PADDY AND MAIZE LEAF DISEASE DETECTION USING DEEP LEARNING TECHNIQUES

A PROJECT REPORT

Submitted by

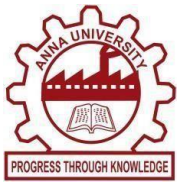
VIGNESH K (191001112)
SHEYNE S CLEETUS (191001090)PRAVEENA S (191001068)
PREETHI R(191001070)

in partial fulfilment for the award of the degreeof

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY



RAJALAKSHMI ENGINEERING COLLEGE, THANDALAM

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report titled **" PADDY AND MAIZE LEAF DISEASE DETECTION USING DEEP LEARNING TECHNIQUES''** is the bonafide workof **"VIGNESH K (191001112), SHEYNE S CLEETUS (191001090)** and **PRAVEENA S(191001068), PREETHI R(191001070)**" who carried out the projectwork under my supervision.

SIGNATURE

Dr. PRIYA VIJAY

THE DEPARTMENT

Information Technology,
Engineering College,
Thandalam,
Chennai – 602105.

SIGNATURE

Ms.A.P.ARUNA JAMEELAHEAD OF SUPERVISOR

Information Technology, Rajalakshmi
Rajalakshmi Engineering College,
Thandalam,
Chennai – 602105.

This project report is submitted for viva voce examination to be held on ……at Rajalakshmi Engineering College, Thandalam.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I would like to thank our Hon. Chairperson **Dr. Thangam Meganathan** and ourHon. Vice-Chairman **Mr. M. Abhay Shankar** for giving us the opportunity to showcase our skill. Then I would also like to extend my heartfelt sincere gratitude to our Principal **Dr. S. N. Murugesan** for providing us this platform toexpose our practical knowledge.

I would also like to extend my heartfelt thanks to **Dr. Priya vijay**(Head of Information Technology Department) for giving us his able support and encouragement.

I must also emphasise the point that this project would not have been possible without the highly informative and guidance of **Ms.A.P.Aruna Jameela**(Associate Professor) whose vast knowledge and experience has greatly helped me in this project. Also, I would like to thank project coordinator **Dr.A.Sathya** for her support throughout the project.

I express my sense of gratitude to all of our staff members and lab assistants of the Information Technology Department who helped me during my project. I am highly grateful to my family and friends for extending their help whenever necessity arose ever and their continuous help to complete my project successfully.

ABSTRACT

In this project presents an automatic approach for early disease and nutrition deficiency detection in Paddy Crop and maize. Agriculture not only provides food for the human existence, it is also a big source for the economy of any country. Millionsof dollars are being spent to safeguard the crops annually. Insects, nutrition deficiency, plant disease and pests damage the crops and, thus, are very dangerous for the overall growth of the crop. One method to protect the crop is early disease detection and nutrition deficiency so that the crop can be protected. The best way to know about the health of the crop is the timely examination of the crop. If disease or nutrition deficiency are detected, appropriate measures can be taken to protect the crop from a big production loss at the end. Early detection would be helpful for minimizing the usage of the pesticides and would provide guidance for the selection of the pesticides. It has become a wide area for research now a days and a lot of research has been carried out worldwide for automatic detection of diseases. Traditional method of examination of the fields is naked eye examination but it is very difficult to have a detailed examination in large fields. To examine the whole field, many human experts are needed which is very expensive and time consuming. Hence an automatic system is required which can not only examine the crops to detectinfestation but also can classify the type of disease on crops. Computer vision techniques provide effective ways for analyzing the images of leaves. Support Vector Machine is used for classification of images with and without disease based on the image features. This technique is simpler as compared to the other automated techniques and provides better results.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACRONYM	EXPLANATION
GA	Genetic Algorithm
RCNN	Regional Convolutional Neural Network
SGDM	Spatial Gray-Level Dependence Matrices
ANN	Artificial Neural Network
DCE	Discrete Contour Evolution
RGB	Red Blue Green
DFD	Data Flow Diagram

1. INTRODUCTION

1.1 GENERAL

Agriculture is the foundation of the Indian economy. Direct or indirect involvement in farming activities accounts for 50% of the population. This region produces a wide range of fruits, cereals, and vegetables that are sold to other nations. So, it is essential to make goods of the highest quality with the highest output. The diagnosis of plant diseases is crucial in the realm of agriculture because plant illnesses cannot be prevented. Diseases can affect several portions of plants, including the fruits, stems, and leaves. The three principal plant diseases include bacterial, fungal, and viral diseases like Alternaria, Anthracnose, bacterial spot, and canker, among others. The bacterial disease is caused by the presence of germs in leaves or plants, the viral disease is caused by environmental changes, and the fungus disease is caused by the leaf has fungus on it. Leaf diseases can be recognised using the proposed approach. Since it is possible to automatically identify diseases from the symptoms that manifest on plant leaves, automatic detection of plant diseases is an important research area. Barbedo proposed a technique for automatically segmenting disease symptoms in digital images of plant leaves that involves applying colour channel manipulation and Boolean operations to a binary mask of the leaf pixels. He suggested a method of manipulating the H and colour channel histograms to semi-automatically segment the symptoms of plant leaf disease. Pang et al. suggested using a local threshold and seeded region growing to automatically segment crop leaf spot disease photos. Singh and Misra suggested employing soft computing approaches for plant leaf disease detection. By using texture-based clustering for segmentation, Prasad et al. suggested a segmentation method for natural plant leaf diseases that is unsupervised and resolution independent. A method to segment leaves in images with non-uniform lighting was proposed by Du and Zhang and is based on genetics and

the maximum entropy algorithms (GA). Dhaygude and Kumbhar suggested employing image processing to identify agricultural plant leaf disease, where the texture statistics are calculated using spatial gray-level dependence matrices (SGDMFor the segmentation of plant disease spots, Diao et al. explored a variety of techniques, such as edge-based, region-based, Artificial Neural Network (ANN), etc. In the literature, various techniques for automatically segmenting leaves and identifying diseases have been suggested.

1.2 SCOPE OF THE PROJECT

Using the right methods to distinguish between healthy and diseased leaves reduces crop loss and boosts productivity. This section includes various RCNN-Deep Learning algorithms that are currently in use to identify plant diseases. The majority of farmers are uneducated and underprivileged, which may lead to problems caused by animals and plant diseases that affect more than half of their crops. Additionally, it can be used in other disciplines of agricultural disease diagnosis in similar application settings.

2.LITERATURE SURVEY

2.1 RELATED WORK

PAPER1:

TITLE: Estimation Of Leaf Angle Distribution Based On Statistical Properties Of Leaf Shading Distribution

AUTHOR: Kuniaki Uto , Mauro Dalla Mura , Yuka Sasaki and Koichi Shinoda

YEAR: 2020

ABSTRACT:

An essential phenotypic characteristic that is connected to photosynthesis is leaf angle dispersion. Leaf-scale airborne photographs with great spectral and spatial resolution are now available, thanks to the recent development of drones and high-resolution photography technologies. This work represents the first attempt to distinguish between plants with various leaf angle distributions using a single leafscale image. A collection of rice leaf surfaces is first approximated by a hemiellipsoid surface, presuming that a rice leaf surface resembles a portion of a hemiellipsoid surface. Under various direct sunlight orientations, timeseries of shade distributions on the hemiellipsoids with various structural parameters are produced. We determined an appropriate time window for image acquisitions, i.e., 11:00–12:30, by examining the statistical properties, i.e., skewness, kurtosis, and the most probable intensity, of the frequencies of the simulated shading intensity that welldifferentiate hemiellipsoids with different structural parameters. Later, under direct sunlight, rice plants with and without silicate fertiliser were captured in time-series leaf-scale pictures and depth maps. . The depth maps proved that leaves treated with silicate fertiliser are more erect than leaves from untreated plants. The most likely intensity of the leaf-scale images taken during the appropriate time slot and 89% and 100% of kurtosis were shown to have consistent relationships with the simulations, indicating that the proposed method can be used to distinguish between different leaf angle distributions based on the frequency of shading intensity in rice leaf images.

