



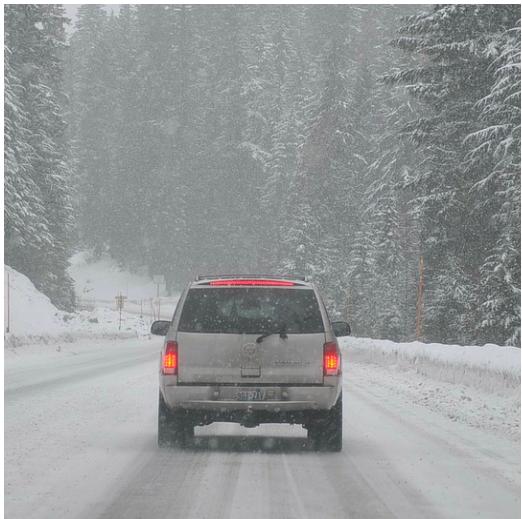
deeplearning.ai

Object Detection

Object localization

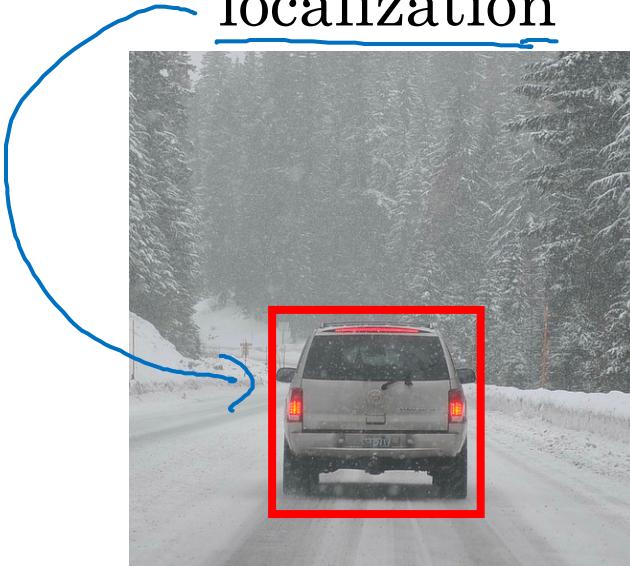
What are localization and detection?

Image classification



"Car"

Classification with
localization



"Car"

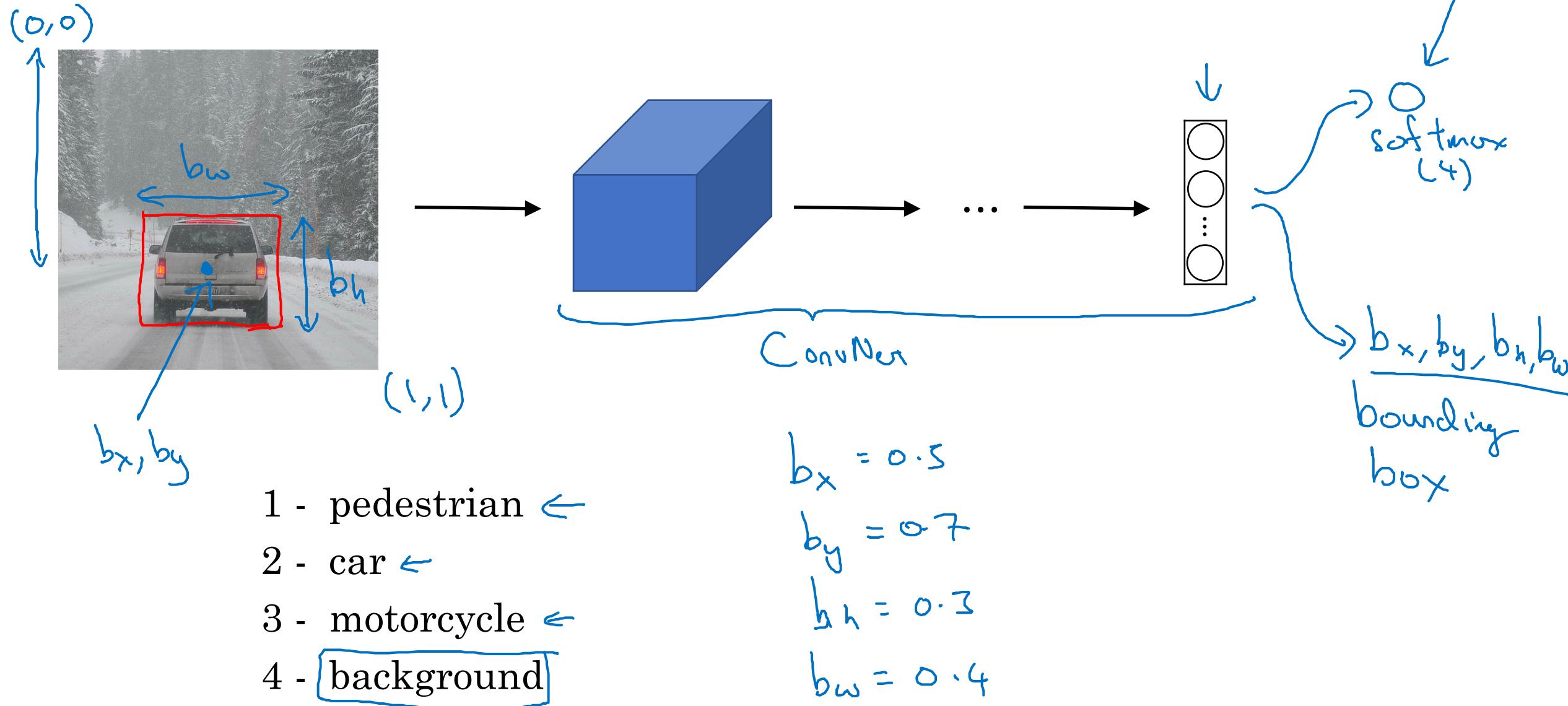
1 object

Detection



multiple
objects

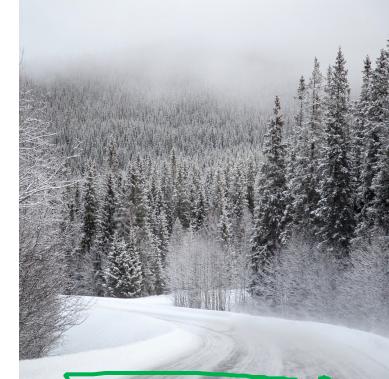
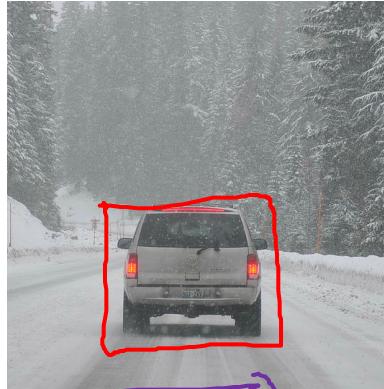
Classification with localization



Defining the target label y

- 1 - pedestrian
- 2 - car ←
- 3 - motorcycle
- 4 - background ←

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

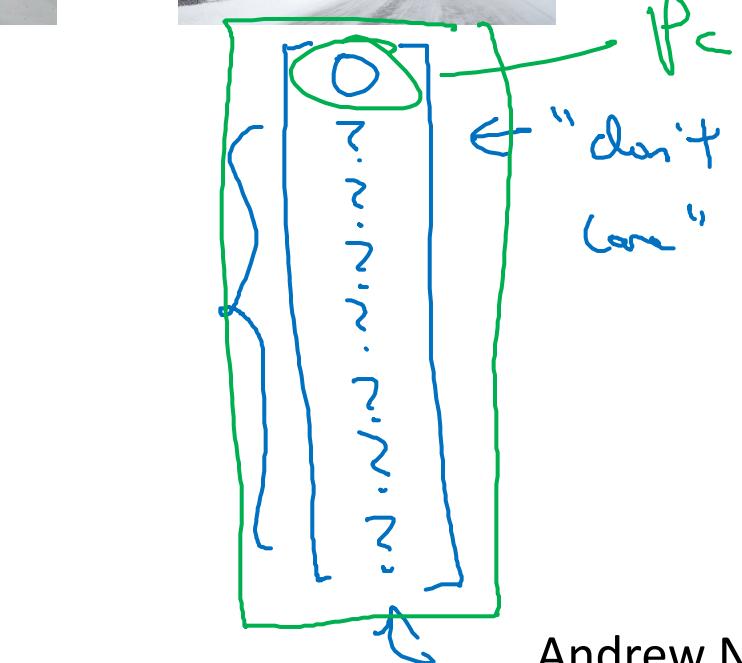


$x =$

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

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad \rightarrow \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad \left\{ \begin{array}{l} \text{is there any} \\ \text{object?} \end{array} \right.$$

