VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Electronics Engineering

Plant Communication System based on Machine Learning Concepts

Neural Networks and Fuzzy Controls

Done By : Team NEURAL BANKERS

Saswat Bhotto Mishra – 20BEC0786
Rastra Yadav – 20BEC0787
Malay Rajpoot – 20BEC0664
Ayan Chattaraj – 20BEC0485
Veerayya Vastrad – 20BEC0741

Motivation

To get a better knowledge of how plants interact with one another and with their environment, as well as how this might be utilised to detect and stop the spread of plant diseases, a project on plant communication systems and disease detection was initiated. Plants have developed a variety of means of communication, such as chemical signals emitted into the soil or atmosphere. These signals can provide details regarding the presence of predators, the accessibility of resources, and other significant elements that influence plant development and survival.
In addition, plants can express themselves to their surroundings physically by changing the way they grow in response to variations in light, temperature, and moisture. Researchers can learn more about how plants respond to various environmental factors and how they interact with other creatures in their habitat by examining these communication networks. Furthermore, early diagnosis and detection of plant diseases are essential for halting their spread and minimising crop damage. Visual inspection is a common component of traditional illness diagnosis techniques, however it can be time-consuming and inaccurate. However, recent technological developments have made it possible to create more effective and trustworthy methods for identifying plant illnesses. Examples include sensors and machine learning algorithms. The overall goal of a research on disease detection and plant communication systems is to increase our knowledge of how plants interact with their environment and to provide new tools and management techniques for plant diseases.

Literature Review and Research Gaps:

Serial no.	Author's name	Title of the paper	Conference name	Key points discussed	Gap identified
1	Muhammad Hammad Saleem Johan Potgieter Khalid Mahmood Arif	Plant Disease Detection and Classification by Deep Learning	IEEE	plant disease; deep learning; convolutional neural networks (CNN)	
2	Sachin D. Khirade A.B. Patil	Plant Disease Detection Using Image Processing	IEEE	Diseases, Feature extraction, Image color analysis, Image segmentation, Monitoring, Classification algorithms	Takes more time for image processing. Minimize sum of square distance between object and centroid. Difficult to predict value of k with fixed number of clusters. Every time need to select proper threshold value for getting better result in

3	Ferentinos, Konstantinos P.	Deep learning models for plant disease detection and diagnosis	IEEE	Convolutional neural networks Machine learning Artificial intelligence Plant disease identification Pattern recognition	
4	Shima Ramesh Mr. Ramachandra Hebbar Mr. P V Vinod	Plant Disease Detection Using Machine Learning	IEEE	Diseased and Healthy leaf, Random forest, Feature extraction, Training, Classification.	Require long training time. Difficult to understand learned function
7	Gurleen Kaur Sandhu Rajbir Kaur	Plant Disease Detection Techniques: A Review	ICACTM	Plant disease detection, image processing, image acquisition, segmentation, feature extraction, classification.	
8	LILI LI SHUJUAN ZHANG AND BIN WANG	Plant Disease Detection and Classification by Deep Learning—A Review	IEEE	Deep learning, plant leaf disease detection, visualization, small sample, hyperspectral imaging.	
9	Garima Shrestha Deepsikha Majolica Das Naiwrita Dey	Plant Disease Detection Using CNN	ASPCON	CNN, image processing, training set, test set	The major limitation was that accuracy of four different diseases was analyzed and the average accuracy is
10	Trimi Neha Tete Sushma Kamlu	Plant Disease Detection Using Different Algorithms	IEEE	Plant Disease, Image processing, Threshold algorithm, K-means cluster, Artificial neural network	Low accuracy
11	Mohameth Bingcai Sada	Plant Disease Detection with Deep Learning and Feature Extraction	IEEE	Plant Diseases Detection, Feature Extraction, Transfer Learning, SVM, KNN	the new features generated are not interpretable by humans.

- ## Aim of the project and Objectives:

The most difficult challenge in the field of agriculture at the moment is effective plant health monitoring. The decline of overall plant health is a negative outcome of physical monitoring of plant health. The solution we suggested provides a platform for the plant to express its emotions by calculating the overall plant health score and generating the relevant emoji on an 8*8 LED matrix (MAX7219). The metrics of plant health will now be communicated to the user by this proposed system via an IoT cloud platform. Soil moisture, light intensity, temperature, humidity, and other factors all affect plant health. Results from this study's single plant can be extrapolated to several plants.

HARDWARE USED:

- ARDUINO UNO
- ESP 32
- SOIL MOISTURE SENSOR
- DHT11 SENSOR
- LDR SENSOR
- LCD
- 8 X 8 LED DOT MATRIX

SOFTWARE USED:

- JUPYTER NOTEBOOK
- REACT JS
- ARDUINO
- THINKSPEAK

Methodology &Block diagram

At present, people are concerned about plants as awareness is increasing. But there are no appropriate tools for monitoring plant health. This leads to easy ignorance of the overall plant health by people in everyday life.

Keeping your plants alive can be quite a challenge as they are very bad at communication.

Moreover, in the last few decades, the pace of life has accelerated and become as fast as ever before. The modern world consistently sets the bar high and invites us to step on the fast track. This is where the problem arises, because of our fast-paced and busy life, we often forget to take care of our plants.

