

**ENHANCING CROP HEALTH BY IDENTIFYING LEAF
DISEASE WITH HIGH ACCURACY USING CNN**

*A Project report submitted in partial fulfillment of the requirements for
the award of the degree of*

**BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMMUNICATION ENGINEERING**

Submitted by

T. Sri Lakshmi Suvarna (319126512118)

N. Navya Sri(319126512101)

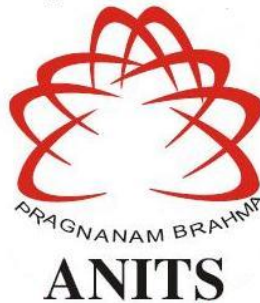
S. Dilleswara Rao(319126512114)

G. Gowtham(319126512084)

Under the guidance of

Mrs. Ch.Anoosha

Assistant Professor



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

**ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(UGC AUTONOMOUS)**

(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC)

Sangivalasa, bheemili mandal, visakhapatnam dist. (A.P)

2022-2023

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(UGC AUTONOMOUS)

(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC)

Sangivalasa, Bheemili mandal, Visakhapatnam dist. (A.P)



CERTIFICATE

This is to certify that the project report entitled "ENHANCING CROP HEALTH BY IDENTIFYING LEAF DISEASE WITH HIGH ACCURACY USING CNN" submitted by T. Sri Lakshmi Suvarna(319126512118), N. Navya Sri(319126512101), S. Dilleswara Rao(319126512114), G.Gowtham(319126512084) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Electronics & Communication Engineering of Anil Neerukonda Institute of technology and Sciences(A), Visakhapatnam is a record of bonafide work carried out under my guidance and supervision.

Project Guide

Mrs. Ch. Anoosha

Assistant Professor

Department of ECE

ANITS

Assistant Professor
Department of E.C.E.

Anil Neerukonda

Institute of Technology & Sciences
Sangivalasa, Visakhapatnam-531 162

Head of the Department

Dr. B. Jagadeesh

Professor&HOD

Department of E.C.E

ANITS

Head of the Department

Department of E C E

Anil Neerukonda Institute of Technology & Science
Sangivalasa - 531 162

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our project guide **Mrs. Ch. Anoosha**, Assistant Professor, Department of Electronics and Communication Engineering, ANITS, for her guidance with unsurpassed knowledge and immense encouragement.

We are grateful to **Dr. B. Jagadeesh**, Head of the Department, Electronics and Communication Engineering, for providing us with the required facilities for the completion of the project work.

We are very much thankful to the **Principal and Management, ANITS, Sangivalasa**, for their encouragement and cooperation to carry out this work.

We express our thanks to all **Teaching faculty** of Department of ECE, whose suggestions during reviews helped us in accomplishment of our project. We would like to thank **all Non-Teaching staff** of the Department of ECE, ANITS for providing great assistance in accomplishment of our project.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

PROJECT STUDENTS

T. Sri Lakshmi Suvarna(319126512118),

N. Navya Sri(319126512101),

S. Dilleswara Rao(319126512114),

G.Gowtham(319126512084)

ABSTRACT

Agriculture is a key source of livelihood. Agriculture provides employment opportunities for village people on large scale in developing country like India. India's agriculture is composed of many crops and according to survey nearly 70% population is depends on agriculture. Most of Indian farmers are adopting manual cultivation due to lagging of technical knowledge. Farmers are unaware of what kind of crops that grows well on their land. When plants are affected by heterogeneous diseases through their leaves that will effects on production of agriculture and profitable loss. Also, reduction in both quality and amount of agricultural production. This project aims to develop an accurate and efficient system for detecting plant diseases using convolutional neural networks (CNN) in Python. The system is trained on a large dataset of images of healthy and diseased plant leaves, and the CNN architecture is optimized to achieve high accuracy in identifying the disease type. The methodology includes data preprocessing, model training, and testing using a validation set. The results show that the CNN-based approach achieves a high accuracy rate in identifying multiple types of plant diseases. The system can be used as an early warning tool for farmers to detect and prevent plant diseases, which can lead to increased crop yields and reduced economic losses. CNN Algorithm is one of the best accuracy providing algorithms compared to K-Means, SVM, RBF, MLP, BPNN etc. We have chosen CNN algorithm because it has an accuracy of above 90% whereas the mentioned algorithms have an accuracy below 90%. We use Python software for the implementation of the leaf disease detection.

Keywords— CNN Classifier, Python Software, Diseased Leaves and Healthy Leaves of plant species

CONTENTS

LIST OF FIGURES	vi
LIST OF TABLES	vii
CHAPTER1 INTRODUCTION	1
1.1 Introduction	2
1.2 Project Objective	3
1.3 Project Outline	5
CHAPTER 2 LITERATURE SURVEY	9
CHAPTER 3 CNN CLASSIFIER	12
3.1 Introduction	13
3.2 Definition	14
3.3 Architecture	14
3.3.1 Convolutional Layer	15
3.3.2 Pooling Layer	20
3.3.3 ReLU Layer	23
3.3.4 Fully Connected Layer	24
3.3.5 Applications.....	26
CHAPTER 4 METHODOLOGY	27
4.1 Data Collection and Preprocessing.....	28
4.2 Data Augmentation	28
4.3 Model Architecture.....	29
4.4 Model Training.....	30
4.5 Model Evaluation	30
4.6 Model Deployment.....	30
4.7 Model Testing.....	30

CHAPTER 5 DATASET	32
CHAPTER 6 PYTHON SOFTWARE.....	36
6.1 Introduction.....	37
6.2 Features	37
6.3 Image Processing Tools	38
6.4 Applications of Python.....	39
6.5 Jupyter IDE	40
CHAPTER 7 OTHER CLASSIFIERS & ITS DRAWBACKS	41
7.1 Different Classifiers	42
7.2 Drawbacks.....	43
CHAPTER 8 RESULTS	46
CONCLUSION	54
REFERENCES	55

LIST OF FIGURES

Figure no	Title	Page no
Fig 1.3.1	Model Architecture	6
Fig 1.3.2	Block Diagram of Architecture	8
Fig. 3.1	Convolutional Layer	16
Fig. 3.2(a)	Basic Convolution Operation	16
Fig. 3.2(b)	Basic Convolution Operation	17
Fig. 3.2(c)	Basic Convolution Operation	17
Fig. 3.2(d)	Basic Convolution Operation	18
Fig. 3.3	Convolution Operation on Volume	18
Fig.3.4	Multi-Convolutional Layer	19
Fig.3.5	1 x 1 Convolution	19
Fig.3.6	One Convolutional Layer	20
Fig.3.7	Convolutional Network	20
Fig.3.8(a)	Pooling Layer	21
Fig.3.8(b)	Max Pooling Layer	22
Fig.3.8(c)	Pooling for Multi Layer	22
Fig.3.9	ReLU Layer	23
Fig.3.10	A Neural Network with Fully Connected Layer	24
Fig.3.11(a)	Block Diagram of CNN	25
Fig.3.11(b)	Block Diagram of CNN	26
Fig.4.1	Flow Chart	31
Fig. 8.1	Accuracy of disease with respect to each plant	52
Fig. 8.2	Training & validation Loss	52
Fig. 8.3	Training & validation Accuracy	52

LIST OF TABLES

Table no	Title	Page no
Table 5.1	Data Set of bell pepper	34
Table 5.2	Data Set of tomato	34
Table 5.3	Data Set of potato	35
Table 8.1	Comparison Table	47
Table 8.2	Results of apple & grape	48
Table 8.3	Results of tomato & pepper	49
Table 8.4	Results of potato & peach	50
Table 8.5	Results of corn	51

CHAPTER 1

INTRODUCTION

CHAPTER 1 INTRODUCTION

1.1 Introduction

India is a developed nation where agriculture supports around 70% of the people. Farmers can choose from a wide variety of eligible crops and choose the right insecticides for their plants. A considerable decrease in both the quantity and quality of crop products is caused by plant disease. Studies on visually discernible patterns on plants are referred to as plant disease research. Plant diseases can occur for a variety of reasons. They are brought on by viruses, nematodes, parasitic plants, molds, bacteria, and nematodes. Diseases are also caused when plants are exposed to excess number of pesticides, environmental pollution, temperature, moisture, light and nutrient abnormalities. When there is nutrient abnormality plants show up discoloration of foliage.