(x, y)



Andrew Ng

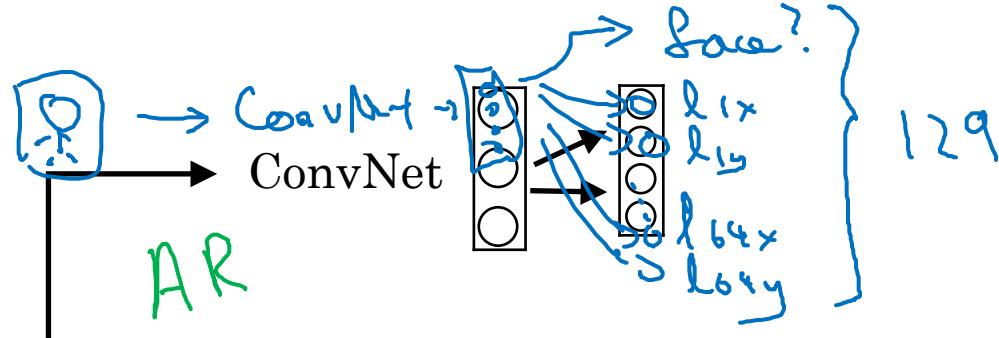


deeplearning.ai

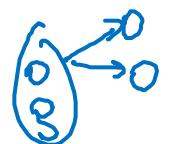
Object Detection

Landmark detection

Landmark detection



b_x, b_y, b_h, b_w



$l_{1x}, l_{1y},$
 $l_{2x}, l_{2y},$
 $l_{3x}, l_{3y},$
 l_{4x}, l_{4y}
:
 l_{64x}, l_{64y}

x, y

$l_{1x}, l_{1y},$
:
 l_{32x}, l_{32y}



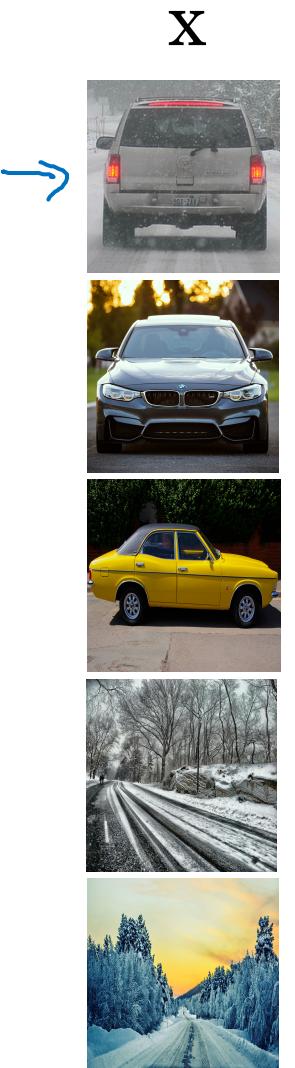
deeplearning.ai

Object Detection

Object detection

Car detection example

Training set:



y

1

1

1

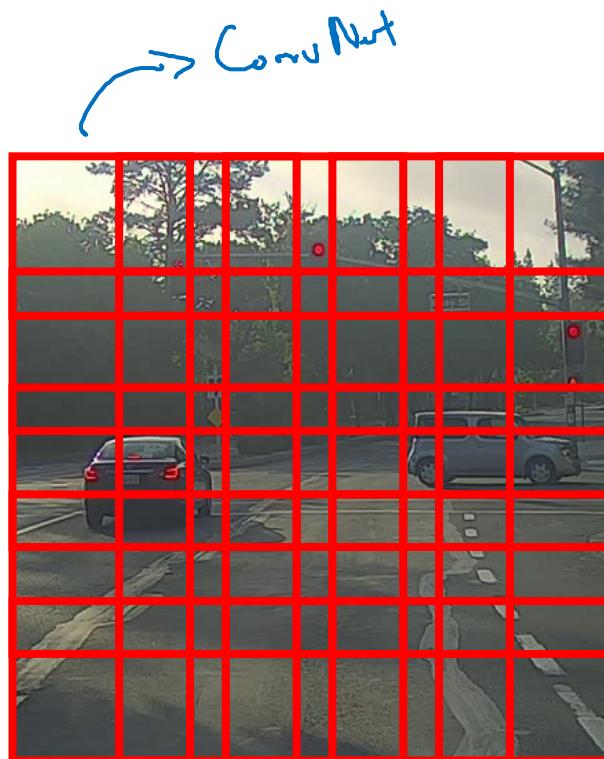
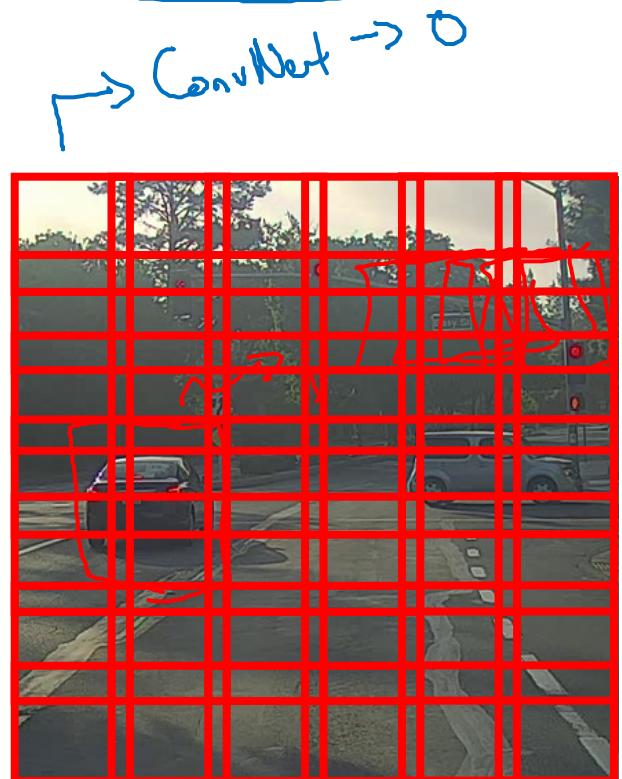
0

0



→ ConvNet → y

Sliding windows detection



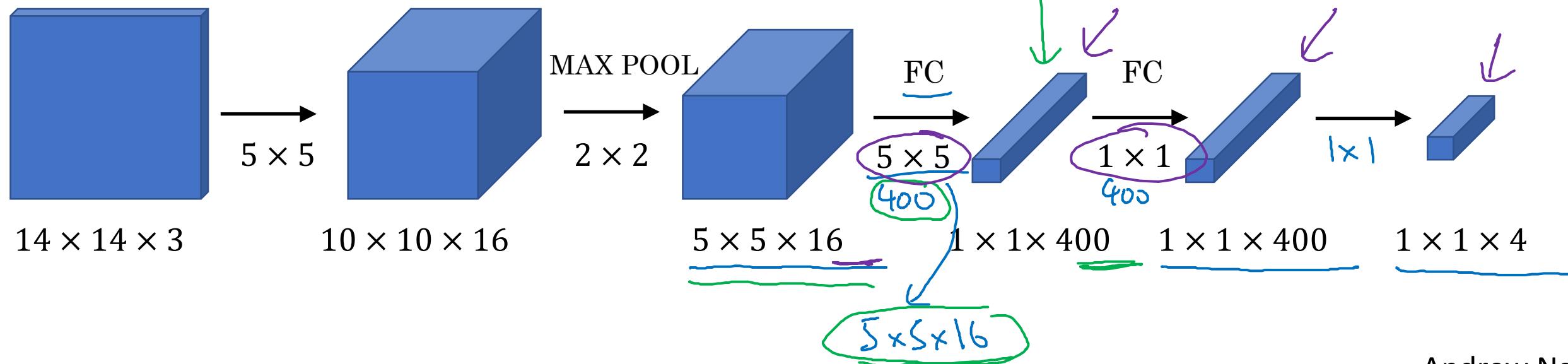
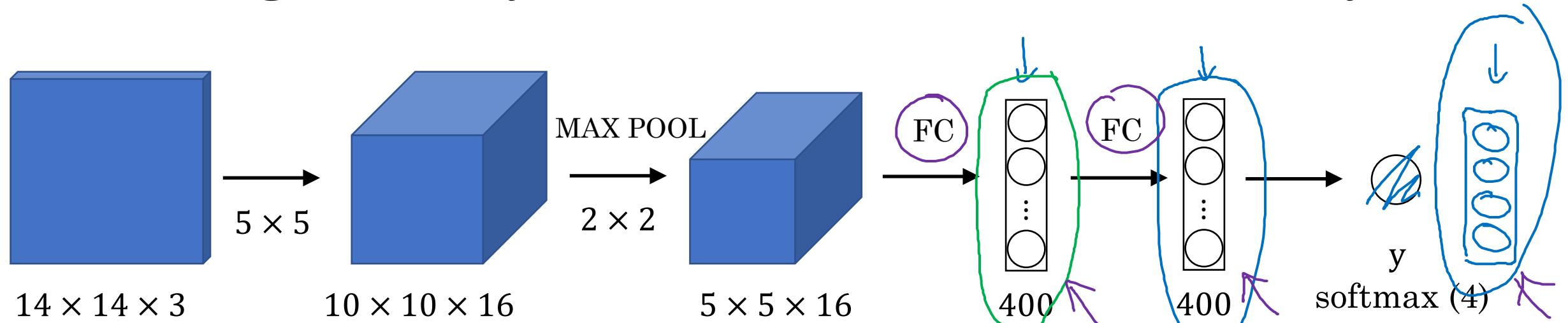


