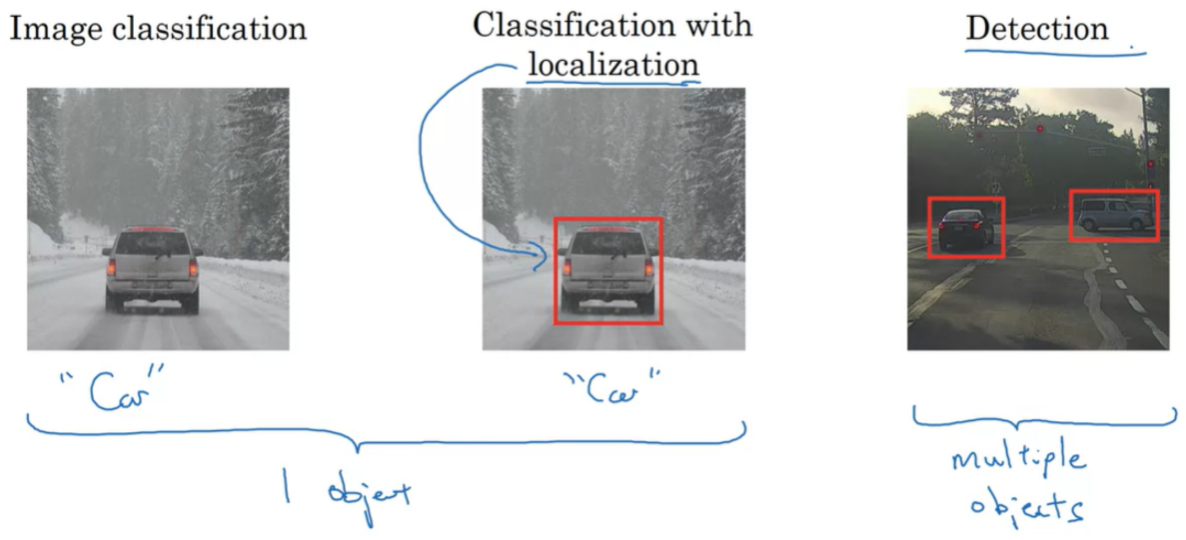
dl.ai_Detection Algorithm

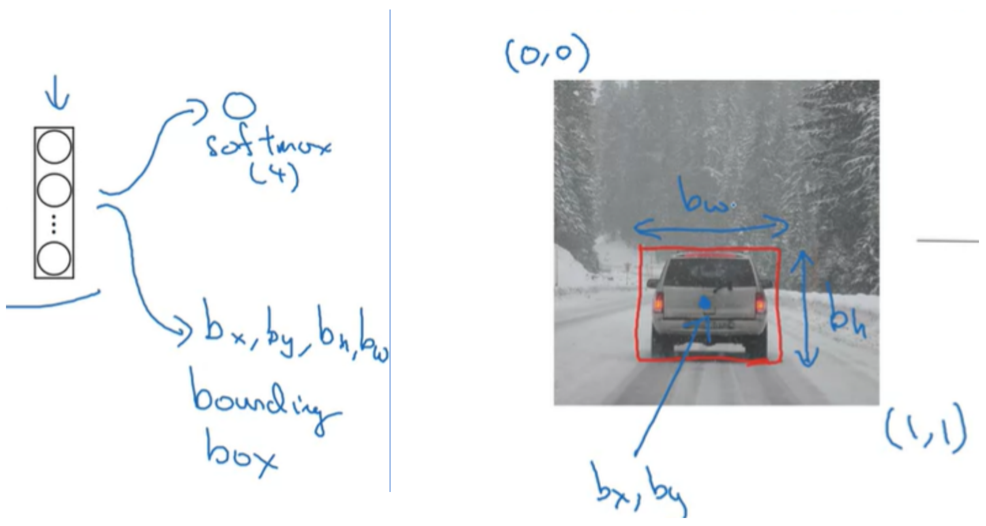
In image classification we are simply trying to classify a image into one of the given targets,in localization we will be trying to classify it and then put a box around the part of the image containing the object(unfortunately we will ultimately consider the full image for classification).The input image generally contains one single object at the center.

