CNN (Convolutional Neural Network) – Image Training

Convolutional neural networks are a specialized type of artificial neural networks that use a mathematical operation called convolution in place of general matrix multiplication in at least one of their layers. They are specifically designed to press pixel data and are used in image recognition and processing.

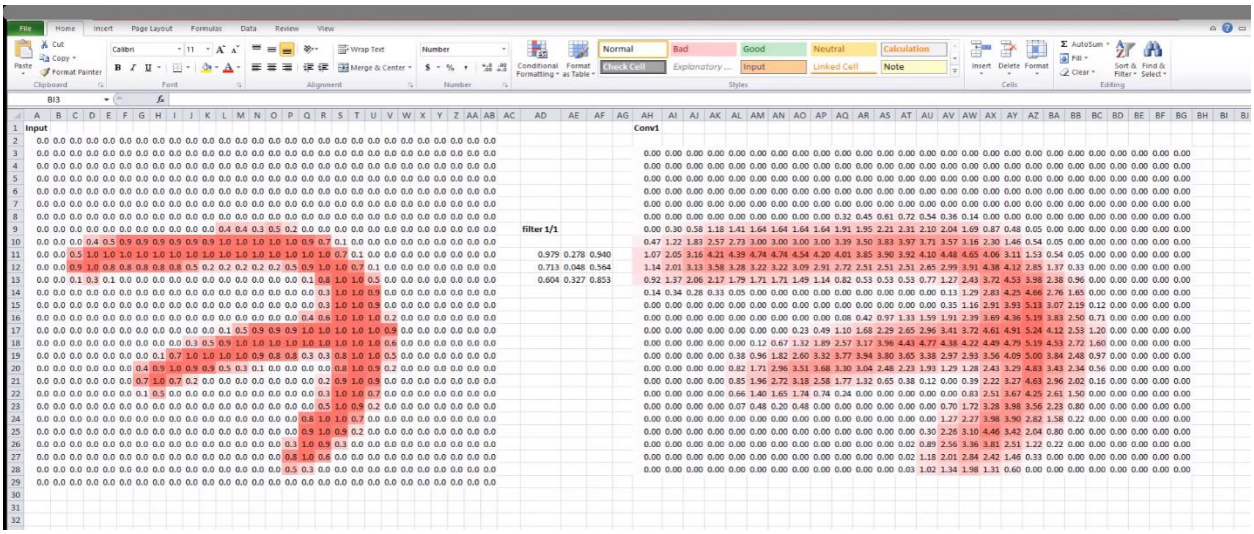


Figure 1: Demonstration of CNN in ExcelSheet

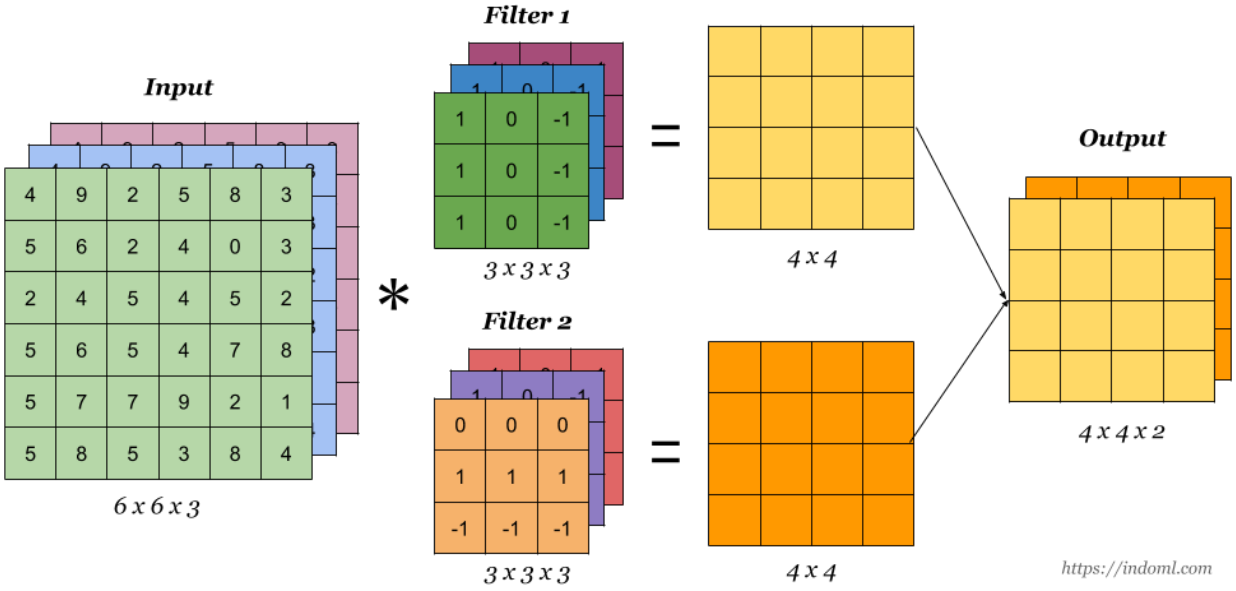


Figure 2: Filtering RGB images

For a layer in the output , where $f^{[l]}, s^{[l]}, p^{[l]}, {n_c}^{[l-1]}, {n_c}^{[l]}$,filter of the layer, stride, padding in layer, number of channel in a previous layer and number of filter in a layer,

Input: ${n_H}^{[l-1]} * {n_W}^{[l-1]} * {n_c}^{[l-1]}$

Output: ${n_H}^{[l]} * {n_W}^{[l]} * {n_c}^{[l]}$

To determine the value of next layer

$${n_H}^{[l]} = \left[\frac{{n_H}^{[l-1]} + 2p^{[l]} - f^{[l]}}{s}\right] + 1$$

$${n_W}^{[l]} = \left[\frac{{n_W}^{[l-1]} + 2p^{[l]} - f^{[l]}}{s}\right] + 1$$

Filter size $X = f^{[l]} * f^{[l]} * {n_c}^{[l-1]}$

Activation Size: $a^{[l]} = {n_H}^{[l]} * {n_W}^{[l]} * {n_c}^{[l]}$

Weights: $f^{[l]} * f^{[l]} * {n_c}^{[l-1]} * {n_c}^{[l-1]}$

Biases: $b^{[l]} = {n_c}^{[l]}$

We determine the activation function

$$Z^{[l]} = W^{[l]}X + b^{[l]}$$

$$a^{[l]} = g(Z^{[l]})$$

Here, activation function is determined which is flattened in deep neural network to train and determine the classes of the objects.

YOLOV3(You Look Only Once)- A Object Detection Algorithm

YOLO is a Convolutional Neural Network (CNN) for performing object detection in real-time. CNNs are classifier-based systems that can process input images as structured arrays of data and recognize patterns between them (view image below). YOLO has the advantage of being much faster than other networks and still maintains accuracy. It allows the model to look at the whole image at test time, so its predictions are informed by the global context in the image. YOLO and other convolutional neural network algorithms "score" regions based on their similarities to predefined classes. High-scoring regions are noted as positive detections of whatever class they most closely identify with.

