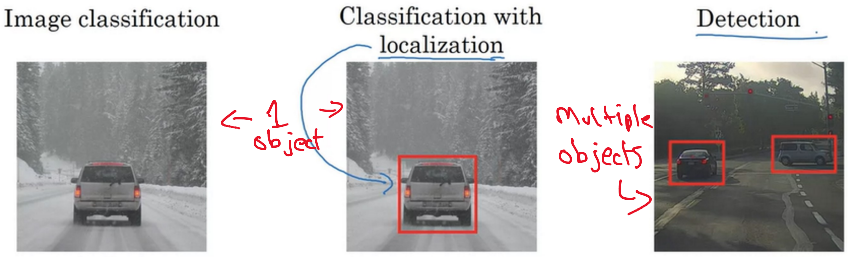
Week 3 - Detection Algorithms

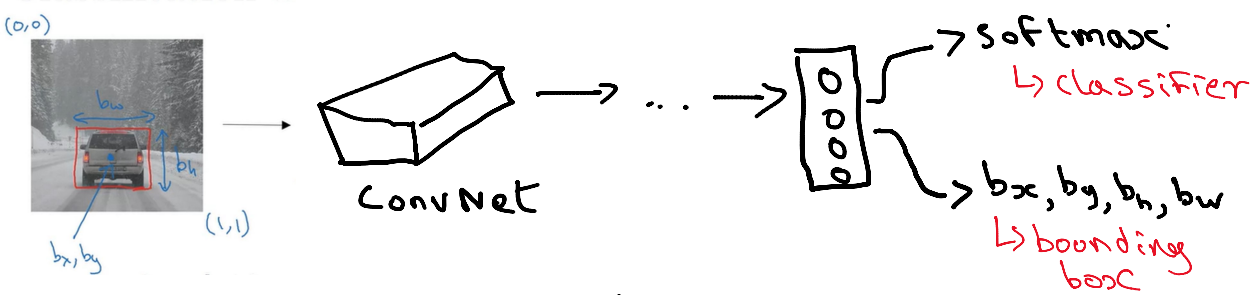
Wednesday, September 9, 2020 11:52 AM

① Object localization

→ localization and detection



→ Classification with localization



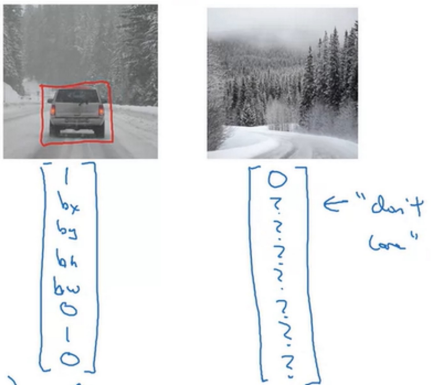
→ Defining the target label y

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

Need to output b_x, b_y, b_h, b_w, class label [1-4]

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

P_c: 1 if any object detected, 0 if background



- $\mathcal{L}(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \dots + (\hat{y}_8 - y_8)^2 & (\text{if } y_1 = 1) \\ (\hat{y}_1 - y_1)^2 & (\text{others don't matter})(\text{if } y_1 = 0) \end{cases}$

② Landmark detection

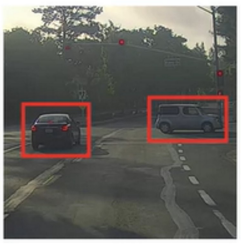
- Sometimes, we might need model to output landmarks (points)
- eg: In face recognition we might need points on the face like eye corners,

- eg: In face recognition we might need points on the face like eye corners, etc.
- Output is $y = [$ thereIsAFace (0 or 1)
  $l1_x, l1_y$
  $l2_x, l2_y$
  $l3_x, l3_y$
  ⋮
  $l64_x, l64_y$ $]$

For 64 points, we have 129 values in y (128 for coordinates, 1 boolean)

