

PLANT DISEASE DETECTION USING MACHINE LEARNING TECHNIQUES

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering
in
Computer Science and Engineering
by

Sushil Athithiyaa.S.K
(Reg.No: 38110582)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU**

March 2022



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade “A” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

www.sathyabama.ac.in

SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of **Sushil Athithiyaa.S.K (Reg.No.38110582)** who underwent the professional training in “**PLANT DISEASE DETECTION USING MACHINE LEARNING TECHNIQUES**” under my supervision from June 2021 to March 2022.

Internal Guide

Dr L. K. Joshila Grace, M.E., Ph.D.,

Head of the Department

**Dr. S. VIGNESHWARI, M.E., Ph.D.,
Dr. LAKSHMANAN L, M.E., Ph.D.,**

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **Sushil Athithiyaa.S.K (RegNo:38110582)** hereby declare that the Professional Training Report “**PLANT DISEASE DETECTION USING MACHINE LEARNING TECHNIQUE**” done by me under the guidance of **Dr.L.K.JOSHILA GRACE,M.E.,Ph.D** Department of Computer Science and Engineering at SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T.SASIKALA, M.E.,Ph.D., Dean,School of Computing** and**Dr.S.VIGNESHWARI,M.E.,Ph.D.,and Dr.L.LAKSHMANAN,M.E.,Ph.D.,Heads of the Department, Dept. of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide, **Dr.L.K.JOSHILA GRACE, M.E., Ph.D** for his valuable suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science and Engineering who were helpful in many ways for the completion of the project.

ABSTRACT

Agricultural productivity is something on which economy highly depends. This is the major reasons that disease detection in plants plays an important role in agriculture field, as having disease in plants are quite natural. If proper care is not taken in this area then it causes serious effects on plants and due to which respective product quality, quantity or productivity is affected. The Existing methods failed to detect plant diseases accurately and efficiently. The proposed method will provide a robust system based on deep learning techniques such as Convolutional neural networks (CNN), ResNet - 152 are used to visualise various plant diseases. The Proposed system will detect plant diseases accurately and efficiently which will meet all the bench marks.

Keywords: Plant disease, Deep learning, Convolutional neural networks (CNN), ResNet - 152.

TABLE OF CONTENTS

Title	Page No.
ABSTRACT.....	vi
ACKNOWLEDGEMENTS.....	iv
LIST OF FIGURES.....	viii
LIST OF TABLES.....	ix
ABBREVIATIONS.....	x
CHAPTER 1 INTRODUCTION.....	1
1.1	
Introduction.....	1
1.2 Statement of	
Problem.....	2
1.3	
Motivation.....	2
1.4	
Objectives.....	3
CHAPTER 2 LITERATURE REVIEW.....	4
CHAPTER 3 METHODOLOGY.....	8

3.1 Proposed	
Method.....	8
3.1.1	
Dataset.....	8
3.1.2 Data Preprocessing and Augmentation	
.....	8
3.1.3 System Overview	
.....	9
3.2 Image	
Processing.....	10
3.3 CNN.....	13
3.3.1 Convolutional Layer.....	14
3.3.2 Pooling Layer.....	14
3.3.3 Activation Layer.....	15
3.3.4 Fully Connected Layer.....	15
3.4 ResNet -	
152.....	15
3.4.1 ResNet Variations You Can Use on PyTorch	17
3.4.2 Options for Running ResNet on PyTorch	
.....	17
3.4.3 Coding a ResNet Architecture Yourself Using PyTorch	
.....	18
3.4.4.Using ResNet50 with Transfer	
Learning.....	20

3.4.5 Scaling ResNet on PyTorch	21
---------------------------------	----

3.4.6 Deep Residual Learning for Image Recognition.....	21
--	----

CHAPTER 4 SYSTEM SPECIFICATION

REQUIREMENTS.....	2
--------------------------	----------

4.1 Hardware Requirements.....	22
-----------------------------------	----

4.2 Software Requirements.....	22
-----------------------------------	----

CHAPTER 5 SYSTEM DESIGN.....	23
---	-----------

5.1 Introduction.....	23
--------------------------	----

CHAPTER 6 SYSTEM TESTING.....	24
--	-----------

6.1 Testing Methodologies.....	24
-----------------------------------	----

6.2 Test Cases.....	30
------------------------	----

6.3

Results.....	31
--------------	----

6.4 Deeper

Studies.....	33
--------------	----

CHAPTER 7 OUTPUT

SCREENS.....	35
---------------------	-----------

CHAPTER 8

CONCLUSION.....	39
------------------------	-----------

CHAPTER 9 FUTURE

SCOPE.....	39
-------------------	-----------

CHAPTER 10

REFERENCES.....	39
------------------------	-----------

CHAPTER 11

APPENDICES.....	42
------------------------	-----------

PROGRAM

OUTCOMES.....	44
----------------------	-----------

PROGRAM SPECIFIC

OUTCOMES.....	45
----------------------	-----------

PROGRAM EDUCATIONAL

OBJECTIVES.....	46
------------------------	-----------

COURSE

OUTCOMES.....	47
----------------------	-----------

MAPPING OF COURSE OUTCOMES WITH	
POs AND	
PSOs.....	48

LIST OF FIGURES

Figure No:	Title	Page No:
3.1	Images from Dataset	8
3.1	Block Diagram Of Identification System	10
3.2	Image Processing	11
3.3	A general CNN architecture	12
3.3	Convolution operation for 5x5 input image and 3x3 filter.	14
3.3	Max pooling with 2x2 filters and stride 2	15
3.4	ResNet-152 Architecture	16
3.4	Layers of ResNet-152	17
6.1	Different Levels of Testing	25
6.2	Plain VGG and VGG with Residual Blocks	32
6.2	ResNet 2 layer and 3 layer Block	33
7.2	Result when the above image has been processed in the software	35

LIST OF TABLES

TableNo.	Title	PageNo.
3.1	Dataset Breakup	9
6.2	Top-1 error on ImageNet Validation	31
6.2	ResNet Architecture Values	33
6.2	Error Rates(%) of single model results	34

ABBREIVATIONS

CNN	Convolutional Neural Networks
ResNet-152	Residual Network
VGG	Visual Geometry Group
PReLU	Parametric ReLU
ReLU	Rectified Linear Activation

1.INTRODUCTION

The primary occupation in India is agriculture. India ranks second in the agricultural output worldwide. Here in India, farmers cultivate a great diversity of crops. Various factors such as climatic conditions, soil conditions, various disease, etc affect the production of the crops. The existing method for plants disease detection is simply naked eye observation which requires more man labor, properly equipped laboratories, expensive devices ,etc. And improper disease detection may led to inexperienced pesticide usage that can cause development of long term resistance of the pathogens, reducing the ability of the crop to fight back. The plant disease detection can be done by observing the spot on the leaves of the affected plant. The method we are adopting to detect plant diseases is image processing using Convolution neural network(CNN).[11] As the time changes we are facing challenges in agricultural fields.Agricultural productivity is something on which economy highly depends. Plant disease detection in early stages is tremendously important so that required precautions are taken by farmers to avoid huge loss. Machine learning has emerged with big data technologies and high-performance computing to create new opportunities for data intensive science in the multi-disciplinary agri-technologies domain. Deep learning with convolutional neural networks (CNNs) has achieved great success in the classification of various plant diseases.Plant disease diagnosis through optical observation of the symptoms on plant leaves, incorporates a significantly high degree of complexity.Here we are using convolutional nueral networks in pytorch to detect if a leaf has a disease and to identify what type of disease it have when given a plant leaf image. In addition, some research gaps are identified from which to obtain greater transparency for detecting diseases in plants, even before their symptoms appear clearly. Major role in economic development of India Agriculture is the largest economic sector. The manual classification and identification methods which are being used to distinguish between different types of leaf diseases that are trusting on human resource. They are subjected to some kind of errors since these techniques are focused by human involvement. Since humans are subjected to tiredness and the automated system also helps to reduce the time consumed by manual techniques. The deficiency of labors, automatic system needs to be incorporated to minimize the work and Many new farming computerization tools are being established by university investigators that pose questions about the Effectiveness with which we succeed current farming practices.Identification of the plant diseases is the key to preventing the losses in the yield and quantity of the agricultural product. The studies of the plant diseases mean the studies of visually observable patterns seen on the plant. Health monitoring and disease detection on plant is very critical for sustainable agriculture. It is

very difficult to monitor the plant diseases manually. It requires tremendous amount of work, expertise in the plant diseases, and also require the excessive processing time. Hence, image processing is used for the detection of plant diseases. It has the following features: Disease detection involves the steps like image pre-processing, image segmentation, feature extraction and classification.[10]

1.2.Statement of Problem:

Plant diseases have turned into a dilemma as it can cause significant reduction in both quality and quantity of agricultural products. Early Detection of plant diseases is an essential research topic as it may prove benefits in monitoring large fields of crops. The proposed system is a software solution for Detection and classification of plant leaf diseases. The scheme consists of four main steps,

- Loading Data -> Data Loaders and Data Sizes
- Organising Data SetBuild and Train Network
- Creating Classifier
- Load Checkpoint and rebuild Model

1.3.Motivation:

- Leaves are delicate part of plant, The evaluation of agricultural harvest Classification is dynamic.
- The most important visual property is leafs texture and colour.
- Hence, classification of leaf disease is necessary in evaluating agricultural produce, increasing market value and meeting quality standards. Identifying and taking further dealings for further diffusion of the diseases it is also helpful.

The process will be too slow, If the identification and categorisation is done through physical

- techniques, we need the experts help sometimes it will be error prone and who are less available.

- The labor's classify based on colour, size etc. if these quality methods are recorded into automatic system by using appropriate program design language then the effort will be error free and faster
- There are two main characteristics of plant disease detection machine-learning methods that must be achieved, they are: speed and accuracy

1.4.Objectives:

- To Detect unhealthy region of plant leaves.
- Classification of plant leaf diseases using texture features.
- Coding is used to analyze the leaf infection.
- To Detect diseased leaf, stem, fruit.
- To Quantify affected area by disease.
- To Find shape of affected area.
- To Determine size & shape of fruits 5. Plant diseases can be identified at early or the initial stage.

