



deeplearning.ai

# Object Detection

---

## Object localization

# What are localization and detection?

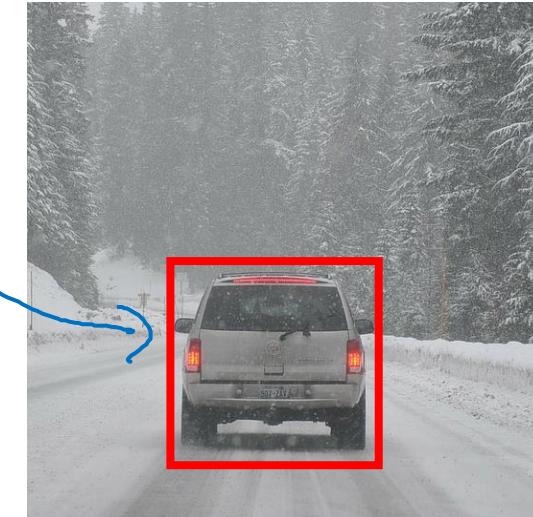
Image classification



"Car"  
|  
object

A blue bracket groups the word "Car" above the image with the word "object" below it, connected by a vertical line.

Classification with  
localization



"Car"

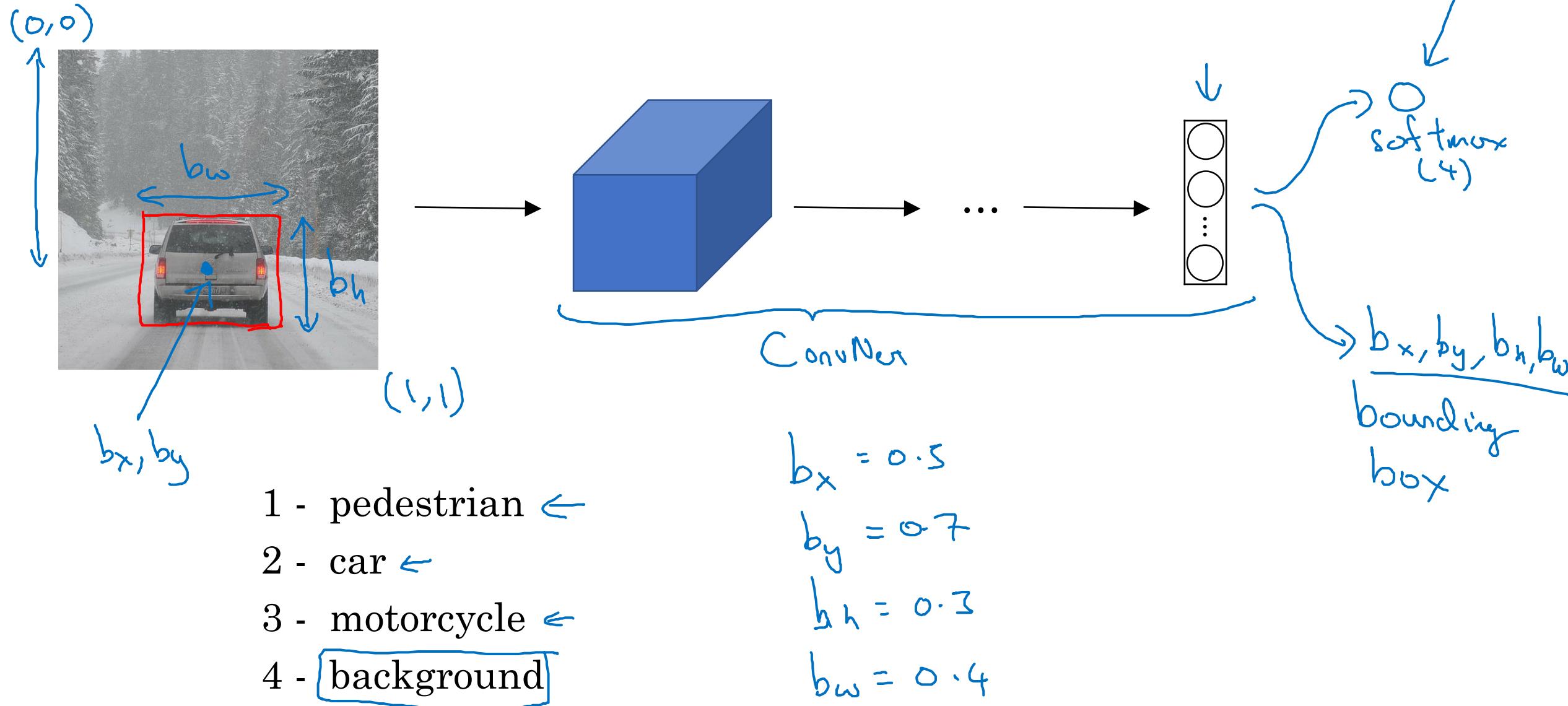
Detection



multiple  
objects

A blue bracket groups the words "multiple objects" below the image.

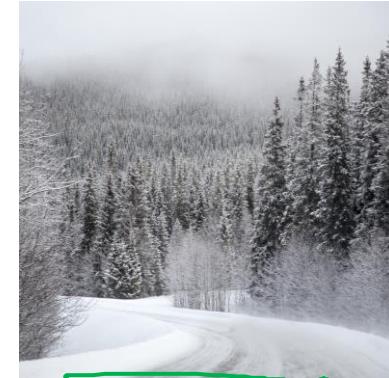
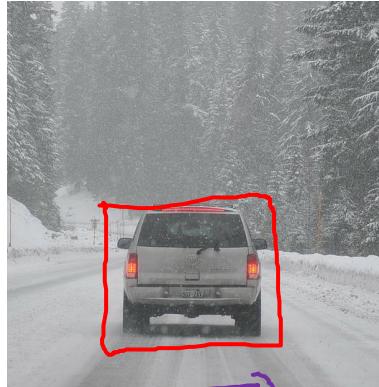
# Classification with localization



# 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)

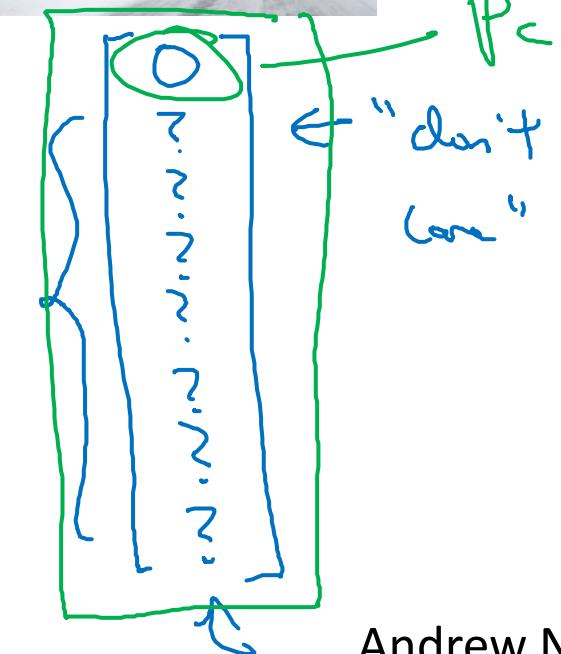
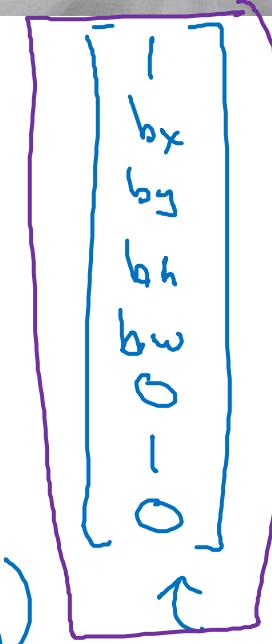


