Abstract

Plant diseases are the biggest barrier to the safe production, quality, and sourcing of important agricultural products. In addition to reducing yield, they are of particular concern because of their direct impacts on human and animal health. To ensure minimal losses to the cultivated crop, it is crucial to supervise its growth. An abundance of direct and indirect methods of plant disease classification exist, including molecular and serological methods, but they require a considerable amount of manual labour and specialized equipment. Hence, to make the process of plant disease detection less laborious as well as a feasible, we aim to build a Utility System to Analyze Diseased Plants and Predict the Stages using Machine Learning Algorithms. The plants here include tomato, bell pepper and potato and some of the diseases that they might have are bacterial spot, Blight, Leaf Moid, mosaic virus, Yellow leaf curl virus etc.

The application of image processing techniques and neural network models are reported to enhance agricultural practices and use in quality and safety inspection of various plants. Few major diseases affecting these crops have been studied and algorithms for the detection of disease and it's stages have been implemented. In this project we aim to use Convolutional Networks as our base Architecture and develop three different models, they are , DenseNet-169, VGG-16 and a Modified CNN model. A comparative analysis of all the three models will also be made available for better performance analysis for the user.

As a result of the project, we aim to design a system which takes an image as the input from the user and passes to each of the models, classifies it against the model and predicts the class with high accuracy, compares the accuracies of each CNN model and identifies the one with the best accuracy. An option to search for the remedy using google search engine is also provided. The project also focuses on the user experience to make the process of detection, classification and prediction for the users simple and easily accessible

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 General Introduction

Plant or Crop diseases are a key danger for food security, but their speedy identification is still difficult in many portions of the world because of the lack of the essential infrastructure. Plant diseases are one of the causes in the reduction of quality and quantity of agriculture crops. Reduction in both aspects can directly affect the overall production of the crop in a country. The main problem is a lack of continuous monitoring of the plants. Sometimes amateur farmers are not aware of the diseases and its occurrence period. Generally, diseases can occur on any plant at any time. However, continuous monitoring may prevent disease infection. The detection of a plant disease is one of the important research topics in the agriculture domain.

In a survey done, it was revealed that agriculturalists found the task of identifying a plant disease to be difficult at times and needed a guiding hand in diagnosing plant diseases and also in figuring out potential curing methods for them.

1.2 Problem Statement

In this Project, using the art of Deep Learning techniques, we aim to demonstrate the feasibility of our approach by using a public dataset for healthy and infected Tomato,Potato and bell pepper Leaves, to produce a model that can be used to classify and identify types of Leaf diseases.

1.3 Objectives of the Project

The objectives of this project are:

- To identify deficiency of particular plant by analysing its leaf efficiently.
- Demonstrating the feasibility of using deep convolutional neural networks to classify Tomato, potato and bell pepper Leaf diseases.
- To obtain a model that can identify and classify accurately.

- To obtain an acceptable accuracy.
- To develop a model that can be practically used in detection of disease and thus enable the users to plan the further approach to handle the disease.

1.4 Project Deliverables

The recent advances in the field of deep learning have enabled us to further understand systems and more accurately predict outcomes based on historical data. Deep learning is successfully able to model large amounts of labeled data accurately. The advantage of a neural network is its ability to generate the best outcome without the need for redesigning output criteria.

Convolutional Neural Networks (CNN or ConvNet) are complex feed forward neural networks. It follows a hierarchical model which works on building a network like a funnel and finally giving out a fully connected layer where all the neurons are connected to each other and the output is processed. They find a particular application in image classification and recognition due to its high accuracy and funnelling technique which enables extraction of useful features in a faster manner.

We thus perform leaf disease detection using the techniques of Image processing and construction of a CNN (Convolutional Neural Networks) which would possess the ability of accurate classification of diseased tomato, potato and bell pepper leaves.

1.5 Current Scope

Constant efforts have been made by various researchers and developers to adopt Image Processing techniques for detection of diseases that produce symptoms like spots, patches, lesions, and variation in the roughness of the surface.

Some proposed papers include various phases of implementation, namely dataset creation, feature extraction, training the classifier and classification. The created datasets of diseased and healthy leaves are collectively trained under Random Forest to classify the diseased and healthy images. For extracting features of an image, they use Histogram of an Oriented Gradient (HOG). Overall, using Machine Learning to train the large data sets available publicly gives us a clear way to detect the disease present in plants on a colossal scale.

Some other papers aim to motivate researchers to focus and develop efficient Machine Learning and classification techniques for leaf identification and disease detection to meet the new challenges in the field of agriculture.
Similarly, there are other manual methods that exist in order to classify and predict leaves and hence the crop as diseased or not. Although there exist quite a few approaches to this problem, the accuracy obtained by any of them was not very high and in cases of manual examination the process becomes slow.

1.6 Future Scope

This project can be further extended to include a larger number of crops. It can also be deployed as a mobile application thus ensuring convenient use. We can also improve classification of the diseased plant by including further factors such as weather data, soil data etc. The application must also be tested in real-time to demonstrate its feasibility and scalability.

Chapter 2

Project Organization

2.1 Software Process Model

This project uses Iterative and Incremental Development Methodology which is best suited for this project.

Agile SDLC model is a combination of Iterative and Incremental Process Models with focus on process adaptability and rapid delivery of working software products.

Iterative Process can be referred to as a process that starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed. Incremental Development is done in steps from analysis design, implementation, testing/verification, maintenance. Each subsequent release of the system adds function to the previous release until all designed functionality has been implemented. Agile Software Development employs both these development techniques but also applies feedback across releases. Iterative and Incremental Development Methodology was developed in response to the weakness of the waterfall model.

Since this project is delivered in small increments adding functionality to each version, Iterative and Incremental Development Methodology would be best suited.

2.2 Roles and Responsibilities

Ritu Sinha	• Project head • Working towards developing Modified CNN model, DenseNet CNN model and VGG CNN model using a public dataset. • To detect the diseases in the plant leaf and analyse the accuracies obtained through the CNN models. • Part of data pre-processing which involves data cleaning, data sampling and data transformation.

S Spandana	• Data collection based on prediction requirements of the project by browsing through Kaggle, GitHub, etc. • Data visualization and labelling of the data using tools like spreadsheets and visuals to represent large information in forms that is easier to understand and analyse. • Documentation of the step by step Achievements of the Project
Shruthi Raju	• Working towards developing the Modified CNN model using the public dataset. • Building a Graphical user interface in the form of a web application to project the accuracies obtained from the CNN models. • Strategizing the entire project through work flow structures and estimation techniques.
Srishti Bijjur	• Part of data pre-processing which involves Data formatting which involves bringing the collected data in the format for instance pixels and image size etc. before being used for training and testing. • Data splitting which involves splitting data into test, training and validation sets. • CNN model evaluation and testing through Graphical User Interface and Documentation

Chapter 3

Literature Survey

3.1 Introduction

Plants are the primary producers of the ecosystem. They are the main source of energy for all animals and are an essential element to the human life cycle. There are several factors that affect the production of plants. Climate change is an extremely important factor in causing diseases in plants. The exponential population is one of the factors contributing to climate change which has caused a decrease in crop yield. India being primarily an agricultural economy, it is essential to detect these diseases in their early stages as they could spread through entire farms, thus causing a huge loss for the farmer as well as placing a strain on the supply of the crop. Correct recognition of the disease forms a crucial step in effective disease management. A common practice is manual disease detection through human experts to identify and recognize plant diseases. This process is laborious, expensive, and requires a considerable amount of human effort. With improvements in technology, the automatic detection of plant diseases from raw images has become a reality. This provides a cost-effective early detection method for monitoring large fields without much human effort.

Smart phones in particular have opened a new avenue in the process of disease detection in plants, owing to their high computational power, high resolution displays and inbuilt set of accessories, such as HD cameras. The factors of high processing power of mobile devices, widespread smartphone penetration and HD cameras has led to the feasibility of an automated system of disease recognition at an unprecedented scale.

Computer vision has made tremendous growth in recent years. M.Mehdipour et.al [12], 2015 Large Scale Visual Recognition Challenge based on the ImageNet dataset serves as a benchmark for numerous visualization related problems in computer vision in computer vision, including object classification. While training large neural networks can be time consuming, the classification of trained models is a quick process, thus making it apt for deployment on smartphones.

Image analysis is used as a fundamental tool for recognizing, differentiating, and quantifying diverse types of images, including grayscale, color and multispectral images. The application of image processing techniques including color imaging, visible and near infrared spectroscopy, hyperspectral imaging and multispectral imaging techniques are reported to enhance agricultural practices and use in quality and safety inspection of various plants. It has been increasingly utilized for the assessment of plant growth and health for decades. Gradually, these techniques were used for plant disease detection, pest detection and identification, plant and plant part identification and so on.

3.2 Related Works

Hand engineered features like SIFT and SURF have been the foundation for image classification tasks. D.L. Borges.et.al [1], 2016 and H. Bay.et.al [2], 2008 propose a method to detect and grade bacterial spot diseases in visible spectrum images of tomato fields is presented. The method applies segmentation and clustering techniques to CIE Lab color space channels. A comparison of the method to expert's evaluation, and a correlation to productivity in the field was demonstrated.

Hand engineered features are then fed into a learning algorithm such as support vector machines or K-means. Satish Madhogaria et.al [3], 2011 proposes a pixel-based, discriminative classification algorithm for automatic detection of unhealthy regions in leaf images. The algorithm is designed to distinguish image pixels as belonging to one of the two classes: healthy and unhealthy. The task is solved in three steps. First, segmentation is performed to divide the image into foreground and background. In the second step, a support vector machine (SVM) is applied to predict the class of each pixel belonging to the foreground. And finally, further refinement is performed by neighbourhood-check to omit all falsely-classified pixels from the second step. The Architecture diagram for this SVM model is shown in Figure 3.2.1.

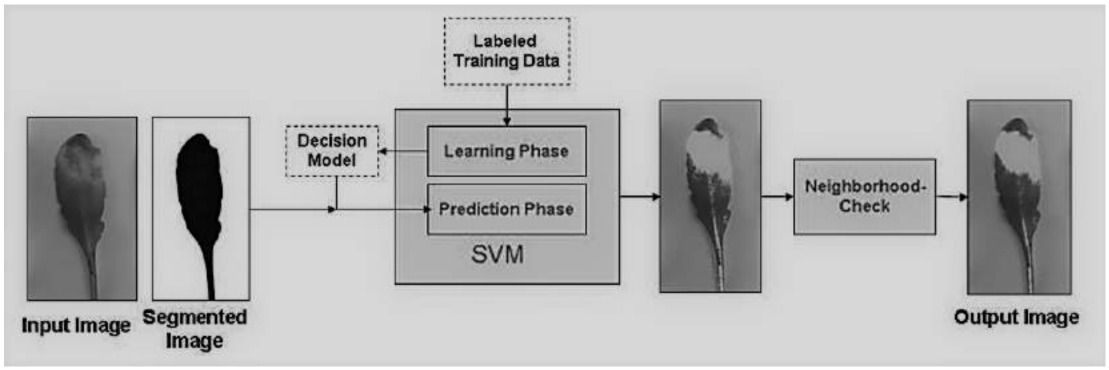


Figure 3.2.1

Architecture Diagram for [3]

Pixel based classification for detecting unhealthy regions

Rupali.et.al [4],2016 has proposed a method that is based on K-means technique is a grape disease partition clustering technique used to partition n number of observations into k clusters .In this technique, k is the number of clusters in the segmented image and colors present in an image are used for the clustering. Then the infected cluster was converted into HIS format from RGB format. In the next step, for each pixel map of the image for only HIS images the SGDM matrices were generated. Finally, the extracted feature was recognized.

Raza.et.al. [5],2016 proposes a feature extraction and classification pipeline which combines thermal and visible light image data with depth information and forms a machine learning system to remotely detect plants infected with the tomato powdery mildew fungus *Oidium neolycopersici*. A novel feature set from the image data using local and global statistics and we see that by combining these with the depth information, the accuracy of detection of the diseased plants can be improved considerably. The Architecture diagram for the above described feature extraction model is shown in Figure 3.2.2.

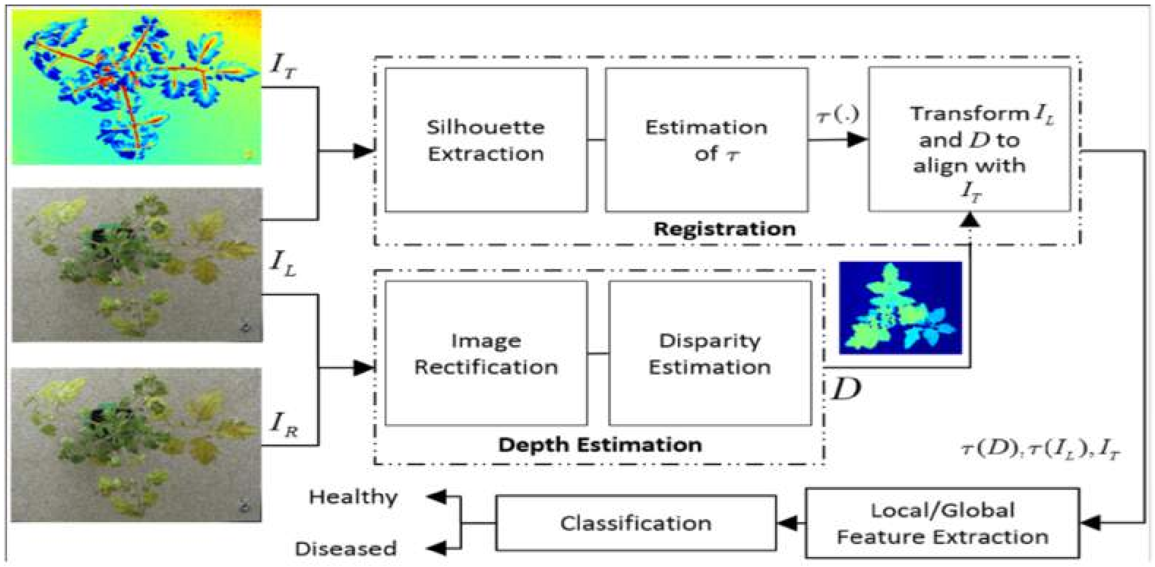


Figure 3.2.2

Architecture Diagram for [5]

Feature Extraction and Classification

Chene.et.al [6], 2012 focused primarily on the assessment of potential depth imaging systems for 3D measurements in the context of phenotyping. It proposes an algorithm for the segmentation of depth images from a single top view of the plant. This paper thus opens the perspective in the direction of high-throughput phenotyping in controlled or field environments.

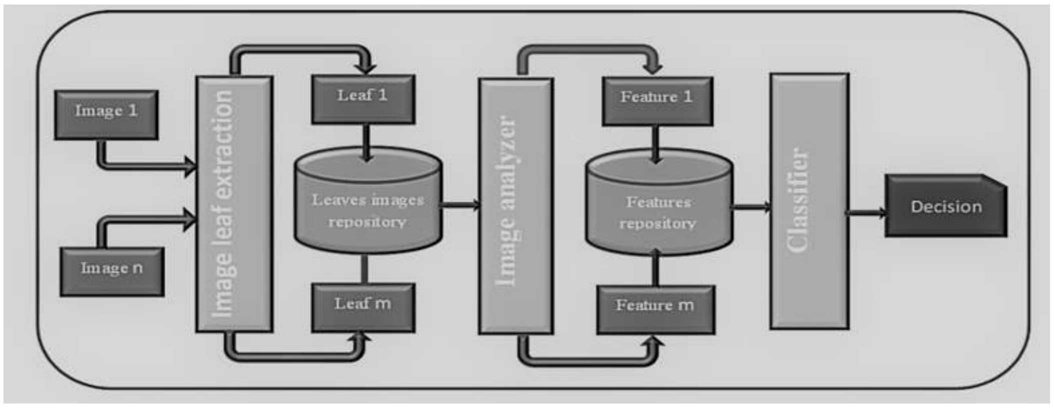


Figure 3.2.3

Architecture Diagram for [7]

Mokhtar.et.al [7],2015 based on the classification of two tomato leaf viruses, has proposed an approach consisting of four main phases; namely pre-processing, image segmentation, feature extraction, and classification phases. Each input image is segmented and descriptor created for each segment. Some geometric measurements are employed to identify an optimal feature subset. Support vector machine (SVM) algorithms with different kernel functions are used for classification. The datasets of a total 200 infected tomato leaf images with TSWV and TYLCV were used for both training and testing phases. N-fold cross-validation technique is used to evaluate the performance of the presented approach. This approach was shown to have obtained an accuracy of 90% on average and 92% based on the quadratic kernel function. The Architecture Diagram for this paper is depicted in Figure 3.2.3.

