Rice Disease Detection using Deep Learning VGG-16 Model and Flask

A
Dissertation
submitted
in partial fulfilment
for the award of the Degree of
Master of Technology

in Department of Computer Science and Engineering

(with specialization in Plant Disease Detection)

Supervisor

Name: R.S. Sharma

Designation: Assistant Professor

Submitted By:

Name of Candidate: Bhairu Jangid

Enrolment No.: 20EUCCS601

Department of Computer Science and Engineering

University Teaching Department

Rajasthan Technical University

March 2023

Candidate's Declaration

I hereby declare that the work, which is being presented in the Dissertation, entitled "Rice Disease Detection using Deep Learning VGG-16 Model and Flask" in partial fulfilment for the award of Degree of "Master of Technology" in Deptt. of Computer Science and Engineering with Specialization in Plant Disease Detection and submitted to the Department of Computer Science and Engineering, University Teaching Department, Rajasthan Technical University is a record of my own investigations carried under the Guidance of Shri R.S. Sharma, Department of Computer Science and Engineering, University Teaching Department, Rajasthan Technical University .

I have not submitted the matter presented in this Dissertation any where for the award of any other Degree.

(Name and Signature of Candidate)

Plant Disease Detection,
Enrolment No.: 20EUCCS601

University Teaching Deptt, Rajasthan Technical University,

Counter Signed by
Ass. Prof. R.S. Sharma

Contents

List of Figures

1 Abstract

As we all know 70 percent population of our country depends on agriculture. Crop fields have high impact on our life. Proper disease management in agriculture leads to a good growth. Farmers have no idea about disease management so they produce less production. India is the second largest producer of rice and wheat.Agriculture is an important sector of Indian of Indian economy as it contributes about 17 percent of the total GDP and provide employment to over 60 percent population.Indian agriculture registered impressive growth over last few years, but due to various kinds of diseases this production rate was affected very much.In agriculture field disease management is the process of reducing disease in crops to enlarge quality and quantity of harvest yield.Creatures that cause contagious disease in crops include fungi, bacteria, viruses, virus-like-organisms, phytoplasmas, protojoa, etc. These organisms creates different types of diseases in plants like bacterial leaf blight, brown spot, hispa, leaf smut, black spot, powdery mildew, downy mildew, canker, rust, late blight, etc. In this research paper we are trying to create an web based application that identify some diseases from these which affects rice plant.We are implementing an Convolution Neural Network(CNN) based model named VGG-16 for classification of diseases in rice plant.As CNN is very highly accurate for prediction in image classification.

Keywords: Disease detection, Image processing, CNN, VGG-16, Deep learning.

2 Acknowledgements

This thesis is premised on research work guided for ”Rice Disease Detection using Convolution Neural Network”.This work would not be possible without one person whose contribution can’t be ignored.
I consider it an glory to work under my guide **Ass. Prof. R.S. Sharma**. This thesis is harvest of their valuable guidelines and proper directions regard this work.He was always available to monitor me for this research work at any time. Thanks to **Ass. Prof. R.S. Sharma** for guiding me in this research work and always support me.
I am also obliged to Rajasthan Technical University for providing everything from faculty guide to resources for this research work. I would like to thanks all my batch-mates and other people who have directly or indirectly helped me to realize this work.

3 Chapter 1 Introduction

3.1 Introduction to Plant Disease Detection

The most important sector of our Economy is Agriculture. Various types of diseases harms the plant leaves and minimise the growth. Regular support to plant leaves is the benefit in agricultural products.
Plant disease detection is the process of identifying disease in any plant. It is the process of classifying diseased and healthy part of an plant. We do this by the help of a well mannered data-set. We can use any technique for this purpose to identifying disease.
Plant disease detection is an important aspect of crop management, as it allows early identification and treatments of diseased plants. This can help to prevent the spread of disease to other healthy plants, as well as minimize crop loss.There are variety of techniques that used for plant disease detection, including visual inspection, laboratory testing and the use of technology such as imaging and sensor system.
One common method of plant disease detection is visual inspection, which involves looking for signs of disease on plant, such as discoloration, wilting or the presence of disease causing organisms. This method can be effective for identifying some diseases, but it can be time consuming and may not detect disease that don't have visible symptoms. So visual inspection is a common method, but new technologies such as imaging and sensor system are showing promise as a way to detect disease at an early stage and with greater accuracy.
It is very difficult to invigilate plants developing in wet-lends like, rice plants, that is the idea of our study in this, as contrast to other plants species. So we use deep learning technique for detecting rice leaf diseases.

Here is the basic steps involved in plant disease detection using image-based systems:-

1. Image Acquisition: The first step in plant disease detection is to acquire images of the plants that are suspected of being infected. This can be done using various methods, such as handheld cameras or drones.

2. Image Pre-processing: The acquired images are then pre-processed to remove any noise or artifacts that may interfere with the disease detection process. This can involve tasks such as image resizing, normalization, and color correction.

3. Feature Extraction: In this step, relevant features are extracted from the pre-processed images. These features can include color histograms, texture features, and shape descriptors, among others.

4. Model Training: The extracted features are then used to train a machine learning model, such as a deep neural network or support vector machine (SVM), to recognize specific diseases.

5. Disease Detection: Once the model has been trained, it can be used to

detect diseases in new images of plants. This involves feeding the new image through the trained model, which outputs a prediction of whether the plant is healthy or diseased, and if diseased, which disease it is.

6. Decision Making: Finally, the output of the disease detection system can be used to make informed decisions about the management of the infected plants. This can involve applying appropriate treatments or removing the infected plants to prevent the spread of disease to neighboring crops.
Here is an illustration of Plant Disease Detection system shown in figure 1.

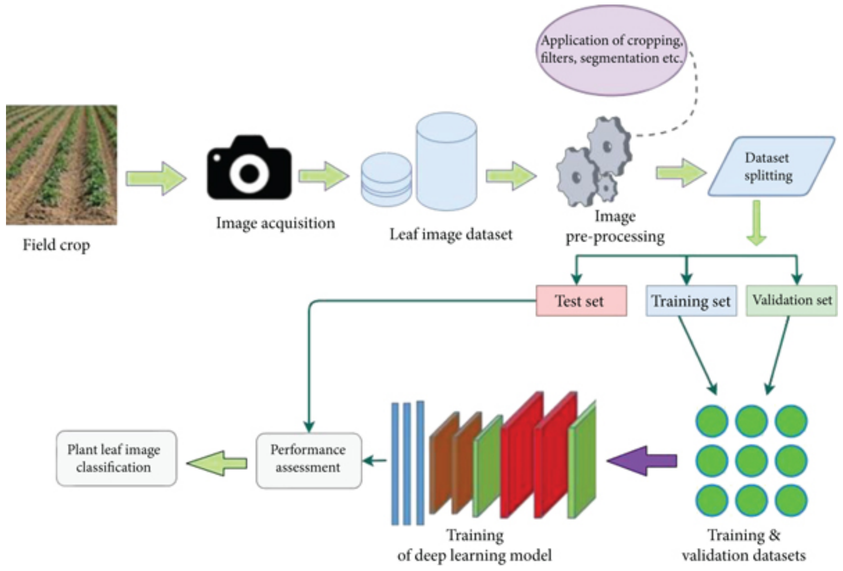

Application of cropping, filters, segmentation etc.
Field crop
Image acquisition
Leaf image dataset
Image pre-processing
Dataset splitting
Test set
Training set
Validation set
Plant leaf image classification
Performance assessment
Training of deep learning model
Training & validation datasets

Figure 1: Illustration of Plant Disease Detection system

3.1.1 Types of Rice Diseases

Figure 2 shows all types of rice diseases:-
1.Rice Stripe Virus Disease
2.Bacterial Sheath Brown Rot
3.Stem Rot
4.Node Neck Blast
5.Sheath Rot
6.Bakanae
7.Red Stripe
8.Narrow Brown Spot
9.Leaf Scald
10.Tungro
11.Sheath Blight

12.Rice Ragged Stunt
13.Rice Grassy Stunt
14.False Smut

Out of these rice diseases we work on mainly three types of diseases, Brown Spot, Leaf Blast, and Neck Blast in this research paper. Image samples of these diseases are mentioned further in figure 4 this report.

3.1.2 About Digital Image Processing

Digital Image Processing (DIP) is a field of study focused on the processing and analysis of digital image using mathematical algorithms and computer programming techniques. DIP involves a series of steps as shown in figure 4 that manipulate digital images to enhance their quality, extract useful information, and transform them for different purposes.
First step is image acquisition means capturing images from fields or directly using any data-set which was already available. Secondly, image segmentation is a process of partitioning a digital image into multiple segments, each of which corresponds to a meaningful part of the image. In third step feature extraction is used, which means extracting features from images.In image processing feature extraction involves extracting visual features such as edges, corners, texture etc. In next step Classification is done based on selected features from above step. And evaluation is done.

Image pre-processing is advancement of image data that compress deformation and enhance some image features which are essential for further processing.
In this study we modify our pre-processing step as it takes input image of size 224 with batch size 64.And dividing the data-set in train data generator and test data generator with data augmentation (process of increasing data-set size by re-scaling, zooming and flipping. Figure 3 shows the process data acquisition and pre-processing.
Algorithm:
• scan the picture files (saved on drive)
• Decrypt the JPG content to RGB grids of pixels with channels
• Covert these into floating-point tensors for input to neural networks
• Formulate the pixel values (between 0, 255) to the [224,224,3] interval (as neural network this length gets efficacious).

