



deeplearning.ai

Object Detection

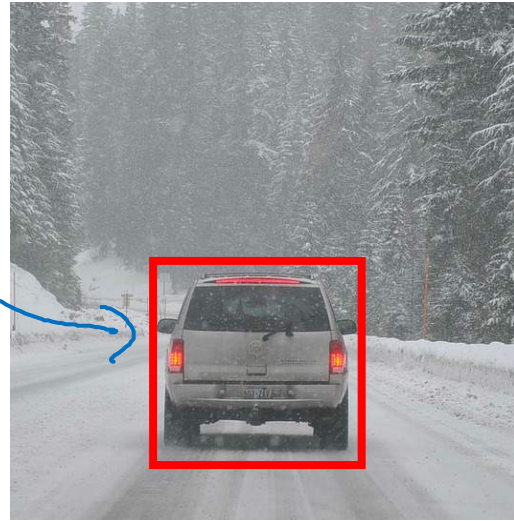
Object
localization

What are localization and detection?

Image classification



Classification with
localization



Detection



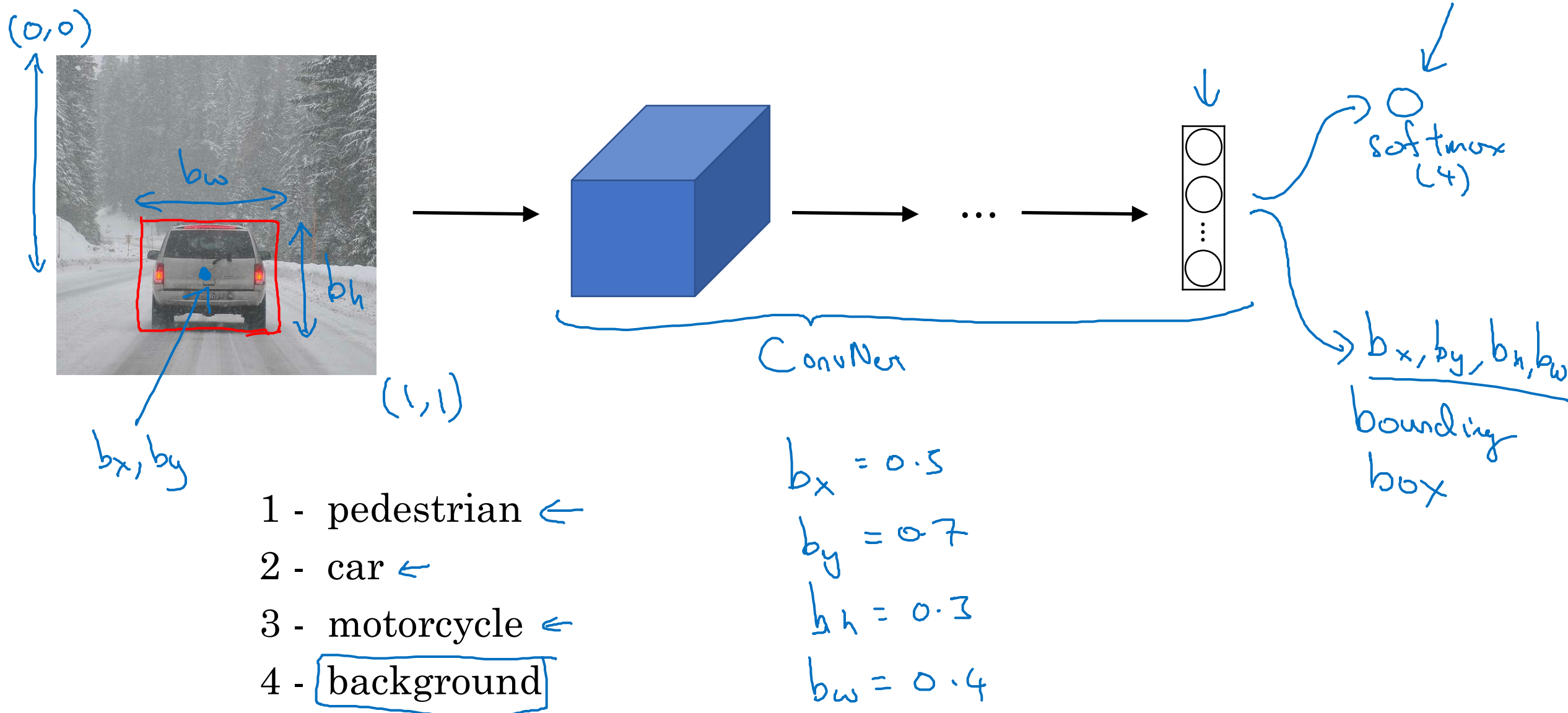
"Car"

"Car"

1 object

multiple
objects

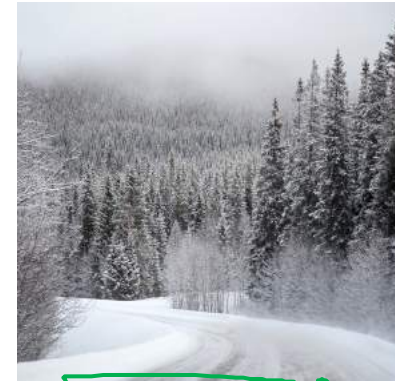
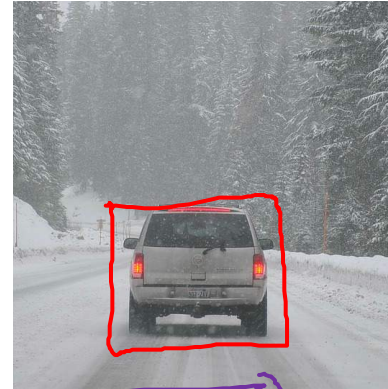
Classification with localization



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)



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

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

(x, y)

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

$$\begin{bmatrix} 0 \\ \vdots \end{bmatrix} \quad \left. \begin{array}{l} P_c \\ \text{"don't care"} \end{array} \right\}$$

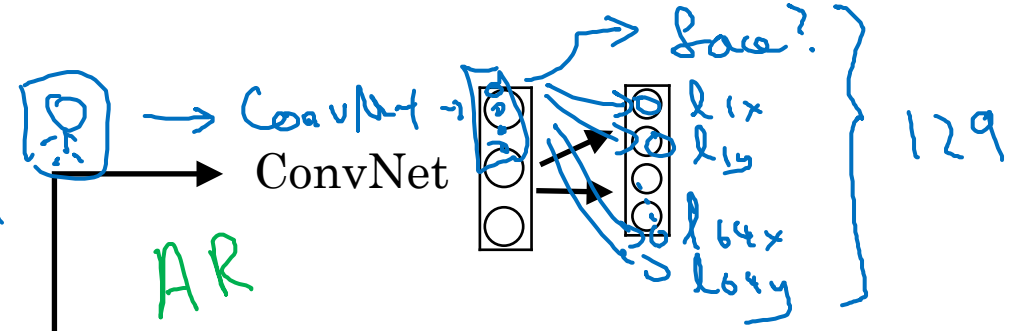


deeplearning.ai

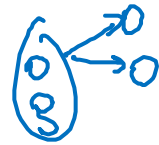
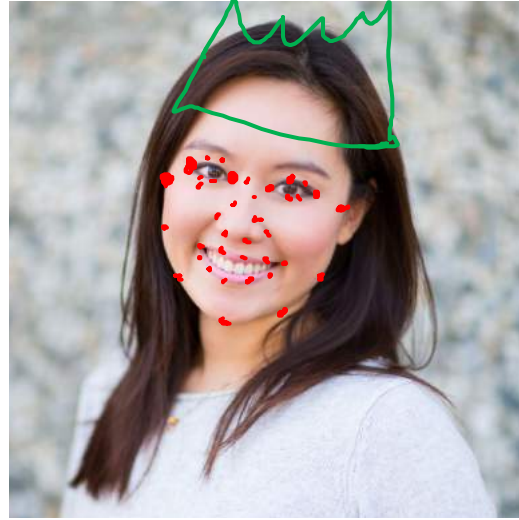
Object Detection

Landmark
detection

Landmark detection



b_x, b_y, b_h, b_w



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

x, y

$l_{1x}, l_{1y}, \dots, l_{32x}, l_{32y}$



deeplearning.ai

Object Detection

Object
detection

Car detection example

Training set:

