National College of Ireland

Configuration Manual

MSc Research Project
MSc. in Data Analytics

Sudharsan Ramesh
Student ID: 21234639

School of Computing
National College of Ireland

Supervisor: Arjun Chikkankod

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sudharsan Ramesh
Student ID:	21234639
Programme:	MSc. in Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Arjun Chikkankod
Submission Due Date:	18/09/2023
Project Title:	Configuration Manual
Word Count:	836
Page Count:	12

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:	Sudharsan Ramesh
Date:	18th September 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 keepa 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

Sudharsan Ramesh

21234639

1 Introduction

This is the configuration manual of the project "**Improvements in Aerial Object Detection: Comparing YOLOv7 with YOLOv5 for Fine Drone and Bird Detection in Volatile Environments**". This setup documentation contains all of the relevant data, including the equipment I utilized, software and hardware specifications, crucial code screenshots, and reproducibility requirements. The specifications are detailed in Section 2, which includes the Software Standard and Hardware Specifications.

2 Specifications

The following chapters go through the software and hardware requirements for this proposed study.

2.1 Software Configurations

As we can see in Table 1, the summary of the software configurations that have been employed during this investigation. Figure 2 demonstrates the hardware and operating system performance.

Table 1: Software

Software	Configuration
Operational System	Windows 10 Home Single Language
Online IDE	Google Colab notebooks
Coding Language	Python
Coding Language Version	Python 3.7
Additional Tools Used	LabelImg, Google Colab

2.2 Hardware Configurations

Table 2 illustrates the hardware configurations used in this investigation.

Table 2: Hardware

Hardware	Configuration
System	Intel(R) Core(TM) 4210U
Operation System	Windows 10 Home Single Language

RAM	6.00 GB
Storage	1 TB
Libraries	cv2, os, torch, tensorflow, yaml, utils
Graphic Card	Intel (R) HD Graphics Family

Item	Value
OS Name	Microsoft Windows 10 Pro
Version	10.0.19045 Build 19045
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	DESKTOP-BFK0U5B
System Manufacturer	LENOVO
System Model	20AMS1XB00
System Type	x64-based PC
System SKU	LENOVO_MT_20AM_BU_Think_FM_ThinkPad X240
Processor	Intel(R) Core(TM) i5-4300U CPU @ 1.90GHz, 2494 Mhz, 2 Core(s), 4 Logical Pr...
BIOS Version/Date	LENOVO GIET98WW (2.48), 08-08-2019
SMBIOS Version	2.7
Embedded Controller Version	1.18
BIOS Mode	UEFI
BaseBoard Manufacturer	LENOVO
BaseBoard Product	20AMS1XB00
BaseBoard Version	0B98401 WIN
Platform Role	Mobile
Secure Boot State	Off
PCR7 Configuration	Elevation Required to View
Windows Directory	C:\Windows
System Directory	C:\Windows\system32
Boot Device	\Device\HarddiskVolume1
Locale	United Kingdom

Figure 1: Device and Windows Specifications

3 Integrated Development Environment

To conduct the research and run the code, Google Colab's was utilized, and Figure 2 below shows how it appears once I start it.

