Crop Design 4 (2025) 100092

Contents lists available at ScienceDirect

Crop Design

journal homepage: www.journals.elsevier.com/crop-design

Research Article

A real time monitoring system for accurate plant leaves disease detection using deep learning

Kazi Naimur Rahman *, Sajal Chandra Banik, Raihan Islam, Arafath Al Fahim

Chittagong University of Engineering & Technology, CUET, 4349, Chittagong, Bangladesh

ARTICLE INFO

Keywords:
Plant disease detection
Convolutional neural network
Deep learning model
Computer vision
Performance evaluation
Image processing
Plant pathogen

ABSTRACT

Accurate and timely detection of plant diseases is crucial for sustainable agriculture and food security. This research presents a real-time monitoring system utilizing deep learning techniques to detect diseases in plant leaves with high accuracy. We combined several plant datasets, including the PlantVillage Dataset, resulting in a comprehensive dataset of 30,945 images across eight plant types (potato, tomato, pepper bell, apple, corn, grape, peach, and rice) and 35 disease classes. Initially, a custom Convolutional Neural Network (CNN) model was developed, achieving a leaf classification accuracy of 95.62 %. Subsequently, the dataset was partitioned for individual plant disease detection, applying nine different CNN models (custom CNN, VGG16, VGG19, InceptionV3, MobileNet, DenseNet121, Xception, and two hybrid models) to each plant type. The highest accuracy rates for disease detection were: 100 % for potato (custom CNN), 98 % for tomato (InceptionV3, custom CNN, VGG16), 100 % for pepper bell (MobileNet, custom CNN), 100 % for apple (MobileNet, Xception), 98 % for corn (custom CNN), 99 % for grape (custom CNN, VGG19, DenseNet121), 100 % for peach (VGG16, custom CNN), and 98 % for rice (DenseNet121). A web and mobile application were developed based on the best-performing models, allowing users to insert or capture images of plant leaves, detect diseases, and receive treatment suggestions with high confidence levels. The results demonstrate the effectiveness of deep learning models in accurately identifying plant diseases, offering a valuable tool for enhancing disease management and crop yields.

1. Introduction

Plant disease affecting leaves, stems, roots, and fruits result in diminished crop quality and quantity, leading to global food deprivation and insecurity [1]. Globally, crop diseases contribute to an estimated annual yield loss of approximately 16 %, serving as a primary factor behind both food shortages and escalated production costs [2]. According to the Food and Agriculture Organization (FAO), the world's population will reach approximately 9.1 billion by 2050. For a steady food supply, about 70 % of food production growth is required [3]. Diseases and disorders occur in plants and their products. Biotic factors include diseases caused by algae, fungi or bacteria and abiotic factors inducing disorders comprise rainfall moisture, temperature and nutrient lack [4]. Crop disease is actually a major crisis in day-to-day life.

1.1. Problem statement

Identifying damage detection — typically, the knowledge of farmer and observation was widely used to diagnose plant leaves for injury such as potato leaves, tomato leaves or pepper-bell leaves. In turn, this method raises a lot of crucial questions.

1. **Inconsistency and Unreliability:** Subjective visual examination by farmers can lead to inconsistent and unreliable diagnoses. Different levels of expertise and observation skills may translate to an under-diagnosed diseases or missed early stages.
2. **Time-Consuming Process:** While manual crop checking is both time-consuming and labor-intensive, rendering it ineffective for large-scale farming enterprises. Such a delay in diagnosis may ultimately prevent the early intervention, thus progresses of diseases.
3. **Limited Scalability:** Traditional methods are not scalable because Figure if they would have to use large agriculture fields at places. The widespread disease outbreaks result from losing control of epidemic behavior over large crop areas.
4. **Economic Impact:** Failure to identify the plants having infections at early stages or defects has a major part in lower crop yield and quality, which eventually causes economic losses. More thorough

* Corresponding author.
E-mail address: naimur@cuet.ac.bd (K.N. Rahman).

https://doi.org/10.1016/j.cropd.2024.100092
Received 25 July 2024; Received in revised form 7 September 2024; Accepted 18 October 2024
Available online 6 January 2025

treatments, or crop replacements could lead to farmers incurring higher production expenses.

5. **Food Security Risks:** Because as the world's population keeps growing and is likely to exceed 9.1 billion by 2050, food security becomes more relevant than ever before. Inability to control plant diseases properly impairs food supply chains, extends poverty and malnutrition. Disease management is key in meeting increasing food demand.
6. **Environmental Concerns:** If a misdiagnosis or late diagnosis is made due to overrelying on chemical treatments hoping it will solve the problems, then there will be pesticide and fungicide raining all over your orchard. It not only rises production costs, but it also is posing danger to our environment and health issues due to residues which leads onto soil, water quality.

Hence there is an immediate requirement of automatic, accurate and large scale-based system for identification and control of crop diseases. The procedure would call upon cutting-edge technologies to provide immediate, accurate diagnoses that could in turn help farmers act quickly and precisely. This could confirm for crop protection, prevent economic losses and foster sustainable farming practices.

1.2. Advancement in research: our contribution

This study solves the limits of existing plant disease diagnosis methods by using advanced Convolutional Neural Networks (CNNs) to detect plant illnesses early and accurately. This research makes significant contributions in the following areas.

- **Comprehensive Dataset:** Several plant datasets were combined with the PlantVillage dataset to create an extensive collection of 30,945 images representing various plant species and disease conditions.
- **Model Development:** Multiple CNN architectures (custom CNN, VGG16, VGG19, InceptionV3, MobileNet, DenseNet121) and hybrid models were investigated to find the best successful methods for various plants and diseases.
- **Application Development:** To make use of the trained models, a web application and a mobile app were created, both of which provide real-time disease diagnosis and therapy recommendations via simple interfaces.
- **Performance Optimization:** The models had high accuracy rates across a wide range of plant diseases, demonstrating their efficiency in effectively identifying and diagnosing plant diseases.
- **Deployment:** The models were integrated into scalable TF server-based platforms, resulting in efficient and extensive deployment for real-time agricultural use.
- **Scope Expansion:** The study focuses on eight major plants (potato, tomato, pepper bell, apple, corn, grape, peach, and rice) and 35 disease classes, with the goal of improving agricultural methods, reducing economic losses, and increasing food security.

This study is essential because it will develop technological innovations to transform agricultural operations, which currently has a great urgency for efficient and scalable plant disease detection. The aim here is to build better tools and technology that accurately recognize plant diseases quickly, so farmers can act fast and treat it appropriately. This in turn aims to increase crop health and yields, eliminate chemical treatment requirement, as well as encouraging a 'greener' approach.

2. Related works

Convolutional neural network (CNN) models have become the standard when it comes to tackling image classification problems in recent time. To overcome the difficulty of detecting rice illnesses, Zhou et al. [5] suggested a K-Means clustering technique and a faster R-CNN Fusion approach. Sharma et al. [6] proposed a comprehensive approach that included image pre-processing, k-means clustering, segmentation, and four classifiers; however, there is still a research gap in determining the comparative effectiveness and optimizing the integration for improved leaf disease detection. Sardogan et al. [7] presented a CNN model with the Learning Vector Quantization (LVQ) algorithm for identifying and classifying tomato leaf diseases. Ozguven et al. [8] modernized the Faster R-CNN architecture and adapted CNN model parameters to detect leaf spot diseases in sugar beet.

Lu et al. [9] proposed a CNN-based innovative rice disease identification model with image preprocessing to identify and recognize rice diseases using 500 natural images from rice empirical fields. Kawasaki et al. [10] introduced a novel viral plant disease detection model based on CNN techniques with image pre-processing, utilizing a 4-fold cross-validation approach and achieving higher accuracy with a dataset of 800 cucumber images. Jiang et al. [11] proposed the INAR-SSD model, applying it to the Caffe structure of GoogLeNet Inception structure with Rainbow concatenation on the GPU platform for real-time disease detection in apple leaves using the Apple leaf disease dataset. Islam et al. [12] proposed a segment-based and multi-SVM-based model to detect potato diseases.

Iqbal, M. A. et al. [13] carried out image segmentation on 450 snapshots of healthy and affected potato leaves, obtained from the free Plant dataset. Mangla et al. [14] presented a method for detecting and diagnosing paddy leaf disease through image pre-processing and segmentation techniques coupled with SVM classifiers. Rizqi et al. [15] introduced a methodology leveraging deep learning, employing VGG16 and VGG19 CNN architectures to classify four distinct diseases in potato plants based on leaf conditions. Andrew et al. [16] utilized diverse transfer learning models to forecast distinct plant diseases, employing data augmentation techniques to expand image samples through minor adjustments in image orientation. Azath et al. [17] devised a smart diagnostic system for identifying diseases in cotton plant leaves through the utilization of a CNN. The collection of cotton leaf samples originated from diverse Arab regions, with the dataset categorized into three distinct infected classes and one healthy class of images.

Md. Manowarul Islam et al. [18] examined CNN, VGG-16, VGG-19, and ResNet-50 models on a PlantVillage 10,000 image dataset to detect crop infections, achieving accuracy rates of 98.60 %, 92.39 %, 96.15 %, and 98.98 %, respectively. This study concludes that ResNet-50 has the best accuracy among all with 98.98 % accurate results. Omkar Kulkarni [19] takes a step towards the same by presenting a deep learning model trained over dataset of healthy and infected leaves. Based on defect patterns found in the leaf images using this program, it helps to correctly classify when a crop is unhealthy and aids by preventing over-application of agrochemicals.

Murk Chohan et al. In this context, because of deep learning as it already stated that [20] had trained a model with the name Plant Disease Detector which is for plant diseases detection in terms Leaf pictures. These data augmentation was done with CNN which includes few convolution and pooling layers, the final test accuracy achieved on PlantVillage data was 98.3 %. Mohanty et al. have used 54,306 plant leaf photos as a dataset [21], he showed that deep learning models can be useful for automation at an accuracy of up to 99.35 %. The authors showed the potential of smartphone technology in large-scale crop disease diagnostics and used them to classify 26 diseases.

Shima Ramesh et al. [22] have used Random Forest method based on Histogram of Oriented Gradients (HOG) to extract the features. Additionally, the paper highlights machine learning in identification of diseases across multiple crops. Minah Jung et al. [23] built a practical disease detection model that used the image pairs with and without diseases of plants to predict whether this plant was diseased or not by 5 steps combined type models through CNN algorithm. Experiments demonstrate that their model can achieve a high accuracy of 97.09 % on identifying crop types and diseases, so they plan to extend it from Solanaceae smart farming with other crops together for training data diversity.

Monika Jhuria et al. [25] presented an image processing system to classify fruits and detect diseases, emphasizing the potential of artificial neural networks in agricultural diagnosis. Kaiyi Wang et al. employed new tactics of identifying insect pests and plant diseases by image processing along with computer vision technologies [26]. Besides, there have been some research on diagnosis of rice leaf disease [27], wheat leaf disease [28] and maize plant diseases [29] by using machine learning algorithms as well deep learning techniques. The studies included are selected by our editors for the unique contributions they make and that represent an important focal point in agriculture research worldwide.

2.1. Research gap from previous work

Building on the foundation built by prior research, this work finds numerous key research needs in the field of plant leaf disease detection, as shown in Table 1.

3. Methodology

For plant leaf damage detection, start by finding a nice dataset so that one can easily apply his project. The model can only be allowed to train fully if the dataset chosen comprises maximum types of plant leaves each suffering from a different type of disease. It chooses the dataset after which it undergoes a lot of cleaning and preprocessing steps as shown below. It involves using tools for feature extraction and data visualization to have a better understanding of the dataset properties, so as its features can be prepared properly removing any inconsistencies or other noise. This stage is very important to clean the data which will be fed into the model, so it should have corrected and relevant details.

