Object (Basketball) Detection Using YOLO Algorithm

Project Report

Subject: RBE 549 (Computer Vision)

Submitted By:

TEAM H

Kshitij Sharma
M.S. Robotics Engineering
Worcester Polytechnic Institute,
Worcester, MA
Email: ksharma@wpi.edu

Shubham Malhotra
M.S. Robotics Engineering
Worcester Polytechnic Institute,
Worcester, MA
Email: smalhotra@wpi.edu

Jasman Deep Singh Narang
M.S. Robotics Engineering
Worcester Polytechnic Institute,
Worcester, MA
Email: jsnarang@wpi.edu

Worcester Polytechnic Institute

Abstract

Object detection has grown since its advent and new, strong, and faster algorithms have paved the way for them to assist in industries like defence, robotics, healthcare etc. In this report we focus on Basketball object detection using the YOLO (You Only Look Once) algorithm. A basketball image dataset was collected and pre-processed and the YOLO algorithm model was trained on our own image dataset to detect the basketball in videos and images successfully.

Index Terms **- Computer Vision, Object Detection, You Only Look Once (YOLO), Basketball, Data, Pre-processing, Jupyter Notebook**

INDEX

List of Figures

Introduction

We humans see and perceive things in a sense which helps us to perform day to day complex tasks such as driving, working with heavy machinery. Human vision system is fast and accurate, and it derives our daily movements (Redmon et al., 2016). For computers or machines such as robots the capacity to interact with the environment is somewhat a matter of great research. Algorithms that are fast and accurate for object detection would allow the computers to perform the human tasks such as driving or making visual calculations and decisions.

Object Detection algorithms have evolved a lot since the Viola Jones Detector in 2001 that detected faces in real-time. Traditional machine learning changed and improved the computer vision domain, but these changes also brought a set of drawbacks (Liu et al., 2020). These machine learning changes were very specific and complicated. Then came in the deep learning applied to computer vision tasks.

In today's time the object detection takes place through neural networks and deep learning models. Big and strong algorithms like R-CNN, Fast R-CNN, Faster R-CNN, all utilize the concepts of deep learning and perform the object detection. These object detection algorithms bridge the gap between the capabilities of a human and a machine. These algorithms use region segmentation and region definition as the ways to generate the bounding boxes and then generate the classifier on the bounding boxes. After all these bounding boxes are rechecked and the duplicate ones are removed. Rescoring the boxes also occur. These process and post processes make these algorithms very slow and therefore cannot always be used for object detection where the data is not much. These algorithms may perform better with a lot of data but for smaller data we can easily use the YOLO algorithm.

YOLO algorithm or You Only Look Once algorithm works in a single pass. From the pixel coordinates we can end up with bounding boxes and class probabilities in one pass with YOLO algorithm (Redmon et al., 2016). In comparison with these heavy-duty big algorithms that perform well with huge data, the YOLO algorithm is very fast and performs very well with smaller data sets. The reason for the YOLO algorithm to be very simple and fast is that it works as a regression problem and is passed through the neural network at test time to predict detections. We can see in Figure 1 The YOLO Detection System that how simple the YOLO neural network is in implementation.

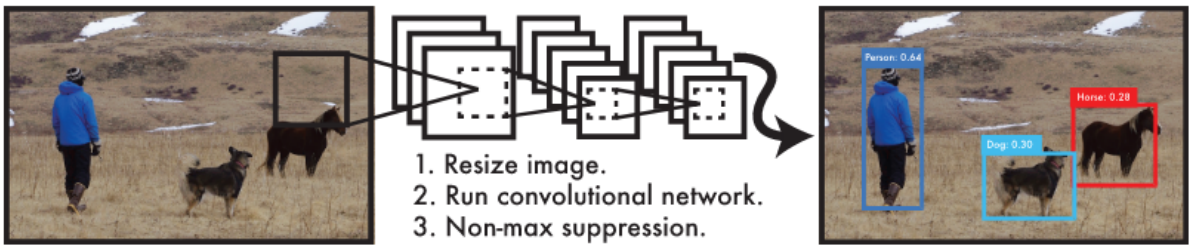

Figure 1 The YOLO Detection System **(Redmon et al., 2016)**

The base network of the YOLO algorithm runs at 45 fps and the GPU enabled network of the YOLO algorithm runs at 150 fps which inherently means that the real-time latency while detecting objects from video is less than 25 milliseconds (Redmon et al., 2016). We can see the YOLO architecture that was implemented in (Redmon et al., 2016) in **Figure 2 The YOLO Architecture**.

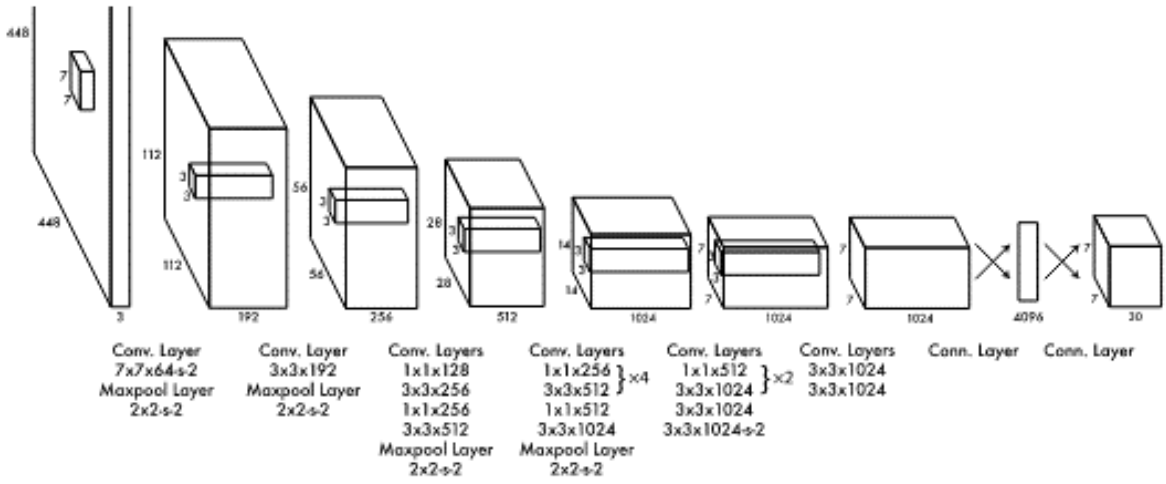

Figure 2 The YOLO Architecture

This report **aims to apply** the concepts of YOLO algorithm as they have been developed over the years to be faster and much more accurate. Here we will make sure that we train the YOLO model on our custom data set of basket balls and make it much more capable of detecting a basketball in an image or a video where the background can be as noisy as possible.

