National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Shaik Rizwana
Student ID: 22114611

School of Computing
National College of Ireland

Supervisor: Vikas Tomar

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Shaik Rizwana
Student ID:	22114611
Programme:	Data Analytics **Year:** 2023
Module:	MSc Data Analytics
Lecturer:	Vikas Tomar
Submission Due Date:	14/12/2023
Project Title:	Firearm detection using Yolov7
Word Count:	1177
Page Count:	9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: .....

Date: 14/12/2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Shaik Rizwana
Student ID: 22114611

1 Introduction

This document serves the purpose of recreating the output provided by the code artifact. We will go through several steps and require libraries to understand how to perform gun *detection using the Yolov7* model.

2 Requirements

There are two phases of working with this code. I have used Google Colab to train the model and perform testing in the local system.

2.1 Google Colab

Since this project is a computation-heavy model, we are going to leverage Google Colab and its GPU. Google Colab provides us with the following specifications –

1. GPU type – Tesla K80 Nvidia
2. GPU provided – 12.6 GB
3. Disk size provided – up to 80GB
4. Up to 5 to 6 hours of free computation time

In order to select the required GPU in Google Colab sheet, you can follow the below screenshots –

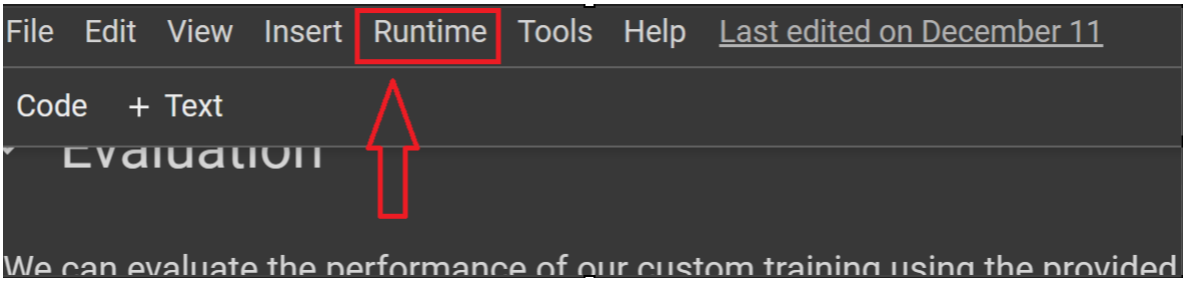
1. 

Figure 1(a) : Step 1 to change runtime GPU in google Colab

2. Click on ‘Change Runtime’ and select T4 GPU like the below screenshot.

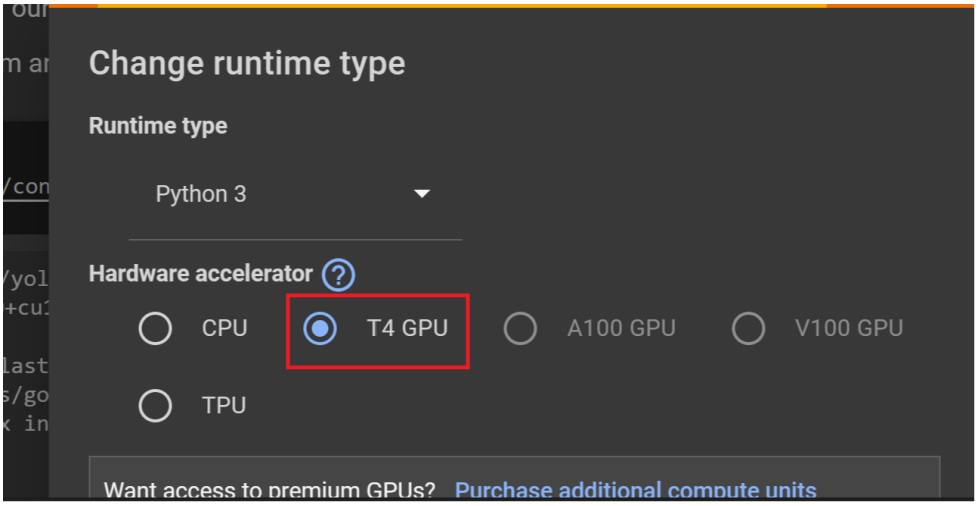


Figure 1(b) : Step 2 to change runtime GPU in google Colab

2.2 System Configuration

1. OS – Windows 11
2. Processor - 12th Gen Intel(R) Core(TM) i7-1250U, 1100 Mhz, 10 Core(s), 12 Logical Processor(s)
3. RAM – 16 GB

3 Software tools

1. Python 3 with PyTorch and TensorFlow
2. Yolov7 – Object detection state-of-the-art algorithm
3. Roboflow – Data management platform

4 Datasets

This project has utilized two different datasets. Both have been cleansed and uploaded on the Roboflow platform, as it is easier for the API to fetch the datasets when required.

