Visvesvaraya Technological University, Belagavi
Government Engineering College, Hassan 573 201

Project report On

Detection and Classification of Tomato Plant Disease

4GH15CS001	Amith Satyanarayan
4GH15CS039	Navya C R
4GH15CS045	Rachana N S
4GH15CS060	Vanishree R R

Under the Guidance of

Mr.M T ThirtheGowda$_{B.E.,M.Tech.,MISTE.}$
Assistant Professor
Dept. of Computer Science & Engineering
Government Engineering College, Hassan

Department of Computer Science & Engineering
Government Engineering College, Hassan
June 2018-19

Government Engineering College, Hassan-573 201
Visvesvaraya Technological University, Belagavi

Certificate

This is to certify that the project work entitled **"Detection and Classification of Tomato Plant Disease"** is a bonafide work carried out by **Amith Satyanarayan (4GH15CS001), Navya C R (4GH15CS0039), Rachana N S (4GH15CS0045), Vanishree R R (4GH15CS0060)** in partial fulfillment of the award of the degree of Bachelor of Engineering in Computer Science Engineering of Visvesvaraya Technological University, Belagavi, during the year 2018-19. It is certified that all corrections / suggestions indicated during internal evaluation have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

Guide

Mr.M T ThirtheGowda

B.E.,M.Tech.,MISTE.

Asst. Professor

Dept of CS & E, GEC, Hassan

Head of the Department

Mr.Chethan K C *B.E.,M.Tech.,*

Head of the Dept, CS & E

GEC, Hassan

Principal

Dr.K C Ravishankar *B.E.,M.Tech.,Ph.D.*

Principal

GEC, Hassan

Date :

Place : Hassan

Acknowledgement

At the outset we express my most sincere grateful thanks to my **Guide, Mr. M T ThirtheGowda, Assistant Professor, Department of CS & E**, for his continuous support and advice not only during the course of the project but also during the period of my stay in GECH.

We express my gratitude to **Mr. Ranganatha S**, Project Co-ordinator, Assistant Professor ,CS & E, for his encouragement and support throughout the work.

We express my gratitude to **Mr. Chethan K C**,Head of the Department,CS & E, for his encouragement and support throughout the work.

We wish to express thanks to our beloved **Principal**, **Dr. K C Ravishankar**, for encouragement throughout my studies.

Finally, We express my gratitude to all teaching and non-teaching staff of Department of CS & E, my fellow classmates and my parents for their timely support and suggestions.

Amith Satyanarayan (4GH15CS001)

Navya C R (4GH15CS039)

Rachana N S (4GH15CS045)

Vanishree R R (4GH15CS060)

Table of Contents

List of Figures

Abstract

Plant leaf diseases and destructive insects are a major challenge in the agriculture sector. Faster and an accurate prediction of leaf diseases in crops could help to develop an early treatment technique while considerably reducing economic losses. Modern advanced developments in Deep Learning have allowed researchers to extremely improve the performance and accuracy of object detection and recognition systems. A deep-learning-based approach to detect leaf diseases in tomato plants using images of plant leaves is proposed. Therefore, we proposed Faster Region-based Convolutional Neural Network (Faster R-CNN) to accomplish the recognition system.

The proposed system can effectively identify different types of diseases with the ability to deal with complex scenarios from the plants area. The proposal of an analysis to determine which kind of images with different color spectrum provide better information to generate a better accuracy. Experimental results on images with different diseases from Tomato disease dataset show that each disease contains valuable information in the infected part of the leaf that responds differently to other uninfected parts of the plant. Then the infected leaf will be used for the classification.

Chapter 1

Introduction

Agriculture is the mainstay of the Indian economy. Immense commercialization of an agriculture has created a very negative effect on our environment. The use of chemical pesticides has led to enormous levels of chemical buildup in our environment, in soil, water, air, in animals and even in our own bodies. Artificial fertilizers give on a short-term effect on productivity but a longer- term negative effect on the environment, where they remain for years after leaching and running of, contaminating ground water. Another negative effect of this trend has been on the fortunes of the farming communities worldwide. Despite this so-called increased productivity, farmers in practically every country around the world have seen a downturn in their fortunes. This is where organic farming comes in. Organic farming has the capability to take care of each of these problems. The central activity of organic farming relies on fertilization, pest and disease control. Plant disease detection through naked eye observation of the symptoms on plant leaves, incorporate rapidly increasing of complexity. Due to this complexity and to the large number of cultivated Crops and their existing Phyto pathological problems, even experienced agricultural experts and plant pathologists may often fail to successfully diagnose specific diseases, and are consequently led to mistaken conclusions and concern solutions. An automated system designed to help identify plant diseases by the plants appearance and visual symptoms could be of great help to amateurs in the agricultural process. This will be proved as useful technique for farmers and will alert them at the right time before spreading of the disease over large area.

Deep learning constitutes a recent, modern technique for image processing and data analysis, with accurate results and large potential. As deep learning has been successfully applied in various domains, it has recently entered also the domain of agriculture. So, we will apply deep learning to create an algorithm for automated detection and classification of plant leaf diseases. Nowadays, Convolutional Neural Networks are considered as the leading method for object detection. Detectors namely

Faster Region-Based Convolutional Neural Network (Faster R-CNN), Region-based Fully Convolutional Networks (R-FCN) and Single Shot Multibox Detector (SSD) can be used. Each of the architecture should be able to be merged with any feature extractor depending on the application or need. The early detection of plant leaf diseases could be a valuable source of information for executing proper diseases detection, plant growth management strategies and disease control measures to prevent the development and the spread of diseases.

Plants are susceptible to several disorders and attacks caused by diseases. There are several reasons that can be characterizable to the effects on the plants, disorders due to the environmental conditions, such as temperature, humidity, nutritional excess or losses, light and the most common diseases that include bacterial, virus, and fungal diseases. Those diseases along with the plants may shows different physical characteristics on the leaves, such as a change in shapes, colors etc. Due to similar patterns, those above changes are difficult to be distinguished, which makes their recognition a challenge, and an earlier detection and treatment can avoid several losses in the whole plant. We use recent detector Faster Region-Based Convolutional Neural Network (Faster R-CNN) to detection and classification of plant leaf diseases that affect in tomato plants.

1.1 Image Processing

A digital image is composed of pixels which can be thought of as small dots on the screen. A digital image is an instruction of how to color each pixel. A typical size of an image is 512-by-512 pixels. It is convenient to let the dimensions of the image to be a power of 2. For example, 2 power 9 equals 512.Image processing is a method of converting an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Usually, Image Processing system includes treating images as two dimensional signals while applying already set signal processing methods to them.

It is among rapidly growing technologies today, with its applications in various aspects of a business. Image Processing forms core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps:

a) Importing the image with optical scanner or by digital photography.

b) Analyzing and manipulating the image which includes data compression and image enhancement and spotting patterns that are not to human eyes like satellite

photographs.

c) Output is the last stage in which result can be altered image or report that is based on image analysis.

Image processing is a very important subject, and finds applications in such fields as photography, satellite imaging, medical imaging, and image compression, just to name a few. In the past, image processing was largely done using analog devices. However, as computers have become more powerful, processing shifted toward the digital domain. Like one-dimensional digital signal processing, digital image processing overcomes traditional analog problems such as noise, distortion during processing, inflexibility of system to change, and difficulty of implementation.

Nowadays, image-processing technologies are frequently combined with machine learning approaches, particularly supervised learning, such as the k-Nearest Neighbors, Support Vector Machine and artificial neural network. The accuracy of such approaches strongly depends on the training dataset. With regard to pixel-based segmentation in our method, training images must cover the full range of information of the targeted parts to be segmented so that accurate results can be obtained. The same can be said of blob-based segmentation. Furthermore, since the developed method requires two types of training datasets for both the pixel-based and blob-based segmentations, the preparation of the training images is a time-consuming task that may prove to be a bottleneck for the proposed method. Semi-automatic preparation of training data for blob-based segmentation using a K-means algorithm, which is used in related research, was also applied. However, this approach was observed to be inefficient for our method because different types of blobs were classified into the same clusters.

1.2 Related works

To analyze the works of the state-of-the-art, we need to give the general architecture of disease classification systems that are based on image proces- sing. These systems contain three phases: image pre-processing, feature extraction, and classification

Pre-processing

The image of a leaf is prepared using some operations. For instance, color space conversion from RGB (Red-Green-Blue) to another space is used for reducing the dependence on device.Also, many works try to remove the background, focusing the analysis on the leaf.Unfortunately, removing back- ground is difficult, and sometimes needs the intervention of the user, which decreases the automation of the system.

Feature extraction

Features proposed by experts (hand-crafted features) are extracted from the image for constructing feature vectors. For example, color moments are used to extract color statistics, in which Gabor Transform (GT) and Wavelet Transform (WT) are combined (GWT) for the extraction of multiscale features. Gray Level Co- occurrence Matrix (GLCM) is used in many previous works.to extract texture features. GLCM is a 256*256 matrix where each position in the matrix counts the co-occurrences of line color and column color in the analyzed image, Scale Invariant Feature Transform (SIFT) is used to analyze the shape features of leaves.