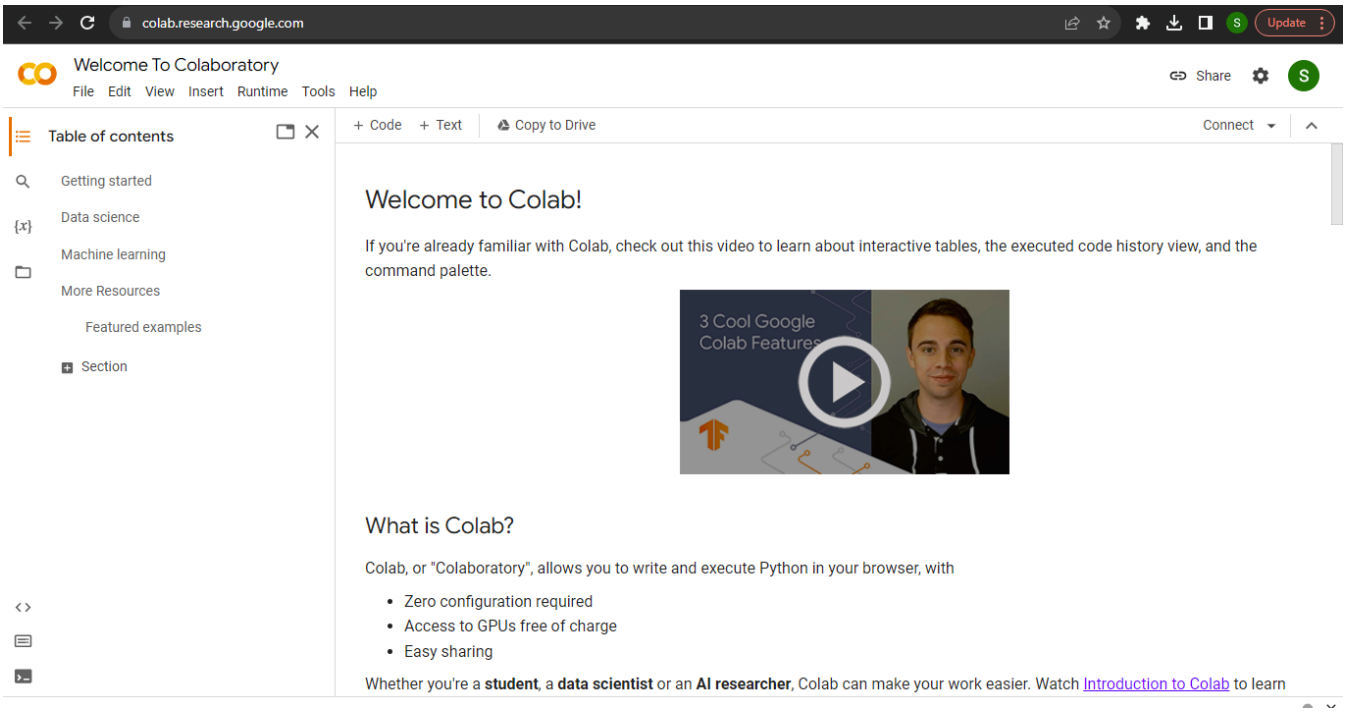

Figure 2: Google Colab launched

4 LabelImg

4.1 Download LabelImg

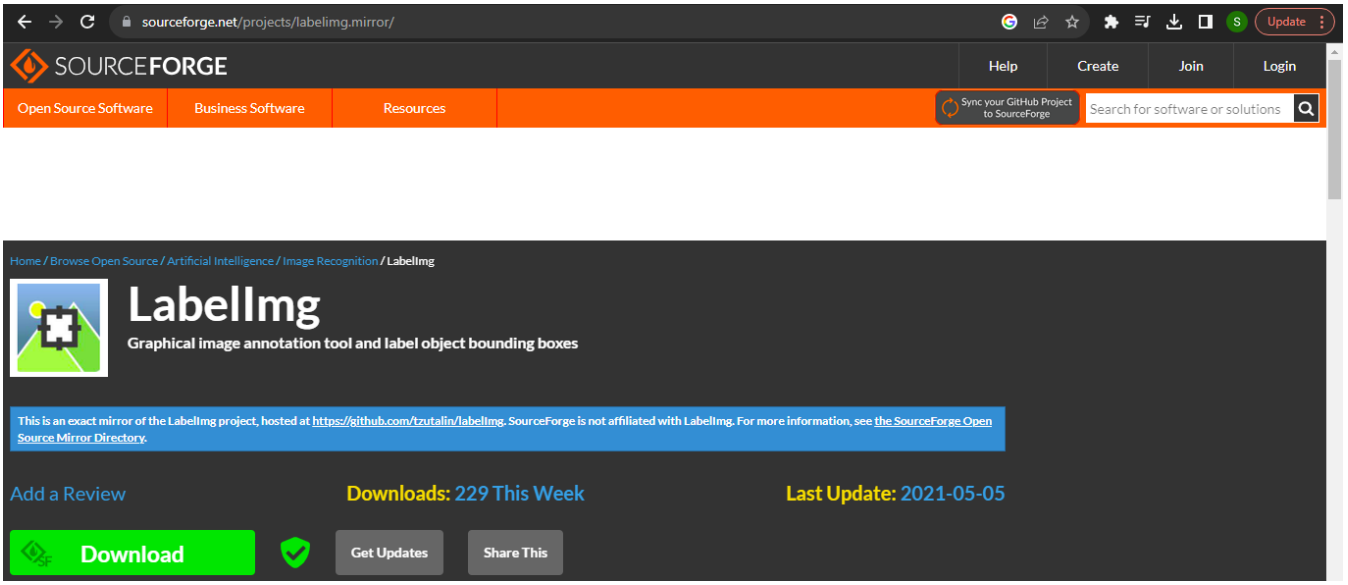

Figure 3: Downloading Software

4.2 Open LabelImg

Figure 4 : LabelImg Application

4.3 Select Directory to open and save images