2.LITERATURE REVIEW

- LEAF DISEASE SEVERITY MEASUREMENT USING IMAGE PROCESSING
Sanjay B. Patil, Dr. Shrikant K. Bodhe, Leaf Disease Severity Measurement Using Image Processing” mentioned in their research that Fungi-caused diseases in sugarcane are the most predominant diseases which appear as spots on the leaves. If not treated on time, causes the severe loss. Excessive use of pesticide for plant diseases treatment increases the cost and environmental pollution so their use must be minimized. This can be achieved by targeting the diseases places, with the appropriate quantity and concentration of pesticide by estimating disease severity using image processing technique. Simple

threshold and Triangle thresh holding methods are used to segment the leaf area and lesion region area respectively. Finally diseases are categories by calculating the quotient of lesion area and leaf area. The accuracy of the experiment is found to be 98.60 %. Research indicates that this method to calculate leaf disease severity is fast and accurate. Disease symptoms of the plant vary significantly under the different stages of the disease so to the accuracy with which the severity of the disease measured is depends upon segmentation of the image. Simple threshold segmentation is used to calculate the leaf area but this method is not suitable to calculate the area of the lesion region because of varying characteristics of the lesion region. Triangle method of the thresh holding used here to segment the lesion region. The average accuracy of the experiment is 98.60 %. Thus image processing technology to measure plant disease severity is convenient and accurate. This eliminates subjectivity of traditional methods and human induced errors. It will helps to farmers to decide the specific quantity for pesticide application which reduces the cost and environmental pollution. [1]

- Jundare Manisha, Jundare Pallavi, Jundare Pragati, Prof. C.S.Aryan, “Plant Disease Detection and its Treatment using Image Processing” mentioned in their research that India is farming nation, Approx. 20 % of harvest yield is missing universal due to pest attack every year which is valued around Rs. 90,000 million. Saving 5 Rs. is like earning 5 Rs. farmers have been using a pesticide, which increases the crop yield to avoid loss. Remains problems in environment due to large use of pesticides harms to soil, sharp toxicity to humans and natural world, change in insect type in agro ecosystems, high cost of control practices etc. Fungus are very known dangerous insects there on grass of plant, send out steamy honeydew, reason loss of leaves and harm the harvest surrender. The visual judgment of farmers is counting of whiteflies has been mostly relied. The illustration decision by farmers for bulk of whiteflies has been less accurate Because of the identification skills has different levels. In laboratory also detection of present whiteflies on leaves, it takes extended time for detection of whiteflies at early on stages has become important because off inimical importance of harvests and strong impacts of damage levels. In proposed solution, using web application, whiteflies on leaves of plant at early stages we are calculating no. of eggs also. farmers to use pesticide as early as possible It will give correct idea . They can avoid damage and control whiteflies, By this

technique, farmers are capable of improve 80 % of lost that will cause due to pest occurrence. Thus, is important to correctly diagnose a disease before proffering management options. Diagnosis, being the process of determining the cause of a problem requires the attention of an expert. extract the features of infected leaf and the classification of plant diseases From these methods, we can accurately identify and classify various plant diseases and provide suitable treatment using image processing techniques.[2]

- Prakash M. Mainkar, Shreekant Ghorpade, Mayur Adawadkar, "Plant "Plant Disease Detection and its Treatment using Image Processing Leaf Disease Detection and Classification Using Image Processing Techniques", mentioned in their research that 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 network based classification. The study reviews and summarizes image processing techniques for several plant species that have been used for recognizing plant diseases. The major techniques used are K-means clustering, GLCM and BPNN. Some of the challenges in these techniques are optimization of the technique for a specific plant, effect of the background noise in the acquired image and automation technique for a continuous automated monitoring of plant leaf diseases under real world field conditions. The proposed approach is a valuable approach, which can significantly support an accurate detection of leaf diseases in a little computational effort. Further future work can be extended by developing better segmentation technique; selecting better feature extraction and classification algorithms and NNs in order to increase the recognition rate of final classification process. Also by computing severity and amount of disease present on the crop, only necessary and sufficient amount of pesticides can be used making agriculture production system economically efficient. So there is a scope of improvement.[3]

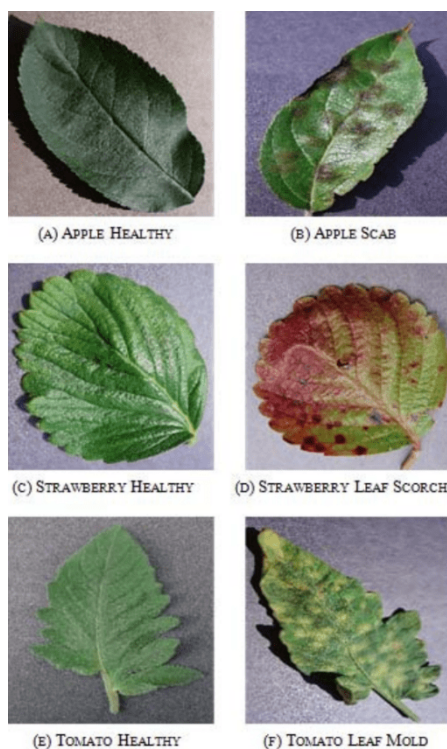
- **SOIL MOISTURE DETERMINATION USING REMOTE SENSING DATA FOR THE PROPERTY PROTECTION AND INCREASE OF AGRICULTURE PRODUCTION.** To provide geospatial data that enables generation of adequate information related to floods and droughts, we applied the remote sensing method that relies on the use of soil-moisture index (SMI) which in its algorithm uses the data obtained from satellite sensors. As presented by Hunt, the index is based on the actual content of water (θ), water capacity and wilting point. Multispectral satellite images from visible (red band) and infrared bands (near infrared and thermal bands) are essential for the calculation of the index.[4]
- Digital image process is the use of computer algorithms to perform image process on digital pictures. It permits a far wider vary of algorithms to be applied to the computer file and might avoid issues like the build-up of noise and signal distortion throughout process. Digital image process has terribly important role in agriculture field. It's widely accustomed observe the crop disease with high accuracy. [5]
- **AN IoT BASED SOIL MOISTURE MONITORING ON LOSANT PLATFORM.** The Internet of Things (IoT) is converting the agriculture industry and solving the immense problems or the major challenges faced by the farmers today in the field. India is one of the 13th countries in the world having scarcity of water resources. Due to ever increasing of world population, we are facing difficulties in the shortage of water resources, limited availability of land, difficult to manage the costs while meeting the demands of increasing consumption needs of a global population that is expected to grow by 70 percent by the year 2050.[6]
- **SMART FARMING: IoT BASED SMART SENSORS AGRICULTURE STICK FOR LIVE TEMPERATURE AND MOISTURE MONITORING USING ARDUINO, CLOUD COMPUTING AND SOLAR TECHNOLOGY.** The Agriculture stick being proposed via this paper is integrated with Arduino Technology, Breadboard mixed with various sensors and live data feed can be obtained online from Thingspeak.com. The product being proposed is tested on Live Agriculture Fields giving high accuracy over 98 percent in data feeds.[7]
- A paper by Sue Han Lee[4] makes use of Caffe framework and Back-Propogation mechanism in training the model. One of the approaches used in understanding the internal working of the CNN model and visualizing the selected filters was done using

Deconvolutional Network. The Deconvolutional Network provides a function that enables us to see the feature map at each layer by deconvoluting and unpooling down to input image pixel. The models were trained on two datasets, MalayaKew Dataset(original) and addition of local leaf data to original dataset. Experimental results show that the model trained on the modified dataset yielded only a small increase in the accuracy of the model when compared to the model trained on the original dataset. In another paper that proposes a plant identification system [6], the standard VGGNet-16 architecture was taken and modified by adding more convolution layers for learning the combined species and organ features resulting the high-level fusion architecture model. The high-level fusion architecture model model and Finetuned VGGNet-16 were trained on PlantClef2015 dataset. According to the experimental results, the HighLevel fusion architecture didn't perform better compared to the fine tuned VGGNet-16. Ignacio's[9] white paper discusses the implementation of a large-scale plant classification model which uses the ResNet-50 architecture. The performance of the model was tested on the Portuguese flora dataset[19], iNaturalist dataset and Google Search Image results.

3.METHODOLOGY

3.1 PROPOSED METHOD

3.1.1. Dataset The dataset used in this study is called the PlantVillage SP Mohanty's Git-Hub repository. The dataset comprised of raw for SVM and other images consisted each segmented images. close to 55k images different classes. classes present in the trained on 29 classes, selecting classes with healthy class each, so distinguish the disease. were used in this study



dataset used in this study is Dataset was obtained from repository. The dataset images, data distribution useful data. The raw of colour, grayscale and Each category resulting which includes 38 Altogether there are 38 dataset, our models were and selection was done by atleast one disease and one that the model learns to Coloured (RGB) images to train and classify the

diseases. Hence the selected dataset comprises of 29 classes and 36k images. Hierarchical division is done on the selected dataset as shown in Table I and resulting into 8 superficial classes. Now the dataset is split into training and validation sets with 80% and 20% of the dataset respectively. Thus, we have 28938 and 7210 images for training and validation sets respectively. Fig. 1 shows images from a few classes of the dataset.

Fig. 1. Images from Dataset

Table I. DATASET BREAKUP

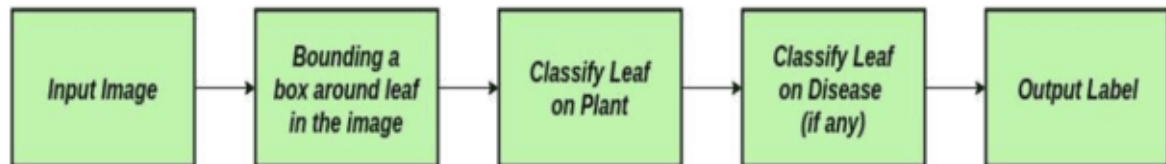
Leaf Category	Images
Apple	3171
Cherry	1906
Grape	4062
Peach	2657
Pepper	2475
Potato	2152
Strawberry	1565
Tomato	18170
Total	36148

3.1.2. Data Preprocessing and Augmentation Image augmentation plays a key role in building an effective image classifiers. Though datasets may contain anywhere from hundreds to a couple of thousand training examples, the variety might still not be enough to build an accurate model. Some of the many image augmentation options are flipping the image vertically/horizontally, rotating through various angles and scaling the image. These augmentations help increase the relevant data in a dataset. The size of each image in the PlantVillage dataset is found to be 256x256 pixels. The data processing and image augmentation are done using the Keras deep-learning framework. The augmentation options used for training are as follows: Rotation - To rotate a training image randomly over various angles Brightness - Helps the model to adapt to variation in lighting while feeding images of varying brightness during training Shear - Adjust the shearing angle of the image in clockwise/counter-clockwise direction Zoom - Provides the input image scaled by various factors Vertical/Horizontal Flip - The image is randomly flipped about

the vertical/horizontal axis.