PAPER-2:

TITLE: Plant Species Identification from Occluded Leaf Images

AUTHOR: Ayan Chaudhury, and John L. Barron

YEAR: 2020

ABSTRACT:

Using a database of whole plant leaves, we offer a method for identifying the plant species using the contour data from occluded leaf images. Matching occluded leaves with entire leaf datasets is an open and under-researched subject, despite the recent considerable study of contour-based 2D form matching. Due to the wide variety and intricate leaf architecture, classifying occluded plant leaves is considerably more difficult than complete leaf matching. Our approach is inherently poor since matching an occluded contour with all the full contours in a database is an NP-hard task. Secondly, a -Spline curve is used to represent the 2D contour points. Following that, we use the Discrete Contour Evolution (DCE) technique to extract interest points from these curves. We employ subgraph matching to create a number of open curves for each closed leaf contour using the DCE points as graph nodes. Then, for each open curve, we compute the similarity transformation parameters (translation, rotation, and uniform scaling). Then, using the Frechet distance metric to assess the quality of the match and keeping the best-matched curves, we "overlay" each open curve with the inverse similarity converted occluded curve. We develop an energy functional that is well connected with the Frechet metric since it is inexpensive to calculate but is not perfectly correlated with the match's quality. The functional makes use of String Cut characteristics, Shape Context descriptors, and local and global curvature. Using a framework of convex-concave relaxation, we minimise this energy functional. The curve with the lowest energy among these optimal curves is regarded as having the greatest overall compatibility with the occluded leaf. Tests on three publicly accessible leaf image databases demonstrate that our method outperforms previous state-of-the-art techniques in terms of effectiveness and efficiency. The percentage of the overall contour (and not the leaf area) that is missing is how occlusion is calculated. We demonstrate that our method can still find the best whole leaf match from the databases even for leaves with significant degrees of occlusion (let's say 50% occlusion).

PAPER-3:

TITLE: Tomato Septoria Leaf Spot Necrotic and Chlorotic Regions Computational Assessment Using Artificial Bee Colony-Optimized Leaf Disease Index.

AUTHOR: Ronnie Concepcion, Sandy Lauguico, Elmer Dadios, Argel Bandala, Edwin Sybingco, Jonnel Alejandrino.

YEAR: 2020

ABSTRACT:

Because to the complicated nature of chlorosis and the error-prone sphere of colours and textures affected by angle photosynthetic light source, visual evaluation of plant health status and disease severity may result in subjective assessments. It is crucial to quantify the effects of damaging diseases on leaves in order to understand how pathogens and plants interact. The integration of computational intelligence and computer vision for tomato Septoria leaf spot necrotic and chlorotic region computational assessment is the suggested answer to this problem. Individually photographed tomato leaves from healthy and diseased plants are included in the dataset. With the aid of the CIELab colour space, non-vegetation pixels were removed. The split leaf was used to extract five Haralick texture features and RGB colour components. The essential predictors leading to the RGB-entropy vector were chosen using a hybrid neighbourhood component analysis and ReliefF algorithm. By normalising apparent red reflectance and introducing red-green and red-blue reflectance ratios to boost Septoria leaf spots pixels and lessen sensitivity to healthy green pixels, a novel tomato leaf disease index (tomLDI) optimised using artificial bee colony (ABC) was created. With a 97.46% accuracy rate, KNN outperformed the classification tree, linear discriminant analysis, and Naive Bayes in identifying Septoria leaf disease. Using tomLDI converted coloured channels and raw infected leaf images, MobileNetV2, ResNet101, and InceptionV3 were used to perform deep transfer image regression. With R2 values of 0.9930 and 0.9484, respectively,.

PAPER:4

TITLE: Leaf Segmentation and Tracking Using Probabilistic Parametric Active Contours

AUTHOR: Jonas De Vylder, Daniel Ochoa, Wilfried Philips, Laury Chaerle, and Dominique Van Der Straeten

ABSTRACT:

Segmentation and tracking frequently use active contours or snakes. The minimization of an energy function, which is often a linear combination of a data fit term and a regularisation term, is necessary for these strategies. To take advantage of the inherent object and image properties, this energy function can be altered. This can be accomplished by adjusting the regularisation and data fit weighting settings. Nevertheless, there is no fixed way to set these settings so that they work best for a certain application. Trial and error parameter estimate is the result of this. In this research, we suggest a novel active contour definition framework based on probability theory. \ Since this novel method uses probability distributions that can be learned from a given training dataset, there is no requirement for ad hoc parameter tuning.

PAPER:5

TITLE: Natural Genetic Variation for Growth and Development Revealed by High-Throughput Phenotyping in Arabidopsis thaliana

AUTHOR: Xu Zhang, Ronald J. Hause, Jr and Justin O. Borevitz

ABSTRACT:

The rate of photosynthesis and carbon fixation of a plant is determined by the growth and development of its leaves. These morphological characteristics result from the gradual blending of hereditary and environmental influences throughout time. Due to the intricacy of the feature and the requirement for real-time assessment, it is challenging to precisely identify the developmental genetic foundation of leaf growth

across a growing season. In this study, we created a method for analysing time-lapse images that tracks leaf expansion in response to changes in seasonal light. To account for developmental variation, three growth traits—rosette leaf area, circle area, and their ratio as compactness—were quantified and normalised on a linear timeframe. For all development characteristics that evolved throughout time, we discovered high heritability. Our research demonstrates a high-throughput, cost-effective phenotyping method that makes it possible to analyse the genetic underpinnings of plant shoot development and growth in a changing environment.

2.2 EXISTING SYSTEM:

Early disease identification is a significant problem in agriculture research under the current system. Plant leaf because they can significantly lower the quality and output of agricultural products, illnesses have become a problem.The primary strategy used in practise for the detection and identification of vegetable illnesses is the naked eye observation of experts. Unfortunately, this necessitates ongoing expert supervision, which in large farms may be prohibitively expensive. However, in some poor nations, farmers may need to travel great distances to see experts, which makes doing so costly and time-consuming. As a result, the system is unable to alert the person who should be informed about the bug in the leaf.

2.3 DISADVANTAGES IN EXISTING SYSTEM:

- Raters might grow weary and lose focus, which would reduce their accuracy.
- Repetition of training can be necessary to keep up quality.
- A rater is costly.
- Visual evaluation may be harmful if samples are taken in the field and evaluated in a lab.
- Significant differences between and within-rater
- Standard area diagrams must be created to facilitate assessment.

3 PROPOSED SYSTEM DESIGN

3.1 PROPOSED SYSTEM

Our suggested approach focuses on the quickly evolving fields of computer vision and image processing; it has the potential to increase the precision and effectiveness of traditional agricultural tasks like disease detection and early warning. We identify numerous leaf parameters in this system. With the aid of digital image processing tools, the characteristics of the leaves are monitored. The controller serves as a means of alerting or providing information.

3.1.1 ADVANTAGES OF PROPOSED SYSTEM:

- Improved image quality for illness detection.
- Clear visual rating in image processing approach
- Comparably lower cost.
- There is no need to create any assisted diagrams for image