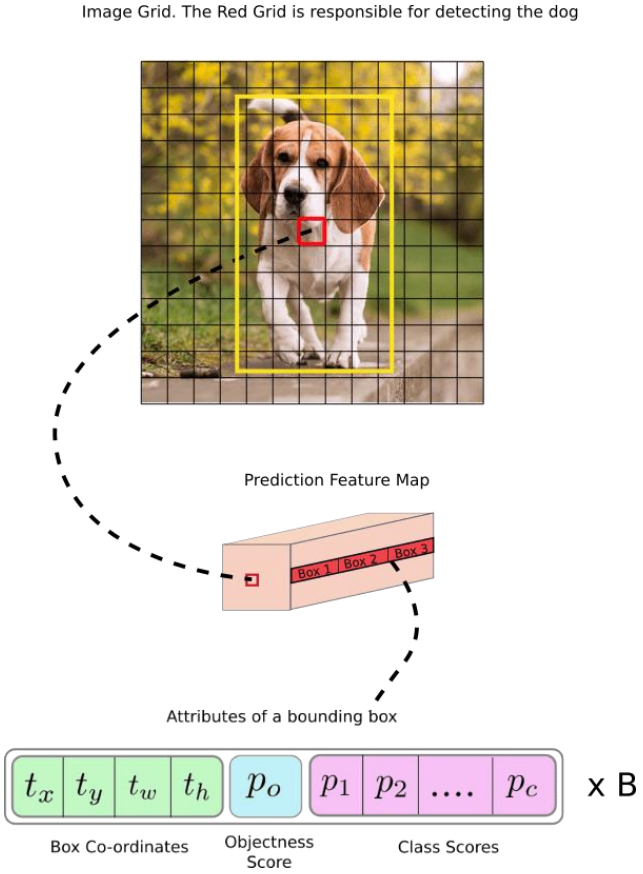
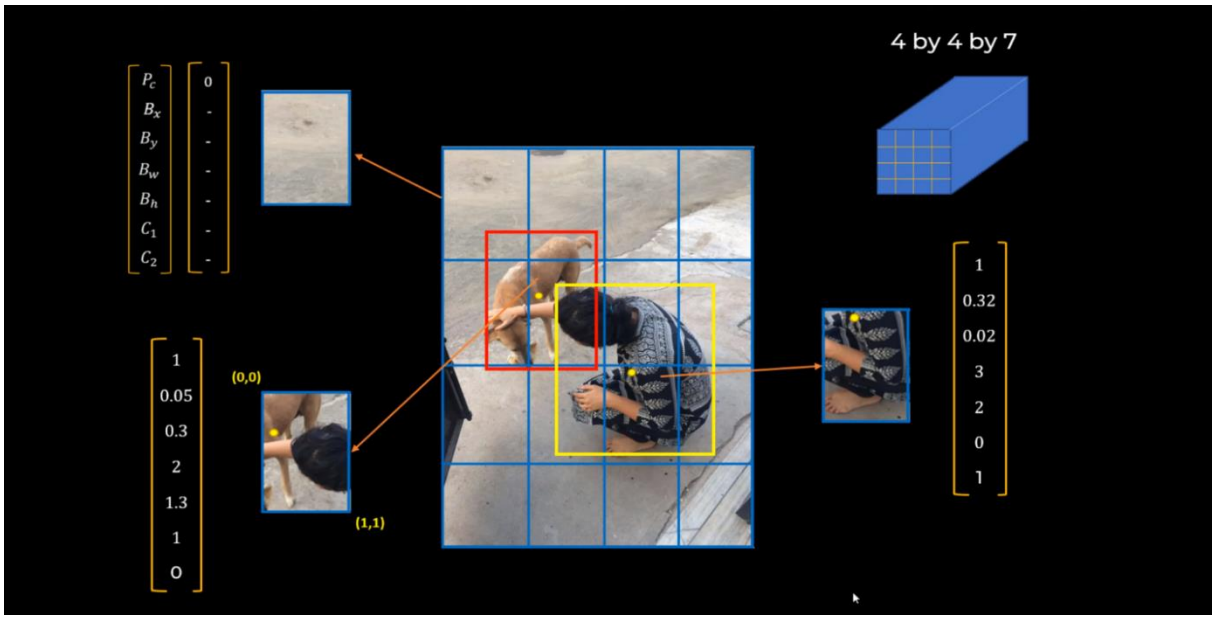
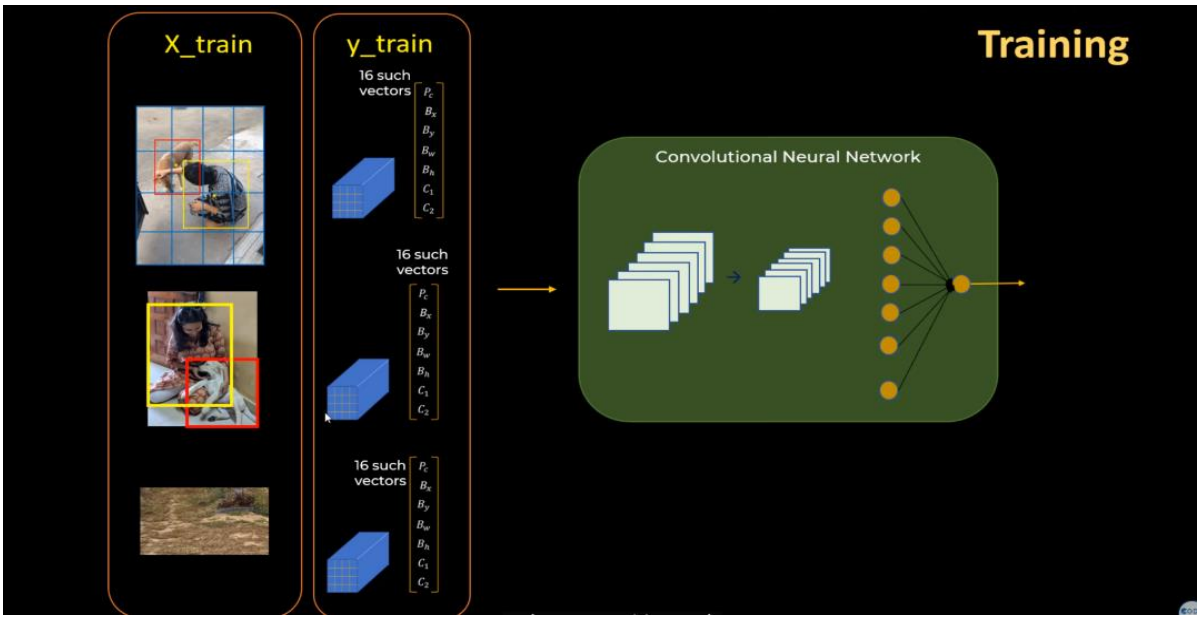