3.1.3. System Overview The whole process is divided into 3 stages as in Fig. 2:

1. An input image is initially taken, A You Only Look Once (YOLOv3)[20] object detector is run over the input image to obtain the coordinates of bounding boxes around leaves present in the image, if any. The detector divides the input image into a grid and then analyzes every cell to identify features of the target object. The adjacent cells where the features are detected with high confidence are then put together to produce the output of the model.
2. The leaves are then cropped out of the image using the OpenCV library using the given coordinates. These extracted images are passed as input to a CNN Classifier which classifies the input into the 8 classes of plants from the dataset.



3. 8 CNN classifiers are trained to identify the diseases of each of the 8 plant classes. The result from stage 2 is used to call the classifier that has been trained to classify the different diseases for that plant. If there are none, the leaf would be classified as 'Healthy'. Every stage in this end-to-end system was trained on the same dataset to provide better accuracy and consistency.

Fig 2: Block Diagram Of Identification System

3.2 Image Processing

- **Image Processing** Doing study on the disease severity or harshness of leaf using image processing techniques. They used feature extraction methods such as threshold and triangular threshold methods. Identification of diseased leaf of brown and blast spot of leaf image processing techniques were carried out by. They used zooming algorithm, SOM neural network for disease detection. The authors made investigation on diseases (for e.g. Early scorch, Ashen mould, Late scorch, Cottony mold and Ting whiteness diseases) of plants using K- Means clustering, Back propagation algorithm. Made study on diseases using image processing techniques which involves morphological processing, colour clustering.

Leaf disease detection of orchid leaf (for e.g.) such as Black leaf spot and Sun scorch was carried out by. They applied pattern classification and border segmentation techniques for detection of diseased on leaf. The present work has been carried out for the automatic disease detection of plant leaf and provide solution about the diseases that solution can be provided using the deep study about the diseases and the soil moisture level detection and also the temperature level detection whether that crop get affected by which reason exactly to determined solution.

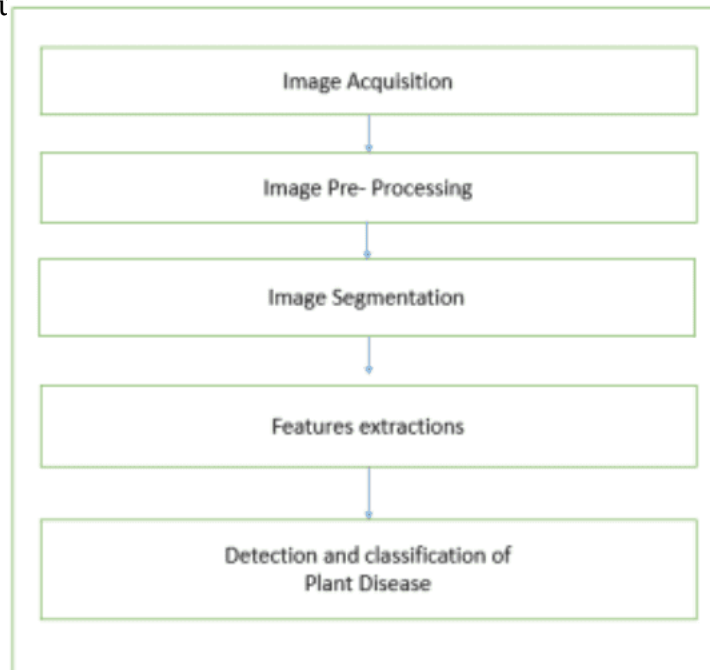


Fig 3: Image Processing

- **Image acquisition** Image acquisition process is use to capturing the images with the help of digital camera. Our study focused on the diseased images of leaf which were stored in digital media for further MATLAB operations.
- **Image Pre- processing** Image Pre- processing is carried out to improve the quality of the image and remove the unwanted noise in image followed by clipping and smoothing of the image. The RGB images can be converted into grey scale images using colour conversion by the following formula. The image enhancement is carried out to increase the contrast. $(x) = 0.2989 \cdot R + 0.5870 \cdot B + 0.114 \cdot B$ Then histogram equalization is applied in which the intensity of the image is distributed using cumulative distribution

function.

- **Image Segmentation** This method is used for the conversion of digital image into various segments having some similarity. Image segmentation helps in the detection of objects and boundary line of the image. In our study K- mean clustering is done for classification of objects based on a set of features into K number of classes. The classification is done by minimizing sum of squares of distance between data objects and corresponding cluster.
- **Feature Extraction** In feature extraction method features such as colour, texture, morphology and structure are used in plant disease detection. Colour co-occurrence method is used in which the texture and colour of the image are considered. The methods used in colour co-occurrence are firstly the RGB image of the leaves are converted into HIS colour space representation. For generation of colour co-occurrence matrix each pixel map is applied which results into three colour co-occurrence matrix one for each of H, S, I [1].

$$X = 0.5 (R-G) + (R-G) Y = (RG)^2 + (R-B) (G-B) = \text{aces}()$$

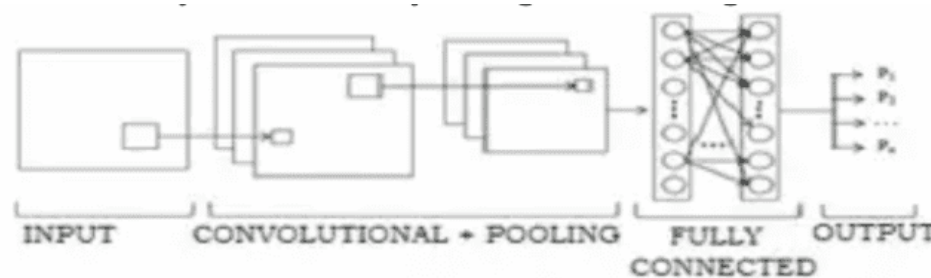
$$H = \arctan\left(\frac{B-G}{R-G}\right) \text{ if } R \neq G \text{ else } 360 - B/G \text{ if } R = G$$

$$S = 1 - \frac{\min(R,G,B)}{\max(R,G,B)} I = (R+G+B)/3$$
- **Classification** Classification is used in the interpretation of the extracted diseased region in an image which helps in the identification of the type of disease infection in leaves. In our analysis back propagation neural network (BPNN) is used which build association between known pattern of input and specific output. The input layer analyses the diseased region while the output layer specifies the disease outcome of the affected region. A hidden layer occurs in between the input and output layer which provides connecting link between the input and output images. It is applied to obtain least error in the classification of disease of the affected region.
- **Architectural Design** The smart agriculture model main aim to provide solution about leaf disease using sensors. It is low cost and efficient system is shown below. It includes Controller, Arduino, sensors like soil moisture and Temperature sensor. **Techniques on image processing** Diseases in plants cause major production and economic losses in agricultural trade worldwide. observation of health and detection of diseases in plants and trees is vital for property agriculture. To the most effective of our data, there's no device commercially accessible for period assessment of health

conditions in trees. The classification strategies are often seen as extensions of the detection strategies, however rather than attempting to observe just one specific sickness totally different conditions and symptoms, these ones attempt to determine and label whichever pathology has effects on the plant. **Neural Networks** This is the strategy to segmentation of the photographs into leaf and background within the following variety of size and colour options are extracted from each the RGB and HSI representations of the image. Those parameters are finally fed to neural networks and applied mathematics classifiers that are accustomed confirm the plant condition

3.3. CNN

Deep



learning is a class of machine learning algorithms that has sequential layers. Each layer uses the output of the preceding layer as input. The learning process can be unsupervised, supervised or semi-supervised. LeCun et al. define the deep learning as a representation learning method. Representation learning algorithms makes optimizations to find the most convenient way to represent the data. Deep learning does not have to divide the feature extraction and the classification because the model automatically extracts the features while training the model. It is used in many research areas such as image processing, image restoration, speech recognition, natural language processing and bioinformatics. CNN is preferred as a deep learning method in this study. CNN, which can easily identify and classify objects with minimal pre-processing, is successful in analyzing visual images and can easily separate the required features with its multi layered structure. It consists of four main layers: convolutional layer, pooling layer, activation function layer and fully connected layer. Fig. 1 shows a general CNN architecture.[25]

Fig. 4. A general CNN architecture

3.3.1.Convolutional Layer

CNN takes its name from the convolution layer. In this layer, a series of mathematical operations are performed to extract the feature map of the input image. The input image is reduced to a smaller size using a filter. The filter is shifted step by step starting from the upper left corner of the image. At each step, the values in the image are multiplied by the values of the filter and the result is summed. A new matrix with a smaller size is created from the input image. Fig. 2 shows the convolution operation in the convolution layer for a 5x5 input image and a 3x3 filter.