Methodology

The proposed approach, illustrated in Figure contains the four components given below:

1. Pre-taining phase: in this phase we train deep architectures on a large dataset like ImageNet using powerful machines. The objective of this phase is the initialization of network weights for the next phase.

2. Training (fine-tuning): we fine-tune the resulted network from the first phase. Also, we replace the output layer of the pre-tained net- works by a new output layer having nine classes.Afterword, the developed deep model is deployed to users machines (Computers, mobiles)

3. Disease classification: in this mode, the user takes a picture of a leaf and use the produced deep model to determine the disease that affects the tomato plant.

4. Symptom detection and visualization: after the disease classification, the user can visualize the regions of leaf image characterizing the disease (Symptoms of disease). This symptom visualization method helps the inexperienced user by giving them more information about the disease mechanism. Also, symptom visualization gives the user a tool to estimate the spread of disease in the other tomato plants.

AlexNet vs. GoogleNet

The results of GoogleNet network overcomes the results of AlexNet in plant disease classification. Although the size of GoogleNet is small (36.6 MB) compared to AlexNet (201 MB), the results of GoogleNet are more accurate than AlexNet results. The accuracy of pre-trained GoogleNet is 99.185 and macro F1 is 98.518 while AlexNet has an accuracy equal to 98.660 and macro F = 97.911. This superiority of GoogleNet is due to the new architecture that is used in this network to increase nonlinearity without an explosion of the number of weights (Szegedy et al. 2015). GoogleNet uses inception module that is inspired from the architecture Network in Network proposed in (Lin, Chen, and Yan 2014). Inception module uses convolutions with filters one by one (1*1), leading to decreases in the depth of input volume. Moreover, the number

of weights decreases, without losing much information. In other words, con- volution using filters of size 1*1 plays the role of filtering information along the layers and this reduces the size of the network.

1.3 Machine Learning

The name machine learning was coined in 1959 by Arthur Samuel.Machine learning explores the study and construction of algorithms that can learn from and make predictions on data such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions,through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or infeasible; example applications include email filtering, detection of network intruders, and computer vision.

Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data mining, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning.

Within the field of data analytics, machine learning is a method used to devise complex models and algorithms that lend themselves to prediction; in commercial use, this is known as predictive analytics. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data.

Machine learning tasks are typically classified into several broad categories :

- **Supervised learning**
  computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs. As special cases, the input signal can be only partially available, or restricted to special feedback.

- **Semi-supervised learning**
  The computer is given only an incomplete training signal: a training set with some (often many) of the target outputs missing.

- **Active learning**
  The computer can only obtain training labels for a limited set of instances (based on a budget), and also has to optimize its choice of objects to acquire labels for. When used interactively, these can be presented to the user for labeling.

- **Unsupervised learning**
  No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

- **Reinforcement learning**
  Data (in form of rewards and punishments) are given only as feedback to the program's actions in a dynamic environment, such as driving a vehicle or playing a game against an opponent.

Machine learning applications are:

- In classification, inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".

- In regression, also a supervised problem, the outputs are continuous rather than discrete.

- In clustering, a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.

- Density estimation finds the distribution of inputs in some space.

- Dimensionality reduction simplifies inputs by mapping them into a lower-dimensional space. Topic modeling is a related problem, where a program is given a list of human language documents and is tasked to find out which documents cover similar topics.

1.3.1 Deep Learning

Deep learning (also known as deep structured learning or hierarchical learning)

It is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised.

Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design and board game programs, where they have produced results comparable to and in some cases superior to human experts.

Deep learning models are vaguely inspired by information processing and communication patterns in biological nervous systems yet have various differences from the structural and functional properties of biological brains (especially human brain), which make them incompatible with neuroscience evidences.

Deep learning is a class of machine learning algorithms that:

- use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.
- learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners.
- learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.

ImageNet

It is an image dataset organized according to the WordNet hierarchy. The ImageNet project is a large visual database designed for use in visual object recognition software research. More than 14 million images have been hand-annotated by the project to indicate what objects are pictured and in at least one million of the images, bounding boxes are also provided. ImageNet contains more than 20,000 categories with a typical category, such as "balloon" or "strawberry", consisting of several hundred images. The database of annotations of third-party image URLs is freely available directly from ImageNet, though the actual images are not owned by ImageNet. Since 2010, the ImageNet project runs an annual software contest, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), where software programs compete to correctly classify and detect objects and scenes. The challenge uses a "trimmed" list of one thousand non-overlapping classes.

1.3.2 Artificial neural network

An artificial neuron network (ANN) is a computational model based on the structure and functions of biological neural networks. Information that flows through the network affects the structure of the ANN because a neural network changes - or learns, in a sense - based on that input and output.

ANNs are considered nonlinear statistical data modeling tools where the complex relationships between inputs and outputs are modeled or patterns are found.

ANN is also known as a neural network.

In our project, we are using three main families of detectors which comes under ANN:

A. Faster Region-Based Convolutional Neural Network (Faster R-Cnn):

Faster R-CNN is one of the Object detection systems, which is composed of two modules. The first module is a deep fully convolutional network that proposes regions. For training the RPNs, the system considers anchors containing an object or not, based on the Intersection-over-Union (IoU) between the object proposals and the ground-truth. Then the second module is the Fast R-CNN detector, that uses the proposed regions. Box proposals are used to crop features from the same intermediate feature map which are subsequently fed to the remainder of the feature extractor in order to predict a class and class-specific box refinement for each proposal. The entire process happens on a single unified network, which allows the system to share full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals.

B. Region-Based Fully Convolution Network(R-FCN):

We develop a framework called Region-based Fully Convolutional Network (R-FCN) for object detection. While Faster R-CNN is an order of magnitude faster than Fast R- CNN, the fact that the region-specific component must be applied several hundred times per image led to propose the R-FCN (Region-based Fully Convolutional Networks) method which is like Faster R-CNN, but instead of cropping features from the same layer where region proposals are predicted, crops are taken from the last layer of features prior to prediction. R-FCN object detection strategy consists of: (i) region proposal, and (ii) region classification. This approach of pushing cropping to the last layer minimizes the amount of per-region computation that must be done. The object detection task needs localization representations that respect translation variance and thus propose a position-sensitive cropping mechanism that is used instead of the more standard ROI pooling operations used in object detection. They show that the R FCN model could achieve comparable accuracy to Faster R-CNN often at faster running times.

C. Single Shot Detector (Ssd):

The SSD approach is based on a feed-forward convolutional network that produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes, followed by a non-maximum suppression step to produce the final detections. This network is able to deal with objects of various sizes by combining predictions from multiple feature maps with different resolutions. Furthermore, SSD encapsulates the process into a single network, avoiding proposal generation and thus saving computational time.

D. Data Collection:

Dataset contains images with several diseases in many different plants. In this System we can consider some of the commercial/cash crops,cereal crops, and vegetable crops and fruit plants such as sugarcane, cotton, potato, carrot, chilly, brinjal, rice, wheat, banana and guava. Diseased leaves, healthy leaves all of them were colleted for those above crops from different sources like images download from Internet, or simply taking pictures using any camera devices or any else.

E. Image Pre-Processing:

Image annotation and augmentation, the task of automatically generating description words for a picture, is a key component in various image search and retrieval application. But in this system, we manually annotate the areas of every image containing the disease with a bounding box and class. Some diseases might look similar depending on its infection status.

1.4 Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community after 30 years.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

PIP

files are preference files that store custom menu and toolbar settings for the Windows version of Microsoft Office. Custom menus and toolbars are generated automatically by Office programs based on what commands and options are used most frequently. They can also be edited manually by selecting the the Customize Toolbars option.

TensorFlow

It is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google, often replacing its closed-source predecessor, DistBelief.

Keras

It is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, or Theano. 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 Franois Chollet, a Google engineer.

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. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel.

Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep learning models on clusters of Graphics Processing Units (GPU) and Tensor processing units (TPU).

Chapter 2

Libraries and Tools

2.1 Anaconda

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. The open-source Anaconda Distribution is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 11 million users worldwide, it is the industry standard for developing, testing, and training on a single machine. Jupyter The Jupyter Notebook is an open-source web application that allows to create and share documents that contain live code, equations, visualizations, and narrative text. Uses include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

2.2 Jupyter

Jupyter has developed and supported the interactive computing products Jupyter Notebook, Jupyter Hub, and Jupyter Lab, the next-generation version of Jupyter Notebook. The Jupyter Notebook application allows to create and edit documents that display the input and output of a Python or R language script.JupyterLab is the next-generation user interface for Project Jupyter. It offers all the familiar building blocks of the classic Jupyter Notebook (notebook, terminal, text editor, file browser, rich outputs, etc.) in a flexible and powerful user interface.

