2024 27th International Conference on Computer and Information Technology (ICCIT)
20-22 December 2024, Cox's Bazar, Bangladesh

Plant Leaf Disease Detection Incorporating IoT and XAI-Enhanced Deep Learning

Pijush Kanti Roy Partho[1*], Md. Abid Hasan Rafi[2*], Pankaj Bhowmik[3†]
[*]Dept. of Electronics and Communication Engineering, [†]Dept. of Computer Science and Engineering
Hajee Mohammad Danesh Science and Technology University, Dinajpur-5200, Bangladesh
[1]pijushkantiroy2040@gmail.com, [2]ahr16.abidhasanrafi@gmail.com, [3]pankaj.cshstu@gmail.com

***Abstract*—Plant diseases represent a significant global threat, affecting both food supply and economic growth. Despite technological advances, early plant disease recognition remains difficult for small-scale farmers because of the lackings of infrastructure and resources. Thus, early disease detection using machine learning, deep learning, and computer vision techniques can improve the prediction process and make it easier. Previous studies that were implemented for plant-specific purposes, require high-resolution images and lack real-time detection methods. Therefore, to address these issues, we proposed a deep learning and IoT-based system that uses convolutional neural networks (CNN) and other supporting models, such as InceptionNetV3, VGG16, and EfficientNetB5, with a fine-tuning process to achieve higher accuracy in predicting 38 diseases over 14 different plant species in lower resolution, noisy data. Besides, to facilitate the system, we incorporated an ESP32CAM microcontroller and a server host that can be used to detect diseases in a real-time environment, which makes our research study robust and meaningful. We also implemented Explainable Artificial Intelligence (XAI), particularly Grad-CAM, along with our CNN model for visualizing the infected region in a plant leaf. The individual-trained models achieved an average accuracy of around 99%. We believe, our study can contribute to the agricultural sector by providing real-time disease detection, smart decision support systems, and low-cost solutions for farmers, mainly in resource-limited territories.**

***Index Terms*—Deep Learning, CNN, Decision Support System, Grad-CAM, ESP32-CAM**

I. Introduction

Human civilization has always relied on agriculture, but crop diseases pose significant risks to food security by reducing yields and burdening the economy. This is an important issue globally, and accurately predicting early-stage diseases is difficult. The lack of skilled pathologists to diagnose agricultural diseases remains an ongoing concern. As a result, infections spread quickly, possibly interrupting food production and impacting the economy. To reduce agricultural losses, there is an urgent need for significantly better crop disease detection, monitoring, and prediction. In this case, combining the Internet of Things (IoT) and Machine Learning (ML) has enormous promise to improve large-scale crop monitoring [1]. Machine learning and deep learning-based models are already playing a vital role in this area to optimize loss by predicting early diseases [6] [8]. Some research demonstrates how different types of deep neural network algorithms when combined with computer vision, perform better than the naked eye [9] [2].
Although several concepts are based on ML, DL, and computer vision models, significant gaps remain. They indeed build some effective models, but the main gap is in real-time performance.
To overcome those circumstances, our main objective is to create an innovative approach to plant disease diagnosis that combines deep neural networks with XAI to offer farmers a smart decision support system. Using cutting-edge deep learning models like Convolutional Neural Networks (CNNs), InceptionNet, VGG16, and EfficientNetB5, we integrate IoT-enabled ESP32-CAM with microcontrollers to provide real-time using cloud-based server disease prediction for 38 different diseases across 14 plant species. The technology has been designed to be adaptable, low-cost, and easily accessible, providing the agriculture industry with immediate data regarding crop health with the help of the decision support system.

