Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

Plant Disease Detection Using Convolutional Neural Networks

Namitha Poduval
Department of E&TC
Vishwakarma Institute of Information Technology
Pune, India

Avantika Ravatale
Department of E&TC
Vishwakarma Institute of Information Technology
Pune, India

Srushti Shindkar
Department of E&TC
Vishwakarma Institute of Information Technology
Pune, India

Nikhil Kandesar
Department of E&TC
Vishwakarma Institute of Information Technology
Pune, India

Rucha Shinde
Department of E&TC
Vishwakarma Institute of Information Technology
Pune, India

Dr. Pravin G. Gawande
Department of E&TC
Vishwakarma Institute of Information Technology
Pune, India

Dr. Yogesh H. Dandawate
Department of E&TC
Vishwakarma Institute of Information Technology
Pune, India

***Abstract*— Plant disease detection is critical in agriculture, as early detection and diagnosis of plant diseases can prevent crop losses. This paper presents an abstract of such a system that uses Convolutional Neural Networks (CNN) to identify and classify plant disease based on images of leaves. The system includes pre-processing techniques for image enhancement, followed by the use of a CNN model for feature extraction and classification. The system achieves high accuracy in identifying plant diseases, thus providing an effective tool for farmers and researchers to monitor and control diseases in crops. This paper concludes that CNN-based plant disease detection systems have significant potential to enhance plant disease diagnosis and treatment efficiency and accuracy, leading to improved crop yields and overall food security.**

***Keywords*—plant disease detection, CNN, deep learning, image processing**

I. INTRODUCTION

Plant disease detection systems are becoming increasingly important in agricultural research due to the need to identify and address plant diseases in a timely and effective manner. In recent years, Convolutional Neural Networks (CNNs) have been used as an effective tool for image recognition and classification. [7] In this paper, we propose a plant disease prediction system using CNN that can accurately detect and classify different types of plant diseases based on leaf images. The proposed system consists of three main components: image preprocessing, feature extraction, and classification. In the preprocessing stage, leaf images are pre-processed to remove any background noise and enhance the contrast of the image. In the feature extraction stage, the CNN model is used to extract the relevant features from the pre-processed images. Finally, in the classification stage, the extracted features are used to classify the plant disease type. To evaluate the performance of the proposed system, we conducted experiments using a dataset consisting of leaf images of different types of plant diseases. The results demonstrate that the proposed system can accurately predict the type of plant disease with high accuracy and low error rate. Also, this provides a valuable tool for farmers and researchers to identify and address plant diseases in a timely and effective manner, thereby improving crop yields and food security.

II. PROPOSED METHOD

A. Dataset

We use the plant village dataset which consists of 20,639 healthy and unhealthy leaf images of tomatoes, potatoes, and bell paper. The image is reduced in size to 256 x 256 pixels, and this compressed image is then optimized and model predictions are made.

Table 1 Total number of leaves

Leaf category	Number of Images of size 256X256
Tomato	16012
Potato	2152
Bell pepper	2475
Total	20639

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

Table 2 Number of healthy and diseased leaf

Leaf category	Healthy leaf	Diseased leaf
Tomato	1591	14421
Potato	152	2000
Bell- pepper	1478	997