In detection we will be trying to detect multiple objects at once. The test image contains multiple objects.

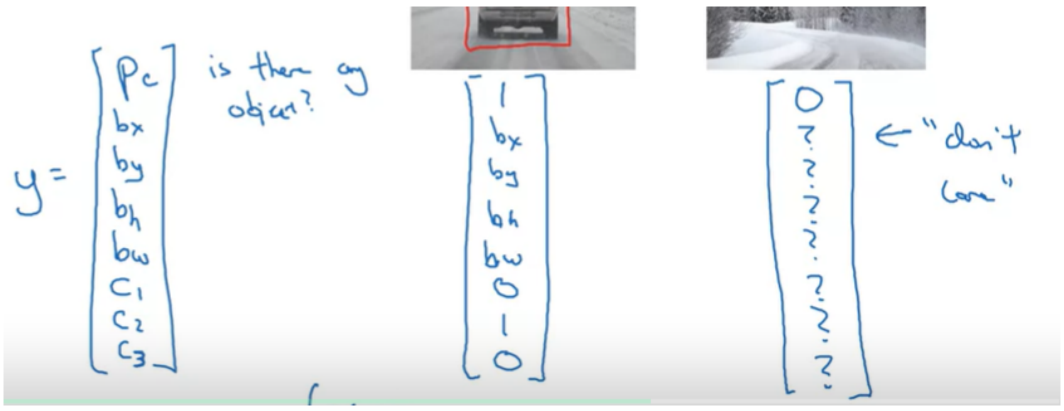


a) Classification → Object Localization

- We will try to find the boundary box just by cnn regression.
- Let say we are only trying to classify the image into 1 of 4 targets, then the output that we get from our convnet will be a vector with 4 softmax output and 4 dimensions to define our boundary box. The dimensions will be 2 coordinates of its center from the top left corner, breadth and height.



- One of our targets will a background which basically denotes the absence of the other three objects. This will be our first element in our output vector. It will be followed by 4 dimensions of the boundary box and three soft-max values for three objects.
- If an object is present Pc would be 1 and the rest of the elements the relevant stuff. If Pc is zero we don't care about the rest of the elements.



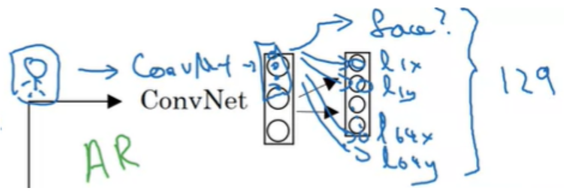
$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$
is there any object?
$\begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$
$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$
← "don't care"

- Our loss function will include all 8 individual loses if Pc(actual)=1 and just Pc if Pc(actual)=0

$$\mathcal{L}(\hat{y},y) = \begin{cases} (\hat{y}_1-y_1)^2+(\hat{y}_2-y_2)^2 \\ +\dots+(\hat{y}_8-y_8)^2 & \text{if } y_1=1 \\ (\hat{y}_1-y_1)^2 & \text{if } y_1=0 \end{cases}$$

b)Object Localization → Landmark detection

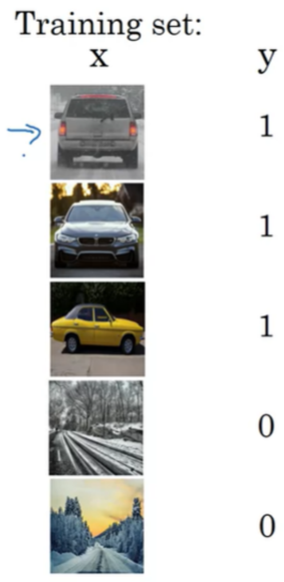
- In landmark detection instead of putting a box over an object will be trying to detect landmark point or points on a image.
- Lets say we want to first classify a image as pen or pencil and then detect the tip of the object from the image.
- We will simply train the image with our output vector looking having 5 elements. The presence of an object, two coordinates to denote the position of the tip in the image and two soft-max outputs.
- A complex use of this will be to detect landmark point across the face for ar filters(without classification)



