Palestine Polytechnic University

College of Information Technology and

Computer Engineering

Computer System Engineering

Monitoring the Health status of the plants using a Computerized system

Team Members:

Ahmad Tebakhi

Amjad Ibrahim Abdeen

Mohammed Sameh Nasereddin

Supervisor:

Dr.Mazen Zalloum

January 4, 2018

Acknowledgement

Firstly, we thanks Allah for helping us to work in this project, We also want to thanks our supervisor Dr.Mazen Zalloum ,he gave us a golden opportunity to work in this project and for his support which helped us to improve our knowledge and performance.

We also want to express our special thanks to our teachers in the collage, they give us their time to help, learn and support us.

We are also thankful to our family, especially our parents to help and support us at every time and with each step in our life. And we thankful to our friends and to Palestine Polytechnic University.

Abstract

Recently, the plants exposed to many serious diseases that may destroy them completely, impacting negatively on the annual production of agricultural crops. It is very difficult to monitor the health of agricultural crops manually specially at night. For this reason, we need to build a system to monitor and examine the state of health of plants and protect them from possible risks.

As we know that plants needs several factors to grow properly, and most important of these factors: the availability of adequate light, the appropriate temperature, the ratio of adequate humidity, in addition to provide the amount of oxygen needed by the plants, so the health status of the plant will be the main idea of our project.

The system that we have built based on the idea of monitoring the health status of the plant by taking a pictures for the plant's leaf periodically, and if a defect was detected on it, the disease will be identified, in addition it will determine the proportion of the disease.

Contents

List of Figures

Chapter 1
Introduction

1.1 Overview of the project

Expert systems are one of the most important new technologies that have been used in many fields in our life. Scientists are trying to develop expert systems to understand the environment then navigate with the capability of avoiding obstacles and move to the target in a less time.

The main problem in our project is to check the environmental conditions surrounding plants, like temperature and humidity, in addition the changes that happened upon the leaves .

Our project depends on deep learning using Tensorflow api. So the main idea of project is to build a system to detect any disease that will effect on the plants.

1.2 Motivation

Intelligent systems have become dramatically required in recent times, some examples of smart device systems, fire detection and theft detection device and many other devices important in our lives. Because of agriculture is important in our society the farmers need a device to protect the products from any damage or defect.

It is very difficult to farmers in pursuing agricultural crops continuously. That's why we want to build a project that saves farmers time and effort. The project pursuing health status of the plants and provide a detailed report to farmers contain state of health of plants, and to provide a system that provides all appropriate climatic conditions of the plants automatically.

1.3 Goals and Objectives

The goal of this system is to detect the disease that affects plant leaves constantly using a dedicated camera.

Project objectives :

1. Detect any possible disease that affecting plant leaves, and giving a certain percentage of the type of the disease affected by the leaf.
2. The system will check the humidity, if the amount of moisture is reached a certain percentage, the system gives a warning alarm.
3. The system will check the temperature, if the available heat of the plant is reached a certain percentage, the system gives a warning alarm.

1.4 Short description of the project

The project consists of two parts, the first one, which depends on the image of the leaves that have been taken, the image will be processed and classified as one of the three classes which are powdery mildew, downy mildew, healthy plant, and the system will give an approximate proportion of the type of the disease.
The second section depend the environment conditions that may affect the crop, so the system examines both the humidity and temperature for the crop environment using a sensor that measure and compare if it is normal degree for the crop or not.

1.5 Problem analysis

The main problem is how the system can reveal the disease and how to deal with it and the most important in this system that will reveal the disease before it's too late, to protect it from damage.
Our system wants to take picture of the plant to be analyzed by "Deep learning" and revealed that the disease is present or not, and if the disease is detected, the system

will classify the disease according to the images entered into the system on which the system was trained , and to provide adequate health for plants we needed sensors like as humidity sensor and temperature sensor in our project.

1.6 List of requirements

The most important requirement that we need in our project:

1. High performance Laptop.
2. Humidity, and temperature sensor.
3. Microcontroller “Arduino”: To deal with motors, and sensors.
4. Camera: To take a picture for the plant’s leaves.

1.7 Expected results

We expected to produce a complete system that classify the images according to many types of leaves diseases using deep learning (through tensorflow api) and giving us an approximate percentage of the disease , in addition to measure the temperature and humidity of the environment of the plant to stay in healthy state.

1.8 Limitations

1. Accuracy : The accuracy of the detecting have many problems like it detect something not belong to the leaves as if it is a leaf.
2. Number of diseases : The number of classes that we classify the leaf to is 3 classes which are (powdery mildew, downy mildew, healthy plant) so in our project we can check the existence of 2 diseases and if the diseases are not exist the plant will be healthy.
3. We collected almost 1500 image for the classes that we have , 500 image for each class , and this number is not that big because if we want to detect more accuracy we should collect more images.

4. the gpu that we used (1050Ti) is kind of good but if we used a better gpu the performance will be faster in training process and in detecting process because we are limited in 4 GB Vram.

1.9 Literature review

In previous studies, we saw the cultivation of the plant in terms of spraying water, looking about the temperature of the plant, but the care of the plant in terms of disease was usually done manually through the farmer.

In our project we would like to add a new one, which is to check the plant using a computer system to protect it from exposure to diseases. Our work will be about the some types of plant, where we will examine it from a number of diseases such as powdery mildew and downy mildew. The following is a picture showing the powdery mildew disease that affects the cucumber, which causes the appearance of white spots on the paper, leading to poisoning of the plant.

Figure 1.1 Powdery mildew disease that infects cucumber leaf

1: Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification

The latest generation of convolutional neural networks (CNNs) has achieved impressive results in the field of image classification. This project is concerned with a new approach to the development of plant disease recognition model, based on leaf image classification, by the use of deep convolutional networks. Novel way of training and the methodology used facilitate a quick and easy system implementation in practice. The developed model is able to recognize 13 different types of plant diseases out of healthy leaves, with the ability to distinguish plant leaves from their surroundings. According to our knowledge, this method for plant disease recognition has been proposed for the first time. All essential steps required for implementing this disease recognition model are fully described throughout the paper, starting from gathering images in order to create a database, assessed by agricultural experts.[1]

2: Using drones to detect crop diseases

For as long as bananas have been grown, banana growers in Southeast Asia have had to struggle with fungal diseases such as the dreaded Panama disease. The loss is enormous and fungicides do not help. Through early detection, the disease can be prevented from spreading. However this is a very labor intensive process.
The use of remote sensing to detect diseases could be an efficient way to identify bad spots. The remote sensing system being developed by Wageningen-based consultancy Triple20 could be used to detect diseases like Panama disease. Growers could use the information gathered to take proactive measures at an early stage [2]

3: Monitoring and Detection of Agricultural Disease using Wireless Sensor Network

Existing systems for forecasting the disease mostly depends on image processing technologies. Drawback of the existing system is that they wait till symptoms appear and then only the disease can be detected. That's why such type of systems is unable

to help treating the disease at an early stage. Grape diseases like downy mildew is mostly dependent upon weather based parameter like humidity, temperature and wind speed. When any favorable weather condition occurs zoospores in downy start generating spores that enters into the leaves of grape via stomata of the leaves. If favorable weather condition and the probability of disease is detected then it it very helpful for farmers to prevent infection of disease and reduce the cost of production. [3]

1.10 Summery

The system will take pictures from period to period for the plant, then analyze them based on the algorithms used and give the result of the state of health of the plant.
In addition, it will monitor the health status of the plant through reading humidity and temperature.

Chapter 2
Background

2.1 Overview

This chapter introduces the theoretical background of the project, provides a description of the disease that affects the plant and how it will be treated, and some description of hardware and software component used in the system, finally some description of design specification and constrains.

2.2 Theoretical background

The problem with greenhouse is that it is possible for the plant to infects a disease that would damage the crop, resulting in significant financial loss [5]. Therefore, the device must detect the type of disease through the work of Deep learning with neural networks (image processing) of the leaf. In addition, plant care in terms of humidity and heat suitable for the plant using the sensors.

2.3 Methodology

2.3.1 Convolutional Neural Network

One of these is neural networks – the algorithms that underpin deep learning and play a central part in image recognition and robotic vision.

A single neuron in the brain, receives signals – as many as 100,000 – from other neurons. When those other neurons fire, they exert either an excitatory or inhibitory effect on the neurons they connect to. And if our first neuron's inputs add up to a

certain threshold voltage, it will fire too.

In an artificial neural network, signals also travel between ’neurons‘. But instead of firing an electrical signal, a neural network assigns weights to various neurons. A neuron weighted more heavily than another will exert more of an effect on the next layer of neurons. The final layer puts together these weighted inputs to come up with an answer.