Then, data augmentation is performed — a crucial stage in artificially enlarging the dataset by generating variations of current photos. This increases the robustness of and in case this one handles a much larger number of events or real-world situations, making it easier to generalize. After data augmentation, a model of Convolutional Neural Network (CNN) is established to effectively classify the extensive range of plant leaves disease. This CNN model is trained on the preprocessed and augmented dataset where it learns to extract different features of human data signatures for each disease.

After the model is trained, its performance checked by various metrics (accuracy, precision, recall, f1 score etc.). This assessment helps to evaluate the suitability of this model and whether there may still need extra work in some areas.

This flowchart is shown in Fig. 1 which illustrates the whole process of plant leaf disease detection from selecting appropriate datasets to evaluating model performances as well. Once a reasonable model has been built, it is saved and sent to TensorFlow Serving (a server-side component) which runs as a local server that can be run on your laptop. This server is used to support real-time predictions by hosting the model and serving inference for other servers. To use the model in a web application, we will also deploy fastAPI service which would assist us in convenient and quick communication between machine learning services & frontend. The trained plant disease detection model resides in the TensorFlow server, and FastAPI acts as an intermediary between this TensorFlow server providing predictions on a part of input data from web being predicted via the application. This design allows users to communicate with the model through a simple interface and extends sophisticated detection of plant disease, such as some farmers or agricultural experts can do. The integration utilizes FastAPI high-performance, the result is an extremely low-response time providing a great degree of scalability for in-web-application illness detection. The benefit of this connection is twofold: it makes the model more user-friendly, but also ensures users can use its predictive capabilities directly in real time which help to decide promptly regarding agricultural activities.

Table 1
Research gaps table.

Author Name	Crop Types	Techniques Used	Research Gaps
Zhou et al. [5]	Rice Leaves Disease Detection	K-Means, Faster R-CNN Fusion	Focus on rice; lacks multi-crop applicability and real-time implementation
Sharma et al. [6], et Mohanty et al. [22]	Multiple Leaves Disease Detection	Image pre-processing, K-Means, Segmentation, Classifiers, CNN	Needs detailed comparative analysis and optimization of techniques for broader crop application
Sardogan et al. [7]	Tomato Leaves Disease Detection	CNN, LVQ	Specific to tomato; lacks exploration of deployment in real-world farming
Ozguven et al. [8]	Sugar beet Leaves Disease Detection	Modified Faster R-CNN	Limited to single crop. No mention of real-time detection or scalability
Lu et al. [9]	Rice Leaves Disease Detection	CNN, Image Preprocessing	Controlled environment study; real-world applicability under diverse conditions not tested
Kawasaki et al. [10]	Cucumber Leaves Disease Detection	CNN, Cross-validation	Lacks assessment of performance under field conditions
Jiang et al. [11]	Apple Leaves Disease Detection	INAR-SSD, GoogleNet Inception	Focus on a single crop; integration with IoT for real-time tracking not addressed
Islam et al. [12]	Potato Leaves Disease Detection	Segment-based, multi-SVM	Lacks discussion on model's scalability for different crops
Iqbal et al. [13]	Potato Leaves Disease Detection	Image Segmentation	No real-time disease progression tracking
Mangla et al. [14]	Paddy Leaves Disease Detection	SVM, Segmentation	Limited to paddy; lacks real-world testing and adaptability
Rizqi et al. [15]	Potato Leaves Disease Detection	VGG16, VGG19, CNN	Need for more comprehensive data augmentation strategies
Andrew et al. [16], Wang et al. [26]	Multiple Leaves Disease Detection	TL, Data Augmentation, Computer Vision, CNN models	General model; lacks implementation details for real-time field application
Azath et al. [17]	Cotton Leaves Disease Detection	CNN	No large-scale deployment or field condition testing
Kulkarni [19], Chohan et al. [20], Ramesh et al. [21]	Various Leaves Disease Detection	Deep Learning, CNN, Data Augmentation, Random Forest, HOG	Need for detailed analysis on real-world integration and user-friendliness
Jung et al. [23], Sannakki et al. [24], Jhuria et al. [25]	Plant Leaves Disease Detection	AI, Image Processing, Pre-trained Models	Specific techniques not generalized for a wider variety of crops
[27–29]	Rice, Wheat, Various Plants, Maize	KNN, J48, Bayesian Network, Logistic Regression, Machine Learning, Leaf Scanning, Data Processing, KNN, RF, DT, NB, SVM	Data versatility not mentioned properly, not scalable and there are no real-world practical insights. Data standardization is also missing.

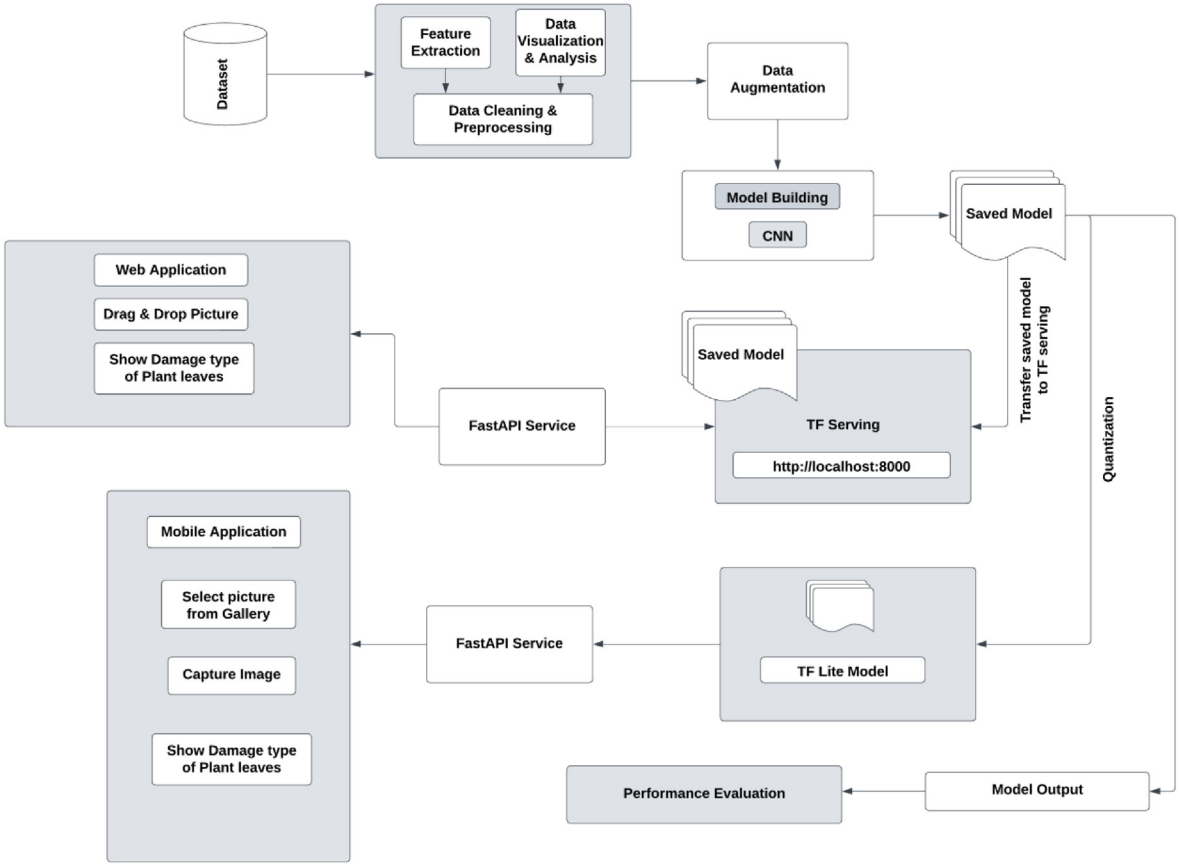


Fig. 1. Flowchart and framework for plant leaves disease Detection using Deep Learning.

The saved model can also be converted to a TensorFlow Lite model using quantization techniques targeting mobile deployment of your models. The FastAPI service is further utilized to deploy the TensorFlow Lite model onto the mobile application for plant disease classification while out in a field. Its wide array of coverages ensures that the plant leaf damage detection system is secure, functional and user-friendly allowing farmers and agricultural experts a new weapon to fight for their plants while keeping your potato in good shape.

3.1. Dataset collection and annotation

The dataset that has been used in this study is derived from PlantVillage, as well new dataset of rice leaf diseases. This combined dataset comprised 30,945 images including eight plant species. All these photos belong to a broad range of plant diseases and healthy cases, which would make it an awesome hybrid dataset, having 35 distinct classes. Classes are organized by plant diseases on a particular crop or the presence of an overall healthiness level for each type of plant encountered in agricultural landscapes.

Photos were annotated with great care, to help build the plant disease detection models in an accurate manner that we can use to train and test machine learning algorithms for leaf identification system. This can be done using one of several annotation programs, such as VGG Image Annotator and LabelMe Image Annotator. We adopt VGG Image Annotator for the sickness regions localization task in this paper, it enables to detect multiple disease locations within one image. Such the detailed annotation helps the model to identify and differentiate between subtle disease manifestations. Table 2 represents distribution of images and corresponding classes among various plant species. Accordingly, we can discover the content of each plant class and disease classes in this dataset.

Table 2
Dataset details with class names.

Plant Name	Number of Classes	Class names
Potato	3	Potato Early Blight, Potato Healthy, Potato Late Blight
Tomato	10	Tomato Early Blight, Tomato Healthy, Tomato Late Blight, Tomato Bacterial Spot, Tomato Leaf Mold, Tomato Mosaic Virus, Tomato Septorial Leaf Spot, Tomato Target Spot, Tomato Spider Mites, Tomato yellow leaf Curl Virus
Pepper-bell	2	Pepper_bell Bacterial Spot, Pepper_bell Healthy
Apple	4	Apple Black Rot, Apple Scab, Apple Healthy, Apple Cedar Rust
Corn	4	Corn Common Rust, Corn Healthy, Corn Northern Leaf Blight, Corn Cercospora Leaf Spot
Grape	4	Grape Black Rot, Grape Esca Black Measles, Grape Isariopsis Leaf Spot, Grape Healthy
Peach	2	Peach Bacterial Spot, Peach Healthy
Rice	6	Rice Brown Spot, Rice Leaf Scald, Rice Narrow Brown Spot, Rice Healthy, Rice Leaf Blast, Rice Bacterial Leaf Blight

3.2. Data cleaning

Feature extraction and analysis extract means the systematic process of cleaning data to be analyzed. Feature extraction includes the selection and transformation of key variables (features) from original data, while Data analysis is concerned with studying these features in depth to acquire some knowledge or patterns about data. This two-pronged strategy aims to make the dataset of higher quality and more relevant for future analytical purposes. This includes dealing with missing values through

imputation or removal, addressing outliers for uniform contribution and encoding the categorical variables.

3.3. Data pre-processing

Pre-processing image datasets, specifically resizing photos and normalizing pixel values, is important before using machine learning algorithms on them. Resizing photos to the same size (a width/height of 256 pixels in this case) which standardizes image dimensions across the dataset. Specifically, uniform block size sizes are crucial to achieve the best results when training a convolutional neural network (CNN) on images. Plus, normalizing pixel values into the value between 0 and 1 if Example when we divide all our pixels by dividing over (255) The purpose of this normalization is to accelerate the speed of convergence for neural network while training, ensuring that input scores have a similar range so as not only stabilize and also enhance how well we can train our model. These processes enhance the quality and efficiency of our dataset, thereby paving way for creation of more accurate, precise models.