X

y



1



1



1



0



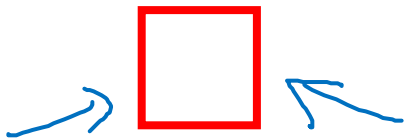
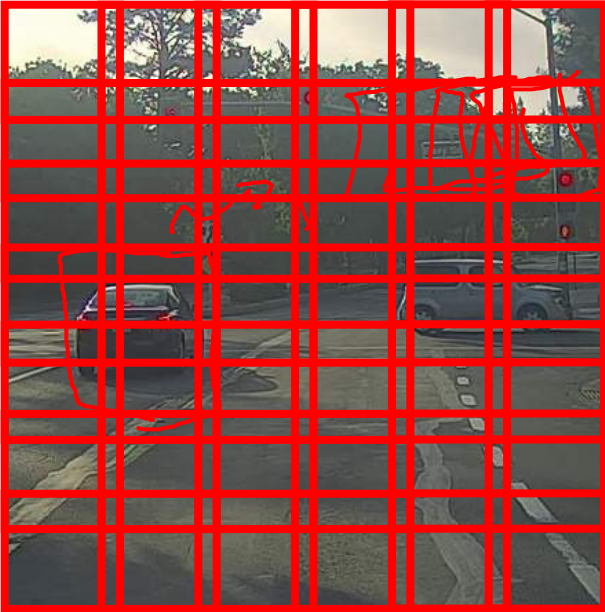
0



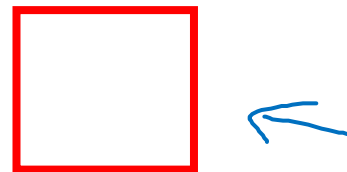
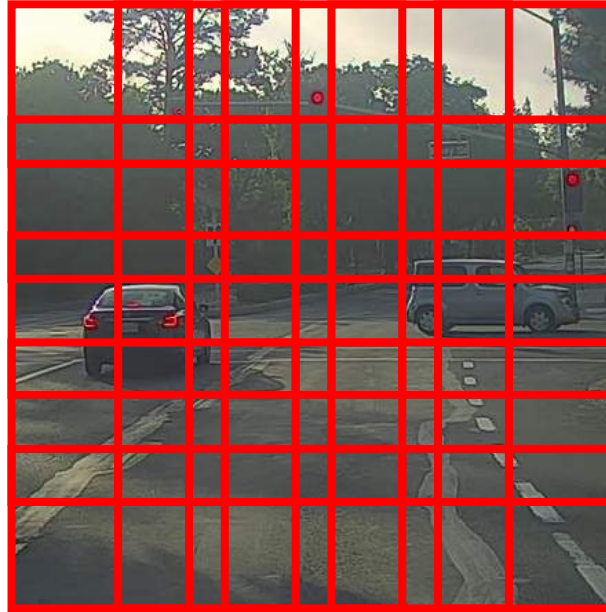
\rightarrow ConvNet $\rightarrow y$