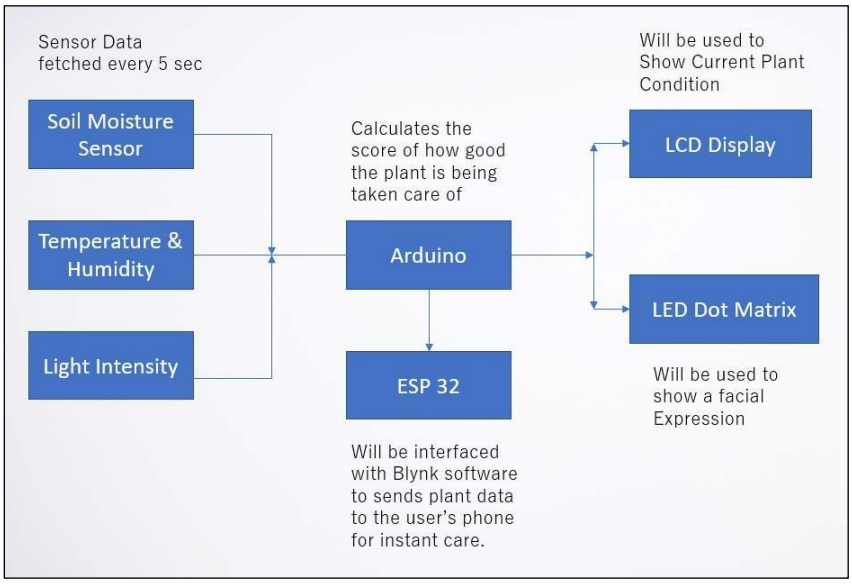
To solve this very problem, we have decided to create a communication tool, which tells us about how the plant feels at any given time by implementing principles of IoT and it would also remind us to fulfill any of its needs. Our work is divided into 2 stages. Stage 1, includes sensor data collecting and hardware implementation. Stage 2, implements the hardware interfacing with the IoT cloud platform and processing of data and sending the processed data over the internet to the ultimate user.

This Project is Divided into 2 parts,
Part I - Sensor Value Collection + Updating the values onto a cloud Server
Part II - Machine Learning implementation based on leaf disease detection

Part I

Sensor Value Collection + Updating the values onto a cloud Server

Block Diagram



Methodology

The brains of our project will be an Arduino – Uno. It is actually a microcontroller with 14 digital I/O pins and 6 analog I/O pins. It can be programmed in C++ language. Further ESP 32 will be used to connect the hardware to the Internet and send information to the mobile app. All our Sensors and displays will be connected to the Arduino using the jumper cables and the breadboard. Our Code will be plant-specific i.e., all the conditions (temperature, light intensity) will be based on a particular plant. This code will be compiled and uploaded to the Arduino with the Arduino IDE.

At first, all the data from the sensors will be collected and displayed to the user one by one with a delay.

Each of the parameters will have a score from 1-3. Based on the conditions of the plant, the score will be higher/ lower. Finally, the average score of all the parameters will be taken, and the user will be shown the score. This score will help the user know exactly how well they have taken care of the plant.

Now based on this score, the 8x8 LED Dot matrix will display a face.

- If the score is high, it will show a smiling face.
- If the score is low, it will show a sad face.

This gives the plant a "Face" to communicate.

System Requirements

1. **Hardware Descriptions**
   • Arduino UNO microcontroller
   • Soil Moisture sensor
   • DHT11 Temperature and Humidity Sensor
   • Photoresistor
   • 10k ohm Resistor
   • 16×2 LCD display with I2C bus
   • 8×8 LED dot matrix • Breadboard
   • Jump wires
   • Power supply

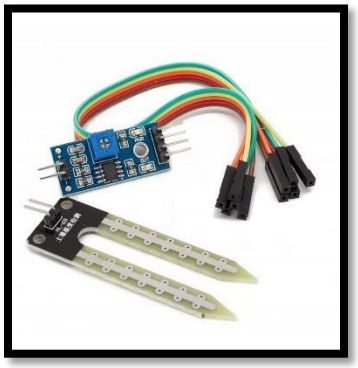
2. **Software Description**
   • Arduino IDE
   • Thingspeak
   • IFTT

STAGE 1

Sensor data collection and Hardware Implementation

(a) Soil Moisture Sensor

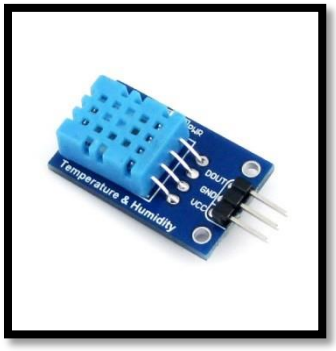
This sensor will be used to measure the amount of water present in the soil. It will be used to check if the plant has been watered properly. This sensor will provide us with Analog data ranging from 0 -1024, which will be then be programmed to identify the suitable amount of water required by the plant. The Analog input pin for the soil moisture sensor is connected to the Analog pin A1 in the Arduino in our circuit.



Soil Moisture Sensor

(b) DHT11 Temperature and Humidity sensor

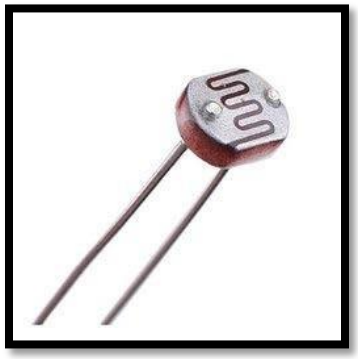
This is a temperature and a moisture sensor. It will help us monitor if the plant is in suitable temperature and moisture conditions. We will use an Arduino library for this sensor. The Data pin for this sensor is connected to the Digital pin 7 in our circuit.



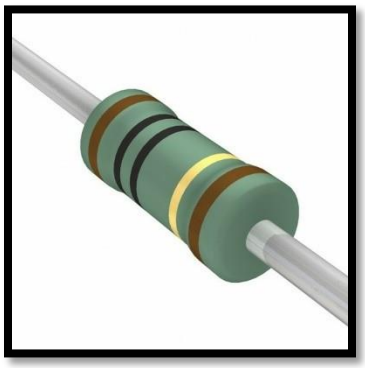
DHT11 temperature and Humidity sensor

(c) Photoresistor

This is a light intensity-based resistor. The more the light intensity more is the more resistance. This will help us in determining if the plant is in suitable light conditions. We have used the Analog pin A0 to get the voltage readings across the resistor.