deeplearning.ai

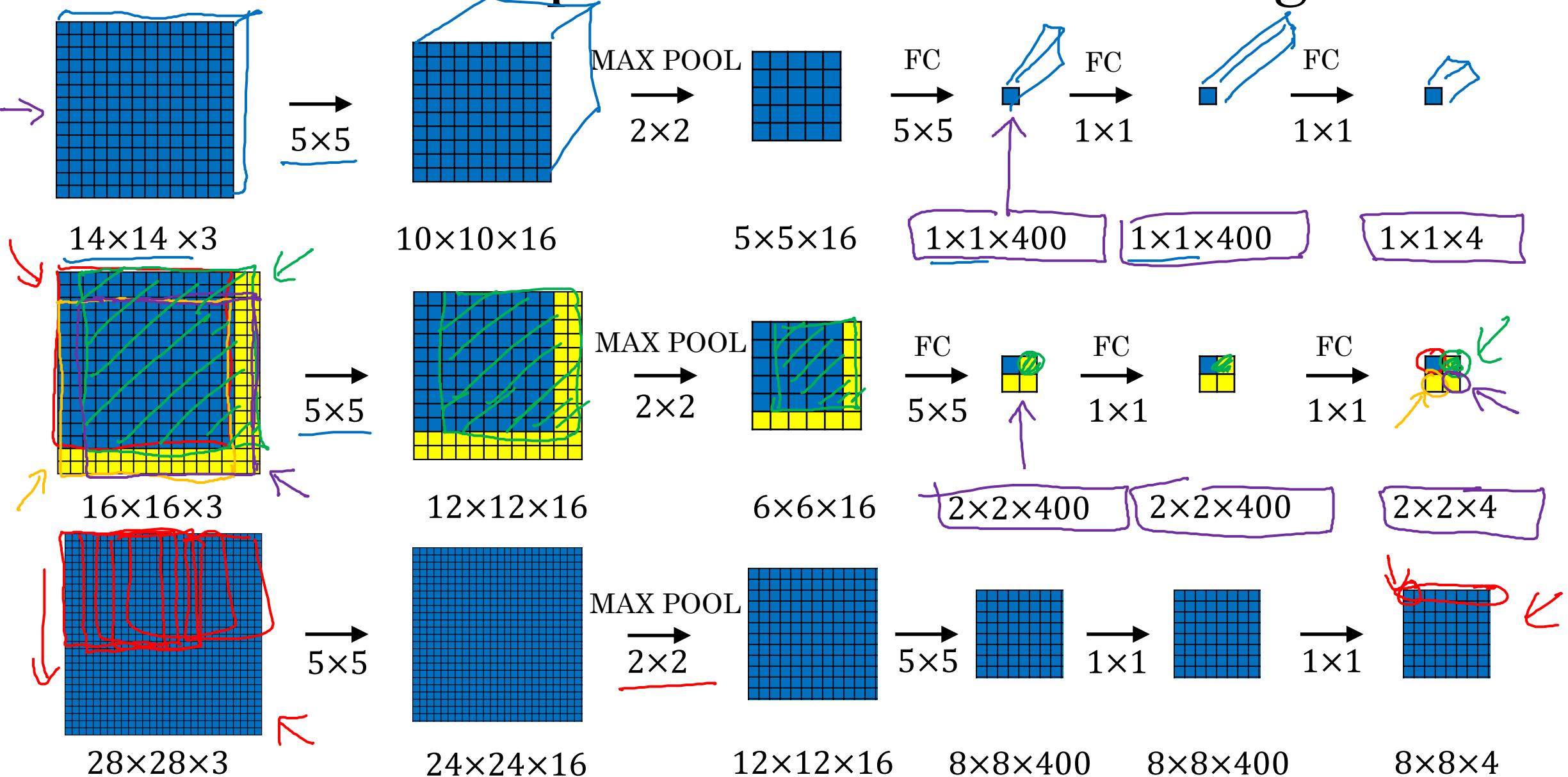
Object Detection

Convolutional implementation of sliding windows

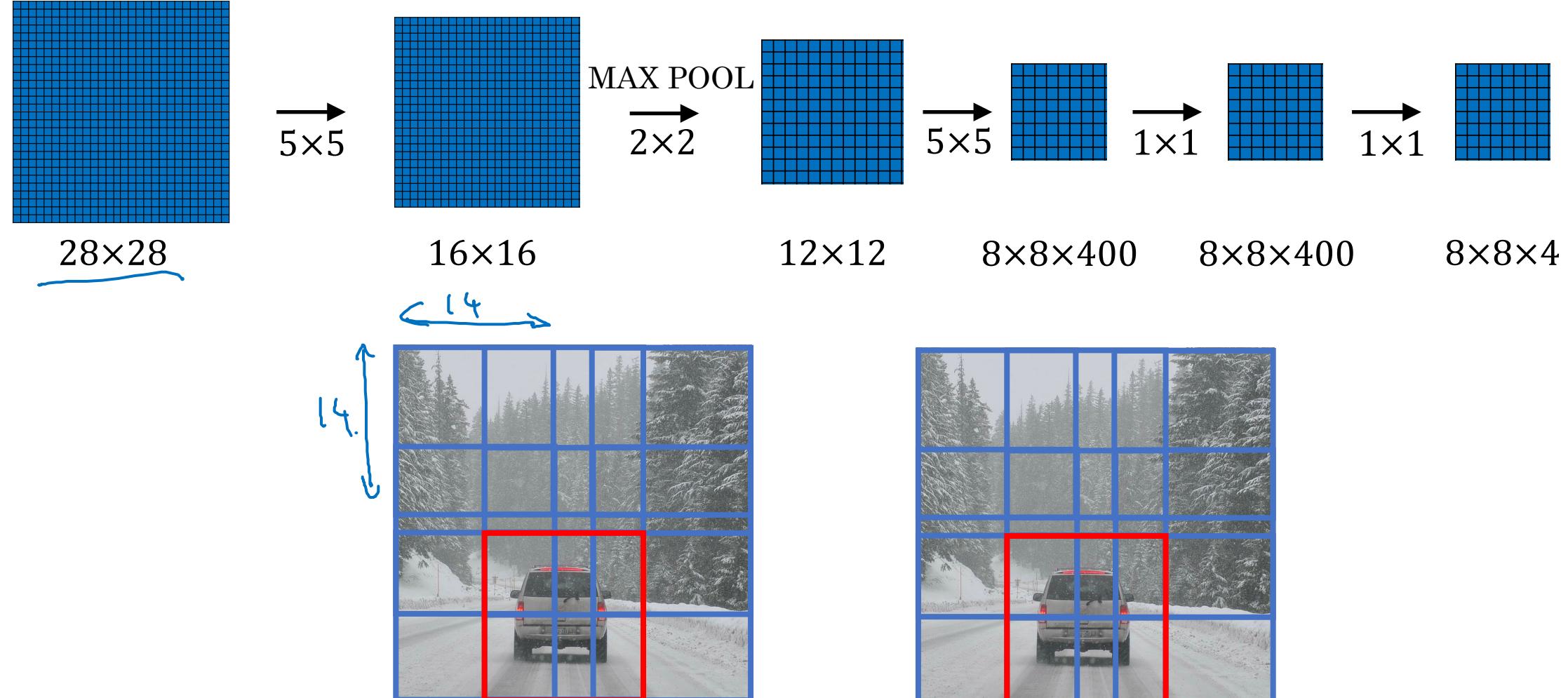
Turning FC layer into convolutional layers



