International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 04 | April - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

Leaf Scanning Disease Detection

Dr. Pratima.A.Kalyankar[1] , Harsh Yadav[2], Pratik Khaladkar[3], Himanshu Rane[4]

[1]Associate Prof. Electronics and Telecommunication JSCOE, Pune
[2]Electronics and Telecommunication JSCOE, Pune
[3]Electronics and Telecommunication JSCOE, Pune
[4]Electronics and Telecommunication JSCOE, Pune

Abstract - Crop diseases pose a significant threat to global food security, with rapid identification challenges in regions lacking essential infrastructure. The confluence of rising smartphone adoption worldwide and recent strides in computer vision through deep learning has opened avenues for smartphone-enabled disease diagnosis. Leveraging a public dataset containing 2,500 images of plant leaves in varied health conditions, acquired under controlled settings, we employed a deep convolutional neural network. This model successfully discerns crop species and identifies diseases or their absence, achieving an impressive 86.35% accuracy on a withheld test set. This underscores the viability of the proposed methodology. In essence, the strategy of training deep learning models on expansive and accessible image datasets signals a promising route for widespread smartphone-assisted crop disease diagnosis on a global scale.

Keywords - crop diseases, machine learning, deep learning, convolutional neural network (CNN)

I. INTRODUCTION

Modern advancements in technology have empowered human society to generate sufficient food resources to meet the needs of a growing population exceeding 7 billion. Despite this progress, food security remains jeopardized by various factors, including climate change, the decline in pollinators, and the prevalence of plant diseases. Plant diseases not only pose a global threat to food security but also have severe consequences for smallholder farmers, who heavily rely on healthy crops for their livelihoods. In the developing world, where over 80% of agricultural production comes from smallholder farmers, reports of significant yield losses due to pests and diseases are common.

Efforts to mitigate crop loss from diseases have evolved from widespread pesticide use to integrated pest management (IPM) approaches. Regardless of the strategy employed, accurately identifying diseases early is critical for effective disease management. Historically, agricultural extension organizations and local plant clinics supported disease identification, with recent efforts leveraging online platforms and, more recently, mobile phones due to the widespread adoption of mobile technology globally. The resulting system can be integrated into precision agriculture.

Smartphones, with their computing power, high resolution displays, and advanced cameras, offer innovative approaches for disease identification. The proliferation of smartphones globally, estimated to reach 5 to 6 billion by 2020, coupled with high-speed internet access, creates an unprecedented opportunity for automated image recognition-based disease diagnosis on a massive scale. This feasibility is demonstrated using a deep learning approach on a dataset of 54,306 images representing 14 crop species with 26 diseases, made publicly available through the Plant Village project.

Deep neural networks have found success in diverse domains, providing end-to-end learning solutions. The mapping of input, such as an image of a diseased plant, to an output, like a crop-disease pair, is achieved through stacked layers of nodes. Training deep neural networks involves tuning parameters to improve the mapping during the process, a task that has seen substantial improvement in recent times.

To develop accurate image classifiers for plant disease diagnosis, a large, verified dataset was essential. The Plant Village project addressed this gap by collecting tens of thousands of images of healthy and diseased crop plants, making them openly available. This study reports on the classification of 26 diseases in 14 crop species using 54,306 images and a convolutional neural network approach. The best-performing model achieved a mean F1 score of 0.9934 (overall accuracy of 99.35%), demonstrating the technical feasibility of smartphone-assisted plant disease diagnosis.

This paper focuses on the development of a CNN-based system that can be used for real-time disease identification through leaf scanning. The approach involves collecting a comprehensive dataset of healthy and diseased plant leaves, training a deep learning model to learn relevant features, and utilizing transfer learning techniques to improve model's performance.