Photoresistor

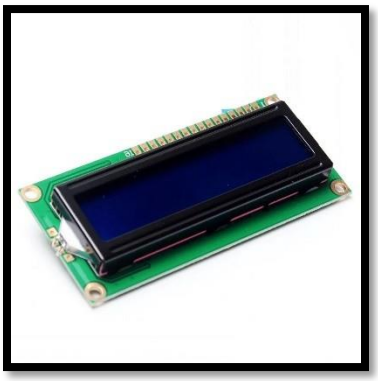


10K Ohm resistor

(d) LCD display

An electronic device that is used to display data and the message is known as LCD 16×2. As the name suggests, it includes 16 Columns & 2 Rows so it can display 32 characters (16 x 2=32) in total & every character will be made with 5×8 (40) Pixel Dots. So, the total pixels within this LCD can be calculated as 32 x 40 otherwise 1280 pixels. 16 x 2 displays mostly depend on multi-segment LEDs. There are different types of displays available in the market with different combinations such as 8×2, 8×1, 16×1, and 10×2, however, the LCD 16×2 is broadly used in devices, DIY circuits, electronic projects due to less cost, programmable friendly & simple to access. In our Project this display is mainly used to display necessary information to the user along with the score that the user has received for taking care of the plant. This display is

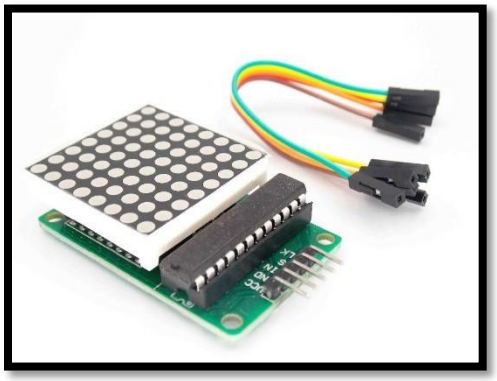
connected to the Arduino with the help of an I2C bus, which helps in reducing the number of cables.



16x2 LCD Display

(e) 8×8 LED dot matrix:

An 8 x 8 LED matrix display is used in this project to display the information. LED matrices are available in different styles like single colour, dual colour, multi-colour, or RGB LED matrix. They are also available in different dimensions like 5 x 7, 8 x 8, 16 x 16, 32 x 32 etc. Based on the arrangement of the LEDs in the matrix, an LED matrix can be either common row anode or common row cathode. In case of common row anode type LED matrix, the current sources (high or positive voltage) are given to the rows A-D and the current sinks (low or negative voltage or ground) are given to the columns 1-4. In case of common row cathode type LED matrix, the current sources (high or positive voltage) are given to the columns 1- 4 and the current sinks (low or negative voltage or ground) are given to the rows A-D. The LED matrix used in this project is a common row cathode-type LED matrix. While developing the project, the type of LED matrix must be known and the program must be written accordingly. This 8x8 matrix of LED will be used to project a face for the plant. If the plant is in healthy conditions, it will display a smile, if not it would display a sad face.



8x8 LED Dot matrix

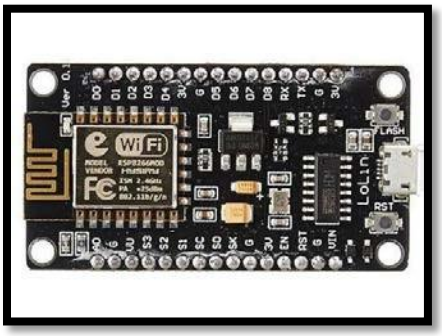
(f) Arduino Uno:

Arduino Uno is a microcontroller board based on the ATmega328P.It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 Analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.

Arduino Uno

(g) Node MCU

NodeMCU [ESP32] is a microcontroller board and is a low cost open source IoT platform. NodeMCU acts as firmware and a microcontroller unit which is further used as IoT development kit. The RX2 and TX2 pins of the NodeMCU are connected with RX and TX pins of Arduino UNO which helps in transmitting the required data over WiFi to the ThingSpeak platform.



Node MCU

STAGE 2

Hardware interfacing with IoT cloud platform

(a) ThingSpeak

ThingSpeak is an IoT analytics service that allows you to aggregate, visualize, and analyze live data streams in the cloud ThingsSpeaks provides instant visualizations posted by our gadgets to ThingSpeak ThingSpeak is free for small non-commercial projects. ThingSpeak includes a Web Service that lets us to collect and store sensor data in the cloud(pall) and develop Internet of Things operations. It works with Arduino, Raspberry Pi, and MATLAB, and should work on all kinds of programming languages since it uses REST API and HTTP In this study we get data (score) from our hardware model and analyze this data we use ThingSpeak inside which we create channel name and channel field (i.e Plant Feelings Communicator and data score in our case). Inside the channel, we generate an API key for our project which is important for our later steps. Then we add a suitable IOT package for ESP 32 and we connect ESP 32 with our own wifi by filling WIFI SSID and KEY with API key in the code blocks inside madecode.microbit.org. The data is viewed that was sent from our channel to the IoT platform which updates every time we make changes in parameters in our hardware model.

