A PROJECT REPORT ON

PLANT DISEASES DETECTION AND CLASSIFICATION USING MACHINE LEARNING

Submitted In Fulfilment Of The Requirements

For The Award of The Degree Of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

MANISH KUMAR SAHOO	**1602-16-735-080**
K. S. V. N. SREE CHARAN	**1602-16-735-105**
GUMMERLA VIDYASAGAR	**1602-16-735-115**

UNDER THE GUIDANCE OF

Dr. K. VEERASWAMY

PROFESSOR

ELECTRONICS AND COMMUNICATION ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VASAVI COLLEGE OF ENGINEERING

(AFFILIATED TO OSMANIA UNIVERSITY)

IBRAHIMBAGH, HYDERABAD-500031

2020

DEPARTMENT OF

ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the Project Report entitled: “PLANT DISEASES DETECTION AND CLASSIFICATION USING MACHINE LEARNING ” is a bonafide work done and submitted.

MANISH KUMAR SAHOO	**1602-16-735-080**
K. S. V. N. SREE CHARAN	**1602-16-735-105**
GUMMERLA VIDYASAGAR	**1602-16-735-115**

In fulfilment of requirement for the award of Bachelor of Engineering degree in Electronics and Communication Engineering during the year 2019-2020.The result embodied in this project report has not been submitted to any other university or institute for the award of any degree

Head of the Department
Dr . E.SREENIVASA RAO
Professor and HOD, ECE Dept.
VCE-HYD

Internal Guide
Dr.K.VEERASWAMY
Professor, ECE Dept.
VCE-HYD

External Examiner

DECLARATION

We hereby declare that the results embodied in this dissertation entitled "PLANT DISEASES DETECTION AND CLASSIFICATION USING MACHINE LEARNING " is carried out by us during the academic year 2019-2020 in fulfilment of the requirement for the award of B.E. (Electronics and Communication Engineering) from "**Vasavi College of Engineering**".

We have not submitted the same to any other university or organization for the award of any other degree.

MANISH KUMAR SAHOO (1602-16-735-080)

K. S. V. N. SREE CHARAN (1602-16-735-105)

GUMMERLAVIDYA SAGAR (1602-16-735-115)

ACKNOWLEDGEMENT

This satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mentioning of the people whose constant guidance and encouragement made it possible. We take pleasure in presenting before you, our project, which is result of studied blend of both research and knowledge.

It is our privilege to express our earnest gratitude and venerable regards to our internal guide **Dr.K.Veeraswamy**, PROFESSOR, E.C.E. Department, Vasavi College of Engineering, Ibrahimbagh, for abounding and able guidance during the preparation and execution of the project work. We are grateful for her cooperation and her valuable suggestions.

We record with pleasure our deep sense of gratitude to **Dr. E.SREENIVASA RAO**, Head of the Department, E.C.E. for his simulating guidance and profuse assistance we have received, which helped throughout the project.

We also thank our Project coordinator, Professor **Dr. N. SIVA SANKAR REDDY**, Department of ECE, for her continuous guidance and valuable suggestions throughout our project.

Also we acknowledge with thanks for the technical help extended by one and all in shaping up our project.

With Regards and Gratitude

MANISH KUMAR SAHOO

K.S.V.N. SREE CHARAN

GUMMERLA VIDYASAGAR

ABSTRACT

Agriculture is the mainstay of the Indian economy. Almost 70% people depend on it & shares major part of the GDP. Diseases in crops mostly on the leaves affects on the reduction of both quality and quantity of agricultural products. Perception of human eye is not so much stronger so as to observe minute variation in the infected part of leaf. In this paper, we are providing software solution to automatically detect and classify plant leaf diseases. In this we are using image processing techniques to classify diseases & quickly diagnosis can be carried out as per disease. This approach will enhance productivity of crops. It includes several steps viz. image acquisition, image pre-processing, segmentation, features extraction and neural.

Recently, many researchers inspired from the success of deep learning in computer vision to improve the performance of plant diseases detection systems. Unfortunately, most of these studies did not leverage the recent deep architectures and based essentially on AlexNet, GoogleNet or similar architectures. Moreover, the deep learning visualization methods are not taken advantage of, which makes these deep classifiers not transparent and qualified as black boxes. In this chapter, we have tested multiple state-of-the-art Convolutional Neural Network (CNN) architectures using three learning strategies on a public dataset for plant diseases classification. These new architectures outperform the state-of-the-art results of plant diseases classification with an accuracy that reached 99.76%. Furthermore, we have proposed the use of saliency maps as visualization method to understand and interpret the CNN classification mechanism.

Contents : Page No:

List of figures:

List of Tables :

Chapter-1

INTRODUCTION

1.1 Introduction To The Topic

In developing countries, farming land can be much larger and farmers can not observe each and every plant, every day. Farmers are unaware of non-native diseases. Consultation of experts for this might be time consuming & costly. Also unnecessary use of pesticides might be dangerous for natural resources such as water, soil, air, food chain etc.as well as it is expected that there need to be less contamination of food products with pesticides. There are two main characteristics of plant disease detection machine-learning methods that must be achieved, they are: speed and accuracy. There is need for developing technique such as automatic plant disease detection and classification using leaf image processing techniques. This will prove useful technique for farmers and will alert them at the right time before spreading of the disease over large area. Solution is composed of four main phases; in the first phase we create a color transformation structure for the RGB leaf image and then, we apply color space transformation for the color transformation structure. Then image is segmented using the K-means clustering technique. In the second phase, unnecessary part (green area) within leaf area is removed. In third phase we calculate the texture features for the segmented infected object. Finally, in the fourth phase the extracted features are passed through a pre-trained neural network.

Plant diseases can cause big damages to agriculture crops which decrease the production significantly. Early blight is a typical example of diseases that can severely decrease the production. Similarly, in a humid climate, late blight is another very destructive disease that is able to affect the plant leaves, stems, and fruits Protecting plants from diseases is vital to guarantee crops quality and quantity. Successful strategy of protection should start by an early detection of the disease in order to choose the appropriate treatment at the right time to prevent its spreading. Usually, this detection is achieved by experts having an academic knowledge reinforced by a practical experience on diseases symptoms and causes. Further more, these experts must monitor plants consistently to avoid disease spreading. This continuous monitoring represents a difficult and time-consuming task for humans, which makes the automation of the plant diseases detection and identification essential to protect plants .Several studies[10,11,12] proposed to detect and classify plant diseases using image processing and machine learning.

These approaches try to build diseases classifiers using images taken from the crops. These classifiers are based on hand-crafted features designed be experts to extract relevant information for image classification. For this reason, these classifiers suffer from the lack of automation because of the hand-crafted features dependency .Moreover, the classifier must be trained using images labeled by experts. Collecting these labeled images is very expensive because it is done manually. This difficulty of data collection has forced the previous studies [10,11,12] to use small datasets to train and test classifiers. Using small labeled datasets is a limiting factor in machine learning, and it can lead to the overfitting. In the last few years, deep learning (DL) is adopted by computer vision community, thanks to its results that outperform the state-of-the-art in many domains. The main advantage of DL in computer vision is the direct exploitation of image without any handcrafted features. DL classifiers are end-to-end systems that form features in a fully automated way without any intervention of human experts. In plant diseases protection, many works have proposed the use of DL to detect and classify diseases. Notably, in more than 54,306 images of diseased and healthy plant leaves are collected which makes the training of DL classifier possible. This new trend produced more accurate classifiers compared to traditional machine learning approaches. Despite these good results, DL research in plant diseases remain immature and require more attention to produce practical systems. For example, many new successful deep architectures are not tested in plant diseases context. Moreover, DL classifiers suffer from lack of interpretability and transparency. These accurate classifiers are often considered as black boxes that give good results but without any explanation and details about the classification mechanism. High accuracy is not sufficient for plan.

- Comparison between state-of-the-art CNN architectures performance in plant diseases protection: this comparison helps researchers to choose the best deep architecture for building a practical system for plant diseases protection.
- Visualization of symptoms used by deep classifiers: visualization methods allow the localization of infected region on the plant and help the users by giving them information about the disease. Also, this biological information is extracted with-out the intervention of agriculture experts. In this study, we propose the saliency map as a visualization method based on derivative of the deep network output with respect to the image.

1.2 Software used:

Python 3.6:

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

Python is a general-purpose high level programming language that is widely used in data science and for producing **deep learning** algorithms. This brief tutorial introduces **Python** and its libraries like Numpy, Scipy, Pandas, Matplotlib; frameworks like Theano, TensorFlow, Keras.

Libraries/Modules Used:

OpenCV:

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

OpenCV uses machine learning algorithms to search for faces within a picture. Because faces are so complicated, there isn’t one simple test that will tell you if it found a face or not. Instead, there are thousands of small patterns and features that must be matched. The algorithms break the task of identifying the face into thousands of smaller, bite-sized tasks, each of which is easy to solve. These tasks are also called classifiers.

Keras:

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet also is the author of the XCeption deep neural network model. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides.

TensorFlow:

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

TensorFlow offers multiple levels of abstraction so you can choose the right one for your needs. Build and train models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy. If you need more flexibility, eager execution allows for immediate iteration and intuitive debugging. For large ML training tasks, use the Distribution Strategy API for distributed training on different hardware configurations without changing the model definition.

Skicit-learn:

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. The scikitlearn library will help us with binarizing our labels, splitting data for training/testing, and generating a training report in our terminal.