3.4. Data augmentation

Data augmentation techniques are essential for data set expansion as well as model robustness in plant leaf damage detection. These methods lead to variable ordinal variation which helps the model generalize better. A random 0°–4'spin would simulate many orientations, while horizontal flipping will flip it to a new orientation. While on the other hand, zooming in emulate different scales, whereas cropping will concentrate to another feature. Changing the brightness factor is intended to capture various lighting conditions and appending noise corrupts runnable codes which effectively boosts performance invariance.

3.5. Dataset splitting & data visualization

The mission of this research study is divided into two primary outlines: Leaf categorization and disease identification. Initially the leaf type must be classified as to whether it belongs to a potato, tomato types or another plant. It strictly splits the dataset into 3 categories, namely training set, test set and validation sets. In general, 80 % for Training and 20 % for test where the other half of testing data is used as validation. This results in the model learning well from most of the data while being validated and tuned on certain subsets. Fig. 2 shows the data visualization for several plant leaf classifications.

The second part is attention to detecting diseases in leaves. It uses a 80–10–10 split ratio of the same dataset. Disease Identification: subdivides the dataset into databases of individual plants. For example, if a leaf is identified as a potato leaf, the dataset specific to potato leaves,

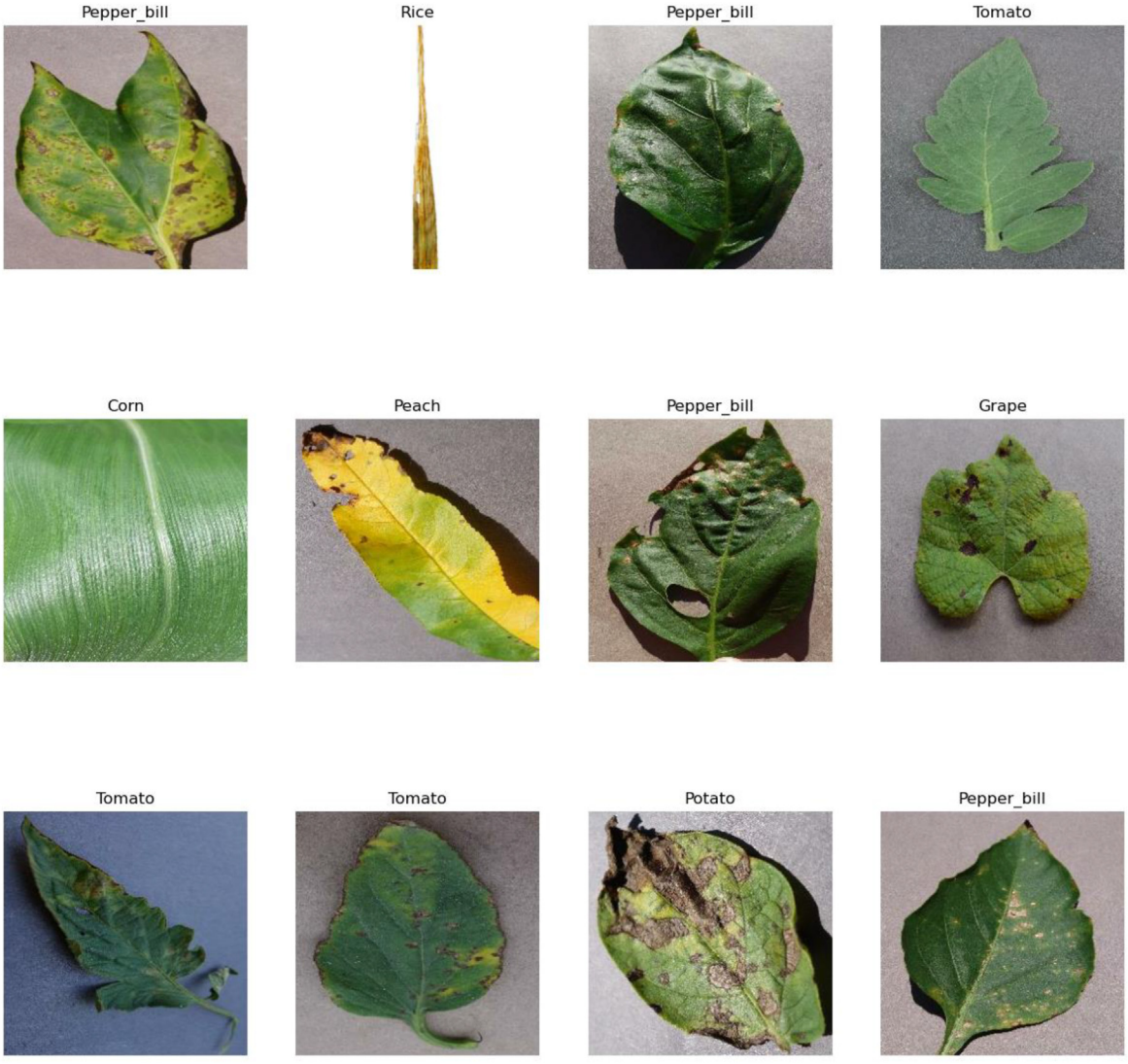


Fig. 2. Data visualization of leaf classification.

which includes three classes of diseases, is utilized. The deep learning model is then trained on this specific dataset to accurately identify which disease is present in the potato leaf. This approach ensures that the model is fine-tuned to the characteristics and disease patterns of each plant type, enhancing its precision in disease detection. Fig. 3 represents the data visualization for Individual plant's multiple categories of leaf disease.

3.6. Model building

Fig. 4 represents the schematic diagram of a basic Convolutional neural network architecture. Every convolutional neural network has some several steps. Firstly, take input images from the dataset. Then some convolutional layers can be added for further steps. This convolutional layer varies because of various CNN model like VGG16, InceptionV3, EfficientNet, MobileNet etc. In every convolutional layer we can use various size matrices of filter.

After using the filter matrices of convolution layer, we can add another layer named pooling layer. Max pooling layer can also be used here for different CNN models. After that CNN model adds a Flatten layer to go further. Then dense layer can be added in CNN model building. After all of these, we can build a perfect CNN model for detecting plant leaves disease detection.

For this research study, we have chosen different CNN algorithms for various plant leaves disease detection like VGG16, VGG19, Custom CNN, InceptionV3, MobileNet, DenseNet121, Xception and two hybrid models. Each model's unique architecture and combination aim to enhance accuracy and performance across different types of plant leaves and diseases. We have made our first hybrid model by concatenating three different CNN algorithms like VGG16, DenseNet121, InceptionV3. Our second hybrid model has been made by concatenating another three CNN algorithms like ResNet50, EfficientNetB0, MobileNetV2.

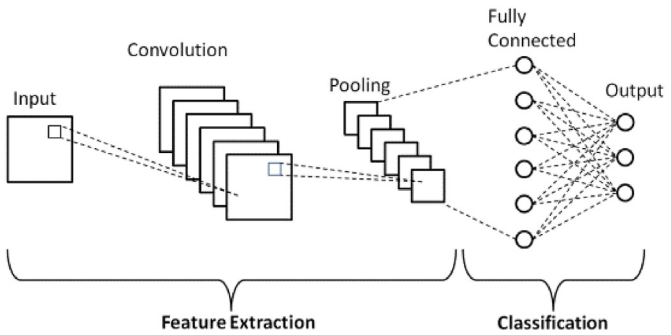


Fig. 4. Schematic Diagram of a basic Convolutional Neural Network (CNN) architecture [30].

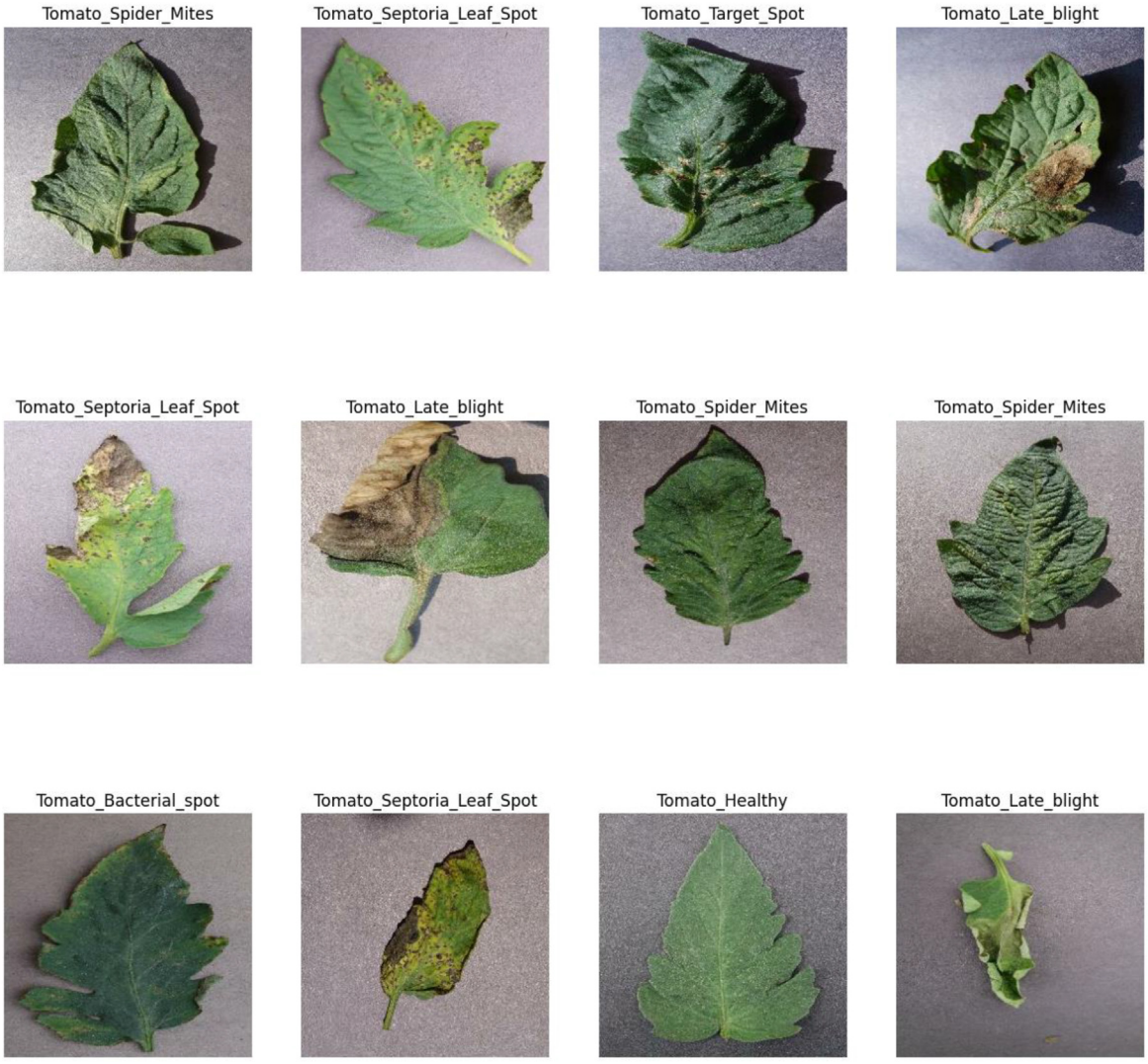


Fig. 3. Data visualization of individual Plant's leaf disease.