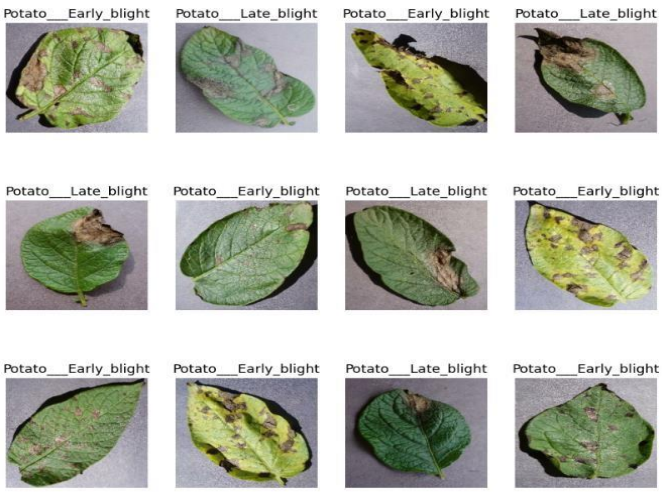

Fig no 1: Sample Potato Leaves with diseases

II. METHODS

A. Dataset Description

We conducted an analysis on a dataset consisting of 3,000 images of plant leaves, each associated with one of 3 class labels representing crop-disease pairs. The objective was to predict the specific crop-disease pair based solely on the image of the plant leaf. In our approach, all images were resized to 256 × 256 pixels, and both model optimization and predictions were performed on these downscaled images.

In Our dataset, we divide image base on their disease. In potatoes leaf there are 3 type of categories: Late blight, Early blight , healthy leaf.

Throughout our experiments, three versions of the Plant Village dataset were utilized. Initially, we used the original Plant Village dataset in color. Subsequently, we experimented with a gray-scaled version, and finally, all experiments were conducted on a version of the dataset where leaves were segmented. Segmentation involved the removal of excess background information, which could potentially introduce biases in the dataset due to the regularized process of data collection in the case of Plant Village. An automated script, tailored for our dataset, was employed for segmentation. The technique utilized a set of masks generated through the analysis of color, lightness, and saturation components in different color spaces (Lab and HSB). This process not only facilitated segmentation but also addressed color casts in some subsets of the dataset, eliminating potential biases.

The purpose of these experiments was to ascertain whether the neural network genuinely grasps the concept of plant diseases or if it merely learns inherent biases within the in

dataset. Figure 2 illustrates various versions of the same leaf for a randomly selected set of leaves.

B. Measurement of Performance

To gain insight into the performance of our methodologies on new, unseen data and to monitor for potential overfitting, we conducted experiments across a diverse range of train-test set splits. These splits encompassed various proportions, including 80–20 (80% for training, 20% for testing), 60–40 (60% for training, 40% for testing), 50–50 (equal distribution for training and testing), 40–60 (40% for training, 60% for testing), and 20–80 (20% for training, 80% for testing). Notably, within the Plant Village dataset, multiple images of the same leaf are often present, captured from different perspectives. We meticulously managed these cases for 2,500 out of the 3,000 images, ensuring that all images of the same leaf were consistently assigned to either the training or testing set across all splits.

Furthermore, for each experiment, we computed metrics such as mean precision, mean recall, mean F1 score, and overall accuracy throughout the training process at regular intervals, typically at the conclusion of each epoch. The final mean F1 score served as the primary metric for comparing results across different experimental configurations. This rigorous methodology allowed us to comprehensively assess model performance under varying conditions while maintaining consistency in data handling across all experiments.

C. Specifications Of Proposed System

An effective host system, a robust server or cloud infrastructure with adequate compute resources is essential. This infrastructure should be capable of handling the computational demands of model inference and serving incoming requests efficiently.

Furthermore, sufficient storage capacity is required to accommodate model files, image data, and logging information. This includes storage for the trained model files, which can be substantial in size, as well as the dataset of images used for training and testing purposes. Additionally, storage space is needed for logging data, such as system metrics, error logs, and user activity logs.

In terms of software requirements, the server should be equipped with an operating system suitable for hosting the system, such as Linux. A Python environment with necessary packages, including TensorFlow for machine learning tasks and Fast API for building web APIs, is also necessary.

For model deployment, TensorFlow Serving or a similar framework can be used to facilitate serving trained models over the network. This ensures that the machine learning models are efficiently deployed and can handle incoming inference requests with low latency. If the system involves the management of large amounts of data, a database system may be necessary for storing and managing image and model data. This database system can help organize and retrieve data efficiently, improving the overall performance of the system.:

II. APPROACH

We evaluate the applicability of deep convolutional neural networks for the classification problem described above. We focus on two popular architectures, namely AlexNet (Krizhevsky et al., 2012), and GoogLeNet (Szegedy et al., 2015), which were designed in the context of the “Large Scale Visual Recognition Challenge” (ILSVRC) (Russakovsky et al., 2015) for the ImageNet dataset (Deng et al., 2009). The AlexNet architecture (see Figure S2) follows the same design pattern as the LeNet-5 (LeCun et al., 1989) architecture from the 1990s.

