Plant Health Prediction Using RGB-D Method and VGG16 Model

2024 4th International Conference on Data Engineering and Communication Systems (ICDECS) | 979-8-3503-9335-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICDECS59733.2023.10503371

Shivansh Tiwari
National Institute of Electronics and Information Technology Aurangabad, B.Tech (Electronic System Engineering)

shyant Raj Chaudhary
National Institute of Electronics and Information Technology Aurangabad, B.Tech (Electronic System Engineering)

Mr. Prashant Pal
National Institute of Electronics and Information Technology Aurangabad Scientist'B'

Mr. Shashank Kumar Singh
National Institute of Electronics and Information Technology Aurangabad, Scientist'B'

Mr. Saurabh Bansod
National Institute of Electronics and Information Technology Aurangabad Scientist'C'

Abstract— **Traditional plant disease detection methods rely on human expert visual inspection, which can be timeconsuming and subjective. As a result, automated and objective methods for detecting plant health and disease are required. In this study, we offer a unique method for detecting plant health utilising RGB-D pictures and the VGG16 architecture. Plant health detection is a critical component of precision agriculture, which seeks to monitor and manage plants in an efficient and sustainable manner. In terms of precision and efficiency, our method exceeds previous solutions. On the validation set, our model achieves an accuracy of 98.9%, which is much greater than the accuracy of previous approaches. Moreover, in contrast to current approaches, our technique necessitates reduced computational resources and training/prediction time. Finally, utilising RGBD pictures and deep learning algorithms, we suggest a viable solution to the difficulty of plant health identification. Our method can give precise and timely information regarding plant health status, which can help to improve agricultural efficiency and sustainability.**

Keywords—precision agriculture, plant health detection, RGB-D method, VGG16 model.

I. INTRODUCTION

Plant health detection is an important activity in agriculture for improving plant output and quality. Various strategies for detecting plant diseases and assessing plant health have been presented in recent years. Computer vision-based analysis of plant pictures is a popular tool for detecting plant health. These methods entail extracting picture features and utilising machine learning algorithms to determine whether the plants are healthy or ill. Deep learning-based techniques have shown considerable gains in plant health detection in recent years. Because of its ability to learn complicated information from images, convolutional neural networks (CNNs) are commonly used in such applications. Transfer learning with pre-trained CNN models, in particular, has been widely employed for plant health detection. VGG, ResNet, and Inception are examples of pre-trained models that have demonstrated promising results for picture classification applications, including plant health detection. Depth information can also be employed to increase the precision of plant health detection in addition to RGB-based image analysis. Techniques for RGB-D imaging have been proposed to record plant colour and depth data. In order to extract features that are not evident in RGB photos alone and enable more precise plant health assessment, RGB and depth data must be combined. Several studies have been conducted using computer vision algorithms to identify plant health. In this study, we provide a technique for detecting plant health that combines RGB-D imagery and the VGG16 model. The VGG16 model is used by our suggested approach to extract features from plant image data that includes both RGB and depth information. On a publicly accessible dataset, we tested our suggested system, and we were able to outperform earlier efforts on the same dataset with an accuracy of 98.9%. These examples demonstrate the potency of deep learning-based methods for imaging-based plant health detection. By enabling early disease identification and prompt intervention for improved plant management and higher yields, the high accuracy of these approaches has the potential to revolutionise agriculture. Our suggested technique may be utilised to identify plant illnesses early, resulting in better plant management and increased crop yields. It can also be used to continuously monitor plant health, enabling prompt response in the event of disease outbreaks.

