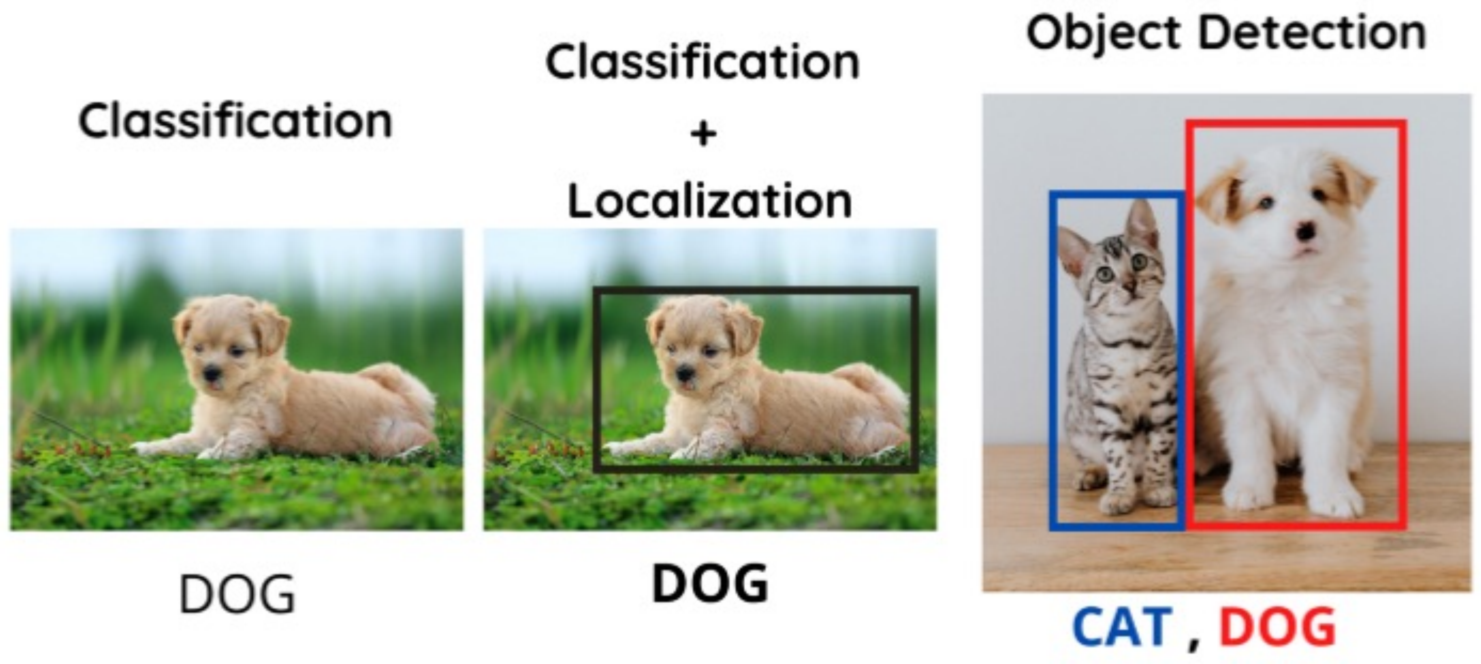
Object Detection

Qingrun Zhang

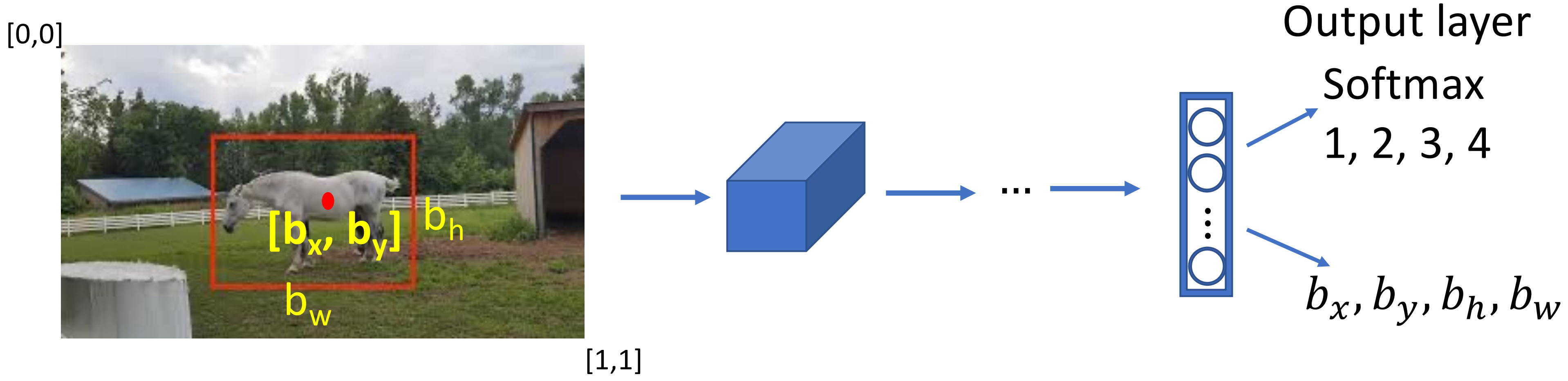
Image Classification vs. Object Detection

- **Image Classification**: Predict the type or class of an object in an image.
    - *Input*: An image with a single object.
    - *Output*: A class label (e.g. one or more integers that are mapped to class labels).
- **Object Localization**: Locate the presence of objects in an image and indicate their location with a bounding box.
    - *Input*: An image with one or more objects.
    - *Output*: One or more bounding boxes (e.g. defined by a point, width, and height).
- **Object Detection**: Locate the presence of objects with a bounding box and types or classes of the located objects in an image.
    - *Input*: An image with one or more objects, such as a photograph.
    - *Output*: One or more bounding boxes (e.g. defined by a point, width, and height), and a class label for each bounding box.

Classification, localization and detection



Classification with localization



1. Horse
2. Cow
3. Human
4. background

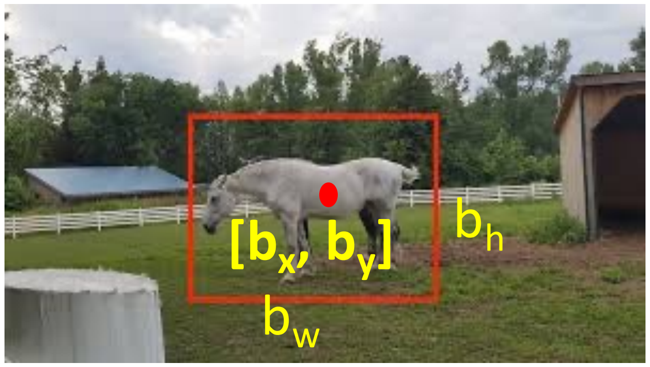
b_x, b_y, b_h, b_w define the bounding box of the object being detected.

$$b_x = 0.5, \quad C_1 = 1$$
$$b_y = 0.5, \quad C_2 = 0$$
$$b_h = 0.5, \quad C_3 = 0$$
$$b_w = 0.35$$

Defining the target label y

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

$p_c = 1$: there is an object, then we need to define the location, and c_i define the probability of which class it is.



1. Horse	$b_x = 0.5,$
2. Cow	$b_y = 0.5,$
3. Human	$b_h = 0.5,$
4. background	$b_w = 0.35$

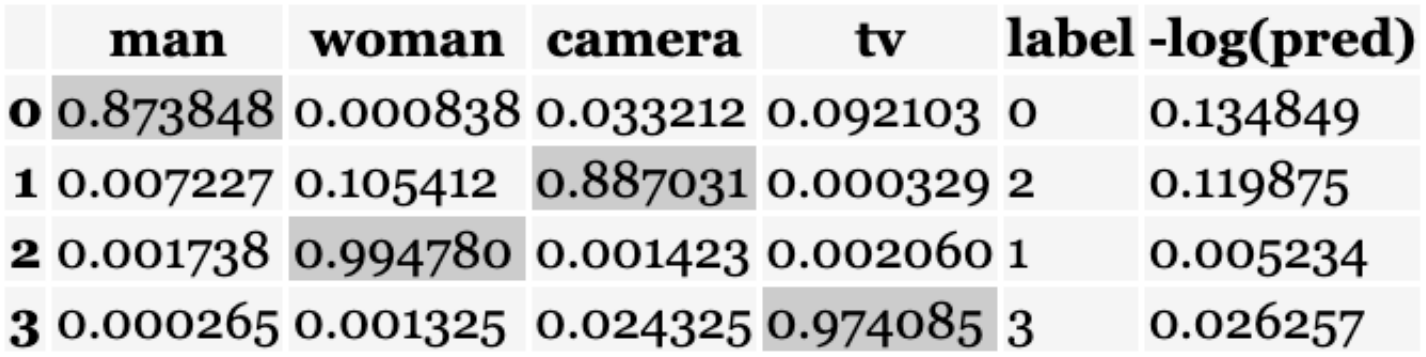
$$\hat{y} = \begin{bmatrix} p_c = 1 \\ b_x = 0.33 \\ b_y = 0.25 \\ b_h = 0.5 \\ b_w = 0.35 \\ c_1 = 1 \\ c_2 = 0 \\ c_3 = 0 \end{bmatrix} \qquad \hat{y} = \begin{bmatrix} p_c = 0 \\ b_x =? \\ b_y =? \\ b_h =? \\ b_w =? \\ c_1 =? \\ c_2 =? \\ c_3 =? \end{bmatrix}$$