1. COCO(Common object in context) dataset - https://universe.roboflow.com/gun-detection-with-yolo/gun-detection-using-yolo-i

2. Custom Dataset - https://universe.roboflow.com/gun-detection-with-yolo/gun-detection-using-yolo_ii

5 Setting up the Environment

We have several steps to import the required repository and use the Yolov7. The official creators of Yolo provide this. i.e., Ultralytics. The datasets are uploaded on Roboflow in separate project repositories.
These steps are performed using Google Colab. Below are the steps.

1. Importing the required repository.

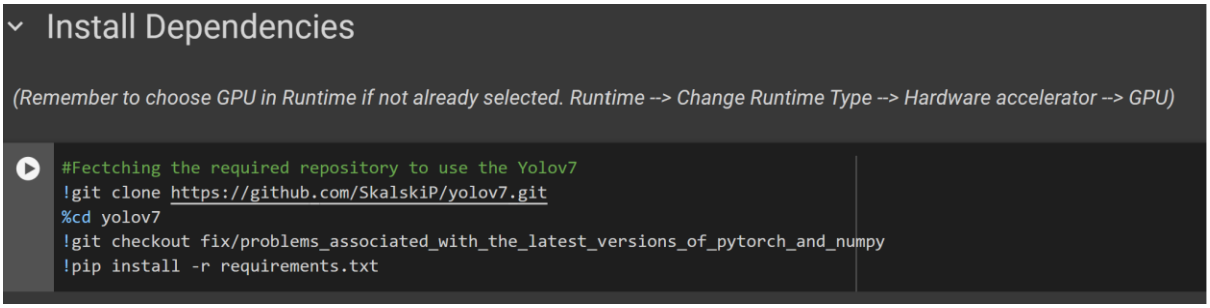


Figure 2 : Installing dependencies in google Colab

2. Import the two datasets –

Figure 3(a): Importing Dataset 2 in google Colab from Roboflow

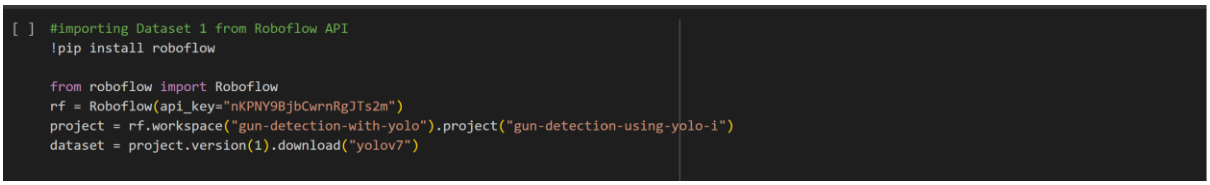


Figure 3(b): Importing Dataset 1 in google Colab from Roboflow

6 Implementation

Let us now look at the steps to implement the model and train, test, and validate it.

1. Download the required pre-existing best weights (a pre-trained) of the model.

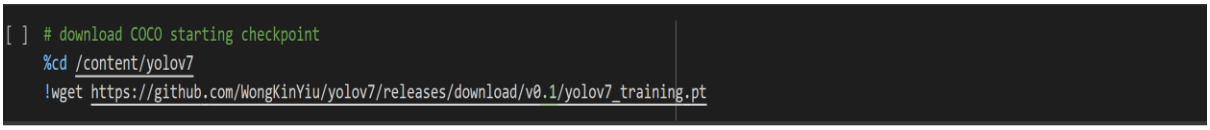


Figure 4: Importing best weight of an existing Yolov7 mode in google Colab

2. We need to edit the ‘coco.yaml’ file so that the train, test and validate data path are automatically picked up by the model.