$$\begin{aligned} &l_{1x},\ l_{1y}, \\ &l_{2x},\ l_{2y}, \\ &l_{3x},\ l_{3y}, \\ &l_{4x},\ l_{4y} \\ &\ \vdots \\ &l_{64},\ l_{64y} \end{aligned}$$

c)Object Detection

- Object detection uses a totally different principle from object localization. The training images will be cropped images of the objects we are trying to detect this pin image in our haystack image.

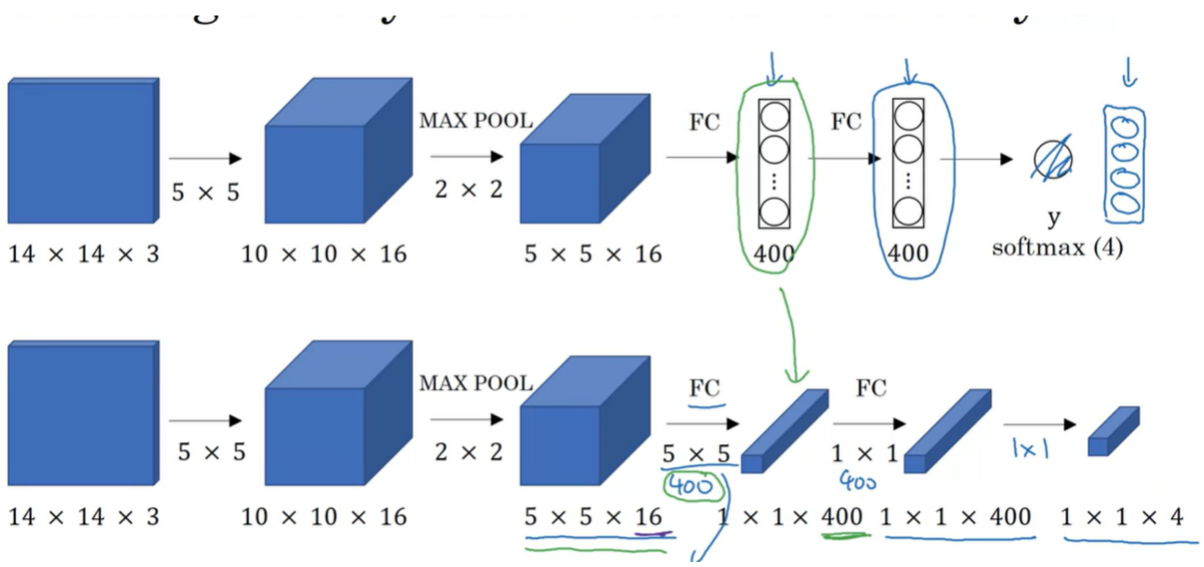


- In sliding window we will be taking a window or a box and we will be only using the part of image inside the box for classification we will move this window across the image.