$x =$

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

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

$(x, y)$



Andrew Ng



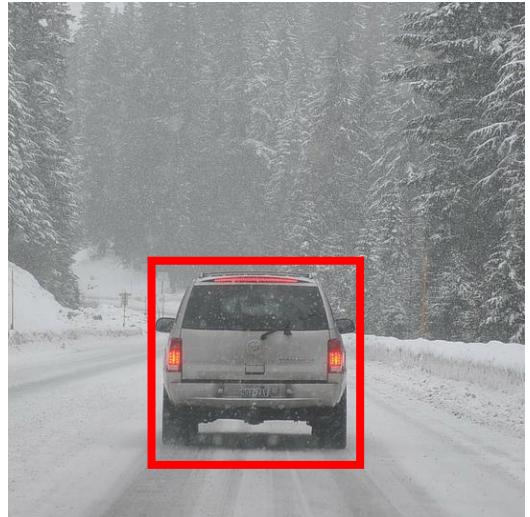
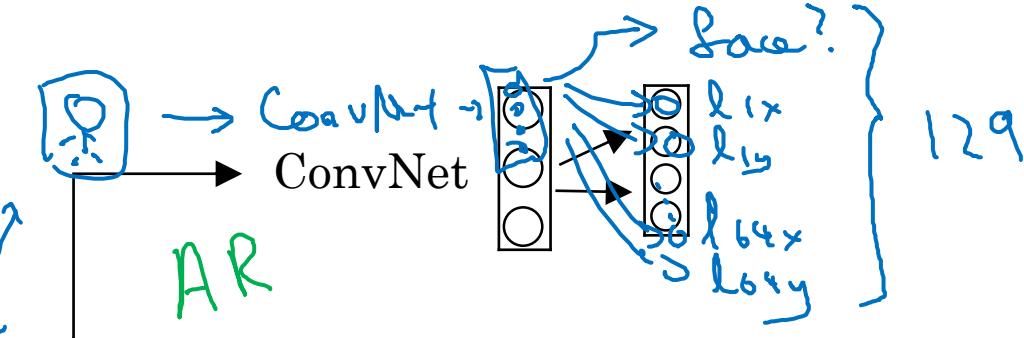
deeplearning.ai

# Object Detection

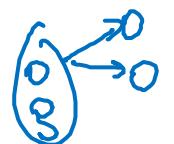
---

## 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

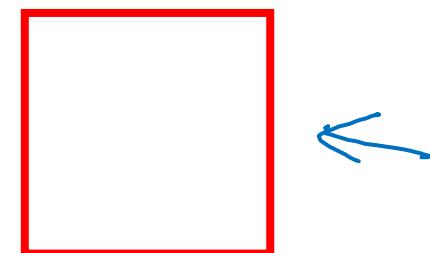
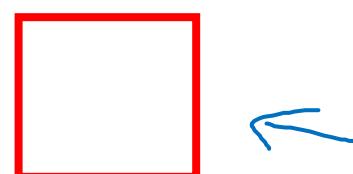
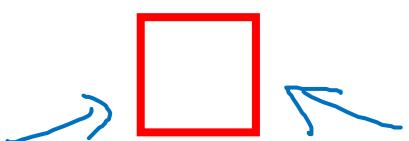
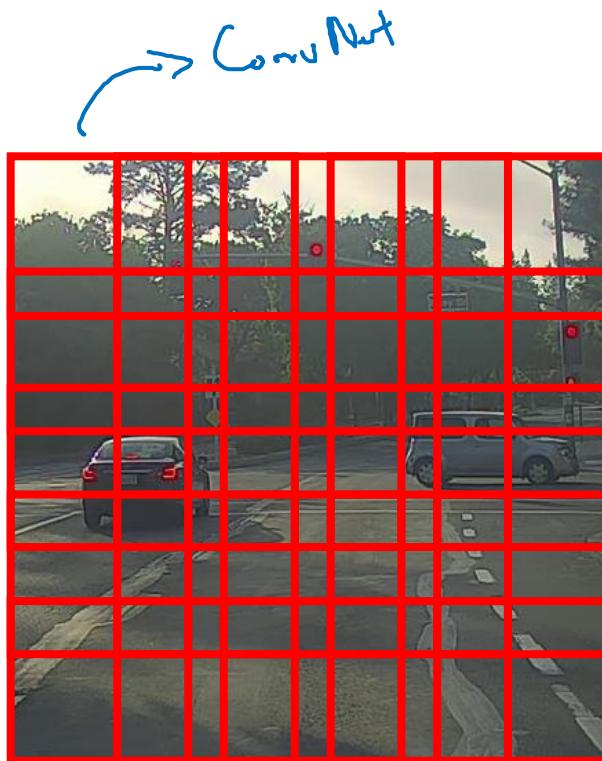
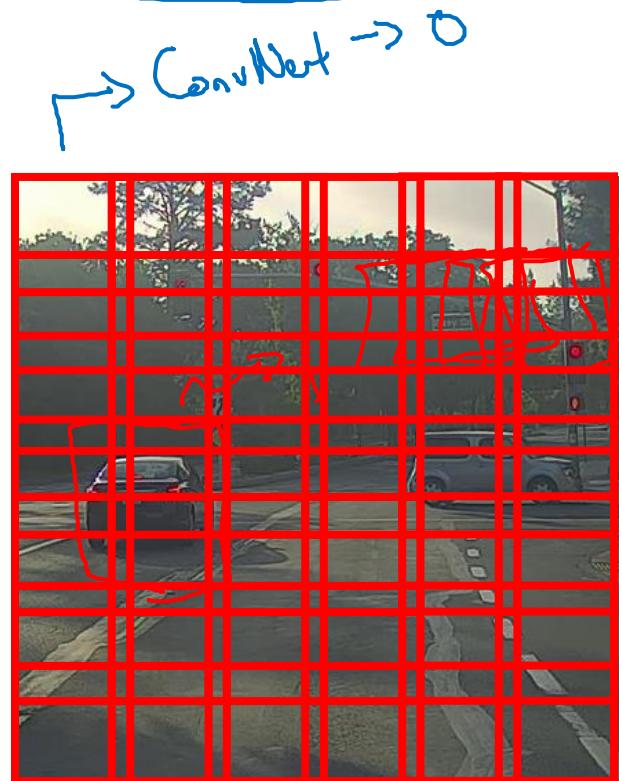
0