Figure 1 Healthy Leaves

Figure 2 Diseased Leaves

1. Bacterial spot: Bacterial spot is a common disease that affects bell peppers and other plants in the Solanaceae family. It is caused by the bacterium Xanthomonas campestris pv. vesicatoria. [8]. By using preventative measures including sanitation, appropriate watering procedures, pruning, and spacing, bacterial spots can be avoided. Work in the garden without watering plants, sprays made of copper, followed by routine monitoring and scouting.
2. Early Blight: Numerous plants, including tomatoes, potatoes, and other solanaceous crops, are susceptible to the widespread fungal disease known as early blight.[8] The fungus Alternaria solani is responsible for the outbreak. Early blight can be prevented by performing crop rotation, proper spacing, applying a layer of organic mulch such as straw, or wood chips, around the base of plants.
3. Late Blight: Late blight is a destructive fungal disease that primarily affects solanaceous crops such as tomatoes and potatoes. It is caused by the pathogen Phytophthora infestans. It is curable by avoiding overhead watering, removing infected plant material, mulching, fungicidal applications, choose varieties that have resistance to late blight. Resistant varieties are less likely to be affected or show milder symptoms when exposed to the pathogen, ensure adequate spacing between plants to promote good air circulation.[8]. Dense foliage can create a favorable environment for late blight development and spread. Follow the recommended spacing guidelines for your specific crop.
4. Target Spot: A fungus called target spot, also known as corynespora leaf spot, damages a wide range of plants, including decorative plants and food crops like tomatoes, peppers, and other vegetables. It is caused by the fungus Corynespora cassiicola.[8] By using preventative measures including sanitation, appropriate watering procedures, pruning, spacing, leaf removal, weed control, monitoring regularly, target spots can be avoided.
5. Mosiac Virus: Mosaic viruses are a group of plant viruses that cause a characteristic mosaic pattern of light and dark green areas on the leaves of infected plants. There are several different types of mosaic viruses that can affect a wide range of plants, including vegetables, ornamentals, and fruit trees. Mosiac viruses can be prevented by using certifies virus-free plants, practice good hygiene, many mosaic viruses are spread by insect pest such as aphids, whiteflies and leaf hoppers.Implement strategies to manage these pesrs, such as insecticidal soaps, sticky traps, or biological control methods.[8]
6. Yellow Leaf Curl Virus: Yellow leaf curl virus (YLCV) is a plant virus that primarily affects tomato plants, but it can also infect other members of the Solanaceae family, including peppers and potatoes. It is transmitted by the whitefly insect (Bemisia tabaci). Yellow leaf curl virus can be prevented by weed control, insect control, using floating row covers or insect-proof nets to physically separate tomato plants from insects, choosing tomato cultivators that are resistant to yellow leaf curl disease, buying healthy plants from reputable nurseries reduces the risk of introducing the disease at first.[8]
7. Septoria Spot: The fungus disease septoria leaf spot, also called septoria leaf blight, commonly affects a variety of plants, including ornamental plants, fruits, and vegetables. A fungus called Septoria spp. is responsible. By using preventative measures including sanitation, appropriate watering procedures, pruning, spacing, leaf removal, weed control, monitoring regularly, septoria spots can be avoided.[8]

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

8. Spider Mites: Spider mites are tiny arachnids that belong to the Tetranychidae family. They are common pests that can infest a wide range of plants, both indoors and outdoors. Spider mites can be cured by regular inspection of leaves for any signs of spider mites, such as webbing, tiny speaks(mites), or stippling (tiny yellow or white spots), avoid using pesticides, use a strong water spray, provide good air circulation.[8]

B. Data Processing and Augmentation

A powerful image classifier must incorporate image augmentation. Even while datasets may have hundreds to a few thousand training samples, the variety may not be sufficient to create a reliable model.[3] The many image enhancement choices include resizing the image, rotating it at different angles, and flipping it vertically or horizontally. These additions aid in increasing the amount of pertinent data in the dataset. It is discovered that each image in the Plant Village collection is 256 × 256 pixels in size. The Keras deep-learning framework is used for data processing and image enhancement.

The following are the augmentation choices used for training: [3]

1. Rotation: To randomly rotate a training image at different angles.
2. Brightness: By feeding the model photos of varied brightness during training, brightness helps the model adjust to changes in lighting.
3. Shear: Adjust the shearing angle.

III. PROPOSED SYSTEM

We are building a neural network model for image classification. This model will be deployed on a website. The website will use this model to identify plant leaf disease in real time.[3]

1. Step 1 is to collect data.
2. The gathered dataset is pre-processed and enhanced using Keras' pre-processing and Image-data generation API.
3. Use pre-trained a CNN (Convolutional Neural Network) model like mobileNet, ResNet50, and Inception for classifying different plant diseases.
4. Build a website using reactjs with FastAPI webserver.

