How Object Detection has been transformed by Deep Learning

Paul Blondel, PhD

July 8th, 2021

Object Detection?

- GOAL: **detect objects** in images or video frames
- Sometimes also **detect several** object classes **simultaneously**

Object Detection?

- GOAL: **detect objects** in images or video frames
- Sometimes also **detect several** object classes **simultaneously**

Object Detection:

A solution to treat the **ever growing** amount of images and video frames:

- Embedded cameras
- Security cameras
- Mobile phones
- ...

Object detection is **not a so easy task**:

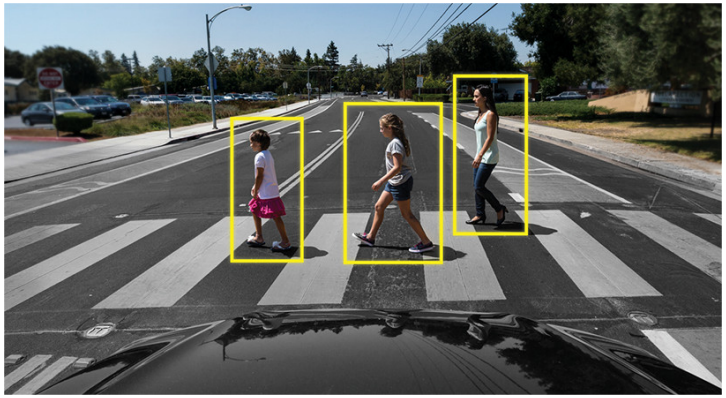
- objects can have **multiple scales**
- objects can be **partially hidden**
- object classes can **look similar**
  - ex: lions and cats
- in a same object class we can have **different textures, colors, etc**
  - ex: human people wearing **different clothes**
- objects can have **multiple orientations and postures**
- Etc.

On top of that, object detection can have other constraints:

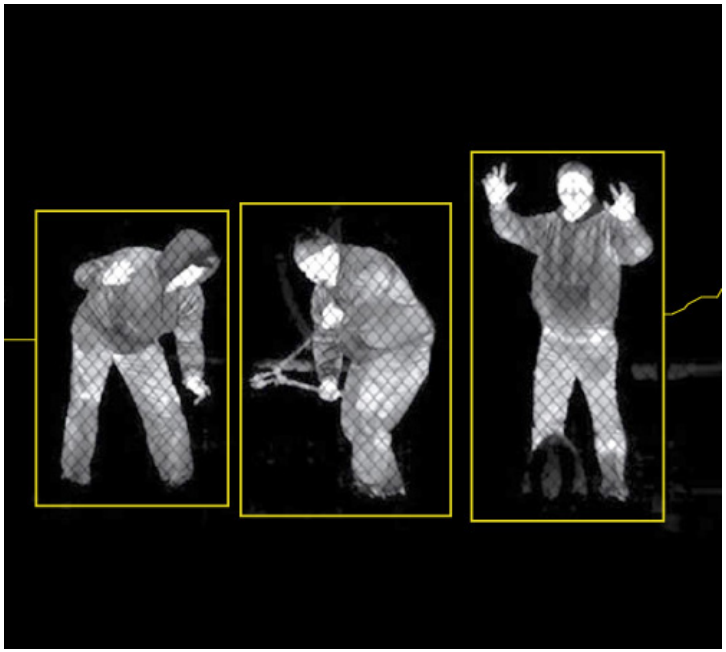
- working in **real-time**
- working in **embedded systems**
  - ex: cars, UAVs, etc.
- working whatever the **weather conditions**
- working at **night**
- Etc.

Examples of application:

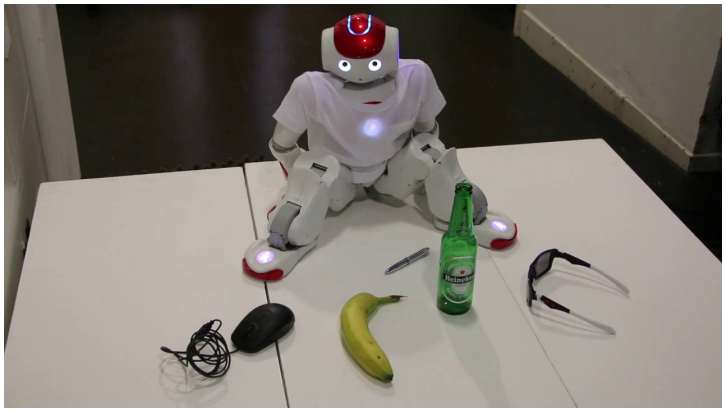
- Advanced Driver Assistance System (ADAS)



- Video surveillance:



- Robots

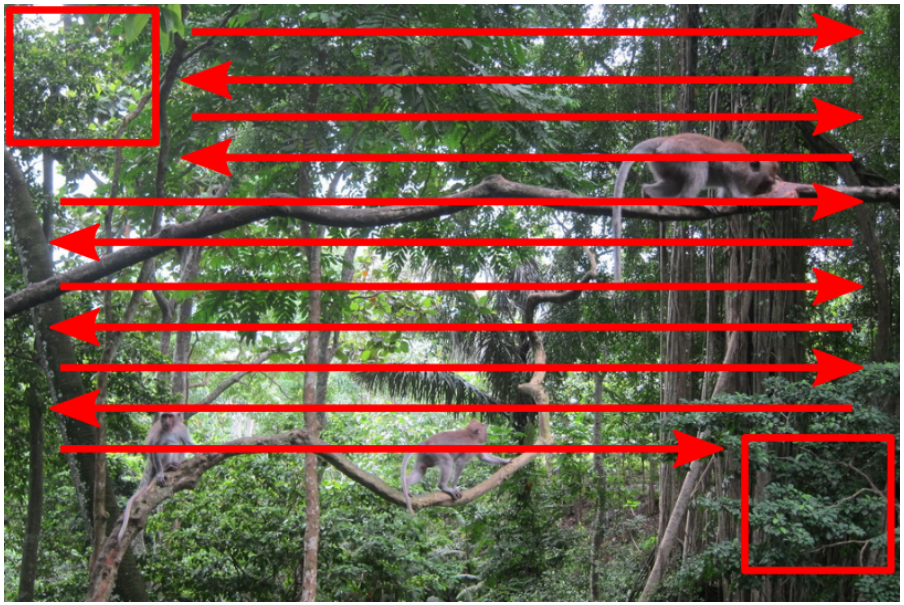


- Face detection

- Etc.

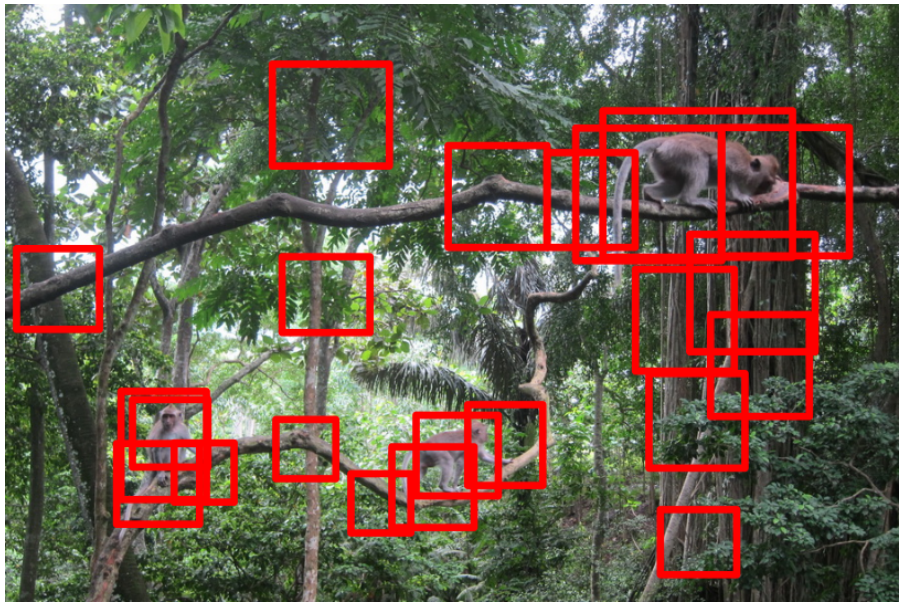
Let's see how to find these monkeys.