Materials And Methods

Dataset

For the purpose of this project, we required a dataset to train a model to recognize basketball. We got the dataset from Kaggle. But the Kaggle dataset alone was not enough thus we clicked pictures from our own smartphone and retrieved some images from open-source websites.

Data as it was, could not be directly run in the algorithm. Thus, after procuring the dataset some steps for pre-processing of the data were performed and the following tasks were carried out:

- Separate the relevant and the irrelevant images from the dataset and remove the irrelevant data i.e., not a basketball image, from the dataset.
- Data augmentation is a method to increase the volume of data by adding slightly changed copies. We performed data augmentation on our image dataset by altering them in the following manner:
    - Making 2 copies of an image and rotating both by 33 degrees in opposite directions,
    - Mirroring an image with respect to. the vertical axis of the image,
    - Cropping the image in different ratios,
    - Isolating the objects in the images,
    - Scaling the images.

After the data was pre-processed, we used **labelImg software** as shown in Figure 3 labelImg software from a github repository to label the images in the YOLO format.

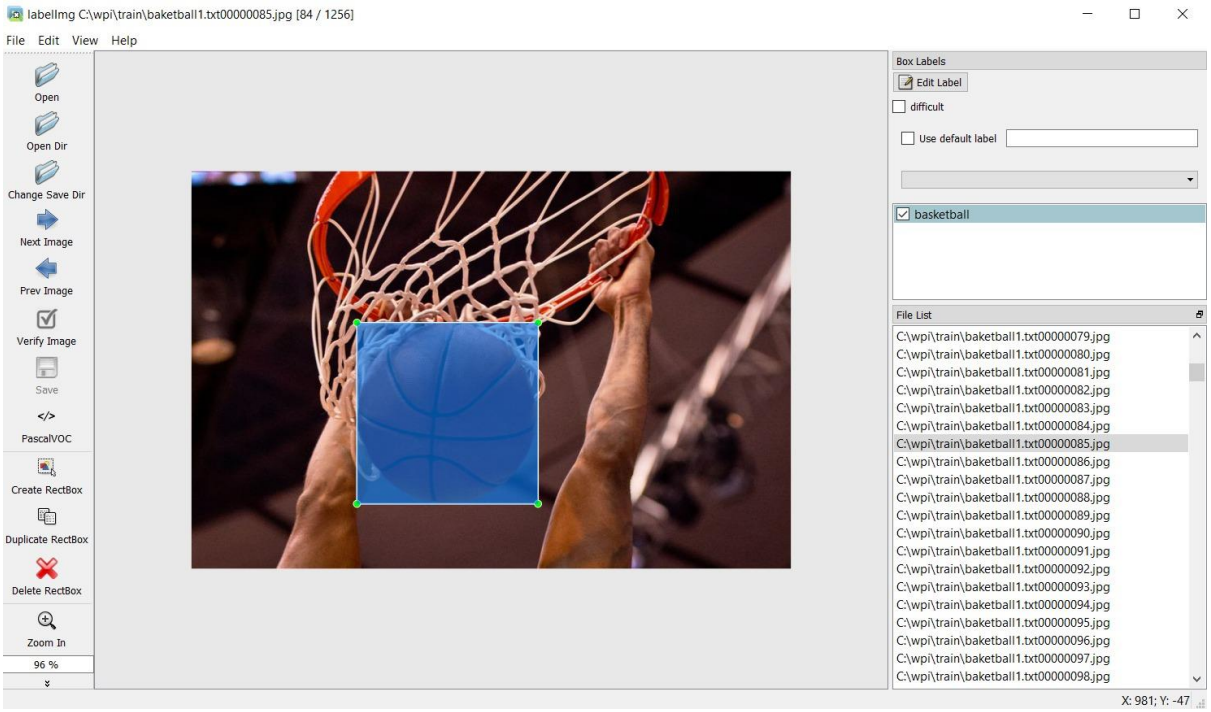

Figure 3 labelImg software

Further, to see the effect of different proportion of train and test data size on the results, we divided the dataset into train and test datasets of varied sizes. We started splitting the dataset into train and test data sizes as 10%-90% respectively and moved on to 20%-80%, and we went up till 90%-10%, to yield different results with different accuracies.

YOLO Algorithm

YOLO algorithm was used to perform object detection on the basketball image dataset. Ultralytics, a group of people developed a customized version of the YOLOv4 official version on their github repository with the name YOLOv5. In the repository, there are 5 models for YOLO called YOLOv5n6, YOLOv5s6, YOLOv5m6, YOLOv5l6, YOLOv5x6. The performance curves for different YOLO models can be seen in the Figure 4 Performance of different YOLOv5 models.

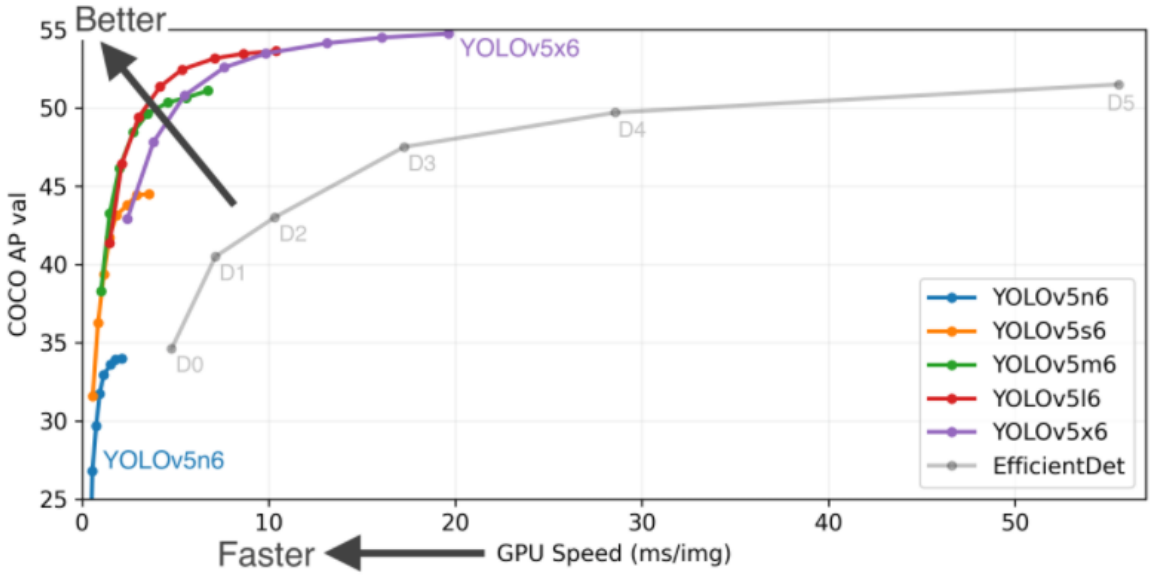

Figure 4 Performance of different YOLOv5 models (github repository)

The dataset was then trained on the 4 models namely, YOLOv5n, YOLOv5s, YOLOv5m, and YOLOv5l and the values were generated for all the models. All these models depend on the data size therefore we trained the 9 datasets on all the 4 models and collected results.