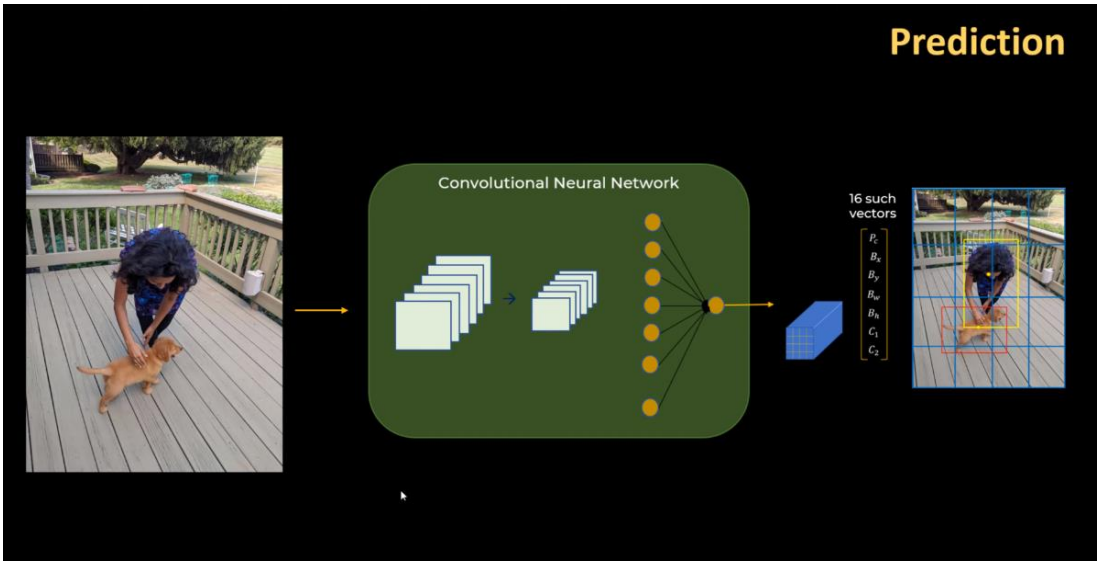
Figure 3 : Vector Representation of the detected object 's centroid onto the grid



When an object is being detected, on the basis of grid on the image and bounding boxes of object, object score is assigned 1 else is assigned 0. Probability of occurrence, bounding boxes and classes of the object are analyzed in a vector. If probability of occurrence or object score is zero , then rest of the portion is ignored , else the bx,by,bh,bw is determined within classes.



