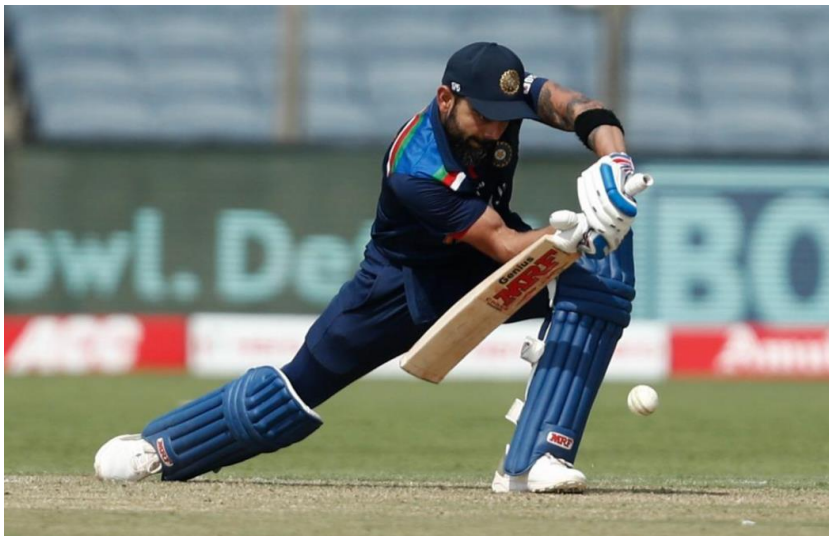
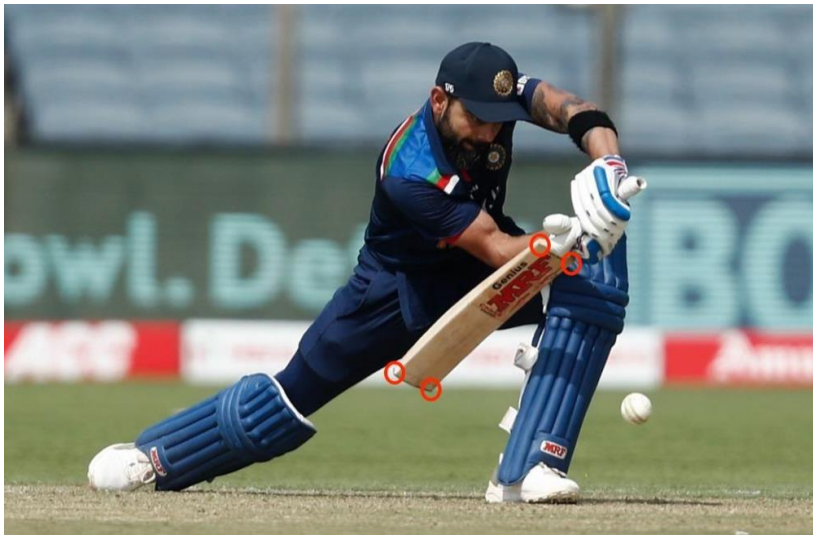
CRICKET BAT CORNER DETECTION

Git repo link : https://github.com/Amalvs007/Cricket_Bat_Corner_Detection_YoloV5

The problem is to predict the 4 corners of the cricket bat in the given input image.



Desired Output



I have decided to use **YoloV5** model for this object detection problem. So, the first step is to collect quality data for my model. I

have checked many sources and I couldn't find any direct source that provide data set for this problem. So, the next step is obviously to build the data set my own. For building the data set, when I researched and I have found some Kaggle data sets that contain images of cricket shots, and another data set with cricket images and cricket player images. I download these datasets and gone through all the images and I only selected good quality images with a condition that is **"Images with a player holding the bat on field"**. I used this condition because the given input image have a player holding the bat on filed.

I only got a very few little images that have good quality and satisfying the condition. So, for rest of the Images, I use google image search. After carefully selecting good quality images and images satisfies the condition *(although I used some images that does not completely satisfies the condition)*, I am left with 110 images.

Since, 110 images is not enough for developing an object detection model, especially when dealing with very small objects, I have decided to augment the data.

Since its an object detection task, labelling of the image is drawing bounding boxes at each object in the image. So, If I do labelling after the image augmentation, that will lead to consuming more time. Since, time is very limited in this project, I have decided to Image augmentation after labelling the images.

I used **Labelimg,** for labelling the images in yolo format. I used a python script file for renaming every image, to numbers before labelling, so that every images and corresponding label text files will have same and unique names.