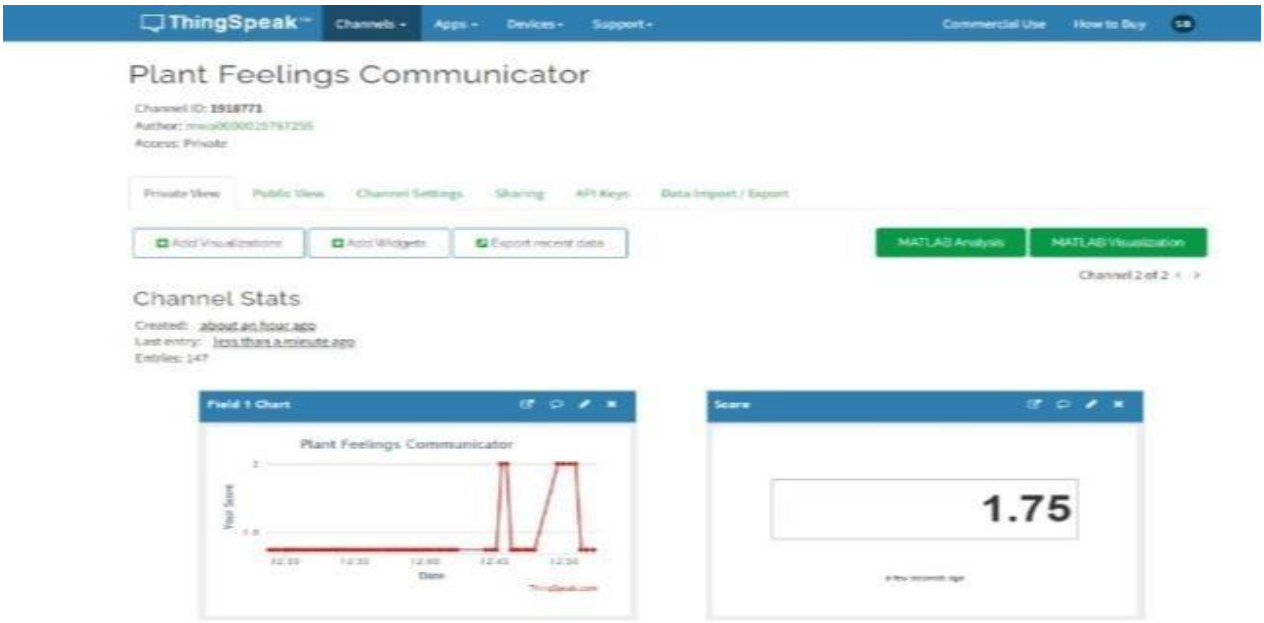


Fig: ThingSpeak Dashboard with Real-Time Data

(b) IFTTT:

It is an acronym of “if this then that”. In fact, it causes a series of chain responses to your website actions with the thing of" Put the internet to work for you", which brings us further convenience in operation. IFTTT connects all kinds of information through the process and also centrally presents our asked information to us, which solves the problem of eclectic information and admits or concentrates on important information. According to IFTTT, the operation of" this" is called " Trigger", that's to say our measures in a certain website while" that" means another conduct "

action" caused by the chain response. Those triggers and conduct are all grounded on a certain website, which is called “Task".
Now in this study, IFTTT is used to receive an email whenever the channel field score arrives at a threshold value where it activates the trigger to perform the designated action.
(i) This operation: From Webhook settings an applet is created where a trigger is chosen by setting down the trigger fields. Thus when the trigger fires every time our service receives a web request to notify it of an event.
(ii) That operation: It sends us an HTML-based email where the parameters of the trigger ares written. The following link is the link to a web request. It plays an important role in the Thingspeak setting later.

ThingSpeak is again used here since we have already uploaded parameters score data to Thingspeak iot platform. ThingSpeak contains multifarious apps but in this study we use Thing HTTP and React apps. A new Thing HTTP service is created in Thingspeak of connection with IFTTT. The URL is the Link of web request where we include the Private Key provided by IFTTT. Content type provided be JSON, because the expected format od IFTTT Maker Channel will be JSON. Within Body, we invoke the data in Channel. This is the data that is going to be sent to IFTTT. After that React Service is created. React work with Thing HTTP to perform actions when channel data meets a certain condition. From Test channel we change into our own channel and set the parameters fields and we check each time the conditions are met or not i.e to see if the parameter(score) value is below 1.9. If it is, then trigger Plant Feelings Communicator service in Thing HTTP. Primarily we use Thing HTTP to send an http request to IFTTT using React.

Fig: HTTP request using React

After the settings are done in ThingSpeak in Thing HTTP we tested it with our hardware Model. The different Parameters *(humidity, temperature, soil moisture, sunlight)* is increased/ decreased thus a different score is observed in the data of Thing HTTP the score has surpassed from the threshold values. And thus, we receive an email regarding the state or condition of our plant in our mail using IFTTT.

Fig: Email received using IFTTT.

Result and Discussion

The experimental result carried out in the proposed work using ThingSpeak and IFFT with our Hardware Model. We contemplated two test cases

Case I:

In the first case, we retain the plant in an ideal condition where all the parameters *(humidity, temperature, soil moisture, sunlight)* are met accordingly and observe a 2.50 score in our LCD panel display and find a happy face in our 8x8 LED dot matrix.
Thus we obtain a score of 2.70 in our Thing HTTP IoT Platform. That means the plant exhibits good condition and care.

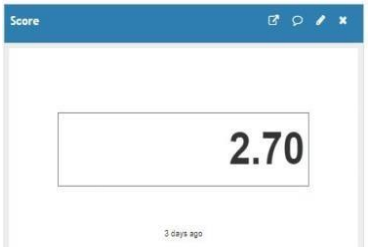


Fig: (a) Score obtained for ideal Case in Thing HTTP

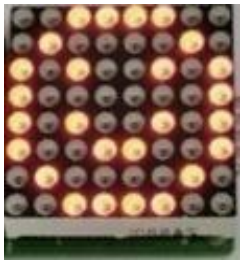


Fig: (b) 8x8 dot matrix displays happy face.

