Integration of IoT and Machine Learning for Real-Time Plant Health Monitoring and Disease Detection System

Bhosale Aditya, Patil Manali, Lohar Sanika
Department of Electronics and Telecommunication Engineering Finolex Academy of Management and Technology, Ratnagiri, Maharashtra, India
{X210402, X210456, X210413}@famt.ac.in
Prof. Saurabh Athalye
Professor, Department of Electronics and Telecommunication Engineering Finolex Academy of Management and Technology, Ratnagiri, Maharashtra, India
saurabh.athalye@famt.ac.in

Abstract* – Agricultural yield is highly dependent on timely disease management and optimal growing conditions. In mango cultivation, especially for the Alphonso variety, diseases such as anthracnose and rust cause significant damage. This paper introduces an IoT-based system that combines environmental monitoring with machine learning-driven leaf disease detection. Temperature, humidity, and soil moisture are tracked using DHT11 and soil moisture sensors interfaced with a NodeMCU ESP8266. This data is visualized on ThingSpeak. For disease diagnosis, a trained Convolutional Neural Network (CNN) model classifies mango leaves into healthy, rust-infected, or fungal-infected categories. The model is deployed via a Streamlit web application, offering users an intuitive interface for image upload and result display. The integrated system supports precision agriculture through timely alerts and remedies, reducing manual inspection and promoting sustainable farming.

Keywords: *Alphonso mango, Convolutional Neural Network (CNN), IoT-based monitoring, Leaf disease detection, Smart agriculture, Internet of Things (IoT).*

I. INTRODUCTION

To address these challenges, this paper proposes a dual-module intelligent system that integrates CNN-based leaf disease classification with IoT-based environmental monitoring. The objective is to develop a smart and automated platform capable of detecting plant diseases at an early stage while simultaneously tracking crucial environmental conditions that influence plant health. The image classification module employs a Convolutional Neural Network (CNN) trained on a custom dataset of Alphonso mango leaf images, categorized into healthy, rust-infected, and fungal-infected classes. This model is embedded into a user-friendly web interface, allowing farmers to upload images for instant diagnosis.
Parallelly, the IoT module leverages a NodeMCU ESP8266 microcontroller integrated with sensors—specifically a DHT11 sensor for temperature and humidity, and a soil moisture sensor—to gather real-time environmental data. This data is transmitted to the ThingSpeak IoT platform, where it is visualized on a dynamic dashboard. The system thus enables remote monitoring of crop health and growing conditions, empowering farmers to make data-driven decisions that enhance productivity and reduce crop loss.
By combining visual plant diagnostics with sensor-based environmental awareness, this hybrid system aims to deliver a comprehensive solution that not only detects diseases early but also anticipates conditions conducive to disease onset. This paper outlines the design, implementation, and evaluation of the proposed system, emphasizing its potential as a cost-effective, scalable, and impactful tool for modernizing agricultural practices in India and beyond.

II. LITERATURE SURVEY

Dlodlo and Foko [1] discussed the growing impact of IoT across various industries, setting the foundation for further research. In agriculture, Khan and Sharma [2] integrated IoT with cloud computing to enable smart farming solutions through real-time monitoring. Shinde and Kulkarni [3] emphasized sensor-based plant health monitoring systems, while Sharma and Mehta [4] applied IoT and machine learning to optimize resource usage in farming. Patel and Shah [5] proposed a CNN-based system for early crop disease detection, and Sutar [6] extended this approach by applying AI-driven CNN models specifically for mango crop disease detection. Furthermore, platforms like ThingSpeak™ [7] support IoT-based agricultural systems by providing efficient data collection, analysis, and visualization tools.

III. PROPOSED SYSTEM

3.1 Methodology

The proposed solution is a dual-module system:
Environmental Monitoring: Collects data from temperature, humidity, and soil moisture sensors connected to NodeMCU. Data is sent to ThingSpeak at regular intervals.
Disease Detection: Users upload leaf images to a Streamlit-based web app. A CNN model processes the image and classifies it into one of three categories. Results include a prediction label, confidence score, and suggested remedy.
This system provides holistic insight into both plant environment and visible disease symptoms, allowing for data-driven interventions.