Numpy:

NumPy is a library for the Python programming language, adding support for large, multidimensional arrays and matrices, along with a large collection of high level mathematical functions to operate on the arrays.

Pandas:

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the Data Frame. Data Frames allow you to store and manipulate tabular data in rows of observations and columns of variables. Using this we can create csv files(excel) for uploading of attendance data in it.

1.3 Flow Chart:

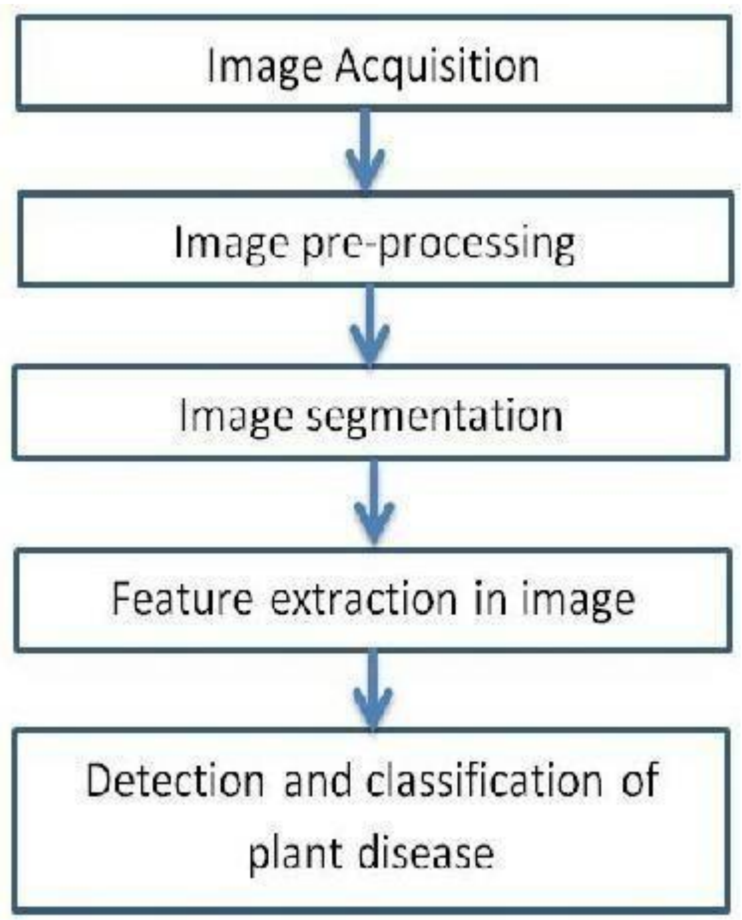


Image Acquisition
Image pre-processing
Image segmentation
Feature extraction in image
Detection and classification of plant disease

Figure 1: This is the flow chart of the implementation of the project

1. Image Acquisition:

The images of the plant leaf are captured through the camera. This image is in RGB (Red, Green And Blue) form. Color transformation structure for the RGB leaf image is created, and then, a device-independent color space transformation for the color transformation structure is applied.

2. Image Pre-processing:

To remove noise in image or other object removal, different pre-processing techniques is considered. Image clipping i.e. cropping of the leaf image to get the interested image region. Image smoothing is done using the smoothing filter. Image enhancement is carried out for increasing the contrast. The RGB images into the grey images using color conversion techniques.

Then the histogram equalization which distributes the intensities of the images is applied on the image to enhance the plant disease images. The cumulative distribution function is used to distribute intensity values.

3. Image Segmentation:

Segmentation means partitioning of image into various part of same features or having some similarity. The segmentation can be done using various methods like otsu' method, k-means clustering, converting RGB image into HIS model etc.

Segmentation using Boundary and spot detection algorithm: The RGB image is converted into the HIS model for segmenting. Boundary detection and spot detection helps to find the infected part of the leaf as discussed in. For boundary detection the 8 connectivity of pixels is consider and boundary detection algorithm is applied[12,13,14,15]

4.Feature Extraction :

Feature extraction plays an important role for identification of an object. In many application of image processing feature extraction is used. Color, texture, morphology, edges etc. are the features which can be used in plant disease detection. Texture means how the color is distributed in the image, the roughness, hardness of the image. It can also be used for the detection of infected plant areas.

1.4 Training accuracy:

Ideally **training accuracy** should be **calculated** including every sample, but practically, it's all right to use a subset of the data as long as its representative of all data. Since the **training** loss is almost always monotonically decreasing, this will tend to overestimate loss (and underestimate **accuracy**.

1.5 Validation loss:

The loss is calculated on training and validation and its interpretation is based on how well the model is doing in these two sets. It is the sum of errors made for each example in training or alidation sets. Loss value implies how poorly or well a model behaves after each iteration of optimization.

1.6 Objectives:

We can reduce the attack of pests by using proper pesticides and remedies .We can reduce the size of the images by proper size reduction techniques and see to it that the quality is not compromised to a great extent. We can expand the projects of the earlier mentioned authors such that the remedy to the disease is also shown by the system . The main objective is to identify the plant diseases using image processing. It also, after identification of the disease, suggest the name of pesticide to be used. It also identifies the insects and pests responsible for epidemic. Apart from these parallel objectives, this drone is very time saving. The budget of the model is quite high for low scale farming purposes but will be value for money in large scale farming. It completes each of the process sequentially and hence achieving each of the output. Thus the main objectives are:

1) To design such system that can detect crop disease and pest accurately.

2) Create database of insecticides for respective pest and disease.

3) To provide remedy for the disease that is detected.

CHAPTER 2

LITERATURE SURVEY

In this section, various method of image processing for plant disease detection is discussed. The vegetation indices from hyper spectral data have been shown for indirect monitoring of plant diseases. But they cannot distinguish different diseases on crop. Wenjiang Huang et al developed the new spectral indices for identifying the winter wheat disease. They consider three different pests (Powdery mildew, yellow rust and aphids) in winter wheat for their study. The most and the least relevant wavelengths for different diseases were extracted using RELIEF-F algorithm. The classification accuracies of these new indices for healthy and infected leaves with powdery mildew, yellow rust and aphids were 86.5%, 85.2%, 91.6% and 93.5% respectively. Enhanced images have high quality and clarity than the original image. Color images have primary colors red, green and blue. It is difficult to implement the applications using RGB because of their range i.e. 0 to 255. Hence they convert the RGB images into the grey images. Then the histogram equalization which distributes the intensities of the images is applied on the image to enhance the plant disease images. Monica Jhuria et al uses image processing for detection of disease and the fruit grading.

They have used artificial neural network for detection of disease. They have created two separate databases, one for the training of already stored disease images and other for the implementation of the query images. Back propagation is used for the weight adjustment of training databases. They consider three feature vectors, namely, color, textures and morphology. They have found that the morphological feature gives better result than the other two features. Zulkifli Bin Husin et al[13], in their paper, they captured the chilli plant leaf image and processed to determine the health status of the chilli plant. Their technique is ensuring that the chemicals should apply to the diseased chilli plant only. They used the MATLAB for the feature extraction and image recognition. In this paper pre-processing is done using the Fourier filtering, edge detection and morphological operations. Computer vision extends the image processing paradigm for object classification. Here digital camera is used for the image capturing and LABVIEW software tool to build the GUI. The segmentation of leaf image is important while extracting the feature from that image.

Mrunalini R. Badnakhe, Prashant R.Deshmukh[16] compare the Otsu threshold and the k-means clustering algorithm used for infected leaf analysis in. They have concluded that the extracted values of the features are less for k-means clustering. The clarity of k-means clustering is more accurate than other method. The RGB image is used for the identification of disease. After applying k-means clustering techniques, the green pixels is identified and then using otsu's method, varying threshold value is obtained. For the feature extraction, color co-occurrence method is used. RGB image is converted into the HSI translation. For the texture statistics computation the SGDM matrix is generated and using GLCM function the feature is calculated. The FPGA and DSP based system is developed by Chunxia Zhang, Xiuqing

Wang and Xudong Li, for monitoring and control of plant diseases. The FPGA is used to get the field plant image or video data for monitoring and diagnosis. The DSP TMS320DM642 is used to process and encode the video or image data. The nRF24L01 single chip 2.4 GHz radio transmitter is used for data transfer. It has two data compress and transmission method to meet user's different need and uses multi-channel wireless communication to lower the whole system cost.Shantanu Phadikar and Jaya Sil uses pattern recognition techniques for the identification of rice disease in. This paper describes a software prototype for rice disease detection based on infected image of rice plant. They used HIS model for segmentation of the image after getting the interested region, then the boundary and spot detection is done to identify infected part of the leaf.

Plant diseases classification can be a very complex task as it relies mainly on experts know-how. Developing a reliable system that is applicable for a large number of classes is a very challenging task. Up to now, most of the approaches for automatic plant diseases classification were depending on machine learning algorithms and basic feature engineering. These approaches usually depend on certain environments and are suited for a smaller number of classes, where some small changes in the system can result in drastic fall in accuracy. In recent years, Convolutional Neural Networks (CNN) have shown great results in many image classification tasks which gave the opportunity for researchers to improve classification accuracy in many fields including agriculture and plant disease.