The LeNet-5 architecture variants are usually a set of stacked convolution layers followed by one or more fully connected layers. The convolution layers optionally may have a normalization layer and a pooling layer right after them, and all the layers in the network usually have ReLu non-linear activation units associated with them. AlexNet consists of 5 convolution layers, followed by 3 fully connected layers, and finally ending with a softMax layer. The first two convolution layers (conv{1, 2}) are each followed by a normalization and a pooling layer, and the last convolution layer (conv5) is followed by a single pooling layer. The final fully connected

layer (fc8) has 38 outputs in our adapted version of AlexNet (equaling the total number of classes in our dataset), which feeds the softMax layer. The softMax layer finally exponentially normalizes the input that it gets from (fc8), thereby producing a distribution of values across the 38 classes that add up to 1. These values can be interpreted as the confidences of the network that a given input image is represented by the corresponding classes.

All of the first 7 layers of AlexNet have a ReLu non-linearity activation unit associated with them, and the first two fully connected layers (fc{6, 7}) have a dropout layer associated with them, with a dropout ratio of 0.5. The GoogleNet architecture on the other hand is a much deeper and wider architecture with 22 layers, while still having considerably lower number of parameters (5 million parameters) in the network than AlexNet (60 million parameters). An application of the "network in network" architecture (Lin et al., 2013) in the form of the inception modules is a key feature of the GoogleNet architecture. The inception module uses parallel 1 × 1, 3 × 3, and 5 × 5 convolutions along with a max-pooling layer in parallel, hence enabling it to capture a variety of features in parallel.

In terms of practicality of the implementation, the amount of associated computation needs to be kept in check, which is why 1 × 1 convolutions before the above mentioned 3 × 3, 5 × 5 convolutions (and also after the max-pooling layer) are added for dimensionality reduction. Finally, a filter concatenation layer simply concatenates the outputs of all these parallel layers. While this forms a single inception module, a total of 9 inception modules is used in the version of the GoogLeNet architecture that we use in our experiments. A more detailed overview of this architecture can be found for reference in (Szegedy et al., 2015).

III. ALGORITHM

The algorithm employed for leaf image classification using Convolutional Neural Networks (CNNs). CNNs have demonstrated remarkable performance in image classification tasks due to their ability to automatically learn hierarchical features from raw pixel data. Our approach involves extracting discriminative features from leaf images and utilizing SoftMax activation for multi-class prediction, while optimizing model parameters through backpropagation for continuous refinement.

1. Data Preprocessing:

Acquisition of Leaf Images: Collect a diverse dataset of leaf images covering various species and environmental conditions.

Data Augmentation: Augment the dataset to increase its size and diversity, including techniques such as rotation, flipping, and scaling.

Normalization: Normalize the pixel values of the images to ensure uniformity and accelerate convergence during training.

2. Model Architecture:

Convolutional Layers: Stack multiple convolutional layers to capture low-level to high-level features present in leaf images.

Pooling Layers: Apply pooling operations to down sample feature maps and enhance the model's spatial invariance.

Fully Connected Layers: Incorporate fully connected layers to facilitate the mapping of extracted features to class labels.

Softmax Activation: Utilize softmax activation at the output layer for multi-class classification, providing probabilities for each class.

3. Training:

Initialization: Initialize the model parameters using appropriate techniques such as Xavier or He initialization.
Loss Function: Define the cross-entropy loss function to measure the discrepancy between predicted and actual class labels.
.
Optimization: Employ backpropagation along with stochastic gradient descent (SGD) or advanced optimizers like Adam to update model parameters iteratively.

Hyperparameter Tuning: Fine-tune hyperparameters such as learning rate, batch size, and regularization strength to optimize model performance.

Early Stopping: Implement early stopping to prevent overfitting by monitoring validation performance and halting training when performance starts to degrade.

4. Evaluation:

Performance Metrics: Assess the model's performance using metrics such as accuracy, precision, recall, and F1score.

Cross-Validation: Employ k-fold cross-validation to obtain robust estimates of model performance.

Visualization: Visualize model predictions, confusion matrices, and feature maps to gain insights into its behaviour and performance
.

5. Inference:

Deployment: Deploy the trained model to classify new simultaneously, making it suitable for deployment in leaf images in real-world scenarios.