Case II:

In our final case, we retained the plant from both sunlight and soil moisture and a score of 1.75 is obtained on the LCD panel.
The 8x8 LED matrix displays a sad face indicating that the plant is in a poor state. Hence which score of 1.75 is observed in our Thing HTTP and triggers the service where we get notified through email. (via IFTTT).

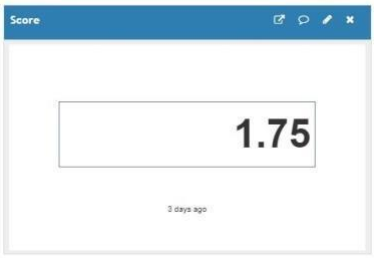


Fig: (a) Score obtained for nonideal case in Thing HTTP

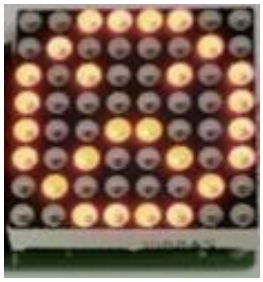


Fig: (b) 8x8 dot matrix displays sad face

Part II

Machine Learning implementation based on leaf disease detection

Introduction

Machine learning is the field of study that gives machines the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one has ever come across. As is evident from the name, it gives the computer something that makes it more like humans the ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect.

For our project we have chosen to use Python as our primary coding language. Along with python we use these libraries primarily

1. **Tensorlflow**
2. **Tensor.keras**
3. **Matplot.lib**

Tensorflow

Tensorflow is an extremely useful tool which helps **us to implement best practices for data automation, model tracking, performance monitoring, and model retraining**. Using production-level tools to automate and track model training over the lifetime of a product, service, or business process is critical to success.

Matplot.lib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

Procedure

For this project we have picked up the potato plant disease detection. A potato plant generally has 2 kinds of disease

i. Early Blight
ii. Late Blight

Our project can primarily detect these 2 diseases in a potato plant. It will be able to separate these two from a healthy plant.

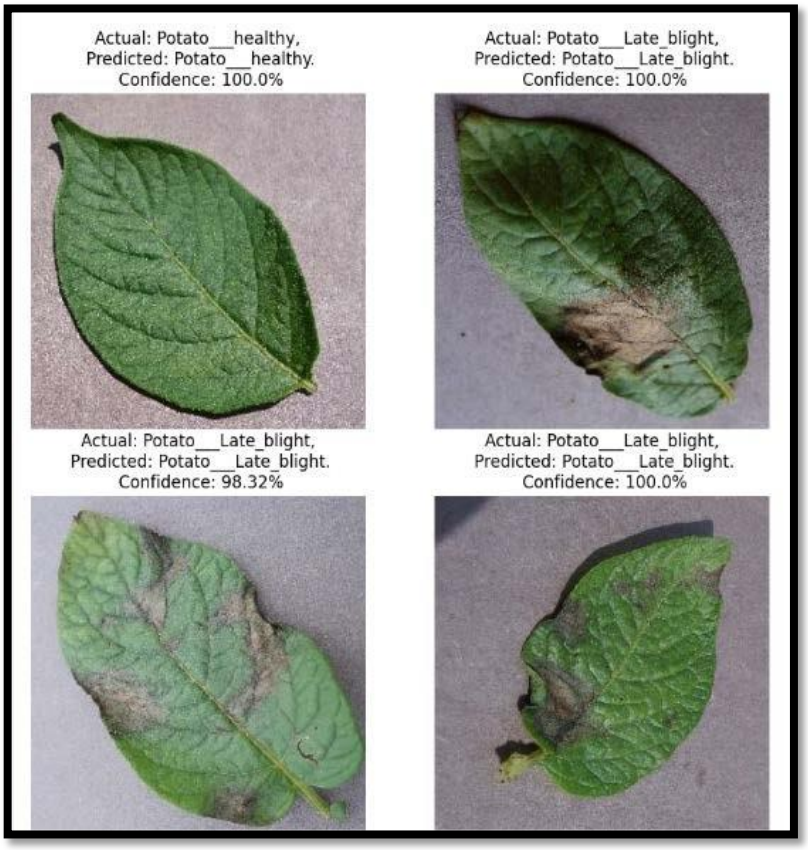
We have taken the data from Kaggle it is an open source where we have found approximately 2512 altogether of which there were approximately 1000 photos of normal leave 756 of early blight and 756 of late blight

After the Training the accuracy came out to be 96.98%

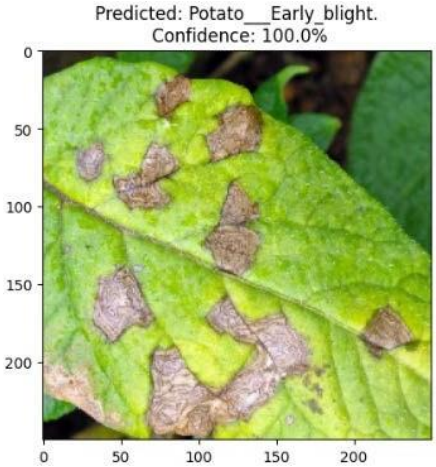
```
scores = model.evaluate(test_ds)
```

```
8/8 [==============================] - 9s 311ms/step - loss: 0.1619 - accuracy: 0.9698
```

For the analysis part, we used the 10% part and most of the images were classified correctly with a confidence of 100%. The image below shows, 4 out of the many image that were correctly classified.



Finally before saving the model, a random image from the internet was tested and the results came out to be correct with 100% confidence.