Leaf diseases are a common problem in plants that can cause damage to the plant's fruit, stem, and leaves. The symptoms of leaf diseases can vary, but common signs include discoloration, wilting, yellowing, spotting, and distortion of leaves. These symptoms can cause reduced plant growth, yield, and quality, and in some cases, can lead to plant death. Leaf diseases can be prevented through various measures such as proper sanitation, cultural practices, and the use of fungicides and other control methods. Disease symptoms are typically visible on the fruit, stem, and leaves. The plant leaf is taken into consideration for disease identification since it exhibits disease signs.

In order to successfully cultivate crops on the farm, it is crucial to monitor plant health and disease. In the beginning, a specialist in that sector would manually monitor and analyze plant illnesses. This involves a huge amount of labour and takes a long time to process. Leaf disease is a major problem affecting the growth and productivity of plants, including crops and ornamental plants. For optimal management and the prevention of plant damage, early detection of leaf diseases is essential. Yet, manual leaf disease identification can be time-consuming and error-prone. Here, the detecting process can be greatly aided by the use of machine learning and computer vision algorithms.

Convolutional Neural Networks (CNNs) are a key tool for machine learning and computer vision in the agricultural sector, specifically for the identification of leaf disease. With the help of deep learning algorithms, CNNs are able to process and analyze enormous amounts of data from images of plant leaves to accurately identify and classify diseases affecting them.

This application's primary goal is to automate the diagnosis of plant diseases, which is traditionally done by human experts through visual inspection. The use of CNNs eliminates the need for manual inspection and enables farmers to take timely preventive measures to protect their crops, leading to increased yield and productivity.

An extensive collection of tagged photos of healthy and diseased leaves is used to train a deep learning network. Once trained, the model can identify the disease on a new leaf image by analyzing its visual features and comparing it with the features learned during training. A variety of plant illnesses, including bacterial, viral, and fungal infections, can be found with this method as it is expanded.

By offering a quick, precise, and automated means to identify and prevent plant illnesses, leaf disease detection using CNNs has the potential to alter the way we approach agriculture.

1.2 Project Objective

To create a machine learning model that can effectively and quickly categorize photos of plant leaves into several disease categories, CNN for leaf disease detection may be used. A dataset of annotated photos of healthy and diseased leaves, with various illnesses, can be used to train the model. The convolutional neural network classifier validates the affected leaf picture over the entire dataset, recognizes the illness term, and offers identification accuracy.

The principle objective of the algorithm is to provide an introduction to basic concepts and techniques of deep learning and to promote interest for further study and research in it. For the proper development of the crop, accurate plant disease identification and classification are crucial, and Deep Learning can be used to accomplish this.

Leaf disease detection using CNN algorithm is a potent method for the automatic detection of plant illnesses. A supervised learning algorithm called CNN is created to analyze and classify images. The CNN algorithm uses image recognition technology to distinguish between healthy and sick plant leaves in the event of leaf disease detection, and then classify the type of disease based on those characteristics.

There are many Python libraries available that can help you implement the above methodology such as TensorFlow, Keras, PyTorch, and scikit-learn. You can also use pre-trained models available in these libraries and fine-tune them for your specific task.

The objective of a project on leaf disease detection using CNN could be to develop a system that can accurately sort plant leaves according to whether they are healthy or sick. The system would be trained with a dataset of pictures of both healthy and ill leaves, CNNs would be used to extract pertinent information from the pictures. Some specific objectives that could be pursued in the project include:

1. Assembling and organizing a sizable dataset of plant leaf photos with proper disease categorization labeling.
2. Constructing and preparing a CNN model for leaf disease classification, optimizing hyper parameters and architecture as needed.
3. Assessing the model's performance using relevant metrics, such as accuracy, precision, recall, and F1 score, on a test dataset.
4. Executing additional tests to look into the effects of various variables, including as the size and standard of the training dataset, the use of transfer learning, and the use of data augmentation techniques, on model performance.
5. Comparing the performance of the CNN model in comparison to other computer vision and conventional image processing methods for detecting leaf illness.
6. Creating an user-friendly user interface that will enable users to upload and examine leaf photos for the leaf disease detection system.

Overall, the objective of a leaf disease detection project using CNNs would be to develop an accurate and effective system to identify and categorize plant diseases, with potential applications in agriculture, plant pathology, and environmental monitoring.

Overall, the objective of using CNN for leaf disease detection is to develop an accurate, efficient, and scalable solution that can help farmers and researchers detect and manage plant diseases effectively. For the automatic detection and categorization of leaf diseases, the CNN algorithm is a potent tool. Quick and precise disease type identification can aid in mitigating its spread and minimizing crop losses for farmers and researchers.

1.3 Project Outline

Here is a general framework for convolutional neural networks (CNNs)-based leaf disease detection:

1. Introduction

Briefly study and understand the topic of leaf disease detection using CNNs and also the importance and relevance of the topic.

2. Literature Review

Conduct a literature review of existing research on leaf disease detection using CNNs. Identify gaps in the literature and areas for further research.

3. Dataset collection

Collecting a high-quality dataset is an essential step in building an accurate leaf disease detection model. Collect a dataset of labeled images of healthy and diseased leaves. Prepare the photos by uniformly scaling them and leveling the pixel values. Create training, validation, and test sets from the dataset. Steps to collect a leaf disease dataset:

- a. Identify the target plant species and its common diseases
- b. Collect images of healthy and diseased leaves
- c. Ensure image quality
- d. Collect a sufficient number of images

- e. Annotate the images
- f. Split the dataset
- g. Augment the dataset
- h. Preprocess the images

4. Model Architecture

Use a deep learning framework like TensorFlow or PyTorch to create a CNN model. One or many convolutional layers, pooling layers, and fully linked layers should all be present in the model. Choose an appropriate CNN architecture for the leaf disease detection task. Define the layers and parameters of the CNN model.

Here is example architecture for leaf disease detection using CNN:

- a. Input layer: Accepts input images of size 224x224x3.
- b. Convolutional layers: includes a series of 3x3 convolutional layers with increasing numbers of filters (e.g., 64, 128, and 256). Batch normalization, ReLU activation, and max pooling layers are placed after each convolutional layer.
- c. Fully connected layers: Consists of two 1024-neuron fully connected layers, followed by dropout regularization.
- d. Output layer: Includes a softmax layer that produces the probability for each class (e.g., healthy or diseased).

Depending on the difficulty of the task and the quantity of the dataset, the number of layers and neurons can be changed. Another way to apply transfer learning is to tweak a pre-trained CNN model like VGG, ResNet, or Inception.

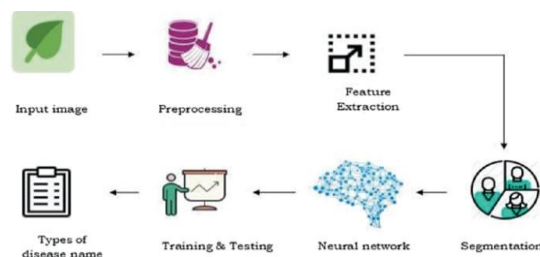


Figure 1.3.1: Model Architecture

5. Model Training

With the training dataset and the suitable optimization strategy, train the CNN model. In order to reduce the discrepancy between the predicted outputs and the true labels, the training procedure entails repeatedly iterating through the training dataset and modifying the model weights after each iteration.