Searching at multiple locations



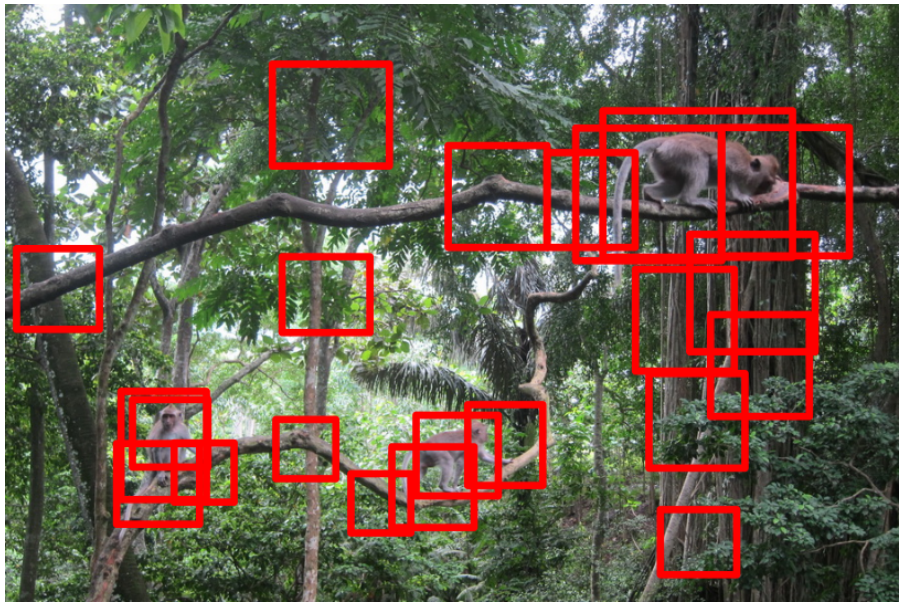
- **Sliding Window (bruteforce): Exhaustive scan of the image**

Searching at multiple locations



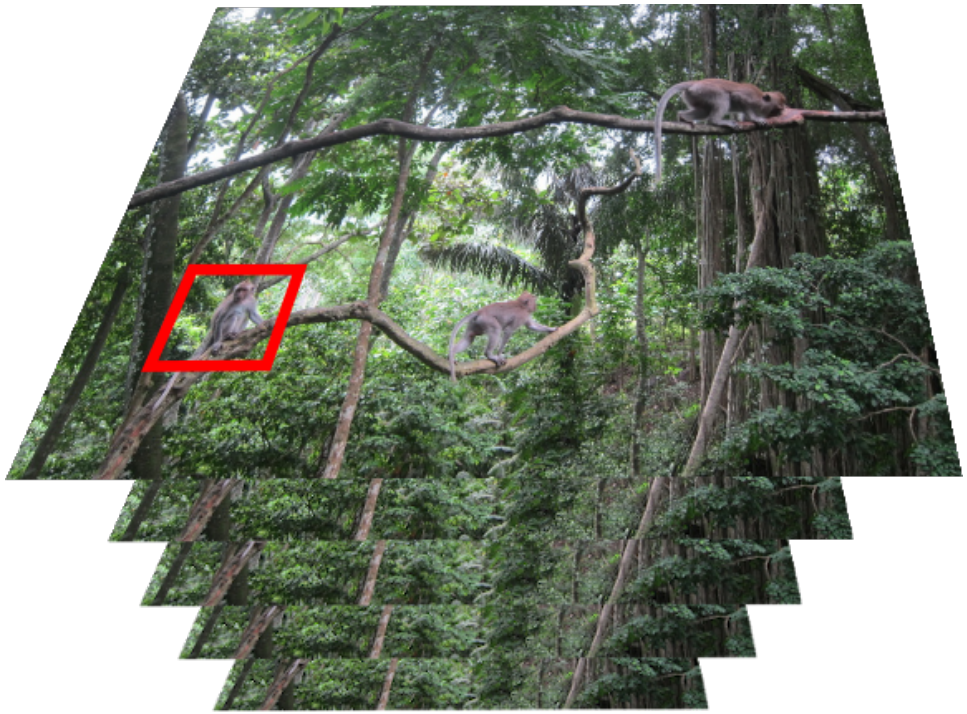
- Sliding Window (bruteforce): Exhaustive scan of the image
- **Generate region proposals: Info-rich regions are proposed**

Searching at multiple locations



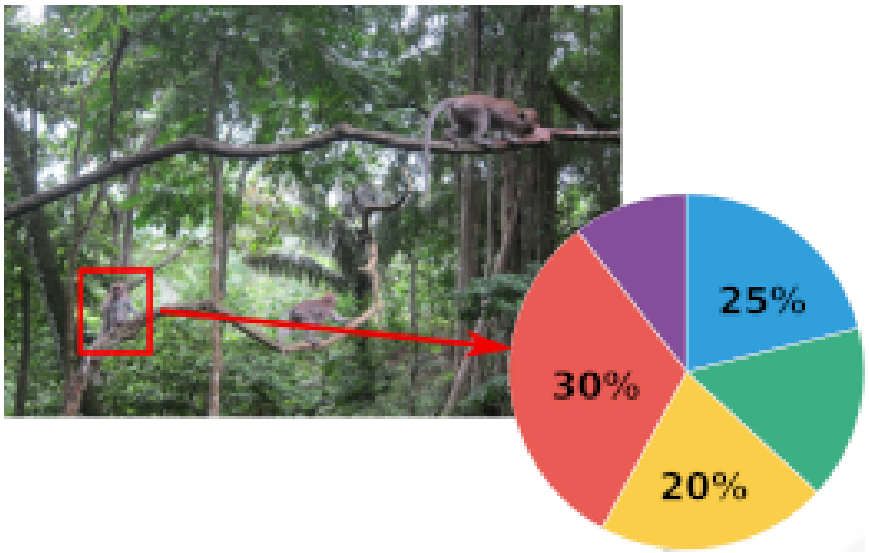
- Sliding Window (bruteforce): Exhaustive scan of the image
- **Generate region proposals: Info-rich regions are proposed**
  - Selective Search algorithm

Searching at multiple scales
(usually paired with a sliding window approach)



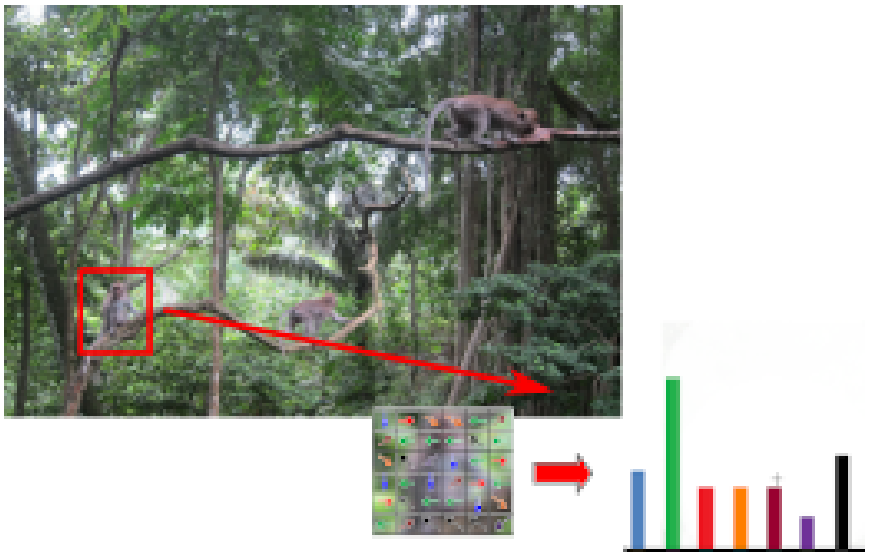
- Analysis window
  - Fixed size
- Image pyramid
  - Down-scaled levels for big objects
  - Up-scaled levels for small objects

Finding clues of the presence of an object in the window (visual features)



- Visual features
  - **Colors**

Finding clues of the presence of an object in the window (visual features)

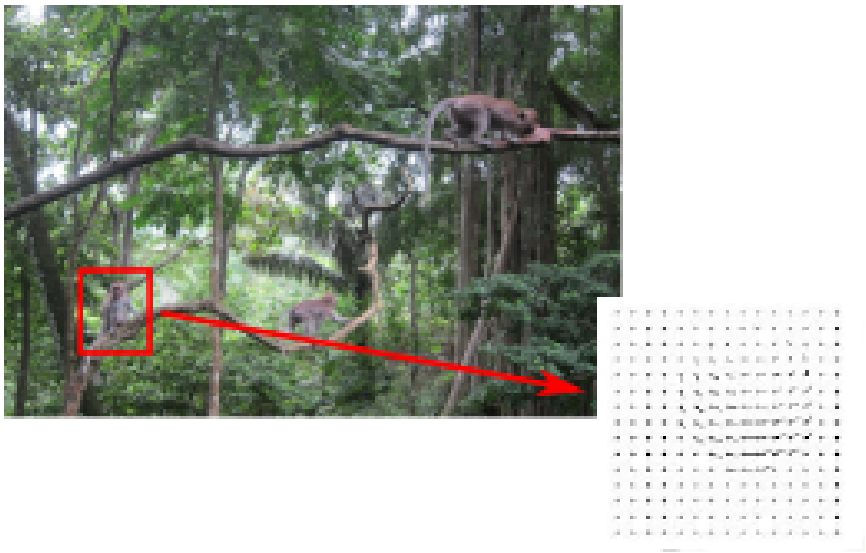


- Visual features
  - Colors
  - **Shapes**

Finding clues of the presence of an object in the window (visual features)

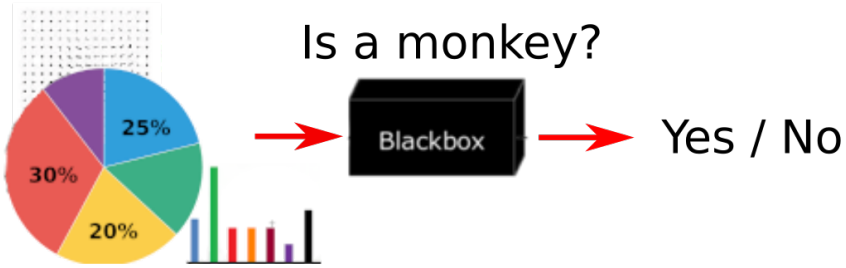
- Visual features
  - Colors
  - Shapes
  - **Depth**

Finding clues of the presence of an object in the window (visual features)



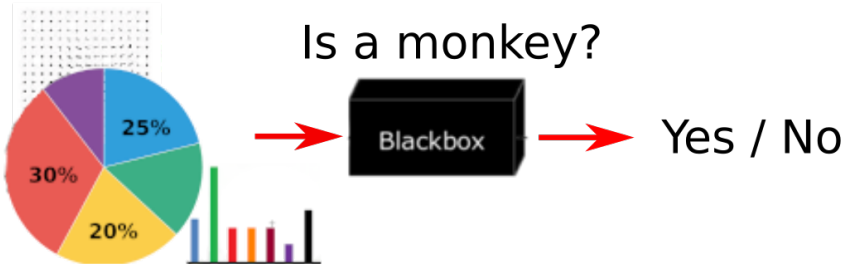
- Visual features
  - Colors
  - Shapes
  - Depth
  - **Movements**
  - **Etc.**

Analyze the collected visual features and ... decide!