3.2 ARCHITECTURE DIAGRAM

3.2.1 TRAINING PART:

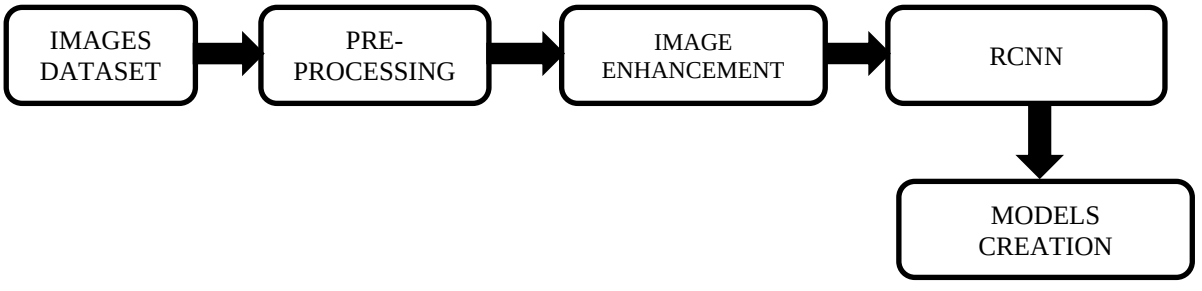


Figure 3.1

3.2.2 TESTING PART:

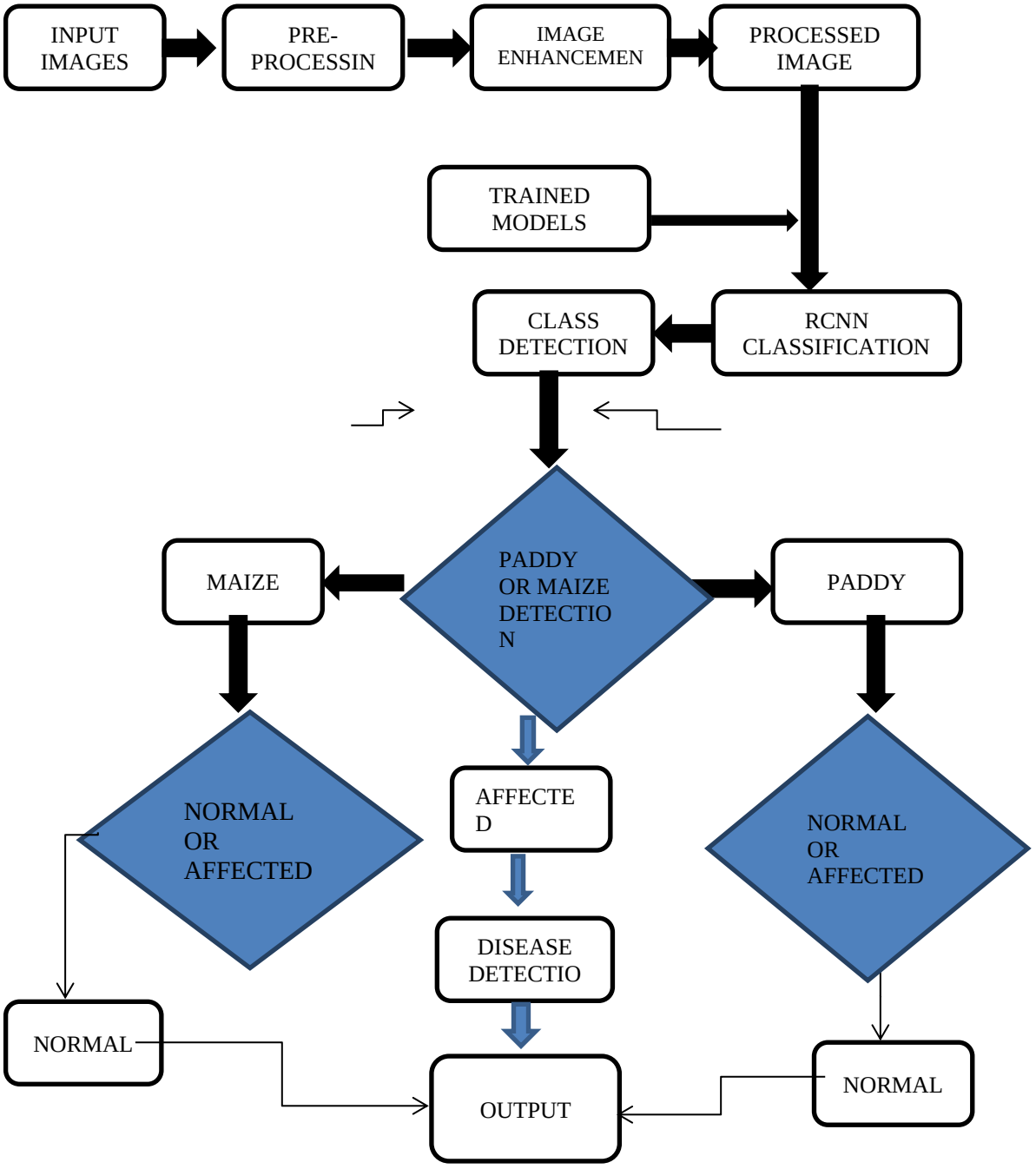


Figure 3.2

3.4 MODULE DESCRIPTION

LIST OF MODULES

1. Image Preprocessing
2. Image Enhancement
3. Feature Extraction
4. Detecting Leaf
5. RCNN Classification

3.4.1 IMAGE PREPROCESSING:

The category of a given input fruit image in the dataset is determined using an image classification task. It is the fundamental undertaking in advanced picture comprehension and can be split into binary and multiple classification tasks. The output layer of an RCNN performs several convolution and pooling operations after which an image is categorised in accordance with the requirements. Between binary and multi classification tasks, the sole distinction is the activation function of the output layer. A good performance in natural image classification, including the use of Region-based Convolution Neural Networks (RCNNs) in JPG/PNG image classification, can be achieved for the image classification task for fruit image analysis that is easily identifiable.

3.4.2 IMAGE ENHANCEMENT:

The RGB image that was obtained is first transformed into grey. Now that our image is more contrasted with the backdrop, a suitable threshold level may be chosen for binary conversion. Image enhancing techniques are required here. The goal of image enhancement is to modify an image so that the final product is better suited for the

intended application than the original image. The features of an image can be played with using a variety of techniques, though not always. Here are a few basic operations that are widely used to improve images.

3.4.3 FEATURE EXTRACTION:

The feature extraction approach is used by To create derived values (features) that are intended to be useful, deep learning, pattern recognition, and image processing informative and non-redundant. Use is made of an initial set of measured data. This approach speeds up generalisation and learning, and in some circumstances, it improves human interpretations. Dimension reduction and feature extraction work together seamlessly. If the input data for an algorithm is too extensive to analyse or is deemed redundant, it may be condensed to a smaller set of attributes (referred to as a feature vector). The process of selecting a portion of the original traits is known as feature selection. The expectation is that the selected characteristics the required information from the input data will be included rather than using the entire starting set of data to carry out the desired activity.

3.4.4 DETECTING LEAF:

Since it is a good leaf, the leaf is a very prominent feature in the picture. When a leaf is running, it must be located. The margins can be used for the same thing because the leaf is good. After the thresholds are appropriately calibrated, it is discovered that canny edge detection produces very good results. Before edge detection, noise in the image can be removed. A group of lines is produced as a result of edge detection. We need to extract the leaf out of it.

3.4.4 DETECTING LEAF:

Since it is a good leaf, the leaf is a very prominent feature in the picture. When a leaf is running, it must be located. The margins can be used for the same thing because the leaf is good. After the thresholds are appropriately calibrated, it is discovered that canny edge detection produces very good results. Before edge detection, noise in the image can be removed. A group of lines is produced as a result of edge detection. We need to extract the leaf out of it.

3.4.5 RCNN CLASSIFICATION

According to region Applications for image and video recognition using convolutional neural networks. The main applications of RCNN in image analysis are segmentation, object identification, and image recognition. The convolutional layer in a conventional neural network connects each input neuron to the next hidden layer. Only a subset of the hidden layer's neurons are connected to those in the input layer. Use the pooling layer to make the feature map less dimensional. In the buried layer of the RCNN, there will be numerous activation and pooling layers. The top tiers of the network are referred to as Fully Connected tiers. The output of the last convolutional or pooling layer is delivered to the layer that is totally connected and flattened there before being put into use.

3.5UML DIAGRAM

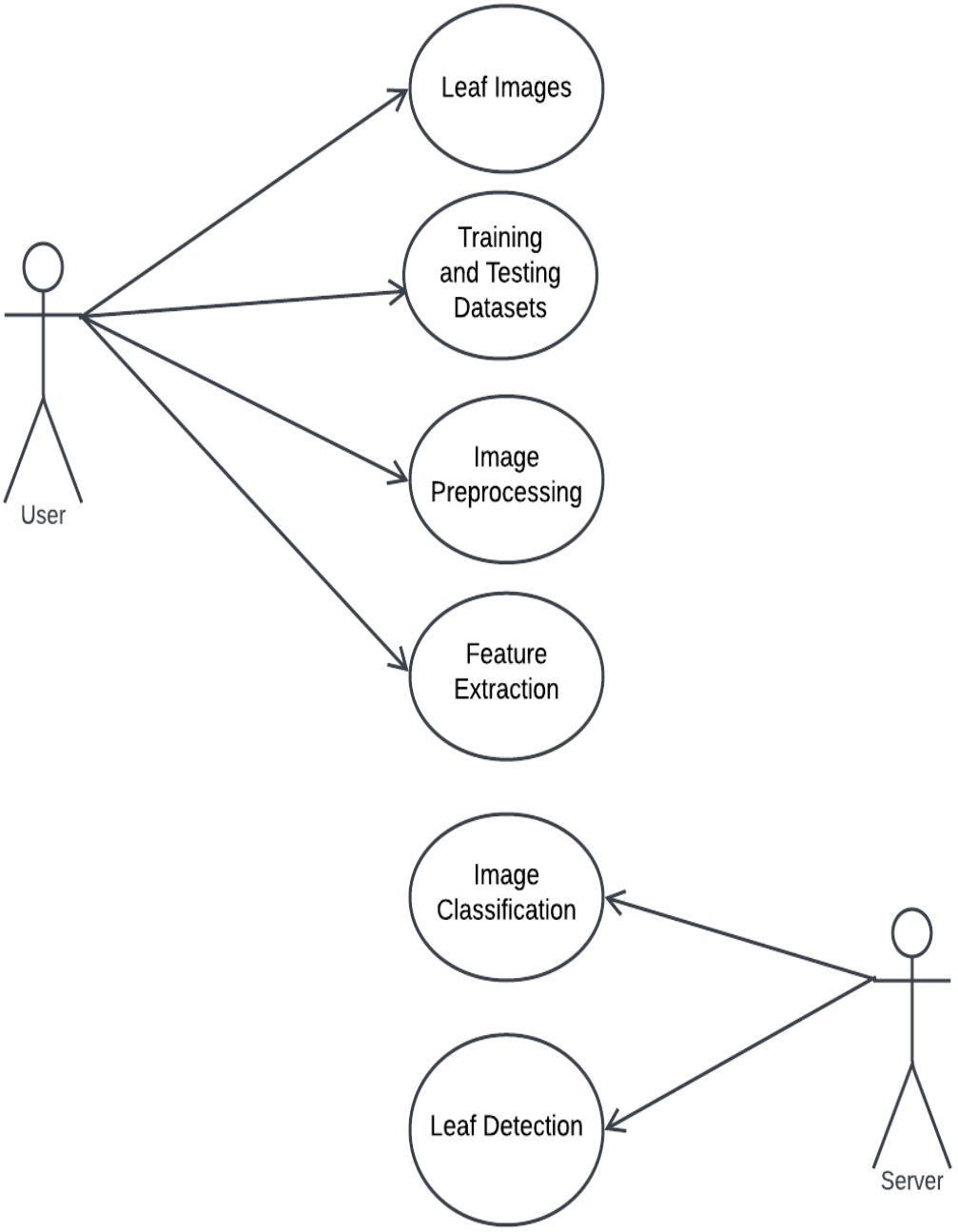


Figure 3.3

4 SYSTEM IMPLEMENTATION:

4.1 SOFTWARE AND HARDWARE REQUIREMENTS:

4.1.1 HARDWARE REQUIREMENTS

- Processor – i3,i5,i7 , AMD Processor
- RAM - 8 GB
- Hard Disk - 500 GB

4.1.2 SOFTWARE REQUIREMENTS

- Operating System : Windows 8/10/11
- Language : Python
- Framework : GUI(Python)
- Tool : IDLE(Python)

4.2 TECHNOLOGIES USED

4.2.1 R-CNN?

A type of machine learning model called R-CNN, also referred to as RCNN, is utilised for computer vision applications, particularly for object detection. Region-Based Convolutional Neural Network is known as R-CNN. After that, we'll look at the RCNN architecture to understand what it is.