3.2 Applications of Plant Disease Detection

Plant disease detection system have variety of applications in agriculture and environmental science fields. Some of the most necessary applications of plant disease detection are as follows:

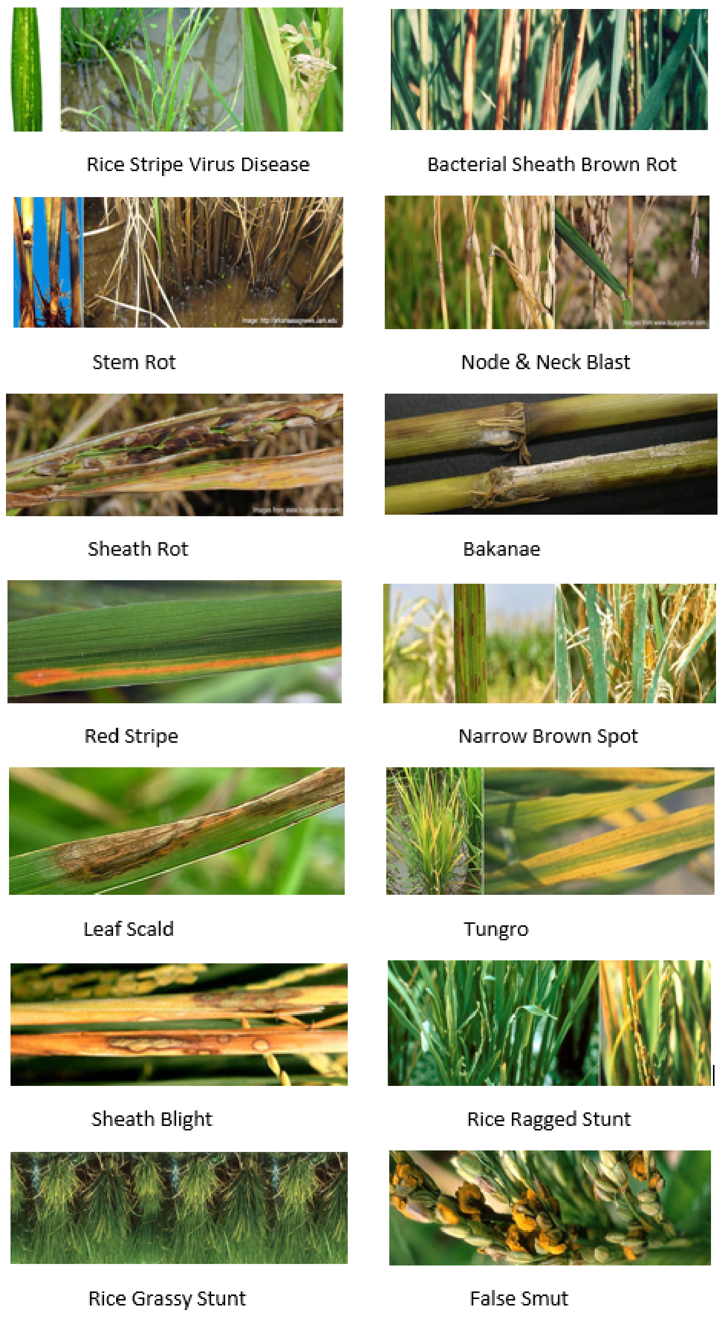


Figure 2: Different Types Of Rice Diseases

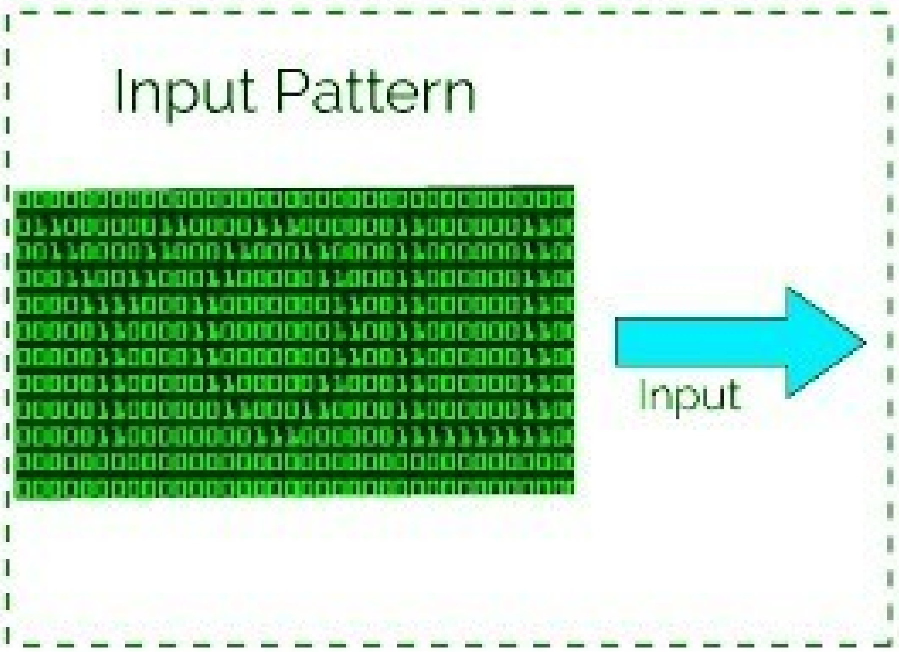


Figure 3: Data Acquisition & Pre-Processing

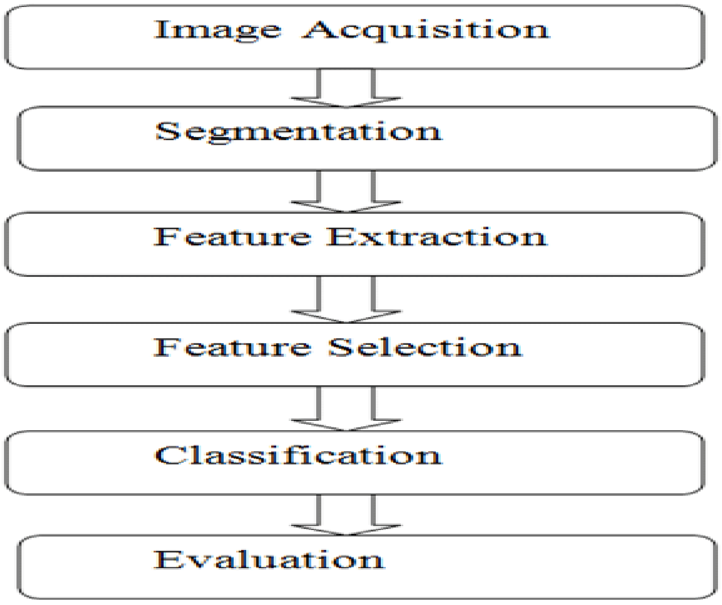


Figure 4: Steps Involves in DIP

1. Early Detection:- The early detection of plant diseases can prohibit significant damages in crop yields and lower the need for pesticides in fields. By using techniques such as imaging and machine learning, early signs of plant diseases can be identified before they circulate all around the crop.

2. Precision Agriculture:- Plant disease detection can be used to develop the use of inputs such as water, fertilizer, and pesticides. By identifying the exact location and strictness of plant diseases, farmers can apply the important treatments only where they are needed, reducing waste and maximizing production.

3. Disease Management:- Plant disease detection can be used to classify the specific bacterium causing plant diseases. This details can be used to develop selected treatments and management plans to lower the transmit of diseases.

4. Environmental Monitoring:- Plant disease detection can be used to monitor the health of natural plant populations. By tracking the transmission of plant diseases, scientists can better know the effect of environmental factors such as climate change and pollution on plants health.

5. Food Safety:- Plant disease detection can be used to classify and monitor plant bacterium that can cause food-borne sickness. By assuring the protection of crops, plant disease detection can help put a stop to food-borne sickness and ease the risk of disruption.

3.3 Objectives

The objective of this study is to focus on the rice leaf disease detection using deep learning CNN's model VGG-16 using flask.
CNN's model VGG-16 is used to examine the diseased and healthy leaves of rice plant.

3.4 Motivation

Rice is a staple food for millions of people worldwide, and its production is essential for food security and economic stability in many countries. However, rice production is often threatened by various diseases, including Brown Spot, Leaf Blast, and Neck Blast, which can cause significant crop losses if left undetected and untreated. Early detection and management of these diseases are crucial for maintaining healthy crop yields, reducing the risk of food insecurity and economic instability.
Motivated by the need for efficient and accurate disease detection tools in rice production, researchers have developed deep learning models for the early detection of rice leaf diseases. These models can provide an effective and scalable solution for disease management, helping farmers to minimize crop losses, optimize yields, and ensure food security.

4 Chapter 2 Literature Survey

4.1 Rice Plant Disease Detection Using Image Processing and Probabilistic Neural Network[1].

Publication Year:- 2022
Author:- İrfan Ökten Uğur Yüzgeç
Publishing Organisation:- International Congress of Electrical and Computer Engineering.
Method:- In this study, author irfan okten use image processing and probabilistic neural network for identifying diseased or healthy leaves of rice plant. Author divides their work in four steps-
1. Image Pre-processing:- In this step they use Median Filter method for preprocessing the image data-set.
2. Image Segmentation:- In this step author use OTSU Thresholding method for segmenting the images.
3. Feature Extraction:- For feature extraction author use Gray Level Cooccurrence Matrix (GLCM) method to extract the features from a rice image data-set.
4. Classification:- At last they use Probabilistic Neural Network for classifying the diseased and healthy leaves of rice plant with an accuracy of 76 percent.

4.2 Artificial Intelligence-Based Drone System for Multiclass Plant Disease Detection Using an Improved Efficient Convolutional Neural Network[2].

Publication Year:- 2022
Author:- Waleed Albattah, Ali Javed, Marriam Nawaz, Momina Masood and Saleh Albahli
Publishing Organisation:- Frontiers in Plant Science.
Method:- In this study a novel approach to detecting plant diseases using drones equipped with cameras and an artificial intelligence-based system. The authors propose an improved version of the EfficientNet architecture for image classification, which is optimized for detecting multiple plant diseases. They also describe the process of collecting and labeling a large data-set of plant images with different diseases.

The proposed system uses a drone to capture images of plants and transmits them to a central server for analysis. The images are then processed using the improved EfficientNet model, which is trained to classify them into one of several disease categories. The authors evaluate their approach on several data-sets and compare it to other state-of-the-art methods, demonstrating that their system achieves high accuracy and outperforms existing approaches.

Overall, the paper demonstrates the potential of using drones and artificial intelligence to improve the detection and monitoring of plant diseases, which could have important implications for agricultural productivity and food security.

4.3 Practical cucumber leaf disease recognition using improved Swin Transformer and small sample size[3].

Publication Year:- 2022
Author:- Fengyi Wang, Yuan Rao, Qing Luo, Xiu Jin, Zhaohui Jiang, Wu Zhang, Shaowen Li
Publishing Organisation:- Computers and electronics in agriculture v.199.
Method:- The research paper presents a new approach for recognizing leaf diseases in cucumber plants using a deep learning model called the Swin Transformer. The authors propose an improved version of the Swin Transformer model that is optimized for small sample size data-sets.

The authors collected a data-set of cucumber leaf images with different diseases, and use this data-set to train and test their model. The proposed model achieves high accuracy in recognizing three common cucumber leaf diseases, which are anthracnose, downy mildew, and powdery mildew.

The proposed approach has several advantages over existing methods. It requires fewer training samples, which reduces the cost and time needed to collect and label a large dataset. It also achieves high accuracy and robustness, making it suitable for practical applications.

4.4 Plant Disease Detection using Image Processing and Machine Learning Algorithm[4].

Publication Year:- 2023
Author:- Rajneni Deepika Sai, Yogeshwari, and Varsha
Publishing Organisation:- Journal of Xidian University.
Method:- The research paper proposes an approach to detect plant diseases using image processing and machine learning algorithms. The authors collected a data-set of images of diseased and healthy plants, and used image processing techniques to extract features from the images. They then applied machine learning algorithms, specifically support vector machines (SVM) and random forests (RF), to classify the images as either healthy or diseased.