Finally the code was demonstrated to the teacher, and a picture was randomly tested again. The following bit of code was used to test the image on out built and saved model using jupyter notebook.

```
dimensions = (256, 256)
image = cv2.resize(image, dimensions)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
plt.imshow(image)
plt.axis("off")

img_batch = np.expand_dims(image,0)

prediction = model.predict(img_batch)
predicted_type = types[np.argmax(prediction[0])]
confidence = np.max(prediction[0])
print(predicted_type,confidence)
```

This time to the algorithm worked flawlessly and gave the result with 100% accuracy. as seen from the image.

Finally with the testing and training complete our model was found to have an Accuracy level of around 98.2756%.

Literature Review

M. Giri introduced a paper in 2013[3], which shared about drip irrigation system using microcontroller and solenoids, that used valve commutation based on the amount of water content present in the soil. Here the benefit achieved is that it successfully decreased soil erosion. However, its major drawback was its lack of use of real-life constraints.

D. Bansal in 2013[4], also proposed another work on WSN with microcontroller and GSM module. This technology is used to inform the farmers about the environmental condition of the crops. Yet this technique was not suitable for India since installing and using GPS in the rural areas is difficult due to poor connectivity.

Devika CM in 2017[5] utilised a simple Arduino in order to introduce automation in watering plants. The drawback of it was it restricted in terms of its resource sharing as the information could not be accessed globally.

M. Ramu in 2013[6] and S.G Zareen in 2016 [7] proposed their work on smart crop irrigation system using GSM. Today, we have much better options other than GSM and Arduino uno such as cloud computing [8] which will be more advantageous in maintaining a database of the different types of soils present in different regions of India.

D.Mishra in 2018[9] and A. Stesel in 2018[10] used Arduino Uno and Arduino Nano independently on with the Wifi module to achieve this. Still the complexity of the circuit can be reduced by using a standalone operation of Wi- Fi module.

S.no	Name of the Author	Title
1	A. Chatzopoulos	An Automated Plant Pot Controlled via the Internet based on Arduino Applications
2	Jude Allan Ambe Urmeneta	An Arduino-based Ph and Moisture Based Soil Plant Identifier
3	S. V. Devika, Sk. Khamuruddeen, Sk. Khamurunnisa, Jayanth Thota, Khalesha Shaik	Arduino Based Automatic Plant Watering System
4	Kritika Shah,	Proposed Automated Plant Watering System Using IoT
5	Yin Yin Nu, San San Lwin	Automatic Plant Watering System using Arduino UNO for University Park

6	Jude Allan Ambe Urmeneta	An Arduino-based Ph Soil Plant Identifier

CODE:

```
import tensorflow as tf
#from tensorflow.keras import models, keras
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
BATCH_SIZE = 32
IMAGE_SIZE = 256
CHANNELS=3
EPOCHS=50
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "PlantVillage",
    seed=123,
    shuffle=True,
    image_size=(IMAGE_SIZE,IMAGE_SIZE),
    batch_size=BATCH_SIZE
)
class_names = dataset.class_names
class_names
for image_batch, labels_batch in dataset.take(1):
    print(image_batch.shape)
    print(labels_batch.numpy())
plt.figure(figsize=(10, 10))
for image_batch, labels_batch in dataset.take(1):
    for i in range(12):
        ax = plt.subplot(3, 4, i + 1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")

len(dataset)
train_size = 0.8
len(dataset)*train_size
54.400000000000006
train_ds = dataset.take(54)
len(train_ds)
test_ds = dataset.skip(54)
len(test_ds)
val_size=0.1
len(dataset)*val_size
6.800000000000001
val_ds = test_ds.take(6)
len(val_ds)
test_ds = test_ds.skip(6)
```

```
len(test_ds)
def get_dataset_partitions_tf(ds, train_split=0.8, val_split=0.1, test_split=0.1, shuffle=True, shuffle_size=10000):
    assert (train_split + test_split + val_split) == 1

    ds_size = len(ds)

    if shuffle:
        ds = ds.shuffle(shuffle_size, seed=12)

    train_size = int(train_split * ds_size)
    val_size = int(val_split * ds_size)

    train_ds = ds.take(train_size)
    val_ds = ds.skip(train_size).take(val_size)
    test_ds = ds.skip(train_size).skip(val_size)

    return train_ds, val_ds, test_ds
train_ds, val_ds, test_ds = get_dataset_partitions_tf(dataset)
len(train_ds)
len(val_ds)
len(test_ds)
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds = test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
```

It seems that you are working on a computer vision task to classify plant diseases using the PlantVillage dataset. The code you provided loads the dataset using the tf.keras.preprocessing.image_dataset_from_directory() function, which is a convenient way to load images from a directory structure.

Then, you split the dataset into training, validation, and testing sets using the get_dataset_partitions_tf() function. This function shuffles the dataset (if specified) and splits it into three parts based on the specified proportions. Finally, you apply caching, shuffling, and prefetching to each of the splits using the .cache(), .shuffle(), and .prefetch() methods of the tf.data.Dataset class. These operations help to speed up data loading and processing during training.

Overall, this code prepares the PlantVillage dataset for use in a machine learning model, and splits it into training, validation, and testing sets for model evaluation.

Malay Rajpoot

```
#BUILDING THE MODEL
#Creating a Layer for Resizing and Normalization

resize_and_rescale = tf.keras.Sequential([
  layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
  layers.experimental.preprocessing.Rescaling(1./255),
```

```
])
#DATA AUGMENTATION

data_augmentation = tf.keras.Sequential([
  layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
  layers.experimental.preprocessing.RandomRotation(0.2),
])
#Applying Data Augmentation to Train Dataset

train_ds = train_ds.map(
    lambda x, y: (data_augmentation(x, training=True), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
#Model Architecture

input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 3

model = models.Sequential([
    resize_and_rescale,
    layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(n_classes, activation='softmax'),
])

model.build(input_shape=input_shape)
model.summary()
```

In this code, you first create two tf.keras.Sequential models: resize_and_rescale and data_augmentation. The resize_and_rescale model applies resizing and rescaling operations to the input images. The data_augmentation model applies data augmentation techniques to the input images during training, such as flipping and rotating the images randomly.