Sliding windows detection

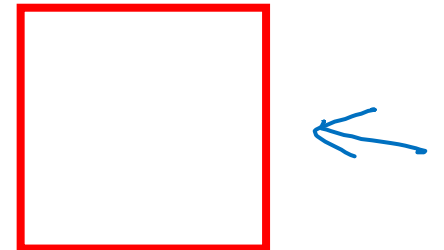
→ ConvNet → 0



→ ConvNet



Computation cost



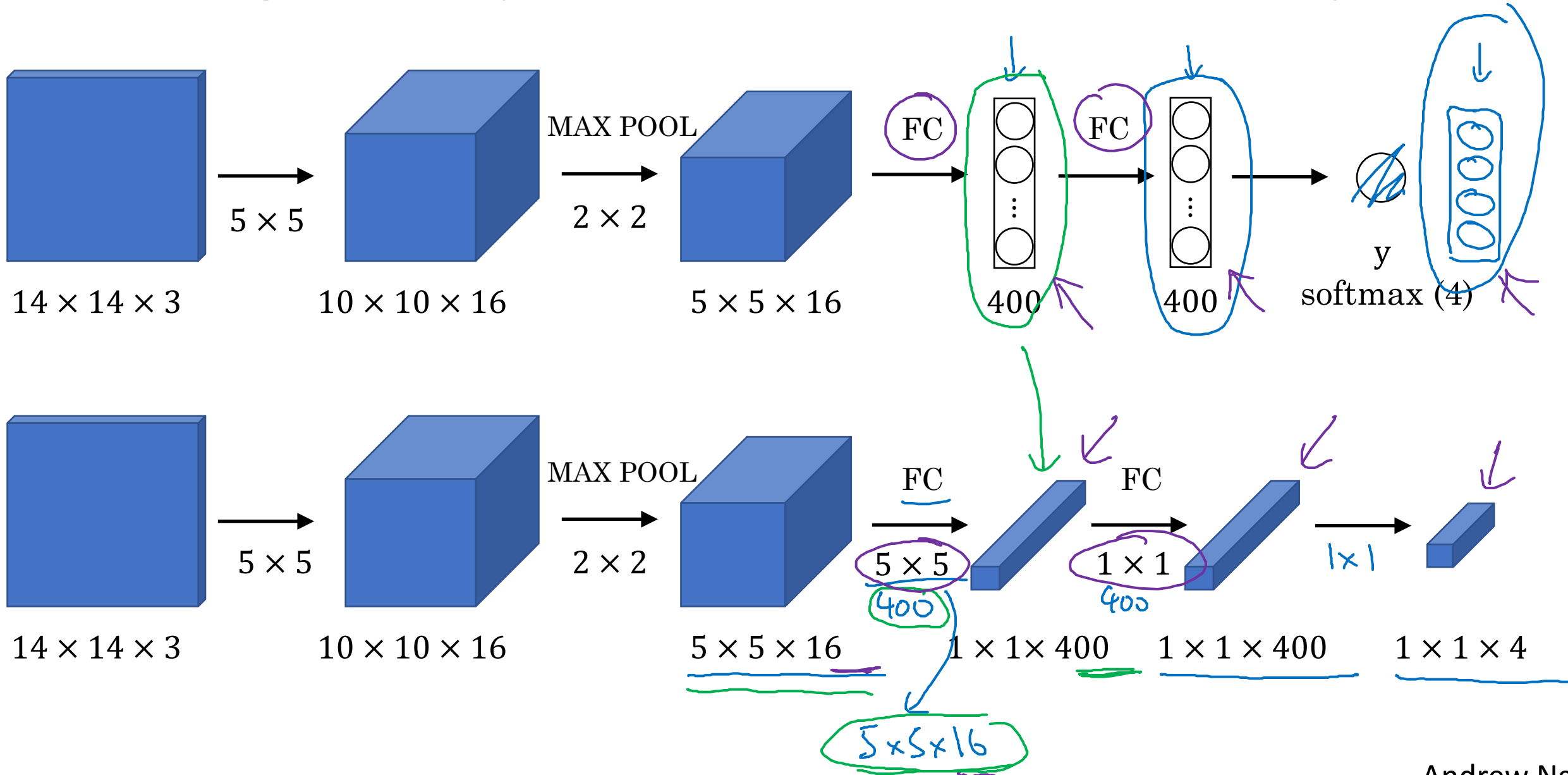


deeplearning.ai

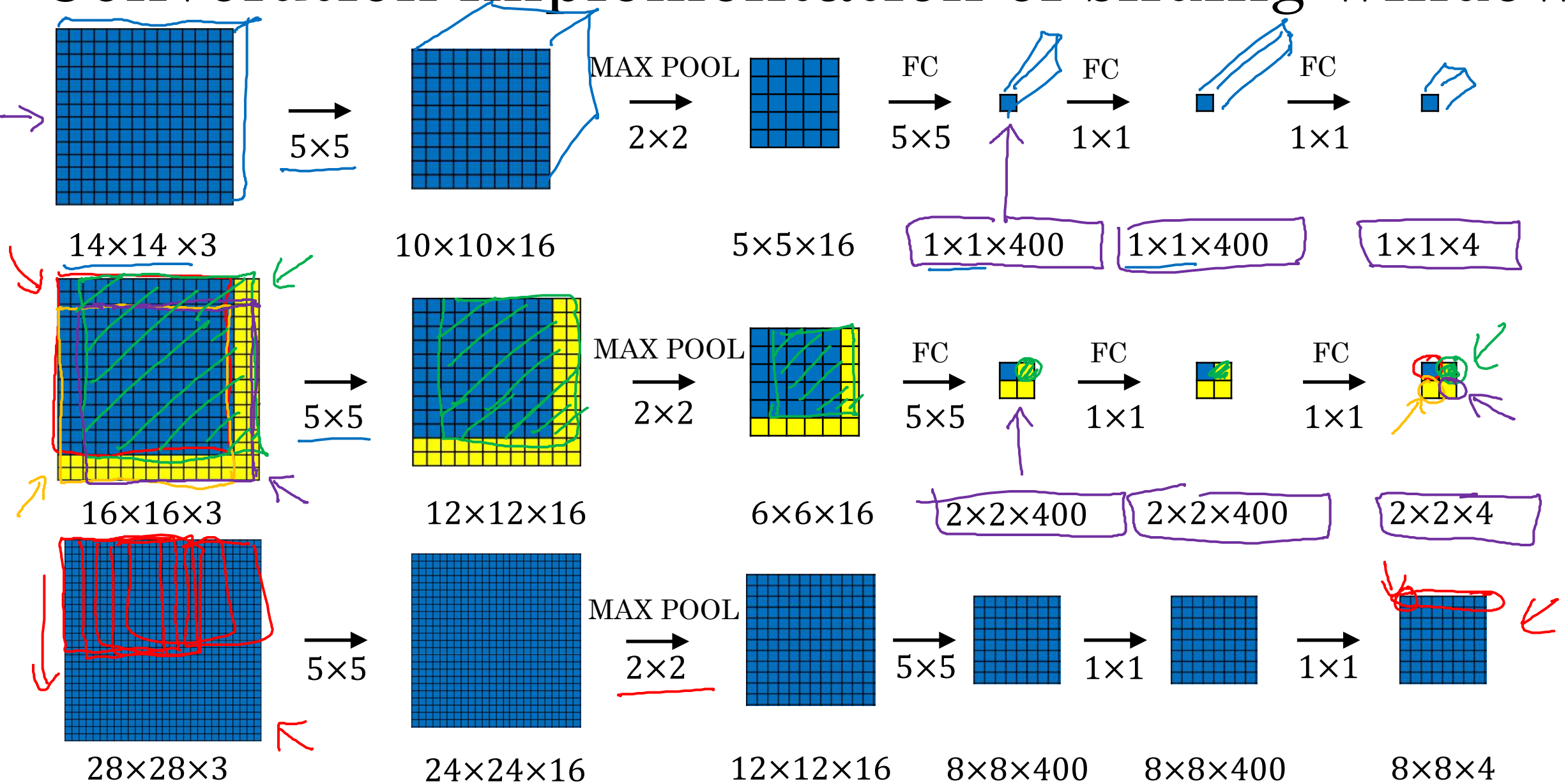
Object Detection

Convolutional
implementation of
sliding windows

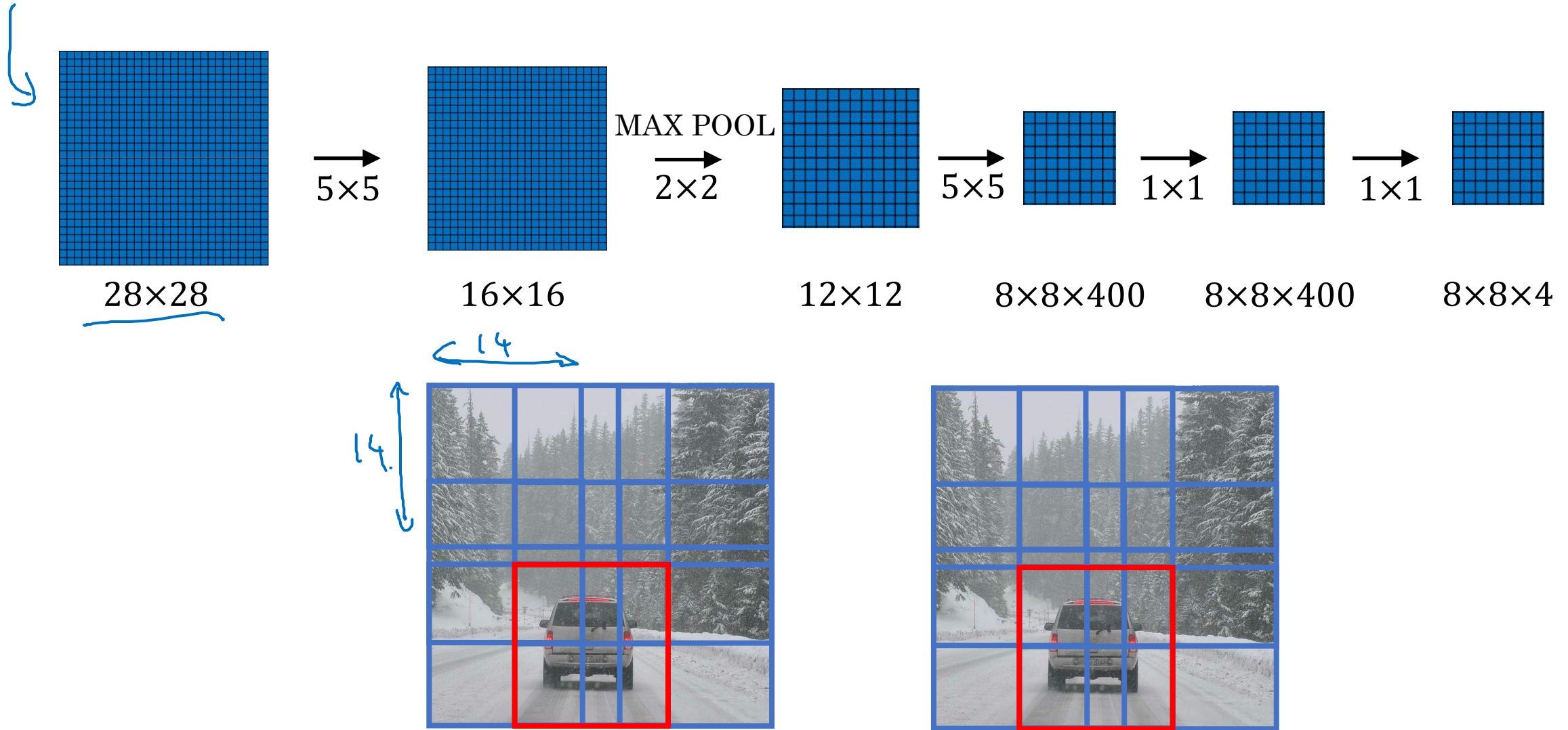
Turning FC layer into convolutional layers