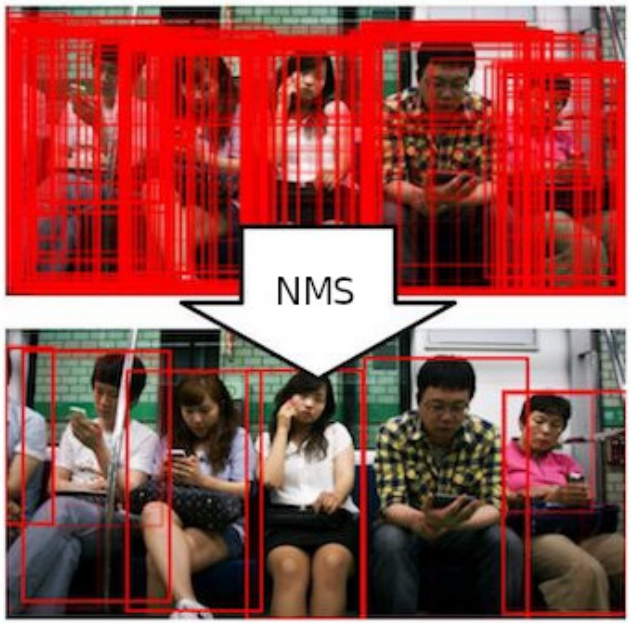
- Classify
  - Monkey's visual features: Yes
  - or not: No

Analyze the collected visual features and ... decide!



- Classify
  - Monkey's visual features: Yes
  - or not: No
- With Deep learning (part 4)
  - All these steps may be combined together

Because detection windows are analyzed at nearby locations the detector may trigger several detections nearby object instances:



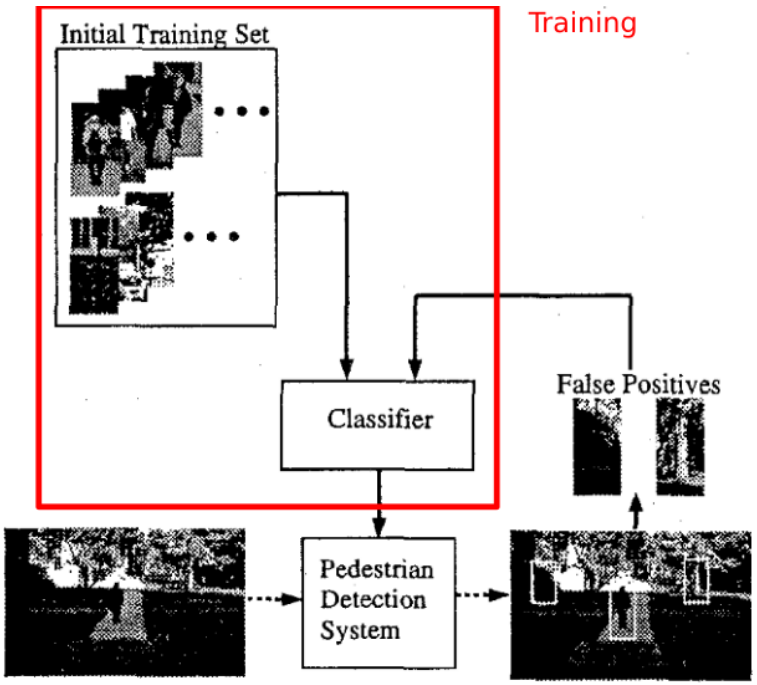
Algorithm 1 Non-Max Suppression

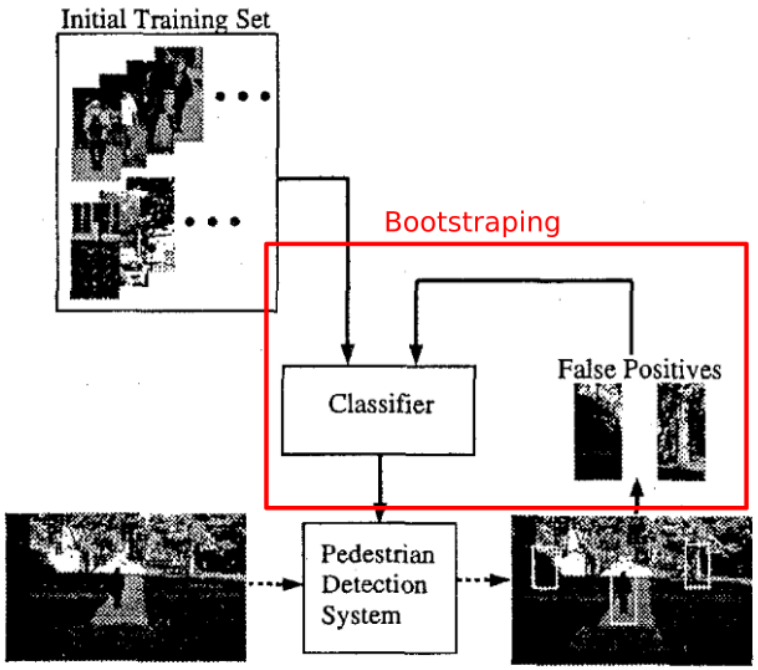
procedure NMS(B,c)
  $B_{nms} \leftarrow \emptyset$ Initialize empty set
  for $b_i \in B$ **do** => Iterate over all the boxes
    $discard \leftarrow$ False Take boolean variable and set it as false. This variable indicates whether b(i) should be kept or discarded
    for $b_j \in B$ **do** Start another loop to compare with b(i)
      if same$(b_i, b_j) > \boldsymbol{\lambda_{nms}}$ **then** If both boxes having same IOU
        if score$(c, b_j) >$ score(c, b_i) **then**
          $discard \leftarrow$ True Compare the scores. If score of b(i) is less than that of b(j), b(i) should be discarded, so set the flag to True.
    if not $discard$ **then** Once b(i) is compared with all other boxes and still the discarded flag is False, then b(i) should be considered. So add it to the final list.
      $B_{nms} \leftarrow B_{nms} \cup b_i$
  return B_{nms} Do the same procedure for remaining boxes and return the final list

One way to only keep the best detections (having the highest scores) is to use: Non-Maximum Suppression(NMS).

The blackbox (classifier) can be trained with Machine Learning

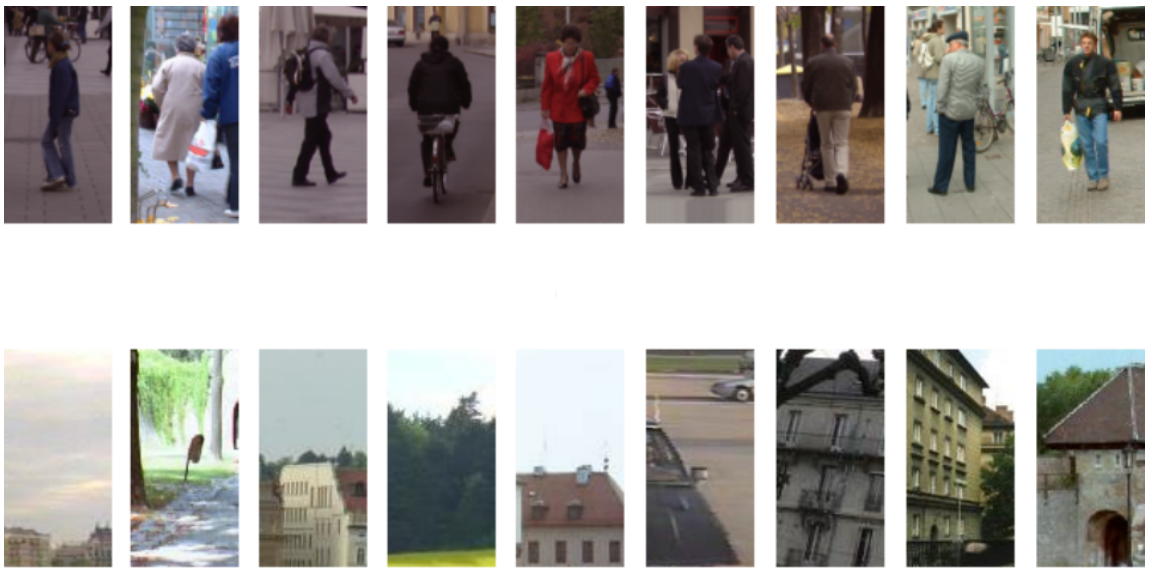
- Papageorgiou et al: First attempt to train a classifier with SVM