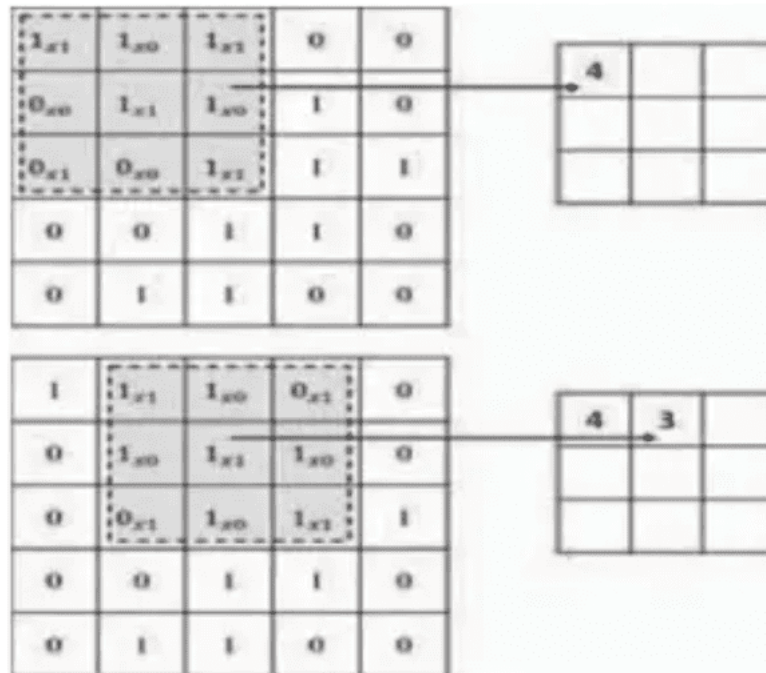
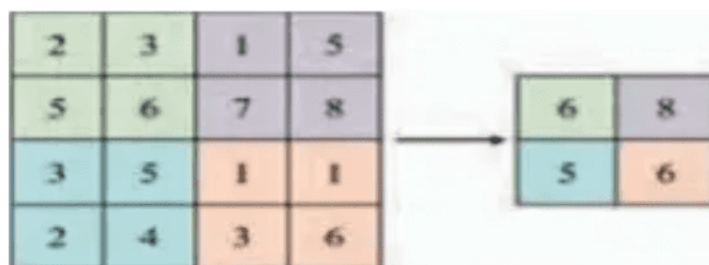


Fig. 5.

Convolution operation for 5x5 input image and 3x3 filter.

3.3.2.Pooling Layer

The pooling layer is usually applied after the convolution layer. The size of the output matrix obtained from the



convolution layer is reduced in this layer. Although filters of different sizes can be used in the pooling layer, generally 2x2 size filter is used. Several functions such as max pooling, average pooling and L2-norm pooling can be used in this layer. Max pooling is done by selecting the largest value in the sub windows and this value is transferred to in a new matrix. Fig. 3 shows an example pooling operation.

Fig. 6. Max pooling with 2x2 filters and stride 2

3.3.3.Activation Layer

In artificial neural networks, the activation function provides a curvilinear relationship between the input and output layers. It affects the network performance. Non-linear learning of the network occurs through the activation function. Several activation functions, such as linear, sigmoid, hyperbolic tangent, exist, but the nonlinear ReLU (Rectified Linear Unit) activation function is usually used in CNN. In ReLU, values less than zero are changed to zero, while values greater than zero are unchanged .

$$f(x) = 0, \text{ if } x < 0.$$

$$f(x) = x, \text{ otherwise.}$$

3.3.4. Fully Connected Layer

The last obtained matrix, after finishing the convolution, pooling and activation operations, is fed into the fully connected layer as input. Recognition and classification are performed in this layer[17]

3.4 ResNet -152

Residual Network (ResNet) is a Convolutional Neural Network (CNN) architecture which can support hundreds or more convolutional layers. ResNet can add many layers with strong performance, while previous architectures had a drop off in the effectiveness with each additional layer.

ResNet proposed a solution to the “vanishing gradient” problem. Neural networks train

via backpropagation, which relies on gradient descent to find the optimal weights that minimize the loss function. When more layers are added, repeated multiplication of their derivatives eventually makes the gradient infinitesimally small, meaning additional layers won't improve the performance or can even reduce it.

ResNet solves this using “identity shortcut connections” – layers that initially don't do anything. In the training process, these identical layers are skipped, reusing the activation functions from the previous layers. This reduces the network into only a few layers, which speeds learning. When the network trains again, the identical layers expand and help the network explore more of the feature space.

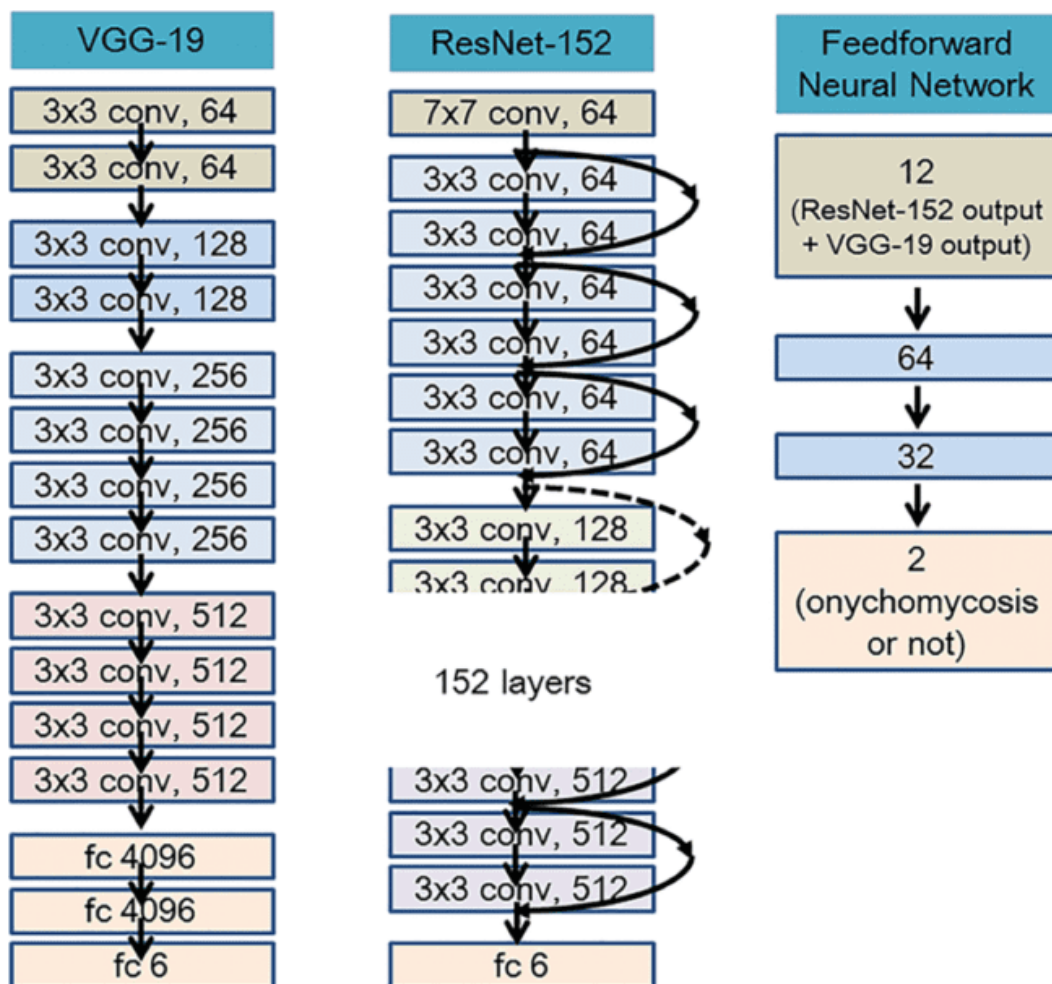


Fig. 7. ResNet-152 Architecture

ResNet was the first network demonstrated to add hundreds or thousands of layers while outperforming shallower networks. Although since its introduction in 2015, newer

architectures have been invented which beat ResNet's performance, it is still a very popular choice for computer vision tasks. A primary strength of the ResNet architecture is its ability to generalize well to different datasets and problems.[21]

3.4.1 ResNet Variations You Can Use on PyTorch : PyTorch supports one ResNet variation, which you can use instead of the traditional ResNet architecture, which is **DenseNet**. DenseNet uses shortcut connections to connect all layers directly with each other. The input of each layer is the feature maps of all earlier layer. Feature maps are joined using depth-concatenation. Concatenating feature maps can preserve them all and increase the variance of the outputs, encouraging feature reuse.

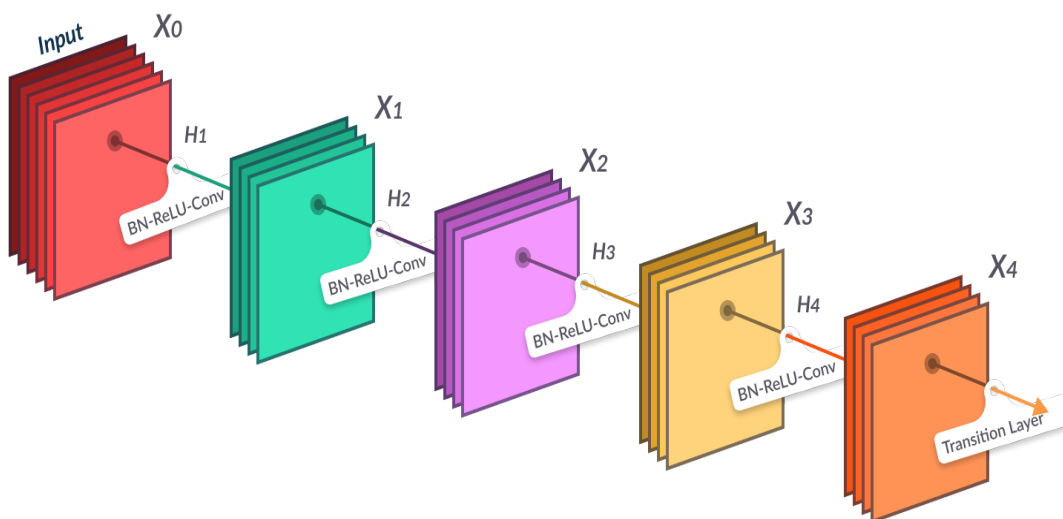


Fig. 8. Layers of ResNet-152

3.4.2 Options for Running ResNet on PyTorch : Built-In PyTorchResNet

```
torchvision.models.resnet18(pretrained=False, **kwargs)
torchvision.models.resnet34(pretrained=False, **kwargs)
torchvision.models.resnet50(pretrained=False, **kwargs)
torchvision.models.resnet101(pretrained=False, **kwargs)
torchvision.models.resnet152(pretrained=False, **kwargs)
torchvision.models.densenet121(pretrained=False, **kwargs)
torchvision.models.densenet161(pretrained=False, **kwargs)
torchvision.models.densenet169(pretrained=False, **kwargs)
torchvision.models.densenet201(pretrained=False, **kwargs)
```

Implementation: torchvision.models PyTorch provides torchvision.models, which include multiple deep learning models, pre-trained on the ImageNet dataset and ready to use. Pre-training lets you leverage transfer learning – once the model has learned many objects, features, and textures on the huge ImageNet dataset, you can apply this learning to your own images and recognition problems. torchvision.models include the following ResNet implementations: ResNet-18, 34, 50, 101 and 152 (the numbers indicate the numbers of layers in the model), and Densenet-121, 161, 169, and 201. The following classes allow you to access ResNet models in PyTorch:[13]

The pretrained parameter specifies whether the model weights should be randomly initialized (false) or pre-trained on ImageNet (true). You can supply your own Python keyword arguments.

