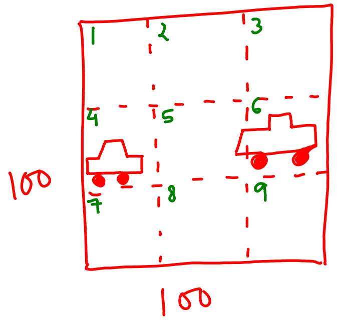
Output Accurate Bounding Boxes

- When using sliding window, it is not necessary that the Image would completely fit in Any sliding window
- Also, it is not necessary that the bounding box must be a square, a rectangle/ circle may better fit the Img

Solution - YoLo Algo :- You only Look once



(Input 100X100X3)

- Consider car detection problem
  4 class detection - car, bike, pedestrian, background

then for each grid cell (9 of them total, we have y)

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

For grid cell (1,2,3,5,7,8,9)

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

For grid cell (4,6)

$$y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_H \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Total volume of the output
= 3X3X8

$$100 \times 100 \times 3 \rightarrow \text{CNN} \rightarrow \text{Max Pool} \;\cdots\; \rightarrow \underset{y}{3 \times 3 \times 8}$$

Adv of Yolo

↳ It gives precise coordinates of bounding boxes

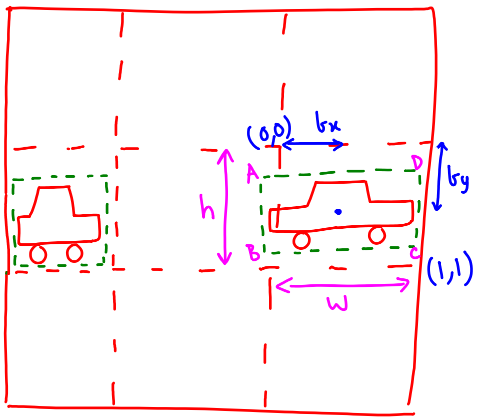
↳ Assumption: only 1 object per grid cell

↳ Actual grid size used in practice - 19X19
   ↳ Granularity ⇒ likelihood of 1 object/grid cell ↑

↳ Algo - look at the midpoint (bx, by) of the object
   & assign it to the appropriate grid
   - in grid 6, the car could have been put in grid 5 as well
   (But the mid point ⇒ grid 6)

- This Algo is also a conv. Implementation ⇒ Shared computation ⇒ fast
- Can be used in Real time object detection

Specifying the bounding boxes



Lets consider the RHS car

$$y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_H \\ b_W \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.4 \\ 0.5 \\ 0.7 \\ 0.9 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

How did we compute b_H, b_W?

$$b_H = \frac{\text{green line height}}{\text{height of grid cell}} = \frac{AB}{h}$$

$$b_W = \frac{\text{green line width}}{\text{width of grid cell}} = \frac{BC}{w}$$

∴ b_x, b_y always b/w 0 and 1

b_H and b_W → could be > 1

Always > 0