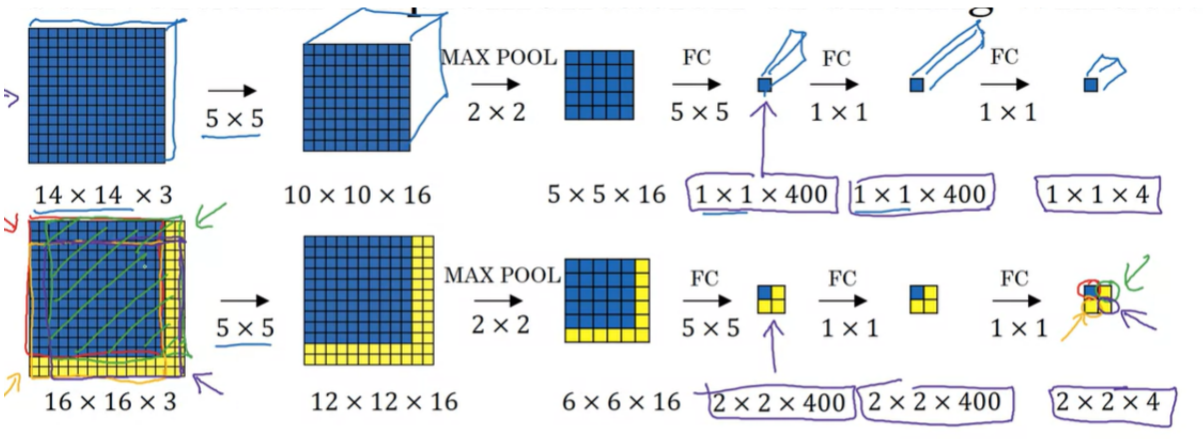
- We will start a reasonable size of box and end at another reasonable value.
- The computational cost of sliding window is very high because the stride associated with every shift must be small to ensure proper detection.

c.1)Convolutional implementation of OD

- We need to first convert our F.C (vector) layers to (F.C) convolutional layers.
- Let say we have a 5×5×16 convolutional layer if we apply a single 5×5×16 filter on it we will get 1×1×1 convolutional layer in which the single element will depend on all 5×5×16 elements in the previous layer(self filtering sort of stuff).



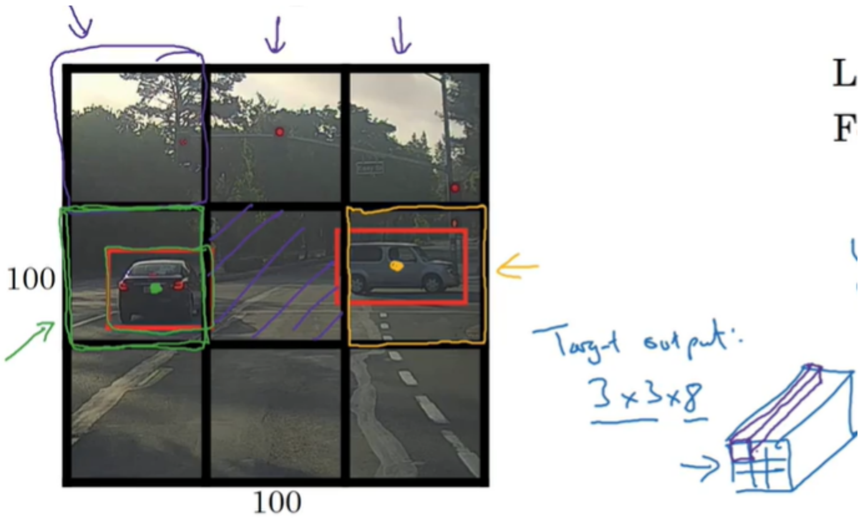
- Here we use 400 filters to get 1×1×400 CI similar to the 1×400 vector, then we use 400: 1×1×400 filter on it to get another 1×1×400 cl we finally use 4: 1×1×400 and softmax.
- Lets assume that 14×14×3 is the size of our window, And after applying all the above filters we get a 1×1×4 CI as our output.
- What will happen if we will apply the same sequence of filters on a 16×16×3 (16=14+2) image which has our 14×14×3 image in its top left corner.
- On applying a 5×5 filter we will get a 12×12 CI (12=10+2) with our 10×10 CI in its top left → On applying the max pooling layer it becomes 6×6(6=5+2/2) CI with our 5×5 in its top left → On applying 5×5 we get 2×2 CI (2=1+1) → etc
- Finally we will get a 2×2×4 CI with each slice corresponding to result of a part of the image, the result we would have got if we have used a 14×14 sliding window with stride 2,the stride 2 appeared because of the maxpooling layer. the Final CI would have been 3×3 if not for the max pooling layer.