③ Object detection

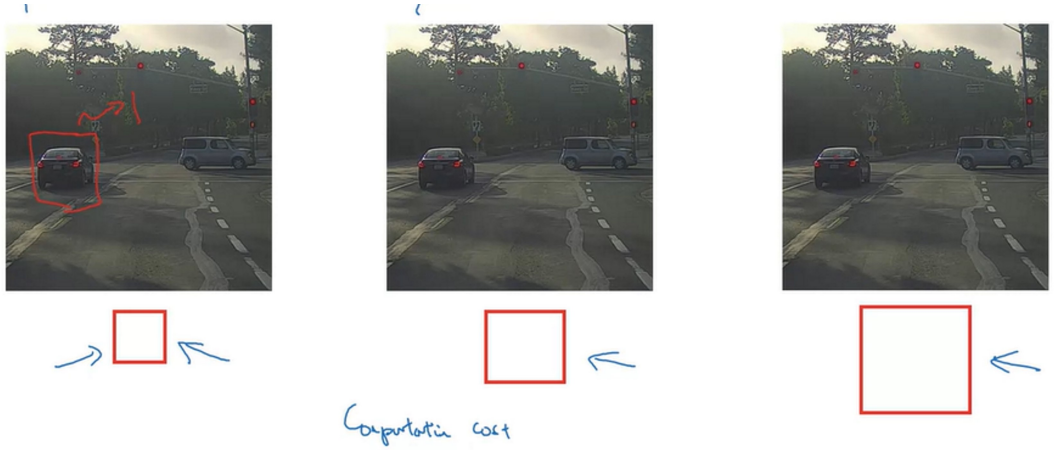
- Sliding windows Algorithm
  → For eg: Car detection
  - Training is done on cropped images of cars using ConvNet



Training set:

x	y
	1
	1
	1
	0
	0

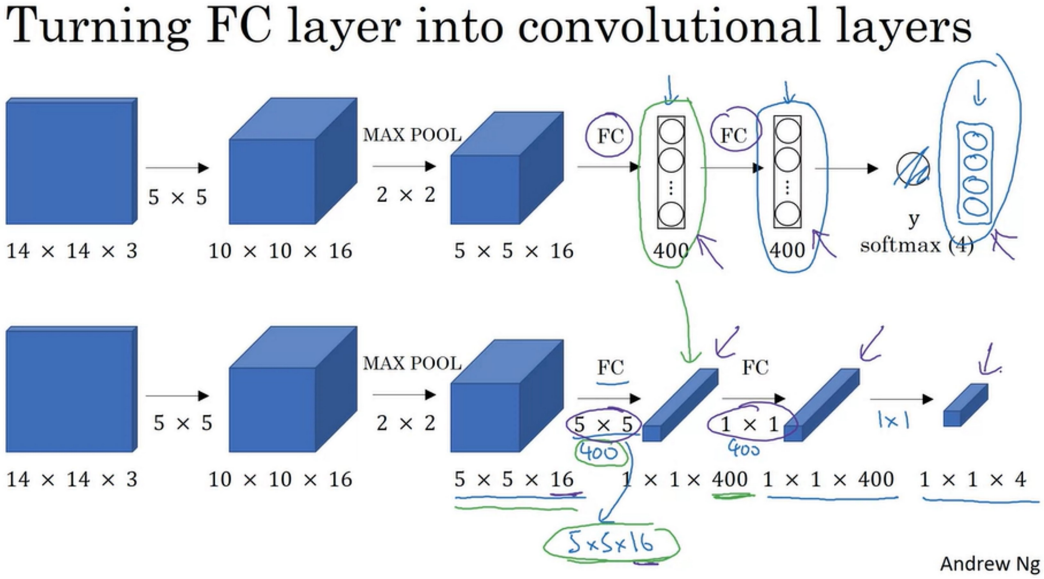
- On test image, we use a small window to pass every part of the image to check if there is a car
- We then change window size and pass again
- store the rectangles in which cars were detected



- Disadvantage is computation time
  - Can be solved by implementing convolutional approach
  - We can also compress our model to save computation

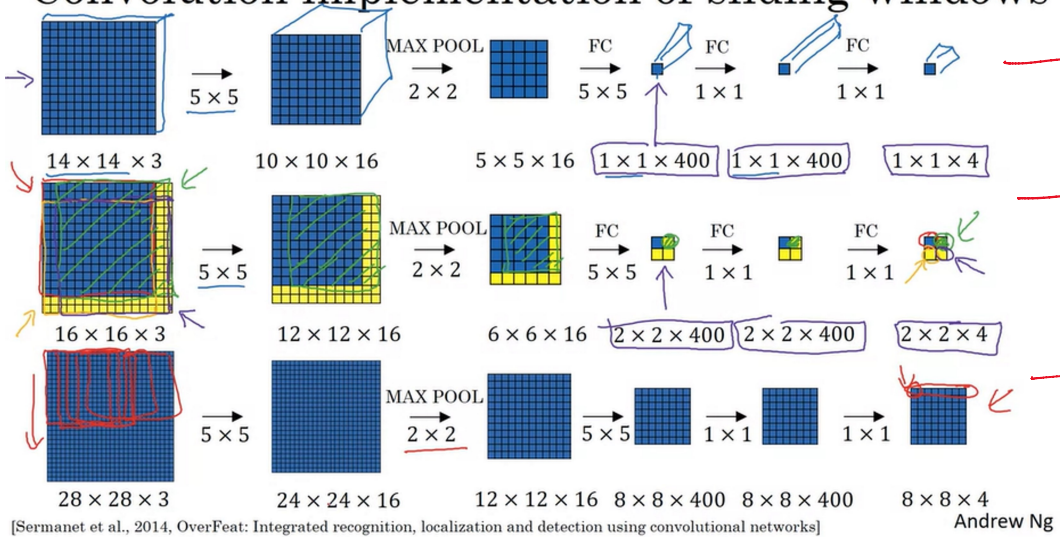
④ Convolutional implementation of Sliding Windows