Convolution implementation of sliding windows



Convolution implementation of sliding windows



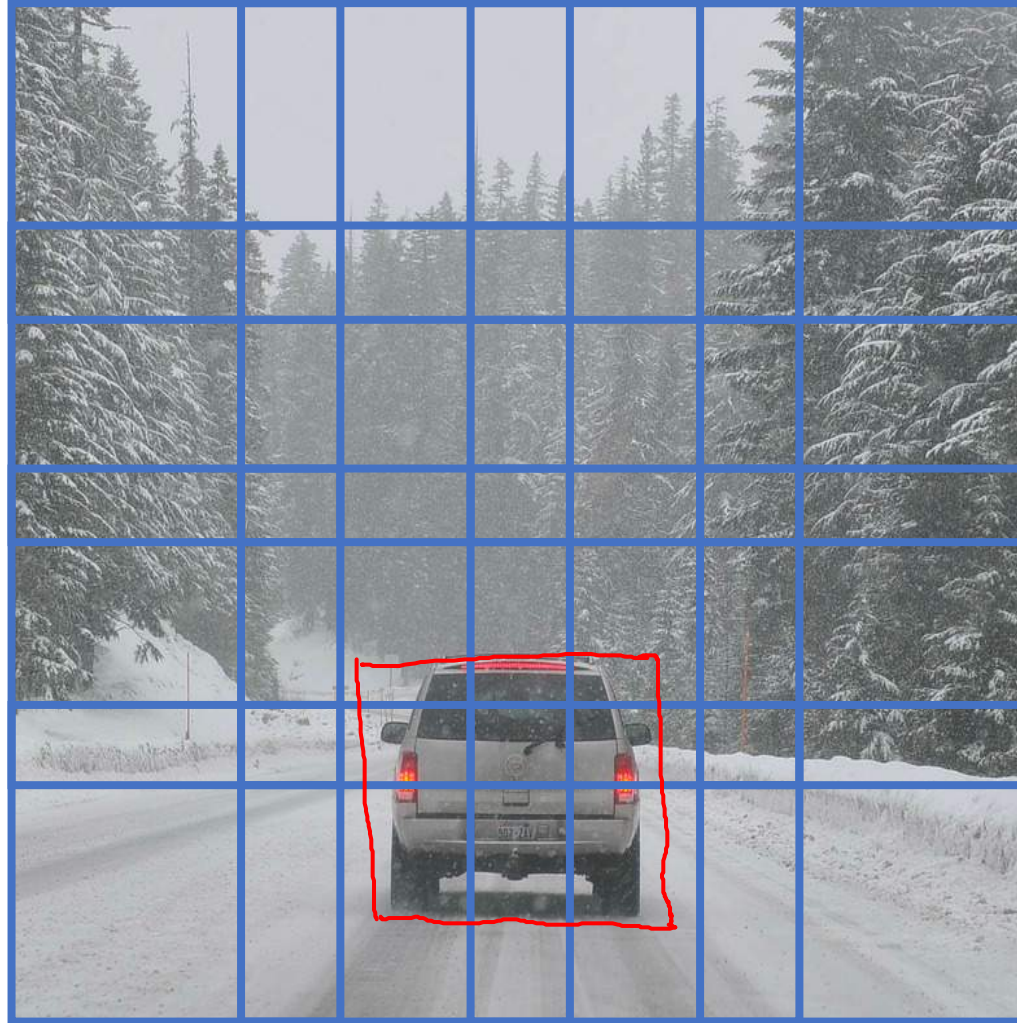


deeplearning.ai

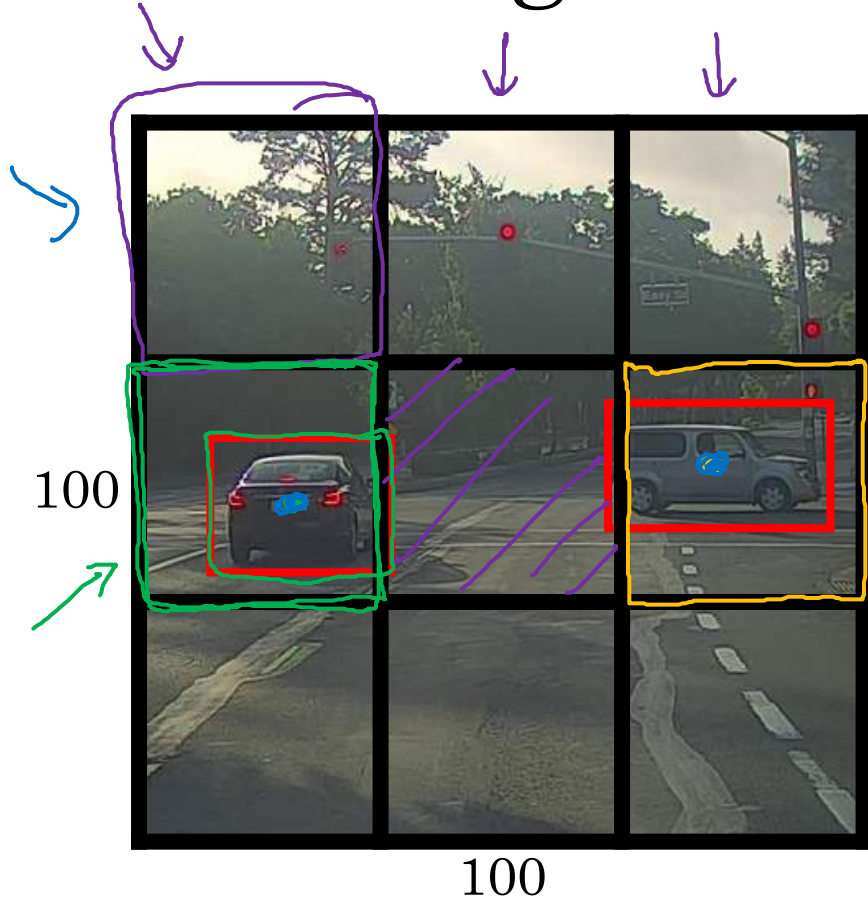
Object Detection

Bounding box
predictions

Output accurate bounding boxes



YOLO algorithm

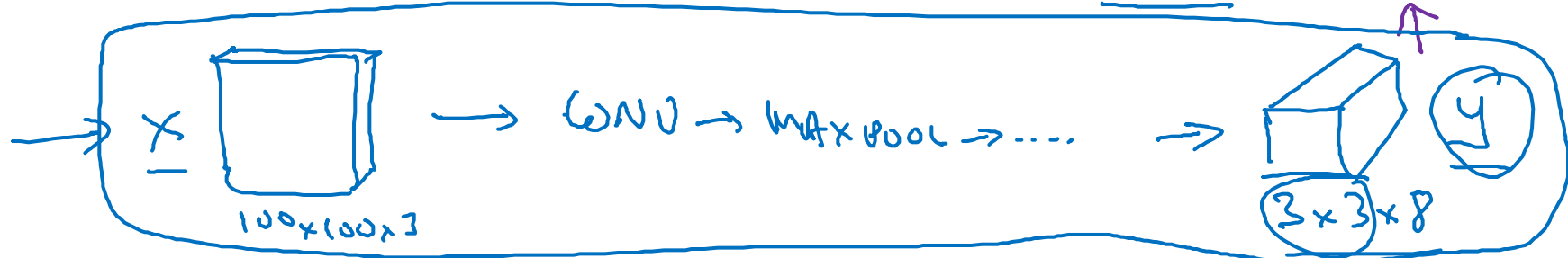
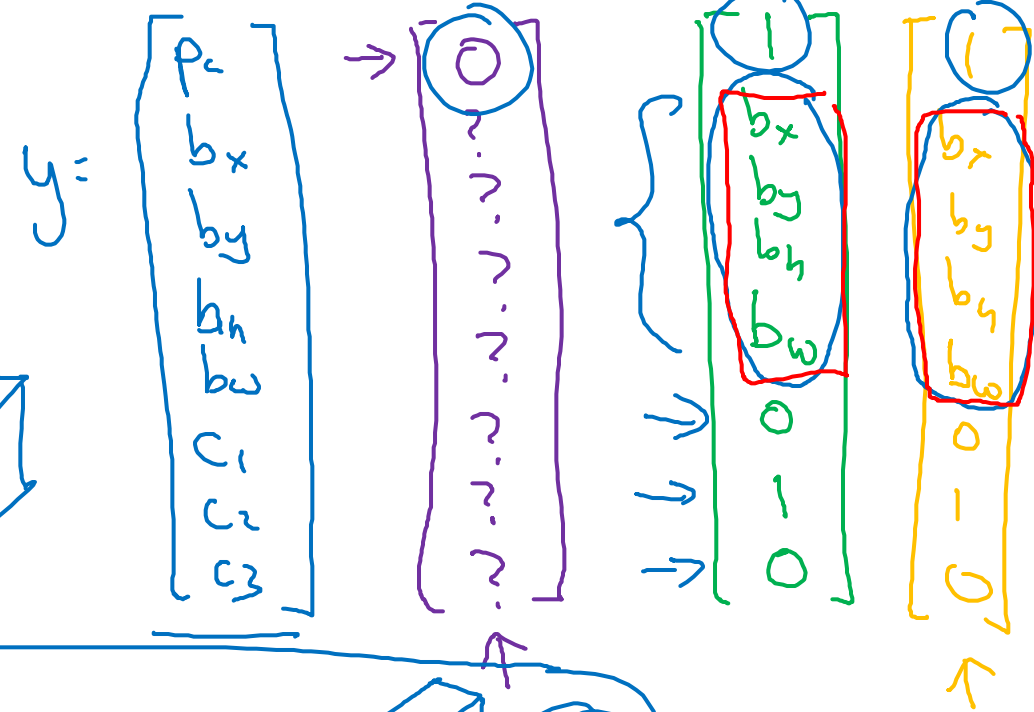
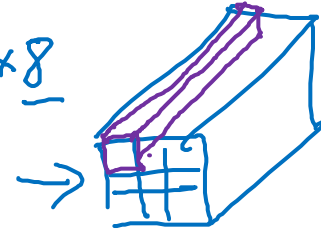