Monitor the training process and adjust the model parameters as needed to improve performance. Model training involves instructing a neural network algorithm to recognize patterns in data by adjusting its parameters to optimize its performance on a particular task. This involves providing the algorithm with a set of input data and the corresponding desired outputs, and then to reduce the gap between the actual and expected outputs, modifies the algorithm's parameters.

During training, the algorithm is exposed to a large amount of data, which it uses to adjust its parameters iteratively. The objective is to reduce the discrepancy between the algorithm's actual output and the desired result. This process is typically repeated many times until the algorithm is able to produce accurate outputs on new, unseen data.

The process of model training can be resource-intensive, requiring significant amounts of computational power and data. However, it is a critical component of the machine learning process, as it enables algorithms to learn from data and make precise assumptions about fresh data.

6. Model Evaluation

Analyze the trained model's performance on the validation dataset to spot any potential problems, such as over- or under-fitting. Performance can be enhanced by modifying the model architecture and hyper parameters as necessary. When evaluating a model for leaf disease detection, there are several metrics that can be used to assess its performance. It is also important to consider factors such as the size and the calibre of the dataset utilized to train and test the model, the choice of features used for classification, and the computational resources required for training and inference.

7. Model Testing

Once you're satisfied with the model's performance, test the finished model on the testing data to get an idea of how well it performs on data that haven't been seen before and to forecast whether the data are healthy or sick.

8. Conclusion

Summarize the findings of the study and discuss the implications and potential applications of the model for detecting leaf disease using CNNs. Overall, the use of CNNs for leaf disease detection is a significant for agriculture applications of machine learning and computer vision, with the potential to improve crop productivity and sustainability.

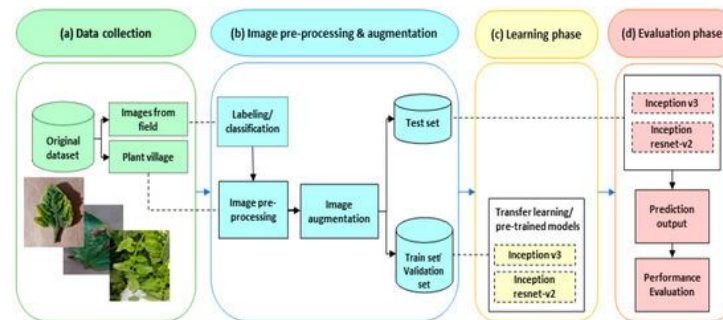


Figure 1.3.2: Block Diagram of Architecture

CHAPTER 2

LITERATURE SURVEY

CHAPTER 2 LITERATURE SURVEY

The Tomato Plant Disease Identification issue was addressed by Sagar Vetall and R.S. Khule² utilising the Image Processing SVM technique. The Multi-class SVM model is trained using 80 photos of a single illness category, and 320 images in total to detect four major diseases. About 15 percent of the image samples are used to validate the created classifier model. The Multi-class SVM was trained using various parameters. The test data for each class of leaves was tested after the training was finished. Better categorization accuracy for all diseases was recorded, with a percentage accuracy of 93.75%.

By utilizing the techniques of artificial intelligence, CNN, and computer science, Tahmina Tashrif Mim¹, Md. Helal Sheikh², Roksana Akter Shampa³, Md. Shamim Reza⁴, and Md. Sanzidul Islam⁵ took on the subject of leaves diseases detection of tomato using image processing. After a successful run of the first epoch, they were able to attain a training accuracy of 58.35% and a validation accuracy of 41.10% at the start of the model's training process. For training this model, they used 30 iterations, and at the last iteration, they got the best results. Validation accuracy is 96.55%, while the final training accuracy is 92.61%.

R. Suguna and Thanjai Vadivel (2022) Automated diagnosis of tomato leaf disease by image processing and fast improved learning. This essay focuses mostly on the tomato disease's planned position according to leaf. Using learning innovation, identification models are created to recognize tomato diseases and bugs, achieving a standard characterisation accuracy of 99.49%. In order to identify the infection in the tomato crop, a CNN-based model is employed. The data on tomato leaves from the Plant Village dataset is used for the analysis. There are 16 different illness classes in the dataset, and each class has a solid image. The model's accuracy in standard testing is 99.5%.

Development of Automated Tomato Plant Diseases Detection System based on Convolutional Neural Network by Raditya Rifqi Rayhan, Muhammad Nurul Puji, and Winda Astuti. In this study, an intelligent system technique will be used to create a disease detection system for tomato plants. The system used clever technology and picture processing. Based on the characteristics of the tomato leaf, the system can identify the type of tomato plant illness. Tomato plant disease detection method based on image processing and convolution neural

networks analysis of tomato leaves (CNN). The system produces accuracy rates for training and testing of 98% and 95%, respectively.

Plant Disease Identification Using Image Processing by Sachin D. Khirade and A. B. Patil. Plant diseases can be found via image processing. Image acquisition, image pre-processing, picture segmentation, feature extraction, and classification are processes in the disease detection process. The methods for identifying plant diseases using photographs of their leaves were covered in this paper. The segmentation and feature extraction algorithms utilised in the identification of plant diseases were also covered in this research.

"Detection of Leaf Disease and Classification Using Image Processing," by Prof. Borkar B. S.5, Tayade Payal, Tilekar Nilesh, Wadekar Satyam, and Abhang Kalyani. Finding whiteites, aphids, and trips on greenhouse vegetables is the first goal. By the use of image processing, they suggest a unique method for the early detection and identification of pests. We utilise a pan-tilt camera with zoom to find items. Hence, we are able to take the image without causing the bugs any discomfort. It serves as an example of how complementing disciplines and methodologies could be combined to create a strong, automated system. Reduced use of insecticides and early pest detection are both beneficial.

"Image-Based Tomato Leaves Diseases Detection using Deep Learning," Belal A. M. Ashqar and Samy S. Abu-Naser. We trained a deep convolutional neural network to recognise 5 diseases using a public dataset of 9000 photos of diseased and healthy tomato leaves taken under controlled settings. The trained model's accuracy on a held-out test set was 99.84%, proving the viability of this method. Overall, the strategy of developing deep learning models on progressively sizable and openly accessible image datasets shows a clear route for massively worldwide smart phone-assisted crop disease diagnosis.

"Plant Disease Detection using CNN," by Nishant Shelar, Suraj Shinde, Shubham Sawant, Shreyash Dhumal, and Kausar Fakir. In this study, a disease recognition model that is based on leaf classification is created. We are using image processing with a convolution neural network to identify plant illnesses (CNN). A type of artificial neural network called a convolutional neural network (CNN) is used for classification and is designed specifically to process pixel input.

CHAPTER 3

CNN CLASSIFIER

CHAPTER 3 CNN CLASSIFIER

3.1 Introduction

Convolutional neural networks (CNNs, or ConvNets) are a type of artificial neural network (ANN) used most frequently in deep learning to interpret visual data. Based on the shared-weight architecture of the convolution kernels or filters that slide along input features and produce translation-equivariant responses known as feature maps, CNNs are also known as Shift Invariant or Space Invariant Artificial Neural Networks (SIANN).

Multilayer perceptrons are modified into CNNs. Fully linked networks, or multilayer perceptrons, are those in which every neuron in one layer is connected to every neuron in the following layer. These networks are prone to over fitting data because of their "full connectedness." Regularizing parameters or preventing over fitting typically entails punishing training variables (such as weight decay) or cutting connectivity (skipped connections, dropout, etc.) By utilizing the hierarchical structure in the data and assembling patterns of increasing complexity using smaller and simpler patterns imprinted in their filters, CNNs adopt a novel strategy for regularization. CNNs are therefore at the lower end of the connectivity and complexity spectrum.