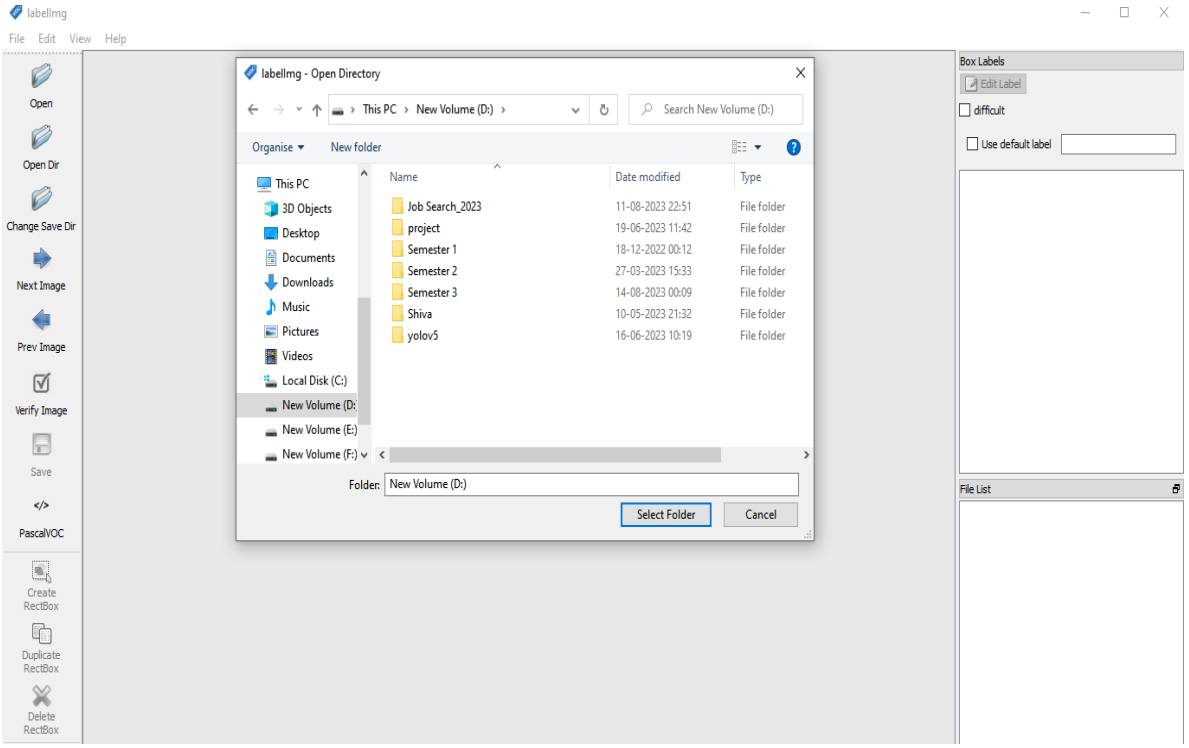

Figure 5: LabelImg Directory Selection

4.4 Draw and Save Annotations

When we select and save, the photograph is stored to the place we have specified previously. Afterwards we just click on the next photograph and repeat the process for each picture.