V. FLOW CHART

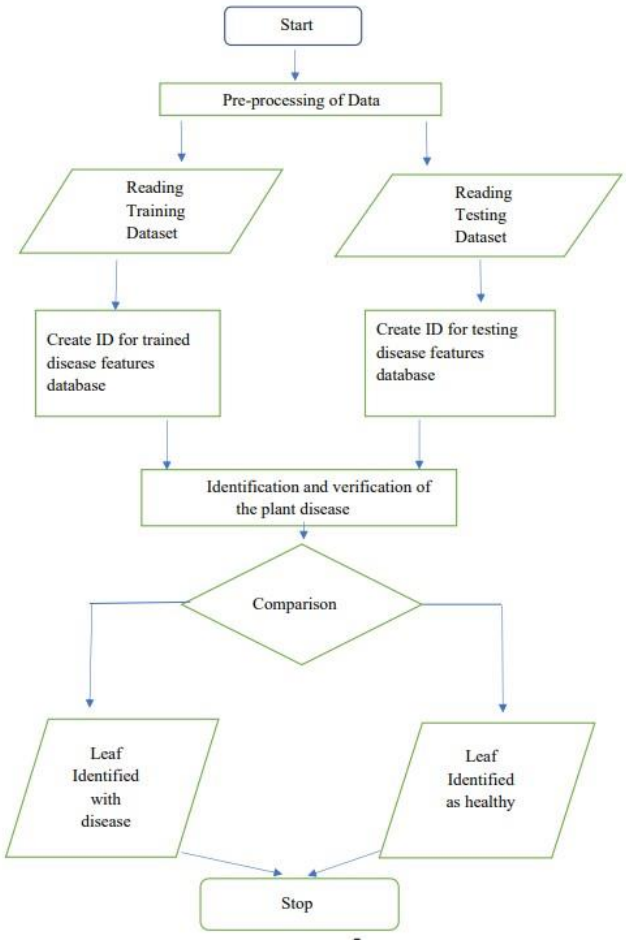

Fig no 2: Flow Chart

VI. MODEL BUILDING

The core of a CNN comprises multiple convolutional layers that apply convolution operations to input images, utilizing learnable filters to detect various features such as edges, textures, and shapes. The architecture's depth and number of convolutional layers can vary based on the problem's complexity and available computational resources.

Activation functions like ReLU (Rectified Linear Unit) are applied after each convolutional layer to introduce nonlinearity into the model. Pooling layers (e.g., max-pooling or average-pooling) are inserted to down sample the spatial dimensions of feature maps, reducing computational load and enhancing translation invariance. A flattening layer is employed to convert the feature maps into a one-dimensional vector, preparing them for fully connected layers.

TensorFlow Serving is utilized for deploying machine learning models, including those created with TensorFlow, for production use. It offers high performance, low latency, and the ability to handle multiple model versions simultaneously, making it suitable for deployment in microservices architectures or containerized environments.

Fast API, a modern and high-performance web framework for building APIs with Python, facilitates the rapid development of APIs. It provides automatic interactive documentation, validation, serialization, and other features, simplifying API development processes.

VII. BLOCK DIAGRAM

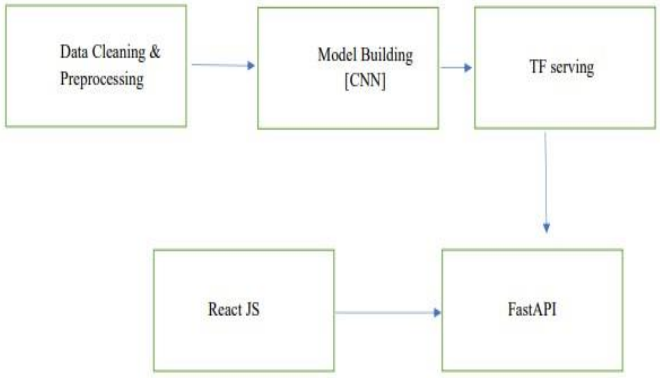

Fig no 3: Block Diagram

VIII. RESULT

1. Summary of Model

Model: "sequential_2"

Layer (type)	Output Shape	Param #
sequential (Sequential)	(32, 256, 256, 3)	0
conv2d (Conv2D)	(32, 254, 254, 32)	896
max_pooling2d (MaxPooling2D)	(32, 127, 127, 32)	0
conv2d_1 (Conv2D)	(32, 125, 125, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(32, 62, 62, 64)	0
conv2d_2 (Conv2D)	(32, 60, 60, 64)	36928
max_pooling2d_2 (MaxPooling2D)	(32, 30, 30, 64)	0
conv2d_3 (Conv2D)	(32, 28, 28, 64)	36928
max_pooling2d_3 (MaxPooling2D)	(32, 14, 14, 64)	0
conv2d_4 (Conv2D)	(32, 12, 12, 64)	36928
max_pooling2d_4 (MaxPooling2D)	(32, 6, 6, 64)	0
conv2d_5 (Conv2D)	(32, 4, 4, 64)	36928
max_pooling2d_5 (MaxPooling2D)	(32, 2, 2, 64)	0
flatten (Flatten)	(32, 256)	0
dense (Dense)	(32, 64)	16448
dense_1 (Dense)	(32, 3)	195