Classification and detection techniques that can be used for plant leaf disease classification. Here preprocess is done before feature extraction. RGB images are converted into white and then converted into grey level image to extract the image of vein from each leaf. Then basic

Morphological functions are applied on the image. Then the image is converted into binary image. After that if binary pixel value is 0 its converted to corresponding RGB image value. Finally by using pearson correlation and Dominating feature set and Naïve Bayesian classifier disease is detected. There are four steps. Out of them the first one is gathering image from several part of the country for training and testing. Second part is applying Gaussian filter is used to remove all the noise and thresholding is done to get the all green color component. K-means clustering is used for segmentation. All RGB images are converted into HSV for extracting feature.

The Brahimi M, Boukhalfa K, Moussaoui A[3] paper presents the technique of detecting jute plant disease using image processing. Image is captured and then it is realized to match the size of the image to be stored in the database. Then the image is enhanced in quality and noises are removed. Hue based segmentation is applied on the image with customized thresholding formula. Then the image is converted into HSV from RGB as it helps extracting region of interest. This approach proposed can significantly support detecting stem oriented diseases for jute plant.

According to DeChant C, Wiesner-Hanks T[6] paper they have proposed for a technique that can be used for detecting paddy plant disease by comparing it with 100 healthy images and 100 sample of disease1 and another 100 sample of disease2. It's not sufficient enough to detect disease or classify it training data is not linearly separable.

In detection of unhealthy plant leaves include some steps are RGB image acquisition. Converting the input image from RGB to HSI format. Masking and removing the green pixels. Segment the components using Ostu's method. Computing the texture features using color-co-occurrence methodology and finally classifying the disease using Genetic Algorithm.

A gray scale image is turned into binary image depending on threshold value. The threshold algorithm is used for image segmentation. The threshold values are given color indices like red, green, blue. But the thresholding is not a reliable method as this technique only distinguishes red tomatoes from other colors. It becomes difficult to distinguish ripe and unripe tomatoes. For this K-means clustering algorithm is used to overcome the drawbacks. K-means create a particular number of non-hierarchical clusters. This method is numerical, unsupervised, non-deterministic and iterative. Then separating the infected parts from the leaf the RGB image was converted into YcbCr to enhance the feature of the image. The final step is the calculation of the percentage of infection and distinguishing the ripe and unripe tomatoes.

The methodology for cucumber disease detection is present. The methodology includes image acquisition, image pre-processing, feature extraction with Grey level co-occurrence matrix (GLCM) and finally classified with two types: Unsupervised classification and supervised classification.

Paddy plant is an important plant in continental region. In paper RGB images are converted into grey scale image using color conversion. Various enhancement techniques like histogram equalization and contrast adjustment are used for image quality enhancement. Different types of classification features like SVM, ANN, FUZZY classification are used here. Feature extract6ion uses different types of feature values like texture feature, structure feature and geometric feature. By using ANN and FUZZY classification, it can identify the disease of the paddy plant.

Popular methods have been utilizes machine learning, image processing and classification based approaches to identify and detect the disease of agricultural product. Image processing technique are used to detect the citrus leaf disease. This system includes: Image pre-processing, segmentation of the leaf using K-means clustering to determine the diseased areas, feature extraction and classification of disease. Uses Grey-Level Co-Occurrence matrix (GLCM) for feature extraction and classification is done using support vector machine (SVM).

The most significant challenge faced during the work was capturing the quality images with maximum detail of the leaf color. It is very typical task to get the image with all the details within a procesable memory. Such images are formed a through high resolution and thus are of 6-10MB of size. This was handled by using a Nikon made D5200 camera which served the task very well. Second challenge faced was to get rid of illumination conditions as from the start to the end of paddy crop season, illumination varies a lot even when the image acquiring time is fixed. However the solution to this is variable user defined thresholding and making necessary adjustments to the shades of LCC.

Sachin D. Khirade & et al… [1] Identification of the plant diseases is the key to preventing the losses in the yield and quantity of the agricultural product. It requires tremendous amount of work, expertize in the plant diseases, and also require the excessive processing time. Hence, image processing is used for the detection of plant diseases. Disease detection involves the steps like image acquisition, image pre-processing, image segmentation, feature extraction and classification. This paper discussed the methods used for the detection of plant diseases using their leaves images. This paper discussed various techniques to segment the disease part of the plant. This paper also discussed some Feature extraction and classification techniques to extract the features of infected leaf and the classification of plant diseases. The accurately detection and classification of the plant disease is very important for the successful cultivation of crop and this can be done using image processing. This paper discussed various techniques to segment the disease part of the plant. This paper also discussed some Feature extraction and classification techniques to extract the features of infected leaf and the classification of plant diseases. The use of ANN methods for classification of disease in plants such as self- organizing feature map, back propagation algorithm, SVMs etc. can be efficiently used. From these methods, we can accurately identify and classify various plant diseases using image processing technique

Mr. Pramod S. landge, Sushil A. Patil& et al… [2]In this propose and experimentally evaluate a software solution for automatic detection and classification of plant diseases through Image Processing. Farmers in rural India have minimal access to agricultural experts, who can inspect crop images and render advice. Delayed expert responses to queries often reach farmers too late. This paper addresses this problem with the objective of developing image processing algorithms that can recognize problems in crops from images, based on colour, texture and shape to automatically detect diseases or other conditions that might affect crops and give the fast and accurate solutions to the farmer with the help of SMS. The design and implementation of these technologies will greatly aid in selective chemical application, reducing costs and thus leading to improved productivity, as well as improved produce.

Rothe et al.has proposed pattern recognition techniques for the detection and classification of cotton leaf diseases of Alternarnia, Myrothecium and Bacterial Blight. The dataset images are taken from the field of Central Institute of Cotton Research Nagpur. Active contour based segmentation algorithm is used for the isolation of diseased spots. Author has also suggested some feature directions to the similar concept for the crops of wheat, orange, citrus and maize etc.

Chapter 3

Convolutional Neural Network

3.1 Convolution neutral network

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

CNNs are regularized versions of multilayer perceptron's. Multilayer perceptron's usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme. Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered.

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is commonly a RELU layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution. Though the layers are colloquially referred to as convolutions, this is only by convention. Mathematically, it is technically a sliding dot product or cross-correlation. This has significance for the indices in the matrix, in that it affects how weight is determined at a specific index point. The Conv layer is the core building block of a Convolutional Network that does most of the computational heavy lifting.

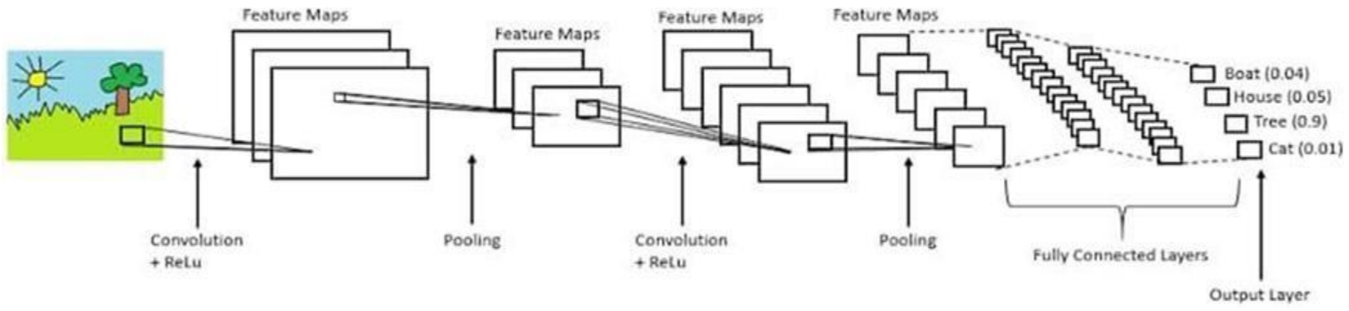


Figure2: Typical processing of image using CNN

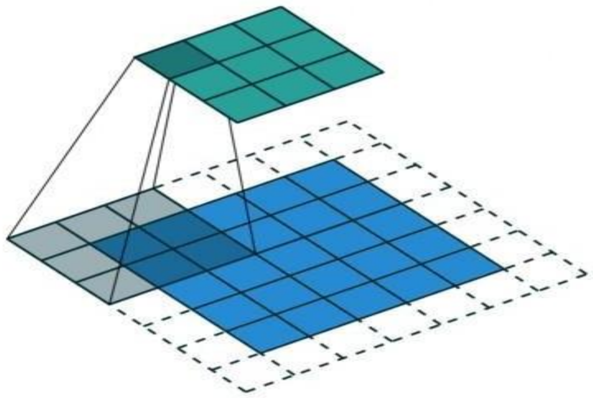


Figure 3:Application of layer of 3*3 on 5*5 with stride 1

Imagine you have an image represented as a 5x5 matrix of values, and you take a 3x3 matrix and slide that 3x3 window around the image. At each position the 3x3 visits, you matrix multiply element wise the values of your 3x3 window by the values in the image that are currently being covered by the window and it also passes through RELU Activation.. This results in a single number the represents all the values in that window of the image.