When labelling, I used only 1 class which is "**corner**", so that the models performance can be improved and complexity can be reduced. After labelling the 110 images, I have decided to start **augmentation**

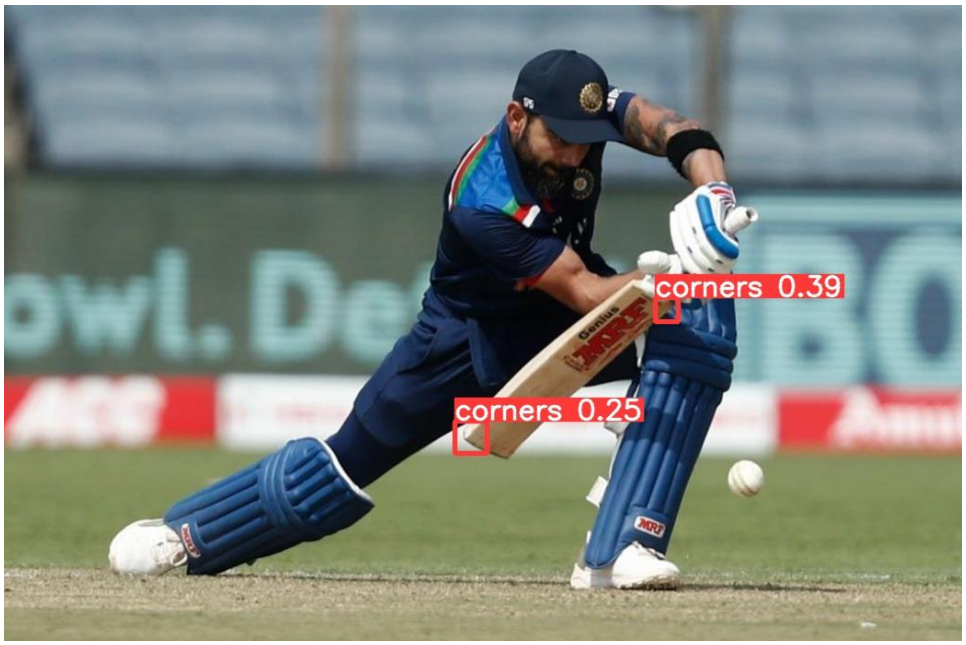
process on images. So, the requirement is to augment the image and also the corresponding label as well.

I used **Albumentations library**, for augmenting both the image its corresponding bounding boxes. After augmenting, the images and labels must be saved also. So, I have created a python notebook named **Augmentation.ipynb**, for this process. I have used 7 different augmentations, that is applied on all 110 images and generated 880 total images after augmentation.

Now for training, I split the data set with **75%** data used for training and **25%** data for testing.

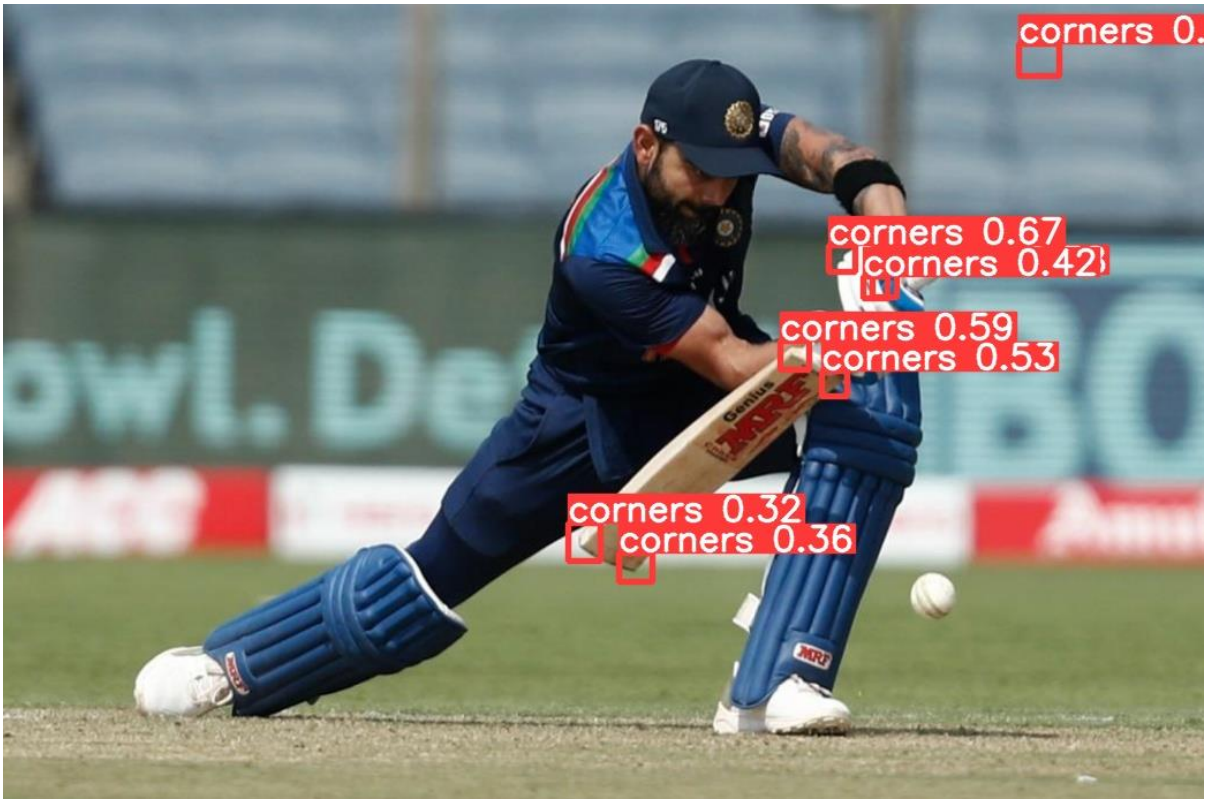
For training purpose, I have created **Cricket Bat Corner Detection.ipynb** file where I clone the official git repository of Yolov5 model and have done the rest of the operations for training the data set.

