A

Project Report

Entitled

Plant Disease Detection using Deep Learning

Submitted to the Department of Electronics Engineering in Partial Fulfilment for the Requirements for the Degree of

Bachelor of Technology

(Electronics and Communication)

: Presented & Submitted By :

Panyam Nagachandra Mouli, Vari Shiva Ram, VSS Neeraj Varma

Roll No. (U19EC031, U19EC067, U19EC077)

B. TECH. IV(EC), 8th Semester

: Guided By :

Dr. Jigisha N. Patel

Associate Professor, DoECE

(Year: 2022-23)

DEPARTMENT OF ELECTRONICS ENGINEERING

SARDAR VALLABHBHAI NATIONAL INSTITUTE OF TECHNOLOGY

Surat-395007, Gujarat, INDIA.

Sardar Vallabhbhai National Institute Of Technology

Surat - 395 007, Gujarat, India

DEPARTMENT OF ELECTRONICS ENGINEERING

CERTIFICATE

This is to certify that the Project Report entitled "**Plant Disease Detection using Deep Learning**" is presented & submitted by Panyam Nagachandra Mouli, Vari Shiva Ram, VSS Neeraj Varma, bearing Roll No. U19EC031, U19EC067, U19EC077, of B.Tech. IV, 8^{th} Semester in the partial fulfillment of the requirement for the award of B.Tech. Degree in Electronics & Communication Engineering for academic year 2022-23.

They have successfully and satisfactorily completed their **Project Exam** in all respects. We, certify that the work is comprehensive, complete and fit for evaluation.

Dr. Jigisha N. Patel
Associate Professor& Project Guide

PROJECT EXAMINERS:

Name of Examiners	Signature with Date
1. _____	_____
2. _____	_____
3. _____	_____
4. _____	_____
5. _____	_____

Dr. (Mrs.) Rasika Dhavse
Head, DoECE, SVNIT

Seal of The Department
(May 2023)

Acknowledgements

This acknowledgement is a mark of gratitude for everyone who helped and encouraged us with their efforts. We would like to take immense pleasure to thank each and every one who helped us with this project “Plant Disease Detection using Deep Learning”.

We are indebted to many individuals who, with their support and encouragement have contributed to the report. We express sincere gratitude to our guide Dr. Jigisha N. Patel for her generous mentoring and for providing expert views on the topic. Many helpful comments and corrections are incorporated in the report with deep appreciation. We also thank Dr. Rasika N. Dhavse, Head of Department, DECE and Dr. Jigisha N. Patel, Dr. Swetha Shah, Coordinator, BTech (IV) for full cooperation as and when needed. Finally, We are thankful to everyone who directly or indirectly helped in this project.

Panyam Nagachandra Mouli
Vari Shiva Ram
VSS Neeraj Varma
Sardar Vallabhbhai National Institute of Technology
Surat

May 2023

Abstract

The advancement of agricultural technology has opened up new possibilities for using image processing techniques to evaluate the quality of plants. In response, this project aims to develop a plant disease detection model using deep learning techniques. To achieve this objective, the project begins by analyzing and comparing various standard machine learning algorithms. The project then proceeds to develop a deep learning model based on AlexNet and Convolutional Neural Networks, and evaluates the models based on accuracy and confusion matrix.

This report explores several studies conducted on the topic of plant disease detection using both machine learning and deep learning approaches. The focus is primarily on the use of Convolutional Neural Networks (CNNs) and Image Processing techniques in these studies. Then Machine learning algorithms like Support Vector Machine, Linear Discriminant Analysis and Random Forest Classifier etc, and compared their accuracies and tested the data with highest accurate algorithm which was Random Forest Classifier. Next we proceeded to use deep learning and employed an AlexNet CNN Model and trained the model for all the plants in PlantVillage dataset and also made a website using FastAPI and ReactJS to detect the disease of a Potato leaf. We have also created a modified model and compared the both models and we observed the performance of AlexNet for various epochs. For all the plant diseases the AlexNet model achieved high accuracy of 96% in detecting the diseases of all the plants compared to modified CNN which obtained only 87%. Overall, this project contributes to the development of automated, accurate, and accessible tools for plant disease detection, and provides a valuable resource for farmers, researchers, and the wider agricultural community.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
API	Application Programming Interface
CART	Classification and Regression Trees
CNN	Convolutional Neural Network
CPU	Central Processing unit
CUDA	Compute Unified Device Architecture
DNN	Deep Neural Network
FC	Fully Connected
GPU	Graphics Processing Unit
HCML	Human Centered concept and Machine Learning
HSV	Hue Saturation Value
KNN	K Nearest Neighbors
LDA	Linear discriminant analysis
LSTM	Long Short Term Memory
ML	Machine Learning
MLP	Multi-Layer Perceptron
NB	Naive
OS	Operating System
RAM	Random Access Memory
R-CNN	Region-Based Convolutional Neural Network
ReLU	Rectified Linear Unit
RF	Random Forest
RNN	Recurrent Neural Network
SSD	Single Shot Detector
SVM	Support Vector Machine
VGG	Visual Geometry Group
YOLO	You Only Look Once

Chapter 1
Introduction

1.1 Background

Agricultural technology has made it possible to use image processing techniques to assess the quality of agricultural products. Researchers have been working on identifying and diagnosing issues such as weed infestations, crop diseases, and insect pests, and have achieved significant progress in these areas. It is important to determine the level of plant diseases for effective management, particularly because disease can significantly impact crop yield.

Figure 1.1: Plant Disease Detection [1]

In most cases, agricultural and forestry experts are used to identify on-site or farmers identify fruit tree diseases and pests based on experience. This method is not only subjective, but also time-consuming, laborious, and inefficient. Farmers with less experience may misjudgment and use drugs blindly during the identification process. Quality and output will also bring environmental pollution, which will cause unnecessary economic losses. To counter these challenges, research into the use of image processing techniques for plant disease recognition has become a hot research topic.

1.2 Motivation

The main motivation behind the project is the number of plant diseases that have been increasing daily due to a variety of environmental and man made issues. Today, plant diseases still have both economic and social effects. Farmers spend a great deal of money on disease management, mostly without correct and much required technical

support, which results in poor disease control. Plant diseases can destroy natural ecosystems, increase environmental problems caused the loss of habitat and poor management of the available land resource. There has not been an optimal solution to detect a particular plant disease in the early stages and hence we intend to take up the issue and provide the best solution to the farmers. With the growth of machine learning in recent years, it has provided a wide spectrum of applications to work on and working on plant disease classification is one of them. With the growth of machine learning every single day, it gives us a chance to explore the latest machine learning trend namely deep learning and the benefits that it provides us in terms of accuracy or any other performance metrics and learn in which way that these models are better than the previous machine learning models.

1.3 Obective

The main objective of this project is to create a plant disease detection model using deep learning.

- The project analyzes and compares different standard Machine Learning and Deep Learning algorithms.
- To develop a modified CNN architecture.
- To develop a deep learning model based on AlexNet.
- To compare and analyze the results of both the models.
- To create a website and mobile application which uses the created model to predict the disease of the plant based on the image of its leaf.

1.4 Flow of project

- In the First chapter, introduction, motivation and objective for plant disease detection is discussed.
- In the second chapter, various literature and works done on plant disease detection ranging from traditional machine learning methods to modern deep learning methods are reviewed and summarized.
- In the third chapter the theoretical background of deep learning, convolutional neural networks and computer vision is explained.

- In the fourth chapter, model is trained for potato diseases using Machine Learing algorithms, AlexNet model and the obtained results are analyzed and website is created to detect the disease of the plant.
- In the fifth chapter, all the plants in PlantVillage are trained using modified CNN architecture and AlexNet model and both the accuracies are compared and the highest accurate model is implemented on the website and mobile application.
- In the Conclusion and Future Scope chapter, all the obtained accuracies are compared and the summary of the project is discussed along with the future works will be implemented.

.

Chapter 2
Literature Review

2.1 Classic Machine Learning Approaches

There is an obvious presence of marks on leaves, stems, flowers, or fruits for any disease infected plants. Generally, each disease or pest condition presents a unique visible pattern that can be used to uniquely diagnose abnormalities. Usually, the leaves of plants are the primary source for identifying plant diseases, and most of the symptoms of diseases may begin to appear on the leaves.

In the traditional way agricultural or forest experts used to identify diseases, a farmer can also identify fruit tree diseases by experience. But this method is subjective and also time consuming. Farmers who are less experienced might misjudge and blindly use drugs on plants. To overcome these challenges research into the image processing technique of plant disease recognition has risen.

The general process of using traditional image recognition processing technology to identify plant diseases used the K-means clustering method to segment the lesions regions.Then combined the global color histogram (GCH) color coherence vector (CCV) local binary pattern (LBP), and completed local binary pattern (CLBP) was used to extract the color and texture features of apple spots, and three kinds of apple diseases were detected and identified based on improved support vector machine (SVM), and the classification accuracy reached 93%[1].

Chai et al. [15] selected four tomato leaf diseases, including early blight and late blight leaf mildew and leaf spot, and extracted 18 features such as color, texture, and shape information of tomato leaf spot images, using stepwise discriminant and Bayesian discriminant principal component analysis (PCA), respectively. Principal component analysis and fisher discriminant methods were used to extract the characteristic parameters and construct the discriminant model. The accuracy of the two methods reached 94.71% and 98.32%, respectively.

Li and He [16] studied 5 kinds of apple leaf diseases (speckled deciduous disease, yellow leaf disease, round spot disease, mosaic disease, and rust disease) as the research objects and extracted 8 features of the apple leaf spot image, such as color, texture, and shape. The BP neural network model was used to classify and recognize the diseases, and the average recognition accuracy reached 92.6%.

Guan et al. [17] extracted 63 features including morphology, color, and texture features of rice leaf disease spots, and applied step-based discriminant analysis and Bayesian discriminant method to classify and recognize three rice diseases (blast, stripe blight, and bacterial leaf blight) with the highest recognition accuracy of 97.2%.

It can be concluded that studies on plant disease recognition based on traditional image processing technology have achieved certain results, with high accuracy of disease recognition, but there are still deficiencies and limitations. The research links and processes are cumbersome, highly subjective, time-consuming and labor-consuming and these are heavily dependent on spot segmentation and on artificial feature extraction. It is difficult to test the disease recognition performance of the model or algorithm in more complex environments. Therefore, it is of great significance to realize intelligent, rapid, and accurate plant leaf disease recognition.

The latest improvements in computer vision formulated through deep learning have paved the method for how to detect and diagnose diseases in plants by using a camera to capture images as basis for recognizing several types of plant diseases. Militante et al [18] studies an efficient solution for detecting multiple diseases in several plant varieties. The system was designed to detect and recognize several plant varieties, specifically apple, corn, grapes, potato, sugarcane, and tomato. The system can also detect several diseases of plants.

Comprised of 35,000 images of healthy plant leaves and infected with the diseases, the researchers were able to train deep learning models to detect and recognize plant diseases and the absence these diseases. The trained model has achieved an accuracy rate of 96.5% and the system was able to register up to 100% accuracy in detecting and recognizing the plant variety and the type of diseases the plant was infected.

2.2 Deep Learning and CNN based architectures

Traditional methods involve the use of semantic features as the classification method. Deep learning is a promising approach for fine-grained disease severity classification for smart agriculture, as it avoids the labor-intensive feature engineering and segmentation-based threshold. L. Ale et al [12] proposed a Densely Connected Convolutional Networks (DenseNet) based transfer learning method to detect the plant diseases, which expects to run on edge servers with augmented computing resources.They proposed a lightweight Deep Neural Networks (DNN) approach that can run on Internet of Things (IoT) devices with constrained resources. To reduce the size and computation cost of the model, they further simplified the DNN model and reduced the size of input sizes. The proposed models are trained with different image sizes to find the appropriate size of the input images. They adopted Adam as the optimization functions for both transfer learning model and light model.Table 1 shows the accuracies achieved by this technique for various input techniques.

Jasim et al. [19] presents a system that is used to classify and detect plant leaf diseases using deep learning techniques. They have taken specific types of plants; including tomatoes, pepper, and potatoes. They have used the convolutional neural network

Table 2.1: Accuracies achieved for various input sizes [12]

Input Sizes	Train Accuracy	Val Accuracy	Train Time
32x32	71.98%	71.49%	2:08:12
64x64	81.12%	74.47%	2:15:54
128x128	86.01%	85.26%	2:45:35
256x256	87.96%	87.92%	5:26:15
512x512	90.00%	89.70	18:06:16

(CNN), through which plant leaf diseases are classified, 15 classes were classified, including 12 classes for diseases of different plants that were detected, such as bacteria, fungi, etc., and 3 classes for healthy leaves. They have obtained an accuracy of 98.29% for training, and 98.029% for testing..

B. Liu et al. [13] has proposed a novel deep convolutional neural network model to accurately identify apple leaf diseases, which can automatically discover the discriminative features of leaf diseases and enable an end-to-end learning pipeline with high accuracy. Table 2.2 shows the accuracies achieved for the different models used in the report [13]. First, image processing methods produced a total of 13,689 images of diseased images such as PCA jittering, light disturbance, and direction disruption. Additionally, a unique design by deleting some of the partial information from the AlexNet model, a deep convolutional neural network was created introducing pooling layers, the Google Network Inception structure, and fully linked layers. Using the suggested network model and the NAG algorithm to optimise network settings helped accurately in classifying the illnesses of apple leaves.

Table 2.2: Accuracies achieved for various models [13]

Method	Accuracy(%)
SVM	68.73%
BP	54.63%
AlexNet	91.19%
GoogleNet	95.69%
ResNet-20	92.76%
VGGNet-16	96.32%

Fan et al. [20] added a batch standardisation layer to the convolutional layer of the Faster R- CNN model, introduced a central cost function to construct a mixed cost function, and used a stochastic gradient descent algorithm to optimize the training model. They used 9 kinds of corn leaf diseases with complex backgrounds in the field as the research object. Under the same experimental environment, the improved method had

an average accuracy increase of 8.86%, and a single image detection time was reduced by 0.139s; compared with the SSD algorithm, the average accuracy was 4.25% higher, and a single image detection time was reduced by 0.018 s.

Wang et al. [2] in order to solve the problems of a long time of training, poor segmentation effect, and susceptibility to illumination and background during the image segmentation of cucumber leaf lesions in traditional convolutional neural networks, they proposed a method based on the full convolution neural network (in which the activation function of rectified linear units (RELU) was replaced by the exponential linear unit (ELU), and the batch normalization function was used to stabilize the model training process, and the softmax of the original CNN was replaced with support vector machine (SVM)). The average pixel segmentation accuracy was 80.46% and the average cross-combination ratio was 70.43% on the 6 kinds of cucumber leaf disease dataset.

