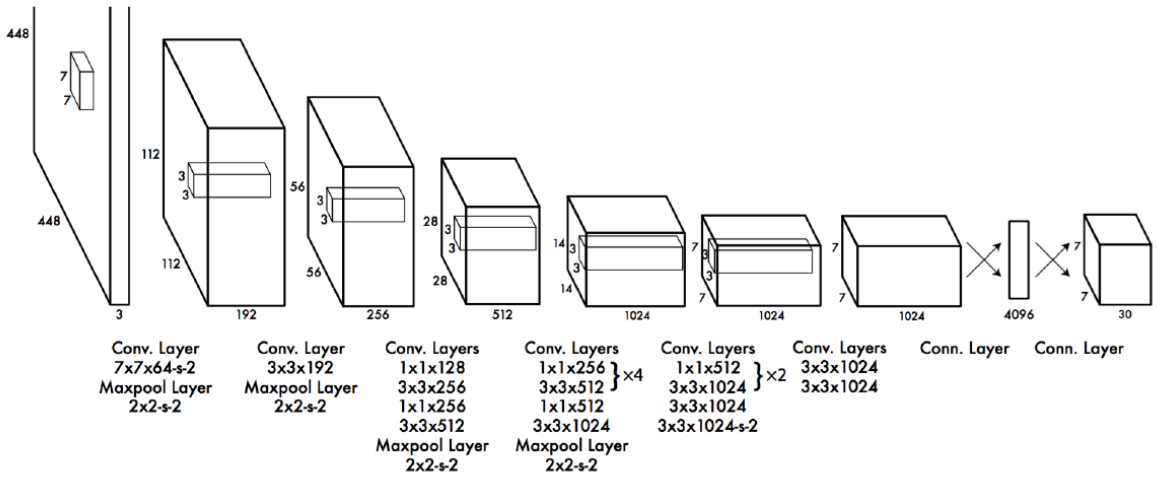
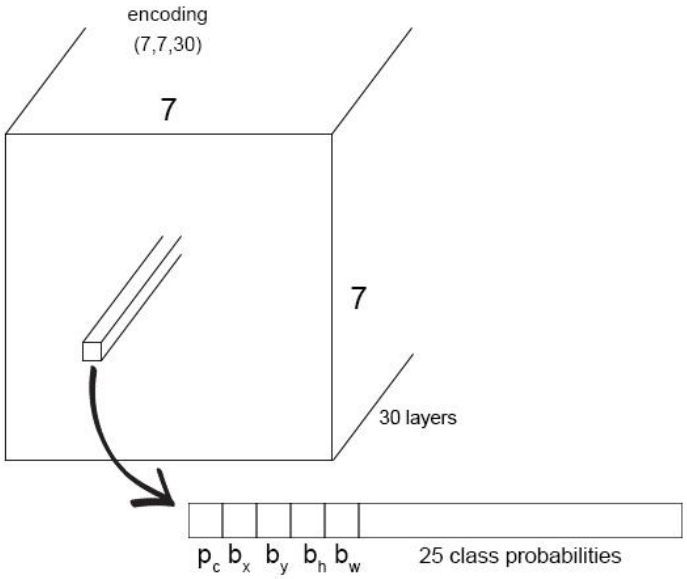
Low level YOLO Implementation using Python

I did a high level overview of YOLO Object Detection algorithm in my previous blog (find it here). Having an overall idea on that will be better to understand this idea. Here, I have done a low-level implementation of object detection for the YOLO algorithm using Python. All code snippets can be found at my GitHub here.

YOLO has the following network architecture. Starting from an input image of size (h=448,w=448,c=3), it uses a series of convolution (CONV) and pooling (MAX POOL) layers and converge to a final output block of size (7,7,30) as shown in the following figure.

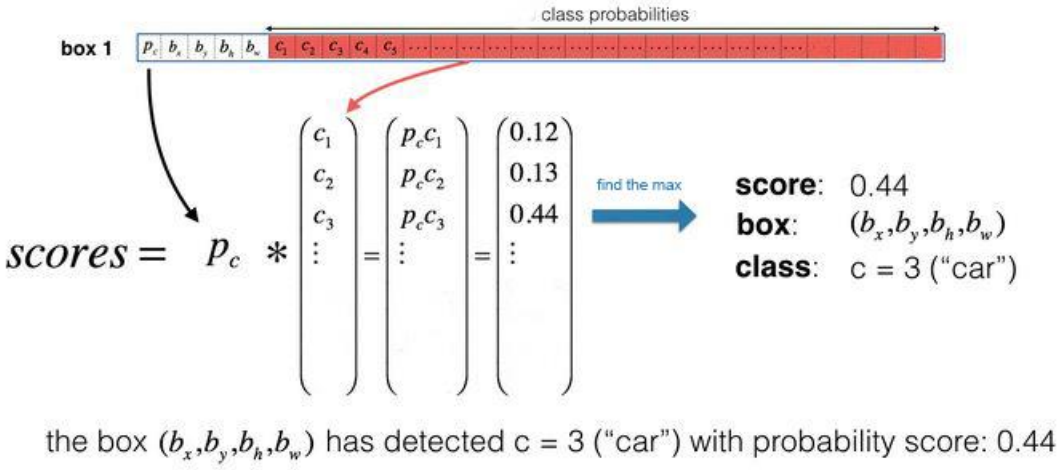


As per the key characteristic of YOLO, if the midpoint of an object falls into a grid cell, that grid cell is responsible for detecting the object. So, each pixel in this output's each layer is responsible for image detection criteria. It is a list of bounding boxes along with the recognized classes. Each bounding box is provided by 6 information namely box probability, box coordinates/dimensions and the class probability/box confidence (pc, bx, by, bh, bw and c).



Each layer in this output could be identified as. The following procedure is done in order to obtain the scores, boxes and classified class labels.

1. First, the corresponding outputs are retrieved from the output block given.
2. Computing the box scores by elementwise multiplication of the box probability and class probability.



3. There can be more number of boxes which corresponds to a single image. This happens at nearby grid cells producing multiple bounding boxes. To get rid of these duplicate boxes, we perform filtering at two stages.
    a. Get rid of boxes with a low score (score thresholding – means that the box is not much confident in detecting a class)
    b. Select only one box when several boxes overlap with each other and detect the same object (Non-Max Suppression). We have to scale up the box coordinates to the image size before performing NMS operation.
4. After Non-Max Suppression (NMS), it then outputs the recognized objects together with the bounding boxes scaled up to the image size.

Above functions can be written using the libraries such as Keras. Here, I have written all those functionalities from scratch in Python. YOLO is a very efficient algorithm as it uses only one forward propagation pass to make the predictions as its name says 'You Only Look Once'. People tend to use YOLO to achieve this higher efficiency. So, The above low level implementation will further contribute in speeding up getting predictions. Also, platforms such as Raspberry Pi's may find difficult to run such libraries. This will avoid that issue.

The code files contain in the repo;

- Boxes_to_Corners
- Filter_Boxes
- IoU_NMS
- Scale_Boxes
- Evaluation

Detailed description of each function has been done in each code file itself.

References: Convolutional Neural Networks by deeplearning.ai by Prof. Andrew Ng. on Coursera: https://www.coursera.org/learn/convolutional-neural-networks