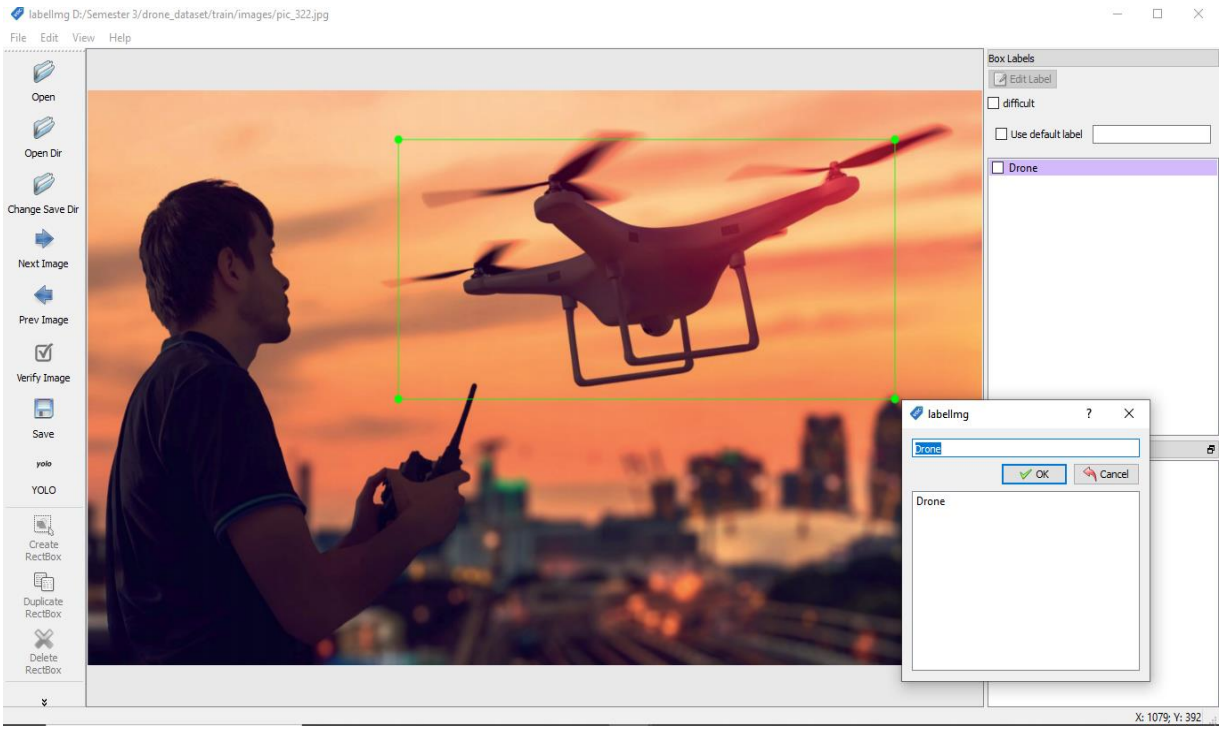

Figure 6: Image labeling

5 YOLOv7 Training Phase

5.1 Installing the YOLOv7 Environment

To get started with YOLOv7, we'll download the repository and apply the requirements. This will prepare our development platform so that object identification training and interpretation instructions can be executed.

```
# Downloading YOLOv7 repository and installing requirements
!git clone https://github.com/WongKinYiu/yolov7
%cd yolov7
!pip install -qr requirements.txt
```

```
Cloning into 'yolov7'...
remote: Enumerating objects: 1191, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 1191 (delta 2), reused 6 (delta 2), pack-reused 1185
Receiving objects: 100% (1191/1191), 74.23 MiB | 17.35 MiB/s, done.
Resolving deltas: 100% (511/511), done.
/content/yolov7/yolov7
```

Figure 7: Clone form Repository

After that, we can look at our Google Colab setup and begin installing requirements. We will be able to reduce training time by using the GPU, Torch is preconfigured on Colab, and it is a beneficial characteristic. As we do not attempt to run this code locally, there are no additional setups to be followed for YOLOv7.