Convolutional Neural Network Model

Since DL models can learn relevant characteristics from input images at different convolutional levels, they are the most popular architecture for CNNs and have recently attracted a lot of attention. This is similar to how the human brain works. Complex issues can be resolved fast and effectively by DL with high classification accuracy and low error rates[1]. The convolutional, pooling, fully connected, and activation layers are some of the components that make up the DL model. Table 3 shows the number of layers and parameter sizes of different CNN models. ResNet50 has a layer size of 50 and 25.6 million parameters whereas inception has 48 layers and 27 million parameters and mobileNet-V2 has a layer size of 28 and 3.37 million parameters. In our work, we have used a pre-trained ResNet50 model.[1]

Table 3 Comparison among different CNN architectures regarding layer number and parameter size

Model	No. of Layers	Parameters (million)
ResNet50	50	25.6
InceptionV3	48	27
MobileNetV2	28	3.37

CNN Model steps:

- Conv2D - Conv2D is a commonly used layer in convolutional neural networks (CNNs) for image processing and computer vision tasks. It stands for two-dimensional convolution and is used to extract features from two-dimensional input data, such as images. In a Conv2D layer, a set of learnable filters (also known as kernels or weights) is convolved with the input data to perform a series of convolution operations. Each filter is a small matrix that is slid across the input data, computing dot products at each location. This process helps to detect different patterns and features present in the input. The output of a Conv2D layer is a set of feature maps, each representing a different learned feature from the input image. These feature maps are then passed on to subsequent layers in the neural network for further processing.[1]
- Maxpooling - Maxpooling is a down-sampling operation commonly used in convolutional neural networks (CNNs) for reducing the spatial dimensions of feature maps and extracting the most important information from them. It is typically applied after Conv2D layers.[1] The purpose of maxpooling is to reduce the size of the feature maps while retaining the most prominent features. It achieves this by dividing the input feature map into non-overlapping rectangular regions (usually squares) and taking the maximum value within each region. This operation effectively downscales the feature maps, reducing their spatial dimensions
- Flatten - In a convolutional neural network (CNN), the Flatten layer is used to convert the two-dimensional (or multi-dimensional) feature maps from the previous convolutional layers into a one-dimensional vector. This operation is often performed before connecting the feature maps to fully connected layers. After the convolutional and pooling layers in a CNN, the feature maps retain their spatial dimensions, typically represented as a three-dimensional tensor (height, width, channels). However, fully connected layers require a one-dimensional input. This is where the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

Flatten layer comes in. The Flatten layer reshapes the input tensor into a single vector by stacking all the elements. It removes the spatial structure and converts the multi-dimensional feature maps into a long linear vector. The output shape of the Flatten layer is determined by the dimensions of the input feature maps.[1]

- Epochs - An epoch refers to a complete pass through the entire training dataset during the training phase. In other words, it means that CNN has seen and processed every training example once. During an epoch, the CNN performs forward propagation to make predictions on the training examples, calculates the loss (error) between the predicted outputs and the actual targets, and then performs backpropagation to update the model's parameters (weights and biases) based on the calculated gradients. This process is typically done in mini-batches, where a subset of the training data is processed at a time to reduce memory requirements and improve computational efficiency.The number of epochs is a hyperparameter that needs to be specified before training the CNN. It determines how many times the entire training dataset will be iterated over by the optimization algorithm. Choosing the appropriate number of epochs is crucial because it can impact both the training time and the model's performance.If the number of epochs is too small, the model may not have sufficient time to learn complex patterns and generalize well to unseen data. On the other hand, if the number of epochs is too large, the model may overfit the training data, meaning it becomes too specialized in capturing the noise and details of the training examples, leading to poor performance on new, unseen data.The optimal number of epochs depends on several factors, including the complexity of the problem, the size of the training dataset, and the architecture of the CNN. It is often determined through experimentation and monitoring the model's performance on a separate validation dataset. Techniques such as early stopping can also be used to automatically stop training when the model's performance on the validation set starts to deteriorate.[1]
- Training Process - The training process in a convolutional neural network (CNN) involves preparing the dataset, designing the model architecture, initializing the weights, performing forward propagation to make predictions, calculating the loss, backpropagating to compute gradients, updating the parameters using an optimization algorithm, repeating this process for multiple epochs while monitoring performance on a validation set, tuning hyperparameters, and finally evaluating the trained model on a separate testing dataset to measure its effectiveness. The goal is to minimize the loss function and optimize the model's parameters to make accurate predictions on new, unseen data.[1]
- Validation Process - The validation process in a convolutional neural network (CNN) involves periodically evaluating the model's performance on a separate validation dataset. During training, after each epoch or a certain number of iterations, the model's predictions are compared to the true labels of the validation examples, and metrics such as loss, accuracy, precision, recall, or F1 score are calculated. This evaluation helps assess the model's generalization ability, indicating how well it is performing on unseen data. The validation results guide the selection of hyperparameters, such as learning rate or regularization techniques, and can help prevent overfitting by identifying when the model's performance on the validation set starts to deteriorate.[1]

Training and Testing Model

Fig 3 shows the block diagram illustrating the training process of CNN.