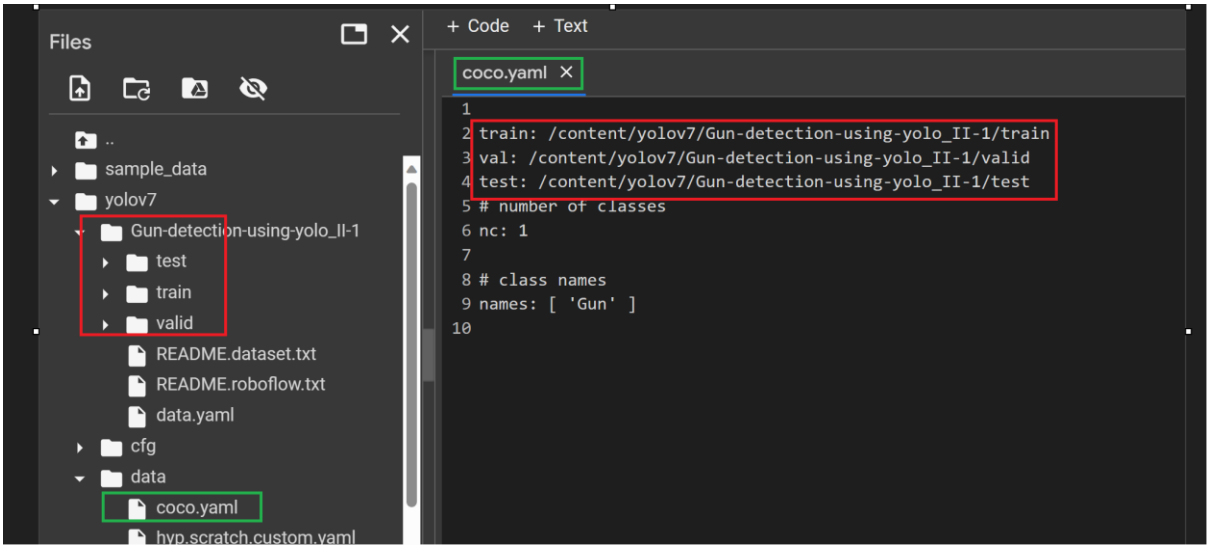


Figure 5: providing the paths for test, valid and train data splits in coco.yaml file

"coco.yaml" file is present under the "data" folder. The path can be provided from the downloaded datasets in red boxes under the 'train, test and valid' path in coco.yaml file.
Save the file for it to reflect for the next steps.

3. We are going to run the training syntax next. The batch size is 16 so there is not a lot of pressure on the GPU for RAM. Epochs chosen can be tested from 50 to 100. Here, we have selected 100 epochs to train the final models well.

```
[ ]  # run this cell to begin training
     %cd /content/yolov7
     !python train.py --batch 16 --epochs 100 --data /content/yolov7/data/coco.yaml --weights 'yolov7_training.pt' --device 0
```
Figure 6: Syntax to run training for the datasets.

```
      Epoch   gpu_mem       box       obj       cls     total    labels  img_size
      96/99     13.1G   0.05525  0.007444 4.111e-05   0.06274        34       640: 100% 31/31 [00:26<00:00,  1.16it/s]
                Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 108/108 [01:03<00:00,  1.69it/s]
                  all        3445        3423       0.712       0.176       0.155      0.0684

      Epoch   gpu_mem       box       obj       cls     total    labels  img_size
      97/99     13.1G   0.05647  0.008157 4.612e-05   0.06467        46       640: 100% 31/31 [00:28<00:00,  1.09it/s]
                Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 108/108 [01:03<00:00,  1.71it/s]
                  all        3445        3423       0.717        0.18       0.155      0.0687

      Epoch   gpu_mem       box       obj       cls     total    labels  img_size
      98/99     13.1G   0.05604  0.007953 4.395e-05   0.06404        29       640: 100% 31/31 [00:28<00:00,  1.10it/s]
                Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 108/108 [01:03<00:00,  1.70it/s]
                  all        3445        3423        0.73        0.18       0.159       0.069

      Epoch   gpu_mem       box       obj       cls     total    labels  img_size
      99/99     13.1G   0.05279  0.007942 4.391e-05   0.06078        22       640: 100% 31/31 [00:28<00:00,  1.08it/s]
                Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 108/108 [01:07<00:00,  1.61it/s]
                  all        3445        3423       0.724       0.178       0.156      0.0669
                  Gun        3445        3421       0.448       0.356       0.313       0.134
               Pistol        3445           2           1           0           0           0
100 epochs completed in 2.734 hours.

Optimizer stripped from runs/train/exp2/weights/last.pt, 74.8MB
Optimizer stripped from runs/train/exp2/weights/best.pt, 74.8MB
```
Figure 7: Final output of the training of the dataset. The red box shows the final output.