Defining loss function

- $\mathcal{L}(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + \cdots + (\hat{y}_8 - y_8)^2, if\ y_1 = 1 (Has\ an\ object) \\ (\hat{y}_1 - y_1)^2, if\ y_1 = 0 \end{cases}$

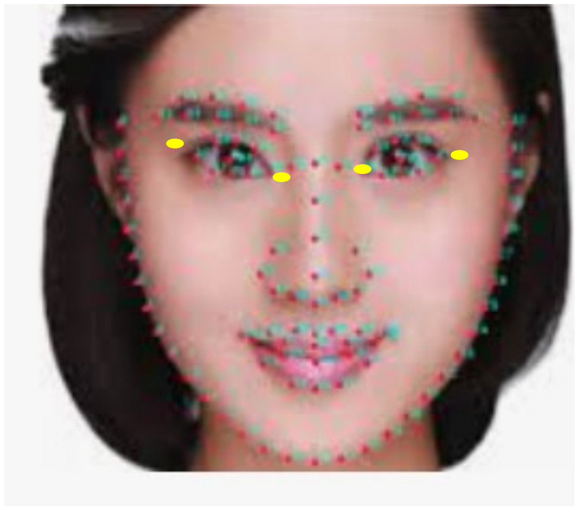
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

In practice you could use a negative log likelihood loss (-ln$\hat{y}$) for c1, c2, c3 to the softmax output. Squared error for the bounding box coordinates and for pc you could use the logistics regression loss.



	man	woman	camera	tv	label	-log(pred)
0	0.873848	0.000838	0.033212	0.092103	0	0.134849
1	0.007227	0.105412	0.887031	0.000329	2	0.119875
2	0.001738	0.994780	0.001423	0.002060	1	0.005234
3	0.000265	0.001325	0.024325	0.974085	3	0.026257

Landmark Detection



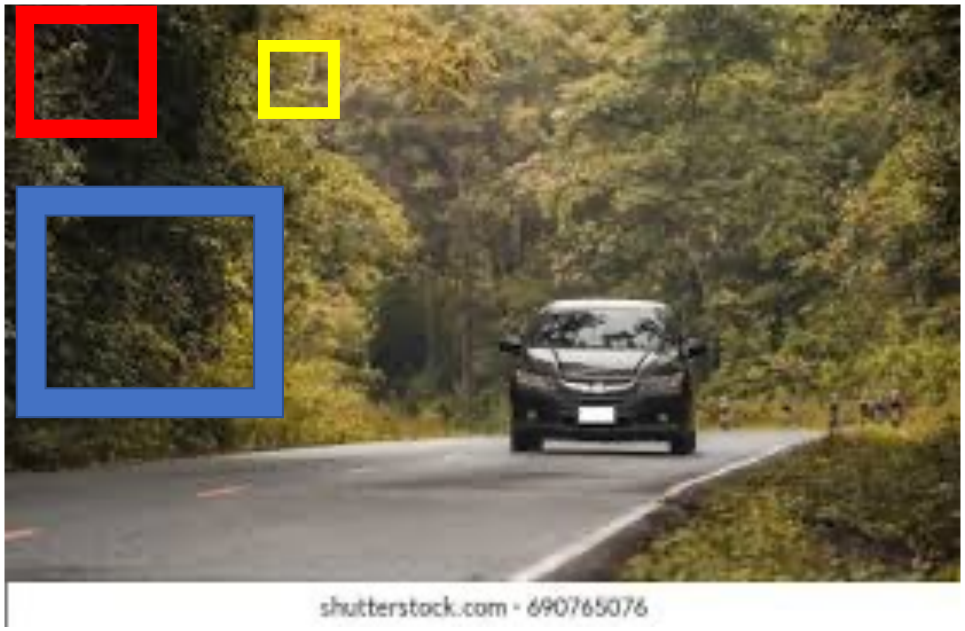
$$x, y = \begin{Bmatrix} L_{1x} & L_{1y} \\ L_{2x} & L_{2y} \\ \vdots & \vdots \\ L_{nx} & L_{by} \end{Bmatrix}$$

Example: face recognition, we would like to detect the edge of eyes, nose, mouth etc, so our neural network output are a list of landmark coordinates.

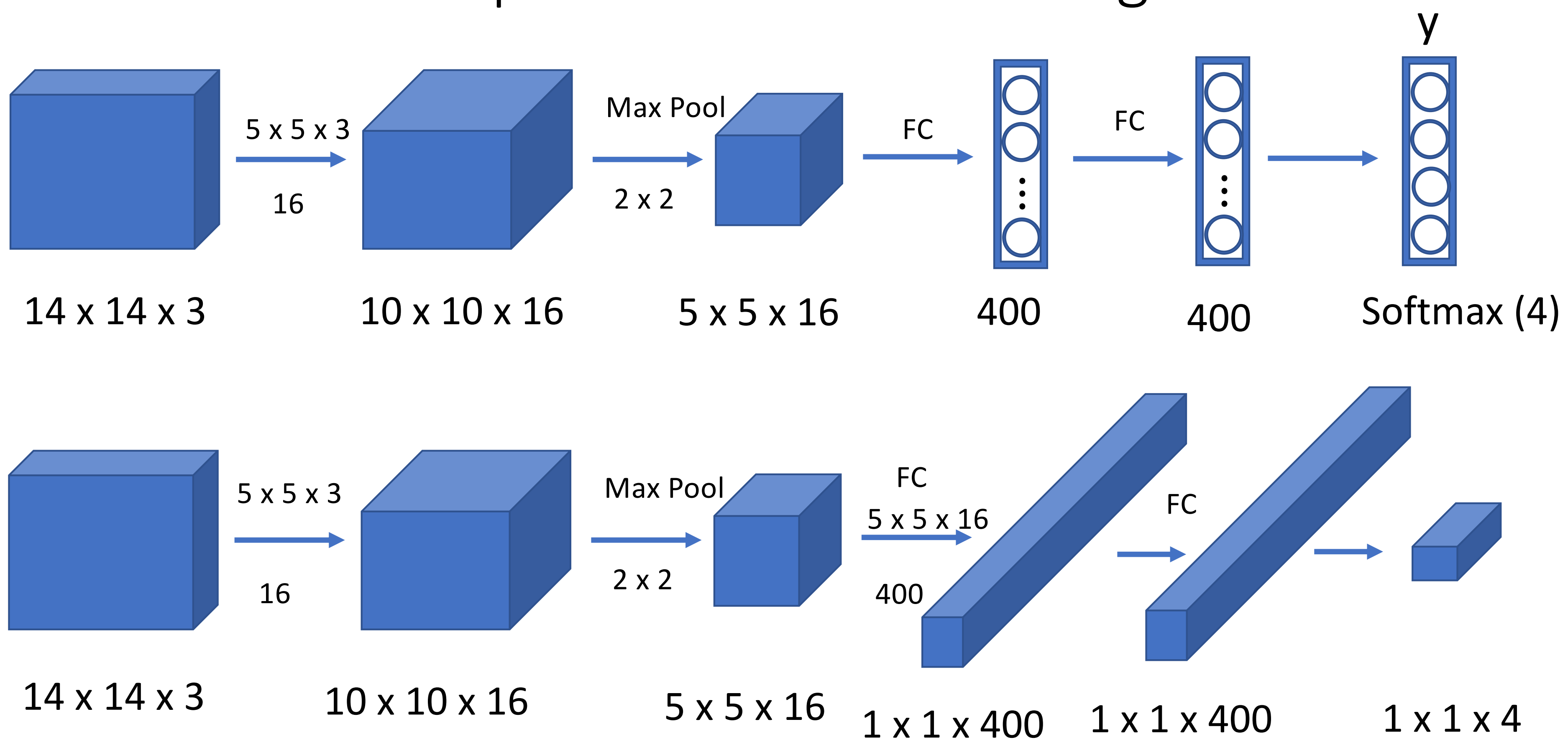
We need pre-labeled dataset to train the NN model

