Configuration Manual

MSc Research Project
Data Analytics

Aditya Raju Pal
Student ID: x20195281

School of Computing
National College of Ireland

Supervisor: Prof. Hicham Rifai

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Aditya Raju Pal
Student ID:	x20195281
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Prof. Hicham Rifai
Submission Due Date:	15/08/2022
Project Title:	Configuration Manual
Word Count:	1254
Page Count:	9

Signature:	
Date:	18th September 2022

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.	☐

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

Configuration Manual

Aditya Raju Pal
x20195281

1 Introduction

This configuration manual gives detailed information about the hardware and software requirements along with all other details about the programming codes written for model implementation and evaluation of the research project: “Can Deep Neural Network with MobileNet V2 + LSTM based hybrid model perform classification of pest-infested citrus leaf images and be used in light weight mobile applications?”.

2 System Configuration

2.1 Hardware Configuration

Table 1 represents the system’s hardware specifications on which the research was carried out for this project.

Table 1: Hardware Specifications

RAM	8 GB
Processor	Intel(R) Core(TM) i5-8300H
Speed	2.30 GHz
Operating System	Windows 10, 64 Bit
Storage	1 TB HDD
GPU	NVIDIA GeForce GTX1650

2.2 Software Configuration

Jupiter Notebook from Anaconda Distribution:

The open-source desktop GUI called Anaconda Navigator is a part of the Anaconda distribution. The distribution provides support to Jupiter Notebooks and it was extremely beneficial when putting machine learning models into practice and running them on the Kaggle research data. This research was carried out using Jupiter notebook version 6.4.12, which included all processes, including data augmentation, feature extraction and developing hybrid machine learning models.

Google Colab

Google colab with GPU has been used for the training and evaluation of some of the parts of this study.

3 Project Development

The Python programming language was used extensively throughout this research project. The Python language was used for the majority of the processes, including data augmentation, feature extraction, model design and assessment. Libraries such as Matplotlib, numpy, pandas, seaborn, and other tools, the main libraries utilized were Keras, TensorFlow, PyTorch, Scikit-Learn and LabelImg. The project has been divided into two main parts:

1. Usage of MobileNet V2 and MobileNet V2 + LSTM Hybrid model to classify a diseased plant leaf.

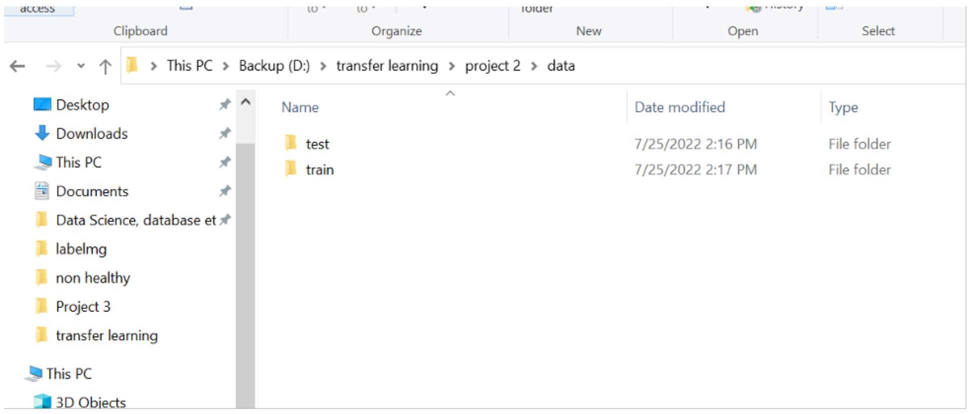
2. Usage of YOLO V5 to detect the diseased area of the plant leaf.

4 MobileNet Model

Usage of MobileNet V2 and MobileNet V2 + LSTM Hybrid model to classify a diseased plant leaf.

4.1 Data Preparation

Jupiter notebook was given access to the datasets that was obtained from Kaggle.com. The datasets contained 2 classes. The train dataset contained 9169 images while the test dataset contained 1206 images Figure 1 is how the directory structure for the train and the test datasets looked like.



Name	Date modified	Type
test	7/25/2022 2:16 PM	File folder
train	7/25/2022 2:17 PM	File folder

Figure 1: Dataset Folders

4.2 Data Augmentation

Augmentation was performed using Tensorflow package in python. The images were augmented on the go for both the training and the testing after some changes in the datasets. The data generator yielded images in batches and performed the aforementioned transformation. Additionally, we kept the validation split = 0.2 to ensure k fold cross validation. The target size of the image was 120 x 120.

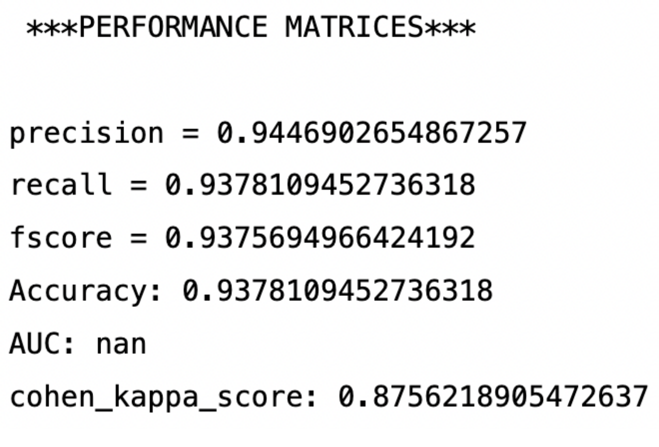
4.3 Performance Evaluation

Since the task is a classification task, we used the following metrics to evaluate the performance of the model.

- Precision
- Accuracy
- F1 Score
- Recall
- Cohen Kappa Score
- ROC curve
- Confusion matrix

4.4 MobileNet V2

Here, we used the base model of MobileNet with the pre-trained weights of ImageNet dataset to extract the features (low level) out of the images. The input shape from the generator was (120, 120, 3). Then we added a custom top for the classification. The custom top had 2 dense layers. The first layer had 1056 neurons. The last layer had 2 neurons (as there were 2 classes). In the first layer the activation function used was 'relu' to have some non linearity's. Since it was a binary classification problem, 'sigmoid' activation function was there at the final layer. The 'crossentropy' was used as loss function as this is a classification problem. 'adamax' optimizer was used since it is among one of the best optimizer for image classification. Additionally, we used two callbacks. One was to save the best weights. The other was monitoring the validation loss. The performance results came have been shown in Figure 2.



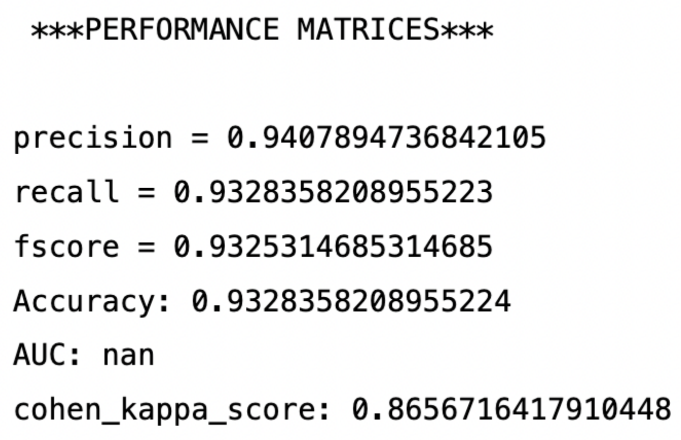
```
 ***PERFORMANCE MATRICES***


precision = 0.9446902654867257
recall = 0.9378109452736318
fscore = 0.9375694966424192
Accuracy: 0.9378109452736318
AUC: nan
cohen_kappa_score: 0.8756218905472637
```

Figure 2: Performance Results for MobileNet V2

4.5 MobileNet V2 + LSTM Hybrid Model

Here, we used the base model of MobileNet with the pre-trained weights of ImageNet dataset to extract the features (low level) out of the images. The input shape from the generator was (120, 120, 3). Then we added a custom top for the classification. The custom top had 2 LSTM layers and a dense layer. The first LSTM layer had 30 neurons. The second LSTM layer had 5 neurons. The last dense layer had 2 neurons (as there are 2 classes - healthy leaf or unhealthy leaf). The activation function for the first two layers was 'relu' as it introduces some non linearity. Since it is a binary classification problem, the final layer's activation function was 'sigmoid'. 'cross-entropy' was used as loss function as this is a classification problem. 'RMSProp' optimizer was used since it is among one of the best optimizer for image classification. Additionally, we used two callbacks. One was to save the best weights. The other was for monitoring the validation loss. The results of performance are represented in Figure 3.



```
 ***PERFORMANCE MATRICES***


precision = 0.9407894736842105
recall = 0.9328358208955223
fscore = 0.9325314685314685
Accuracy: 0.9328358208955224
AUC: nan
cohen_kappa_score: 0.8656716417910448
```

Figure 3: Performance Results for MobileNet V2 + LSTM

5 YOLO V5 Model

Usage of YOLO V5 to detect the diseased area of the plant leaf.

5.1 Data Preparation

The data was collected from Kaggle.com and then annotated in YOLO format using labelImg. The application was cloned using the following command from the code

```
git clone https://github.com/heartexlabs/labelImg
```
⊤

Then the application was run using the following commands from the command line

```
cd labelImg
python labelImg.py
```

The interface of the application looks like in figure part (a) and with clicking on the Open Dir we can select the folder where the training images reside as shown in figure part (b) of Figure 5.

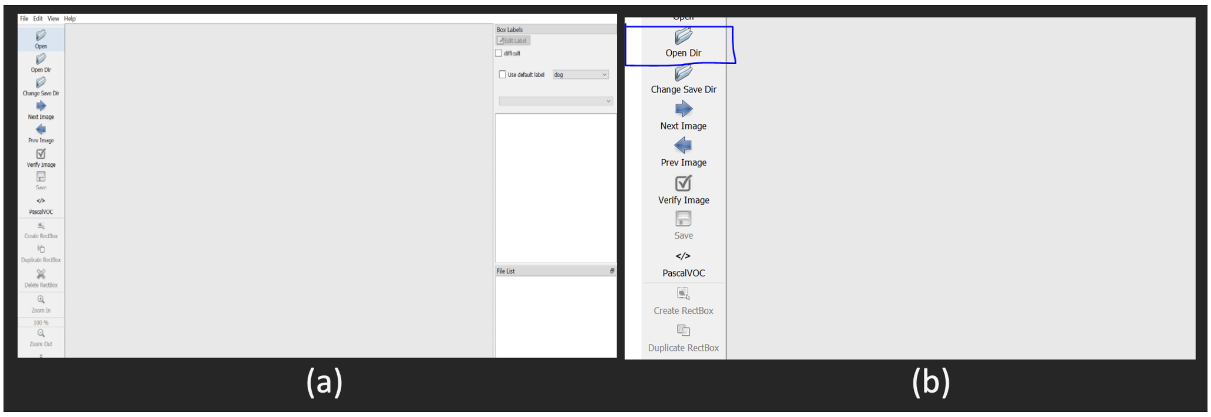


Figure 4: View of LabelImg tool

The part (c) in Figure 5 shows how to click on the change 'Save Dir' to select the directory where the annotations are to be saved. Prior to saving the annotations, change the annotations format to "YOLO". Part (d) of the Figure 5 shows how to hit ‘w’ and select the region and name it as disease and then hit save. In the last part, (c) it shows how to click on ‘Next Image’ and reiterate until all the images are labelled. A total of 53 images were annotated for the training process.

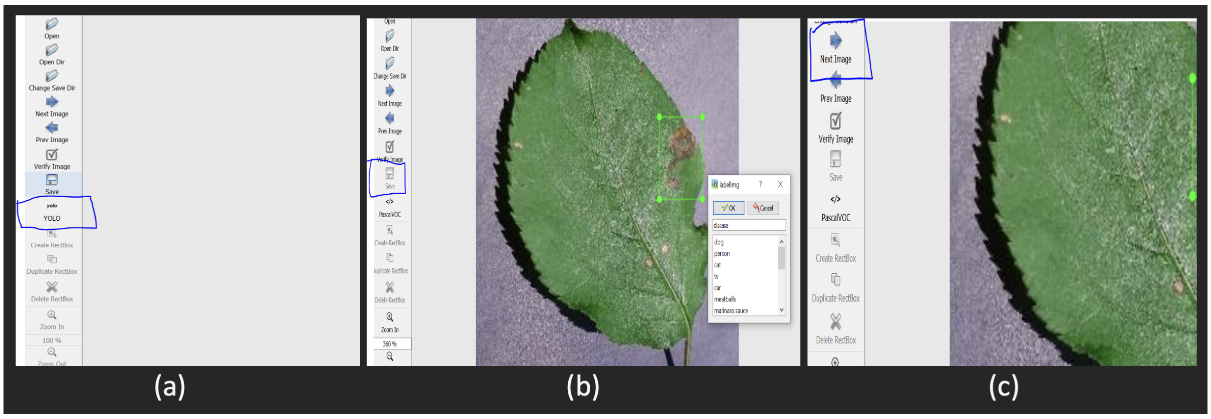


Figure 5: Steps for Annotation

5.2 Model Preparation

Google Colab is used to train and test the model.

Installing dependencies

The dependencies were cloned and installed form ultralytics (PyTorch) using the below code.

```
#clone YOLOv5 and
!git clone https://github.com/ultralytics/yolov5  # clone repo
%cd yolov5
%pip install -qr requirements.txt # install dependencies
%pip install -q roboflow

import torch
import os
from IPython.display import Image, clear_output  # to display images

print(f"Setup complete. Using torch
{torch.__version__}
({torch.cuda.get_device_properties(0).name if torch.cuda.is_available()
else 'CPU'})")
```

5.3 Assembling the dataset

The train and the test datasets were placed in the following directory
yolov5/data/images/

And for the labels, a separate directory with the name of labels was created in yolov5/data and the labels were uploaded in the folder.
yolov5/data/labels/

Additionally, a separate file dataset.yaml was created. In the dataset.yaml file, we are first setting the path to the train and then the test directories. And then defining the number of classes. nc: 16 was used as there were already 15 built-in classes in the LabelImg output file.

```
# train and val data as 1) directory: path/images/,
2) file: path/images.txt, or 3) list: [path1/images/, path2/images/]
path: /content/yolov5/data/
train: images
val: images
test: images
# number of classes
nc: 16
# class names
names: ['dog', 'person','cat','tv','car','meatballs','marinara sauce',
'tomato soup','chicken noodle soup','french onion soup','chicken breast',
'ribs','pulled pork','hamburger','cavity','disease']
```

The dataset.yaml file was kept in the directory
yolov5/

5.4 Train the model

Run the following command to start training the model on 150 epochs with the YOLO V5 original weights.

```
!python train.py --img 416 --batch 16 --epochs 150 --data dataset.yaml
--weights yolov5s.pt {cache
```

5.5 Predictions on the test data

To start predictions on the test data, following command was run

```
!python detect.py --weights runs/train/exp/weights/best.pt --img 416
--conf 0.1 --source '/content/yolov5/data/images'
```

Then to check the results of the prediction with bounding boxes, the following command was run. This will basically plot each of the predicted images with bounding box.

```
# plotting prediction results

import glob
from IPython.display import Image, display

for imageName in glob.glob('/content/yolov5/runs/detect/exp/*.JPG'):
#assuming JPG
    display(Image(filename=imageName))
    print("\n")
```

The output is shown in Figure 6

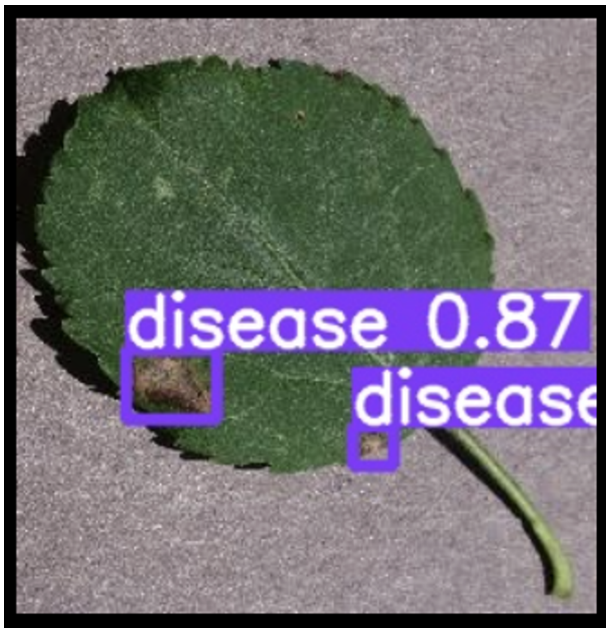


Figure 6: YOLO Prediction

5.6 Performance Evaluation

To evaluate the performance of the model tensorboard was used. The following commands are used to evaluate the performance.

```
# Start tensorboard
# Launch after you have started training
# logs save in the folder "runs"
%load_ext tensorboard
%tensorboard --logdir runs
```

The results of the model is represented in Figure 7

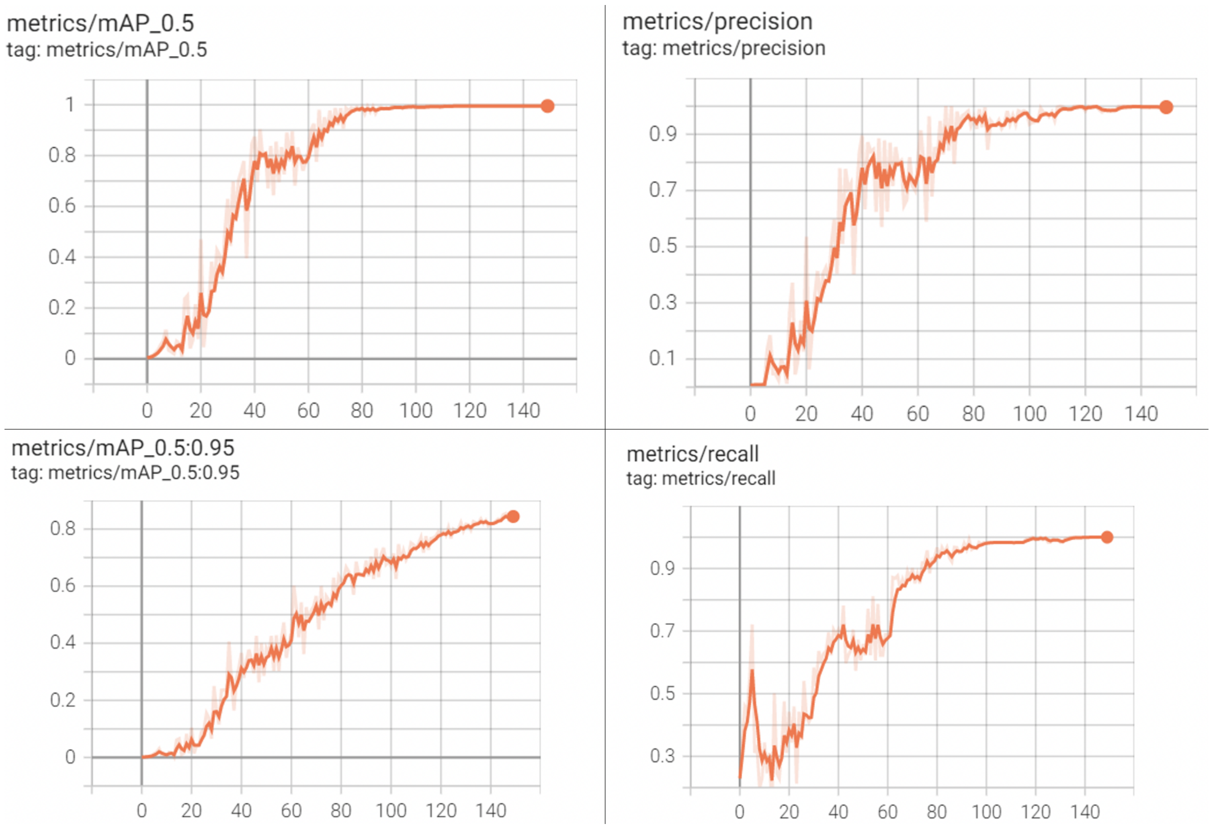


Figure 7: YOLO Prediction