Convolution implementation of sliding windows



Convolution implementation of sliding windows



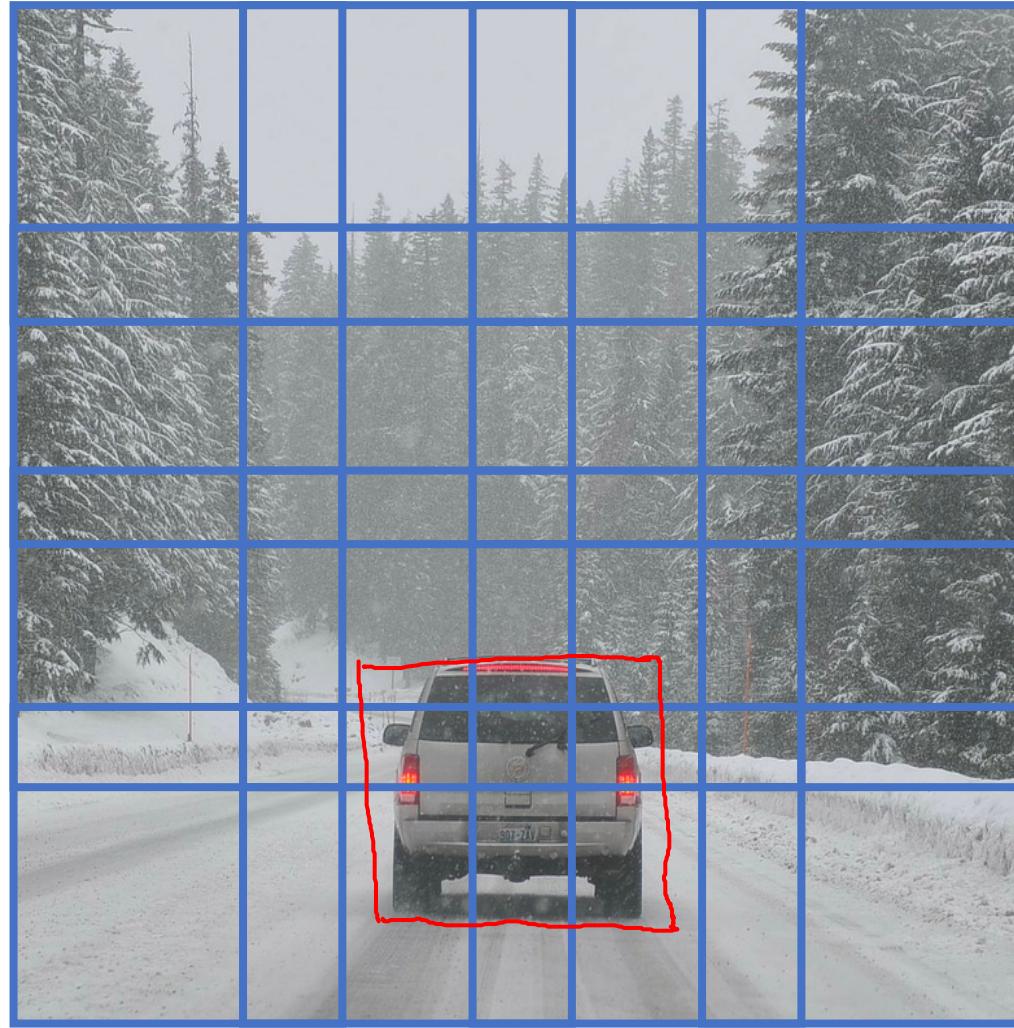


deeplearning.ai

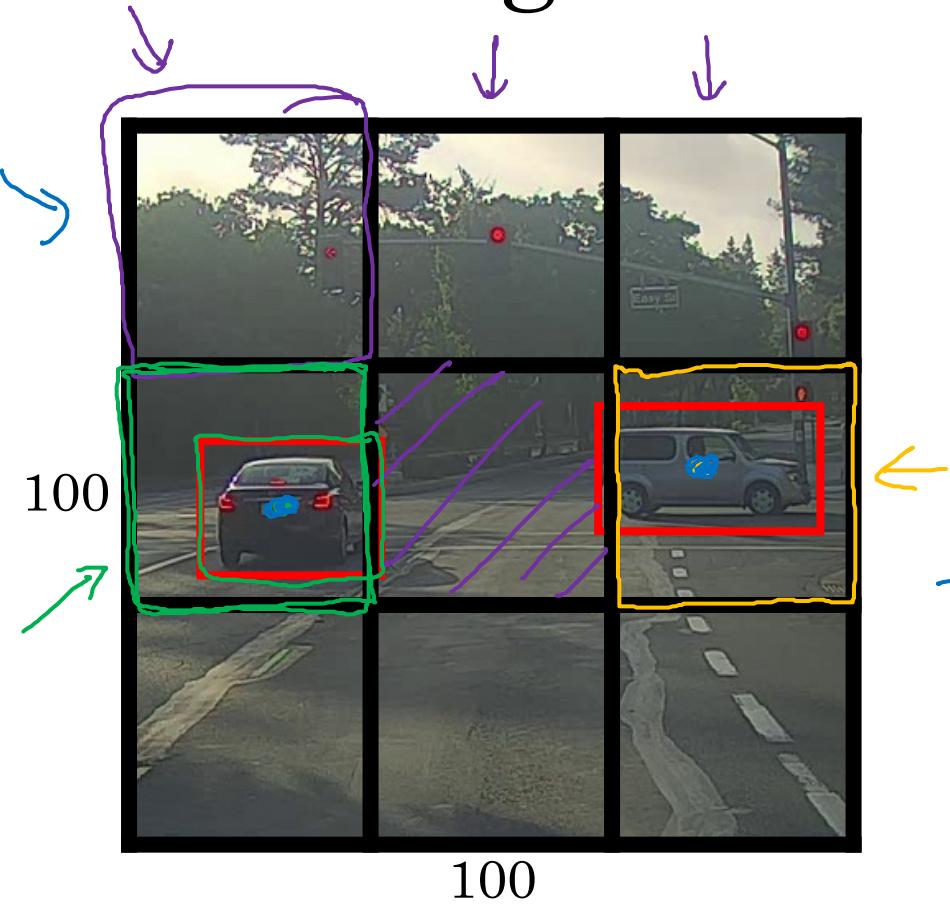
Object Detection

Bounding box
predictions

Output accurate bounding boxes

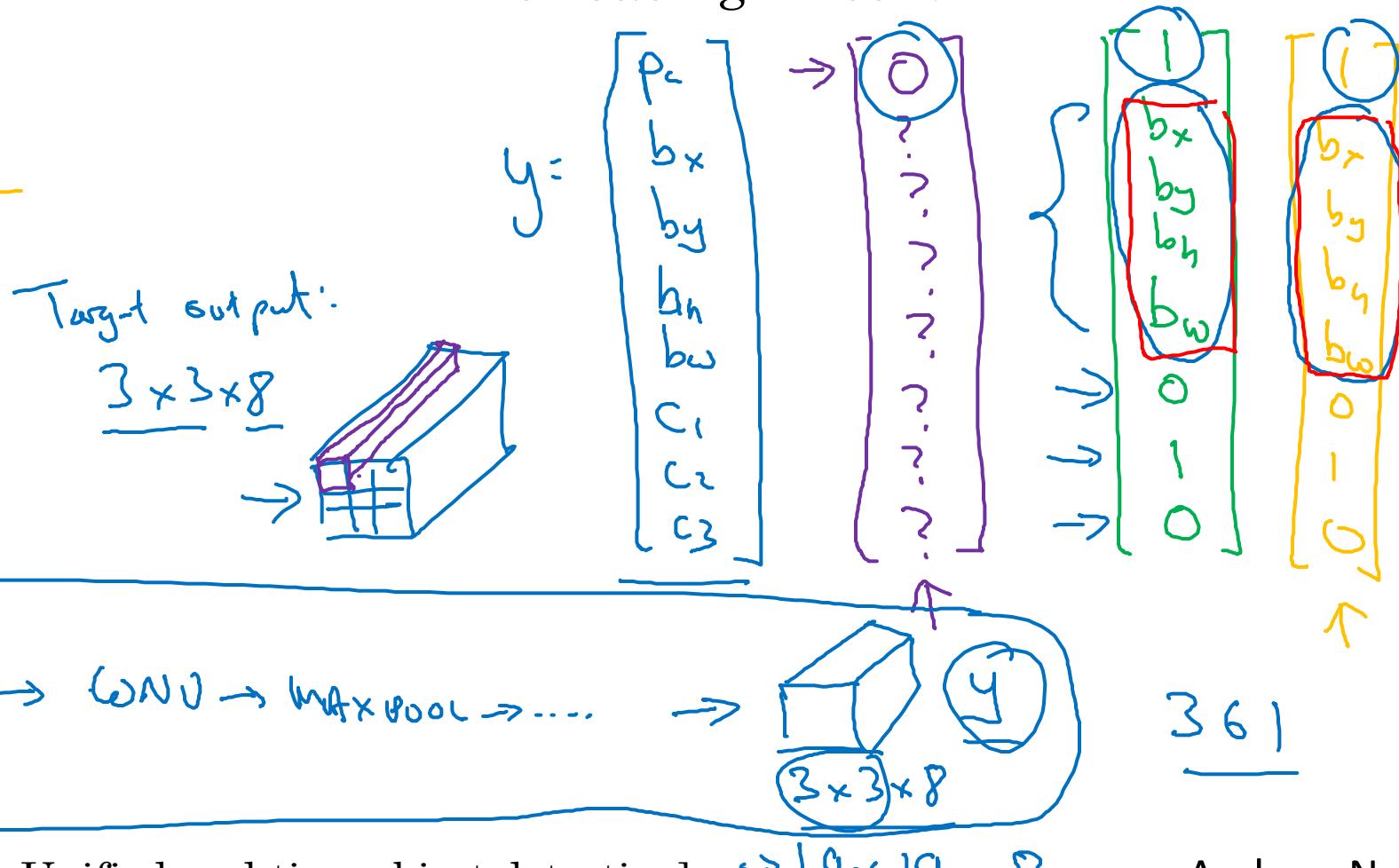


YOLO algorithm

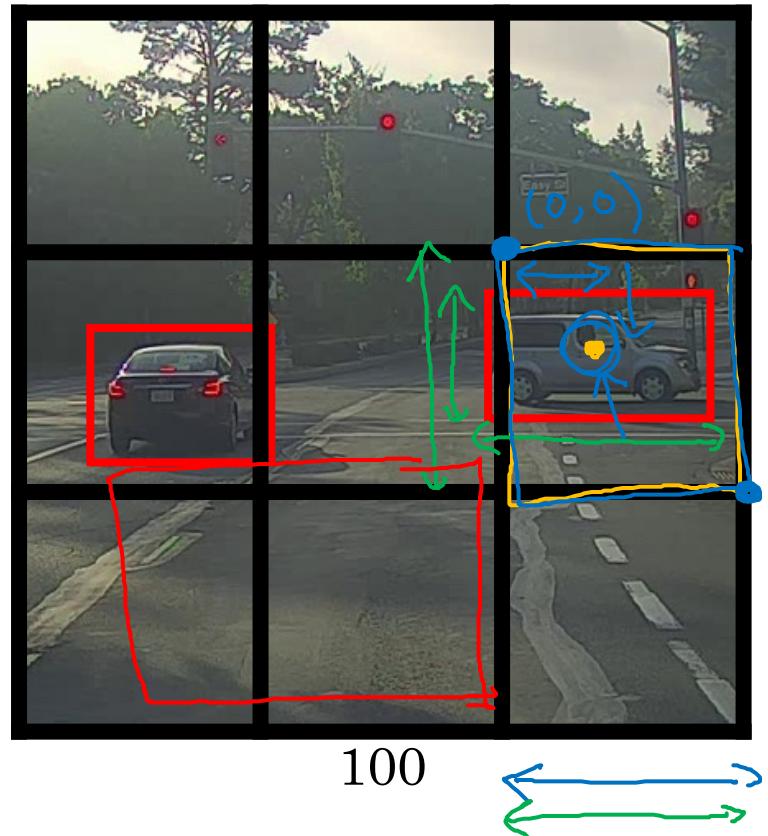


Labels for training

For each grid cell:



Specify the bounding boxes