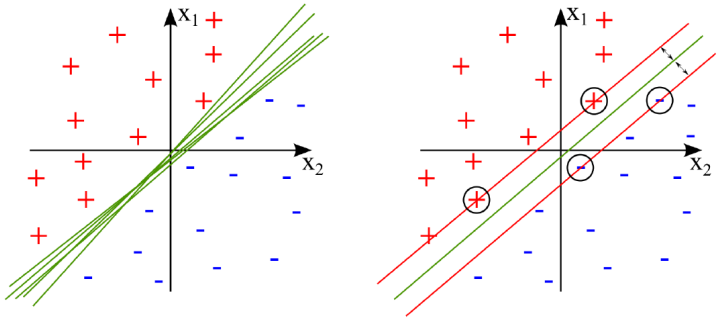
In order to **train the classifier** we **need a lot of examples**:

- images of **object** (ex: images of people)
- images of **random background**



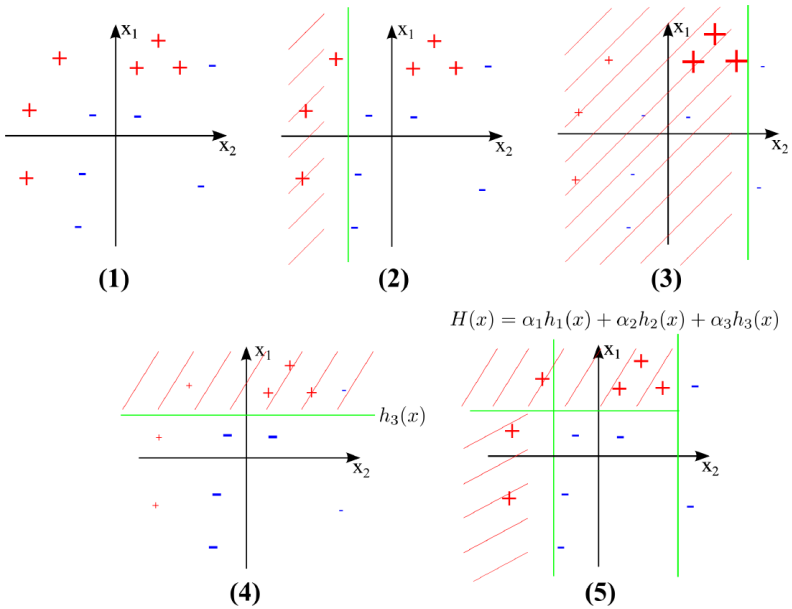
Then, we can **train the classifier** with a Machine Learning algorithm:

Support Vector Machine (SVM)



- **Goal**: Maximize margin of class separating hyperplane
- **How**: Lagrangian dual problem optimization
- +: positive class (people)
- -: negative class (background)

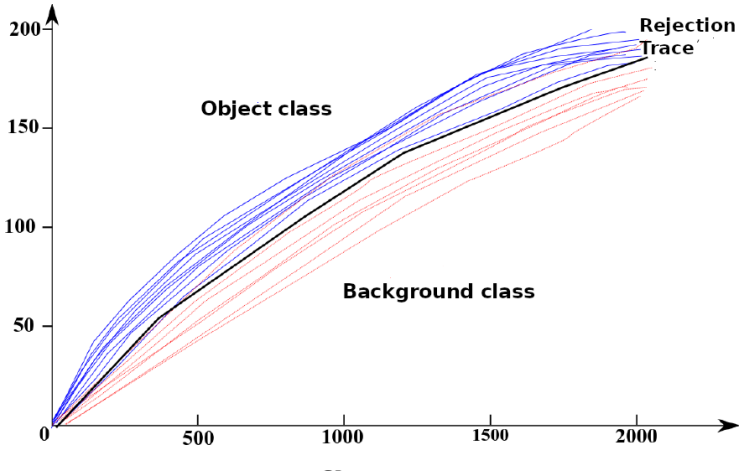
Boosting (AdaBoost)

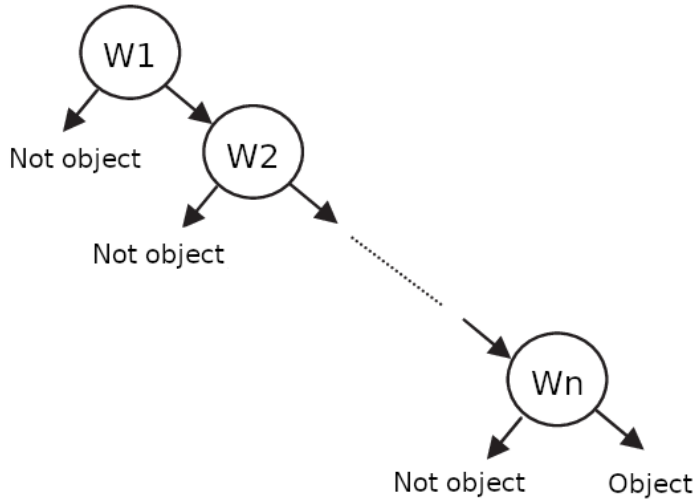


- **Goal**: Train discriminative classifier (H) made of weak classifiers (h_i)
- **How**: Train successive weak classifiers

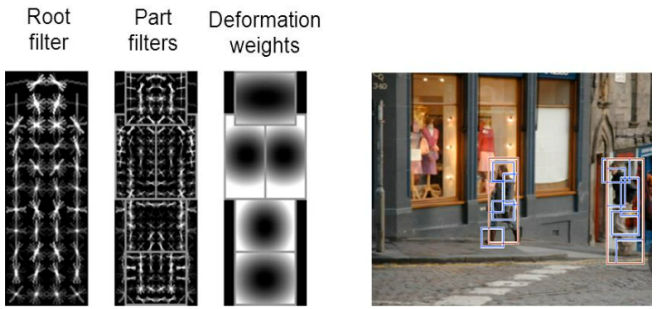
Improvements of Machine Learning-based Object Detection:

- Faster classifier inference (Soft-Cascade Boosting):





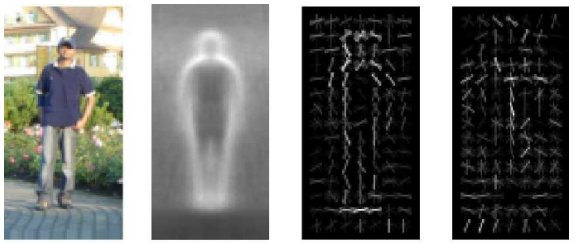
- Part-based detector for articulated objects (DPM/Latent-SVM)



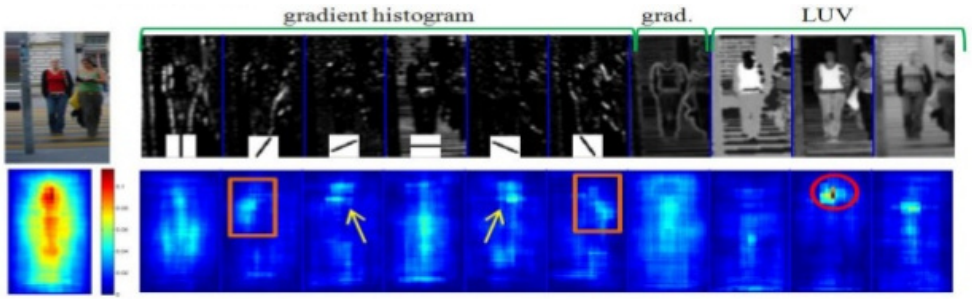
  - Classifier form:
    $f(x) = max_z(w.H(x,z))$
    - ★ w: support vectors
    - ★ z: latent variables
  - Training, initiate w and iterate:
    - ★ Fix w and find z (part positions)
    - ★ Fix z and solve w (classic SVM)

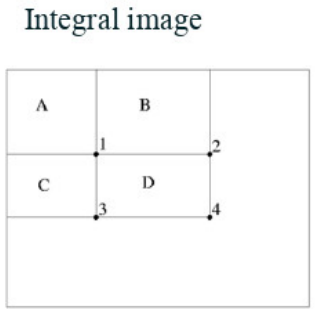
Finer and finer visual features extraction:

- Histogram of Oriented Gradients (HOG)



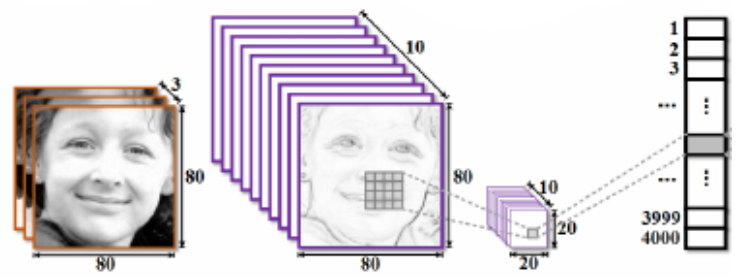
- Integral Channel Features (ICF)





Sum of all pixels in
D = 1+4-(2+3)
= A+(A+B+C+D)-(A+C+A+B)
= D

- Aggregated Channel Features (ACF)



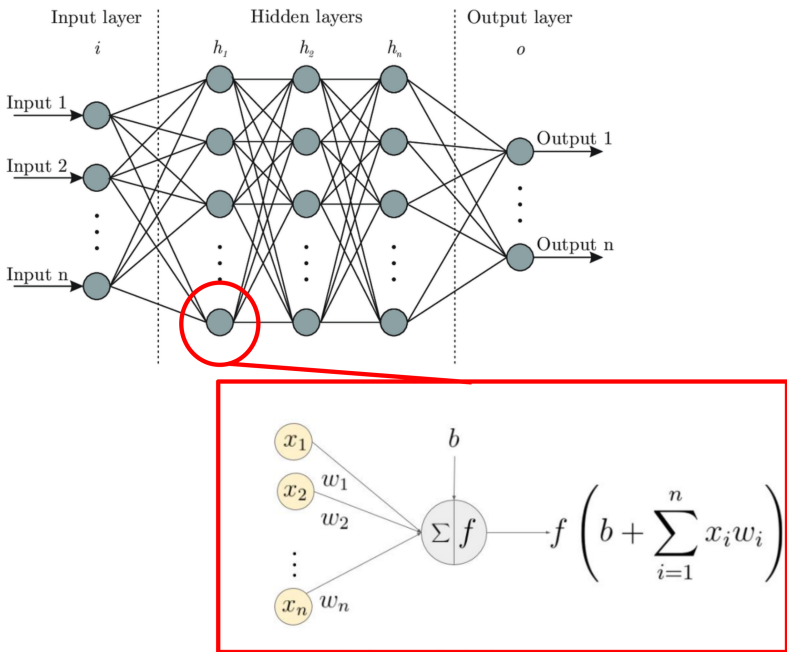
Machine Learning for Object Detection

No big improvements until 2014, until the democratization of Deep Learning for Object Detection.