II. Literature Review

Recent research on plant disease detection has increasingly focused on deep learning techniques, particularly the application of Convolutional Neural Networks (CNNs) for image classification [11] used in the agriculture sector aims a great purpose. Today's world very much depends on technology, there are already existing technologies in this field using machine learning, deep learning, IoT, etc. for advanced time basis solutions. This section reviews some previous research that will help to detect plant leaf disease detection using machine learning and deep learning approaches.
Chen *et al.* [4], made an IoT and machine learning combination system named RiceTalk that detects rice blast disease in its early stages by analyzing nonimage IoT real-time data like weather and soil sensors. They used CNN to train the sensor data and got an average prediction accuracy of 89.4%. Another DL-based approach that detects plant disease was proposed in [7]. They proposed a CNN model, obtaining 97.77% accuracy with ResNet38 and 97.58% accuracy with VGG16 for diagnosing 38 plant diseases. Atila *et. al* [2] proposed a DL-based framework, which achieved the best plant leaf disease classification accuracy using deep learning methods, surpassing ResNet50 and Inception V3. However, it faces challenges including dataset dependency, high computational requirements, and the need for real-world validation. Panchal *et. al* [14] build the different traditional machine learning models to detect and classify plant disease with 98% for random forest classifier, 94% DT, 92% KNN, and 90% in SVM.

979-8-3315-1909-4/24/$31.00 ©2024 IEEE

However, this study deals with several drawbacks focusing on different disease scopes and environment variability.
An ESP32 camera module using a PIR sensor for motion-triggered image collection and incorporating with an Arduino-based security system produced a 100% success rate in photo delivery for remote surveillance sending the images to Telegram [18]. According to an analysis of the Industrial Internet of Things, low-level microcontrollers such as STM32 and ESP32-CAM are ideal for building efficient security devices. To meet this demand for efficient modern technology a cost-effective smart security alarm was developed by via protocols like LoRa, ZigBee, and MQTT [16]. Another cost-effective surveillance system was developed using ESP32, OV2640 camera, PIR sensor utilizing TWILIO API for SMS alerts, making it suitable for areas with insufficient security infrastructure with half what conventional systems cost [13]

TABLE I: RECENT BENCHMARK STUDIES

Reference	Dataset	Model	Performance	Limitations
[1]	PlantVillage	CNN	94%	Limited disease coverage
[6]	The PlantVillage	EfficientNetV2	95%	Overfitting
[8]	PlantVillage	CNN, InceptionV3	99.34%	Slight overfitting
[9]	PlantVillage	CNN	98.29%, 98.03%	Lack of real-world testing
[10]	Apple leaf Disease Detection(ALDD)	VGG-FCN-VD16, VGG-FCN-S	97.95%, 95.12%	Environmental interference
[20]	PlantVillage	ResNet18	96%	Overfitting risk

In essence, a significant number of research works focused on theoretical reasoning rather than real-time plant leaf disease detection. Along with this, there are scenarios when experiments are just centered on a specific disease, which is unsuitable for environmental variation. We believe that most of these models are not appropriate for detecting plant disease in a real-time environment. Therefore, we planned to develop an IoT-based real-time early disease detection system incorporating an XAI-supported CNN model that can assist agriculture stakeholders in making smart decisions.

III. METHODOLOGY

In this section, Figure 1 illustrates the detailed description of our proposed framework incorporating data preprocessing techniques, CNN architecture, Fine-tuning, Grad-Cam, and other DL techniques.

A. Dataset Description

The dataset used in this study is a modified version of the PlantVillage dataset. This dataset has around 87,000 RGB images of both healthy and diseased plant leaves, divided into 38 different classes of plant disease individuals. 80% (70295 images) are provided for training, and 20% (17572 images) for validation [3]. Figure 2 illustrates the sample images with 256×256 pixel resolution from labeled random classes.