The authors evaluated the performance of their approach on several plant disease data-sets and compared it with other state-of-the-art methods. The results demonstrate that their approach achieves high accuracy in detecting plant diseases, particularly for data-sets with relatively simple backgrounds and few variations in lighting and color.

The proposed approach has several advantages over traditional methods, such as visual inspection by experts. It is automated, fast, and objective, and can be used to detect diseases early, which is critical for effective disease management.

4.5 Plant Disease Detection Using Quantum Image Processing[5].

Publication Year:- 2022
Author:- Aishwarya K, Sreekumar N R
Publishing Organisation:- International Conference on Industry 4.0 Technology.
Method:- The research paper proposes an approach to detect plant diseases using quantum image processing (QIP). The authors suggest that QIP could provide advantages over classical image processing techniques, such as improved speed and efficiency in processing large data-sets of images.

The proposed approach involves encoding plant images as quantum states and performing quantum operations to extract features and classify the images as either healthy or diseased. The authors also suggest using quantum machine learning algorithms, specifically the quantum support vector machine (QSVM), to improve the accuracy and efficiency of disease detection.

The authors evaluate the performance of their approach on several plant disease data-sets and compare it with classical image processing techniques. The results demonstrate that their approach achieves high accuracy in detecting plant diseases and outperforms classical methods in terms of speed and efficiency.

The proposed approach has several advantages over classical methods, such as improved speed and accuracy, and the potential for scalability to process large data-sets. However, the approach is still in its early stages of development and requires further research to optimize the quantum algorithms and improve the accuracy of the disease detection.

4.6 Plant Disease Detection Using Deep Convolutional Neural Network[6].

Publication Year:- 2022
Author:- J. Arun Pandian, V. Dhilip Kumar, Oana Geman, Mihaela Hnatiuc, Muhammad Arif, K. Kanchanadevi
Publishing Organisation:- Journal of Applied Science.

Method:- The research paper proposes an approach to detect plant diseases using a deep learning model called a convolutional neural network (CNN). The authors collected a data-set of images of healthy and diseased plants, and used the CNN to classify the images as either healthy or diseased.

The authors evaluated the performance of their approach on several plant disease data-sets and compared it with other state-of-the-art methods. The results demonstrate that their approach achieves high accuracy in detecting plant diseases, particularly for data-sets with relatively simple backgrounds and few variations in lighting and color.

The proposed approach has several advantages over traditional methods, such as visual inspection by experts. It is automated, fast, and objective, and can be used to detect diseases early, which is critical for effective disease management.

4.7 Plant Disease Detection and Classification Method Based on the Optimized Lightweight YOLOv5 Model[7].

Publication Year:- 2022
Author:- Haiqing Wang, Shuqi Shang, Dongwei Wang, Xiaoning He, Kai Feng, Hao Zhu
Publishing Organisation:- Journal of Agriculture.
Method:- The research paper proposes an approach to detect and classify plant diseases using a deep learning model called the lightweight YOLOv5. The authors optimized the architecture of the model to reduce its computational complexity while maintaining its accuracy in detecting and classifying plant diseases.

The authors collected a data-set of images of healthy and diseased plants, and used the optimized lightweight YOLOv5 to classify the images into one of several disease categories. They evaluated the performance of their approach on several plant disease data-sets and compared it with other state-of-the-art methods. The results demonstrate that their approach achieves high accuracy in detecting and classifying plant diseases, particularly for data-sets with relatively simple backgrounds and few variations in lighting and color.

The proposed approach has several advantages over traditional methods, such as visual inspection by experts. It is automated, fast, and objective, and can be used to detect and classify diseases early, which is critical for effective disease management.

4.8 A Performance-Optimized Deep Learning-Based Plant Disease Detection Approach for Horticultural Crops of New Zealand[8].

Publication Year:- 2022
Author:- Muhammad Hammad Saleem, Johan Potgieter, Khalid Mahmood Arif
Publishing Organisation:- IEEE Access.
Method:- The research paper proposes an approach to detect plant diseases in horticultural crops of New Zealand using a deep learning model. The authors collected a data-set of images of healthy and diseased crops, including kiwifruit, apple, and pear, and used a deep learning model to classify the images into one of several disease categories.

The authors optimized the performance of the deep learning model by experimenting with different architectures and training parameters, and evaluated the performance of their approach on several disease data-sets. The results demonstrate that their approach achieves high accuracy in detecting plant diseases, particularly for kiwifruit and apple data-sets.

The proposed approach has several advantages over traditional methods, such as visual inspection by experts. It is automated, fast, and objective, and can be used to detect diseases early, which is critical for effective disease management. The approach has the potential to improve the efficiency and accuracy of plant disease monitoring and management in New Zealand's horticultural industry.

4.9 Different stages of disease detection in squash plant based on machine learning[9].

Publication Year:- 2022
Author:- R Ganesh Babu, Chellaswamy C
Publishing Organisation:- National Center for Biotechnology Information.
Method:- The research paper proposes an approach to detect different stages of disease in squash plants using machine learning algorithms. The authors collected a data-set of images of squash plants at different stages of disease progression, including healthy plants and plants with early and advanced stages of disease.

The authors used several machine learning algorithms, including support vector machine (SVM) and k-nearest neighbor (KNN), to classify the images into one of several disease categories. They evaluated the performance of their approach on several disease data-sets and compared it with other state-of-the-art methods. The results demonstrate that their approach achieves high accuracy in detecting different stages of disease in squash plants.

The proposed approach has several advantages over traditional methods, such as visual inspection by experts. It is automated, fast, and objective, and can be used to detect diseases early, which is critical for effective disease management.

4.10 A fast accurate fine-grain object detection model based on YOLOv4 deep neural network[10].

Publication Year:- 2022
Author:- Arunabha M. Roy, Rikhi Bose, Jayabrata Bhaduri
Publishing Organisation:- Neural Computing and Applications.
Method:- The research paper proposes an approach to detect fine-grained objects, such as small objects with intricate details, using a deep neural network called YOLOv4. The authors optimized the architecture of the YOLOv4 model to improve its accuracy and speed in detecting fine-grained objects.

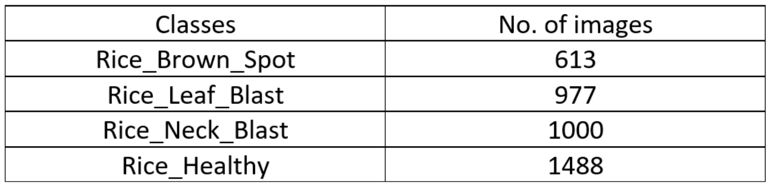
The authors collected a data-set of images containing fine-grained objects, such as insects and small animals, and used the optimized YOLOv4 model to detect and classify the objects. They evaluated the performance of their approach on several data-sets and compared it with other state-of-the-art methods. The results demonstrate that their approach achieves high accuracy and speed in detecting fine-grained objects.

The proposed approach has several advantages over traditional methods, such as manual inspection and counting by experts. It is automated, fast, and objective, and can be used to detect fine-grained objects in a variety of settings, such as in agriculture, biology, and ecology.

5 Chapter 3 Methodology

5.1 Data-set

The data-set used in our research work is downloaded from Kaggle Web-Page[15]. This data-set consists of four classes, three are diseased classes and one for healthy leaves. This data-set is already arranged in a well manner, thanks to the data-set up-loader. This data-set consists of total 4500 images in these four classes.



Classes	No. of images
Rice_Brown_Spot	613
Rice_Leaf_Blast	977
Rice_Neck_Blast	1000
Rice_Healthy	1488

Figure 5: Rice Leaf Data-set

In figure 5 & 6, we can see the images from all four classes of rice leaf disease data-set. First one is from brown spot, second is from healthy class, third one is from leaf blast and at last fourth one is from neck blast class. In our study we classify these diseased and healthy images using our trained model.

5.2 Tools & Technologies

The tools and technologies we used in this research of Rice Disease Detection system are explained below.

5.2.1 3.2.1 About Python

Python is high-level interpreted programming language that is used for a wide variety of applications, including web development, data science, artificial intelligence, and many more. It was first released in 1991 by Guido Van Rossum and from that it becomes most popular programming language in the world.

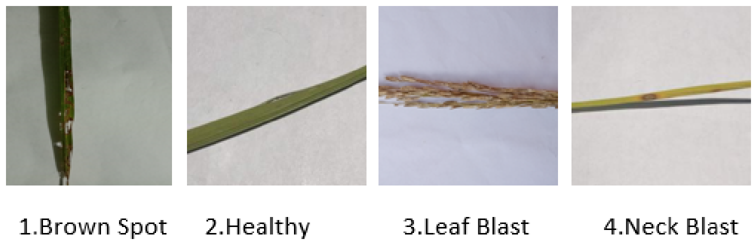


Figure 6: Four Classes of Data-set

One key feature of python is its simplicity and ease of use. Python is an interpreted language, which means it does not need to be compiled before execution. This allows for faster development and testing of code, which is helpful in rapid iterative development environment.
Another strength of python is, its extensive standard library, which provides developers with a wide range of pre-built modules and functions which are directly used in code. Additionally, there are numerous third party libraries available that extend python's capabilities, such as NumPy for scientific computation, pandas for data manipulation and TensorFlow for machine learning.
Python is also highly portable with all versions of operating systems, like- Windows, Macos, Linux etc.

5.2.2 NumPy & TensorFlow

NumPy (Numerical Python) is most useful and powerful python library which is used in every aspect of python code. It is popular for scientific computation. It was created in 2005 and has since become a fundamental tool for scientific computing and data analysis in python.
NumPy also includes a wide range of mathematical functions, including arithmetic, logarithmic, exponent and trigonometric functions. NumPy also provides support to linear algebra operations such as matrix multiplication, matrix inversion and matrix decomposition.

TensorFlow is powerful and flexible machine learning library that provide a wide range of tools and API's for building and training machine learning models on Google Colaboratory on any other GPU. Its ability to perform distributed training and its rich ecosystem of tools and libraries makes it a popular choice for developers and researchers in the machine learning community.

5.2.3 Keras

Keras is an open-source deep learning library written in Python. It was developed by Francois Chollet and first released in 2015. Keras provides a user-friendly interface to build and train deep learning models on top of TensorFlow, Theano, or Microsoft Cognitive Toolkit (CNTK).

Some of the features and benefits of Keras include:
1. User-friendly API: Keras provides a simple and intuitive interface that allows users to quickly build and train deep learning models.
2. Modular architecture: Keras allows users to easily build complex models by connecting pre-built building blocks such as layers and activation functions.
3. Supports multiple back-ends: Keras can run on top of TensorFlow, Theano, or CNTK, giving users the flexibility to choose the back-end that best fits their needs.
4. Comprehensive documentation: Keras has extensive documentation that provides examples, tutorials, and API references.
5. Wide range of applications: Keras can be used for various deep learning tasks, including classification, regression, image processing, and natural language processing.
6. Easy to extend: Keras is built on top of Python, which makes it easy to extend with custom layers, loss functions, and metrics.
7. Support for GPU acceleration: Keras supports running models on GPUs, which can significantly speed up training times.