Deep Learning permitted the **come back of Artificial Neural Networks**:

- Artificial Neural Networks (ANN) **exist for a very long time** (50's)
- But for a long time, performances obtained with ANNs = not satisfactory
- Deep Learning, means **learning a network with more than 4/5 hidden layers**

Artifical Neural Networks recalls:



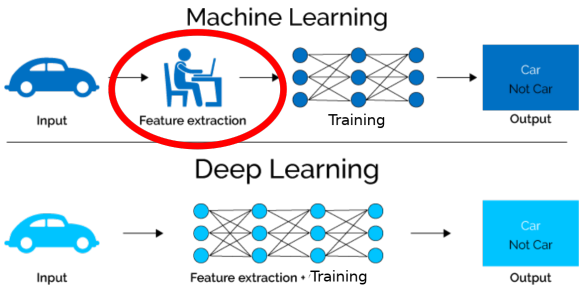
- All the weights w_i between all the nodes constitutes the model
- Nodes output values with respect to an activation function f and weights w_i
- trained by back propagation (gradient of loss)
- Goal: optimize a loss function on weights w_i

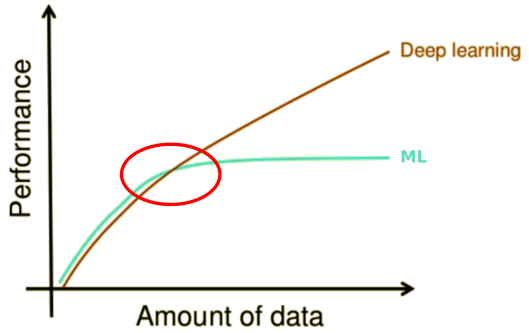
The reemergence of ANNs with Deep learning is due to several factors:

- **Fixing vanishing gradient** (ReLU, Normalized Init, ResNet, BatchNorm...)
- **Fixing exploding gradient** (Gradient norm, Normalized Init and clipping...))
- **More data** available everywhere
  - increasing amount of unstructured data (videos, images, Etc.)
  - collaborative labeling approaches (Amazon Mech. Turk, CVAT, Etc.)
- **Increasing processing power** (NVIDIA TITAN V GPU, Etc.)
- **New "layers"** (RNN, LSTM, Dropout, Attention, Transformer...)
- **Others...** (Deep Reinforcement Learning,...)

The case of Object Detection

Amongst the different types of ANNs: Convolutional Neural Networks (CNNs) are particularly suitable in Computer Vision, and thus for Object Detection.





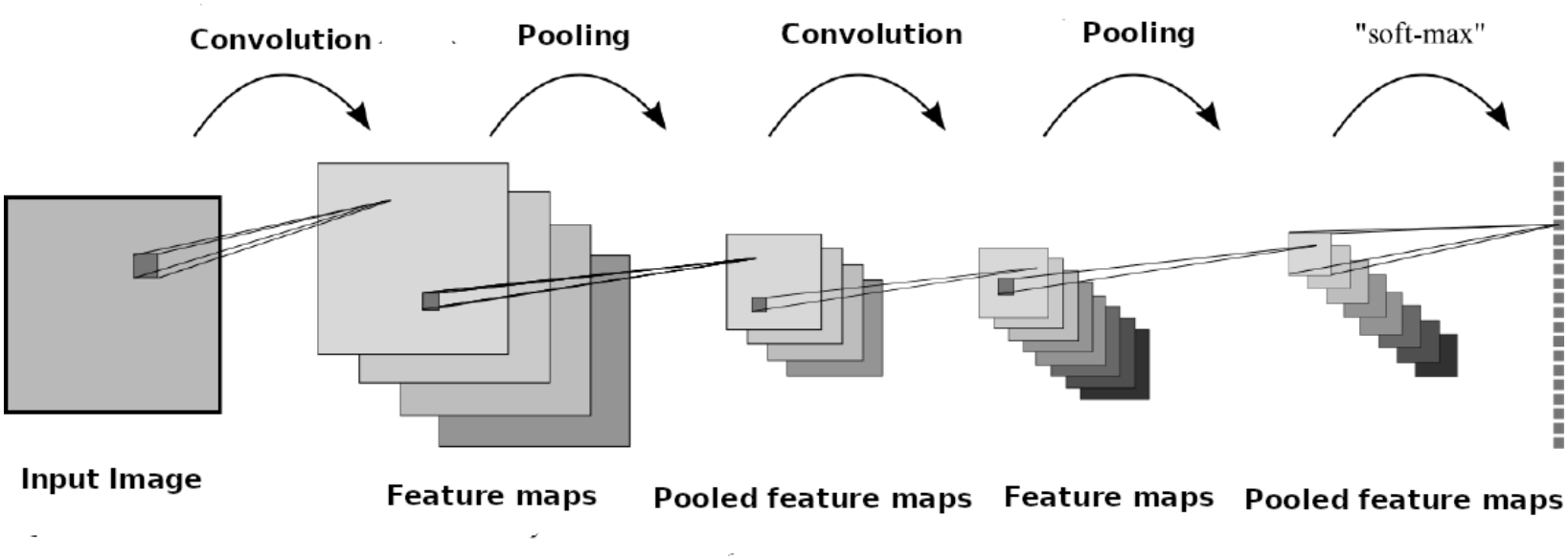
Advantages:

- No need of difficult feature engineering (like: HOG, ICF and ACF))
  - Features will be learned during the training, in incremental way
- Huge number of parameters: greater amplitude of improvement
- Fast inference

Disadvantages:

- Longer to train (ML models are usually faster to train
- Require a lot more training data (big number of parameters)
- Blackbox... almost impossible to explain how the trained model works

The Convolutional Neural Network (CNN) is as follow:

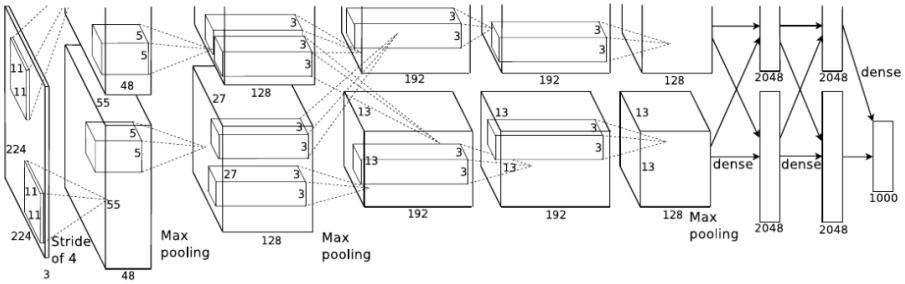


- Convolution: **local pixels** are **connected to the same pool node**
- Pooling: **features** computed in the convolution layers **are aggregated** (max or average over a pool)
- Images = many pixels, thanks to CNN: we **don't need a tremendeous number of connections**

Some of the major breakthroughs in Object Detection with Deep Learning:

- AlexNet
- R-CNN
- Fast R-CNN
- Faster R-CNN
- ResNet
- Feature Pyramid Network
- RetinaNet
- Yolo
- EfficientNet

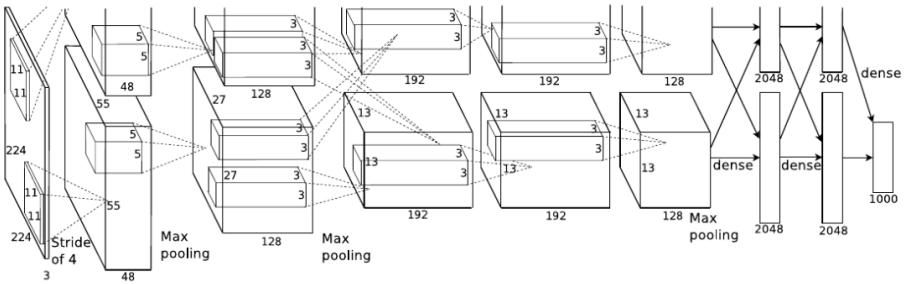
AlexNet:



- Two floors network architecture (trained and inferred by 2 GPUs)
- Made of CNN layers
- Using non-linear ReLU activation functions
- 80 millions of parameters to train
- Dropout + Data augmentation to avoid over-fitting

- AlexNet is the first DNN to win the ImageNet contest:
  - 1.6 millions of images, 1000 object classes
  - drop to 37.5% error rate (previous best: 45.1%)
  - A major breakthrough at that time