This research developed a novel identification approach of cucumber leaf diseases based on small sample size and deep convolutional neural network. The lesion images were acquired by one two-stage segmentation method that offered strong discrimination ability to extract disease spots from cucumber leaves with little human intervention. The high-quality training samples were generated under the operation of rotation, translation, and AR-GAN. With the improvement in convolutional layers, the proposed DICNN, shown in figure 2.1, it exhibited powerful feature extraction and fast convergence ability. Experimental results demonstrated that the proposed approach could effectively identify cucumber leaf diseases. The research explored a feasible way for field agricultural IoTs to timely implement the identification of plant leaf diseases, which is of great practical significance. The future work will focus on identifying more types of cucumber leaf diseases, as well as extending our approaches into the disease recognition of more plants.

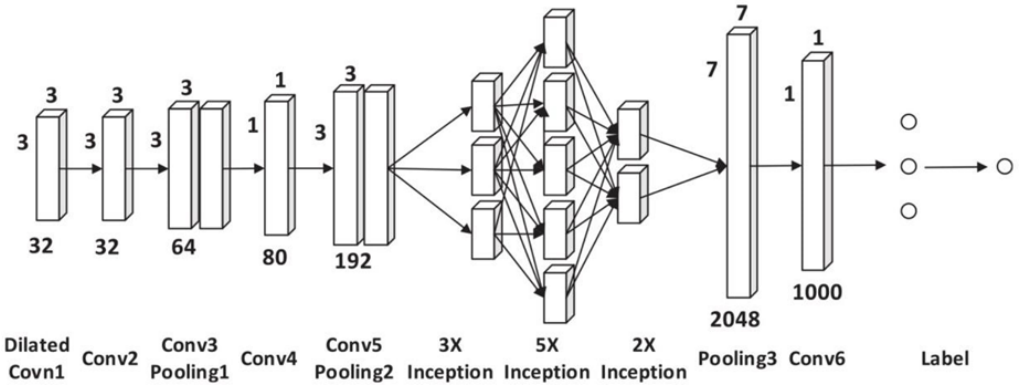

Figure 2.1: The Structure of DICNN [2]

Agarwal et al. [3] developed a CNN model with 3 convolutional layers, 3 maximum

pooling layers, and 2 fully connected layers, and each layer had a different number of filters to detect 9 types of tomato leaf diseases. The experimental results showed that the average accuracy of the proposed model on the test set reached 91.2%, and its performance was much better than VGG16, MobileNet, and Inception.

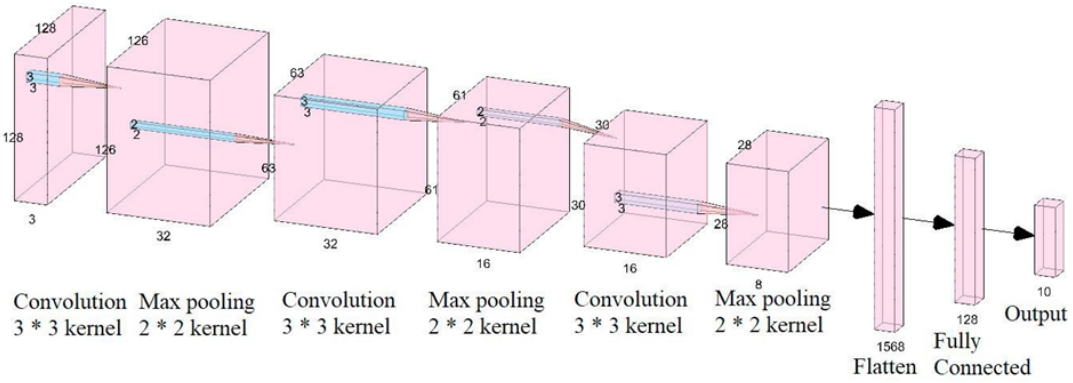

Figure 2.2: Pictorial representation of the proposed convolution network. [3]

From above figure 2.2, a CNN based model to detect the disease in tomato crop. In the proposed CNN based architecture there are 3 convolution and max pooling layers with varying number of filters in each layer. For the experiment purpose, we have taken the tomato leaf data from PlantVillage dataset. In the dataset there are 9 disease classes and class which has healthy images. As the images inside class is not balanced, data augmentation techniques have been applied to balance the images inside the class. Experimentally, it is observed the testing accuracy of the model is ranging from 76% to 100% for the classes. Moreover, the average testing accuracy of the model is 91.2%. The storage space needed by proposed model is of the order of 1.5 MB.

Fuentes et al. [21] used Faster R-CNN, R-FCN, and SSD architectures to locate lesion areas of 9 kinds of tomato leaf diseases and insect pests, and classify them according to the bounding box. And explored the influence of different CNN architectures on the detector. The results showed that ResNet50 as the feature extractor achieved a mean average accuracy (mAP) of 85.98 and the detection time was about 160 ms per image.

Jiang et al. [22] proposed a novel method that is the SSD with inception module and rainbow concatenation (INAR-SSD). And the VGG16 feature extractor used in the INAR-SSD network was a modification by replacing two convolution layers (Conv4_1 and Conv4_2) with inception modules, fully connected layers of VGG16 were also replaced with 1 1 convolutions. On a dataset of 5 kinds of apple leaf diseases, the proposed INAR-SSD network achieved the highest mAP of 78.8% compared with the Faster R-CNN (73.78%) and SSD (75.82%). Meanwhile, the detection speed of the model was 23.13 FPS.

Li et al. [23] took 5 kinds of bitter gourd leaf diseases taken in the eld as the research object, modified the Faster R-CNN by increasing the size of the regional suggestion frame and integrating the feature pyramid network (FPN) based on ResNet50. The research results showed that after integrating the feature pyramid network, the average accuracy of the obtained model was 86.39%, higher than the original model (7.54%), and the accuracy of gray spot detection was improved by 16.56%. The detection time of each image is 0.322s, which can guarantee real-time detection.

A region proposal network is combined with a CNN-based classifier in Faster R-CNN models. During the classification phase, only the recommended areas, instead of moving window, are classified with Faster R-CNN in all feature maps. Moving windows are applied to the last convolution layer of CNN. Each moving window produces k binding boxes. All these k boxes with different sizes and different width-height ratios are centered in the current moving window [4]. We can see from figure 2.3 there is an increase in the size of the input layer from 32x32 pixels to 600x600 pixels and the development of an automatic detection and recognition system for leaf spot disease. Ozguven and Adem developed Faster R-CNN and achieved an accuracy of 95.48% compared to 92.89% achieved by Faster R-CNN.

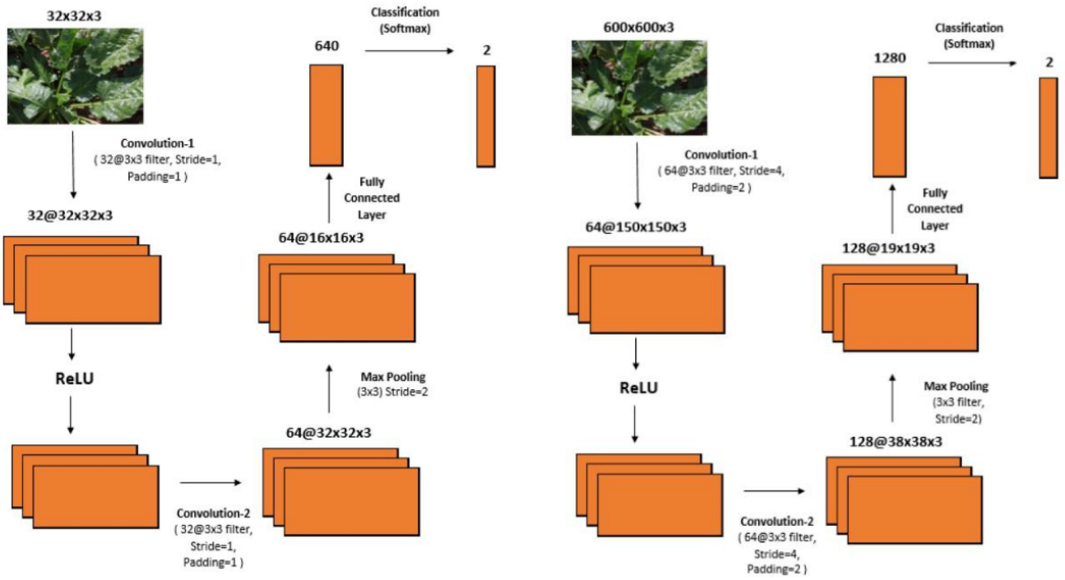

Figure 2.3: Faster R-CNN model with different window sizes for classification [4]

Aiming at the problem that the classification accuracy of the classification model for the severity of crop diseases and insect pests is not high enough, Yu et al. [24] proposed an improved ResNet50 model (CDCNNv2) combined with deep transfer learning and developed a classification system for the severity of crop diseases and insect pests. In addition to real time and fully automatic detection of crop pests and diseases, the system also implements a series of supporting functions such as prevention and control

recommendations and drug recommendations.

Table 2.3: Summary of recent Deep Learning research works

Reference No	**Year**	**Algorithm**	**Object**	**Dataset**	**Accuracy**
[12]	2019	DenseNet	Plant	Plant Village	89.70%
[19]	2020	CNN	Tomato, pepper and potato	Plant Village	98.03%
[13]	2017	AlexNet, NAG	Apple	Self acquired in field	97.62%
[20]	2020	R-CNN	Corn	China Science Data Network and Self acquired in field	96.04%
[2]	2021	DICNN	Cucumber	Self acquired in field	80.46%
[3]	2020	Improved CNN	Tomato	Plant Village	91.20%
[21]	2017	R-CNN, R-FCN, SSD	Tomato	Self acquired in field	85.98%
[22]	2019	INAR-SSD	Apple	Plant Village and Self acquired in field	78.80%
[23]	2020	Faster R-CNN	Bitter gourd	Self acquired in field	86.38%
[4]	2019	Faster R-CNN	Sugar beet	Sugar beet leaf images dataset	95.48%
[24]	2020	CDCNNv2. ResNet 50	Plant	AI Challenger 2018	99.35%

Overall, Table 2.3 provides a summary of recent research works in the field of plant disease detection using deep learning techniques, including the dataset used, the deep learning model applied, and the achieved accuracy.

Chapter 3
Theoretical Background

3.1 Machine Learning

What are the fundamental statistical computational-information-theoretic laws that govern all learning systems, including computers, humans, and organisations? How can one construct computer systems that automatically improve through experience? Machine learning addresses this question. It is one of today's most rapidly growing technical fields, lying at the intersection of computer science and statistics, and at the core of artificial intelligence and data science. Recent progress in machine learning has been driven both by the development of new learning algorithms and theory and by the ongoing explosion in the availability of online data and low-cost computation. The adoption of data-intensive machine-learning methods can be found throughout science, technology and commerce, leading to more evidence-based decision-making across many walks of life, including health care, manufacturing, education, financial modelling, policing, and marketing. Machine-learning methods can be roughly divided into three major cat-

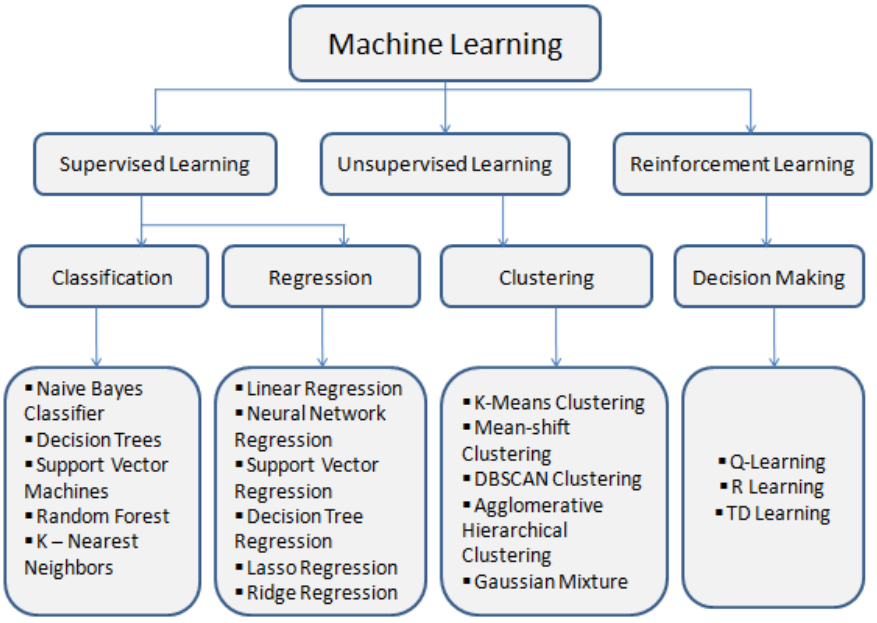

Figure 3.1: An overview of machine learning algorithms and their applications [5]

egories: supervised, unsupervised, and reinforcement learning. Fig.3.1 illustrates the family tree of machine learning with applications in wireless networks. Other learning

schemes, such as semi supervised, online, and transfer learning, can be viewed as variants of these three basic types. In general, machine learning involves two stages, i.e., training and testing. In the training stage, a model is learned based on training data, whereas in the testing stage, the trained model is applied to produce the prediction.

3.2 Deep Learning

Deep Learning is a subset of machine learning. Deep learning mainly consists of Neural Network, which is a machine learning (ML) technique that is inspired by and resembles the human nervous system and the structure of the brain. It consists of processing units organized in input, hidden and output layers. The nodes or units in each layer are connected to nodes in adjacent layers. Each connection has a weight value. The inputs are multiplied by the respective weights and summed at each unit. The sum then undergoes a transformation based on the activation function, which is in most cases is a sigmoid function, tan hyperbolic or rectified linear unit (ReLU). These functions are used because they have a mathematically favorable derivative, making it easier to compute partial derivatives of the error delta with respect to individual weights. Sigmoid and tanh functions also squash the input into a narrow output range or option, i.e., 0/1 and -1/+1 respectively. They implement saturated nonlinearity as the outputs plateaus or saturates before/after respective thresholds. ReLu on the other hand exhibits both saturating and non-saturating behaviors with f (x) = max(0, x). The output of the function is then fed as input to the subsequent unit in the next layer. The result of the final output layer is used as the solution for the problem [6].