Labels for training

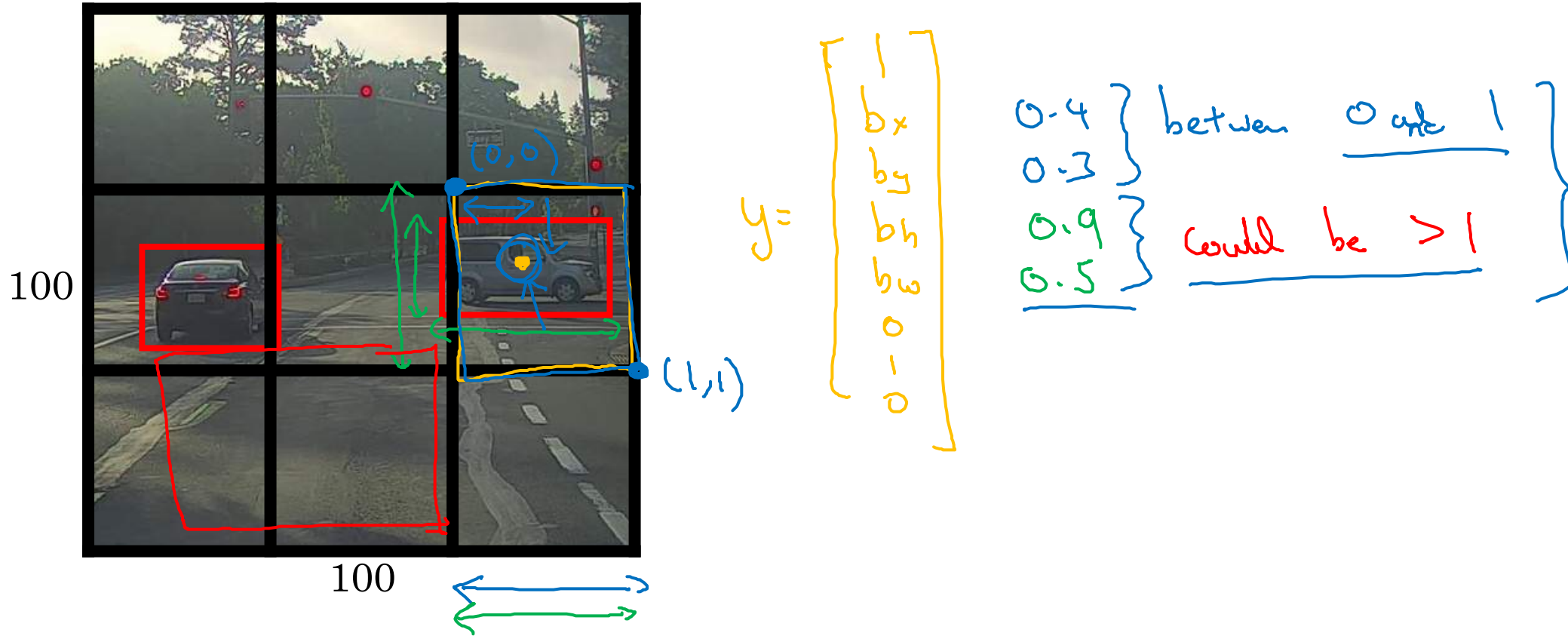
For each grid cell:

Target output:

$3 \times 3 \times 8$



Specify the bounding boxes



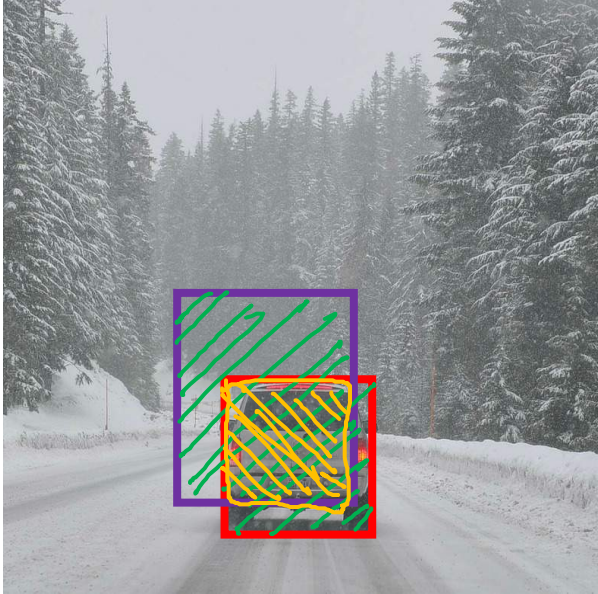


deeplearning.ai

Object Detection

Intersection
over union

Evaluating object localization



Intersection over Union (IoU)

$$= \frac{\text{size of } \text{[yellow hatched box]}}{\text{size of } \text{[green hatched box]}}$$