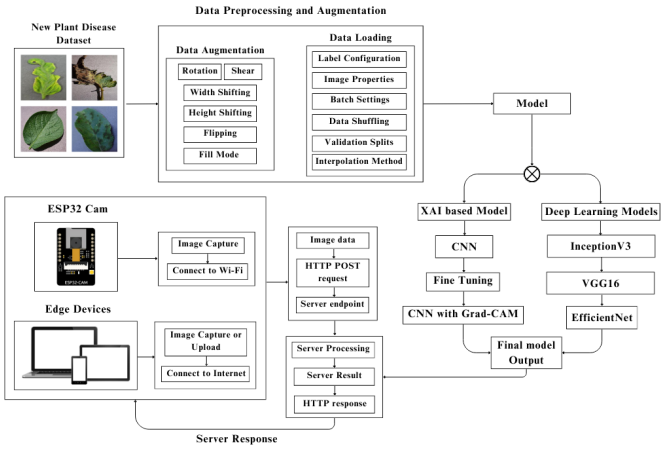


Fig. 1: Proposed framework

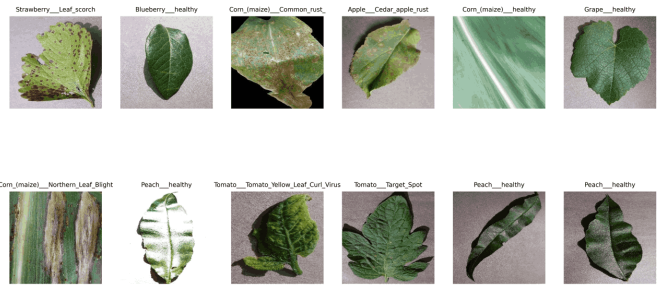


Fig. 2: Sample images from random classes with labels

B. Image Processing and Data Augmentation

For dataset improvement, additional processes were applied like scaling the picture sizes, normalizing the pixel values, and data augmentation. Algorithm 1 discusses the process of data preprocessing and augmentation.

Algorithm 1 Data Preprocessing and Augmentation

Input: Dataset D with images I
Output: Processed dataset $D_{processed}$
$D_{processed} \leftarrow \emptyset$
for each image $I \in D$ **do**
  Resize I to $(128, 128)$
  Normalize pixel values: $I \leftarrow \frac{I-127.5}{127.5}$
  Augment I:
    Randomly apply:
      Rotation $\theta \sim \text{Uniform}(-15°, 15°)$
      Width shift $w \sim \text{Uniform}(-0.1, 0.1)$
      Height shift $h \sim \text{Uniform}(-0.1, 0.1)$
      Shear $s \sim \text{Uniform}(-0.1, 0.1)$
      Zoom $z \sim \text{Uniform}(0.8, 1.2)$
  Add augmented I to $D_{processed}$
end for
Return $D_{processed}$

IV. Empirical Analysis of The Proposed System

*a) **Hardware Setup**:* We set up the hardware using ESP32-CAM to capture and transmit images to a server for disease categorization and recognition. The server delivers an API, the backbone of an ESP32-CAM setup adept at handling POST requests. Image capture and transmission process has been described in algorithm 2. A web interface was also developed to allow devices with a functioning internet browser to connect with the server and employ its performance. The ESP32-CAM, along with other edge devices, interact with the server through POST requests in the same manner, ensuring a fixed and regular response to any inquiry on image classification. The circuit diagram of ESP32CAM with Arduino UNO is shown in Figure 3 below. The server infrastructure primarily comprises an API capable of handling POST requests. The ESP32-CAM and other edge devices interact with the server through identical POST request mechanisms, enabling consistent and reliable output from the model's image classification process. In Figure 4, we build a user interface for edge devices to perform the prediction in the local server by uploading images. The left side of this figure shows the user interface on the local machine and the right side, shows the backend server logs. It will take images as input, then process with our pre-defined models and give the prediction result as a particular disease name. The developed interface frontend code and server backend code for our system has been provided in our GitHub repository[1].

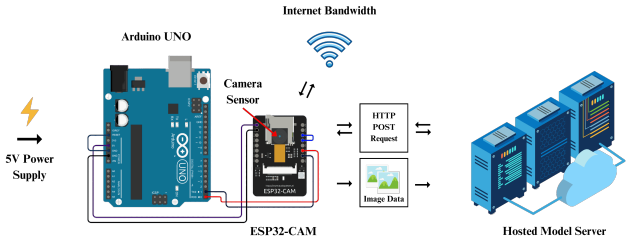
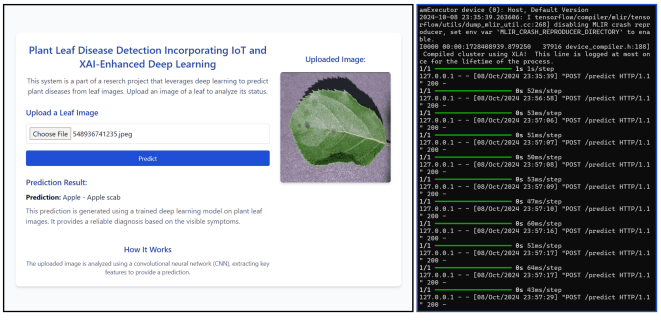


