CNN Part - 5

20 September 2023 01:05 PM

Loss Functions in object detection

If $P_c = 1$:

Lets say we are using squared loss,

$$\Rightarrow L(y,\hat{y}) = (y_1-\hat{y}_1)^2 + (y_2-\hat{y}_2)^2 + \dots + (y_8-\hat{y}_8)^2$$

$$L(y,\hat{y}) = \sum_{i=1}^{8} (y_i-\hat{y}_i)^2$$

→ This loss is for 1 data point

$$\begin{array}{c} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{array} \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

for 'N' data points

$$\Rightarrow L(y,\hat{y}) = \sum_{j=1}^{N} \sum_{i=1}^{8} \left[y_i^{(j)} - \hat{y}_i^{(j)} \right]^2$$

If $P_c = 0$:

$$L(y,\hat{y}) = (y_1 - \hat{y}_1)^2$$

{We don't care about the rest of the values as $P_c = 0 \Rightarrow$ The object is not there in the image.}

Output Layer

Sigmoid Activation ← P_c → Squared Loss / Binary Cross Entropy

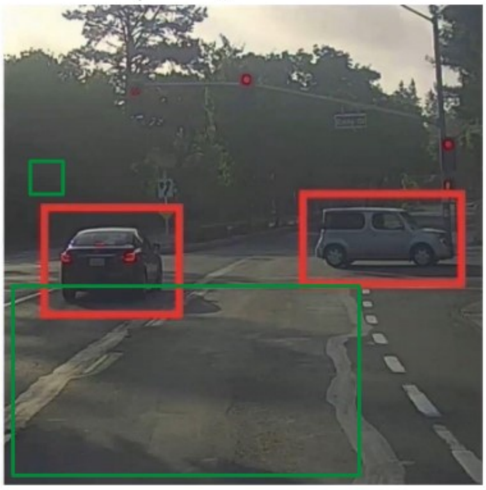
Linear Activation ← {b_x, b_y, b_h, b_w} Squared Loss

Softmax Activation ← {c_1, c_2, c_3} Cross Entropy

{In place of squared loss any other loss can also be used}

Object Detection (YOLO)

Task: The object might be at multiple locations in the image, you need to put a bounding box around all of them



Sliding Window

Training Set (y)


1


1


1


0


0

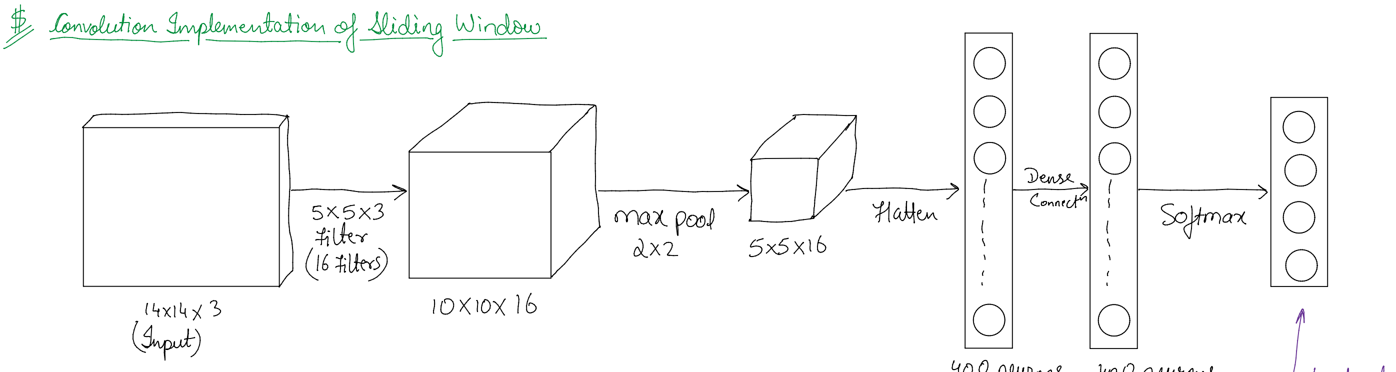
* So, we want to build a CNN model to learn to predict when there is a car.

Now, one way to solve this problem is to take the trained CNN model, and run it on a portion of the image. That portion can be made using a sliding window and the model will tell whether the car is there in that portion/window or not.

