Chess Objects Detection

Introduction :

There is a lot of objects detection methods, But Most of the recent successful methods are based on convolutional neural networks (CNNs), One of the Neural network approaches is You Only Look Once (YOLO), which we going to deal with in this project.

The first thing that I came through is to find the data set, either install a pretrained one which is for a chess object could be difficult due to the variation in chess objects shapes and colors, so the other choice was to train my own one. Starting by collecting images for my chess board using my phone camera.

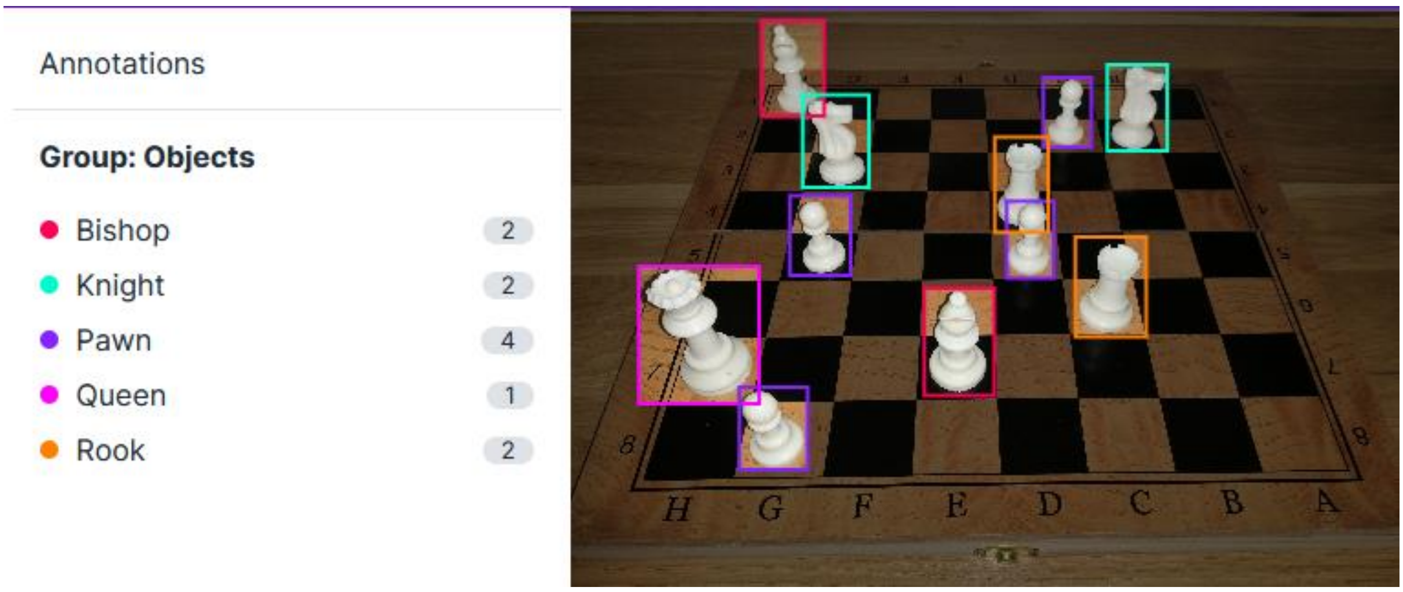
I start taking some photos as shown below :

But I came across some difficulties with training process due to the light and the colors contrast.

So I made my small studio and tried again, and after hours of training the detection accuracy was almost non-existent, and I found out that it's because the objects are looking so similar to each other which need a lot more time training, which was not a choice.

Until I found those old chess objects which make the operation much easier. (a lot of pieces were missing ^^, But if it worked for one it works for the rest).

Taking 33 images from the same perspective, with a rando m positioning of the objects, and with the help of Roboflow framework, which made the annotation process (specifying the coordinated of the bounding boxes around each object to use it in the training process) , Preprocessing the images and resizing them more organized.