Neural Networks can be used in a variety of problems including pattern recognition, classification, clustering, dimensionality reduction, computer vision, natural language processing (NLP), regression, predictive analysis, etc.

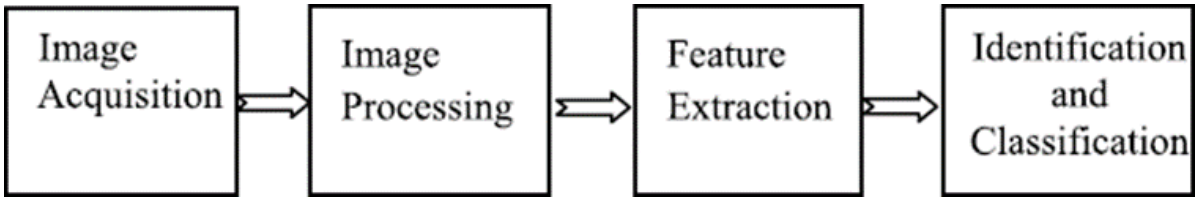

Figure 3.2: Traditional image recognition processing [6]

Figure 3.2 shows how a deep neural network can learn hierarchical levels of representations from a low-level input vector and successfully identify the higher-level object. The implementation of neural networks consists of the following steps:
1. Acquire training and testing data set

2. Train the network
3. Make prediction with test data

3.2.1 Background

In 1957, Frank Rosenblatt created the perceptron, the first prototype of what we now know as a neural network. It had two layers of processing units that could recognize simple patterns.A breakthrough in DNN occurred with the advent of backpropagation learning algorithm. It was proposed in the 1970s but it wasn't until mid-1980s that it was fully understood and applied to neural networks. DNN is a type of neural network modeled as a multilayer perceptron (MLP) that is trained with algorithms to learn representations from data sets without any manual design of feature extractors. As the name Deep Learning suggests, it consists of higher or deeper number of processing layers, which contrasts with shallow learning model with fewer layers of units. The shift from shallow to deep learning has allowed for more complex and non-linear functions to be mapped, as they cannot be efficiently mapped with shallow architectures [6].

3.2.2 DNN Architectures

Deep neural network consists of several layers of nodes. Different architectures have been developed to solve problems in different domains or use-cases. E.g., CNN is used most of the time in computer vision and image recognition, and RNN is commonly used in time series problems/forecasting. On the other hand, there is no clear winner for general problems like classification as the choice of architecture could depend on multiple factors. Nonetheless evaluated 179 classifiers and concluded that parallel random forest, which is essentially parallel implementation of variation of decision tree, performed the best. Below are four of the most common architectures of deep neural networks.
1. Convolution Neural Network
2. Autoencoder
3. Restricted Boltzmann Machine (RBM)
4. Long Short-Term Memory (LSTM)
Out of these four we mainly explore Convolutional Neural Network for Classification of Plant Diseases [6].

3.2.3 Convolutional Neural Network

Convolutional Neural Network has had ground breaking results over the past decade in a variety of fields related to pattern recognition; from image processing to voice

recognition. The most beneficial aspect of CNNs is reducing the number of parameters in ANN. Fig. 3.3 shows the feature extraction process by a CNN network through number of layers and followed by classification. This achievement has prompted both researchers and developers to approach larger models in order to solve complex tasks, which was not possible with classic ANNs [7].

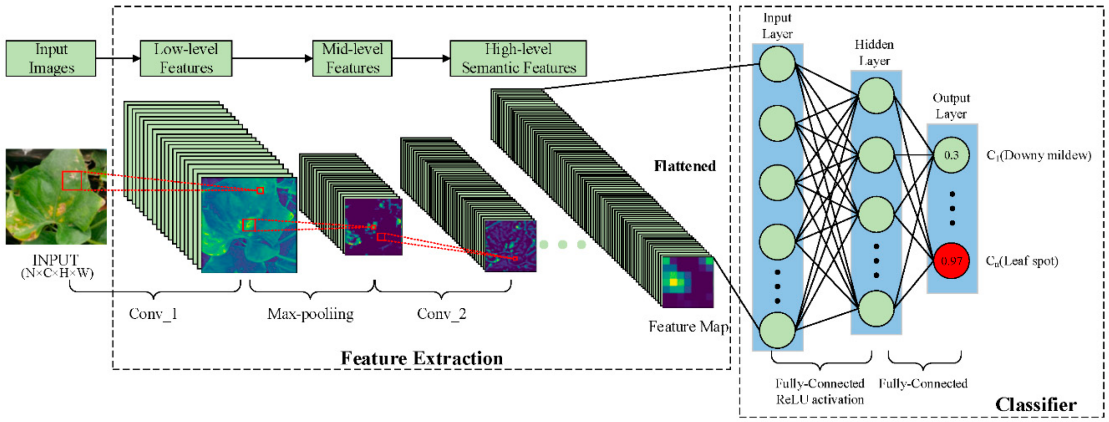

Figure 3.3: A typical diagram of Convolutional Neural Network [7]

The analysis of basic operations performed in a typical CNN is discussed below:

A. Convolution

Convolution is a mathematical operation that allows the merging of two sets of information. In the case of CNN, convolution is applied to the input data to filter the information and produce a feature map. Fig. 3.4 shows the process of convolution performed between an image and a filter. It shows the pixel level multiplications involved in a convolution process.

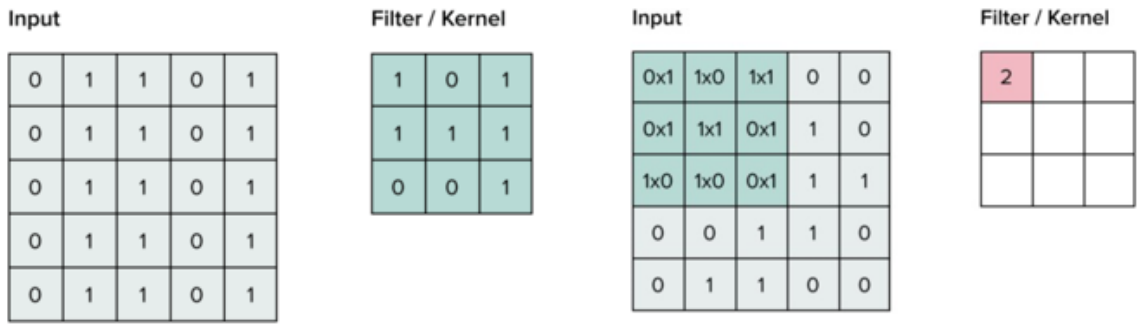

Figure 3.4: Performing Convolution [8]

This filter is also called a kernel, or feature detector, and its dimensions can be, for example, 3x3. To perform convolution, the kernel goes over the input image, doing matrix multiplication element after element. The result for each receptive field (the area where convolution takes place) is written down in the feature map [8].

B. Stride

Let's assume that the network receives raw pixels as input. Therefore, to connect the input layer to only one neuron (e.g. in the hidden layer in the Multi- Layer perceptron), as an example, there should be 32×32×3 weight connections for the CIFAR-10 dataset. In fact, CNN has more options which provide a lot of opportunities to even decrease the parameters more and more, and at the same time reduce some of the side effects. One of these options is stride. In the above mentioned example, it is simply assumed that the next layer's node has lots of overlaps with their neighbors by looking at the regions. We can manipulate the overlap by controlling the stride.

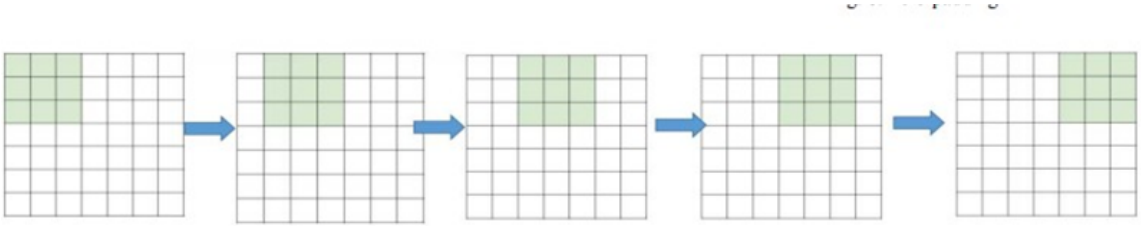

Figure 3.5: Stride 1, the filter window moves only one time for each connection [8]

Fig. 3.5, shows a given 7×7 image. If we move the filter one node every time, we can have a 5x5 output only. Note that the output of the three left matrices, have an overlap (and three middle ones together and three right ones also). However, if we move and make every stride 2, then the output will be 3x3. Put simply, not only overlap, but also the size of the output will be reduced.

$$O = 1 + (N - F)/S \tag{3.1}$$

Above equation 3.1, formalize this, given the image N×N dimension and the filter size of the F×F, the output size O as shown in Fig. 3.5 Where N is the input size, F is the filter size, and S is the stride size [8].

C. Padding

One of the drawbacks of the convolution step is the loss of information that might exist on the border of the image. Because they are only captured when the filter slides, they never have the chance to be seen. A very simple, yet efficient method to resolve

the issue, is to use zero-padding. The other benefit of zero padding is to manage the output size. For example, in Fig. 3.6, with N=7 and F=3 and stride 1, the output will be 5×5 (which shrinks from a 7×7 input). However, by adding one zero-padding, the output will be 7×7, which is exactly the same as the original input (The actual N now becomes 9, use the equation 3.1. The modified formula including zero-padding is

$$O = 1 + (N + 2P - F)/S \tag{3.2}$$

Where P is the number of the layers of the zero-padding (e.g. P=1 in Fig. 3.6), This padding idea helps us to prevent network output size from shrinking with depth. Therefore, it is possible to have any number of deep convolutional networks [8].

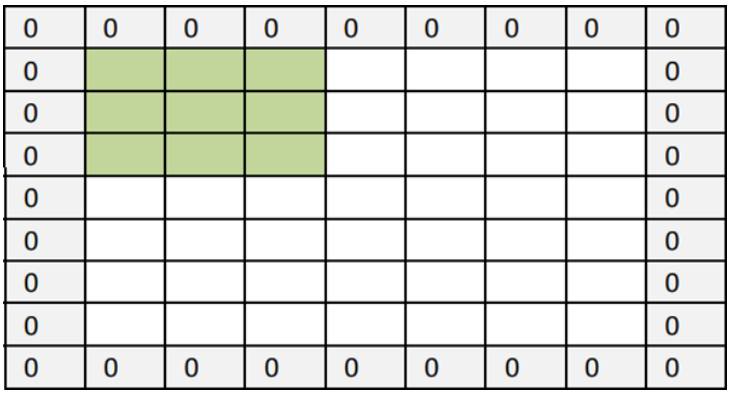

Figure 3.6: Zero Padding [8]

D. Pooling

Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network. The pooling layer summarises the features present in a region of the feature map generated by a convolution layer. So, further operations are performed on summarised features instead of precisely positioned features generated by the convolution layer. This makes the model more robust to variations in the position of the features in the input image [8].

Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

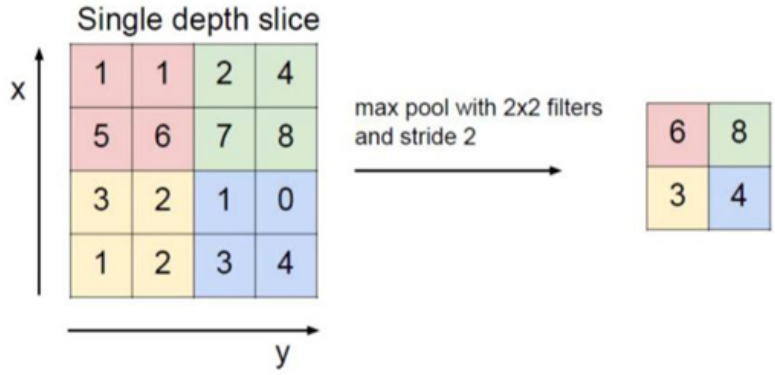

Figure 3.7: Max pooling feature [8]

Average pooling computes the average of the elements present in the region of feature map covered by the filter. Thus, while max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in a patch. Fig. 3.7 shows max pooling operation done on an image with stride 2 and filter size as 2x2.

3.3 Image Processing

3.3.1 Introduction

Computer vision algorithms can be said to be a mathematical model for processing images. It is more common in the field of artificial intelligence and has strong technical advantages. In essence, it obtains information from images or data. For humans, facing different images can be directly analysed according to their own understanding, but using a computer for image analysis will produce more interpretation methods, which are more complicated overall. The image is interpreted through a physical probability model, and finally a computer vision algorithm is formed. In the application process of this method, it can flexibly identify the image accurately by its own advantage and draw a three-dimensional model and predictive simulation based on the image to provide people with convenient services

Image Segmentation for object detection is the most vital part of computer vision where the computer has to identify objects differently from the background whether it is a face or hand or a man or simply static objects. It is the process of dividing an image into different regions based on the characteristics of pixels to identify objects or boundaries to simplify an image and more efficiently analyse it. Visual similarity can be based on brightness, color, position, depth, motion, texture and material. If the contrast difference of the background with the foreground is high then the detection

is simple. But if the background is chaotic and there is a little difference between the background and the object then it becomes difficult for the system to identify the edges from the background. The major issues in object detection are the shape variance, lighting variance and objects' pose variances.

The Gestalt psychologists identified a series of factors, which they felt predisposed a set of elements to be grouped. These factors are important because it is quite clear that the human vision system uses them in some way. Furthermore, it is reasonable to expect that they represent a set of preferences about when tokens belong together that lead to a useful intermediate representation [5].

