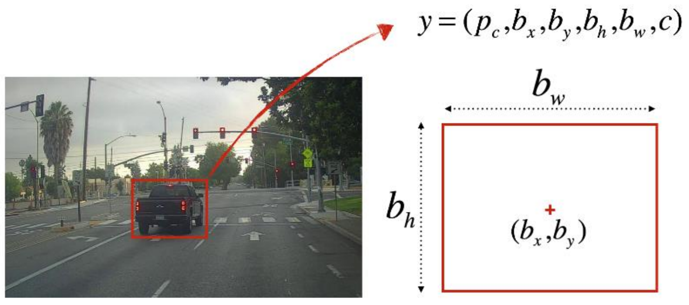
Convolutional neural networks – week 3: Car detection with YOLO

Model details

Input: The input is a batch of images, and each image has the shape (m, 608, 608, 3).

Output: The output is a list of bounding boxes along with the recognized classes. Each bounding box is represented by 6 numbers $(p_c, b_x, b_y, b_h, b_w, c)$. Label c is either an integer from 1 to 80 (80 classes), or an 80-dimensional vector. If you expand c into an 80-dimensional vector, each bounding box is then represented by 85 numbers.

Definition of the bounding box:

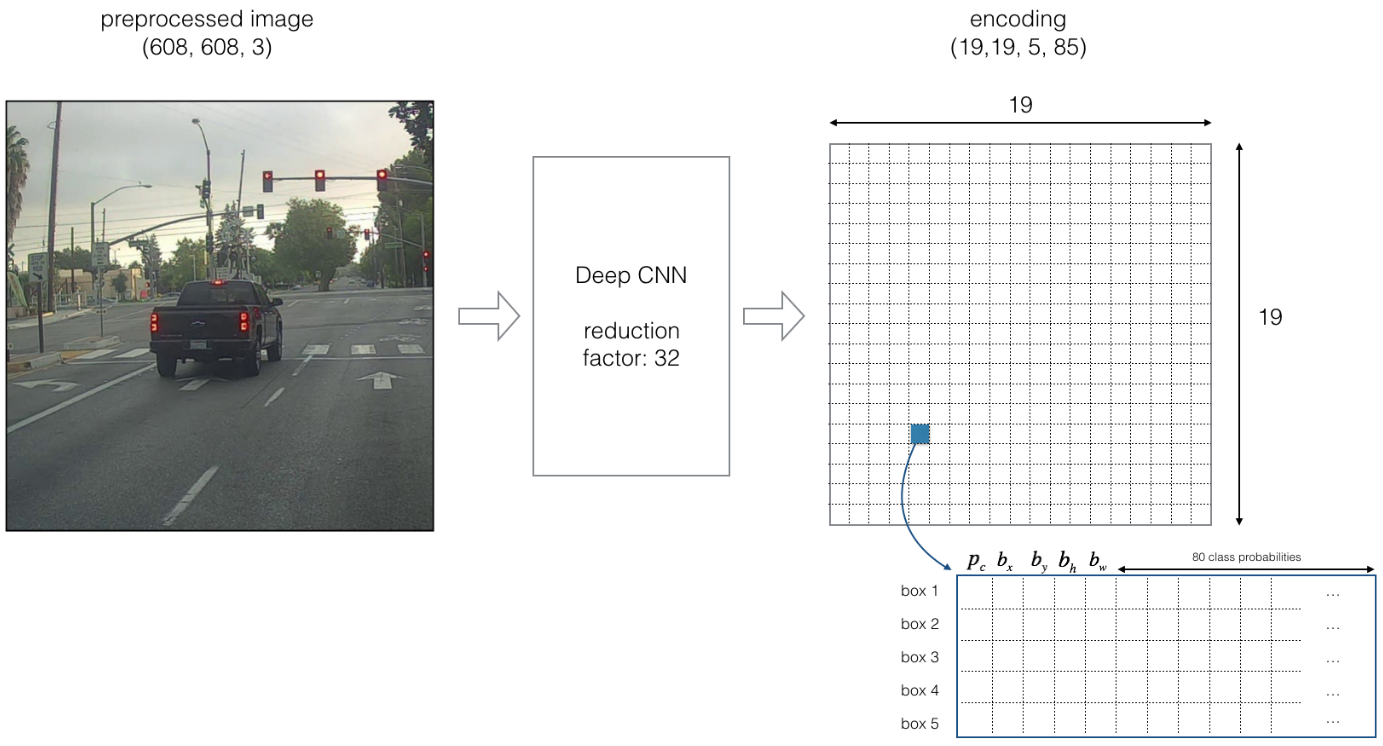


$p_c = 1$: confidence of an object being present in the bounding box
$c = 3$: class of the object being detected (here 3 for "car")

Anchor Boxes

- Anchor boxes are chosen by exploring the training data to choose reasonable height/width ratios that represent the different classes. 5 anchor boxes were chosen (to cover the 80 classes), and stored in the file `'./model_data/yolo_anchors.txt'`
- The dimension for anchor boxes is the second to last dimension in the encoding: $(m, n_H, n_W, anchors, classes)$.
- The YOLO architecture is: IMAGE (m, 608, 608, 3) -> DEEP CNN -> ENCODING (m, 19, 19, 5, 85).