All labels have to be consistent across different images

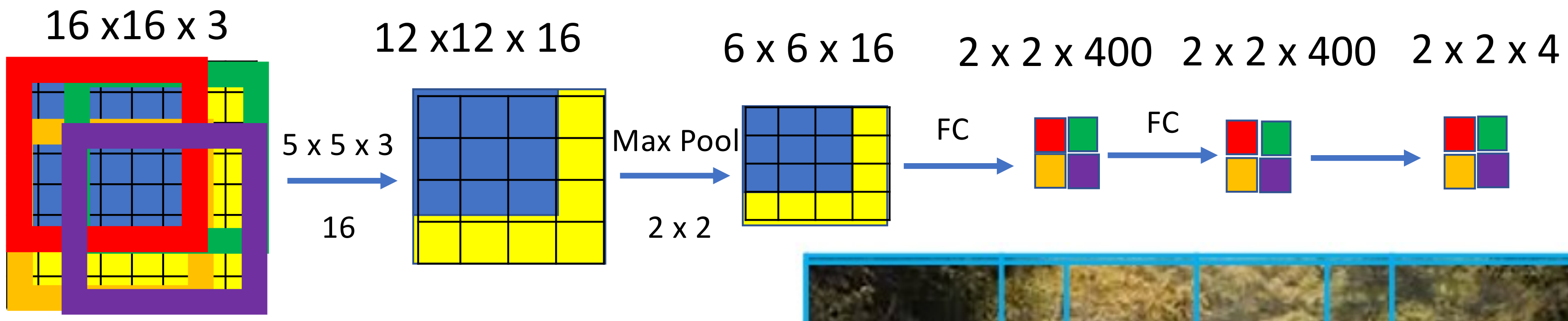
Sliding window detection



Convolutional implementation of sliding windows

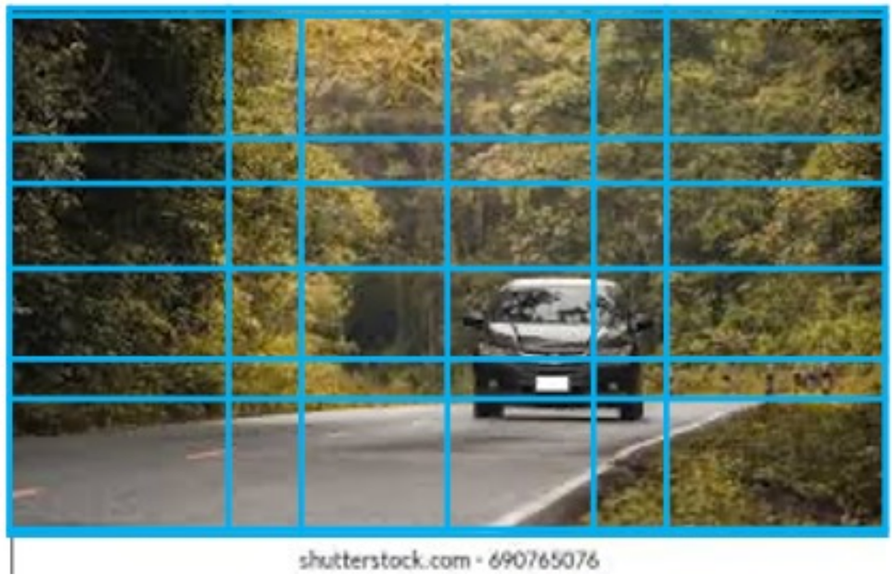


Convolutional implementation of sliding windows



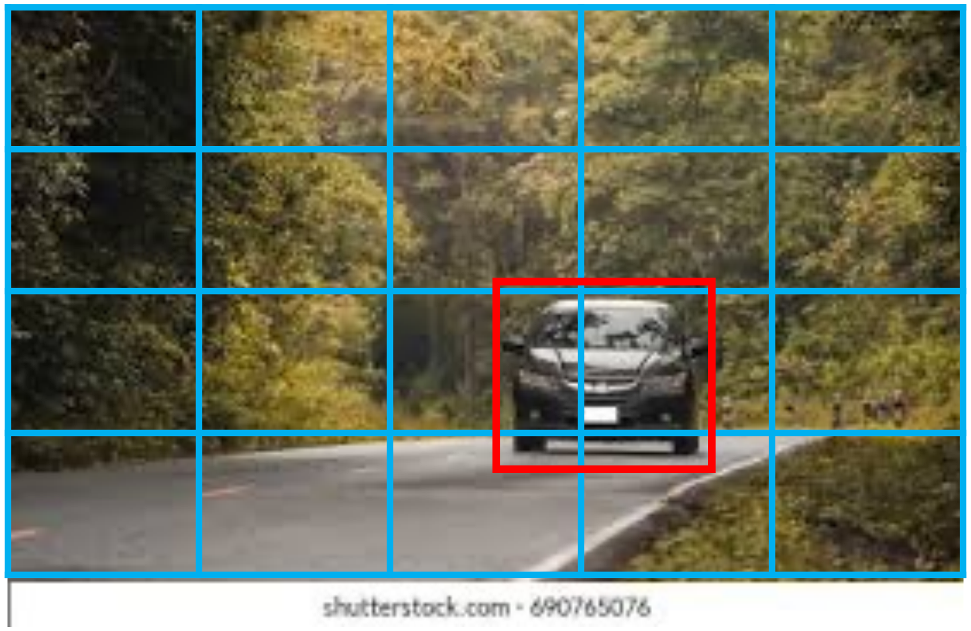
Pros: Instead of doing it sequentially, one can implement the entire image, make all the predictions by one forward pass through

Cons: the position of the bounding boxes is not accurate



Sermanet et al, 2014 OverFeat: Integrated recognition, localization and detection using convolutional networks

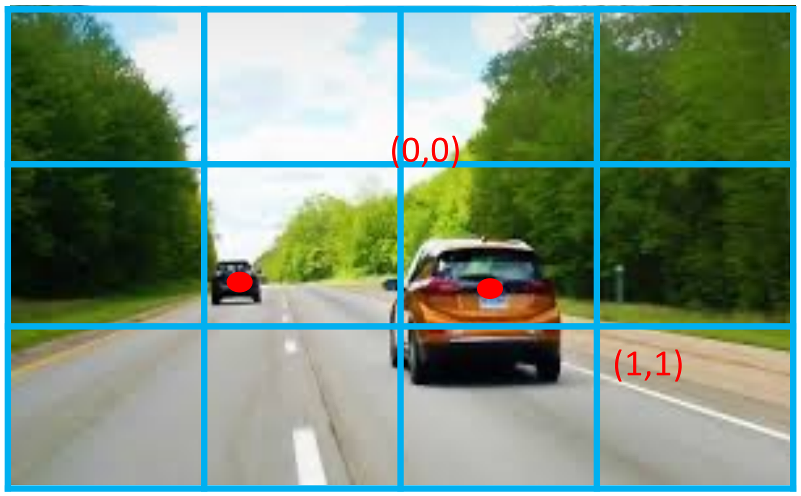
Bounding box prediction



- With sliding windows, in the case, none of the boxes really match up perfectly with the position of the car
- Our goal is to get more accurate bounding boxes
- Solution: YOLO (You Only Look Once) algorithm

Redmon et al., 2015 You Only Look Once: Unified, Real-Time Object Detection

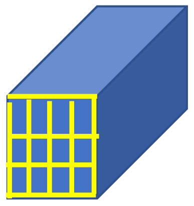
Bounding box prediction



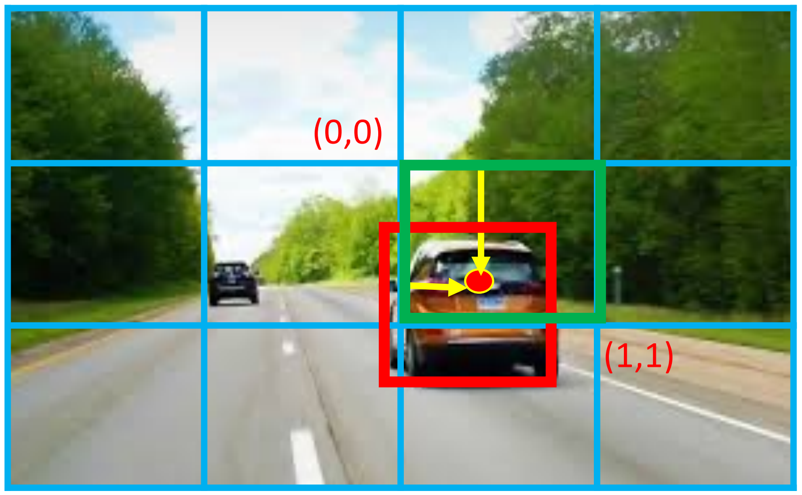
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 0.1 \\ 0.4 \\ 0.8 \\ 0.8 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

- YOLO algorithm takes the midpoint of reach objects and then assigns the object to the grid cell containing the midpoint.
- The target output will be 3 x 4 x 8, because 3 x 4 cells
- Using finer grid to reduce the chance of multiple objects in one cell

- Place down a grid on the input image.
- Apply image classification and localization algorithm on each of the grids
- Define labels Y for each of the grid cells.


3 x 4 x 8

How to encode bounding box?