$$y = \begin{bmatrix} b_x \\ b_y \\ b_h \\ b_w \\ o \end{bmatrix}$$

0.4 } between 0 and 1
0.3 }
0.9 }
0.5 } Could be > 1

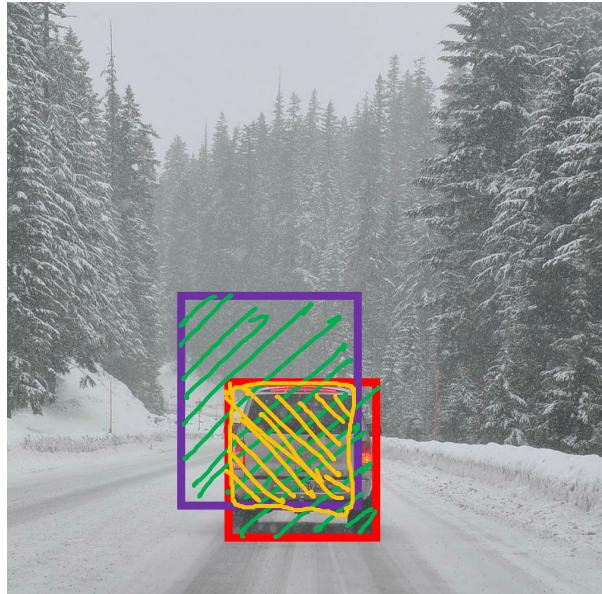


deeplearning.ai

Object Detection

Intersection
over union

Evaluating object localization



Intersection over Union (IoU)

$$= \frac{\text{Size of intersection}}{\text{Size of union}}$$

“Correct” if $\text{IoU} \geq 0.5$

0.6

More generally, IoU is a measure of the overlap between two bounding boxes.