“Correct” if IoU \geq 0.5 \leftarrow

0.6 \leftarrow

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



deeplearning.ai

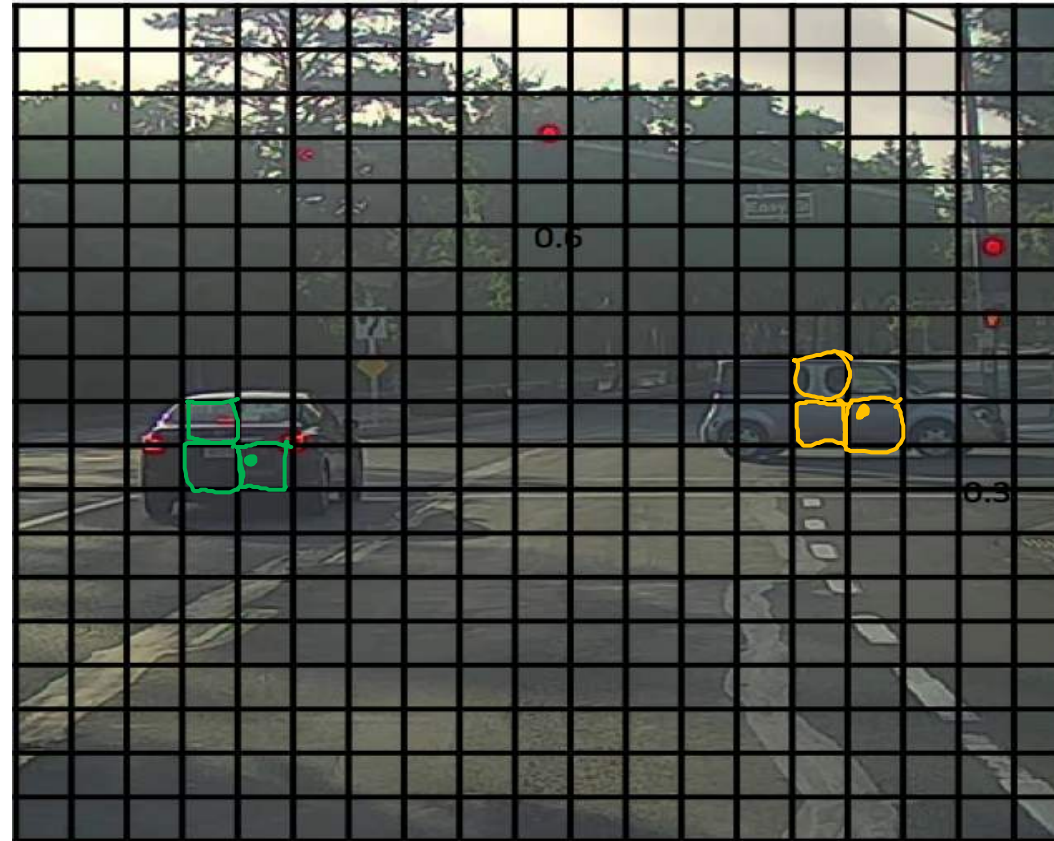
Object Detection

Non-max
suppression

Non-max suppression example

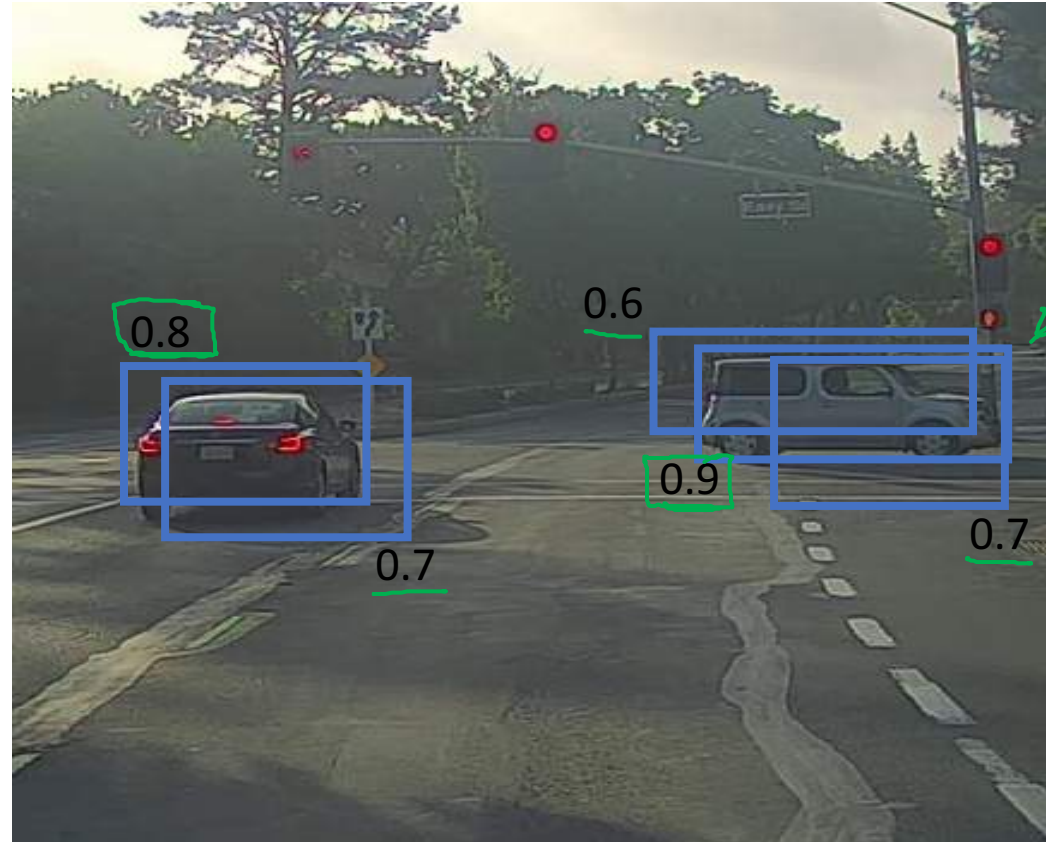


Non-max suppression example



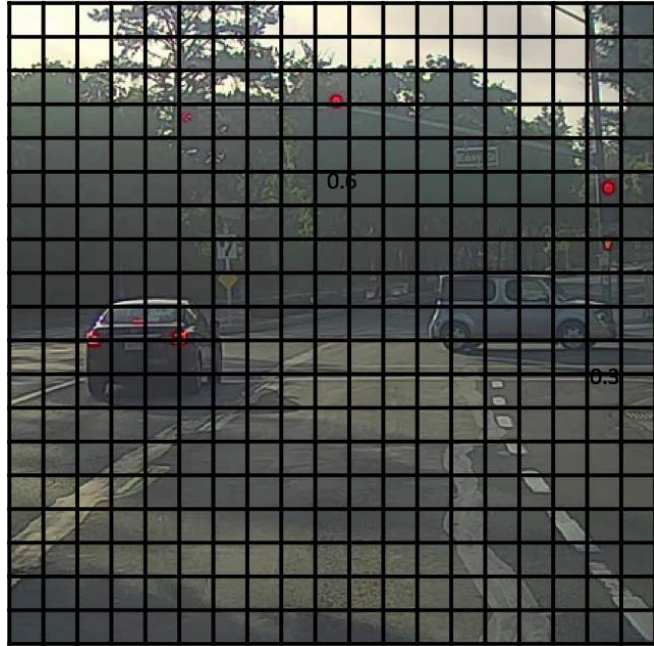
19x19

Non-max suppression example



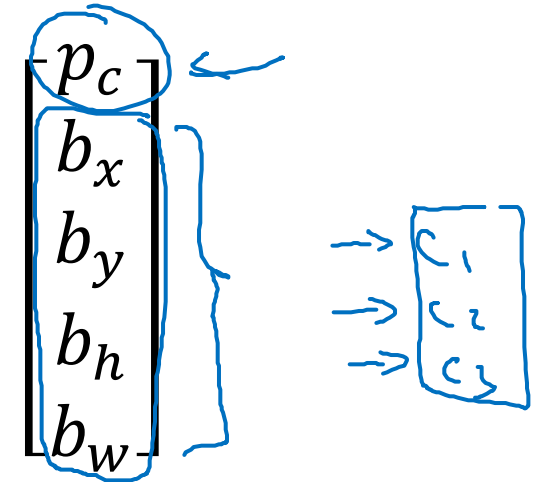
P_c

Non-max suppression algorithm



19x19

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

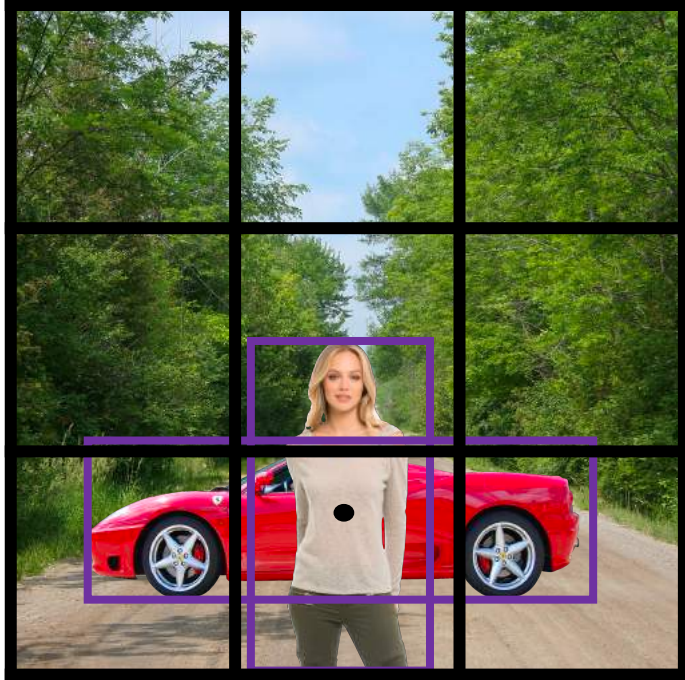


deeplearning.ai

Object Detection

Anchor boxes

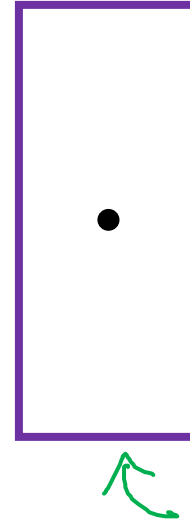
Overlapping objects:



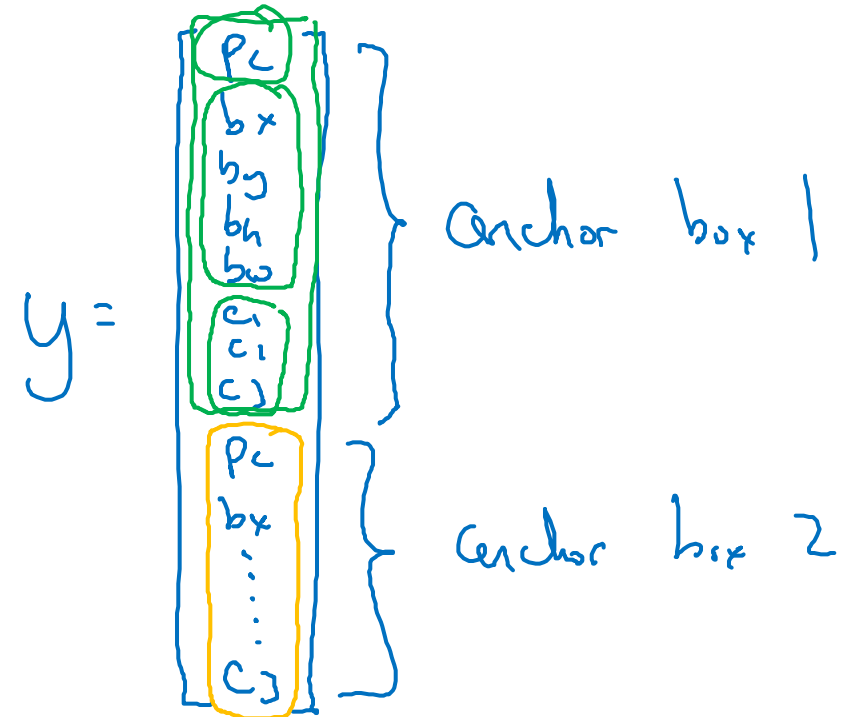
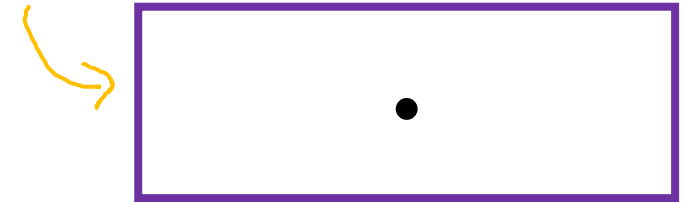
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Annotations: A green arrow points to p_c , a blue arrow points to b_x , and a blue bracket groups c_1, c_2, c_3 .