Fig. 3: Circuit diagram of ESP32CAM with Arduino UNO

Algorithm 2 Image Capture and Transmission

1: **Initialize:** Connect to WiFi.
2: **Camera Setup:** Configure camera.
3: **Capture Image:** Capture with ESP32-CAM.
4: **Error Handling:** If capture fails, log and retry.
5: **Prepare Request:** Define server and set headers.
6: **Transmit Image:** Send the image via POST and await a response.
7: **Handle Response:** Log success or error, and retry if needed.



User Interface on Local Machine | Localhost Server Logs

Fig. 4: User interface for edge devices

*b) **Custom CNN**:* Convolutional Neural Networks (CNN) are a type of Artificial Neural Network (ANN) useful for processing images [11]. In our CNN model, we used three different approaches i.e., building the model and training, fine-tuning, and finally implementing XAI (Grad-CAM) to visualize the infected places. The proposed architecture doesn't use a pre-trained model like VGG16 or InceptionNet, and it was manually built by adding convolutional layers, max-pooling layers, dropout, and fully connected layers. These parameters (filter sizes, number of layers, and dropout values) make the model custom-designed for our disease detection purpose.

*c) **Model Structure and Train**:* We built a custom CNN model that takes 128×128 shape images as input, and this architecture has `Conv2D` layers having 32, 64, 128, 256, and 512 filters and each of them followed by max-pooling layers for downsampling. We added the dropout layers (0.25) and 0.4 to avoid overfitting issues and flatten the output layer. A 2048-dimensional vector obtained from the flattened final feature maps was passed through a dense layer of 1500 units, followed by a 38-unit softmax output for disease classification. It was optimized with Adam optimizer (0.0001 learning rate), had 7,842,762 trainable parameters, and used categorical cross-entropy loss. Algorithm 3 demonstrates the architecture of our custom CNN model. In this model convolutional operation was applied to the input image using filters or kernels and can be described by the following equation:

$$S_{ij} = (I * K)_{ij} = \sum_{a=0}^{m-1} \sum_{b=0}^{n-1} I_{i+a,j+b} K_{a,b} \tag{1}$$

where S_{ij} is the output feature map at position (i,j), I is the input feature, K is the filter or kernel, and m and n are the dimensions of filters.

The softmax function is used in the output layer for multi-class classification:

$$\sigma(\vec{z}_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_i}} \tag{2}$$

where, $\sigma(z_i)$ is the softmax output for class i, z_i is the input score (logit) for class i before applying softmax, e^{z_i} is the exponential function applied to the score z_i. For multi-class classification, the loss function used is categorical cross-entropy, which is defined as:

$$H(y, \hat{y}) = -\sum_{i} y_i \log(\hat{y}_i) \tag{3}$$

[1]https://github.com/AbidHasanRafi/Prediction-System

where y is the true predicted label and $\hat{y}$ is the predicted probability.

Algorithm 3 CNN Model Architecture

Input: Preprocessed dataset $D_{processed}$
Output: Trained CNN model M
Initialize model M
Add Layer: $C_1 = \text{Conv2D}(32, (3,3), \text{activation="ReLU"})$
Add Layer: $P_1 = \text{MaxPooling2D}((2,2))$
Add Layer: $C_2 = \text{Conv2D}(64, (3,3), \text{activation="ReLU"})$
Add Layer: $P_2 = \text{MaxPooling2D}((2,2))$
Flatten: $F = \text{Flatten}()$
Add Layer: $D = \text{Dense}(128, \text{activation="ReLU"})$
Add Output: $O = \text{Dense}(38, \text{activation="softmax"})$
Compile model M with optimizer and loss function $L(y, \hat{y})$
Return M

*d) **Fine-tuning**:* Fine-tuning in machine learning is the process of modifying a pre-trained model for particular tasks. Due to its advantages, transfer learning or fine-tuning a pre-trained model is gradually gaining popularity in the present day [15]. The working process has been demonstrated on Algorithm 4 to fine-tune the CNN model.