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

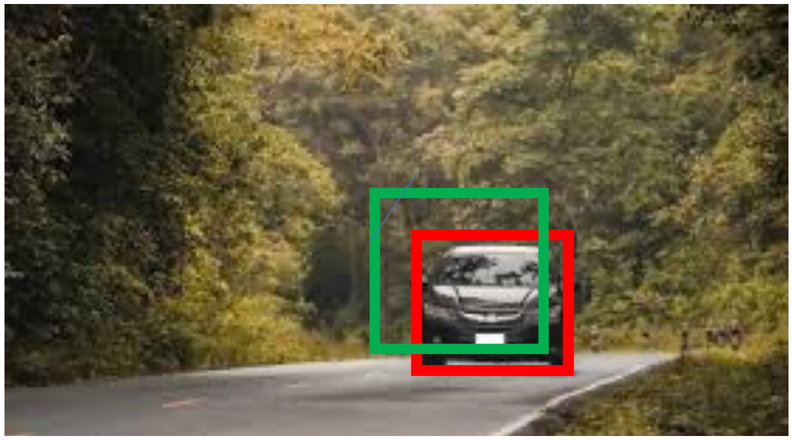
- $b_x = 0.4$
- $b_y = 0.7$
- $b_h = 1.0$
- $b_w = 0.8$

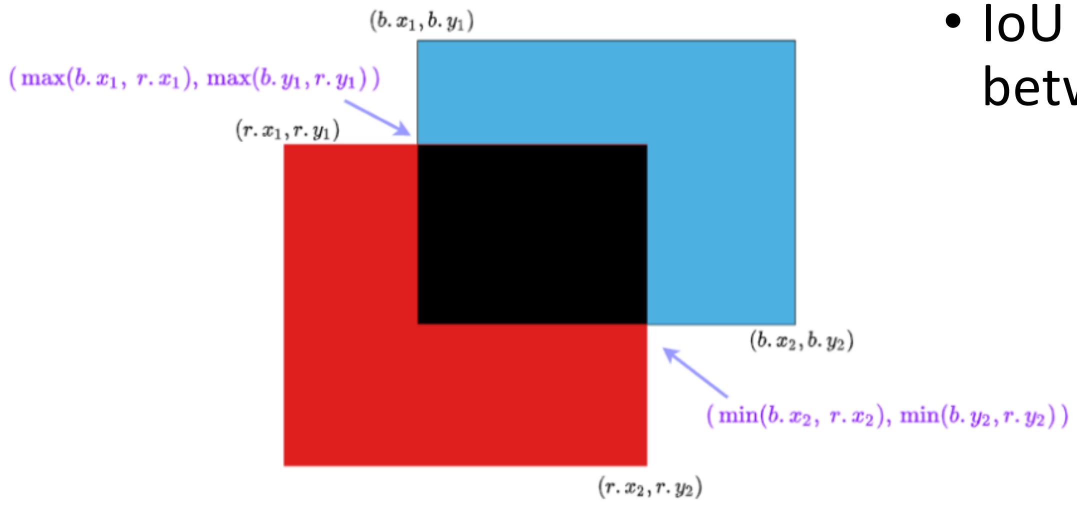
b_x, b_y, b_h, b_w are specified relative to the grid cell.

b_x, b_y, should between 0 and 1. but b_h, b_w could be greater than 1

Redmon et al., 2015 You Only Look Once: Unified, Real-Time Object Detection

Evaluating object localization accuracy





$(b.x_1, b.y_1)$
$(\max(b.x_1,\ r.x_1),\ \max(b.y_1, r.y_1))$
$(r.x_1, r.y_1)$
$(b.x_2, b.y_2)$
$(\min(b.x_2,\ r.x_2),\ \min(b.y_2, r.y_2))$
$(r.x_2, r.y_2)$

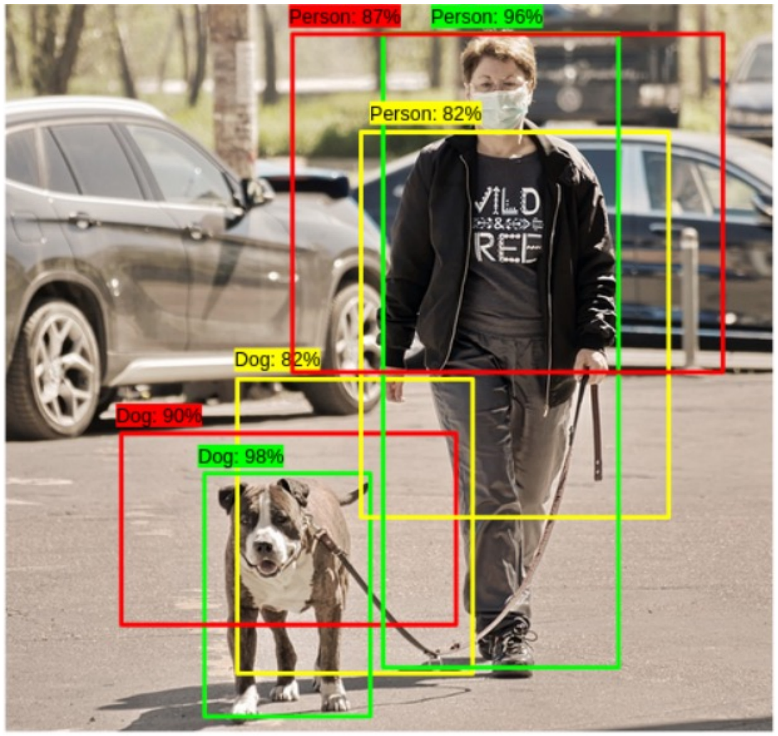
- Intersection over Union (IoU)
- Green is the algorithm identified.
- Red is the ground truth
- $IoU = \dfrac{intersection(Red,Green)}{Union(Red,Green)}$
- Correct when $IoU \geq 0.5$
- IoU is used the measure the similarity between two bounding boxes.

$$r_{\text{area}} = (r.\,x_2 - r.\,x_1) \times (r.\,y_2 - r.\,y_1)$$
$$b_{\text{area}} = (b.\,x_2 - b.\,x_1) \times (b.\,y_2 - b.\,y_1)$$

$$\text{width} = \min(b.\,x_2,\ r.\,x_2) - \max(b.\,x_1, r.\,x_1)$$
$$\text{height} = \min(b.\,y_2,\ r.\,y_2) - \max(b.\,y_1, r.\,y_1)$$

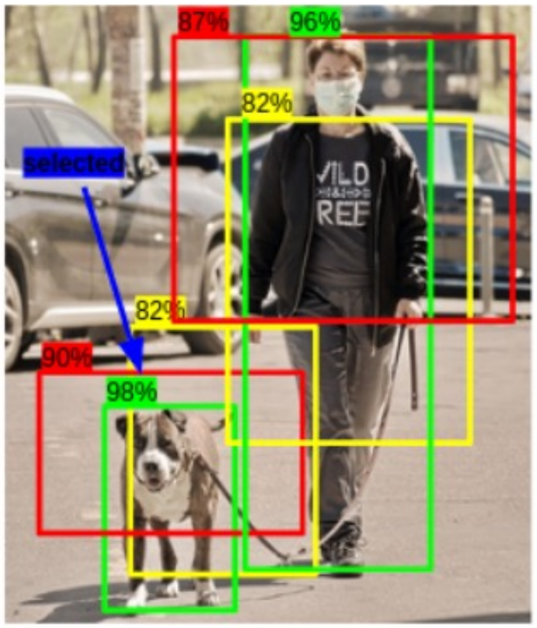
$$\text{intersection} = \text{width} \times \text{height}$$
$$\text{union} = r_{\text{area}} + b_{\text{area}} - \text{intersection}$$

Non Max Suppression (NMS)

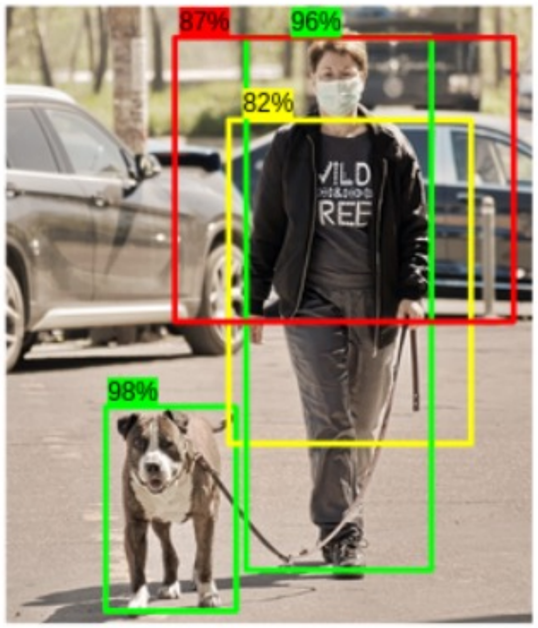


1. Define a value for Confidence_Threshold, and IOU_Threshold.
2. Sort the bounding boxes in a descending order of confidence.
3. Remove boxes that have a confidence < Confidence_Threshold
4. Loop over all the remaining boxes, starting first with the box that has highest confidence. Box_S
5. Calculate the IOU of the Box_S, with every remaining box
6. If the IOU of the 2 boxes > IOU_Threshold, remove the box with a lower confidence from our list of boxes.
7. Repeat this operation until we have gone through all the boxes in the list.