AlexNet:



- Two floors network architecture (trained and inferred by 2 GPUs)
- Made of CNN layers
- ReLU activation functions
- 80 millions of parameters to train
- Dropout + Data augmentation to avoid over-fitting

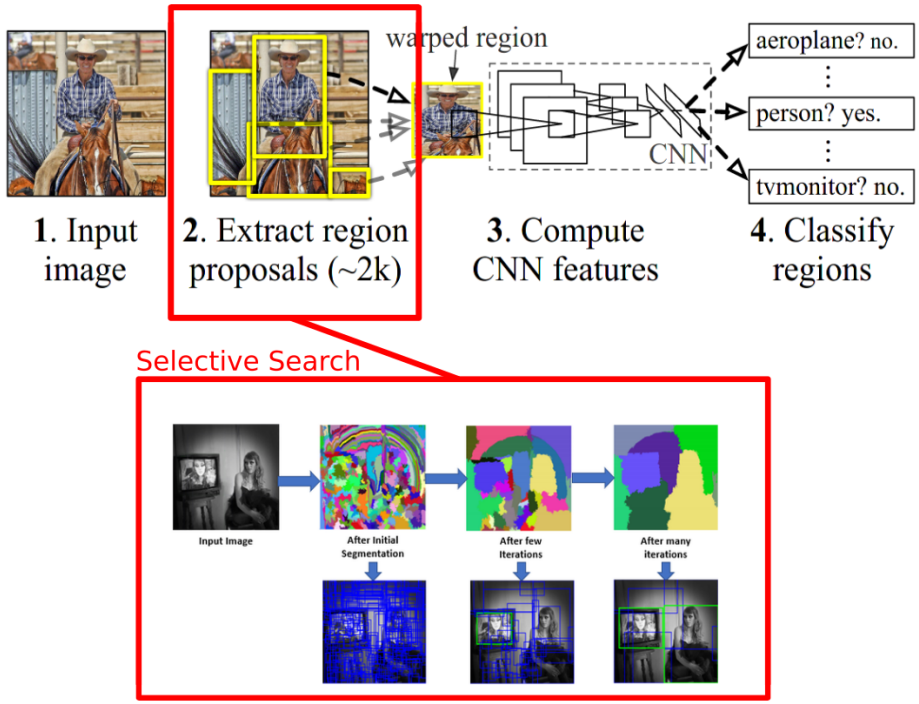
- AlexNet is the first DNN to win the ImageNet contest:
  - 1.6 millions of images, 1000 object classes
  - drop to 37.5% error rate (previous best: 45.1%)
  - A major breakthrough at that time

But ...

A **breakthrough in Object Recognition and not in Object Detection** (detecting = recognizing objects at different scales and locations)

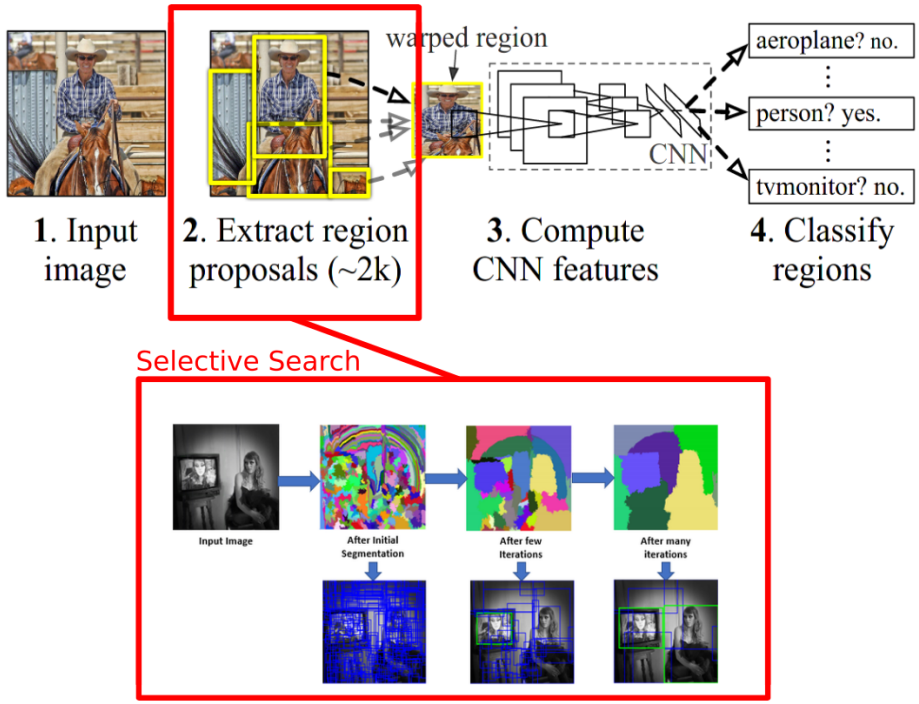
Region-based CNN (R-CNN):

- A solution to the detection problem
- Use **Selective Search** to generate region proposals to analyze
- The CNNs are only used to generate features
- Classification is performed using old-school linear SVMs
- 53.7% of mAP PASCAL 2010 (previous best: 33.4%)



Region-based CNN (R-CNN):

- A solution to the detection problem
- Use **Selective Search** to generate region proposals to analyze
- The CNNs are only used to generate features
- Classification is performed using old-school linear SVMs
- 53.7% of mAP PASCAL 2010 (previous best: 33.4%)



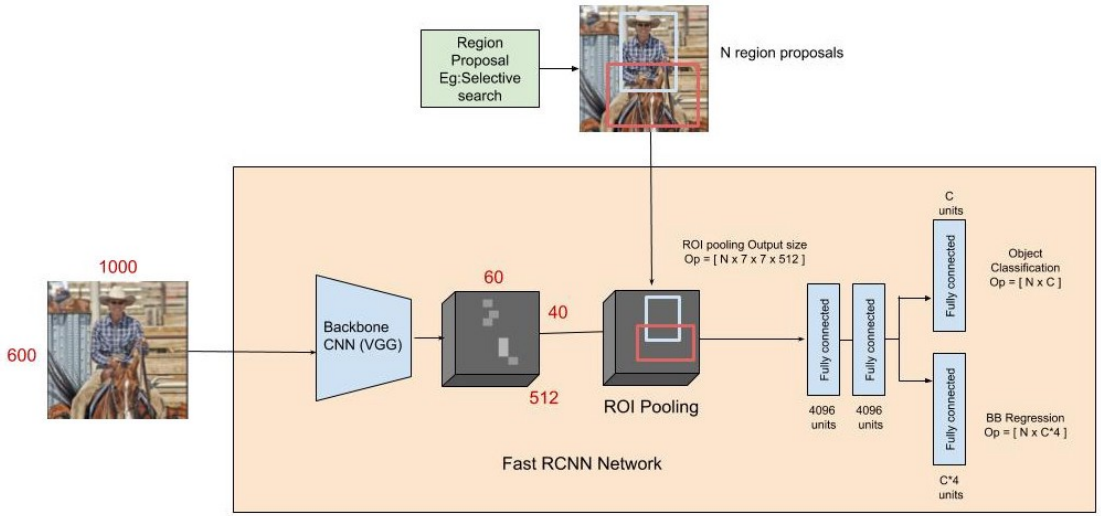
But ...

R-CNN is slow:

- Selective Search is slow...
- Features has to be recomputed with the DNN for each region proposal

Fast R-CNN:

- Still use Selective Search to generate region proposals
- Compute **all the features of the input image once** and use ROI Pooling
  - Similar to other CNN's pooling techniques
  - But used to extract region proposal features
  - Output feature vectors of size Nx7x7x512 fed to fully connected layers
  - Classification is performed by a softmax (no external classifiers)

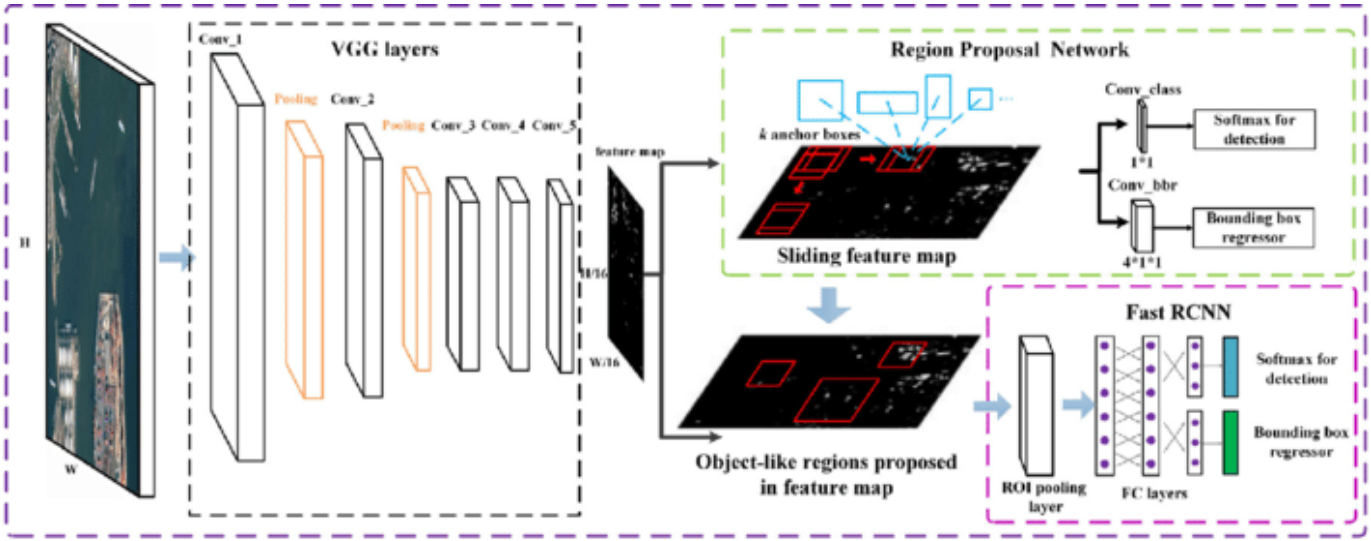


[1]