Wetterich.et.al[8],2012 proposes a system with an aim to develop and evaluate computer vision and machine learning techniques for classification of Huanglongbing-(HLB)-infected and healthy leaves using fluorescence imaging spectroscopy. The fluorescence images were segmented using normalized graph cut, and texture features were extracted from the segmented images using co-occurrence matrix. The extracted features were used as an input into the classifier, support vector machine (SVM). The classification results were evaluated based on classification accuracy and number of false positives and false negatives. The results indicated that the SVM could classify HLB-infected leaf fluorescence intensities with up to 90% classification accuracy.

Dong.et.al [9], 2013 studied the disease of cucumber downy mildew, powdery mildew and anthracnose leaf image processing and recognition technologies. It applies a median filtering method of filtering noise, leaf spot disease of cucumber leaf color range segmentation part ; extraction of color feature parameters of the lesion site, characteristic parameters of the shape and texture parameters by using gray level co-occurrence matrix. Based on the shortest distance methods to identify diseases of images, the technique was able to obtain a disease recognition accuracy of more than 96%.

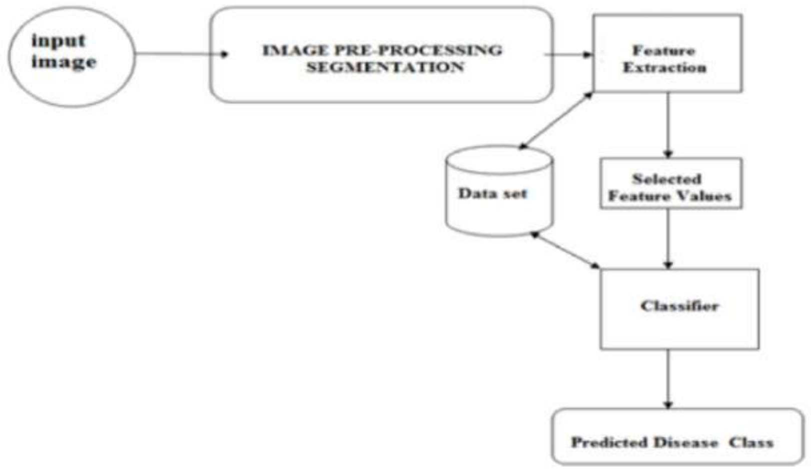


Figure 3.2.4

System Architecture for [11]

A. Krizhevsky et.al [10], 2012 proposes a system in which they trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes and prove that deep neural networks are more capable of capturing abstract concepts invariant to phenomena in the visual world.

Anand.H.Kulkarni et.al [11], 2012 presents a methodology for early and accurate plant diseases detection, using Artificial Neural Network (ANN) and diverse image processing techniques. As the proposed approach is based on ANN Classifier for classification and Gabor Filter for feature extraction, it gives better results with a recognition rate of up to 91%. An ANN based classifier classifies different plant diseases and uses the combination of textures, color and features to recognize those diseases. The Architecture Diagram for this paper is depicted in Figure 3.2.4.

Mr. Sachin B. Jagtap et.al. [23], 2014 describes a system consisting of four stages: the first stage is the image enhancement, which includes, histogram analysis, HSI enhancement and intensity adjustment. Fuzzy c-means algorithm is used for segmentation of captured images. Color, shape of spot, size are three features used to extract features from leaf. Then classification is based on Back Propagation based Neural Networks.

Prof. Sanjay, B. Dhaygude et.al. [24] , 2013 has explained the application of texture statistics for detecting plant leaf disease by color transformation structure. RGB is converted into HSV space because HSV is a good color descriptor. Masking and removing of green pixels is performed with pre-computed threshold level. Then in the next step, segmentation is performed using 32X32 patch size and useful segments are obtained. These segments are used for texture analysis by Color Co-occurrence Matrix. Finally the texture parameters are compared to texture parameters of normal leaf.

In recent years, convolutional neural networks, a special type of DNN have emerged to be a powerful framework for the recognition and classification of a variety of image domains. M.Havaei.et.al [13], 2017 proposes a system where they have drawn attention with respect to semantic segmentation, L.A. Alexandre et. al [14], 2016 proposes a system that focuses on object detection, H. Xue.et.al [15], 2016 proposes a system that focuses on video analysis etc. One of the applications which CNNs were recently introduced to is plant disease classification.

Grinblat.et.al [16], 2017 proposed a deep convolutional neural network for the problem of identification of vein morphological patterns. The introduction of a CNN avoids the use of handcrafted feature extractors as it is standard in the state of the art pipeline. This approach was shown to have improved the classification accuracy of the regular pipeline and was able to classify three species of legumes (soybean, white bean and red bean) using CNNs of up to 6 layers.

Mustafa.et.al [17], 2017 proposed a comparative study of the well known CNN architectures AlexNet, GoogleNet and VGGNet to identify the plant species in photographs. Using the plant task datasets of LifeCLEF 2015 augmented by rotation, translation, reflection, and scaling, they applied Transfer Learning to fine-tune these pretrained models. Furthermore, the parameters of the networks were adjusted and different classifiers fused to improve the overall performance.

Recently deep CNNs have been used for the diagnosis of plant diseases. Mohanty.et.al [18], 2016 dealt with the classification of 26 diseases in 14 crop species of 54306 images. Two popular architectures AlexNet and GoogleNet were used with different versions of the images (grayscale, original, background-removed). This trained model was able to achieve an accuracy of 99.35% on a held out test set, demonstrating its feasibility. The segmented versions of the whole dataset was also prepared to investigate the role of the background of the images in overall performance, the performance of the model using segmented images was consistently better than that of the model using gray-scaled images, but slightly lower than that of the model using the colored version of the images.

Sladojevic.et.al [19],2016 is concerned with a new approach to the development of plant disease recognition models, based on leaf image classification, by the use of deep convolutional networks. Novel way of training and the methodology used facilitate a quick and easy system implementation in practice. The developed model is able to recognize 13 different types of plant diseases out of healthy leaves, with the ability to distinguish plant leaves from their surroundings. According to our knowledge, this method for plant disease recognition has been proposed for the first time. .This model was able to achieve an average accuracy of 96.3% on random test sets, with accuracy between 91% and 98%.

Chowdhury Rafeed Rahman.et.al [20], 2018 proposes a system where CNNs have applied for the classification of rice diseases using a dataset of 500 images obtained from the experimental field. This was able to successfully classify 10 types of rice diseases with an accuracy of 95.48%, thus demonstrating the feasibility of this method in the experimental field.

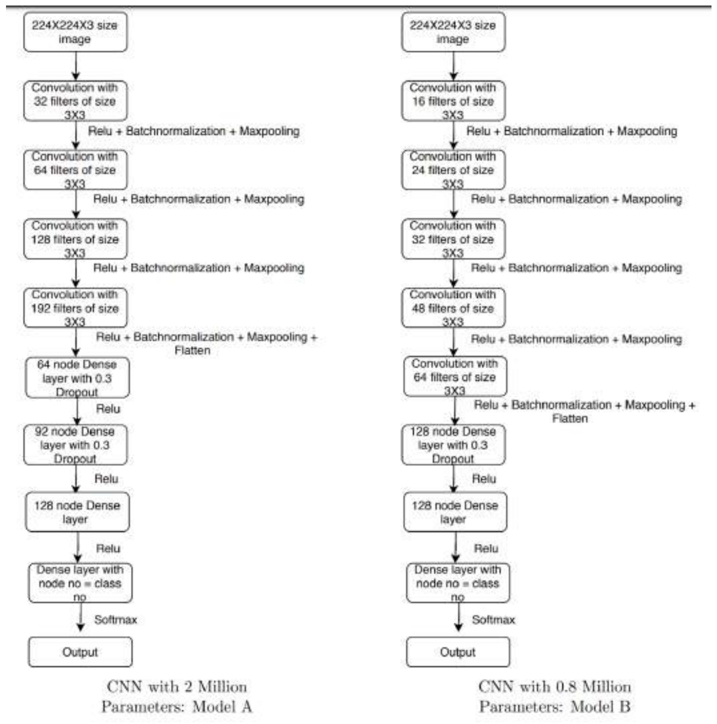

224X224X3 size image
Convolution with 32 filters of size 3X3
Relu + Batchnormalization + Maxpooling
Convolution with 64 filters of size 3X3
Relu + Batchnormalization + Maxpooling
Convolution with 128 filters of size 3X3
Relu + Batchnormalization + Maxpooling
Convolution with 192 filters of size 3X3
Relu + Batchnormalization + Maxpooling + Flatten
64 node Dense layer with 0.3 Dropout
Relu
92 node Dense layer with 0.3 Dropout
Relu
128 node Dense layer
Relu
Dense layer with node no = class no
Softmax
Output
CNN with 2 Million Parameters: Model A
224X224X3 size image
Convolution with 16 filters of size 3X3
Relu + Batchnormalization + Maxpooling
Convolution with 24 filters of size 3X3
Relu + Batchnormalization + Maxpooling
Convolution with 32 filters of size 3X3
Relu + Batchnormalization + Maxpooling
Convolution with 48 filters of size 3X3
Relu + Batchnormalization + Maxpooling
Convolution with 64 filters of size 3X3
Relu + Batchnormalization + Maxpooling + Flatten
128 node Dense layer with 0.3 Dropout
Relu
128 node Dense layer
Relu
Dense layer with node no = class no
Softmax
Output
CNN with 0.8 Million Parameters: Model B

Figure 3.2.5

CNN Architecture Diagram for [21]

CNN models similar to VGG16

Lu.et.al [21] , 2017 improved the work of Chowdhury Rafeed Rahman. It presents deep learning based approaches to detect diseases and pests in rice plants using images captured in real life scenarios. We have experimented with various state-of-the-art CNN architectures on our large dataset of rice diseases and pests collected manually from the field, which contain both inter-class and intra-class variations and have nine classes in total. The results show that we can effectively detect and recognize rice diseases and pests using CNN with the best accuracy of 99.53% on a test set using CNN architecture, VGG16. CNN models similar to VGG16 are represented in Figure 3.2.5.

Brahimi.et.al [22], 2017 a recently published work, has proposed a method where the approach of transfer learning has been used to identify diseases in tomato plants. This method uses visualization methods to identify and localize diseased regions of the leaf and has been able to achieve a test accuracy of 99.18%. Thus it was proved that this could be used as a practical tool for farmers against tomato plants.

3.3 Conclusion

We see that the initial approaches to leaf disease classification included hand engineered features fed to a learning algorithm. The drawback of these methods is that the results heavily depend on the features which have been used. Feature engineering itself is a complex and tedious process involving which requires re-visitation if there are any considerable changes with respect to the problem at hand or the dataset used. It has been proven that deep neural networks are more capable of capturing abstract concepts invariant to phenomena in the visual world.

We also conclude that Convolutional Neural Networks, a subset of DNNs proves to be effective especially in the image domain, and has been successfully applied to many disease classification problems. It is also able to achieve a higher accuracy and does not have a dependency on feature engineering.

Chapter 4

Project Management Plan

4.1 Schedule of the Project

4.1.1 Gantt Chart

A Gantt Chart is a timeline that is used as a project management tool to illustrate how the project will run. It shows individual tasks, their durations and the sequencing of these tasks. The overall timeline of the project and the expected completion date for our project, which is Building Utility System to Analyse Diseased Plants and Predict the Stages using Machine Learning Algorithms is represented in Figure 4.1.1.

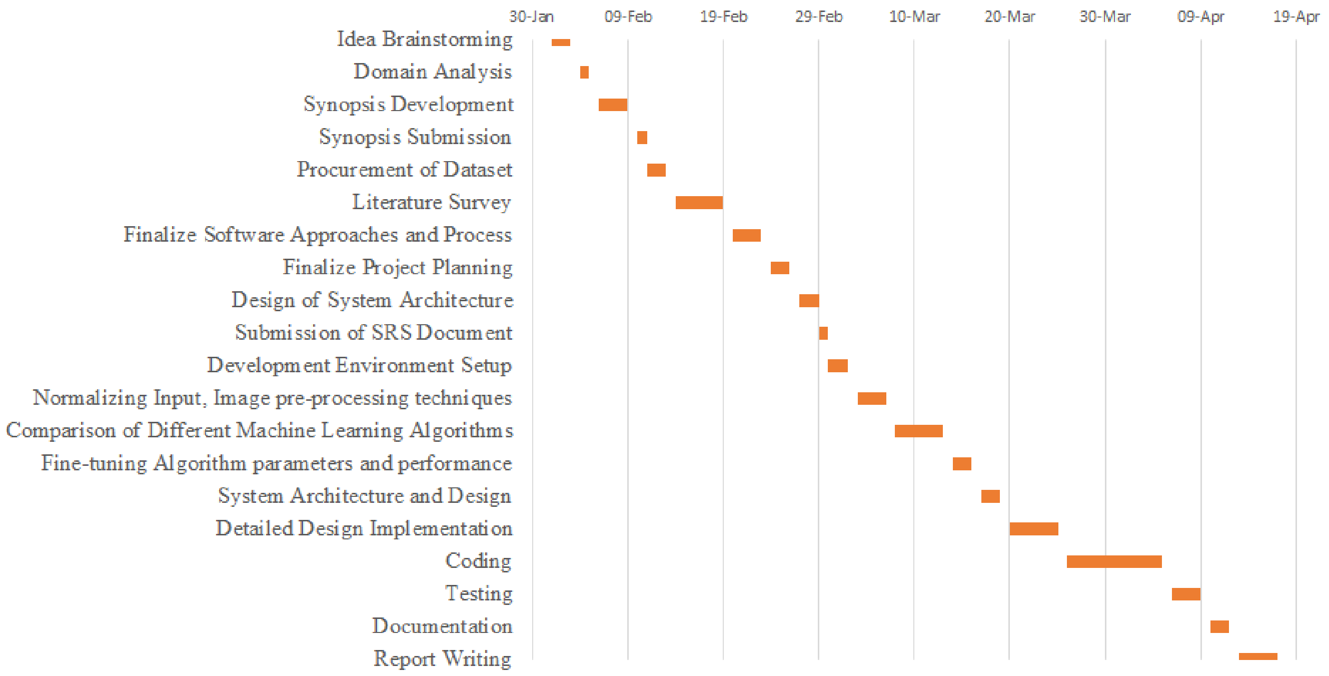


Figure 4.1.1 Gantt Chart for our Project

4.2 Risk Identification

4.2.1 Image Resolution

The resolution of the image is necessary for proper analysis and classification of the disease. A decrease in image resolution and clarity reduces the chances of a correct classification.

Mitigation Steps:

Ensure the image taken is clear from a camera with good resolution. Perform image enhancement techniques to ensure that a lower quality image can also be correctly identified.