Overall, Keras is a powerful and popular deep learning library that makes it easy for researchers and developers to build and train deep learning models.

5.2.4 Flask

Flask is a mini Web Framework dictated in Python.It is categorize as a mini framework because it doesn't needs appropriate tools or libraries. It has no database consideration layer from validation or any other elements where preceding third party libraries supply common functions.Flask is used for developing web applications using python.

Advantages

1.There is a built in development server and a fast debugger provided.
2.Scalable
3.Easy to negotiate
4.Lightweight
5.Not a lot of tools
6.Flexible

5.2.5 Google Colaboratory

Google Colaboratory also known as Colab, is a cloud based platform developed by google for training machine learning models and data analysis. It allows user to write, run and share code in a collaborative environment, without the need of any special hardware and software.
Colab is built on top of jupyter notebooks. Most useful feature of colab is its ability to run code on Google cloud infrastructure, which provides access to powerful computing resources, including GPUs and TPUs. With the help of this we can train complex machine learning model with large data-set without need of any expensive hardware. It access data-set directly from website or we can save our data-set in google drive and provide link of it in the code.
Google Colab is a excellent platform for Deep Learning admirer, and it can also be used to test basic machine learning models, achieve experience and establish an intuition about deep learning condition such as hyperparameter tuning, pre-processing data, model-complexity , over-fitting and more.
We train our model on **Google Colab**, because Google Colab provide us the facility of Python3 Google Compute Engine Back-end (GPU) and TPU as the run-time environment.As we know we can't train our model on our system CPU because it takes a day or sometime even weeks to train and for management of time we can't do this.And on google colab we will able to train our model within 2 hours.

5.3 VGG-16 Model

VGG-16 is a convolution neural network model that was trained on the image net data-set. It was developed by Visual Geometry Group(VGG) at the university of Oxford and introduced in 2014. The 16 in VGG-16 refers to the numeral of weight layers in the model.VGG-16 is known for its deep architecture, with a total of 16 layers and for its use of small convolutional filter (3*3) that are stacked together to form a deep network. It is widely used for image classification and object detection tasks.

VGG is although a relatively large-scale network with a total of 138 million parameters- it's massive even by today's recognized. Still the integrity of the vgg16 architecture is its main interest.The vgg architecture associates the most needy convolutional neural network features.
The VGG-16 model achieved state-of-the-art results on the Image-Net Large Scale Visual Recognition Challenge (ILSVRC) in 2014, with a top-5 error rate of 7.3% . Since then, it has become a popular model for transfer learning and has been used as a starting point for many other CNN architectures.

In the VGG-16 model, feature extraction (as shown in figure 7) is performed using convolutional layers. The input to the model is a 224x224 RGB image. The first few layers of the model are convolutional layers with small filter sizes

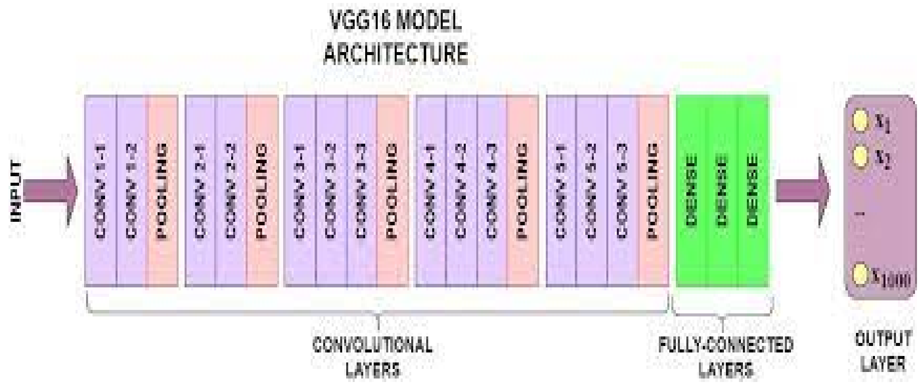


Figure 7: Architecture of VGG-16 Model

(3x3) and a small stride (1), followed by max-pooling layers with a stride of 2. These layers are designed to extract low-level features such as edges, corners, and blobs.

As we go deeper into the network, the convolutional layers become more complex and have larger filter sizes. These layers are able to extract higher-level features such as shapes and textures. The final few convolutional layers in the VGG-16 model are fully connected layers that take the output of the previous convolutional layers and produce a feature vector that can be used for classification.

During training, the weights of the convolutional layers are learned using back propagation and gradient descent to minimize a classification loss function. Once the model is trained, the convolutional layers can be used as a feature extractor for other computer vision tasks such as object detection and semantic segmentation. By removing the fully connected layers and using the output of the convolutional layers as features, we can extract features from images that can be used as inputs to other machine learning models.

The configuration of VGG-16 model is given in figure 9.

A vgg network comprise of limited convolution filters. VGG-16 has three fully connected layers and thirteen convolution layers.

Here is a rapid outline of the vgg architecture:-
Input- VGGNet takes a 224×224 image input.In the image-net clash, the model originator's kept the input image size unvarying by cropping 224×224 section from the center of image.
Convolution Layers-the convolutional feature of vgg use the slightest desirable approachable field of 3×3 . VGG also uses 1×1 convolution filter as the

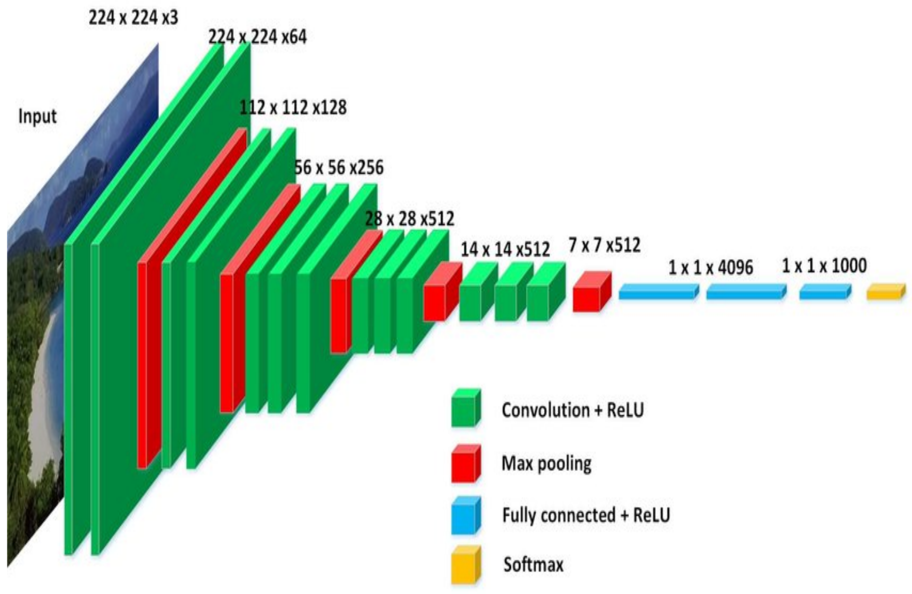


Figure 8: Architecture of Feature Extraction by VGG-16

input straight transformation.
reLu Activation- next is the rectifier Linear unit (reLu) activation function element, Alex-net big modernization for lowering training time.
Hidden Layers- all the vgg network's hidden layers are reLu instead of limited response normalization like alexnet.
Pooling Layers- A pooling layer chases several convolutional layers - this support lower the dimensionality and the number of parameters of the feature maps constructed by each convolutional stride.
Fully Connected Layers- The VGG-16 (Visual Geometry Group) model has three fully connected layers at the end of the network, just before the final classification layer. These fully connected layers are also referred to as the "dense" layers.
The first fully connected layer has 4,096 neurons, the second has 4,096 neurons, and the third has 1,000 neurons. The first two fully connected layers use the Re-LU activation function, which helps to introduce non-linearity into the network and improve its ability to learn complex patterns in the data. The third fully connected layer uses the Soft-max activation function, which produces a probability distribution over the 1,000 classes in the Image-Net data-set.

In brief:-

* All manuscript of VGG have block structures.

* A VGG block comprises multiple convolutional layers followed by a max-pooling layer. The authors used a uniform kernel size of 3x3 for all the convolutional layers and applied a padding size of 1 to maintain the output size after each convolution. Additionally, they employed a max-pooling layer of size 2x2 with a stride of 2 to reduce the resolution by half after each block.

* Every VGG model is composed of one fully connected output layer and two fully connected hidden layers.

To begin our study, we collected a data-set of rice plant leaves with both healthy and diseased conditions from Kaggle [15].

We used this data-set to train a neural network, which was subsequently evaluated on the same data-set. The performance of the network was assessed by measuring its accuracy in classifying healthy and diseased rice plant leaves. Our experiments yielded a testing accuracy of 90%, indicating the effectiveness of the network in this task.

5.4 Proposed Model

Process of our proposed model is shown in figure 10.

In our study we use built in VGG-16 model with some modification. Here we can see the detailed explanation of our **code** step by step:-

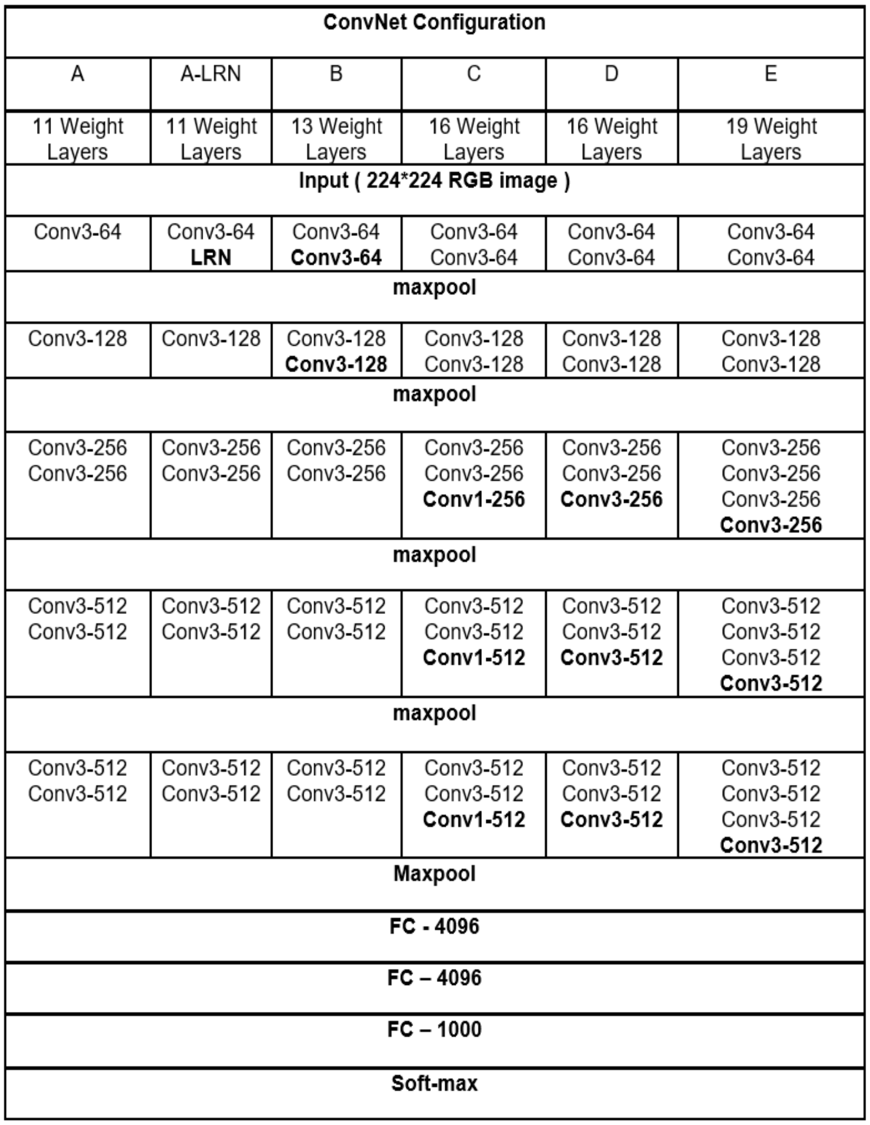
Step 1.
```
import tensorflow as tf
import os
import numpy as np
```

→ In first step we have to import important libraries.