3.6.1. Deep learning models

Custom CNN: A custom CNN typically consists of several convolutional layers followed by pooling layers for spatial reduction and feature extraction. The exact configuration can vary widely based on the specific design choices. Commonly used activation function in custom CNNs is ReLU (Rectified Linear Unit) as it adds non-linearity to the model. While the overall number of layers in a custom-made CNN may vary by task difficulty, it typically consists of several convolutional and pooling layers followed by fully connected classification layers. In the custom CNN we used in with 6 conv2D and 5 maxpooling layers. The relu activation method was obviously used in our custom CNN process then we combined it with the SoftMax. We used different kernel sizes and filter matrices to obtain correct disease detection results among a wide variety of plant leaves.

VGG16: VGG16 is a deep convolutional neural network design that pays attention to the simplicity, efficiency of learning and use. The CNN has 16 layers (13 conv and 3 fc). The maxpooling layers are interleaved between the convolutional layers to reduce spatial dimensionality. All of these layers work perfectly with ReLU except for the output layer that uses SoftMax activation function when it comes to Categorical (multi-class) classification. It is well-liked by image classification people because of its highly efficient design and competitive performance on standard datasets.

VGG19: This is a new version of VGG which has 19 layers. It consists of 16 convolutional layers and three fully connected layers. The feature map sizes are reduced using maxpooling layers after every two or three convolutional layers. The remaining layers apply a Rectified Linear Unit (ReLU) activation function, which is key to modeling complex and nonlinear functions. Most of the time, if you have multiple classes (like in multi-class classification tasks), SoftMax activation is used at this final layer to generate class probabilities. VGG19 has gained recognition for its high accuracy as it derives from deep network layering and simple architecture compared to networks like ResNet or Inception.

InceptionV3: The InceptionV3 is a convolutional neural network with 48 layers. Build an architecture with conv layers to extract features, pool layers to shrink spatial dimensions between convolutional blocks and finally fully connected layer for determining output (predictions). In each layer, the main activation function used is ReLU (Rectified Linear Unit) to introduce nonlinearity and help the network learn complex patterns in Image data. This architecture is designed to balance the trade-off between depth and computational efficiency which can be better applied in tasks such as image classification, object recognition etc.

MobileNet: MobileNet is a small extreme computation-efficient convolutional neural network architecture for mobile and embedded vision applications. In all it is a 28-layer deep network, where depth wise separable convolutions have been used which tremendously cuts down parameter count compared to regular convolutional layers. MobileNet has no maxpooling layers as usual and reduces the spatial complexity using depth wise convolutions with stride 1. ReLU (Rectified Linear Unit) is the activation function used in the entire network that makes feature extraction and creates nonlinearity Efficiency and speed are kept in mind while designing this architecture which makes it suitable for real-time applications on less-resourced devices.

DenseNet121: DenseNet121 is a kind of convolutional neural network in which each layer directly accesses the information from all the different layers that are present before it. Totally, it has 121 layers (including convolutional, pooling and fully linked layers) In a usual architecture, the use of maxpooling layers is replaced here by densely connected blocks where each layer takes as input all preceding feature maps and gives its output to subsequent convolutional blocks with concatenation. This method strengthens the feature propagation and enhances the feature learning, which in turn improves model performance. However, in this implementation they have used ReLU which often works best for the activation function and allows us to learn 2D patterns with dense inputs.

Xception: Xception is a deep and complex convolutional neural network architecture. It consists of 71 layers, in which there are both convolution layers and depth wise separable convolution layers as well. Regarding the fact that Xception has depth wise separable convolutions, as opposed to standard CNNs, which learn spatial and channel-wise correlations. This architecture often aims to capture the fine-grained detail in data while still being computationally efficient. Xception utilizes linear activation functions for its convolutional layers and uses ReLU (Rectified Linear Unit) activation in other parts of the network to bring nonlinearity necessary for learning features, classifying objects.

Hybrid Models: The first hybrid model is described in the below Fig. 5, where VGG16+InceptionV3+DenseNet121 architectures combine to serve as a single trained neural network. It uses these networks to perform the task better.

Based on what architecture is used, the total number of layers in that hybrid model varies but most often it goes beyond 100+ layer. Maxpooling layers are added to each component architecture to decrease the spatial dimensions of feature maps increase model efficiency and performance. The second hybrid model also combines EfficientNetB0, ResNet50, and MobileV2 with ReLU activation throughout the network as shown in Fig. 6. These activation functions help to introduce nonlinearity which is crucial in training deep neural networks and performing better on classification tasks. In this paper, the hybrid model for plant disease diagnosis is expected to be capable of achieving a higher accuracy by taking advantage of above-mentioned features in those complex architectures and deploying sophisticated activation functions.

3.6.2. Architectural comparison of various CNN model

Today Convolutional Neural Networks (CNN) has become a basic building block in computer vision, and CNNs have provided the solution for many tasks like Image Classification, Object Detection or Semantic Segmentation. CNN is an evolving project, and several different architectures have been developed, each with their own design philosophy. Table 3 demonstrates an in-depth architectural comparison between various CNN models, illustrating key attributes such as image size, model size, layers count depth-wise and involutional, parameter total. This comparison provides a deep dive into the pros and cons of each architecture, helping you to choose which one is relevant for your application.

3.6.3. Tools and system configuration

This section gives a brief recap of the high-end hardware and software components selected in this research work, highlighting each element role and contribution to the project. The selective nature of this paradigm permits handling massive data, efficient model training, and rapid build-up of rich web + mobile applications. Therefore, the powerful architecture shows in Table 4 is very important to our plant leaf damage detection system because it needs a high quality of performance and reliability. Several kernel crashes are observed when any CNN model is trained on a low-end machine.

The architecture was selected to meet the resource demands of deep learning model training, ensuring hardware wouldn't hold up research progress while making resulting applications more reliable and scalable for further developments.

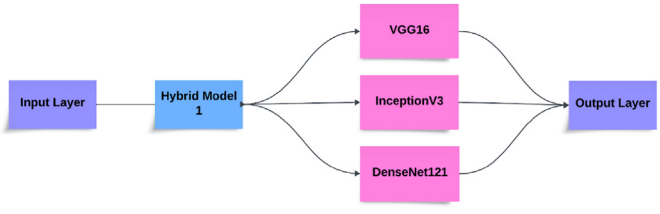


Fig. 5. Hybrid model 1 schematic block.

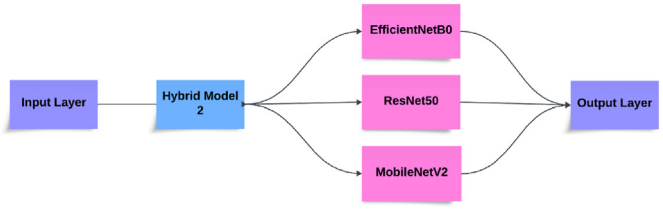


Fig. 6. Hybrid model 2 schematic block.

3.7. Converting model into TensorFlow serving

There are several key steps required to convert a model into something that can be served with TensorFlow Serving such as fast deployment and real-time inference. Once training of the model is done, the first thing to do would be saving the created model using keras i.e. Keras format. The next step is to perform the implementation of TensorFlow Serving, which provides comprehensive and high-performance serving for machine learning models. Especially the operational part, this is recommended to use Docker since its dynamic and very elastic deployment helps at using it.

The saved model is in a directory structure organized by version, with each version at the subdirectory of its base model path. This versioning gives the ability to perform simple updates and rollbacks without downtime. To start the TensorFlow Serving container, we map the base model directory to it and then provide the name of our models. This organization also enables us to provision many versions of the model, facilitating continual innovation by delivery and testing new models.

3.8. TF-lite model conversion process

TFlite model conversion is an essential component to implement machine learning models within devices at the edge, such as mobile phones and IOT sensors with limited processing power. You begin this method with a trained TensorFlow model, which is usually saved in the Saved Model format. Then the TensorFlow Lite model is added to TensorFLow Lite Converter, which is a special tool inside the library itself. There are many optimizations with conversion process, major one is quantization, which reduces the model size and improves inference time as it converts floating-point data to integer representations. Quantization can be done in different ways depending on the desired performance/accuracy trade-off: dynamic range, full integer or float16. After converting the model to a tflite file, the model is tested so that it functions correctly and gives consistent results as of original TensorFlow Model. This involves executing the TFLite model on test data and verifying it against that of original model. After validation, the TFLite model is used with TensorFlow Lite Interpreters and APIs for Android interfaces or iOS interfaced as well as embedded Linux through which it can be integrated into a mobile application or an IoT device in the end, we test it on our target device is achieved of all those performance and accuracy criteria. Further optimizations like model pruning and hardware acceleration could be employed to increase performance. This complete conversion process ensures that the model becomes efficient, lightweight and works well in low-resource devices while maintaining accuracy needed for practical applications.

Table 4
System Configuration and necessary software.

Component	Specification	Role/Description
Graphics Card	Asus ROG Strix GeForce RTX 3090 OC Edition 24 GB	Handles computationally intensive tasks of deep learning models without kernel crash issues
Processor	Intel 11th Gen Core i9-11900K Rocket Lake	Provides high-speed processing power for data handling and multitasking
RAM	Corsair VENGEANCE 16 GB DDR5 6000 MHz	Supports high-speed data processing and multitasking
Motherboard	Asus ROG CROSSHAIR X670E HERO DDR5 AMD AM5 ATX	Ensures optimal compatibility and performance of all components
Storage	Seagate FireCuda 530 2 TB Gen4 M.2 2280 PCIe NVMe Gaming SSD	Offers fast read/write speeds for efficient handling of large datasets
Development Environment	Visual Studio Code	Provides an integrated environment for web and mobile application development
Frontend Framework (Web)	React	Utilized for its component-based architecture and efficient rendering
Frontend Framework (Mobile)	React Native	Enables cross-platform development, ensuring compatibility with both Android and iOS
Backend Framework	FastAPI	Modern web framework used for building high-performance backend services
Database	MongoDB	Scalable and flexible NoSQL database used for storing user records
Interactive Environment	Jupyter Notebook	Used for developing and executing deep learning algorithms
GPU Toolkit	CUDA Toolkit and cuDNN	Facilitates efficient computation by leveraging GPU capabilities

3.9. Development phase

After converting the TensorFlow model to TensorFlow Serving and as well a lightweight format version for mobile called 'TensorFlow Lite', these models were utilized by us to create robust backend API service that allowed users with confidence score. The backend was tested extensively using Postman for accuracy and stability before being weaved together with the frontend components. The backend API was carefully crafted with FastAPI, a modern and fast web framework for building APIs with Python. The front-end of the web application was created with React, a widely employed JavaScript framework lauded for its productivity and expandability. We successfully connected the backend api with our react frontend, so users could interact using drag and drop leaf images. The app will then look over the photos for current symptoms and offer suggestions on how to treat them. In addition, a mobile application was developed using the React Native Expo framework that supports Android and iOS. This mobile application directly connects to the RESTful backend API also made more effective through ngrok for allowing real-time interaction. A secure database connection with MongoDB was established to maintain user data records and for reliable data storage & retrieval. This comprehensive approach with extensive backend testing, frontend integration and cross-platform

Table 3
Architectural Comparison of several CNN models.

	VGG16	VGG19	InceptionV3	MobileNet	DenseNet121	Xception
Image size	256*256*3	256*256*3	256*256*3	256*256*3	256*256*3	256*256*3
Model Size	528 MB	549 MB	92 MB	16 MB	30.5 MB	88 MB
Total Layer	16	19	48	28	121	71
Convolutional Layer	13	16	23	27	117	36
Total Parameters	138.4 million	143.7 million	23.9 million	13 million	7.2 million	22.8 million

mobile development proves power of the system. Using the latest frameworks and technologies to detect plant disease, which are beneficial for better user experience as well as make sure that system growing is properly scalable and closed.