4.2.2 Misclassification of Image

The leaf can be misclassified. A wrong classification could have disastrous consequences. It may discourage users from trusting the application and thus reduce its viability

Mitigation Steps:

Maintenance of the database is important. Accuracy can also be improved by retraining the required model

4.2.3 Incorrect Input

The user may input the wrong image to the application. This image forms noise for the data and should not be classified under any category.

Mitigation Steps:

The application must be able to detect if an image is not a leaf image and must be able to discard this data accordingly.

4.2.4 Complexity

Leaf disease is a combination of various factors and thus identifying the required disease simply from the image becomes a difficult task. We must ensure that diseased plants are not detected healthy.

Mitigation Steps:

A larger training set can ensure greater accuracy and increased feature learning. Expert guidance may also aid in improvement of accuracy.

Risk	Probability	Impact	Experience	Mitigation Plan
Image Resolution	High	High	High	1. Ensure high image clarity and superior resolution 2. Image Enhancing Techniques
Misclassification of Images	Low	Medium	Medium	1. Database enhancement 2. Retraining
Wrong Input	Medium	Medium	Medium	1. Removal of data as noise
Complexity	Medium	Medium	Medium	1. Large amount of training data 2. Expert guidance

Table 4.2: Risk identification and Mitigation step

Chapter 5

Software Requirements and Specifications

5.1 Purpose

In this Project, we aim to demonstrate the accuracy of various CNN models and showcase the feasibility of our approach by using a public dataset for healthy and infected Tomato, Potato and Bell pepper Leaves and we wish to classify and identify types of Leaf diseases. Through this, The plant disease detection can be done by observing the spot on the leaves of the affected plant.

5.2 Project Scope

The method that we are adopting to detect plant diseases is image processing using Convolution neural network (CNN). The advantages of the approach that we are using is that the existing method for plants disease detection is simply naked eye observation which requires more man labor, properly equipped laboratories, expensive devices etc can be reduced to a large extent. Another advantage is that our approach facilitates the avoidance of improper disease detection which may lead to inexperienced pesticide usage that can cause development of long-term resistance of the pathogens, reducing the ability of the crop to fight back. The project is developed keeping in mind the benefits that it would provide to the farmers and agricultural sector .The developed system can detect disease. By proper knowledge of the disease and the remedy can be taken for improving the health of the plants.

5.3 Overall Description

5.3.1 Product Perspective

The primary occupation in India is agriculture. India ranks second in the agricultural output worldwide. Here in India, farmers cultivate a great diversity of crops. Various factors such as climatic conditions, soil conditions, various disease, etc affect the production of the crops. In this project, The identified approach is to perform a comparative analysis of the performance between three kinds of Convolutional Neural Networks which are DenseNet-169, vgg-16 and modified CNN. These models are used to identify and classify plant leaf disease detection. Some of these diseases are early blight, late blight, bacterial spot, mosaic virus etc. The accuracy obtained from the above models is visualized through a web application and the model with the best accuracy is obtained. After the disease is successfully predicted with a good accuracy, the farmers then are able to analyze the problems of the infected leaves based on which the remedies can be decided.

5.3.2 Product Features

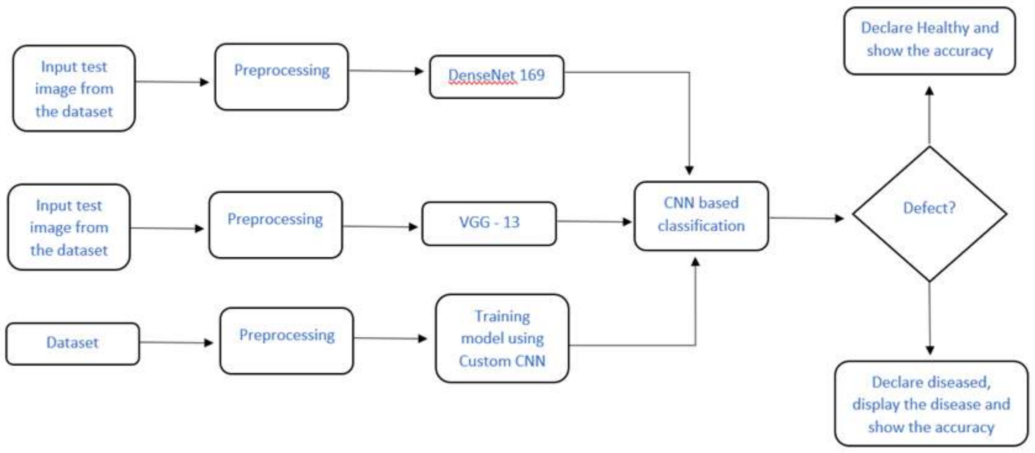

Input test image from the dataset
Preprocessing
DenseNet 169
Declare Healthy and show the accuracy
Input test image from the dataset
Preprocessing
VGG - 13
CNN based classification
Defect?
Dataset
Preprocessing
Training model using Custom CNN
Declare diseased, display the disease and show the accuracy

Figure 5.3.1 Product Features

The database is Pre-processed using techniques such as image reshaping, resizing and conversion to an array form. Similar processing is also done on the test image. A database consisting of many different plant species is obtained, out of which any image can be used as a test image for the software. The selected database is properly segregated and pre-processed and then renamed into proper folders. The train database is used to train the model (CNN) so that it can identify the test image and the disease it has .CNN has different layers that are Dense, Dropout, Activation, Flatten, Convolution, MaxPooling. After successful training and pre-processing ,comparison of the test image and trained model takes place to predict the disease. The accuracies of each CNN model i.e. DenseNet, VGG and modified CNN are compared and the one with the best accuracy is identified for further usage.

5.3.3 Operating Environment

The project includes a web based visualization and comparative analysis of the accuracies. The minimum hardware requirements for this project are CPU quad-core or hexa-core for the processor, 40GB HDD free space for the hard disk and 8GB for RAM .The software requirements include Operating system - Windows 7, 8, 10, Server 2008, Server 2012, 64 bits. The Framework used for development includes Python and jupyter notebook.

5.4 External Interface Requirements

5.4.1 User interfaces

The Graphical user interface consists of a web application. A user can upload the image of the diseased plant on the website. The GUI shows the resultant disease predicted by the various models. Models can be selected specifically as well. The GUI shows the input image, classification results and accuracy. The user can also view results of the saliency experiment (estimate the importance of each pixel, using only one forward and one backward pass through the network) on the web application. This is a cross platform application and can thus be used on the mobile as well.

5.4.2 Hardware Interfaces

The hardware requirement for the utility system development has been identified as shown in table below:-

Component	**Sub-component**	**Minimum requirement**
Laptop/ computer	Processor	Quad core Intel Core i7 Skylake or hexa-core
	Hard Disk	40 GB HDD Free Space
	Graphic Card	Nvidia Geforce GTX 1080
	GPU	Not essential, but preferable for training
	RAM	8GB
Mobile	IP Webcam	

Table 5.4.1 Hardware Interfaces

5.4.3 Software Interfaces

The software requirement for the utility system development has been identified as shown in table below:-

	Software	**Purpose**
Operating System	Windows 7, 8, 10, Server 2008, Server 2012, 64 bits	
Software	Microsoft Excel	Gantt Chart
	Python (Jupyter Notebook)	Development Tools
	StarUML	Data flow diagrams

Table 5.4.2 - Software Interfaces

5.4.4 Communication Interfaces

The web application is created using Flask and it connects the user to the application. IP Webcam is used to capture the required images.

5.5 System Features

5.5.1 Functional Requirements

"Utility System to Analyze Diseased Plants and Predict the Stages of Disease using Machine Learning Algorithms" is a fully-featured software system that begins immediately at the input of plant images, followed by analysis and detection of diseased plants by comparing the results of different learning architectures.
In order to successfully and accurately accomplish this detection, it is necessary to satisfy the following functional requirements.

5.5.1.1 Application Portal to Capture Input: the system should allow input of plant images through appropriate user interfaces.

5.5.1.2 Image Pre-Processing Methodology: the system should employ an image pre-processing methodology to improve the image data by suppressing unwanted distortions or enhancement of image features.

5.5.1.3 Detection through modified CNN: the system should employ a modified CNN architecture to detect plant disease and it's stages with appropriate accuracy.

5.5.1.4 Detection through DenseNet: the system should employ a Dense Convolutional Network (DenseNet), which connects each layer to every other layer in a feed-forward fashion, to detect plant disease and it's stages with appropriate accuracy.

5.5.1.5 Detection through VGG16: the system should employ VGG, a convolutional neural network model to detect plant disease and it's stages with appropriate accuracy.

5.5.1.6 Comparison of Result: the system must conduct comparative analysis of the results of all convolutional architectures, and present the user with the most accurate output.

5.5.1.7 Application User Interface to View Result: the system must allow the user to view the output parameters obtained after training and prediction in a suitable user interface.

5.5.2 Non-functional Requirements

5.5.2.1 Performance: the system should be able to perform image processing, detection through training and prediction, and comparison of result with acceptable speed of response, throughput, execution time.

5.5.2.2 Scalability: the system should be able to handle a growing amount of work by using efficient and adaptable algorithms, and have capability to cope and perform well under an increased or expanding workload or scope.

5.5.2.3 Reliability: the system should be able to function under stated conditions for a specified period of time, without any failure, and provide accurate results.

5.5.2.4 Usability: the system should employ a sufficiently effective, efficient and satisfying experience for the user.

5.5.3 Use case description

5.5.3.1 Use Case 1: Input Images in Application Portal

Purpose: This use case specifies how the user must input the sample plant images through our application portal.

Actors(s): User

Pre-Conditions: The user must have a set of plant images available, in a readable format (*.jpeg, *.png, etc.)

Basic Flow:

B.1 The user navigates to our application.

B.2 The user selects the 'upload' button to upload images.

B.3 The user selects the images they would like to upload as the plant image dataset, and selects 'ok'.

Post Conditions:

The plant images have been successfully uploaded by the user.

Functional Requirements:

The system should have a user interface for uploading images.

The system should be able to capture the uploaded images and transfer them to the processing pipeline.

5.5.3.2 Use Case 2: Detection through modified CNN

Purpose: This use case details the detection of plant disease and it's stages for the input plant dataset provided by the user, through modified CNN.

Actor(s): User

Pre-Conditions: The user must have uploaded their plant image dataset through the application portal.

Basic Flow:

B.1. The user selects the learning architecture ‘modified CNN’ from the options provided in the application portal.

B.2 The input dataset is appropriately pre-processed to reduce distortion and enhance image features.

B.3 The processed data is fed to our modified CNN model, which predicts the presence of plant disease and it’s stages.

B.4 The result of modified CNN classification is displayed to the user, through the application portal interface.

Post Conditions:

The input plant dataset has been successfully classified through the use of modified CNN.

Functional Requirements:

The system should employ an image pre-processing methodology to improve the image data by suppressing unwanted distortions or enhancement of image features.

The system should employ a modified CNN architecture to detect plant disease and it’s stages with appropriate accuracy.

5.5.3.3 Use Case 3: Detection through DenseNet (Dense Convolutional Network)

Purpose: This use case details the detection of plant disease and it’s stages for the input plant dataset provided by the user, through DenseNet.

Actor(s): User

Pre-Conditions: The user must have uploaded their plant image dataset through the application portal.

Basic Flow:

B.1. The user selects the learning architecture ‘DenseNet’ from the options provided in the application portal.

B.2 The input dataset is appropriately pre-processed to reduce distortion and enhance image features.

B.3 The processed data is fed to the DenseNet model, which predicts the presence of plant disease and it’s stages.

B.4 The result of DenseNet classification is displayed to the user, through the application portal interface.

Post Conditions:

The input plant dataset has been successfully classified through the use of DenseNet.

Functional Requirements:

The system should employ an image pre-processing methodology to improve the image data by suppressing unwanted distortions or enhancement of image features.

The system should employ DenseNet architecture to detect plant disease and it's stages with appropriate accuracy.

5.5.3.3 Use Case 4: Detection through VGG16

Purpose: This use case details the detection of plant disease and it's stages for the input plant dataset provided by the user, through VGG16.

Actor(s): User

Pre-Conditions: The user must have uploaded their plant image dataset through the application portal.

Basic Flow:

B.1. The user selects the learning architecture 'VGG' from the options provided in the application portal.

B.2 The input dataset is appropriately pre-processed to reduce distortion and enhance image features.

B.3 The processed data is fed to the VGG model, which predicts the presence of plant disease and it's stages.

B.4 The result of VGG classification is displayed to the user, through the application portal interface.

Post Conditions:

The input plant dataset has been successfully classified through the use of VGG.

Functional Requirements:

The system should employ an image pre-processing methodology to improve the image data by suppressing unwanted distortions or enhancement of image features.

The system should employ VGG architecture to detect plant disease and it's stages with appropriate accuracy.

5.5.3.3 Use Case 5: Comparative Analysis of Plant Disease Detection through different Learning Architectures

Purpose: This use case details the detection of plant disease and it's stages for the input plant dataset provided by the user, through a comparative analysis of the result of processing with modified CNN, DenseNet, and VGG models.

Actor(s): User

Pre-Conditions: The user must have uploaded their plant image dataset through the application portal.

Basic Flow:

B.1. The user selects the 'Comparative Analysis' feature provided in the application portal.

B.2 The input dataset is appropriately pre-processed to reduce distortion and enhance image features.

B.3 The processed data is fed to the modified CNN, DenseNet and VGG model, which predict the presence of plant disease and it's stages.

B.4 The result of modified CNN, DenseNet and VGG classification is displayed side-by-side, to allow the comparison of performance of all three models. These results are displayed through the application results interface.

Post Conditions:

The input plant dataset has been successfully classified through the use of modified CNN, DenseNet and VGG.

Functional Requirements:

The system should employ an image pre-processing methodology to improve the image data by suppressing unwanted distortions or enhancement of image features.

The system should employ modified CNN, DenseNet and VGG architectures to detect plant disease and it's stages with appropriate accuracy.

5.5.3.3 Use Case 6: View Saliency Maps

Purpose: This use case allows the user to view saliency maps to monitor the output of the classifier, and understand if the model is truly identifying image features, or just using the surrounding context.

Actor(s): User

Pre-Conditions:

The user must have uploaded their plant image dataset through the application portal.

The system must have successfully completed classification through the use of modified CNN, DenseNet, or VGG architectures.

Basic Flow:

B.1. The user selects the option to view saliency maps, through the application's results user interface.

B.2 The relevant saliency maps for the employed models are displayed to the user through the application.

Post Conditions:

The saliency maps produced as a result of classification have been successfully produced and displayed to the user through the application results portal.

Functional Requirements:

The system should employ modified CNN, DenseNet and VGG architectures to detect plant disease and it's stages with appropriate accuracy.

The system should be able to produce relevant saliency maps through the results of classification.

5.5.3.4 Use Case 7: View Remedy Details

Purpose: This use case allows the user to view the remedies for the identified disease by redirecting them to google search engine.

Actor(s): User

Pre-Conditions: The system must have successfully completed classification through the use of modified CNN, DenseNet, or VGG architectures, and must have identified a disease in the leaf.

Basic Flow:

B.1. The user selects the option to view Remedy, through the application's remedy button provided below the results.

B.2 The relevant Remedies for the identified disease are displayed to the user through google search engine.

Post Conditions:

The remedy produced as a result of google search based been successfully produced and displayed to the user through the google search engine.

Functional Requirements:

The system should employ modified CNN, DenseNet and VGG architectures to detect plant disease and it's stages with appropriate accuracy.

The system should be able to redirect user to the results of google search engine based on the identified engine.

5.5.4 Use Case Diagram and Activity Diagram

A use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behaviour, and not the exact method of making it happen. Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modelling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behaviour in the user's terms by specifying all externally visible system behaviour. The use case Diagram for our Utility system is depicted in Figure 5.5.1.