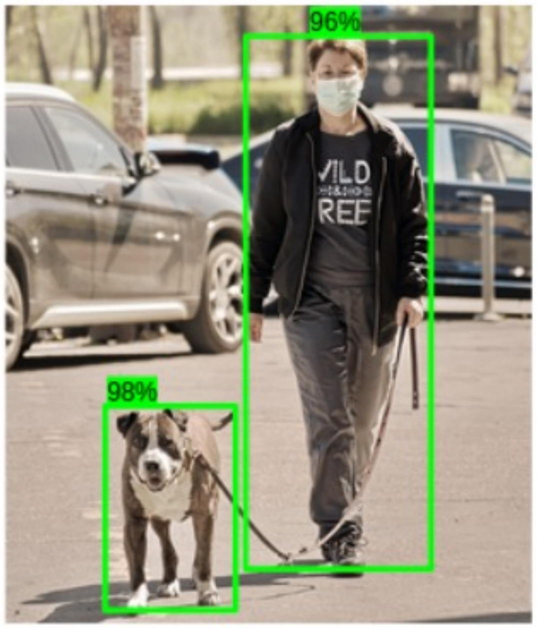
Non Max Suppression (NMS)



Step 1: Selecting Bounding box with highest score

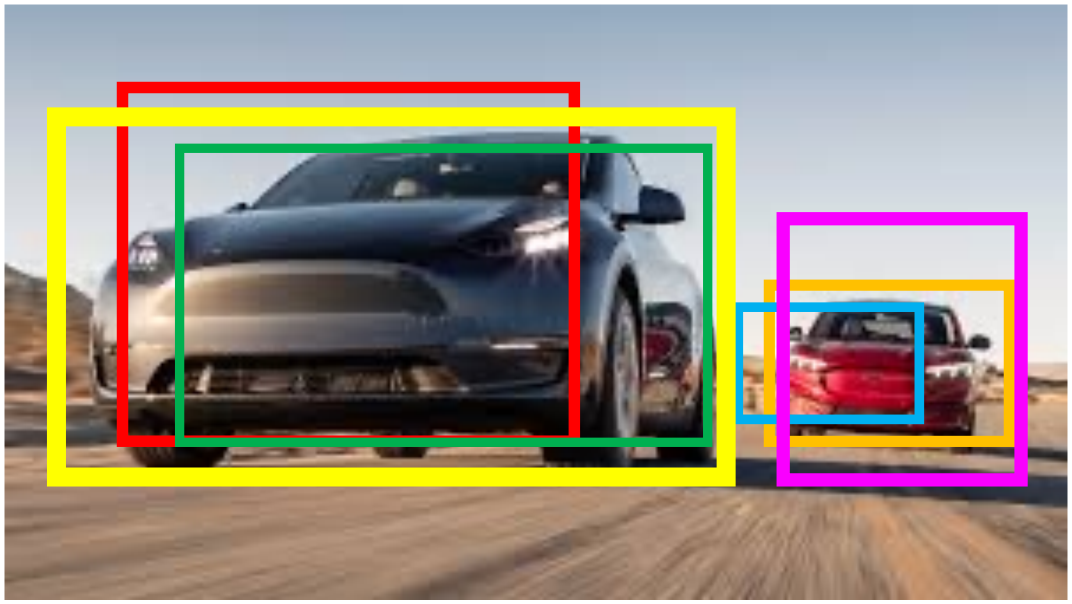


Step 3: Delete Bounding box with high overlap



Step 5: Final Output

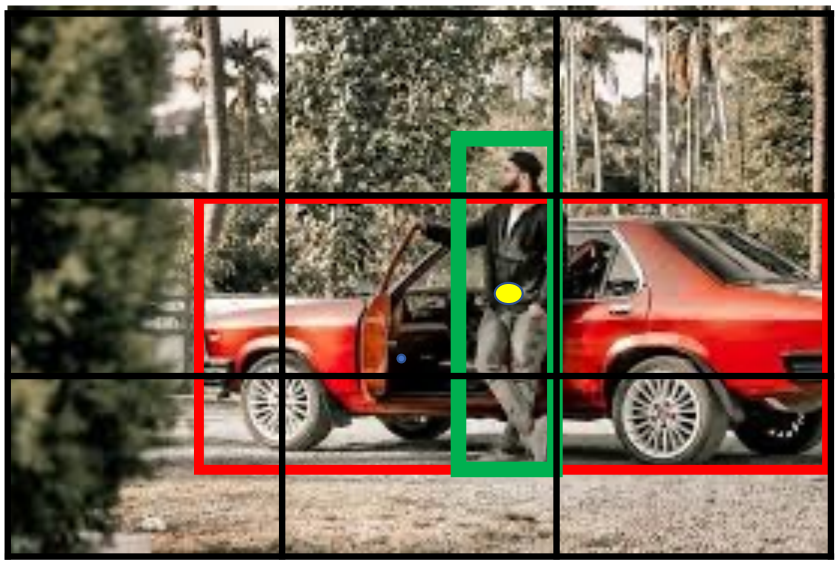
Non Max Suppression (NMS)

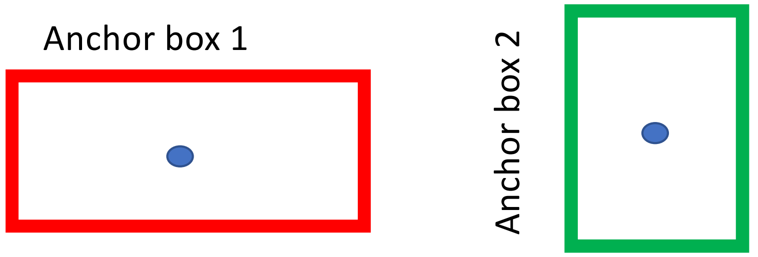


0.98, x1, x2, x3, x4
0.43, x1, x2, x3, x4
0.72, x1, x2, x3, x4
0.68, x1, x2, x3, x4
0.96, x1, x2, x3, x4
0.89, x1, x2, x3, x4

Discard all boxes with $p_c \leq 0.6$,
Pick the box with the largest p_c, put it in the prediction category. box_yellow
Discard any remaining box with $IoU \geq 0.5$ *with the box_yellow*
Repeat the process until no more left

Anchor boxes





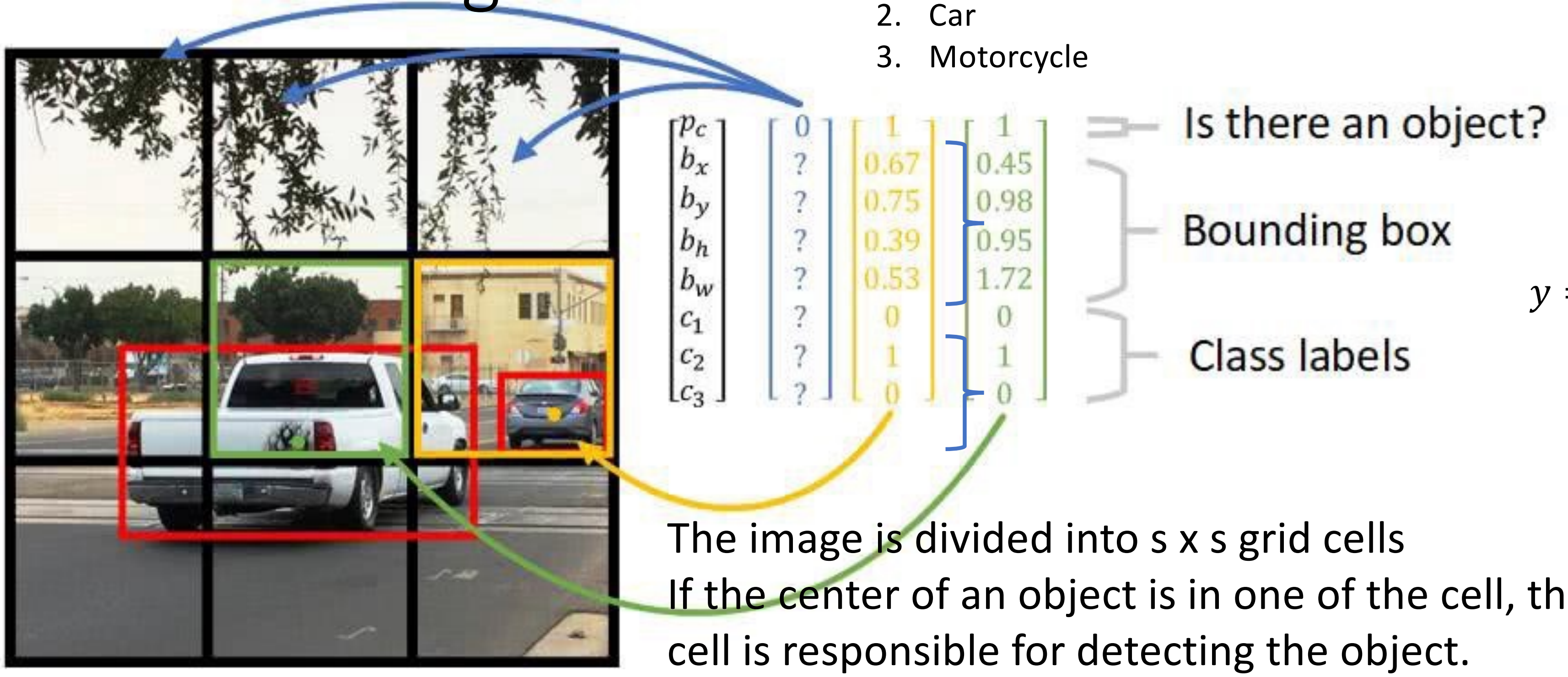
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