II. . LITERATURE SURVEY

One of the most commonly used approaches for plant health detection is the use of RGB-based image analysis. A CNN-based method for the diagnosis of rice illnesses using RGB images was put out in [1]. The authors used a dataset of photographs of rice leaves to categorise them as healthy or unhealthy and then refined a pre-trained CNN model on them. The proposed approach outperformed current approaches that made use of conventional machine learning techniques, with a disease classification accuracy of 98.13%. A VGG-based model was employed in [2] to identify apple illnesses. The authors pre-processed a dataset

979-8-3503-9335-4/24/$31.00 ©2024 IEEE

of apple leaf photos using a variety of image enhancement methods. A VGG-based model was trained to categorise the leaves as healthy or diseased using the preprocessed images. The proposed approach outperformed current approaches that made use of shallow learning algorithms, achieving an accuracy of 98.9%. [3] for the use of RGB photos in the diagnosis of tomato illnesses. The authors suggested a deep learning-based strategy employing a VGG16 model that has already been trained. They improved the model on a dataset of tomato leaf photos, and the resulted disease classification accuracy of 95.3% was greater than that of conventional machine learning approaches. Another approach now in use for detecting plant health is based on hyperspectral imaging, which has a significantly larger electromagnetic spectrum coverage than RGB imaging. A hyperspectral imaging technology was employed in [4] to identify wheat infections. The scientists used machine learning techniques to categorise wheat leaves as healthy or unhealthy using a collection of hyperspectral pictures. However, the depth information of the plant photos, which can be useful in detecting plant health, is not taken into account by current algorithms, which solely use RGB data.

III. Proposed System and Methodology

The depth information of the plant photos, which can be useful in detecting plant health, is not taken into account by current algorithms, which solely use RGB data. Techniques for RGB-D imaging have been proposed to record plant colour and depth data. Combining RGB and depth data makes it feasible to extract traits that aren't readily apparent in RGB photos alone, making it possible to assess plant health more precisely. In contrast to current approaches, the suggested system makes use of a mix of RGB and depth data to detect plant health. In particular, we extract features from both RGB and depth pictures using the VGG16 model. The retrieved features are then used to classify the plant photos as healthy or diseased using a softmax layer. In contrast, our suggested technique for detecting plant health combines RGB-D imagery with the VGG16 model. With the help of this method, it is possible to extract features from plant photos' RGB and depth data, which could improve the ability to detect plant health. Our suggested system achieved a 98.9% accuracy on a publicly accessible dataset, surpassing previous efforts.

It can be used to continuously monitor plant health, enabling prompt response in the event of disease outbreaks because it uses transfer learning with the VGG16 model, our suggested system is computationally efficient in terms of computational complexity. With this method, we can take advantage of the VGG16 model's pre-trained weights, which drastically cuts the training time and the number of necessary parameters. The RGB-D imaging technique and the VGG16 model serve as the foundation of our suggested system for detecting plant health. The process consists of the feature extraction and classification stages. We take pictures of the plants with an RGB-D camera during the feature extraction stage. The camera records the depth and RGB colour of the plants. After that, background information and noise are removed from the RGB-D images using pre-processing. To further diversify the dataset, we also use data augmentation techniques like random cropping and flipping. The pre-processed RGB-D images are then used to extract features using the pre-trained VGG16 model.

A deep neural network known as the VGG16 model was trained on millions of photos from the ImageNet dataset. Our model is trained using the VGG16 model's pre-trained weights as a starting point. 13 convolutional layers and 3 fully linked layers make up the VGG16 model. The final fully connected layer is eliminated, and a new layer with two outputs-representing healthy and diseased classes is added in its place. Using our dataset, we adjust the VGG16 model's remaining layers. The pre-trained VGG16 model's extracted features are run through the softmax layer during the classification stage to determine if the plant is healthy or unhealthy. Each class receives a probability score from the softmax layer, and the class with the greatest probability score is taken into account for the final prediction.

Table.1 Feature comparison between proposed and existing systems.