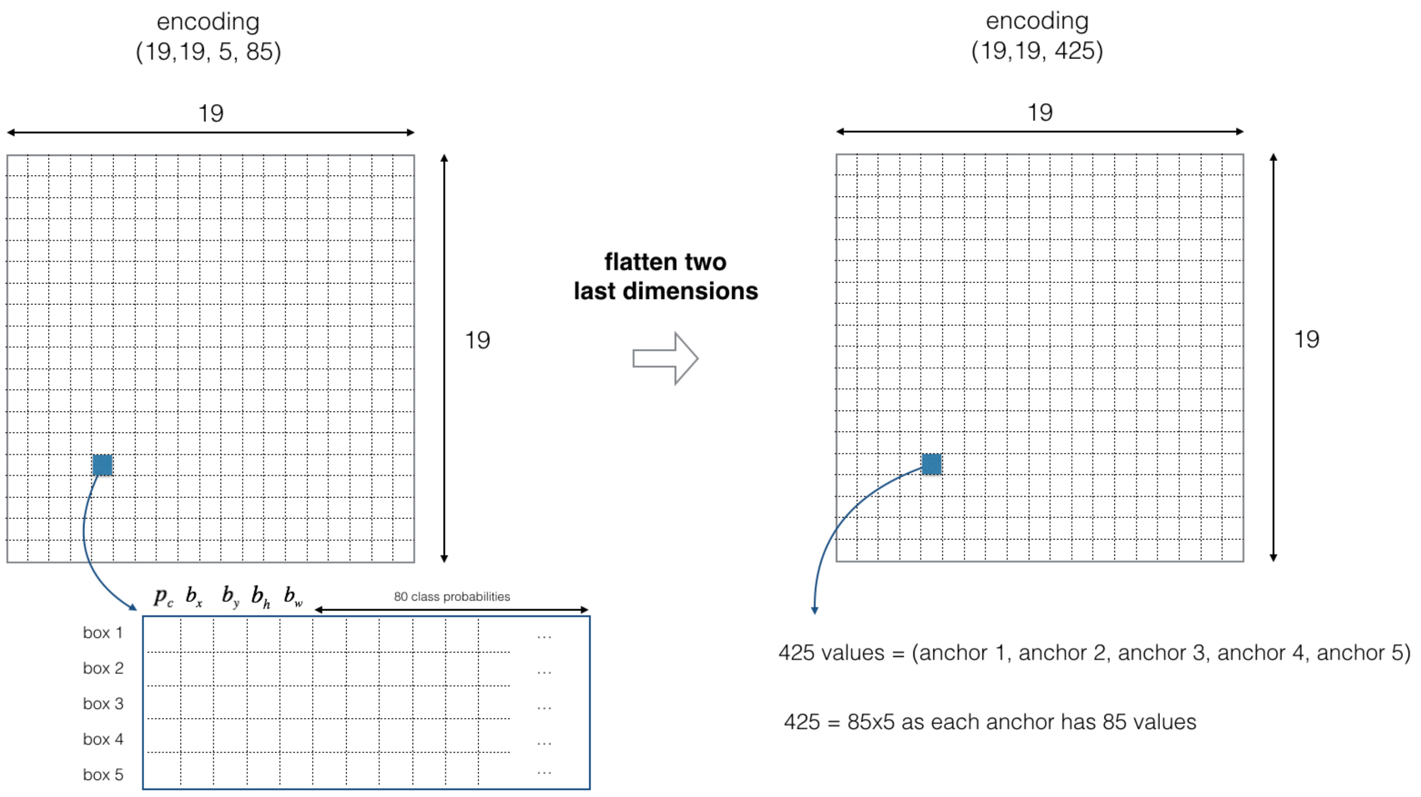
Encoding



If the center/midpoint of an object falls into a grid cell, that grid cell is responsible for detecting that object.