Because of how closely the connectivity pattern between neurons mirrors the structure of the animal visual cortex, convolutional networks were inspired by biological processes. Only in the constrained area of the visual field known as the receptive field do individual cortical neurons respond to inputs. Different neurons' receptive areas partially overlap one another to fill the whole visual field.

Comparatively speaking to other image classification algorithms, CNNs employ a minimal amount of pre-processing. This means that, unlike traditional methods where these filters are hand-engineered, the network learns to optimize the filters (or kernels) through automatic learning. This feature extraction's independence from prior information and human interaction is a significant benefit.

3.2 Definition

Deep learning models known as convolutional neural networks (CNNs) are capable of learning to recognize patterns and characteristics in images. CNNs have excelled in a variety of computer vision applications, such as segmentation, object identification, and image categorization. CNNs can be trained on a dataset of annotated photos of healthy and sick leaves, with various sorts of diseases, in the context of detecting leaf diseases. An advanced form of artificial neural network known as a convolutional neural network substitutes the mathematical operation known as convolution for generic matrix multiplication in at least one of its layers. They are employed in image processing and recognition since they were created primarily to process pixel data. A common type of neural network used in image identification and classification applications is the convolutional neural network (CNN). Convolution is a mathematical procedure that is used to extract features from the input data, and it is the basis for this technique.

The input data for a CNN is passed through a number of convolutional layers, which apply filters to the data and extract pertinent features. In order to inject non-linearity into the model, the output of each convolutional layer is then processed through a non-linear activation function, such as a ReLU (Rectified Linear Unit). One or more fully connected layers, which are comparable to the layers in a conventional neural network, are then given the output of the final convolutional layer. The input data is divided into one or more categories using the fully linked layers. To reduce the discrepancy between the predicted outputs and the actual labels, the weights of the filters in the convolutional layers and the weights of the fully connected layers are changed during the training phase using an optimization approach, such as stochastic gradient descent. In a variety of image identification and classification applications, including object detection, facial recognition, and picture segmentation, CNNs have proven to be quite successful.

3.3 Architecture

An input layer, hidden layers, and an output layer make up a convolutional neural network. Any middle layers in a feed-forward neural network are referred to as hidden layers since the activation function and final convolution hide their inputs and outputs. The hidden layers in a convolutional neural network contain convolutional layers. This typically contains a layer that does a dot product of the input matrix of the layer with the convolution kernel. The activation

mechanism for this product, which is often the Frobenius inner product, is frequently ReLU. The convolution procedure develops a feature map as the convolution kernel moves across the input matrix for the layer, adding to the input of the following layer. This is followed by other layers like pooling layers, fully connected layers, and normalization layers.

3.3.1 Convolutional Layer

The input to a CNN is a tensor with the following dimensions: (number of inputs) \times (input height) \times (input width) \times (input channels). The image is abstracted to a feature map, also known as an activation map, with the following dimensions: (number of inputs) \times (feature map height) (feature map width) \times (feature map channels).

Convolutional layers combine the input and send the resulting information to the following layer. This is comparable to how a neuron in the visual brain might react to a particular stimulus. Data processing for each convolutional neuron's specific receptive field only. Although fully linked feed forward neural networks can be used to categorize data and learn features, this architecture is typically unsuitable for larger inputs (such as high-resolution photos), which would necessitate enormous numbers of neurons because each pixel is an important input feature. Each neuron in the second layer of a fully connected layer for an image of size 100×100 has 10,000 weights. Convolution limits the amount of open parameters, enabling a deeper network.

For instance, just 25 neurons are needed when employing a 5×5 tiling region with identical shared weights for each tile. The vanishing gradients and inflating gradients issues that were present during back propagation in older neural networks are avoided by using regularized weights across fewer parameters.

Depth wise separable convolutional layers, which are built on a depth wise convolution followed by a point wise convolution, can be used in place of normal convolutional layers to speed up processing. Whereas the point wise convolution is a typical convolution limited to the usage of 1×1 kernels, the depth wise convolution is a spatial convolution applied individually over each channel of the input tensor.

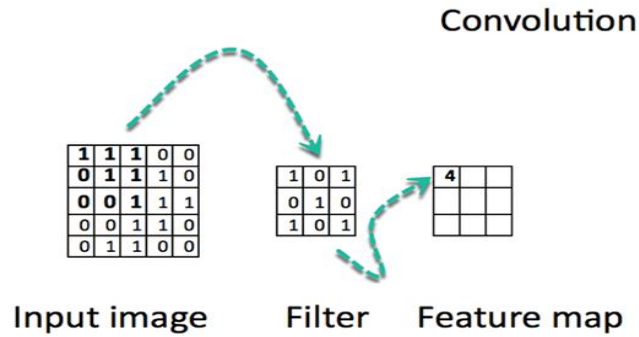


Figure 3.1: Convolutional Layer

Why Convolutions?

Parameter sharing: A feature detector that is helpful in one area of the image is probably also helpful in another area of the image (for example, a vertical edge detector).

Sparsity of connections: Each output value in a layer only depends on a small number of inputs.

Basic steps for Convolution Operation:

Step 1: Add the output after performing element-wise multiplication and overlaying the filter on the input.

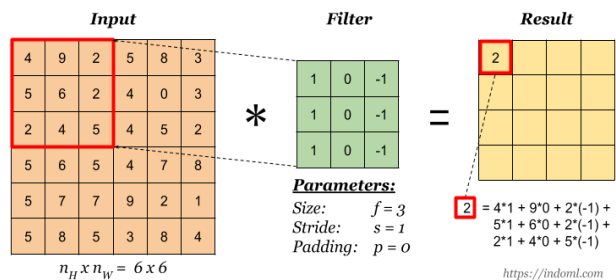


Figure 3.2(a): Basic Convolution Operation

Step 2: To obtain the following outcome, simply move the overlay to the right one place (or in accordance with the stride setting). And so on.

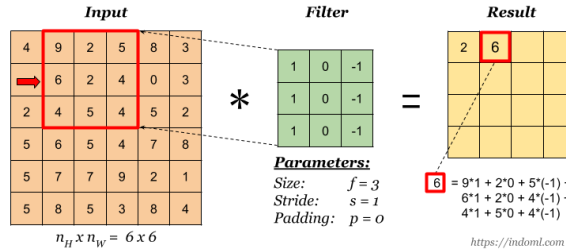


Figure 3.2(b): Basic Convolution Operation

The total number of multiplications required to get to the given conclusion is $(4 \times 4) \times (3 \times 3)$, which equals 144.

Stride

The stride determines how many input cells the filter must travel before the next cell in the result may be calculated. The filters are moved to 1 pixel at a time when the stride is 1. The filters are moved to 2 pixels at a time when the stride is 2, and so on.

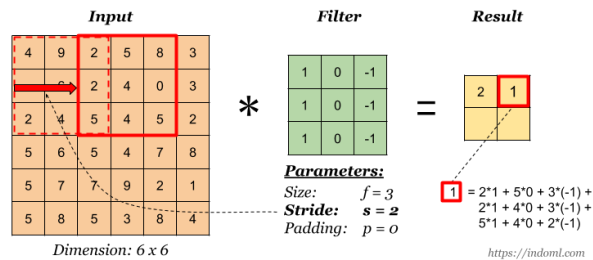


Figure 3.2(c): Basic Convolution Operation

The answer above required a total of 36 multiplications $(2 \times 2) \times (3 \times 3)$

Padding

The advantages of padding include:

- It enables us to use a CONV layer without necessarily reducing the volumes' height and width. This is crucial for creating deeper networks since, in the absence of it, the height/width would decrease as we added more layers.

- b. It enables us to maintain more details at an image's border. Without padding, the boundaries of a picture would have relatively little impact on values at the next layer.