0



→ ConvNet → y

# Sliding windows detection



Computation cost



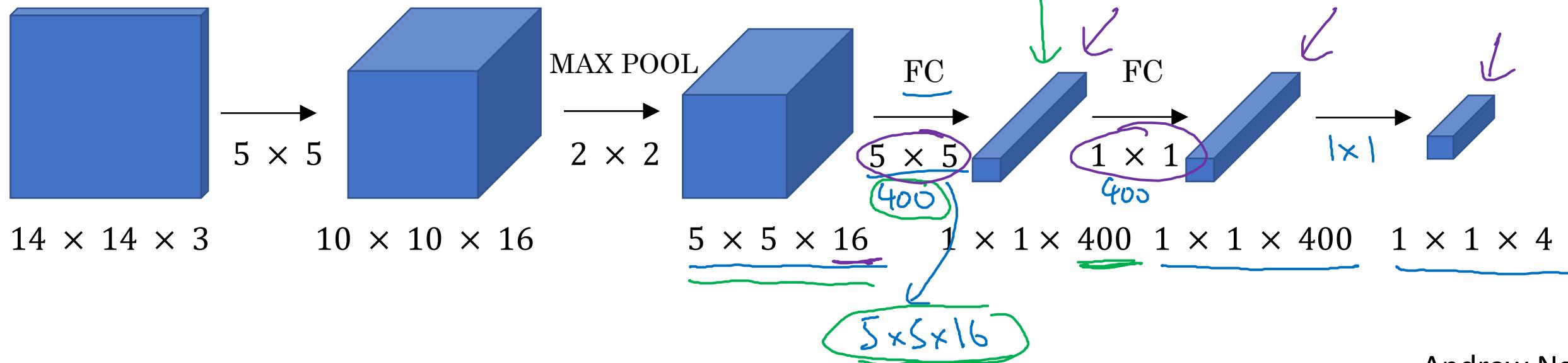
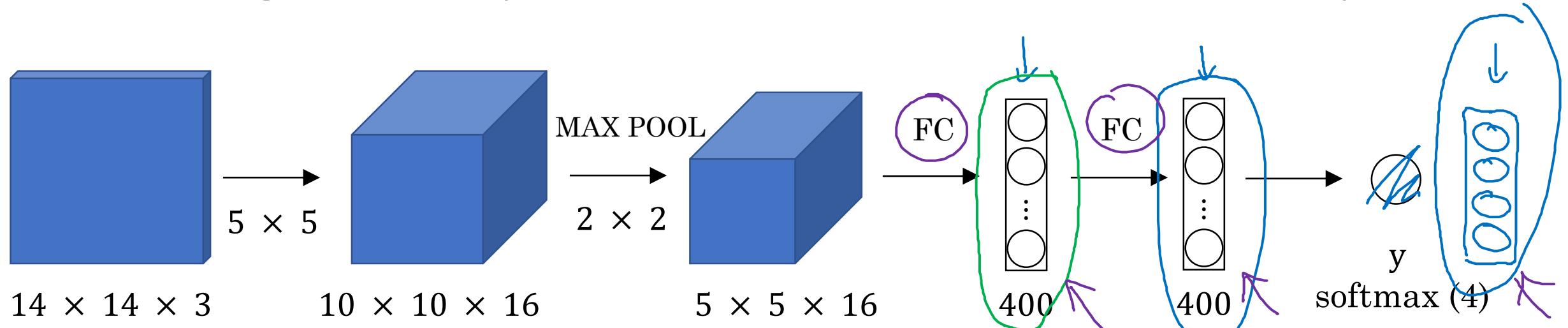
deeplearning.ai