3.4.3 Coding a ResNet Architecture Yourself Using PyTorch : If you want to create a different ResNet architecture than the ones built into PyTorch, you can create your own custom implementation of ResNet. Liu Kuang created an extensive code example that shows how to implement the building blocks of ResNet in PyTorch. Here is a PyTorch ResNet example of how to create the basic identity block:

```

class BasicBlock(nn.Module):
    expansion = 1

    def __init__(self, in_planes, planes, stride=1):
        super(BasicBlock, self).__init__()
        self.conv1 = nn.Conv2d(in_planes, planes, kernel_size=3, stride=stride,
padding=1, bias=False)
        self.bn1 = nn.BatchNorm2d(planes)
        self.conv2 = nn.Conv2d(planes, planes, kernel_size=3, stride=1,
padding=1, bias=False)
        self.bn2 = nn.BatchNorm2d(planes)

        self.shortcut = nn.Sequential()
        if stride != 1 or in_planes != self.expansion*planes:
            self.shortcut = nn.Sequential(
                nn.Conv2d(in_planes, self.expansion*planes, kernel_size=1,
stride=stride, bias=False),
                nn.BatchNorm2d(self.expansion*planes)

```

```

class ResNet(nn.Module):
    def __init__(self, block, num_blocks, num_classes=10):
        super(ResNet, self).__init__()
        self.in_planes = 64

        self.conv1 = nn.Conv2d(3, 64, kernel_size=3, stride=1, padding=1,
bias=False)
        self.bn1 = nn.BatchNorm2d(64)
        self.layer1 = self._make_layer(block, 64, num_blocks[0], stride=1)
        self.layer2 = self._make_layer(block, 128, num_blocks[1], stride=2)
        self.layer3 = self._make_layer(block, 256, num_blocks[2], stride=2)
        self.layer4 = self._make_layer(block, 512, num_blocks[3], stride=2)
        self.linear = nn.Linear(512*block.expansion, num_classes)

```

And here is how the entire ResNet network is structured using this and other building blocks:

3.4.4.Using ResNet50 with Transfer Learning : A simple way to perform transfer learning with PyTorch's pre-trained ResNets is to switch the last layer of the network with one that suits your requirements. Here is how to do this, with code examples

by Prakash Jain. The process is to freeze the ResNet layer you don't want to train and pass the remaining parameters to your custom optimizer.

```
model_conv=torchvision.models.resnet50(pretrained=True)
```

Loading the model:

```
num_fts = model_conv.fc.in_features  
model_conv.fc = nn.Linear(num_fts, n_class)
```

Change the first layer:

```
for name, child in model_conv.named_children():  
    for name2, params in child.named_parameters():  
        print(name, name2)
```

The `model_conv` object has child containers, each with its own children which represent the layers. Here is how to freeze the last layer for ResNet50:

```
conv1 weight  
bn1 weight  
bn1 bias  
fc weight  
fc bias
```

This displays a long list of parameters, including these:

```

if ct < 7:
    for name2, params in child.named_parameters():
        params.requires_grad = False

```

```

ct = 0
for name, child in model_conv.named_children():
    ct += 1

```

Here is how to freeze one or more layers. For example, to freeze the first 7 layers:

To adapt the layer to fit your data, consider how the underlying layers are represented.

3.4.5 Scaling ResNet on PyTorch : We explained the basics of ResNet and provided two ways to run ResNet on PyTorch: pre-trained models in the `pytorch.vision` modules, or by coding ResNet components yourself directly in PyTorch. We also reviewed a simple application of transfer learning with ResNet-50. Training ResNet is extremely computationally intensive and becomes more difficult the more layers you add. Instead of waiting for hours or days for training to complete, use the MissingLink deep learning framework to automate the process:

- **Define a group of machines** either on-premises or in the cloud, and automatically run deep learning jobs on them
- **Scale out ResNet** automatically across numerous machines
- **Avoid idle time** by scheduling a queue of experiments and utilizing machines to the max

3.4.6 Deep Residual Learning for Image Recognition : Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers---8x deeper than VGG nets but still having lower complexity. An

ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers. The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions, where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation. A pre-trained model has been previously trained on a dataset and contains the weights and biases that represent the features of whichever dataset it was trained on. Learned features are often transferable to different data. For example, a model trained on a large dataset of bird images will contain learned features like edges or horizontal lines that you would be transferable your dataset. Pre-trained models are beneficial to us for many reasons. By using a pre-trained model you are saving time. Someone else has already spent the time and compute resources to learn a lot of features and your model will likely benefit from it.[15]

4.SYSTEM SPECIFICATION REQUIREMENTS

4.1.Hardware Requirements:

Processor	:	NVIDIA GEFORCE DRIVER
Hard Disk	:	500GB
RAM	:	4GB or more

4.2.Software Requirements:

Operating System	:	Windows XP/2003 or MacOS
Programming Language	:	Python 3
IDE/Workbench	:	Anaconda, Jupyter Notebook

5. System Design

5.1.Introduction

The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affecting the quality of the software; it has a major impact on the later phase, particularly testing, maintenance. The output of this phase is the design document. This document is similar to a blueprint for the solution and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phases System Design and Detailed Design.

System Design also called top-level design aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of the system design all the major data structures, file formats, output formats, and the major modules in the system and their specifications are decided.

During, Detailed Design, the internal logic of each of the modules specified in system design is decided. During this phase, the details of the data of a module is usually specified in a high-level design description language, which is independent of the target language in which the software will eventually be implemented.

In system design the focus is on identifying the modules, whereas during detailed design the focus is on designing the logic for each of the modules. In other words, in system design the attention is on what components are needed, while in detailed design how the components can be implemented in software is the issue.

Design is concerned with identifying software components specifying relationships among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is minimal clearly specified.

During the system design activities, Developers bridge the gap between the requirements specification, produced during requirements elicitation and analysis, and the system that is delivered to the user.

Design is the place where the quality is fostered in development. Software design is a process through which requirements are translated into a representation of software.

6. System Testing

6.1 Testing Methodologies

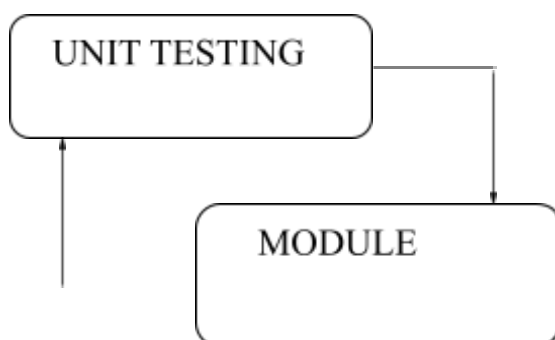
Testing is the process of finding differences between the expected behavior specified by system models and the observed behavior implemented system. From modeling point of view , testing is the attempt of falsification of the system with respect to the system models. The goal of testing is to design tests that exercise defects in the system and to reveal problems.

The process of executing a program with intent of finding errors is called testing. During testing , the program to be tested is executed with a set of test cases , and the output of the program for the test cases is evaluated to determine if the program is performing as expected . Testing forms the first step in determining the errors in the program. The success of testing in revealing errors in program depends critically on test cases.

Strategic Approach to Software Testing:

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirements analysis where the information domain , functions , behavior , performance , constraints and validation criteria for software are established. moving inward along the spiral , we come to design and finally to coding . To develop computer software we spiral in along streamlines that decreases the level of abstraction on each item.

A Strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing will progress by moving outward along the spiral to integration testing , where the focus on the design and the concentration of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed . Finally we arrive at system testing , where the software and other system elements are tested as a whole .



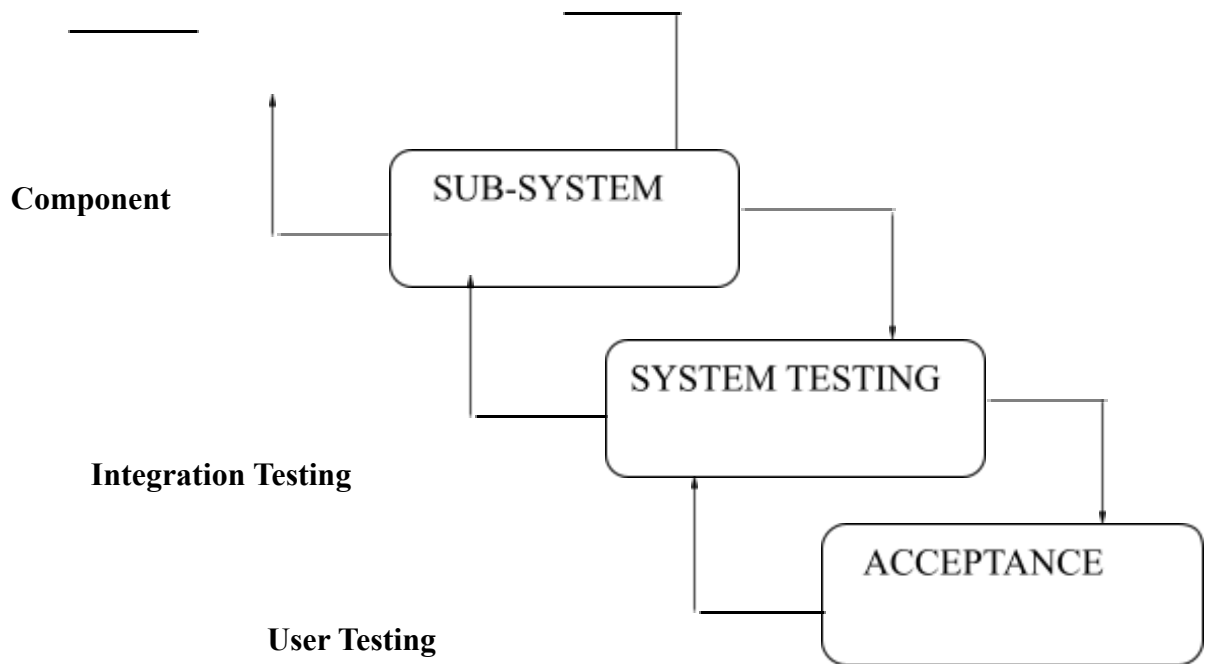


Fig. 9.Different Levels of Testing

Client Needs
Requirements
Design
Code

Acceptance Testing
System Testing
Integration Testing
Unit Testing

Testing is the process of finding difference between the expected behavior specified by system models and the observed behavior of the implemented system.