4. Experimental results

We have then trained numerous CNN models on our dataset before evaluating it by technical aspects such as accuracy, precision recall and F1 score. We divided our training process into 2 phases: leaf classification and plant level disease detection. Leaf classification was done using our own CNN model that has been trained on a dataset containing 30,945 images of different plants. Due to the dataset's large size, we initially trained the model for 20 epochs, achieving a time/step of 753 s per epoch. This training phase yielded an accuracy of 95.62 % and a loss of 0.1343. To enhance the model's performance, we extended the training to 40 epochs, which resulted in an improved accuracy of 97.28 % and a reduced loss of 0.0982. The batch size for this training was set to 32. Table 5 represents the comparison of custom CNN model for leaf classification on large dataset based on various number of epochs. From that table it can be concluded that more number of epochs causes more accuracy in this case.

4.1. Performance evaluation on individual Plant's disease detection

After the successful leaf classification, the next step involved proceeding to individual plant disease detection. This phase required partitioning the dataset into specific plant disease subsets, which subsequently reduced the time/steps per epoch, thereby shortening the overall model training time. There is eight individual plant's leaves disease dataset which is to be evaluated on various CNN model like VGG16, VGG19, Custom CNN etc. For this research study, we have implemented 7 CNN model algorithms on each plant's leaves and also applied two hybrid models. From all that model's performance evaluation, we have selected the best model for our development purpose.

4.1.1. Potato Plant's performance evaluation

For the classification of potato plant diseases, a meticulously curated dataset comprising a total of 2152 images of potato plant leaves was utilized. This dataset is organized into three distinct classes: potato early blight, potato healthy, and potato late blight, as indicated in Table 2. Specifically, the dataset consists of 1000 images of potato leaves affected by early blight, 152 images of healthy potato leaves, and an additional 1000 images depicting late blight symptoms. The performance of various CNN models for classifying potato leaf diseases is summarized in Table 6, which includes metrics such as accuracy, precision, recall, and F1-score. From this analysis, the custom CNN model emerged as the best performer, demonstrating superior accuracy and robust metrics across the board.

4.1.2. Tomato Plant's performance evaluation

For the classification of tomato plant diseases, a meticulously curated dataset comprising a total of 13,226 images of tomato plant leaves was utilized with 10 classes mentioned in Table 2. Specifically, the dataset consists of 2127 images of tomato leaves affected by bacterial spot, 1000 images of tomato early blight, 1592 images of tomato healthy, 1909 images of tomato late blight, 952 images of tomato leaf mold, 373 images of tomato mosaic virus, 1771 images of tomato septoria leaf spot, 1676 images of tomato spider mites, 1404 images of tomato target spot and an additional 422 images depicting tomato yellow leaf curl virus symptoms. From Table 7, best model Xception was selected for tomato plant leaves disease detection.

Table 6
Performance evaluation for potato plants on various CNN models.

CNN Models	Accuracy	Precision	Recall	F1-score
Custom CNN	100 %	0.9967	0.9967	1.0000
VGG16	97 %	0.95	0.9633	0.9533
VGG19	97 %	0.9467	0.9467	0.9467
InceprionV3	97 %	0.98	0.93	0.9533
MobileNet	98 %	0.9733	0.9633	0.97
DenseNet121	98 %	0.99	0.96	0.9733
Xception	96 %	0.9633	0.9233	0.9433
Hybrid Model 1	98.84 %	0.9884	0.9884	0.9880
Hybrid Model 2	99.53 %	0.9953	0.9953	0.9951

4.1.3. Pepper bell Plant's performance evaluation

For the classification of pepper bell plant diseases, a well-prepared dataset was utilized, consisting of a total of 2475 images of pepper bell plant leaves, encompassing 2 classes as detailed in Table 2. Specifically, the dataset includes 997 images of pepper bell leaves afflicted by bacterial spot and 1478 images of healthy pepper bell leaves. From Table 8, best model custom CNN was selected for pepper bell plant leaves disease detection.

4.1.4. Apple Plant's performance evaluation

For the classification of apple leaf diseases, a meticulously curated dataset comprising 2536 images of apple leaves was employed, featuring 4 distinct classes as delineated in Table 2. Specifically, this dataset includes 496 images of apple leaves afflicted by black rot, 220 images exhibiting cedar rust, 1316 images of healthy apple leaves, and 504 images showing symptoms of apple scab. From Table 9, best model Xception was selected for apple plant leaves disease detection.

4.1.5. Corn Plant's performance evaluation

For the classification of corn leaf diseases, a meticulously curated dataset comprising 3080 images of corn leaves was employed, featuring 4 distinct classes as outlined in Table 2. Specifically, this dataset includes 410 images of corn leaves afflicted by cercospora leaf spot, 953 images exhibiting common rust, 929 images of healthy corn leaves, and 788 images displaying symptoms of northern leaf blight. From the analysis of Table 10, we have selected the custom CNN model as the best model for future prediction in real time.

Table 7
Performance evaluation for tomato plants on various CNN models.

CNN Models	Accuracy	Precision	Recall	F1-score
Custom CNN	98 %	0.9820	0.9750	0.9790
VGG16	98 %	0.9800	0.9810	0.9800
VGG19	97 %	0.9750	0.9710	0.9720
InceprionV3	98 %	0.9780	0.9770	0.9800
MobileNet	96 %	0.9620	0.9510	0.9560
DenseNet121	97 %	0.9710	0.9670	0.9670
Xception	98 %	0.9850	0.9800	0.9820
Hybrid Model 1	96.02 %	0.9602	0.9600	0.9598
Hybrid Model 2	97.03 %	0.9699	0.9710	0.9657

Table 5
Comparison of model based on number of epochs.

Number of Epochs	Accuracy	Loss	Batch Size	Time/steps per epoch	Precision	Recall	F1-score
20	95.62 %	0.1343	32	753s	0.9578	0.9522	0.9643
40	97.28 %	0.0982	32	738s	0.9739	0.9687	0.9837

Table 8
Performance Evaluation for Pepper bell Plants on Various CNN Models.

CNN Models	Accuracy	Precision	Recall	F1-score
Custom CNN	100 %	1.0000	1.0000	1.0000
VGG16	98 %	0.9750	0.9700	0.9700
VGG19	95 %	0.9600	0.9450	0.9500
InceprionV3	99 %	0.9850	0.9900	0.9850
MobileNet	100 %	1.0000	1.0000	1.0000
DenseNet121	92 %	0.9200	0.9200	0.9150
Xception	93 %	0.9400	0.9300	0.9300
Hybrid Model 1	99.60 %	0.9960	0.9960	0.9956
Hybrid Model 2	97.57 %	0.9757	0.9757	0.9760

Table 9
Performance evaluation for apple plants on various CNN models.

CNN Models	Accuracy	Precision	Recall	F1-score
Custom CNN	84 %	0.8400	0.8125	0.8225
VGG16	97 %	0.9800	0.9500	0.9625
VGG19	98 %	0.9825	0.9600	0.9725
InceprionV3	98 %	0.9900	0.9700	0.9800
MobileNet	100 %	0.9975	0.9925	0.9975
DenseNet121	95 %	0.9625	0.9325	0.9425
Xception	100 %	1.0000	1.0000	1.0000
Hybrid Model 1	99.80 %	0.9980	0.9980	0.9976
Hybrid Model 2	99.60 %	0.9960	0.9960	0.9957

Table 10
Performance evaluation for corn plants on various CNN models.

CNN Models	Accuracy	Precision	Recall	F1-score
Custom CNN	98 %	0.9675	0.9700	0.9700
VGG16	96 %	0.9450	0.9450	0.9450
VGG19	95 %	0.9300	0.9325	0.9325
InceprionV3	92 %	0.8950	0.8900	0.8925
MobileNet	95 %	0.9300	0.9325	0.9325
DenseNet121	94 %	0.9225	0.9250	0.9250
Xception	97 %	0.9525	0.9675	0.9600
Hybrid Model 1	96.74 %	0.9674	0.9674	0.9670
Hybrid Model 2	95.28 %	0.9528	0.9528	0.9525

4.1.6. Grape Plant's performance evaluation

For the classification of grape leaf diseases, a meticulously curated dataset comprising 3251 images of grape leaves was utilized, featuring 4 distinct classes as outlined in Table 2. Specifically, this dataset includes 944 images of grape leaves afflicted by black rot, 1107 images exhibiting symptoms of esca black measles, 339 images of healthy grape leaves, and 861 images showing signs of isariopsis leaf spot. From Table 11, best model custom CNN was selected for apple plant leaves disease detection.

4.1.7. Peach Plant's performance evaluation

To classify peach leaf illnesses, a rigorously curated dataset of 2126 photos of peach leaves was used, with two unique classifications as shown in Table 2. This dataset contains 1838 photos of peach leaves with bacterial spots and 288 images of healthy peach leaves. This large dataset was critical in training the algorithm to effectively distinguish between healthy leaves and those with bacterial spot, ensuring precise and consistent disease diagnosis. Table 12 shows that the best model for detecting illness in peach plant leaves is VGG16.

4.1.8. Rice Plant's performance evaluation

A rigorously curated collection of 2100 rice leaf pictures was used for disease classification, with six unique classifications as shown in Table 2. This dataset includes 350 photos of rice leaves infected with bacterial leaf blight, brown spot, healthy leaves, leaf blast, leaf scald, and narrow brown spot. This comprehensive dataset played a crucial role in training the model to accurately differentiate between different diseases affecting rice leaves, ensuring robust and accurate disease detection capabilities. Best model DenseNet121 was selected for rice plant leaves disease detection from the analysis of Table 13.

4.2. Optimal hyperparameters selection for CNN architecture

In deep learning, hyperparameters are crucial settings that influence how a model is trained and its ultimate performance. These include the learning rate, batch size, number of epochs, and network architecture details like the number of layers and nodes per layer. Unlike model parameters, hyperparameters are not learned during training; they must be set beforehand. Finding the optimal hyperparameters often involves a trial-and-error process, as their values significantly affect the model's accuracy, convergence speed, and ability to generalize. Table 14 represents the optimal hyperparameters for our proposed CNN architectures of various plant leaves according to our research.

4.3. Confusion matrix evaluation of plant diseases

As we have used multiple models for different plant leaves disease, we have chosen best models according to accuracy, precision, f1-score and recall. These features have been achieved according to the confusion matrix of various plant leaves based on test dataset. Fig. 7 represents the best model's confusion matrices of various plant disease.

4.4. Analysis of best Model's training & validation accuracy, loss curves

According to Fig. 8, we have seen best model's Accuracy Vs Epochs Curve and Loss Vs Epochs Curve of various plant leaves diseases. From that figure, we can conclude that accuracy increases over epoch and loss simultaneously decreases over epochs.

4.5. Overfitting issue solution

The concern of overfitting with large models like VGG16, DenseNet121, and hybrid models was addressed in my research which was mitigated through several strategies to ensure robust performance across plant species and diseases. Key strategies included extensive data augmentation (e.g., random rotations, flipping, and scaling) to enhance dataset diversity, regularization techniques like L2 regularization and Dropout to reduce model complexity, and k-fold cross-validation to

Table 11
Performance evaluation for grape plants on various CNN models.