# Object Detection

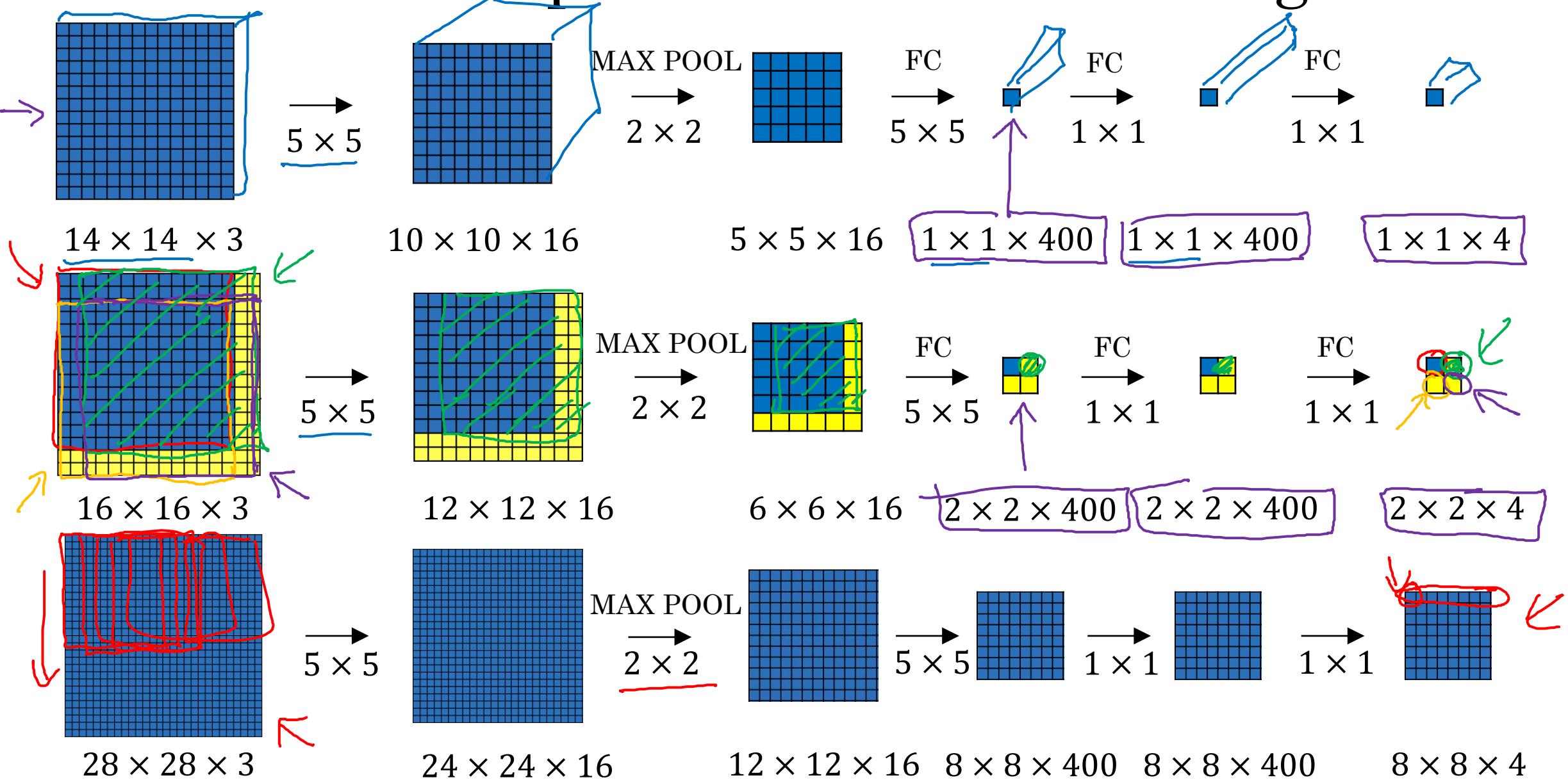
---

## Convolutional implementation of sliding windows

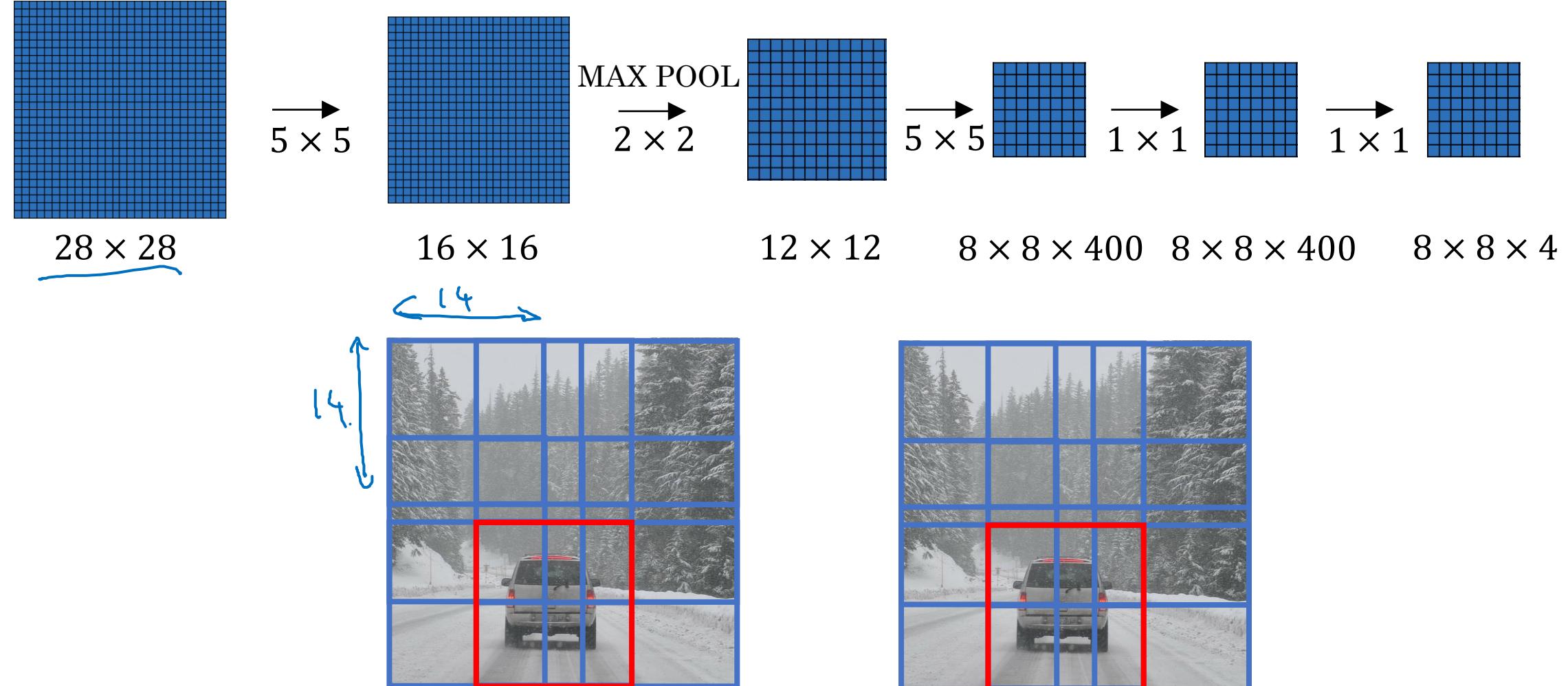
# Turning FC layer into convolutional layers



# Convolution implementation of sliding windows



# Convolution implementation of sliding windows





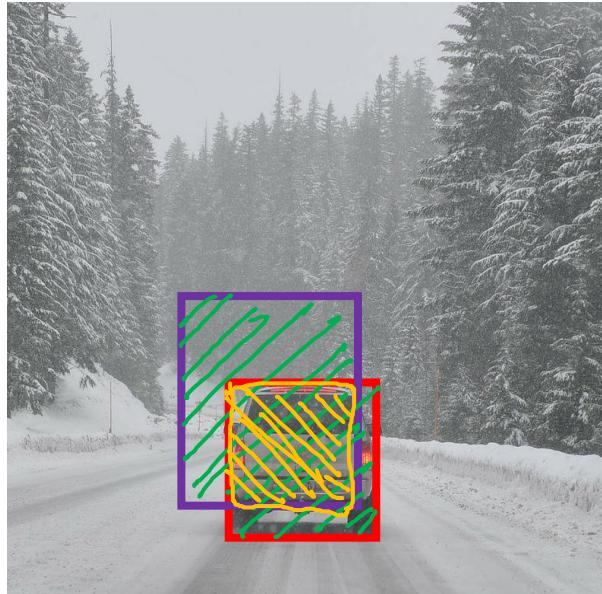
deeplearning.ai

# Object Detection

---

Intersection  
over union

# Evaluating object localization



Intersection over Union (IoU)

$$= \frac{\text{Size of intersection}}{\text{Size of union}}$$
A diagram showing two overlapping rectangles. The top rectangle is yellow with diagonal hatching, and the bottom one is green with horizontal hatching. Their overlapping area is shaded with both patterns, representing the intersection. The non-overlapping parts of each rectangle are also shaded, representing the 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