Algorithm 4 Fine-Tuning

Input: Pre-trained model $M_{pretrained}$, dataset $D_{processed}$
Output: Fine-tuned model $M_{fine-tuned}$
Set learning rate $\alpha \leftarrow 1e^{-5}$ //small learning rates
Freeze layers of $M_{pretrained}$
Compile model with new learning rate α
Train model on $D_{processed}$ for E epochs
Return $M_{fine-tuned}$

*e) **Grad-Cam Implement**:* We used an XAI method called Grad-CAM (Gradient-weighted Class Activation Mapping) [17], to show class-specific activations in our CNN model. Figure 5, reflects the Grad-CAM visualizing techniques. From Selvaraju *et al.* study "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization" presents a unique approach called Gradient-weighted Class Activation Mapping [17]. Grad-CAM analyzes the gradient information in CNN's last convolutional layer to identify the weight of each neuron for a particular choice. According to the final convolutional layer feature map, the Grad-CAM approach consists of processing the neuron significance weights. a_k over the width and height dimensions noted by i and j. It calculates the gradient of score y_c for class c concerning feature map activation A_k of the convolutional layer. Eq. (4) demonstrates how the neuron importance weights are computed [12]:

$$a_k = \frac{1}{N} \sum_i \sum_j \frac{\partial y_c}{\partial A_{k_{ij}}} \tag{4}$$

where N is the number of pixels in the feature map. Algorithm 5 shown below presents the working method of Grad-CAM:

Algorithm 5 Grad-CAM Implementation

Input: Image I, model M, last conv layer L_{last}
Output: Grad-CAM heatmap H
Compute gradients: $\frac{\partial y^c}{\partial A}$ (y^c is the predicted class)
Pool gradients: $G \leftarrow \frac{1}{N} \sum_i \sum_j \frac{\partial y_c}{\partial A_{k_{ij}}}$
Compute activation maps: $A \leftarrow L_{last}(I)$
Compute heatmap: $H = \text{ReLU}\left(\sum_k G_k \cdot A_k\right)$
Normalize heatmap: $H \leftarrow \frac{H - \min(H)}{\max(H) - \min(H)}$
Resize H to match I
Superimpose H on I
Return Superimposed image

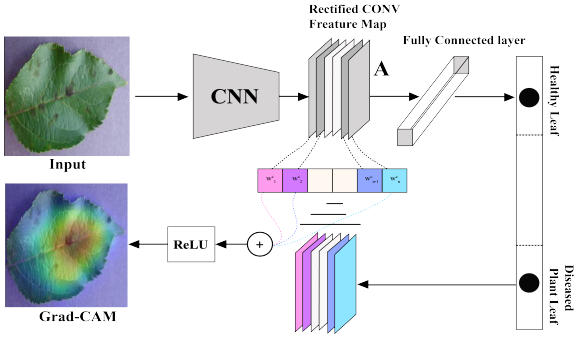


Fig. 5: Grad-CAM visual explanation mechanism

*f) **Other Deep Learning Models**:* We also developed some other deep learning models (i.e., EfficientNetB5, VGG16, InceptionV3) for comparison between our models.

Transfer Learning Method with EfficientNetB5: Transfer learning is a significant approach in machine learning for handling insufficient training data [19]. The last few layers of a pre-trained network may be removed and retrained with new layers for the current objective [2]. In this model structure, we loaded the pre-trained EfficientNetB5 having `imagenet` weights and skipped the output layers. Then we froze the pre-trained neural networks. In the transfer learning part, we applied data augmentation and the model generated the feature maps for input images. Then we added a fully connected dense layer with 256 neurons with RelU activation with BatchNormalization and a dropout layer (0.3). Before compiling, we used softmax activation in the output layer. Then we compiled this model with Adam optimizer using a small learning rate (0.0005) including categorical cross-entropy loss. Finally, we trained this model with 10 epochs and set `ModelCheckpoint`, `EarlyStopping`, and `ReduceLROnPlateau` callbacks to monitor training, and finally applied fine tunning methods (EfficientNetB5) by freezing BatchNormalization layer with ($\alpha = 0.00001$) learning rate. This process was discussed in algorithm 4.

