We are using open source library called YOLO for training our model from scratch.

**Pre Processing:**

A python script (PreProcessing.ipynb) has been written for creating train.txt and test.txt which will further be used for training purpose. Train.txt will contain the list of all the images along with their normalized boundary box co-ordinates. Test.txt will contain the name of all images in test Set. Dependencies for running this script are OpenCV, numpy and pandas.

As we are just concerned about detecting co-ordinates the boundary box of the object, there will be only one class and their normalized boundary box coordinates.

**Training the Model:**

After making the YOLO for object detection, setup the configuration file (.cfg) as classes = 1(as we want only bbox) and filters = 18 ( (classes + 5) \* 3) and learning rate = 0.001.

We have done training on all the labeled images(14k) given to us. Start the training by providing all the paths in the .data file. We started training using mAP(mean average precision chart). After every 1000 iterations we are saving weight files into a folder and keep monitoring results of every epochs.

We stopped training when average loss was around 0.04 and was not decreasing anymore and mAP = 90.9 percent.

Once the training was completed we wrote a scrips called map.py to select top5 weights with highest mean Average precision. In our case, we choose early stage weight files in Order to overcome overfitting.

Once we select the weights of highest precision we can run our script test.py to generate the co-ordinates and write it to a csv file.

**Guidelines to run the source code:**

1 ) Open PreProcessing.ipynb file and provide path of ‘training.csv’ to generate a training.txt file which will keep the class Id and the normalized co-ordinates. Similarly provide path of testing.csv to generate test.txt which will keep all the image names with their absolute path. Also provide absolute path of images folder in **img\_path\_dir\_name.**

2) Due to Size restriction of 20MB, we are not attaching our trained weight file, kindly refer our google drive link to download and put in the same directory.

Link:

https://drive.google.com/open?id=1WMTH7UhIePUA71a4ewKqQITppeiPgpff

3) Put all the files in same directory and run test.py Script. It will create a new file test\_label.csv file.

4) Test.py script will iterate over all the **image\_names** in test set(test.txt) and performs detection on them. Their co-ordinates will be saved in test\_label.csv file.