Anchor box 1:



Anchor box 2:

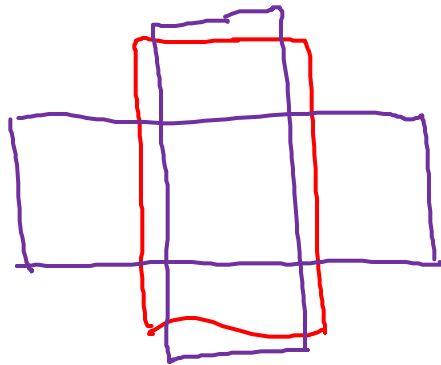


Anchor box algorithm

Previously:

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

Output y :
 $3 \times 3 \times 8$



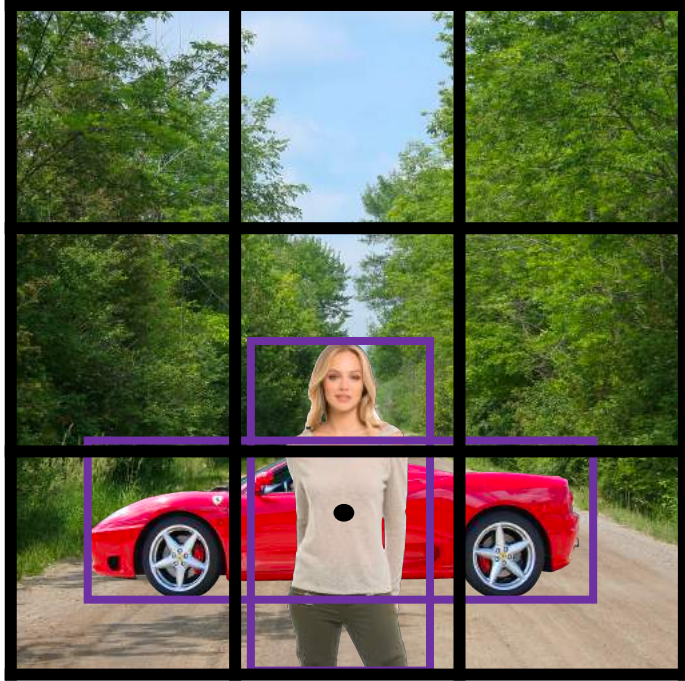
With two anchor boxes:

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.

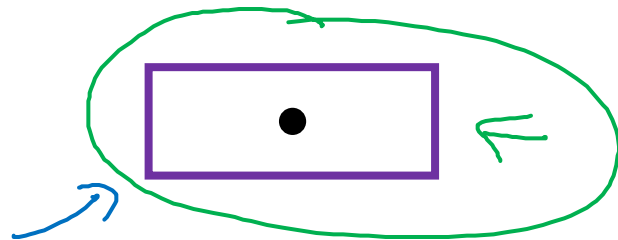
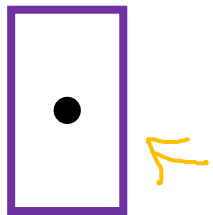
(grid cell, anchor box)

Output y :
 $3 \times 3 \times 16$
 $3 \times 3 \times 2 \times 8$

Anchor box example



Anchor box 1: Anchor box 2:



$y =$

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

Handwritten values for the first set of parameters (vertical):

- p_c : 1
- b_x : 1
- b_y : 1
- b_h : 1
- b_w : 1
- c_1 : 1
- c_2 : 0
- c_3 : 0
- p_c : 1
- b_x : 1
- b_y : 1
- b_h : 1
- b_w : 1
- c_1 : 0
- c_2 : 1
- c_3 : 0

Handwritten values for the second set of parameters (vertical):

- p_c : 1
- b_x : 1
- b_y : 1
- b_h : 1
- b_w : 1
- c_1 : 1
- c_2 : 0
- c_3 : 0
- p_c : 1
- b_x : 1
- b_y : 1
- b_h : 1
- b_w : 1
- c_1 : 0
- c_2 : 1
- c_3 : 0

Labels and annotations:

- Car only?
- anchor box 1
- anchor box 2



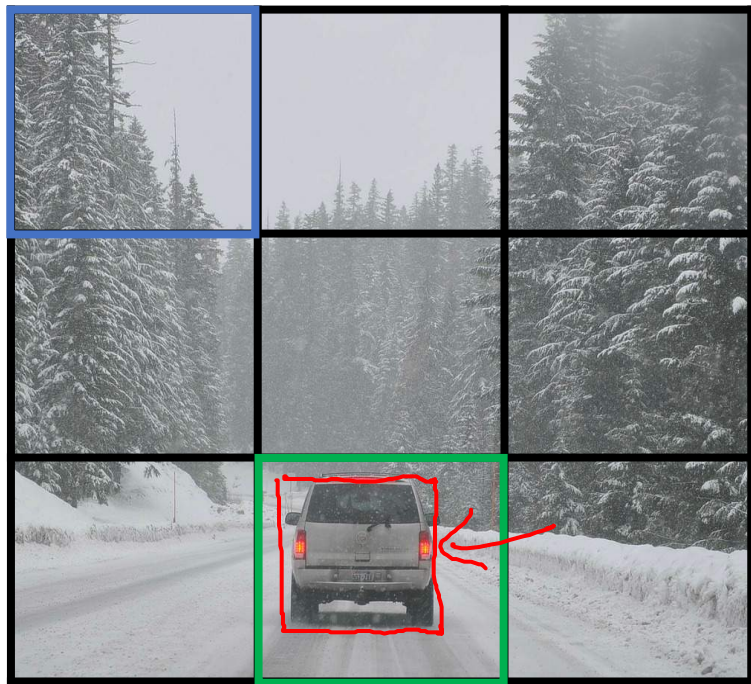
deeplearning.ai

Object Detection

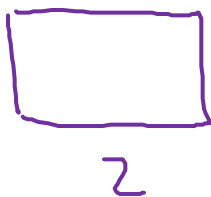
Putting it together:
YOLO algorithm

Training

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



$y =$



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

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

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

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

$19 \times 19 \times 16$

$19 \times 19 \times 40$

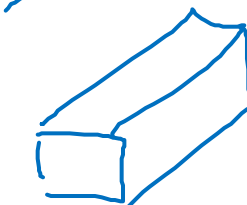
↑
#anchors

↑
 $5 + \#classes$



$100 \times 100 \times 3$

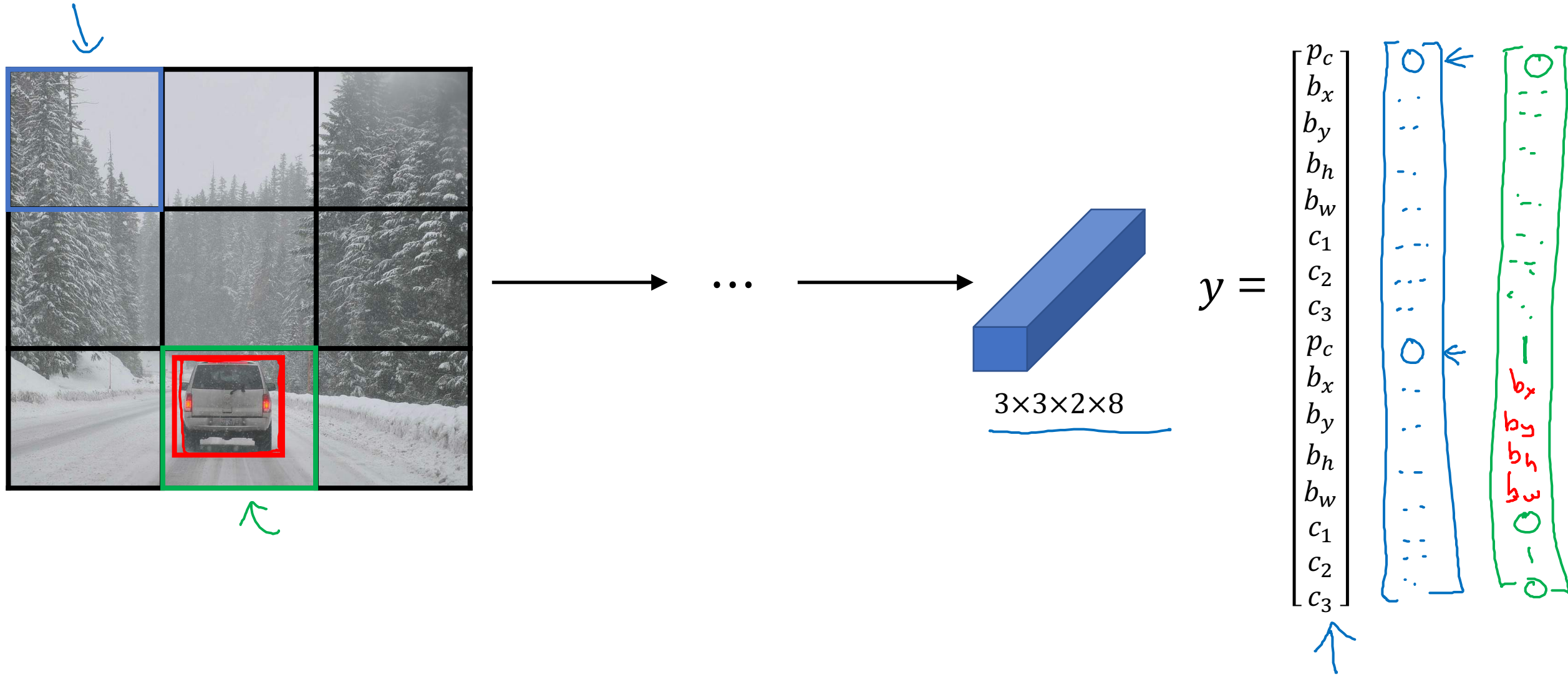
→ ConvNet →



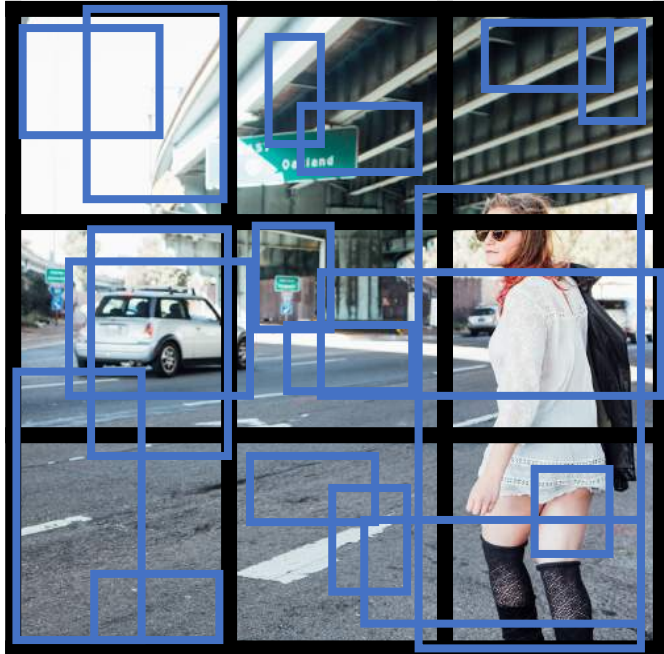
$3 \times 3 \times 16$

Andrew Ng

Making predictions



Outputting the non-max suppressed outputs



- For each grid cell, 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.

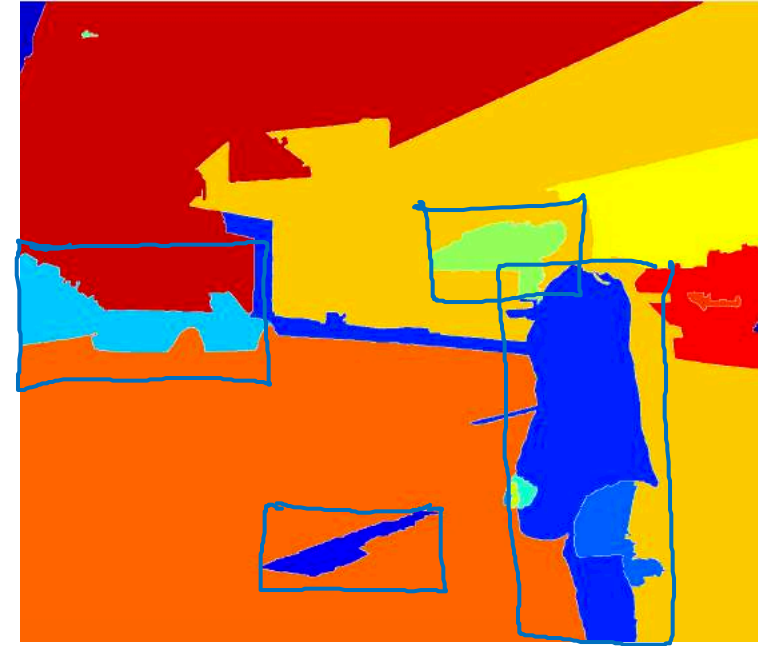
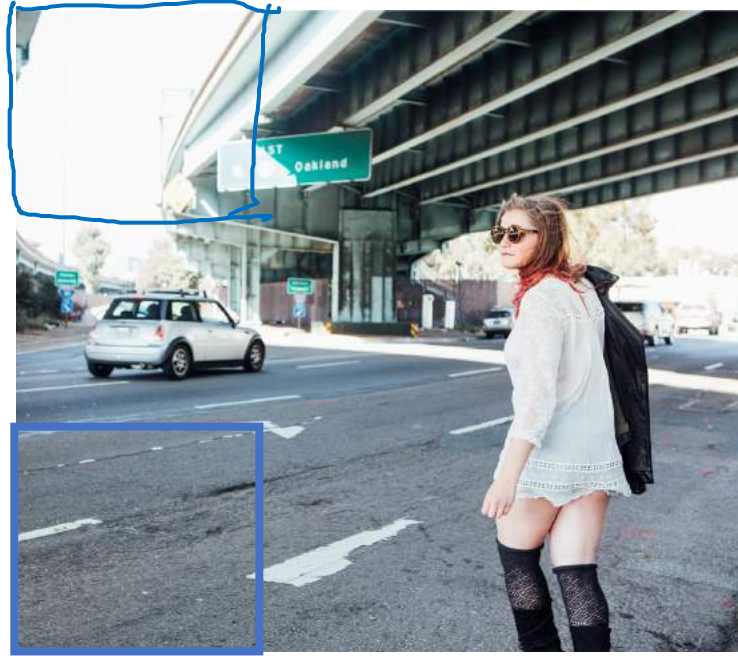
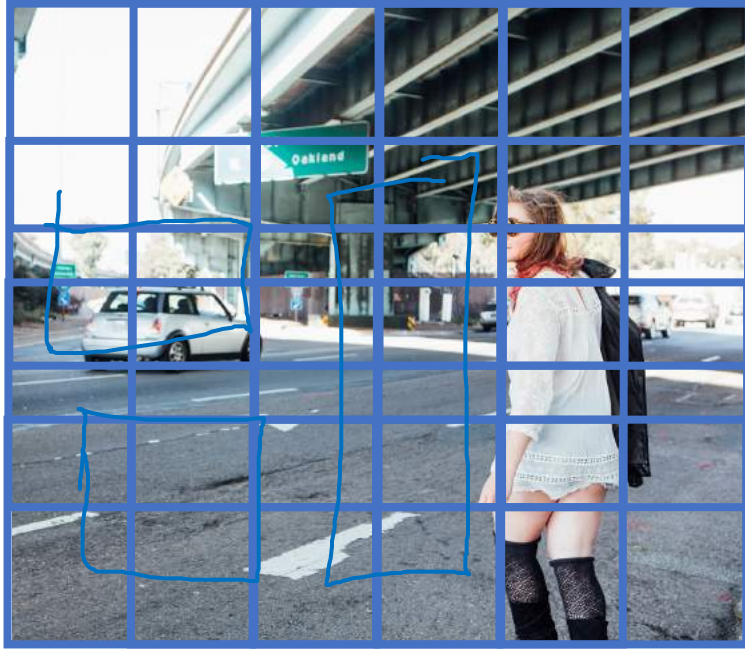


deeplearning.ai

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