CNN Models	Accuracy	Precision	Recall	F1-score
Custom CNN	99 %	0.9950	0.9925	0.9925
VGG16	93 %	0.9225	0.9300	0.9200
VGG19	99 %	0.9925	0.9900	0.9900
InceprionV3	93 %	0.9150	0.9425	0.9250
MobileNet	96 %	0.9650	0.9650	0.9650
DenseNet121	99 %	0.9900	0.9900	0.9900
Xception	98 %	0.9875	0.9850	0.9850
Hybrid Model 1	98.46 %	0.9846	0.9846	0.9845
Hybrid Model 2	98.30 %	0.9830	0.9830	0.9832

Table 12
Performance evaluation for peach plants on various CNN models.

CNN Models	Accuracy	Precision	Recall	F1-score
Custom CNN	100 %	1.0000	0.9850	0.9900
VGG16	100 %	0.9850	1.0000	0.9900
VGG19	99 %	0.9800	0.9750	0.9750
InceprionV3	97 %	0.9400	0.9450	0.9400
MobileNet	99 %	0.9750	0.9950	0.9850
DenseNet121	99 %	0.9750	0.9950	0.9850
Xception	99 %	0.9450	0.9950	0.9650
Hybrid Model 1	99.76 %	0.9976	0.9976	0.9972
Hybrid Model 2	99.76 %	0.9976	0.9976	0.9969

Table 13
Performance evaluation for rice plants on various CNN models.

CNN Models	Accuracy	Precision	Recall	F1-score
Custom CNN	90 %	0.9117	0.9033	0.9050
VGG16	91 %	0.9200	0.9100	0.9083
VGG19	95 %	0.9533	0.9433	0.9433
InceprionV3	91 %	0.9150	0.9067	0.9100
MobileNet	93 %	0.9200	0.9300	0.9217
DenseNet121	98 %	0.9833	0.9850	0.9850
Xception	94 %	0.9283	0.9267	0.9283
Hybrid Model 1	92.86 %	0.9286	0.9286	0.9279
Hybrid Model 2	88.57 %	0.8857	0.8857	0.8859

Table 14
Hyperparameters selected for the proposed CNN architecture.

Hyperparameters	Optimal Hyperparameter
Learning Rate	0.0001
Batch Size	32 (potato, pepper bell, apple, corn, grape, peach, rice) 16 (tomato)
Epochs	50 (potato, pepper bell, apple, corn, grape, peach) 100 (tomato, rice)
Number of Dense Layers	2
Number of Nodes in Dense Layers	64 (potato, pepper bell, corn, grape) 512 (tomato, apple, peach, rice)
Activation Function	SoftMax

assess model performance across different data subsets. Additionally, batch normalization helped stabilize training, while transfer learning from pre-trained models like ImageNet improved generalization. Early stopping was also implemented to halt training when validation performance plateaued, preventing overfitting. Together, these techniques effectively controlled overfitting, enabling large models to generalize well across different plant disease categories, as demonstrated by the high accuracy achieved in both training and validation phases.

4.6. Reason for proposing multiple models in our research

In our research, multiple models were proposed for the following reasons.

- Comprehensive Performance Evaluation: Different models have unique strengths in feature extraction, computational efficiency, and generalizability. Evaluating several models helped identify the most robust architecture for diverse plant species and diseases.
- Optimization for Deployment: Models like MobileNet are lightweight and ideal for mobile devices, while more complex models like DenseNet121 and InceptionV3 provide enhanced feature extraction. This allows adaptability for various platforms (web, mobile).
- Generalizability and Robustness: Using multiple models ensures the system performs well under different real-world conditions, such as lighting and image quality variations, improving overall robustness.
- Facilitating Future Research: The comparative analysis provides insights for future studies, offering a foundation for further refinement of model architectures or development of hybrid approaches.
- Addressing Plant and Disease Variations: Certain models excel for specific plants or diseases. By testing multiple models, we ensured higher accuracy and reliability across a range of agricultural applications.

This strategic approach ensures flexibility, scalability, and the highest possible accuracy in plant disease detection.

4.7. Backend API and database setup for user records

In this research study, we have developed the backend service for the plant disease detection system utilizing FastAPI and MongoDB. We structured the project architecture to separate models, schemas, database connections, and CRUD operations, ensuring a clean and maintainable codebase. FastAPI was chosen for its performance and ease of use, while MongoDB provided a scalable NoSQL database solution. To manage user authentication, we implemented a system requiring users to provide a username, email, and password during registration. Using custom utility functions imported from 'app.utils', passwords were hashed before storage to ensure security. The MongoDB driver, motor, facilitated efficient database interactions. Pydantic models were used to validate and serialize data, maintaining data integrity. This robust backend setup enabled secure storage of user credentials with hashed passwords, providing a reliable foundation for integrating the plant disease detection models with web and mobile applications. We have selected the database name as Plant_manage. Inside it, we have created a directory named users for keeping user records. Fig. 9 represent the database of users.

4.8. Web Application Development

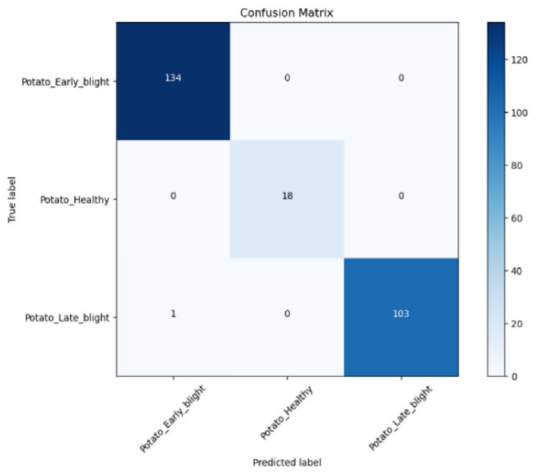
For the development of the web application, we utilized the React framework for its robust frontend capabilities. The web application was designed with a user-friendly interface featuring a drag-and-drop functionality, allowing users to easily upload images of plant leaves. Upon image upload, the application interfaces with our backend service to detect and identify the disease present in the leaf. Fig. 10 represents the homepage graphical user interface of web application where eight several plants portion exists. If upload image button is clicked, then user can go to the individual plant's portion to upload their plant image.

The application displays the disease name along with the confidence level of the detection, providing users with an accurate assessment. Additionally, it offers detailed instructions for disease management and treatment. This comprehensive approach not only aids in the rapid identification of plant diseases but also empowers users with actionable information to address the detected issues, thereby enhancing the overall utility of the web application in agricultural disease management.

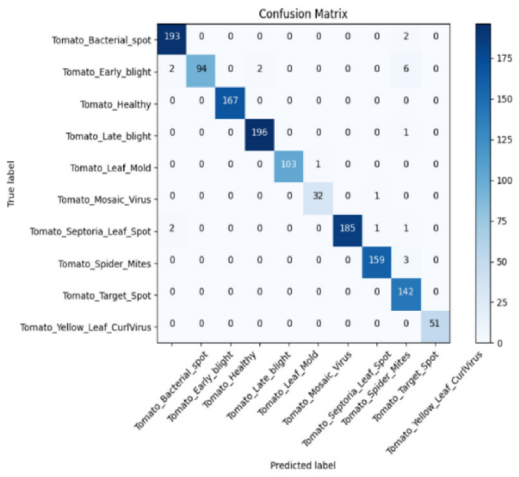
4.9. Mobile Application Development

For the development of the mobile application, we utilized the React Native framework, a cross-platform JavaScript framework designed to enable seamless operation on both Android and iOS devices. This method offers broad accessibility and a uniform user experience across multiple platforms. User Registration and Login, which is essential to have secure authentication took place in our database in back-end side inside the mobile application. Users need to created account and have successfully logged in using application functionalities. Once logged in, users can upload images of plant leaves to identify the invading diseases. Leaf classification is performed when a user submits the image via backend API service. Following that, an additional API request is immediately launched to detect specific diseases affecting plant leaves. This two-step process ensures accurate and detailed disease detection.

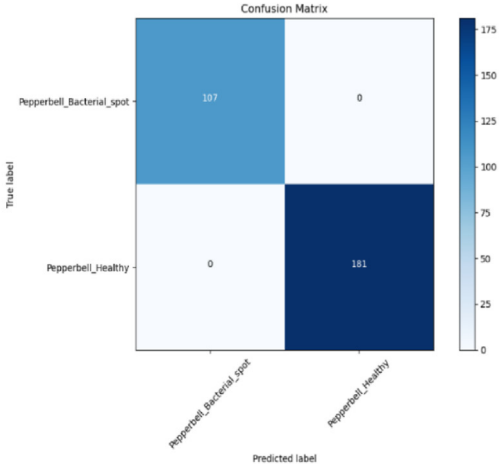
The mobile application's graphical user interface (GUI) is designed for ease of use, providing an intuitive and streamlined user experience. The practical implementation of the mobile application, alongside the web application, is documented in the **appendix** section, showcasing the operational effectiveness and user-friendly nature of our plant disease detection system.



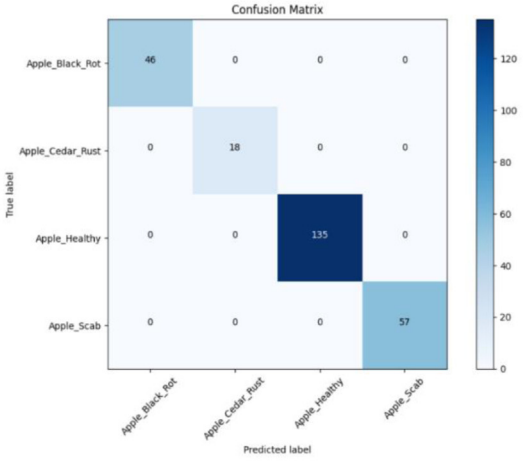
(a) Potato



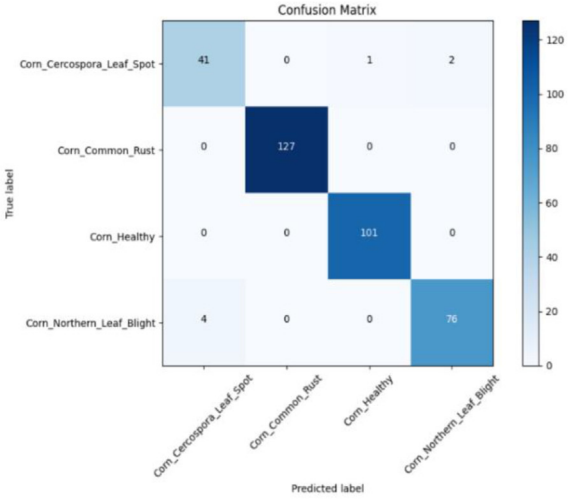
(b) Tomato



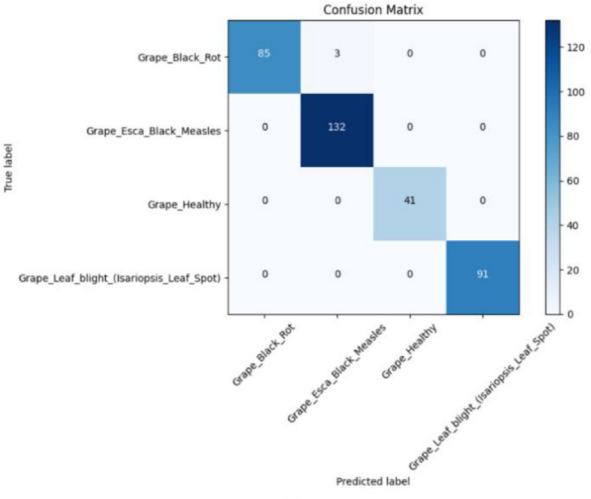
(c) Papper-bell



(d) Apple

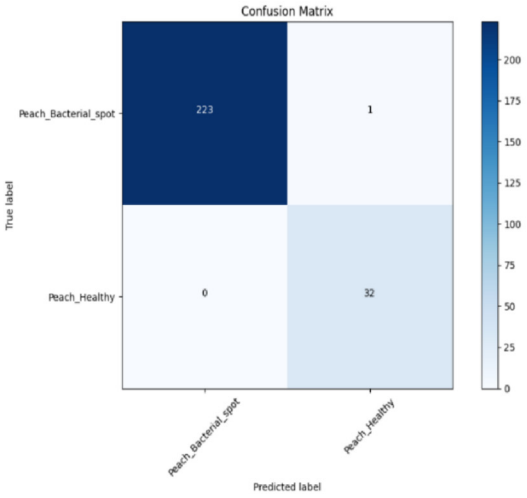


(e) Corn

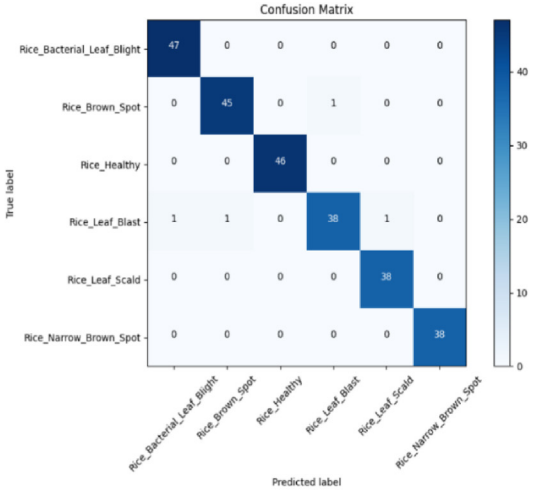


(f) Grape

Fig. 7. Best Model's confusion matrix of various Plant's diseases.