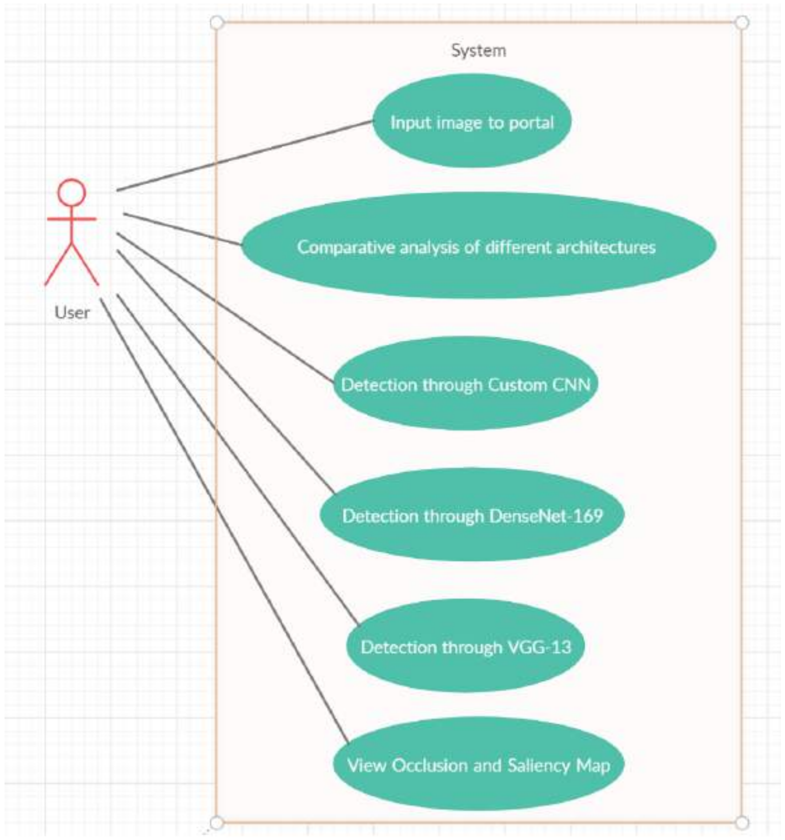

System
Input image to portal
Comparative analysis of different architectures
Detection through Custom CNN
Detection through DenseNet-169
Detection through VGG-13
View Occlusion and Saliency Map
User

Figure 5.5.1: Use Case Diagram

Activity diagram is another important behavioural diagram to describe dynamic aspects of the system. Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination. The activity diagram for our utility system is depicted in Figure 5.5.2.

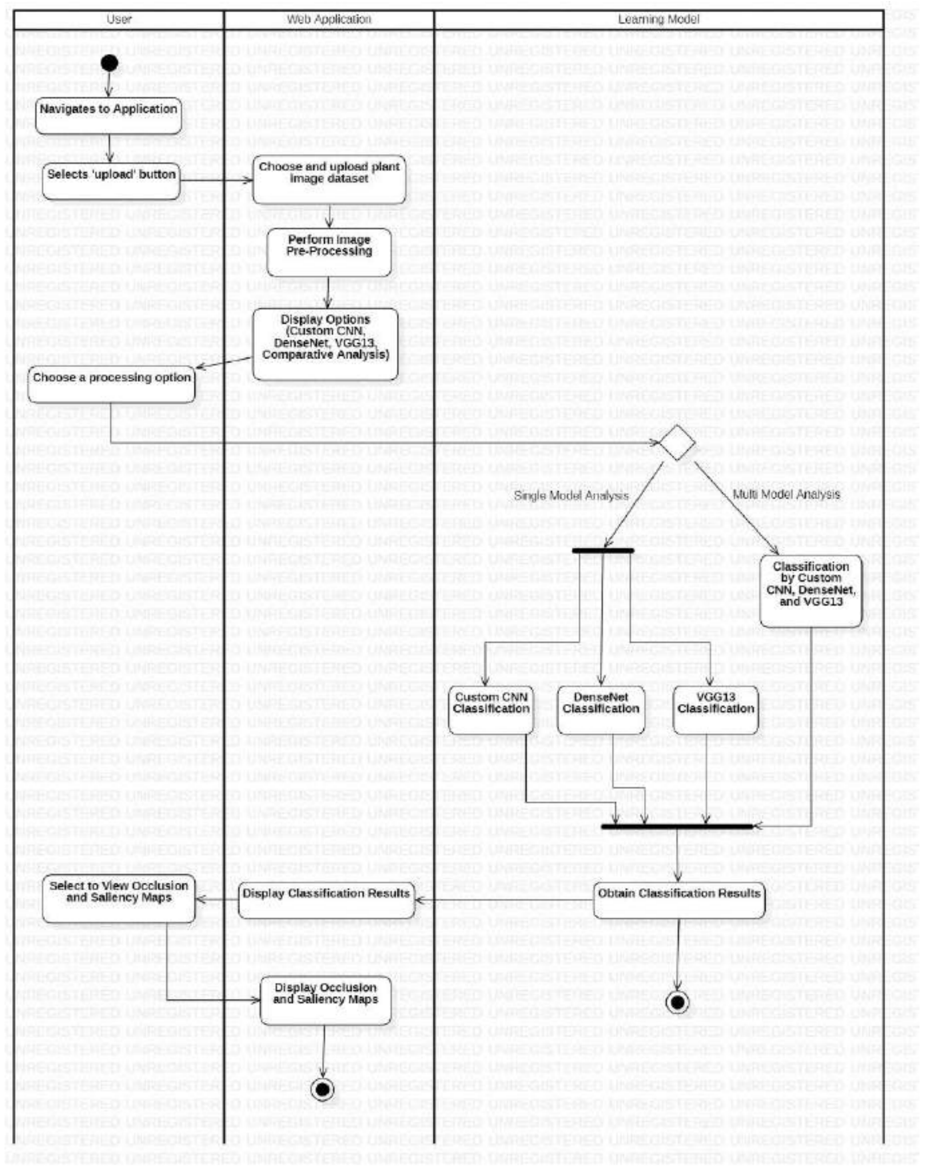

User
Web Application
Learning Model
Navigates to Application
Selects 'upload' button
Choose and upload plant image dataset
Perform Image Pre-Processing
Display Options (Custom CNN, DenseNet, VGG13, Comparative Analysis)
Choose a processing option
Single Model Analysis
Multi Model Analysis
Classification by Custom CNN, DenseNet, and VGG13
Custom CNN Classification
DenseNet Classification
VGG13 Classification
Select to View Occlusion and Saliency Maps
Display Classification Results
Obtain Classification Results
Display Occlusion and Saliency Maps

Figure 5.5.2: Activity Diagram

Chapter 6

Design

6.1 Introduction

The image is captured through a digital camera. This image is uploaded to the web application which performs the required pre-processing and is fed to the Convolutional Neural Networks as specified by the user. The resultant classification is displayed with a certain level of confidence. Since a neural network is generally viewed as a black box with a lack of interpretability, we use saliency experiments. Saliency map is an analytical method that allows to estimate the importance of each pixel, using only one forward and one backward pass through the network. Thus, we display all this information to the user through the web application.

6.2 Architecture Design

An Architecture Diagram is a representation of the entire workflow and steps of a project in the form of flow charts or graphs. It gives an overview of the entire project in a single Diagram which includes the project's principles, elements and components. For our project that is Building Utility System to Analyse Diseased Plants and Predict the Stages using Machine Learning Algorithms, the Architecture diagram is depicted in the Figure 6.2.1.

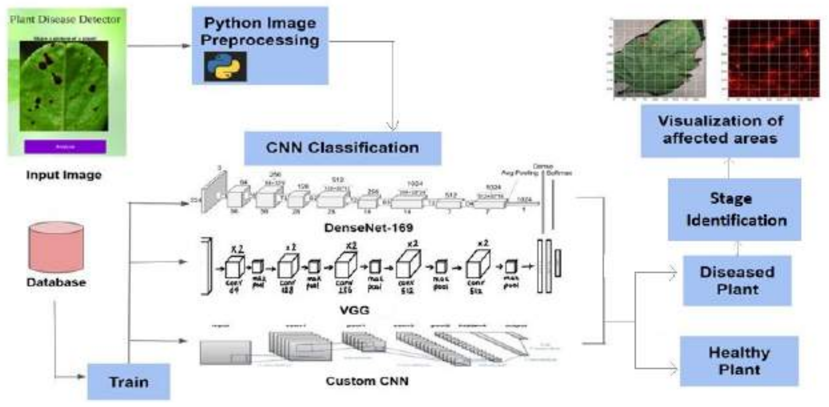
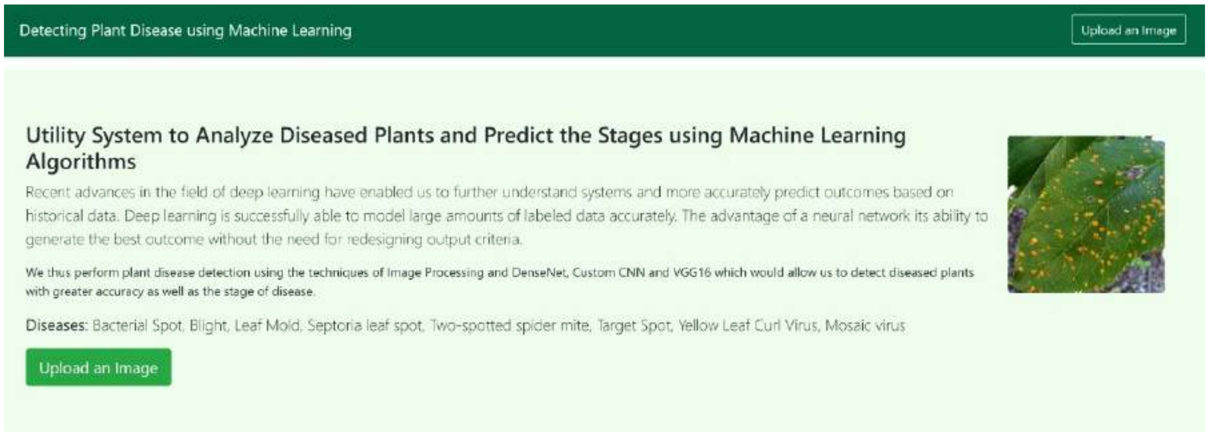


Figure 6.2.1 The diagram above represents the architectural design for the proposed Utility System to detect plant leaf diseases using convolutional neural networks. The image is the input which is then processed and passed through the network to generate the required prediction. The system also offers visualization of the results in the form of a saliency map indicating the affected areas of the plant. 3 different CNN architectures have been used to generate results which can be compared to ensure further surety of the results obtained.

6.3 User Interface Design

User Interface Design refers to the design of the user interface for software, websites or applications. Through this project we are providing an interactive platform in the form of a website where the user can upload the image of the diseased plant in any of the formats specified. The user is then given an option to choose the CNN model which they would want to run on the plant leaf. The CNN model which we developed would then perform the detection and classify the plant leaf. Similarly, any of the models can be chosen and the accuracies can be compared to obtain the most efficient results. These results can then be facilitated to the stakeholders like the farmers and the needful actions could be taken. The screenshots of the Graphical user interface is shown in the Figures 6.3.1, 6.3.2 and 6.3.3.



Utility System to Analyze Diseased Plants and Predict the Stages using Machine Learning Algorithms

Recent advances in the field of deep learning have enabled us to further understand systems and more accurately predict outcomes based on historical data. Deep learning is successfully able to model large amounts of labeled data accurately. The advantage of a neural network its ability to generate the best outcome without the need for redesigning output criteria.

We thus perform plant disease detection using the techniques of Image Processing and DenseNet, Custom CNN and VGG16 which would allow us to detect diseased plants with greater accuracy as well as the stage of disease.

Diseases: Bacterial Spot, Blight, Leaf Mold, Septoria leaf spot, Two-spotted spider mite, Target Spot, Yellow Leaf Curl Virus, Mosaic virus

Figure 6.3.1 The above figure illustrates the homepage of our WebApp for Utility System for Detecting Plant Disease and it's Stages using Convolutional Neural Networks. It was important to us to maintain a user-friendly, and intuitive interface. To accomplish this, we used a front-end HTML5 framework called Bootstrap.

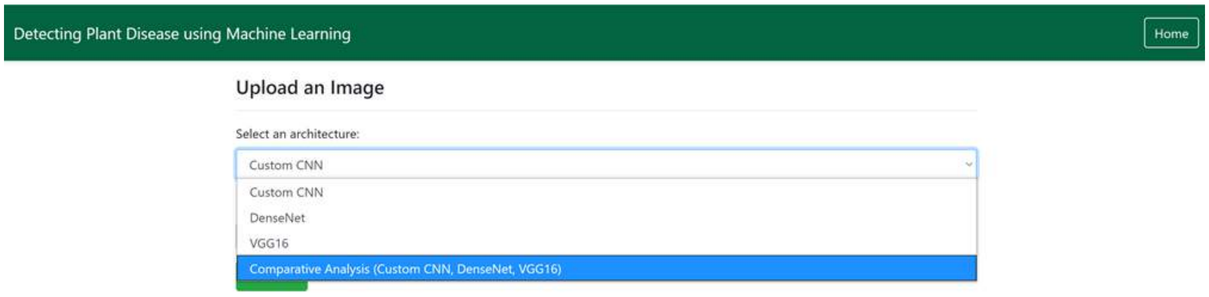
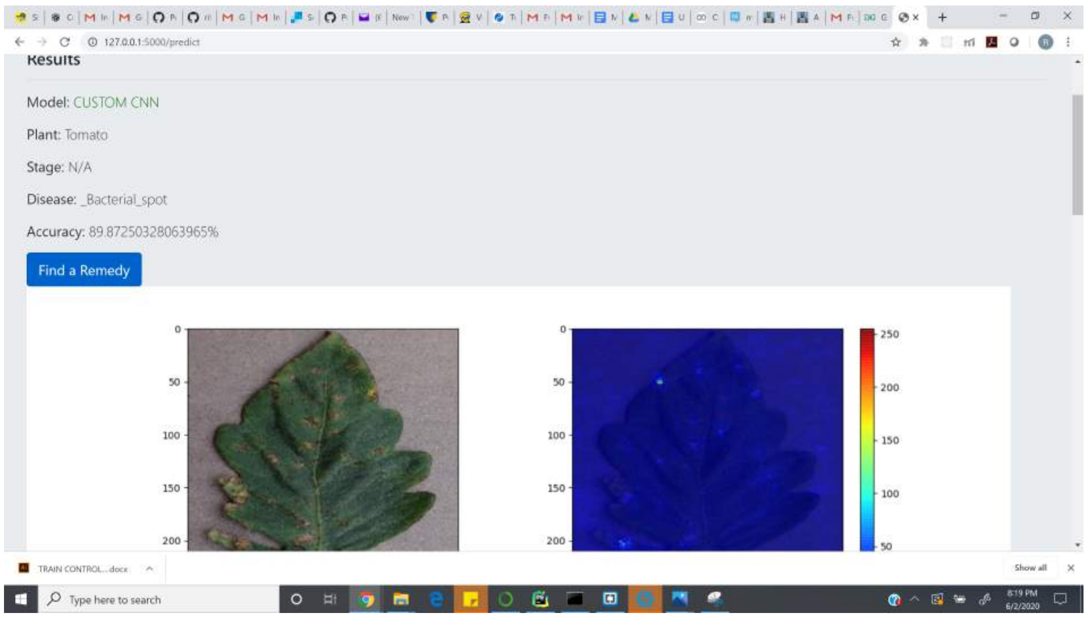

Detecting Plant Disease using Machine Learning
Home
Upload an Image
Select an architecture:
Custom CNN
Custom CNN
DenseNet
VGG16
Comparative Analysis (Custom CNN, DenseNet, VGG16)

Figure 6.3.2 The above figure illustrates our WebApp allowing the user to select the architecture with which they'd like to perform classification. The options are as follows: Custom CNN, DenseNet, VGG-16, and a Comparative Analysis option to compare the results of all three architectures.