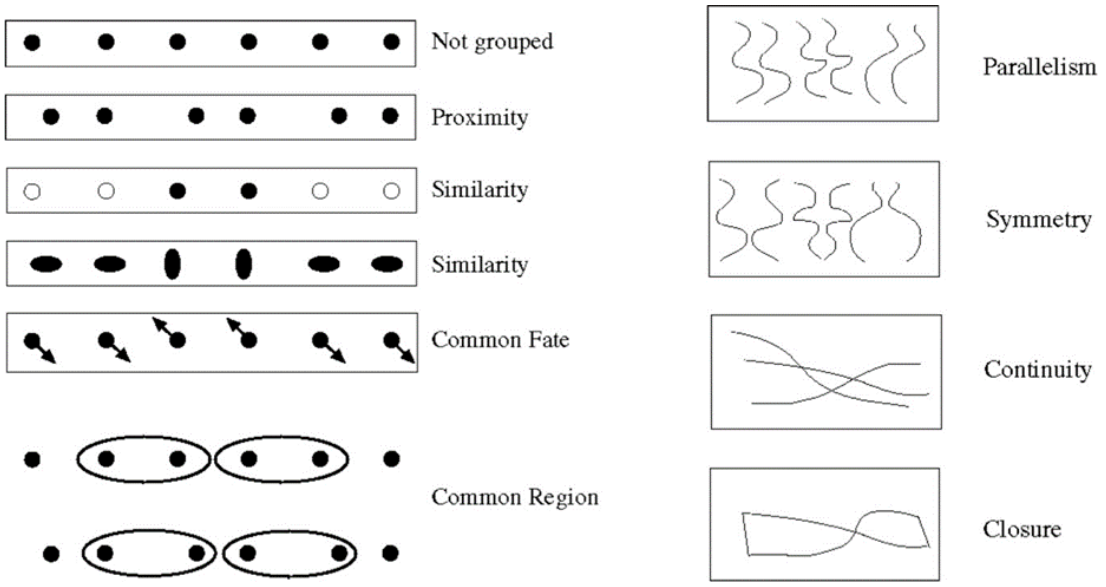

Figure 3.8: Examples of Gestalt factors that make humans group things together [5]

As shown in Fig. 3.8 there are several key principles or factors that are central to Gestalt psychology, including:

Figure-ground: This principle refers to the way our minds distinguish between the main object or figure in a scene and the background or ground against which it appears. For example, if you look at a tree in a park, the tree is the figure and the grass and other elements in the scene make up the ground. Our minds naturally separate the two and prioritize the figure as the most important element.

Proximity: This principle refers to the way we group objects together based on their proximity or closeness to one another. Objects that are close together are often perceived as belonging together or forming a group. For example, if you see a group of people standing close together, you might assume they are a group.

Similarity: This principle refers to the way we group objects together based on their similarity or likeness to one another. Objects that are similar in size, shape, color, texture, or other characteristics are often perceived as belonging together or forming a group. For example, if you see a group of balloons that are all the same color, you might assume they belong together.

Closure: This principle refers to the way our minds fill in missing information or gaps in a scene to create a complete picture. For example, if you see a circle with a small part missing, your mind might fill in the missing piece to perceive it as a complete circle.

Continuity: This principle refers to the way we perceive a continuous flow or movement in a scene. Our minds naturally group together objects that appear to be moving in the same direction or following a similar path. For example, if you see a line of people walking in the same direction, your mind might perceive it as a procession or parade.

Symmetry: This principle refers to the way we perceive balance and harmony in a scene. Our minds are drawn to symmetry and often perceive objects that are symmetrical or evenly balanced as more pleasing or aesthetically pleasing. For example, if you see a flower with identical petals on both sides, your mind might perceive it as more beautiful than a flower with uneven or asymmetrical petals.

3.3.2 Image Segmentation by Clustering Pixels

Clustering is a process whereby a data set is replaced by clusters, which are collections of data points that belong together. It is natural to think of image segmentation as clustering; we would like to represent an image in terms of clusters of pixels that belong together. The specific criterion to be used depends on the application.

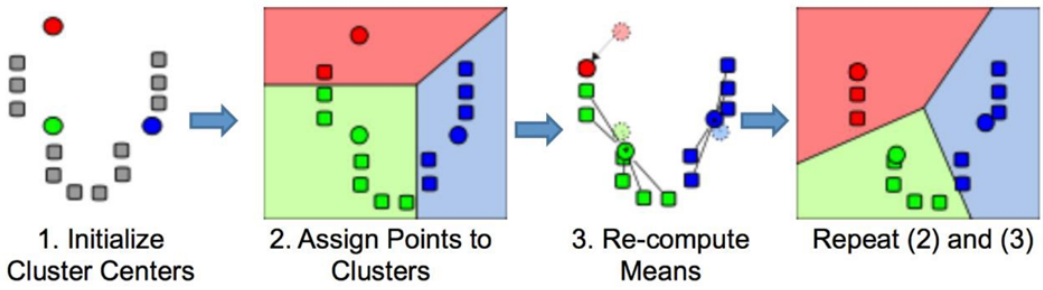

Figure 3.9: K Means algorithm flow [9]

Pixels may belong together because they have the same color, they have the same

texture, they are nearby, and so on. K -means clustering algorithm is an unsupervised algorithm and it is used to segment the interest area from the background. Fig. 3.9 shows the flow of K Means algorithm on a group of data points.

Mean shift is an unsupervised learning that assigns the data points to the clusters iteratively by shifting points towards the mean. Unlike the popular K-Means cluster algorithm, mean- shift does not require specifying the number of clusters in advance.

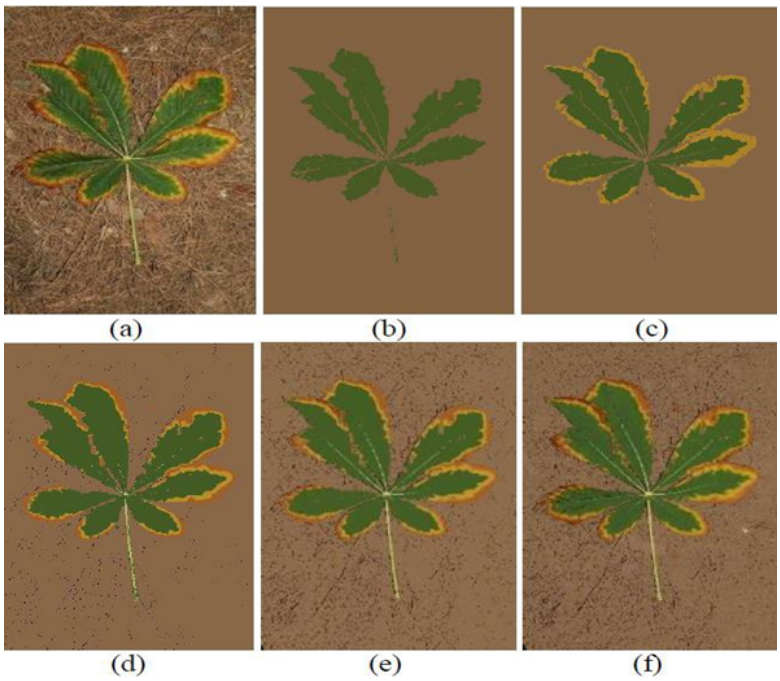

Figure 3.10: Mean Shift algorithm (a) Original image (b) h=2 (c) h=15 (d) h=10 (e) h=5 (f) h=3 [10]

This algorithm is mostly used for clustering. It is widely used in image segmentation because it's non-parametric and doesn't require any predefined shape of the clusters in the feature space. The mean shift smoothing process is shown in Fig.3.10 when the bandwidth h is set to different values. we can observe that Fig.3.10 (b) has lost the gradient area of the edge of the leaf, and Fig.3.10 (c) to Fig.3.10 (e) still retains the gradient area of the edge of the leaf.

Chapter 4
Plant Disease Detection

4.1 Dataset

The PlantVillage dataset is a publicly available dataset of plant images, created to support research in computer vision and machine learning for plant disease identification and classification. The dataset contains a total of 54,305 images of 14 different crop species, including both healthy and diseased plants. The images are 256x256 healthy and unhealthy leaf images divided into 38 categories by species and disease named as *species_disease* or *species_healthy*.
The following are the classes and the number of images included in each class in the PlantVillage dataset:
1. Apple - 4440
2. Blueberry - 462
3. Cherry - 3370
4. Corn - 12333
5. Grape - 3117
6. Orange - 534
7. Peach - 1561
8. Pepper - 2606
9. Potato - 1922
10. Raspberry - 1687
11. Soybean - 5093
12. Squash - 1927
13. Strawberry - 1901
14. Tomato - 18185

Each class contains both healthy and diseased plant images, with varying degrees of severity and symptoms. The images were collected from various sources, including field surveys and laboratory experiments, and were reviewed by experts to ensure accuracy of labeling and classification. The dataset is available for download from the PlantVillage website and can be used for training and evaluating machine learning models for plant disease identification and classification.

The diseases and symptoms that are present in the dataset vary depending on the plant class. Here are some examples of the diseases and symptoms present in the PlantVillage dataset:

1. Apple: Apple Scab, Cedar Apple Rust, Powdery Mildew
2. Blueberry: Mummy Berry, Phomopsis Twig Blight, Anthracnose Fruit Rot
3. Cherry: Cherry Leaf Spot, Powdery Mildew, Cherry Fruit Fly
4. Corn: Common Rust, Northern Leaf Blight, Gray Leaf Spot
5. Grape: Black Rot, Downy Mildew, Powdery Mildew
6. Orange: Citrus Greening, Alternaria Brown Spot, Phytophthora Foot Rot
7. Peach: Bacterial Spot, Peach Leaf Curl, Brown Rot
8. Pepper: Bacterial Spot, Powdery Mildew, Tomato Spotted Wilt Virus
9. Potato: Early Blight, Late Blight, Blackleg
10. Raspberry: Raspberry Leaf Curl, Anthracnose, Spur Blight
11. Soybean: Soybean Rust, Septoria Brown Spot, Brown Stem Rot
12. Squash: Powdery Mildew, Zucchini Yellow Mosaic Virus, Squash Bug
13. Strawberry: Anthracnose Crown Rot, Botrytis Fruit Rot, Powdery Mildew
14. Tomato: Late Blight, Early Blight, Tomato Yellow Leaf Curl Virus

The dataset includes multiple images of each disease and symptom, with varying degrees of severity and progression. The images were labeled and classified by experts to ensure accuracy and consistency.

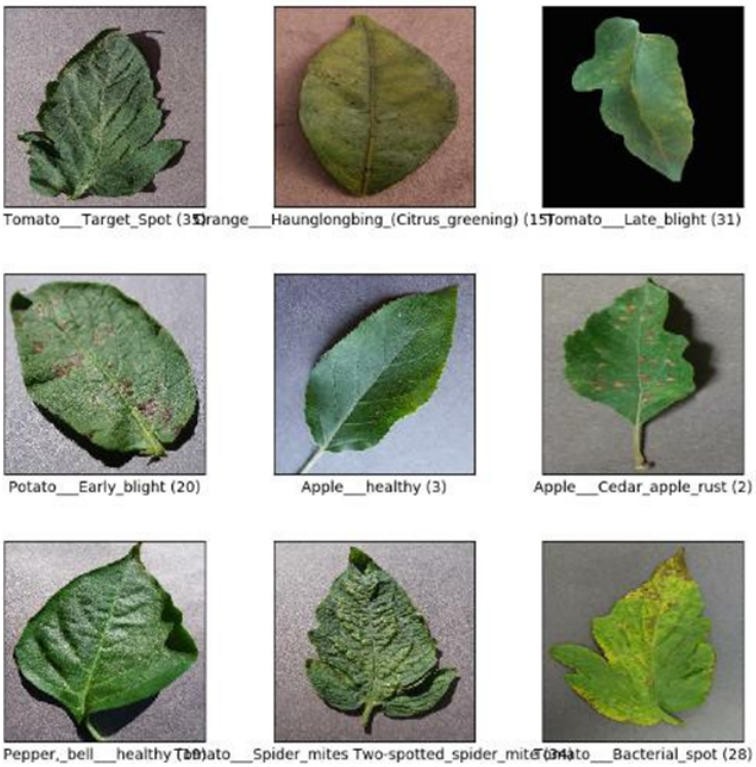

Figure 4.1: Example images from PlantVillage Dataset

4.2 Using Machine Learning

Below Figure 4.2 shows the flowchart of the steps followed in implementing the first model where we compared the accuracy of various standard machine learning algorithms.

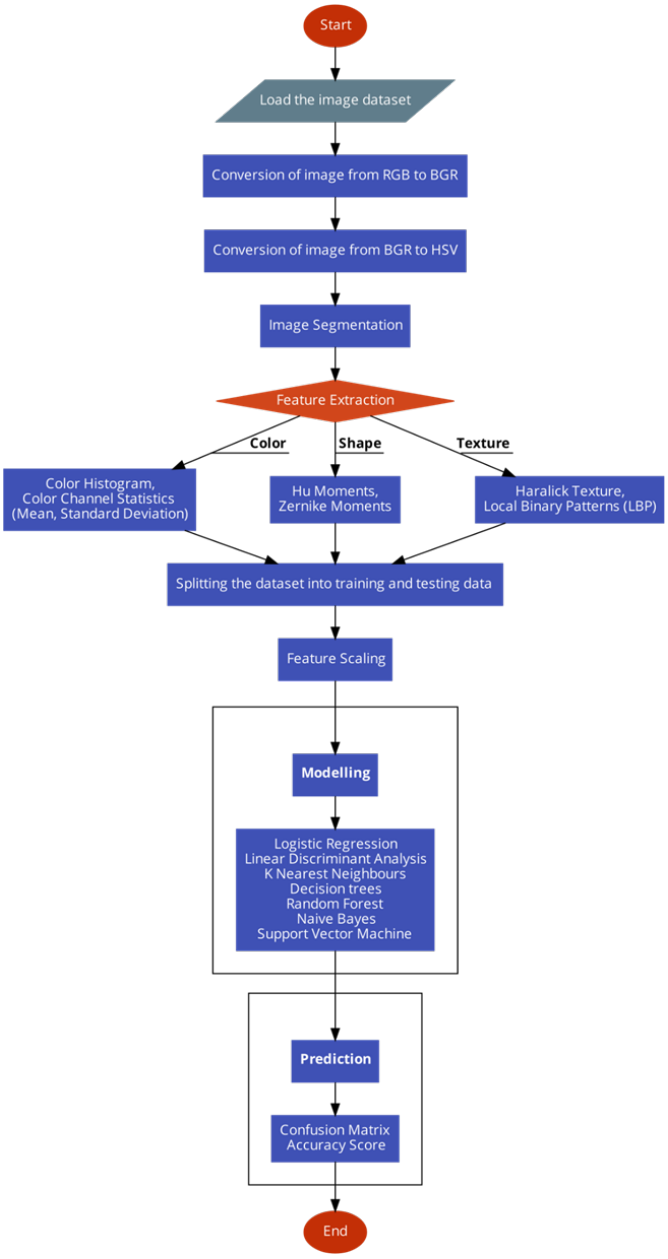

Figure 4.2: Flowchart showing the process of disease detection

Conversion of image from RGB to BGR. Since OpenCV accepts images in BGR colouring format so it needs to be converted to the original format that is BGR format. Con-

version of image from BGR to HSV is essential as HSV separates luma, or the image intensity, from chroma or the color information is done.Fig. 4.3 and Fig. 4.4 shows the conversions respectively.