- Convolution helps us to paralize all the operations for a given size of the sliding window at the same time. It doesnt solve the issue associated with multiple window sizes,rectangular boundary boxes still.

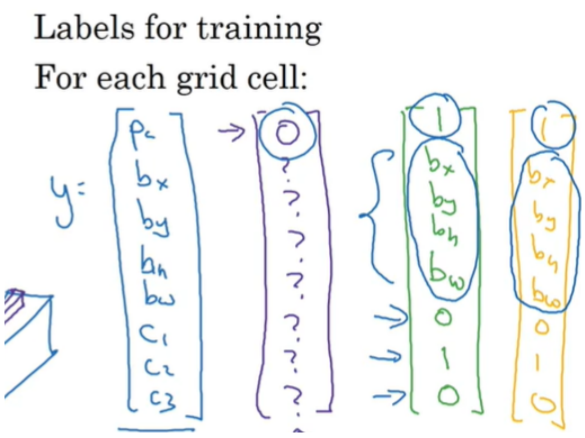
c.2)YOLO ALGORITHM

- In yolo we will be dividing the image into grids and try to predict the boundary of the object just like we could do in Object localization.
- In real life we generally use 19×19 grids but here we will be using 3×3 grids



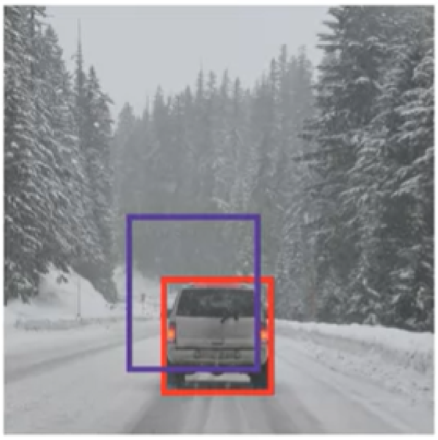
- We will be using convolutional implementation of Fc to get our final output as a 3d matrix. We can that YOLO is like a sliding window where stride is the size of the window and we are not just trying to classify but also put a boundary around the object inside the grid.**Sliding window → YOLO is anologous to Object classification → Object Localization.**

- Any object detected will be mapped to the grid containing the center and the height and length of the boundary can be greater than the dimensions of the grid they are mapped to.