- Turning FC layer into conv. layer



- Convolution implementation of Sliding windows

Convolution implementation of sliding windows



[Sermanet et al., 2014, OverFeat: Integrated recognition, localization and detection using convolutional networks]

Andrew Ng

- we just feed the image directly to the convnet
- we get output for each window
- this improves efficiency greatly because the computations are shared
- weakness is that the position of bounding box isn't that accurate

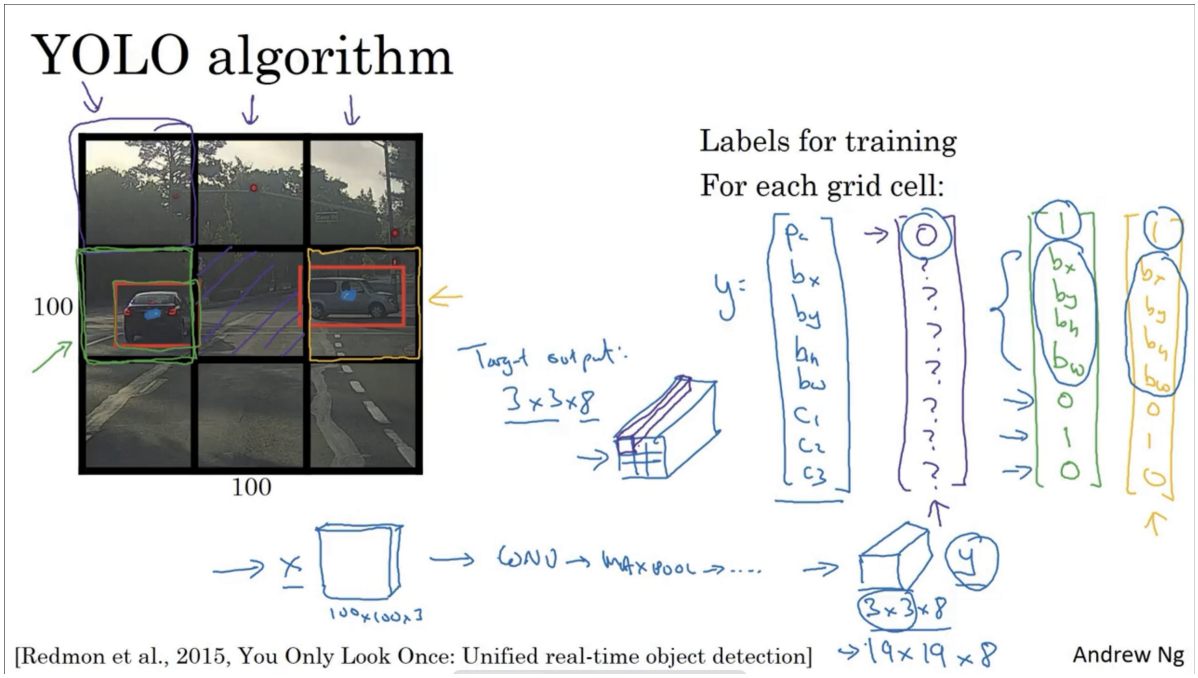
(5) Bounding box prediction

- YOLO algorithm

→ If image is 100x100 (x3)
→ Place a 3x3 grid (for finer results use larger grid size like 19x19)

→ Apply classification and localization algorithm to each grid
bx & by represents center of object & bh & bw will give height & width

→ Output Y → 3x3x8
(3x3 → no. of grids; 8 → output for each grid)



→ One downside is when there is more than one object in one grid box

→ It is very fast since it is a convolutional implementation

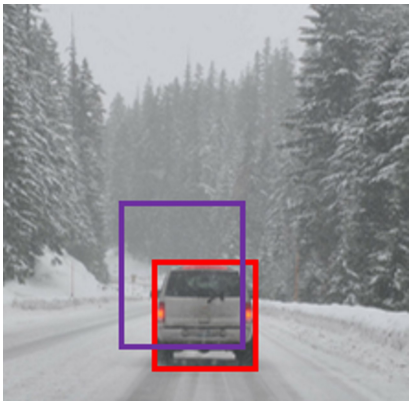
→ Outputs explicit bounding boxes (not dictated by grid size)

- Specifying the bounding boxes

$$y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \begin{matrix} \rightarrow \text{bool} \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} \text{0 to 1} \\ \hookrightarrow \text{relative to grid} \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} \text{in pixels} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{bool} \end{matrix}$$

⑥ Intersection over Union (IoU)

- Evaluating object localization



$$\text{IoU} = \frac{\text{size of intersection}}{\text{size of union}}$$

Prediction is "correct" if

$\text{IoU} \geq 0.5$

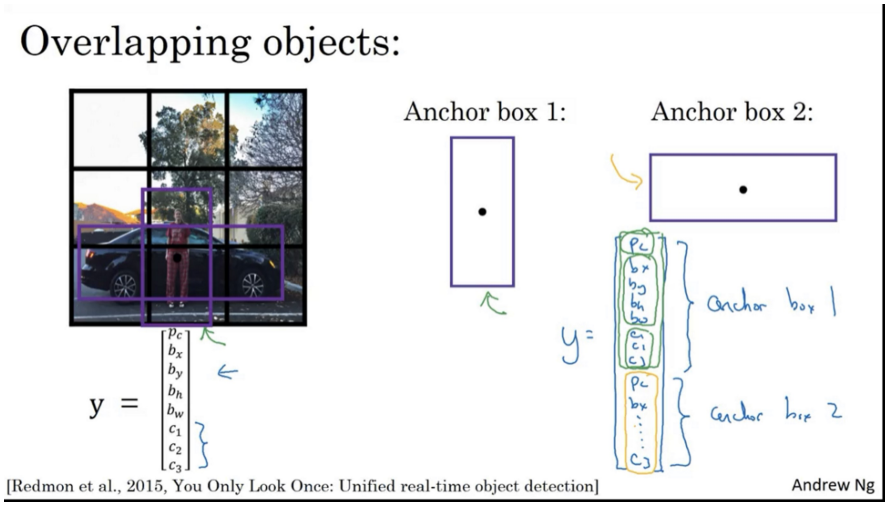
- It is a measure of overlap between 2 bounding

- It is a measure of overlap between 2 bounding boxes.
- The higher the IoU the better is the accuracy

(7) Non-Max suppression

- Each object may be detected two or more times with different probabilities
- Non-max Suppression is used to eliminate possible duplicates
  - i. for this case, assume output is just one class
  - ii. Y shape is $[P_c, b_x, b_y, b_h, b_w]$ where P_c is the probability of the object occurring
  - iii. Discard all boxes with $P_c < 0.6$
  - iv. For the remaining boxes
    - → Pick box with largest P_c
    - → compute IoU with other boxes and the one picked above and discard anything with IoU > 0.5
- If we output multiple classes, we should run Non-max suppression once for every output class

(8) Anchor boxes

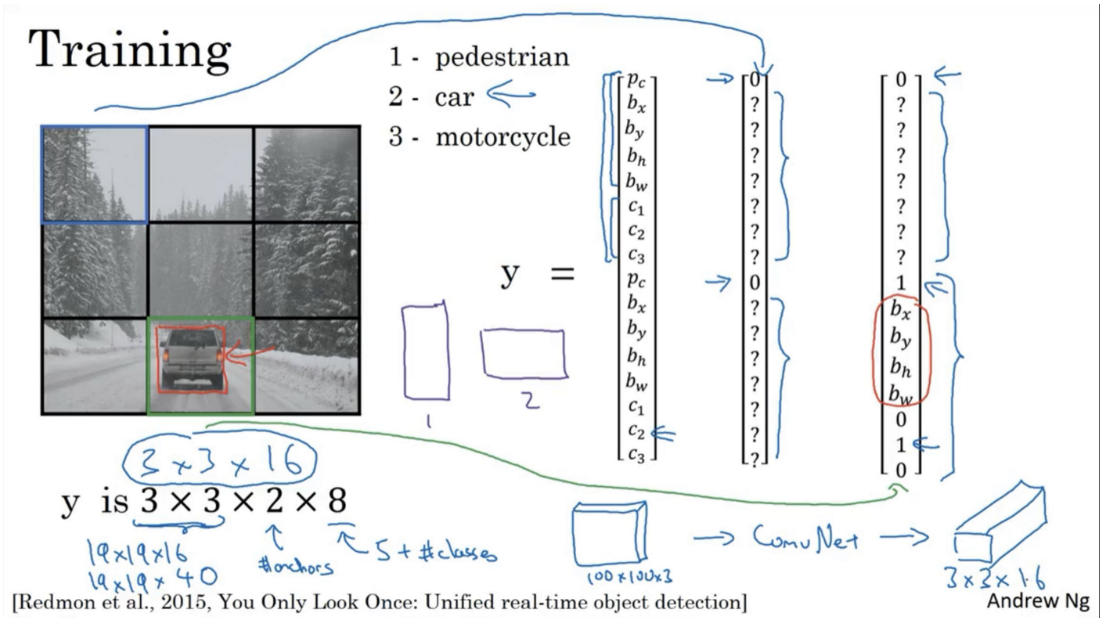


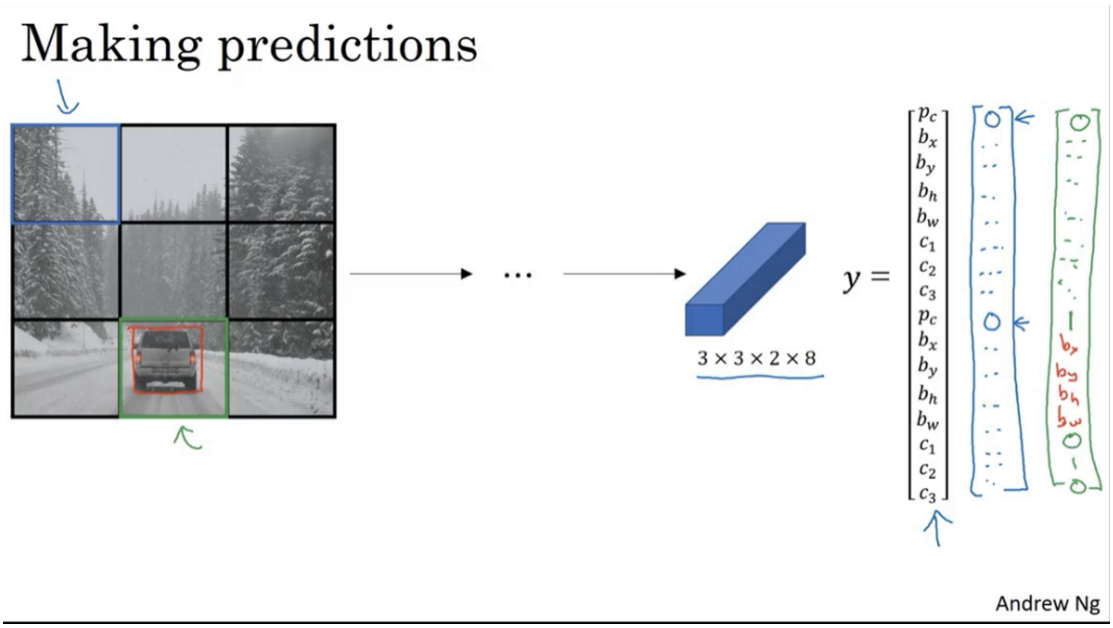
- each object is assigned to grid cell that contains the object's midpoint and additionally anchor box for the grid cell with highest IoU
  - output is 3X3X16 (or 3X3X2X8) as 2 anchor boxes are used
- Doesn't work well if we have 2 anchor boxes

2 anchor boxes are used

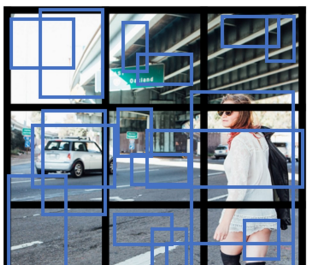
- Doesn't work well if we have 2 anchor boxes but 3 objects or if we have 2 objects with same anchor box shape.

⑨ YOLO Algorithm



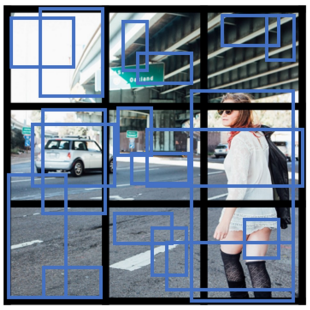


- Making Predictions
  - → The ConvNet always outputs 2 boxes for each grid even if no object is present



Total no of generated boxes

= grids width * grid height * no. of anchors



Total no of generated boxes

= grids width * grid height * no. of anchors

→

- After removing low probability predictions

→

- Finally, we apply non-max suppression to get the best output

⑩ Region Proposals

- R-CNN

Region proposal: R-CNN

Segmentation algorithm

~2,000

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

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]

- Still slower than YOLO