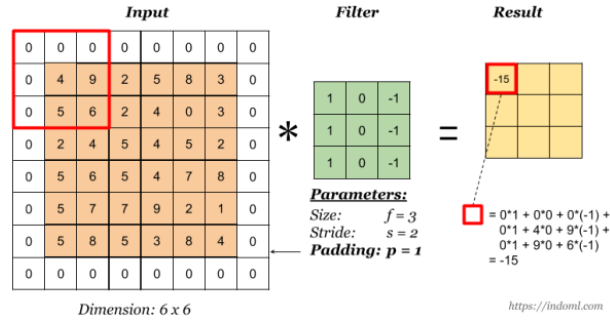


Figure 3.2(d): Basic Convolution Operation

See how the result's dimension has altered as a result of padding. For information on calculating output dimension, see the section below. Many padding jargons:

- a. “valid” padding: no padding
- b. “same” padding: padding to ensure that the output dimension matches the input size

Convolution Operation on Volume

The filter should have the same number of channels as the input when it contains multiple channels (such as an RGB image). Convolution must be applied to each matched channel before the result is added to determine one output cell.

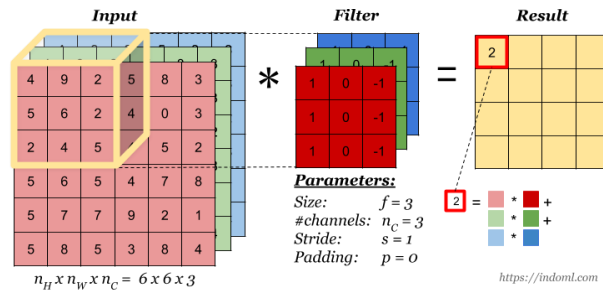


Figure 3.3: Convolution Operation on Volume

$(4 \times 4) \times (3 \times 3 \times 3)$ is the total number of multiplications needed to calculate the answer, which is 432.

Convolution Operation with Multiple Filters

A convolution layer can employ many filters to recognize various features. The layer's output will then contain exactly as many channels as there are filters in the layer.

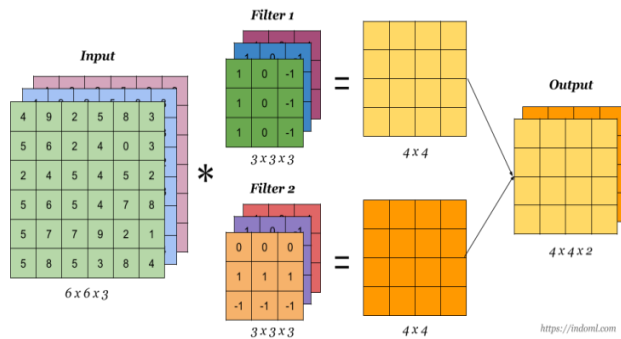


Figure 3.4: Multi-Convolutional Layer

The total number of multiplications required to arrive to the answer is $(4 \times 4 \times 2) \times (3 \times 3 \times 3)$, which results in 864.

1 x 1 Convolutions

Convolution with a 1×1 filter is used here. The result is a flattening or "merging" of the channels, which might help the network, avoid further computations:

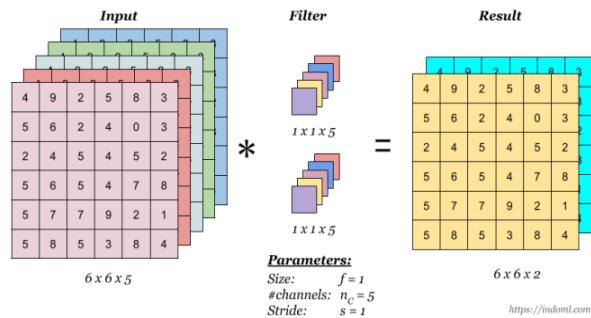


Figure 3.5: 1×1 Convolution

One Convolution Layer

Lastly, a bias (R) is added and an activation function like ReLU or tanh is used to create a convolution layer.

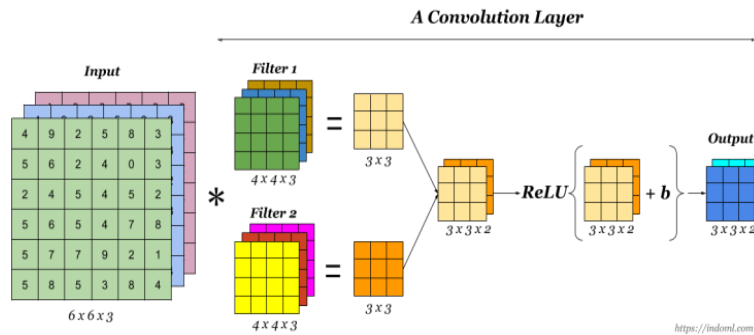


Figure 3.6: One Convolutional Layer

Sample Complete Network

An example network with three convolution layers is shown here. The output of the convolution layer is flattened and connected to a logistic regression or a softmax output layer at the network's end.

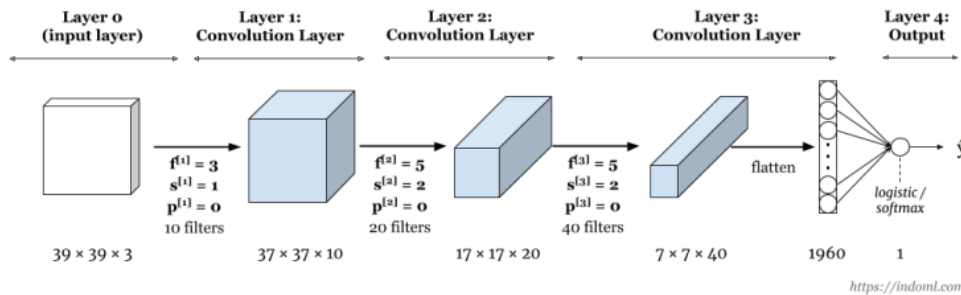


Figure 3.7: Convolutional Network

3.3.2 Pooling Layer

The introduction of a pooling layer allows for faster calculations, smaller representations, and slightly more reliable detection of specific features. Pooling, a type of non-linear down-sampling, is another crucial idea in CNNs. The most popular non-linear function for pooling implementation is max pooling. It divides the input image into a number of rectangles and

outputs the maximum for each of these sub-regions. It makes sense that a feature's approximate placement in relation to other features is more significant than its precise location. Convolutional neural networks use pooling because of this theory. The pooling layer controls over fitting by gradually reducing the spatial size of the representation, the number of parameters, the memory footprint, and the amount of processing in the network. It is referred to as down sampling.

In a CNN architecture, it is normal to sporadically introduce a pooling layer between succeeding convolutional layers, each of which is typically followed by an activation function, such as a ReLU layer. Pooling layers can help with local translation invariance, but until global pooling is implemented, they cannot guarantee global translation invariance in a CNN. The pooling layer typically resizes the input spatially and independently on each depth, or slice. Pooling units may also use average pooling or l2-norm pooling in addition to maximum pooling. Max pooling, who often outperforms average pooling in practise, has recently gained popularity in place of average pooling, which was once widely employed.

The two most widely used types of pooling are maximum and average. Average pooling employs the average value while max pooling uses the maximum value of each local cluster of neurons in the feature map. By merging the outputs of neuron clusters at one layer into a single neuron at the following layer, pooling layers minimise the dimensionality of data. Little clusters are combined using local pooling; 2 2 tile sizes are frequently employed. All of the neurons in the feature map are affected by global pooling.