2.3 Keras

Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, or Theano. 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 Francois Chollet, a Google engineer.

They 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. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel.

Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep learning models on clusters of Graphics Processing Units (GPU) and Tensor processing units (TPU).

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

2.4 TensorFlow

TensorFlow is an open source software library released in 2015 by Google to make it easier for developers to design, build, and train deep learning models. TensorFlow originated as an internal library that Google developers used to build models in-house, and we expect additional functionality to be added to the open source version as they are tested and vetted in the internal flavor.

At a high level, TensorFlow is a Python library that allows users to express arbitrary computation as a graph of data flows. Nodes in this graph represent mathematical operations, whereas edges represent data that is communicated from one node to another. Data in TensorFlow are represented as tensors, which are multidimensional arrays.

2.5 Labelme

Labelme is a graphical image annotation tool, It is written in Python and uses Qt for its graphical interface.

2.6 NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on the arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.Numpy provides a high-performance multidimensional array and basic tools to compute with and manipulate these arrays. SciPy builds on this and provides a large number of functions that operate on numpy arrays and are useful for different types of scientific and engineering applications.

2.7 Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

2.8 Conv2D

This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If use_bias is True, a bias vector is created and added to the outputs. Finally, if activation is None, it is applied to the outputs as well. When using this layer as the first layer in a model, provide the keyword argument input_shape (tuple of integers, does not include the batch axis.

2.9 MaxPooling2D

Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned. This is done in part to help over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation. Max pooling is done by applying a max filter to (usually) non-overlapping subregions of the initial representation. Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned. This is done in part to help over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation. Max pooling is done by applying a max filter to (usually) non-overlapping subregions of the initial representation.

2.10 Flatten

A flatten operation on a tensor reshapes the tensor to have a shape that is equal to the number of elements contained in the tensor.

2.11 Dense

A dense layer is a classic fully connected neural network layer : each input node is connected to each output node. A dropout layer is similar except that when the layer is used, the activations are set to zero for some random nodes.

2.12 VGG

VGG stands for the Visual Geometry Group from the University of Oxford.VGG16 is a convolutional neural network architecture. VGGNet consists of 16 convolutional layers and is very appealing because of its very uniform architecture. It is currently the most preferred choice in the community for extracting features from images. The weight configuration of the VGGNet is publicly available and has been used in many other applications and challenges as a baseline feature extractor. However, VGGNet

consists of 138 million parameters, which can be a bit challenging to handle. VGG-19 is a convolutional neural network that is trained on more than a million images from the ImageNet database. The network is 19 layers deep and can classify images into 1000 object categories.

Chapter 3

Requirement Specification

A System Requirements Specification (SRS) (also known as a Software Requirements Specification) is a document or set of documentation that describes the features and behaviour of a system or software application. It includes a variety of elements (see below) that attempts to define the intended functionality required by the customer to satisfy their different users.

Depending on the methodology employed the level of formality and detail in the SRS will vary, but in general a SRS should include a description of the SRS will vary, but in general a SRS should include a description of the functional requirements, system requirements, constraints, assumptions and acceptance criteria.

3.1 Functional Requirements

Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish. Functional requirements are supported by non-functional requirements (also known as "quality requirements"), which impose constraints on the design or implementation (such as performance requirements, security, or reliability).

Generally, functional requirements are expressed in the form "system must do requirement". Functional requirements specify particular results of a system. This should be contrasted with non-functional requirements, which specify overall characteristics such as cost and reliability. Functional requirements drive the application architecture of a system.

3.2 Non-functional Requirements

A non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions.The plan for implementing non-functional requirements is detailed in the system architecture, because they are usually architecturally significant requirements.

Non-functional requirements are in the form of "system shall be ¡requirement¿", an overall property of the system as a whole or of a particular aspect and not a specific function. The system's overall properties commonly mark the difference between whether the development project has succeeded or failed.Non-functional requirements are often called "quality attributes" of a system. Other terms for non-functional requirements are "qualities", "quality goals", "quality of service requirements", "constraints", "non-behavioral requirements", or "technical requirements".

3.3 Hardware Requirements

Processor : Intel CORE i5 or higher
RAM : 4GB RAM
Monitor : EGVGA Compatible
Keyboard : Normal keyboard(QWERTY)
Hard Disk : 500GB or higher recommended

3.4 Software Requirements

Operating system : GNU/Linux.
Language : Python.
Libraries : Python Libraries, Keras Libraries, TensorFlow
IDE : Anaconda.
Tools: : Labelme.

Chapter 4

Objectives

Plant diseases cause a major production and economic losses in the agricultural industry. The disease management is a challenging task. Usually the diseases or its symptoms such as colored spots or streaks are seen on the leaves of a plant. In plants most of the leaf diseases are caused by fungi, bacteria, and viruses. The diseases caused due to these organisms are characterized by different visual symptoms that could be observed in the leaves or stem of a plant. Usually, these symptoms are detected manually. Automatic detection of various diseases can be detected with the help of image processing. Image processing plays a crucial role in the detection of plant diseases since it provides best results and reduces the human efforts. The image processing could be used in the field of agriculture for several applications. It includes detection of diseased leaf, stem or fruit, to measure the affected area by disease, to determine the color of the affected area. Tomato cultivation is one of the most remunerative farming enterprises in India. The naked eye observation by the experts is approach usually taken in identification and detection of plants. This approach is time consuming in huge farms or land areas. The use of image processing techniques in detection and identification of tomato plant diseases in the earlier stages and thereby the quality of the product could be increased. These systems monitor the plant such as leaves and stem and any variation observed from its characteristic features, variation will be automatically identified and also will be informed to the user.

Chapter 5

Literature Survey

Here, we take some of the papers related to Plant leaf diseases detection using various advancedtechniques and some of them shown below,

In paper [1], author described as an in-field automatic wheat disease diagnosis system based on a weekly supervised deep learning framework, i.e. deep multiple instance learning, which achieves an integration of identification for wheat diseases and localization for disease areas with only image-level annotation for training images in wild conditions. Furthermore, a new infield image dataset for wheat disease, Wheat Disease Database 2017 (WDD2017), is collected to verify the effectiveness of our system.nUnder two different architectures, i.e. VGG-FCNVD16 and VGG-FCN-S, our system achieves the mean recognition accuracies of 97.95% and 95.12% respectively over 5-fold cross validation on WDD2017, exceeding the results of 93.27% and 73.00% by two conventional CNN frameworks, i.e. VGG-CNN-VD16 and VGG-CNN-S. Experimental results demonstrate that the proposed system outperforms conventional CNN architectures on recognition accuracy under the same amount of parameters, meanwhile maintaining accurate localization for corresponding disease areas. Moreover, the proposed system has been packed into a Realtime mobile app to provide support for agricultural disease diagnosis.

In paper [2], author discussed and to perform a survey of 40 research efforts that employ deep learning techniques, applied to various agricultural and food production challenges. Examine the particular agricultural problems under study, the specific models and frameworks employed the sources, nature and pre-processing of data used, and the overall performance achieved according to the metrics used at each work under study. Moreover, study comparisons of deep learning with other existing popular techniques, in respect to differences in classification or regression performance. Findings indicate that deep learning provides high accuracy, outperforming existing

commonly used image processing techniques.

In paper [3], author discussed about convolutional neural network models were developed to perform plant disease detection and diagnosis using simple leaves images of healthy and diseased plants, through deep learning methodologies. Training of the models was performed with the use of an open database of 87,848 images, containing 25 different plants in a set of 58 distinct classes of [plant, disease] combinations, including healthy plants. Several model architectures were trained, with the best performance reaching a 99.53% success rate in identifying the corresponding [plant, disease] combination (or healthy plant). The significantly high success rate makes the model a very useful advisory or early warning tool, and an approach that could be further expanded to support an integrated plant disease identification system to operate in real cultivation conditions.

In paper [4] author describes a methodology for early and accurately 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 91classifies different plant diseases and uses the combination of textures, color and features to recognize those diseases.

In paper [5] authors presented disease detection in Malus domestica through an effective method like K-mean clustering, texture and color analysis. To classify and recognize different agriculture, it uses the texture and color features those generally appear in normal and affected areas.

In paper [6] authors compared the performance of conventional multiple regression, artificial neural network (back propagation neural network, generalized regression neural network) and support vector machine (SVM). It was concluded that SVM based regression approach has led to a better description of the relationship between the environmental conditions and disease level which could be useful for disease management.

In paper [7] authors described disease detection, in which image processing is first step for obtaining image in digital form and pre-processing to remove noise and other object from image. Pre-processing also convert RGB images into grey images using equation f(x) = 0.2989*R + 0.5870*G + 0.114*B and makes histogram equalization. Image segmentation is done using boundary and spot detection algorithms for finding infected part of leaf. Classifications of objects are done using K-means

clustering method. Otsu threshold algorithm is used for thresholding which creates binary images from grey images. With the help of feature extraction color, texture, morphology, edges are used in plant disease detection. Leaf color extraction using H B components and Color co-occurrence method are feature extraction methods in image processing. Classifications of diseases are done using artificial neural network (ANN) and Back propagation network.