YOLOv5 Experiments

For our project, similar experiments were performed for all the 9 datasets and 4 models. The difference was in the weights of the parameters used for each of the models and the size of the 9 datasets.

We imported a few libraries and installed a few dependencies. Then we trained each of our model from scratch for all the datasets that we had created. The trained models generated a bunch of results on precision, recall, losses, confusion matrices and a result.csv file as well as trained weights that were specific to our dataset.

After training the model and generating dataset specific results we loaded our custom model to make predictions and detections on the test images that we had. This was the test of the trained model on the final test images. The resulting images were supposed to have a bounding box on top of them with a probability index as well.

Another test that we performed on the trained model was that we loaded a custom video file of a college basketball game, and the model was able to perform real-time object detections. There were some false results as well but all it was able to perform alright.

RESULTS

YOLOv5 Algorithm

The results were generated for each of the model based on the data that was observed we are only showing the results for 3 datasets:

YOLOv5n

1. For Dataset – Train 10% to Test 90%

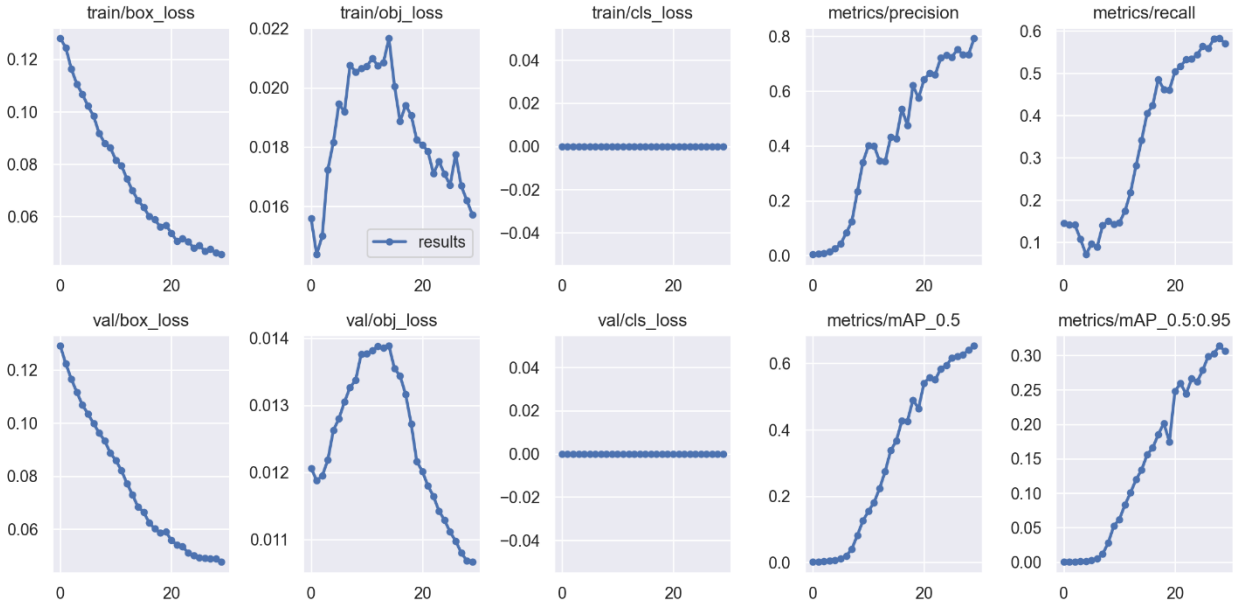

Figure 5 Precision Recall Plots for Dataset Train 10 - Test 90 YOLOv5n

2. For Dataset – Train 50% to Test 50%

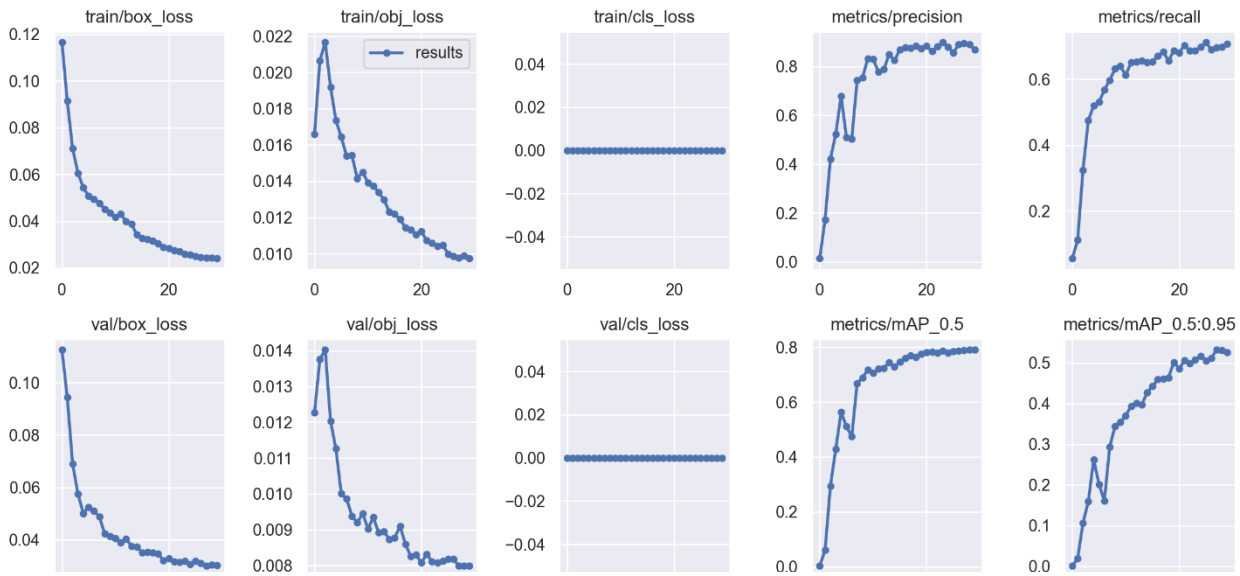

Figure 6 Precision Recall Plots for Dataset Train 50 - Test 50 YOLOv5n

3. For Dataset – Train 90% to Test 10%

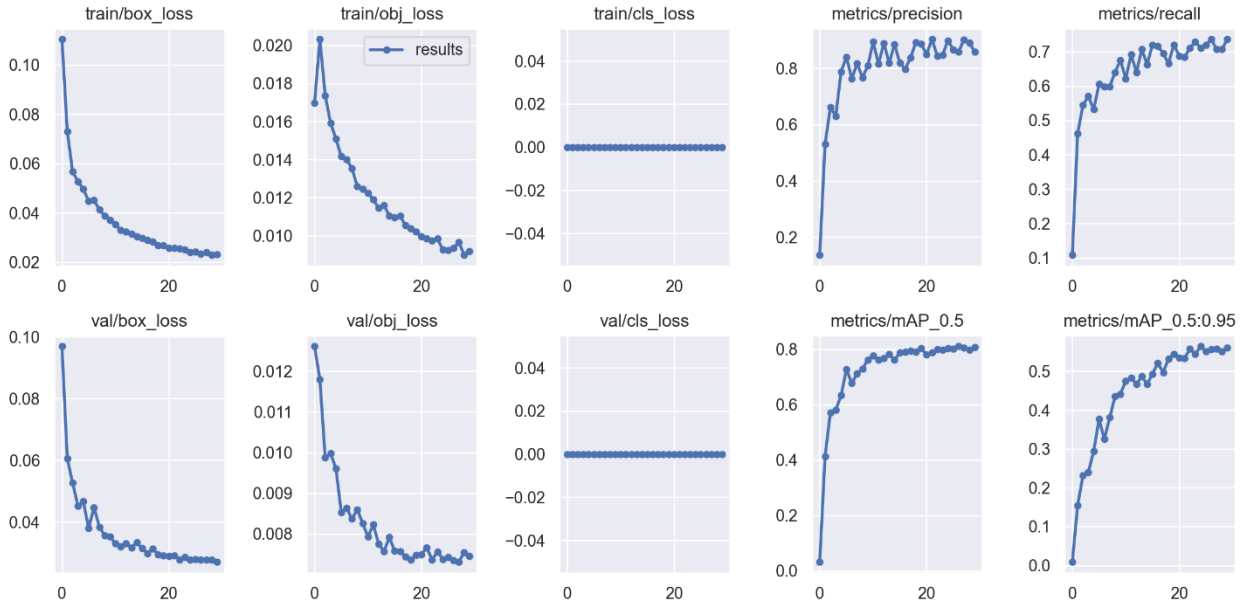

Figure 7 Precision Recall Plots for Dataset Train 90 - Test 10 YOLOv5n

Here are a few observations:

a. We can see that the rate of decrease of box loss for the train and test datasets increased with the increase in the size of the train dataset.
b. The precision and recall follow a similar trend that the rate of increase of precision and recall increased with the increase in the size of the train dataset.