Figure 4.3: An image of leaf from PlantVillage Dataset

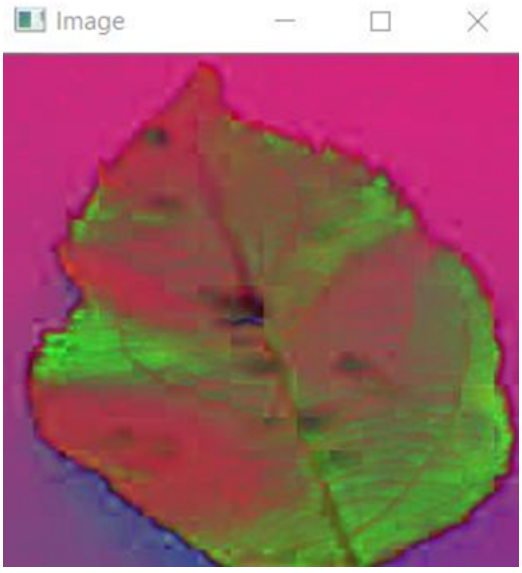

Figure 4.4: HSV converted image

Image Segmentation for extraction of colors. In order to separate the picture of leaf from the background segmentation has to be performed. The color of the leaf is extracted from the image. We have created a mask containing the range of colors a leaf

can have in RGB values and segmented the image of leaf out of the background.

Figure 4.5: Image segmented out of background

Applying Global Feature Descriptor. Global features are extracted from the image using three feature descriptors namely :
• **Color :** Color Channel Statistics (Mean, Standard Deviation) and Color Histogram
• **Shape :** Hu Moments, Zernike Moments
• **Texture :** Haralick Texture, Local Binary Patterns (LBP)

Shape features usually consist of two geometrical features aspect ratio (ratio between length and width of the leaf) and compactness or roundness which is ratio between area of the leaf and square of perimeter of the leaf. Other important shape features are rectangularity or similarity between a leaf and a rectangle, solidity or a ratio between the area of the leaf and the area of its convex hull, area of the leaf witch is the number of pixels that leaf contains.

It is important to mention that some plant species have leafs with irregular shapes. Color features are used by some methods but due to the fact than many plant species have similar leaf colors this features is not of crucial importance. However, information from histogram can be used for classification of leafs. Texture features are important as well. They can be used to describe vain structure of the leaf or venation. This feature proved to be very important, right after the shape, for recognition of leafs. Fig. 4.5 shows a image of leaf from the dataset and Fig. 4.6 shows its color histogram.

After extracting the feature of images the features are stacked together using numpy

function “np.stack”. The Dataset is splitted into training and testing set with the ratio of 80/20 respectively. Feature Scaling Feature Scaling is performed during the data pre-

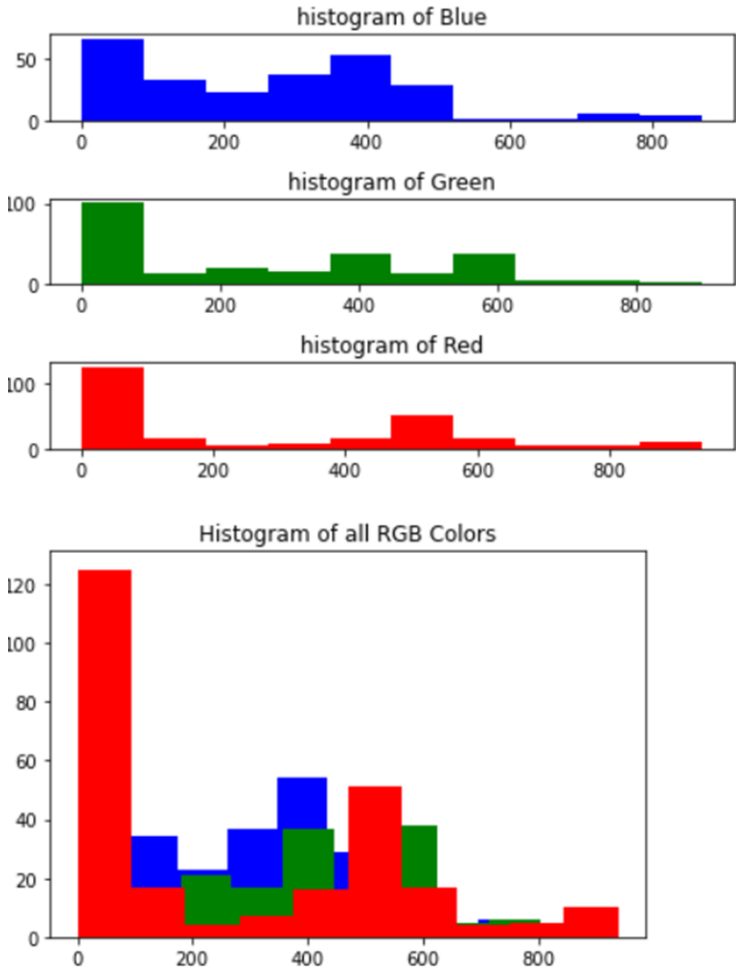

Figure 4.6: Histogram of a leaf image in all colors

processing to handle highly varying magnitudes or values or units. Here, we have used Min-Max Scaler. This scaling brings the value between 0 and 1.

After features are extracted from the images they are saved in HDF5 file. The Hierarchical Data Format version 5 (HDF5), is an open source file format that supports large, complex, heterogeneous data. Modelling: The Model is trained over 7 machine learning models named :

• Logistic Regression
• Linear Discriminant Analysis
• K Nearest Neighbours
• Decision Trees

• Random Forest
• Naïve Bayes
• Support Vector Machine

And the model is validated using a 10 k fold cross validation technique. Fig.4.7 shows the comparison of performance of machine learning algorithms. It is observed that Random Forest Classifier has high accuracy for the training data. So it has been proceeded to perform on test data.

```
LR: 0.920312 (0.025627)
LDA: 0.917188 (0.022964)
KNN: 0.925781 (0.025496)
CART: 0.907031 (0.030549)
RF: 0.961719 (0.015007)
NB: 0.866406 (0.015007)
SVM: 0.919531 (0.023189)
```

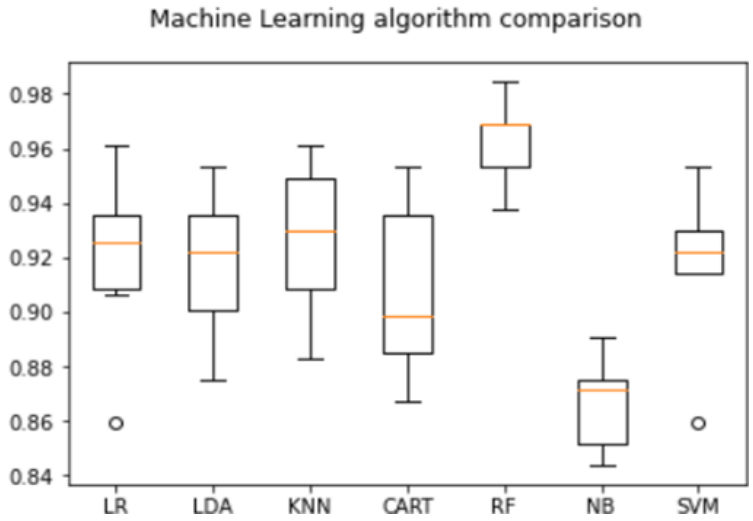

Figure 4.7: Comparison of all ML algorithms performance while cross validation

The models with best performance is them trained with whole of the dataset and score for testing set is predicted using 'Predict' function. Random Forest Classifier has better performance and testing has been performed using it. Confusion matrix has been calculated. The accuracy achieved is 93.75

4.3 Convolutional Neural Network

4.3.1 Method

This section explains the steps how to design the Web application for the Alexnet. Thus, it has been designed according to Human-centered Concept and Machine Learning

(HCML) in one or more objects in the lifecycle. Next, for implementation purpose, Jupyter is used in collaboration with the Python programming language and Integrated Development Environment (IDE), as well as several supporting libraries such as Tensor Flow, Keras, NumPy, Scikit-learn, OS and all of Python's built- in libraries. All these libraries are used to manipulate and engineer raw data for it to match the requirements for input into the main library. The steps carried out in this project are described in below Fig 4.8.

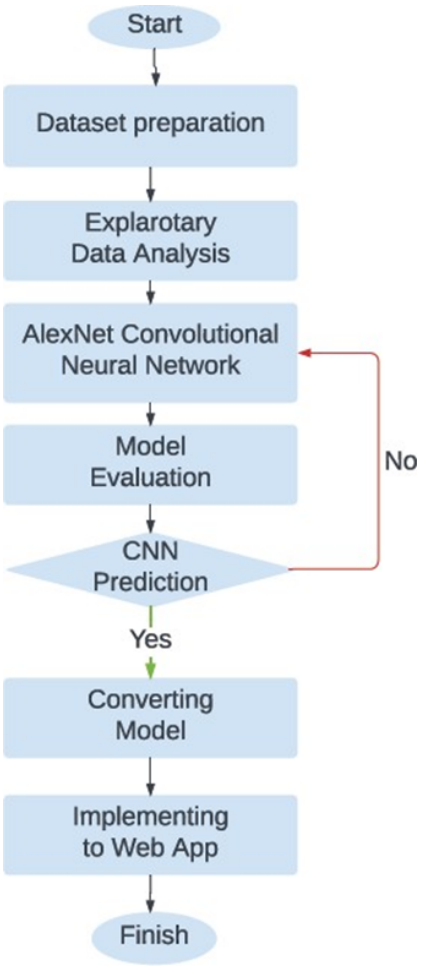

Figure 4.8: Flow chart showing the steps involved in developing this model [11]

4.3.2 Data Analysis

The dataset is obtained from Kaggle and can be accessed through the application programming interface (API). On the kaggle.com webpage, any user could easily obtain the "kaggle.json" API files. For this part of the report only Potato plant has been considered and the results shown from now are of the same.

Table 4.1: The details of potato diseases in the dataset consisting of tested dataset and trained dataset.

Type of Leaf	Training Dataset	Validation Dataset	Testing Dataset
Late Blight	800	100	100
Healthy	122	15	15
Early Blight	800	100	100

Then, the data collected during the exploration phase, EDA, are analyzed. The goal of EDA is to deliver data in such a way that the amount of data in the training and test data, as well as the relationship between variables, can be comprehended. By importing Python libraries and the exploratory data analysis function, the dataset can be divided into three categories of data: training data, validation data and testing data each of which contains 3 different forms of leaf diseases in tomato plants as shown in table 4.1.

4.3.3 AlexNet Convolutional Neural Network

Firstly a layer is created for Resizing and Normalization before we feed our images to network, the it is resized to the desired size. Moreover, to improve model performance, then normalize the image pixel value (keeping them in range 0 and 1 by dividing by 256). This should happen while training as well as inference. Hence we can add that as a layer in our Sequential model. Then further the layers of AlexNet architecture are defined.

AlexNet architecture was developed Alex krishevesky et al, which won the ILVRC (ImageNet Large Scale Visual Recognition Challenge) in 2012. AlexNet has about 650,000 neurons with a total of 60 million parameters, compared to Lenet-5, the network scale has been greatly improved. In the 2012 ImageNet image classification competition, it has achieved a huge advantage of 11% higher accuracy than the second place [14].

AlexNet is an 8-layer convolutional neural network, it is consisted of 5 convolutional layers and 3 full connection layers, of which the maximum pooling operation is performed after three convolutional layers as shown in Fig. 4.9. Different from previous neural networks, AlexNet uses ReLU as the activation function to instead of the traditional sigmoid and tanh functions.

$$ReLU(x) = max(0, x) \tag{4.1}$$

ReLU is a non-saturated activation function, which not only effectively improves

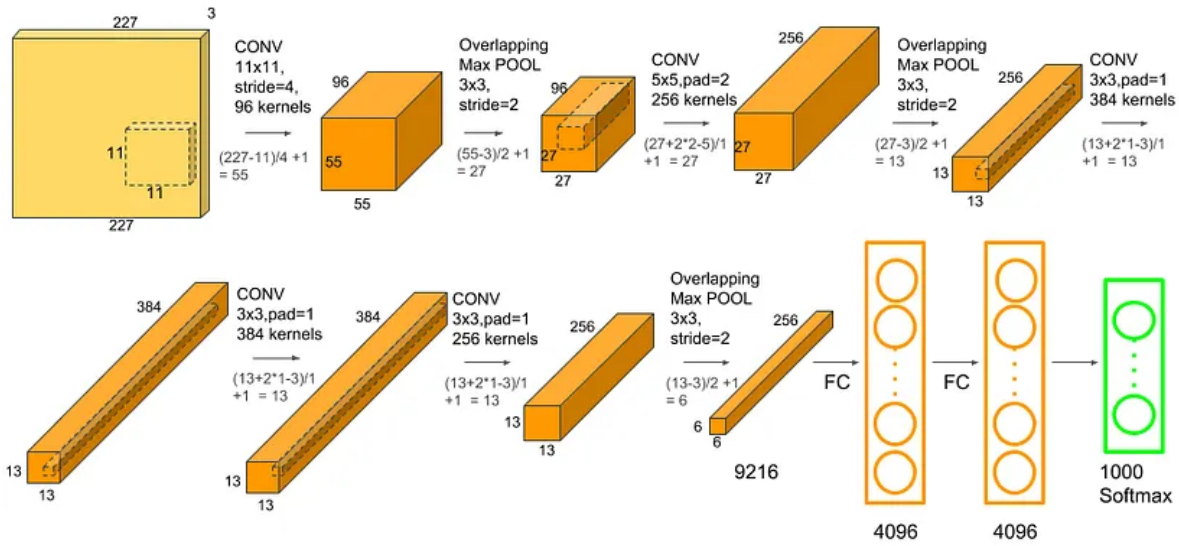

Figure 4.9: AlexNet Architecture [11]

the training speed of the model, but also better controls the problem of gradient disappearance and gradient explosion, it is easy to train a deeper network. The form of ReLU function is shown in equation 4.1.

To achieve generalization, some units or connections with a certain probability within the network are randomly skipped (dropout). The AlexNet model executes dropout after several fully connected layers in it.