4. In order to save the output with the graphs and best weights we can use the below code to download the zip folder of the experiments.

```
]  from google.colab import files

   # Define the file name you want to download
   file_name = "/content/yolov7/runs/train/exp"

   # Use the files.download method to save the file to your local system
   files.download(file_name)
```

Figure 7: Code to download any folder from the Google Colab.

5. Let us not switch to the visual studio and pull the repository into our local systems using the same code as step1, shown in figure 2.

```
1   Select a Language, or fill with template, or open a different editor to get started.
    Start typing to dismiss or don't show this again.

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Rizwana\Desktop\exp\yolov7-main\yolov7-main> #Fectching the required repository to use the Yolov7
>> !git clone https://github.com/SkalskiP/yolov7.git
>> %cd yolov7
>> !git checkout fix/problems_associated_with_the_latest_versions_of_pytorch_and_numpy
>> !pip install -r requirements.txt
```

Figure 8: Code in visual studio command line to pull the required repository.

6. After all the folders are pulled and requirements are installed, We copy the best.pt in the root of the project to use it for testing. A simple drag and drop of the best.pt should do the trick. I have renamed them according to the dataset I trained the Yolov7 model with for my convenience.

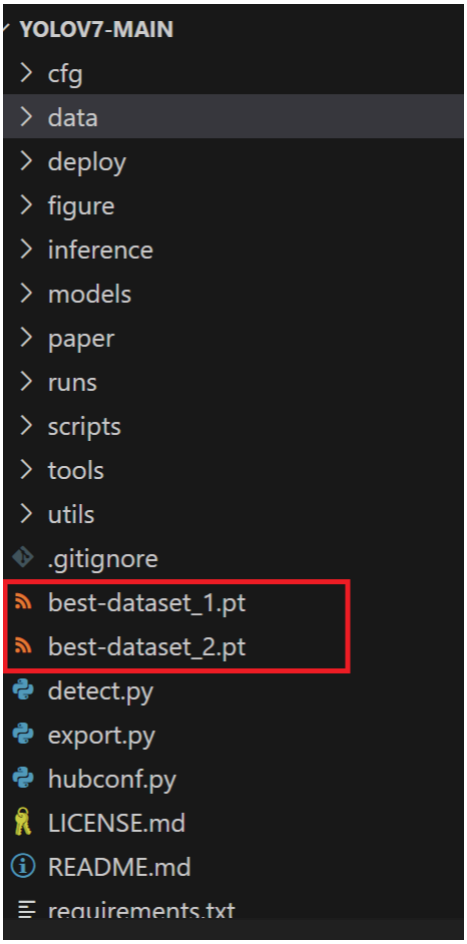


Figure 9: Best weights are pasted in the root path of the project pulled into Visual studio.

7. One final change before we start testing our model. The 'coco.yaml' file under 'data' folder needs to be changed according to the dataset we are using. Change the path of 'valid' dataset path so that model picks it up for testing. This has been set as default value to be picked up. We need to change only path for '**val**' with test dataset split.

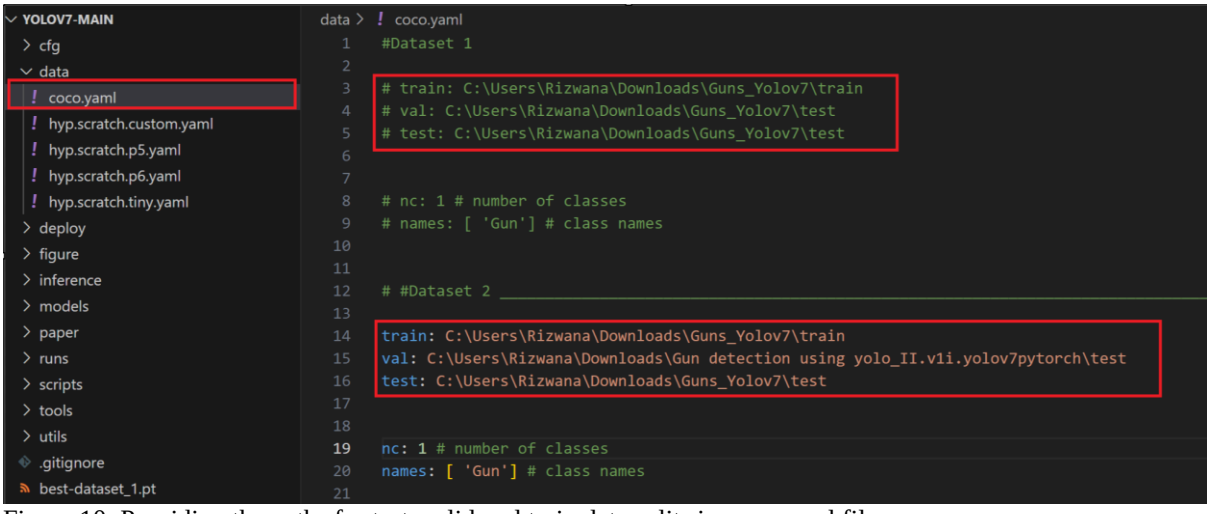


Figure 10: Providing the paths for test, valid and train data splits in coco.yaml file