4.2.2 How does R-CNN work?

The picture below exemplifies the idea of a local CNN. (R-CNN). After using bounding boxes to build object regions then each Region of Interest (ROI) is analysed individually using convolutional networks to categorise different picture regions. In order to address challenges with image detection, the RCNN architecture was created.

Mask R-CNN is built on the R-CNN architecture, which was also used to develop

What is R-CNN faster at?

Region Proposal Network, a faster form of R-CNN designs with two steps, (RPN). RPN is only a Neural Network that suggests various items that are present in a specific image. Quick R-CNN. Each candidate box is used to extract features using RoIPool (Region of Interest Pooling), and then classification and bounding-box regression are carried out. A tiny feature map can be extracted from each RoI in a detection using the RoIPool function..

Faster R-CNN advances this stream by utilising a Region Proposal Network and Fast R-CNN architecture to comprehend the attention process. "Fast R-CNN" is quicker than R-CNN since it does not require the convolutional neural network's 2 000 region proposals to be fed continually. Instead, the convolution operation—which is only applied once per image—produces a feature map. Additionally, because Faster R-CNN was developed to accelerate computing, it is an upgraded form of R-CNN.

The key difference between the two RCNNs is that Faster R-CNN uses a "Region Proposal Network," whereas Fast R-CNN uses selective search to produce Regions of Interest. (RPN). Now let's look at how Mask-R-CNN was constructed using Faster R-CNN.

Describe Mask R-CNN.

Mask R-CNN, sometimes known as Mask RCNN, is the most advanced Convolutional Neural Network (CNN) for instance and picture segmentation. Faster R-CNN, a region-based convolutional neural network, served as the foundation for Mask R-CNN. knowing the idea of image segmentation is a prerequisite for knowing

how Mask R-CNN operate. The task of computer vision The technique of dividing a digital image into several parts is called image segmentation. (sets of pixels, also known as image objects). In this segmentation, borders and objects are located. (lines, curves, etc.). Mask R-CNN encompasses two primary forms of picture segmentation: Instance-Based Semantic Segmentation

Describe Mask R-CNN.

Mask R-CNN, sometimes known as Mask RCNN, is the most advanced Convolutional Neural Network (CNN) for instance and picture segmentation. Faster R-CNN, a region-based convolutional neural network, served as the foundation for Mask R-CNN. knowing the idea of image segmentation is a prerequisite for knowing how Mask R-CNN operate. The task of computer vision The technique of dividing a digital image into several parts is called image segmentation. (sets of pixels, also known as image objects). In this segmentation, borders and objects are located. (lines, curves, etc.). Mask R-CNN encompasses two primary forms of picture segmentation: Instance-Based Semantic Segmentation.

The benefits of Mask R-CNN

Performance: On every job, Mask R-CNN exceeds all currently available single-model entries. Efficiency: The technique is extremely effective and only slightly increases the overhead of Faster R-CNN. Flexibility: Mask R-CNN is simple to apply to different tasks. For instance, in the same framework, Mask R-CNN can be used to estimate human pose.

The essential component of Fast/Faster R-CNN that is lacking from Mask R-CNN is the pixel-to-pixel alignment. The same two-step method, with an identical first stage, is used by Mask R-CNN. (which is RPN). In the second stage, Mask R-CNN additionally produces a binary mask for each RoI in addition to class and box offset predictions. In contrast, the

majority of modern systems rely on mask predictions for classification. Furthermore, the Faster R-CNN framework, which supports a large variety of customizable architecture designs, makes it easy to develop and train Mask R-CNN. Additionally, the mask branch only incurs a minor computational cost, allowing for quick testing and a quick system.

4.2.3 Semantic Segmentation

Semantic segmentation divides each pixel into a preset set of categories without distinguishing between different object instances. To put it another way, semantic segmentation identifies and groups similar objects into a single category at the pixel level. Everything was grouped together, as can be seen in the image above. (person). Semantic segmentation, also referred to as background segmentation, separates the image's subjects from its backdrop.

4.2.4 Instance Segmentation

The method of precisely recognising each Instance segmentation, also known as instance recognition, is the process of identifying each instance of an object in an image and meticulously segmenting each instance. It integrates object detection, object positioning, and object categorization as a result. To put it another way, this type of segmentation takes a step further to highlight the differences between each object and those that are grouped together as comparable instances. As shown in the example image above, all objects for the segmentation example are humans; nevertheless, throughout the segmentation process, each person is regarded as a separate entity. Semantic segmentation, also known as foreground segmentation, focuses the subjects of the image rather than the background.

5 PERFORMANCE ANALYSIS

5.1 TYPE OF TESTING USED

5.1.1UNIT TESTING

By creating test cases for unit testing, you can make sure that the core programme logic is operating with accuracy and that inputs into the programme create legitimate outputs. It is crucial to examine the internal code flow and each decision branch. It entails testing every software element of every programme separately. After each unit but before integration, it is finished. To conduct this intrusive structural test, a comprehension of its construction is required. Unit tests, which closely investigate a particular configuration of a system, application, or business process, are used to do fundamental testing at the component level. Unit tests make certain that every phase of a business process properly adheres to the established standards and has inputs and outputs that are well defined.

5.1.2 INTEGRATION TESTING

There are software components that have been subjected to integration tests to ascertain whether they actually operate as a single programme. Event-driven testing stresses a screen's or field's primary result. Even though unit testing of the individual components demonstrated that they passed successfully, integration tests show that the components are correct and consistent when integrated. Integrity testing seeks to draw attention to problems that develop during component fusion.

5.1.3 FUNCTIONAL TEST

Functional tests offer thorough proof that the features being tested are functional and satisfy all business and technical requirements, as well as those listed in the system documentation and user manuals.

The following are the areas where functional testing is most prevalent:

Valid Input: Classes of valid input must be recognised and accepted.

Illegal Input: Some forms of illegal input must be disregarded.

Functions: Use of the following functions is required.

Some application classes' output needs to be tested.

Utilize integrated systems or procedures. Functional testing is organised and prepared with a focus on the requirements, crucial functions, or unique test cases. Complete data field coverage, established practices, adherence to protocols, and business process activities must all be considered during testing. Before functional testing is complete, further tests are discovered, and the worth of the ones that are currently in place is evaluated.

5.1.4 SYSTEM TESTING

To make certain that the integrated software system as a whole is reliable, system testing is required complies with requirements. It assesses a setup to produce consequences that are known and foreseeable. System testing is exemplified by the system integration test with a focus on configuration. The foundation process flows and descriptions are part of system testing, along with an emphasis on integration points and pre-driven links**.**

5.1.5 WHITE BOX TESTING

White box testing can be carried out by software testers who are conversant with the software's internal workings, structure, and terminology or who the very least are aware of what it is designed to perform. It serves a function. It is employed to investigate problems covered by the black box level of restriction.

5.1.6 BLACK BOX TESTING

Software is tested in a "black box," without understanding the architecture, language, or internal workings of the module being tested. such the vast majority of other test types, black box tests a transparent source document, such as a requirements or specification document must be used as the foundation for creation.The computer code being tested is treated as a "black box" in this kind of examination. a transparent source document, such requirements or a specification document, must be used as the foundation for creation . Software engineering requirements for Chapter V:

5.2 PYTHON INTRODUCTION

Python is an advanced object-oriented programming language that Guido van Rossum created. It is also referred to as a general-purpose programming language because it is used in almost every sector of business imaginable, including those in the following list. Web development, software development, game development, and data analytics. This list might go on, but let's focus on the reasons why Python is so well-liked in the next section.

WHY PROGRAM IN PYTHON?

Why Python? is a question that some of you may have. Why not utilise a different programming language?

