+ <u>Object Localization</u>

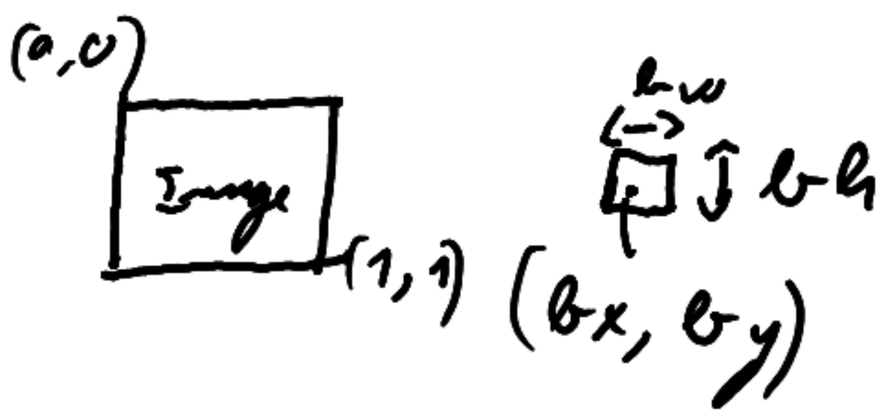
+ → Two sub problem, - Classification
- Classification with localization
- Detection

Image → ConvNet → $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ → Softmax.

• <u>Classification with localization</u>

Img → Conv Net → $\begin{bmatrix} 0 \\ 0 \end{bmatrix}_{(4)}$ → Softmax

If we wanted to localize the detected img then network will output (b_x, b_y, b_h, b_w) (bounding box)



So in our example ⇒ 4 outputs
1. Pedestrian
2. Car
3. Motorcycle
4. Background.

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

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

P_c → Probability of object.
b_x, b_y, b_h → Bounding box.
c_1, c_2, c_3 → class

Example for conv $\Rightarrow$ $\begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$ No object. $\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$ ← "don't care"

<u>Loss function</u>: $\mathcal{L}(\hat{y}, y) = (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \dots$

$\dots + (\hat{y}_8 - y_8)^2$ if $y_1 = 1$

$(\hat{y}_1 - y_1)^2$ if $y_1 = 0$

+ <u>Landmark Detection:</u>

If we don't want a box, just a point, we could use l_x, l_y.

For example we could have a variety of landmarks l_{eyex}, l_{eyey}

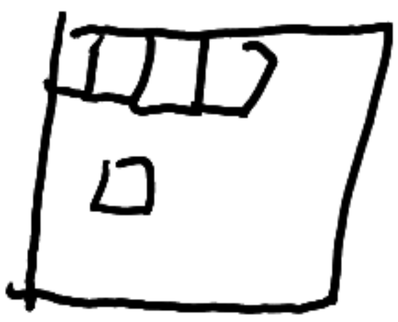
Image $\Rightarrow$ conv Net $\Rightarrow \begin{bmatrix} 0 \\ 0 \\ \vdots \end{bmatrix} \rightarrow \begin{bmatrix} \text{face} \\ \rightarrow l_{x_1} \dots l_{x_n} \\ l_{y_1} \dots l_{y_n} \end{bmatrix}$ 129 outputs.

+ <u>Object Detection</u>:

Using a conv net with sliding window algorithm

Training set => closely cropped.

· Once we have trained it, the sliding window algorithm, the image is segmented.



=> If there is a car a 1 is outputed

The computational cost is very high.

+ <u>Convolutional implementation of sliding windows</u>

Turning fully connected layers into conv layers.

AFC layer over $\begin{bmatrix} \circ \\ \circ \\ \circ \end{bmatrix}$ => $1 \times 1 \times 400$ on a conv.
400

Even in the final layer $1 \times 1 \times 4$

So to implement.

We crop -> We run through sequences not in sequence but in one go using a Conv net.

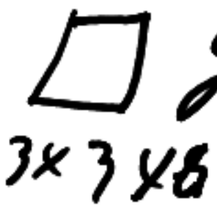
Main weakness => Bounding box will be inaccurate.

+ Bounding Box Predictions

YOLO => you only look once.

A label for each grid cell:

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad (3 \times 3 \times 8)$$

X  $\longrightarrow$ CNN $\rightarrow$ Max Pool .... $\rightarrow$  y.

100 x 100 x 3   3 x 3 x 8

+ Intersection Over Union

$$\frac{\text{Size of Intersection}}{\text{Size of Union}} \Rightarrow \text{correct if } IOU \geq 0,5.$$

+ Non-max Suppression

Some boxes will overlap.

Step. 1. Discard boxes with $P_c \leq 0,6$.

2. The remaining boxes:

- Pick the box with highest P_c => Prediction
- Discard any remaining box with $IOU \geq 0,5$

+ <u>Anchor Boxes:</u>

Predetermined box shapes. (for different anchors)

So instead of $y \Rightarrow \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$ $\Rightarrow$ Now $\Rightarrow y = \begin{bmatrix} \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \\ \begin{bmatrix} p_c \\ b_x \\ \vdots \end{bmatrix} \end{bmatrix}$ $\Rightarrow$ anchor 1, $\Rightarrow$ anchor 2.

$3 \times 3 \times 8$

grid cell, anchor box

$(3 \times 3 \times 16)$

<u>Summary</u>

Previously: Each object in training image is assigned to grid cell that contains that objects midpoint

With two anchor boxes: Each object in training image is assigned to grid cell that contains objects midpoint and anchor box for the grid cell with highest IoU

+ Semantic Segmentation with U-Net.

Rather than object detection, each pixel is labeled,

Used in self driving. or X-ray reading.

- U-Net (segmentation)

If looking for a car in an image, 0 will be not a car and 1 car.

:) segmentation map is generated.

One key step is to make the image bigger the bigger and width will increase the deeper we go into the U-Net.

+ Transpose Convolution:

Normal Convolution turns the image into a smaller output

A transpose conv does the opposite.

$(2 \times 2) \Rightarrow \xrightarrow[(3\times3)]{\text{Filter}}$ Filter is applied to the output. (4×4)

filter $f \times f$ (3×3)

Stride $s=2$

Padding $p=1$

+ U-Net Architecture

$h \times w \times 3 \Rightarrow$ U-Net $\Rightarrow h \times w \times$ number.