In paper [8] authors used both LABVIEW and MATLAB software for image processing to detect chili plant disease. This combined technique detects disease through leaf inspection in early stage. The Image is captured using LABVIEW IMAQ Vision and MATLAB is used for further operations of image processing. Image pre-processing operations are Fourier filtering, edge detection and morphological operations. In feature extractions, the color clustering is used to distinguish between chili and non-chili leaves. Then image recognition and classification determine the healthiness of each chili plant. This technique results in reducing use of harmful chemicals for chili plant which reduces production cost and increases high quality of chili.

In paper [9] authors present image-processing technique for Leaf stem disease detection. The author used a set of leaf images from Jordans Al -Ghor area. The five plant diseases namely: Early scorch, Ashen mold, Late scorch, Cottony mold and Tiny whiteness is tested by image processing technique. In this technique at starting, image acquisition is obtained and then K-Means clustering method is used for segmentation. After that in feature extraction, CCM (Colour Co-occurrence Method) is used for texture analysis of infected leaf and stem. Lastly paper presents Back propagation algorithm for neural network in classification of plant diseases. Result of this image processing technique shows accurate detection and classification of plant diseases with high precision around 93

In paper [10] authors present image processing technique for Rice disease identification and considered the two most common diseases in the north east India, namely Leaf Blast (Magnaporthe Grisea) and Brown Spot (Cochiobolus Miyabeanus). Image acquisition is basic step, after that author use segmentation, boundary detection and spot detection method for feature extraction of the infected parts of the leave. In this paper author introduces zooming algorithm in which SOM (Self Organising Map) neural network is used for classification diseased rice images. There are two methods to make input vector in SOM. First method is the padding of zeros and the second method is the interpolation of missing points. For fractional zooming to normalize the spots size, interpolation method is applied. Image transformation in frequency domain does not give better classification. For testing purposes, four different types of images are applied; the zooming algorithm gives satisfactory results of classification for test images.

In paper [11] authors present image processing technique for detecting the Malus Domestica leaves disease. Intensity values of grayscale images are obtained by histogram equalization method. In image segmentation, Co-occurrence matrix method algorithm is used for texture analysis and K-means clustering algorithm is used for color analysis. Texture analysis is characterization of regions in an image by texture content. Color analysis refers to minimizing the sum of squares of distance between objects and class centroid or corresponding cluster. In threshold matching process individual pixels value is compared with threshold value, if value is greater than threshold then it is marked as object pixel. The texture and color analysis images are compared with the previous images for detection of plant diseases. Author will use Bayes and K-means clustering in future.

In paper [12] authors present image processing techniques for detecting the Bacterial infection in plant. Common infection seen on plant is Bacterial leaf scorch and early detection of this helps in improvement of plant growth. The image processing starts with image acquisition which involves basic steps such as capturing of image and converting it to computer readable format. Then clustering is done to separate foreground and background image with help of K-means clustering method in image segmentation. Clustering is based on intensity mapping and leaf area highlighting is done by subtracting the clustered leaf images from base images. Compared to Fuzzy logic, K-means clustering algorithm is simple and effective in detecting the infected area with reduced manual cluster selection requirement. With ADSP target boards and FPGA tools, further implementation is possible.

In paper [13] authors present image processing technique for detection of unhealthy region of Citrus leaf. There are four types of citrus diseases namely: (i) Citrus canker, (ii) Anthracnose, (iii) Overwatering, (iv) Citrus greening. Author proposed methodology in which image acquisition is first step for capturing image by digital camera in high resolution to create database. Color space conversion and image enhancement is done in image pre-processing. Discrete cosine transform domain is used for color image enhancement. YCbCr color system and L*a*b* color space are chosen for color space conversion. In feature extraction author present statistical method, using Gray-Level Co-Occurrence Matrix (GLCM) to see statistics such as contrast, energy, homogeneity and entropy using graycoprops function. Two types support vector machine (SVM) classifiers: SVMRBF and SVMPOLY are used for differentiating citrus leaf diseases.

In paper [14] authors present image processing technique for Orchid leaf disease detection. Black leaf spot and Sun scorch are two types of orchid leaf diseases mostly found. The basic step of image processing is image acquisition for capturing images

and stores it in computer for further operation. Image pre-processing involves histogram equalization, intensity adjustment and filtering for enhancing or modifying the image. Three morphological processes are used in border segmentation technique for remove small object and preserve large object in image. Thresholding in segmentation is used for start and stop point of line to trace edges. Author added ROI (region of interest) in GUI. After the border segmentation process a classification is done by calculating white pixels in image. This system gives high accuracy and low percentage of error in result.

In paper [15] authors present image processing technique for Tomato leaves diseases detection. In image acquisition phase, digital images of infected tomato leaves are collected which include two types of tomato diseases namely: Early blight and Powdery mildew. In pre-processing phase some techniques are techniques are applied for image enhancement, smoothness; remove noise, image resizing, image isolation, and background removing. Author introduced Gabor wavelet transformation and Support vector machine for identification and classification of tomato diseases. In feature extraction phase with the help of Gabor wavelet transform feature vectors are obtained for next classification phase. In classification phase, support vector machine (SVM) is trained for identifying the category of tomato diseases. The inputs of SVM are feature vectors and corresponding classes, whereas the outputs are the decision that detect tomatos leaf disease. SVM is employed using Invmult Kernel, Cauchy Kernel and Laplacian Kernel functions. Grid search and N-fold cross-validation techniques are used for performance evaluation.

Chapter 6

Proposed Design

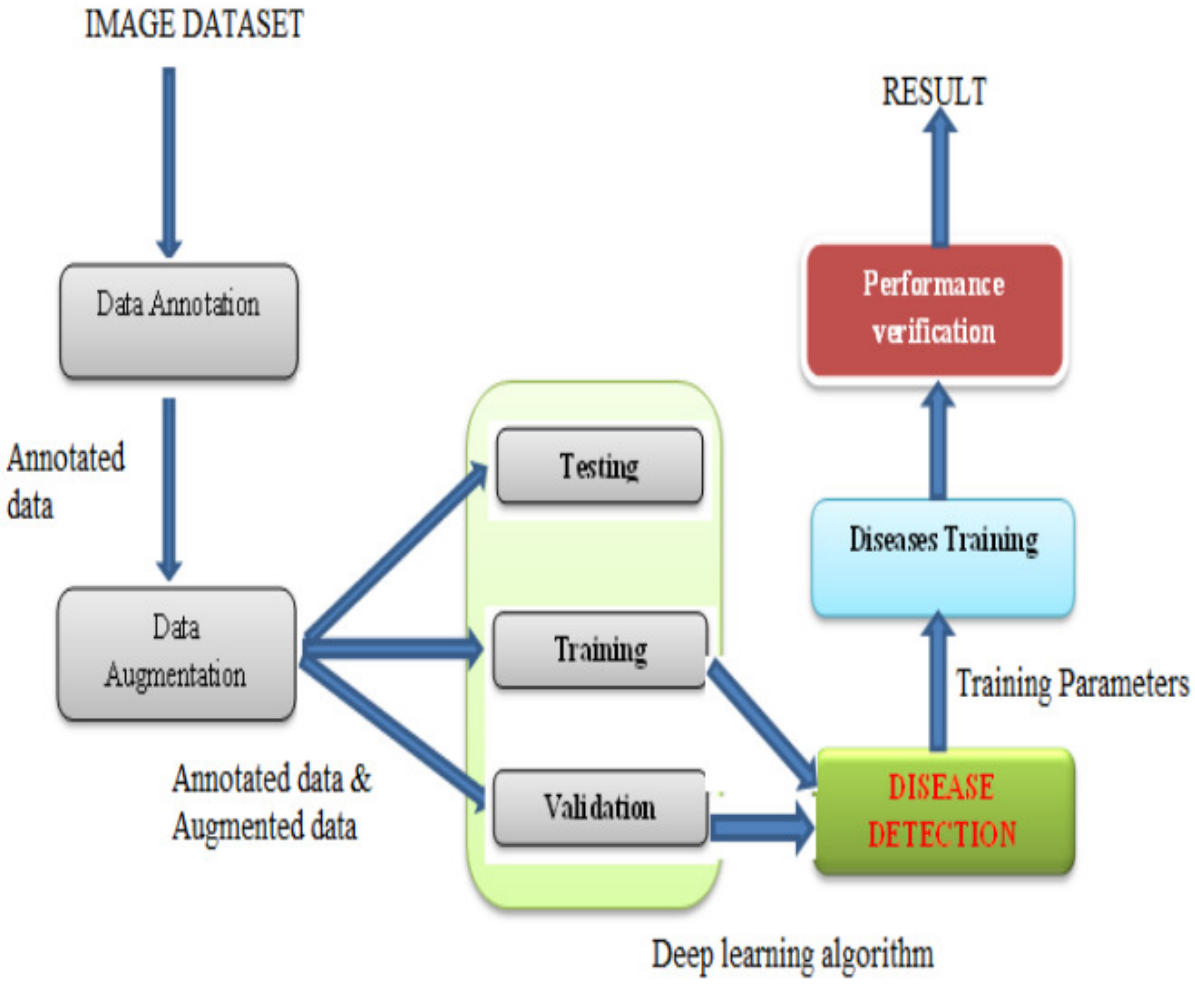

Figure 6.1: Flow chart of the deep meta-architecture approach

Problem statement is to detect if the plant is affected or not. This approach consists of several steps that use input images as a source of information, and provide

detection results in terms of class and location of the infected area of the plant in the image.