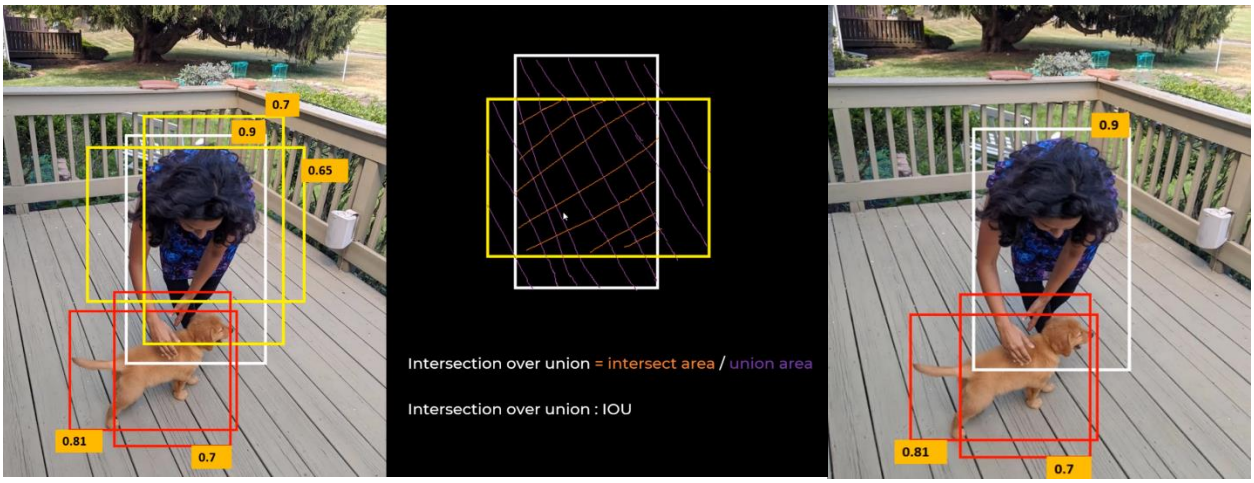
The images are trained with assigning boundary boxes and classes and we assign the probabilistic vector with score, boundary box and its classes. Each of the trained vector data are accumulated into a bulk of images with their vector . Here there is 16 number of grids causing 16 vectors formation. The vector undergoes CNN algorithm to determine the classes of images being trained.



For prediction, the images are scanned and is convolved with the filter to give the class of the boundary images with unique id through CNN

IOU(Intersection over Union)

IOU(Intersection over Union) is a term used to describe the extent of overlap of two boxes. The greater the region of overlap, the greater the IOU. IOU is mainly used in applications related to object detection, where we train a model to output a box that fits perfectly around an object. For example in the image below, we have two yellow box, and a white box around a human. The white box represents the correct box, and the two yellow box represents the prediction from our model. The aim of this model would be to keep improving its prediction, until the white box and the yellow boxes perfectly overlap, i.e. the IOU between the two boxes becomes equal to 1.



IOU is also used in non max suppression, which is used to eliminate multiple boxes that surround the same object, based on which box has a higher confidence. I will go over non max suppression in more detail in my next post.

Calculating IOU: Let us assume that box 1 is represented by [x1, y1, x2, y2], and box 2 is represented by [x3, y3, x4, y4]. (We will use this convention later to calculate the areas.)

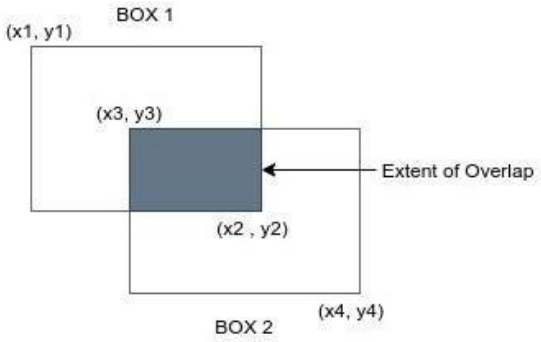


Figure 1: Intersection of two boxes

$$\textbf{IOU} = \frac{\text{Area of Intersection of two boxes}}{\text{Area of Union of two boxes}}$$