Next, you apply the data augmentation model to the training dataset using the .map() method of the tf.data.Dataset class. This maps the data augmentation model to each batch of images and labels in the training dataset, producing a new dataset with augmented images.

Then, you define the model architecture using the models.Sequential class. This model includes several convolutional layers followed by max pooling layers to downsample the input images. After the final max pooling layer, the model flattens the output and passes it through two fully connected layers (Dense layers) to perform classification. The final layer uses a softmax activation function to output class probabilities.

Finally, you build the model using the build() method and print a summary of the model's architecture using the summary() method.

Malay Rajpoot

```
#compiling the model

model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
history = model.fit(
    train_ds,
    batch_size=BATCH_SIZE,
    validation_data=val_ds,
    verbose=1,
    epochs=50,
)
```

It seems that you have successfully compiled and trained your model. Do you have any questions or concerns regarding the code?

Malay Rajpoot

```
scores = model.evaluate(test_ds)
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']
plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(range(EPOCHS), acc, label='Training Accuracy')
plt.plot(range(EPOCHS), val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
```

```
plt.plot(range(EPOCHS), loss, label='Training Loss')
plt.plot(range(EPOCHS), val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
#Run prediction on a sample image
import numpy as np
for images_batch, labels_batch in test_ds.take(1):

    first_image = images_batch[0].numpy().astype('uint8')
    first_label = labels_batch[0].numpy()

    print("first image to predict")
    plt.imshow(first_image)
    print("actual label:",class_names[first_label])

    batch_prediction = model.predict(images_batch)
    print("predicted label:",class_names[np.argmax(batch_prediction[0])])
#Write a function for inference
def predict(model, img):
    img_array = tf.keras.preprocessing.image.img_to_array(images[i].numpy())
    img_array = tf.expand_dims(img_array, 0)

    predictions = model.predict(img_array)

    predicted_class = class_names[np.argmax(predictions[0])]
    confidence = round(100 * (np.max(predictions[0])), 2)
    return predicted_class, confidence
plt.figure(figsize=(15, 15))
for images, labels in test_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))

        predicted_class, confidence = predict(model, images[i].numpy())
        actual_class = class_names[labels[i]]

        plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n Confidence: {confidence}%")

        plt.axis("off")
#Saving the Model
#We append the model to the list of models as a new version
model.save('model.h5')
#INFO:tensorflow:Assets written to: ../models/3/assets
#model.save("../potatoes.h5")
model_2 = models.load_model('model.h5')
plt.figure(figsize=(15, 15))
```

```
for images, labels in test_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))

        predicted_class, confidence = predict(model_2, images[i].numpy())
        actual_class = class_names[labels[i]]

        plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n Confidence: {confidence}%")

        plt.axis("off")
```

Great! It looks like you have trained a CNN model for image classification using TensorFlow and Keras. You have also evaluated the model and visualized the training and validation accuracy and loss, and performed inference on some test images. Finally, you have saved the trained model and loaded it again to perform inference. Well done!

Conclusion

In this paper we proposed a feelings communication system for plants which is simple and effective communication tool which help us to understand, the basic needs of the plant. This system will indicate and remind us if the plant is facing any scarcity of the basic needs. In this busy world all the people are busy in their works this communication system will help them to take care of the plants. This plant communication system can be used for household plants, offices and also in any other places, where growing up plants are necessary. The cost of implementation of this plant communication system is also low and can be further reduced with the aid of process specific printed circuit boards.

Comparison with other similar works

Plant disease detection and plant communication systems have been the subject of numerous works. Here are some examples of comparable projects and how they stack up against one that focuses on these subjects: Numerous studies have been conducted on plant communication systems, including how plants communicate with one another and with their environment by using chemical and physical signals. These investigations have shown that intricate communication is possible between plants, and that this can have significant ecological and evolutionary ramifications. The mechanics and purposes of plant communication, however, are still largely unknown, and further study in this field is required to advance our understanding. Plant disease identification can be done using a variety of conventional techniques, some of which might be time-consuming and imprecise, such as visual inspection. The employment of numerous technologies, including sensors, spectroscopy, and machine learning algorithms, is a feature of more contemporary approaches of plant disease detection. These techniques may be more effective and dependable than conventional techniques, but they might also call for more sophisticated tools and training to use. Integration of Plant Communication and Disease Detection: Research has been done on the application of plant communication systems for the detection of diseases, including the use of volatile organic compounds (VOCs) released by plants as disease indicators. However, there is still much to learn about the integration of disease detection and plant communication, and additional study is required to create efficient and useful methods for this. In conclusion, despite the numerous studies on plant communication systems and disease detection, there is still much to learn about these subjects, and further study is required to provide more efficient and useful approaches to controlling plant health.