VGG16: The VGG16 architecture we used in our project was a pre-train Sequential model with the input shape (224, 224, 3) and applied the Lambda function to clearly define the input shape. The VGG16 model was employed with

the top fully connected layers and froze the base VGG16 layers. After this, we included the pre-trained model and `GlobalAveragePooling2D` to replace the flattened layers. Then the fully connected layers with dense having 1500 units with ReLU activation and a dropout (0.4) layer were also provided to prevent the overfitting problem. In the prediction layer, we used 38 units with softmax activation and finally, we compiled the model with Adam optimizer with 0.0001 learning rate and categorical cross-entropy loss. Then we trained the model with 10 epochs for the training and validation set.

InceptionV3: In this study, we used the InceptionV3 base model to predict our results. The input shape of this model was (224, 224, 3) using `imagenet` weights. The first 10 layers of our model were kept frozen, and then custom layers were added on top of the base model. A single layer contains `GlobalAveragePooling2D`, a dense layer with 1024 units along with ReLU activation, and a dropout layer of 0.5 that was used to prevent overfitting. Finally, there was a dense layer having 38 units (38 different classes) with softmax activation which produced the prediction. Then we compiled this model with the Adam optimizer with a learning rate of 0.0001 and categorical cross-entropy loss. We trained this model with only 12 epochs with 32 batch sizes for both training and validation purposes.

V. Result and Discussion

After training all the models on the New Plant Disease Dataset, we got the validation accuracy based on various evaluation parameters. To find the model performance, we analyzed some matrices of the model which demonstrate the result using precision, recall, and F1 score. The resultant outcome offered high precision values indicating an accurate prediction of the true positive rate for each class by minimizing false positive outcomes. Similarly, recall predictions gave the best result by anticipating true positive values. An elevated precision and recall level was used to calculate the F1 score. Figure 6 describes the accuracy and loss for our CNN model, and Figure 7 shows the ROC curve on the left and the precision-recall (PR) Curve on the right. The ROC curve in Figure 7, carried maximum true positive rate (AUC=1.00) for all individual classes. Respectively, in the precision-recall curve, the AUC score is greater than 0.99, but in a few instances, some classes (Tomato late blight, Tomato early blight, Tomato target spot, etc.) have comparatively lower AUC scores of around 0.95.

TABLE II: Performance Comparison Of The Deep Learning Models

Model Name	Training Accuracy	Validation Accuracy	F1 Score
Custom CNN	99.44	99.23	99.34
EfficientNetB5	99.30	99.15	99.14
VGG16	99.48	97.50	97.00
InceptionV3	99.81	99.00	99.00

The custom CNN model showed the highest validation accuracy compared to the other deep learning models. From Table II, we can spot the CNNs model producing the best

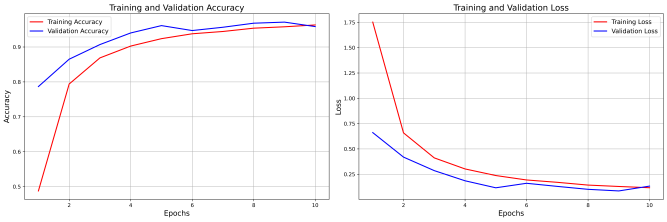


Fig. 6: Accuracy and loss for custom CNN model

outcome out of all the other models. The training and validation accuracy we achieved for this model is 99.40% and 99.23%, that's a quite decent score. In the case of another deep learning model, InceptionV3 has an accuracy of 99%, VGG16 has 97.50% and EfficientNetB5 has 99.15%.