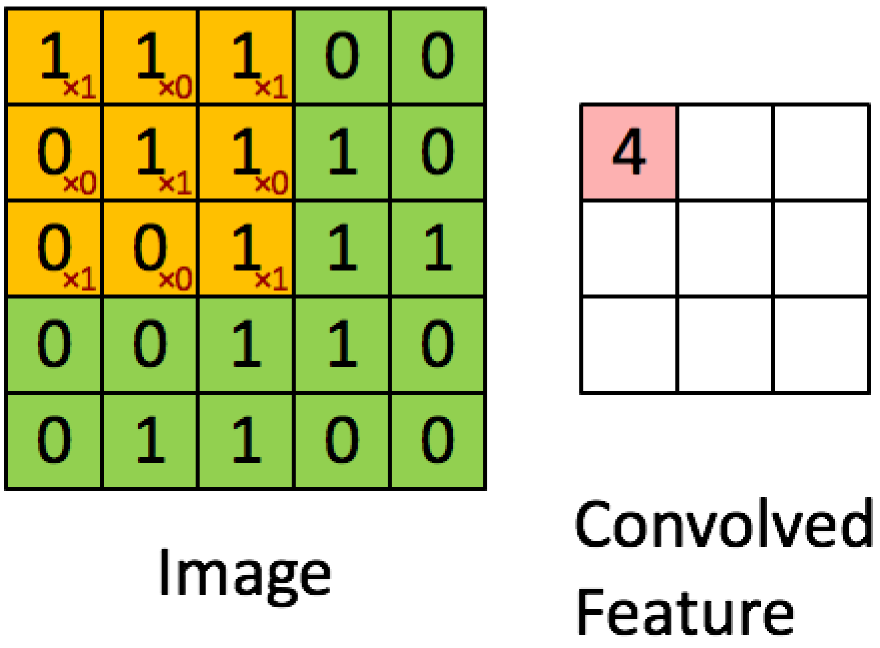


Figure 4: array form of image converted to a convoluted feature

The “window” that moves over the image is called a **kernel.**

3.2 Pooling layer

Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters at one layer into a single neuron in the next layer. For example, **max pooling** uses the maximum value from each of a cluster of neurons at the prior layer. Another is **average pooling**, which uses the average value from each of a cluster of neurons at the prior layer.

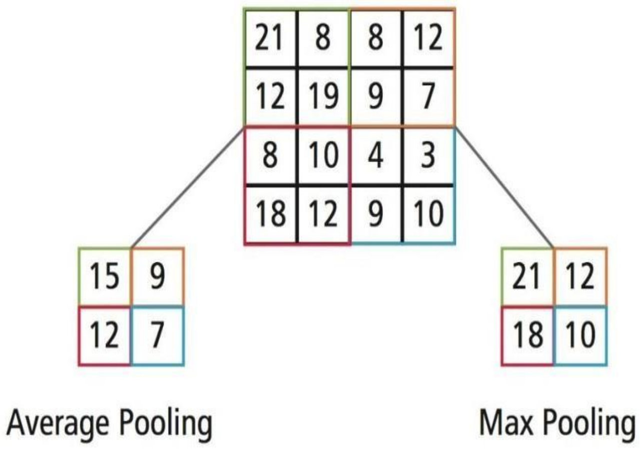


Figure 5:Example of Pooling

3.3 Activation layer

Activation functions are important for a Artificial Neural Network to learn and understand the complex patterns. The main function of it is to introduce non-linear properties into the network. What it does is, it calculates the ‘weighted sum’ and adds direction and decides whether to ‘fire’ a particular neuron or not. There are several kinds of non-linear activation functions, like Sigmoid, Tanh, ReLU and leaky ReLU. The non linear activation function will help the model to understand the complexity and give accurate results.

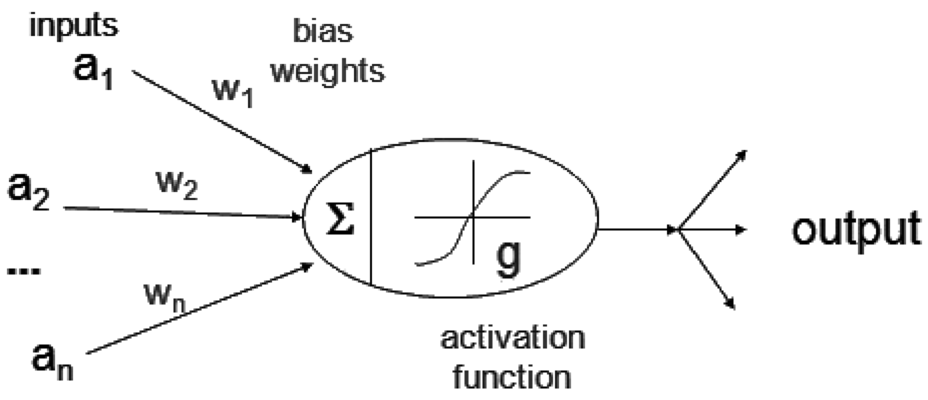


Figure 6:Representation of typical Activation Function

3.4 Applications of CNN

1. Decoding Facial Recognition

Facial recognition is broken down by a convolutional neural network into the following major components –

- Identifying every face in the picture
- Focusing on each face despite external factors, such as light, angle, pose, etc.
- Identifying unique features
- Comparing all the collected data with already existing data in the database to matcha face with a name.
- A similar process is followed for scene labeling as well.

2. Analyzing Documents

- Convolutional neural networks can also be used for document analysis.

3. Historic and Environmental Collections

- CNNs are also used for more complex purposes such as natural history collections. These collections act as key players in documenting major parts of history such as biodiversity, evolution, habitat loss, biological invasion, and climate change.

4. Understanding Climate

- CNNs can be used to play a major role in the fight against climate change, especially in understanding the reasons why we see such drastic changes and how we could experiment in curbing the effect.

5. Grey Areas

- Introduction of the grey area into CNNs is posed to provide a much more realistic picture of the real world.

6. Advertising

- CNNs have already brought in a world of difference to advertising with the introduction of programmatic buying and data-driven personalized advertising.

7. Other Interesting Fields

- CNNs are poised to be the future with their introduction into driverless cars, robots that can mimic human behavior, aides to human genome mapping projects, predicting earthquakes and natural disasters, and maybe even self-diagnoses of medicalproblems.

Chapter 4

Plant Disease Detection Using Deep Learning

4.1 Data collection: is an important stage for developing any data-driven application. Particularly, deep models require a large dataset to avoid overfitting which presents a major challenge[6]. Up to now, only a few datasets are publicly available for diseased plants. Most of the works in this area are conducted on Plant Village public dataset or private datasets.

Dataset structure Plant Village dataset. This dataset includes 54323 images of 14 crop species with 15 classes of diseases or healthy plants, as shown in Table 1.All used images in the experimental tests are randomly cropped to be 224*224 or 299*299 according to the network input size

Table 1 PlantVillage dataset details

Sno:	Disease name	No:of images
1	Pepper_bell_bacterial_spot	2343
2	Pepper_bell_healthy	3213
3	Potato_Early_blight	2354
4	Potato_healthy	3435
5	Potato_late_blight	2243
6	Tomato_Target_spot	2900
7	Tomato_Tomato_mosaic_virus	2876
8	Tomato_Tomato_Yellowleaf_Curl_Virus	2856
9	Tomato_Bacterial_spot	2999
10	Tomato_Early_blight	2232
11	Tomato_healthy	3562
12	Tomato_late_blight	2134
13	Tomato_leaf_mold	1343
14	Tomato_Septoria_leaf_spot	3465
15	Tomato_Spider_mites_two_spotted_spid_	3421

4.2 Splitting of Dataset

It is typical to allocate a percentage of your data for training and a smaller percentage of your data for testing. The scikit-learn provides a handy train_test_split function which will split the data for us. Both trainX and testX make up the image data itself while trainY and testY make up the labels. Our class labels are currently represented as strings; however, Keras will assume that both:

- Labels are encoded as integers
- And furthermore, one-hot encoding is performed on these labels making each label represented as a vector rather than an integer

A call to fit_transform finds all unique class labels in trainY and then transforms them into one- hot encoded labels. A call to just .transform on testY performs just the one-hot encoding step — the unique set of possible class labels was already determined by the call to .fit_transform.