Dataset contains images with several diseases and pests in tomato plants. Using simple camera devices, the images were collected under several conditions depending on the time (e.g., illumination), season (e.g., temperature, humidity), and place where they were taken (e.g., greenhouse).

Starting with the dataset of images, we manually annotate the areas of every image containing the disease or pest with a bounding box and class. Some diseases might look similar depending on the infection status that the present; therefore, the knowledge for identifying the type of disease or pest has been provided by experts in the area. That has helped us to visibly identify the categories in the images and infected areas of the plant. This process aims to label the class and location of the infected areas in the image. The outputs of this step are the coordinates of the bounding boxes of different sizes with their corresponding class of disease and pest.

As the number of images in the data set is less data augmentation is necessary to increase the training set. These techniques consist of geometrical transformations (resizing, crop, rotation, horizontal flipping) and intensity transformations (contrast and brightness enhancement, color, noise).

By using Deep Meta Architecture for object detection [10], the architecture will be merged with feature extractors (VGG, Residual Networks ResNet)

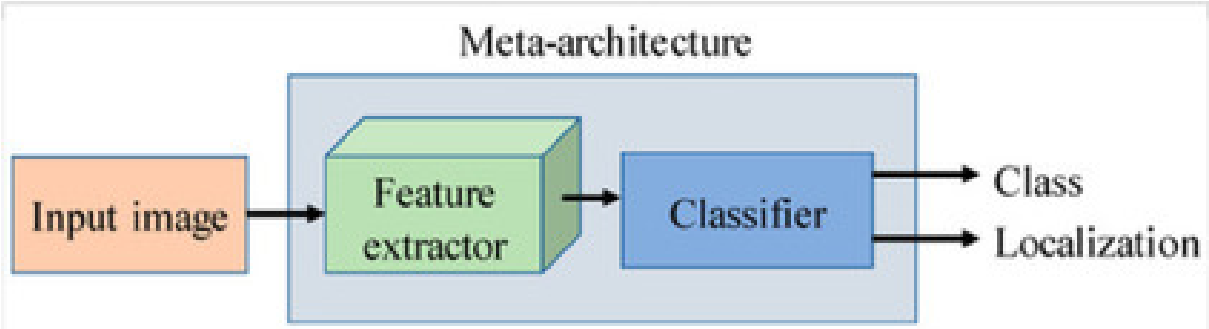

Figure 6.2: System Overview

our system proposes to treat the deep meta-architecture as an open system on which different feature extractors can be adapted to perform on our task. The input image captured by a camera device with different resolutions and scales is fed into our system, which after processing by our deep network (feature extractor and classifier) results in the class and localization of the infected area of the plant in the image. Thus, we can provide a nondestructive local solution only where the damage is present.

We extend the application of Faster R-CNN for object recognition and its

Region Proposal Network (RPN) to estimate the class and location of object proposals that may contain a target candidate. The RPN is used to generate the object proposals, including their class and box coordinates. Then, for each object proposal, we extract the features with an RoI Pooling layer and perform object classification and bounding box regression to obtain the estimated targets.

Chapter 7

Snapshots

This chapter consists the snapshots of different data pre-processing techniques.