5.2 Download Correctly Formatted Custom Dataset

Loading into this notebook our data

```
!unzip /content/Drone.zip -d /content/yolov7/

Archive:  /content/Drone.zip
   creating: /content/yolov7/train/images/
  inflating: /content/yolov7/train/images/0080.jpg
  inflating: /content/yolov7/train/images/0081.jpg
  inflating: /content/yolov7/train/images/0082.jpg
  inflating: /content/yolov7/train/images/0083.jpg
  inflating: /content/yolov7/train/images/0084.jpg
  inflating: /content/yolov7/train/images/0085.jpg
  inflating: /content/yolov7/train/images/0086.jpg
  inflating: /content/yolov7/train/images/0087.jpg
  inflating: /content/yolov7/train/images/0088.jpg
  inflating: /content/yolov7/train/images/0089.jpg
  inflating: /content/yolov7/train/images/0090.jpg
  inflating: /content/yolov7/train/images/0091.jpg
  inflating: /content/yolov7/train/images/0092.jpg
  inflating: /content/yolov7/train/images/0093.jpg
  inflating: /content/yolov7/train/images/0094.jpg
  inflating: /content/yolov7/train/images/0095.jpg
  inflating: /content/yolov7/train/images/0096.jpg
  inflating: /content/yolov7/train/images/0097.jpg
  inflating: /content/yolov7/train/images/0101.jpg
  inflating: /content/yolov7/train/images/1.png
  inflating: /content/yolov7/train/images/100.png
  inflating: /content/yolov7/train/images/101.png
```

Figure 8: Import Dataset

5.3 Model Architecture

We can use the pre-created yaml file because it specifies our model's characteristics, such as the number of classes, anchors, and layers.

data.yaml

D: > Semester 3 > drone_dataset > Drone > ! data.yaml

```
1  train: /content/yolov7/train/images
2  val: /content/yolov7/valid/images
3  nc: 2
4  names: [drone, bird]
5
```

Figure 9: Yaml data

```
# define number of classes based on YAML
import yaml
with open("/content/data.yaml", 'r') as stream:
    num_classes = str(yaml.safe_load(stream)['nc'])
```

```
[ ] num_classes

'2'
```

Figure 10: Importing Yaml file

5.4 Train the Model

Based on the information we have Yolov7 is now ready to train with our yaml files on hand. To start training, execute the training instruction with the following parameters:

Table 3: Train model

Image	416 x 416
Size of the batch	16
Epoch	300
Data location	/content/data.yaml
Weights	yolov7.pt
Name	Yolov7_result