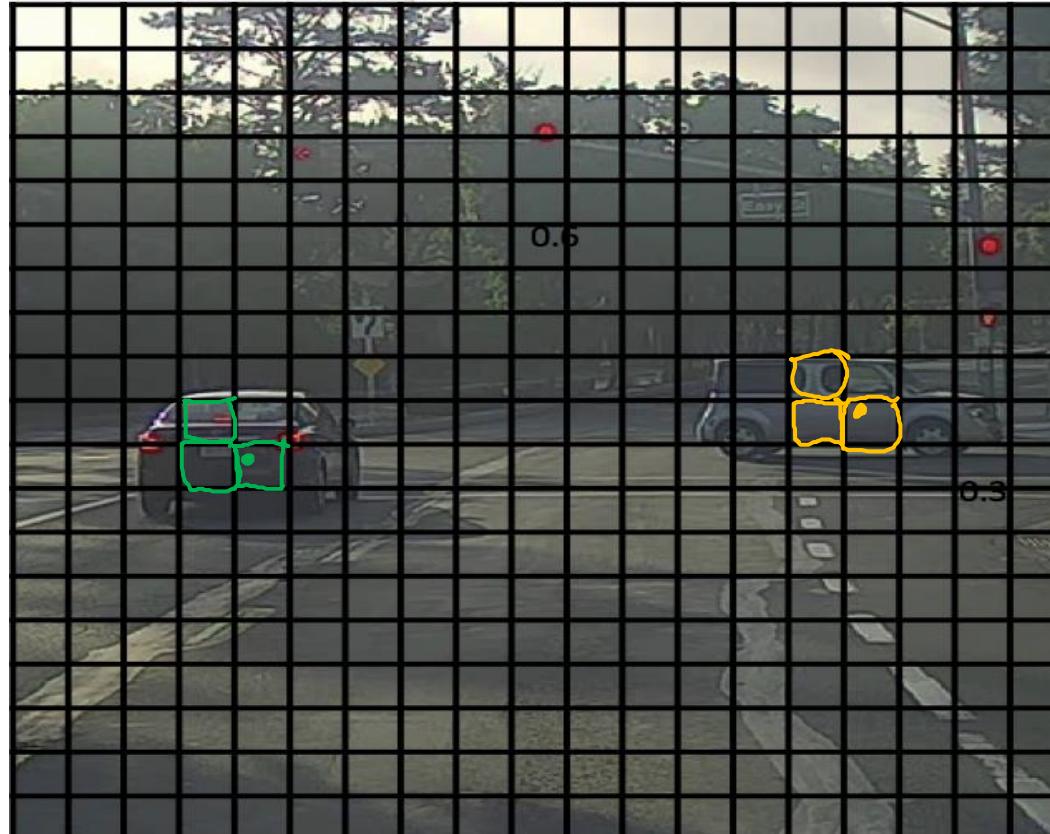
---

Non-max  
suppression

# Non-max suppression example

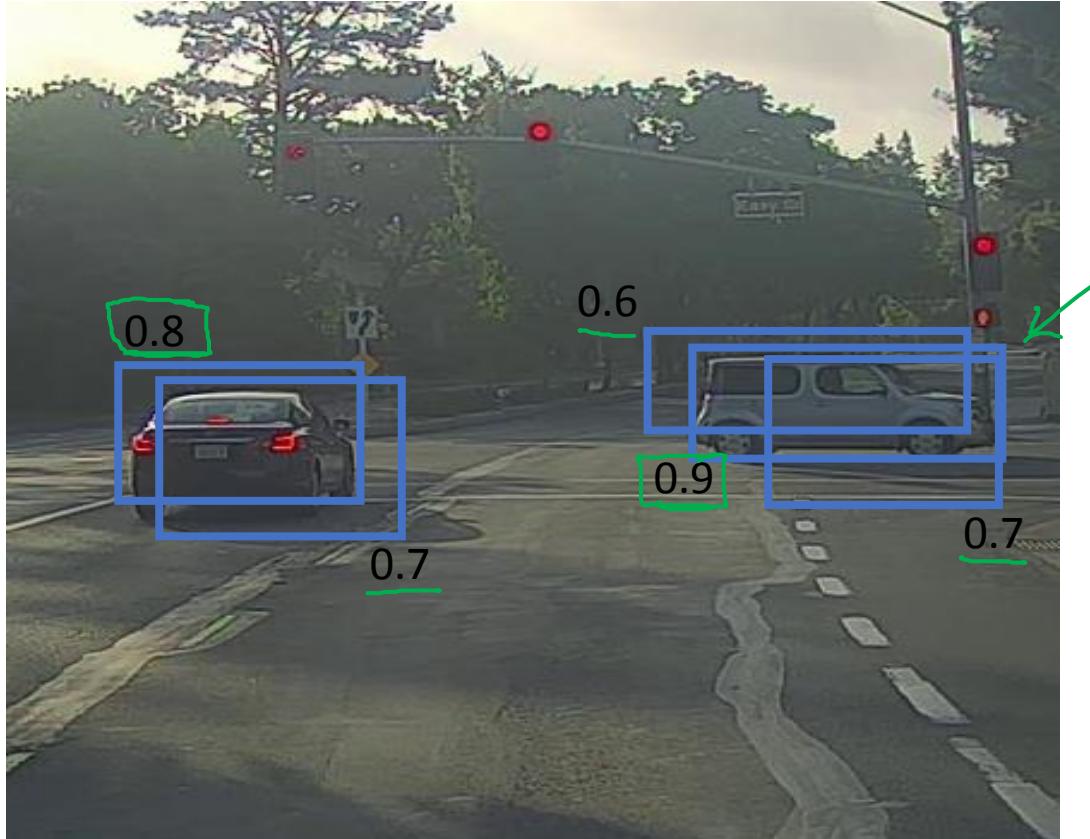


# Non-max suppression example

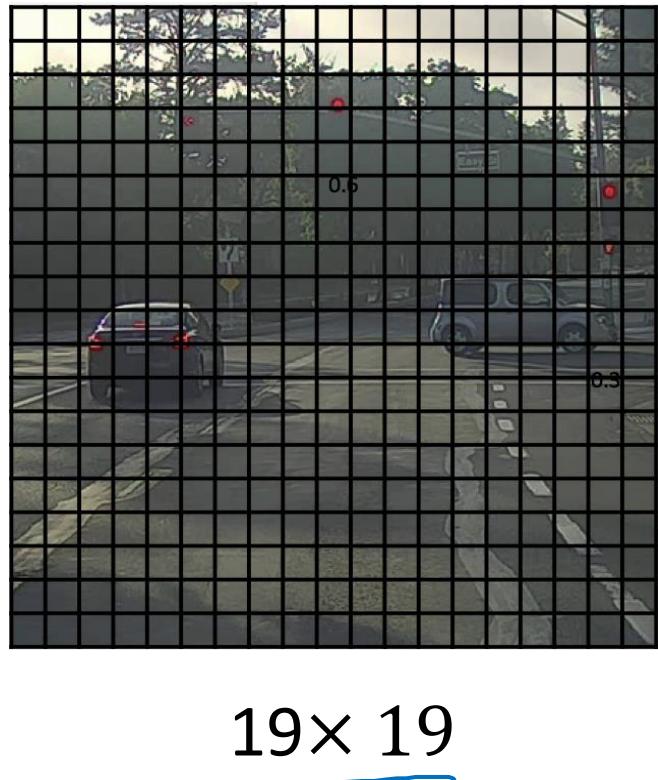


19x19

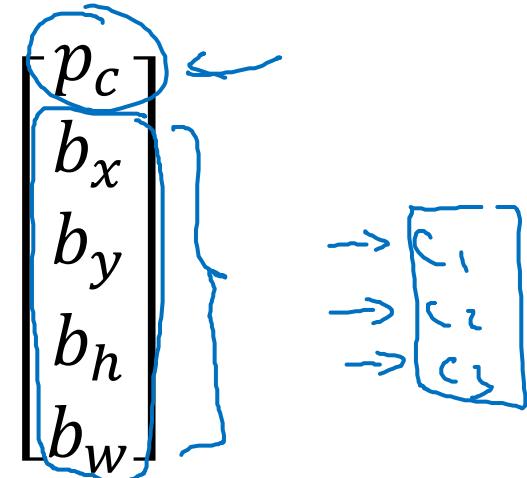
# Non-max suppression example



# Non-max suppression algorithm



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



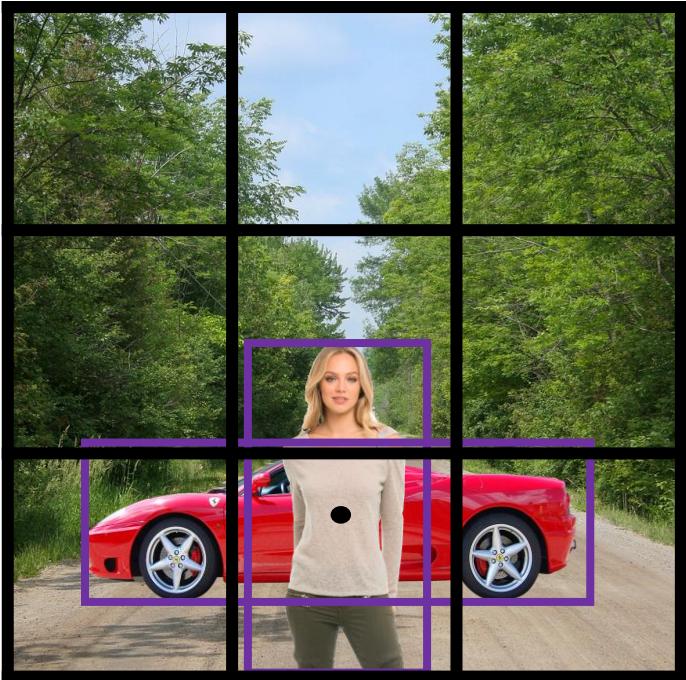
deeplearning.ai

# Object Detection

---

## Anchor boxes

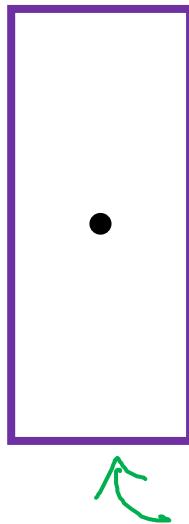
# Overlapping objects:



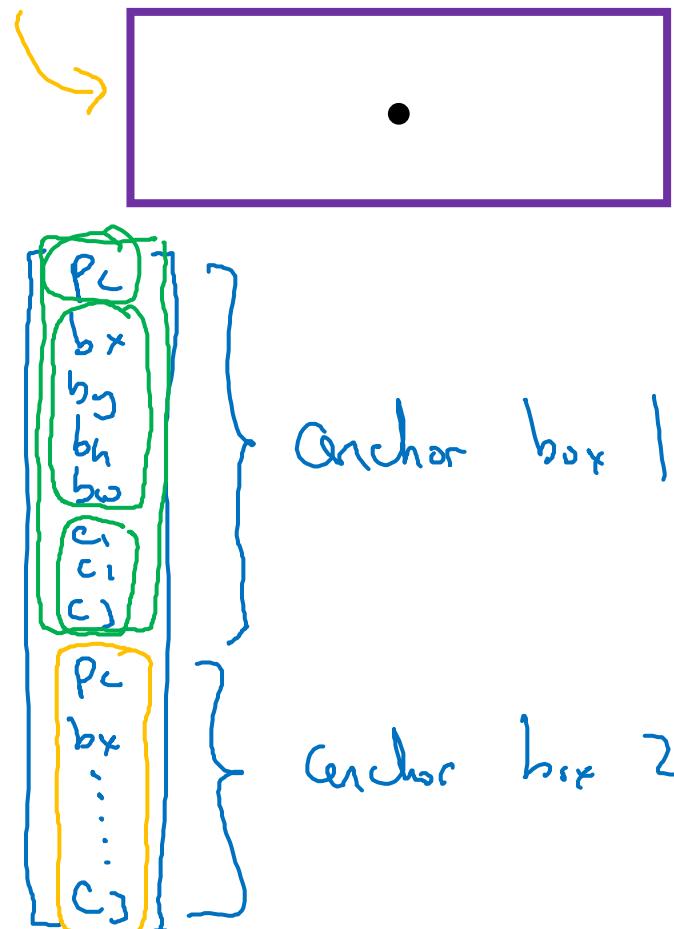
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

A green bracket above the first four columns indicates they are grouped together. A blue bracket below the last four columns indicates they are grouped together.

Anchor box 1:



Anchor box 2:



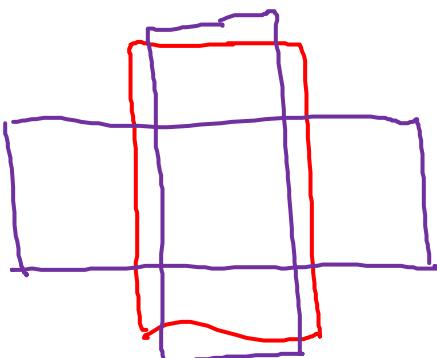
# Anchor box algorithm

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.

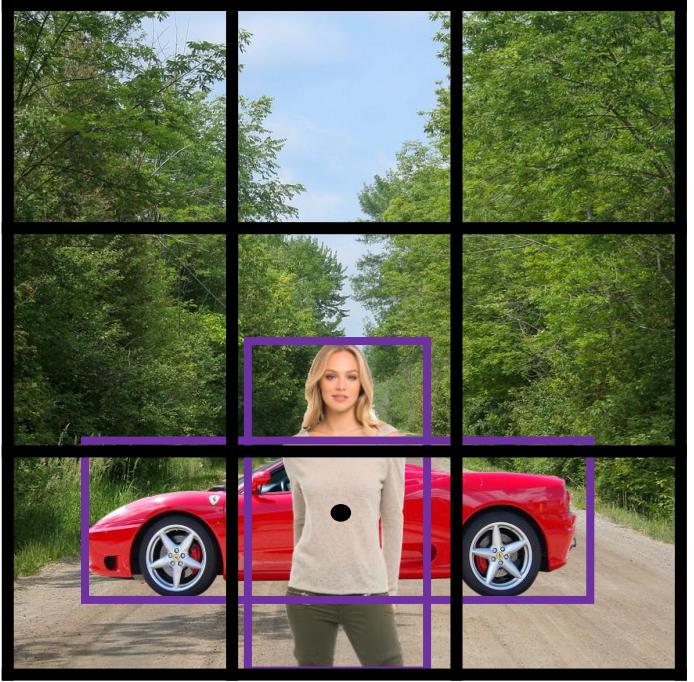
(grid cell, anchor box)

Output y:

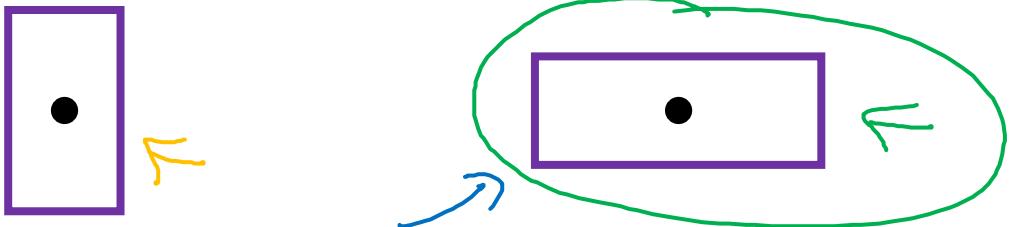
$3 \times 3 \times 16$

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

# Anchor box example

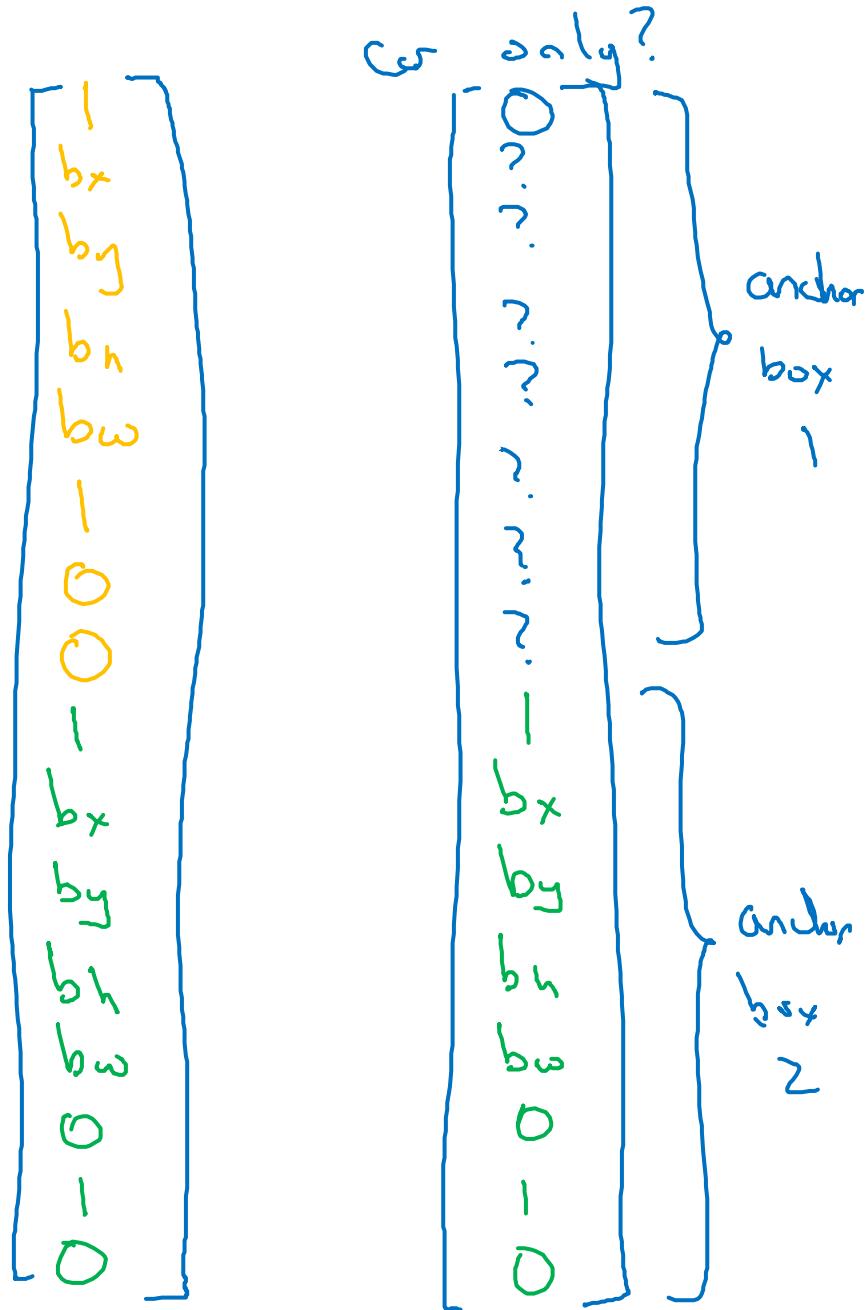


Anchor box 1:    Anchor box 2:



$$y =$$

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





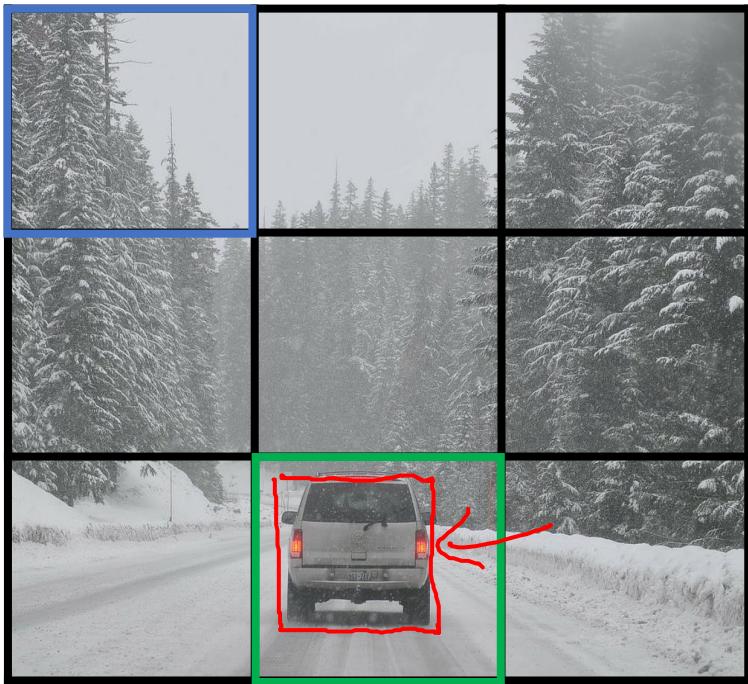
deeplearning.ai

# Object Detection

---

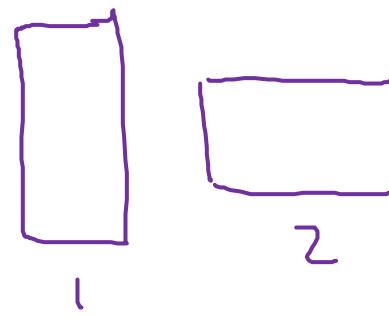
Putting it together:  
YOLO algorithm

# Training



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

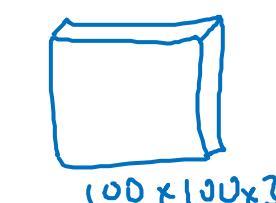
$y =$



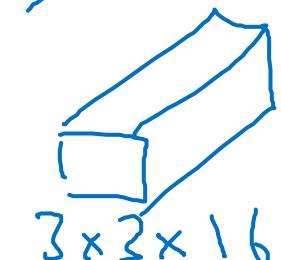
$y$  is  $3 \times 3 \times 2 \times 8$

$3 \times 3 \times 16$  (circled)  
 $19 \times 19 \times 16$   
 $19 \times 19 \times 40$

$\uparrow$  #anchors     $\uparrow$   $5 + \# \text{classes}$



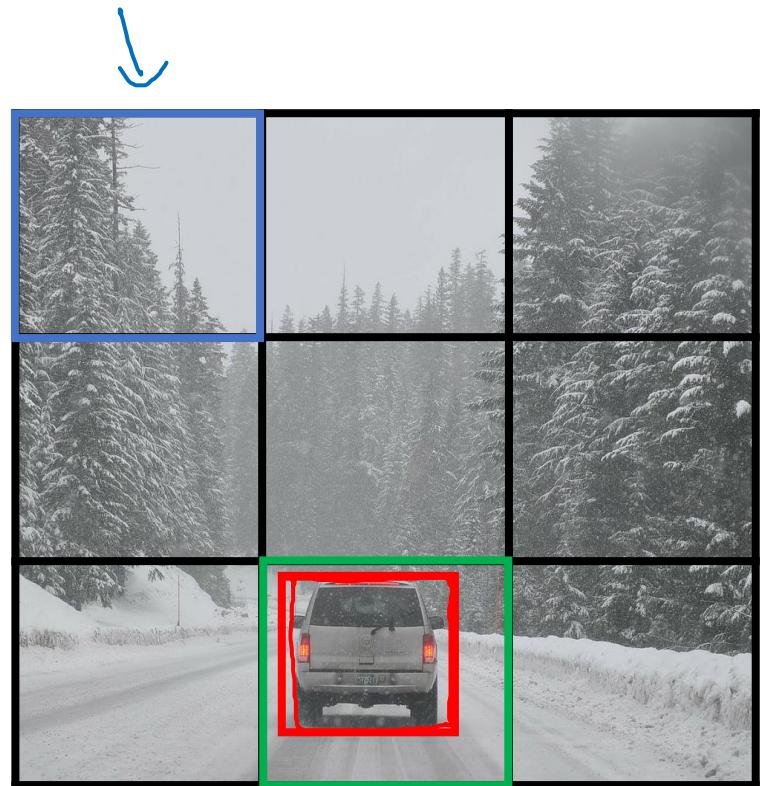
$\rightarrow$  ConvNet



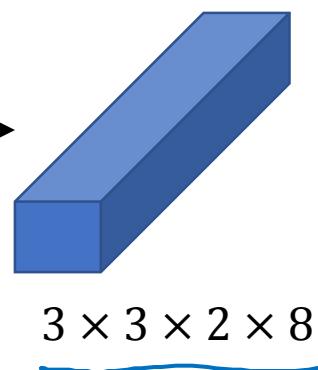
[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

Andrew Ng

# Making predictions



...