I'll now explain:

Every programming language has a function or use case that is specific to a given domain. For instance, Javascript is the most popular language used by web developers

because it allows the developer the ability to manage applications using various frameworks, such as react, vue, and angular, which are used to create attractive User Interfaces. They also simultaneously have advantages and disadvantages. Python is therefore considered general-purpose, which means it is extensively used in all domains. This is due to the fact that it is relatively simple to comprehend and scalable, which contributes to the rapidity of development.. Now that you understand why learning Python is not a prerequisite, you can see why Python is popular among developers as well. Python's syntax is comparable to that of the English language, making it easier for programmers to construct programmes with fewer lines of code. Since it is open-source, a variety of libraries are available that simplify the work of developers and increase output. They is simple to concentrate on business logic and its difficult skills in the digital age because information is readily available in vast data sets.

THE PYTHON STANDARD LIBRARY

The Python Language Reference covers the precise syntax and semantics of the Python language, while this library reference article explains what the standard library is provided with Python. A couple of the optional parts that are typically added in Python versions are also discussed.A good illustration of the vastness and versatility of the Python standard library is provided by the lengthy table of contents that follows. Both pre-built (C) modules that allow access to the system for Python programmers resources like file I/O that they otherwise wouldn't have, and Python modules built to offer common solutions for many problems that arise in everyday programming are included in the library.A number of these modules were created by abstracting away platform-specifics into platform-neutral APIs in order to enable and improve the portability of Python programmes. The majority of these, The Python installers for Windows usually include the standard library in addition to countless other parts. Since Python is often provided as a a collection of programmes for Unix-

like operating systems, it could be required to utilise the operating system's packaging tools to acquire any or all of the supplementary elements. Besides the traditional library, the Python Package Index provides a growing number of several thousand components, ranging from standalone programmes and modules to comprehensive application development frameworks.

A Python Package: What Is It?

We'll take a quick look at scripts and modules before discussing Python packages. In the shell, a "script" is something you run to complete a certain purpose. You would enter your code into your preferred text editor and save it with the.py extension to create a script. Then, you may run your script using the python command at a terminal.

On the other hand, a module is a Python programme that you import into other programmes or into interactive mode. The name "module" is actually a catch-all for reusable code. Typically, a Python package is made up of numerous modules. A package is a folder that physically contains modules as well as maybe other folders that in turn may include still more folders and modules. In terms of concept, it's a namespace. This merely means that a package's modules are linked together by a name that may be used to refer to them.

In reference to our earlier definition of a module as reusable, importable code, we should point out that while every package is a module, not every module is a package. One file called __init__.py is typically found in a package folder, and it basically informs Python, "Hey, this directory is a package!" The init file may be blank or it may have code that will run when the package is initialised.

Probably as well, you've heard the word "library."

Although a library's definition in Python is less explicit than that of a package or a module, it is generally accepted that if a package has been made public, it may also be referred to as a library.

HOW TO USE A PYTHON PACKAGE

Namespaces, publishing packages, and importing modules have all been mentioned. We can help if any of these words or ideas aren't totally obvious to you. We'll go through all you need to know in this section so that you may use Python packages in your code with confidence.

Python Package Import

We'll use the import statement to import a package:

Assume that no packages have yet been installed. The Python Standard Library is a sizable collection of pre-installed packages that comes with Python. It has features for many different use cases, like text processing and math. Importing the latter

An import statement could be viewed as a module's search trigger. Searches are well planned: Python first searches the cache for a module before moving on to the standard library and then a list of paths. Once sys has been imported, you can view this list. (another standard library module).

The directories that Python will search for packages in are all listed by the sys.path command. You might download a package and attempt to import it, but you encounter an error:

In these situations, make sure your imported package is listed in one of Python's search pathways. If not, you can always add more search paths to your list:

After receiving an import statement, the interpreter will then have more than one

place to look for packages.

Aliases and Namespaces

We initialised the math namespace after importing the math module. As a result, we can now use "dot notation" to refer to the math module's functions and classes:

Assume that we are solely interested in the factorial function from our math module and that we are also sick of dot notation. Then, we can move forward as follows:

You can simply comma-separate the resources in the import statement if you want to import more than one resource from the same source:

However, there is always a slight possibility that your variables will conflict with those in your namespace. What if another variable in your programme has the same name as log? It would replace the log function, which would lead to errors. It's best to import the package like we did before to prevent that. You can alias your package to give it a shorter name to save down on typing time:

Aliasing is a very typical tactic. There are some packages with widely used aliases: For instance, "np" is virtually always imported when the NumPy library of numerical computations is used.

Another choice is to import every resource from a module into your namespace:

This strategy, however, has a significant risk because it's uncommon for you to be familiar with every name found in a package, which raises the possibility of your variables being overwritten. The majority of seasoned Python programmers will advise against using the wildcard * in imports because of this. Namespaces are another fantastic notion, as The Zen of Python puts it.

Installing Python Packages

What about software programmes that are not included in the standard library? The Python Package Index, more commonly referred to as PyPI, is the official repository for discovering and downloading such third-party programmes. Use the package installer pip to install packages from PyPI:
Python packages can be installed with pip from any repository, not just PyPI. The conda command can be used to install Python packages if Anaconda or Miniconda were used to install Python.

Conda is relatively user-friendly but less adaptable than pip. So you may always try pip if conda is unable to install a package for you.
Loading a Module Again
Even if you provide another import statement, changes to a module's script made while programming in interactive mode won't be imported. Use the reload() function from the importlib package in this situation:

How to Create Your Own Python Package

You may or may not want your code to be published to PyPI when you package it for future usage. Maybe all you want to do is give it to a friend or use it again. Whatever your goal, there are a number of files you need to incorporate into your project. The __init__.py file has already been described. The setup.py file is another crucial one. This file uses the setuptools package to offer comprehensive details about your project and a list of all dependencies, or packages needed by your code to function correctly. Beyond the scope of this basic course is publishing to PyPI. However, if you do have a package ready for release, your project needs two more files: a Markdown README.md and a licence. If you wish to learn more, refer to the official Python Packaging User Guide (PyPUG).

INSTALLING PACKAGES

This section covers the basics of how to install Python packages.

It's important to note that the term "package" in this context is being used to describe a bundle of software to be installed (i.e. as a synonym for a distribution). It does not to refer to the kind of package that you import in your Python source code (i.e. a container of modules). It is common in the Python community to refer to a distribution using the term "package". Using the term "distribution" is often not preferred, because it can easily be confused with a Linux distribution, or another larger software distribution like Python itself.

Requirements for Installing Packages

This section describes the steps to follow before installing other Python packages.

Ensure you can run Python from the command line

Before you go any further, make sure you have Python and that the expected version is available from your command line. You can check this by running:

Unix/macOS

```
python3 --version
```

Windows

You should get some output like `Python 3.6.3`. If you do not have Python, please install the latest 3.x version from python.org or refer to the Installing Python section of the Hitchhiker's Guide to Python.

Note

If you're a newcomer and you get an error like this:

```
>>> python --version
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'python' is not defined
```

It's because this command and other suggested commands in this tutorial are intended to be run in a *shell* (also called a *terminal* or *console*). See the Python for Beginners getting started tutorial for an introduction to using your operating system's shell and interacting with Python.

Note

If you're using an enhanced shell like IPython or the Jupyter notebook, you can run system commands like those in this tutorial by prefacing them with a `!` character:

```
In [1]: import sys
        !{sys.executable} --version
Python 3.6.3
```

It's recommended to write `{sys.executable}` rather than plain `python` in order to ensure that commands are run in the Python installation matching the currently

running notebook (which may not be the same Python installation that the python command refers to).

Note

Due to the way most Linux distributions are handling the Python 3 migration, Linux users using the system Python without creating a virtual environment first should replace the python command in this tutorial with python3 and the python -m pip command with python3 -m pip --user. Do *not* run any of the commands in this tutorial with sudo: if you get a permissions error, come back to the section on creating virtual environments, set one up, and then continue with the tutorial as written.

Ensure you can run pip from the command line

Additionally, you'll need to make sure you have pip available. You can check this by running:

Unix/macOS

```
python3 -m pip --version
```

Windows

If you installed Python from source, with an installer from python.org, or via Homebrew you should already have pip. If you're on Linux and installed using your OS package manager, you may have to install pip separately, see Installing pip/setuptools/wheel with Linux Package Managers.

If pip isn't already installed, then first try to bootstrap it from the standard library:

Unix/macOS

```
python3 -m ensurepip --default-pip
```

Windows

If that still doesn't allow you to run `python -m pip`:

- Securely Download get-pip.py 1
- Run `python get-pip.py`. 2 This will install or upgrade pip. Additionally, it will install setuptools and wheel if they're not installed already.

Warning

Be cautious if you're using a Python install that's managed by your operating system or another package manager. get-pip.py does not coordinate with those tools, and may leave your system in an inconsistent state. You can use `python get-pip.py --prefix=/usr/local/` to install in `/usr/local` which is designed for locally-installed software.