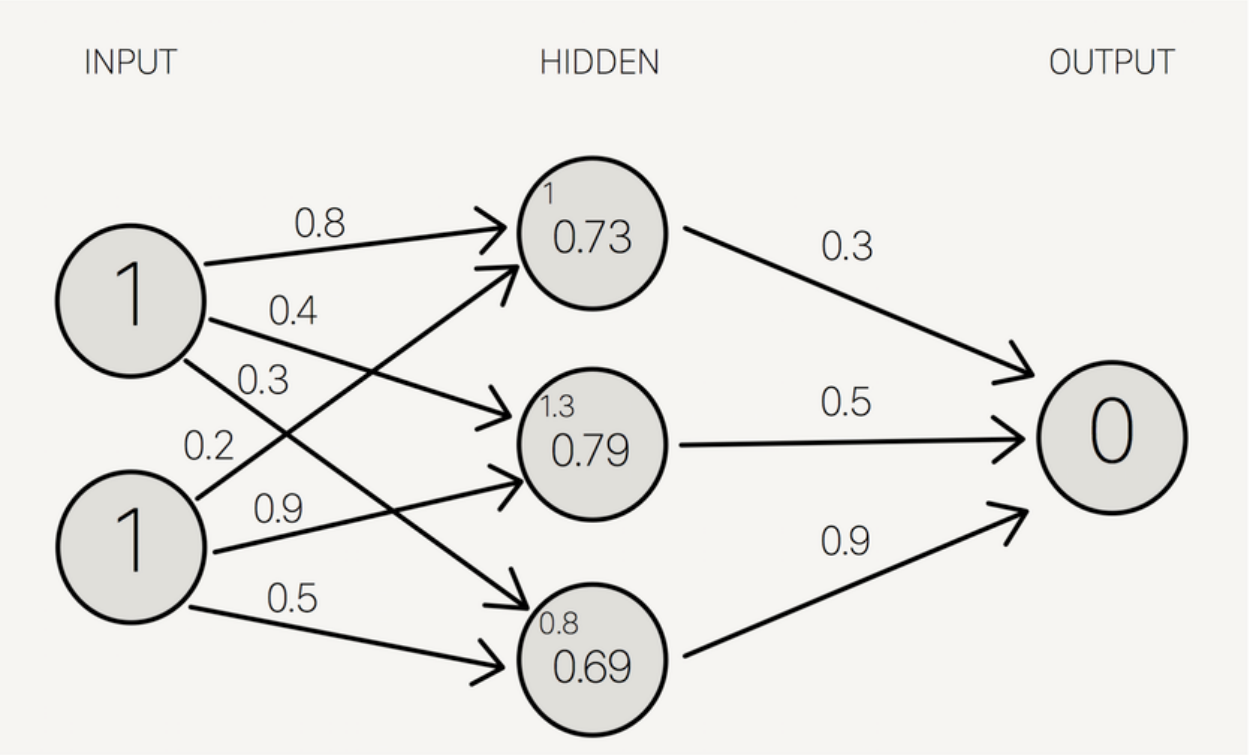

Figure 2.1 Neural Network example

Let’s say we want a neural network to recognise photos that contain at least one leaf. But leaves don’t all look exactly alike – consider The leaves come in different shapes and sizes. Nor do photos necessarily show them in the same light, at the same angle and at the same size.

So we need to compile a training set of images – thousands of examples of leaves, which we (humans) label “leaf”, and pictures of objects that aren’t leaves, labelled (you guessed it) “not leaf”.

These images are fed into the neural network , an image is converted into data which moves through the network and various neurons assign weights to different elements. At the end, the final output layer puts together all the pieces of information – pointed Cuticle, Guard cell, Stoma, Phloem, itc… – and spits out an answer: leaf.

The neural network compares this answer to the real, human-generated label. If it matches, great ! If not , the neural network makes note of the error and goes back and adjusts its neurons' weightings . The neural network then takes another image and repeats the process, thousands of times, adjusting its weightings and improving its cat-recognition skills – all this despite never being explicitly told what "makes" a leaf.

Unsupervised learning, on the other hand, uses unlabelled data. Neural networks must recognise patterns in data to teach themselves what parts of any photo might be relevant

Powerful graphics processing units, or GPUs, burst onto the scene, meaning researchers could run, manipulate and process images on desktop computers rather than supercomputers.

As with traditional neural networks, convolutional counterparts are made of layers of weighted neurons. But they're not just modelled on the workings of the brain; they, appropriately enough, take inspiration from the visual system itself.

Every layer within a convolutional neural network applies a filter across the image to pick up specific patterns or features. The first few layers detect larger features, such as diagonal lines, while later layers pick up finer details and organise them into complex features such as an Phloem.

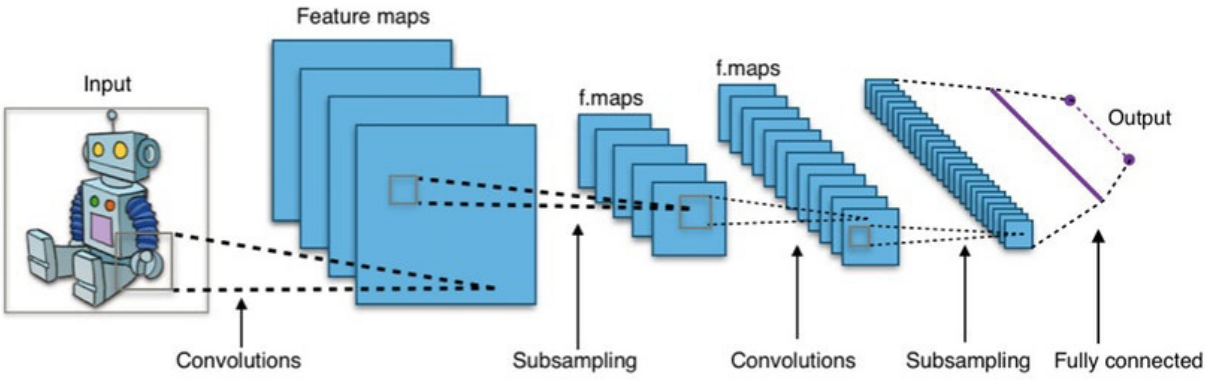

Figure 2.2 convolutional neural network

The final output layer, like an ordinary neural network, is fully connected (that is, all neurons in that layer are connected to all neurons in the previous layer). It puts together highly specific features –to produce an ultra-precise classification: leaf.[4]

2.3.2 Faster R-CNN

State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations. Advances like SPPnet and Fast R-CNN have reduced the running time of these detection networks, exposing region proposal computation as a bottleneck. In this work, Region Proposal was introduced as Network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully-convolutional network that simultaneously predicts object bounds and objectness scores at each position. RPNs are trained end-to-end to generate high quality region proposals, which are used by Fast R-CNN for detection. With a simple alternating optimization, RPN and Fast R-CNN can be trained to share convolutional features.

Recent advances in object detection are driven by the success of region proposal methods) and region-based convolutional neural networks (R-CNNs) . Although region-based CNNs were computationally expensive as originally developed in , their cost has been drastically reduced thanks to sharing convolutions across proposals. The latest incarnation, Fast R-CNN , achieves near real-time rates using very deep networks, when ignoring the time spent on region proposals. Now, proposals are the computational bottleneck in state-of-the-art detection systems.

Region proposal methods typically rely on inexpensive features and economical inference schemes. Selective Search (SS), one of the most popular methods, greedily merges superpixels based on engineered low-level features. Yet when compared to efficient detection networks, Selective Search is an order of magnitude slower, at 2s

per image in a CPU implementation. EdgeBoxes currently provides the best tradeoff between proposal quality and speed, at 0.2s per image. Nevertheless, the region proposal step still consumes as much running time as the detection network.

One may note that fast region-based CNNs take advantage of GPUs, while the region proposal methods used in situation are implemented on the CPU, making such runtime comparisons inequitable. An obvious way to accelerate proposal computation is to re-implement it for the GPU. This may be an effective engineering solution, but re-implementation ignores the down-stream detection network and therefore misses important opportunities for sharing computation.

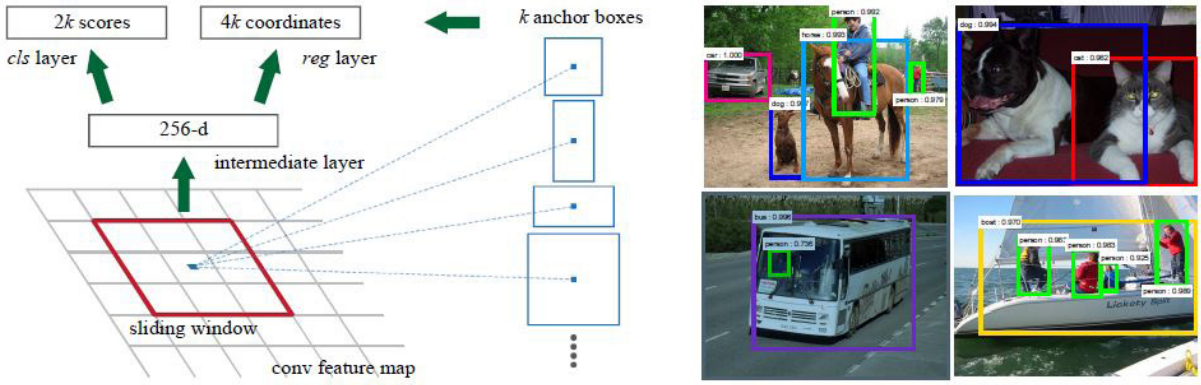

Figure 2.3 Faster rcnn

For training RPNs, its assigned a binary class label (of being an object or not) to each anchor. Its assigned a positive label to two kinds of anchors: (i) the anchor/anchors with the highest Intersectionover- Union (IoU) overlap with a ground-truth box, or (ii) an anchor that has an IoU overlap higher than 0.7 with any ground-truth box. Note that a single ground-truth box may assign positive labels to multiple anchors.its assigned a negative label to a non-positive anchor if its IoU ratio is lower than 0.3 for all ground-truth boxes. Anchors that are neither positive nor negative do not contribute to the training objective.[5]

2.4 Hardware components

2.4.1 Laptop

We will use the laptop to analyze the images that were captured by the camera through (Tensorflow API) using a dedicated video card (1050 Ti) and showing the results of the plant, and their classification.