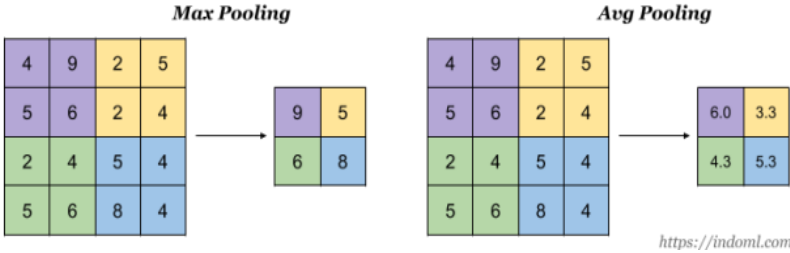


Figure 3.8(a): Pooling Layer

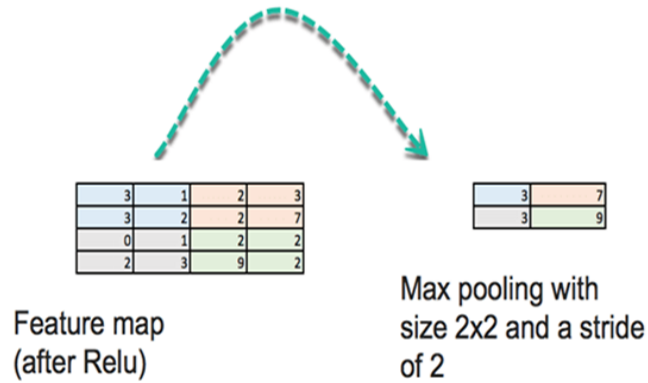


Figure 3.8(b): Max Pooling Layer

Interesting pooling layer characteristics:

- a. it has hyper-parameters:
 - size (f)
 - stride (s)
 - type (max or avg)
- b. but it doesn't have parameter; there's nothing for gradient descent to learn

Pooling reduces the height and width (nW and nH) but leaves the channel count (nC) same when applied to input with many channels:

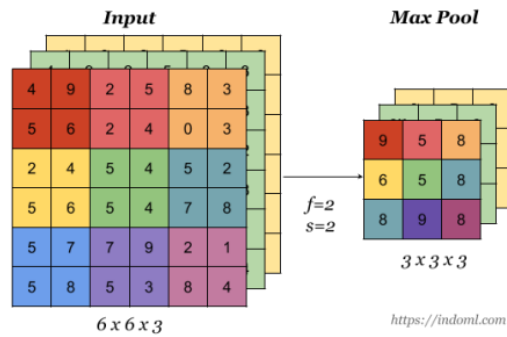


Figure 3.8(c): Pooling for Multi Layer

3.3.3 ReLU Layer

Rectified linear unit, often known as ReLU, applies the non-saturating activation function $f(x) = \max(0, x)$. By setting negative values to zero, it effectively eliminates them from an activation map. Without changing the receptive fields of the convolution layers, it causes nonlinearities to the decision function and the entire network.

Other functions, such as the saturating hyperbolic tangent $f(x) = \tanh(x)$, $f(x) = |\tanh(x)|$, and the sigmoid function $f(x) = \frac{1}{1+e^{-x}}$, can also be employed to improve nonlinearity. ReLU trains the neural network several times quicker than other functions without significantly degrading generalization accuracy, which is why it is frequently chosen over other functions.

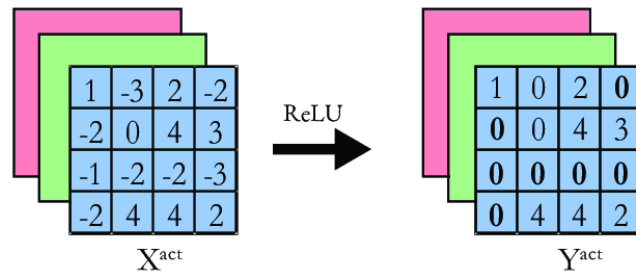


Figure 3.9: ReLU Layer

Compared to the conventional logistic or hyperbolic tangent activation functions, the corrected linear units provide the following three important advantages in convolutional neural networks:

- Rectified linear units effectively propagate the gradient, which lowers the possibility of the vanishing gradient problem, which is frequent in deep neural architectures.
- Rectified linear units solve the cancellation issue and provide a considerably more sparse activation volume at their output by setting their negative value threshold to zero. The sparsity is advantageous for a variety of reasons, but it mostly offers resilience against minor input changes like noise.
- Rectified linear units are substantially more effective to incorporate in convolutional neural networks because they just involve simple computations (mostly comparisons).

3.3.4 Fully Connected Layer

A neural network with fully connected layers is one in which each neuron uses a weights matrix to apply a linear transformation to the input vector. As a result, every input of the input vector influences every output of the output vector, and all layer-to-layer relationships are present. Every neuron in one layer communicates with every other layer's neuron through fully connected layers. The structure is identical to that of a conventional multilayer perceptron neural network (MLP). To categorize the photos, the flattened matrix passes through a layer that is fully connected.

The final classification is carried out using fully connected layers after a number of convolutional and max pooling layers. In conventional (non-convolutional) artificial neural networks, neurons in a fully connected layer have connections to all activations in the preceding layer. As a result, it is possible to compute their activations as an affine transformation with matrix multiplication and bias offset (vector addition of a learned or fixed bias term).

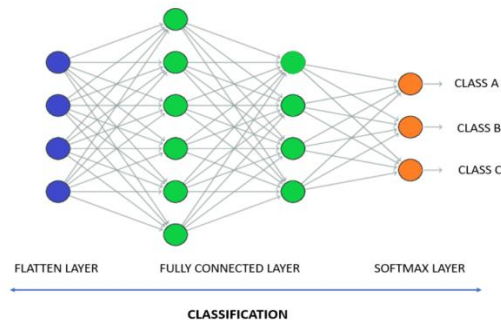


Figure 3.10: A Neural Network with Fully Connected Layer

The Deep Neural Network's Classification section includes the Flatten Layer, Fully Connected Layer, and Softmax Layer combinations. To reduce the discrepancy between the predicted outputs and the actual labels, the weights of the filters in the convolutional layers and the weights of the fully connected layers are changed during the training phase using an optimization approach, such as stochastic gradient descent. In a variety of image identification and classification applications, including object detection, facial recognition, and picture segmentation, CNNs have proven to be quite successful.

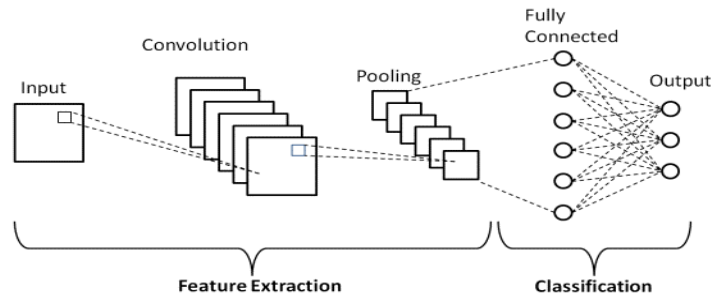


Figure 3.11(a): Block Diagram of CNN

Convolutional Neural Networks (CNNs), which have been demonstrated to be particularly useful in detecting and classifying pictures for computer vision, must include fully linked layers. Convolution and pooling, which divide the image into features and analyze each one separately, are the first steps in the CNN process. A fully connected neural network structure receives the output of this procedure and uses it to determine the final classification.

Why are layers that are fully connected needed?

The entire neural network (used for classification) can be divided into two sections:

- **Feature extraction:** In order to make the classification work in the traditional classification techniques, such as SVMs, we used to extract features from the data. The convolutional layers have the same feature extraction function. CNNs record data more accurately, therefore feature engineering is not necessary.
- **Classification:** A fully connected (FC) neural network can be used to categorise the data into several classes after feature extraction. We may also use a standard classifier, such as SVM, in place of fully linked layers. Yet in order to make the model end-to-end trainable, we typically end up adding FC layers. As an output from the convolutional layers, the fully connected layers learn a (potentially non-linear) function between the high-level features.

