ≤ threshold



Anchor Box

- Possible to detect two classes that are overlapping and share the same grid with anchor boxes



Target variable :

$$\hat{y} = [\underbrace{p_c \quad b_x \quad b_y \quad b_h \quad b_w \quad c_1 \quad c_2 \quad \dots \quad c_8}_{\text{Anchor Box 1}} \quad \underbrace{p_c \quad b_x \quad b_y \quad b_h \quad b_w \quad c_1 \quad c_2 \quad \dots \quad c_8}_{\text{Anchor Box 2}}]^T$$

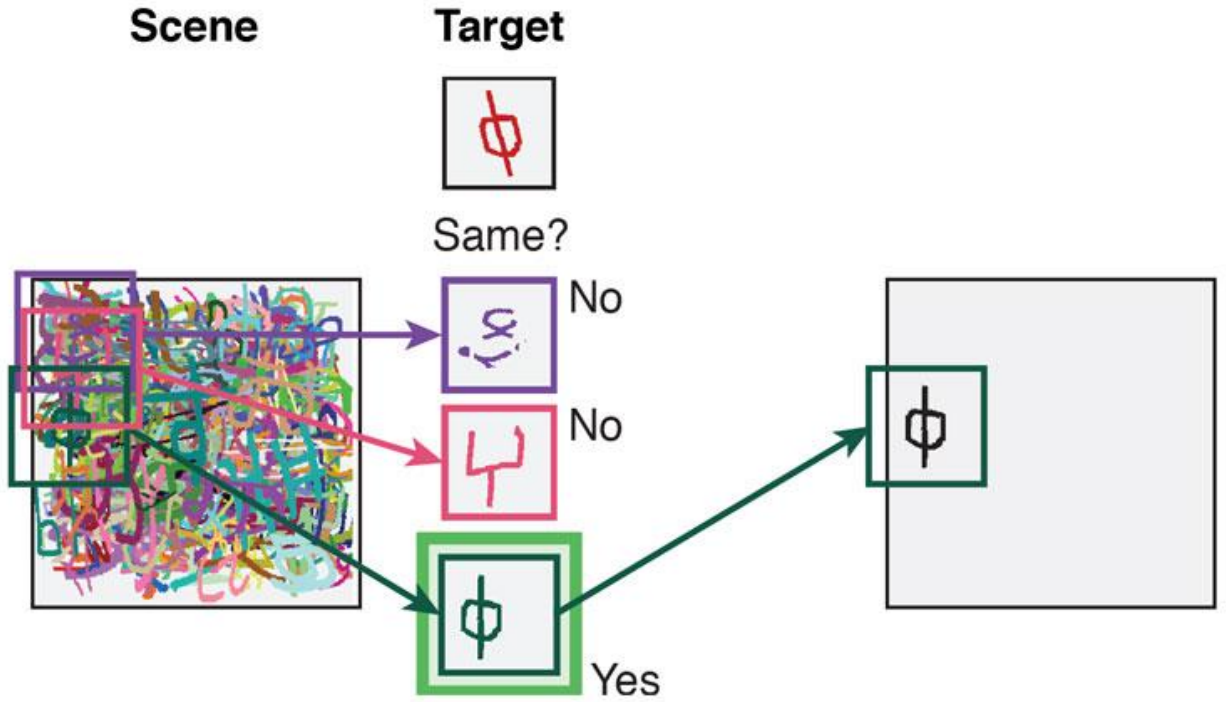
Result

https://www.youtube.com/watch?v=1_SiUOYUoOI

Face Detection

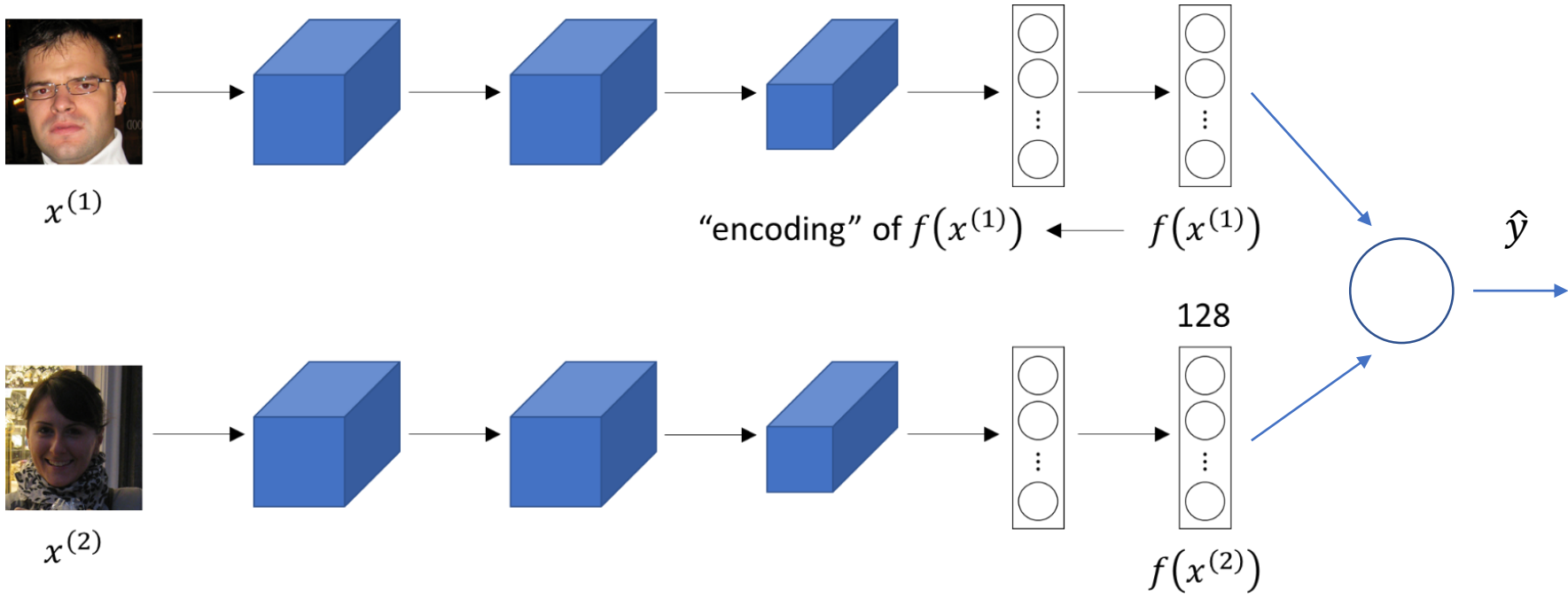
One-shot Learning

- Learn from a couple of examples to recognize the person again (very small training set) with different facial expressions, lighting conditions, accessories,…
- Learn a similarity function d (degree of difference between images)



Siamese Network