First, I trained the model with only **50 epochs**, and from that check point I only got **precision value of 49.4%** and a **recall value of 32.1%**. This comparatively a low performance model and there is still room for improvement. Even though the performance is low, the model still predicts two corners correctly from the input image.



So, I decided to make the model even better and decided to train from the last check point. I have given **300 epochs** and the compiler stops running after **280 epochs** due to some unexpected error. Luckily, the checkpoints were saved at regular intervals automatically, So that I got a model with 280 epochs. So, now the obtained model is trained with **330 epochs** *(Since we started from last check point)* and have **precision value of 90.2%** and a **recall value of 83.5%**. Now the performance, of the model is really great in numbers as it shows, but model is actually Overfitting. when I checked with some images, the model is detecting unwanted objects along with correct predictions.

The prediction of the input image is shown below.



The model actually detected the 4 corners as per requirement, but additionally, unwanted objects are also detected. This is not good for the model.

So, I decided to train the model once again from start with **150 epochs** and image size as **512**, and I have started training but unfortunately, I couldn’t complete the training. As, I was using

Google Collab, the free time for using GPU given by google collab was expired. So, no more training was possible.

Also, I have created a **Final_output.ipynb** file for predicting and displaying the obtained output.

(Since the submission dead line was on Sunday, I need to submit at least now. So, I can't wait for GPU recovery as it will recover only after so many hours. Also, I need to mention that I have started doing this project from the next day I got the mail, but due to an unexpected situation that is a complaint with my laptop, I couldn't do the work on Thursday and Friday. I resumed the work only on Saturday evening. That's the reason, I couldn't complete the project on Sunday. I was only training the model on Monday morning, and the GPU exhausted at evening and has not recovered till now. So, I am submitting the project with what I have got.)

Challenges faced and how I handled them

- Collecting the exact images was the most challenging part in this project. Finding the images with the satisfied condition that I mentioned above was really challenging and time consuming. Taking images and selecting from Kaggle data set was easy and I do save some time there.

  But from google images, it was tiresome process. Some images will be in webp file, and it needs to be converted to jpg. So, I have used online converter which takes links of image source and give jpg image for downloading.

- For making dataset big, augmentation is used. It will be time consuming and tiresome work if labelling is done after augmentation. So, I have decided to do augmentation after labelling and make augmentations for labels also. This, speed up the process by 70X, with maintaining exact accuracy of labelling.

- The next challenging part was using Albumntations library, as I have never used this library before and I have never done augmentations for both images and labels before. So, understanding it's concepts and using the library was really time consuming. But I have spent enough time on it and try out with sample images and checked my self how it works.

- Creating the **Augmentations.ipynb** file was also a challenging part, as the code contain so many functions for different requirements and I really spent so many time for finding the code for certain functions. I got so many error and with my debugging skills, I finally made the code error free and with the code we can fully automate the process of augmenation.

- Not having a guidance was a major challenging part as I was having so many ideas for crafting code, and I was having uncertainty whether it will work or not. So, my only option is to try it out by myself and If I got error, I will have to do another way. This process has taken so much of my time while doing the project. Also, I saved some time as some of my doubts were cleared by chat GPT.