3.2 Details of Hardware and Software

a) Hardware Components

NodeMCU ESP8266: Wi-Fi-enabled microcontroller used for sensor data transmission.
DHT11 Sensor: Measures ambient temperature and humidity.
Soil Moisture Sensor: Detects moisture level in soil to evaluate irrigation needs.
Power Supply, Breadboard, Jumper Wires: For circuit setup and testing.

b) Software Components

Arduino IDE: For programming and flashing code to the NodeMCU.
ThingSpeak: Used as a cloud-based dashboard to visualize environmental data.
Python, TensorFlow, Keras: Used for building and training the CNN model.
Streamlit: Used to create a clean, responsive web app for image-based disease detection.

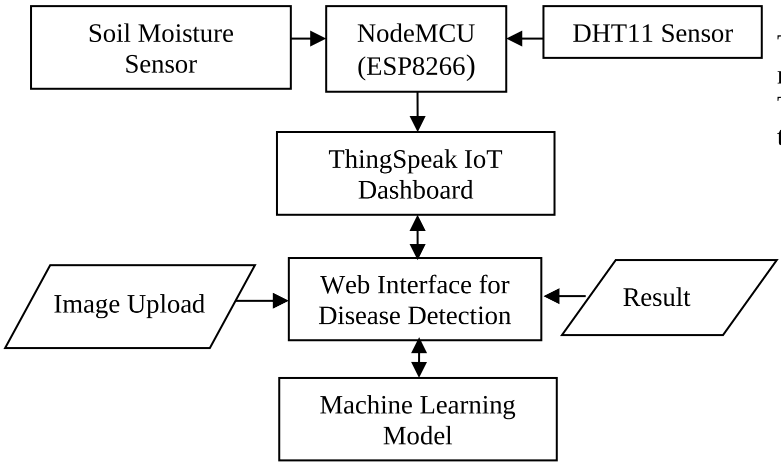


Figure 1: Block Diagram

3.3 Design Details

The design of the proposed system combines embedded hardware with cloud-based monitoring and machine learning-powered disease detection. The architecture is optimized for affordability, modularity, and ease of deployment in agricultural fields. This section elaborates on the hardware circuit, data processing, CNN model, software tools, and system integration.

a) Hardware Circuit Design

The system's physical layer includes the **NodeMCU ESP8266 microcontroller**, which is chosen for its compact form factor and built-in Wi-Fi capabilities. It serves as the central unit interfacing with two types of sensors:

- **DHT11 Sensor**: A digital sensor used to monitor ambient **temperature** and **humidity**. It is connected to a digital GPIO pin.
- **Soil Moisture Sensor**: An analog sensor that measures volumetric water content in soil and is connected to the **A0 analog input** of the NodeMCU.

The components are arranged on a breadboard, powered through a 5V USB input. Jumper wires are used to establish connections between the sensors and the NodeMCU pins. This circuit reads environmental parameters every 15 seconds and transmits them to the **ThingSpeak cloud platform** via HTTP POST requests using the microcontroller's Wi-Fi interface.

b) Firmware and Sensor Data Processing

The firmware running on NodeMCU is developed using the **Arduino IDE** and leverages the following libraries:

- ESP8266WiFi.h: Enables internet access via Wi-Fi.
- DHT.h: Reads temperature and humidity from the DHT11 sensor.
- ThingSpeak.h: Facilitates cloud connectivity with ThingSpeak API.

c) Software Stack and CNN Model Architecture

The disease detection module is powered by a **Convolutional Neural Network (CNN)** built using the **Keras** API with a **TensorFlow** backend. The image classification task involves identifying the leaf as either:

- **Healthy**
- **Rust-Infected**
- **Fungal-Infected (Anthracnose)**

The dataset is manually curated, consisting of clear images of mango leaves in various lighting and background conditions. The dataset is augmented with flipping, rotation, and scaling techniques.

d) Libraries Used:

- TensorFlow / Keras – Deep learning model design and training
- NumPy – Numerical array processing
- OpenCV – Image manipulation
- Matplotlib – Visualization during model training
- PIL (Pillow) – Image loading and formatting
- Streamlit – Frontend deployment of the detection app

e) CNN Model Layers:

- Input layer with image size **256×256×3**
- 3 × Convolution layers (filters: 32, 64, 128)
- ReLU activation after each convolution
- Max Pooling layers after each convolution block
- Dropout layer (rate 0.3) for regularization
- Flatten layer
- Dense layer (128 neurons)
- Output layer with Softmax activation for 3-class classification