Testing Activities

Different levels of testing are used in the testing process , each level of testing aims to test different aspects of the system. the basic levels are:

Unit testing

Integration testing

System testing

Acceptance testing

Unit Testing

Unit testing focuses on the building blocks of the software system, that is, objects and sub system. There are three motivations behind focusing on components. First, unit testing reduces the complexity of the overall tests activities, allowing us to focus on smaller units of the system. Second, unit testing makes it easier to pinpoint and correct faults given that few components are involved in this test. Third, Unit testing allows parallelism in the testing activities, that is each component can be tested independently of one another. Hence the goal is to test the internal logic of the module.

Integration Testing

In the integration testing, many test modules are combined into sub systems , which are then tested . The goal here is to see if the modules can be integrated properly, the emphasis being on testing module interaction.

After structural testing and functional testing we get error free modules. These modules are to be integrated to get the required results of the system. After checking a module, another module is tested and is integrated with the previous module. After the integration, the test cases are generated and the results are tested.

System Testing

In system testing the entire software is tested . The reference document for this process is the requirement document and the goal is to see whether the software meets its requirements. The system was tested for various test cases with various inputs.

Acceptance Testing

Acceptance testing is sometimes performed with realistic data of the client to demonstrate that the software is working satisfactory. Testing here focus on the external behavior of the system , the internal logic of the program is not emphasized . In acceptance testing the system is tested for various inputs.

Types of Testing

1. Black box or functional testing
2. White box testing or structural testing

Black box testing

This method is used when knowledge of the specified function that a product has been

designed to perform is known . The concept of black box is used to represent a system whose inside workings are not available to inspection . In a black box the test item is a "Black" , since its logic is unknown , all that is known is what goes in and what comes out , or the input and output.

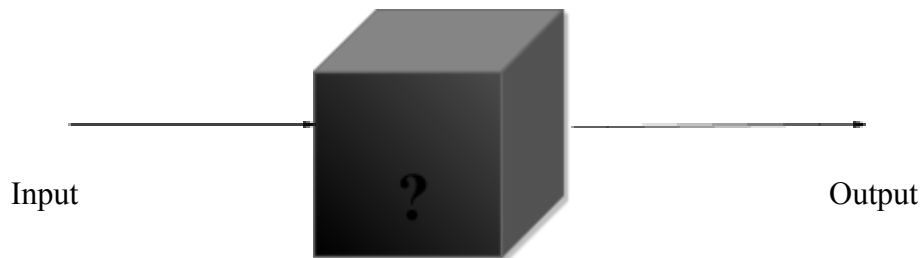
Black box testing attempts to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structure
- Performance errors
- Initialization and termination errors

As shown in the following figure of Black box testing , we are not thinking of the internal workings , just we think about

What is the output to our system?

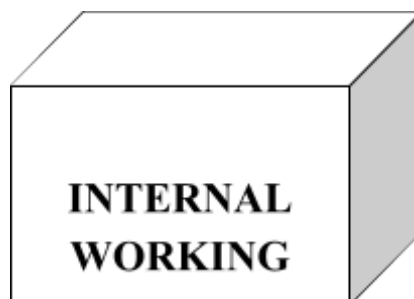
What is the output for given input to our system?



The Black box is an imaginary box that hides its internal workings

White box testing

White box testing is concerned with testing the implementation of the program. the intent of structural is not to exercise all the inputs or outputs but to exercise the different programming and data structure used in the program. Thus structural testing aims to achieve test cases that will force the desire coverage of different structures . Two types of path testing are statement testing coverage and branch testing coverage.





Input

Output

The White Box testing strategy, the internal workings

Test Plan

Testing process starts with a test plan. This plan identifies all the testing related activities that must be performed and specifies the schedules, allocates the resources, and specified guidelines for testing . During the testing of the unit the specified test cases are executed and the actual result compared with expected output. The final output of the testing phase is the test report and the error report.

Test Data:

Here all test cases that are used for the system testing are specified. The goal is to test the different functional requirements specified in Software Requirements Specifications (SRS) document.

Unit Testing:

Each individual module has been tested against the requirement with some test data.

Test Report:

The module is working properly provided the user has to enter information. All data entry forms have tested with specified test cases and all data entry forms are working properly.

Error Report:

If the user does not enter data in specified order then the user will be prompted with error messages. Error handling was done to handle the expected and unexpected errors.

6.2.TEST CASES:

Take a plain network (VGG kind 18 layer network) (Network-1) and a deeper variant of it (34-layer, Network-2) and add Residual layers to the Network-2 (34 layer with residual connections, Network-3).Designing the network:

- Use 3*3 filters mostly.
- Down sampling with CNN layers with stride 2.
- Global average pooling layer and a 1000-way fully-connected layer with Softmax in the end.

There are two kinds of residual connections:

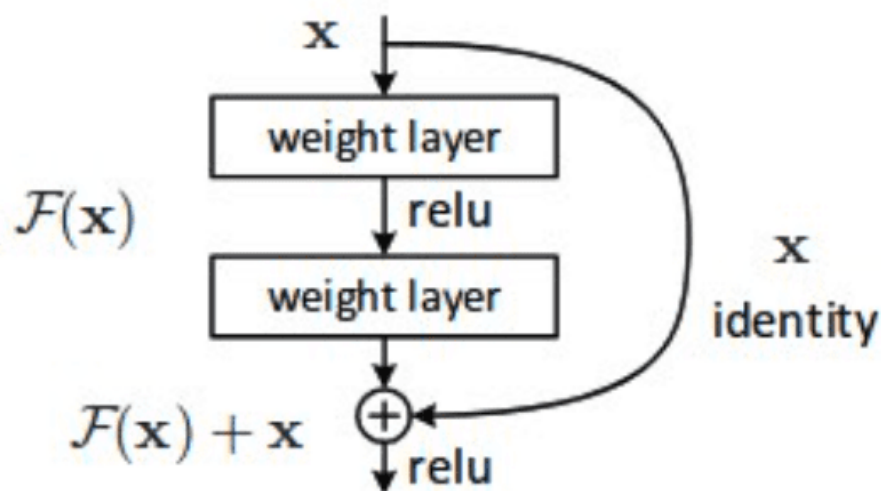


Fig. 9.Residual block

1.The identity shortcuts (x) can be directly used when the input and output are of the same dimensions.

$$y = F(x, \{W_i\}) + x. \quad (1)$$

Residual block function when input and output dimensions are same

$$y = F(x, \{W_i\}) + W_s x. \quad (2)$$

2. When the dimensions change, A) The shortcut still

performs identity mapping, with extra zero entries padded with the increased dimension.

B) The projection shortcut is used to match the dimension (done by 1×1 conv) using the following formula

Residual block function when the input and output dimensions are not same.
The first case adds no extra parameters, the second one adds in the form of $W_{\{s\}}$

6.3.Results:

Even though the 18 layer network is just the subspace in 34 layer network, it still performs better. ResNet outperforms by a significant margin in case the network is deeper

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	25.03

Table 2. Top-1 error (% , 10-crop testing) on ImageNet validation. Here the ResNets have no extra parameter compared to their plain counterparts. Fig. 4 shows the training procedures.

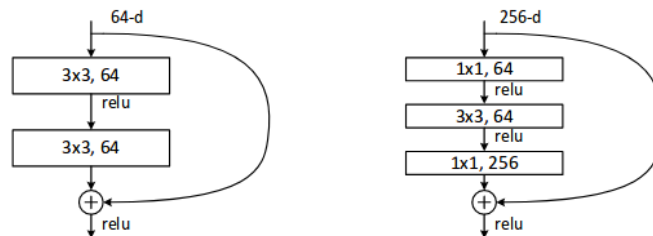
ResNet Model comparison with their counter plain nets

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ResNet Architectures

Table 3 : ResNet Architecture Values

Each ResNet block is either 2 layer deep (Used in small networks like ResNet 18, 34) or 3



layer deep(ResNet 50, 101, 152).

Fig 10. ResNet 2 layer and 3 layer Block

Pytorch Implementation can be seen here:

The Bottleneck class implements a 3 layer block and Basicblock implements a 2 layer block. It also has implementations of all ResNet Architectures with pretrained weights trained on ImageNet.

Observations:

- ResNet Network Converges faster compared to plain counter part of it.
- Identity vs Projection shortcuts. Very small incremental gains using projection shortcuts (Equation-2) in all the layers. So all ResNet blocks use only Identity shortcuts with Projections shortcuts used only when the dimensions changes.

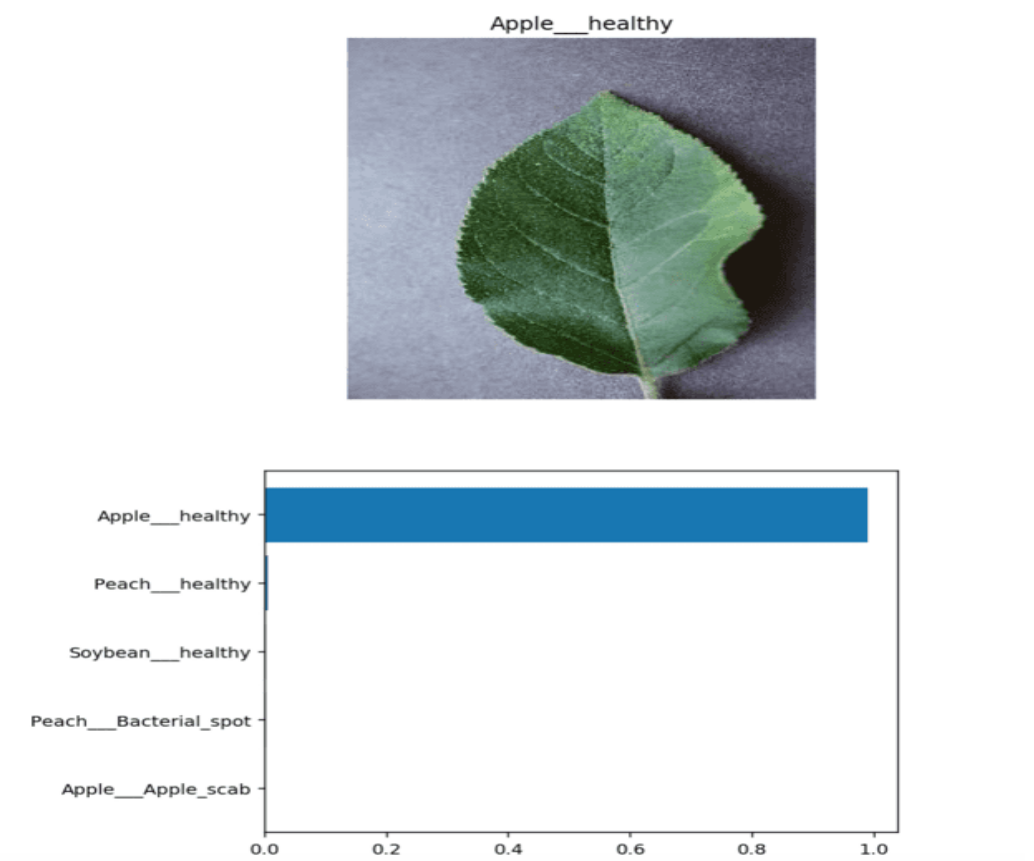
method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43 [†]
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	19.38	4.49

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except [†] reported on the test set).

