# Yolo V3 using openCV_ object detection :

+V1 :

· yolov3 → trained on the coco dataset

· the loaded network (yolov3 [architecture + weights]) only accept blob format.

· Darknet : NN framework for training, testing computer vision models

# object detection → looking in an image, try to find all of the objects, put bounding boxes around them and say what's these objects are.

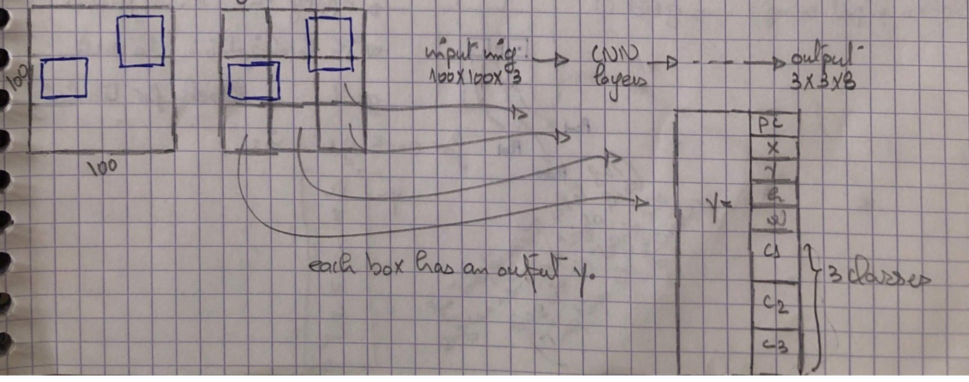
· in past, object detection sys will take an image, split it into bunch of regions, run a classifier on each batch, and high scales for that classifier would be considered detections. but this involves running a classifier thousand of times over an image.

· Non-maximal Suppression (NMS) → yolo uses NMS to only keeps the best bounding box.

· Anchor boxes → allow one grid to detect multiple objects.

· Prior detection systems apply the model to an image at multiple locations and scales, multiple times. high scoring are considered detections.

· Yolo uses a totally different approach. it applies a single NN to the full image, this network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.



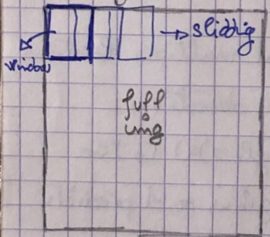
- classification with localization: → training set contains not just the object class label but also (x, y, h, w) bounding box parameters, so the output will contain the class labell + the object location.

(for the full img)!

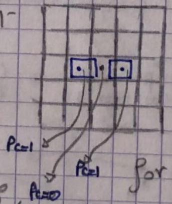
$$y = \begin{bmatrix} P_c \\ x \\ y \\ h \\ w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \qquad \hat{y} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \qquad loss = (y_1-\hat{y}_1)^2 + \dots + (y_8-\hat{y}_8)^2$$

- training a convNet car / not car } bounded → use this model in sliding windows detection.

- sliding window detection: picking a window size □ and the input this window to your model (convNet) (car/not car). slide this window by shifting little bit and pass to convNet. Go trough all the image.
Repeat the operation using different window size.
at the end there will be a window which can bounded our object. ⟹ computational cost. X



- Yolo: input image → grid it → apply the "classification with localization" to each grid. → labels for training



(Yolo takes the midpoint of each object and then assigns the object to the grid containing the midpoint)

for each grid is:

$$y = \begin{bmatrix} P_c \\ x \\ y \\ h \\ w \\ c_1 \\ c_2 \end{bmatrix}$$

- one of the problems of object Detection is that your algorithm may find multiple detections of the same object, so rather than detecting an object just once it might generate multiple bounded boxes around one object → Solution: Non-max suppression.