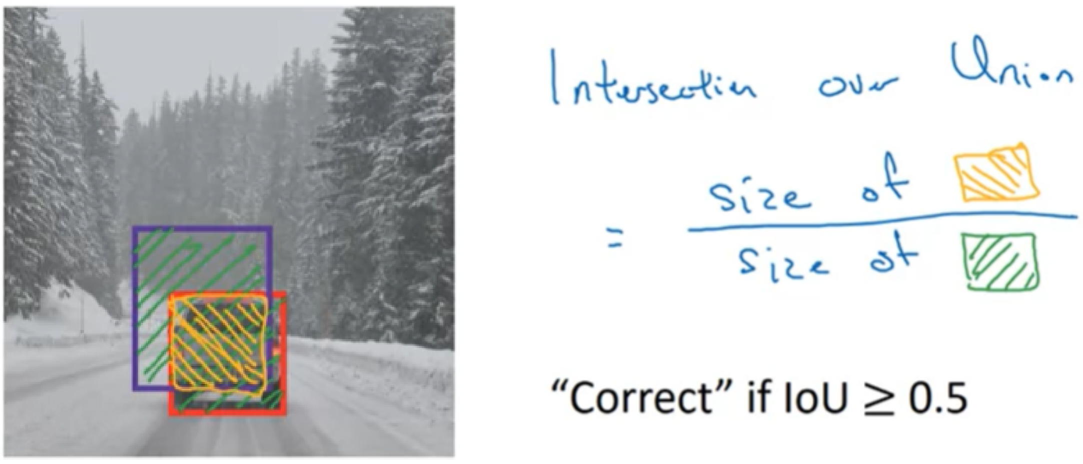


- Only the green and the yellow grids have an object detected rest 8 have Pc=0
- Because of the convolutional implementation of Yolo it runs even in real time detection
- Bx ,By ,Bh ,Bw are all relative to the dimensions of the grid ,Bx and By must lie in between 0 and 1.
- There are still issues which could be addressed like
  a) what happens if the model detects a center of an object at multiple points and put multiple boundaries for same object.
  b) if you try to remove these extra boundaries just based on confidence how will you differentiate two boundaries belonging to different objects. We cant simply take the first 2 boundaries with the highest confidence.
  c)what happens if two object have center in the same grid

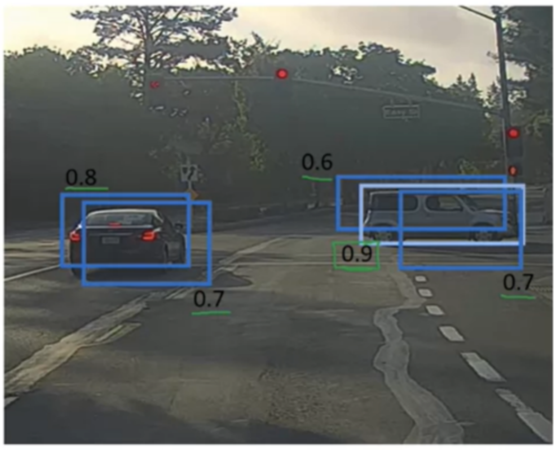
Non Max Suppression



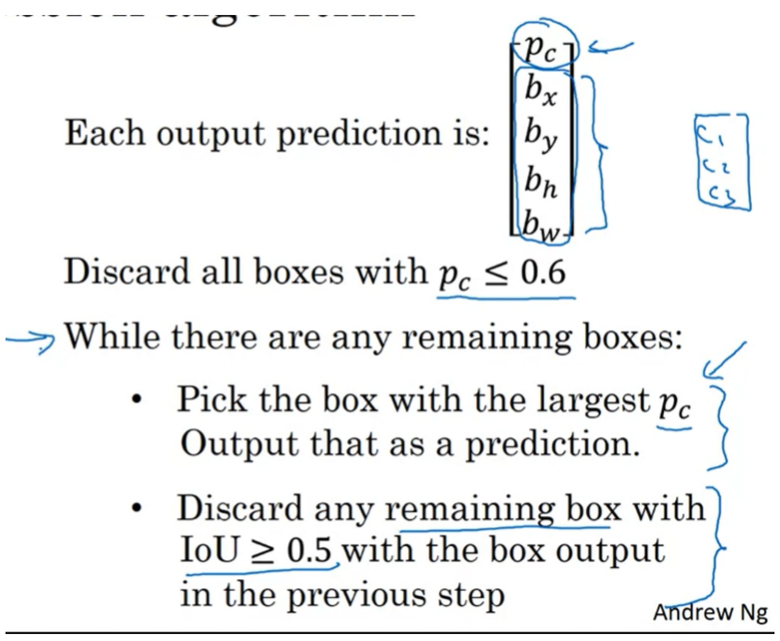
The red box is the ground truth and the purple box is the boundary predicted by the model. How to find the closeness of the prediction we use **Intersection over union IOU**



- we won't be using iou to check wether a boundary is correct(where we need high IOU) or not instead we would be using it remove duplicates(where we dont need high IOU).