We also integrated Grad-CAM into our custom CNN model to visualize the infected region of the leaf. Figure 8 shows the

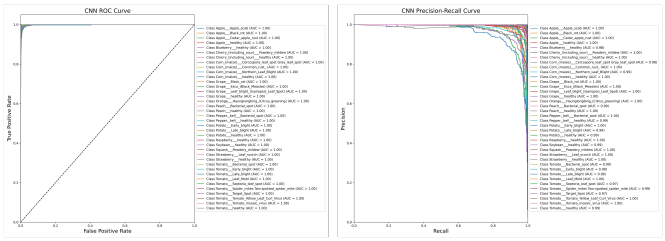


Fig. 7: ROC Curve (Left) and PR Curve (Right) of CNN model

heatmap and Grad-CAM applied to the image. In Figure 8, the intensity of the yellow color gradually increases, with the diseased area of a leaf highlighting the highest intensity. In the

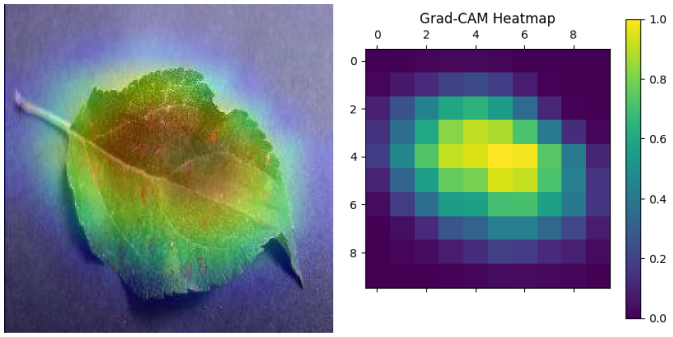


Fig. 8: Grad-CAM Visualization Using CNN Model

Grad-CAM section, the last convolution layer from the model is used to show the infected area of the leaf. It analyzes the input image, predicts the class, and then overlays the heatmap over the original image to improve clarity in plant disease classification [17].

Here, a comparison between the proposed model and different existing works is shown in Table III.

From this table, the proposed system is compared with other baseline studies. The majority of the research works in this table used the CNN model including other traditional models. For example, Hirani *et al.* [8], Jasim *et al.* [9] and

TABLE III: COMPARISON WITH BASELINE STUDIES

Author	Year	Model	Accuracy (%)
Ahmed et al. [1]	2021	CNN	94.00 %
Gurusamy et al. [5]	2024	CNN	98.1%
Hema et al. [7]	2021	ResNet34, VGG16	97.77%, 97.58%
Hirani et al. [8]	2021	Custom CNN, InceptionV3	95.56 %
Jasim et al. [9]	2020	CNN	98.029%
Panchal et al. [14]	2019	K-mean Clustering, RF classifier,HSV	98%
Our proposed model	2024	Custom CNN	99.23%

Ahmed *et al* [1] used the CNN model and achieved 95.56%, 98.029% and 94.00% respectively. It shows that our proposed model outperformed the listed benchmark studies, indicating our systems' effectiveness and robustness. Besides, we also implemented Grad-CAM in our study so that it can enhance the models' explainability. However, none of the studies listed in the table applied XAI techniques.

VI. CONCLUSION

In the field of agriculture, crop disease poses a great threat to the minimal landowners, and it becomes even more dreadful because of the limited resources and utilities for early detection of the diseases. This study proposes an integrated strategy for plant leaf disease detection that employs a custom convolutional neural network with Explainable AI techniques, especially Grad-CAM, for enhanced visualization of affected areas on leaf surfaces. We analyzed the performance of multiple deep learning models, including InceptionNet, VGG16, and EfficientNet, to improve detection accuracy and adaptability. Moreover, integrating IoT devices using microcontroller systems enables real-time classification and forecasting, which maybe applied in the field to help farmers at the grassroots level. This technology has the potential to significantly improve agricultural activities by early disease prediction, cost-effectiveness, minimizing crop loss, and maximizing overall production.

For future enhancement, we are planning to explore hybrid modeling techniques by creating ensemble models that include multiple deep-learning models or applying traditional machine-learning algorithms. The hybrid methods could boost the performance of the classification system, resulting in more accurate and reliable detection in real-time scenarios.

REFERENCES

[1] Ahmed Abdelmoamen Ahmed and Gopireddy Harshavardhan Reddy. A mobile-based system for detecting plant leaf diseases using deep learning. *AgriEngineering*, 3(3):478–493, 2021.