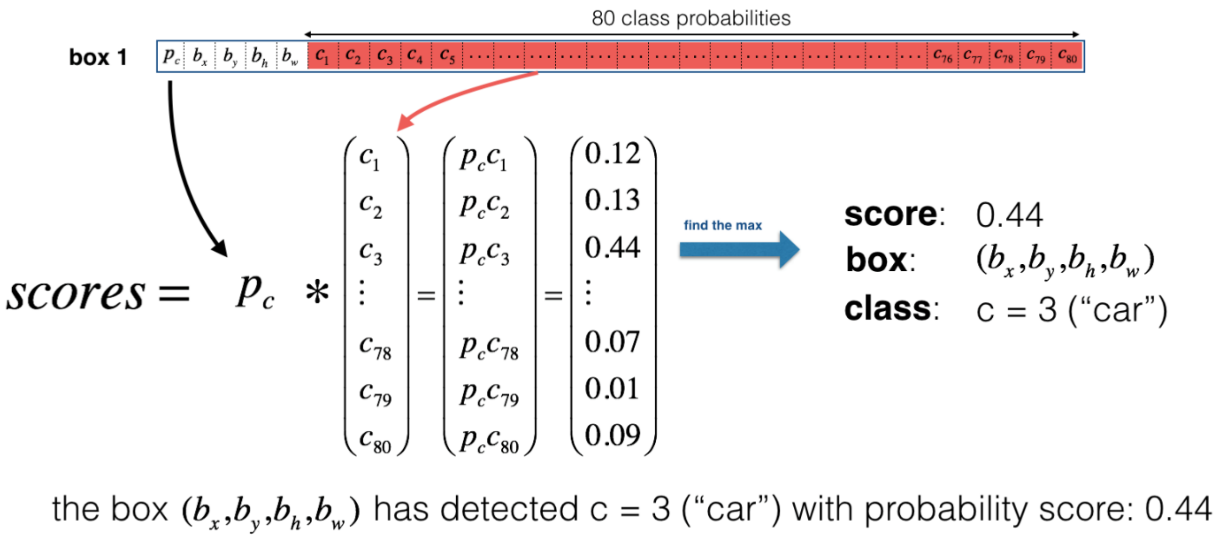
Since we are using 5 anchor boxes, each of the 19x19 cells thus encodes information about 5 boxes. Anchor boxes are defined only by their width and height.

For simplicity, we will flatten the last two last dimensions of the shape (19, 19, 5, 85) encoding, So the output of the Deep CNN is (19, 19, 425).


encoding
(19,19, 5, 85)
19
19
p_c b_x b_y b_h b_w
80 class probabilities
box 1
box 2
box 3
box 4
box 5
flatten two
last dimensions
encoding
(19,19, 425)
19
19
425 values = (anchor 1, anchor 2, anchor 3, anchor 4, anchor 5)
425 = 85x5 as each anchor has 85 values

Class score

For each box (of each cell) we will compute the following element-wise product and extract a probability that the box contains a certain class.
The class score is $score_{c,i} = p_c \times c_i$: the probability that there is an object (p_c) times the probability that the object is a certain class (c_i).


80 class probabilities
box 1
p_c b_x b_y b_h b_w c_1 c_2 c_3 c_4 c_5 … c_{76} c_{77} c_{78} c_{79} c_{80}
$scores = p_c * \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{78} \\ c_{79} \\ c_{80} \end{pmatrix} = \begin{pmatrix} p_c c_1 \\ p_c c_2 \\ p_c c_3 \\ \vdots \\ p_c c_{78} \\ p_c c_{79} \\ p_c c_{80} \end{pmatrix} = \begin{pmatrix} 0.12 \\ 0.13 \\ 0.44 \\ \vdots \\ 0.07 \\ 0.01 \\ 0.09 \end{pmatrix}$
find the max
score: 0.44
box: (b_x, b_y, b_h, b_w)
class: c = 3 (“car”)
the box (b_x, b_y, b_h, b_w) has detected c = 3 (“car”) with probability score: 0.44

Non-Max suppression

- Get rid of boxes with a low score (meaning, the box is not very confident about detecting a class; either due to the low probability of any object, or low probability of this particular class).
- Select only one box when several boxes overlap with each other and detect the same object.

Filtering with a threshold on class scores

By using threshold, we get rid of any box for which the class "score" is less than the chosen threshold.