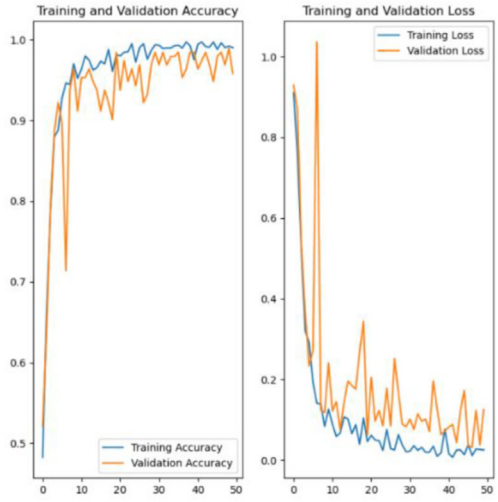


(g) Peach

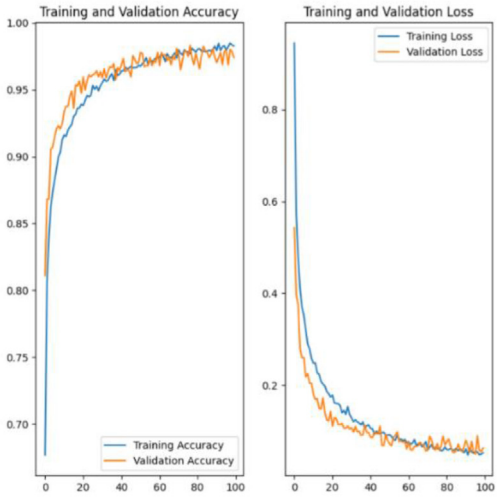


(h) Rice

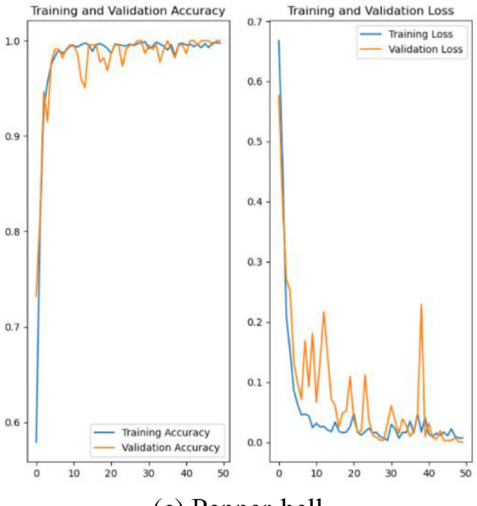
Fig. 7. (*continued*).



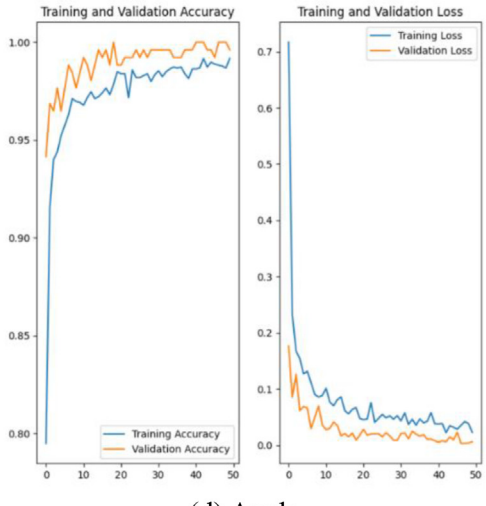
(a) Potato



(b) Tomato



(c) Pepper-bell



(d) Apple

Fig. 8. Best Model's training & validation accuracy vs epoch, loss vs epochs of various Plant's disease.

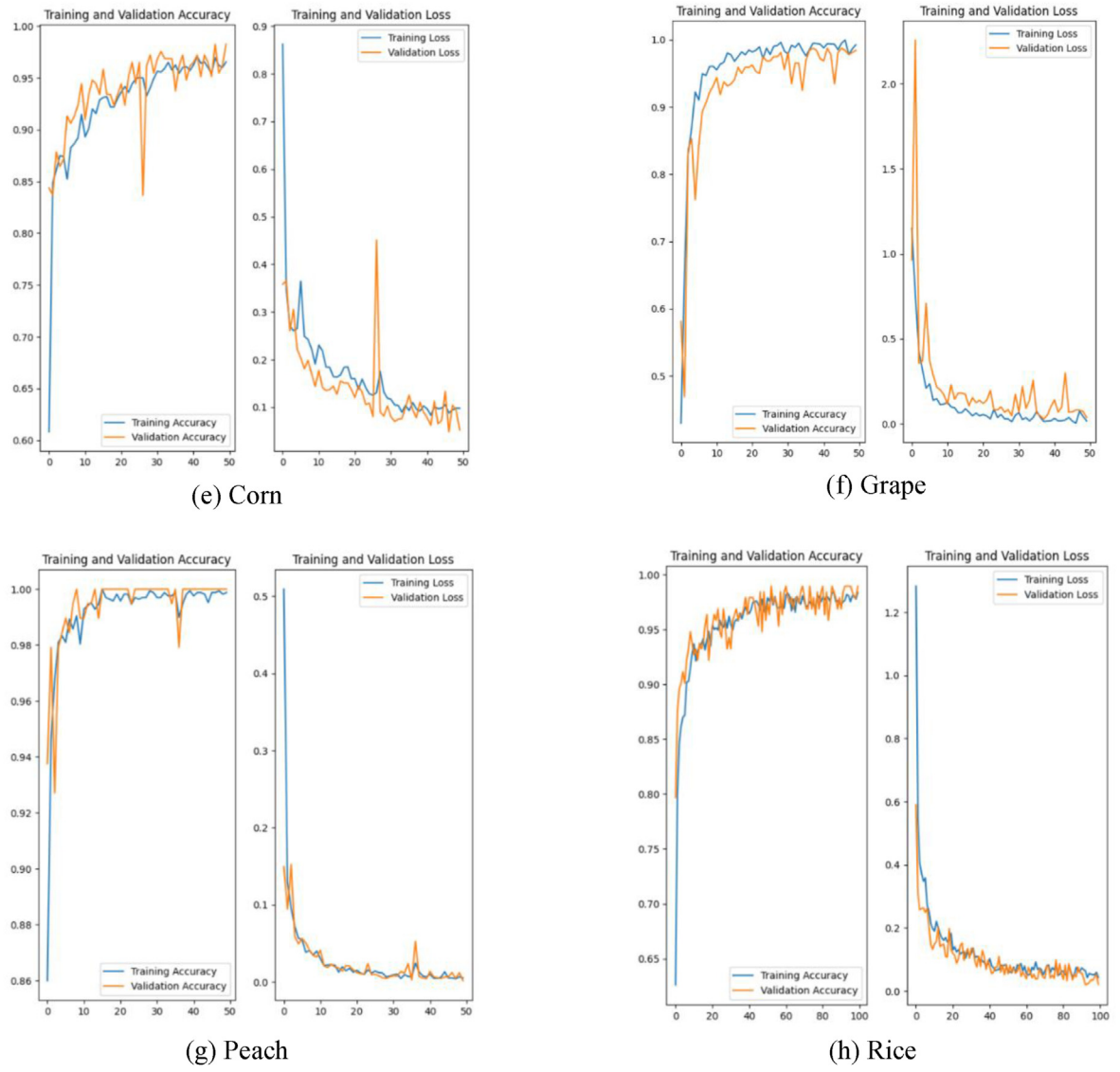


Fig. 8. (*continued*).

```
_id: ObjectId('6689abc59c66d99c79405e33')
username: "naimur"
email: "naimur@gmail.com"
hashed_password: "$argon2id$v=19$m=65536,t=3,p=4$LiXEWMv5n1Pq/f8fw9h7zw$rdgMRwCNtDNBajyp…"

_id: ObjectId('66935fcf1bc07b61767a5fd6')
username: "raihan islam"
email: "raihanislam@gmail.com"
hashed_password: "$argon2id$v=19$m=65536,t=3,p=4$XetdqxUCYGwN4Vzr3TtnLA$8bJ0UecyyaG0Ognn…"
```

Fig. 9. Database records of user registration.

5. Conclusion

With a few recommendations for best practices in the future, this section assesses and concludes the research study presented in this section. Future research direction is also mentioned in this section.

5.1. Summary of research

This research aimed to develop a robust real-time monitoring system for accurate plant leaf disease detection using deep learning. By leveraging a comprehensive dataset of 30,945 images across eight different plant species, we implemented and compared various convolutional neural network (CNN) models. The custom CNN demonstrated superior performance, achieving high accuracy rates in both leaf classification and some individual plant disease detection. Other models like Xception, DenseNet121 and VGG16 performs well for some plants. The developed models were successfully integrated into a web and mobile application, providing users with an intuitive interface to identify plant diseases and receive management recommendations. The deployment process involved converting the trained models to TensorFlow Serving

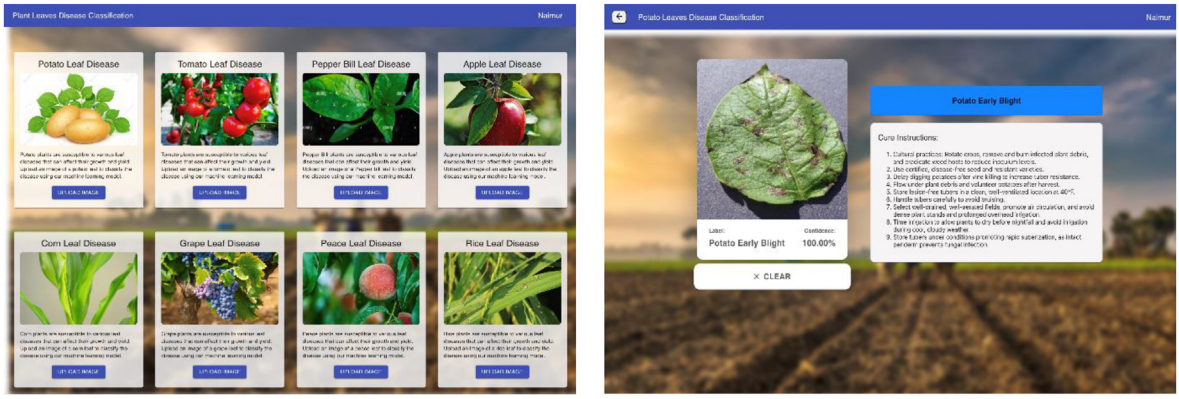


Fig. 10. Home Page and Individual plant's Image Upload Page of Web Application.

and TensorFlow Lite formats, facilitating backend API services developed using FastAPI. The frontend components for the web and mobile applications were built using React and React Native, respectively, providing a seamless user experience across platforms. MongoDB was employed for database management, securely storing user credentials and interaction data.