Step 2.
```
base_dir=r”/content/drive/MyDrive/Thesis Dataset”
```

→ In second step we have to import our drive in google colaboratory for accessing our saved data-set by using above command.

Step 3.
```
IMAGE_SIZE=224
BATCH_SIZE=32
train_datagen=tf.keras.preprocessing.image.ImageDataGenerator(
```



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 Weight Layers	11 Weight Layers	13 Weight Layers	16 Weight Layers	16 Weight Layers	19 Weight Layers
Input (224*224 RGB image)					
Conv3-64	Conv3-64 **LRN**	Conv3-64 **Conv3-64**	Conv3-64 Conv3-64	Conv3-64 Conv3-64	Conv3-64 Conv3-64
maxpool					
Conv3-128	Conv3-128	Conv3-128 **Conv3-128**	Conv3-128 Conv3-128	Conv3-128 Conv3-128	Conv3-128 Conv3-128
maxpool					
Conv3-256 Conv3-256	Conv3-256 Conv3-256	Conv3-256 Conv3-256	Conv3-256 Conv3-256 **Conv1-256**	Conv3-256 Conv3-256 **Conv3-256**	Conv3-256 Conv3-256 Conv3-256 **Conv3-256**
maxpool					
Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512 **Conv1-512**	Conv3-512 Conv3-512 **Conv3-512**	Conv3-512 Conv3-512 Conv3-512 **Conv3-512**
maxpool					
Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512 **Conv1-512**	Conv3-512 Conv3-512 **Conv3-512**	Conv3-512 Conv3-512 Conv3-512 **Conv3-512**
Maxpool					
FC - 4096					
FC – 4096					
FC – 1000					
Soft-max					

Figure 9: VGG-16 Configuration

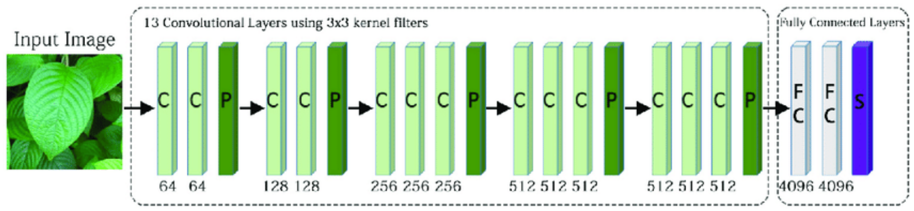

Input Image
13 Convolutional Layers using 3x3 kernel filters
Fully Connected Layers
C C P C C P C C C P C C C P C C C P F C F C S
64 64 128 128 256 256 256 512 512 512 512 512 512 4096 4096

Figure 10: Process of Proposed Model

```
    rescale=1./255,

    zoom_range=0.2,

    horizontal_flip=True)
validation_datagen=tf.keras.preprocessing.image.ImageDataGenerator(
rescale=1./255,

    zoom_range=0.2,

    horizontal_flip=True,

    validation_split=0.1)
```

→ This step is also known as pre-processing step, as the code snippet is setting up two image data generators, one for training data and another for validation data.
IMAGE_SIZE=224 defines the target size of the input images, which will be resized to 224 x 224 pixels by the data generators.
BATCH_SIZE=32 specifies the number of images that will be processed in each training iteration.
The train_datagen object is configured to apply data augmentation techniques such as re-scaling the pixel values of the input images to be between 0 and 1, randomly zooming into the images, and randomly flipping the images horizontally. These techniques can help to prevent overfitting and improve the generalization performance of the model.
The validation_datagen object is configured with the same data augmentation techniques as the train_datagen, with the additional validation_split parameter set to 0.1, which indicates that 10% of the training data will be used for validation. This generator will be used to create a validation data-set that can be used to evaluate the performance of the model during training.

Step 4.

```
train_generator=train_datagen.flow_from_directory(
    base_dir,

    target_size=(IMAGE_SIZE,IMAGE_SIZE),

    batch_size=BATCH_SIZE,

    subset='training'
)
validation_generator=validation_datagen.flow_from_directory(

    base_dir,

    target_size=(IMAGE_SIZE,IMAGE_SIZE),

    batch_size=BATCH_SIZE,

    subset='validation'
)
```

→ This code snippet is using Keras' ImageDataGenerator to create data generators for training and validation sets.
The train_datagen and validation_datagen objects are instances of the ImageDataGenerator class, which define the transformations to be applied to the images. These transformations can include resizing, cropping, and data augmentation such as flipping or rotating the images.
The flow_from_directory method is called on these objects to specify the directory where the image files are located, the target size of the images, and the batch size. The subset argument is used to indicate whether the generator is for the training or validation set.
In summary, these data generators will be used to load and pre-process images on-the-fly during training and validation of a deep learning model. The generators will take batches of images from the directories specified and apply the specified transformations.

Step 5.
```
from tensorflow.keras.layers import Input,Flatten,Dense
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.models import Sequential
from glob import glob
```

→ This code snippet is importing the necessary modules and classes from the TensorFlow Keras library to create a deep learning model using the VGG16 architecture.
The Input, Flatten, and Dense classes are used to define the input layer, flatten

layer, and output layer of the model, respectively. The Model class is used to create a functional API model, which can handle more complex architectures compared to the Sequential model. The VGG16 class is imported from the Keras applications module, which provides pre-trained deep learning models for various tasks.
The glob function is imported from the Python glob module, which is used to retrieve a list of file paths that match a specified pattern. In this case, it is used to retrieve a list of file paths for the image classes in the data-set, which will be used to set the number of output nodes in the model's output layer.
Overall, this code snippet sets the stage for defining a VGG16-based deep learning model that can be trained on image data. The Sequential or functional API can be used to add layers to the model and the pre-trained weights from the VGG16 model can be optionally loaded for transfer learning.

Step 6.

```
IMAGE_SIZE=[224,224]
vgg=VGG16(input_shape=IMAGE_SIZE+[3],weights='imagenet',include_top=
False)
vgg.output
```

→ In this code snippet, the VGG16 model from Keras applications is initialized with input_shape parameter set to [224, 224, 3], which specifies the expected size of the input images to the model. The weights parameter is set to 'imagenet', which indicates that the pre-trained weights for the model on the ImageNet data-set should be used. Finally, include_top parameter is set to False to exclude the fully connected layers at the top of the model.
The resulting vgg object is an instance of the Model class that represents the VGG16 model with the pre-trained weights loaded.
The vgg.output expression simply returns the output tensor of the last layer of the VGG16 model. This output tensor has shape (None, 7, 7, 512) which corresponds to a feature map of size 7x7 with 512 channels for each input image. This output can be used as input to additional layers in a custom neural network architecture built on top of the VGG16 base.

Step 7.

```
for layer in vgg.layers:
layer.trainable=False
```

→ In this code snippet, each layer in the pre-trained VGG16 model is set to be untrainable by setting the trainable attribute to False. This is done by iterating over the layers attribute of the vgg object, which is a list of all the layers in the model. By setting trainable to False, the weights of these layers will not be updated during training of the model.
Setting the layers of the pre-trained model to be untrainable is often done when using transfer learning, where the pre-trained model is used as a base for a new model that is trained on a different data-set or for a different task. In this case,

the weights of the pre-trained layers are kept fixed, and only the weights of the new layers added on top of the pre-trained layers are updated during training. This approach can speed up training and improve performance, especially when the size of the new data-set is small.

Step 8.
folders=glob("/content/drive/MyDrive/Thesis Dataset/*")
print(len(folders))

→ In this step we print the length of our data-set folder.

Step 9.
x=Flatten()(vgg.output)
prediction=Dense(len(folders),activation='softmax')(x)
model=Model(inputs=vgg.input,outputs=prediction)
model.summary()

→ In this code snippet, a new neural network model is built on top of the pre-trained VGG16 model.
First, the output tensor of the VGG16 model is passed through a Flatten layer, which reshapes the tensor from shape (None, 7, 7, 512) to shape (None, 25088). This flattened tensor is then passed through a Dense layer with len(folders) output nodes, where folders is a list of the directories in the dataset. The softmax activation function is used to ensure that the output values are probabilities that sum to 1.
The resulting prediction tensor is used to create a new Model object with the same input layer as the VGG16 model and the prediction tensor as the output layer. This creates a new model architecture with the pre-trained VGG16 layers frozen and the new dense layer trainable.
The model.summary() function call is used to display a summary of the architecture of the new model. The summary shows the layers in the model, their output shapes, and the number of parameters in each layer. This can be useful for debugging the model and ensuring that it has been built correctly.
The summary of model is shown in figure 12.
Step 10.
model.compile(loss='binary_crossentropy',
optimizer='adam',
metrics=['accuracy'])

→ In this code snippet, the compile method is called on the model object to configure its learning process. The loss parameter is set to 'binary_cross-entropy', which is a common loss function used for binary classification problems. The optimizer parameter is set to 'adam', which is a popular optimization algorithm that adjusts the weights of the neural network during training to minimize the loss function. The metrics parameter is set to ['accuracy'], which specifies that the model's performance during training should be evaluated based on its accu-

racy in predicting the correct class label for each input image.
Once the model is compiled, it is ready to be trained on the data-set using the fit method, which takes as input the training and validation data, the number of epochs to train for, and other optional parameters. During training, the model will update its weights based on the back-propagation of errors through the network, using the optimizer to adjust the weights in the direction that minimizes the loss function.