Results

Model: CUSTOM CNN

Plant: Tomato

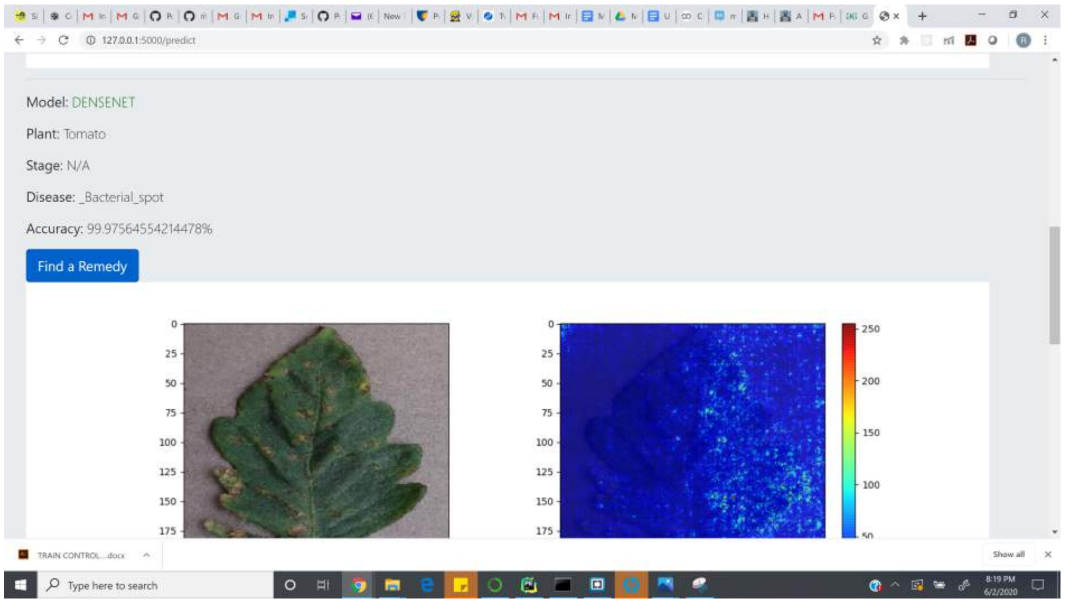
Stage: N/A

Disease: _Bacterial_spot

Accuracy: 89.87250328063965%

Figure 6.3.3 This diagram represents the results which can be viewed on our application when the user chooses to obtain the results generated by the modified CNN architecture. The application also provides visualization of the CNN results in the form of a saliency map, indicating regions which were important during prediction. The user can also view the accuracy of the generated result, along with the stage of the disease and an option to obtain

remedies. Thus the results obtained are easy to understand and visualization could be used for further analysis.


Model: DENSENET

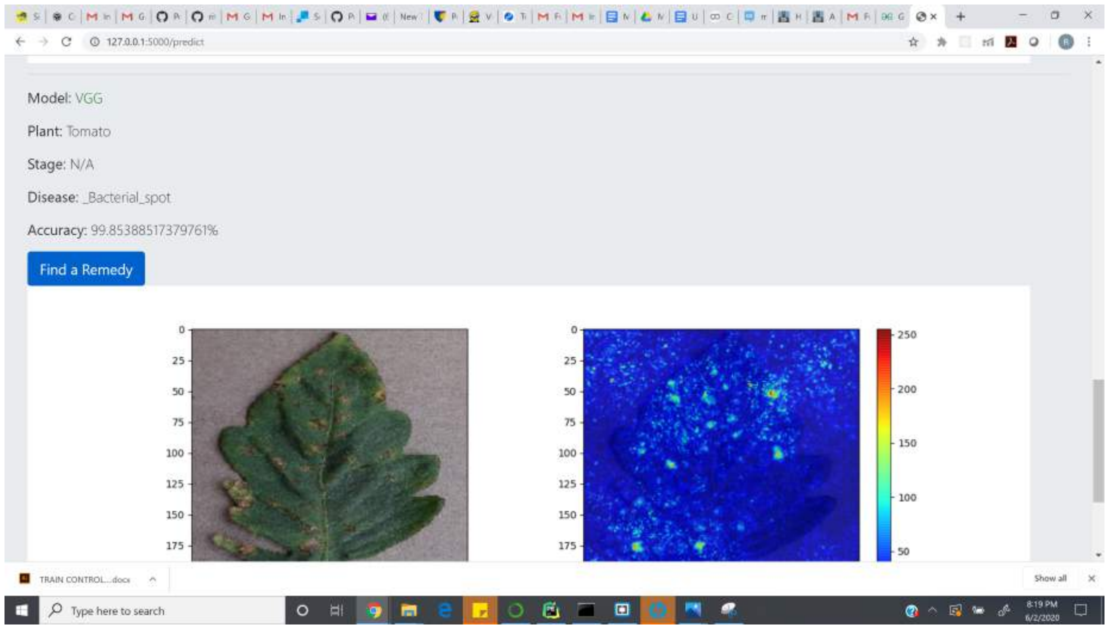
Plant: Tomato

Stage: N/A

Disease: _Bacterial_spot

Accuracy: 99.97564554214478%

Figure 6.3.4 This diagram represents the results which can be viewed on our application when the user chooses to obtain the results generated by DenseNet architecture. The application also provides visualization of the DenseNet results in the form of a saliency map, indicating regions which were important during prediction. The user can also view the accuracy of the generated result, along with the stage of the disease and an option to obtain remedies. Thus the results obtained are easy to understand and visualization could be used for further analysis.


Model: VGG

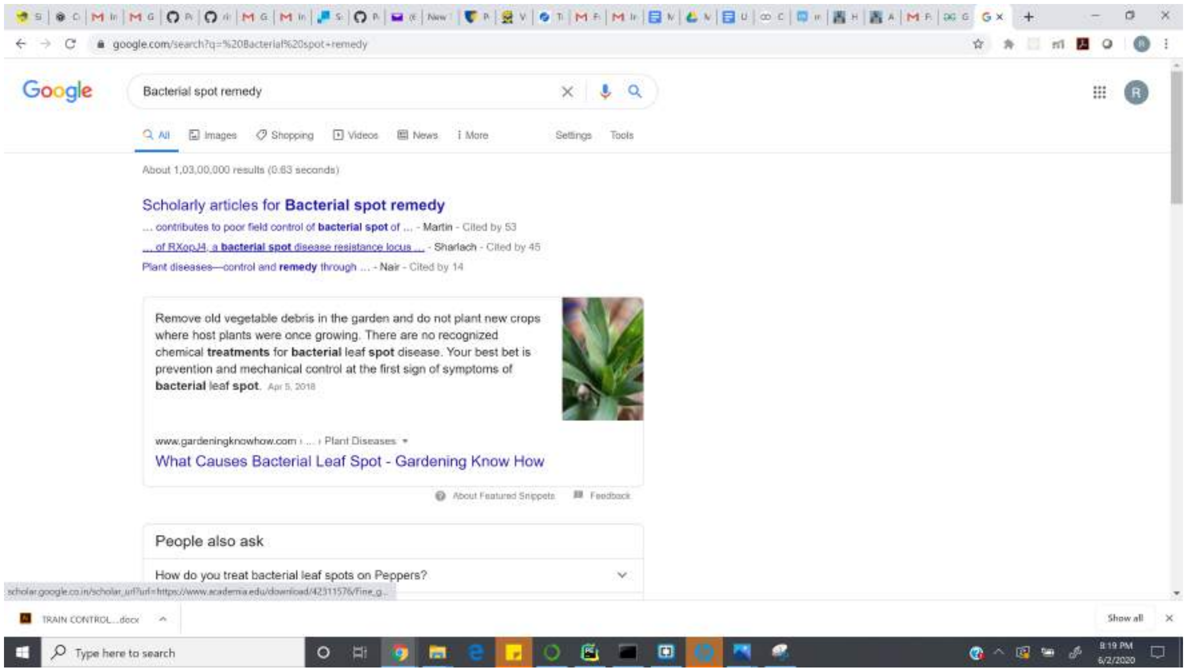
Plant: Tomato

Stage: N/A

Disease: _Bacterial_spot

Accuracy: 99.85388517379761%

Figure 6.3.5 This diagram represents the results which can be viewed on our application when the user chooses to obtain the results generated by the VGG-16 architecture. The application also provides visualization of the VGG-16 results in the form of a saliency map, indicating regions which were important during prediction. The user can also view the accuracy of the generated result, along with the stage of the disease and an option to obtain remedies. Thus the results obtained are easy to understand and visualization could be used for further analysis.



google.com/search?q=%20Bacterial%20spot+remedy

Bacterial spot remedy

About 1,03,00,000 results (0.63 seconds)

Scholarly articles for **Bacterial spot remedy**

... contributes to poor field control of **bacterial spot** of ... - Martin - Cited by 53

... of RXopJ4, a **bacterial spot** disease resistance locus ... - Sharlach - Cited by 45

Plant diseases—control and **remedy** through ... - Nair - Cited by 14

Remove old vegetable debris in the garden and do not plant new crops where host plants were once growing. There are no recognized chemical **treatments** for **bacterial** leaf **spot** disease. Your best bet is prevention and mechanical control at the first sign of symptoms of **bacterial** leaf **spot**. Apr 5, 2018

www.gardeningknowhow.com › ... › Plant Diseases

What Causes Bacterial Leaf Spot - Gardening Know How

People also ask

How do you treat bacterial leaf spots on Peppers?

scholar.google.co.in/scholar_url?url=https://www.academia.edu/download/42311576/Fine_g...

Figure 6.3.6 The above figure illustrates the Google search results for the remedy of the

disease classified. Our WebApp makes it easy to access this remedy information through our ‘Find a Remedy’ button in classification results, which directs the user to the appropriate Google search webpage.

6.4 Low Level Design Flowchart

Low-level design fills in some of the gaps to provide extra detail that’s necessary before developers can start writing code. It gives more specific guidance for how the parts of the system will work and how they will work together. It refines the definitions of the database, the major classes, and the internal and external interfaces. The low level design flowchart is depicted in Figure 6.4.1.

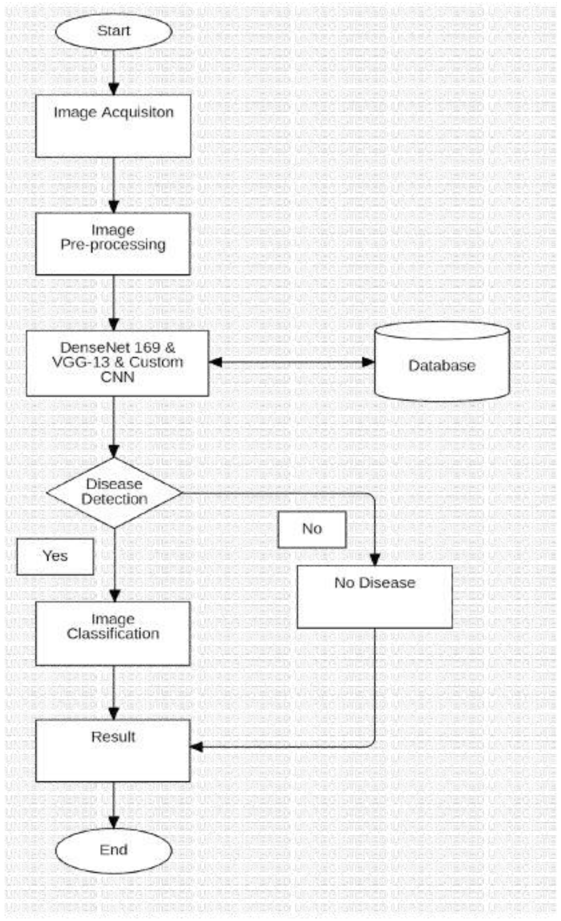


Figure 6.4.1 The above figure illustrates the Low Level Design Flowchart for Utility System for Detecting Plant Disease and it's Stages using Convolutional Neural Networks. Our first step is gathering the data through Image Acquisition, after which we subject it to various pre-processing steps to prepare it for classification. Once our data has been pre-processed, it can be classified by either Custom CNN, DenseNet, VGG-16, or a combination of these. We then present the user with the result of this classification, whether or not the input plant image is diseased or not.

6.5 Conclusion

The aim of this project is to design a system which takes an image as the input from the user and passes to each of the models, classifies it against the model and predicts the class with high accuracy, compares the accuracies of each CNN model and identifies the one with the best accuracy. Also the feel of the GUI must be easy to the user.

Chapter 7

Implementation

7.1 Tools Introduction

7.1.1 Keras

Keras is a high-level neural networks library which runs on top of Tensorflow, CNTK and Theano. It enables easy and fast prototyping and also runs seamlessly on CPU and GPU. This framework is written in Python code which enables easy debugging and ease of extensibility. It has a simple optimized user interface and provides clear feedback for user errors. It is aso modular and composable, making it easy to use for research and deployment. We have used Keras to develop the architecture of our neural networks.

7.1.2 Google Colab

This is a tool similar to Jupyter Notebook which allows us to run the code in the cloud. It comes with Data Science such as Tensorflow, Keras, OpenCV libraries preinstalled. Additionally it offers free GPU/TPU enabling faster network training without increasing load on the user's system. This environment is being utilized to train our models for the detection of diseases in plants.

7.1.3 Pycharm

Pycharm is a popular IDE used for the Python scripting language. It offers features such as code completion, Git visualization, package management etc. It provides support for web programming and frameworks such as Django and Flask. We have thus used Pycharm for the development of our Flask web application.

7.2 Technology Introduction

7.2.1 Python language

Python code is understandable by humans, which makes it easier to build models for machine learning. Since Python is a general-purpose language, it can do a set of complex machine learning tasks and enable you to build prototypes quickly that allow you to test your product for machine learning purposes. It consists of various libraries like NumPy for high-performance scientific computing and data analysis. We use python for the development of our models.

7.2.2 Flask framework

Flask is a micro web framework for Python, meaning that it provides functionality for building web applications, including managing HTTP requests and rendering templates.
In our project, Flask is used for development of our web application.

7.2.3 HTML + CSS + Bootstrap

HTML, CSS and Bootstrap are used to build the front end that is the web application where the user sends a request as uploading the image and receives the response in the form of results**.**

7.3 Overall view of the project in terms of implementation

7.3.1 Image Acquisition:

This is the first step involved in the development of the image classifier. Images of diseased leaves are collected and used to train the classifier. To train and test the system, we use sample images of diseased and healthy leaves. The leaf diseases that are tested include Blight, Bacterial Spot, Leaf Mold, Septoria Leaf Spot, Target Spot, Mosaic Virus, Yellow Leaf Curl Virus and Two-Spotted Spider Mite. The application was executed on Pycharm IDE. The application programming interface was deployed using Flask.