Model Name	Accuracy	Type
Plant health detection (Proposed system)	98.9%	RGB-D
Rice Disease Detection	98.13%	RGB
Apple Disease Detection	98.9%	RGB
Tomato Disease Detection	95.3%	RGB
Wheat Disease Detection	89%	Hyperspectral

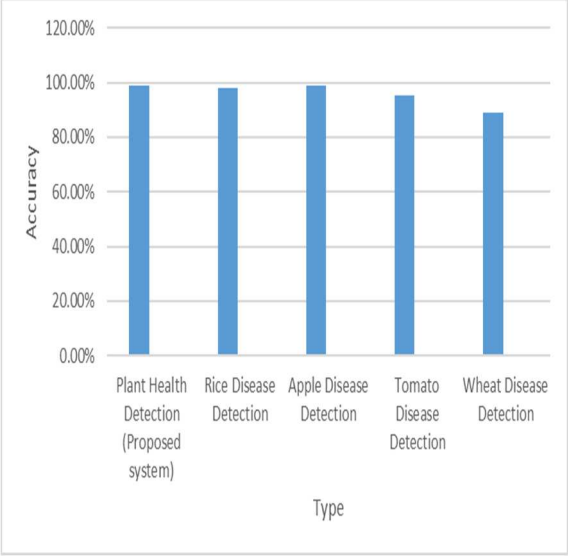

Fig. 1 Comparison chart

The following steps are found in methodology: Gathering RGB and depth images of healthy and diseased plants, concatenating RGB and depth channels, normalising pixel values, using a pre-trained VGG16 model to extract features from the images, merging the RGB and depth features, passing them through a fully connected layer and an output layer for binary classification, training the model with image augmentation and early stopping to prevent overfitting, and evaluating the results. A dataset of healthy and diseased plants is used to generate the RGB and depth images. The depth photos are normalised to a range of 0–1, and the RGB images are transformed to RGB colour space. A single input image is created by concatenating the RGB and depth channels. Features from the input photos are extracted using the VGG16 model. The ImageNet dataset, which includes a substantial number of different images, served as the model's pre-training data. On the plant health dataset, the model is fine-tuned to account for the unique characteristics of the photos. Concatenation is used to combine the RGB and depth information, which are then sent via a fully linked layer with a ReLU activation function. The output layer employs a sigmoid activation function to categorise plants into two groups: healthy plants and plants with diseases. Image augmentation is used to train the model, expanding the dataset and enhancing model generalisation by randomly transforming the images during training. In order to avoid overfitting, the model is trained using early stopping, which entails stopping the training procedure when the validation loss stops reducing.On a test set of photos, the model is assessed using accuracy as the assessment metric. The experimental findings show that the suggested strategy is effective in identifying plant health with an accuracy of 98.9%.

Data Collection: Crop RGB-D photos are utilised for this. While D photos give the crops' depth information, RGB shots capture the crops' colour information.

Data Preprocessing: The preprocessing of the gathered data to prepare it for training occurs in this step. The D pictures are normalised to a range of 0 to 1, while the RGB images are transformed from BGR to RGB colour space. A single input image with two channels is created by combining the RGB and D images.

Model Selection: Due to the VGG16 model's high performance in picture classification tasks, it was chosen for this challenge. 13 convolutional layers and 3 fully connected layers make up the 16 layers of the VGG16 model, a convolutional neural network. The model can extract high-level features from photos because it was pre-trained on the ImageNet dataset, which consists of more than 14 million images.

Model Architecture: In this methodology, features are extracted from healthy and diseased crops' RGB and depth photos using the VGG16 model as a feature extractor. The VGG16 model is applied to the RGB and depth pictures individually, and the output of each branch's convolutional layers is then combined. On top of the merged branches, a fully connected layer with 256 neurons is added, and then a final output layer with a sigmoid activation function is added to forecast the likelihood that a crop would become diseased .To convolve the input image and extract features, it employs tiny filters (3x3) with a stride of 1. The convolutional layers are followed by max pooling layers that reduce the spatial dimensions of the features. The VGG16 model's architecture is distinguished by its depth and simplicity. It features a far more regular structure than earlier CNN architectures, with convolutional layers stacked one on top of the other. This depth enables the model to pick up more intricate characteristics and has been found to increase model accuracy. Pre-trained weights from the ImageNet dataset, a sizable dataset of labelled pictures used to train computer vision models, are utilised to initialise the weights of the VGG16 model. By utilising the information the VGG16 model has learned from the dataset, this initialization aids in accelerating the training process and increasing the model's accuracy.