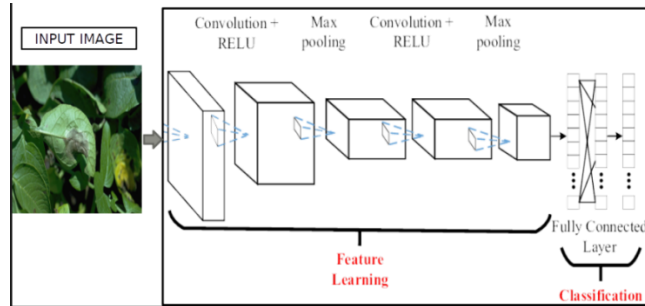


Figure 3.11(b): Block Diagram of CNN

3.3.5 Applications

1. Image recognition
2. Facial emotion recognition
3. Auto translation
4. Handwritten character recognition
5. X-ray image analysis
6. Cancer detection
7. Document classification
8. Biometric authentication

CHAPTER 4

METHODOLOGY

CHAPTER 4 METHODOLOGY

The methodology used in the code in the link is a CNN. This kind of network is employed for tasks like leaf disease detection as well as image classification and object recognition. The input layer, convolutional layers, pooling layers, and output layer are some of the layers that make up the CNN. The image is transformed into an array of pixel values in the input layer. Convolutional procedures are used in the convolutional layers to extract features from the image. The feature vectors' size is decreased using the pooling layers. The retrieved features are then utilized in the output layer to produce a prediction regarding the illness type visible in the image. Popular deep learning frameworks like Tensor Flow or PyTorch can be used to create a CNN in Python.

One common use of computer vision is the detection and diagnosis of leaf diseases using deep learning methods like CNN. In this approach, we'll create a CNN model that can distinguish between healthy and unhealthy leaves using Python and Jupyter Notebook. A step-by-step tutorial for developing this methodology is provided here:

4.1 Data Collection and Preprocessing

Gathering leaf photos of both healthy and sick leaves is the first stage. You can use a number of publicly accessible internet datasets, like the Plant Village dataset. There are train and validation sets for the dataset. The next step is to preprocess the photographs after you have acquired them. Resizing the photos to a common size, converting them to grayscale or RGB, standardizing the pixel values to a range between 0 and 1, and using data augmentation strategies like random rotations and flips to expand the training set are all examples of this. Following preprocessing, the photos are saved as NumPy arrays for training and validation.

4.2 Data Augmentation

Data augmentation is a potent method for expanding the training set for a machine learning system, such as CNN, which can help improve its accuracy and generalization ability.

When it comes to leaf disease detection using CNN, there are several ways to augment the data:

- Flipping: It can be useful to create new training data that is comparable to the original photos but with a different orientation by flipping the image horizontally or vertically.
- Rotation: rotating the image at different angles can help create new training data that is different from the original images.
- Zooming: zooming in or out of the image can help create new training data that focuses on different parts of the image.
- Brightness adjustment: Changing the image's contrast or brightness can aid in producing new training data under various lighting conditions.
- Cropping: cropping the image at different locations can help create new training data that focuses on different parts of the image.
- Adding noise: In order to produce training data that is distinct from the original images, noise can be added to the image.
- Scaling: The creation of new training data that concentrates on various details in the image can be aided by scaling the image up or down.

Through the use of data augmentation techniques, CNN may access more training data, which will help it perform better when identifying leaf diseases. The Keras Picture Data Generator class can be used for this.

4.3 Model Architecture

The CNN model must then be constructed. The Keras package, which offers a simple-to-use API for creating deep learning models, is used to define the CNN model. A number of convolutional and pooling layers are present in the model, which is then followed by a few fully connected layers.

Pooling layers are used to decrease the dimensionality of the feature maps, and convolutional layers are used to extract features from the input images. The features are divided into one of the ten kinds of disorders using fully connected layers. Hyper parameter tweaking is used to optimise the model architecture. To fine-tune a pre-trained model for our particular goal, we can start with one like VGG16 or ResNet50 as the basis model and add our own layers on top.

4.4 Model Training

Once the model is built, we can train it on our preprocessed and augmented dataset. For a specific number of epochs, we can train the model using Keras' fit () function. With the training dataset and the suitable optimization strategy, train the CNN model. In order to reduce the discrepancy between the predicted outputs and the true labels, the training procedure entails repeatedly iterating through the training dataset and modifying the model weights after each iteration. To enhance performance, keep an eye on the training procedure and tweak the model's parameters as necessary.

4.5 Model Evaluation

We must assess our model's performance after training. The effectiveness of the model can be assessed using metrics like accuracy, precision, recall, and F1-score. Analyze the trained model's performance on the validation dataset to spot any potential problems, such as over- or under-fitting. Performance can be enhanced by modifying the model architecture and hyper parameters as necessary. To evaluate the model's effectiveness for each class, a confusion matrix and a classification report are generated.

4.6 Model Deployment

Finally, we can deploy our model to classify new leaf images as healthy or diseased. We can use a Flask web application or a simple Python script to classify new images.

4.7 Model Testing

Test the final model on the test dataset after being pleased with its performance to estimate how well it will perform on data that have not yet been seen and to anticipate whether those data will be healthy or diseased. The learned model is then used to forecast the illness category for fresh photos. Similar to how the training and validation photos were pre-processed; the fresh images are then fed to the trained model for prediction. Summarize the findings of the study and discuss the implications and potential applications of the leaf disease detection model using CNNs. As we launch the program, an input image has to be given to the testing phase. Some image pre-processing steps are taken out like normalization, skew correction, image scaling, noise removal,

thinning and skeletonization, gray scale image and thresholding or binarization etc. Labeling is given to all the images in the dataset. Later augmentation process is taken out in which transformed versions of images are created for the process to recognize a variety of pictures. After the above processes neural network training is given to the process. This training basically refers to a machine learning algorithm which consumes images as an input and gives weights to each of the different image objects. Due to this process differentiation of objects takes place. The testing phase involves two fields they are classified disease and healthy leaf images. Both the images are compared to produce the results. If the output result is not accurate the CNN processing takes back the output and starts reprocessing to produce the effective output results.