And run the training command:

```
#customize iPython writefile so we can write variables
from IPython.core.magic import register_line_cell_magic

@register_line_cell_magic
def writetemplate(line, cell):
    with open(line, 'w') as f:
        f.write(cell.format(**globals()))
```

```
# train yolov7 on custom data for 300 epochs
# time its performance
%%time
%cd /content/yolov7/
!python train.py --img 416 --batch 16 --epochs 300 --data '/content/data.yaml' --weights 'yolov7.pt' --name yolov7_results  --cache
```

```
/content/yolov7
2023-08-12 23:15:48.836487: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use availabl
To enable the following instructions: AVX2 AVX512F FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-08-12 23:15:49.800573: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
YOLOR 🚀 v0.1-126-g84932d7 torch 2.0.1+cu118 CUDA:0 (Tesla T4, 15101.8125MB)

Namespace(weights='yolov7.pt', cfg='', data='/content/data.yaml', hyp='data/hyp.scratch.p5.yaml', epochs=300, batch_size=16, img_siz
tensorboard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
hyperparameters: lr0=0.01, lrf=0.1, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1,
wandb: Install Weights & Biases for YOLOR logging with 'pip install wandb' (recommended)
Overriding model.yaml nc=80 with nc=2

                 from  n    params  module                                  arguments
  0                -1  1       928  models.common.Conv                      [3, 32, 3, 1]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     36992  models.common.Conv                      [64, 64, 3, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  1      8320  models.common.Conv                      [128, 64, 1, 1]
  5                -2  1      8320  models.common.Conv                      [128, 64, 1, 1]
  6                -1  1     36992  models.common.Conv                      [64, 64, 3, 1]
  7                -1  1     36992  models.common.Conv                      [64, 64, 3, 1]
  8                -1  1     36992  models.common.Conv                      [64, 64, 3, 1]
  9                -1  1     36992  models.common.Conv                      [64, 64, 3, 1]
 10  [-1, -3, -5, -6]  1         0  models.common.Concat                    [1]
```

```
# train yolov7 on custom data for 300 epochs
# time its performance
%%time
%cd /content/yolov7/
!python train.py --img 416 --batch 16 --epochs 300 --data '/content/data.yaml' --weights 'yolov7.pt' --name yolov7_results  --cache
```

```
     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     0/299     1.31G   0.08471  0.008546   0.01797    0.1112        40       416: 100% 25/25 [00:30<00:00,  1.23s/it]
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 4/4 [00:06<00:00,  1.73s/it]
                 all         125         139      0.0585       0.113      0.0249     0.00364

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     1/299     5.66G   0.07564  0.008919   0.01452   0.09907        38       416: 100% 25/25 [00:11<00:00,  2.24it/s]
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 4/4 [00:01<00:00,  3.05it/s]
                 all         125         139       0.039       0.193      0.0304     0.00626

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     2/299      6.6G    0.0691  0.009287   0.01157   0.08996        19       416: 100% 25/25 [00:11<00:00,  2.09it/s]
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 4/4 [00:01<00:00,  3.64it/s]
                 all         125         139       0.129       0.343       0.12      0.0338

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     3/299      6.6G   0.06107   0.01001  0.008984   0.08007        40       416: 100% 25/25 [00:12<00:00,  2.05it/s]
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 4/4 [00:01<00:00,  3.96it/s]
                 all         125         139       0.233       0.297       0.256      0.0846

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
     4/299      6.6G   0.05867  0.008997  0.007804   0.07548        27       416: 100% 25/25 [00:12<00:00,  2.08it/s]
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 4/4 [00:00<00:00,  4.06it/s]
```

Figure 11: Training the YOLOv7

5.5 Evaluate Custom YOLOv7 Detector Performance

```
from PIL import Image
from IPython.display import display

# Open the image file using Image.open()
image = Image.open('/content/yolov7/runs/train/yolov7_results/results.png')
# Display the image
display(image)
```