Step 11.
epoch=50 history=model.fit(train_generator,
steps_per_epoch=len(train_generator),
epochs=epoch,
validation_data=validation_generator,
validation_steps=len(validation_generator)
)

→ In this code snippet, the fit method is called on the model object to train the model on the training data-set. The train_generator object is used as input to the fit method to generate batches of training data, while the validation_generator object is used to generate batches of validation data.
The steps_per_epoch parameter is set to the number of batches in the training data-set, which is calculated as the length of the train_generator object. The validation_steps parameter is similarly set to the number of batches in the validation data-set, which is calculated as the length of the validation_generator object.
The epochs parameter is set to 50, which specifies the number of times the model should iterate over the entire training data-set during training.
The history variable is assigned the result of the fit method, which is an object containing information about the training process, such as the loss and accuracy of the model on the training and validation data-sets at each epoch. This information can be used to evaluate the performance of the model and make decisions about how to modify it to improve its performance.

Step 12.
import matplotlib.pyplot as plt

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)

plt.plot(epochs, loss, 'bo', label='T raining loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

```
plt.figure()

plt.plot(epochs, acc, 'bo', label='T raining acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.show()
```

→ In this code snippet, the history object returned by the fit method is used to plot the training and validation loss and accuracy for each epoch of training. The accuracy, val_accuracy, loss, and val_loss properties of the history object contain lists of the model's accuracy and loss on the training and validation datasets at each epoch. The range function is used to generate a list of integers from 1 to the number of epochs.
The plot function from the matplotlib.pyplot library is used to create two subplots. The first subplot shows the training and validation loss as a function of the epoch number, while the second subplot shows the training and validation accuracy. The legend function is used to label each line in the plots.
Finally, the show function is called to display the plots.
By examining the plots, it is possible to evaluate the performance of the model during training and determine whether it is overfitting or under-fitting the data. If the validation loss and accuracy start to diverge from the training loss and accuracy, it may indicate that the model is overfitting the training data and is not generalizing well to new data. On the other hand, if the validation loss and accuracy are not improving significantly over time, it may indicate that the model is under-fitting the data and may need to be made more complex.

Step 13.

```
!pip install pyyaml h5py
```

→ Required to save models in HDF5 format.

Step 14.

```
import os
import tensorflow as tf
from tensorflow import keras
print(tf.version.VERSION)
```

→ This code snippet uses TensorFlow to print the version number of the installed TensorFlow package.
The import statements at the top of the code bring in the required modules and functions from the TensorFlow and Keras libraries.
The print statement at the bottom of the code uses the VERSION attribute of the tensorflow.version module to print the version number of the installed TensorFlow package.

This information can be useful when debugging or troubleshooting issues with a TensorFlow program, as different versions of TensorFlow may have different behavior or require different code to function correctly.

Step 15.

```
model_json = model.to_json()
with open("vgg-16_model.json", "w") as json_file:
json_file.write(model_json)
tf.keras.models.save_model(model,"vgg-16_model.h5")
```

→ In this code snippet, the trained model is saved in two different formats: as a JSON file and as an HDF5 file.
The to_json method of the model object is used to convert the model architecture to a JSON string, which is then written to a file named "vgg-16_model.json" using a with statement and the open function.
The save_model function from the tf.keras.models module is used to save the model weights and other information to an HDF5 file named "vgg-16_model.h5". The model object is passed as the first argument to this function, and the path to the output file is passed as the second argument.
These two formats are commonly used for saving models in TensorFlow, and can be loaded back into a Model object using the model_from_json and load_model functions, respectively. This allows the saved model to be used for future inference or fine-tuning, without having to retrain the model from scratch.

The present study comprises of four steps in order to find out the health status of the rice plants.

→ In the 1st step, taking rice images as input and Data Augmentation is carried out to increase the size of data-size taken.

→ In the second part, image pre-processing is carried out to process all the input images to prepare pictures data for model input.For instance, convolution neural network fully connected layers demanded that all the images be in arrays of the same size. Additionally, model pre-processing may shorten model training time and speed up model inference.

→ In third step, Neural Network model VGG-16 was applied for classification of leaves either it is healthy or diseased plant leaf.

→ In fourth step, we create a flask web app for making prediction using VGG-16 model.
In this study we make some modification in inbuilt vgg-16 model as shown in above code explanation.
In this study we train our model for approx 4500 image data-set downloading

from Kaggle[15].
We have four classes in given data-set as bacterial leaf blight, brown spot, leaf smut and healthy leaf's.In training phase we have 4000 images belonging to four classes and in validation phase we have 500 images which are also belongs to four classes.
After compiling the model we have to train the model.
After training we get **90%** Accuracy, which is far better then over base paper[1].

The flowchart for prospective system is as revealed below in figure 11.

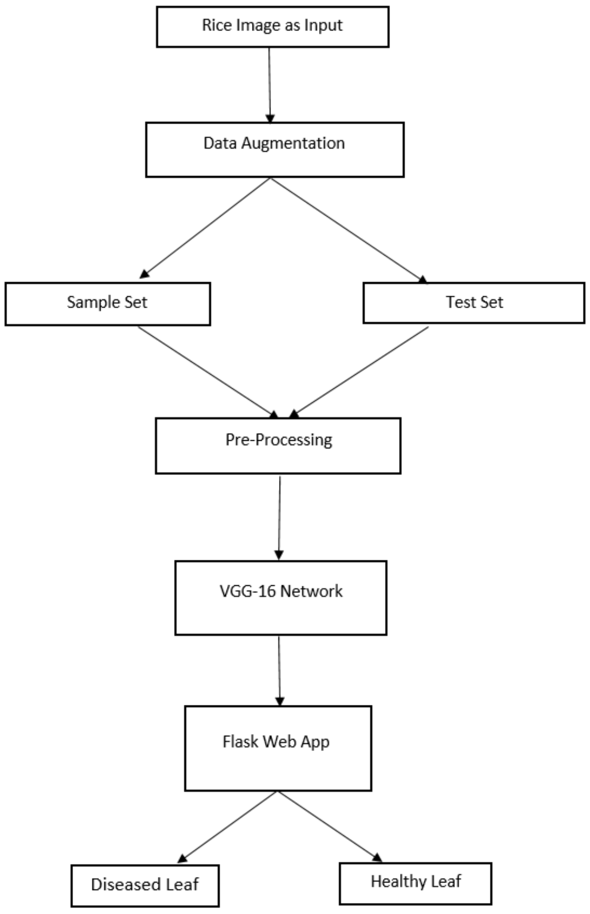
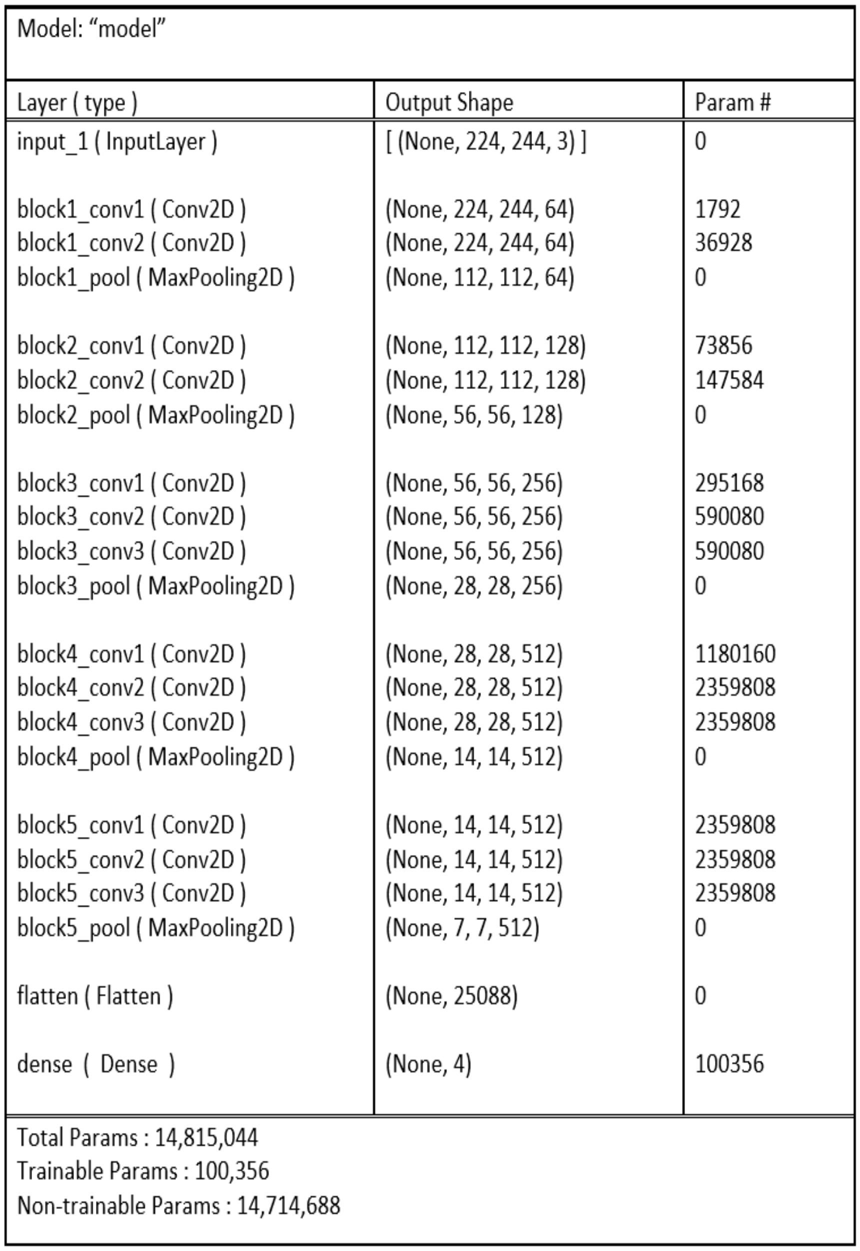


Figure 11: Flowchart For Proposed System

5.5 Computations in VGG-16

The method involved in determining the output size from each convolution layer is written as -
[(N-f)/S]+1
Where, N = input size (224), f = Kernel size (3), S = Strides
The model summary is as follows:
The summary of a model provides a compact overview of its architecture, including the number of layers, the size of the inputs and outputs, the number of parameters, and the computational complexity. The VGG-16 model is a deep convolutional neural network that was introduced in 2014 for image classification tasks.
The first two layers are Input and the VGG16 model from Keras library, respectively. The input shape is (224, 224, 3) and the output is (7, 7, 512) tensor. After freezing all the layers, a Flatten layer is added to convert the 3D feature maps to 1D feature vectors, which is then followed by a Dense layer with a soft-max activation function. The Dense layer has 10 neurons because there are 10 classes to predict.
As we can see there are 14 million plus parameters in VGG-16 for training, out of which we only use 1 lac plus parameters for our model training with given data-set.And remaining parameters are still available for training.



