National College of Ireland

Configuration Manual

MSc Research Project
MSc in Data Analytics

Ziyi Yan
Student ID: x22198512

School of Computing
National College of Ireland

Supervisor: Prof. Jorge Basilio

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ziyi Yan
Student ID:	X22198512
Programme:	MSc in Data Analytics **Year:** 2024
Module:	Research Project
Lecturer:	Prof. Jorge Basilio
Submission Due Date:	12/08/2024
Project Title:	How Well Can MobileNetV3 Perform in Detecting Diseases in Tomato Plants
Word Count:	656 **Page Count:** 8

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: Ziyi Yan

Date: 11/08/2024

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

Ziyi Yan
x22198512

1 Introduction

This configuration manual lists the details about how to set up the running environment for the project's code and the development process of the project.

2 Software

- Operating System: Windows 11.
- Python Version: Python 3.11.
- IDE: VS Code, Jupyter Notebook.
- Libraries: PyTorch, Torchvision, Pandas, PIL(Python Imaging Library), Matplotlib, os, Seaborn, scikit-learn.

3 Hardware

- Laptop Model: Mechanical Revolution Winged Dragon 15 Pro (机械革命翼龙 15 Pro).
- Processor: AMD Ryzen 7 8845H.
- GPU: NVIDIA RTX 4060.
- Memory: 32GB RAM.
- Storage: 512GB SSD.

4 Setp-by-step Setup

4.1 Install Python

Download and install python from the official website: https://www.python.org/downloads/

4.2 Install Jupyter Notebook

We can install it by running the following command in the terminal:
pip install notebook

When the installation is complete, we can start the Jupyter Notebook by running the following command:
jupyter notebook
this command will open Jupyter Notebook in our web browser, it looks like this:
http://localhost:8888/tree?

4.3 Install Required Libraries

Install the libraries by running the following command in the terminal:
pip install torch torchvision scikit-learn pandas matplotlib seaborn pillow

If some libraries are not found when the code is running, look carefully for the error message, and install the missing library separately with the pip command.

4.4 Download the dataset

Download the dataset from Kaggle: https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset

The dataset's directory looks like this:

```
plantvillage/
│
├── color/
│   ├── Apple___Apple_scab/
│   ├── Apple___Black_rot/
│   ├── ...
│   └── Tomato___Tomato_Yellow_Leaf_Curl_Virus/
│
├── grayscale/
│   ├── Apple___Apple_scab/
│   ├── Apple___Black_rot/
│   ├── ...
│   └── Tomato___Tomato_Yellow_Leaf_Curl_Virus/
│
└── segmented/
    ├── Apple___Apple_scab/
    ├── Apple___Black_rot/
    ├── ...
    └── Tomato___Tomato_Yellow_Leaf_Curl_Virus/
```

Figure 1 Dataset's directory

In this project, the directories of code and dataset are as follows:
D:\workspace\NCI_Study\Semester3\ResearchProject\code\PlantVillage_MobileNetV3.ipynb
D:\workspace\NCI_Study\Semester3\ResearchProject\code\Prediction.ipynb
D:\workspace\NCI_Study\Semester3\ResearchProject\dataset\plantvillage

4.5 How to run the code

You can run the code in VS Code or Jupyter Notebook. There is one thing to note that we need to install the Hupyter notebook support if we run the code in VS Code.

Figure 2 VS Code extension for Jupyter notebook support

5 Code instructions

5.1 Import libraries and modules

Import necessary libraries for each jupyter notebook file.

```
import os
import torch
from torchvision import transforms, models
from torch.utils.data import DataLoader, Dataset, random_split
import torch.nn as nn
import torch.optim as optim

from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, confusion_matrix
import matplotlib.pyplot as plt
from PIL import Image
import seaborn as sns
import random
import pandas as pd
```

Figure 3 Libraries for PlantVillage_MobileNetV3.ipynb

```
import torch
from torchvision import models, transforms
from PIL import Image
import torch.nn.functional as F

```

Figure 3 Libraries for Prediction.ipynb

5.2 Define custom class to load the images

There are not only tomato but also other types of plant's leaf images in the dataset, we create a custom class ImageDataset load the images we need in the project.

```python
# dataset path
data_dir = "../dataset/plantvillage/color"

# Custom dataset classes
class ImageDataset(Dataset):
    def __init__(self, root_dir, transform=None, plant_type=None):
        self.root_dir = root_dir
        self.transform = transform
        if plant_type:
            self.classes = [d for d in os.listdir(root_dir) if d.startswith(plant_type)]
        else:
            self.classes = os.listdir(root_dir)

        self.image_paths = []
        self.labels = []
        print(self.classes)
        for label, class_dir in enumerate(self.classes):
            class_dir_path = os.path.join(root_dir, class_dir)
            for img_name in os.listdir(class_dir_path):
                self.image_paths.append(os.path.join(class_dir_path, img_name))
                self.labels.append(label)

    def __len__(self):
        return len(self.image_paths)

    def __getitem__(self, idx):
        img_path = self.image_paths[idx]
        image = Image.open(img_path).convert('RGB')
        label = self.labels[idx]

        if self.transform:
            image = self.transform(image)

        return image, label

# Load the dataset
plant = "Tomato"
tomato_datasets = ImageDataset(root_dir=data_dir, transform=data_transforms['train'], plant_type=plant)
tomato_datasets_no_aug = ImageDataset(root_dir=data_dir, transform=data_transforms_no_aug['train'], plant_type=plant)
```

Figure 4 Custom Class to load the images

5.3 Model initialization

We are planning to train 5 models in the study, so, it is necessary to define a function to handle this.

```python
# Define model functions
def initialize_model(model_name, num_classes, feature_extract, use_pretrained=True):
    if model_name == "resnet":
        model = models.resnet50(pretrained=use_pretrained)
        set_parameter_requires_grad(model, feature_extract)
        num_ftrs = model.fc.in_features
        model.fc = nn.Linear(num_ftrs, num_classes)
    elif model_name == "efficientnet":
        model = models.efficientnet_b0(pretrained=use_pretrained)
        set_parameter_requires_grad(model, feature_extract)
        num_ftrs = model.classifier[1].in_features
        model.classifier[1] = nn.Linear(num_ftrs, num_classes)
    elif model_name == "pretrained_mobilenetv3":
        model = models.mobilenet_v3_large(pretrained=use_pretrained)
        set_parameter_requires_grad(model, feature_extract)
        num_ftrs = model.classifier[3].in_features
        model.classifier[3] = nn.Linear(num_ftrs, num_classes)
    elif model_name == "scratch_mobilenetv3":
        model = models.mobilenet_v3_large(pretrained=False)
        set_parameter_requires_grad(model, feature_extract)
        num_ftrs = model.classifier[3].in_features
        model.classifier[3] = nn.Linear(num_ftrs, num_classes)
    elif model_name == "transfer_mobilenetv3":
        model = models.mobilenet_v3_large(pretrained=use_pretrained)
        set_parameter_requires_grad(model, feature_extract)
        num_ftrs = model.classifier[3].in_features
        model.classifier[3] = nn.Linear(num_ftrs, num_classes)
    else:
        print("Invalid model name, exiting...")
        exit()

    return model

```

Figure 5 Model initialize function

5.4 Model training

Model training is the most important part of the study, we will train the model with two optimizers separately(SGD and Adam), and also training the model with and without data enhancement processing respectively, the total number of models need to train are up to 20, this is a very time-consuming process.

```python
# model initialization and training
models_to_train = ["pretrained_mobilenetv3", "scratch_mobilenetv3", "transfer_mobilenetv3", "efficientnet", "resnet"]
optimizers = ["Adam", "SGD"]
feature_extract = False

trained_models = {}

def reset_model_and_optimizer(model_name, optimizer_name, num_classes, feature_extract):
    model = initialize_model(model_name, num_classes, feature_extract, use_pretrained=True)
    model = model.to(device)

    if optimizer_name == "Adam":
        optimizer = optim.Adam(model.parameters(), lr=0.001)
    elif optimizer_name == "SGD":
        optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)

    return model, optimizer

for model_name in models_to_train:
    for optimizer_name in optimizers:
        # training with the augmented data
        print(f"Training {model_name} model with {optimizer_name} optimizer with data augmentation...")
        model, optimizer = reset_model_and_optimizer(model_name, optimizer_name, num_classes, feature_extract)
        criterion = nn.CrossEntropyLoss()

        model, train_loss, val_loss, train_acc, val_acc = train_model(model, dataloaders, criterion, optimizer, num_epochs=25)
        trained_models[f"{model_name}_{optimizer_name}_aug"] = model

        plot_training_history(train_loss, val_loss, train_acc, val_acc, f"{model_name}_{optimizer_name}_aug")
        save_model(model, f"{model_name}_{optimizer_name}_aug_model.pth")

        # # training with the data without augmentation
        print(f"Training {model_name} model with {optimizer_name} optimizer without data augmentation...")
        model, optimizer = reset_model_and_optimizer(model_name, optimizer_name, num_classes, feature_extract)
        model, train_loss, val_loss, train_acc, val_acc = train_model(model, dataloaders_no_aug, criterion, optimizer, num_epochs=25)
        trained_models[f"{model_name}_{optimizer_name}_no_aug"] = model

        plot_training_history(train_loss, val_loss, train_acc, val_acc, f"{model_name}_{optimizer_name}_no_aug")
        save_model(model, f"{model_name}_{optimizer_name}_no_aug_model.pth")

```

Figure 6 Model training

5.5 Model evaluation

Use multiple metrics and confusion matrix to evaluate the model's performance.

```python
# model evaluation
def evaluate_model(model, dataloader, model_name):
    model.eval()
    all_preds = []
    all_labels = []

    with torch.no_grad():
        for inputs, labels in dataloader:
            inputs = inputs.to(device)
            labels = labels.to(device)

            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)

            all_preds.extend(preds.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())

    accuracy = accuracy_score(all_labels, all_preds)
    f1 = f1_score(all_labels, all_preds, average='weighted')
    precision = precision_score(all_labels, all_preds, average='weighted')
    recall = recall_score(all_labels, all_preds, average='weighted')

    # Generate confusion matrix
    cm = confusion_matrix(all_labels, all_preds)
    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title(f'Confusion Matrix for {model_name}')
    plt.show()

    return accuracy, f1, precision, recall

```

Figure 6 Model evaluation

5.6 Some of the outcomes

	Accuracy	F1 Score	Precision	Recall
pretrained_mobilenetv3_Adam_aug	0.978524	0.978470	0.978605	0.978524
pretrained_mobilenetv3_Adam_no_aug	0.761564	0.758518	0.853378	0.761564
pretrained_mobilenetv3_SGD_aug	0.983756	0.983700	0.983951	0.983756
pretrained_mobilenetv3_SGD_no_aug	0.835903	0.837414	0.876363	0.835903
scratch_mobilenetv3_Adam_aug	0.959802	0.959656	0.960117	0.959802
scratch_mobilenetv3_Adam_no_aug	0.806993	0.815255	0.851869	0.806993
scratch_mobilenetv3_SGD_aug	0.938326	0.938598	0.940750	0.938326
scratch_mobilenetv3_SGD_no_aug	0.772852	0.771470	0.793417	0.772852
transfer_mobilenetv3_Adam_aug	0.976872	0.976839	0.977275	0.976872
transfer_mobilenetv3_Adam_no_aug	0.798733	0.810795	0.882465	0.798733
transfer_mobilenetv3_SGD_aug	0.987059	0.987084	0.987226	0.987059
transfer_mobilenetv3_SGD_no_aug	0.865363	0.868061	0.892663	0.865363
efficientnet_Adam_aug	0.974119	0.974135	0.974745	0.974119
efficientnet_Adam_no_aug	0.814978	0.830210	0.889781	0.814978
efficientnet_SGD_aug	0.988436	0.988457	0.988505	0.988436
efficientnet_SGD_no_aug	0.883260	0.879453	0.894579	0.883260
resnet_Adam_aug	0.970540	0.970488	0.971175	0.970540
resnet_Adam_no_aug	0.759086	0.784152	0.870011	0.759086
resnet_SGD_aug	0.990363	0.990353	0.990389	0.990363
resnet_SGD_no_aug	0.891520	0.889868	0.908285	0.891520

Tabel 1 Metrics of the models

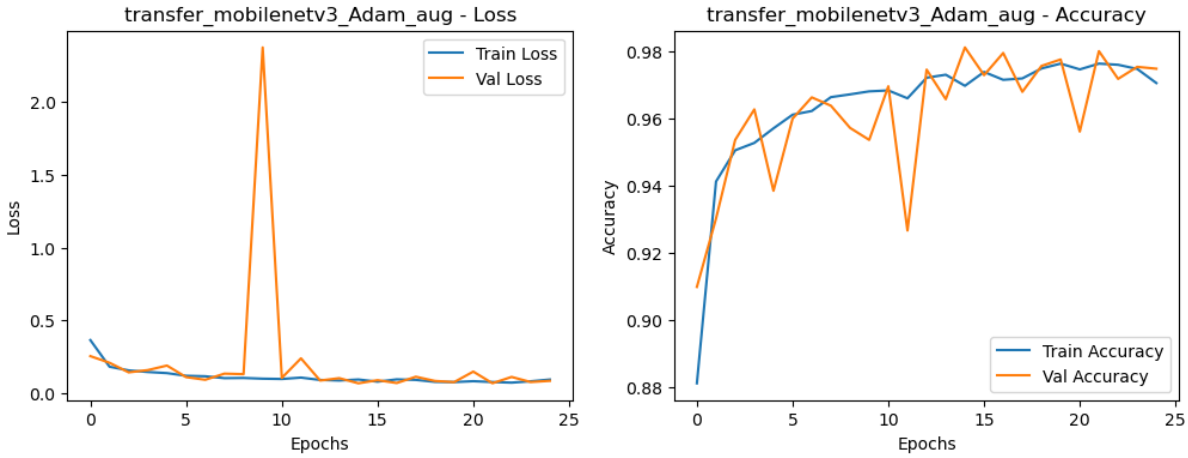
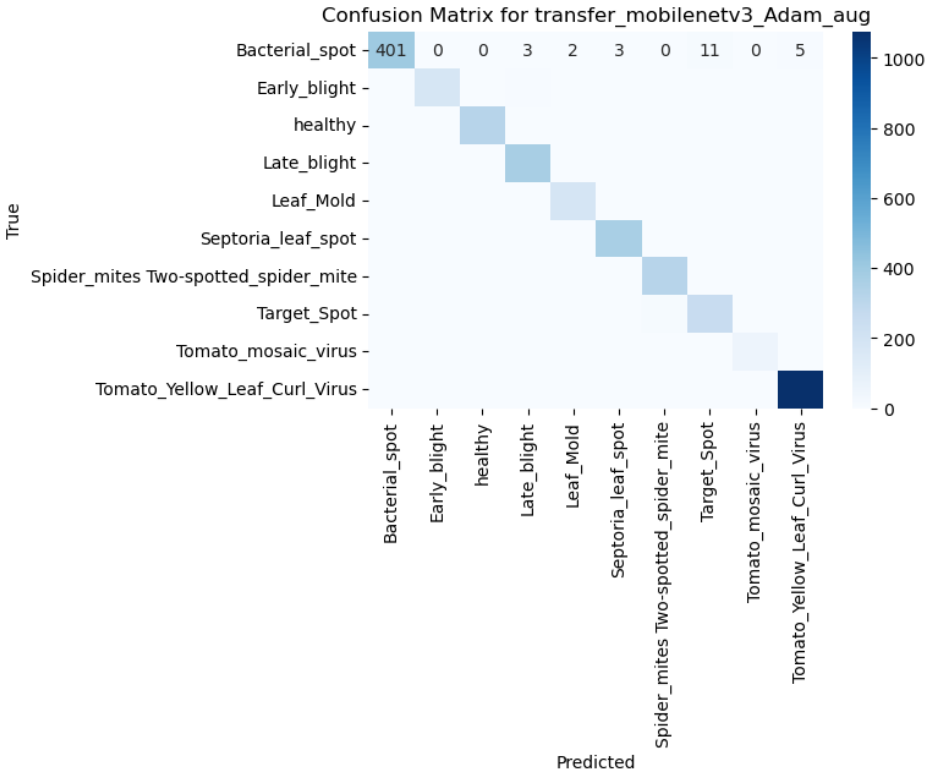


Figure 7 Training history



5.7 How to use the models

The code in the file prediction. ipynb is mainly used to detect the health state of tomato leaf by using the model we trained. The input parameter is a picture, and the output result is whether the tomato leaf in the picture is healthy and the reliability of the Prediction.

```
# Example usage
image_path = "../dataset/plantvillage/color/Tomato___Early_blight/0abc57ec-7f3b-482a-8579-21f3b2fb780b___RS_Erly.B 7609.JPG"
# image_path = "../dataset/plantvillage/grayscale/Tomato___Leaf_Mold/0ae36892-5cb1-476e-8a51-b7fd8183a535___Crnl_L.Mold 6728.JPG"
predicted_label, confidence = predict(image_path)
print(f"Predicted: {predicted_label}, Confidence: {confidence:.2f}")

✓ 0.0s
Predicted: Tomato___Early_blight, Confidence: 1.00
```

Figure 8 Using our trained model to detect disease