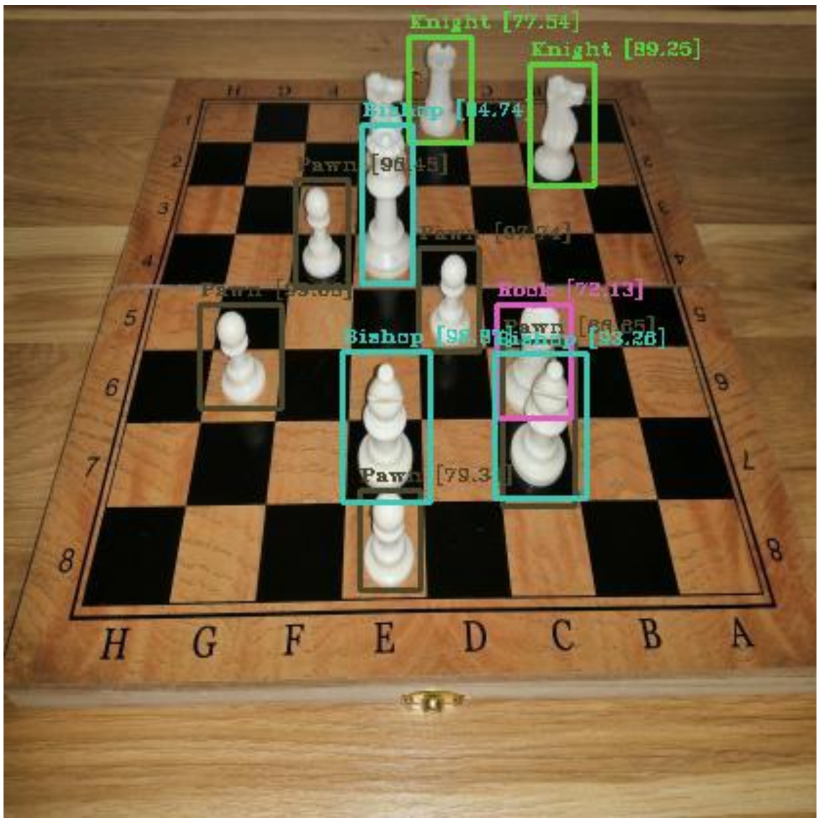
Annotations

Group: Objects

● Bishop	2
● Knight	2
● Pawn	4
● Queen	1
● Rook	2

After we annotate our images it's time for training, and because it takes a long time on non-gpu-based machine, I had to use google colab framework for the rest of my project.

Starting by configuring the machine, and installing Darknet for Yolov4, Specifying the number of classes which had to be 5 instead of 6 due to – mysterious disappearance of the king – , followed by making a custom configuration file with a maximum number of batches equal to 500 (it had to be 10000 but this going to take days of training), finally after the training process we get the .**weight** file which we going to use in the detection soon.

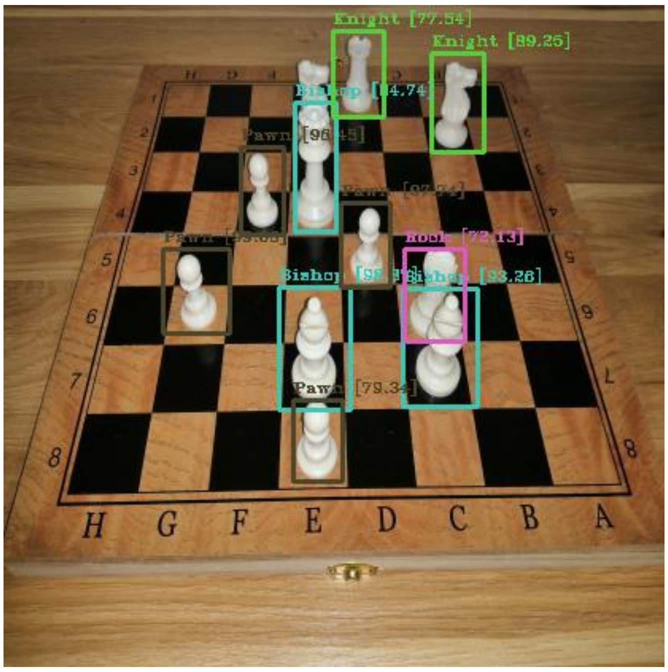


the detection result.