7.3.2 Image Pre-Processing:

Image preprocessing refers to the process of ensuring that the physical geometry of the input to the Deep Convolutional Neural Network is consistent across the dataset. Deep Convolutional neural networks can perform feature extraction and classification automatically, a consistent input needs to be presented to the network to ensure that time is spent in learning the required parameters and not peculiarities in the data. The steps involved in this process include:

- Reading the image: The required RGB image is read from the directory using the OpenCV library. This information is loaded into arrays.

  image = cv2.imread (image_dir)
- Resizing the image: Images present in the dataset vary in size, and thus we establish a base size for the images fed into the algorithm. The Convolutional Neural Network architecture accepts an input size of (256,256); whereas the VGG16 architecture and DenseNet-169 architecture accepts an input size of (224,224).

  image = cv2.resize (image, default_image_size)
- Label Binarization: Each image label was converted into binary levels, which are saved for further processing.

  labelbinarizer = LabelBinarizer()

  img_labels = labelbinarizer.fit_transform (label_ls)
- Image Scaling: The data is further pre-processed by scaling the data points from (0,225) (the minimum and maximum RGB values of the image); thus converting it to the range [0,1].

  np_img_list = np.array (image_ls, dtype = np.float16) / 255.0
- Image Augmentation: An image data generator object is used, which performs random flips, shifts, crops, and sheers on the image dataset. The degree range for random rotations is set to be 25. The shear intensity, i.e, shear angle in counter-clockwise direction, is set to be 0.2 degrees and the range for random zoom is set as 0.2. Width_shift_range and height_shift range are the fraction of total width and total height respectively which are set as 0.1. The input is randomly flipped horizontally. The fill_mode is set to be "nearest", i.e, default value, where points outside the boundaries of the input are filled according to the aaaaaaaa|abcd|dddddddd format.

```
aug = ImageDataGenerator (rotation_range = 25, width_shift_range = 0.1,
height_shift_range = 0.1, shear_range=0.2, zoom_range=0.2, horizontal_flip=True,
fill_mode="nearest" )
```

7.3.3 Model Creation:

The data is split into a training and validation set in the ratio 4:1. This data is then further used for the training of the DenseNet-169, VGG-16 and Convolutional Neural Network model. These models are then used for the prediction of the diseased plant.

7.4 Explanation of Algorithm and how it is been implemented

Convolutional Neural Network

Deep Learning Convolutional Neural Network (CNN) is a neural network model which is used extensively in the domain of image recognition and classification. The funneling technique applied in CNNs ensure faster processing and extraction of features. Three main layers are used to build a CNN architecture:

- Convolutional layer: This is the layer used to extract features from an input image. It takes two inputs; the image matrix and a filter/kernel; and then the convolution of the image matrix is multiplied with the filter matrix produces a feature map used for classification. A given stride is used to calculate the number of pixel shifts over the input image matrix. Padding is added when the filter does not perfectly fit the input image as well. ReLU (Rectified Linear Unit) is added to introduce non-linearity to the CNN;
- Pooling layer: This is used to reduce the number of parameters when the images are large. Thus it reduces dimensionality while retaining the required information. This can be of the types Max Pooling(largest element taken from feature map), Average Pooling (average of elements) and Sum Pooling (sum of all elements).
- Fully Connected Layer: The matrix obtained from the pooling layer is flattened to a vector and fed into a fully connected layer, where the features are combined to create a model. Finally, an activation function is used for the purpose of classification

Thus the algorithm for a CNN is as follows:

- Provide pre-processed input image to convolution layer
- Choose parameters, apply strides and padding if required. Perform convolution on the image and apply ReLU activation.
- Perform pooling to reduce dimensionality size
- Add convolutional layers until the required criteria is met
- Flatten the output and feed it to a fully connected layer
- Output the class using an activation function and classify images

The architectures applied for the purpose of this project are:

1. Convolutional Neural Network architecture:
    - The model is defaulted to the channel_last architecture. The first convolutional layer has 32 channels with a 3x3 kernel size and ReLU activation. Batch Normalization and dropout is applied for the purpose of regularization
    - The second and third convolutional blocks have 64 channels of 3x3 kernel size and ReLU activation. Max pooling is applied with a stride of (2,2). Dropout of 0.25 is applied to prevent overfitting.
    - This is followed by 2 blocks each with 128 channels of 3x3 kernel size and ReLU activation. Max pooling is applied with a pool size of (2,2) followed by a dropout of 0.25.
    - One set of Fully Connected Layers is used followed by ReLU activation. This is then followed by a fully connected layer and softmax activation which allows for the classification for the required diseased plant.
    - The optimizer used here is the Adam optimizer. The model was trained for 25 epochs.

. VGG16:

- The input to the model is an image vector of dimensions of 224x224x3.
- The first two layers have 64 channels with filter size 3x3 and same padding, followed by a max pooling layer of stride (2,2)

- This is followed by two layers with 128 channels and 3x3 kernel size; followed by a max pooling layer of stride (2,2)
- There are three convolutional layers of filter size 3x3 and 256 filters.
- This s followed by a maxpool layer of stride (2,2)
- This is followed by two sets of 3 convolutional layers, each with 512 filters of (3,3), followed by a maxpool layer.
- This vector obtained is then flattened and two Dense layers of 4096 units is added followed by one Dense softmax layer with the number of units equal to the number of classes.
- We use a VGG16 model with pretrained weights (pretrained on Imagenet) where we do not include the fully connected layers and make the last block trainable on the dataset.
- We then add the required classifier layers of 4096 units followed by a Dense softmax layer with the number of units equal to the number of classes.
- A stochastic gradient descent optimizer is used with a learning rate of 0.01 and a decay of 0.005. This model was trained for 25 epochs.

. DenseNet-169:

- A problem with CNNs as they go deeper is that the path to the input to the output layer becomes so large that the gradient vanishes before reaching the other side.
- Densenets do not sum the output feature maps, they concatenate them. Every layer has access to its preceding feature maps, and thus they use collective knowledge.
- Thus Densenets strengthen feature propagation, encourage reuse and reduce the number of parameters.
- We use a Densenet-169 model which is pretrained (Imagenet) where the fully connected layers are not included. We then add a max pooling layer, which is followed by a fully connected layer.
- The optimizer used is Adam optimizer with a learning rate of 10e-5. This model was trained for 10 epochs.

7.5 Information about the implementation of Modules

7.5.1 GUI Module:

The GUI module is the front end of the project. The GUI Module allows users to upload leaf images and obtain result using the other three models where each one is a type of CNN model
(VGG, Densenet, Modified).The GUI takes in the user input and displays the output of each sub models.

7.5.2 VGG -16 Module:

The key feature in this architecture is the definition and repetition of what we will refer to as VGG-blocks. These are groups of convolutional layers that use small filters (e.g. 3×3 pixels) followed by a max pooling layer. The image is passed through a stack of convolutional (conv.)
layers, where we use filters with a very small receptive field: 3 x 3 (which is the smallest size to capture the notion of left/right, up/down, center). Max-pooling is performed over a 2 x 2 pixel window, with stride 2.

7.5.3 Densenet module

DenseNet is a new CNN architecture that reached State-Of-The-Art (SOTA) results on classification datasets (CIFAR, SVHN, ImageNet) using less parameters. Its new use of residual it can be deeper than the usual networks and still be easy to optimize. DenseNet is composed of Dense blocks. In those blocks, the layers are densely connected together: Each layer receives as input all previous layers output feature maps.

7.5.4 Modified CNN module

A CNN model with modified architecture is trained from scratch for over 10000 images. It has
different layers such as Dense, Dropout, Activation, Flatten, Convolution, MaxPooling. The module is loaded in the webapp and the classification is done and the output of the module is sent over to the webapp.

7.6 Conclusion

This application is developed using various tools like Google Colab and PyCharm. Using technologies like python language, flask API, HTML, CSS and Bootstrap we were able to develop the Web Application as well as all the desired CNN models. Once the building was done, we used these models to visualize and identify diseases in plant leaf. The results of CNN models are rendered on to web application in the form of accuracy, necessary details and comparative analysis.

Chapter 8

Testing

8.1 Introduction

Software testing is the process done to check the quality of the product and to verify and validate if the product meets its specified requirements. The methods for testing involves unit testing which tests if the classification of disease and the GUI work fine. After testing all the modules they are added into the GUI. The whole module is then again evaluated on various parameters.
The different phases are split using various system designs like iterative and incremental model. After the implementation, the model is tested and unified. Once the functional and non functional testing is done the User uploads the image is released to the market.

For test environment, key area to set up includes System and applications,Test data ,Front end running environment, Documentation required like bibliography, reference documents/configuration guides/installation guides/ user manuals. The challenges faced during the testing environment includes Proper plannig on image acquisition and resource usage, Configuring test bed etc. Sometimes the testing process can pose certain limitations like remote environment, shared usage by teams and complex test configuration. These issues have to be handled accordingly.

8.2 Test cases

Use Case	Test Case	Description	Procedure	Expected result
UC01 : Input Images in Application Portal	TC01: User uploads the image	The user selects the image and upload it in the portal	The user gives a new input to the models	The user uploads the image successfully
UC02 : Detection through modified CNN	TC01: The user selects the learning architecture 'modified CNN' from the options provided	The user chooses the modified CNN architecture and the classification result along with accuracy is printed.	The user chooses the architecture and the result is printed.	The classification of the input is correct.

UC03: Detection through DenseNet	TC01: The user selects the learning architecture 'DenseNet' from the options provided	The user chooses the densenet architecture and the classification result along with accuracy is printed.	The user chooses the architecture and the result is printed.	The classification of the input is correct.
UC04: Detection through vgg16	TC01: The user selects the learning architecture 'vgg16' from the options provided	The user chooses the vgg16 architecture and the classification result along with accuracy is printed.	The user chooses the architecture and the result is printed.	The classification of the input is correct.
UC05: Comparative Analysis of Plant Disease Detection through different Learning Architectures	TC05: The user selects the 'Comparative Analysis' feature from the options provided	The user selects the comparative analysis feature and the data is fed to the custom cnn, densenet, and vgg13 models, which classify the image and the result is printed.	The user selects the feature and the comparison of performance of all three models is displayed side-by-side.	The classification of all three models is printed.
UC06: View Saliency Maps	TC06: The user selects the option to view saliency maps,	The user selects the option 'view saliency maps' to view saliency	The user selects the option and the relevant maps are	The relevant saliency maps are displayed.

	through the application's results user interface	maps and the relevant saliency maps for the employed models are displayed.	produced through the results of the classification.	
UC07 : Obtain remedy for the identified disease	TC07 : the user selects the remedy button	The user selects the remedy button on the result page to obtain the remedy for the identified disease and eventually use it to treat the disease	The user selects the option to obtain remedy. Through this the user is redirected to Google search Engine displaying the results in the form of remedies for the identified disease	Relevant Remedies based on the disease and its stage is displayed

Chapter 9

Results & performance analysis

9.1 Result Snapshots

9.1.1 Model Development

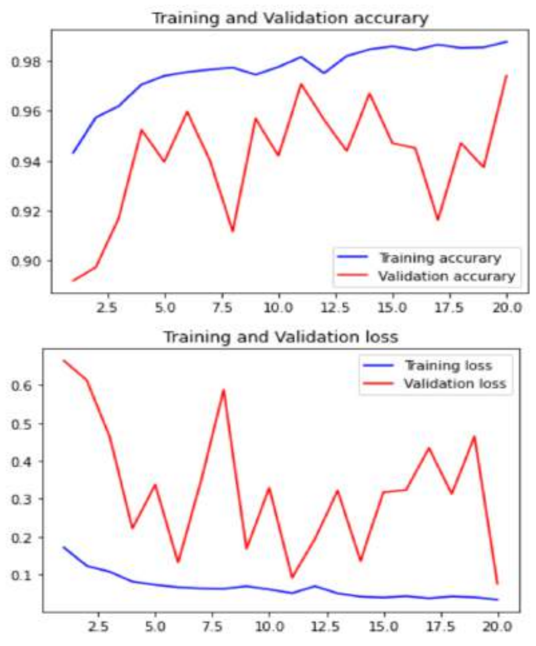


Figure 9.1.1. The diagram above represents the training accuracy and loss of the modified CNN. Loss refers to the objective function which is to be minimized in order to minimize error and accuracy refers to the percentage of correct predictions which are obtained through the model. The plot indicates the required values over the entire training period and indicates the maximum training accuracy obtained as well.

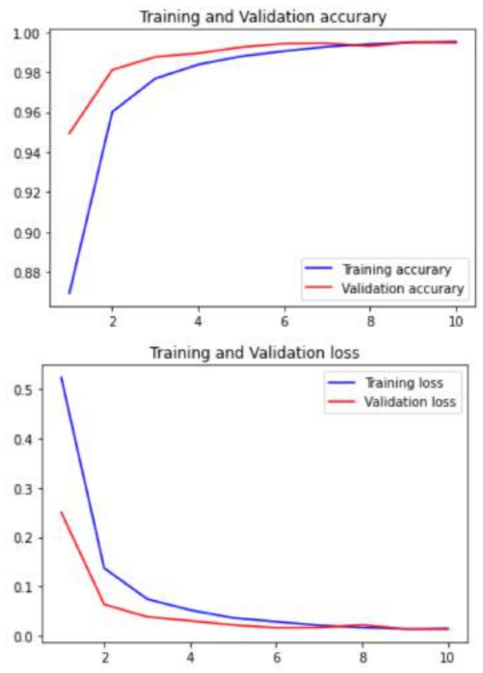
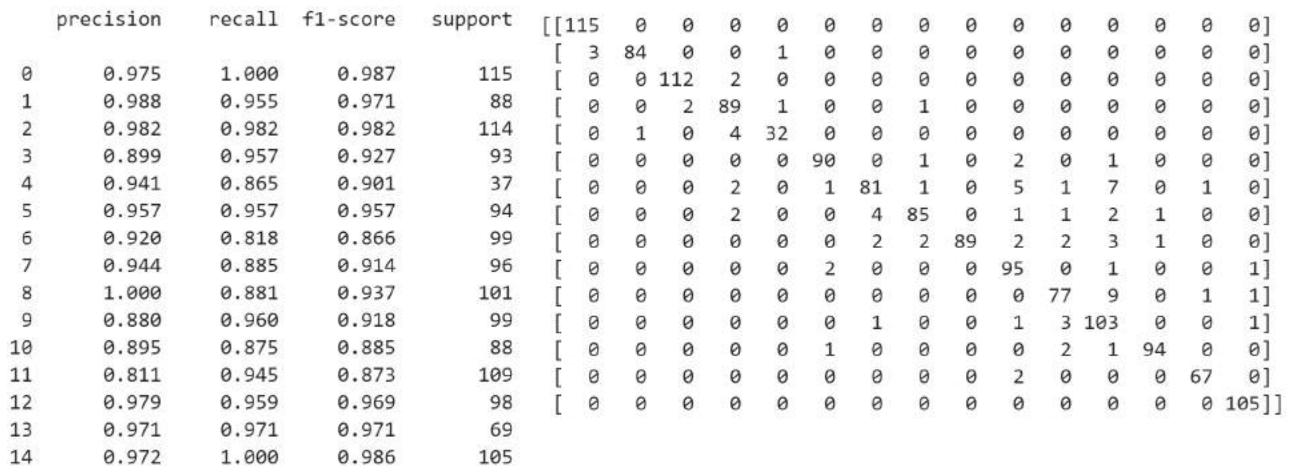


Figure 9.1.1.2 The diagram above represents the training accuracy and loss of DenseNet. Loss refers to the objective function which is to be minimized in order to minimize error and accuracy refers to the percentage of correct predictions which are obtained through the model. The plot indicates the required values over the entire training period and indicates the maximum training accuracy obtained as well.

Figure 9.1.1.3 The diagram above represents the training accuracy and loss of VGG-16. Loss refers to the objective function which is to be minimized in order to minimize error and accuracy refers to the percentage of correct predictions which are obtained through the model. The plot indicates the required values over the entire training period and indicates the maximum training accuracy obtained as well.

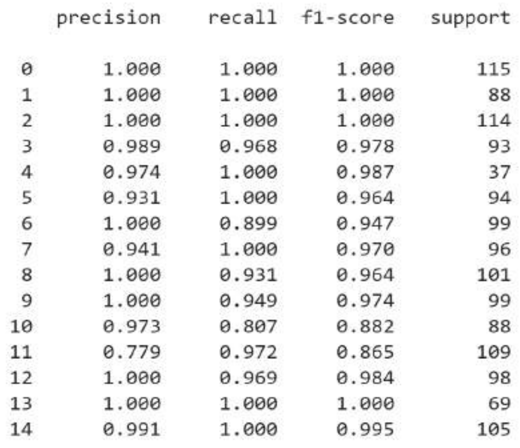
Figure 9.1.1.4 The diagram above represents the different performance measures which have been applied on the modified CNN model and the results obtained. Recall refers to the ratio of correctly predicted positive observations to all observations in the actual class. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. F1 Score refers to the weighted average of Precision and Recall, thus taking both false positive and negatives into account. A confusion matrix is a table that is often used to describe the performance of the classifier on a set of test data for which the true values are known. The required results are displayed and provide us with a better insight as to the accuracy of prediction, the number of false positive and false negatives and so on.