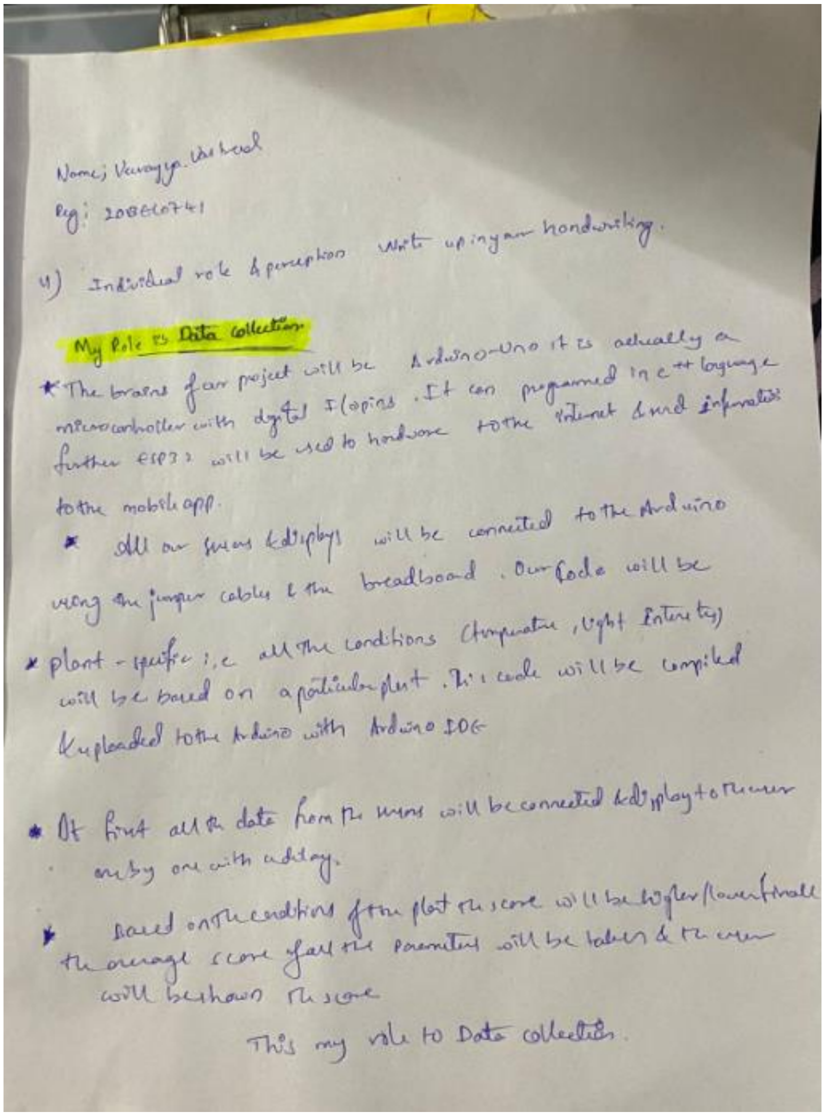
Conclusion

In this paper we proposed a feelings communication system for plants which is simple and effective communication tool which help us to understand, the basic needs of the plant. This system will indicate and remind us if the plant is facing any scarcity of the basic needs. In this busy world all the people are busy in their works this communication system will help them to take care of the plants. This plant communication system can be used for household plants, offices and also in any other places, where growing up plants are necessary. The cost of implementation

of this plant communication system is also low and can be further reduced with the aid of process specific printed circuit boards

REFERENCES

[1] Shaikh Sheroz Mohd Hasan, 'auto irrigation using arduinio'2016.
[2] Automatic Plant Watering and Monitoring System using NodeMCU
[3] M. Giri and D.N. Wavhal, "Automated Intelligent Wireless Drip Irrigation Using Linear Programming," International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) vol. 2, no. 1, pp. 1–5, 2013.
[4] D. Bansal and S. R. N. Reddy, "WSN Based Closed Loop Automatic Irrigation System," International Journal of Engineering Science and Innovative Technology (IJESIT), vol. 2, no. 3, pp. 229–237, 2013.
[5] Devika, C. M., K. Bose, and S. Vijayalekshmy. "Automatic plant irrigation system using Arduino." IEEE International Conference on Circuits and Systems (ICCS), Thiruvananthapuram, India 20-21 Dec,2017.
[6] M. Ramu and C. H. Rajendra, "Cost effective atomization of Indian agricultural system using 8051 microcontroller," International Journal of Advanced Research in Computer and Communication Engineering, vol. 2, no. 7, pp. 2563-2566 , 2013.Devika et al., International Journal of Advanced Research in Computer Science and Software Engineering 4(10), October - 2014, pp. 449-456
[7] S. G. Zareen, K. S. Zarrin, A. R. Ali and S. D. Pingle "Intelligent Automatic Plant Irrigation System,", International Journal of Scientific Research and Education, vol. 4, no. 11, pp. 6071–6077, 2016. Emerging Technologies for Monitoring Plant Health in Vivo Jenna M. Roper, Jose F. Garcia, and Hideaki Tsutsui.
[8] https://www.elecfreaks.com/blog/post/how-to-sendtemperature-threshold-value-alarm-email-via-ifttt.html
[9] D. Mishra, A. Khan, R. Tiwary and S. Upadhay "Automated Irrigation System-IoT Based Approach," 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU). IEEE, Bhimtal, India 23-24 Feb., 2018.



Name; Varayya. Vishal

Reg: 20BEC0741

4) Individual role & perception Write up in your handwriting.

My Role is **Data collection**

* The brains for our project will be Arduino-Uno it is actually a microcontroller with digital I/O pins. It can programmed in C++ language further ESP32 will be used to hardware to the internet & send information to the mobile app.
* All our sensors & displays will be connected to the Arduino using the jumper cables & the breadboard. Our code will be
* plant-specific i,e all the conditions (temperature, light intensity) will be based on a particular plant. This code will be compiled & uploaded to the Arduino with Arduino IDE
* At first all the data from the sensors will be connected & display to the user one by one with a delay.
* Based on the conditions for the plant the score will be higher/lower/finally the average score for all the parameters will be taken & the user will be shown the score

This my role to Data collection.