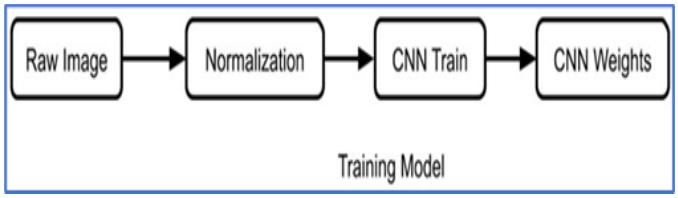

Figure 3 Training Model [6]

1. Raw Image: The training process starts with a raw image, which serves as the input to the CNN. The raw image typically consists of pixel values representing the intensity or colour information.
2. Normalization: Before feeding the raw image into the CNN, it is common practice to apply normalization techniques to improve training efficiency and performance. Normalization involves scaling the pixel values to a standardized range, such as [0, 1] or [-1, 1], to ensure that the input data has a consistent scale and distribution. This step helps prevent certain features from dominating the learning process.
3. CNN Training - The normalized image is then fed into the CNN for training. The CNN consists of multiple layers, including convolutional layers, pooling layers, activation functions and possibly fully connected layers. The purpose of the CNN is to learn and extract relevant features from the input image through the iterative process of forward propagation, loss calculation, backpropagation, and parameter updates. This process continues for multiple epochs until the model's performance converges or reaches a satisfactory level.
4. CNN Weights - During training, the CNN's weights (also known as parameters) are updated iteratively to minimize the loss function and improve the model's performance. The weights represent the learned patterns and features within the CNN. After the training process, the CNN model contains optimized weights that capture the knowledge learned from the training dataset.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

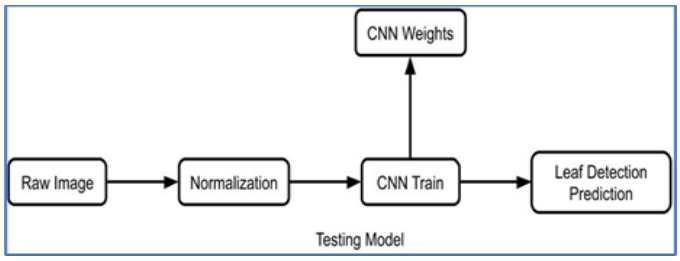

Figure 4 Testing Model [6]

The trained CNN model with its weights encapsulates the learned representations and can be used for various tasks, such as image classification, object detection, or image segmentation. These trained weights represent the network's knowledge of the important features and relationships in the input data and enable the model to make accurate predictions on new, unseen images.
Fig 4. shows the block diagram illustrating the testing process of CNN.

1. Raw Image - The testing process begins with a raw image of a leaf that needs to be evaluated for disease detection.
2. Normalization - Similar to the training process, the raw image is normalized to ensure consistent scaling and distribution of the pixel values. Normalization helps in achieving better performance and comparability between images during the testing phase.
3. CNN Training - Prior to testing, the CNN model is trained on a labeled dataset containing images of healthy and diseased leaves. This training phase involves feeding normalized images into the CNN and adjusting the model's weights iteratively to learn the discriminative features that differentiate healthy leaves from diseased ones. The CNN is trained to generalize these learned patterns to unseen images.
4. CNN Weights - After completing the training phase, the CNN model contains optimized weights that capture the learned representations and features. These weights represent the knowledge gained by the model during training about the distinguishing characteristics of healthy and diseased leaves.
5. Leaf Disease Detection - In the testing phase, the normalized raw image is passed through the trained CNN model with its learned weights. The model performs forward propagation, applying the learned features and patterns to the input image. The CNN outputs a prediction or probability score indicating the likelihood of the leaf being diseased.
   This output can help in determining the presence and type of disease in the leaf, aiding in early detection and appropriate intervention. The trained CNN model, with its optimized weights, acts as a powerful tool for automated leaf disease detection. By leveraging the learned representations from the training phase, the model can provide accurate and efficient predictions on new, unseen leaf images, facilitating timely decision-making for plant health management.