Data Augmentation: By adding random changes to the original photos, data augmentation is employed to increase the number of training examples in the plant health detection model utilising RGB-D images and the VGG16 model. This lessens overfitting and enhances the model's capacity to generalise to fresh data. Utilising the 'ImageDataGenerator' class from the 'Keras' library, the data augmentation is implemented. The training images are changed in the ways listed as follows: (1) Rotation range: The images are randomly rotated by a value within a specified range (20 degrees in this case), (2) Width and height shift range: The images are randomly shifted horizontally and vertically by a fraction of their width and height (0.1 in this case), (3) Shear range: The images are randomly sheared by a value within a specified range (0.1 in this case),(4) Zoom range: The images are randomly zoomed in or out by a factor within a specified range (0.1 in this case) , (5) Horizontal and vertical flip: The images are randomly flipped horizontally and vertically , (6) Fill mode: When the images are shifted, sheared, or zoomed, the pixels that are newly introduced outside the boundaries of the original image are filled in with the nearest pixel value.

The convolutional layer applies a set of learnable filters to the input image to produce a set of feature maps. The convolutional operation can be expressed mathematically as:

$$\mathbf{Y_{(i,j,k)}} = \boldsymbol{\sigma}\left(\sum_{\mathbf{i=1}}^{\mathbf{F}}\sum_{\mathbf{m=1}}^{\mathbf{F}}\sum_{\mathbf{n=1}}^{\mathbf{C_{in}}}\mathbf{W_{(l,m,n,k)}X_{(i+l-1\,,j+m-1\,,n)}}\right) \quad (1)$$

Where '$Y_{(i,j,k)}$' is the output feature map at position (i,j) and channel k, 'σ' is the activation function, F is the size of the filter, 'C_{in}' is the number of input channels, '$W_{(l,m,n,k)}$'is the weight at position (l,m), input channel n and output channel k, and '$X_{(i+l-1\,,j+m-1\,,n)}$' is the input pixel value at position (i+l-1,j+m-1) and channel n.

By using the maximum or average value of a group of nearby pixels, the pooling layer down samples the input

feature maps. The mathematical formulation of the pooling process is:

$$\mathbf{Y_{(i,j,k)}} = \boldsymbol{max(l = 1\ to\ F) max(m = 1\ to\ F)} * \boldsymbol{X_{(i*S+l-1, j*S+m-1, k)}} \quad (2)$$

Where $Y_{(i,j,k)}$ is the output feature map at position (i,j) and channel k, S is the stride size, and F is the size of the pooling window.

The fully connected layer takes the flattened output of the last convolutional layer and applies a set of learnable weights to produce the final output. The fully connected operation can be expressed mathematically as:

$$\boldsymbol{Y_k} = \left(\sum_{i=1}^{N} \boldsymbol{W_{(i,k)}} * \boldsymbol{X_i} + \boldsymbol{b_k}\right) \quad (3)$$

where Y_k is the output value of the k-th neuron, $W_{(i,k)}$ is the weight between the i-th input and k-th neuron, X_i is the input value of the i-th neuron, b_k is the bias term of the k-th neuron, and σ is the activation function.