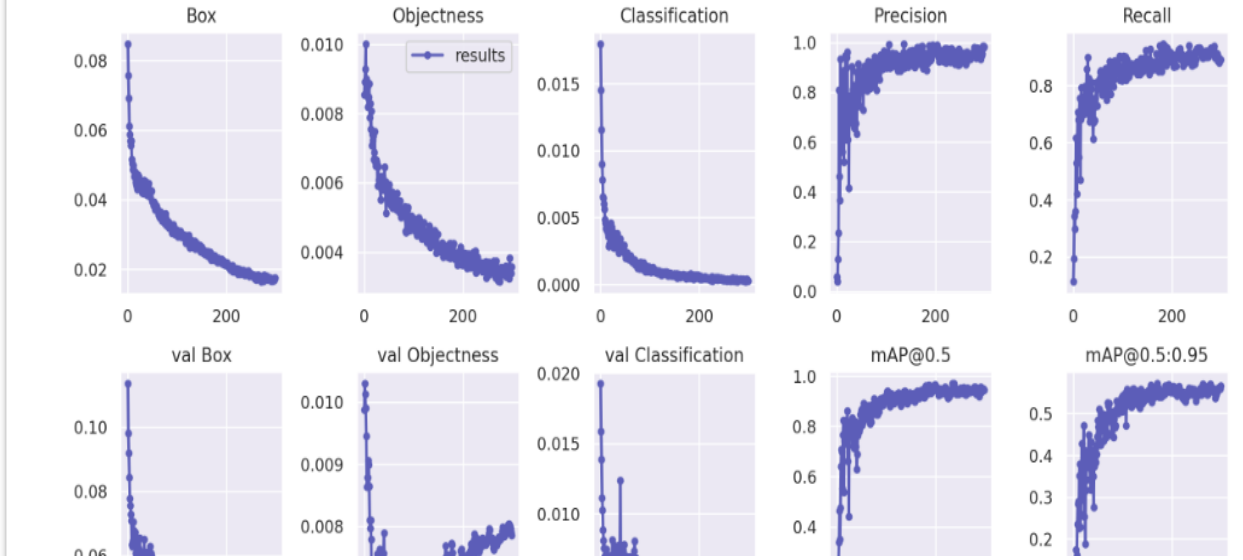

Figure 12: Result Visualization

5.6 Run Inference with Trained Weights

```
# trained weights are saved by default in our weights folder
%ls runs/
```

```
train/
```

```
[ ] %ls runs/train/yolov7_results/weights
```

```
best_278.pt   epoch_049.pt  epoch_149.pt  epoch_249.pt  epoch_297.pt  last.pt
best.pt       epoch_074.pt  epoch_174.pt  epoch_274.pt  epoch_298.pt
epoch_000.pt  epoch_099.pt  epoch_199.pt  epoch_295.pt  epoch_299.pt
epoch_024.pt  epoch_124.pt  epoch_224.pt  epoch_296.pt  init.pt
```

```
[ ] # when we ran this, we saw .007 second inference time. That is 140 FPS on a TESLA P100!
    # use the best weights!
    %cd /content/yolov7/
    !python detect.py --weights /content/yolov7/runs/train/yolov7_results/weights/best.pt --img 416 --conf 0.1
```

```
/content/yolov7
Namespace(weights=['/content/yolov7/runs/train/yolov7_results/weights/best.pt'], source='/content/yolov7/v
YOLOR 🚀 v0.1-126-g84932d7 torch 2.0.1+cu118 CUDA:0 (Tesla T4, 15101.8125MB)

Fusing layers...
RepConv.fuse_repvgg_block
RepConv.fuse_repvgg_block
RepConv.fuse_repvgg_block
/usr/local/lib/python3.10/dist-packages/torch/functional.py:504: UserWarning: torch.meshgrid: in an upcomi
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
Model Summary: 306 layers, 36485311 parameters, 6194944 gradients, 103.2 GFLOPS
 Convert model to Traced-model...
 traced_script_module saved!
 model is traced!
```

Figure 13: Detecting the class after YOLOv7

5.7 Export Trained Weights for Future Inference

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

5.8 Detecting Drone and Bird Images

```
import os
import random
import matplotlib.pyplot as plt
from PIL import Image

# Path to the directory containing the images
image_dir = '/content/yolov7/runs/detect/exp'

# Get a list of image file names in the directory
image_files = [f for f in os.listdir(image_dir) if f.lower().endswith(('.jpg', '.jpeg', '.png', '.gif', '.bmp'))]

# Randomly select 20 images
random_images = random.sample(image_files, 20)

# Create a 4x5 grid for the collage
num_rows = 4
num_cols = 5

# Set up the figure
fig, axes = plt.subplots(num_rows, num_cols, figsize=(15, 12))
plt.subplots_adjust(wspace=0.2, hspace=0.4)

# Loop through the axes and display images
for i, ax in enumerate(axes.flat):
    img_path = os.path.join(image_dir, random_images[i])
    img = Image.open(img_path)
    ax.imshow(img)
    ax.axis('off')
    ax.set_title(random_images[i])

plt.show()
```

Figure 14: Show the detected Image

5.9 Final Output