ResNet50:

ResNet-50 (short for Residual Network-50) is a convolutional neural network (CNN) architecture that consists of 50 layers. It was introduced by researchers at Microsoft Research in 2015 and has become one of the most popular and influential CNN architectures for various computer vision tasks, including image classification, object detection, and image segmentation.[1]
The architecture of ResNet-50 is based on a series of convolutional layers, followed by a global average pooling layer and a fully connected layer for classification. It has a total of 50 layers, which are organized into different blocks. The key building block of ResNet-50 is the "residual block," which consists of two or three convolutional layers with shortcut connections that bypass the intermediate layers. This allows the network to learn residual mappings instead of directly learning the desired mappings.
ResNet-50 has been pre-trained on large-scale image datasets such as ImageNet, which contains millions of labeled images. This pre-training enables the network to learn rich and discriminative features that can be transferred to other related tasks or fine-tuned on specific datasets with smaller amounts of labeled data.

Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

Architecture:

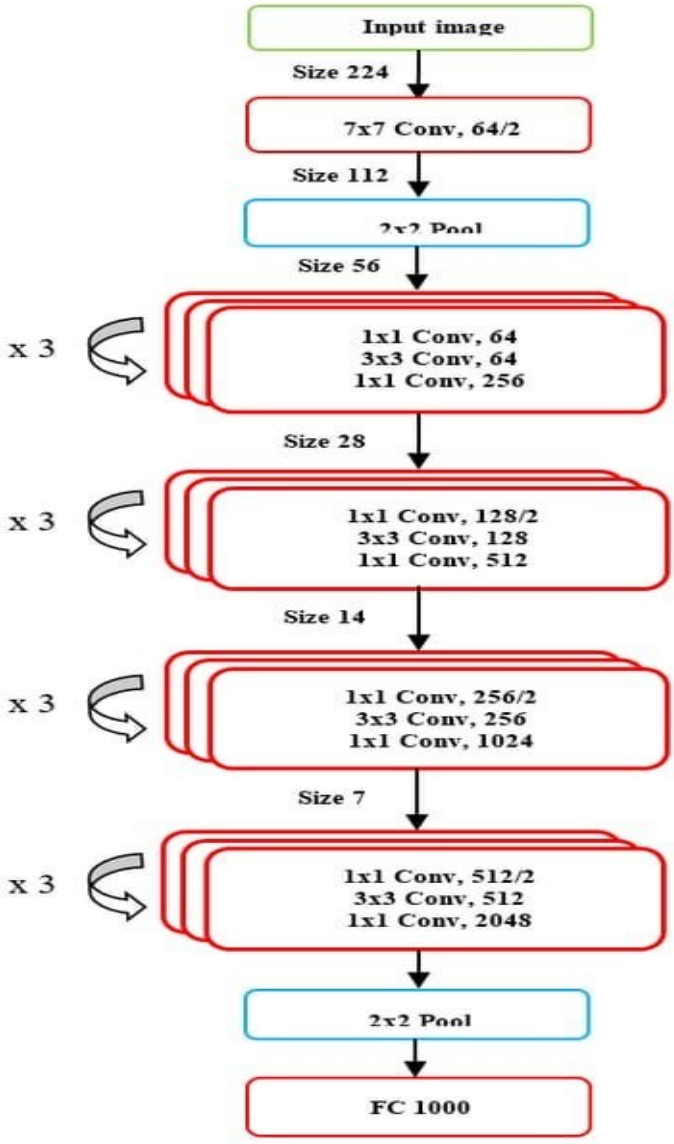

Figure 5 Resnet50 CNN architecture [9]