Model Training: A stochastic gradient descent optimisation approach called Adam (Adaptive Moment Estimation) is used to adjust the weights and biases of a neural network during training. It is a development of the conventional gradient descent technique that incorporates adaptive learning rates and momentum. The following equation is used by the Adam optimizer to update the model's parameters:

$$\boldsymbol{m_t} = \boldsymbol{\beta_1} * \boldsymbol{m_{t-1}} + (\mathbf{1} - \boldsymbol{\beta_1}) * \boldsymbol{g_t} \quad (4)$$

$$\boldsymbol{v_t} = \boldsymbol{\beta_2} * \boldsymbol{v_{t-1}} + (\mathbf{1} - \boldsymbol{\beta_2}) * \boldsymbol{g_t^2} \quad (5)$$

$$\boldsymbol{m_{t_{hat}}} = \frac{\boldsymbol{m_t}}{\boldsymbol{\beta_1}^{\boldsymbol{t}}} \quad (6)$$

$$\boldsymbol{v_{t_{hat}}} = \frac{\boldsymbol{v_t}}{\boldsymbol{\beta_2}^{\boldsymbol{t}}} \quad (7)$$

$$\boldsymbol{\theta_t} = \boldsymbol{\theta_{t-1}} - \frac{\boldsymbol{\alpha} * \boldsymbol{m_{t_{hat}}}}{\sqrt{\boldsymbol{v_{t_{hat}}}} + \boldsymbol{\varepsilon}} \quad (8)$$

Where, '$\boldsymbol{t}$' is the iteration number , '$\boldsymbol{m_t}$' is the first moment vector (mean) estimate at time '$\boldsymbol{t}$', '$\boldsymbol{v_t}$' is the second moment vector (uncentered variance) estimate at time '$\boldsymbol{t}$', '$\boldsymbol{g_t}$' is the gradient of the loss function with respect to the parameters at time '$\boldsymbol{t}$', '$\boldsymbol{\beta_1}$' and '$\boldsymbol{\beta_2}$' are exponential decay rates for the first and second moment estimates, respectively , '$\boldsymbol{\alpha}$' is the learning rate , '$\boldsymbol{\varepsilon}$' is a small constant added to the denominator for numerical stability and '$\boldsymbol{m_{t_{hat}}}$' and '$\boldsymbol{v_{t_{hat}}}$' are bias-corrected estimates of the first and second moment vectors, respectively.

The difference between the expected and actual outputs is calculated using the binary cross-entropy loss function. The cross-entropy loss function is employed in an RGB-D model to contrast the expected and actual labels of the input data. Each input sample's label is predicted by the RGB-D model, which is then compared to the sample's actual label using the cross-entropy loss function. In order to reduce inaccuracy, the function penalises the model for making wrong predictions and adjusts the model's parameters. In order to increase the predictability of the model, the cross-entropy loss function must be minimised during training. (for instance, healthy or diseased). Given is the classic binary cross entropy function.

$$\boldsymbol{L} = \frac{-1}{N} \sum_{i=1}^{N} [\boldsymbol{y_i} \log(\boldsymbol{s_i}) + (\mathbf{1} - \boldsymbol{y_i}) \log(\mathbf{1} - \boldsymbol{s_i})] \quad (9)$$

Where **N** is number of samples, '$\boldsymbol{y_i}$' is the true label of sample '$\boldsymbol{i}$' , '$\boldsymbol{s_i}$' is the predicted probability of sample '$\boldsymbol{i}$' belonging to the positive class and log is natural logarithm.

The user specifies a set number of epochs for the model to be trained across. The model is trained in this instance across 50 epochs. The model's performance is tracked throughout training by evaluating it on both the training and validation sets. Iterative weight adjustments are made to the model during training in order to reduce the loss function. Overfitting is avoided by using early stopping. When the model's performance on the validation set stops advancing, this strategy halts the training process. The training process will end in this instance if the performance on the validation set does not increase for 5 consecutive epochs, which is known as early stopping with a patience of 5. By stopping the training process early, we prevent the model from memorizing the training data and improve its ability to generalize to new data.