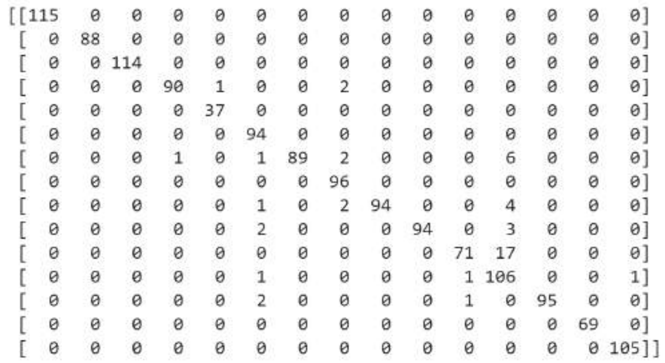
```
    precision    recall  f1-score   support

0       0.975     1.000     0.987       115
1       0.988     0.955     0.971        88
2       0.982     0.982     0.982       114
3       0.899     0.957     0.927        93
4       0.941     0.865     0.901        37
5       0.957     0.957     0.957        94
6       0.920     0.818     0.866        99
7       0.944     0.885     0.914        96
8       1.000     0.881     0.937       101
9       0.880     0.960     0.918        99
10      0.895     0.875     0.885        88
11      0.811     0.945     0.873       109
12      0.979     0.959     0.969        98
13      0.971     0.971     0.971        69
14      0.972     1.000     0.986       105
```

```
[[115   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  3  84   0   0   1   0   0   0   0   0   0   0   0   0   0]
 [  0   0 112   2   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   2  89   1   0   0   1   0   0   0   0   0   0   0]
 [  0   1   0   4  32   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0  90   0   1   0   2   0   1   0   0   0]
 [  0   0   0   2   0   1  81   1   0   5   1   7   0   1   0]
 [  0   0   0   2   0   0   4  85   0   1   1   2   1   0   0]
 [  0   0   0   0   0   0   2   2  89   2   2   3   1   0   0]
 [  0   0   0   0   0   2   0   0   0  95   0   1   0   0   1]
 [  0   0   0   0   0   0   0   0   0   0  77   9   0   1   1]
 [  0   0   0   0   0   0   1   0   0   1   3 103   0   0   1]
 [  0   0   0   0   0   1   0   0   0   0   2   1  94   0   0]
 [  0   0   0   0   0   0   0   0   0   2   0   0   0  67   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0 105]]
```

Figure 9.1.1.5 The diagram above represents the different performance measures which have been applied on VGG-16 model and the results obtained. Recall refers to the ratio of correctly predicted positive observations to all observations in the actual class.Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. F1 Score refers to the weighted average of Precision and Recall, thus taking both false positive and negatives into account. A confusion matrix is a table that is often used to describe the performance of the classifier on a set of test data for which the true values are known. The required results are displayed and provide us with a better insight as to the accuracy of prediction, the number of false positive and false negatives and so on.



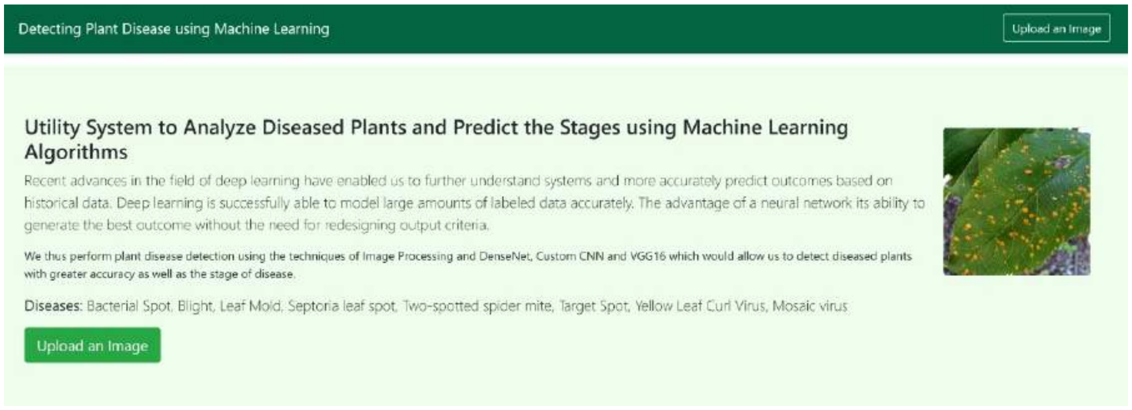
```
    precision    recall  f1-score   support

0       1.000     1.000     1.000       115
1       1.000     1.000     1.000        88
2       1.000     1.000     1.000       114
3       0.989     0.968     0.978        93
4       0.974     1.000     0.987        37
5       0.931     1.000     0.964        94
6       1.000     0.899     0.947        99
7       0.941     1.000     0.970        96
8       1.000     0.931     0.964       101
9       1.000     0.949     0.974        99
10      0.973     0.807     0.882        88
11      0.779     0.972     0.865       109
12      1.000     0.969     0.984        98
13      1.000     1.000     1.000        69
14      0.991     1.000     0.995       105
```



```
[[115   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0  88   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0 114   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0  90   1   0   0   2   0   0   0   0   0   0   0]
 [  0   0   0   0  37   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0  94   0   0   0   0   0   0   0   0   0]
 [  0   0   0   1   0   1  89   2   0   0   0   6   0   0   0]
 [  0   0   0   0   0   0   0  96   0   0   0   0   0   0   0]
 [  0   0   0   0   0   1   0   2  94   0   0   4   0   0   0]
 [  0   0   0   0   0   2   0   0   0  94   0   3   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  71  17   0   0   0]
 [  0   0   0   0   0   1   0   0   0   0   1 106   0   0   1]
 [  0   0   0   0   0   2   0   0   0   0   1   0  95   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  69   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0 105]]
```

Figure 9.1.1.6 The diagram above represents the different performance measures which have been applied on DenseNet model and the results obtained. Recall refers to the ratio of correctly predicted positive observations to all observations in the actual class.Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. F1 Score refers to the weighted average of Precision and Recall, thus taking both false positive and negatives into account. A confusion matrix is a table that is often used to describe the performance of the classifier on a set of test data for which the true values are known. The required results are displayed and provide us with a better insight as to the accuracy of prediction, the number of false positive and false negatives and so on.

9.1.2 Graphical user Interface



Utility System to Analyze Diseased Plants and Predict the Stages using Machine Learning Algorithms

Recent advances in the field of deep learning have enabled us to further understand systems and more accurately predict outcomes based on historical data. Deep learning is successfully able to model large amounts of labeled data accurately. The advantage of a neural network its ability to generate the best outcome without the need for redesigning output criteria.

We thus perform plant disease detection using the techniques of Image Processing and DenseNet, Custom CNN and VGG16 which would allow us to detect diseased plants with greater accuracy as well as the stage of disease.

Diseases: Bacterial Spot, Blight, Leaf Mold, Septoria leaf spot, Two-spotted spider mite, Target Spot, Yellow Leaf Curl Virus, Mosaic virus

Figure 9.1.2.1 The above figure illustrates the homepage of our WebApp for Utility System for Detecting Plant Disease and it’s Stages using Convolutional Neural Networks. It was important to us to maintain a user-friendly, and intuitive interface. To accomplish this, we used a front-end HTML5 framework called Bootstrap.

This is the homepage of the website which describes our vision and mission i.e. detection of various diseases and its stages. The upload Image button redirects user to another page where the user can choose between various architectures for prediction

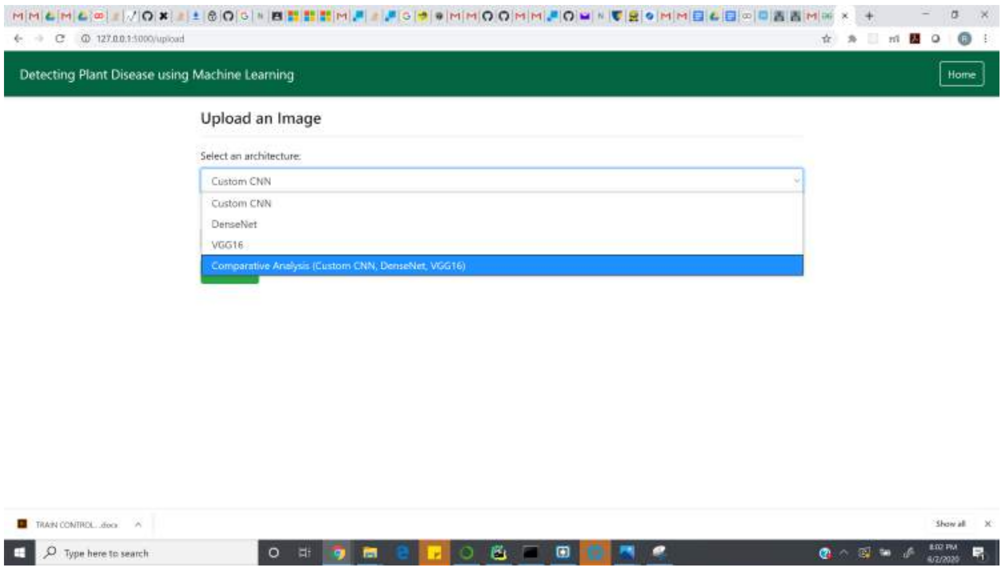
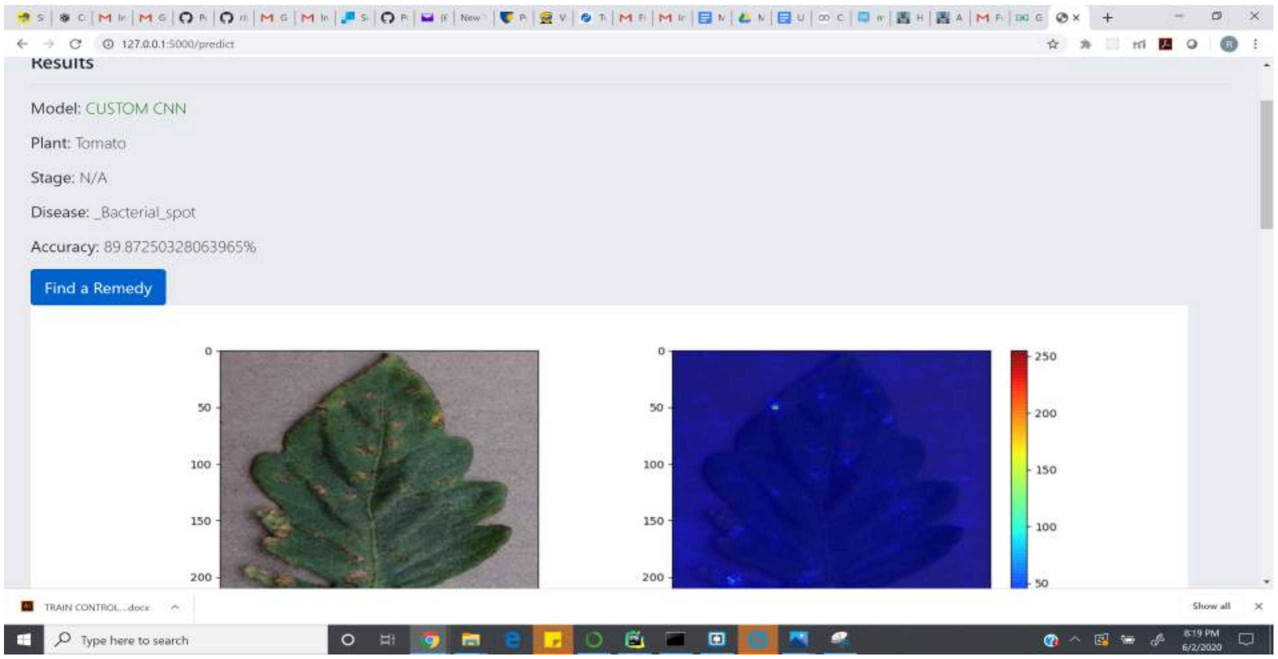

Detecting Plant Disease using Machine Learning
Home
Upload an Image
Select an architecture:
Custom CNN
Custom CNN
DenseNet
VGG16
Comparative Analysis (Custom CNN, DenseNet, VGG16)

Figure 9.1.2.2 The above figure illustrates our WebApp allowing the user to select the architecture with which they'd like to perform classification. The options are as follows: Custom CNN, DenseNet, VGG-16, and a Comparative Analysis option to compare the results of all three architectures.

This page provides the user with a dropdown menu where user can choose between three CNN architectures provided. They are CNN modified, DenseNet and VGG-16. The user is also given another option of comparative analysis which shows the results of all the three models. Through this the user can perform better analysis of the results. Once the architecture is chosen, the images are uploaded and the calculate button is clicked too obtain results.



Results

Model: CUSTOM CNN

Plant: Tomato

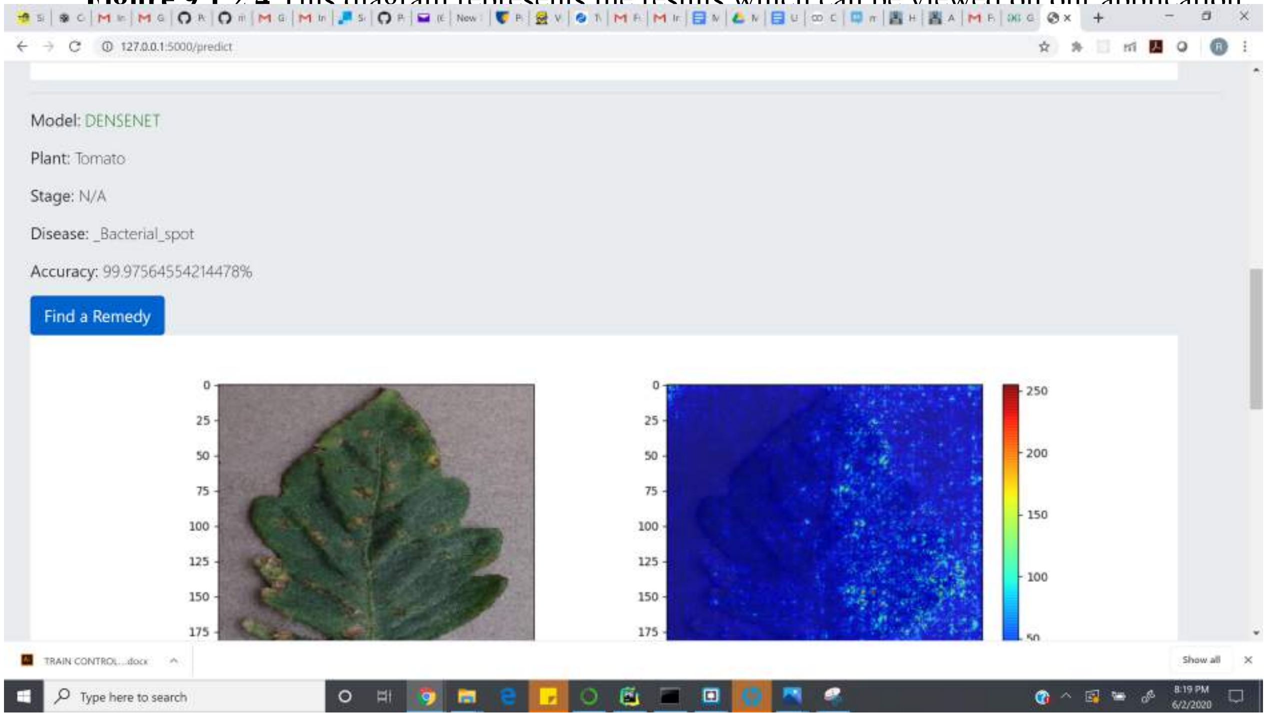
Stage: N/A

Disease: _Bacterial_spot

Accuracy: 89.87250328063965%

Figure 9.1.2.3 This diagram represents the results which can be viewed on our application when the user chooses to obtain the results generated by the modified CNN architecture. The application also provides visualization of the CNN results in the form of a saliency map, indicating regions which were important during prediction. The user can also view the accuracy of the generated result, along with the stage of the disease and an option to obtain remedies. Thus the results obtained are easy to understand and visualization could be used for further analysis.