- Predefine several anchor boxes for each grid cell.
- Each object in the training image is assigned to grid cell that contains the midpoint and anchor box for the grid cell with highest IoU
- Not working well:
  - When the # of objects in one grid cell is more than # of anchor boxes
  - Or several objects in the same grid cell, but have same anchor box shape

Redmon et al. 2015, You Only Look Once: Unified real-time object detection

YOLO algorithm

1. Pedestrian
2. Car
3. Motorcycle

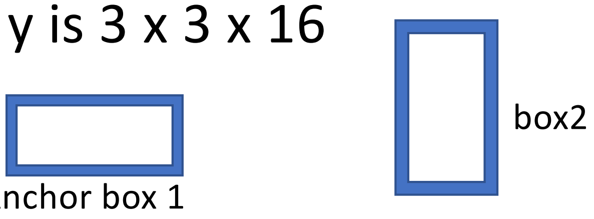


$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \begin{bmatrix} 1 \\ 0.67 \\ 0.75 \\ 0.39 \\ 0.53 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0.45 \\ 0.98 \\ 0.95 \\ 1.72 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

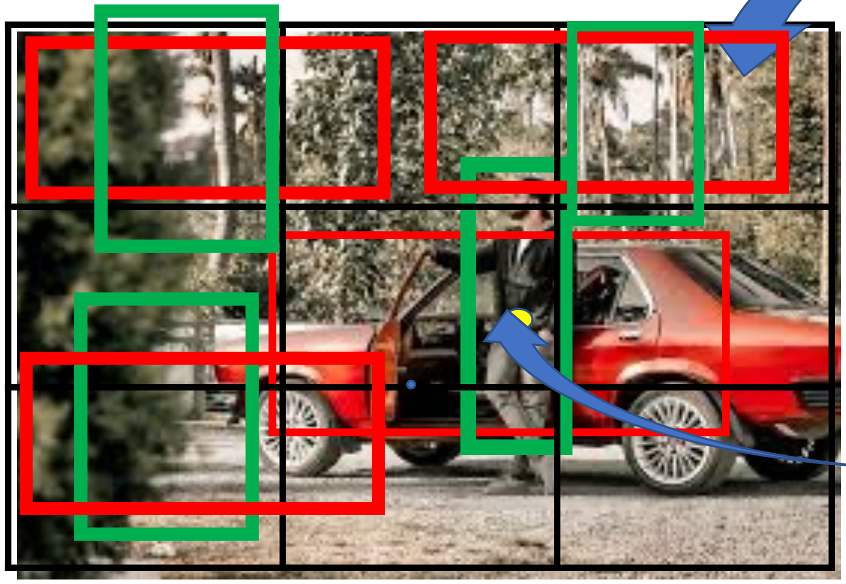
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ 0.6 \\ 0.5 \\ 1 \\ 1.6 \\ 0 \\ 1 \\ 0 \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

The image is divided into s x s grid cells
If the center of an object is in one of the cell, that cell is responsible for detecting the object.
Each of the cells is responsible for predicting 'B' boxes each contain an object and confidence score for the object present in the box
YOLO reasons at the level of the overall picture

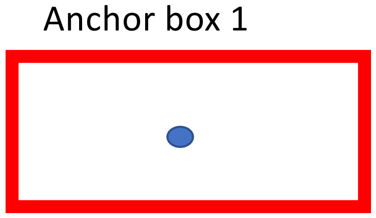
If you use 2 anchor boxes:
y is 3 x 3 x 16

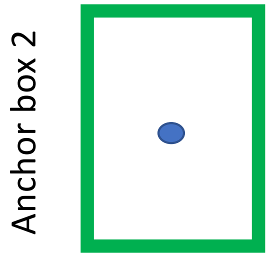


YOLO algorithm



$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ 0.5 \\ 0.8 \\ 1.2 \\ 2.0 \\ 0 \\ 1 \\ 0 \\ 0.98 \\ 0.5 \\ 0.8 \\ 2 \\ 0.3 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad y = \begin{bmatrix} 0 \\ \\ \\ \\ \\ 0 \\ 0 \\ 0 \\ 0 \\ \\ \\ \\ \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



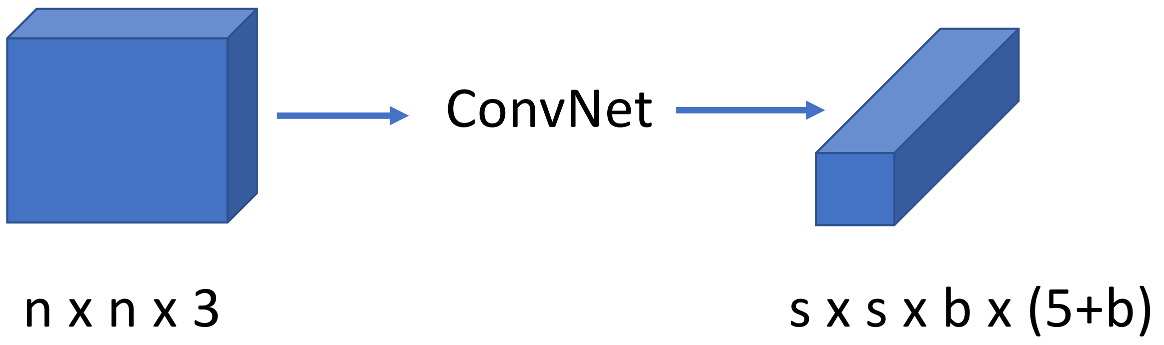


Does the anchor box size predefined or H/W ratio?

Redmon et al. 2015, You Only Look Once: Unified real-time object detection

YOLO algorithm

- For each grid cell, get ”B” predicted bounding boxes
- Get rid of low probability predictions
- For each class (pedestrian, car, motorcycle), independently run Non-Max Suppression to get the final predictions

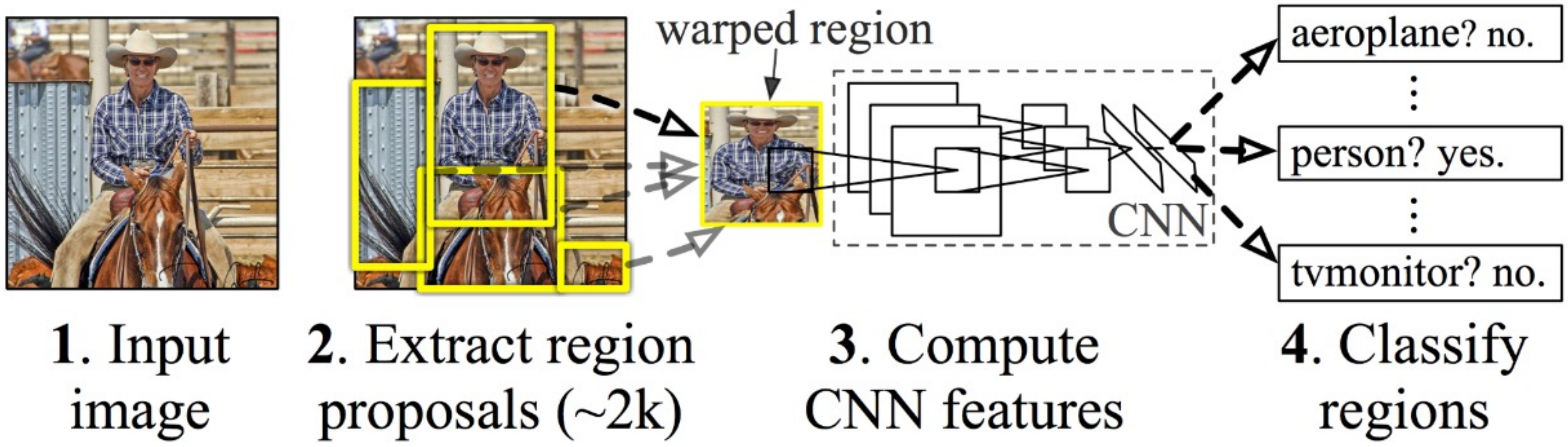


S x S: # of grid

B : # of classes

5: IoU and b_x, b_y, b_w, b_h

R-CNN (Region-based Convolutional Neural Networks)