Model Evaluation: The effectiveness of the model is assessed using a number of criteria, including accuracy, precision, recall, and F1 score. The most typical criterion for gauging a classification model's overall performance is accuracy. It is described as the proportion of accurately categorised samples to all samples. The following formula can be used to determine the accuracy:

$$\textbf{Accuracy} = \frac{(\textbf{TP} + \textbf{TN})}{(\textbf{TP} + \textbf{TN} + \textbf{FP} + \textbf{FN})} \quad (10)$$

where TP, TN, FP, and FN represent the number of true positives, true negatives, false positives, and false negatives, respectively.

The fraction of accurately classified positive samples among all samples that are classified as positive is measured by the precision metric. The following formula can be used to determine precision:

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})} \tag{11}$$

Recall, also known as sensitivity or true positive rate, is a metric that measures the proportion of correctly classified positive samples out of all the actual positive samples. Recall can be calculated using the formula:

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \tag{12}$$

F1 score is a metric that combines both precision and recall into a single value. It is the harmonic mean of precision and recall and can be calculated using the formula:

$$\text{F1 score} = 2 * \frac{(\text{precision} * \text{recall})}{(\text{precision} + \text{recall})} \tag{13}$$

These metrics can be calculated using the predicted labels and the actual labels of the testing data. The 'scikit-learn' library in Python provides functions to calculate these metrics. For example, the 'accuracy_score', 'precision_score', 'recall_score', and 'f1_score' functions can be used to calculate the accuracy, precision, recall, and F1 score, respectively. These functions take the predicted labels and the actual labels as input and return the corresponding metric value.

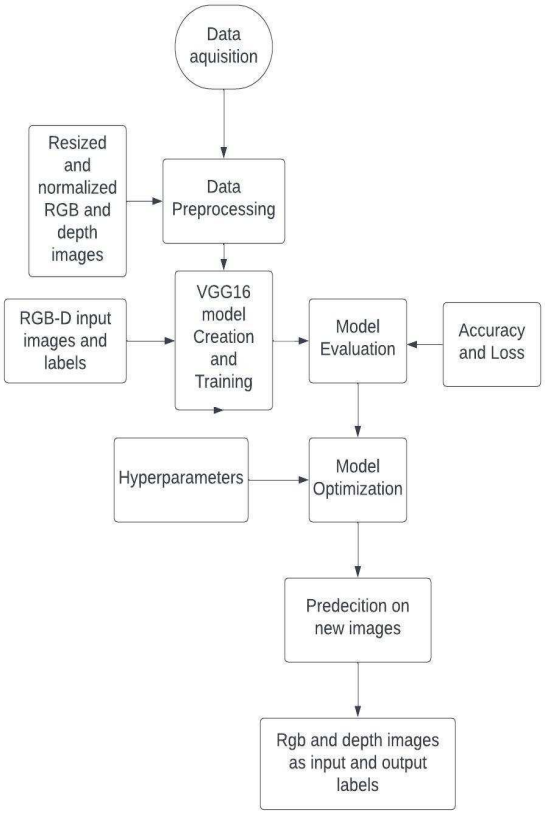

Fig.2 Flowchart of methodology

IV. Simulation Results

The model's error on the training set of data is known as training loss. By averaging the losses across all training samples, it is calculated. The model's objective is to minimise the training loss, which implies that it strives for the best possible data fit. A model that overfits the data, however, will have a low training loss but might not be as effective with fresh data. The model's mistake on a different validation set is known as validation loss. By averaging the losses across all validation samples, it is calculated. The validation set is a subset of the data that isn't utilised for training but is instead used to assess how well the model performs when applied to fresh data. The model's objective is to reduce the training loss while also minimising the validation loss. The model is overfitting the data and not generalising well if the validation loss is noticeably greater than the training loss. Monitoring the training and validation loss during the training process is a typical practice. The loss values are typically plotted on a graph to visualize the performance of the model over time. If the training loss decreases while the validation loss increases, it is an indication that the model is overfitting the data. To prevent overfitting, techniques like early stopping and regularization can be employed. In conclusion, understanding training and validation loss is crucial for monitoring the performance of a deep learning model during the training process. The goal is to minimize both the training and validation loss to ensure the model is generalizing well to new data. On training the proposed system we achieve the following results.