Redundant Bounding Boxes:

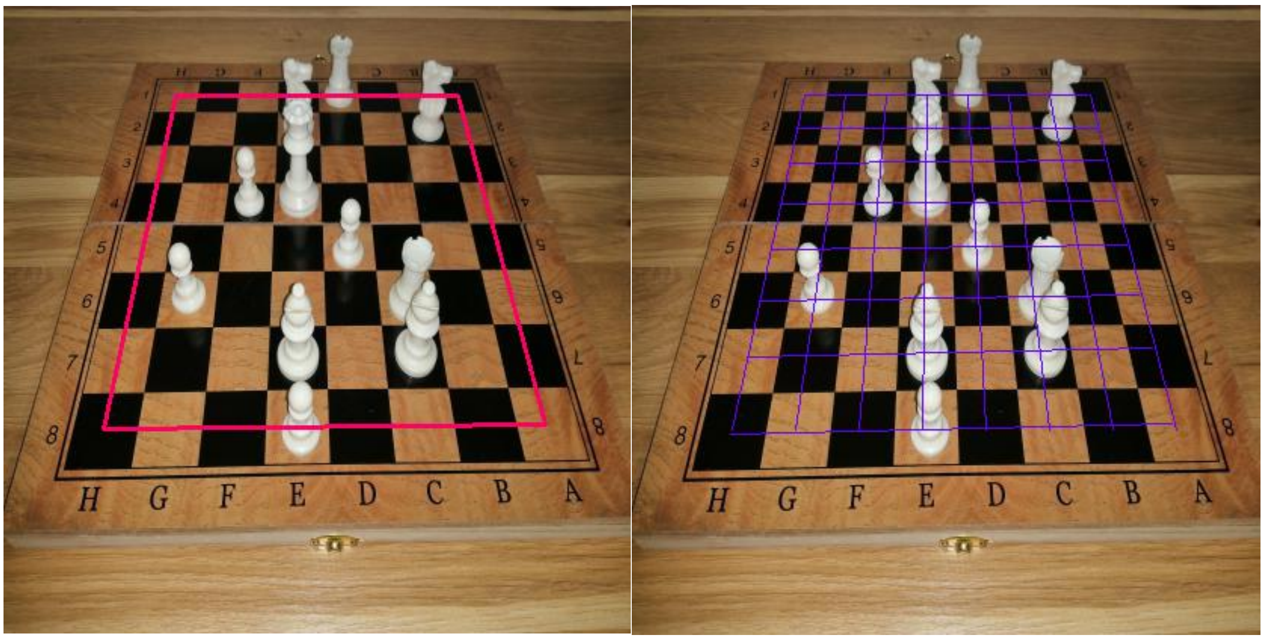
if you looked carefully in the above image, you will notice on location "C7" the bishop had two predictions, one as bishop with confidence of 93.28%, and the second one as Pawn with 88.87% confidence.

To deal with that, there is a commonly used technique called **Non Maximum Suppression (NMS)** that select one entity (e.g., bounding boxes) out of many overlapping entities.



Locating The Objects On The Board:

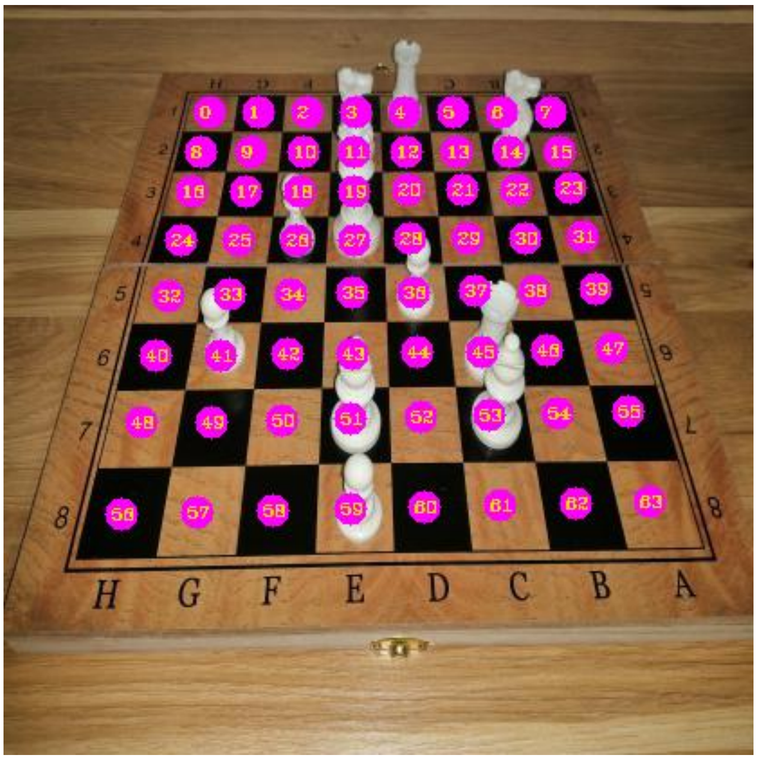
Since we are looking at the board from the same side the simplest procedure I saw is to detect the corners of the board. Which I made it manually, connecting the points together, then divide each line into 8 points, connecting each point on each line. to the corresponding point on the corresponding line. Until we got the grid shape.