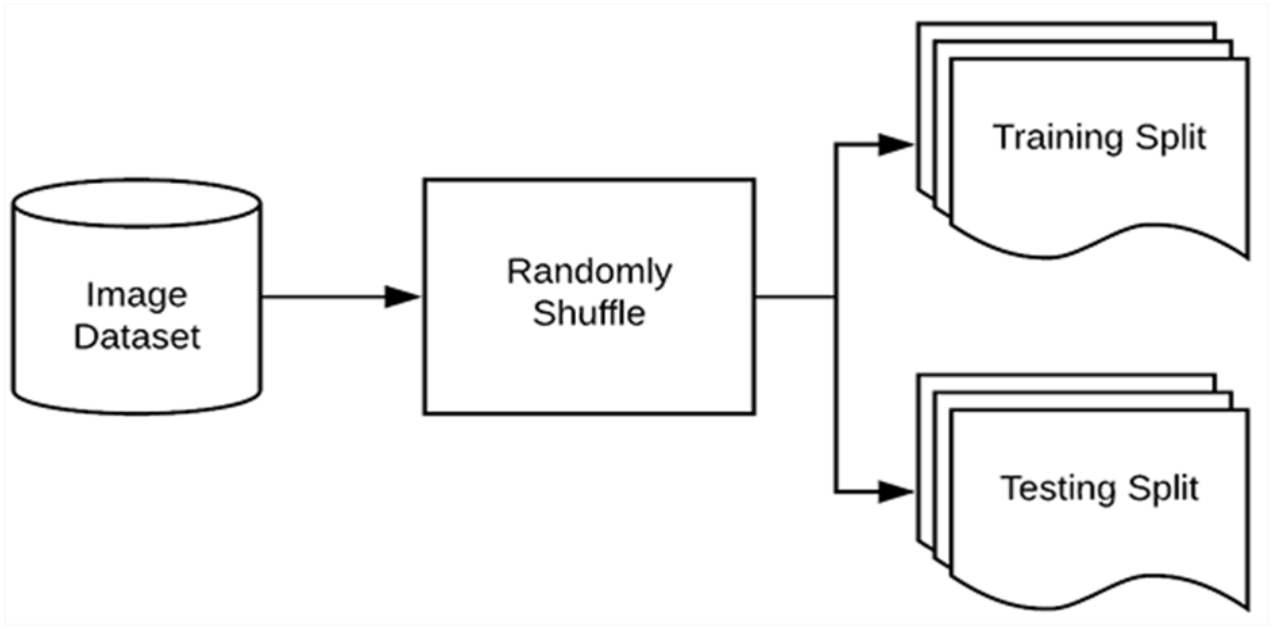


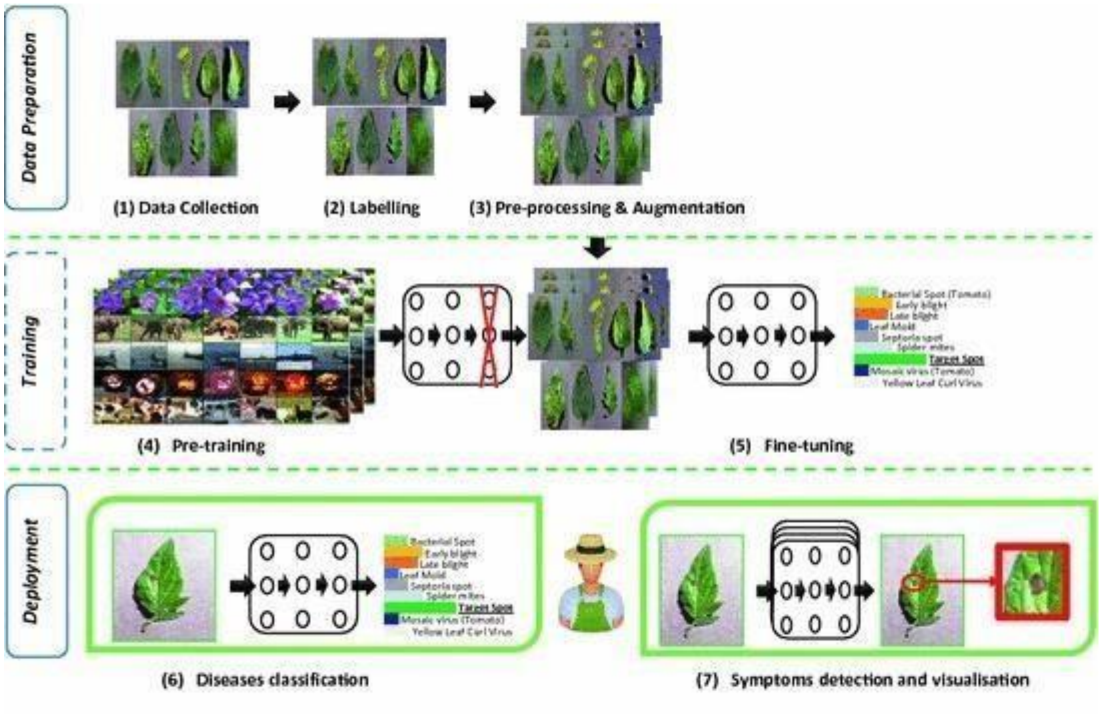
Figure7:Image showing the splitting of data `

4.3 Labeling:

The labeling process consists of annotating the collected images by a human expert. This expert labels images according to two possible strategies[5,6,8]:

Weak labeling: where the agriculture expert identifies only the disease in each plant without any additional information about this disease.

Strong labeling: where the agriculture expert determines, in addition to the disease, the infected regions on plant. This labeling strategy is expensive and time-consuming because it requires the patience of the expert where he uses the annotation software. For this reason, most of the available dataset are weakly labeled.



Data Preparation
(1) Data Collection
(2) Labelling
(3) Pre-processing & Augmentation
Training
(4) Pre-training
(5) Fine-tuning
Deployment
(6) Diseases classification
(7) Symptoms detection and visualisation

Figure 8: Representation of major steps in implementation

4.4 Data augmentation and pre-processing: deep models like CNN are very greedy to labeled data as discussed in Sect. 3. Unfortunately, data collection and labeling are very tedious and expensive tasks. For addressing this problem, data augmentation techniques are used commonly by DL researchers. Augmentation techniques aim to increase the size of the dataset and include more variations. These techniques consist of geometrical transformations (resizing, crop, rotation, horizontal flipping) and intensity transformations (contrast and brightness enhancement, color, noise). Moreover, image preprocessing is used to normalize the images of the dataset. The most used techniques in DL context are image resizing and mean subtraction. The resizing is used to convert input images to the size of the network input layer. However, mean subtraction is used to center the data which accelerate the optimization using gradient descent algorithm. After the process of data preparation, deep models and particularly CNN models are trained using back propagation algorithm. This algorithm aims to minimize a cost function that measures the total error of the model on the training dataset. To reduce this error, the gradient of this cost function is calculated with respect to all weights. Then, gradient descent algorithm is used to find the optimum of the cost function.

4.5 Normalization

In image processing, normalization is a process that changes the range of pixel intensity values. Applications include photographs with poor contrast due to glare, for example. Normalization is sometimes called contrast stretching or histogram stretching. In more general fields of data processing, such as digital signal processing, it is referred to as dynamic range expansion.The purpose of dynamic range expansion in the various applications is usually to bring the image, or other type of signal, into a range that is more familiar or normal to the senses, hence the term normalization. For example, a newspaper will strive to make all of the images in an issue share a similar range grayscale. Normalization transforms an n-dimensional grayscale image with intensity values in the range (Min, Max), into a new image with intensity values in the range (newMin, newMax).The linear normalization of a gray scale digital image is performed according to the formula .For example, if the intensity range of the image is 50 to 180 and the desired range is 0 to 255 the process entails subtracting 50 from each of pixel intensity, making the range 0 to 130. Then each pixel intensity is multiplied by 255/130, making the range 0 to 255. The model type that we will be using is Sequential. Sequential is the easiest way to build model in Keras. It allows you to build a model layer by layer.

4.6 Building the model

We use the ‘add()’ function to add layers to our model.

Our first 2 layers are Conv2D layers. These are convolution layers that will deal with our input images, which are seen as 2-dimensional matrices.

64 in the first layer and 32 in the second layer are the number of nodes in each layer. This number can be adjusted to be higher or lower, depending on the size of the dataset. In our case, 64 and 32 work well, so we will stick with this for now.

Kernel size is the size of the filter matrix for our convolution. So a kernel size of 3 means we will have a 3x3 filter matrix. Refer back to the introduction and the first image for a refresher on this.

Activation is the activation function for the layer. The activation function we will be using for our first 2 layers is the ReLU, or Rectified Linear Activation. This activation function has been proven to work well in neural networks.

Our first layer also takes in an input shape. This is the shape of each input image, 256,256,1 as seen earlier on, with the 1 signifying that the images are greyscale.