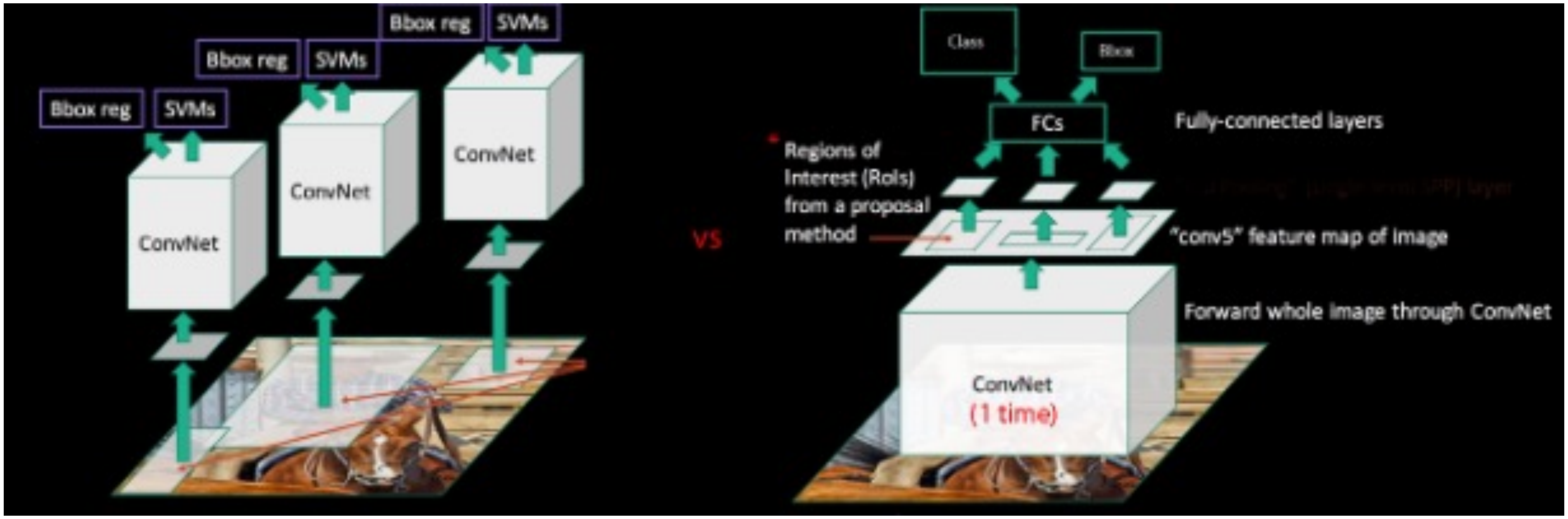


- The selective search algorithm: similar to sliding window exhaustive search combines with the segmentation of the colours presented in the image to select region of interest.
- Then the image with regions goes through a CNN to extracts the objects from the region.
- R-CNN reshape the region into 227 x 227 x 3 size images

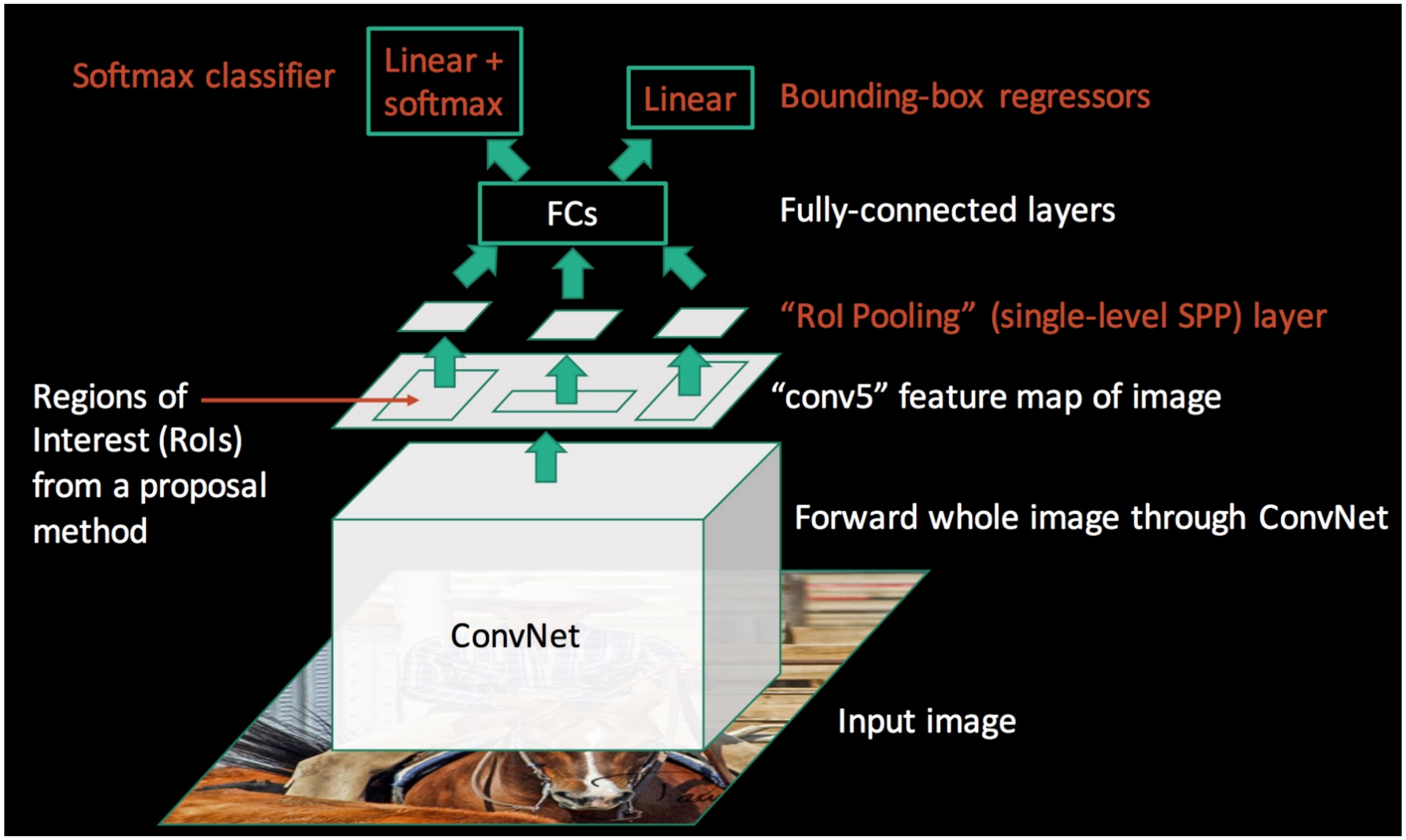
Grishick et al, 2014 Rich feature hierarchies for accurate object detection and semantic segmentation

SPPNet (Spatial Pyramid Pooling Network)

- SPPNet instead of working on the 2000 region to convert them into feature maps, it converts the whole image on the feature map at once.

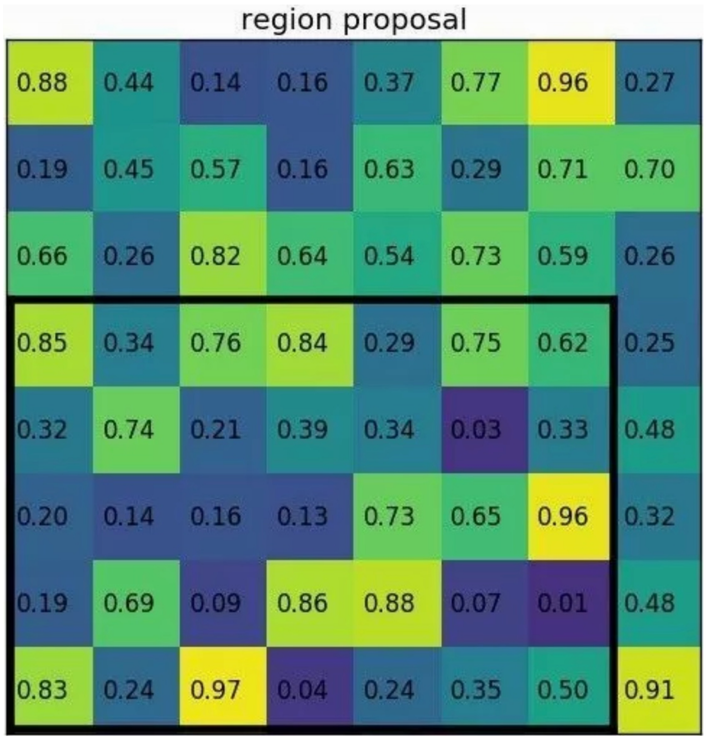

Bbox reg
SVMs
Bbox reg
SVMs
Bbox reg
SVMs
ConvNet
ConvNet
ConvNet
VS
Class
Bbox
FCs
Fully-connected layers
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
(1 time)