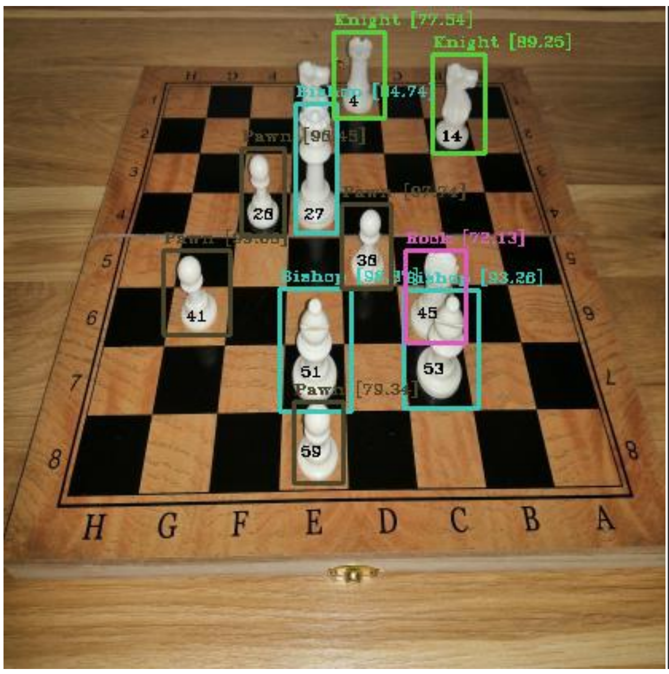
Now due to what's called **Fisheye Effect**, the horizontal lines must not be equally separated, But for our fixed view they follow this pattern of ratios (found by testing)

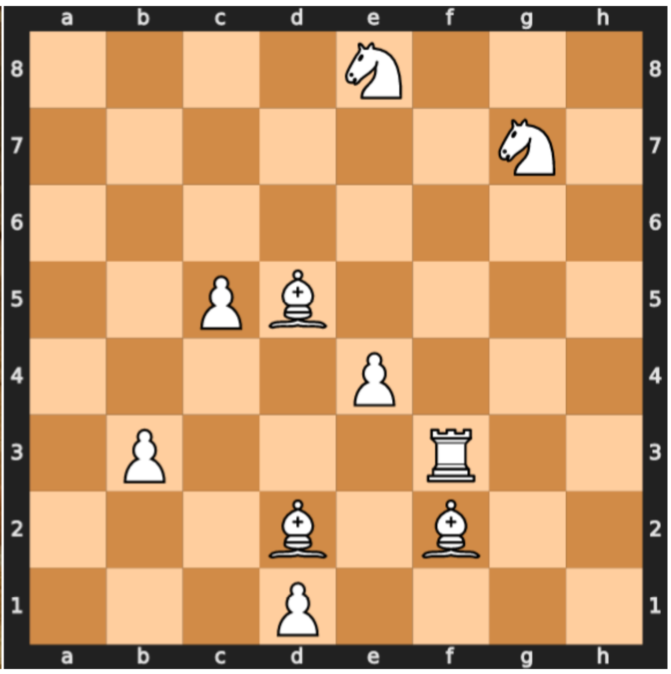
```
Ratios = [0, 0.7, 0.7, 0.75, 0.8, 0.85, 0.9, 1]
```

After that we find all the intersection points knowing the points on each side of each line.



