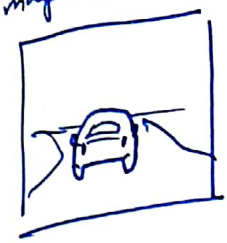
Week 3

Detection Algorithms

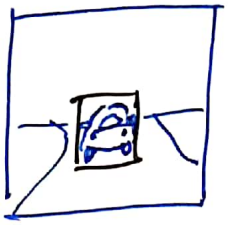
Object localization:

Image Classification



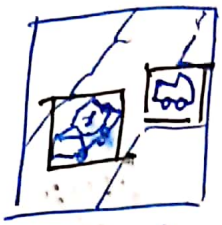
"car"

Classification with localization

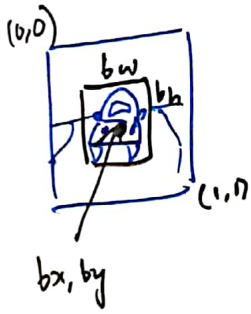


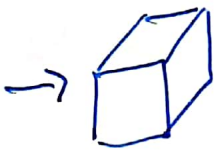
"car" and bounding box

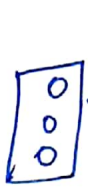
Detection



Multiple objects & bounding box (can even have multiple classes)







$\rightarrow$ 0

softmax (4)

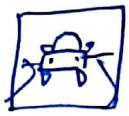
1. pedestrian
2. car
3. motorcycle
4. background

$\rightarrow$ b_x, b_y, b_h, b_w (bounding box)

Need to output b_x, b_y, b_h, b_w, class label (1-4)

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

P_c ← is there any object?

If $x =$ 

$$\begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

If $x =$ 

$$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

← don't care (?)

Loss:

$$\mathcal{L}(\hat{y}, y) = (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 \ldots + (\hat{y}_8 - y_8)^2$$

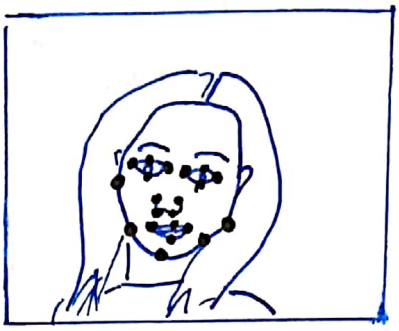
if $y_1 = 1$

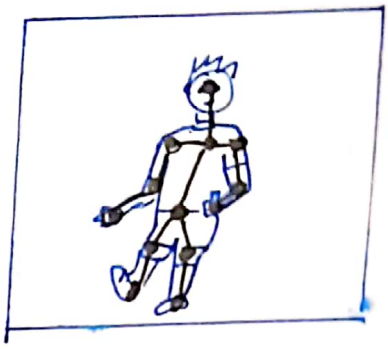
$$\mathcal{L}(\hat{y}, y) = (\hat{y}_1 - y_1)^2$$

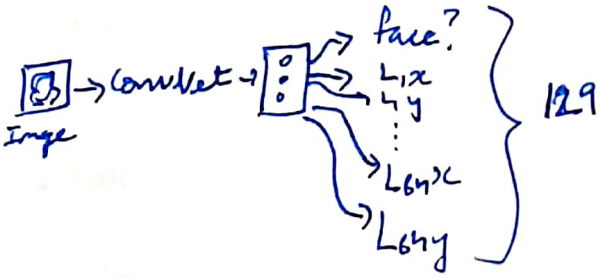
if $y_1 = 0$

Landmark detection:

Rather than finding a bounding box, we can find important coordinates (landmarks) in the image.







$l_1x, l_1y,$
$\vdots$
$l_{32}x, l_{32}y$

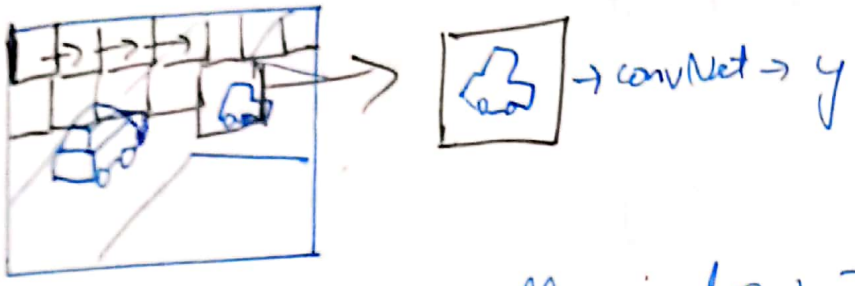
$l_1x, l_1y,$
$l_2x, l_2y,$
$l_3x, l_3y,$
$\vdots$
$l_{64}x, l_{64}y$

x, y coordinates of 64 landmarks

They use this in snapchat AR filters.