The laptop that we is Dell Inspiron 7567 and it has the following specification that help us in training images and image recognition process.

- 7th Generation Intel® Core™ i7-7700HQ Quad Core
- 16GB RAM
- NVIDIA® GeForce® GTX 1050 Ti with 4GB GDDR5
- Windows 10 Home 64-bit English [6]

Figure 2.4 Dell Inspiron 7567

2.4.2 Camera

We will connect a high-resolution camera to the laptop, which will capture images of the plant's leaf during certain predetermined periods using Webcam Surveyor application that will do the job for us.

The photos that have been taken will be send to the system and checked if the plant affected by the disease or not.

Figure 2.5 Webcam

2.4.3 Arduino uno microcontroller

Is an open source computer hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world.[7]

In this microcontroller We have linked some of the sensors, one of these sensors is humidity and temperature sensor .In addition, we connected servo motor for controlling the camera to move in many directions.

Figure 2.6 Arduino uno microcontroller

2.4.4 Humidity and Temperature sensor

A humidity sensor (or hygrometer) senses, measures and reports the relative humidity in the air. It therefore measures both moisture and air temperature. Relative humidity is the ratio of actual moisture in the air to the highest amount of moisture that can be held at that air temperature. The warmer the air temperature is, the more moisture it can hold. Humidity / dew sensors use capacitive measurement, which relies on electrical capacitance. Electrical capacity is the ability of two nearby electrical conductors to create an electrical field between them. The sensor is composed of two metal plates and contains a non-conductive polymer film between them. This film collects moisture from the air, which causes the voltage between the two plates to change. These voltage changes are converted into digital readings showing the level of moisture in the air. [8] And the purpose of humidity and temperature sensor in our project is to check the humidity and temperature in the air and according to its output we will decide to make some steps to decrease it because some diseases of the plants are appeared because of the high ratio of humidity and temperature.

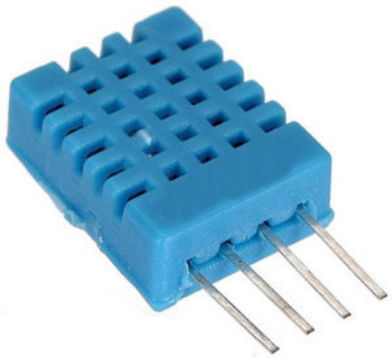

Figure 2.7 Humidity sensor

2.4.5 Servo motor.

A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor

coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors. [9]

We used Servo motor In order to place the camera on it, to control the angles through a program installed on the phone (Android platform) or by programming it to move is a certain directions without using a phone.

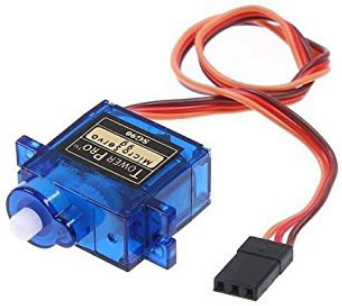

Figure 2.8 Servo motor

2.4.6 LCD 16x2

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications [10], and in our project we used it to display humidity and temperature degree.

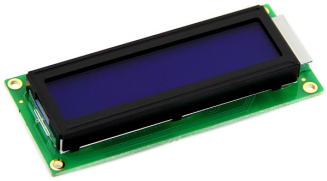

Figure 2.9 LCD 16x2

Chapter 3
System Design

3.1 Overview

This chapter illustrates the conceptual design of the system, it also shows a block diagram and all of its components, flowchart of the system and finally software requirements.

3.2 Brief description of the system

We have done a shape that contains the plant and tools used in a smooth and flexible, so that it is easy for the user to work without difficulties. The camera will take a picture of the plant and give it to the Laptop for analysis.

The Arduino will take the readings from the sensors, and control the temperature and humidity operations. Also, control the servo motor rotation.

The Arduino display the sensor's readings on the LCD and when the temp. and humidity are abnormal the lcd will warn us.

3.3 System Diagrams

3.3.1 Block diagram

It shows the principal parts or functions used and performed by the device.

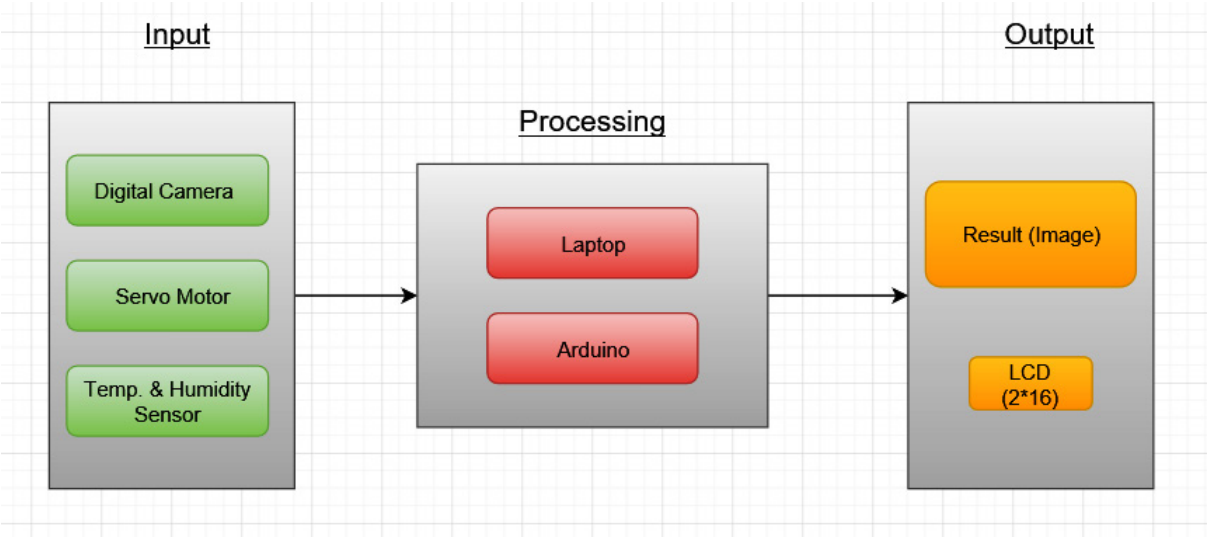

Figure 3.1 shows block diagram of the system

3.3.2 Description diagram

It shows the hardware components and how they will work together.

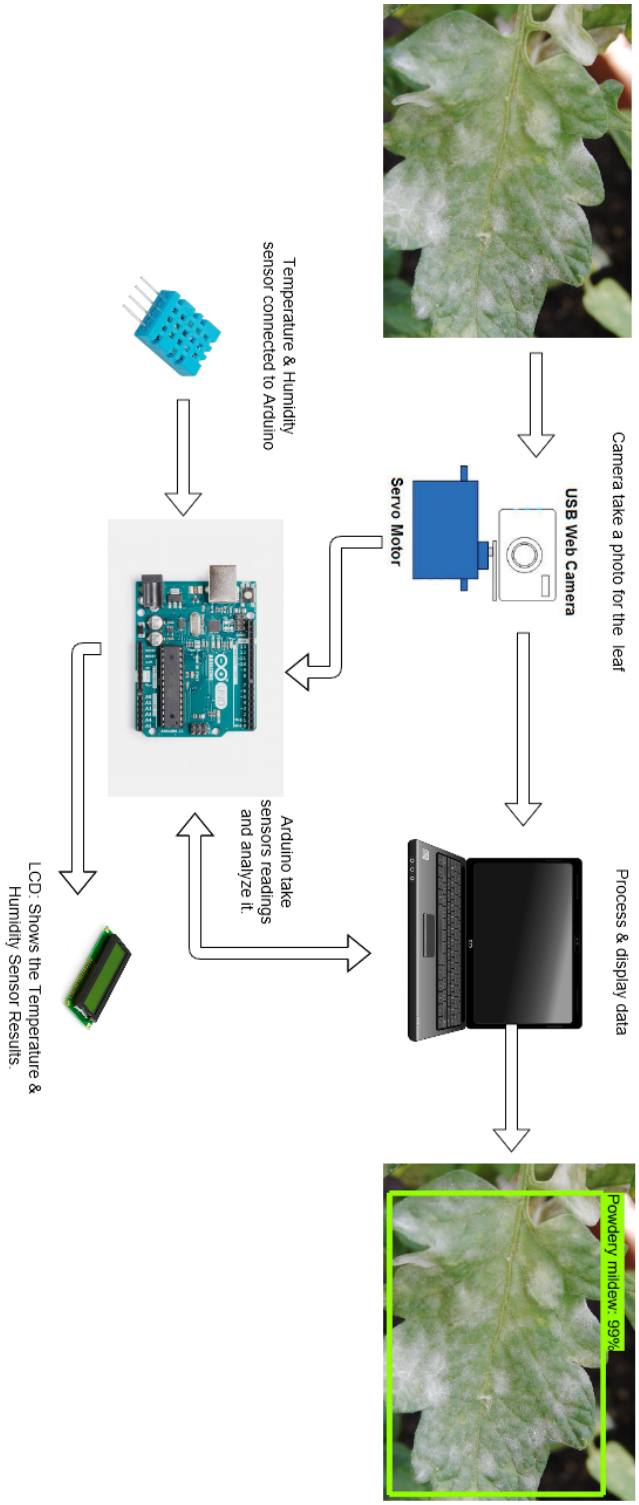

Figure 3.2 description diagram of the system

3.4 System Flowchart

It shows the sequential steps that the system performs.

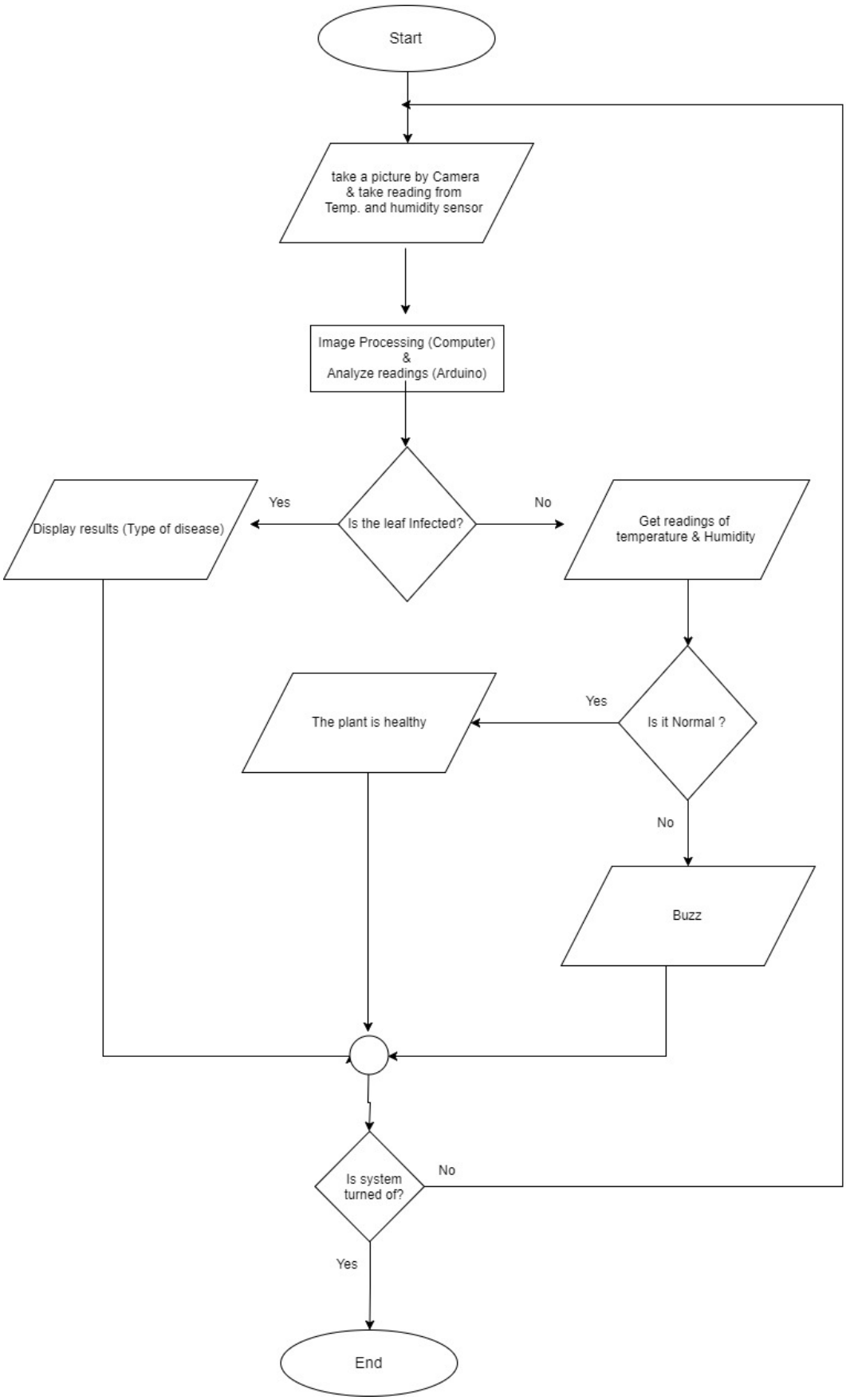

Figure 3.3 Flow chart of the system.

3.5 Software design

In this section we will briefly mention the software that we used to implement our system. The software includes the following:

Python: The language that we used to build the project.

Tensorflow API: Is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and also used for machine learning applications such as neural networks.
Android app.: To control the servo motor.[11]

Webcam Surveyor: Program to take a pictures from time to time using webcam.

Jupyter Notebook : A web-based notebook environment for interactive computing [12] which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc..) as well as executable documents which can be run to perform data analysis, The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access or can be installed on a remote server and accessed through the internet , In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.[13]
Fritzing: To move from physical prototyping to actual product.

Chapter 4
Implementation

4.1 Hardware implementation

We connected dht sensor with the Arduino in order to measure the humidity and temperature and the readings displayed on the lcd , in addition we connected servor motor in order to control the direction of the camera which is mounted upon it , and this circuit showing us how all of the component connected with the Arduino.

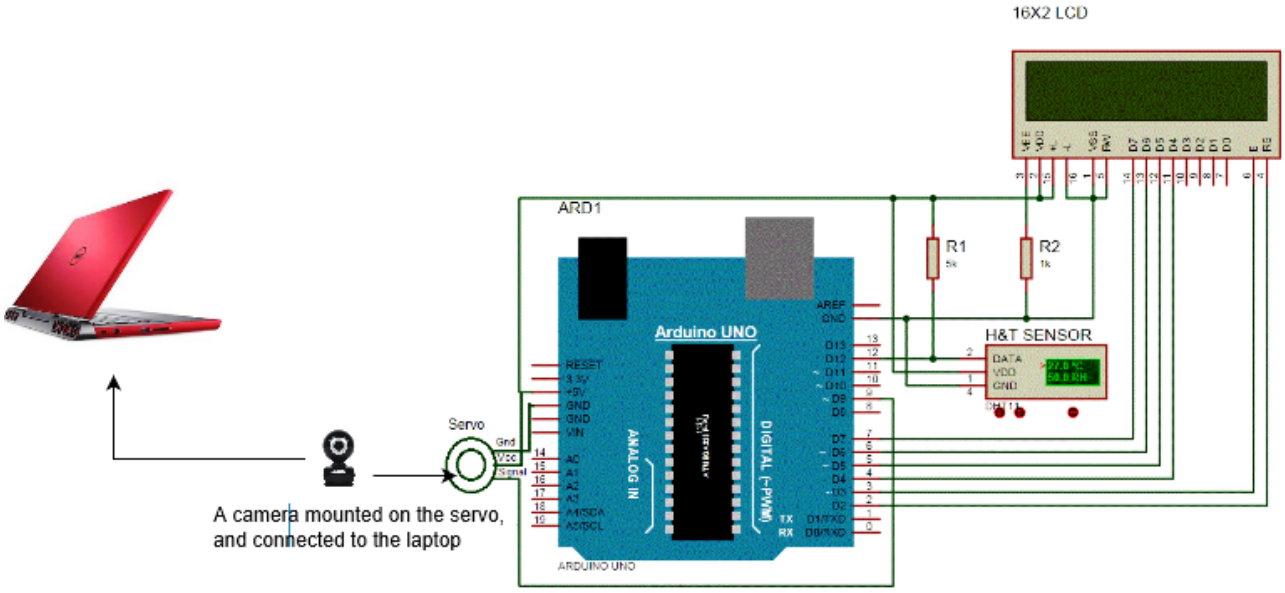

Figure 4.1 Hardware Circuit

4.2 Software implementation

We used Google Images, Bing, and ImageNet sites to collect our images dataset of plant's leaves and these leaves contain 3 classes which is two classes of diseased leaves (powdery mildew and downy mildew) and one class for healthy leaves . In general, the images that we collected are around 1500 image, 500 image for each class , the images contain many kinds of plant's leaves like cucumbers, squashes (including pumpkins), melons, watermelons, lemon, grape, furthermore the images have a large variations in scale, pose and lighting and the dimensions of the images ranging from 144x151px to 5456x3632px , which means that we collect (low - medium - high) resolutions.

Figure 4.2 Sample of the images

, and then we annotate the images manually using 'LabelImg' app (LabelImg is a graphical image annotation tool that is written in Python und uses Qt for the graphical interface. It supports Python 2 and 3. It's easy to use and the annotations are saved as XML files in the PASCAL VOC format[14]) by Drawing a boxes on the objects (plant's leaves in our case),and then we add a title or name to each box and we repeated this process until we had a bunch of labeled images.

The below figure describes labelling process using LabelImg app.

Figure 4.3 Labelling images process “LabelImg”

Mentioning that XML files contains some details about the image and the labels that we annotated to like width ,height ,xmin , ymin , xmax , ymax , and after labeling the images we collected all of these images and their XML’s together.

In order to convert them to a singular CSV file (which is a big database contains the details of images and their label’s name, dimensions) ,

,mentioning that most of images contains many leaves , which means we drew many labels for almost every single image , and after counting number of labels we have had 4758 label for 1453 image , all of them contained in two CSV file (test_labels. csv) and (train_labels.csv).

And we convert the xml files to csv files using xml_to_csv script ,the below figure contains a script that gathers xml details in CSV files.

generate_tfrecord.py | xml_to_csv.py

```python
import os
import glob
import pandas as pd
import xml.etree.ElementTree as ET


def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            value = (root.find('filename').text,
                     int(root.find('size')[0].text),
                     int(root.find('size')[1].text),
                     member[0].text,
                     int(member[4][0].text),
                     int(member[4][1].text),
                     int(member[4][2].text),
                     int(member[4][3].text)
                     )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df


def main():
    for directory in ['train','test']:
        image_path = os.path.join(os.getcwd(), 'images/{}'.format(directory))
        xml_df = xml_to_csv(image_path)
        xml_df.to_csv('data/{}_labels.csv'.format(directory), index=None)
        print('Successfully converted xml to csv.')


main()
```

Figure 4.4 xml_to_csv script

, then csv files can be converted to TFRecord files that the Tensorflow api can deal with, so we used 1360 images for training (train.records) and 75 images for testing (test.records) and we used generate_tfrecord.py script to do the job.

```
22
23  flags = tf.app.flags
24  flags.DEFINE_string('csv_input', '', 'Path to the CSV input')
25  flags.DEFINE_string('output_path', '', 'Path to output TFRecord')
26  FLAGS = flags.FLAGS
27
28
29  # TO-DO replace this with label map
30  def class_text_to_int(row_label):
31      if row_label == 'Powdery mildew':
32          return 1
33      elif row_label == 'Downy mildew':
34          return 2
35      elif row_label == 'Healthy plant':
36          return 3
37      else:
38          None
39
40
```

Figure 4.5 generate_tfrecord script

After we created the required input file for the tensorflow API (TFRecord files), we now can train our model so we starting by setup the configuration of the model Anyway we have two options to start. First one We can use a pre-trained model, and then use transfer learning to learn a new object , or we could learn new objects entirely from scratch. The benefit of transfer learning is that training can be much quicker, and the required data that we might need is much less. For this reason,it is always recommended to use a checkpoint pre-trained model to start from rather than training from scratch because training from scratch process could take days before we get good results, so we have used transfer learning in our project.

By the way TensorFlow has a few pre-trained models with checkpoint files available, along with configuration files,some of them have high speed with low accuracy and others have low speed with high accuracy, so we used faster_rcnn_resnet101_coco model because it's more accuracy than others like ssd_mobilenet_v1_coco but its

speed of detecting is less, and in our project we focused at accuracy of detecting objects more than the speed , and before start training ,we used faster_rcnn_resnet101_coco. config as a configuration file that is suited with our pre-trained model, and in the configuration file, we modify batch size , it is set to 24 by default. Other models may have different batch sizes.

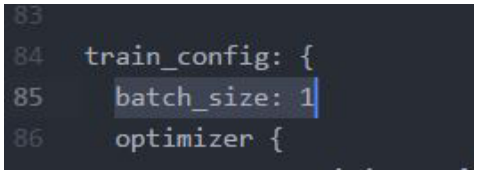

Figure 4.6 batch_size

In our case we faced a lot of memory errors which is resolved by decreasing the batch size to 1 to get the model to fit in our VRAM.

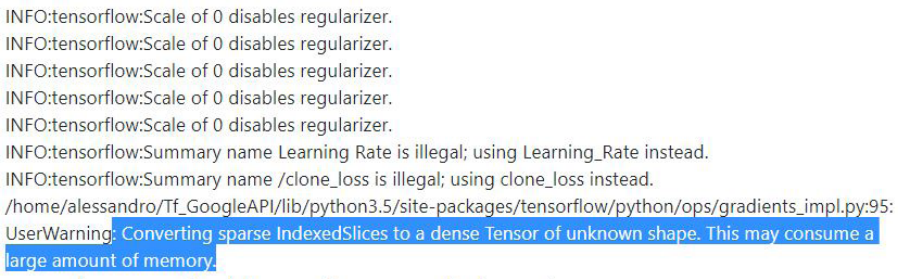

Figure 4.7 memory error

Finally, we made some changes on the configuration file first we change the checkpoint name/path according to pre-trained model that we used which is faster_ rcnn_resnet101_coco model and num_classes to 3 because we have three classes as we mentioned before (powdery mildew , downy mildew, Healthy plant) and we changed label_map_path: “training/object-detect.pbtxt” and also TFRecord files path to make tf_record_input_reader read them correctly

```
    num_steps: 200000
    data_augmentation_options {
      random_horizontal_flip {
      }
    }
  }

  train_input_reader: {
    tf_record_input_reader {
      input_path: "data/train.record"
    }
    label_map_path: "training/object_detection.pbtxt"
  }

  eval_config: {
    num_examples: 8000
    # Note: The below line limits the evaluation process to 10 evaluations.
    # Remove the below line to evaluate indefinitely.
    max_evals: 10
  }

  eval_input_reader: {
    tf_record_input_reader {
      input_path: "data/test.record"
    }
    label_map_path: "training/object_detection.pbtxt"
    shuffle: false
    num_readers: 1
    num_epochs: 1
  }
```

Figure 4.8 configuration file

Then we modify object-detect.pbtxt file, this file contains the name of the labels that we used in our project.

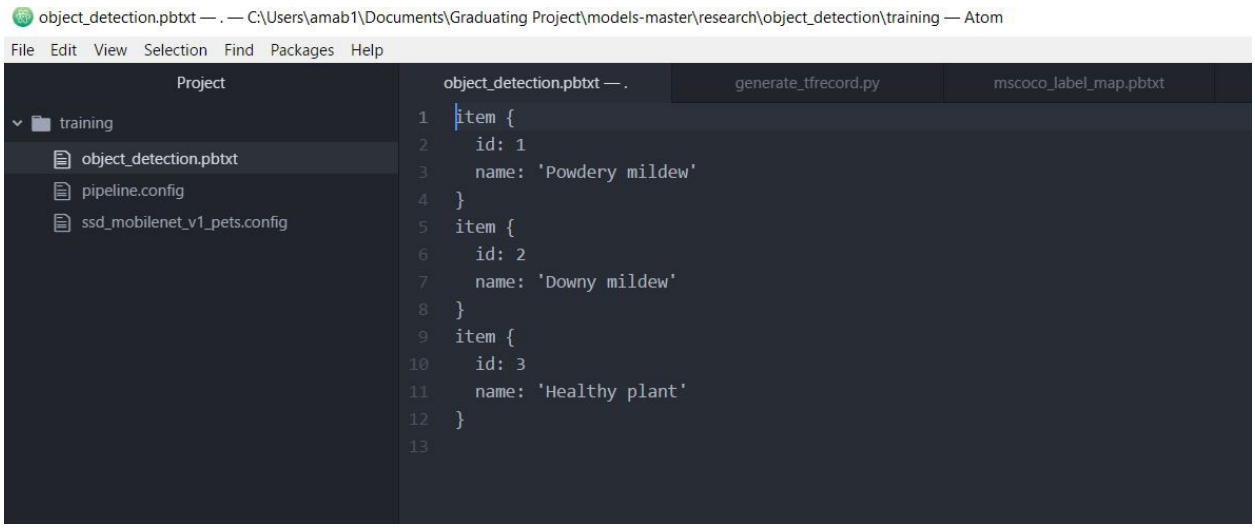

Figure 4.9 object-detect.pbtxt file

Training process can be either done locally or on the cloud (AWS, Google Cloud etc.).in our case we done the training process locally through DELL Inspiron 7567 Laptop using a dedicated GPU (1050ti) which is using Pascal Architecture and Frame Buffer of 4 GB GDDR5 and 768 of CUDA Cores , which helped us a lot to make the training process as fast as possible , in other word the training process walking through number of iterations , every single iteration taking more or less 1 second (depending on the dedicated gpu used and the batch size) in our case every iteration take almost ~0.9s and we continue in training process until we reached 200k iterations which make the error rate number decreasing over time and make the detection more accuracy, and this process took almost 3 days or more until it finished , and this picture describing what happened in training process.

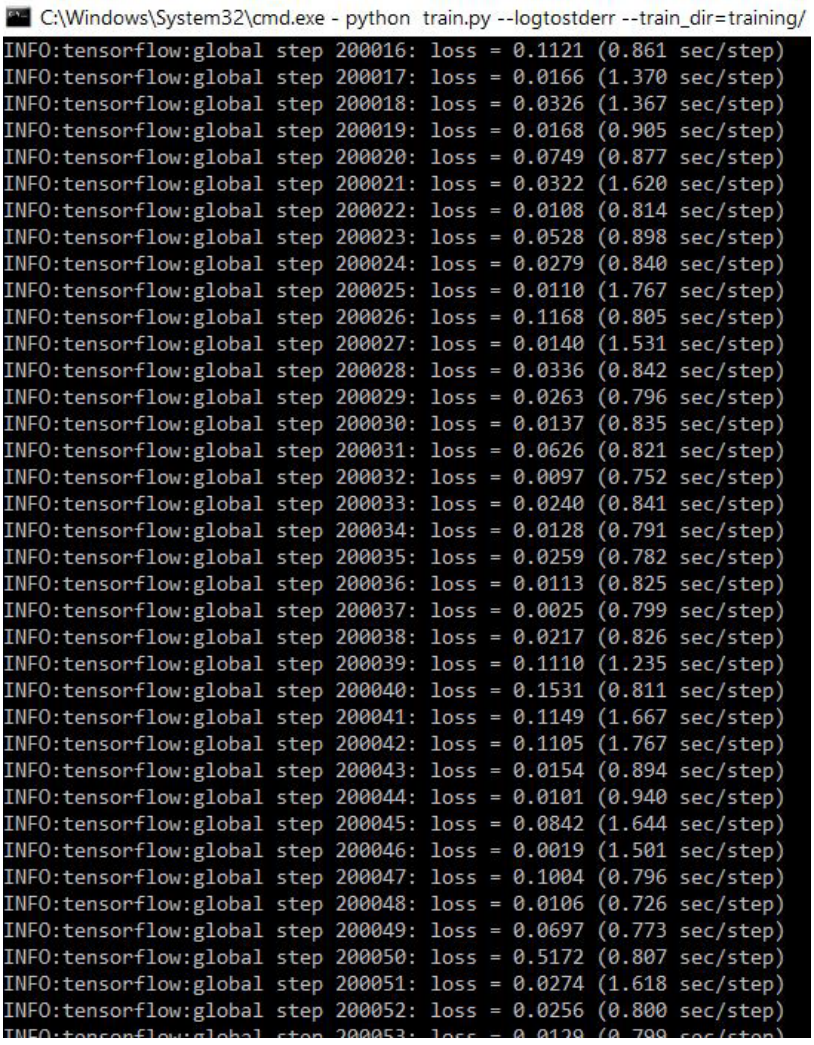

Figure 4.10 Training steps

We can check the total loss rate by open Tensorboard ,and this figure describes how loss rate decreasing by time and if we see the loss rate at the beginning it was very high (more than 1) which means that the error rate will high also so if we try to detect a leaf or anything else the system will detect it randomly so the accuracy is very low , but when the total loss rate decreased to (0.01245 or less) after 200k steps which is 3 days and 10 hours in time the system will be more accuracy to detect leaves and their diseases.

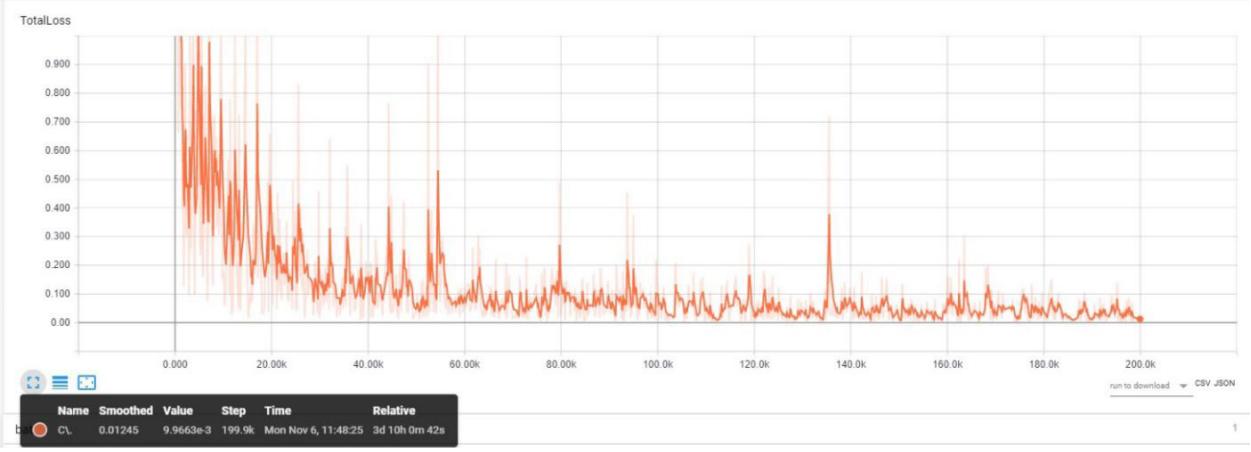

Figure 4.11 Total Loss

So after finishing with training, we exported the trained model and now we are ready to start taking photos for the leaves diseases and check whenever the leaf has a disease or not , and this figure showing the folder for the final model for leave's diseases detection.

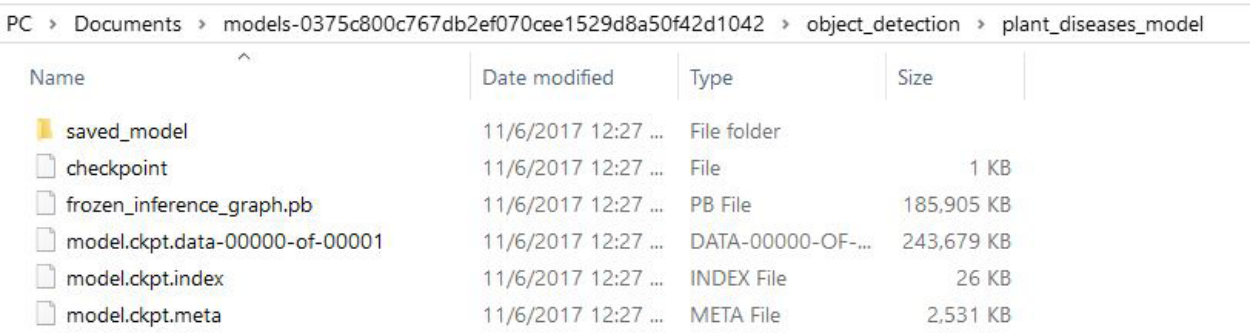

Figure 4.12 Plant_diseases_model folder

Chapter 5
Testing (Results)

5.1 Results

Our system made to detect just leaves and their diseases so we used Jupyter notebook to produces and shows the results ,When the notebook is executed (either cell-by-cell or with menu Cell -> Run All), the kernel performs the computation and produces the results. Depending on the type of computations (image recognition for plant's leaf disease in our case), the kernel may consume significant CPU and RAM[15]. Note that the RAM is not released until the kernel is shut-down.

So we tried to classify 100 images for many leaves to recognize diseases on them after processing them through "Jupyter notebook" using our trained model (Plant_diseases_ model) and we have 73% accuracy in predict these images and figure showing us how much the system consume of laptop resources (using task manager), like it takes 2-5 GB Ram and the dedicated gpu keep running along with the cpu until it processed all of the input images.

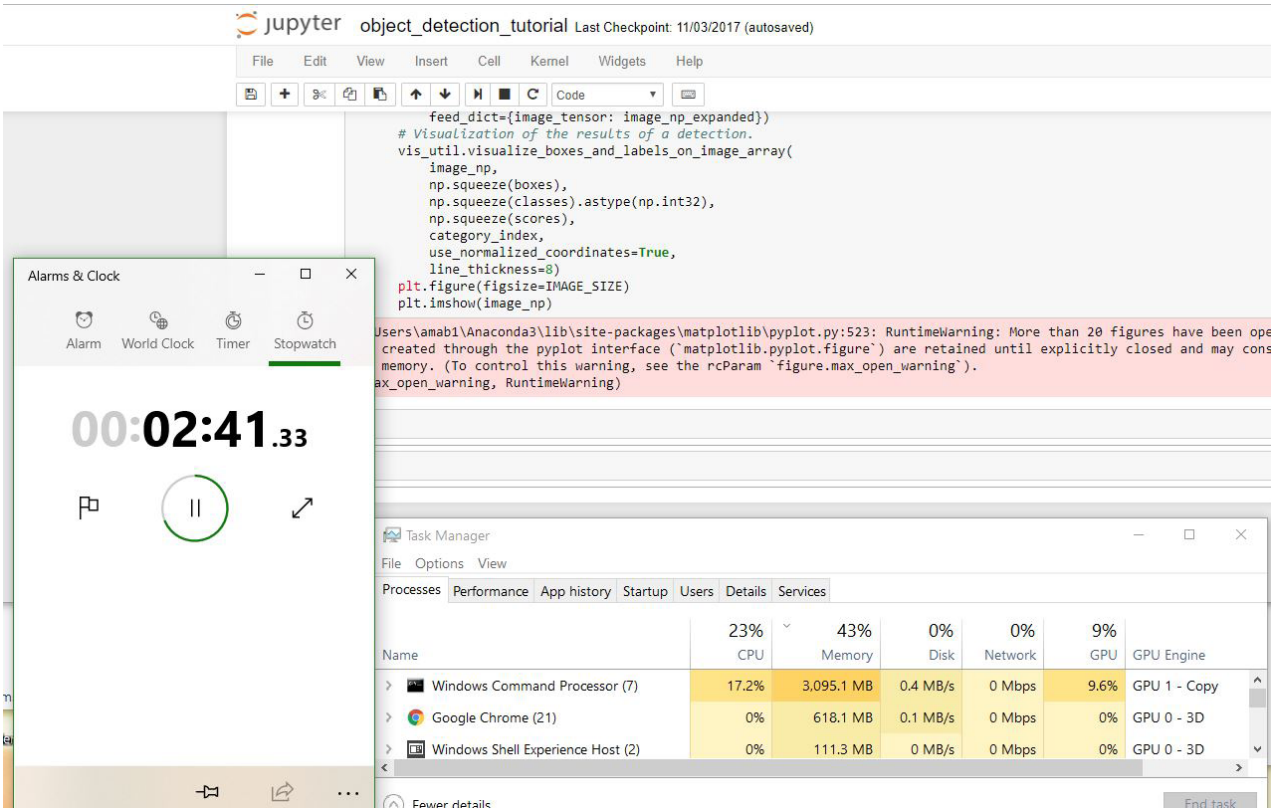

Figure 5.1 Jupyter notebook while processing

and after about 4 minutes the system finished image-processing for 100 image and it starts to produce labeled images like the below figure.

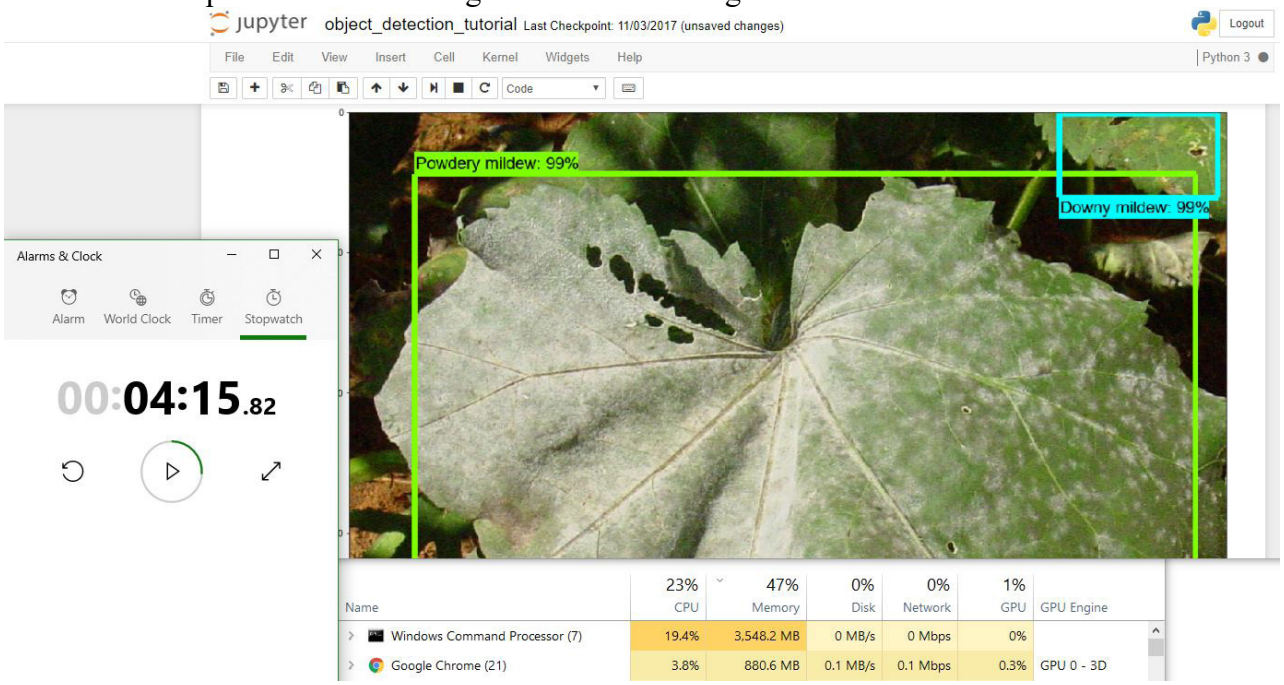

Figure 5.2 Jupyter notebook after processing

in the following figure we have 3 leaves with 3 different classes and the system predect them correctly mensioning that this image is not one of the trained images and the system classify it according to the trained images.

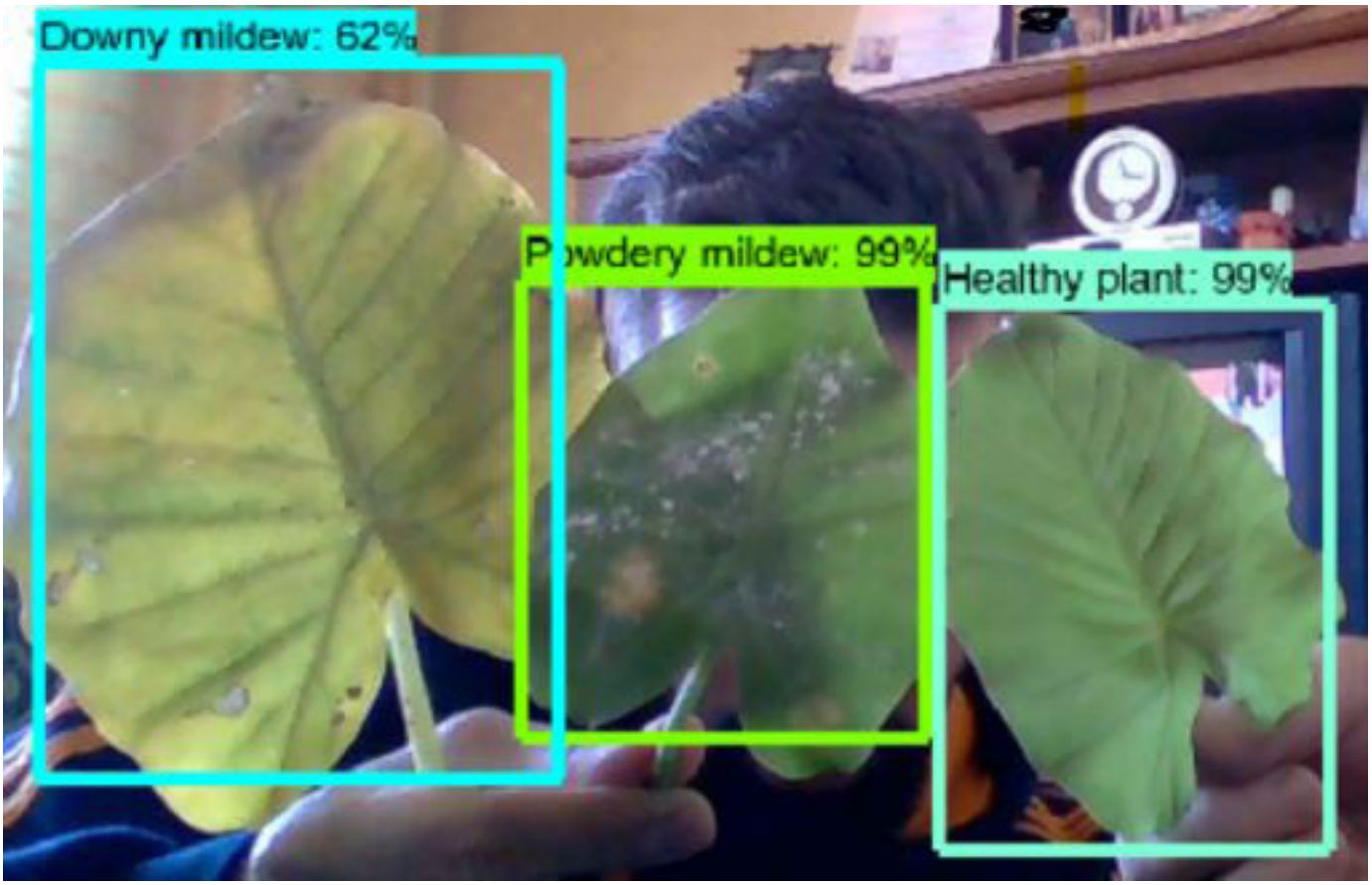

Figure 5.3 3 leaves with 3 different classes after processing

5.2 Drawbacks

There are some misclassifications while detecting images other than leaves. This is logical as we only trained the model on a small dataset which is almost 1500 image with three classes. To create a more generalized and robust leaves diseases detector, we just need much more data and classes to make the system more intelligence. That's just one of the limitations of AI right now , so our system is effective if it is used only in the field of plants. If another scope is used it will show us inaccurate (meaningless) results. for example the following image is not a plant leaf but the system recognize it as if it is a leaves, because of this the system will have many misclassifications if we make it try to recognize something else.

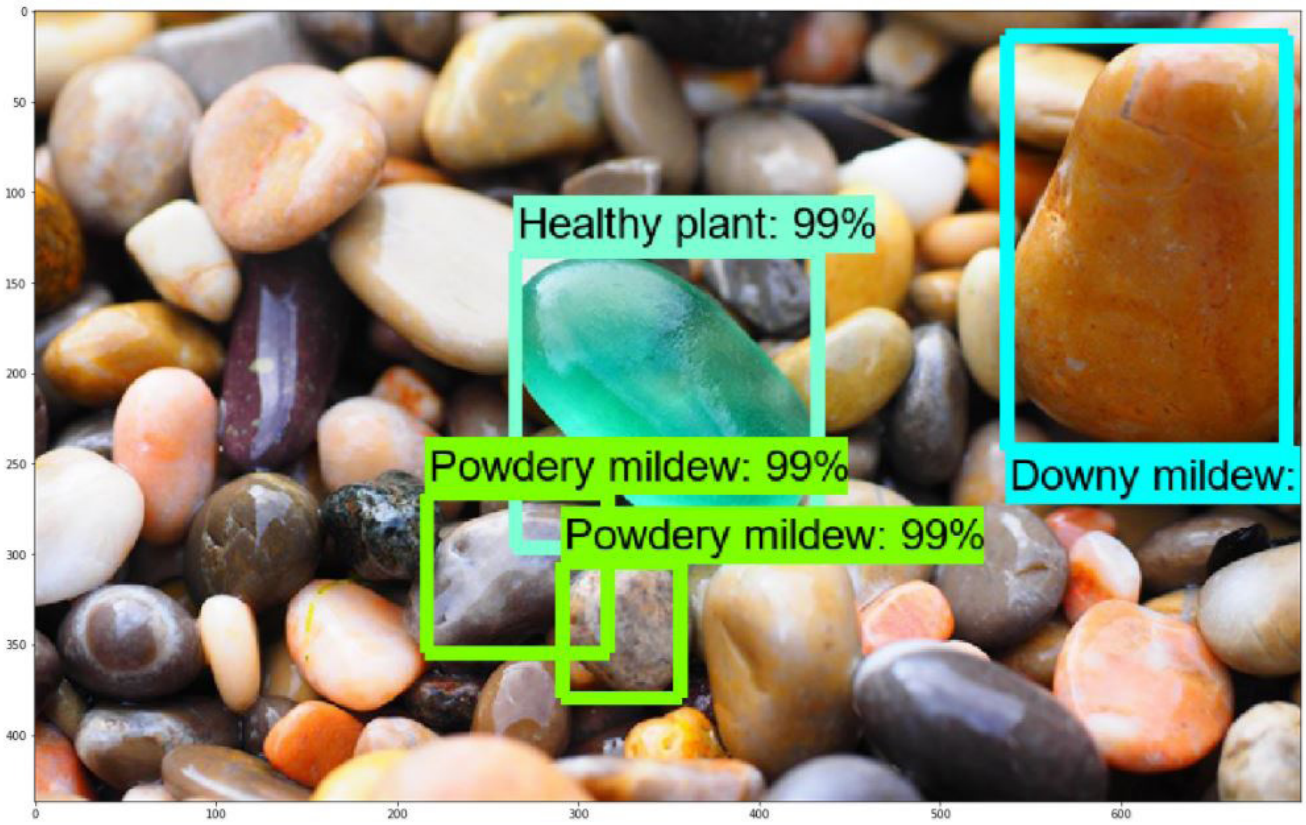

Figure 5.4 misclassifications image

5.3 Design option

We used laptop instead of Raspberry pi. This is because the specifications of the best Raspberry pi currently available are not sufficient to complete the process of training our module and it can't run the project efficiently.

The table below showing us the comparison between Raspberry pi 3 model B and Laptop (Dell Inspiron 7567) which is the laptop that we used in training and image-processing and detection.[16]

Comparison/ Device	Laptop (Dell Inspiron 7567)	Raspberry pi 3 model B
CPU	7th Generation Intel® Core™ i7-7700HQ Quad Core (6MB Cache, up to 3.8 GHz)	Quad Cortex A53 @ 1.2 GHz
Memoryi	16GB, 2400MHz, DDR4	1GB SDRAM
Video Card	NVIDIA® GeForce® GTX 1050Ti with 4GB GDDR5	400MHZ VideoCore IV

Our system requires high specifications laptop to do the training process using the gpu , and because tensorflow api uses cuda while processing so it needs just an NVIDIA GPU because it's the only gpus that supports cuda , in addition when we use better gpu the training and detection process will be faster and will take much less time and also we can use multiple gpus to give a very fast performance and this is used when we have a huge dataset(millions of images).

We can run this project using raspberry pi but of course the result that we will have will be as a percentage. for example, powdery mildew (score=0.96091) downy mildew(score=0.00091), Healthy plant(score=000063) without labelling the images

that we used as an input and this method will not be clear enough for us because the system is mainly used for farmers which means that in every image we will take will contain 10s of leaves and by labelling method every leaf will have a label according to its situation , not like the percentage method that is used for only an image with one leaf , mentioning that training images will be impossible in raspberry pi and this figure showing us the steps of the training and how much the system consuming resources while training ,for example the memory sometimes reaches 7 GB which is impossible to be handled by raspberry pi.

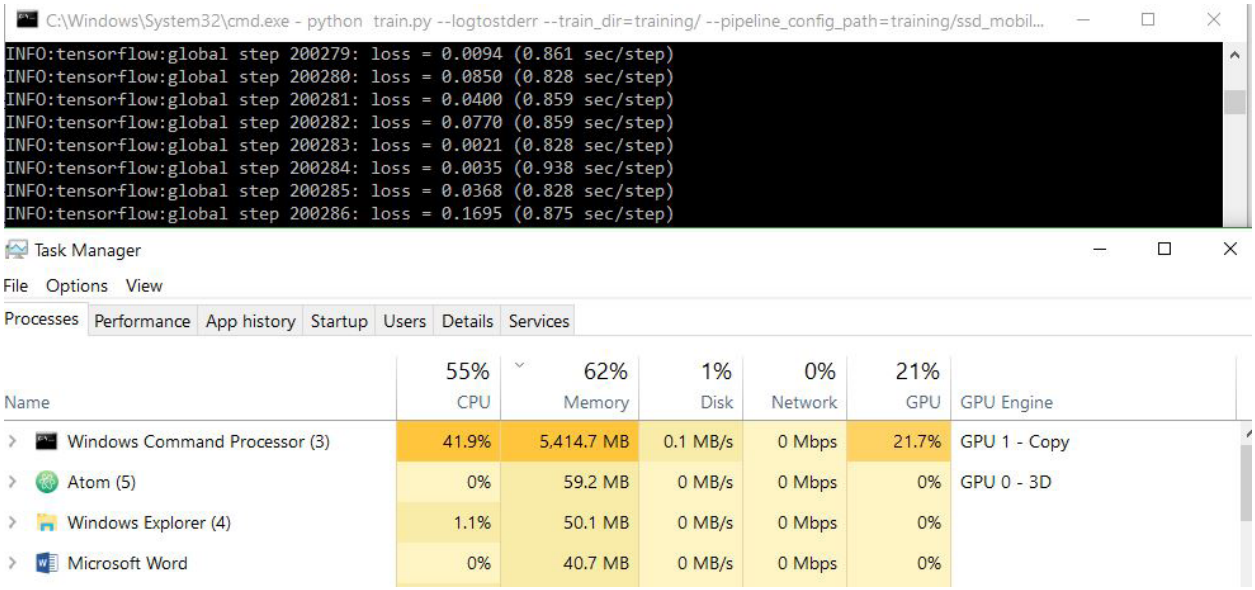

Figure 5.5 usage of laptop resources while Training the module

Chapter 6

Conclusion & Recommendations

6.1 Conclusion

After testing the results, we conclude that our system gives a good results when we test it on leaves images.But if we test it other types of images like cars, humans, etc... it may give us imprecise results.
In the future, We could add more classes of images to make the system more intelligence to have more accurate results.

6.2 Recommendations

1- In the case of a disease detection on the plant, it is possible to add an insecticide spraying device to ensure that the crop is not spoiled.

2- We can use a Drone (small remote controlled plane), so that it is linked with a high quality camera to check all of the harvest, in addition to a small container that containing insecticide.
Where a real-time video streaming checks the situation of the leaf and If the disease is detected, the insecticide is sprayed, then the drone will continue to another leaf to check it.

Figure 6.1 Drone with camera

References

[1] PMC, " Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification ", 2016 Jun 22.
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4934169/

[2] Foodvalleyupdate, " Using drones to detect crop diseases ", 2014 Dec 19.
http://www.foodvalleyupdate.com/news/using-drones-to-detect-crop-diseases/

[3] Sarika Datir K.J.College of Engineering and Management Research ,Pune(University of Pune) Sanjeev Wagh,Ph.D K.J.College of Engineering and Management Research ,Pune(University of Pune) , " Monitoring and Detection of Agricultural Disease using Wireless Sensor Network", India, 2014 Fe 4.

[4] Cosmosmagazine, " What is Deep Learning and how does it work?", 2017 Aug 24.
https://cosmosmagazine.com/technology/what-is-deep-learning-and-how-does-it-work/

[5] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, " Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks ", Microsoft Research, 2015 Jun 4.

[6] Dell , " Dell-Inspiron7567 ".
http://laptops.reviewed.com/content/dell-inspiron-7567-laptop-review.

[7] Wikipedia, " Arduino".
https://en.wikipedia.org/wiki/Arduino

[8] Futureelectronics, " Humidity and temperature sensor ".
http://www.futureelectronics.com/en/sensors/humidity-dew.aspx

[9]Wikipedia, " Servomotor".
https://en.wikipedia.org/wiki/Servomotor

[10] engineersgarage, 16x2-lcd
https://www.engineersgarage.com/electronic-components/16x2-lcd-module-datasheet

[11] wikipedia, TensorFlow
https://en.wikipedia.org/wiki/TensorFlow

[12] github, jupyter notebook

https://github.com/jupyter/notebook

[13] jupyter-notebook-beginner-guide, jupyter notebook

http://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html

[14] github , labelImg

https://github.com/tzutalin/labelImg

[15] jupyter-notebook-beginner-guide, jupyter notebook

http://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html

[16] dell, inspiron-15-7567-laptop

http://www.dell.com/en-us/shop/cty/pdp/spd/inspiron-15-7567-laptop