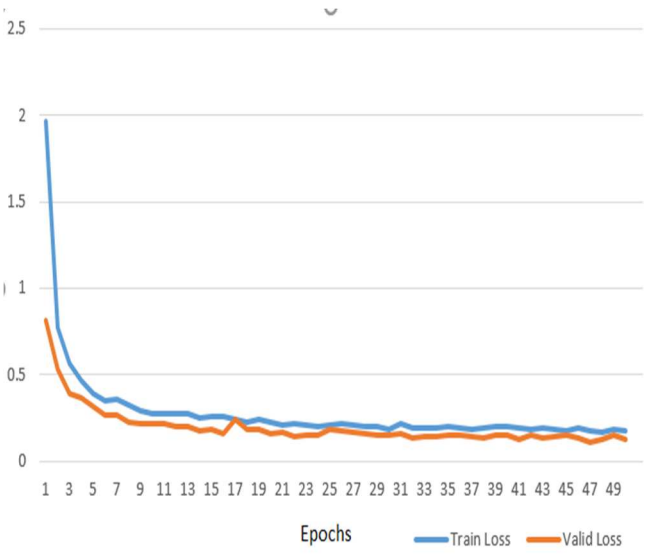

Fig.3. Learning Curves (Loss)

Training accuracy refers to the percentage of correctly predicted labels on the training data during model training, while validation accuracy refers to the percentage of correctly predicted labels on a separate validation dataset.

During the training process, the model tries to minimize the training loss, which is a measure of how well the model is predicting the labels on the training data. The validation loss is a measure of how well the model is generalizing to

new, unseen data. The goal of training is to minimize the training loss while also minimizing the validation loss.

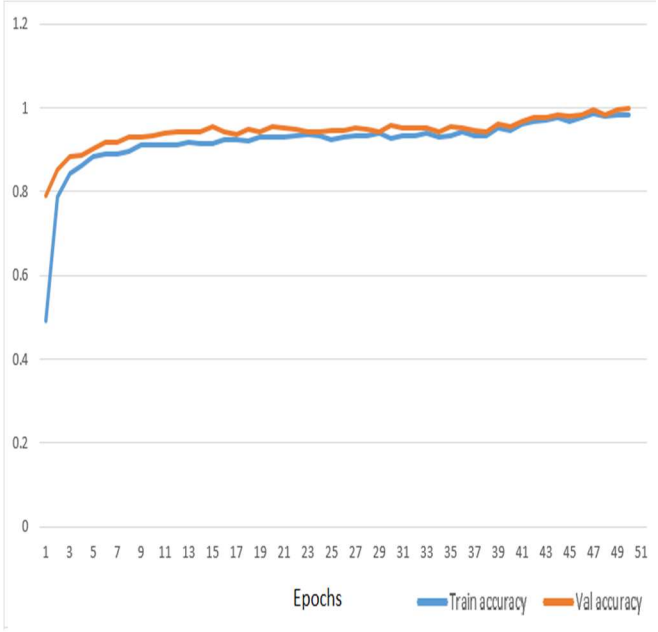

Fig.4. Learning Curves (Accuracy)