Figure 9.1.2.4 This diagram represents the results which can be viewed on our application



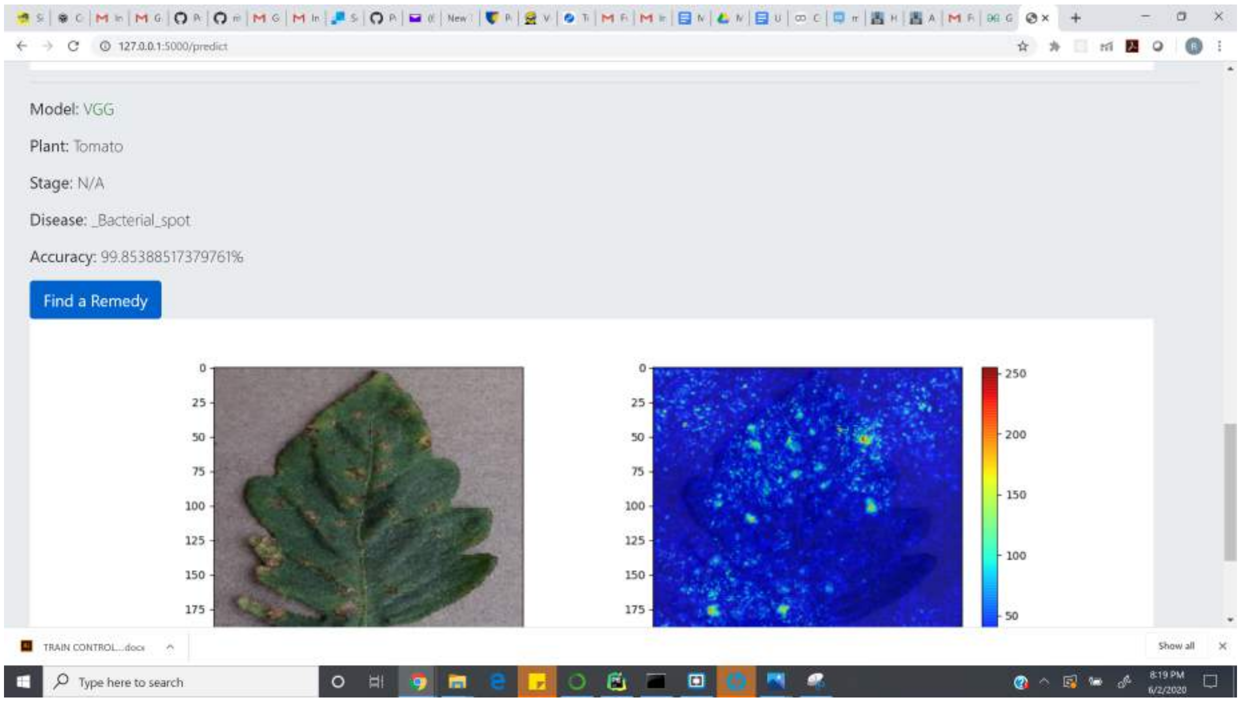
Model: DENSENET

Plant: Tomato

Stage: N/A

Disease: _Bacterial_spot

Accuracy: 99.97564554214478%


Model: VGG

Plant: Tomato

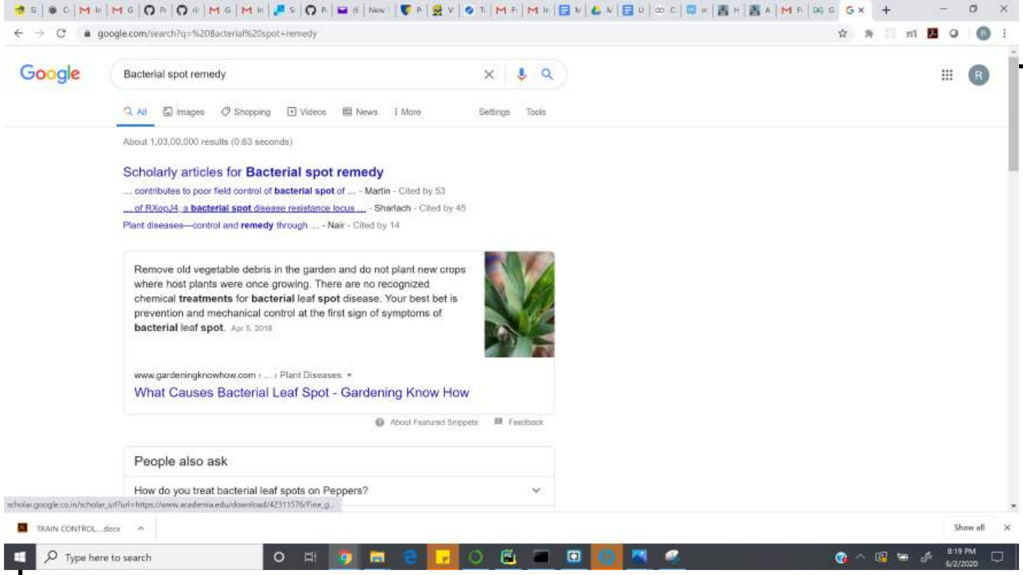
Stage: N/A

Disease: _Bacterial_spot

Accuracy: 99.85388517379761%

Figure 9.1.2.5 This diagram represents the results which can be viewed on our application when the user chooses to obtain the results generated by VGG-16 architecture. The application also provides visualization of the VGG-16 results in the form of a saliency map, indicating regions which were important during prediction. The user can also view the accuracy of the generated result, along with the stage of the disease and an option to obtain remedies. Thus the results obtained are easy to understand and visualization could be used for further analysis.

Based on the model chosen by the user the classification is performed and the prediction is displayed. The displayed details include the Model Chosen, The Plant name, The stage of the disease, the recognized disease and the accuracy obtained for that particular image. Along with the above-mentioned details, the original leaf image as well as a Saliency Map is also displayed for the users to check the accuracy of the identified areas in the Saliency Map. In Comparative Analysis, results of all three models are displayed. This is shown in Figure 9.1.2.3, Figure 9.1.2.4 and Figure9.1.2.5


google.com/search?q=%20Bacterial%20spot+remedy

Bacterial spot remedy

About 1,03,00,000 results (0.63 seconds)

Scholarly articles for **Bacterial spot remedy**

... contributes to poor field control of **bacterial spot** of ... - Martin - Cited by 53

... of RXopJ4, a **bacterial spot** disease resistance locus ... - Sharlach - Cited by 45

Plant diseases—control and **remedy** through ... - Nair - Cited by 14

Remove old vegetable debris in the garden and do not plant new crops where host plants were once growing. There are no recognized chemical **treatments** for **bacterial** leaf **spot** disease. Your best bet is prevention and mechanical control at the first sign of symptoms of **bacterial** leaf **spot**. Apr 5, 2018

www.gardeningknowhow.com › ... › Plant Diseases

What Causes Bacterial Leaf Spot - Gardening Know How

People also ask

How do you treat bacterial leaf spots on Peppers?

scholar.google.co.in/scholar_url?url=https://www.academia.edu/download/42311576/Fine_g...

Figure 9.1.2.6 The above figure illustrates the Google search results for the remedy of the disease classified. Our WebApp makes it easy to access this remedy information through our 'Find a Remedy' button in classification results, which directs the user to the appropriate Google search webpage.Once the disease and its stage is identified, the webpage provides a find remedy button which provides solutions or remedies to the particular disease by redirecting the user to google search engine.

9.1.3 Accuracy Details

Model	Accuracy
Modified CNN	0.974
VGG16	0.936
DenseNet-169	0.995

Chapter 10

Conclusion & scope for future work

10.1 Findings and suggestions

We developed an application that takes care of disease detection using CNN models at the backend. Through this application life for farmers is made much easier by facilitating a platform that helps them identify various diseases and its stages without any prior knowledge of how it is done. All that has to be done is, upload an image, choose a model or choose all 3 CNN models i.e Densenet, VGG and Modified, to perform a comparative analysis and the result is displayed which involves the identified disease and its stage along with accuracy. It also displays Query Image, Saliency Map.

We suggest looking closely at some details to be taken care of while using this application. Focus on the properties of the image being uploaded such as the type, quality, size etc. Another point to be noted is to make sure to use the comparative analysis feature in order to be able get a better idea on accuracy details.

10.2 Significance of the Proposed Research Work

Agriculture comes under the primary sector in India. Most part of India is dependent on the agriculture for their livelihood. So loss of crops from plant disease may result in hunger and starvation. Leaves are important because they are the primary source of photosynthesis, which is how plants feed themselves and hence feed us. It is very difficult to the farmers to diagnose the disease only with the help of observation. Hence, Identification of diseases of plant leaf which is a very important and challenging task, becomes extremely significant.

10.3 Limitation of this Research Work

Every project has its own limitations and can be worked upon in the future. Even though we have achieved an almost efficient system, It still has certain limitations:

- The image which is uploaded needs to have specific file properties before being uploaded on to the webpage
- Sometimes false positive cases of disease maybe be obtained due to inaccuracy of the image

- The implementation is done only for tomato, bell pepper and potato. Hence, Only the diseases with respect to these crops are detected.

10.4 Directions for the Future works

We have developed a project for disease detection in the leaves of the plants tomato, potato and bell pepper. There are several other crops which also get affected that need to be identified . hence, we aim to expand the already existing application to other crops. It can be extended to building an Android application to make the access easier since everyone has access to mobile phones.

References

[1] - D.L. Borges, S.T.d.M. Guedes, A.R. Nascimento, P. Melo-Pinto, “Detecting and grading severity of bacterial spot caused by Xanthomonas spp. in tomato (Solanum lycopersicon) fields using visible spectrum images” Computers and Electronics in Agriculture, Vol. 125 No., pp. 149-159, 2016

[2] - H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (SURF), Computer Vision and Image Understanding, Volume 110, Issue 3, 2008, Pages 346-359

[3] - Satish Madhogaria, Marek Schikora , Wolfgang Koch , Daniel Cremers “Pixel-Based Classification Method for Detecting Unhealthy Regions in Leaf Images”, Conference: 6th workshops on Sensor Data Fusion (SDF) , 2011, 1-6

[4] - Rupali Patil1, Sayali Udgave,Supriya More, Dhanashri Nemishte, Monika Kasture “Grape Leaf Disease Detection Using K-means Clustering Algorithm” ,IRJET Volume: 03 Issue: 04,2016, 2330-2333

[5] - Raza S-e-A, Prince G, Clarkson JP, Rajpoot NM “Automatic Detection of Diseased Tomato Plants Using Thermal and Stereo Visible Light Images”, PLoS ONE 10(4): e0123262, 2016, 1-20

[6] - Chéné, Y., Rousseau, D., Lucidarme, P., Bertheloot, J., Caffier, V., Morel, P,“On the use of depth camera for 3d phenotyping of entire plants”, Computers and Electronics in Agriculture 82, 2012, 122–127

[7] -Mokhtar, U., M. A. S. Ali, A. E. Hassanien, and H. Hefny, "Identifying Two of Tomatoes Leaf Viruses Using Support Vector Machine", Second International Conference, vol. 339, 2015 , pp. 771-782

[8] - Wetterich, C. B., Kumar, R., Sankaran, S., Junior, J. B., Ehsani, R., and Marcassa, L. G. “A comparative study on application of computer vision and fluorescence imaging

spectroscopy for detection of huanglongbing citrus disease in the usa and brazil", Journal of Spectroscopy,2012, 2314-2420

[9] - Dong Pixia, Wang Xiangdong,"Recognition of Greenhouse Cucumber Disease Based on Image Processing Technology", Open Journal of Applied Sciences 3(01), 2013, 27-31

[10] - A. Krizhevsky, I. Sutskever, G.E. Hinton, "Imagenet classification with deep convolutional neural networks, Advances in neural information processing systems", 2012, 1-9

[11] - Anand.H.Kulkarni , Ashwin Patil R. K "Applying image processing technique to detect plant diseases ". January 2012

[12] - Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. "ImageNet large scale visual recognition challenge", International Journal of Computer Vision volume 115, 211–252(2015)

[13] - M. Havaei, A. Davy, D. Warde-Farley, A. Biard, A. Courville, Y. Bengio, C. Pal, P.-M. Jodoin, H. Larochelle, "Brain tumor segmentation with deep neural networks, Medical image analysis", Volume 35, January 2017, Pages 18-31

[14] - L.A. Alexandre, "3D object recognition using convolutional neural networks with transfer learning between input channels", Intelligent Autonomous Systems 13 Proceedings of the 13th International Conference IAS-13, 2016, (pp.889-898)

[15] - H. Xue, Y. Liu, D. Cai, X. He, "Tracking people in RGBD videos using deep learning and motion clues", Neurocomputing Volume 204, 5 September 2016, Pages 70-76

[16] - G.L. Grinblat, L.C. Uzal, M.G. Larese, P.M. Granitto, "Deep learning for plant identification using vein morphological patterns", Computers and Electronics in Agriculture, Volume 127, September 2", Neurocomputing, Volume 235, 26 April 2017, Pages 228-235

[17] - Mehdipour Ghazi, Mostafa, Yanikoglu, Berrin Aptoula, Erchan, “Plant Identification Using Deep Neural Networks via Optimization of Transfer Learning Parameters”, Neurocomputing, Volume 235, 11 January 2017

[18] - S.P. Mohanty, D.P. Hughes, M. Salathé “Using Deep Learning for Image-Based Plant Disease Detection”, Frontiers in Plant Science, 2016, 7 pages

[19] - S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, D. Stefanovic, “Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification”, Computational Intelligence and Neuroscience, 2016, 11 pages

[20] - Chowdhury Rafeed Rahman , Preetom Saha Arko , Mohammed Eunus Ali , Mohammad Ashik Iqbal Khan , Sajid Hasan Apon , Farzana Nowrinb , Abu Wasif – “ Identification and Recognition of Rice Diseases and Pests Using Convolutional Neural Networks”,ArXiv 2018, Computer Science

[21] - Y. Lu, S. Yi, N. Zeng, Y. Liu, Y. Zhang, “Identification of Rice Diseases using Deep Convolutional Neural Networks”, Neurocomputing 267, 2017, Pages 378-384

[22] - M. Brahimi, K. Boukhalfa, A. Moussaoui “Deep Learning for Tomato Diseases: Classification and Symptoms Visualization”, Applied Artificial Intelligence Volume 31, Issue 4 , 2017, Pages 299-315

[23] - Sachin B. Jagtap, Shailesh Hambarde “Morphological Feature Extraction Based Image Processing System For Agricultural Plant Disease Detection And Diagnosis” , International Journal of Engineering Sciences 7(2), 2014, 61-66

[24] - Sanjay B. Dhaygude, Mr.Nitin P.Kumbhar “Agricultural plant Leaf Disease Detection Using Image Processing “, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 2, Issue 1, 2013, Pages 599-602