Total params: 183,747
Trainable params: 183,747
Non-trainable params: 0

2.Number of Epoch are use:

By monitoring changes in epoch accuracy and loss throughout the training process, practitioners can make informed decisions regarding model adjustments, such as tuning hyperparameters, applying regularization techniques, or modifying the model architecture. Additionally, visualizing these changes over epochs allows for a deeper understanding of the training dynamics and helps diagnose potential issues such as underfitting or overfitting. Overall, analyzing epoch accuracy and loss changes provides valuable feedback for refining neural network models and improving their performance on classification tasks.

```
Epoch 1/10
54/54 [==============================] - 115s 2s/step - loss: 0.2101 - accuracy: 0.9323 - val_loss: 0.2884 - val_accuracy: 0.87
67
Epoch 2/10
54/54 [==============================] - 116s 2s/step - loss: 0.2006 - accuracy: 0.9253 - val_loss: 0.1659 - val_accuracy: 0.93
52
Epoch 3/10
54/54 [==============================] - 117s 2s/step - loss: 0.2255 - accuracy: 0.9144 - val_loss: 0.3291 - val_accuracy: 0.85
76
Epoch 4/10
54/54 [==============================] - 117s 2s/step - loss: 0.1308 - accuracy: 0.9525 - val_loss: 0.1223 - val_accuracy: 0.96
12
Epoch 5/10
54/54 [==============================] - 117s 2s/step - loss: 0.1097 - accuracy: 0.9606 - val_loss: 0.1566 - val_accuracy: 0.94
21
Epoch 6/10
54/54 [==============================] - 116s 2s/step - loss: 0.1038 - accuracy: 0.9641 - val_loss: 0.0880 - val_accuracy: 0.96
53
Epoch 7/10
54/54 [==============================] - 117s 2s/step - loss: 0.1288 - accuracy: 0.9497 - val_loss: 0.1171 - val_accuracy: 0.95
14
Epoch 8/10
54/54 [==============================] - 118s 2s/step - loss: 0.0729 - accuracy: 0.9740 - val_loss: 0.0985 - val_accuracy: 0.96
70
Epoch 9/10
54/54 [==============================] - 117s 2s/step - loss: 0.0865 - accuracy: 0.9653 - val_loss: 0.1370 - val_accuracy: 0.94
79
Epoch 10/10
54/54 [==============================] - 118s 2s/step - loss: 0.0668 - accuracy: 0.9740 - val_loss: 0.1503 - val_accuracy: 0.94
50
```