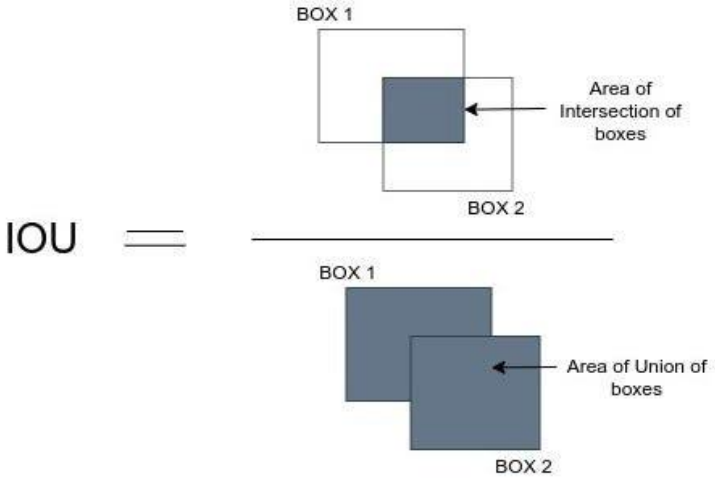
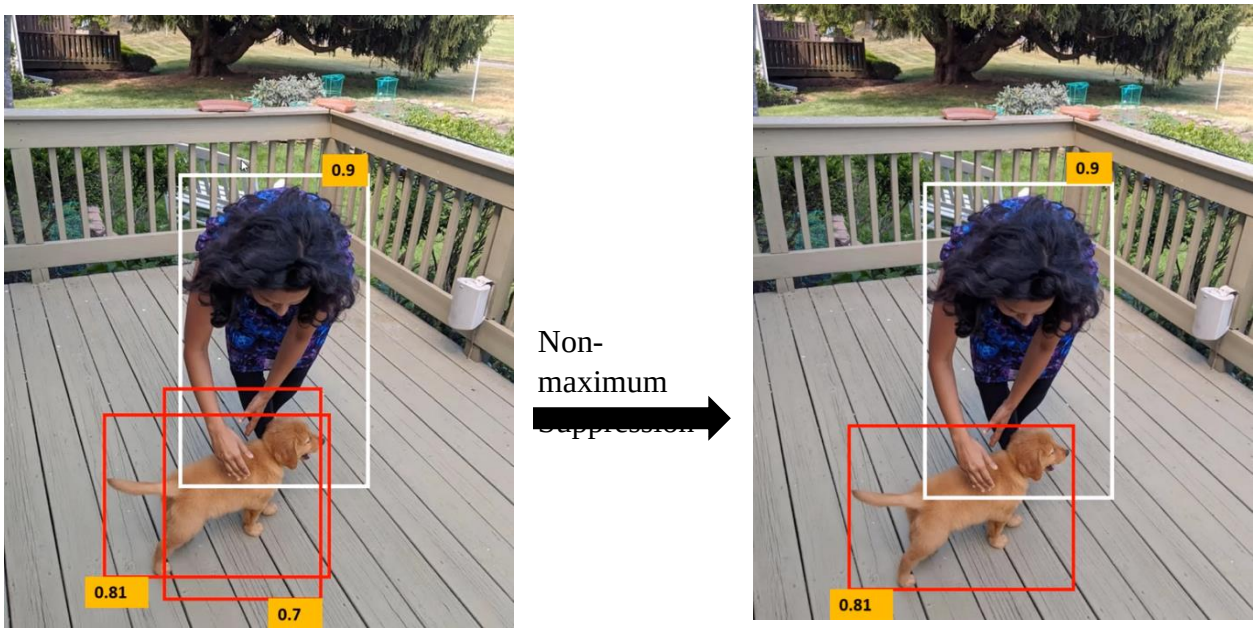