Fast R-CNN

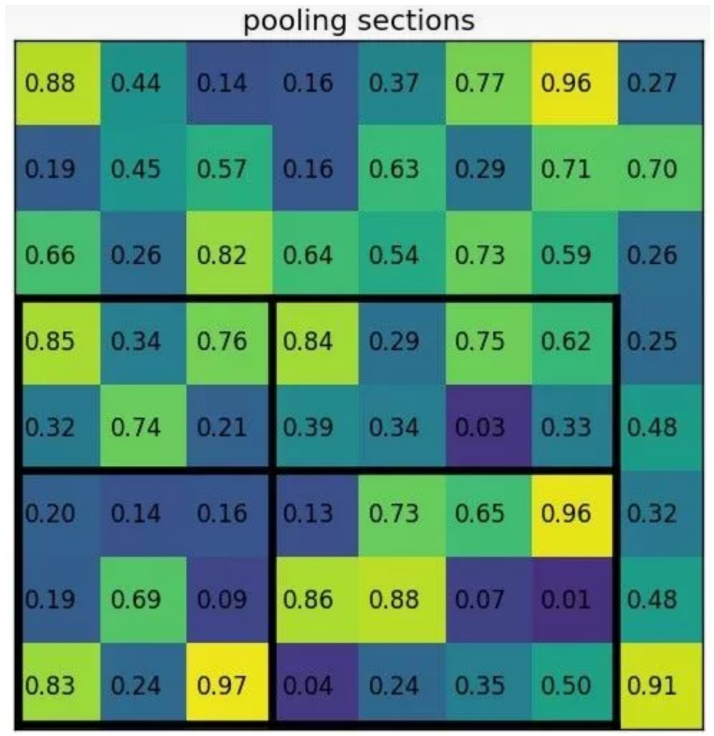

Softmax classifier
Linear + softmax
Linear
Bounding-box regressors
FCs
Fully-connected layers
"RoI Pooling" (single-level SPP) layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image

Girshick 2015 Fast R-CNN

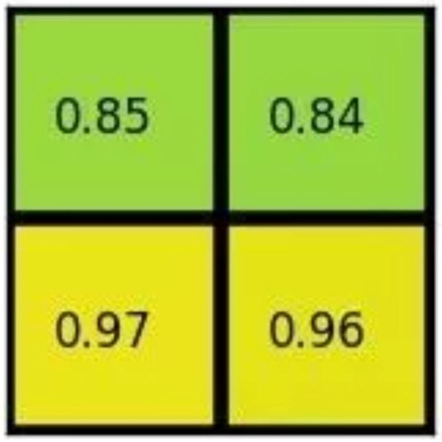
ROI pooling



0.88	0.44	0.14	0.16	0.37	0.77	0.96	0.27
0.19	0.45	0.57	0.16	0.63	0.29	0.71	0.70
0.66	0.26	0.82	0.64	0.54	0.73	0.59	0.26
0.85	0.34	0.76	0.84	0.29	0.75	0.62	0.25
0.32	0.74	0.21	0.39	0.34	0.03	0.33	0.48
0.20	0.14	0.16	0.13	0.73	0.65	0.96	0.32
0.19	0.69	0.09	0.86	0.88	0.07	0.01	0.48
0.83	0.24	0.97	0.04	0.24	0.35	0.50	0.91

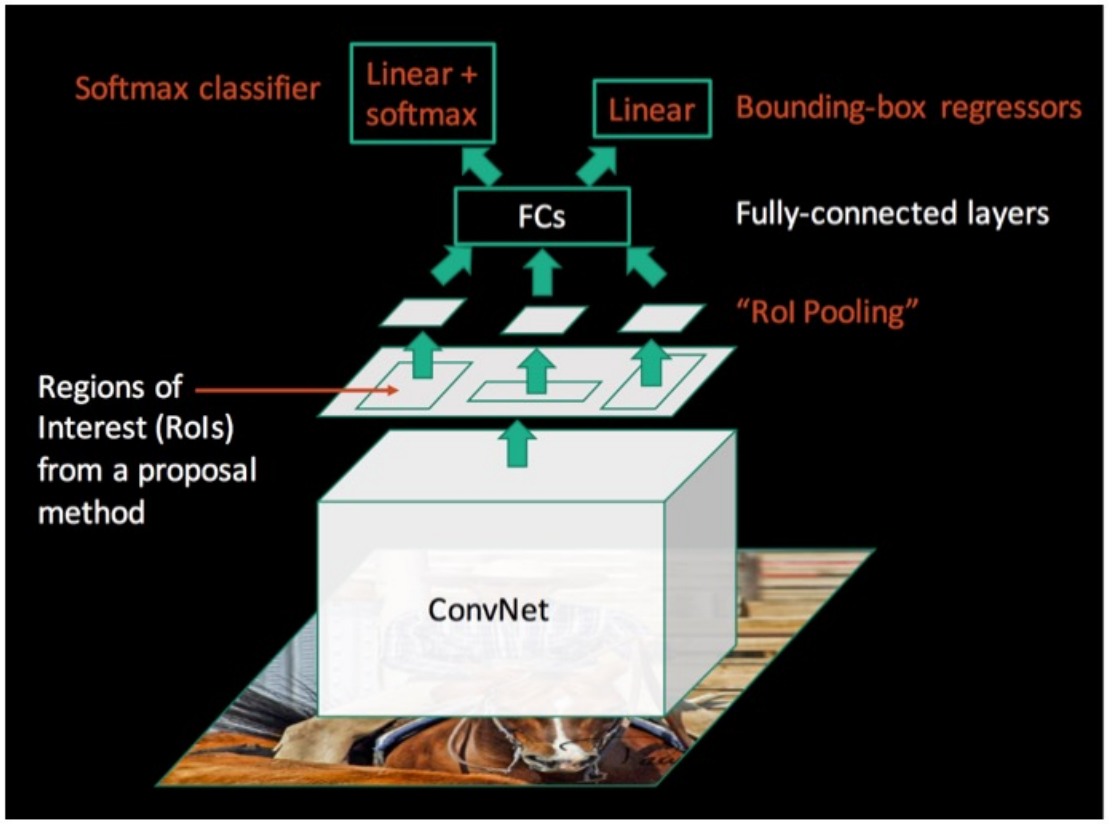


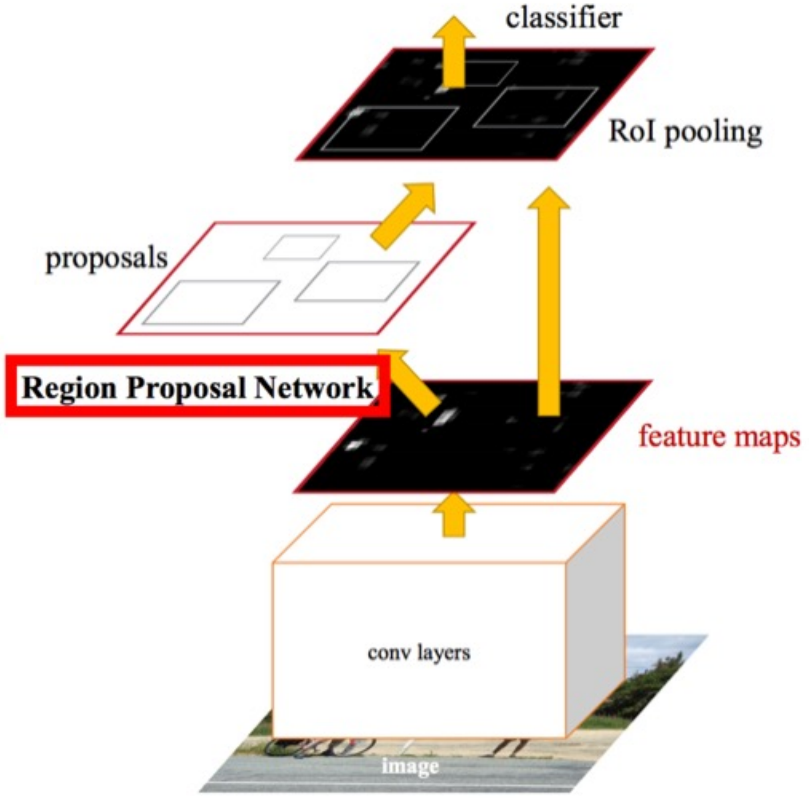
0.88	0.44	0.14	0.16	0.37	0.77	0.96	0.27
0.19	0.45	0.57	0.16	0.63	0.29	0.71	0.70
0.66	0.26	0.82	0.64	0.54	0.73	0.59	0.26
0.85	0.34	0.76	0.84	0.29	0.75	0.62	0.25
0.32	0.74	0.21	0.39	0.34	0.03	0.33	0.48
0.20	0.14	0.16	0.13	0.73	0.65	0.96	0.32
0.19	0.69	0.09	0.86	0.88	0.07	0.01	0.48
0.83	0.24	0.97	0.04	0.24	0.35	0.50	0.91



0.85	0.84
0.97	0.96

Faster R-CNN





Ren, He, Girshick and Sun 2016 Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

More readings

- R-CNN, fast RCNN and faster RCNN
- https://analyticsindiamag.com/r-cnn-vs-fast-r-cnn-vs-faster-r-cnn-a-comparative-guide/
- Object detection code example
- https://www.tensorflow.org/hub/tutorials/object_detection
- https://github.com/ultralytics/yolov5
- YOLO implementation
- https://docs.ultralytics.com
- https://www.analyticsvidhya.com/blog/2018/12/practical-guide-object-detection-yolo-framewor-python/