- In the above image Each boundary has a confidence associated with it which is simply the Pc or the probability of finding that specific boundary(Pc is not related to IOU).
- The algorithm will first take the boundary with highest confidence across the image (it can't differentiate between the left car and right car) which happens to be the left car. Then we take all the boundaries which has IOU above a threshold say 0.5 and remove them
- We will be then go to the next highest confidence boundary and repeat the process
- We can optimize this process by removing all the boxes with Pc <0.6 or something (which can't be a proper boundary) beforehand



Each output prediction is: $\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$ $\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$

Discard all boxes with $p_c \leq 0.6$

While there are any remaining boxes:

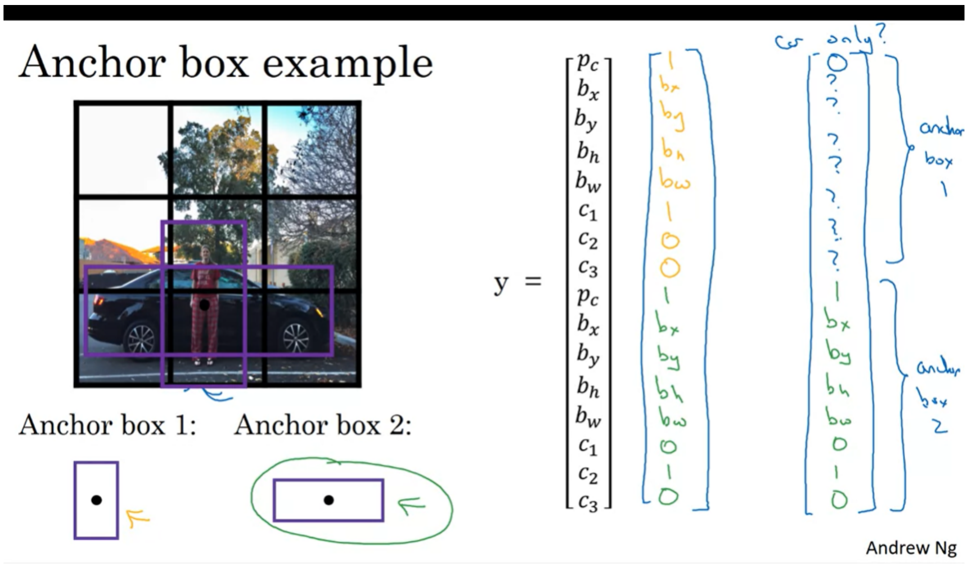
- Pick the box with the largest p_c Output that as a prediction.
- Discard any remaining box with IoU ≥ 0.5 with the box output in the previous step

Andrew Ng

- For better results we would need to do Yolo multiple times for each class separately.

Anchor Box

- How can we differentiate two overlaping objects with same center. We can make double the length of our output from each gridcell, we will output two values of Pc Bx By Bh Bw etc.
- How can we differentiate these two sets of values like when will Pc1 be 1 and Pc2 be 0 or viceversa. To solve this issue we use anchor boxes
- We basically have two predefined shapes anchor box1 and anchor box2 lets say they are a vertical box for people and horizontal box for car.
- We will use these two predefined anchor boxes to classify all boundaries as belonging to one of the two types based on which anchor box has highest IOU with the boundary box.



Anchor box example
Anchor box 1:
Anchor box 2:
y = [pc, bx, by, bh, bw, c1, c2, c3, pc, bx, by, bh, bw, c1, c2, c3]
1, bx, by, bh, bw, 1, 0, 0, 1, bx, by, bh, bw, 0, 1, 0
car only?
0, ?, ?, ?, ?, ?, ?, ?, 1, bx, by, bh, bw, 0, 1, 0
anchor box 1
anchor box 2
Andrew Ng

- The output will have dimension 3×3×2×8 because of 2 types of anchor boxes.
- This is the final algorithm we follow

- For each grid call, get 2 predicted bounding boxes.
- Get rid of low probability predictions.
- For each class (pedestrian, car, motorcycle) use non-max suppression to generate final predictions.