deeplearning.ai

Object Detection

Non-max suppression

Non-max suppression example

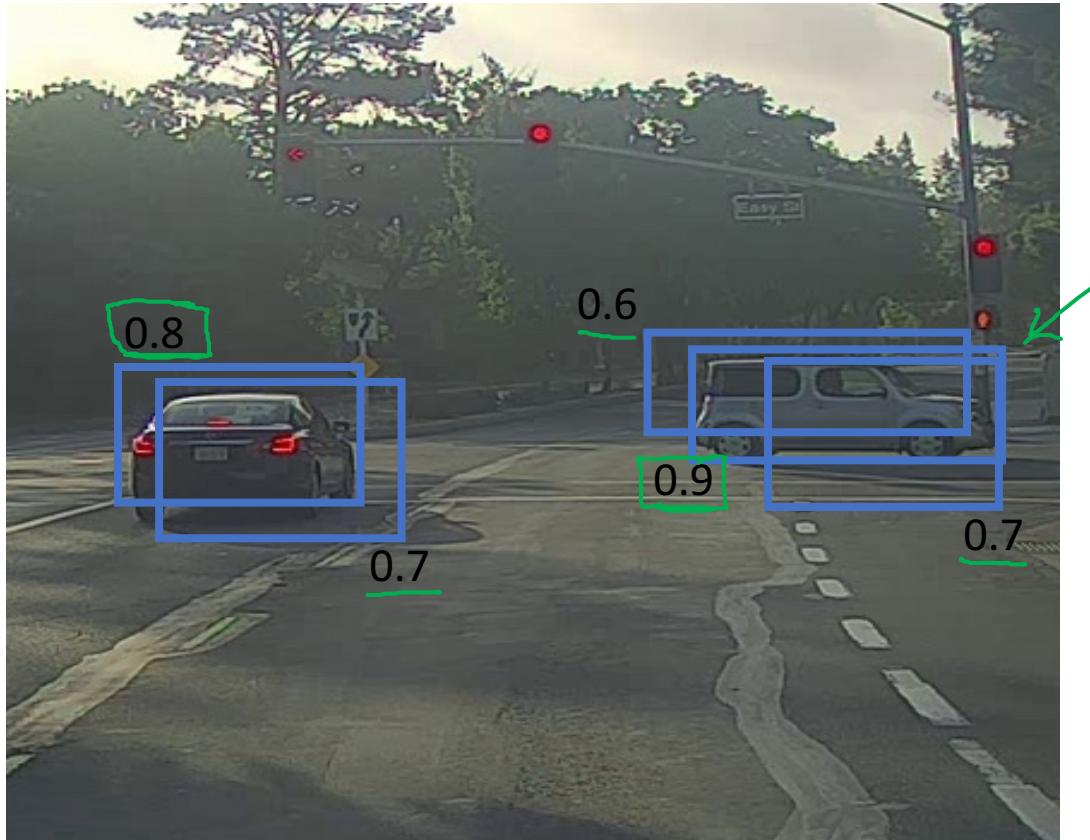


Non-max suppression example

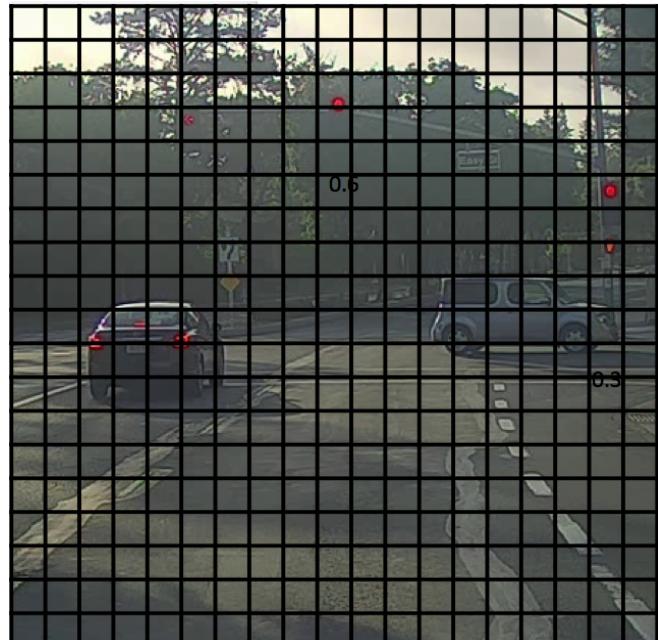


19x19

Non-max suppression example

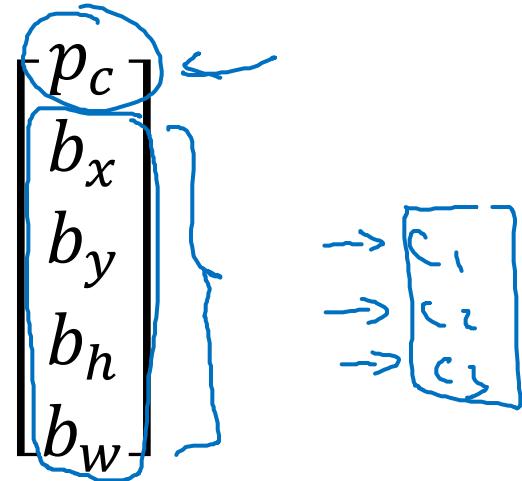


Non-max suppression algorithm



19×19

Each output prediction is:



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 $\text{IoU} \geq 0.5$ with the box output in the previous step

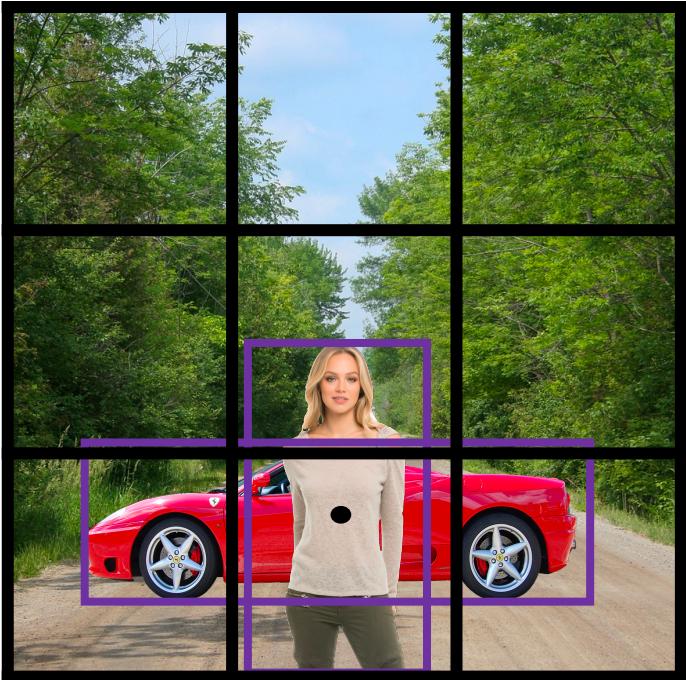


deeplearning.ai

Object Detection

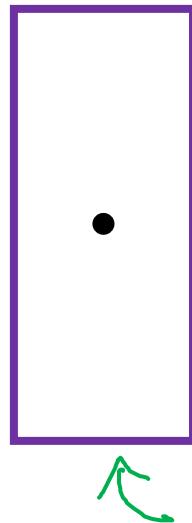
Anchor boxes

Overlapping objects:

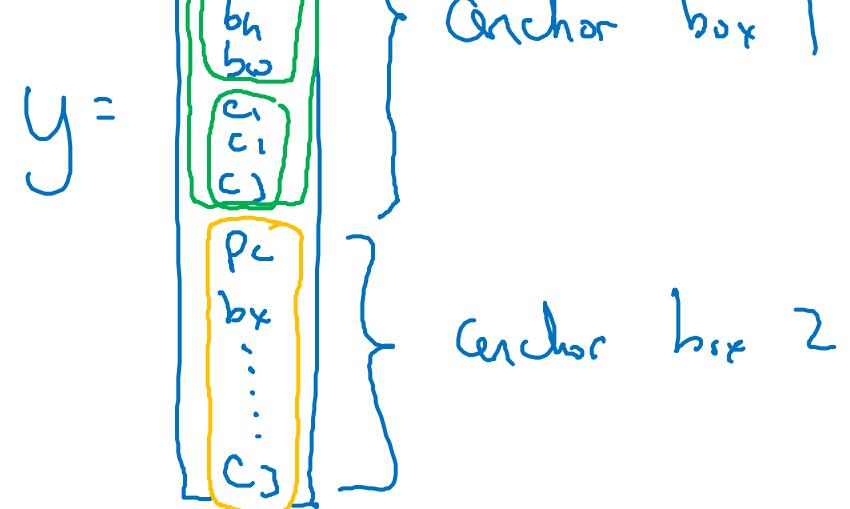
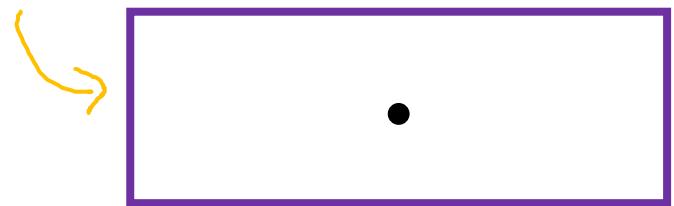


$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Anchor box 1:



Anchor box 2:



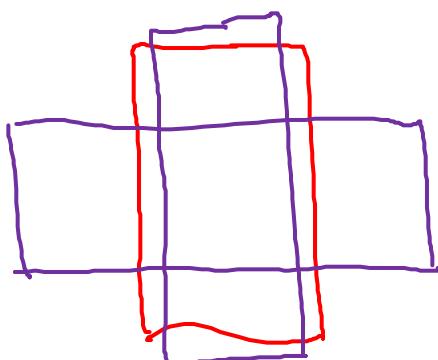
Anchor box algorithm

Previously:

Each object in training image is assigned to grid cell that contains that object's midpoint.

Output y:

$3 \times 3 \times 8$



With two anchor boxes:

Each object in training image is assigned to grid cell that contains object's midpoint and anchor box for the grid cell with highest IoU.

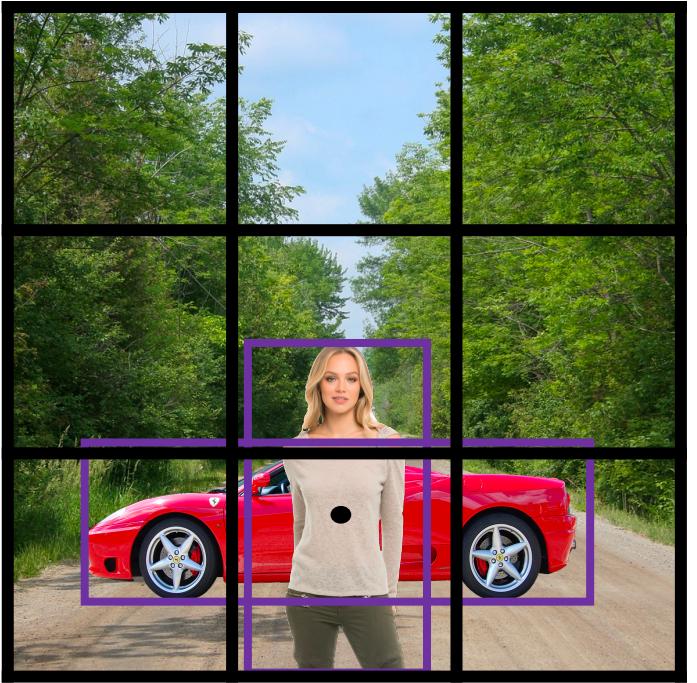
(grid cell, anchor box)

Output y:

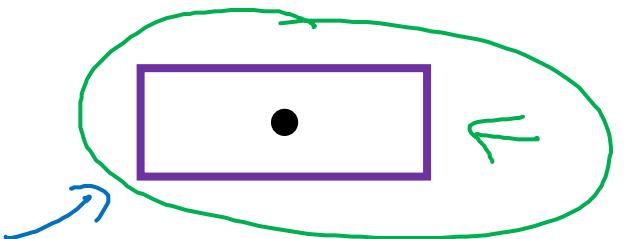
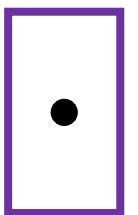
$3 \times 3 \times 16$