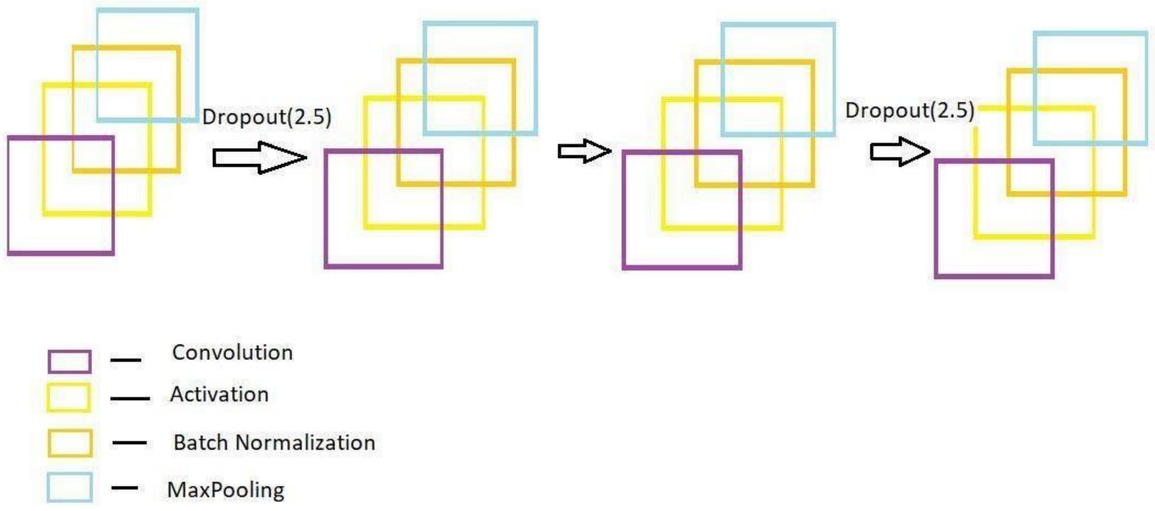


Figure 9: Architecture of cnn used in the project

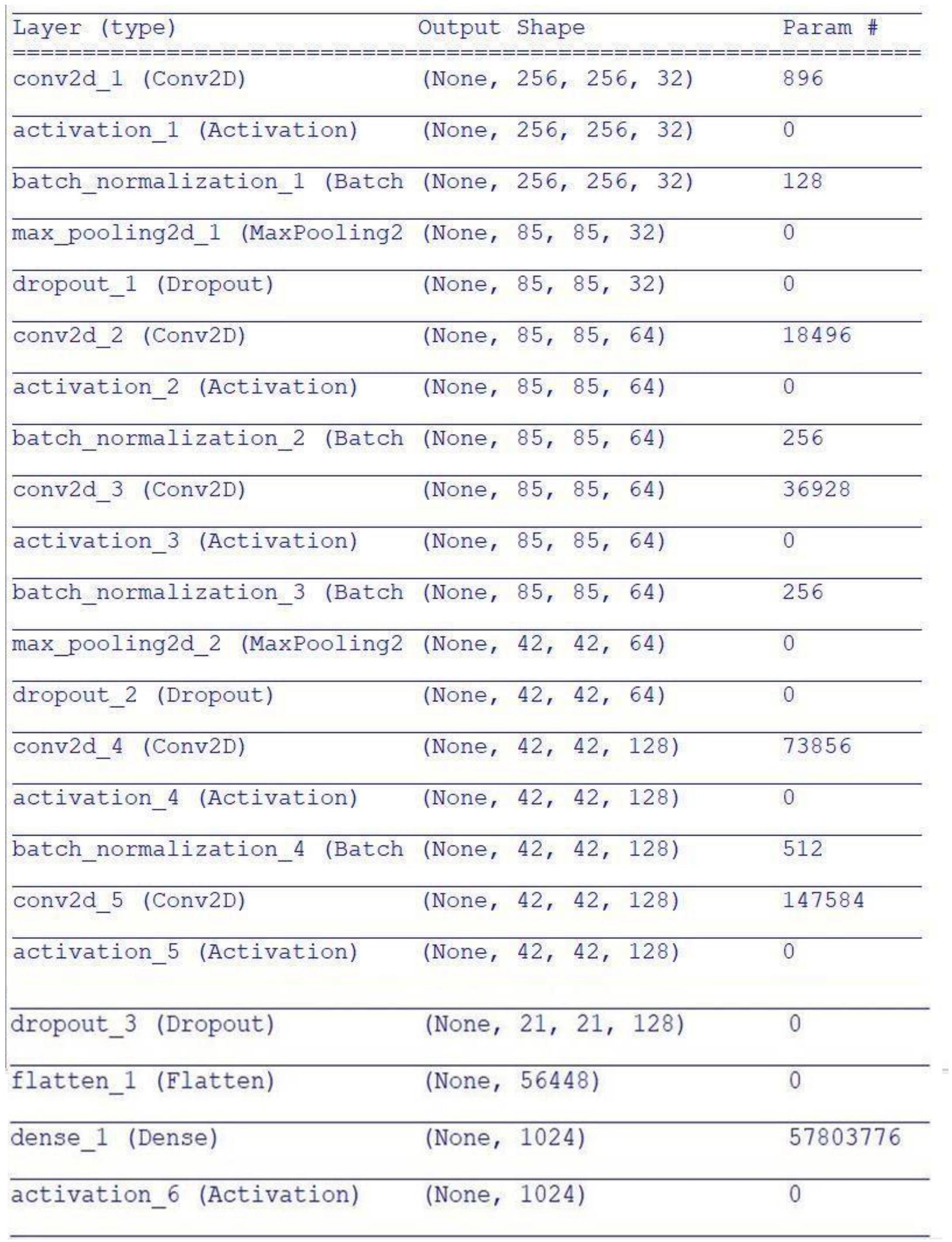
In between the Conv2D layers and the dense layer, there is a ‘Flatten’ layer. Flatten serves as a connection between the convolution and dense layers. ‘Dense’ is the layer type we will use in for our output layer. Dense is a standard layer type that is used in many cases for neural networks. We will have 10 nodes in our output layer, one for each possible outcome (0–9).The activation is ‘softmax’. Softmax makes the output sum up to 1 so the output can be interpreted as probabilities. The model will then make its prediction based on which option has the highest probability.

4.7 Training the model

Now we will train our model. To train, we will use the ‘fit()’ function on our model with the following parameters: training data (train_X), target data (train_y), validation data, and the number of epochs.For our validation data, we will use the test set provided to us in our dataset, which we have split into X_test and y_test.

The number of epochs is the number of times the model will cycle through the data. The more epochs we run, the more the model will improve, up to a certain point. After that point, the model will stop improving during each epoch. For our model, we will set the number of epochs to 25.Now we will train our model. To train, we will use the ‘fit()’ function on our model with the following parameters: training data (train_X), target data (train_y), validation data, and the number of epochs.For our validation data, we will use the test set provided to us in our dataset, which we have split into X_test and y_test.

The number of epochs is the number of times the model will cycle through the data. The more epochs we run, the more the model will improve, up to a certain point. After that point, the model will stop improving during each epoch. For our model, we will set the number of epochs to 25.



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 256, 256, 32)      896
_________________________________________________________________
activation_1 (Activation)    (None, 256, 256, 32)      0
_________________________________________________________________
batch_normalization_1 (Batch (None, 256, 256, 32)      128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 85, 85, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 85, 85, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 85, 85, 64)        18496
_________________________________________________________________
activation_2 (Activation)    (None, 85, 85, 64)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 85, 85, 64)        256
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 85, 85, 64)        36928
_________________________________________________________________
activation_3 (Activation)    (None, 85, 85, 64)        0
_________________________________________________________________
batch_normalization_3 (Batch (None, 85, 85, 64)        256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 42, 42, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 42, 42, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 42, 42, 128)       73856
_________________________________________________________________
activation_4 (Activation)    (None, 42, 42, 128)       0
_________________________________________________________________
batch_normalization_4 (Batch (None, 42, 42, 128)       512
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 42, 42, 128)       147584
_________________________________________________________________
activation_5 (Activation)    (None, 42, 42, 128)       0