The model achieves over **94% validation accuracy** and is saved as a .h5 file for later use in deployment.

f) Image Processing and Prediction Pipeline

When a user uploads a leaf image on the Streamlit interface, the image undergoes preprocessing:

1. **Resized** to 256×256
2. **Normalized** (pixel values scaled to 0–1)
3. **Expanded** into a 4D tensor
4. Passed to the model for prediction
5. The predicted class and confidence are displayed to the user

g) Web Interface Integration

The entire detection workflow is embedded into a **Streamlit web app**, designed for ease of use by non-technical users. The app features:

- A file uploader for mango leaf images

- Display of predicted disease class and confidence score
- Suggested remedies or care instructions
- A button or link to navigate to the ThingSpeak dashboard for sensor data

The web app runs on localhost or cloud services and loads the .h5 model dynamically for quick, real-time prediction without requiring an external API.

This integration enables the system to act as both a **smart sensor dashboard** and a **disease detection advisor**, offering a complete solution to farmers and agricultural researchers.

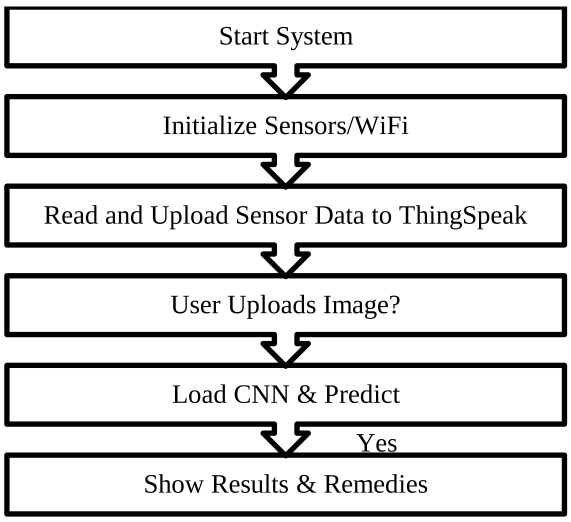


Figure 2: Flow Diagram

3.4 Pseudo Code

3.4.1 Sample preprocessing on NodeMCU (pseudocode):

```
BEGIN
Step 1: Import required libraries
{
   Import ESP8266WiFi.h // For Wi-Fi connection
   Import DHT.h // For temperature and humidity sensor
}
Step 2: Define Wi-Fi and ThingSpeak parameters
{
   Set API Key for ThingSpeak
   Define Wi-Fi SSID and Password
   Define ThingSpeak server address
}
Step 3: Initialize DHT sensor and Wi-Fi connection
{
   Begin serial communication
   Initialize DHT11 sensor
   Connect to Wi-Fi network
   While Wi-Fi not connected:
      Print connection status
}
Step 4: Start main loop
{
   Read humidity and temperature values from DHT sensor
   If sensor reading is not valid:
      Print error message
      Return to start of loop
}
Step 5: Connect to ThingSpeak server
{
   If connection successful:
      Create HTTP POST request
      Attach temperature and humidity data
      Send data to ThingSpeak
}
Step 6: Display sensor data on serial monitor
{
   Print temperature value
   Print humidity value
}
Step 7: Close connection
{
   Stop client connection
}
Step 8: Add delay before next data transmission
{
   Wait for 1 second (1000 milliseconds)
}
END
```

3.4.2 Pseudo Code for Model Training

```
BEGIN
Step 1: Import required libraries
{
   Import TensorFlow, Keras layers, ImageDataGenerator
   Import NumPy, Matplotlib, OpenCV, Pickle
   Import train_test_split, LabelBinarizer from sklearn
}
Step 2: Define parameters
{
   Set epochs, learning rate, batch size, and image dimensions
}
Step 3: Load and preprocess images
{
   Define function to read images and convert them to arrays
   Resize images to 256×256 pixels
   Normalize pixel values
   Label images according to disease class
}
Step 4: Encode labels
{
   Apply LabelBinarizer to convert disease names into binary format
   Save the label binarizer using Pickle
}
Step 5: Split dataset
{
   Split images and labels into training and testing sets (80%-20%)
}
Step 6: Data augmentation
{
   Apply random rotations, flips, shifts, and zooms to images using ImageDataGenerator
}
Step 7: Build CNN model
```

```
{
    Initialize a Sequential model
    Add convolutional, activation, batch normalization,
pooling, and dropout layers
    Flatten output
    Add fully connected dense layers
    Use Softmax activation at the output layer
}
Step 8: Compile model
{
    Set loss function to binary_crossentropy
    Use Adam optimizer and monitor accuracy
}
Step 9: Train model
{
    Train the model using the training set
    Validate using the test set
}
Step 10: Plot training and validation performance
{
    Plot graphs for accuracy and loss over epochs
}
Step 11: Evaluate model
{
    Calculate final test accuracy
    Print test results
}
Step 12: Save the trained model
{
    Serialize the model using Pickle
}
Step 13: Load model and predict
{
    Load saved model
    Preprocess a new image
    Predict disease class and probability
    Print predicted disease name and confidence
}
END
```