Ensure pip, setuptools, and wheel are up to date

While `pip` alone is sufficient to install from pre-built binary archives, up to date copies of the `setuptools` and `wheel` projects are useful to ensure you can also install from source archives:

Unix/macOS

```
python3 -m pip install --upgrade pip setuptools wheel
```

Windows

Optionally, create a virtual environment

See section below for details, but here's the basic venv 3 command to use on a typical Linux system:

Unix/macOS

```
python3 -m venv tutorial_env
source tutorial_env/bin/activate
```

Windows

This will create a new virtual environment in the tutorial_env subdirectory, and configure the current shell to use it as the default python environment.

Creating Virtual Environments

Python "Virtual Environments" allow Python packages to be installed in an isolated location for a particular application, rather than being installed globally. If you are looking to safely install global command line tools, see Installing stand alone command line tools.

Imagine you have an application that needs version 1 of LibFoo, but another application requires version 2. How can you use both these applications? If you install everything into /usr/lib/python3.6/site-packages (or whatever your platform's standard location is), it's easy to end up in a situation where you unintentionally upgrade an application that shouldn't be upgraded.

Or more generally, what if you want to install an application and leave it be? If an application works, any change in its libraries or the versions of those libraries can break the application.

Also, what if you can't install packages into the global site-packages directory? For instance, on a shared host.

In all these cases, virtual environments can help you. They have their own installation directories and they don't share libraries with other virtual environments.

Currently, there are two common tools for creating Python virtual environments:

- venv is available by default in Python 3.3 and later, and installs pip and setuptools into created virtual environments in Python 3.4 and later.
- virtualenv needs to be installed separately, but supports Python 2.7+ and Python 3.3+, and pip, setuptools and wheel are always installed into created virtual environments by default (regardless of Python version).

The basic usage is like so:

Using venv:

Unix/macOS

```
python3 -m venv <DIR>
source <DIR>/bin/activate
```

Windows

Using virtualenv:

Unix/macOS

```
python3 -m virtualenv <DIR>
source <DIR>/bin/activate
```

Windows

For more information, see the venv docs or the virtualenv docs.

The use of **source** under Unix shells ensures that the virtual environment's variables are set within the current shell, and not in a subprocess (which then disappears, having no useful effect).

In both of the above cases, Windows users should _not_ use the **source** command, but should rather run the **activate** script directly from the command shell like so:

```
<DIR>\Scripts\activate
```

Managing multiple virtual environments directly can become tedious, so the dependency management tutorial introduces a higher level tool, Pipenv, that automatically manages a separate virtual environment for each project and application that you work on.

Use pip for Installing

pip is the recommended installer. Below, we'll cover the most common usage scenarios. For more detail, see the pip docs, which includes a complete Reference Guide.

Installing from PyPI

The most common usage of pip is to install from the Python Package Index using a requirement specifier. Generally speaking, a requirement specifier is composed of a project name followed by an optional version specifier. **PEP 440** contains a **full specification** of the currently supported specifiers. Below are some examples.

To install the latest version of "SomeProject":

Unix/macOS

```
python3 -m pip install "SomeProject"
```

Windows

To install a specific version:

Unix/macOS

```
python3 -m pip install "SomeProject==1.4"
```

Windows

To install greater than or equal to one version and less than another:

Unix/macOS

```
python3 -m pip install "SomeProject>=1,<2"
```

Windows

To install a version that's **"compatible"** with a certain version: 4

5.3 CODING

MAIN.PY

```
import tensorflow as tf
from keras_preprocessing.image import ImageDataGenerator
from keras_preprocessing import image
import numpy as np
import easygui
from keras.models import load_model
import os
import serial
import tkinter as tk
from tkinter import *
from tkinter import filedialog
from tkinter.filedialog import askopenfile
from PIL import Image, ImageTk
my_w = tk.Tk()
sw=my_w.winfo_screenwidth()
sh=my_w.winfo_screenheight()
my_w.geometry('%dx%d'%(sw,sh))
my_w.title('Leaf Detection')
my_font1=('times', 18, 'bold')

bg = ImageTk.PhotoImage(file='leaf.webp')
bgLabel = Label(my_w, image=bg)
bgLabel.place(x=0, y=0)

l1 = tk.Label(my_w,text='Upload Files & get
```

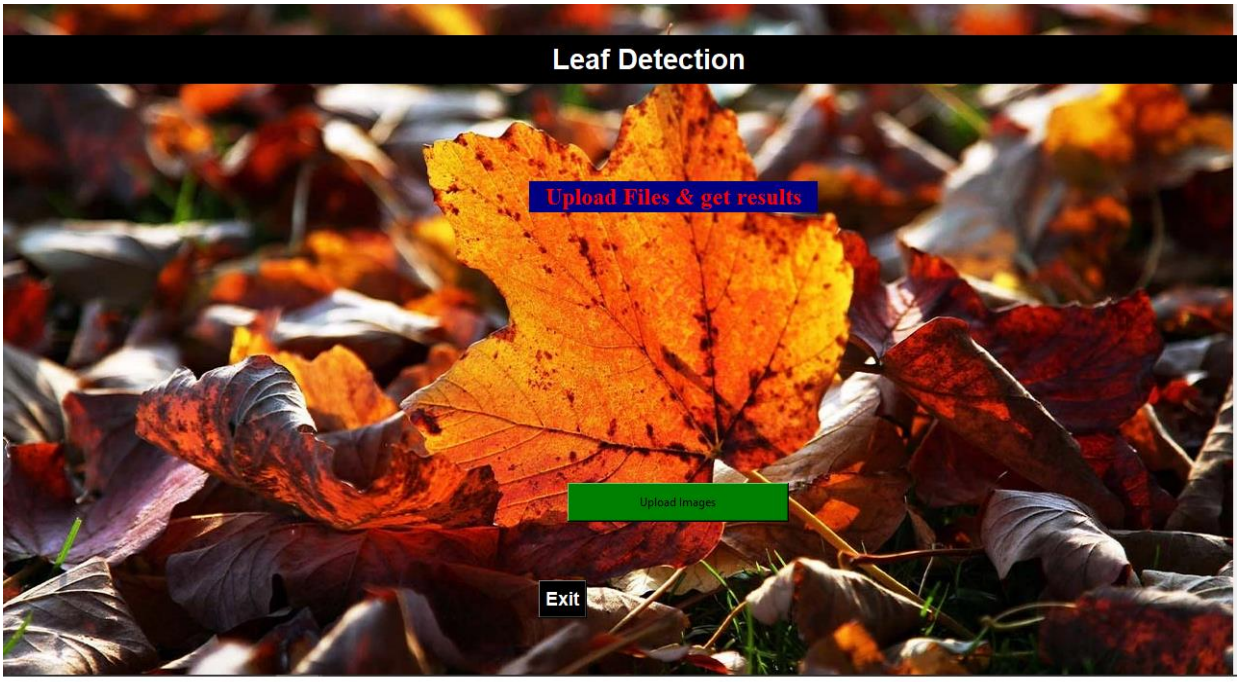
```
results',width=30,font=my_font1,bg='#000080',
                fg='red',)
l1.place(x=550, y=190, width=300)
b1 = tk.Button(my_w, text='Upload Images',
  width=20,command = lambda:result(), activebackground='#000080', bg='green')
b1.place(x=590,y=500, width=230, height=40)

print(tf.__version__)

def close():
  my_w.destroy()


titleLabel = Label(my_w, text='Leaf Detection', font=('italic', 22, 'bold '), bg='black',
                fg='white', )
titleLabel.place(x=0, y=40, width=1350, height=50)

endbtn=Button(my_w,text="Exit",font='italic 14
bold',bg='black',fg='white',command=close)
endbtn.place(x=560,y=600,width=50)

model1 = load_model('model/Class1/model_Class1.h5')
model2 = load_model('model/Class2/model_Class2.h5')
model3 = load_model('model/Class3/model_Class3.h5')
model4=load_model('model/Class4/model_Class4.h5')
model5=load_model('model/Class5/model_Class5.h5')
```

```
def result():

  filename =upload_file()
  test_image2 = image.load_img(filename, target_size = (64, 64))
  test_image2 = image.img_to_array(test_image2)
  test_image2 = np.expand_dims(test_image2, axis = 0)
  # cnn prediction on the test image
  result2 = model1.predict(test_image2)
  print(result2)
  result3 = model2.predict(test_image2)
  print(result3)
  result4 = model3.predict(test_image2)
  print(result4)
  result5=model4.predict(test_image2)
  print(result5)
  result6=model5.predict(test_image2)
  print(result6)
  if result2[0][0]==1:
    if result3[0][0]==0:
      if result4[0][0]==1:
        prediction2='Paddy: Leaf Smut'
      else:
        prediction2='Paddy: Brown Spot'
    else:
      prediction2='Paddy: Healthy'
  else:
    if result5[0][0]==0:
```