At the end of the feature extraction stage (accomplished by convolutional layers), three fully connected layers are introduced which perform classification globally. The final layer of AlexNet architecture which acts as the output layer uses the softmax activation function. The softmax function is given by:

$$S(yi) = \frac{exp(yi)}{\sum_{j=1}^{n} exp(ji)} \tag{4.2}$$

where *yi* is the element of the input vector, is the number of classes which, in our case is two–defect and defect-free [14]

Table 4.2: Layers of the AlexNet architecture [14]

ID	**Layer Type**	**Layer Parameters(f=no. of feature maps, k=kernel size, s=stride, act=activation function)**	**Size of feature Map**
0	Input layer	Input image size=(227x227) Pixels, Channels=3	227x227x3
1	Conv2D	f=96, k=(l 1x11), s=4, act=ReLU	55x55x96
2	Max Pool	f=96, k=(3X3), s=2	27x27x96
3	Batch normalization	N/A	27x27x96
4	Conv2D	f=256, k=(5 X 5), s=1, act=ReLU	27x27x96
5	Max Pool	f=256, k=(3 X 3), s=2	13x13x256
6	Batch Normalization	N/A	13x13x256
7	Conv2D	f=384, k=(3 x 3),s=I, act=ReLU	13x13x384
8	Batch normalization	N/A	13x13x384
9	Conv2D	f=384, k=(3 x 3),s=I, act=ReLU	13x13x384
10	Batch normalization	N/A	13x13x384
11	Conv2D	f=256, k=(3 X 3), s=I, act=ReLU	13x13x256
12	Max Pool	f=256, k=(3 X 3), s=2	6x6x256
13	Batch Normalization	N/A	6x6x256
14	Dropout	Rate=0.5	6x6x256
15	FC	f, k, s are N/A,act=ReLU	4096
16	Dropout	Rate=0.5	4096
17	FC	f, k, s are NIA, act=ReLU	1024
18	Dropout	Rate=0.5	1024
19	FC	f, k, s are N/A, act=softmax	3

4.3.4 Model Evaluation

The model is evaluated to determine the loss and accuracy values that are contained in the model. The model prediction outcomes are displayed in the form of accuracy, precision, recall, and F- measure in the confusion matrix generated during the training.

Here are some key analyses that can be obtained from epochs and validation curves:

Loss Analysis:

- Training Loss: The training loss indicates how well the model is learning from the training data. It represents the discrepancy between the predicted outputs and the true labels during training. Analyzing the training loss over epochs can help you determine if the model is converging or overfitting. A decreasing training loss generally indicates that the model is learning and improving.

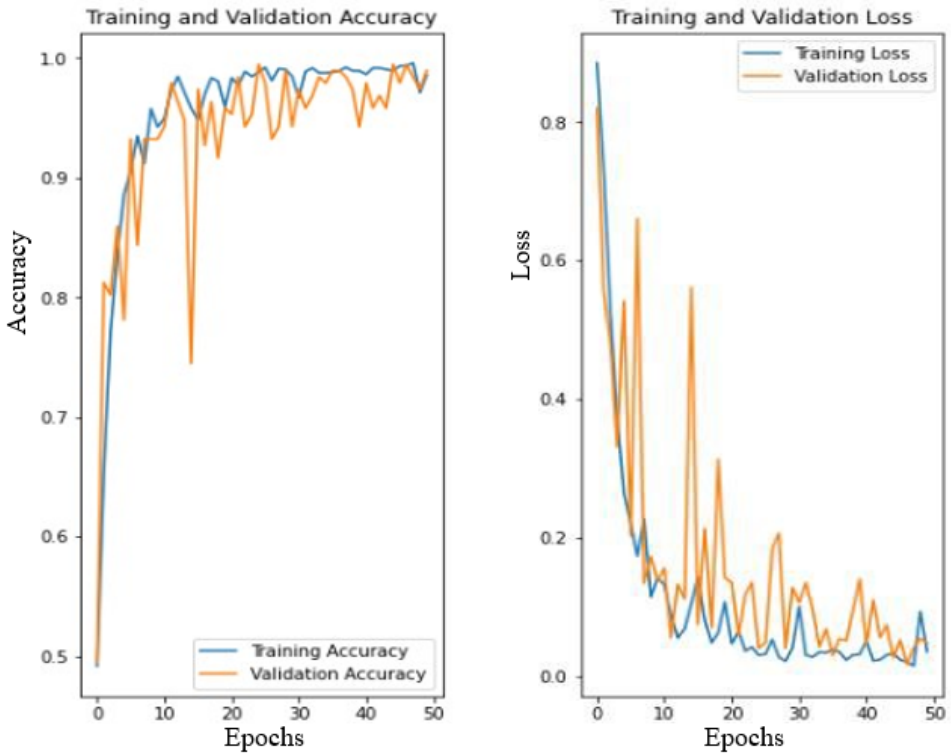

Figure 4.10: Plot of comparing accuracy and loss of Training and Validation.

- Validation Loss: The validation loss measures the performance of the model on a separate validation dataset. It provides an estimate of how well the model generalizes to unseen data. Monitoring the validation loss over epochs helps identify if the model is overfitting or underfitting. A decreasing validation loss indicates that the model is learning and generalizing well.

- Overfitting: If the training loss continues to decrease, but the validation loss starts to increase or remains stagnant, it suggests overfitting. Overfitting occurs when the model memorizes the training data instead of learning the underlying patterns, resulting in poor generalization. Regularization techniques or adjusting the model's complexity may be necessary to mitigate overfitting.

Accuracy Analysis:

- Training Accuracy: The training accuracy measures the model's performance on the training dataset. It represents the percentage of correctly predicted samples during training. Analyzing the training accuracy over epochs provides insights into the model's learning progress. An increasing training accuracy indicates that the model is improving its ability to classify the training data correctly.

- Validation Accuracy: The validation accuracy evaluates the model's performance on the validation dataset, which consists of unseen data. It indicates how well the model generalizes to new samples. Monitoring the validation accuracy over epochs helps assess the model's ability to generalize beyond the training data. An increasing validation accuracy suggests that the model is learning and generalizing well.
- Overfitting: Similar to the loss analysis, if the training accuracy keeps increasing while the validation accuracy plateaus or starts to decrease, it indicates overfitting. It means the model is becoming too specific to the training data and does not perform well on new, unseen data.

```
[INFO] Calculating model accuracy
8/8 [==============================] - 1s 15ms/step - loss: 0.0519 - accuracy: 0.9844
Test Accuracy: 98.44000000000001%
```

Figure 4.11: Accuracy of model on testing data

It is an effficient training with optimal loss. From Fig. 4.10 it is observed that as the number of epochs increased the training and validation curves coincided which justifies the efficient training done. The implementation of the model evaluation is carried out by using the evaluate () function in Keras libraries.

We have achieved an accuracy of 98.44% for potato plant classification using AlexNet, whereas the referenced paper [14] reported a slightly lower accuracy of 98.35%.

4.3.5 Detection for random test data

This step is carried out to measure the ability of the best CNN model to predict from the results of previous training. The CNN model was given input from the test data. For each incoming leaf disease image, the CNN output layer will give the probability of each class using the Softmax activation function formula. The result of CNN is an i-th neuron with the highest output probability value prediction. The final step is to convert the CNN model so that the CNN model becomes an entity that can provide predictive results from live data. Then, the CNN model is converted to Tensorflow Lite. Fig. 4.12 shows the test results obtained by predicting through the trained model.

4.3.6 Designing a Web Application

Leaf disease image recognition in tomato plants (image recognition) is implemented into a web- based application by using FastAPI as web framework. FastAPI is a Web

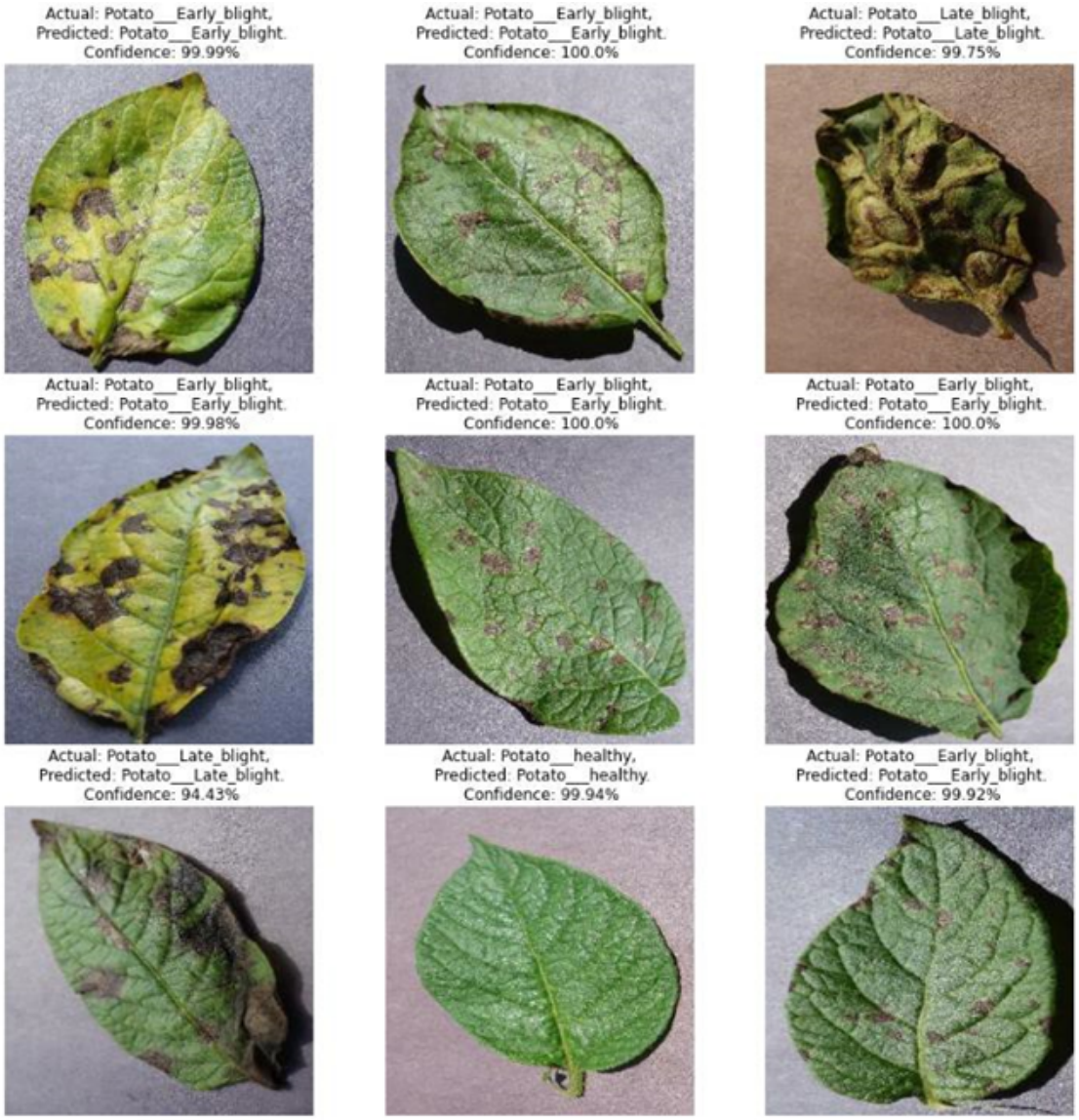

Figure 4.12: Some predicted images from the results of previous training

framework for developing RESTful APIs in Python. FastAPI is based on Pydantic and type hints to validate, serialize, and deserialize data, and automatically auto- generate OpenAPI documents. It fully supports asynchronous programming and can run with Gunicorn and ASGI servers for production such as Uvicorn and Hypercorn.

React is a JavaScript library for building user interfaces. It allows developers to create reusable UI components. It uses a virtual DOM (a lightweight in-memory representation of the actual DOM) to optimize updates to the real DOM, making it faster than other JavaScript libraries that manipulate the DOM directly. React follows a unidirectional data flow, which means that the parent component passes data down to its children through properties. This makes it easy to reason about the flow of data in a

React application, and helps to minimize the potential for unexpected behaviour.

This application is executed using ReactJS with Deep Learning library, in which there are algorithms and other libraries related to the image recognition model of leaf diseases in tomato plants.

4.3.7 Results

Below figures Fig. 4.13, Fig. 4.14 and Fig.4.15 are the results obtained after uploading Early Blight, late blight and a healthy images of Potato Leaves respectively.

Figure 4.13: Leaf classified as early blight with 100% prediction confidence

Figure 4.14: Leaf classified as healthy with 97.82% prediction confidence

The results of this study demonstrate the effectiveness of the developed convolutional neural network (CNN) model for accurately detecting early blight, late blight, and healthy leaves of potato plants. The model achieved 100% prediction confidence in correctly identifying leaves with early blight and late blight, and 97.82% prediction

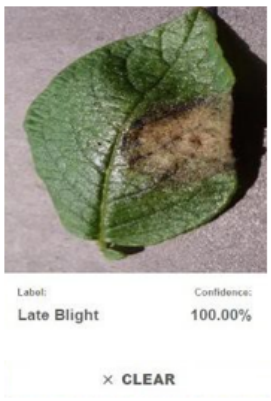

Figure 4.15: Leaf classified as late blight with 100% prediction confidence

confidence in identifying healthy leaves.

These results highlight the potential of CNN models in accurately and efficiently detecting plant diseases, which can aid in early detection and effective management of plant diseases. The use of such models can also reduce the reliance on manual detection methods, which can be time-consuming and may require specialized expertise.

Overall, the results of this study demonstrate the potential of deep learning models in the agricultural sector, particularly in the detection and management of plant diseases. Further research and development in this area can lead to the creation of more accurate and efficient tools for disease detection and management, ultimately leading to increased crop yield and food security.

Chapter 5
Model Development for PlantVillage Dataset

In this chapter a CNN architecture has been developed and it has been trained with the whole PlantVillage dataset which consists of 54,305 images of 14 different crop species, including both healthy and diseased plants. The images are 256x256 healthy and unhealthy leaf images divided into 38 categories by species and disease named as *species_disease* or *species_healthy*.

A CNN architecture is designed to extract features from the input images using a series of convolutional and pooling layers, followed by two fully connected layers to classify the input image into one of the 38 disease classes. The use of data augmentation in the architecture helps in improving the performance of the model by introducing variations in the training data, thus making the model more robust to real-world variations. The implementation of this model can help in accurate and early detection of plant diseases, which can potentially reduce crop losses and ensure food security.