4. For Dataset – Train 10% to Test 90%

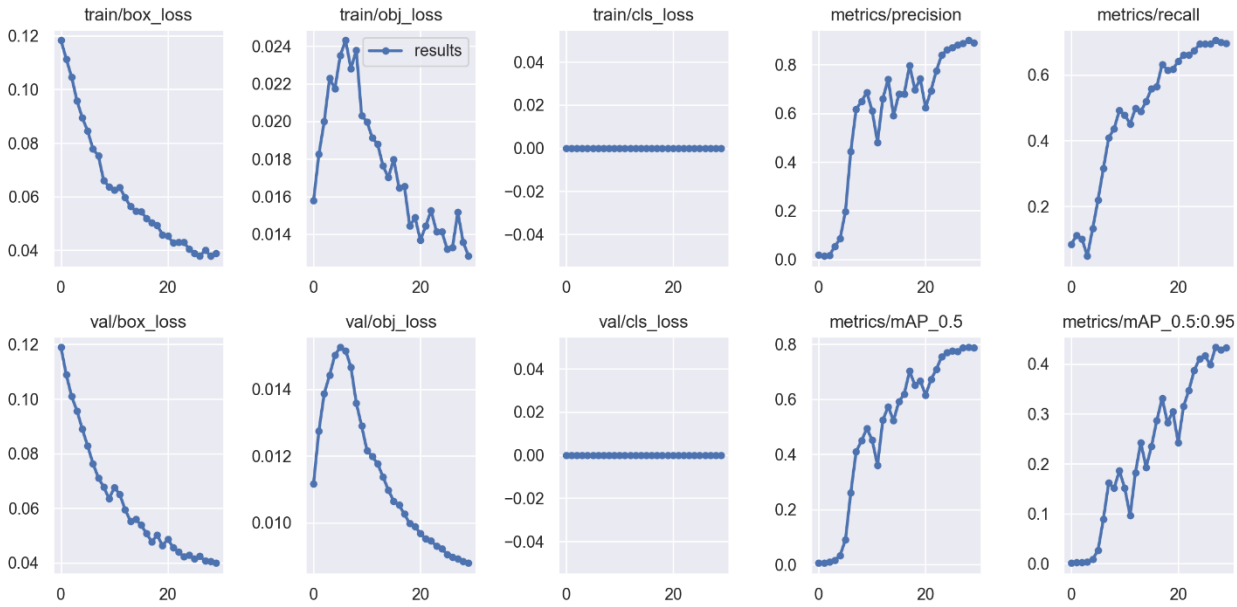

Figure 8 Precision Recall Plots for Dataset Train 10 - Test 90 YOLOv5s

5. For Dataset – Train 50% to Test 50%

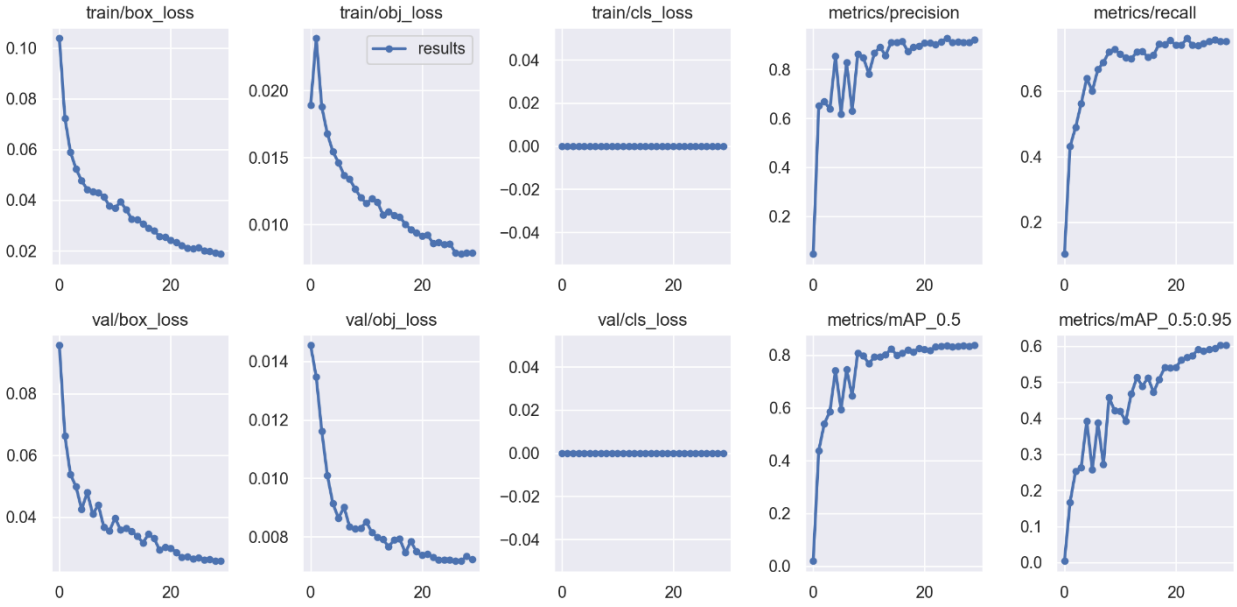

Figure 9 Precision Recall Plots for Dataset Train 50 - Test 50 YOLOv5s

6. For Dataset – Train 90% to Test 10%

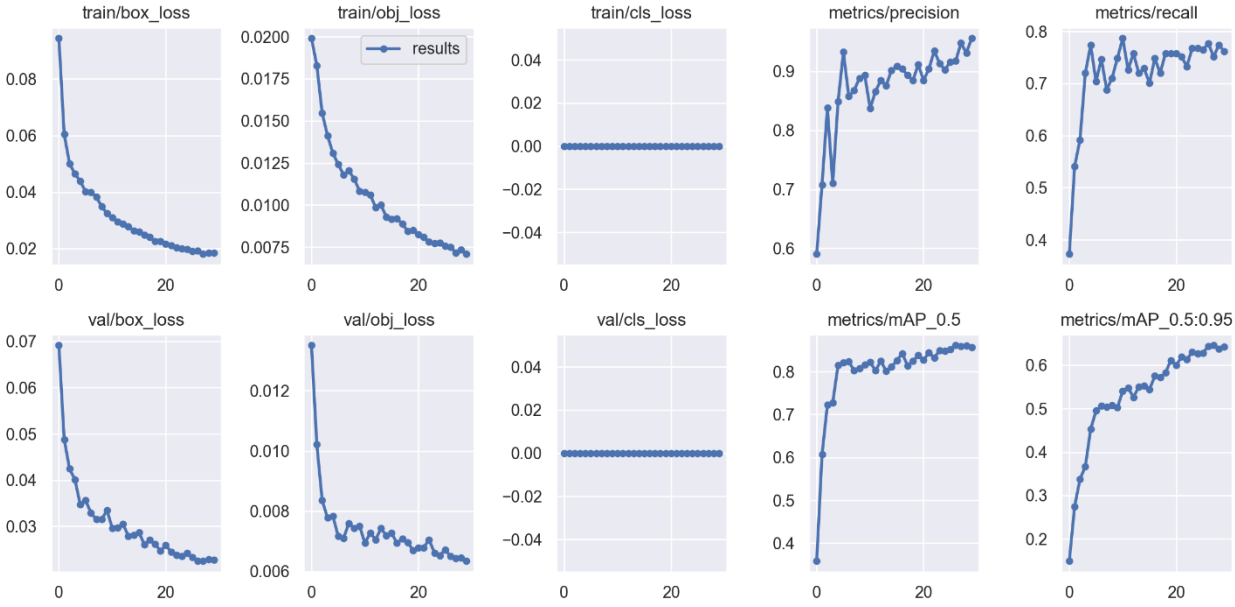

Figure 10 Precision Recall Plots for Dataset Train 90 - Test 10 YOLOv5s

Here are a few observations:

a. We can see that the rate of decrease of box loss for the train and test datasets increased with the increase in the size of the train dataset.
b. The precision and recall follow a similar trend that the rate of increase of precision and recall increased with the increase in the size of the train dataset.