7.1 Data Annotation

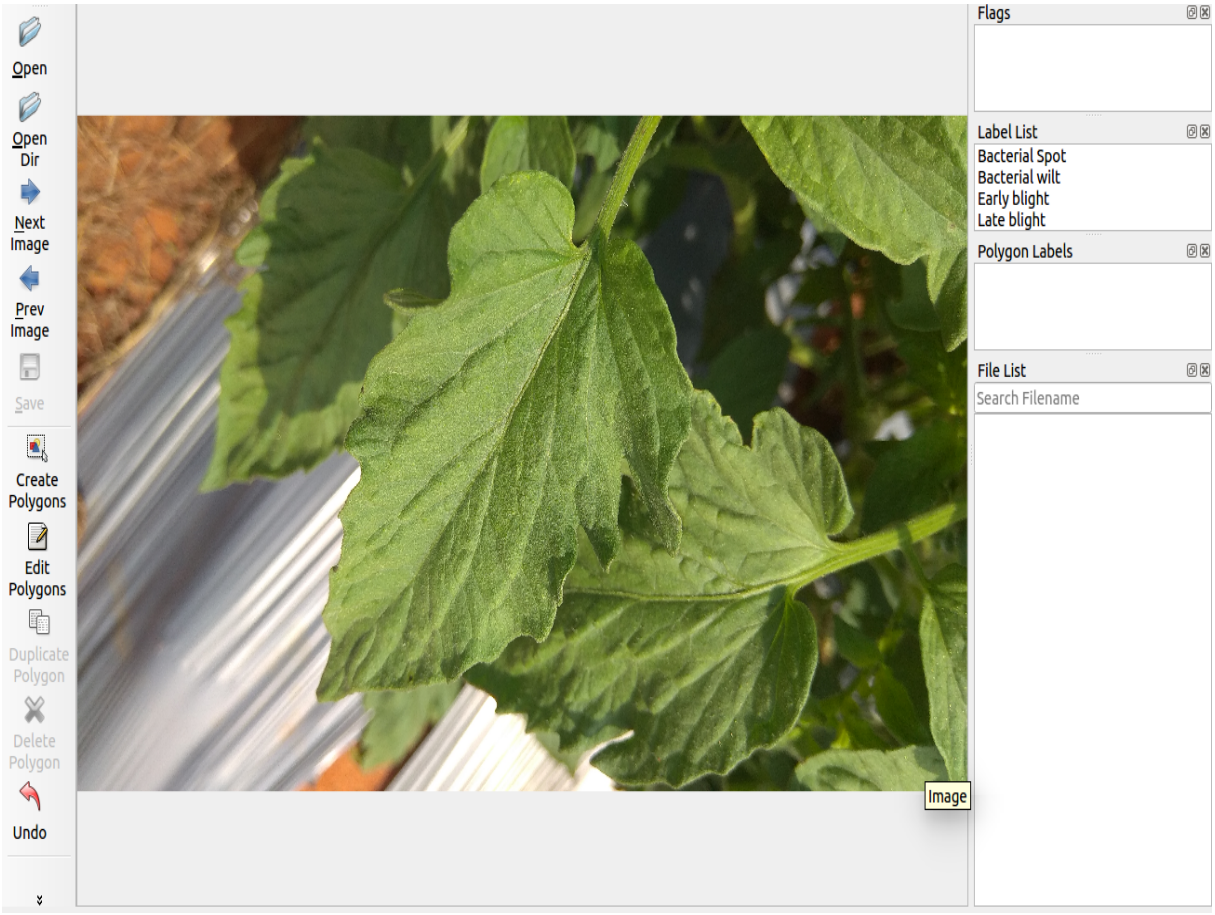

Figure 7.1: Healthy Tomato leaf

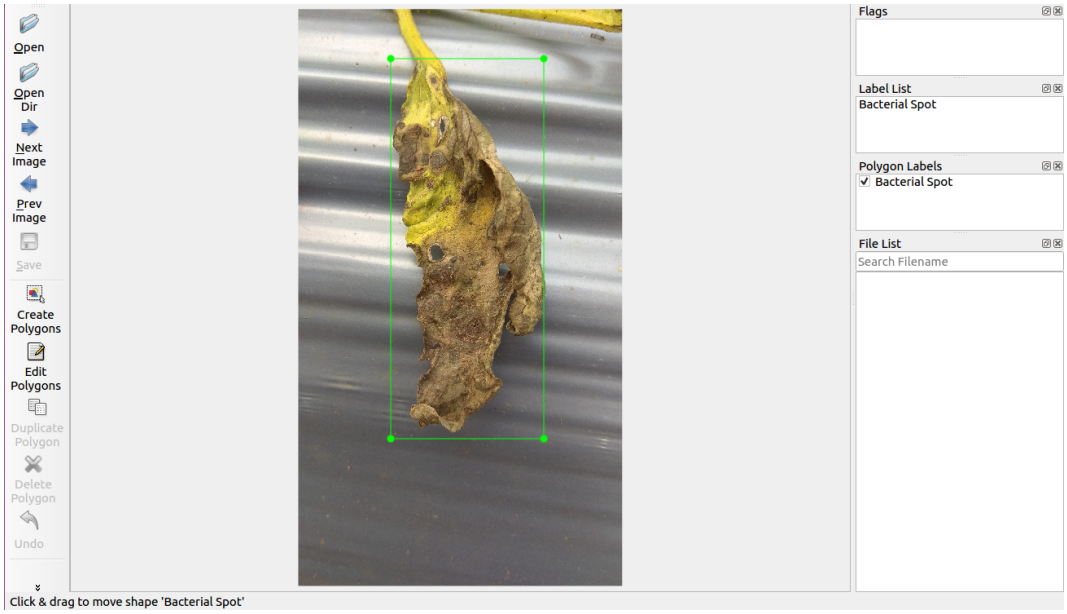

Figure 7.2: Annotated Leaf of Bacterial spot disease

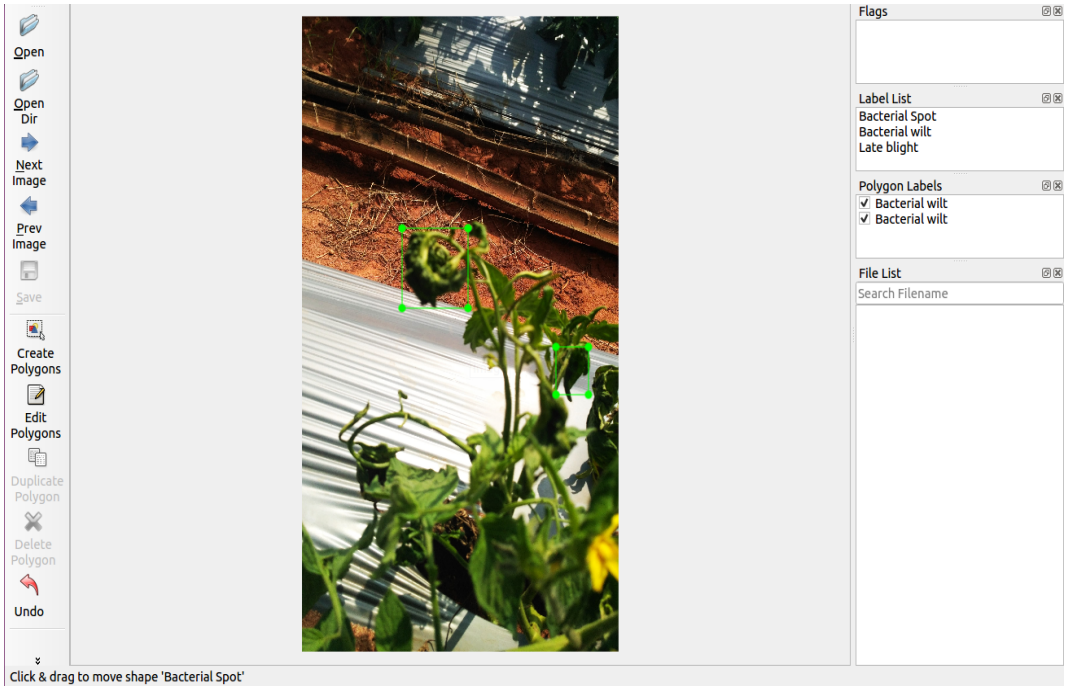

Figure 7.3: Annotated Leaf of Bacterial Wilt disease

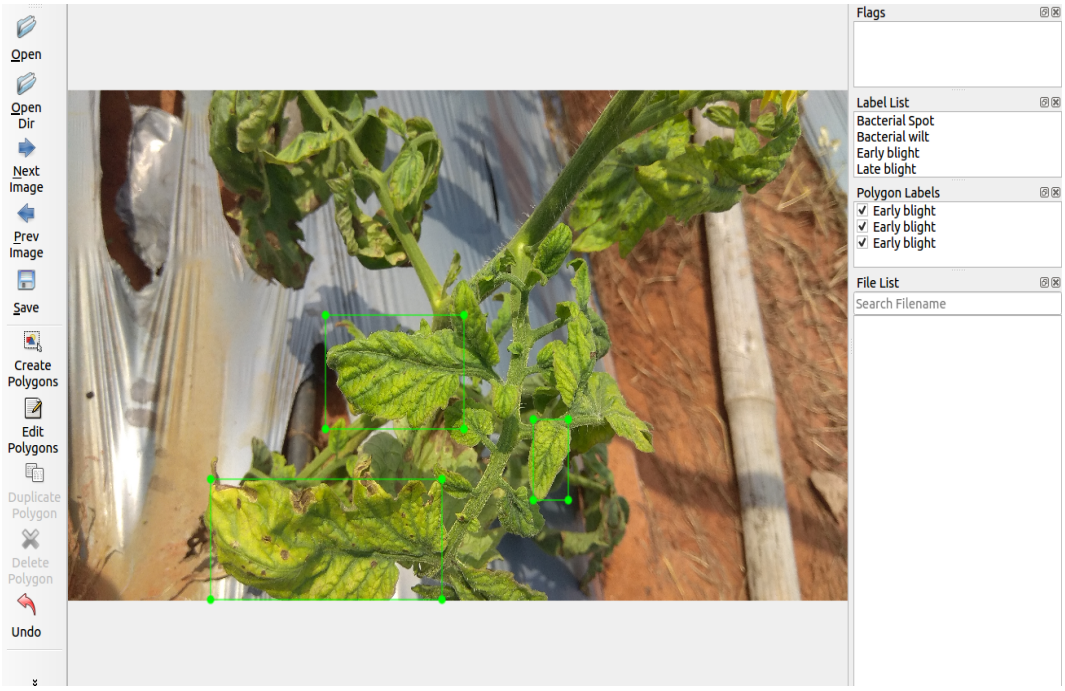

Figure 7.4: Annotated Leaf of Early Blight disease

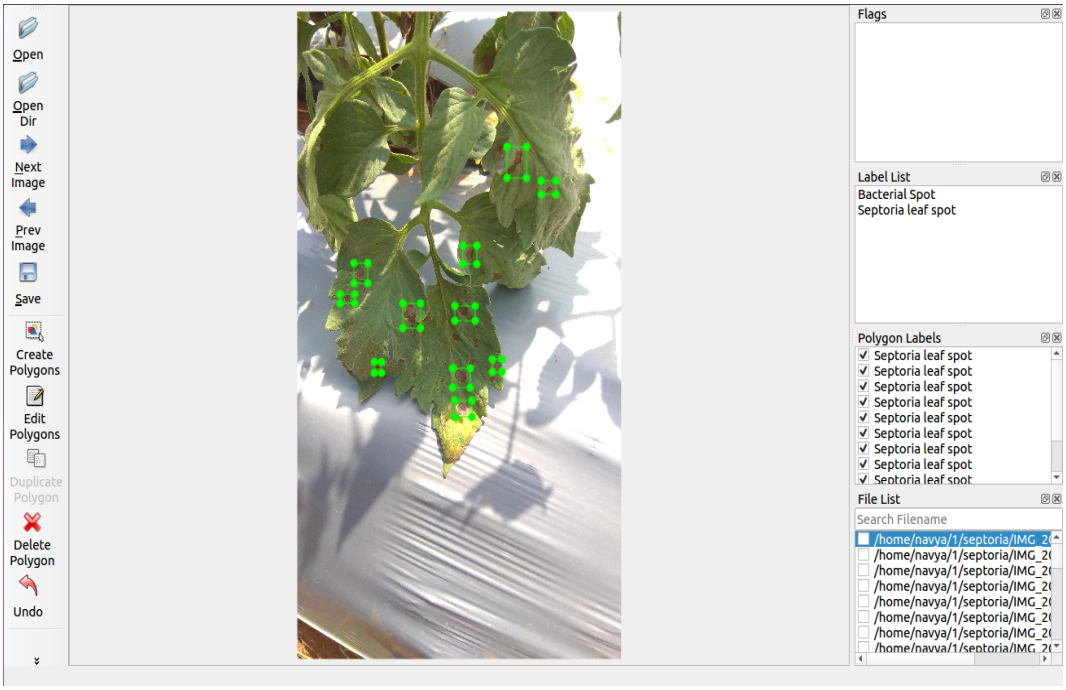

Figure 7.5: Annotated Leaf of Septorial leaf spot disease

7.2 Data Augmentation

Figure 7.6: Augmented Tomato Leaves

7.3 Model Summary

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 64, 64, 3)         0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 64, 64, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 64, 64, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 32, 32, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 32, 32, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 32, 32, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 16, 16, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 16, 16, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 8, 8, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 8, 8, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 4, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 2, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               262272
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 774
=================================================================
Total params: 14,977,734
Trainable params: 263,046
Non-trainable params: 14,714,688
_________________________________________________________________
```

Figure 7.7: Sequential model built by adding all layers

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 2, 2, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              2098176
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 6150
=================================================================
Total params: 16,819,014
Trainable params: 9,183,750
Non-trainable params: 7,635,264
_________________________________________________________________
```

Figure 7.8: Pre-built VGG16 model

The project is built using two models. One is built by adding different layers to the model and another is built by VGG16.