5.2. Suggestions of improvements

Future work will focus on expanding the dataset to include more plant species and disease classes, enhancing the system's versatility and accuracy. Integrating additional image augmentation methods and exploring the use of generative adversarial networks (GANs) for synthetic data generation could also help address the challenges posed by limited real-world data. Moreover, extending the system's capabilities to support real-time video analysis for continuous crop monitoring could provide more comprehensive disease management solutions.

CRediT authorship contribution statement

Kazi Naimur Rahman: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Sajal Chandra Banik:** Writing – review & editing, Supervision, Formal analysis. **Raihan Islam:** Writing – review & editing, Visualization, Formal analysis, Data curation. **Arafath Al Fahim:** Writing – review & editing, Visualization, Formal analysis, Data curation.

Data availability

https://www.kaggle.com/datasets/naimur006/plant-leaves-disease-detection/data.

Practical Implementation of Web Application Video: https://drive.google.com/file/d/1U0WBKGbw7WYFL4HJkuTuXj-z0Z0UWQDQ/view?usp=sharing.

Practical Implementation of Mobile Application Video: https://drive.google.com/file/d/1keglEueOaFsYZ_4bP5XFBtcRnlAaC5il/view?usp=sharing.

Funding

There was no explicit support for this study from the government, industry and non-profit funding organization. It was self-funded research.

Declaration of interest statement

The authors declare that there are no conflicts of interest regarding the publication of this research. This study was conducted independently, and no financial support or incentives were received from any commercial or private entities that could influence the results or interpretation of the findings.

All authors contributed significantly to the research and development of the real-time monitoring system for plant leaf disease detection using deep learning. The research was driven solely by academic and scientific interests, aimed at advancing knowledge in the field of agricultural technology and improving disease management practices for crops.

The results presented in this study are based on objective analysis and have not been influenced by any external factors. The authors affirm their commitment to maintaining the highest standards of scientific integrity and transparency throughout the research process.

Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of any affiliated institutions or organizations.

Acknowledgement

This work is M.Sc. thesis research work titled “A Real Time Monitoring System for Accurate Plant Leaves Disease Detection Using Deep Learning” which reflects on a real time monitoring system that has been built for detecting plant leaves diseases accurately. This study also reflects on cure instruction for various plant leaves according to specified diseases.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.cropd.2024.100092.

References

[1] Z. Iqbal, M.A. Khan, M. Sharif, J.H. Shah, M.H. ur Rehman, K. Javed, An automated detection and classification of citrus plant diseases using image processing techniques: a review, Comput. Electron. Agric. 153 (2018) 12–32, https://doi.org/10.1016/j.compag.2018.07.032.

[2] E.-C. Oerke, Crop losses to pests, J. Agric. Sci. 144 (1) (2006) 31–43, https://doi.org/10.1017/S0021859605005708.

[3] K. Golhani, S.K. Balasundram, G. Vadamalai, B. Pradhan, A review of neural networks in plant disease detection using hyperspectral data, Inform. Process. Agric. 5 (3) (2018) 354–371, https://doi.org/10.1016/j.inpa.2018.05.002.

[4] S.S. Chouhan, A. Kaul, U.P. Singh, S. Jain, Bacterial foraging optimization based radial basis function neural network (BRBFNN) for identification and classification of plant leaf diseases: an automatic approach towards plant pathology, IEEE Access 6 (2018) 8852–8863, https://doi.org/10.1109/ACCESS.2018.2800685.

[5] G. Zhou, W. Zhang, A. Chen, M. He, X. Ma, Rapid detection of rice disease based on FCM- KM and faster R-CNN fusion, IEEE Access 7 (2019) 143190–143206, https://doi.org/10.1109/ACCESS.2019.2943454.

[6] P. Sharma, P. Hans, S.C. Gupta, Classification of plant leaf diseases using machine learning and image preprocessing techniques, in: 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida,

India, IEEE, Jan. 2020, pp. 480–484, https://doi.org/10.1109/Confluence47617.2020.9057889.
[7] M. Sardogan, A. Tuncer, Y. Ozen, Plant leaf disease detection and classification based on CNN with LVQ algorithm, in: 2018 3rd International Conference on Computer Science and Engineering (UBMK), IEEE, Sarajevo, Sep. 2018, pp. 382–385, https://doi.org/10.1109/UBMK.2018.8566635.
[8] M.M. Ozguven, K. Adem, Automatic detection and classification of leaf spot disease in sugar beet using deep learning algorithms, Phys. Stat. Mech. Appl. 535 (2019) 122537, https://doi.org/10.1016/j.physa.2019.122537.
[9] Y. Lu, S. Yi, N. Zeng, Y. Liu, Y. Zhang, Identification of rice diseases using deep convolutional neural networks, Neurocomputing 267 (2017) 378–384, https://doi.org/10.1016/j.neucom.2017.06.023.
[10] Y. Kawasaki, H. Uga, S. Kagiwada, H. Iyatomi, Basic study of automated diagnosis of viral plant diseases using convolutional neural networks, in: G. Bebis, R. Boyle, B. Parvin, D. Koracin, I. Pavlidis, R. Feris, T. McGraw, M. Elendt, R. Kopper, E. Ragan, Z. Ye, G. Weber (Eds.), Advances in Visual Computing, vol. 9475, Springer International Publishing, Cham, 2015, pp. 638–645, https://doi.org/10.1007/978-3-319-27863-6_59.
[11] P. Jiang, Y. Chen, B. Liu, D. He, C. Liang, Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks, IEEE Access 7 (2019) 59069–59080, https://doi.org/10.1109/ACCESS.2019.2914929.
[12] M. Islam, Anh Dinh, K. Wahid, P. Bhowmik, Detection of potato diseases using image segmentation and multiclass support vector machine, in: 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), Windsor, ON, IEEE, Apr. 2017, pp. 1–4, https://doi.org/10.1109/CCECE.2017.7946594.
[13] Md A. Iqbal, K.H. Talukder, Detection of potato disease using image segmentation and machine learning, in: 2020 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET), IEEE, Chennai, India, Aug. 2020, pp. 43–47, https://doi.org/10.1109/WiSPNET48689.2020.9198563.
[14] Priyanka B. Raj Mangla, Soumya G. Hedge, R. Pooja, Paddy leaf disease detection using image processing and machine learning, Int. J. Innovative Res. Electrical, Electronics, Instrum. Control Engineer. 7 (2) (2019) 97–99, https://doi.org/10.17148/IJIREEICE.2019.7220.
[15] H.D. Rizqi, A.S. Purnomo, The ability of brown-rot fungus Daedalea dickinsii to decolorize and transform methylene blue dye, World J. Microbiol. Biotechnol. 33 (5) (2017) 92, https://doi.org/10.1007/s11274-017-2256-z.
[16] A. M, M. Zekiwos, A. Bruck, Deep learning-based image processing for cotton leaf disease and pest diagnosis, J. Electrical Comput. Engineer. 2021 (2021) 1–10, https://doi.org/10.1155/2021/9981437.
[17] J. Andrew, D.E. Popescu, M.K. Chowdary, J. Hemanth, Deep learning-based leaf disease detection in crops using images for agricultural applications, Agron. 12 (10) (2022) 2395, https://doi.org/10.3390/agronomy12102395.
[18] M. Islam, M. Adil, M.A. Talukder, M.K. Ahamed, M.A. Uddin, M. Hasan, S. Sharmin, M. Rahman, S. Debnath, DeepCrop: deep learning-based crop disease prediction with web application, J. Agric. Food Res. 14 (2023) 100764, https://doi.org/10.1016/j.jafr.2023.100764.
[19] O. Kulkarni, Crop disease detection using deep learning, in: Proceedings of the 2018 International Conference on Current Trends toward Converging Technologies (ICCTCT), 2018, pp. 1–4, https://doi.org/10.1109/ICCUBEA.2018.8697390.
[20] M. Chohan, A. Khan, R. Chohan, S. Katper, M. Mahar, Plant disease detection using deep learning, Int. J. Recent Technol. Eng. 9 (2020) 909–914, https://doi.org/10.35940/ijrte.A2139.059120.
[21] S.P. Mohanty, D.P. Hughes, M. Salathé, Using deep learning for image-based plant disease detection, Front. Plant Sci. 7 (2016), https://doi.org/10.3389/fpls.2016.01419.
[22] S.R. Maniyath, V.P. M, N. M, P. R, N, P, N. S, H.R. Ram, Plant disease detection using machine learning, in: Proc. Int. Conf. Design Innov. For 3Cs: Compute Communicate Control (ICDI3C), 2018, pp. 41–45, https://doi.org/10.1109/ICDI3C.2018.00017.
[23] M. Jung, J. Song, A.-Y. Shin, B. Choi, S. Go, S.-Y. Kwon, J. Park, S. Park, Y.-M. Kim, Construction of deep learning-based disease detection model in plants, Sci. Rep. 13 (2023), https://doi.org/10.1038/s41598-023-34549-2.
[24] S. Sannakki, V. Rajpurohit, V. Nargund, P. Kulkarni, Diagnosis and classification of grape leaf diseases using neural networks, in: 2013 4th International Conference on Computing, Communications and Networking Technologies (ICCCNT 2013), 2013, pp. 1–5, https://doi.org/10.1109/ICCCNT.2013.6726616.
[25] M. Jhuria, A. Kumar, R. Borse, Image processing for smart farming: detection of disease and fruit grading, in: 2013 IEEE Second International Conference on Image Information Processing (ICIIP-2013), 2013, pp. 521–526, https://doi.org/10.1109/ICIIP.2013.6707647. Shimla, India.
[26] K. Wang, S. Zhang, Z. Wang, Z. Liu, F. Yang, Mobile smart device-based vegetable disease and insect pest recognition method, Intelligent Automation & Soft Computing 19 (2013), https://doi.org/10.1080/10798587.2013.823783.
[27] K. Ahmed, T. Shahidi, S.I. Alam, S. Momen, Rice leaf disease detection using machine learning techniques, in: Presented at 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), 2019, pp. 1–5, https://doi.org/10.1109/STI47673.2019.9068096. Dhaka, Bangladesh.
[28] A. Dixit, S. Nema, Wheat leaf disease detection using machine learning method-a review, Int. J. Comput. Sci. Mobile Comput. 7 (5) (2018) 124–129.
[29] K. Panigrahi, H. Das, A. Sahoo, S. Moharana, Maize leaf disease detection and classification using machine learning algorithms, in: Proceedings of the International Conference on Advanced Machine Learning Technologies and Applications, 2020, p. 66, https://doi.org/10.1007/978-981-15-2414-1_66.
[30] D.M. Phung, J.T. Rhee, A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets, Appl. Sci. 9 (21) (2019) 4500, https://doi.org/10.3390/app9214500.