IV. RESULT

Despite the existence of numerous developed methods for detecting and classifying plant diseases based on diseased leaves, a commercially viable and efficient solution for disease identification is still lacking. While significant progress has been made in research and academia, translating these advancements into practical, effective solutions that meet commercial requirements, such as scalability, real-time performance, ease of use, and integration with existing agricultural practices, remains a challenge. In our study, we investigated the application of three distinct deep learning models, namely InceptionV3, ResNet50, and MobileNetV2, for the purpose of plant disease detection using images of both healthy and diseased leaves. Upon analyzing the results presented in Table 4, it was evident that ResNet50 consistently achieved remarkable training accuracy, validation accuracy, and test accuracy. As a result, we opted to employ the ResNet50 convolutional neural network model as the primary approach in our research endeavor.

Table 4 Comparison of Train Accuracy and Train Loss among Different CNN Models

Model	Train acc (%)	Train Loss
ResNet50	97.99	0.2316
InceptionV3	88.32	27.214
MobileNetV2	96.28	0.545

Table 5 Comparison of Validation Accuracy and Validation Loss among different CNN models

Model	Validation acc (%)	Validation Loss
ResNet50	96.68	2.031
InceptionV3	74.84	104.009
MobileNetV2	89.86	2.053

Table 6 Comparison of Epoch and Average time among different CNN models

Model	Epoch	Avg time (s/epoch)
ResNet50	10	286.8
InceptionV3	10	364.7
MobileNetV2	10	150.4

Table 7 Comparison of Test accuracy and Test loss among different CNN models

Model	Test acc (%)	Test Loss
ResNet50	96.73	2.327
InceptionV3	75.29	105.199
MobileNetV2	88.84	3.022

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 02, February 2024

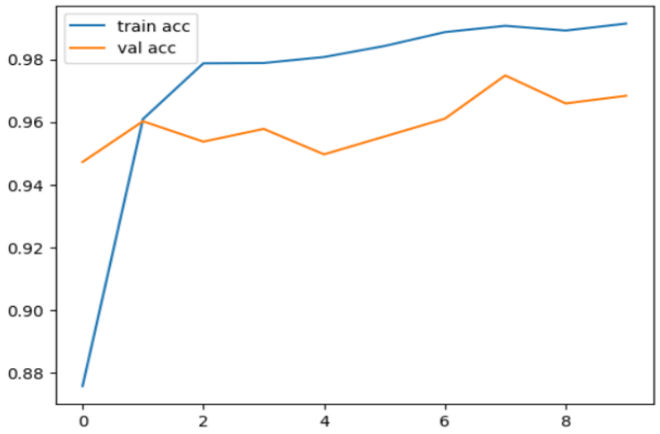

Figure 6 Training accuracy vs Validation accuracy

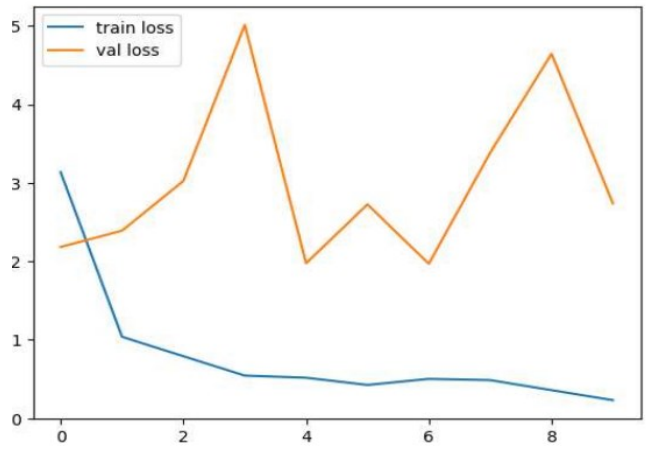

Figure 7 Training loss vs Validation loss

Website-

Step 1: Saved model into the local device
Step 2: Built a FastAPI web server, loaded the model, and finally tested it using the Postman application.
Step 3: Built a Testpad website in React JS that can support drag and drop the plant leaf images
Step 4: Dropping the image on the website, it calls the FastAPI backend to perform the inference.