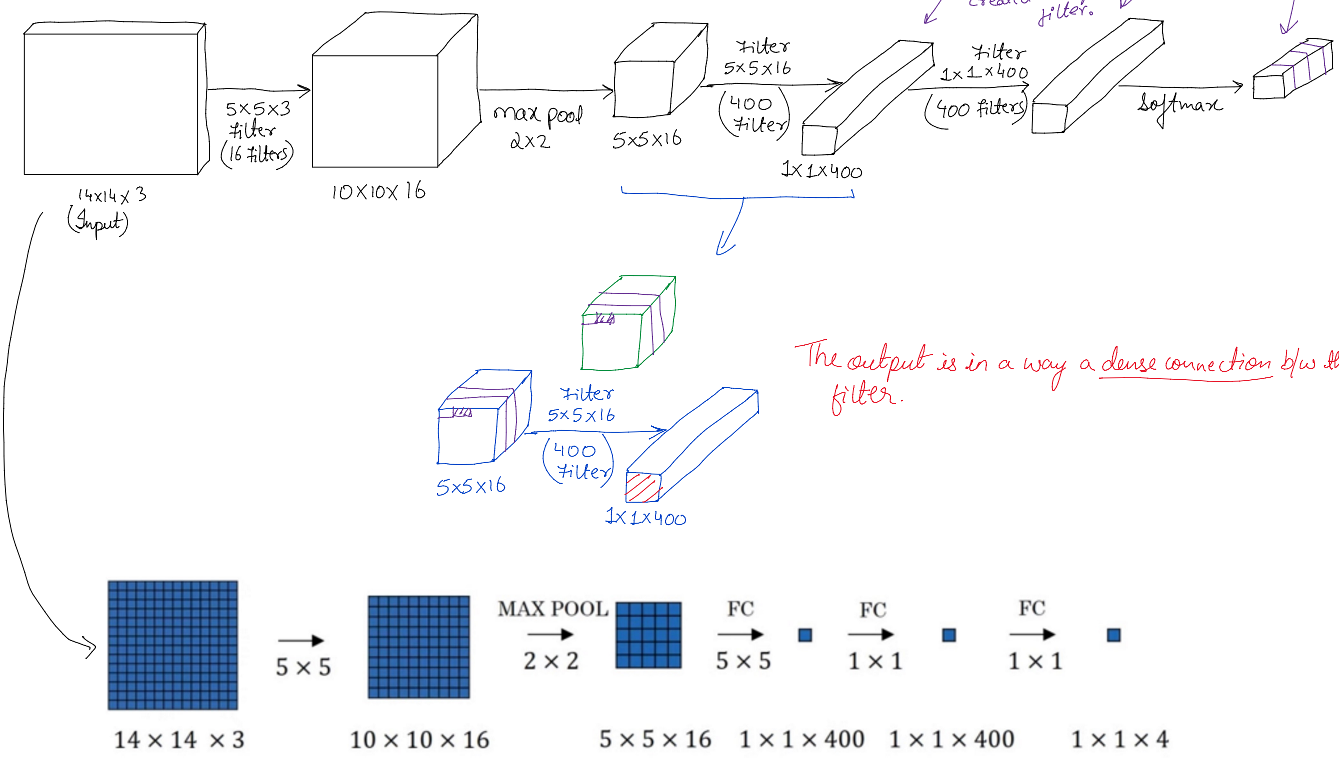
→ The main problem with this approach is the window size; a small window will be really good for cars that are distant (small) in image but will fail to see the nearby (larger) cars.

→ Another issue is that, we are performing the same computation repeatedly because of the window, thereby making this process computationally very expensive.