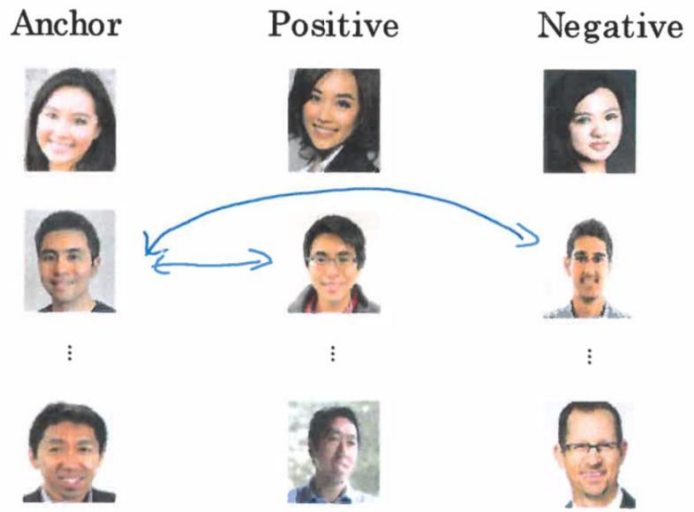
- Network used to learn the similarity function d



$$d\left(x^{(1)}, x^{(2)}\right) = \left\| f\left(x^{(1)}\right) - f\left(x^{(2)}\right) \right\|^2$$

Triplet Loss

- Loss function where a baseline input (anchor) is compared to a positive input and a negative input :



- Often used for learning similarity
- Loss function is described using a Euclidean distance function

$$\mathcal{L}(A, P, N) = \max\left(\|\mathrm{f}(A) - \mathrm{f}(P)\|^2 - \|\mathrm{f}(A) - \mathrm{f}(N)\|^2 + \alpha, 0\right)$$

- Cost function :

$$\mathcal{J} = \sum_{i=1}^{M} \mathcal{L}\left(A^{(i)}, P^{(i)}, N^{(i)}\right)$$

Security

Watch out with your algorithms !

https://www.youtube.com/watch?v=SA4YEAWVpbk

Two-Minute Papers

Quiz

https://b.socrative.com/login/student/

Room : CONTI6128