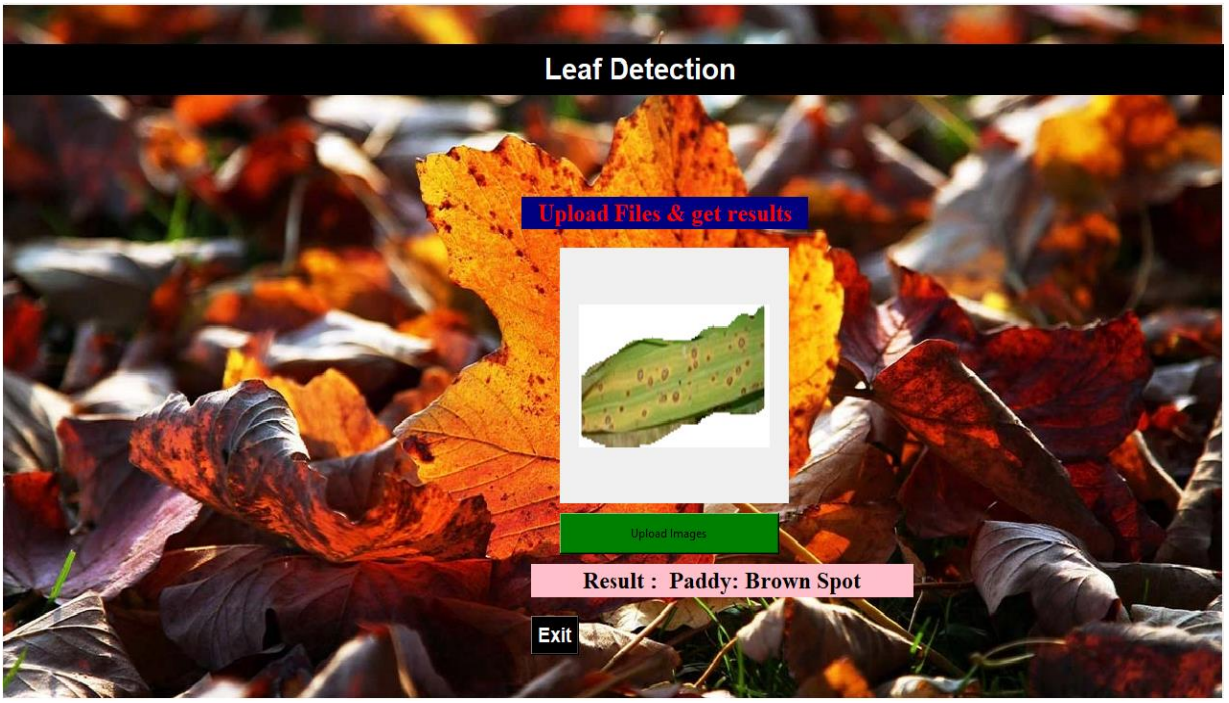
```
        if result6[0][0]==1:
            prediction2='Maize: Gray Leaf Spot'
        else:
            prediction2='Maize: Common Rust'
      else:
        prediction2='Maize: Healthy'
      print(prediction2)
   prediction=prediction2
   l2 = tk.Label(my_w,text="Result :
"+prediction,width=50,font=my_font1,bg='pink',
                 fg='black',)
   l2.place(x=560, y=550, width=400)
   return filename

def upload_file():
    filename=easygui.fileopenbox()
    img=Image.open(filename) # read the image file
    img=img.resize((200,140)) # new width & height
    img=ImageTk.PhotoImage(img)
    e1 =tk.Label(my_w)
    e1.place(x=590, y=240, width=240, height=250)
    e1.image = img
    e1['image']=img
    return filename

my_w.mainloop()
```

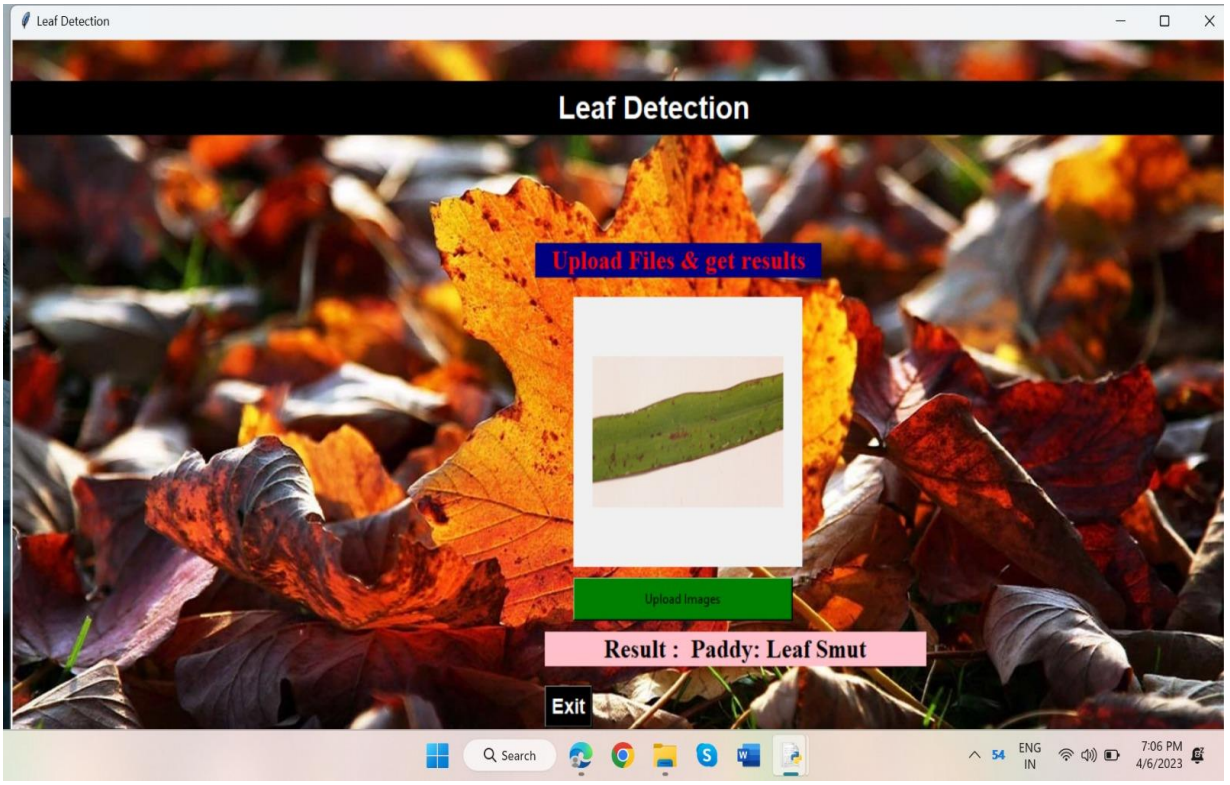
5.3.1 OUTPUT SCREENSHOTS:

Home Page

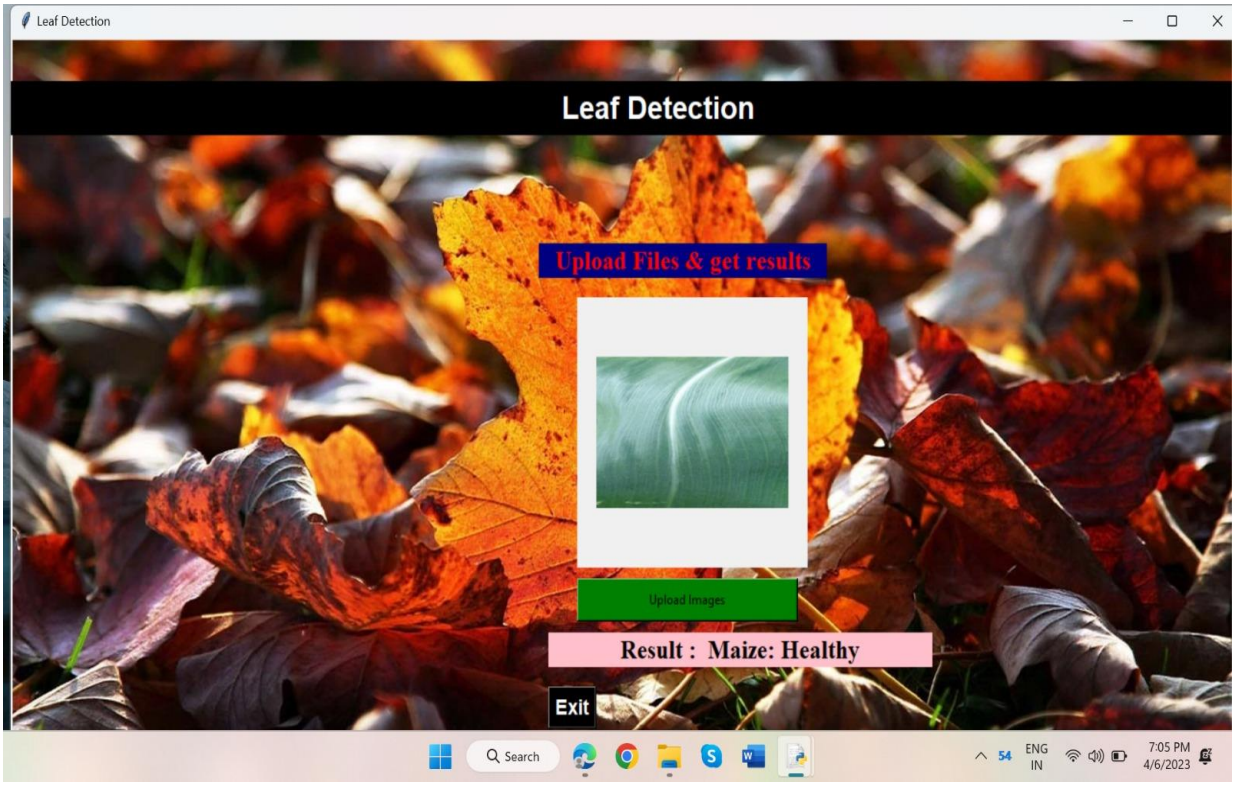


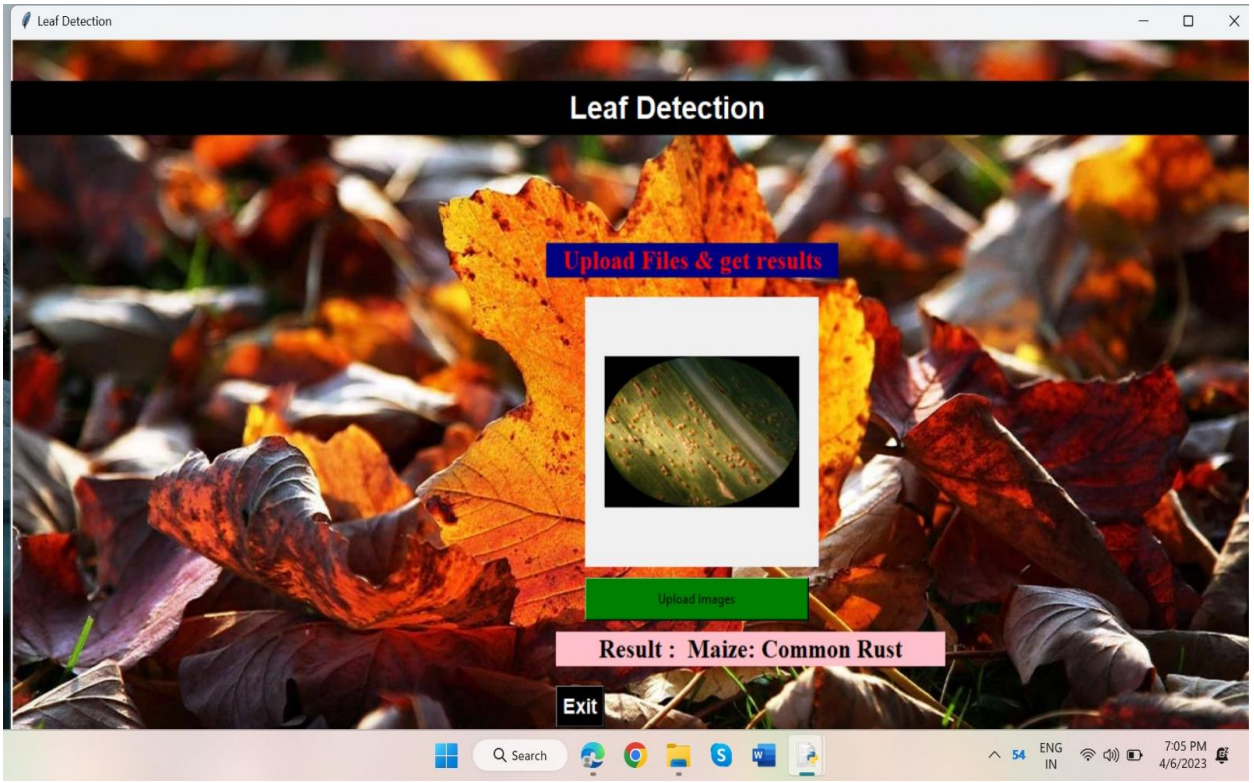
Paddy Leaf:




Leaf Detection
Leaf Detection
Upload Files & get results
Upload Images
Result : Paddy: Leaf Smut
Exit
Search
ENG
IN
7:06 PM
4/6/2023

Maize Leaf:


Leaf Detection
Leaf Detection
Upload Files & get results
Upload Images
Result : Maize: Healthy
Exit
Search
ENG
IN
7:05 PM
4/6/2023


Leaf Detection
Leaf Detection
Upload Files & get results
Upload Images
Result : Maize: Common Rust
Exit
Search
ENG
IN
7:05 PM
4/6/2023


Leaf Detection
Upload Files & get results
Upload Images
Result : Maize: Gray Leaf Spot
Exit

6 CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

Implemented is a technique for paddy and maize leaf diseases early detection and classification. The dataset of paddy and maize leaves from sick plants is utilised to train the region-based convolutional neural network model. The paddy and maize leaf disease is then classified according to its stage using a deep learning system that uses RCNN. Disease detection using the approach is being tried on paddy and maize leaf tissue. Further research will be done to classify diseases in many plant species and to increase the classification's accuracy.

6.2FUTURE WORK:

Our upcoming projects include Leaf Counting as well. Data association is a part of this assignment, thus we must determine whether or not a leaf was visible in the previous image frame. To identify leaf, feature matching, tracking, and association approaches are needed.

REFERENCES:

[1] X.-S. Wei, Y.-Z. Song, O. M. Aodha, J. Wu, Y. Peng, J. Tang, J. Yang, and S. Belongie, ‘‘Finegrained image analysis with deep learning: A survey,’’ IEEE Trans. Pattern Anal. Mach. Intell., early access, Nov. 13, 2021, doi: 10.1109/TPAMI.2021.3126648.

[2] J. Yin, A. Wu, and W. S. Zheng, ‘‘Fine-grained person re-identification,’’ Int. J. Comput. Vis., vol. 128, no. 12, pp. 1654–1672, 2020.

[3] S. D. Khan and H. Ullah, ‘‘A survey of advances in vision-based vehicle re-identification,’’ Comput. Vis. Image Understand., vol. 182, pp. 50–63, May 2019.

[4] X.-S. Wei, Q. Cui, L. Yang, P. Wang, and L. Liu, ‘‘RPC: A large-scale retail product checkout dataset,’’ 2019, arXiv:1901.07249.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ‘‘ImageNet classification with deep convolutional neural networks,’’ in Proc. Adv. Neural Inf. Process. Syst. (NIPS), Lake Tahoe, NV, USA, Dec. 2019, pp. 1097–1105. [32]

[6] K. Simonyan and A. Zisserman, ‘‘Very deep convolutional networks for large-scale image recognition,’’ 2020, arXiv:1409.1556.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, ‘‘Going deeper with convolutions,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Boston, MA, USA, Jun. 2020-, pp. 1–9.

[8] K. He, X. Zhang, S. Ren, and J. Sun,‘‘Deep residual learning for image recognition,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, USA, Jun. 2020, pp. 770–778.

[9] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, ‘‘ Densely connected convolutional networks,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Honolulu, HI, USA, Jun. 2019, pp. 2261–2269.

[10] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang, ‘‘Transfer learning using computational intelligence: A survey,’’ Knowl.-Based Syst., vol. 80, pp.

Paddy and Maize leaf disease detection using deep learning techniques

ORIGINALITY REPORT

10%	3%	4%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to SASTRA University Student Paper	2%
2	R. Meena Prakash, G.P. Saraswathy, G. Ramalakshmi, K.H. Mangaleswari, T. Kaviya. "Detection of leaf diseases and classification using digital image processing", 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2017 Publication	1%
3	www.ijsred.com Internet Source	1%
4	Febina S. N, Mohamed Mubarak T. "Predicting The Level Of Anxiety Of Coronavirus Pandemic Among College Students Using Different Classifier Algorithms", 2022 International Conference on Innovations in Science and Technology for Sustainable Development (ICISTSD), 2022 Publication	1%

5	Revathi Vankayalapati, Akka Lakshmi Muddana. "Accurate Brain Tumor Recognition Using Double-Weighted Feature Extraction Labelling Model with Priority Weighted Feature Selection", Traitement du Signal, 2021 Publication	1 %
6	Submitted to The University of Wolverhampton Student Paper	1 %
7	Submitted to Visvesvaraya Technological University, Belagavi Student Paper	1 %
8	Submitted to University of Lancaster Student Paper	1 %
9	knowledgecommons.lakeheadu.ca Internet Source	1 %
10	Submitted to Houston Community College Student Paper	<1 %
11	Submitted to St. Patrick's College Student Paper	<1 %
12	ukdiss.com Internet Source	<1 %
13	"Intelligent Computing Theories and Application", Springer Science and Business Media LLC, 2017 Publication	<1 %