3.Visualization using Line graph

Fig no 4: Visualization of Line Graphs

4. Model Testing

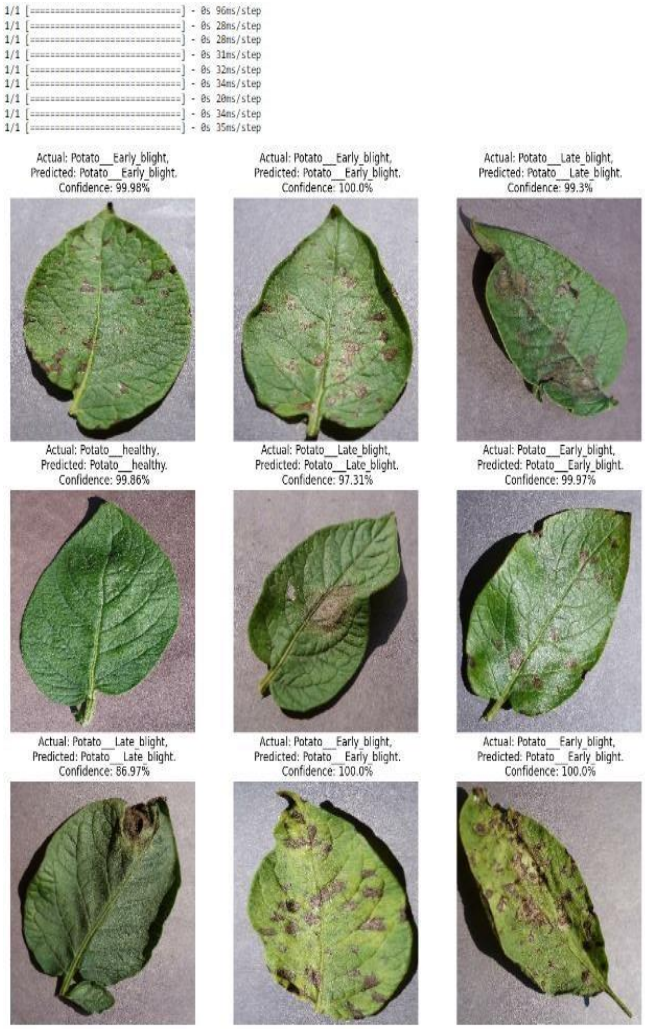

Fig no 5: Potato Leaves with Disease detection

IX. RELEVANCE TO THE PRESENT INDUSTRIAL

A plant disease detection system using advanced technologies like CNN-based models is highly relevant to the present industrial scenario and agriculture sector for several reasons:

Crop Protection and Yield Optimization: Agriculture is a critical industry that directly impacts global food security. Plant diseases can lead to significant crop losses, which can threaten the food supply. An automated disease detection system helps protect crops by identifying diseases at an early stage, allowing for timely intervention and reducing yield losses. Sustainable Agriculture: The agriculture industry is increasingly focused on sustainability. Implementing advanced disease detection systems reduces the need for excessive use of pesticides and fungicides, which can have negative environmental impacts.

By targeting treatments more effectively, these systems contribute to sustainable farming practices. Precision Agriculture: Modern agriculture is moving toward precision

farming, which involves using data-driven insights to make informed decisions. Automated disease detection aligns with this trend by providing farmers with real-time information about the health of their crops, enabling precise actions to be taken only when necessary.

Labor Efficiency: Agriculture often faces labor shortages. Automated disease detection can reduce the need for manual inspections, enabling farmers to allocate their labor more efficiently to other essential tasks. Advanced Technology Adoption: Many industries, including agriculture, are embracing advanced technologies to improve efficiency and productivity. The use of CNN based models for disease detection represents the integration of cutting-edge technology into an age-old industry.

X. ACKNOWLEDGEMENT

We take this opportunity to present our seminar report on “LEAF SCANNING DISEASE DETECTION USING ML”. We express our sincere thanks to our project guide Dr. P.A.Kalyankar for his valuable help, guidance and the confidence which she gave us at all stages of the project work. We also express our gratitude to Dr. S. M. Hambarde, Head of the E&TC Dept. for providing us the necessary facilities in the laboratory as well as his kind support. Finally, we are grateful to all faculty members of our department for their cooperation and valuable help.

XI. CONCLUSION

In conclusion, the development of a plant disease detection system using Convolutional Neural Networks (CNNs) is a significant and timely endeavor with the potential to bring transformative benefits to the agricultural industry. This technology, aligned with the present industrial scenario, offers a promising solution to address critical challenges in agriculture, enhance crop protection, and promote sustainable farming practices.

XIII. REFRENCES

1. Shweta Verma, et al. (2017). "Automated Detection of Plant Diseases through Image Processing Techniques."

2. Narges Sarraf and Sinisa Todorovic (2016). "Deep Learning for Plant Disease Detection and Diagnosis."

3. Savita Gupta and Sandeep Kautish (2020). "Machine Learning-Based Leaf Disease Detection: A Review."

4. Moumita Dey and P. Balamurugan (2021). "A Survey of Deep Learning Techniques for Plant Disease Detection."

5. Neelam Sinha and Jyoti Sinha (2019). "Computer Vision-Based Tomato Disease Detection: A Comprehensive Review."

6. Sankaran, S., Mishra, A., Ehsani, R., & Davis, C. (2010). "A review of advanced techniques for detecting plant diseases." Computers and Electronics in Agriculture, 72(1), 1-13.

7. Sladojevic, S., Arsenovic, M., Anderla, A., & Culibrk, D. (2016). "Deep neural networks based recognition of plant diseases by leaf image classification." Computational Intelligence and Neuroscience, 2016.

8. Barbedo, J. G. A. (2019). "Plant disease identification from individual lesions in images: An object-based approach." Computers and Electronics in Agriculture, 165, 104972.

9. Ghosal, S., Blystone, D., Singh, A. K., Ganapathy subramanian, B., Singh, A., & Sarkar, S. (2018).