3.4.3 Pseudo Code for Web Interface Deployment:

```
BEGIN
Step 1: Import required libraries
{
    Import Streamlit for web UI components
```

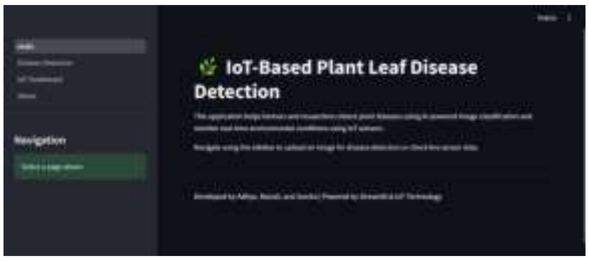


Figure 3: Main Page of Web Interface

```
    Import TensorFlow/Keras to load trained CNN model
    Import NumPy for numerical processing
    Import PIL (Pillow) for image handling
}
Step 2: Load the trained CNN model
{
    Use Kerasload_model() function to load the .h5 model
file
}
Step 3: Define preprocessing function for uploaded images
{
    Resize image to 256×256 pixels
    Normalize pixel values to 0–1 scale
    Expand dimensions to match model input shape
}
Step 4: Create Streamlit web application layout
{
    Set page title and sidebar navigation
    Create file uploader widget for leaf image upload
}
Step 5: Handle user input
{
    If image is uploaded:
        Preprocess the image
        Predict the disease class using CNN model
        Display predicted disease name
        Display confidence score
}
Step 6: Provide additional user information
{
    Suggest basic remedies for the detected disease
    Provide a navigation link/button to the ThingSpeak IoT
dashboard
}
END
```

IV. EXPERIMENTAL RESULTS

1. The IoT system was implemented and tested using the NodeMCU ESP8266 microcontroller with DHT11 and soil moisture sensors. Real-time data on temperature, humidity, and soil moisture was successfully transmitted to the ThingSpeak cloud platform every 15 seconds.
2. For the disease detection module, a Convolutional Neural Network (CNN) model was trained using a curated dataset of mango leaf images categorized into healthy, rust-affected, and fungal-infected types. The model reached a training accuracy of 96.2% and a validation accuracy of 94%, delivering predictions with high confidence. When tested with new images, the model consistently provided results within one second, ensuring suitability for real-time detection.
3. The complete system was deployed through a user-friendly web application developed using Streamlit. Users could upload leaf images and instantly receive disease classification results along with care suggestions. The integration of live sensor data with AI-powered diagnosis provided a practical and efficient tool for monitoring plant health, suitable for use in real agricultural environments.

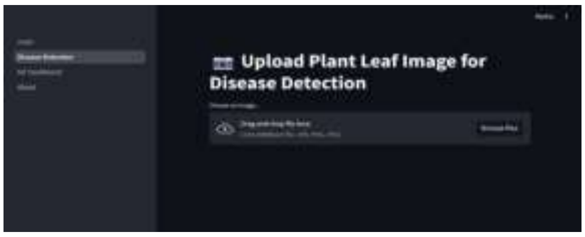
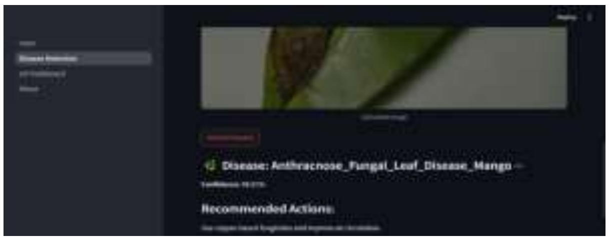
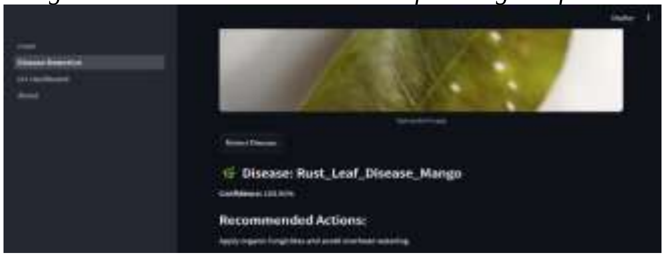