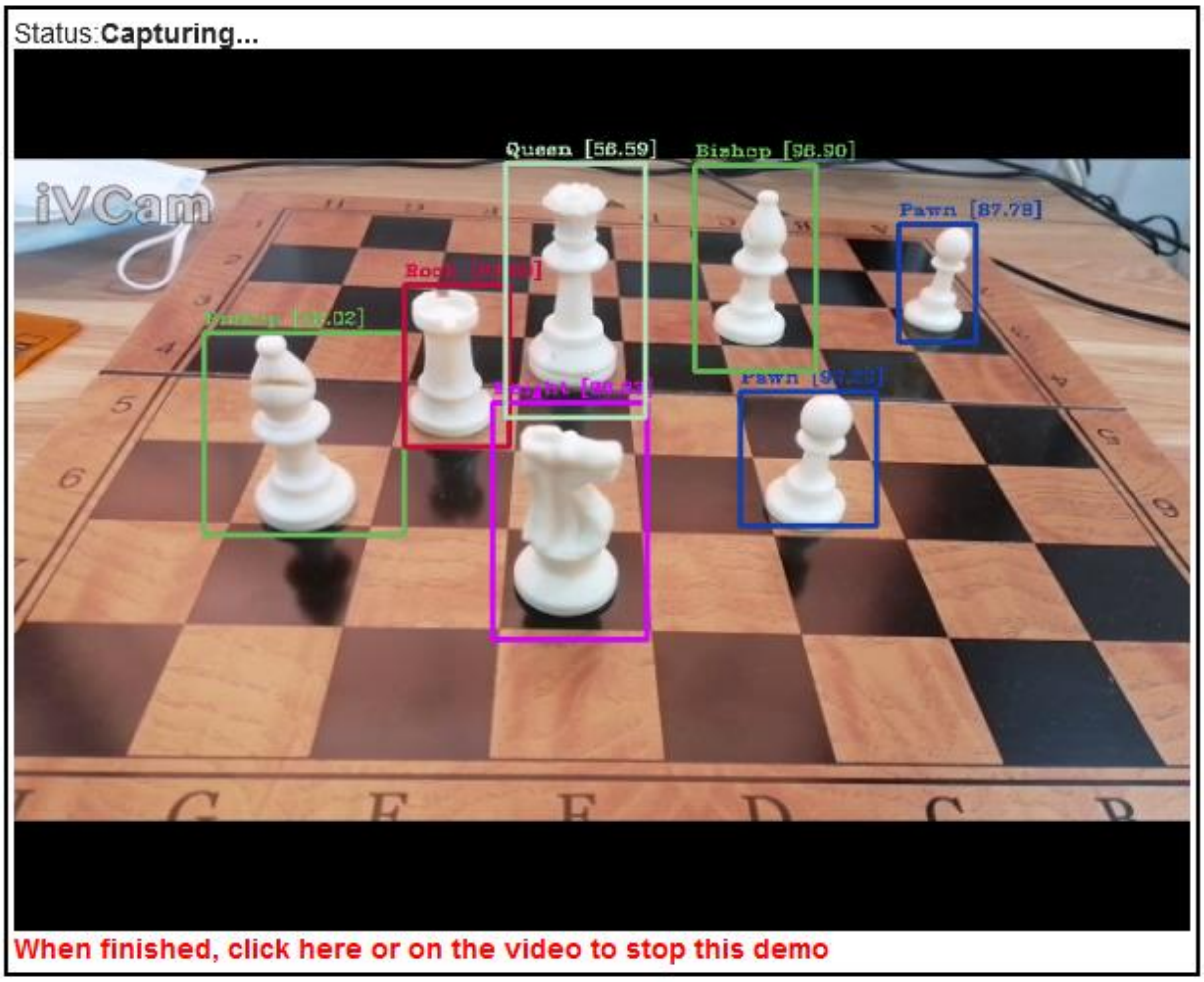
In the end, we just combine the detected objects locations with the grid informations to point on each object and convert the view into two dimensional one.





Of course, because the training period was relatively short, sometimes we will see some lack of accuracy. But giving it more time of training should deal with that.

Real Time Detection:



Status:**Capturing...**

When finished, click here or on the video to stop this demo

Because we are working on colab, which dosen't have a build in video webcam utility, So I borrowed a custom made snippit of code that helped with that

Thank you for reading

Osama Alsharif

My Github Repo : https://github.com/o54ma-4l5h4r1f/ChessObjectsDetection