Object Detection:

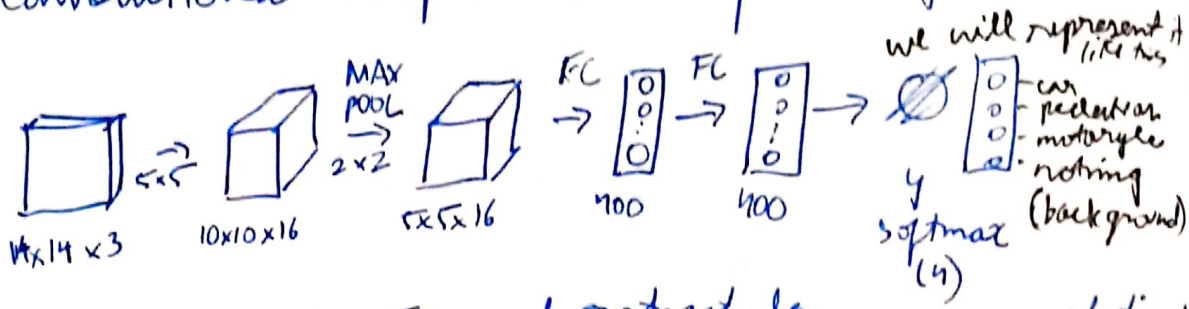
If we have a test image containing multiple objects (and classes), we use a sliding window to crop out smaller images that are then fed into the convNet.



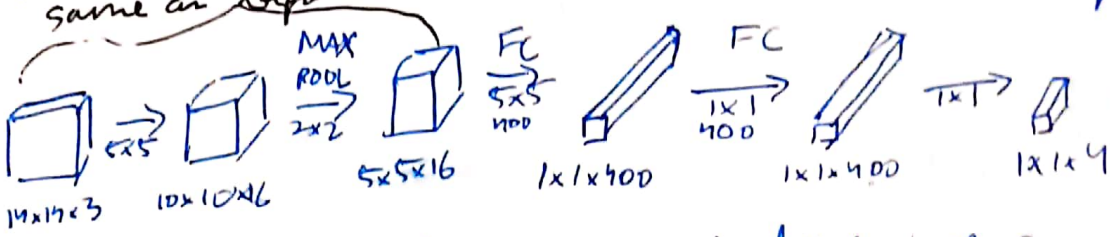
we start with a small window, then gradually increase it (maybe use 3 different sized windows) and we can use different stride to make sure we get the car in the picture.

However this is computationally very costly since we need to pass so many images into convNet. So we use a convolution implementation of Sliding Windows.

Convolutional Implementation of Sliding Window:

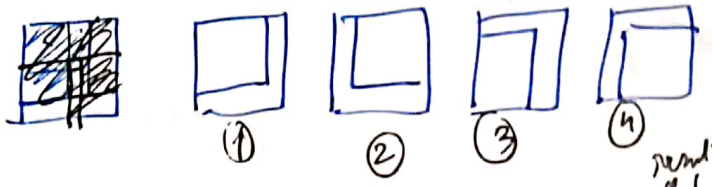


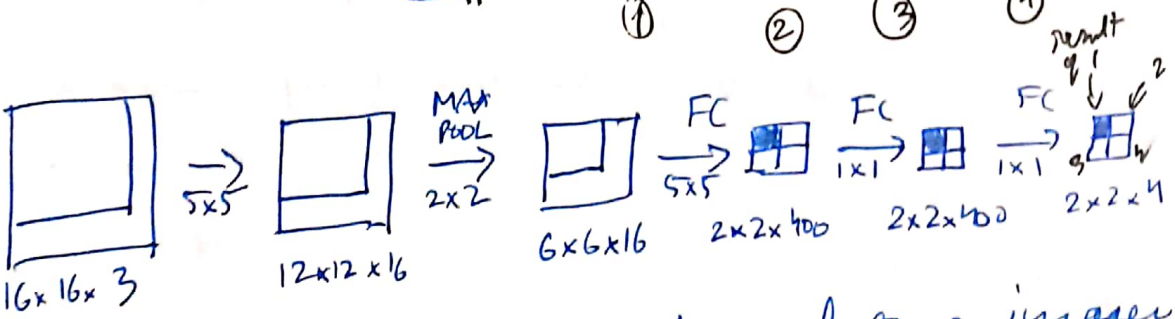
We can turn the FC and output layer into convolutional layer:



Rather than sliding a window, we can simultaneously find all the values of a sliding window using convolution in a single iteration.

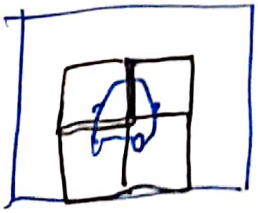
suppose our image is 16x16x3 and our window is 14x14x3 with a stride of (2,2)





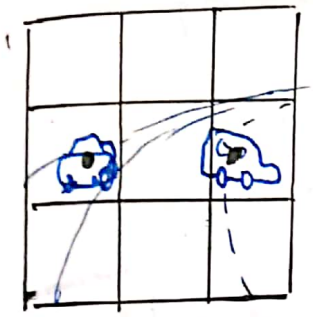
We can even do this for larger images with more strides.

Disadvantage: Positioning of the bounding boxes isn't going to be accurate. We'll see how to fix this.



Bounding Box Predictions:

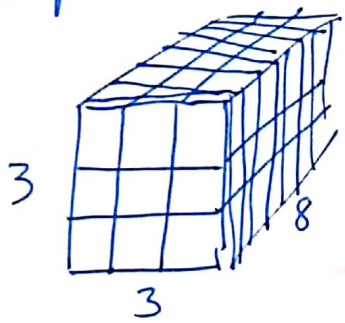
YOLO (You Only Look Once) algorithm:
We divide the image into ~~squares~~ grids and and pass them through ConvNet to get an output matrix



Labels for training for each grid cell

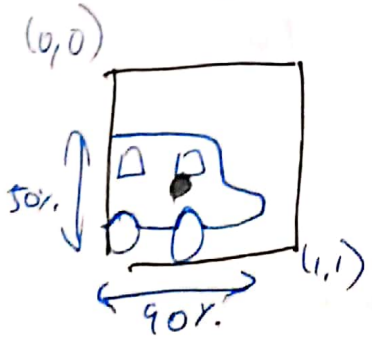
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad \text{For car} \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{For not car} \begin{bmatrix} 0 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

We get a matrix combining all these vectors



$\underbrace{3 \times 3}_{\text{image}} \times \underset{\text{8 output values}}{8}$

It finds the midpoint of the car and then assigns bounding box. - the midpoint is assigned to only one grid



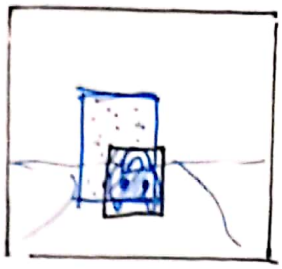
$b_x = 0.4$, $b_y = 0.3$ } between 0 and 1

$b_h = 0.5$, $b_w = 0.9$ } could be greater than 1 - if car is in more than a single grid

Typically smaller grids are used so 2 cars don't come in the same grid (19x19x8)

Intersection over Union (IOU)

It is a method used to evaluate object localisation



- the bounding box we are using
- the perfect bounding box
- the intersection of the 2 boxes

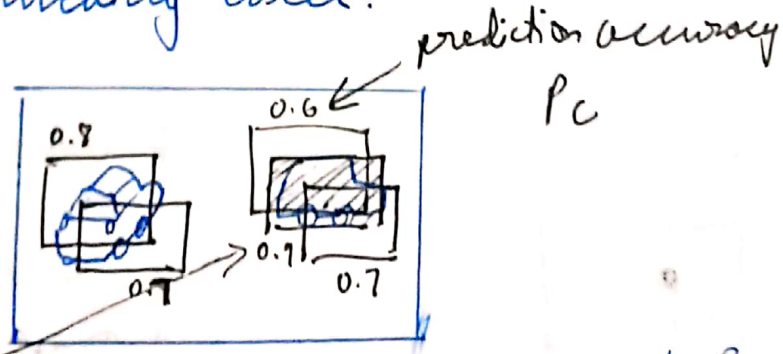
$$IOU = \frac{\text{size of } \text{inter}(A,B) \text{ - intersection}}{\text{size of } \text{ - our bounding box} \quad \text{ - } A+B - \text{inter}(A,B)}$$

The bounding box is correct if $IOU \geq 0.5$

ie. IOU is the measure of overlap between 2 bounding boxes.