$3 \times 3 \times 2 \times 8$

Anchor box example



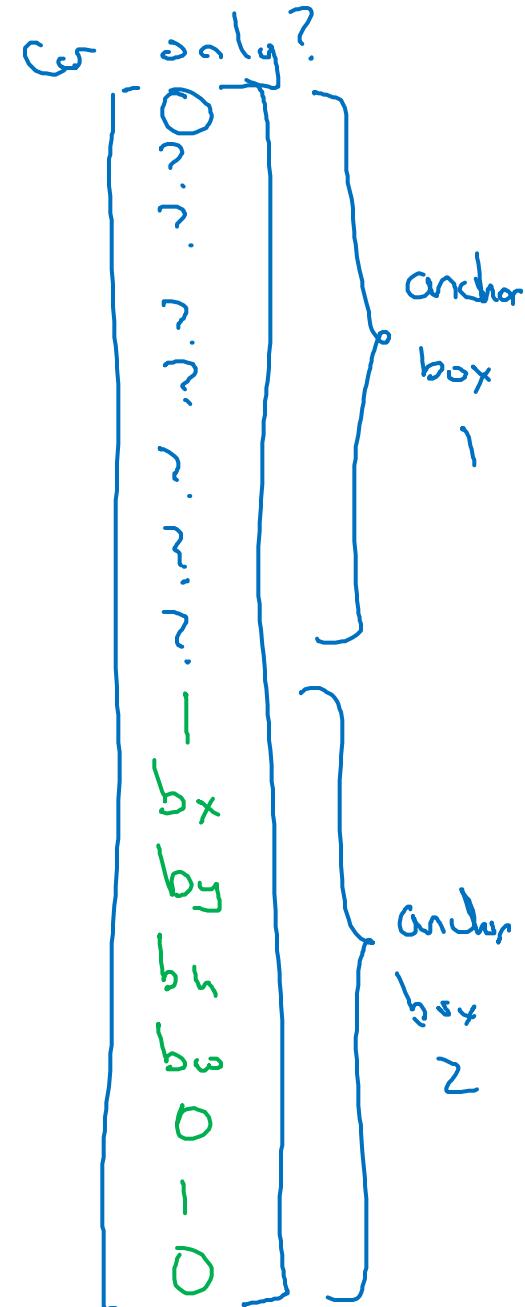
Anchor box 1: Anchor box 2:



$$y =$$

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

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



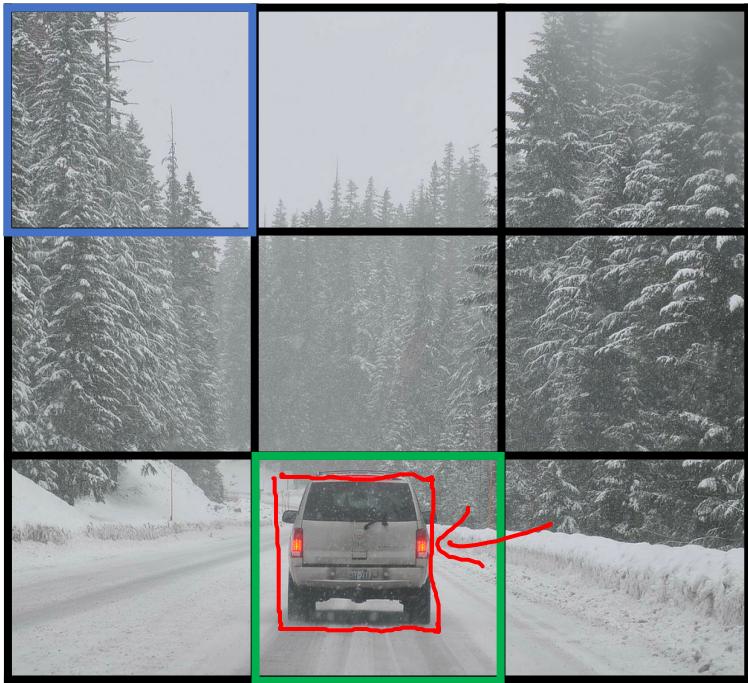


deeplearning.ai

Object Detection

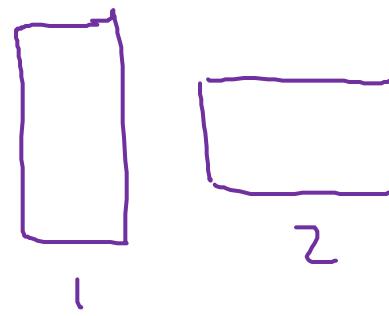
Putting it together:
YOLO algorithm

Training



- 1 - pedestrian
- 2 - car ←
- 3 - motorcycle

$y =$

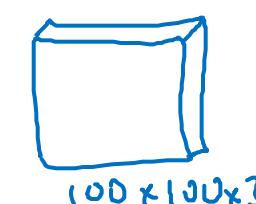
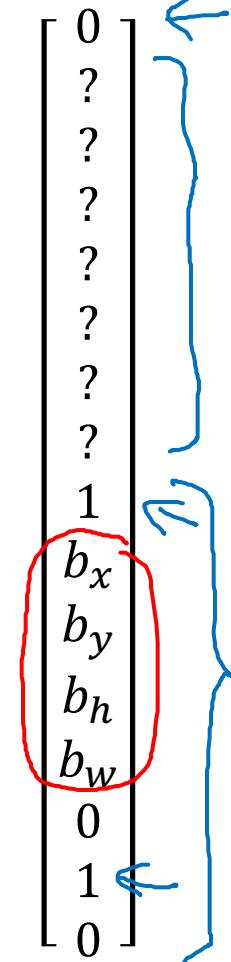
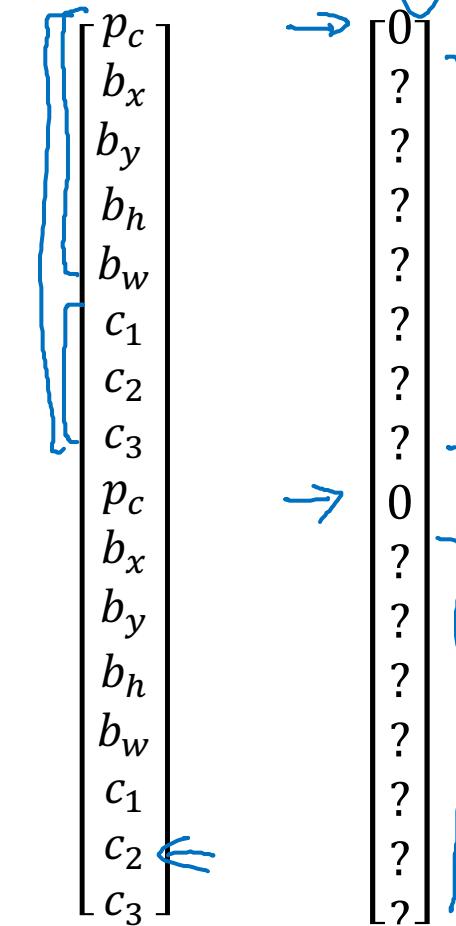


y is $3 \times 3 \times 2 \times 8$

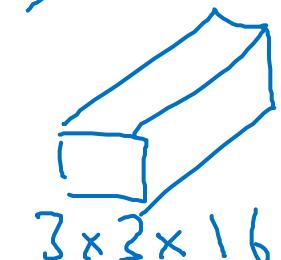
$3 \times 3 \times 16$

$19 \times 19 \times 16$
 $19 \times 19 \times 40$

#anchors $\rightarrow 5 + \#classes$

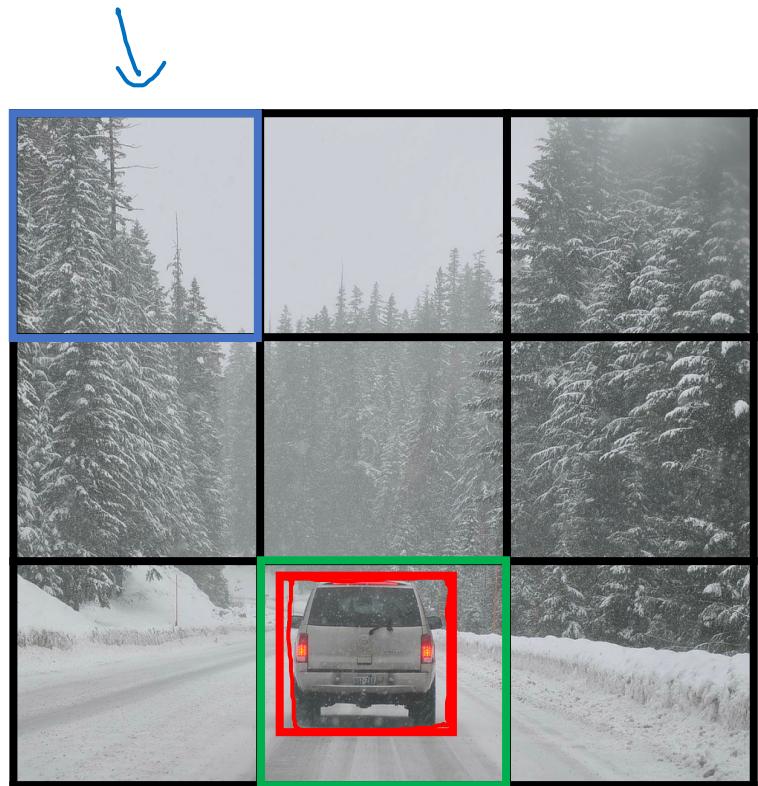


\rightarrow ConvNet



Andrew Ng

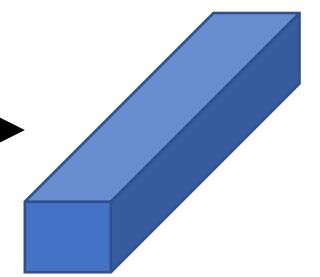
Making predictions



→

...