5.1 Modified CNN Architecture

The architecture consists of the following layers:

- A preprocessing layer that scales the input image to a smaller size and rescales the pixel values between 0 and 1.
- A data augmentation layer that randomly applies various transformations to the input images during training, such as random rotations, zooming, and flipping.
- A 2D convolutional layer with 32 filters of size 3x3 and ReLU activation function. This layer takes the preprocessed input image as the input.
- A max-pooling layer that downsamples the output of the previous convolutional layer by taking the maximum value within a 2x2 window.
- Additional ‘Conv2D‘ and ‘MaxPooling2D‘ layers with increasing numbers of filters (64 in each) and decreasing spatial dimensions are stacked one after the other. These layers learn higher-level representations by convolving with increasingly larger receptive fields while reducing the spatial dimensions of the input feature maps. The output of the last ‘MaxPooling2D‘ layer is a 2D tensor with a spatial dimension of 4x4.

- A layer that flattens the 4D tensor into a 1D vector. This layer prepares the output of the convolutional layers for the fully connected layers.
- Two fully connected layers with 64 and n_classes neurons, respectively. These layers apply a linear transformation to the input features and then apply the ReLU and softmax activation functions, respectively. The output of the last 'Dense' layer is the predicted class probabilities.

5.2 Comparison of modified CNN and Alexnet model

The AlexNet model which has been trained with only Potato plant earlier has been again trained with the whole dataset. The performance of Alexnet model is exemplary when compared with modified CNN model. The modified CNN model is more compatible with size. So, it can be used for space optimised applications.

Table 5.1: Overall comparison of modified CNN and Alexnet model

Model	**Size**	**Accuracy**	**Parameters(trainable+non-trainable)**	**Epochs**
Alexnet	611MB	0.96	58,439,782 + 2,752 = 58,442,534	50
Modified CNN	37MB	0.87	186,022 + 0 = 186,022	50

Better accuracy and high performance can be provided by Alexnet model. Table 5.1 illustrates the comparison of both models and also give some insights into the accuracy. Fig. 5.1 and Fig 5.2 show the testing and validation accuracies of Alexnet and modified CNN architectures respectively.

AlexNet:

- Size: The AlexNet model has a size of 611MB, indicating the amount of memory required to store the model's parameters and architecture.
- Parameters: The total number of parameters in AlexNet is 58,442,534, which includes both trainable and non-trainable parameters.
- Accuracy: The AlexNet model achieves an accuracy of 0.96 as shown in Fig. 5.1, indicating its performance on a given dataset or task.

Modified CNN:

- Size: The modified CNN model has a significantly smaller size of 37MB compared to AlexNet, making it more compatible with space-optimized applications.

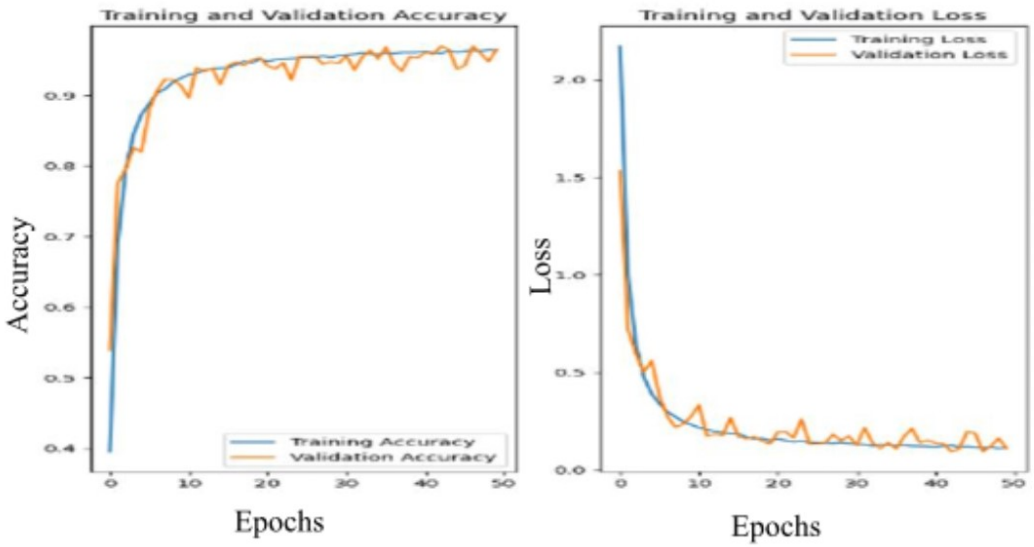

Figure 5.1: Training and validation curve of Alexnet Model.

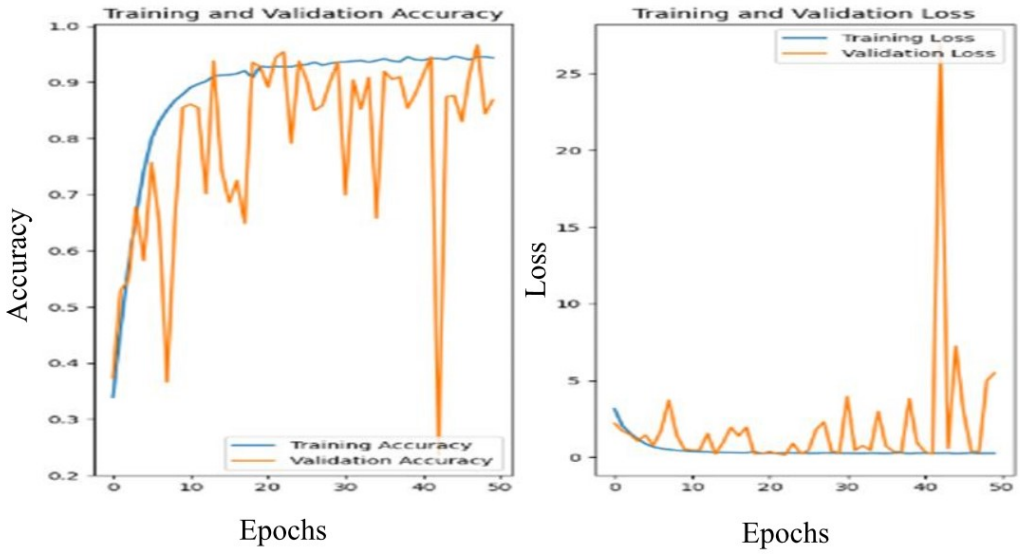

Figure 5.2: Training and validation curve of modified CNN Model.

- Parameters: The modified CNN has a total of 186,022 parameters, which includes both trainable and non-trainable parameters. It suggests that the modified CNN has a simpler architecture or fewer layers, resulting in reduced complexity.
- Accuracy: The modified CNN model achieves an accuracy of 0.87 as shown in Fig. 5.2. This suggests that, in the given comparison, the AlexNet model outperforms the modified CNN in terms of accuracy.

In summary, from the observed information indicates that the AlexNet model has a

larger size, higher number of parameters, and better accuracy compared to the modified CNN model. However, the modified CNN model has a smaller size and may be more suitable for space-optimized applications where memory constraints are a concern. The choice between the two models depends on the specific requirements and trade-offs of the application at hand.

We have successfully achieved an accuracy of 98.44% for potato plant classification using AlexNet, whereas the referenced paper [14] reported a slightly lower accuracy of 98.35%. Furthermore, when considering the entire Plant Village Dataset, we have obtained an overall accuracy of approximately 96%. These results demonstrate the effectiveness of our approach in accurately classifying potato plants and highlight the strong performance of our model on a broader range of plant species in the dataset.

5.3 Problem of Overfitting and Underfitting

Overfitting and Underfitting are the two main problems that occur in deep learning and degrade the performance of the machine learning models. The main goal of each machine learning model is to generalize well. Here generalization defines the ability of a model to provide a suitable output by adapting the given set of unknown input. It means after providing training on the dataset, it can produce reliable and accurate output. Hence, the underfitting and overfitting are the two terms that need to be checked for the performance of the model and whether the model is generalizing well or not.

- Signal: It refers to the true underlying pattern of the data that helps the machine learning model to learn from the data.
- Noise: Noise is unnecessary and irrelevant data that reduces the performance of the model.
- Bias: Bias is a prediction error that is introduced in the model due to oversimplifying the machine learning algorithms. Or it is the difference between the predicted values and the actual values.
- Variance: If the machine learning model performs well with the training dataset, but does not perform well with the test dataset, then variance occurs.

Underfitting occurs when our CNN model is not able to capture the underlying trend of the data. To avoid the overfitting in the model, the fed of training data can be stopped at an early stage, due to which the model may not learn enough from the training data. As a result, it may fail to find the best fit of the dominant trend in the data. Here 40 epochs is an underfitting case. In the case of underfitting, the model is not able to learn enough from the training data, and hence it reduces the accuracy and

Table 5.2: Performance of Alexnet for different epochs

Epochs	**Test Accuracy**	**Validation accuracy**	**Loss at fifth epoch**
40	0.76	0.80	2.71
75	0.81	0.83	2.89
50(Good fitting)	0.94	0.96	0.38

produces unreliable predictions. An underfitted model has high bias and low variance. Fig. 5.3 gives the understanding of the underfitting using of Alexnet model with 40 epochs. Table 5.2 shows the various accuracies for different epochs for the model.

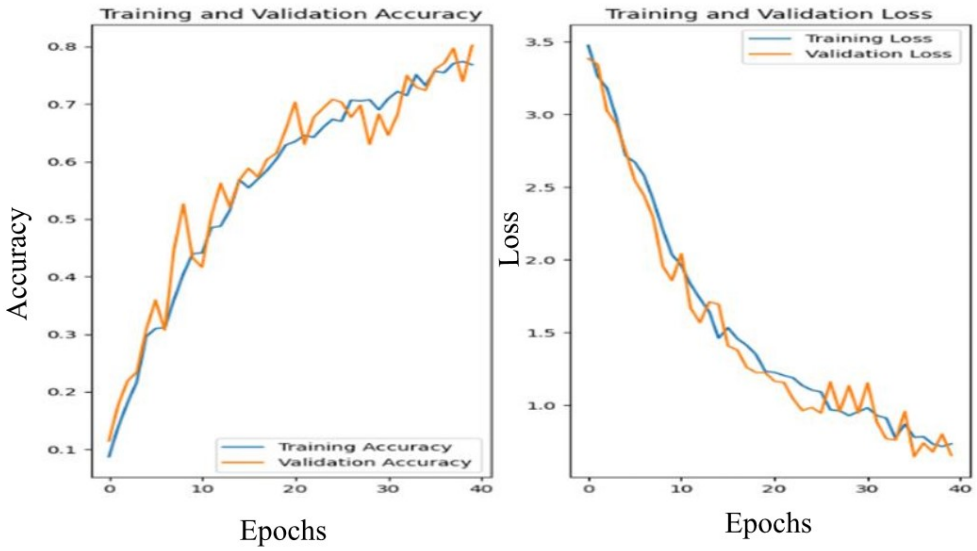

Figure 5.3: Underfitting curve of Alexnet with 40 epochs

Overfitting occurs when our machine learning model tries to cover all the data points or more than the required data points present in the given dataset. Because of this, the model starts caching noise and inaccurate values present in the dataset, and all these factors reduce the efficiency and accuracy of the model. The overfitted model has low bias and high variance.The chances of occurrence of overfitting increase as much we provide training to our model. Here, In the below case 75 epochs are used which increased the training. It means the more we train our model, the more chances of occurring the overfitted model. The concept of the overfitting can be understood by the Fig. 5.4 of the Alexnet with 75 epochs output.

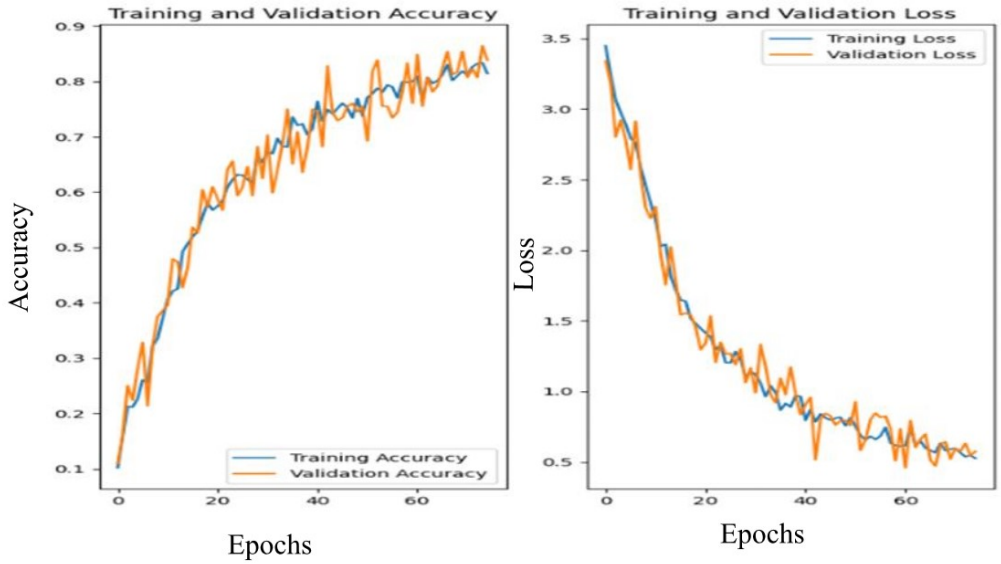

Figure 5.4: Overfitting curve of Alexnet with 75 epochs.

5.4 Project Deployment

Plant disease detection using deep learning involves several steps shown in Fig.5.5, starting from data collection, data preprocessing, model building, model deployment, and finally the development of a website/application that can be used to detect plant diseases.

The first step is data collection, which involves obtaining a suitable image dataset of various plant diseases. This dataset should be diverse enough to cover different types of plants and their diseases. The dataset should also be labelled with the corresponding plant disease for each image.

Once the dataset is obtained, the next step is data preprocessing. This involves preparing the dataset for training by applying transformations such as image resizing, normalization, and augmentation. Image resizing ensures that all images are of the same size while normalization scales the pixel values to a range of 0 to 1. Data augmentation involves applying random transformations such as flipping, rotating, and zooming to increase the diversity of the dataset.