7.4 Model Training

```
In [10]: training_images = 2298
         validation_images = 1114

         history = model.fit_generator(training_generator,
                           steps_per_epoch = 2200,  # this should be equal to total number of images in training set. But to speed up the execution, I a
                           epochs = 3,  # change this for better results
                           validation_data = validation_generator,
                           validation_steps = 1688 )  # this should be equal to total number of images in validation set.

         Epoch 1/3
         2200/2200 [==============================] - 2619s 1s/step - loss: 0.1305 - acc: 0.9561 - val_loss: 0.5324 - val_acc: 0.8297
         Epoch 2/3
         2200/2200 [==============================] - 2716s 1s/step - loss: 0.0363 - acc: 0.9866 - val_loss: 0.5611 - val_acc: 0.8617
         Epoch 3/3
         2200/2200 [==============================] - 2619s 1s/step - loss: 0.0228 - acc: 0.9925 - val_loss: 0.3838 - val_acc: 0.9012
```

Figure 7.9: Model trained using 3 epochs for built model

```
Epoch 1/30
23/22 [==============================] - 174s 8s/step - loss: 1.2560 - acc: 0.5242 - val_loss: 0.8311 - val_acc: 0.6562
Epoch 2/30
23/22 [==============================] - 169s 7s/step - loss: 0.8323 - acc: 0.7159 - val_loss: 1.1276 - val_acc: 0.5727
Epoch 3/30
23/22 [==============================] - 172s 7s/step - loss: 0.6413 - acc: 0.7681 - val_loss: 1.2578 - val_acc: 0.5925
Epoch 4/30
23/22 [==============================] - 168s 7s/step - loss: 0.5767 - acc: 0.7967 - val_loss: 1.3400 - val_acc: 0.6598
Epoch 5/30
23/22 [==============================] - 167s 7s/step - loss: 0.4993 - acc: 0.8321 - val_loss: 0.7642 - val_acc: 0.7585
Epoch 6/30
23/22 [==============================] - 168s 7s/step - loss: 0.4993 - acc: 0.8333 - val_loss: 1.4474 - val_acc: 0.5682
Epoch 7/30
23/22 [==============================] - 168s 7s/step - loss: 0.4189 - acc: 0.8646 - val_loss: 1.1076 - val_acc: 0.6239
Epoch 8/30
23/22 [==============================] - 167s 7s/step - loss: 0.4529 - acc: 0.8425 - val_loss: 0.6429 - val_acc: 0.7666
Epoch 9/30
23/22 [==============================] - 165s 7s/step - loss: 0.4607 - acc: 0.8491 - val_loss: 0.5427 - val_acc: 0.8016
Epoch 10/30
23/22 [==============================] - 164s 7s/step - loss: 0.4761 - acc: 0.8460 - val_loss: 1.0307 - val_acc: 0.6382
Epoch 11/30
23/22 [==============================] - 180s 8s/step - loss: 0.4513 - acc: 0.8460 - val_loss: 0.8287 - val_acc: 0.6741
Epoch 12/30
23/22 [==============================] - 177s 8s/step - loss: 0.4051 - acc: 0.8725 - val_loss: 1.7910 - val_acc: 0.5628
Epoch 13/30
23/22 [==============================] - 184s 8s/step - loss: 0.3434 - acc: 0.8850 - val_loss: 1.3448 - val_acc: 0.5826
Epoch 14/30
23/22 [==============================] - 174s 8s/step - loss: 0.5074 - acc: 0.8390 - val_loss: 0.5575 - val_acc: 0.7864
Epoch 15/30
23/22 [==============================] - 164s 7s/step - loss: 0.3696 - acc: 0.8873 - val_loss: 0.7174 - val_acc: 0.7280
Epoch 16/30
23/22 [==============================] - 165s 7s/step - loss: 0.4325 - acc: 0.8686 - val_loss: 0.9096 - val_acc: 0.7217
Epoch 17/30
23/22 [==============================] - 163s 7s/step - loss: 0.4172 - acc: 0.8747 - val_loss: 1.0426 - val_acc: 0.6741
Epoch 18/30
23/22 [==============================] - 164s 7s/step - loss: 0.4047 - acc: 0.8904 - val_loss: 0.7325 - val_acc: 0.7047
Epoch 19/30
23/22 [==============================] - 164s 7s/step - loss: 0.4331 - acc: 0.8746 - val_loss: 0.6834 - val_acc: 0.7361
Epoch 20/30
23/22 [==============================] - 164s 7s/step - loss: 0.2961 - acc: 0.9116 - val_loss: 1.4415 - val_acc: 0.6463
Epoch 21/30
23/22 [==============================] - 164s 7s/step - loss: 0.3267 - acc: 0.8994 - val_loss: 1.6375 - val_acc: 0.6625
Epoch 22/30
23/22 [==============================] - 199s 9s/step - loss: 0.5088 - acc: 0.8795 - val_loss: 0.7401 - val_acc: 0.7540
Epoch 23/30
23/22 [==============================] - 205s 9s/step - loss: 0.2792 - acc: 0.9012 - val_loss: 0.8774 - val_acc: 0.7145
Epoch 24/30
23/22 [==============================] - 185s 8s/step - loss: 0.3668 - acc: 0.8869 - val_loss: 0.5140 - val_acc: 0.8330
Epoch 25/30
23/22 [==============================] - 163s 7s/step - loss: 0.4686 - acc: 0.8795 - val_loss: 0.5437 - val_acc: 0.7962
Epoch 26/30
23/22 [==============================] - 164s 7s/step - loss: 0.3016 - acc: 0.9121 - val_loss: 0.8185 - val_acc: 0.7926
Epoch 27/30
23/22 [==============================] - 164s 7s/step - loss: 0.4438 - acc: 0.8776 - val_loss: 4.0668 - val_acc: 0.5206
Epoch 28/30
23/22 [==============================] - 164s 7s/step - loss: 0.3304 - acc: 0.9039 - val_loss: 0.4476 - val_acc: 0.8483
Epoch 29/30
23/22 [==============================] - 163s 7s/step - loss: 0.3734 - acc: 0.8830 - val_loss: 0.6933 - val_acc: 0.7567
Epoch 30/30
23/22 [==============================] - 163s 7s/step - loss: 0.4757 - acc: 0.8767 - val_loss: 0.8758 - val_acc: 0.6966
```

Figure 7.10: Model trained using 30 epochs for VGG16 model

The built model is trained using 3 epochs and 2200 images, Whereas vgg16 model is trained using 30 epochs and 22 images.

7.5 Training and Validation accuracy

```
print ('teTraining Accuracy = ' + str(history.history['acc']))
print ('Validation Accuracy = ' + str(history.history['val_acc']))
```

```
teTraining Accuracy = [0.95602392478058462, 0.98655653659246889, 0.9924807747080604]
Validation Accuracy = [0.82966051209871083, 0.86174806427962725, 0.90124329945883486]
```

Figure 7.11: Accuracy for built model

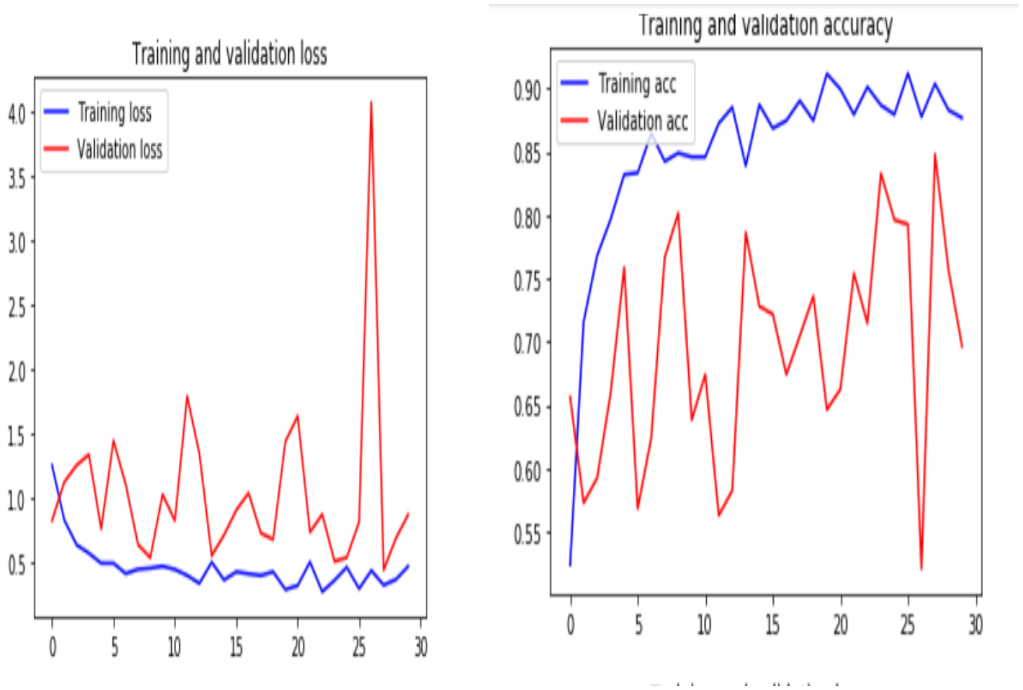

Figure 7.12: Plot of Training and Validation Accuracy/Loss in VGG16

The Training accuracy for the built model is more than the VGG16 model.