→

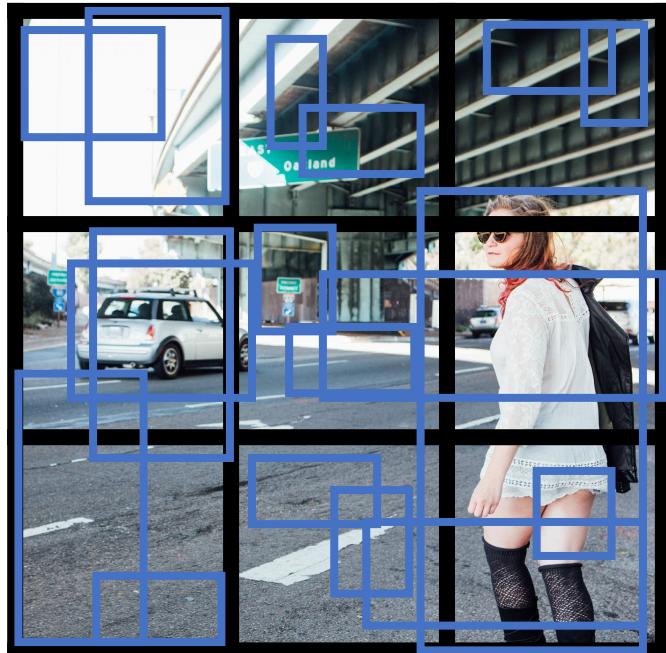


$y =$

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

Diagram illustrating the output vector y for three classes (c_1, c_2, c_3). The vector is composed of 15 elements: the first two elements are p_c (confidence scores), followed by four bounding box parameters (b_x, b_y, b_h, b_w) for each class, and finally three class labels (c_1, c_2, c_3). Handwritten annotations in blue and red are present: a blue bracket groups the first two elements as $[O]$; blue arrows point to the first two elements and the last element; red annotations show the labels b_x, b_y, b_h, b_w next to their respective bounding box parameters, and a green circle highlights the third element c_1 . A blue arrow also points to the bottom element c_3 .

Outputting the non-max suppressed outputs



- 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.

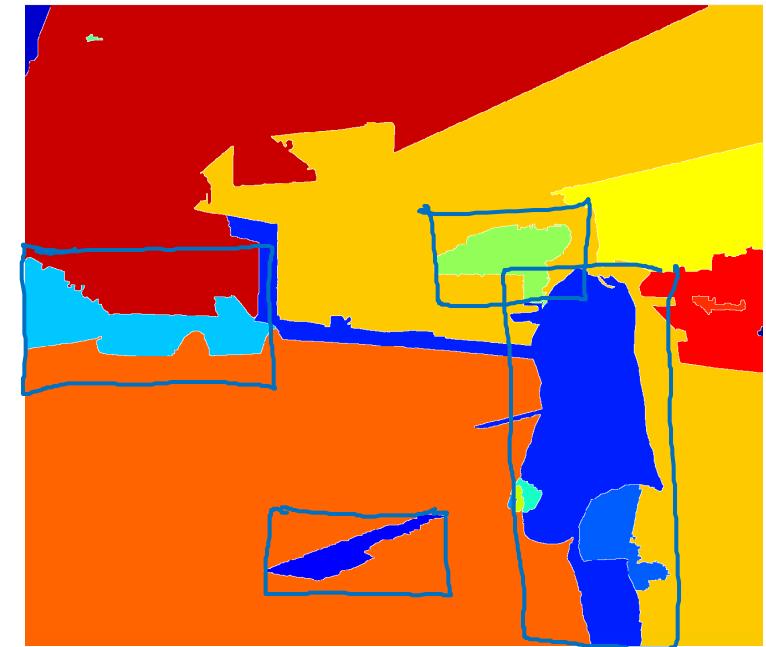
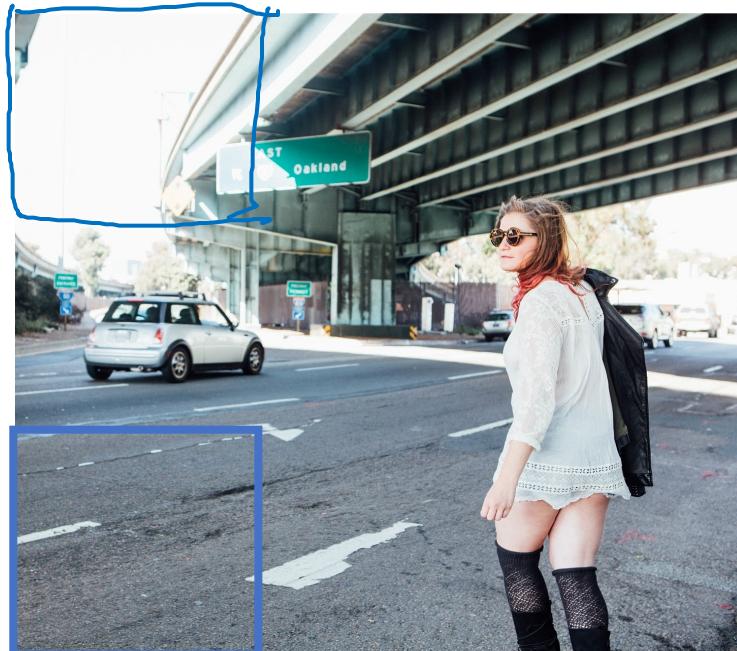
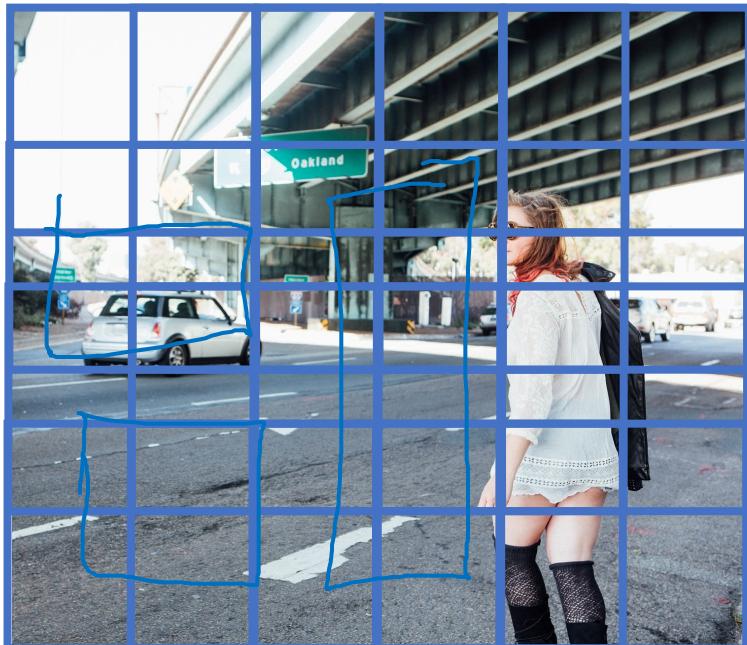


deeplearning.ai

Object Detection

Region proposals
(Optional)

Region proposal: R-CNN



Segmentation algorithm

~ 2,000

Faster algorithms

- 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.

[Girshik et. al, 2013. Rich feature hierarchies for accurate object detection and semantic segmentation]

[Girshik, 2015. Fast R-CNN]

[Ren et. al, 2016. Faster R-CNN: Towards real-time object detection with region proposal networks]

Andrew Ng