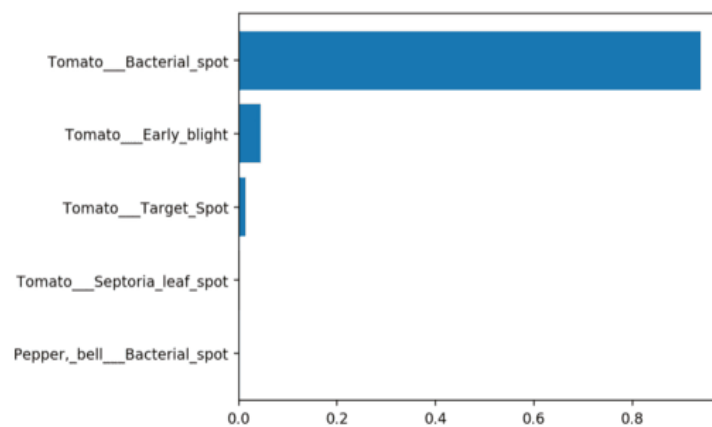
- ResNet-34 achieved a top-5 validation error of 5.71% better than BN-inception and VGG. ResNet-152 achieves a top-5 validation error of 4.49%. An ensemble of 6 models with different depths achieves a top-5 validation error of 3.57%. Winning the 1st place in ILSVRC-2015[16]

7. Output Screens

Fig.11: Result when the above image has been processed in the software



Tomato__Bacterial_spot



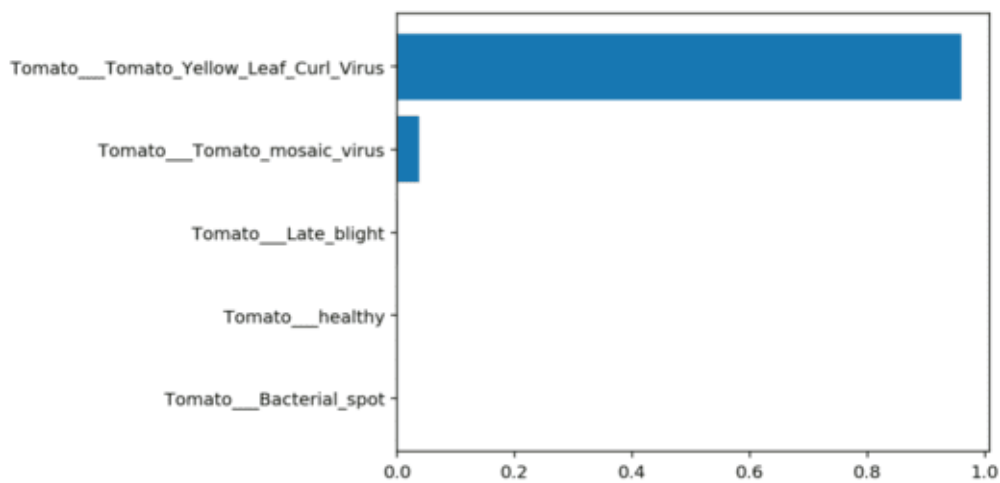
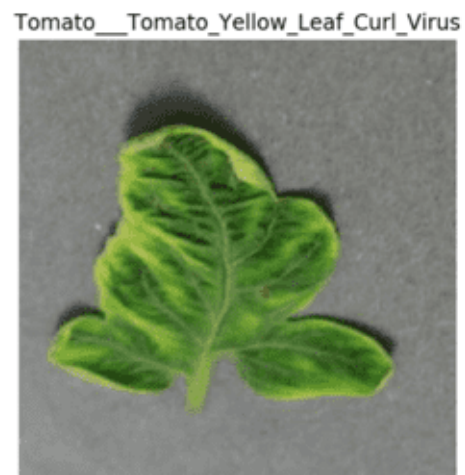


Fig.12: Result when the above image has been processed in the software

Fig.13: Result when the above image has been processed in the software

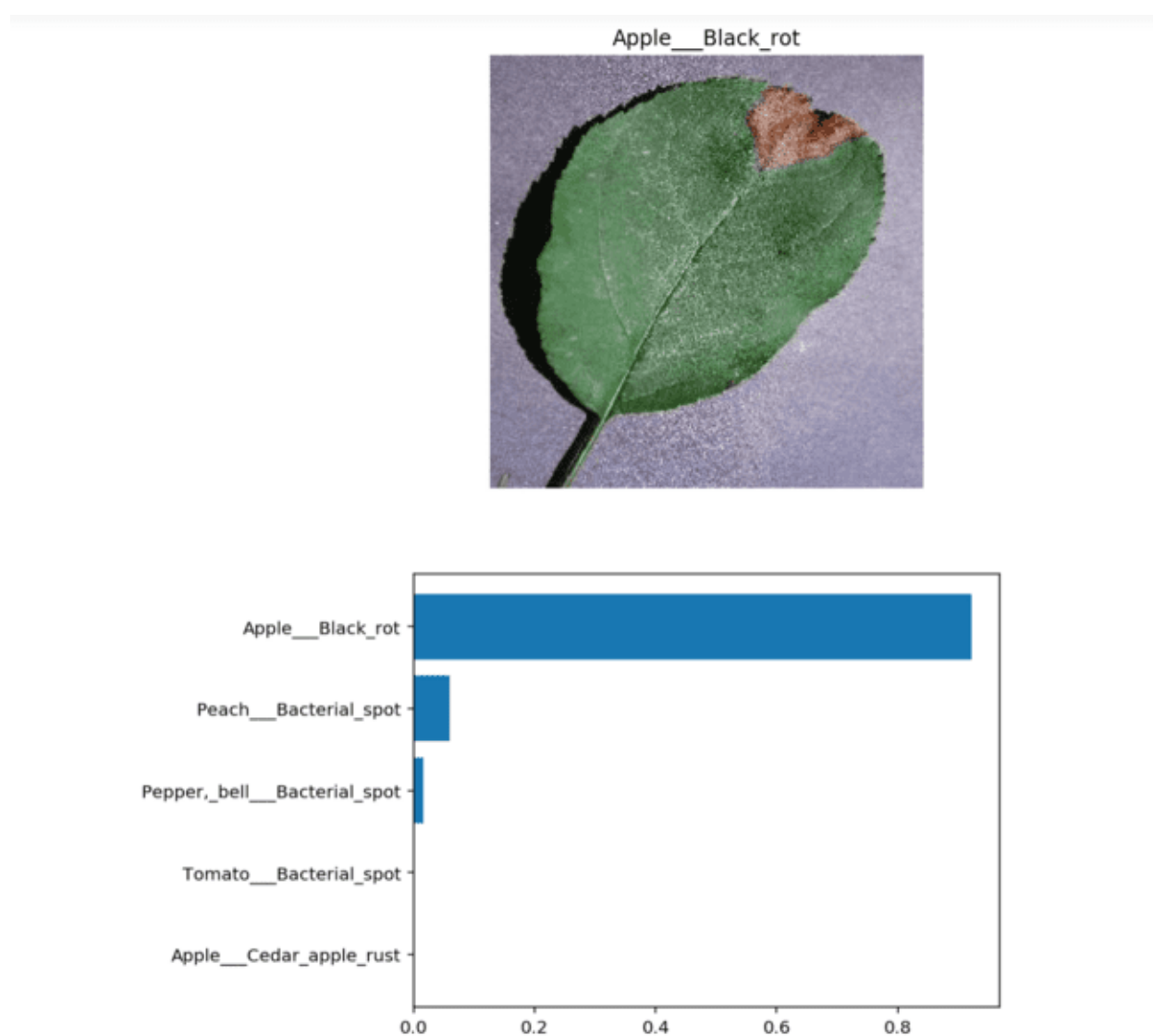


Fig.14: Result when the above image has been processed in the software.

8. Conclusion.

The proposed CNN based leaves disease identification model is capable of classifying four different deficiencies in leaves. Since CNN does not require any tedious preprocessing of input images and hand designed features, faster convergence rate and good training performance, it is preferred for many applications rather than the conventional algorithms. The classification accuracy can be further increased by providing more images in the dataset and tuning the parameters of the CNN model

9. Future Scope

- A. Using Reinforcement learning through time to time feedbacks to improve the accuracy our product.
- B. Work can be stretched out by utilizing quad copter at field level to catch pictures of the leaves of the different plants in the farm
- C. For further handling, this framework can be associated with the server.
- D. Group leaf ailments utilizing picture preparing instruments and send all data about the sickness through the GSM module to the farmer's mobile phone.
- E. Using machine learning algorithms ,recommendation system can be build for various remedies associated with particular leaf disease.