Figure 2: Diagrammatic Representation of the formula to calculate IOU

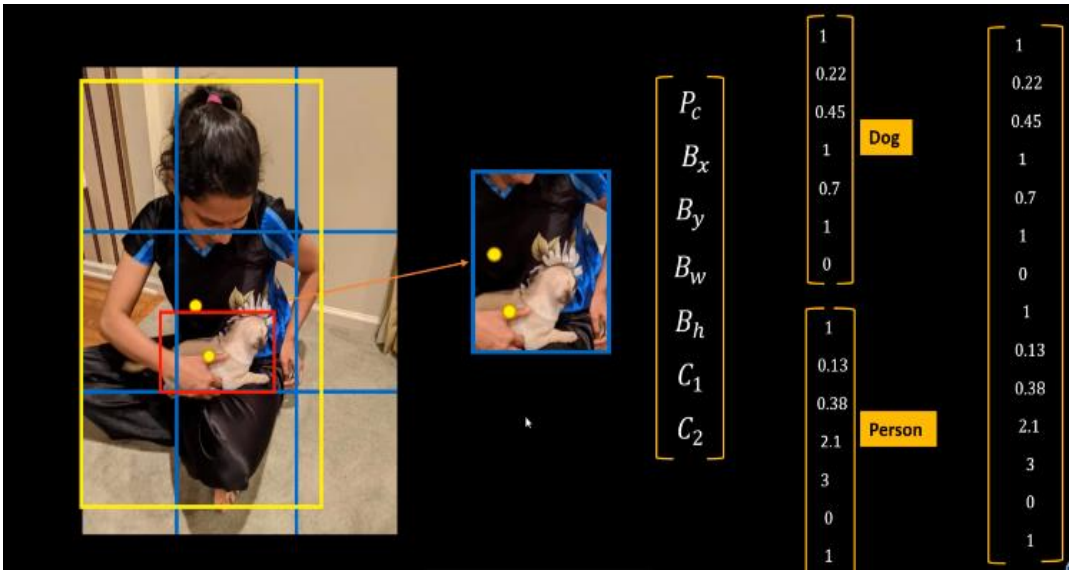
Non max suppression



Non-max suppression is a technique used mainly in object detection that aims at selecting the best bounding box out of a set of overlapping boxes. It is a class of algorithms to select one entity (e.g. bounding boxes) out of many overlapping entities. The selection criteria can be chosen to arrive at particular results. Most commonly, the criteria are some form of probability number along with some form of overlap measure (e. g. IOU).

In object Detection, there is a problem that an algorithm detects multiple bounding boxes for a single object. To solve this problem there, is a technique called non-max suppression. Non-max suppression cleans up thee multiple detection and end with just one detection per object. For this it chooses the bounding boxes with highest probability and suppressed all the other bounding boxes whose IOU with it is greater, so in last only one bounding box is left which is more accurate. In nutshell, the overlapping portion is judge through probability of occurrence of centroid and min value of IOU (Intersection Over Union).

Anchor Boxes



If the centroid of two objects lies on the same grid in yolo, detection of the object becomes difficult with normal bounding vector and so anchor box accumulates two vector into a vector , so to get detected.

LOSS FUNCTION

Regression Loss:

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2]$$

Confidence Loss:

$$\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}[\hat{C}_i \log(C_i) + (1-\hat{C}_i)\log(1-C_i)] + \lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{noobj}[\hat{C}_i \log(C_i) + (1-\hat{C}_i)\log(1-C_i)] +$$

Classification Loss:

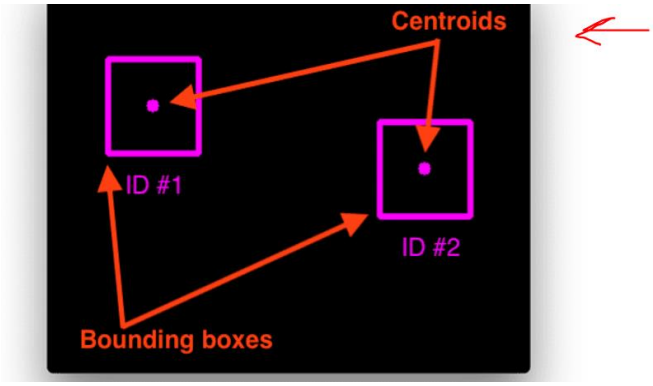
$$\sum_{i=0}^{S^2} I_{ij}^{obj} \sum_{c \in classes}[\hat{p}_i \log(p_i(c)) + (1-\hat{p}_i(c))\log(1-p_i(c))]$$

Deep Sort(Simple Online Realtime Tracking): Deep Sort is an effective object tracking algorithm since it uses deep learning techniques. It works with YOLO object detection and uses Kalman filters for its tracker to get the track of the data with the features of the objects. It comprises of detection, estimation, association and track identity creation and destruction.

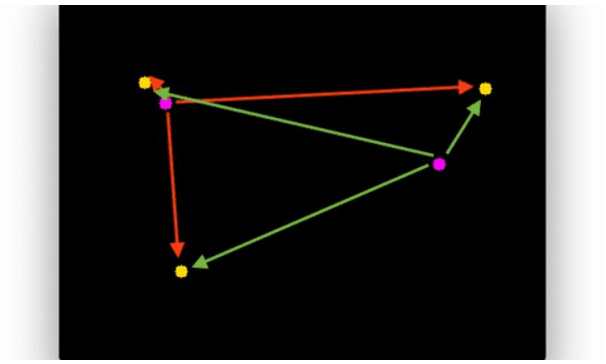
Centroid Tracking algorithm: The centroid Tracking algorithm is a combination of multiple-step processes. It uses a simple calculation to track the point using Euclidean distance. This Tracking algorithm can be implemented using our custom code as well.