YOLOv5m

7. For Dataset – Train 10% to Test 90%

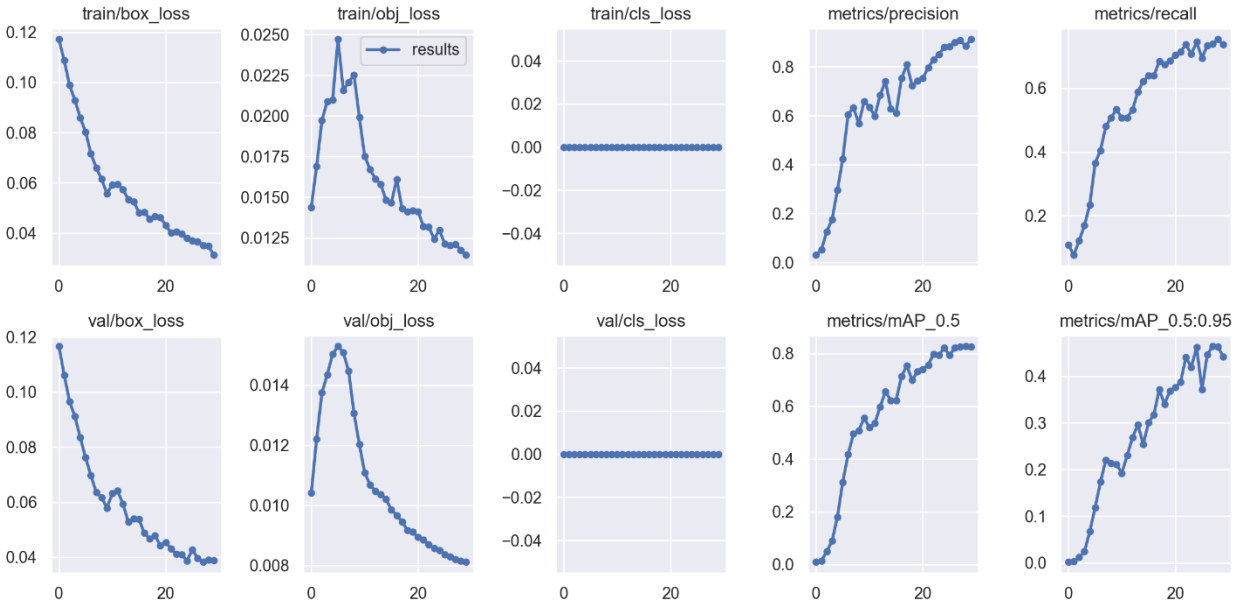

Figure 11 Precision Recall Plots Dataset Train10 – Test90 YOLOv5m

8. For Dataset – Train 50% to Test 50%

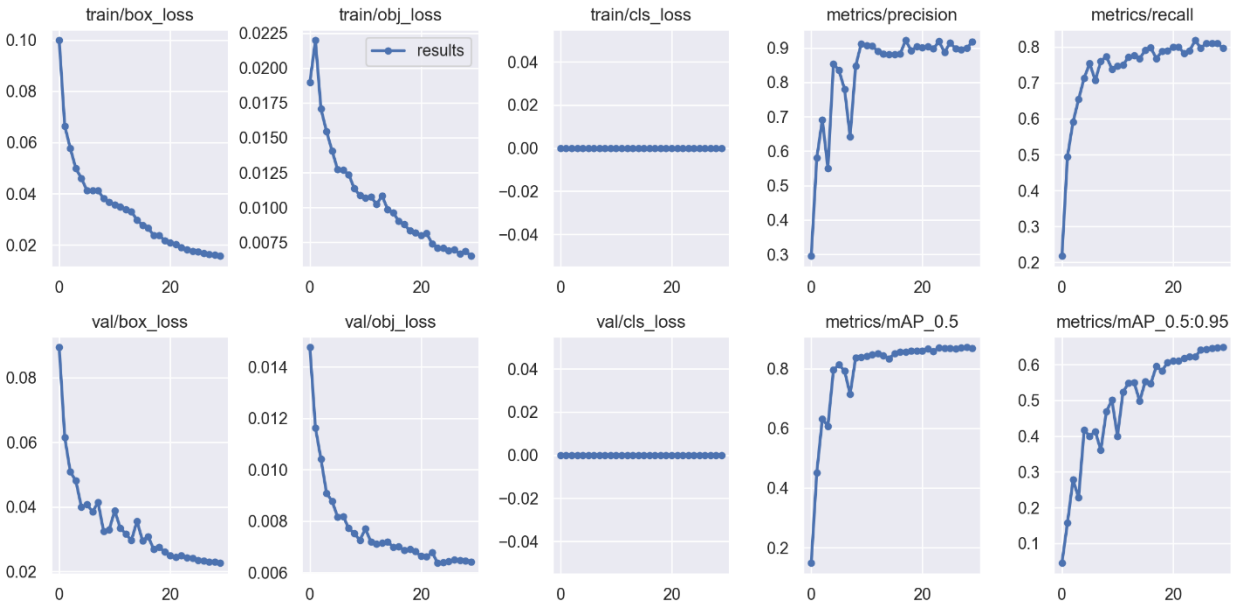

Figure 12 Precision Recall Plots Dataset Train50 - Test50 YOLOv5m

9. For Dataset – Train 90% to Test 10%

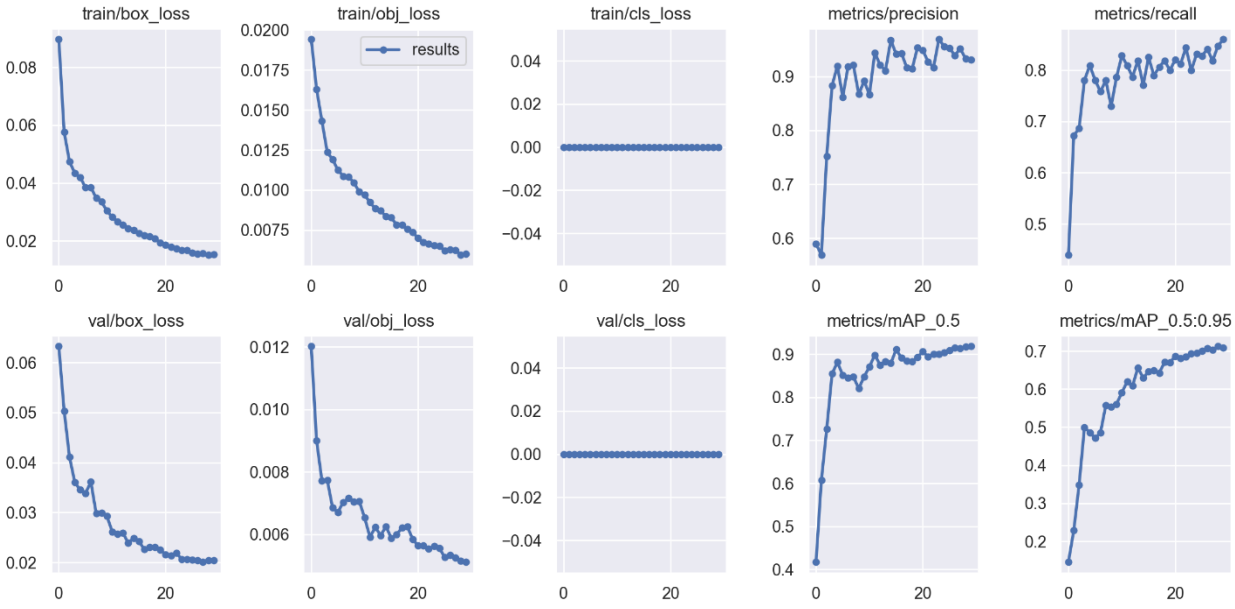

Figure 13 Precision Recall Plots Dataset Train90 - Test10 YOLOv5m

Here are a few observations:

a. We can see that the rate of decrease of box loss for the train and test datasets increased with the increase in the size of the train dataset.
b. The precision and recall follow a similar trend that the rate of increase of precision and recall increased with the increase in the size of the train dataset.