After data preprocessing, the next step is model building. In this project, the deep learning AlexNet model has been used. The model architecture involves several convolutional and pooling layers followed by fully connected layers. The model is trained using the preprocessed dataset and optimized using backpropagation and gradient descent. The objective is to minimize the loss function and maximize the accuracy of the model. Once the model is trained, the next step is model deployment. This involves deploying the model to the cloud using the Google Cloud Platform (GCP). GCP pro-

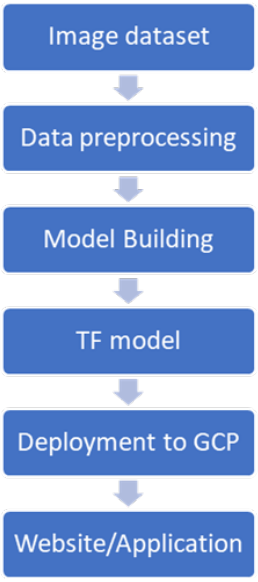

Figure 5.5: Flowchart of building the mobile application

vides a scalable and flexible infrastructure for deploying deep learning models. The model is deployed using TensorFlow, a popular deep learning framework, and can be accessed via an API. Finally, a website/application is developed to enable users to detect plant diseases using the deployed model. The website/application should have an intuitive user interface that allows users to upload images of plants and receive predictions of the corresponding plant disease. The website/application should also display the confidence level of the prediction and provide information on the plant disease.

Detecting plant diseases using the deep learning is a complex process that involves several steps. Each of these stages is essential in ensuring the accuracy and efficiency of the final system. To successfully detect plant diseases, a diverse dataset, meticulous data preprocessing, optimization of the model, deployment to GCP, and website/application development are required. Overall, if all these steps are completed successfully, a reliable and precise plant disease detection system will be created. This system can have a significant impact on promoting sustainable agriculture, improving plant health, and reducing crop losses due to plant diseases.

5.4.1 TF Model

TensorFlow provides different formats for saving models, including the SavedModel format, Keras format, and HDF5 format.

- **SavedModel** format is the default format for TensorFlow models, and it includes the model architecture, weights, and optimizer state. This format is intended for

use with TensorFlow Serving, which provides a way to serve the model as an API.

- **Keras** format is a simplified format that only includes the model architecture and weights. This format is designed to be used with Keras, a popular high-level API for building deep learning models.

- **HDF5** stands for Hierarchical Data Format version 5. It is a binary format that includes the model architecture and weights, similar to the Keras format. However, HDF5 allows for compression, making the model files smaller and easier to transfer. Out of these formats, the HDF5 format (.h5) is often preferred because of its simplicity and flexibility. The HDF5 format is platform-independent and can be used with a variety of programming languages, making it easy to transfer and use the model across different platforms. Additionally, the HDF5 format supports compression, making it possible to save and transfer large models efficiently. Moreover, the HDF5 format allows for easy integration with other libraries and frameworks, such as OpenCV and scikit-learn, which can be used for further processing or analysis of the model's output. Finally, the HDF5 format is supported by many cloud platforms, including GCP and AWS, making it easy to deploy and serve the model as an API. TensorFlow provides multiple formats for saving models, but the HDF5 format (.h5) is often preferred for our project to its simplicity, flexibility, platform-independence, compression, easy integration with other libraries, and support by many cloud platforms.

5.4.2 Deployment to Google Cloud Platform (GCP)

Google Cloud Platform (GCP) is a cloud computing platform provided by Google that offers a variety of services, including compute, storage, networking, machine learning, and more. GCP provides a highly dependable and adaptable infrastructure that is secure and cost-effective, which makes it an ideal option for implementing the plant disease detection system. Although there are other cloud computing platforms available, GCP was selected for this project due to its strong capabilities and its easy integration with other Google services, making it an all-inclusive solution for hosting and deploying the application which is shown in Fig. 5.6.

In our project the trained AlexNet model is saved in .h5 format and is deployed to GCP. To deploy a TensorFlow (.h5) model on Google Cloud Platform (GCP), the first step is to create a GCP account and a new project on the GCP console. Next, a GCP bucket should be created and the TensorFlow model should be uploaded to the bucket in the path "models/potato-model.h5".
Afterward, the Google Cloud SDK should be installed and authenticated. Then we have

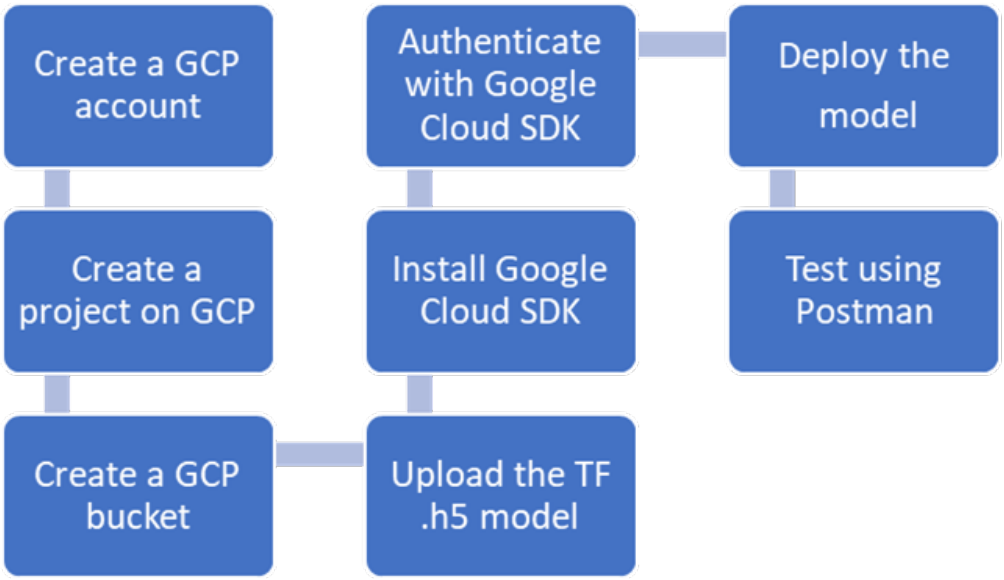

Figure 5.6: Flow of deploying the saved model to Google Cloud Platform (GCP)

to deploy the model. Finally, Postman can be used to test the GCP Function by sending a POST request to the Trigger URL provided by GCP. By following these steps, the TensorFlow model will be deployed on GCP, and it can be used to detect plant diseases.

5.4.3 Android Application

In today's technology-driven world, the use of mobile applications has become increasingly popular across various industries. The agriculture industry is no exception, as farmers are always looking for new ways to improve their crop yields and protect their plants from diseases.

One way to achieve this is by leveraging the power of machine learning to develop a mobile application that can help farmers quickly and accurately detect plant diseases.To accomplish this, we will build a mobile application in React Native, a popular open-source mobile application framework.

The mobile application will be designed to allow farmers to take a picture of a plant leaf using their mobile device's camera and automatically detect if the plant has a disease or not. The app will utilize the pre-trained AlexNet model to detect the disease based on the picture of the plant leaf. along with additional information such as the recommended treatments.

Features:
The mobile application we are building in React Native will have several key features to help farmers quickly and accurately detect plant diseases. These features include:

- User-friendly interface: The app will have a simple and intuitive user interface that is easy for farmers to use.
- Cloud Integration: The app will be integrated with a cloud function hosted on the GCP cloud that will provide the prediction results to the user.
- User authentication: The app will have a secure login system that will require users to authenticate themselves before using the app.
- Image capture: The app will utilize the mobile device's camera to allow farmers to take a picture of a plant leaf.
- Disease detection: The app will use a pre-trained deep learning model to analyze the picture of the plant leaf and detect if the plant has a disease or not.
- Treatment suggestions: If the app detects a disease, it will suggest ways to prevent or treat the detected disease.
- Multi-lingual support: The app will support multiple languages, making it accessible to farmers from different regions and backgrounds.

Overall, the mobile application will be a powerful tool for farmers, providing them with an easy and accessible way to detect plant diseases and take appropriate action to protect their crops.

Chapter 6
Conclusion and Future Scope

Deep learning has shown to be a promising approach for the detection of plant diseases. By training a deep neural network on a dataset of images of diseased and healthy plants, it is possible to accurately classify new images and detect the presence of diseases. In the past few years, several studies have demonstrated the effectiveness of using deep learning approaches for the detection of various plant diseases.

In this project, first, we used several machine learning algorithms, including Support Vector Machine, Linear Discriminant Analysis, and Random Forest Classifier, to classify plant diseases. We compared their accuracies and found that the Random Forest Classifier performed the best.

After that, we trained an AlexNet CNN model to identify plant diseases for potato diseases and for all the plants in Plantvillage Dataset. We created a modified CNN model and compared its performance with the AlexNet model. Additionally, we analyzed the performance of the AlexNet model at various epochs explaining the underfitting and overfitting. For Potato diseases both our model and the AlexNet model obtained an accuracy of 98% where as for all the plant diseases our modified CNN architecture model acquired an accuracy of 87% where as AlexNet model achieved an impressive accuracy of 96% in detecting diseases across all plant types and then we proceeded to develop a website and an android application.

Overall, the use of deep learning for plant disease detection has the potential to significantly improve the efficiency and accuracy of plant disease diagnosis. This can ultimately lead to better crop yields and more sustainable agricultural practices. However, it is important to continue to research and refine these approaches to ensure their continued effectiveness and to address any potential limitations.

In future we are going to work on,

- Exploring different Deep Learning Techniques: Investigating other deep learning techniques such as transfer learning and reinforcement learning could lead to new models with higher accuracy and more robust performance.

- Further the integration of R-CNN into our project report enhances the accuracy and reliability of our object detection and localization pipeline, ensuring the successful achievement of our project goals.

- Increasing the dataset size: Collecting more data for different plant species and disease types can improve the accuracy of disease detection models.

- Building a comprehensive plant disease database: Developing a comprehensive database of plant diseases that includes information about their symptoms, causes, and treatments can provide a valuable resource for researchers and farmers alike.

References

[1] S. R. Dubey and A. S. Jalal, "Adapted approach for fruit disease identification using images," in *Image processing: Concepts, methodologies, tools, and applications*. IGI Global, 2013, pp. 1395–1409.

[2] L. Li, S. Zhang, and B. Wang, "Plant disease detection and classification by deep learning—a review," *IEEE Access*, vol. 9, pp. 56 683–56 698, 2021.

[3] M. Agarwal, A. Singh, S. Arjaria, A. Sinha, and S. Gupta, "Toled: Tomato leaf disease detection using convolution neural network," *Procedia Computer Science*, vol. 167, pp. 293–301, 2020.

[4] M. M. Ozguven and K. Adem, "Automatic detection and classification of leaf spot disease in sugar beet using deep learning algorithms," *Physica A: statistical mechanics and its applications*, vol. 535, p. 122537, 2019.

[5] D. A. Forsyth and J. Ponce, *Computer vision: a modern approach*. prentice hall professional technical reference, 2002.

[6] S. R. Karanam, Y. Srinivas, and M. V. Krishna, "Study on image processing using deep learning techniques," *Materials Today: Proceedings*, vol. 10, p. 2020, 2020.

[7] K. Stergiou, C. Ntakolia, P. Varytis, E. Koumoulos, P. Karlsson, and S. Moustakidis, "Enhancing property prediction and process optimization in building materials through machine learning: A review," *Computational Materials Science*, vol. 220, p. 112031, 2023.

[8] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 international conference on engineering and technology (ICET)*. Ieee, 2017, pp. 1–6.

[9] S. A. I. Laboratory. Image segmentation. [Online]. Available: https://ai.stanford.edu/~syyeung/cvweb/tutorial3.html#:~:text=Another%20important%20subject%20within%20computer,and%20more%20efficiently%20analyze%20it.

[10] Q. Wang, W. Du, C. Ma, and Z. Gu, "Gradient color leaf image segmentation algorithm based on meanshift and kmeans," in *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 5. IEEE, 2021, pp. 1609–1614.

[11] H.-C. Chen, A. M. Widodo, A. Wisnujati, M. Rahaman, J. C.-W. Lin, L. Chen, and C.-E. Weng, "Alexnet convolutional neural network for disease detection and classification of tomato leaf," *Electronics*, vol. 11, no. 6, p. 951, 2022.

[12] L. Ale, A. Sheta, L. Li, Y. Wang, and N. Zhang, "Deep learning based plant disease detection for smart agriculture," in *2019 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2019, pp. 1–6.

[13] B. Liu, Y. Zhang, D. He, and Y. Li, "Identification of apple leaf diseases based on deep convolutional neural networks," *Symmetry*, vol. 10, no. 1, p. 11, 2017.

[14] S. Arya and R. Singh, "A comparative study of cnn and alexnet for detection of disease in potato and mango leaf," in *2019 International conference on issues and challenges in intelligent computing techniques (ICICT)*, vol. 1. IEEE, 2019, pp. 1–6.

[15] A. Chai, B. Li, Y. Shi, Z. Cen, H. Huang, J. Liu *et al.*, "Recognition of tomato foliage disease based on computer vision technology." *Acta Horticulturae Sinica*, vol. 37, no. 9, pp. 1423–1430, 2010.

[16] Z.-R. Li and D.-J. He, "Research on identify technologies of apple's disease based on mobile photograph image analysis," *Computer Engineering and Design*, vol. 31, no. 13, 2010.

[17] Z. Guan, J. Tang, B. Yang, Y. Zhou, D. Fan, Q. Yao *et al.*, "Study on recognition method of rice disease based on image." *Chinese Journal of Rice Science*, vol. 24, no. 5, pp. 497–502, 2010.

[18] S. V. Militante, B. D. Gerardo, and N. V. Dionisio, "Plant leaf detection and disease recognition using deep learning," in *2019 IEEE Eurasia conference on IOT, communication and engineering (ECICE)*. IEEE, 2019, pp. 579–582.

[19] M. A. Jasim and J. M. Al-Tuwaijari, "Plant leaf diseases detection and classification using image processing and deep learning techniques," in *2020 International Conference on Computer Science and Software Engineering (CSASE)*. IEEE, 2020, pp. 259–265.

[20] X.-P. Fan, J.-P. Zou, and Y. Xu, "Recognition of field maize leaf diseases based on improved regional convolutional neural network," *Journal of South China Agricultural University*, vol. 41, no. 6, pp. 82–91, 2020.

[21] A. Fuentes, S. Yoon, S. C. Kim, and D. S. Park, "A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition," *Sensors*, vol. 17, no. 9, p. 2022, 2017.

[22] P. Jiang, Y. Chen, B. Liu, D. He, and C. Liang, "Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks," *IEEE Access*, vol. 7, pp. 59 069–59 080, 2019.

[23] J. Li, L. Lin, K. Tian, and A. Alaa, "Detection of leaf diseases of balsam pear in the field based on improved faster r-cnn," *Transactions of the Chinese Society of Agricultural Engineering*, vol. 36, no. 12, pp. 179–185, 2020.

[24] X. Yu, M. Yang, H. Zhang, D. Li, Y. Tang, and X. Yu, "Research and application of crop diseases detection method based on transfer learning," *Trans. Chin. Soc. Agricult. Eng.*, vol. 51, no. 10, pp. 252–258, 2020.