7.6 Result

```
[[  0.00000000e+00   0.00000000e+00   1.72781119e-31   1.00000000e+00
    0.00000000e+00   0.00000000e+00]]
b403.jpg:Tomato mosaic
[[ 0.  0.  0.  1.  0.  0.]]
h404.png:Tomato mosaic
[[ 0.  0.  0.  1.  0.  0.]]
l403.png:Tomato mosaic
[[  0.00000000e+00   0.00000000e+00   0.00000000e+00   1.00000000e+00
    2.45003267e-18   0.00000000e+00]]
y707.jpg:Tomato mosaic
[[ 0.  0.  0.  1.  0.  0.]]
sep403.jpg:Tomato mosaic
[[ 0.          0.          0.          0.99880004  0.00119995  0.        ]]
m202.jpg:Tomato mosaic
[[  0.00000000e+00   0.00000000e+00   0.00000000e+00   7.16129989e-10
    1.00000000e+00   0.00000000e+00]]
m204.jpg:septorial leaf spot
[[ 0.  0.  0.  1.  0.  0.]]
sep405.jpg:Tomato mosaic
[[ 0.  0.  0.  1.  0.  0.]]
h405.png:Tomato mosaic
[[  0.00000000e+00   0.00000000e+00   0.00000000e+00   1.32500347e-13
    1.00000000e+00   5.09351976e-35]]
y709.jpg:septorial leaf spot
[[ 0.  0.  0.  1.  0.  0.]]
y702.jpg:Tomato mosaic
[[  0.00000000e+00   0.00000000e+00   0.00000000e+00   2.75237921e-09
    1.00000000e+00   1.24274777e-12]]
y710.jpg:septorial leaf spot
[[  5.49072000e-24   0.00000000e+00   0.00000000e+00   1.00000000e+00
    0.00000000e+00   0.00000000e+00]]
h402.png:Tomato mosaic
[[ 0.  0.  0.  1.  0.  0.]]
sep402.jpg:Tomato mosaic
[[  0.00000000e+00   0.00000000e+00   0.00000000e+00   1.00000000e+00
    1.36946244e-31   0.00000000e+00]]
y703.jpg:Tomato mosaic
[[ 0.  0.  0.  1.  0.  0.]]
h401.png:Tomato mosaic
[[ 0.  0.  0.  1.  0.  0.]]
h403.png:Tomato mosaic
[[ 0.  0.  0.  1.  0.  0.]]
```

Figure 7.13: Prediction of disease with probaility using built model

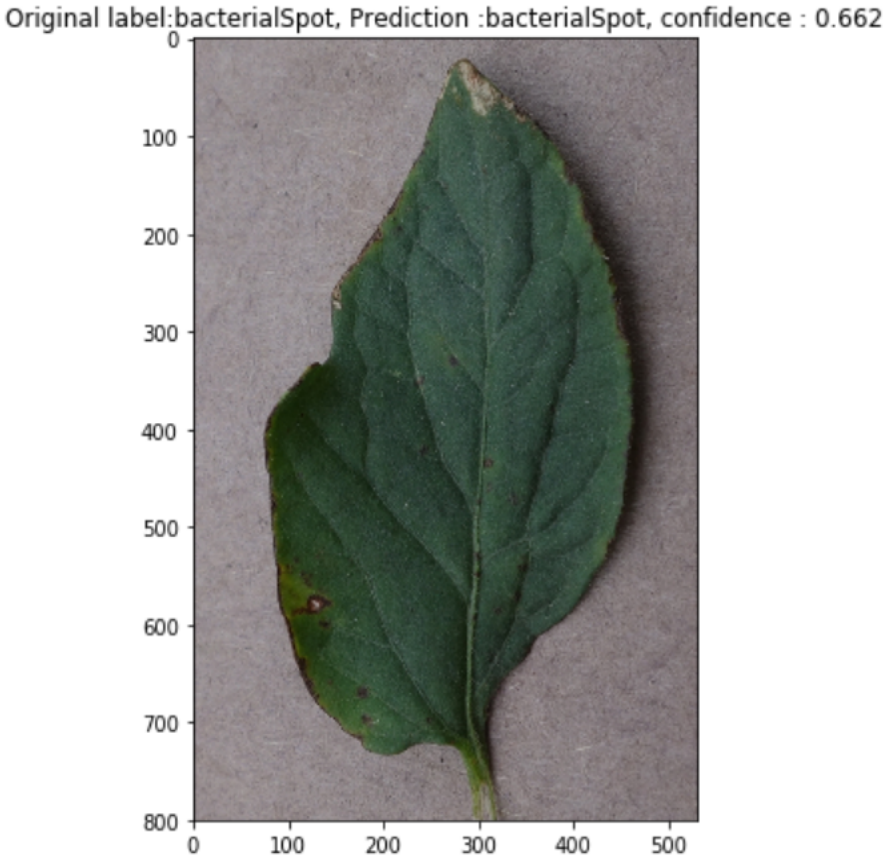

Figure 7.14: Predictiion of disease from VGG16 model

The number of diseases predicted correctly using built model is less than vgg16 model. In vgg16 model we got correct prediction with 70% confidance.

Chapter 8

Conclusion and Future Enhancement

8.1 Conclusion

The collection of various diseases of tomato plants are taken and later model is created by training them. Tomato mosaic, YellowCurved, Septoria leaf spot, Bacterial spot and Early Blight are the detected diseases. For detection purpose VGG16 and VGG19 is used to train a model and predict diseases in tomato plant. Python programming language is used to implement this project. It is found that using technique based data annotation and augmentation results in better performance. This system is capable of detecting five class of diseases and healthy plant. In order to detect other class of diseases, related data has to be trained. The main challenge while developing disease detection model on machine learning was to collect large number train images with different shapes, sizes, with different background, light intensity, orientation and aspect ratio.

8.2 Future Enhancement

- In order to detect other class of diseases in different plants, related data has to be trained.
- Disease localization can be implemented.

References

[1] Jiang Lu, Jie Hu, Guannan Zhao, Fenghua Mei, Changshui Zhang, An infeldautomatic wheat disease diagnosis system, Computers and Electronics in Agriculture 142 (2017) 369379.

[2] Andreas Kamilaris, Francesc X. Prenafeta-Boldu Deep learning in agriculture: Asurvey, Computers and Electronics in Agriculture 147 (2018) 7090.

[3] Konstantinos P. Ferentinos, Deep learning models for plant disease detection and diagnosis Computers and Electronics in Agriculture 145 (2018) 311318.

[4] Kulkarni Anand H, Ashwin Patil RK. Applying image processing technique to detect plant diseases. Int J Mod Eng Res 2012;2(5):36614.

[5] Bashir Sabah, Sharma Navdeep. Remote area plant disease detection using image processing. IOSR J Electron Commun Eng 2012;2(6):314. ISSN: 2278-2834.

[6] Rakesh Kaundal, Amar S Kapoor and Gajendra PS Raghava Machine learning technique in disease forecasting: a case study on rice blast prediction, BMC Bioinformatics, 2006.

[7] Srdjan Sladojevic, Marko Arsenovic, Andras Anderla, Dubravko Culibrk, and DarkoStefanovic, Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classifcation, Hindawi Publishing Corporation, Computational Intelligence and Neuroscience Volume 2016,Article ID 3289801,11 pages http://dx.doi.org/10.1155/2016/3289801.

[8] C. C. Stearns and K. Kannappan, Method for 2-D afne transformation of images,US Patent No. 5,475,803, 1995.

[9] Sankaran, S. Mishra, A.; Ehsani, R. A review of advanced techniques for detecting plant diseases. Comput. Electron. Agric. 2010, 72, 113.

[10] Alvaro Fuentes, Sook Yoon, Sang Cheol Kim and Dong Sun Park A Robust Deep Learning- Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition, Sensors 2017, 17, 2022; doi:10.3390/s17092022.

[11] Ren, S., He, K., Girshick, R., Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Trans. Pattern Anal. Mach. Intell. 2016, 39, 11371149.