_________________________________________________________________
dropout_3 (Dropout)          (None, 21, 21, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 56448)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              57803776
_________________________________________________________________
activation_6 (Activation)    (None, 1024)              0
_________________________________________________________________
```

Figure 10:This the summary of the model which we have built representing all the layer along with order of the input parameters to each layer and also the structure of output obtained from the layer

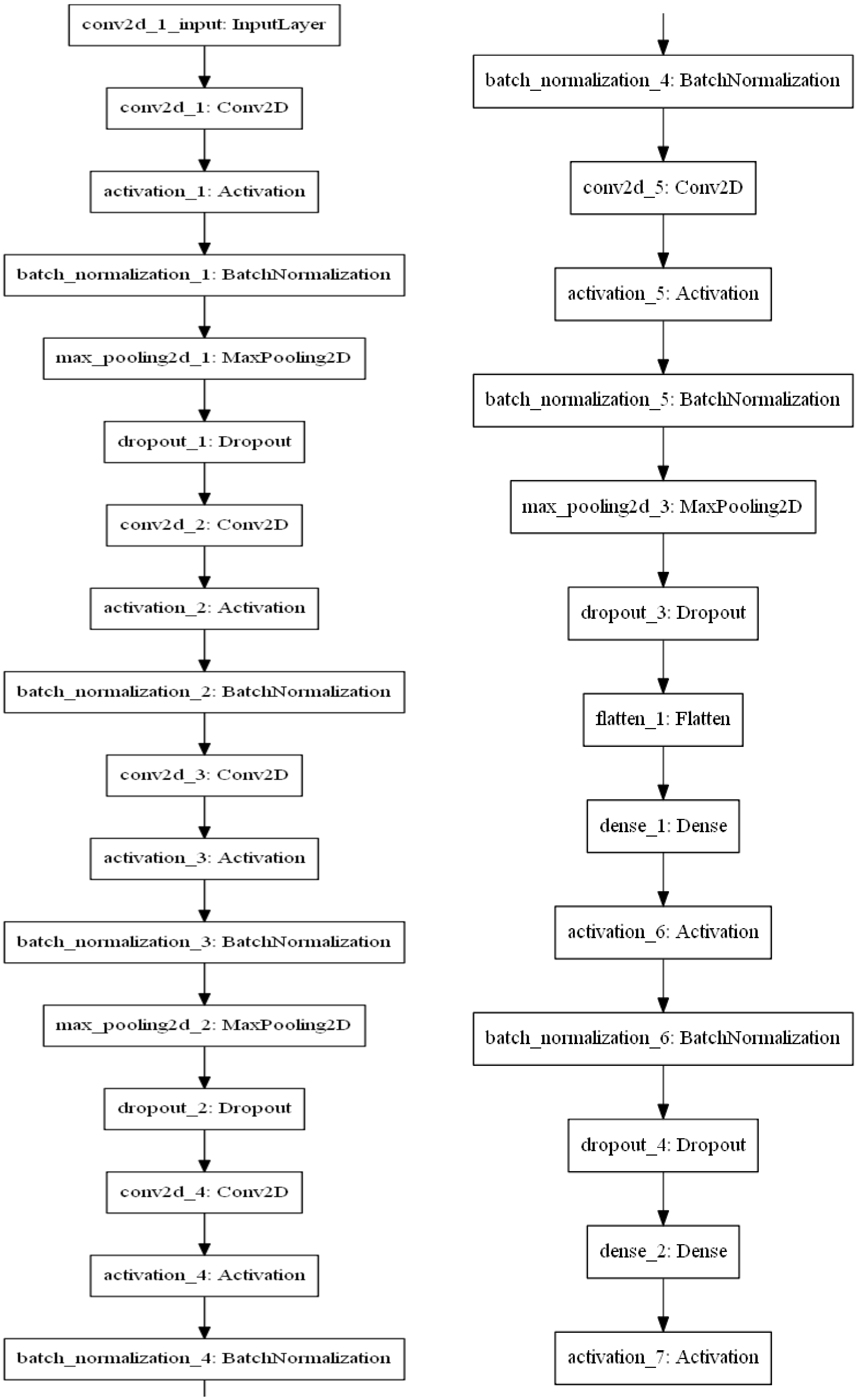
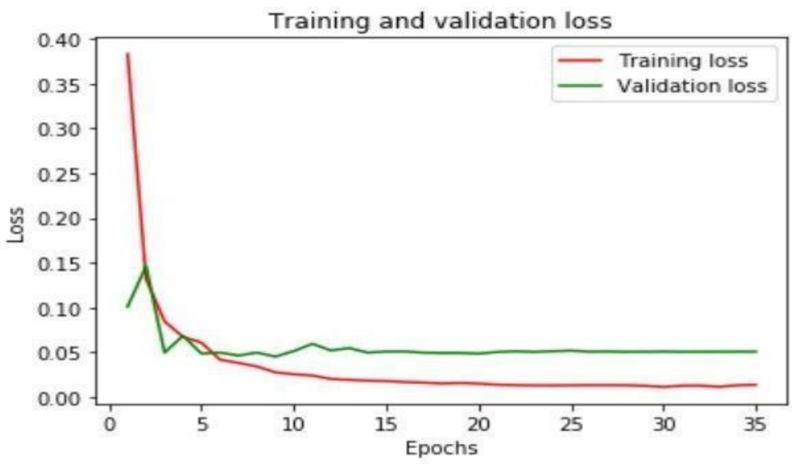

conv2d_1_input: InputLayer
conv2d_1: Conv2D
activation_1: Activation
batch_normalization_1: BatchNormalization
max_pooling2d_1: MaxPooling2D
dropout_1: Dropout
conv2d_2: Conv2D
activation_2: Activation
batch_normalization_2: BatchNormalization
conv2d_3: Conv2D
activation_3: Activation
batch_normalization_3: BatchNormalization
max_pooling2d_2: MaxPooling2D
dropout_2: Dropout
conv2d_4: Conv2D
activation_4: Activation
batch_normalization_4: BatchNormalization
batch_normalization_4: BatchNormalization
conv2d_5: Conv2D
activation_5: Activation
batch_normalization_5: BatchNormalization
max_pooling2d_3: MaxPooling2D
dropout_3: Dropout
flatten_1: Flatten
dense_1: Dense
activation_6: Activation
batch_normalization_6: BatchNormalization
dropout_4: Dropout
dense_2: Dense
activation_7: Activation

Figure 11: The above picture represents the flow of layers of the built model.

4.8 Model Evaluation:

Model Evaluation is an integral part of the model development process. It helps to find the best



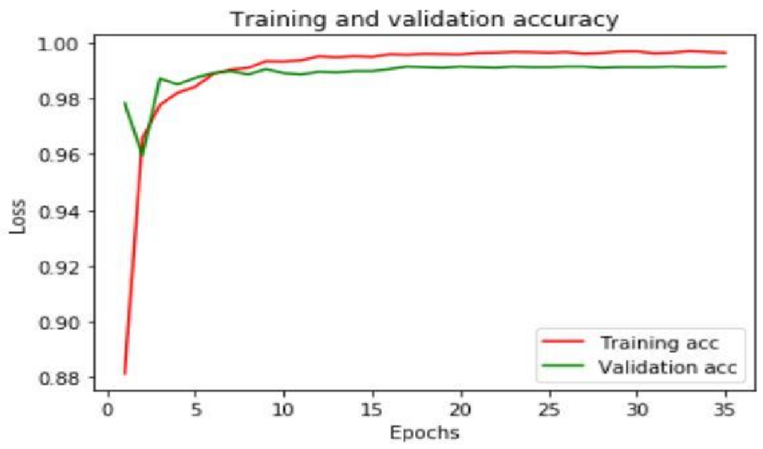


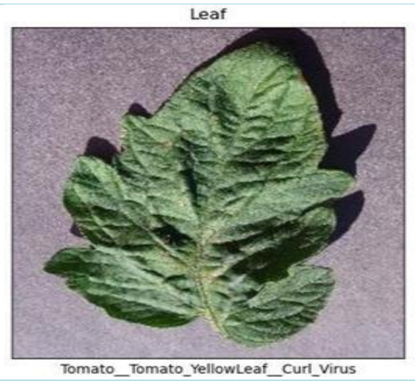
Figure 12 and 13: Graphs of the performance metric

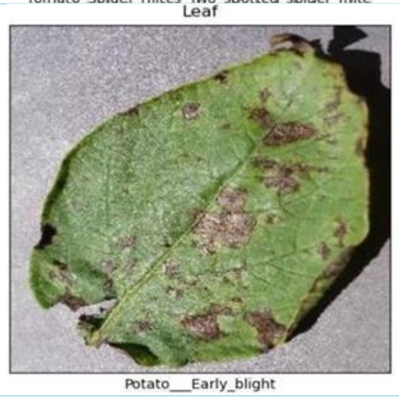
model that represents our data and how well the chosen model will work in the future. … To avoid overfitting, both methods use a test set (not seen by the model) to evaluate model performance. Model evaluation aims to estimate the generalization accuracy of a model on future (unseen/out-of-sample) data. Here we are plotting the graphs of Loss and Accuracy against epochs

4.9 Output Images Obtained On Testing

Table 2: The Test images along with disease depicted have been tabulated

Sno:	Test image given to the model	Disease detected
1)		Pepper_bell_healthy
2)		Tomato_mosaic_virus
3)		Tomato_Target_Spot
4)		Tomato_Septoria_leaf_spot

5)		Tomato_YellowLeaf_Curl_Virus
6)		Pepper_bell Bacterial_spot
7)		Tomato_Target_Spot
8)		Tomato_Bacterial_spot

9)		Potato_Early_blight
10)		Tomato_Spider_mites_Two_spotted-Spider_mite
11)		Tomato_Spider_mites_Two_spotted-Spider_mite
12)		Potato_Early_blight

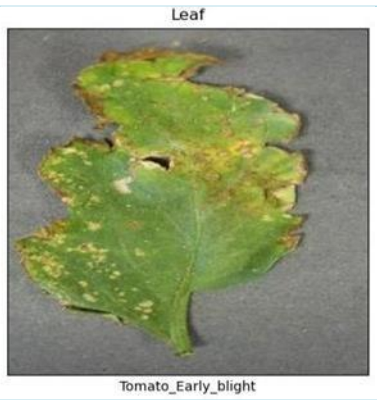
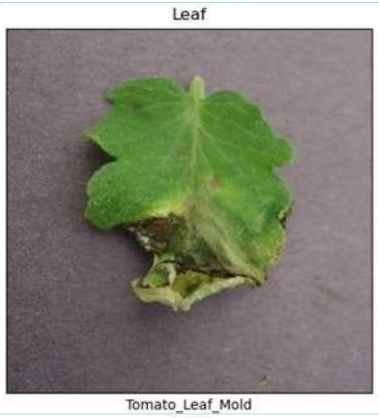
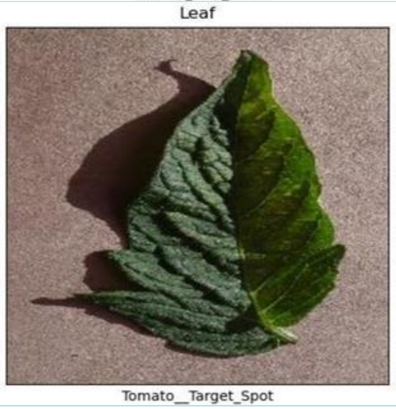
13)		Tomato_Early_blight
14)		Tomato_Leaf_Mold
15)		Tomato_Target_Spot

Figure14-29:The above are some of the random images given to the prediction model and we have received the disease of the leaf as the label given to the input image, and fortunately all the labels which we have got as output are correct.

4.1 Deployment:

The trained models can be deployed to user’s machines (computers, mobiles...etc. and can be used in two modes:

• Diseases classification: a captured image is used as an input of the model, then, the output of the network determines which diseases are present in the plant.

• Symptoms detection and visualization: user can visualize regions that characterize the identified disease. These symptoms visualization methods are very useful for inexperienced users by giving them more information about the diseases alteration on the plant.

Chapter-5
CONCLUSION AND FUTURESCOPE

Conclusion

We studied in this chapter a recent trend for building a system able to detect and classify plant diseases. After analyzing and comparing the previous works based on DL, we have concluded that these studies use principally two CNN architectures (AlexNet and GoogleNet). For this reason, we have evaluated the state-of-the-art CNN architectures using a public dataset of plant diseases. The results of this evaluation show clearly that we can improve the accuracy using a new CNN architectures such as inceptionV3 which achieved an accuracy of 99.76 %. In addition to this improvement in accuracy, we have investigated in increasing the transparency of deep models based on visualization techniques. In this context, the saliency map method is introduced for localizing the infected regions of the plant after the identification of diseases. Despite the fact that the training images are weakly labeled, this visualization has succeeded in the extraction of the infected regions without any expert intervention. Furthermore, this visualization method shows a precise and sharp visualization which helps the inexperienced users to understand the diseases. As a limitation of this study, we can notice that visualization method is not evaluated quantitively using a defined measure. However, the images are assessed based on expert defined visual symptoms. This qualitative evaluation is motivated by the weak labeling of the dataset. Therefore, our future work will focus on the preparation of strong labeled dataset, which makes possible to measure numerically the performance of saliency maps visualization.

FUTURE SCOPE:

This Model can be linked with pesticides dealers and Government. The pesticide dealers can give the best solution to the disease and at same time can promote their products. The Government will take the action if the disease is wide spread in the area and also takes preventing measures in order to stop the spread of the diseases. `

References :

[1] Sachin D. Khirade, A.B Patil, “Plant Disease Detection Using Image Processing”, International Conference on Computing Communication Control and Automation”, 2015.

[2] Mr. Pramod S. landge, Sushil A. Patil, Dhanashree S. Khot, “Automatic Detection and Classification of Plant Disease through Image Processing”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 7, 2013

[3] Brahimi M, Boukhalfa K, Moussaoui A (2017) Deep Learning for Tomato Diseases: Classification and Symptoms Visualization. Applied Artificial Intelligence 31(4):1–17

[4] Hughes D, Marcel Salathe (2015) An open access repository of images on plant health to enable the development of mobile disease diagnostics pp 1–13

[5] Dandawate Y, Kokare R (2015) An automated approach for classification of plant diseases towards development of futuristic Decision Support System in Indian perspective. In: 2015 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2015, IEEE, Kochi,India,pp 794–799

[6] DeChant C, Wiesner-Hanks T, Chen S, Stewart EL, Yosinski J, Gore MA, Nelson RJ, Lipson H (2017) Automated Identification of Northern Leaf Blight Infected Maize Plants from Field Imagery Using Deep Learning. Phytopathology pp PHYTO–11–16–0417–R

[7] Papandreou G, Chen LC, Murphy KP, Yuille AL (2015) Weakly-and semisupervised learning of a deep convolutional network for semantic image segmentation. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp 1742–1750

[8] Shinozaki T (2016) Semi-supervised Learning for Convolutional Neural Networks Using Mild Supervisory Signals, Springer International Publishing, Cham, pp 381–388

[9] Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. CoRR abs/1609.02907

[10] Hanssen IM, Lapidot M (2012) Major Tomato Viruses in the Mediterranean Basin.

Advances in Virus Research 84:31–66

[11] Blancard D (2012) 2 - Diagnosis of Parasitic and Nonparasitic Diseases. Academic Press The Netherlands

[12] Al Hiary H, Bani Ahmad S, Reyalat M, Braik M, ALRahamneh Z (2011) Fast and Accurate Detection and Classification of Plant Diseases. International Journal of Computer Applications 17(1):31–

[13] Zulkifli Bin Husin et al, Plant disease detection using machine learning

[14] Wang K, Zhang D, Li Y, Zhang R, Lin L (2017) Cost-effective active learning for deep image classification. CoRR abs/1701.03551

[15] Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8689

[16] Mrunalini R. Badnakhe, Prashant R. Deshmukh, "Infected LeafAnalysis and Comparison by Otsu Threshold and k-Means Clustering", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue3, March 2012.

ANNEXURE