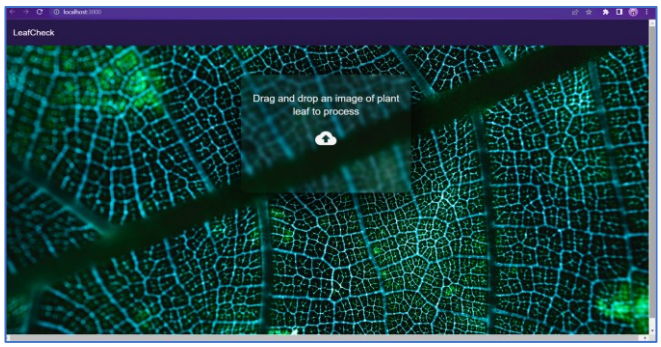

Figure 8 LeafCheck Website

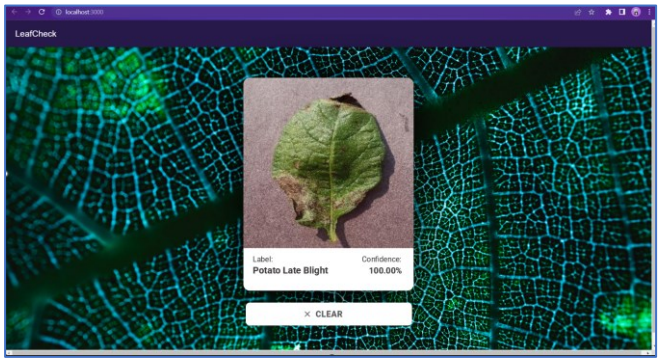

Figure 9 Result of Disease Detection of Potato Leaf

Fig 8. showcases the LeafCheck website interface, where users can conveniently drag and drop plant images. In this particular instance, a potato leaf image was dropped onto the interface. Moving on to Fig 9., the displayed outcome reveals the identification of the potato leaf as being afflicted with potato late blight, with a high confidence level of 100%.

V. CONCLUSION

We are successful in developing disease categorization methods that may be utilized to find plant leaf diseases. A deep learning model is developed that can automatically identify and categorize plant leaf diseases. Three species—tomato, potato, and bell pepper—are used to test the proposed paradigm. We were able to do image-processing operations as a result. Additionally, using the data, we were able to build the ResNet50 model, an advanced convolution model, and train it for prediction. Our model made a forecast that was 96.7% accurate. Additionally, the LeafCheck website which allows for disease detection was successfully built.

REFERENCES

[1] Sk Mahmudul Hassan, Arnab Kumar Maji, Michal Jasinki, Zbigniew Leonowicz, and Elzbieta Jasinka “Identification of Plant-Leaf Diseases Using CNN and Transfer-Learning Approach,” 9 June 2021
[2] Nishant Shelar, Suraj Shinde, Shubham Sawant, Shreyash Dhumal, and Kausar Fakir, “Plant Disease Detection Using CNN” 2022
[3] Sumit Kumar, Veerendra Chaudhary, Ms. Supriya Khaitan Chandra,” Plant Disease Detection using CNN”, 23 May 2021
[4] Jeon, Wang-Su, and Sang-Yong Rhee. "Plant leaf recognition using a convolution neural network." 2017
[5] Lee, Sue Han, et al. "How deep learning extracts and learns leaf features for plant classification.",2017
[6] Prakanshu Srivastava, Vibhav Awasthi, Vivek Kumar Sehu, Pawan Kumar Pal,”Plant Disease Detection Using Convolutional Neural Networks”, Article in International Journal of Advanced Research, January 2021
[7] Serawork A. Wallelign, Mihai Polceanu, Cedric Buche, “Soybean Plant Disease Identification Using Convolutional Neural Network”, Oct 2014
[8] Dimitrios I. Tsitsigiannis, Polymnia P. Antoniou, Sotirious E. Tjamos, Epaminondas J. Paplomatas, “Major Diseases of Tomato, Pepper and Eggplant in Greenhouses”, The European Journal of Plant Science and Biotechnology, 27 Oct, 2008.
[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, “Deep Residual Learning for Image Recognition”, 10 Dec 2015.
[10] Sharada P. Mohanty, David P. Hughes, Marcel Salathe, “Using Deep Learning for Image- Based Plant Disease Detection”, Front Plant Sci., 22 Sept, 2016.