8. Now run the code to test the model –
python test.py --weights "Path to the best weight without the quotes" --conf 0.4

```
PS C:\Users\Rizwana\Desktop\exp\yolov7-main\yolov7-main> python test.py --weights C:\Users\Rizwana\Desktop\exp\yolov7-main\yolov7-main\best-dataset_2.pt --conf 0.4
```

Figure 11(a): Syntax to run the test run for the model

```
val: Scanning 'C:\Users\Rizwana\Downloads\Guns_Yolov7\test\labels.cache' images and labels... 593 found, 10 missing, 0 empty, 0
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100%|| 19/19 [12:28<00:00, 39.40s
                 all         603         698       0.715       0.133       0.125      0.0606
Speed: 1235.2/0.3/1235.5 ms inference/NMS/total per 640x640 image at batch-size 32
```

Figure 11(b): Output of the test syntax

We can interpret the precision, recall rate and mAP from this. All the graphs and images are saved under *run -> test -> exp*

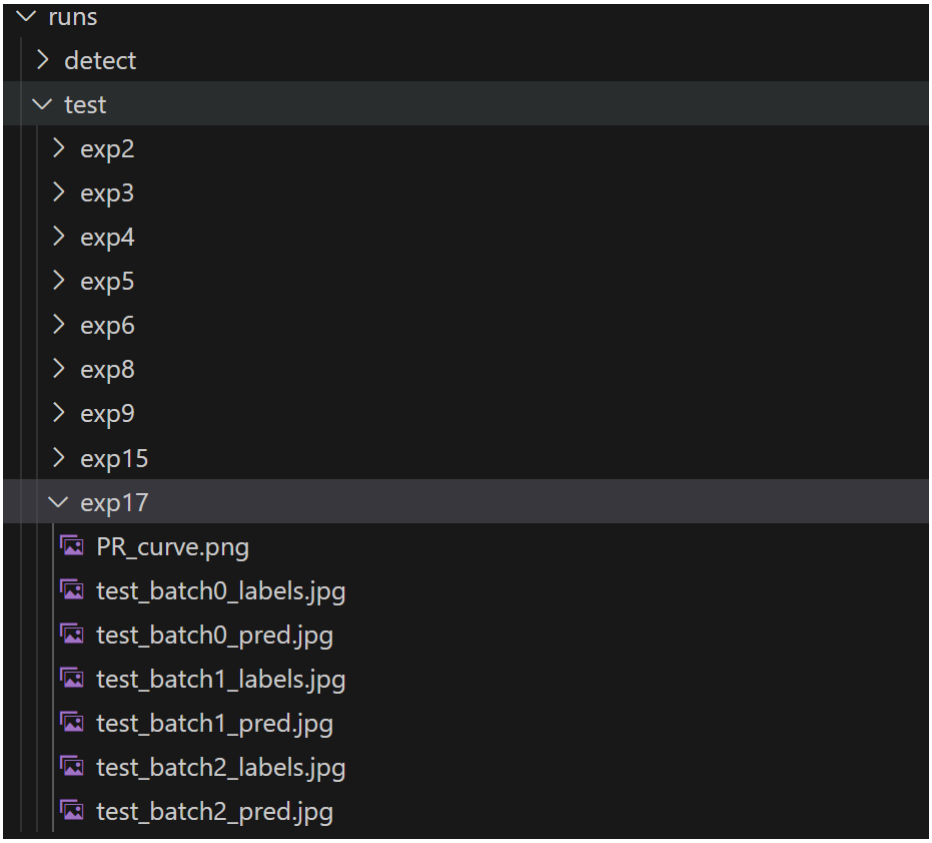


Figure 11(c): Output of the test syntax saved under exp folders

9. Another important note. When running the code for different datasets, the model has run so many times that it might have an issue of overfit. Hence I recommend now to run the model too many times. This can result in the images to be distorted and yield bad result metrics.

References