Steps Involved in Centroid Tracking Algorithm

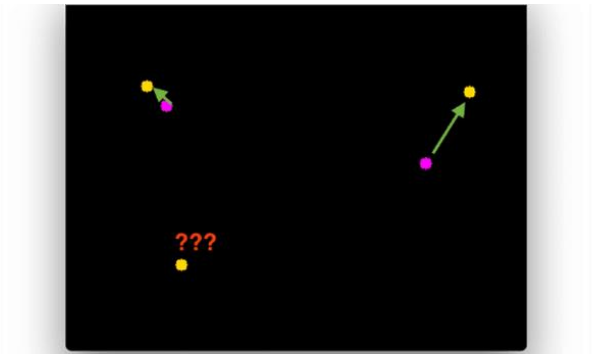
Step 1. Calculate the Centroid of detected objects using the bounding box coordinates.



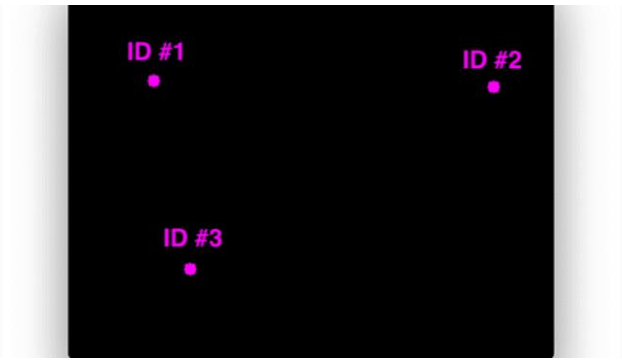
Step 2. For every ongoing frame, it does the same; it computes the centroid by using the coordinates of the bounding box and assigns an id to every bounding box it detects. Finally, it computes the Euclidean distance between every pair of centroids possible.



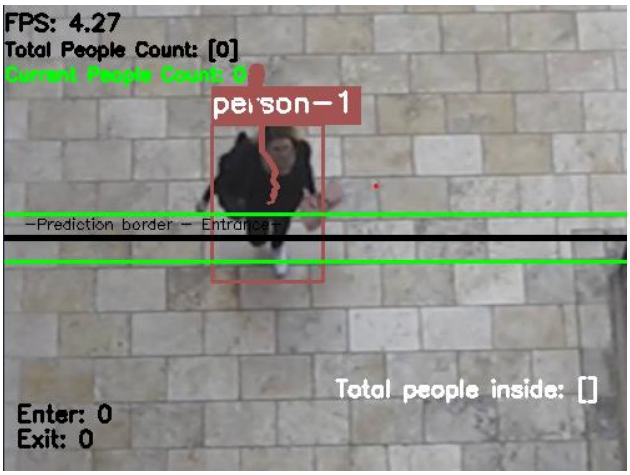
Step 3. We assume that the same object will be moved the minimum distance compared to other centroids, which means the two pairs of centroids having minimum distance in subsequent frames are considered to be the same object.

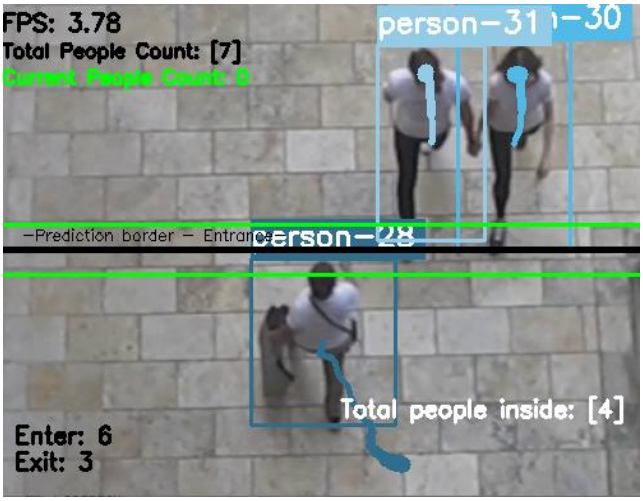


Step 4.



Now it’s time to assign the IDs to the moved centroid that will indicate the same object. We will use the frame subtraction technique to capture a moving object . F(t+1) -F(t) => moved object.





Here ,The trajectory link represents the motion of the tracks held onto by kalman's filter and the track within the prediction border is determined using Centroid tracking for calculating number of counts.