[2] Ümit Atila, Murat Uçar, Kemal Akyol, and Emine Uçar. Plant leaf disease classification using efficientnet deep learning model. *Ecological Informatics*, 61:101182, 2021.

[3] Samir Bhattarai. New plant diseases dataset. https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset/data, 2018. Accessed: 2024-10-05.

[4] Wen-Liang Chen, Yi-Bing Lin, Fung-Ling Ng, Chun-You Liu, and Yun-Wei Lin. Ricetalk: Rice blast detection using internet of things and artificial intelligence technologies. *IEEE Internet of Things Journal*, 7(2):1001–1010, 2020.

[5] Sriram Gurusamy, B Natarajan, R Bhuvaneswari, and M Arvindhan. Potato plant leaf diseases detection and identification using convolutional neural networks. In *Artificial Intelligence, Blockchain, Computing and Security Volume 1*, pages 160–165. CRC Press, 2024.

[6] Bahaa S Hamed, Mahmoud M Hussein, and Afaf M Mousa. Plant disease detection using deep learning. *International Journal of Intelligent Systems and Applications*, 15:38–50, 2023.

[7] MS Hema, Niteesha Sharma, Y Sowjanya, Ch Santoshini, R Sri Durga, and V Akhila. Plant disease prediction using convolutional neural network. *EMITTER International Journal of Engineering Technology*, 9(2):283–293, 2021.

[8] Ebrahim Hirani, Varun Magotra, Jainam Jain, and Pramod Bide. Plant disease detection using deep learning. In *2021 6th International Conference for Convergence in Technology (I2CT)*, pages 1–4. IEEE, 2021.

[9] Marwan Adnan Jasim and Jamal Mustafa Al-Tuwaijari. Plant leaf diseases detection and classification using image processing and deep learning techniques. In *2020 International Conference on Computer Science and Software Engineering (CSASE)*, pages 259–265. IEEE, 2020.

[10] Peng Jiang, Yuehan Chen, Bin Liu, Dongjian He, and Chunquan Liang. Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks. *Ieee Access*, 7:59069–59080, 2019.

[11] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[12] Hicham Moujahid, Bouchaib Cherradi, Mohammed Al-Sarem, Lhoussain Bahatti, Abou Bakr Assedik Mohammed Yahya Eljialy, Abdullah Alsaeedi, and Faisal Saeed. Combining cnn and grad-cam for covid-19 disease prediction and visual explanation. *Intelligent Automation & Soft Computing*, 32(2), 2022.

[13] Kennedy Okokpujie, Imhade P Okokpujie, Fortune T Young, and Roselyn E Subair. Development of an affordable real-time iot-based surveillance system using esp32 and twilio api. *Journal homepage: http://iieta. org/journals/ijsse*, 13(6):1069–1075, 2023.

[14] Poojan Panchal, Vignesh Charan Raman, and Shamla Mantri. Plant diseases detection and classification using machine learning models. In *2019 4th international conference on computational systems and information technology for sustainable solution (CSITSS)*, pages 1–6. IEEE, 2019.

[15] Ramaprasad Poojary and Akul Pai. Comparative study of model optimization techniques in fine-tuned cnn models. In *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, pages 1–4. IEEE, 2019.

[16] RB Salikhov, V Kh Abdrakhmanov, and IN Safargalin. Internet of things (iot) security alarms on esp32-cam. In *Journal of Physics: Conference Series*, volume 2096, page 012109. IOP Publishing, 2021.

[17] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[18] Gauhar Zareen Shaikh, Sangeeta Kamite, Aniket Gunvant Khandekar, Aniket Govind Kandangire, and Tukaram Khillare. Arduino based security system using esp32 camera microcontroller with telegram notifications. *INTERNATIONAL JOURNAL OF ENGINEERING DEVELOPMENT AND RESEARCH*, 11(2):1–9, 2023.

[19] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27*, pages 270–279. Springer, 2018.

[20] Aravindhan Venkataramanan, Deepak Kumar P Honakeri, and Pooja Agarwal. Plant disease detection and classification using deep neural networks. *Int. J. Comput. Sci. Eng*, 11(9):40–46, 2019.