$ Convolution Implementation of Sliding Window

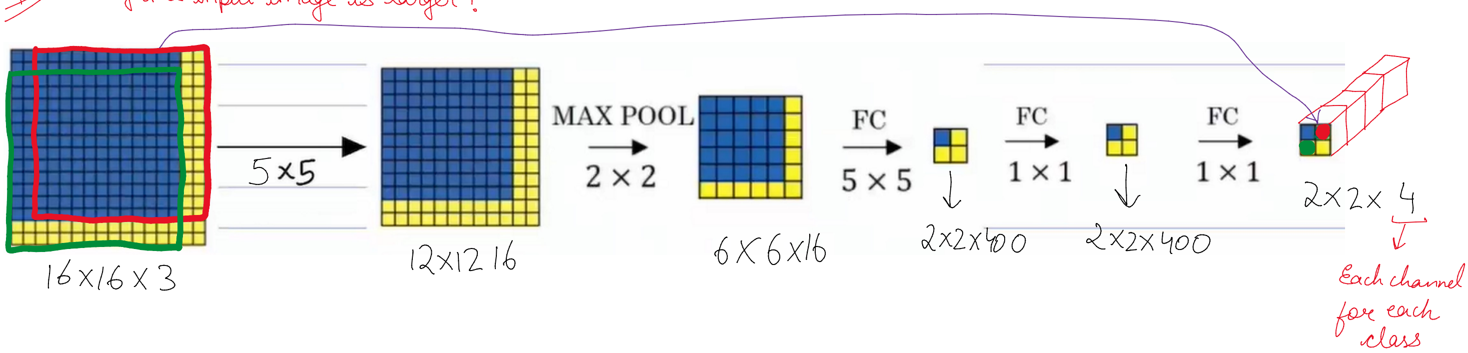


Traditional CNN architecture.



The output is in a way a dense connection b/w the input & filter.

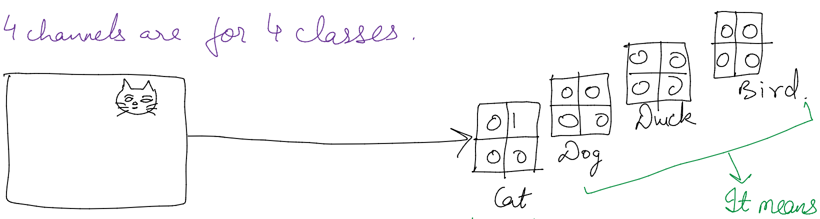
Q. What if the input image is larger?



wherever the first filter goes in the output the pixel which is closer to 1 ⇒ object is in that part of image.

And the 4 channels are for 4 classes.

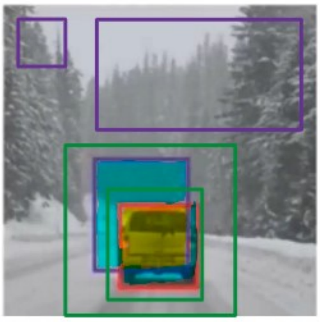
Eg.



It means they weren't detected

It means cat was detected in top right corner

Evaluating a Localization



Red Box → True Box
Blue Box → Predicted Box (By model)

To evaluate, how good the bounding box is, we need to consider → Position of the box
→ Size of the box.

So, in order to evaluate, we use the following metric

$$\text{Intersection Over Union (IOU)} = \frac{\text{area of intersection}}{\text{area of union}}$$

* It means the area of intersection between the True Box and predicted box should be as high as possible, while the total area should be as small as possible.

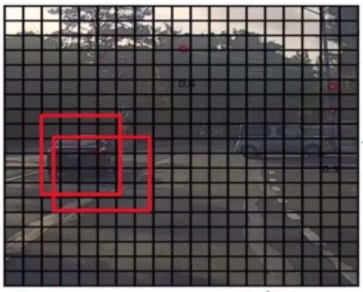
* If the boxes are concentric but the predicted Box is very large compared to true box then the area of union will be large making IOU small.

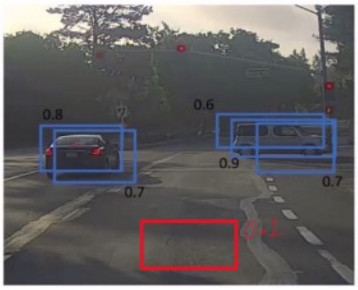
$$\boxed{0 \leq IOU \leq 1}$$ Range of IOU

(Bad) (Good)

Removing Overlapping Boxes

While implementing sliding window using convolution, we might end up getting multiple boxes detecting the target object.
Now, the task is to find the best bounding box out of all the boxes.