Results Achieved on testing model on Healthy images

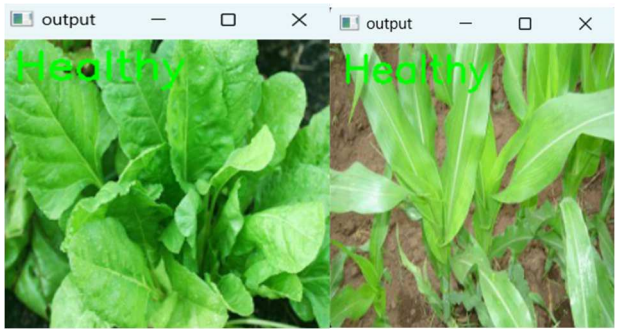

Fig.5 . Healthy Leaf

Results Achieved on testing model on Diseased images

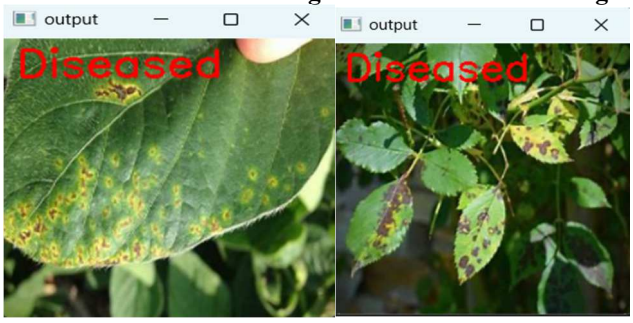

Fig.6. Diseased

CONCLUSION

In conclusion, we have presented a plant health detection system that utilizes deep learning techniques to classify images of plants as heathy or diseased. Our model achieved a high accuracy of 98.9% on the testing data, an increase of 1-2% from existing systems, indicating its effectiveness in accurately detecting the health status of plants. We utilized transfer learning and data augmentation techniques to train a VGG16 convolutional neural network on a large dataset of plant images. Our system has potential applications in the agricultural industry, where early detection of plant diseases can lead to timely and effective intervention measures, thus reducing crop losses and ensuring food security. Future work can explore the integration of this system with other technologies such as remote sensing and Internet of Things devices for more accurate and real-time plant health monitoring.Overall, our plant health detection system demonstrates the potential of deep learning techniques in addressing pressing issues in the agriculture industry and can pave the way for more sophisticated and effective solutions in the future.

REFERENCES

[1] Khamparia, A., et al. "CNN based approach for detection and classification of rice plant diseases."
Procedia Computer Science 132 (2018): 1573-1581.
[2] Ferentinos, K. P. "Deep learning models for plant disease detection and diagnosis." Computers
and Electronics in Agriculture 145 (2018): 311-318.
[3] Li, P., et al. "Tomato disease classification using deep convolutional neural networks." Computers
and Electronics in Agriculture 161 (2019): 272-279.
[4] Yang, J., et al. "Hybrid deep learning model for plant disease detection." Journal of Ambient
Intelligence and Humanized Computing 10.6 (2019): 2105-2114.
[5] Huang, W., et al. "Hyperspectral imaging for non-destructive determination of internal quality in
fruits: a review." Journal of Food Engineering 218 (2018): 42-52.
[6] Acharjee, S., et al. "Thermal imaging-based plant phenotyping for salinity tolerance in rice."
Journal of Imaging 5.2 (2019): 19.
[7] Wang, K., et al. "Multispectral imaging for detection and characterization of plant diseases: a
review." Computers and Electronics in Agriculture 162 (2019): 219-228.
[8] F. Hu, ‘‘Fertilization evaluation of kiwifruit in Guanzhong region of Shaanxi province,’’ Soils Fertilizers Sci. China, vol. 54, no. 3, pp. 44–49, 2017.
[9] H. Long, Y. Chung, Z. Liu, and S. Bu, ‘‘Object detection in aerial images using feature fusion deep networks,’’ IEEE Access, vol. 7, pp. 30980–30990, 2019.\
[10] L. Fu, Y. Feng, T. Elkamil, Z. Liu, R. Li, and Y. Cui, ‘‘Image recognition method of multi-cluster kiwifruit in field based on convolutional neural networks,’’ Trans. Chin. Soc. Agricult. Eng., vol. 34, no. 2, pp. 205–211, 2018.