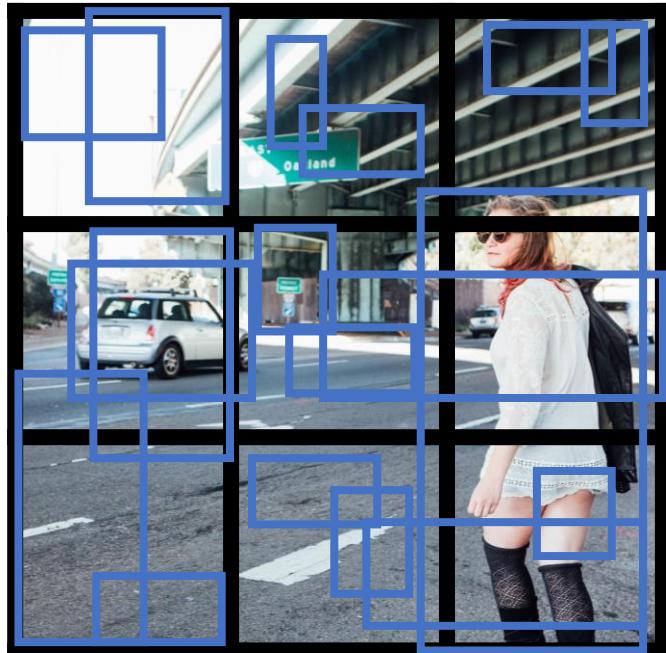
$$y =$$

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

Handwritten annotations are present on the right side of the output vector:

- Blue arrows point to the first four elements ( $p_c, b_x, b_y, b_h$ ) and the last three elements ( $c_1, c_2, c_3$ ) of the vector.
- Red arrows point to the fifth element ( $b_w$ ) and the second element ( $b_x$ ) of the vector.
- A green arrow points to the third element ( $b_y$ ) of the vector.
- A blue arrow points up from the bottom of the vector to the underlined dimension 8.

# 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.



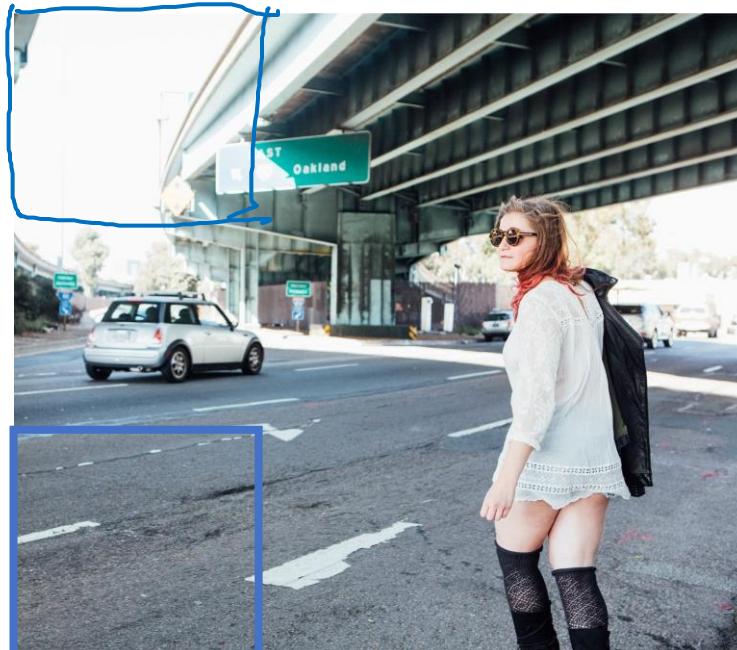
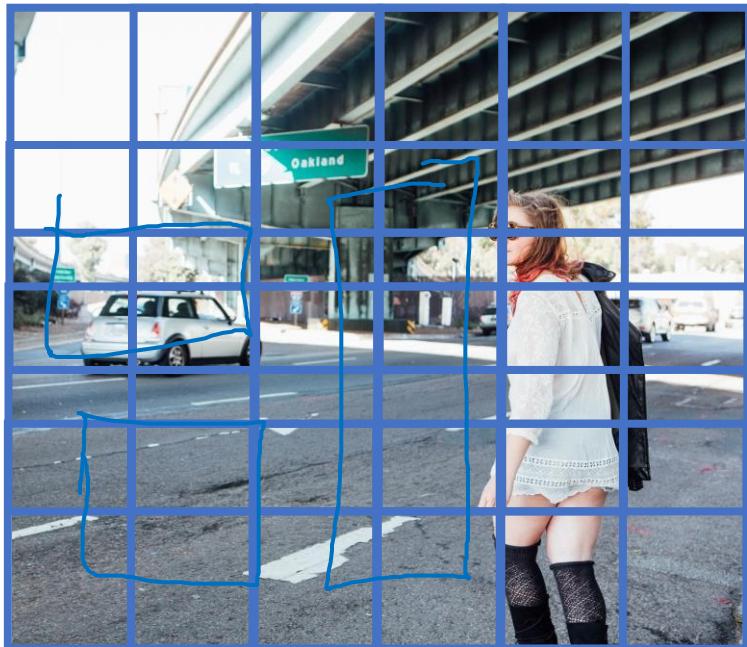
deeplearning.ai

# Object Detection

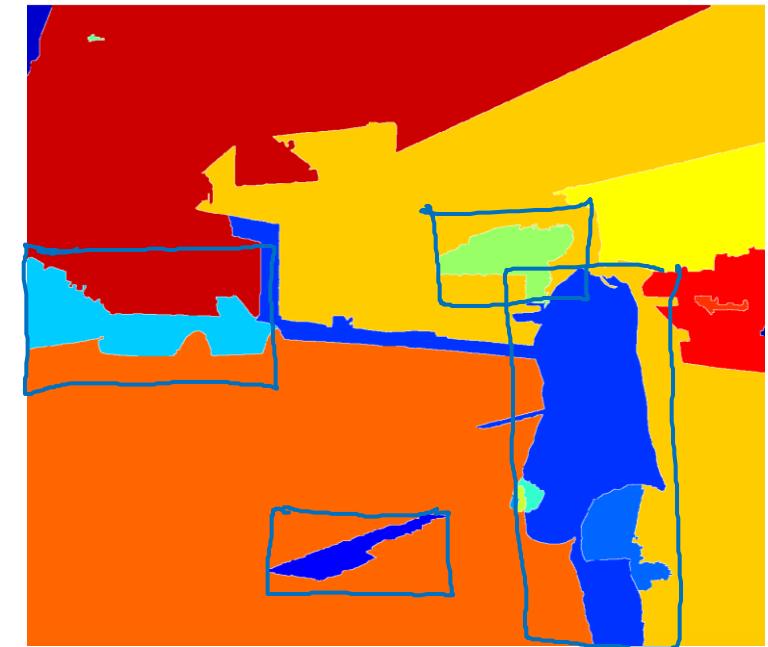
---

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