10. References

- [1] of Advanced Research in Electrical, Electronics and Instrumentation Engineering(An ISO 3297: 2007 Certified Organization)Vol. 5, Issue 3, March 2016 Sanjay B. Patil, Dr. Shrikant K. Bodhe, Leaf Disease Severity Measurement Using Image Processing” International Journal of Engineering and Technology Vol.3 (5), 2011, 297-301
- [2] Jundare Manisha, Jundare Pallavi, Jundare Pragati, Prof. C.S.Aryan, “Plant Disease Detection and its Treatment using Image Processing” International Journal
- [3] Prakash M. Mainkar, Shreekant Ghorpade, Mayur Adawadkar, ”Plant Leaf Disease

- Detection and Classification Using Image Processing Techniques” International Journal of Innovative and Emerging Research in Engineering e-ISSN: 2394 – 3343 p-ISSN: 2394 – 5494 [4] Vaibhavi S. Bharwad , Kruti J. Dangarwala,” Recent Research Trends of Plant Disease Detection” International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064 Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611
- [5] Prof. Sanjay B. Dhaygude, Mr.Nitin P.Kumbhar, “Agricultural plant Leaf Disease Detection Using Image Processing” International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 2, ISSN: 2278 – 8875, Issue 1, January 2013
- [6] Surender Kumar , Rupinder Kaur ,” Plant Disease Detection using Image Processing A Review” International Journal of Computer Applications (0975 – 8887) Volume 124 – No.16, August 2015
- [7] Sushil R. Kamlapurkar. “Detection of Plant Leaf Disease Using Image Processing Approach” International Journal of Scientific and Research Publications, Volume 6, Issue 2, February 2016 ISSN 2250-3153
- [8] Y.Sanjana , Ashwath Sivasamy , SriJayanth,” Plant Disease Detection Using Image Processing Techniques” International Journal of Innovative Research in Science, Engineering and Technology (An ISO 3297: 2007 Certified Organization) Vol. 4, Special Issue 6, May 2015
- [9] Jayamala K. Patil , Raj Kumar, “Advances In Image Processing For Detection Of Plant Diseases” Journal of Advanced Bioinformatics Applications and Research ISSN 0976-2604 Vol 2, Issue 2, June2011, pp 135-141
- [10] M.Malathi , K.Aruli , S.Mohamed Nizar , A.Sagaya Selvaraj, “A Survey on Plant Leaf Disease Detection Using Image Processing Techniques” International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 p-ISSN: 2395-0072 Volume: 02 Issue: 09 | Dec-2015
- [11] S.Raj Kumar , S.Sowrirajan,” Automatic Leaf Disease Detection and Classification using Hybrid Features and Supervised Classifier”, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 5, Issue 6,2016..
- [12] Tatem, D. J. Rogers, and S. I. Hay, “Global transport networks and infectious disease spread,” *Advances in Parasitology*, vol. 62, pp. 293–343, 2006. View at Publisher · View at Google Scholar · View at Scopus.
- [13] J. R. Rohr, T. R. Raffel, J. M. Romansic, H. McCallum, and P. J. Hudson, “Evaluating the links between climate, disease spread, and amphibian declines,” *Proceedings of the*

National Academy of Sciences of the United States of America, vol. 105, no. 45, pp. 17436–17441, 2008. View at Publisher · View at Google Scholar · View at Scopus.

[14] T. Van der Zwet, “Present worldwide distribution of fire blight,” in Proceedings of the 9th International Workshop on Fire Blight, vol. 590, Napier, New Zealand, October 2001.

H. Cartwright, Ed., Artificial Neural Networks, Humana Press, 2015.

[15] Steinwart and A. Christmann, Support Vector Machines, Springer Science & Business Media, New York, NY, USA, 2008. View at MathSciNet.

[16] Steinwart and A. Christmann, Support Vector Machines, Springer Science & Business Media, New York, NY, USA, 2008. View at MathSciNet.

[17] S. Sankaran, A. Mishra, R. Ehsani, and C. Davis, “A review of advanced techniques for detecting plant diseases,” Computers and Electronics in Agriculture, vol. 72, no. 1, pp. 1–13, 2010. View at Publisher · View at Google Scholar · View at Scopus.

[18] P. R. Reddy, S. N. Divya, and R. Vijayalakshmi, “Plant disease detection technique tool—a theoretical approach,” International Journal of Innovative Technology and Research, pp. 91–93, 2015. View at Google Scholar.

[19] A.-K. Mahlein, T. Rumpf, P. Welke et al., “Development of spectral indices for detecting and identifying plant diseases,” Remote Sensing of Environment, vol. 128, pp. 21–30, 2013. View at Publisher · View at Google Scholar · View at Scopus.

[20] Alom M. Z., Taha T. M., Yakopcic C., Westberg S., Hasan M., Eses B. C. V., et al. (2018). The history began from alexnet: a comprehensive survey on deep learning approaches. arXiv:1803.01164. [[Google Scholar](#)]

[21] Atabay H. A. (2017). Deep residual learning for tomato plant leaf disease identification. J. Theor. Appl. Inform. Technol. 95, 6800–6808. [[Google Scholar](#)]

[22] Barbedo J. G. A. (2016). A review on the main challenges in automatic plant disease identification based on visible range images. Biosyst. Eng. 144, 52–60. 10.1016/j.biosystemseng.2016.01.017 [[CrossRef](#)] [[Google Scholar](#)]

[23] Barbedo J. G. A. (2018a). Factors influencing the use of deep learning for plant disease recognition. Biosyst. Eng. 172, 84–91. 10.1016/j.biosystemseng.2018.05.013 [[CrossRef](#)] [[Google Scholar](#)]

[24] Barbedo J. G. A. (2018b). Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. Comput. Electron. Agric. 153, 46–53. 10.1016/j.compag.2018.08.013 [[CrossRef](#)] [[Google Scholar](#)]

- [25] Bassil K., Vakil C., Sanborn M., Cole D., Kaur J., Kerr K. (2007). Cancer health effects of pesticides: systematic review. *Can. Fam. Physician* 53, 1704–1711. [[PMC free article](#)] [[PubMed](#)] [[Google Scholar](#)]
- [26] Bengio Y. (2012). Practical recommendations for gradient-based training of deep architectures. *arXiv:1206.5533*. Available online at: <http://arxiv.org/abs/1206.5533>
- [27] Brahimi M., Arsenovic M., Laraba S., Sladojevic S., Boukhalfa K., Moussaoui A. (2018). Deep learning for plant diseases: detection and saliency map visualisation, in *Human and Machine Learning*, eds Zhou J., Chen F. (Cham: Springer International Publishing;), 93–117. [[Google Scholar](#)]
- [28] Brahimi M., Boukhalfa K., Moussaoui A. (2017). Deep learning for tomato diseases: classification and symptoms visualization. *Appl. Artif. Intell.* 31, 299–315. 10.1080/08839514.2017.1315516 [[CrossRef](#)] [[Google Scholar](#)]
- [29] Buda M., Maki A., Mazurowski M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Netw.* 106, 249–259. 10.1016/j.neunet.2018.07.011 [[PubMed](#)] [[CrossRef](#)] [[Google Scholar](#)]

11. Appendices:

- (1) Practical Convolutional Neural Networks: Implement Advanced Deep Learning Models Using Python Book by Md. Rezaul Karim, Mohit Sewak, and Pradeep Pujari
- (2) Python Deep Learning: Develop Your First Neural Network in Python Textbook by Samuel Burns
- (3) Wikipedia, URL: <http://www.wikipedia.org>.
- (4) Answers.com, Online Dictionary, Encyclopedia and much more, URL: <http://www.answers.com>
- (5) Google, URL: <http://www.google.co.in>
- (6) Project Management URL: <http://www.startwright.com/project.html>

(7) IEEE Xplore Digital Library URL: <https://ieeexplore.ieee.org/Xplore/home.jsp>

(8) Kaggle URL: <https://www.kaggle.com/pytorch/resnet152#resnet152.pth>

(9) GitHub URL: <https://github.com/>

PROGRAM OUTCOMES

After the completion of the program, a successful student will be able to:

1. Acquire knowledge of mathematics, sciences and concepts of Computer Sciences and Engineering.
2. Ability to perform analysis of electronic systems, computer systems and software systems to meet the requirements.
3. Design and develop computer, software, mobile, embedded systems and high performance computing systems.
4. Skills to solve problems in hardware and software systems.
5. Use of computer science principles and modern tools to computing systems engineering practice.
6. Create solutions of social context the impact of Computer Science and Systems Engineering.
7. Practice computer sciences and engineering in compliance with environmental standards.
8. Follow ethical code of conduct in professional activities.
9. Achieve personal excellence and ability to work in groups.
10. Develop effective communication in professional transactions.
11. Life skills for effective project management.
12. Appreciate the significance and applications of computer science and engineering and to engage in lifelong learning for knowledge and skill upgradation.

PROGRAM SPECIFIC OUTCOMES

On successful completion of the Program, the graduates will be able to

1. Acquire knowledge of mathematics, Computer Science and Systems Engineering to solve complex engineering problems.
2. Identify, Analyze, Design among alternatives and Develop software for applications and systems in the domain of Computers and its based Systems to meet the societal needs.
3. Use the research-based knowledge and methods to solve realworld problems in the fields of Computer Science and Systems Engineering.
4. Apply appropriate techniques, use modern programing languages, and packages to simulate and develop software by thoroughly understanding the requirements of the system and its constraints in Computer Science and Engineering.

PROGRAM EDUCATIONAL OBJECTIVES

After few years of graduation:

1. Graduate will pursue advanced studies in Computer Science domain and Management.
2. Graduates will be employed in reputed Software Industries and develop Quality Software Systems.
3. Graduates will have career progression through professional skill development, continuing education with ethical attitude.

Mapping of Course Outcomes with Project:

Mapping of Course Outcomes with POs and PSOs:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	✓												✓			
CO2		✓												✓		
CO3			✓												✓	
CO4				✓												✓
CO5					✓											
CO6						✓										
CO7							✓									
CO8								✓								

CC	CO4, CO8	Algorithmic approach	✓
CC	CO3, CO8	Simulation Model/Experiments Design	✓
CC	CO2	Identification of metrics related to the work	✓
CC	CO2, CO5	Comparison of results with rival methods	✓
	CO6	Society related Problem	✓
	CO9	Self Dependency in Performing Tasks/teamwork	✓
	CO12	Is the abstract appropriate to the project?	✓
	CO1	Introduction & literature Review	✓
	CO3, CO8	Modeling and Implementation	✓
	CO7	Results for sustainable development	✓
	CO11	Feasibility study ,Cost Model	✓
	CO10, CO12	Conclusions and Future work	✓