YOLOv5l

10. For Dataset – Train 10% to Test 90%

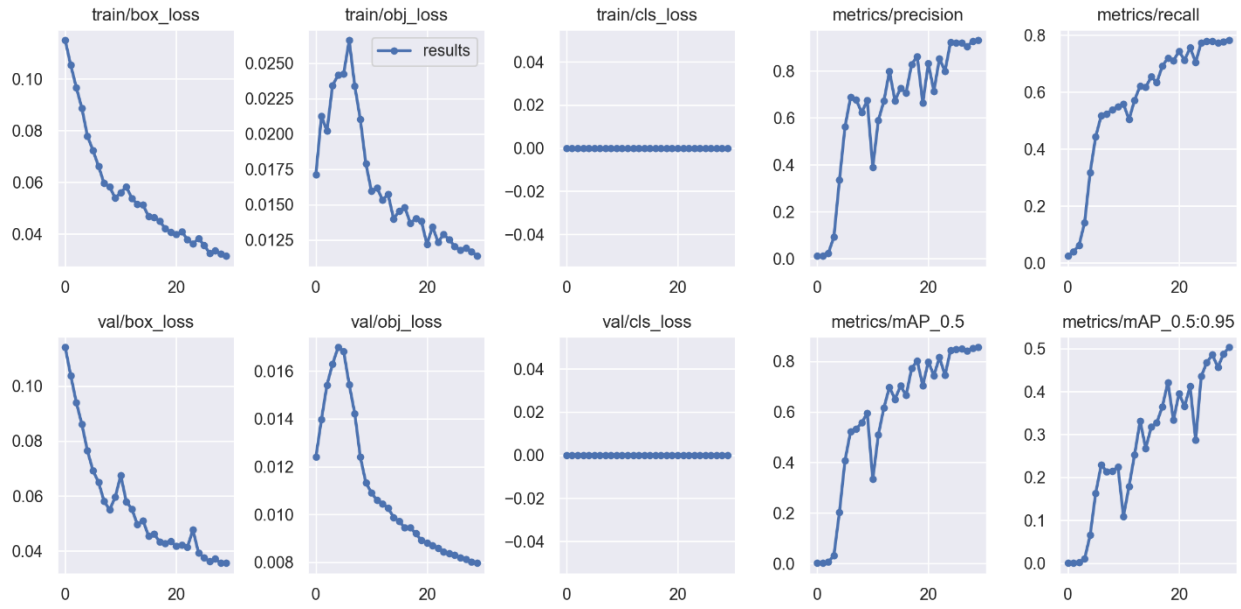

Figure 14 Precision Recall Plots Dataset Train10 - Test90 YOLOv5l

11. For Dataset – Train 50% to Test 50%

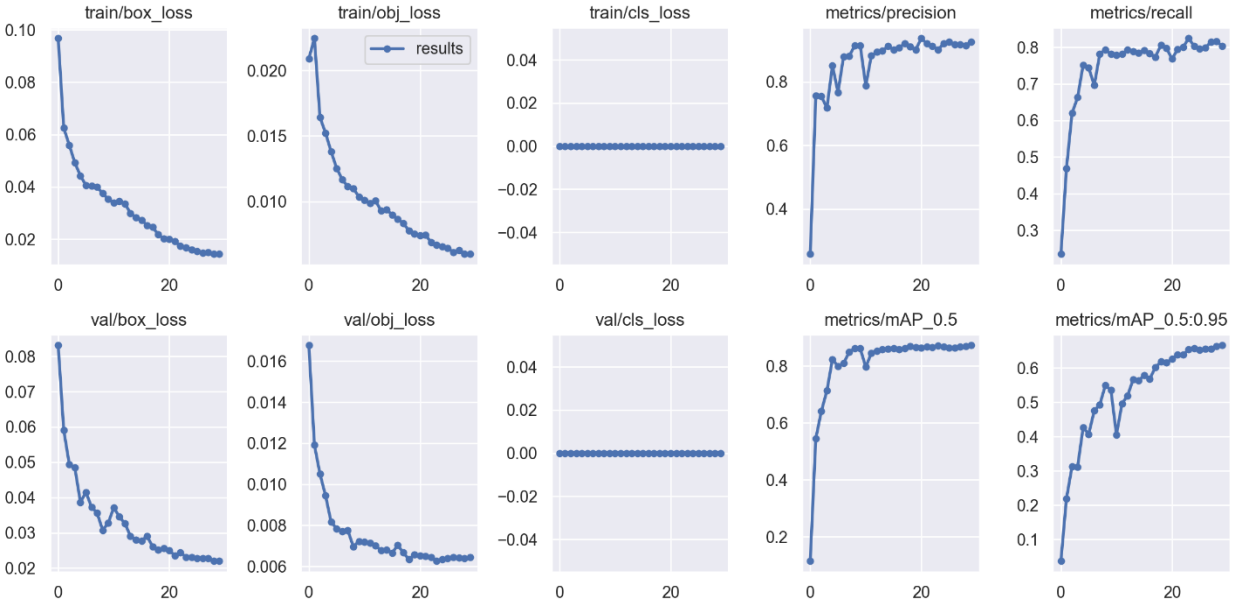

Figure 15 Precision Recall Plots Dataset Train50 - Test50 YOLOv5l

12. For Dataset – Train 90% to Test 10%

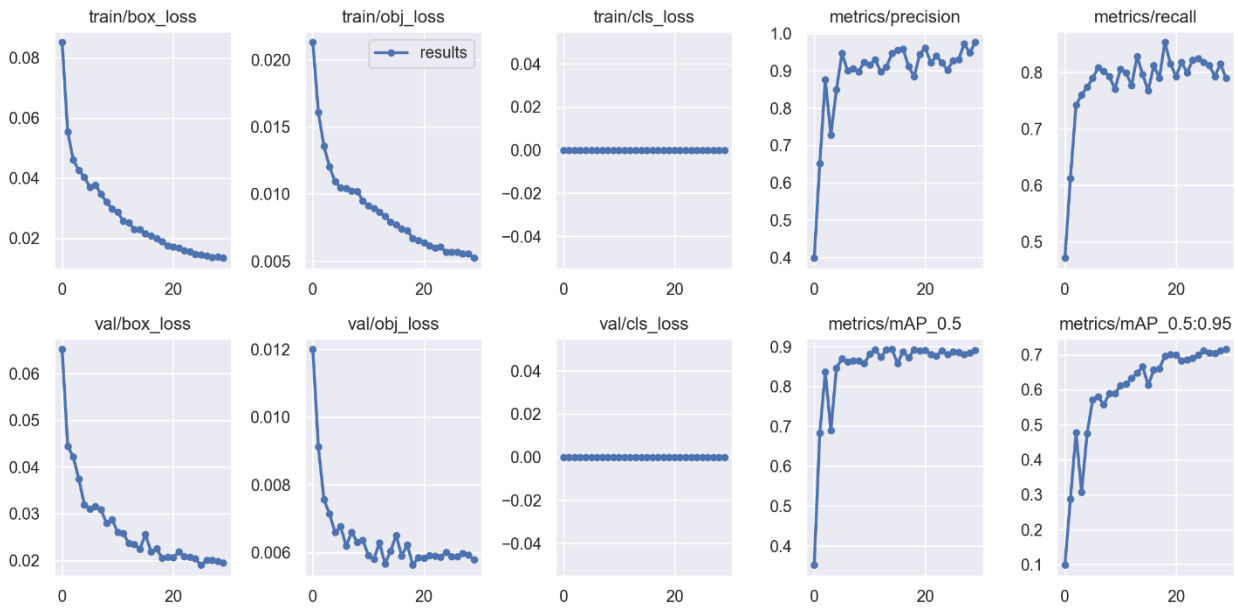

Figure 16 Precision Recall Plots Dataset Train90 - Test10 YOLOv5l

Here are a few observations:

a. We can see that the rate of decrease of box loss for the train and test datasets increased with the increase in the size of the train dataset.
b. The precision and recall follow a similar trend that the rate of increase of precision and recall increased with the increase in the size of the train dataset.