Model: “model”

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 244, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 244, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 244, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 4)	100356

Total Params : 14,815,044
Trainable Params : 100,356
Non-trainable Params : 14,714,688

Figure 12: Model Summary

6 Chapter 4 Discussion & Result

6.1 Input GUI

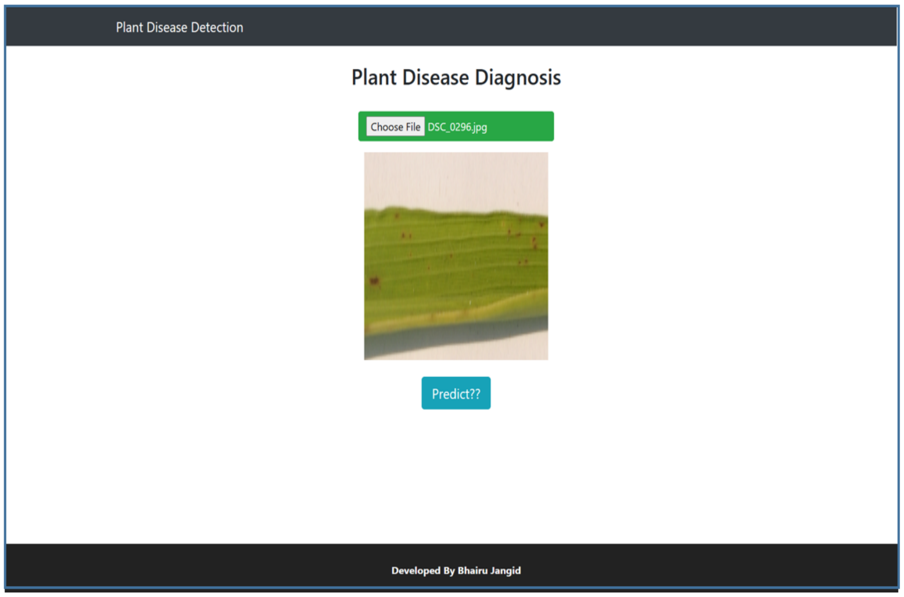


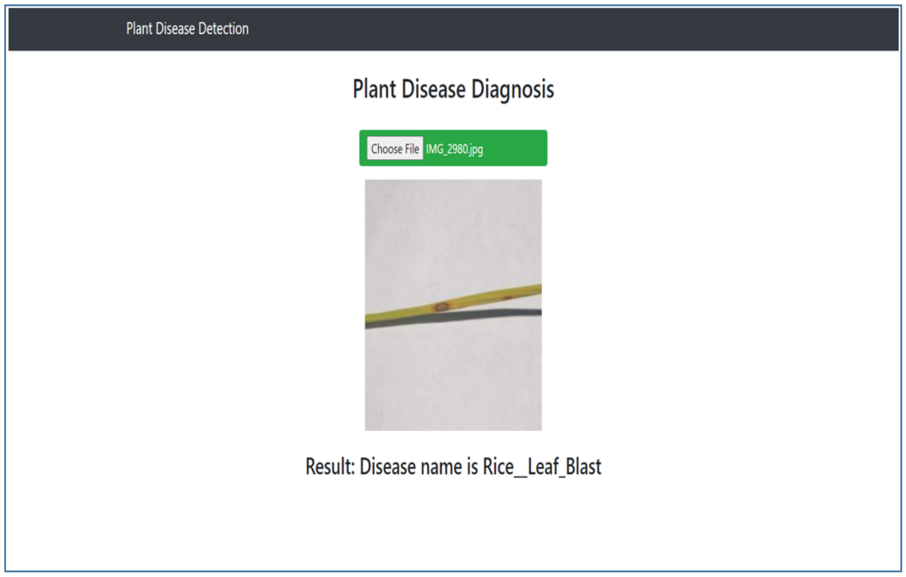
Figure 13: The intrusion of Plant Disease Detection GUI

Figure 13 shows the illustration of input graphical user interface.
The Input GUI is usually designed to be user-friendly and intuitive, so that users can easily understand how to provide input and interact with the system. It may also include features like validation to ensure that the input provided by the user is correct and appropriate.
Overall, the Input GUI plays an important role in facilitating communication between users and software systems, making it easier for users to perform various tasks and accomplish their goals within the software application.

In this interference firstly, we load an image from disk (system) and after loading we have to click on Predict?? button to running the model and predicting about diseases present on rice leaf.

6.2 Output GUI

Figure 14 shows the illustration of output GUI.



Result: Disease name is Rice__Leaf_Blast

Figure 14: Output GUI of Rice Disease Detection System

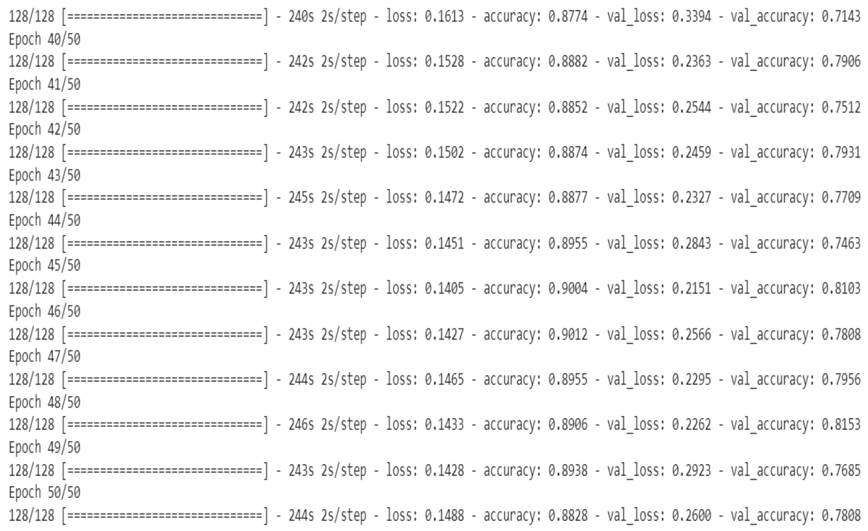
The Output GUI is usually designed to be visually appealing and easy to understand, so that users can quickly and easily interpret the information being presented. It may also include features like interactive elements or animations to help users better understand the output.

Overall, the Output GUI plays an important role in conveying information to the user, allowing them to make informed decisions and take appropriate actions based on the output provided by the software application.
In output GUI the result is predicted and shown below the input image.

4.2 Result

As we know, the result is typically the output produced by the model when it is given a particular input. For example, in image classification, the result may be the predicted class label of the input image. This result can be evaluated by comparing it to the ground truth label of the image to determine the accuracy of the model. And this is what we evaluate in this study is shown in figure 14.

When we train our inbuilt vgg-16 model with some modification then the accuracy of trained model on 50 epochs is as shown in figure 14. Figure 14 shows the training and validation accuracy of trained model with training loss and validation loss.
In trained model the training and validation accuracy is increased epoch per epoch as well as the training and validation loss is decreased epoch per epoch.



```
128/128 [==============================] - 240s 2s/step - loss: 0.1613 - accuracy: 0.8774 - val_loss: 0.3394 - val_accuracy: 0.7143
Epoch 40/50
128/128 [==============================] - 242s 2s/step - loss: 0.1528 - accuracy: 0.8882 - val_loss: 0.2363 - val_accuracy: 0.7906
Epoch 41/50
128/128 [==============================] - 242s 2s/step - loss: 0.1522 - accuracy: 0.8852 - val_loss: 0.2544 - val_accuracy: 0.7512
Epoch 42/50
128/128 [==============================] - 243s 2s/step - loss: 0.1502 - accuracy: 0.8874 - val_loss: 0.2459 - val_accuracy: 0.7931
Epoch 43/50
128/128 [==============================] - 245s 2s/step - loss: 0.1472 - accuracy: 0.8877 - val_loss: 0.2327 - val_accuracy: 0.7709
Epoch 44/50
128/128 [==============================] - 243s 2s/step - loss: 0.1451 - accuracy: 0.8955 - val_loss: 0.2843 - val_accuracy: 0.7463
Epoch 45/50
128/128 [==============================] - 243s 2s/step - loss: 0.1405 - accuracy: 0.9004 - val_loss: 0.2151 - val_accuracy: 0.8103
Epoch 46/50
128/128 [==============================] - 243s 2s/step - loss: 0.1427 - accuracy: 0.9012 - val_loss: 0.2566 - val_accuracy: 0.7808
Epoch 47/50
128/128 [==============================] - 244s 2s/step - loss: 0.1465 - accuracy: 0.8955 - val_loss: 0.2295 - val_accuracy: 0.7956
Epoch 48/50
128/128 [==============================] - 246s 2s/step - loss: 0.1433 - accuracy: 0.8906 - val_loss: 0.2262 - val_accuracy: 0.8153
Epoch 49/50
128/128 [==============================] - 243s 2s/step - loss: 0.1428 - accuracy: 0.8938 - val_loss: 0.2923 - val_accuracy: 0.7685
Epoch 50/50
128/128 [==============================] - 244s 2s/step - loss: 0.1488 - accuracy: 0.8828 - val_loss: 0.2600 - val_accuracy: 0.7808
```

Figure 15: Accuracy Result of Trained Network

6.3 Training and Validation Curve

The training and validation curves are important tools to assess the performance of machine learning models during training. They help in monitoring the progress of the model, detecting overfitting, and tuning hyper-parameters.

In the case of the VGG-16 model, the training and validation curves are plotted using the training history obtained from the model.fit() method. The training curve shows how well the model performs on the training data across the epochs, while the validation curve shows its performance on the validation data.

The history object returned by the model.fit() method contains the training and validation loss and accuracy for each epoch, which can be plotted against the number of epochs using Matplotlib to create the curves.

The training and validation loss curves illustrate how well the model is fitting the training and validation data over the epochs. Ideally, the training loss should decrease with each epoch, while the validation loss should also decrease but not as rapidly. If the validation loss starts to increase while the training loss is still decreasing, it is an indication of overfitting, and the model should be adjusted to prevent it.

Similarly, the training and validation accuracy curves show how well the model is predicting the training and validation data over the epochs. Both the training and validation accuracy should ideally increase with each epoch, but the validation accuracy may plateau or even decrease at some point. If the training accuracy continues to increase while the validation accuracy starts to decrease, it is an indication of overfitting, and the model should be adjusted to prevent it.

The training and validation accuracy and loss is shown in fig. 10.