Non-max Suppression ~~Example~~

Sometimes a single object can get recognised multiple times resulting in multiple bounding boxes.



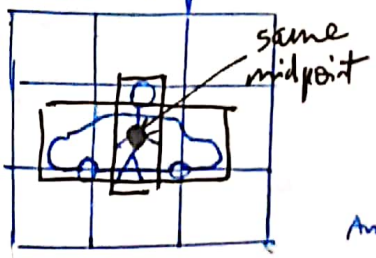
Here we take the box with the highest Pc (ex. 0.9). Then we find IOU for all the other boxes wrt to this box. We remove any box with $IOU \geq 0.5$

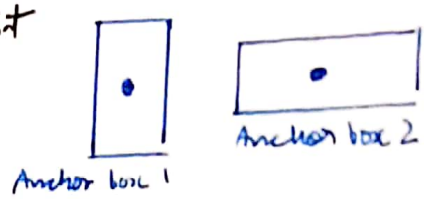
Complete Non-max suppression algorithm:

1. Discard all boxes with $p_c \leq 0.6$
2. While there are any remaining boxes:
   3. Pick the box with the largest p_c. Output that as a prediction.
   4. Discard any remaining box with IoU ≥ 0.5 with the box output in the previous step

Anchor boxes

One of the problems is a grid can only have 1 midpoint. But what if 2 images end up sharing the same midpoints? Then we use anchor boxes





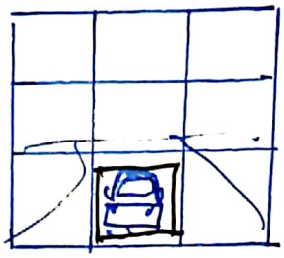
and $y = \begin{bmatrix} p_c \\ \vdots \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ \vdots \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$ Anchor box 1 / Anchor box 2

So previously, we would assign an object to a grid cell containing that object's midpoint. But with 2 anchor boxes, we assign each object to a grid cell that contains its midpoint as well as the anchor box with the highest IOU (the man's box has a higher IOU with anchor box 1)

However it's very rare for objects to share a midpoint since we use small grids (19x19x8)

YOLO Algorithm summary

1. The input image is broken into grids (usually 19x19, but 3x3 for this example)



2. each grid is checked to see if it contain the midpoint of the car (class = 2)

For no class detected

For class detected

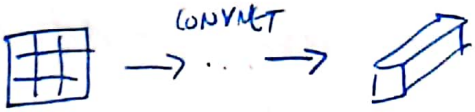
$$y = \begin{bmatrix} P_c \\ b_x \\ \vdots \\ c_3 \\ P_c \\ \vdots \\ c_3 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ ? \\ \vdots \\ ? \\ 0 \\ ? \\ \vdots \\ ? \end{bmatrix} \qquad \begin{bmatrix} 0 \\ ? \\ \vdots \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

← since anchor box 2 fits better

3. Y is 3x3 x 16

2 x 8

anchor ← 5 + # classes



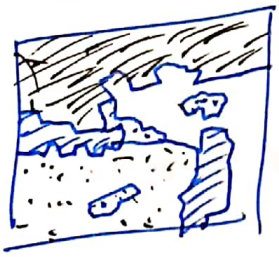
3x3x16

4. Run Non-max suppression seperately for each class

Region Proposals (R-CNN)

Sometimes YOLO ~~doesn't~~ predicts cars where there are no cars (not than specific I just took car as example) so regional proposals were created

1. Perform segmentation algorithm



These gives different coloured regions based on the image

2. Take ~2000 blobs and run CONVNET there

R-CNN: Propose regions. Classify proposed regions one at a time. Output label + bounding box

Fast R-CNN: Propose regions. Use convolution implementation of sliding windows to classify all the proposed regions.

Faster R-CNN - Use convolutional network to propose regions.