Both red boxes will detect car

For each prediction →

$$\hat{y} = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad P_c \to [0,1]$$

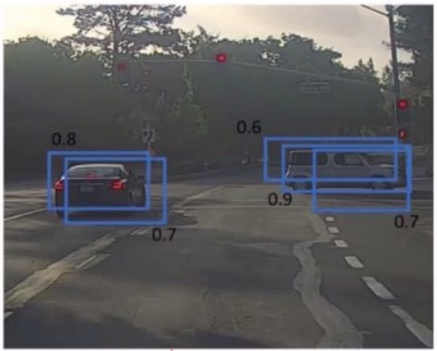
0.95 → Present
0.05 → Absent

{ P_c basically acts like a confidence value, as in it gives prob. (Sigmoid) of presence of an object at that window. }

Similar to logistic regression where $y = 0$ or 1 but $\hat{y} = [0,1]$ which gives a range and by using sigmoid we apply a threshold.
In the same way for P_c →

> 0.5 → Object is there
< 0.5 → object is not there.

Now, to select the best bounding box:



1.) Remove all the boxes for which $P_c < 0.5$

2.) Pick the box which has highest P_c. $[Box_i]$

3.) Remove all the boxes which have high IoU with Box_i, report Box_i and remove it.

The threshold needs to be defined. (Hyperparameter)

4.) Repeat Step 2.

Highest IoU.

High Intersection

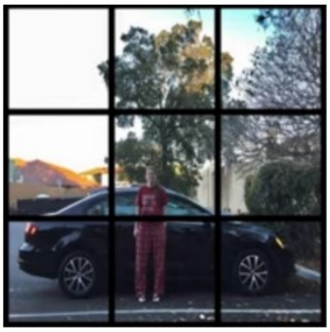
High Intersection

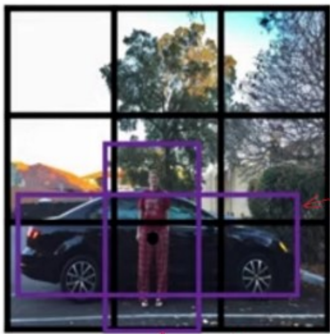
Eg IoU = [0.9, 0.6, 0.7, 0.8, 0.7]

highest IoU

Lowest IoU (0) ⇒ different Object

Anchor Boxes





Good for detecting a car

Good for detecting a person.

When working with object detection, we have to work with multiple different Boxes, called Anchor Boxes, to identify various objects in the image.

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

This is for one box

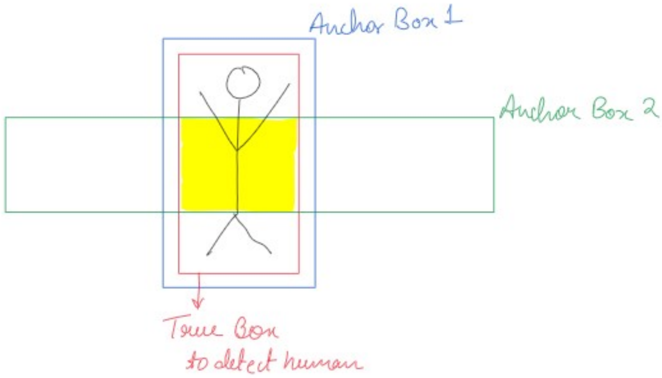
Now lets say we have to detect 2 objects in the above image, so we need 2 boxes

Anchor Box-2

Anchor Box-1

So, as a result the output size will also increase:

$$y = \begin{bmatrix} \left.\begin{matrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{matrix}\right\} \text{Anchor Box-1} \\ \left.\begin{matrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{matrix}\right\} \text{Anchor Box-2} \end{bmatrix}$$



Now in order to select the Best Anchor Box out of the 2 as both are detecting the human, the Anchor Box with the highest IoU with true Box is selected. In above eg, it will be Anchor Box 1

References for YOLO:

- https://pjreddie.com/darknet/yolo/
- YOLO Research paper: https://arxiv.org/pdf/1506.02640v5.pdf
- YOLO Improved Research Paper: https://arxiv.org/pdf/1612.08242v1.pdf
- YOLO V3 Research Paper: https://arxiv.org/pdf/1804.02767.pdf
- https://medium.com/analytics-vidhya/yolo-explained-5b6f4564f31
- https://www.analyticsvidhya.com/blog/2018/12/practical-guide-object-detection-yolo-framewor-python/
- https://www.analyticsvidhya.com/blog/2021/06/implementation-of-yolov3-simplified/