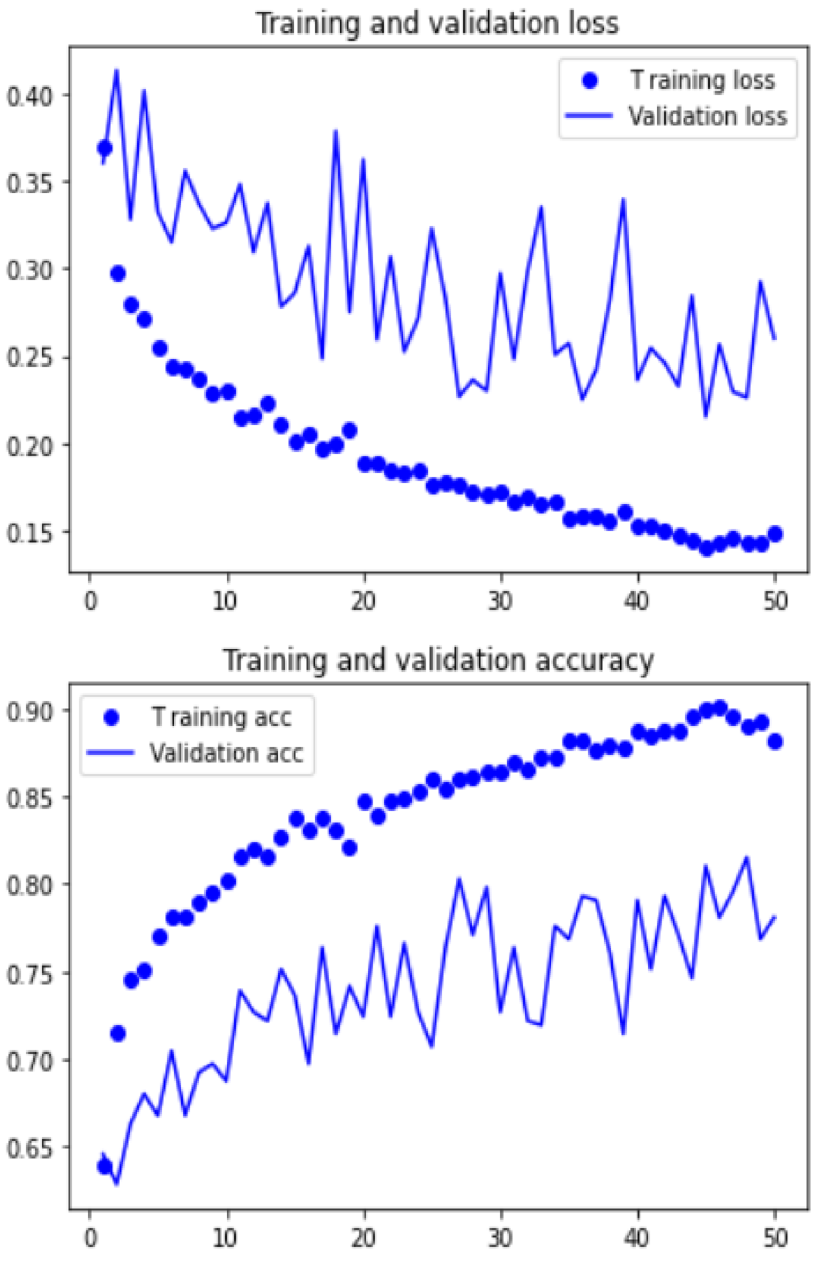


Figure 16: Training and Validation Curve

7 Conclusion

Inside the outlook of our study, the disease of rice plants was exposed using VGG-16 model, which is one of the Convolution Neural Network. For the identification of diseases, leaves pictures of 4500 rice plants were collected and VGG-16 model was trained with these pictures.Here a new method of using deep learning was examined in order to spontaneous classify and discover plant disease from leaves pictures. An accuracy rate of 90% was achieved when the rice plants pictures that it had never watched were provide to the VGG-16. Additionally, a GUI has been improved to accomplish all these actions. When the detection of disease in the rice plant is collated with other neural network models, it has been noticed that the training duration is much smaller.
In this study, the improved model was able to detect leaves between healthy and diseased.

8 Future Scope

There is a lot of potential for future advancements in rice disease detection systems. Some possible directions for future research and development in this area include:-

• Improved accuracy: One of the key challenges in rice disease detection is achieving high accuracy in disease identification. In future study, we can focus on developing more accurate algorithms and models that can detect a wider range of diseases and accurately distinguish between different types of diseases.

• Integration with other technologies: Rice disease detection systems can be integrated with other technologies such as drones, sensors, and machine learning algorithms to improve accuracy and reduce the need for manual labor.

• Cost-effective solutions: Rice farmers in developing countries often cannot afford expensive disease detection systems. So in future research we can focus on developing cost-effective solutions that can be easily adopted by small-scale farmers.

• Disease prediction: In addition to detecting diseases, future systems can predict the likelihood of disease outbreaks based on weather conditions, soil moisture, and other factors. This can help farmers take preventive measures before the disease occurs, leading to better crop yields and reduced losses.

Also i will try to implement it as an Application, with extending it to more types of diseases, means training it on large size of data-set with some more modification in VGG-16 model to increase accuracy. And also provide proper guidance for particular disease like, treatment, pesticides, etc. An application which can able to run on any device either android or IOS, without any error. Which helps farmers without any effort. And will try to upload it on Google Play-Store.

9 References

[**1**] Irfan Okten and Ugur Yuzgec,Rice plant disease detection using image processing and PNN,ICST Institute for Computer sciences, informatics and Telecommunication Engineering,2022.

[**2**] Albattah, W., Javed, A., Nawaz, M., Masood, M., Albahli, S. (2021). Artificial Intelligence-Based Drone System for Multi-class Plant Disease Detection Using an Improved Efficient Convolutional Neural Network. IEEE Access, 9, 70719-70728.

[**3**] Rao, Y., Wang, F. (2022). Cucumber plant disease detection using improved SWIM transformer algorithm. Computers and Electronics in Agriculture, 201, 107295.

[**4**] Rajneni Deepika Sai, Yogeshwari, and Varsha (2023), Plant Disease Detection using Image Processing and Machine Learning Algorithm, Journal of Xidian University.

[**5**] Aishwarya K, Sreekumar N R (2022) , Plant Disease Detection Using Quantum Image Pro- cessing, International Conference on Industry 4.0 Technology.

[**6**]Pandian, J.A., Kumar, V.D. (2022). Plant Disease Detection Using Deep Convolutional Neural Network. Applied Science, 12(3), 934.

[**7**] Haiqing Wang, Shuqi Shang, Dongwei Wang, Xiaoning He, Kai Feng,Plant Disease Detection and Classification Method Based on the Optimized Lightweight YOLOv5 Model,Agriculture, 2022.

[**8**]Saleem, M. H., Potgieter, J., Arif, K. M. (2022). A Performance-Optimized Deep Learning-Based Plant Disease Detection Approach for Horticultural Crops of New Zealand. IEEE Access, 10, 105965-105975. doi: 10.1109/ACCESS.2022.3050809.

[**9**] R Ganesh Babu C Chellaswamy, Different stages of disease detection in squash plant based on machine learning, Journal of Biosciences, 2022.

[**10**] Chen, H.-C., Widodo, A. M., Wisnujati, A., Rahaman, M. (2022). AlexNet Convolutional Neural Network for Disease Detection and Classification of Tomato Leaf. Electronics, 11(2), 250.

[**11**] Roy, A. M., Bose, R., Bhaduri, J. (2022). A fast accurate fine-grain object detection model based on YOLOv4 deep neural network. Neural Computing and Applications, 34(2), 385-394.

[**12**] Santosh, B., Rakesh, S., Kalifungwa, P., Sahoo, P. R. (2022). Plant

Disease Detection Using Image Processing Methods in Agriculture Sector. In Intelligent Communication Technologies and Virtual Mobile Networks (pp. 1-9). Springer, Cham.

[13] P. Praveen, Mandala Nischitha, Chilupuri Supriya, Mitta Yogitha Aakunoori Suryanandh, To Detect Plant Disease Identification on Leaf Using Machine Learning Algorithms, Intelligent System Design,2022.

[14] Pal, A., Kumar, V. (2022). Plant leaf disease severity classification using agriculture detection framework. Engineering Applications of Artificial Intelligence, 107, 104429. doi: 10.1016/j.engappai.2022.104429.

[15] https://www.kaggle.com/datasets/nafishamoin/new-bangladeshi-crop-disease.

[16] H. Samiullah, M. Awais, and M. Saqib, "Deep convolutional neural networks for efficient and accurate plant disease detection," Computers and Electronics in Agriculture, vol. 195, 2022.

[17] N. Ramzan, S. A. Raza, S. Akbar, and S. M. Shafique, "A novel deep learning approach for plant disease detection using ensemble of convolutional neural networks," Neural Computing and Applications, vol. 34, pp. 10613–10623, 2022.

[18] K. Roy and A. K. Das, "Plant disease detection using a hybrid approach of deep convolutional neural network and extreme learning machine," Computers and Electronics in Agriculture, vol. 195, 2022.

[19] R. Zhao, X. Zhang, Y. Liu, and D. Li, "A novel plant disease detection method based on convolutional neural network with attention mechanism," Multimedia Tools and Applications, vol. 81, pp. 17511–17530, 2022.

[20] Y. Li, X. Li, and X. Liu, "Plant disease detection using a deep neural network with transfer learning," Computers and Electronics in Agriculture, vol. 195, 2022.

[21] Ali, M., Li, Y., Li, R., Ahmad, B., Imran, M. (2022). A Deep Learning-Based Framework for Automated Detection and Classification of Rice Diseases. IEEE Access, 10, 109979-109989.

[22] Kansakar, S. P., Kim, J., Cho, K. H. (2022). A novel deep learning approach for automated rice disease detection and classification. Computers and Electronics in Agriculture, 201, 107044.

[23] Hu, Y., Gao, Y., Zhang, R., Chen, L., Zhang, J. (2022). Rice disease detection based on deep convolutional neural network with multi-feature fusion. Journal of Applied Remote Sensing, 16(1), 016511.

[24] Xie, X., Wang, X., Chen, C., Zhang, W., Chen, X. (2022). Deep Learning-Based Rice Disease Detection Using Combined RGB and Thermal Infrared Images. Frontiers in Plant Science, 13, 839.

Thesis

by Ab De

Thesis

ORIGINALITY REPORT

7%	5%	7%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	"Biometric Recognition", Springer Science and Business Media LLC, 2016 Publication	1%
2	link.springer.com Internet Source	1%
3	dokumen.pub Internet Source	1%
4	vbook.pub Internet Source	1%
5	cdmana.com Internet Source	1%
6	"Recent Trends in Image Processing and Pattern Recognition", Springer Science and Business Media LLC, 2019 Publication	1%
7	epage.pub Internet Source	1%
8	junyelee.blogspot.com Internet Source	1%

9	Thimira Amaratunga. "Deep Learning on Windows", Springer Science and Business Media LLC, 2021 Publication	1 %
10	"Proceedings of the International Conference on Cognitive and Intelligent Computing", Springer Science and Business Media LLC, 2022 Publication	1 %

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	On		