[1] VGG is an improvement of AlexNet (using fixed size kernel for convolution)

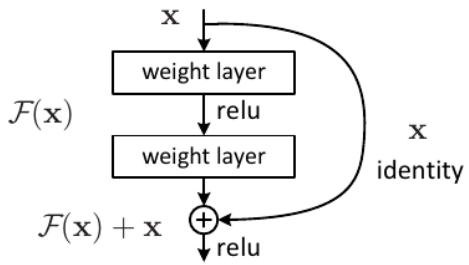
Faster R-CNN:

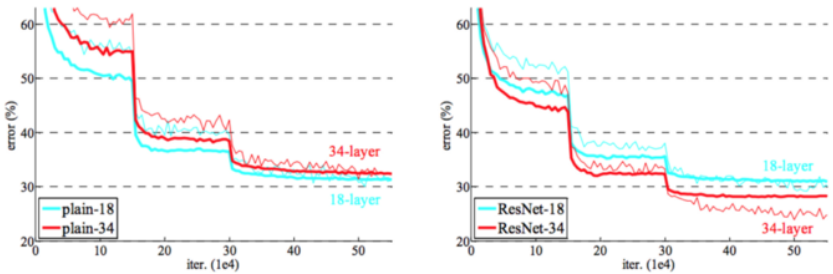
- Region proposals generated by neural network
  - Using anchors on a grid: evaluating 3 anchor boxes per cell
  - Analyzing anchor boxes permit to extract region proposals
- Now two parts to train independently:
  - Region Proposal Network (Backbone + RPN)
  - Classification Network (Backbone + Head)
- The generation of region proposals is now faster and improved
- Although training is trickier, inference is now faster



ResNet (Backbone):

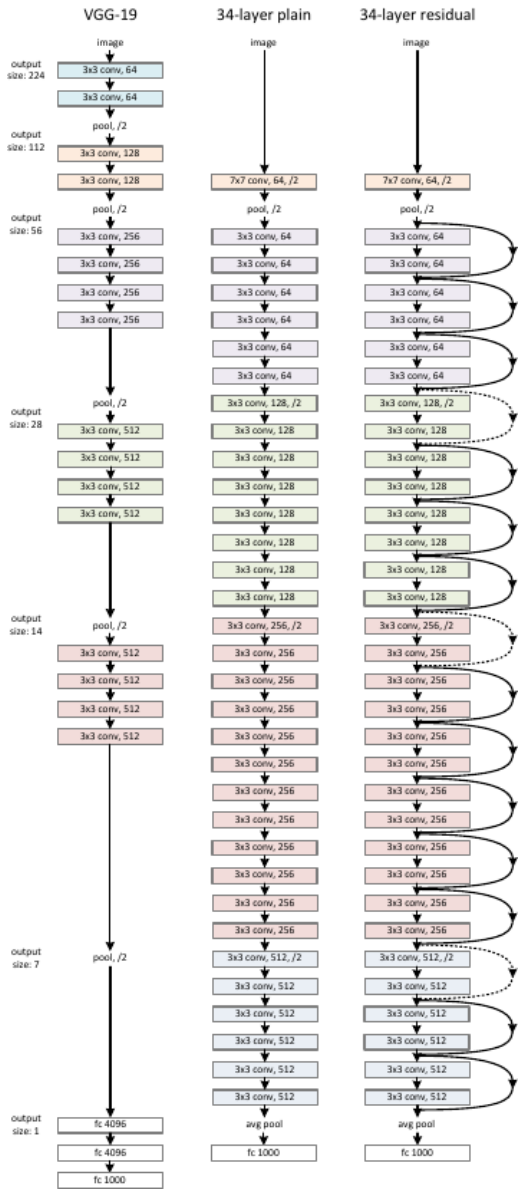
- Residual Mapping: skipped connections (inspired by biological neurons)
- Residual Mapping permit to reduce accuracy saturation drastically
  - Beginning of train: few layers trained (skipped connections)
  - Restore the skipped layers as it learns the feature space
  - This approach permit a more gradual exploration of the feature space
- With this approach learning is faster (reducing even more the vanishing gradient problem)





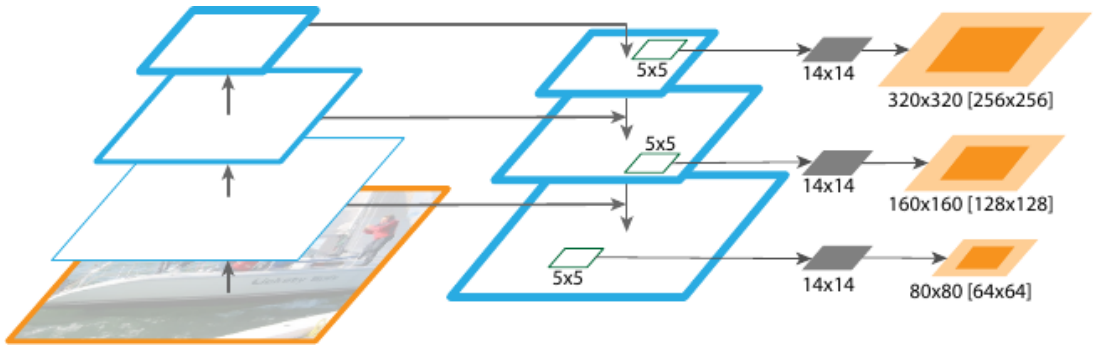
Results:

- The architecture of the network can have >100 layers with this!
- 19.38% error rate on ImageNet! (previous best: 37.5%)


VGG-19
34-layer plain
34-layer residual
image
output size: 224
3x3 conv, 64
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
3x3 conv, 128
pool, /2
output size: 56
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
output size: 28
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 14
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 7
fc 4096
fc 4096
fc 1000
output size: 1
7x7 conv, 64, /2
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
avg pool
fc 1000

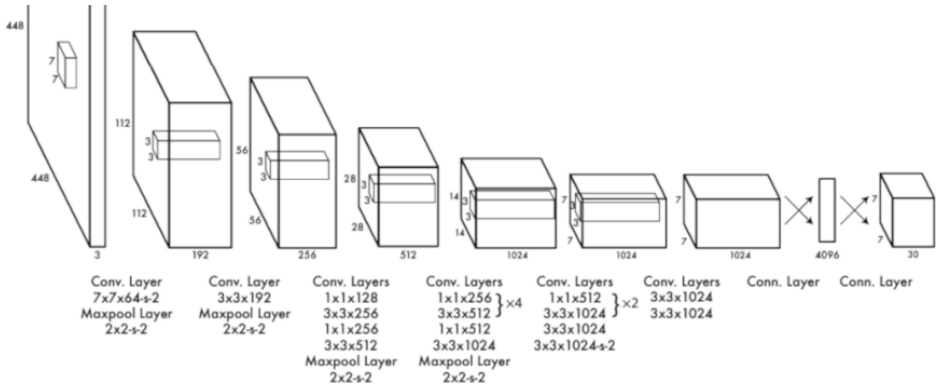
Feature Pyramid Network or FPN (modify the backbone):

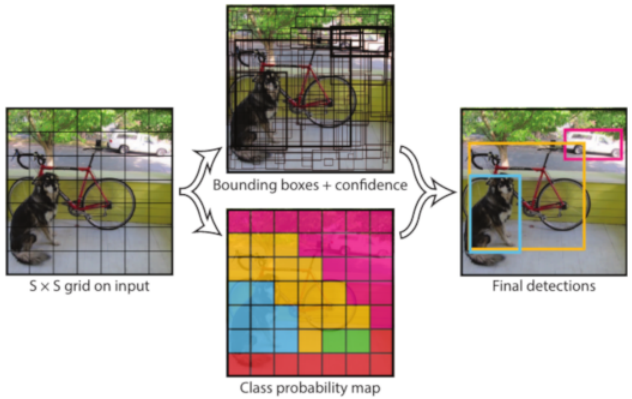
- Until now: object scaling problem was not directly addressed
- FPN permits to generate multiple feature map at multiple scales:
  - A top-down pathway restores resolution with rich semantic information
  - Lateral connections add more precise object spatial information
- Improve accuracy by 8 points on COCO, 12.9 points for small objects



You Only Look Once (YOLO):

- A unified approach: one network for all
- End-to-end training (one network):
  - Predict bounding boxes and classes
- Real-time performance with high accuracy
- But not as accurate as Faster R-CNN...