The plot results that are defined above show that the YOLOv5l algorithm had the highest precision and recall values for the train 90% and test 10% dataset but, we still couldn't use them because at the train dataset size 10% there was underfitting and when the train dataset was 90% there was overfitting.

Hence, the best results were obtained at the 50 – 50 dataset sizes where the precision recall was high enough but there was not much overfitting of the datset.

Object Detection from Images

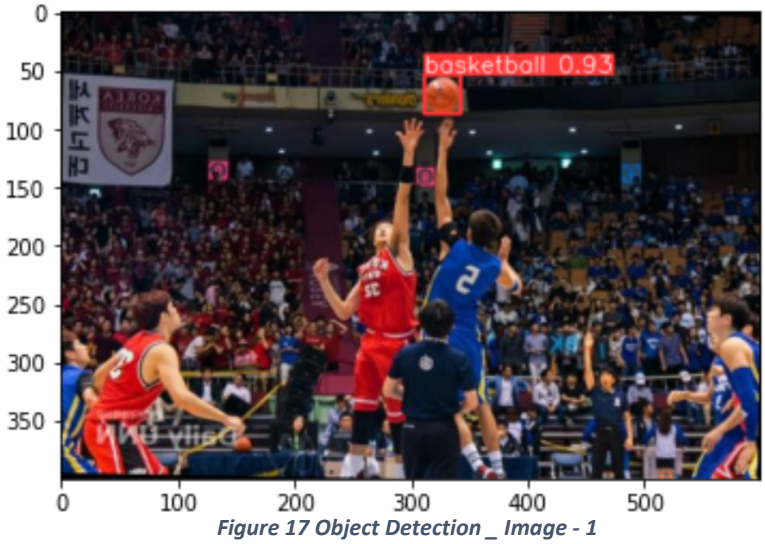

Figure 17 Object Detection _ Image - 1

Figure 18 Object Detection _ Image - 2

Figure 19 Object Detection _ Image - 3

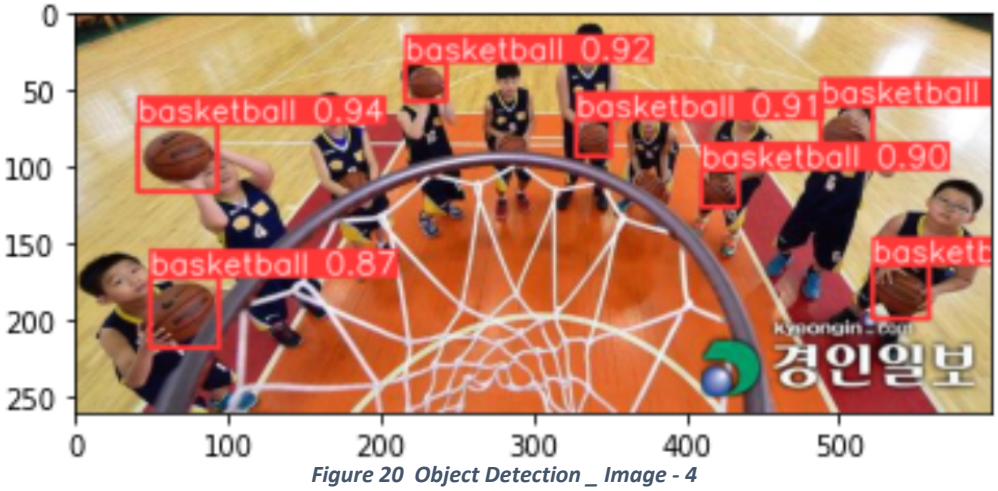

Figure 20 Object Detection _ Image - 4

Our trained algorithm was able to detect the basketball from images very much accurately and create a bounding box that included the probability of the object being inside the bounding box to be a basketball.

As you can see in the last image that some of the basketballs were not exactly detected therefore, we can say that there was a little bit of omission in the trained algorithm. But still the results are pretty accurate.

Object Detection from Videos

Youtube Link

Limitations and Future work

This work successfully detects the desired object (basketball) with great precision and accuracy. Although some limitations were observed: At higher ratio of train to test data size overfitting was observed such that model labelled some non-basketball objects as basketball. On the other hand, when the ratio of train to test was reduced to improve computation time, it was observed that model severely under fit.

The work can be further extended to detect different objects simultaneously. With enough computational capabilities this model can be implemented to detect object in real-time.

Conclusion

In this work, different version of Yolov5 (YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l) algorithms were implemented on a basketball image data set. The basketball image dataset was converted into a Yolo readable file format (.YAML). Lastly all these algorithms were implemented with different ratios of test and train data set. We successfully detected our desired object (basketball) in different images with ambient background as well in video with great precision and accuracy. Based on the results it was concluded although YOLOv5l performed overall better, but the best results were obtained at YOLOv5m when the dataset test train split ratio was 50/50. The algorithm converged to a value of 90% recall in under 10 epochs.

Acknowledgement

This project was performed under the guidance of Dr. Michael A.Gennert. We, as team H, would like to express our appreciation for Dr. Gennert's great and insightful contributions to the team throughout the duration of the course. We would like to thank Dr. Gennert for his valuable and constructive suggestions through the assignments and lectures conducted for the duration of this project work.

Our grateful thanks are also extended to each of the team H's members for working hard on this project and constantly inspiring, motivating each other while creating a comfortable yet productive environment.

References:

[1] YOLO-Tomato: A Robust Algorithm for Tomato Detection Based on YOLOv3

Liu, G.; Nouaze, J.C.; Touko Mbouembe, P.L.; Kim, J.H. YOLO-Tomato: A Robust Algorithm for Tomato Detection Based on YOLOv3. *Sensors* **2020**, *20*, 2145. https://doi.org/10.3390/s20072145

[2] You Only Look Once: Unified, Real-Time Object Detection

Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788

Appendix

The code was generated in Jupyter Notebook and is attached with the submission file. As well as attached below.

Label BB Images.ipynb

YOLO.ipynb