The model gives you a total of 19x19x5x85 numbers, with each box described by 85 numbers. It is convenient to rearrange the (19,19,5,85) (or (19,19,425)) dimensional tensor into the following variables:

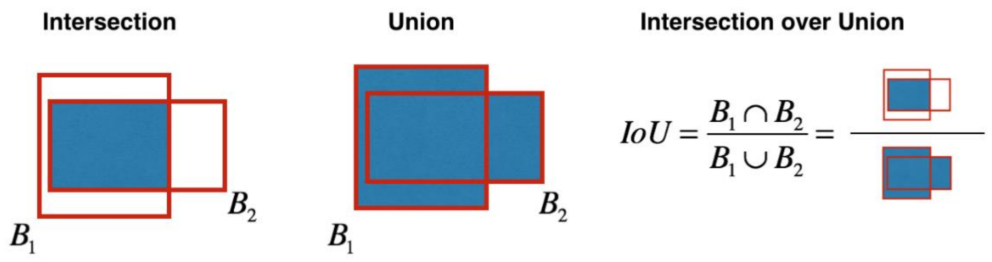
- `box_confidence`: tensor of shape (19×19,5,1) containing p_c (confidence probability that there's some object) for each of the 5 boxes predicted in each of the 19x19 cells.
- `boxes`: tensor of shape (19×19,5,4) containing the midpoint and dimensions $\left(b_x, b_y, b_h, b_w\right)$ for each of the 5 boxes in each cell.
- `box_class_probs`: tensor of shape (19×19,5,80) containing the "class probabilities" $(c_1, c_2, \ldots, c_{80})$ for each of the 80 classes for each of the 5 boxes per cell.

`yolo_filter_boxes():`

1) Compute box scores: $class\ score_{c,i} = p_c \times c_i$
2) For each box find:
    - The index of the class with the maximum box score
    - The corresponding box score.
3) Create a mask by using a threshold. The mask is `True` for boxes you want to keep.
4) Use TensorFlow to apply the mask to `box_class_scores` and `box_classes` to filter out the boxes we don't want.

Non-max suppression

Intersection over Union (IOU) – `iou()`



<u>Convention</u> - (0,0) is the top left corner of an image. (1,0) is the upper-right corner, (1,1) is the lower-right corner.

A box is defined by 2 corners: upper left (x_1, y_1) and lower right (x_2, y_2). The rectangle area: $(y_2 - y_1) \cdot (x_2 - x_1)$.

<u>Finding intersection</u>

- The top left corner of the intersection (xi_1, yi_1) is found by comparing the top left corners (x_1, y_1) of the two boxes and finding a vertex that has an x-coordinate that is closer to the right, and y-coordinate that is closer to the bottom.
- The bottom right corner of the intersection (xi_2, yi_2) is found by comparing the bottom right corners (x_2, y_2) of the two boxes and finding a vertex whose x-coordinate is closer to the left, and the y-coordinate that is closer to the top.
- The two boxes may have **no intersection.** You can detect this if the intersection coordinates you calculate end up being the top right and/or bottom left corners of an intersection box. Another way to think of this is if you

calculate the height $(y_2 - y_1)$ or width $(x_2 - x_1)$ and find that at least one of these lengths is negative, then there is no intersection (intersection area is zero).

- The two boxes may intersect at the **edges or vertices**, in which case the intersection area is still zero. This happens when either the height or width (or both) of the calculated intersection is zero.

YOLO non-max suppression - `yolo_non_max_suppression()`

1) Select the box that has the highest score.
2) Compute the overlap of this box with all other boxes and remove boxes that overlap significantly (iou >= `iou_threshold`).
3) Go back to step 1 and iterate until there are no more boxes with a lower score than the currently selected box.

Wrapping up the filtering

`yolo_eval()` – Takes the output of the YOLO encoding and filters the boxes using score threshold and NMS. This function uses `yolo_boxes_to_corners()` function which converts between representing boxes via their corners or via their midpoint and height/width (Converts the yolo box coordinates (x, y, h, w) to box corners' coordinates (x_1, x_2, y_1, y_2) to fit the input of `yolo_filter_boxes`).

Summary for YOLO

- Input image (608, 608, 3)
- The input image goes through a CNN, resulting in a (19,19,5,85) dimensional output.
- After flattening the last two dimensions, the output is a volume of shape (19, 19, 425):
  - Each cell in a 19x19 grid over the input image gives 425 numbers.
  - 425 = 5 x 85 because each cell contains predictions for 5 boxes, corresponding to 5 anchor boxes.
  - 85 = 5 + 80 where 5 is because $(p_c, b_x, b_y, b_h, b_w)$ has 5 numbers, and 80 is the number of classes we'd like to detect
- You then select only few boxes based on:
  - Score-thresholding: throw away boxes that have detected a class with a score less than the threshold
  - Non-max suppression: Compute the Intersection over Union and avoid selecting overlapping boxes
- This gives YOLO's final output.