$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2]$$

$$+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2$$

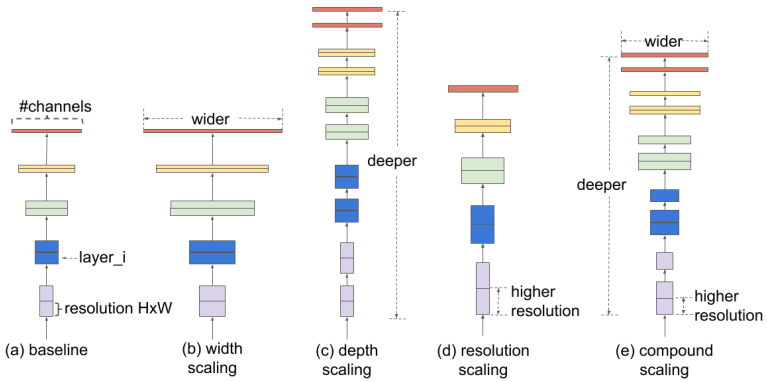
$$+ \sum_{i=0}^{S^2} \mathbf{1}_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2$$

2

[2] Above, predictions are encoded in a S × S × (B * 5 + C) tensor, where, S = 7, B = 2 and C = 20: 20 object classes, 2 bounding boxes, 4 spacial information (x, y, w, h) and a confidence score

EfficientNet (Backbone) (2020)

- Before: no real scaling guideline when designing a backbone
- Based on the intuitions that we should coordinate and balance the scaling of the dimensions
- EfficientNet are optimized in width, depth and resolution networks
- EfficientNet-B0 achieves 77.3% accuracy on ImageNet:
  - With only 5.3M parameters!
  - Resnet-50 provides 76% accuracy with 26M parameters
- Since EfficientNet have less parameters, there are faster



$$depth = \alpha^{\Phi}$$

$$width = \beta^{\Phi}$$

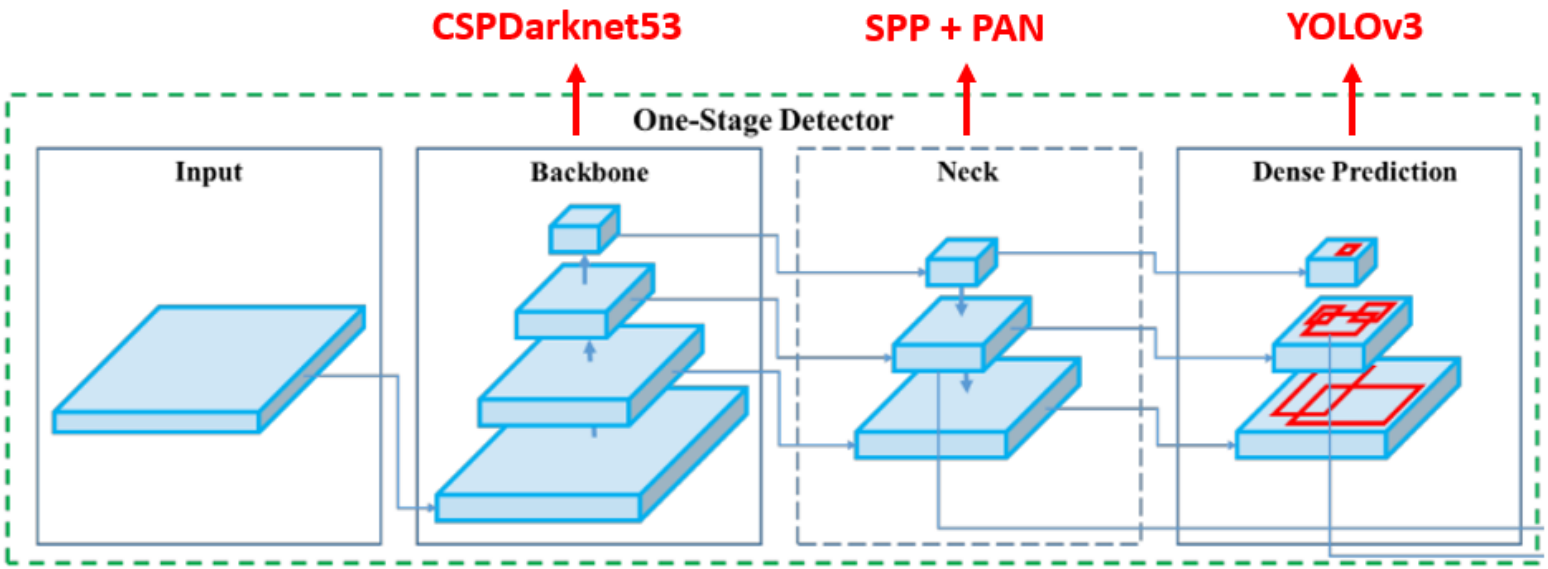
$$resolution = \gamma^{\Phi}$$

$$s.t.\ \alpha^{\Phi} \times \beta^{\Phi} \times \gamma^{\Phi} \approx 2 \text{ and } \alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

α, β, γ are found by search grid Φ is user specified a compound coefficient to uniformly scale the network

YOLO v4 (2020):

- This is YOLO, on steroids...
- Still a regression approach
- But embeds improvements of other detectors:
  - SAT Data Augmentation, New Loss, DropBlock, Etc.
  - SPP/PAN, Attention layers, Etc.
- Achieves state-of-the-art results in real time on MS COCO:
  - 43.5 % AP
  - 65 FPS on a Tesla V10



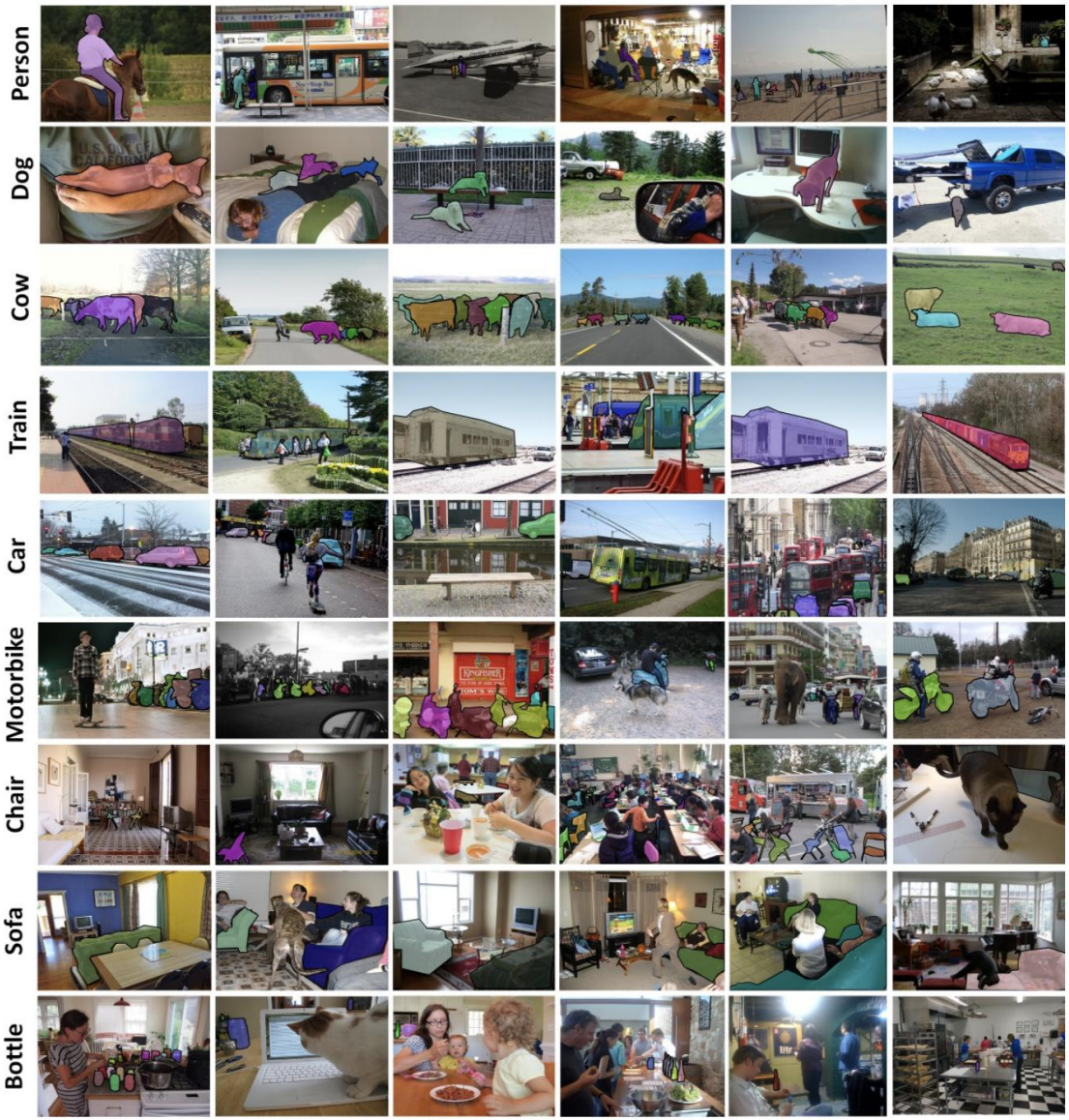
To conclude:

- Former Machine Learning approaches for OD: obsolete
- Performances improved thanks to Deep Learning
- Year after year, Deep Learning-based Object Detection becomes ...
  - ... simpler (one step training, etc.)
  - ... more accessible (cheaper and cheaper powerful GPU, etc.)
  - ... more accurate (new optimization, etc.)
  - ... speeder.
- Trend 1: one unique network for all detection steps
- Trend 2: optimized networks
- Trend 3: important re-usage of techniques and tricks

In CVPR 2021: detection in 3d (point cloud)

The course continue...

MS COCO dataset:



PASCAL VOC 2007 dataset: