

PLANT DISEASE DETECTION AND GROWTH MONITORING USING IOT

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

KONKA ANUDEEP (Reg. No. 37110362)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI
SALAI, CHENNAI – 600 119**

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **KONKA ANUDEEP** (Reg. No. 37110362) who carried out the project entitled "**PLANT DISEASE DETECTION AND GROWTH MONITORING USING IOT**" under my supervision from **December 2020** to **March 2021**.

Internal Guide

Mrs. Ramya G. Franklin, M.E.,

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **KONKA ANUDEEP** hereby declare that the Project Report entitled “**PLANT DISEASE DETECTION AND GROWTH MONITORING USING IOT**” is done by me under the guidance of **Mrs. Ramya G. Franklin, M.E.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mrs. Ramya G. Franklin, M.E., Assistant Professor**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

As we all know that agriculture play a vital role for establishing agricultural society or country. Agriculture has been the career for people beyond a century however because of herbal aspect of climatic circumstance farmers are going through plenty of issues which might also additionally result in address critical issues in the society . So IOT can be a crucial key to help the farmers automate the whole process and deep leaning algorithms like CNN can be implemented for plant disease detection. This device will assist to govern the diverse environmental situations which include soil moisture, temperature and humidity for the growth of the plant and detect the plant disease. This paper presents the monitoring of the plant through Thinkspeak and disease detection through CNN.

TABLE OF CONTENTS

	ABSTRACT	v
	LIST OF FIGURES	viii
CHAPTER No.	TITLE	PAGE No.
1.	INTRODUCTION	1
	1.1 PLANT MONITORING	1
	1.2 DISEASE DETECTION	2
	1.3 AIM OF PROJECT	3
2.	LITERATURE SURVEY	4
	2.1 RELATED WORK	4
3.	SYSTEM DESIGN & METHODOLOGY	6
	3.1 ALGORITHM	6
	3.1.1 <i>CNN</i>	6
	3.2 DOMAIN	9
	3.2.1 <i>IOT</i>	9
	3.2.2 <i>DEEP LEARNING</i>	9
	3.3 IDE	10
	3.3.1 <i>ARDUINO</i>	10
	3.3.3 <i>ANACONDA</i>	12
4.	RESULTS AND DISCUSSION	13
	4.1 PROPOSED WORK	13
	4.1.1 <i>MODULES</i>	14
	4.2 THINKSPEAK	17
	4.3 RESULTS	17

6.	CONCLUSION AND FUTURE WORK	20
	REFERENCES	21
	APPENDIX	
	A. SOURCE CODE	22

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
1	ARCHITECTURE OF CNN	7
2	ARDUINO	10
3	PLANT MONITORING ARCHITECTURE	13
4	DETAILED REPORT OF PLANT MONITORING	17
5	PLANT DISEASE DETECTED USING DEEP LEARNING	18
6	HEALTHY PLANT	19

CHAPTER 1

INTRODUCTION

1.1 PLANT MONITORING

Agriculture, as the primary source of food grains and other raw materials, is known as the base of human life. It plays an important role in the growth of the national economy and provides individuals with various business opportunities. The development of the agrarian sector is critical for the country's economic well-being.. Unfortunately, many farmers continue to use conventional production methods, resulting in low harvest and daily item yields. However, wherever robotization has been implemented and people have been replaced by programmed machines, productivity has increased. In order to increase yield, it is essential to understand current science and advancement in the cultivating section.

The majority of the papers suggest the use of a remote sensor, which collects data from various types of sensors and sends it to a guideline server via remote display after a short time. The accumulated data provides knowledge about various environmental components, which is then used to screen the structure. Checking standard components isn't a sufficient or full answer to increase harvest yield. There are several segments that have an unfathomable effect on productivity. These factors include bug and nuisance attacks, which can be mitigated by spraying the harvest with enough bug spray and pesticides. When the reaper matures, he will be ambushed by wild animals and fowls. When yield is at the time of a social affair, there is also a chance of robbery.

Ranchers face issues away from the collected crop in the same way as they face issues with the collected crop. As a result, in order to provide answers to any such question, it is important to build a streamlined system that will manage all segments that affect profitability at each level, such as growth, reaping, and post-collection stockpiling. This paper then proposes a method that is useful for monitoring field data as well as managing field

assignments, allowing for flexibility. The paper focuses on using computerization and IoT advances to make farming more savvy. This paper has the ability to perform tasks such as weeding, showering, detecting soggy, feathered animal surprising, keeping wariness, etc. Moreover, it fuses splendid water framework with sharp control reliant on steady field data. Third, a clever distribution focus company that combines temperature regulation and stockroom moisture management. Both of these exercises will be operated by any sharp contraption or device connected to the Internet, and the errands will be carried out using sensors and a Wi-Fi module with a scaled down scale controller.

1.2 DISEASE DETECTION

The state of the plants is crucial to the farmers' ability to make a profit. In order to prevent disease from affecting plants, proper plant health monitoring is needed at various stages of plant development. The existence of pests and diseases has a major effect on crop cultivation estimation and crop yield. The current method is based on naked eye observation, which is a time-consuming process. To diagnose plant disease at an early stage, automated detection of plant disease can be used. Farmers also used a number of disease control techniques on a daily basis to eliminate plant diseases. We're using IOT and image processing to build a leaf disease detection system. This paper discusses the role of internet of things (IOT) technology in agricultural disease and introduces the idea of IOT technology as a means of perceiving knowledge. IoT-enabled smart farming With the rise of IoT, there is hope for developing a digital infrastructure for agriculture that will assist farmers in making educated decisions regarding their farms and proactively resolving certain undesirable situations. In the farming sector, technology improves crop production. The state of the plants is crucial to the farmers' ability to make a profit. In order to prevent disease from affecting plants, proper plant health monitoring is needed at various stages of plant development. Visual detection of plant diseases is a more challenging task. Using the automatic technique, on the

other hand, would require less effort. Image analysis is used to predict disease leaf in plants. According to a study, the most common forms of plant disease are brown and yellow spots, early and late scorch, and fungal, infectious, and bacterial diseases. The Median filter is used to smooth out images and minimise noise. In terms of image de-noising, median filtering is very efficient. This form of noise reduction is a common pre-processing phase used to enhance the results of subsequent processing. In general, the colour and texture of plant leaves are distinguishing characteristics that are used to determine whether a leaf is diseased or healthy. It was accomplished by connecting various sensors to the NodeMcu module. Various sensors are used in agriculture to calculate parameters such as soil moisture, temperature, and humidity, fertiliser, and contribute to the farm's productivity. Farmers and agricultural experts must be able to identify diseases in plants at all times. The proposed system's main goal is to use IoT to detect plant diseases. The disease begins on the plant leaves in the majority of cases. As a consequence, we have considered detection of plant disease on leaves in the proposed work. Temperature, humidity, and colour changes may be used to differentiate between regular and damaged plant leaves.

1.3 AIM OF THE PROJECT

The whole proposal's intention is to aid farmers or the people who are into gardening but couldn't monitor them at regular intervals, ex-watering the plants, controlling the temperature, humidity as some of the crops require closed monitoring sheds to grow.so to achieve this we require two domains iot and cnn.

CHAPTER 2

LITERATURE SURVEY

2.1 RELATED WORK

Arti N. Rathod et.al., provides a variety of techniques for detecting leaf sickness using image processing. The techniques under investigation are aimed at improving throughput while ignoring the subjectivity of human experts when it comes to detecting leaf disease. A technique for enhancing the quality of an image is digital image processing. Automatic symptom identification is helpful in improving agricultural items.

The Raspberry Pi, Arduino micro - controller, Xbee modules, and relays are among the moderate, close to the edge components proposed by Nikhil Agarwal et al. for a home security system. Since additive effects are used in the basic value, the method is adaptive and robust. The Raspberry Pi processes the consumer's commands using the Python programming language. The Raspberry Pi sends on-off instructions to Arduino microcontrollers via the ZigBee protocol. This serves as the Raspberry Pi's and end-devices' contact bridge. Raspberry Pi instructs Arduino microcontrollers to turn on and off. The foundation for connectivity between both the raspberry pi and end devices is indeed the star ZigBee topology.

Leaf image identification through some kind of server-based fully monitoring and control system, pressure and temperature sensing, and relative humidity sensing are just a few of the features offered by Apeksha Thorat et

al. It measures moisture temperature and humidity using sensor networks rather than a manual scan. A Raspberry PI controller has been used in a variety of locations around farms(RPI) to power these sensors. A camera and RPI can be used to detect leaf disease. The credibility of a plant, such as leaf sickness, as well as numerous external conditions influencing the plant, such as moisture content, temperatures, and precipitation, are simultaneously sent to the farm workers via RPI via WIFI Server.

K.Lakshmi et al. have built a pesticide control system that removes the need for human intervention in crop cultivation. It prevents the negative effects of overuse of insecticides, which result in reduced crop cultivation. This version will achieve 83 excellent results for most types of vegetation, as well as the target of regulated irrigation techniques.

Melike Sardogan et al. developed a CNN model for image retrieval automatically. In the study of leaf image, colour records are commonly used. The filtering is applied to different positive significant correlation on RGB components from their model. The output characteristic vector of the convolution component was fed into the Lvq to train the network. Experiments show that the suggested approach correctly distinguishes four distinct types of tomato leaf diseases.

Mitul Sheth et.al., A deadline for retrieving data and syncing it with the internet via WIFI was discussed. When the water level falls below the set point, the customer is informed. Their paper shows how the NodeMCU can observe wireless circuit diagrams and view the results with the Blynk App. When it senses low wetness and temperature, a message is sent to NodeMCU and Blynk App, and the motor in the home or farm is automatically started.

In the IOT age, Samuel Olawepo et al. show the benefits of intelligent field optimization for optimal crop production and food safety. The whole paper,

which would also be linked to internet and has live event stats presented via communications or a custom web app, recommends the activities needed for intelligent grass integration using current infrastructure, such as integrated circuits and detectors, which is linked to a data cloud.

CHAPTER 3

SYSTEM DESIGN AND METHODOLOGY

3.1 ALGORITHM

3.1.1 CNN

Model preprocessing and coaching (CNN): Preprocessing includes image reshaping, resizing, and converting the dataset to an array form. The check picture is also subjected to a similar method. A dataset of about nineteen different plant species is collected, and any image from that dataset can be used as a check image for the computer code. The train dataset is used to train the CNN model so that it can assess the check image as well as the disease. Dense, Dropout, Activation, Flatten, Convolution2D, and MaxPooling2D are the various layers that CNN has. If the plant species is present within the dataset and the model is successfully educated, the computer code will assess the disease. When coaching and preprocessing, the check image and trained model are compared in order to predict the disease.

CNN Model steps-

Conv2D-This is the layer that flexes the image into several images and activates them. MaxPooling2D- It's popular to gamma hydroxybutyrate pool the value from a given size matrix, and the following two layers use the

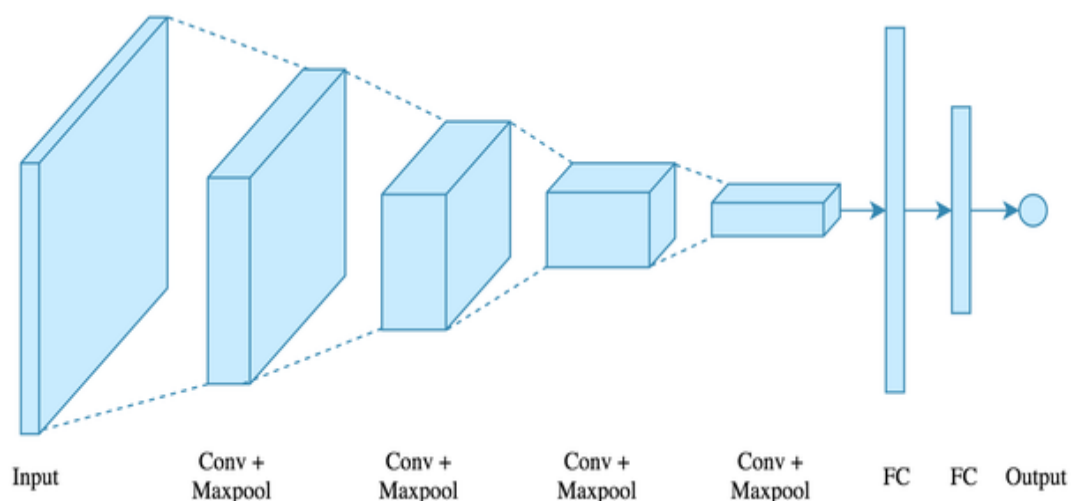
same form.

Flatten- It's common practise to flatten the scale when convolving a frame.

Dense- This is the hidden layer, and it's popular to make it a completely linked model.

Dropout- Avoiding overfitting on the dataset is standard practise, and dense means that the output layer only contains one nerve cell that tries to match the class image. Image knowledge generator- It resizes the image, applies shear to specific areas, zooms it in, and flips it horizontally. All possible image orientations are included in this Image Information Generator. In the Training Process-Train datagen, the flow from directory feature is used to prepare information from the train dataset directory. target size determines the image's target size. The test datagen flow from directory is used to organise the model's test data, and each one is equal to higher than fit generator; other variables used include steps per epochs, which tells us how many times the model will execute for the coaching information. Epochs: The number of forward and backward passes the model has been trained. Validation procedure: Validation data is used to feed validation/test information into the model. Validation steps reflect the number of validation/test samples.

Architecture of CNN-



An Associate in Nursing input nodes, convolutional nodes, fully-connected

nodes, and an Associate in Nursing output nodes are all part of a typical convolutional neural network architecture. With a few tweaks, CNN is based on the LeNet definition. It has a half-dozen layers without accounting for input and output. The image must be processed before ever being loaded further into input nodes since it has well before, defined measurements. Forty-eight X forty-eight pixel normalised grey scale images from a given dataset were used for coaching, validation, and testing Laptop computer digital camera photos are used for analysis, and the victim's face is detected and cropped during the process. In OpenCV, the Haar-cascade Classifier is used.

Convolution and Pooling (ConvPool) Layers-

Most every batches consists N images, and the CNN filter weights for those batches are adjusted. A four-dimensional image batch with N x Colour-Channel x dimension x height is fed into any convolution sheet. A four-dimensional convolutional function map or filter is also a four-dimensional convolutional function map or filter (Number of feature maps in, range of feature maps out, filter dimension, filter height). Four-dimensional convolutions are measured between image batches and each convolution layer includes maps. The only variables that change when using convolution are the image dimension and height.

Previous image dimension – filter dimension + one = new image dimension

Previous image height – filter height + one = new image height.

For structural material contraction, downward sampling/subsampling is performed every after-convolution sheet. Pooling is the term for this practise. Pooling methods such as Goop pooling and average pooling are well-known. Goop pooling is done once convolution is completed in this project. The pool size is (2x2), which divides the image into a grid of 2x2 blocks that take up the majority of the four pixels. When you combine height and dimension, only height and dimension are affected. Two convolution layers and a pooling layer may be used in the architecture. The input image batch's convolution layer size is initially Nx1x48x48. Every image height and dimension has forty-eight components, and the image batch size is N. The

colour channel range is one. The convolution with feature map of $1 \times 20 \times 5 \times 5$ results image batch has a batch size of $N \times 20 \times 44 \times 44$. When convolution pooling with a pool size of 2×2 is complete, an image batch of size $N \times 20 \times 22 \times 22$ is generated, which is then followed by a 2d convolution layer with a characteristic map of $20 \times 20 \times 5 \times 5 \times 20 \times 5 \times 5$, producing an image batch of size $N \times 20 \times 18 \times 18$. This is often followed by a pooling layer with a pool size of 2×2 , completing the $N \times 20 \times 9 \times 9$ image batch size. Fully Connected Layer: This layer is impressed by the way neurons send signals across the brain. It transforms a large number of input options using layers connected by trainable weights. Two hidden layers with sizes of 500- and 300-units square are used in the fully connected layer. To train the weights of such layers, forward propagation of coaching knowledge and backward propagation of its errors are used. Back propagation starts with measuring the difference between prediction and true value, then calculating the burden adjustment for each board. By standardising hyper-parameters like learning rate and network density, we can monitor the coaching speed as well as the design performance. This sheet's hyper-parameters include learning rate, momentum, regularisation parameter, and decay. The second pooling layer's output is $N \times 20 \times 9 \times 9$, and the fully-connected layer's initial hidden layer's input is $N \times 500$. As a result, the pooling layer's output is two-dimensional and $N \times 1620$ in size, and it is fed to the initial hidden layer. The first hidden layer's output is passed on to the second hidden layer. The second hidden layer's output, which is $N \times 300$ in size, is fed to an output layer with an appropriate range of facial feature categories. Output layer: The current hidden layer's sensor is connected to an output nodes with seven different categories, and the possibilities for each one of the seven categories are used to generate output. The actual class is the one with best hope of achieving success.

3.2 DOMAIN

3.2.1 IOT

IOT, relates to the trillions of serial interfaces worldwide that are hooked up

to the internet and store as well as exchange information. Thanks to the growing popularity of wireless devices and the invention of mega integrated circuits, it's now capable of transforming everything, from a pill to a plane, into a part of the IOT. Bringing all of these disparate objects together and connected detectors to them gives machines that would otherwise be dumb a degree of digital intelligence, allowing them to exchange real-time data without requiring a person. The Internet of Things (IoT) connects the digital and physical environment to make our surroundings smarter and more adaptive. Any problem can be identified by its embedded computing unit, but it can also work with the current Internet infrastructure. Professionals expect that by the year 2021, the internet of things would have grown to about 50 billion devices.

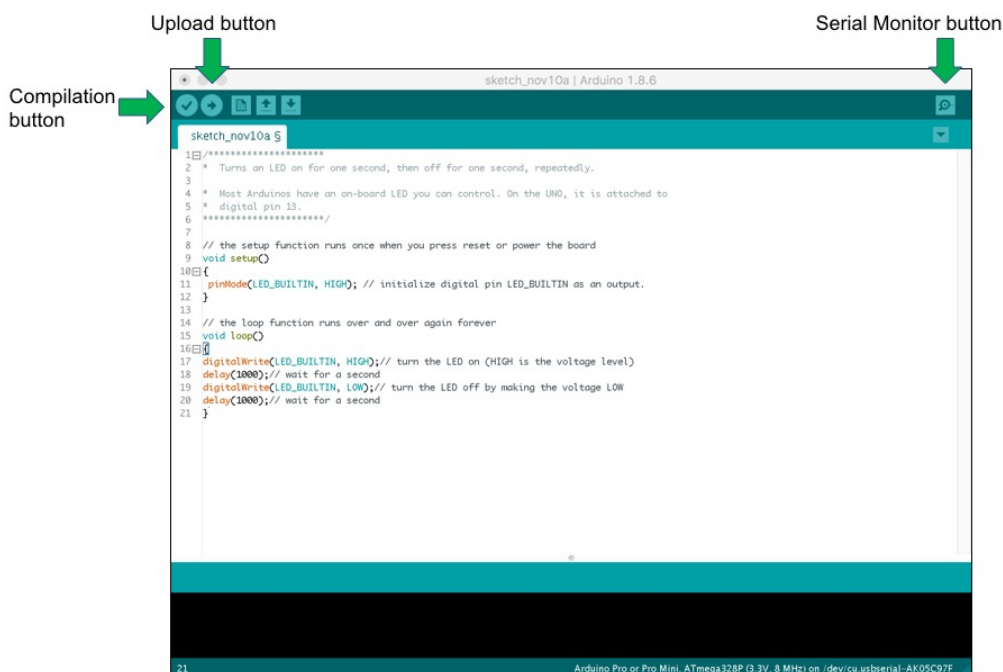
3.2.2 DEEP LEARNING

DL is a form of ML that relies solely on artificial neural networks for its support. Since neural networks are designed to imitate the human brain, deep learning is also a kind of brain mimic. It's turned off today because we didn't have enough process capacity and expertise earlier. Neurons are a proper definition of DL. DL is a form of explicit machine learning that teaches the computer to see the world as a nested hierarchy of concepts with each thought outlined in terms of less abstract ideas, and they are computed in terms of less abstract representations. This can be a picture of a personal nerve cell in the brain, which has roughly a hundred billion nerve cells at any given time, and each neuron is connected to thousands of its neighbours. The question is how it manages to replicate these neurons in a laptop. As a result, everywhere there are nodes or neurons, it generates a man-made structure known as a man-made neural internet. It has several neurons for input value and a few for output value, with variant neurons intertwined within the hidden layer in between.

3.3 IDE

3.3.1 ARDUINO

To write computer code and upload it to the physical board, the Arduino IDE (Integrated Development Environment) is used. The Arduino IDE is extremely simple, which is probably one of the reasons Arduino has become so popular. We may confidently claim that compatibility with the Arduino IDE is now one of the most important criteria for a new microcontroller board. Many useful features have been introduced to the Arduino IDE over the years, and you can now handle third-party libraries and boards from the IDE while retaining the board's simplicity. The Arduino IDE's main window is shown below, along with the simple Blink illustration.



3.3.2 ANACONDA

Anaconda is a Python and R programming language distribution aimed at simplifying package management and deployment in scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, and so on). Data-science packages for Windows, Linux, and macOS are included in the distribution Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012, develops and maintains it. Anaconda Distribution or Anaconda Individual Edition are other Anaconda, Inc. products, while Anaconda Team Edition and Anaconda Business Edition, all of which are not free, are other Anaconda, Inc. products.

Conda, Anaconda's package management framework, maintains track of package versions. This package manager was spun out as a separate open-source package because it turned out to be useful for more than just Python. Miniconda is a lightweight, bootstrapped version of Anaconda that only includes conda, Python, and the packages they rely on, as well as a few other packages.

Over 250 packages are installed by default in the Anaconda distribution, and over 7,500 additional open-source packages, as well as the conda package and virtual environment manager, can be installed from PyPI. Anaconda Navigator, a graphical user interface, is used as an alternative to the command line interface (CLI).

Conda and the pip package manager vary greatly in how package dependencies are handled, which is a major challenge for Python data science and the reason for conda's existence.

When pip installs a package, it also installs any dependent Python packages without testing to see if they clash with other packages already installed. [requires citation] Regardless of the state of the current installation, it can instal a kit and all of its dependencies. As a result, a user who previously had a working installation of Google Tensorflow can discover that it no longer works after using pip to instal a different package that needs a

different version of the based numpy library than Tensorflow. In some cases, the package may appear to work but produce different results in detail.

Conda, on the other hand, examines the current environment, including all that is currently installed, and, along with any version limitations defined (for example, the user may want Tensorflow version 2,0 or higher), determines how to instal a compatible set of dependencies, and issues an alert if this is not possible.

The conda update command can be used to instal open source packages from the Anaconda registry, Anaconda Cloud (anaconda.org), or the user's own private repository or mirror. Anaconda, Inc. compiles and instals the packages in the Anaconda repository and distributes binaries for Windows 32/64 bit, Linux 64 bit, and MacOS 64 bit. Pip can instal anything available on PyPI into a conda environment, and conda will keep track of what it has installed and what pip has installed..

The conda build command can be used to create custom packages, which can then be shared with others by uploading them to Anaconda Cloud, PyPI, or other repositories.

Anaconda2 comes with Python 2.7 by default, while Anaconda3 comes with Python 3.7. However, any version of Python bundled with conda can be used to build new environments.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 PROPOSED WORK

In this proposed system, I've included an ESP8266 microcontroller, which serves as the system's brain because it stores all of the system's software instructions. For plant growth, we use a dht11 sensor to record the surrounding temperature and humidity. For the plants, the LDR sensor detects the presence of sunlight. In the absence of sunlight, the plants' LEDs automatically switch on. This detector is used to determine if the soil is wet or dry. The pump motor automatically turns on when the inserted medium is dry, to supply water to the plants. All the information is monitored using IOT and also displayed in the LCD [fig 3]. This data is sent to Thingspeak cloud where we can easily create a channel, by entering the channel id and API key details we can directly monitor the temperature, humidity, moisture and light for every 120 seconds. Using CNN, we train the datasets in python for 19 different type of plants and their diseases and can identify the diseased plant by image processing.

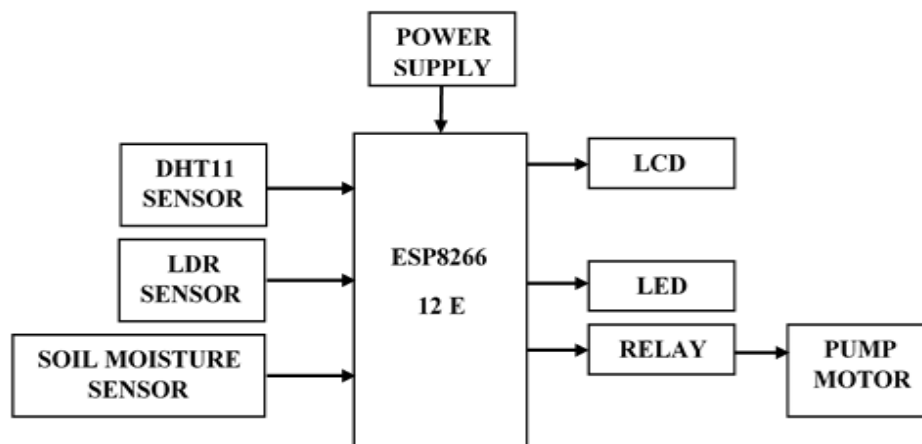


Fig 3. Plant monitoring Architecture

4.1.1 MODULES

The modules we require are plant monitoring part:

DHT11 SENSOR

LDR SENSOR

SOIL MOISTURE SENSOR

LCD

NODEMCU ESP8266

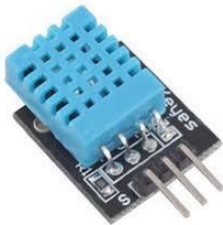
LED

RELAY

PUMP MOTOR

DHT11 SENSOR-

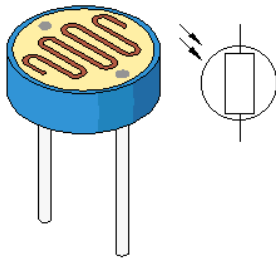
The DHT11 Temperature measurement Device consists a soil moisture sensor complex with a standardized transistor output. It uses an exclusive interactive signal accounting equation and weather moisture measuring techniques to improve good durability and long-term sustainability.



LDR SENSOR-

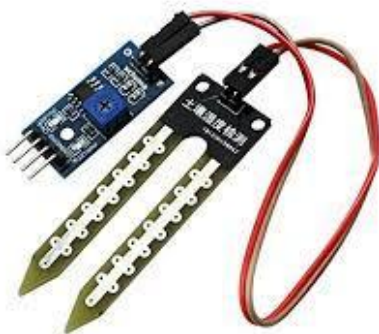
A photoresistor or a cadmium sulphide (CdS) cell is another term for a Light

Based Resistor (LDR). A photoconductor is another name for it. It's essentially a photocell that operates on the photoconductivity theory. The passive part is essentially a resistor whose resistance decreases as the light intensity decreases. This optoelectronic device is most commonly used in light-varying sensor systems.



SOIL MOISTURE SENSOR-

This detector can be used to check the moisture of soil. When the soil is dry, the module output is high, and when the soil is wet, the output is low. This sensor can also be used to mechanically water flower plants or any other plants that need automatic watering.



NODEMCU-

Ai-thinker Team created the ESP-12E Wi-Fi module. Tensilica I106 is encapsulated in the center processor esp8266 in smaller module sizes. With

the sixteen bit brief mode clock velocity aid eighty MHz one hundred sixty MHz helps the RTOS, integrated Wi-Fi and on-board antenna, It features an industry-leading incredibly less power 32-bit MCU microcontroller.



RELAY-

This is a device that opens or closes contacts in order to monitor the action of other electric controls. It detects an insupportable or undesirable situation in a designated location and sends commands to the circuit breaker to sever the affected region, resulting in a more efficient working module.



PUMP MOTOR-

A pump motor is a DC motor that serves as a medium for moving fluids. It operates on the principle that when a cutting-edge wearing conductor is placed in a magnetic field, it produces torque and has a tendency to travel, which is known as motoring motion. When the cutting-edge path within the cord is reversed, the rotation path is reversed as well. Pumps operate by using a few mechanisms (usually reciprocating or rotary) and using energy to perform mechanical work. Pumps use a variety of power sources,

including guidance service and energy, and come in a variety of sizes, ranging from microscopic pumps for research applications to massive business pumps.



4.2 ThinkSpeak

ThingSpeak is a cloud-based IoT analytics application that lets you compile, visualize, and analyze live data streams. ThingSpeak offers real-time visualisations of data submitted to the network by your computers. With the ability to run MATLAB code in ThingSpeak, you can analyse and process data as it arrives in real time. ThingSpeak is often used for IoT device prototyping and proof of concept that require analytics.

4.3 RESULTS

The section put forth a model for monitoring and detection of plant disease, below figure[fig 4] shows the values displayed on thinkspeak website after acquiring through the sensors.



Fig 4 Detailed report of plant monitoring

Fig 4 above shows the values of temperature, humidity, moisture, light values at regular intervals of time.

Fig 5 determines a cherry plant diseased by powdery mildew and required solution is given for curing it by applying Rex lime sulphur to reduce next season's powdery mildew potential.

plant leaf disease clasification

Predicting

Agriculture

PLANT LEAF DISEASE CLASIFICATION USING DEEP CNN

Browse and Predict Leaf Disece(click)

All The Best For Agriculture(click)

PREDICTION

Cherry_(including_sour)_Powdery_mildew

SOLUTION

Rex Lime Sulfur Apply to reduce next season's powdery mildew potential. Apply within 30 days after harvest. Optimum timing is 7 to 10 days after harvest. Full wetting of the foliage is required.



Fig 5. Plant disease detected using deep learning.

Below Fig 6 determines a healthy cherry leaf without being effected by any disease.

plant leaf disease clasification

PLANT LEAF DISEASE CLASIFICATION USING DEEP CNN

Predicting

Browse and Predict Leaf Disece(click)

Agriculture

All The Best For Agriculture(click)

PREDICTION

Cherry_(including_sour)_Powdery_mildew

SOLUTION

Rex Lime Sulfur Apply to reduce next season's powdery mildew potential. Apply within 30 days after harvest. Optimum timing is 7 to 10 days after harvest. Full wetting of the foliage is required.

PREDICTION

Pepper_bell_healthy

SOLUTION

NO PROBLEM...!



Fig 6 healthy plant

CHAPTER 5

CONCLUSION AND FUTURE WORK

CONCLUSION:

The plant disease detection and growth monitoring using iot has been verified to satisfactorily work by displaying the data in thinkspeak using the ESP 8266 microcontroller the disease detection of the plant has been successfully identified using CNN algorithm.

FUTURE WORK:

Application developers can work together with the monitoring part and the

disease detection part together to develop an app which could eventually determine the values of the environmental conditions, plant Realtime growth from sapling to tree stage and the diseases effecting the plants from growing healthily.

REFERENCES

- [1]. Apeksha Thorat; Sangeeta Kumari; Nandakishor D. Valakunde.“An IoT based smart solution for leaf disease detection”, 2017 International Conference on Big Data, IoT and Data Science (BID),IEEE,2017.
- [2]. Arti N. Rathod, Bhavesh Tanawal, Vatsal Shah,“ Image Processing Techniques for Detection of Leaf Disease”, International Journal of Advanced Research in Computer Science and Software Engineering, Vol 3, Issue no 11,pp397-399,2013.
- [3]. Lakshmi.K, Gayathri.S,“ Implementation of IoT with Image processing in plant growth monitoring system”, Journal of Scientific and Innovative Research,pp 80-83,2017.
- [4]. Mitul Sheth,Pinal Rupani.“ Smart Gardening Automation using IoT With BLYNK App”,3rd International Conference on Trends in Electronics and Informatics (ICOEI),IEEE,2019.
- [5].Melike Sardogan; Adem Tuncer; Yunus Ozen.“ Plant Leaf Disease Detection and Classification Based on CNN with LVQ Algorithm”, 3rd International Conference on Computer Science and Engineering (UBMK),IEEE,2018.
- [6].Nikhil Agrawal; Smita Singhal,“Smart drip irrigation system using raspberry pi and arduino”, nternational Conference on Computing, Communication & Automation,IEEE,2015.
- [7].Samuel Olawepo; Ayodele Adebisi; Marion Adebisi; Olatunji Okesola,“ An Overview Of Smart Garden Automation”, International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS),IEEE.2020.

APPENDIX

A. SOURCE CODE

Arduino code:

```
#include"dht11.h"
DHT dht;
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ThingSpeak.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

const char* ssid = "Anudeep";
const char* password = "89609408";
WiFiClient client;
unsigned long myChannelNumber = 568876;
const char * myWriteAPIKey = "O7T32VB4D50PUDUJ";
uint8_t temper, humidity, k = 0, l = 0;

/*****/
void setup()
{
  lcd.begin();
  lcd.backlight();
  Serial.begin(9600);
  pinMode(D3, INPUT);
  pinMode(D7, INPUT);
  pinMode(D5, OUTPUT);
  pinMode(D6, OUTPUT);
  digitalWrite(D5, HIGH);
  digitalWrite(D6, LOW);
  delay(10);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("CONNECTING..");
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
```

```

    Serial.print(".");
}
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("CONNECTED..");
Serial.println("");
Serial.println("WiFi connected");
Serial.println(WiFi.localIP());
ThingSpeak.begin(client);
delay(2000);
}
/*****/
void loop()
{
    dht.dht_read(D0);
    temper=dht.temperature;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("T:");
    lcd.print(dht.temperature);
    lcd.setCursor(8, 0);
    lcd.print("H:");
    lcd.print(dht.RH);

    lcd.setCursor(0, 1);
    lcd.print("M:");
    lcd.print(digitalRead(D3));

    lcd.setCursor(8, 1);
    lcd.print("L:");
    lcd.print(digitalRead(D7));
    if(digitalRead(D7)==1)
    {
        digitalWrite(D6,HIGH);
    }
    else
    {
        digitalWrite(D6,LOW);
    }
    if (digitalRead(D3) == 1)
    {
        digitalWrite(D5, LOW);
    }
    else
    {
        digitalWrite(D5, HIGH);
    }
    ThingSpeak.writeField(myChannelNumber, 1, dht.temperature, myWriteAPIKey);
}

```

```

delay(15000);
ThingSpeak.writeField(myChannelNumber, 2, dht.RH, myWriteAPIKey);
delay(15000);
ThingSpeak.writeField(myChannelNumber, 3, digitalRead(D3), myWriteAPIKey);
delay(15000);
ThingSpeak.writeField(myChannelNumber, 4, digitalRead(D7), myWriteAPIKey);
delay(15000);
}

```

Python code:

```

import warnings
warnings.filterwarnings('ignore')
from tkinter import *
import numpy as np
import pandas as pd
from tkinter.filedialog import askopenfile
import tkinter as tk
import tkinter.ttk as ttk
import os
from tkinter import Message ,Text
import shutil
from PIL import Image, ImageTk
import pandas as pd
import datetime
import time
import tkinter.font as font
import imageio

import numpy as np
from PIL import Image
import os
import tkinter as tk
import tkinter.ttk as ttk
import os
import numpy as np
import numpy as np
from PIL import Image
import os
import imageio

#from moviepy.video.io.ffmpeg_tools import ffmpeg_extract_subclip
import tkinter

root = Tk()
root.geometry("1100x600")

```

```

root.title("plant leaf disease clasification ")

canvas = Canvas(root,width=100,height=100)
canvas.pack()
photo = PhotoImage(file ='demo1.png')
canvas.create_image(0,0,image=photo,anchor=NW)


def Give_the_details():
    from tkinter import filedialog


def Train_the_model():
    from tkinter import filedialog
    slabel = tkinter.Label(root, text="PREDICTION")

    slabel.pack()

    filename = filedialog.askopenfilename(initialdir = "/",title = "Select file",filetypes =
(("jpeg files", "*.jpg"),("all files", "*..*")))

    load = Image.open(filename)
    render = ImageTk.PhotoImage(load)
    img = Label(image=render)
    img.image = render
    img.place(x=0, y=300)
    import numpy as np
    import tensorflow as tf
    from tensorflow import keras
    import h5py
    models=keras.models.load_model('plantoutput3.h5')
    from tensorflow.keras.preprocessing import image
    test_image=image.load_img(filename,target_size=(225,225))
    test_image=image.img_to_array(test_image)
    test_image=np.expand_dims(test_image,axis=0)
    result=models.predict(test_image)
    prediction = result[0]
    prediction=list(prediction)

classes=['Apple___Apple_scab','Apple___healthy','Cherry_(including_sour)___Powd
ery_mildew','Cherry_(including_sour)___healthy','Corn_(maize)___Cercospora_leaf
spot
Gray_leaf_spot','Corn_(maize)___healthy','Grape___Black_rot','Grape___healthy','O

```

```

range__Haunglongbing_(Citrus_greening)','Peach__Bacterial_spot','Peach__hea
lthy','Pepper,_bell__Bacterial_spot','Pepper,_bell__healthy','Potato__Early_blight'
,'Potato__healthy','Strawberry__Leaf_scorch','Strawberry__healthy','Tomato__E
arly_blight','Tomato__healthy']
output=zip(classes,prediction)
output=dict(output)
if output['Apple__Apple_scab']==1.0 :
    print("Apple_scab")
    label = tkinter.Label(root, text="Apple_scab")

    label.pack()

elif output['Apple__healthy']==1.0:
    print("Apple_healthy")
    label = tkinter.Label(root, text="Apple_healthy")
    label.pack()

elif output['Cherry_(including_sour)__Powdery_mildew']==1.0:
    print("Cherry_(including_sour)_Powdery_mildew")
    label = tkinter.Label(root, text="Cherry_(including_sour)_Powdery_mildew")
    label.pack()
elif output['Cherry_(including_sour)__healthy']==1.0:
    print('Cherry_(including_sour)__healthy')
    label = tkinter.Label(root, text="Cherry_(including_sour)__healthy")
    label.pack()
elif output['Corn_(maize)__Cercospora_leaf_spot Gray_leaf_spot']==1.0:
    print('Corn_(maize)_Cercospora_leaf_spot Gray_leaf_spot')
    label = tkinter.Label(root, text="Corn_(maize)_Cercospora_leaf_spot
Gray_leaf_spot")
    label.pack()

elif output['Corn_(maize)__healthy']==1.0:
    print("Corn_(maize)_healthy")
    label = tkinter.Label(root, text="Corn_(maize)_healthy")
    label.pack()
elif output['Grape__Black_rot']==1.0:
    print("Grape_Black_rot")
    label = tkinter.Label(root, text="Grape_Black_rot")
    label.pack()

elif output['Grape__healthy']==1.0:
    print("Grape_healthy")
    label = tkinter.Label(root, text="Grape_healthy")
    label.pack()
elif output['Orange__Haunglongbing_(Citrus_greening)']==1.0:
    print("Orange_Haunglongbing_(Citrus_greening)")
    label = tkinter.Label(root, text="Orange_Haunglongbing_(Citrus_greening)")
    label.pack()

```

```

elif output['Peach___Bacterial_spot']==1.0:
    print("Peach_Bacterial_spot")
    slabel = tkinter.Label(root, text="Peach_Bacterial_spot")
    slabel.pack()
elif output['Peach___healthy']==1.0:
    print("Peach_healthy")
    slabel = tkinter.Label(root, text="Peach_healthy")
    slabel.pack()
elif output['Pepper,_bell___Bacterial_spot']==1.0:
    print("Pepper_bell_Bacterial_spot")
    slabel = tkinter.Label(root, text="Pepper_bell_Bacterial_spot")
    slabel.pack()
elif output['Pepper,_bell___healthy']==1.0:
    print("Pepper_bell_healthy")
    slabel = tkinter.Label(root, text="Pepper_bell_healthy")
    slabel.pack()
elif output['Potato___Early_blight']==1.0:
    print('Potato_Early_blight')
    slabel = tkinter.Label(root, text="Potato_Early_blight")
    slabel.pack()

elif output['Potato___healthy']==1.0:
    print('Potato_healthy')
    slabel = tkinter.Label(root, text="Potato_healthy")
    slabel.pack()

elif output['Strawberry___Leaf_scorch']==1.0:
    print("Strawberry_Leaf_scorch")
    slabel = tkinter.Label(root, text="Strawberry_Leaf_scorch")
    slabel.pack()
elif output['Strawberry___healthy']==1.0:
    print("Strawberry_healthy")
    slabel = tkinter.Label(root, text="Strawberry_healthy")
    slabel.pack()

elif output['Tomato___Early_blight']==1.0:
    print("Tomato_Early_blight")
    slabel = tkinter.Label(root, text="Tomato_Early_blight")
    slabel.pack()

elif output['Tomato___healthy']==1.0:
    print("Tomato_healthy")
    slabel = tkinter.Label(root, text="Tomato_healthy")
    slabel.pack()

slabel = tkinter.Label(root, text="SOLUTION")

slabel.pack()

```

```

if output['Apple___Apple_scab']==1.0 :
    slabel = tkinter.Label(root,justify=RIGHT, text="Apply as a soil drench or
foliar spray (3-4 tsp/ gallon of water) to prevent and attack fungal problems.")
    slabel.pack()

elif output['Apple___healthy']==1.0:
    slabel = tkinter.Label(root,justify=RIGHT, text="NO PROBLEM...!")
    slabel.pack()

elif output['Cherry_(including_sour)___Powdery_mildew']==1.0:
    slabel = tkinter.Label(root,justify=RIGHT, text="Rex Lime Sulfur Apply to reduce
next season's powdery mildew potential. Apply within 30 days after harvest.
Optimum timing is 7 to 10 days after harvest. Full wetting of the foliage is required.")
    slabel.pack()
elif output['Cherry_(including_sour)___healthy']==1.0:
    slabel = tkinter.Label(root,justify=RIGHT, text="NO PROBLEM...!")
    slabel.pack()
elif output['Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot']==1.0:
    slabel = tkinter.Label(root,justify=LEFT,
text="Corn_(maize)_Cercospora_leaf_spot Gray_leaf_spot")
    slabel.pack()

elif output['Corn_(maize)___healthy']==1.0:
    slabel = tkinter.Label(root, text="NO PROBLEM...!")
    slabel.pack()
elif output['Grape___Black_rot']==1.0:
    slabel = tkinter.Label(root, text="Grape_Black_rot")
    slabel.pack()

elif output['Grape___healthy']==1.0:
    slabel = tkinter.Label(root, text="NO PROBLEM...!")
    slabel.pack()
elif output['Orange___Haunglongbing_(Citrus_greening)']==1.0:
    slabel = tkinter.Label(root, text="Orange_Haunglongbing_(Citrus_greening)")
    slabel.pack()
elif output['Peach___Bacterial_spot']==1.0:
    slabel = tkinter.Label(root, text="Peach_Bacterial_spot")
    slabel.pack()
elif output['Peach___healthy']==1.0:
    slabel = tkinter.Label(root, text="NO PROBLEM...!")
    slabel.pack()
elif output['Pepper,_bell___Bacterial_spot']==1.0:
    slabel = tkinter.Label(root, text="Pepper_bell_Bacterial_spot")
    slabel.pack()
elif output['Pepper,_bell___healthy']==1.0:
    slabel = tkinter.Label(root, text="NO PROBLEM...!")

```

```

        slabel.pack()
    elif output['Potato___Early_blight']==1.0:
        slabel = tkinter.Label(root, text="Potato_Early_blight")
        slabel.pack()

    elif output['Potato___healthy']==1.0:
        slabel = tkinter.Label(root, text="NO PROBLEM...!")
        slabel.pack()

    elif output['Strawberry___Leaf_scorch']==1.0:
        slabel = tkinter.Label(root, text="Remove the older and infected leaves from
runner plants before setting.Take care in spacing runner plants in matted-row
culture.Plant in light, well-drained soil in a location exposed to all-day sun and good
air circulation.Control weeds in the planting. Weeds reduce air circulation and
increase drying time for leaves. (Leaves stay wet longer in weedy
plantings.)Removing infected leaves after harvest (during renovation) is helpful in
reducing inoculum and controlling all the leaf diseases")
        slabel.pack()
    elif output['Strawberry___healthy']==1.0:
        slabel = tkinter.Label(root, text="NO PROBLEM...!")
        slabel.pack()

    elif output['Tomato___Early_blight']==1.0:
        slabel = tkinter.Label(root, text="Chlorothalonil is a synthetic fungicide that
controls both early blight and late blight. It can be used as soon as tomato plants are
subjected to humid or rainy conditions that can cause blight. Premixed and
concentrated chlorothalonil are available, and spraying it every seven to 10 days is
ideal for prevention of blight. It can be used at any time before harvesting
tomatoes.")
        slabel.pack()

        slabel.pack()
    elif output['Tomato___healthy']==1.0:
        slabel = tkinter.Label(root, text="Tomato_healthy")
        slabel.pack()

def getImages():
    # Path for face image database
    Train_the_mode.set('0')

```

```
def Finder():
```

Output.insert(END, 'In India, the agriculture and the allied sector contributes less than 17% to its US\$ 3 trillion economy today (Sunder, 2018). Average annual farmer incomes remain abysmally low at ₹77,976 (or US\$ 1,128; Financial Express, 2018). In terms of land and labour inputs, nearly 50% of India's land is tilled (Agriculture Census, 2015-16), and agriculture directly employs 41.1% of India's working age population (World Bank, 2018). This low output to input ratio points to the huge systemic flaws in India's agronomy that brings severe hardship to farmers and agricultural labourers, who bear the brunt of rising input costs, falling productivity, climate change, water scarcity, poor market access, technological stagnation, and so on. Although the agricultural predicament in India needs to be addressed at multiple levels, this article discusses the role of technology – in particular, artificial intelligence (AI) – in increasing agricultural output, and therefore, farmer incomes in India.')

```
# Heading
```

```
w2 = Label(canvas, justify=CENTER, text="PLANT LEAF DISEASE  
CLASIFICACION USING DEEP CNN ", fg="red")
```

```
w2.config(font=("Elephant", 20))
```

```
w2.grid(row=1, column=0, columnspan=2, padx=100)
```

```
# labels
```

```
BrpoquLb = Label(canvas, text="Predicting", fg="white", bg="blue")
```

```
BrpoquLb.grid(row=19, column=0, pady=10, sticky=W)
```

```
Play_video = Label(canvas, text="Agriculture", fg="white", bg="blue")
```

```
Play_video.grid(row=27, column=0, pady=10, sticky=W)
```

```
but5=Button(canvas, padx=0, pady=0, width=30, bg='white', fg='black', relief=GROOVE,  
command=Train_the_model, text='Browse and Predict Leaf  
Disece(click)', font=('helvetica 15 bold'))
```

```
but5.grid(row=19, column=1, pady=10, sticky=W)
```

```
but10=Button(canvas, padx=0, pady=0, width=30, bg='white', fg='black', relief=GROOVE
```

```
,command=Finder,text='All The Best For Agriculture(click) ',font=('helvetica 15 bold'))  
but10.grid(row=27, column=1, pady=10, sticky=W)
```

```
root.mainloop()
```