Upload Plant Leaf Image for Disease Detection

Figure 4: Leaf Disease Detection Interface


Disease: Anthracnose_Fungal_Leaf_Disease_Mango

Recommended Actions:

Figure 5: Disease Detection Result for Fungal Infection


Disease: Rust_Leaf_Disease_Mango

Recommended Actions:

Figure 6: Disease Detection Result for Rust Infection

IoT Sensor Data Dashboard

Figure 7: IoT Dashboard Link and Monitoring Interface

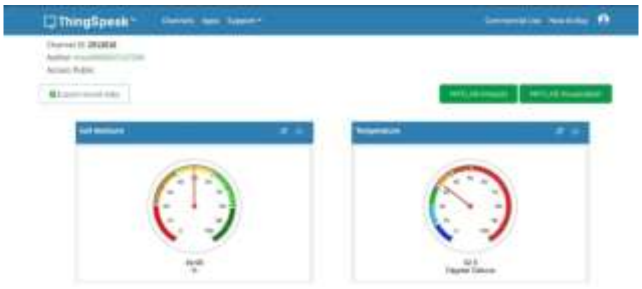


Figure 8(a): IoT Sensor Data Output 1

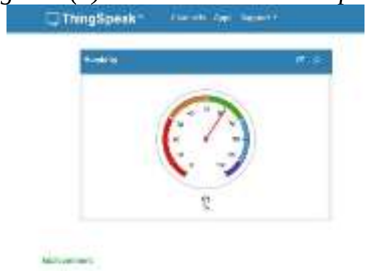


Figure 8(b): IoT Sensor Data Output 2

V. CHALLENGES FACED

1. **Limited Dataset Availability:** There was a lack of publicly available, high-quality mango leaf images categorized by specific diseases. This required manual collection and labeling of images, increasing the time and effort needed for training the machine learning model.
2. **Unstable Network Connectivity:** Maintaining consistent Wi-Fi connectivity in rural or farm environments was challenging, occasionally interrupting the data transmission to the ThingSpeak cloud platform and affecting real-time monitoring.
3. **Sensor Inconsistencies:** The DHT11 sensor produced slight fluctuations in temperature and humidity readings. To improve accuracy, additional filtering techniques and averaging methods had to be implemented in the firmware.

VI. REFERENCES

[1] Dlodlo, N., and Foko, T., 2012, *The State of Affairs in Internet of Things Research*, Springer, New York.

[2] Khan, M., and Sharma, R., 2021, "Smart Agriculture Using IoT and Cloud Computing," *International Journal of Recent Technology and Engineering (IJRTE)*, 9(6), pp. 34–39.

[3] Shinde, S., and Kulkarni, R., 2022, "Analysis of Sensor-Based Plant Health Monitoring Using IoT," *Journal of Emerging Technologies and Innovative Research (JETIR)*, 9(5), pp. 102–108.

[4] Sharma, R., and Mehta, A., 2023, "Optimizing Resource Usage in Smart Farming with IoT and ML," *International Journal of Computer Applications*, 185(42), pp. 19–25.

[5] Patel, K., and Shah, M., 2020, "IoT-Based Crop Disease Prediction System Using Convolutional Neural Networks," *International Conference on Intelligent Computing and Control Systems (ICICCS)*, IEEE, pp. 570–574.

[6] Sutar, A., 2023, "Implementation of AI-Driven Disease Detection in Mango Crops Using CNN," *M.Tech. Thesis*, Vellore Institute of Technology, Vellore, India.

[7] ThingSpeak™, "ThingSpeak™ Documentation," [Online]. Available: https://thingspeak.com/docs/. [Accessed: Apr. 9, 2025].