```
#Code for Training
import numpy as np
import pickle
import cv2
from os import listdir
from sklearn.preprocessing import LabelBinarizer from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation, Flatten, Dropout, Dense from keras import backend as K
from keras.models import Sequential#new
from keras.models import Model#new
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array from sklearn.preprocessing import
MultiLabelBinarizer from sklearn.model_selection import train_test_split from keras.utils import
plot_model
import matplotlib.pyplot as plt
import os
EPOCHS = 25
INIT_LR =
1e-3 BS = 32
default_image_size = tuple((256, 256))
image_size = 0
directory_root = 'C:/Users/sreecharanksvn/AppData/Local/Programs/Python/Python37/Testing'
width=256
height=256
depth=3
```

```
def convert_image_to_array(image_dir):
    try:
        image = cv2.imread(image_dir)
        if image is not None :
            image = cv2.resize(image, default_image_size)
            return img_to_array(image)
        else :

            return np.array([])
     except Exception as e:
         print(f"Error : {e}")
        return None
image_list, label_list = [],
[] try:
    print("[INFO] Loading images
    ...") root_dir =
    listdir(directory_root) for
    directory in root_dir :
        # remove .DS_Store from list
         if directory == ".DS_Store" :
             root_dir.remove(directory)



    for plant_folder in root_dir :

        plant_disease_folder_list = listdir(f"{directory_root}/{plant_folder}")
        for disease_folder in plant_disease_folder_list :
            # remove .DS_Store from list
            if disease_folder == ".DS_Store" :
                plant_disease_folder_list.remove(disease_folder)


        for plant_disease_folder in plant_disease_folder_list:
            print(f"[INFO] Processing {plant_disease_folder} ...")
            plant_disease_image_list                                                                                     =
listdir(f"{directory_root}/{plant_folder}/{plant_disease_folder}/")


            for single_plant_disease_image in plant_disease_image_list :
                if single_plant_disease_image == ".DS_Store" :
                    plant_disease_image_list.remove(single_plant_disease_image)


            for image in plant_disease_image_list[:200]:

                image_directory = f"{directory_root}/{plant_folder}/{plant_disease_folder}/{image}"

                if image_directory.endswith(".jpg") == True or image_directory.endswith(".JPG") ==

True
:                   image_list.append(convert_image_to_array(image_directory)) label

                    _list.append(plant_disease_folder)
    print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")
```

```
image_size = len(image_list)
label_binarizer=LabelBinarizer()
image_labels = label_binarizer.fit_transform(label_list)
pickle.dump(label_binarizer,open('label_transform.pkl', 'wb'))
n_classes = len(label_binarizer.classes_)
print(n_classes)
print(label_binarizer.classes_)
np_image_list = np.array(image_list, dtype=np.float16) / 225.0
print("[INFO] Spliting data to train, test")
x_train, x_test, y_train, y_test = train_test_split(np_image_list, image_labels, test_size=0.2,
random_state = 42)

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")

model = Sequential()

inputShape = (height, width, depth)
chanDim = -1
if K.image_data_format() ==
   "channels_first": inputShape = (depth,
   height, width) chanDim = 1
model.add(Conv2D(32, (3, 3), padding="same",input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())

model.add(Dense(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(n_classes))
model.add(Activation("softmax"))
model.summary()
plot_model(model,to_file='model.png')
opt=Adam(lr=INIT_LR,decay=INIT_LR/EPOCH
S) # distribution
model.compile(loss="binary_crossentropy", optimizer=opt,metrics=["accuracy"])
# train the network
print("[INFO] training network...")
```

```
history = model.fit_generator(
   aug.flow(x_train, y_train,
   batch_size=BS), validation_data=(x_test,
   y_test), steps_per_epoch=len(x_train) //
   BS, epochs=EPOCHS, verbose=1
   )

acc = history.history['accuracy']

val_acc =history.history['val_accuracy'] loss = history.history['loss']
val_loss =history.history['val_loss']
epochs = range(1, len(acc) + 1)
#Train and validation accuracy
plt.plot(epochs, acc, 'b', label='Training accurarcy')
plt.plot(epochs, val_acc, 'r', label='Validation accurarcy')
plt.title('Training and Validation accurarcy')
plt.legend()
plt.figure()
#Train and validation loss
plt.plot(epochs, loss, 'b', label='Training loss') plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()
print("[INFO] Calculating model accuracy")
scores = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {scores[1]*100}")
# save the model to disk
print("[INFO] Saving model...")
pickle.dump(model,open('cnn_model.pkl', 'wb'))
```

```
#Code for Testing:

from PIL import Image
import numpy as np
import pickle
import cv2

import matplotlib.image as iplot
import matplotlib.pyplot as plt
from os import listdir
from sklearn.preprocessing import LabelBinarizer from keras.preprocessing.image import img_to_array
default_image_size = tuple((256, 256))
image_size = 0

directory_root = 'C:/Users/sreecharanksvn/AppData/Local/Programs/Python/Python37/Testing'
def convert_image_to_array(image_dir):
  try:

    image = cv2.imread(image_dir)
    if image is not None :
      image = cv2.resize(image, default_image_size)
      return img_to_array(image)
    else :

      return np.array([])
  except Exception as e:
    print(f"Error : {e}")
```

```
        return None
file_object                                                                                    =
open('C:/Users/sreecharanksvn/AppData/Local/Programs/Python/Python37/cnn.pkl', 'rb') model
= pickle.load(file_object)


fp=open('C:/Users/sreecharanksvn/AppData/Local/Programs/Python/Python37/label_transfor
m.pkl', 'rb')
fodel=pickle.load(fp)
imgpath='C:/Users/sreecharanksvn/Desktop/brit/3.JPG'
img = Image.open(imgpath)
imar = convert_image_to_array(imgpath)
npimagelist = np.array([imar], dtype=np.float16) / 225.0
PREDICTEDCLASSES2 =
model.predict_classes(npimagelist)
f=fodel.classes_[PREDICTEDCLASSES2]
l=str(f[0])
#print(type(l))
plt.imshow(iplot.imread(imgpath))
plt.title('Leaf')
plt.xlabel(l)
plt.xticks([]
)
plt.yticks([]
) plt.show()
```