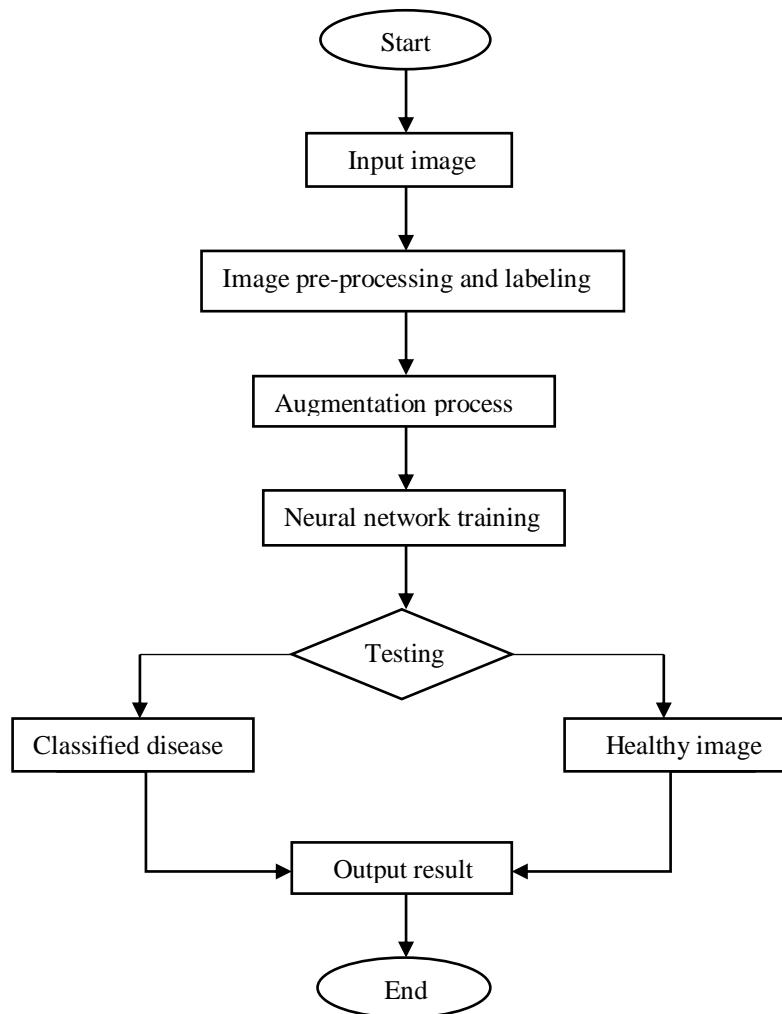


Figure 4.1: Flow Chart

CHAPTER 5

DATASET

CHAPTER 5 DATASET

Finding and treating plant illnesses early on helps prevent them from spreading and causing serious harm, making leaf disease detection a crucial responsibility in agriculture. Machine learning models can be trained using a variety of datasets that are accessible for the detection of leaf diseases. Plant Village is a large dataset of plant images, including those of diseased leaves. It covers over 14 crop species and has over 87,000 images of both healthy and diseased plants. When choosing a dataset for leaf disease detection, it's important to consider the specific crop species you're interested in, as well as the types of diseases you want to detect. Also, you must consider the dataset's quality, which includes the quantity of photos, the precision of the labels, and the variety of the images. Collecting a high-quality dataset for leaf disease detection can be a challenging task, as it requires a significant amount of domain expertise and time. Here are some steps for collecting a good dataset for leaf disease detection:

Identify the target crop species: The first step is to identify the crop species that you want to focus on. Different crops are susceptible to different diseases, so it's important to choose a crop that is commonly grown in the region and has a variety of diseases.

Gather both healthy and ailing leaves: Collecting images of healthy leaves is as important as gathering images of diseased leaves. This helps in creating a balanced dataset, which is necessary for training accurate machine learning models.

Collect images in different lighting conditions: It is important to capture images of leaves in various lighting conditions, as the lighting can affect the appearance of the leaves and the accuracy of the model.

Ensure accurate labeling of images: For a machine learning model to be successfully trained, pictures must be accurately labeled. It is critical that the photos be appropriately labeled by subject-matter specialists.

Collect a sufficient number of images: You need an enough number of photos to cover all potential illness and leaf appearance changes before you can train a machine learning model. In general, the more images you have, the better.

Include a variety of disease severities: Collect images of leaves at different stages of disease development, including leaves that are just starting to show symptoms and those that are severely affected. This helps in creating a diverse dataset and training an accurate model.

Make the dataset available: Once the dataset is collected, it can be made available to the community to help advance research in the field. This can be done by sharing the dataset on online platforms such as Kaggle or GitHub, or by publishing the dataset in a research paper.



	<p>Pepper : Unhealthy leaf (Bacterial spot)</p>
	<p>Pepper : Healthy leaf</p>

Table 5.1: Data Set of Bell Pepper

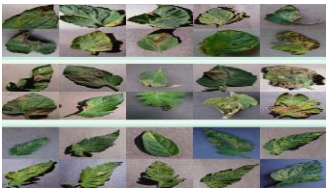

	<p>Tomato : Unhealthy leaf (Leaf mold, Septoria leaf spot, early blight, late blight, mosaic virus)</p>
	<p>Tomato : Healthy leaf</p>

Table 5.2: Data Set of Tomato



	<p style="text-align: center;">Potato : Unhealthy leaf (Early Blight, Late Blight)</p>
	<p style="text-align: center;">Potato : Healthy leaf</p>

Table 5.3: Data Set of Potato

These are some of the leaves of plant species in plant village dataset and the other plants are like corn, apple, peach and grape. In summary, collecting a good dataset for leaf disease detection requires careful planning and attention to detail. The dataset should be diverse, accurately labeled, and include a sufficient number of images to train an accurate machine learning model.

CHAPTER 6

PYTHON SOFTWARE

CHAPTER 6 PYTHON SOFTWARE

6.1 Introduction

What is python?

A high-level, interpreted programming language called Python is employed for a variety of tasks including web development, data analysis, artificial intelligence, scientific computing, and automation. It was made by Guido van Rossum and initially made available in 1991.

Python is renowned for being straightforward, readable, and user-friendly. It has a sizable and vibrant developer community that has produced numerous helpful libraries and frameworks that make working with Python easier. NumPy, Pandas, Django, Flask, TensorFlow, and PyTorch are a few of the well-known Python libraries and frameworks.

Since Python is an open-source language, anyone is free to use and alter the source code. All popular operating systems, including Windows, macOS, and Linux, support it. Many large corporations, like Google, Facebook, Dropbox, and Netflix, also utilize Python.

6.2 Features

High-level programming language Python is renowned for its readability, versatility, and simplicity. Python has a number of important features, including:

Easy to Learn: Even for beginners, Python is basic and concise in its syntax, which makes it simple to learn and comprehend.

Interpreted: Python code can be executed directly without the requirement for a separate compilation process because it is interpreted rather than compiled.

Object-Oriented: Python offers object-oriented programming (OOP), which enables you to write modular, easily understandable code that is reusable.

Dynamically Typed: Python is a dynamically typed language, so rather than being decided at build time, variable types are decided at run time.

Cross-Platform: Without requiring any modifications, Python code can be run on a variety of operating systems, including Windows, Linux, and Mac OS.

Large Standard Library: Python makes it simple to complete tasks like file I/O, networking, and database programming because to its extensive standard library, which offers a wide selection of modules and functions.

Third-Party Libraries: NumPy, Pandas, TensorFlow, and Django are just a few examples of the many third-party libraries available for Python that may be quickly installed and used.

High-level Language: Because Python code is designed at a high degree of abstraction, it is straightforward and readable to convey complicated ideas.

Free and Open-Source: As a free and open-source language, Python can be used, altered, and distributed without the need for a license.

Community Support: There is a sizable and active Python developer community that contributes to the language, builds libraries, and offers help via online forums and communities.

6.3 Image Processing Tools

Python provides various image processing tools that can be used for tasks such as reading, manipulating, and analyzing digital images. The most well-liked Python image processing utilities include:

Pillow: Pillow provides a suite of tools for viewing, processing, and saving several image file formats, including JPEG, PNG, and BMP. It is a fork of the Python Imaging Library (PIL).

OpenCV: The well-known open-source OpenCV (Open Source Computer Vision Library) library offers a variety of image processing capabilities, such as object detection, image segmentation, and feature detection.

Scikit-image: The Python image processing package scikit-image offers a selection of methods for image augmentation, segmentation, and feature extraction.

NumPy: For scientific computing, NumPy is a potent Python library that supports multidimensional arrays and matrix operations. It is often used in conjunction with other image processing libraries, such as Pillow or OpenCV.

Matplotlib: An image display and manipulation tool called Matplotlib is a data visualisation library. It offers a variety of picture viewing tools, including the ability to plot histograms and color maps.

SciPy: Image filtering, segmentation, and feature identification are just a few of the capabilities for image processing offered by the SciPy Python library for scientific computing.

Mahotas: A variety of image analysis capabilities, such as feature extraction and segmentation, are offered by the Python package Mahotas for computer vision and image processing.

These image processing technologies can be applied to a variety of tasks, such as computer vision, picture recognition, and imaging in the medical field, among others.

6.4 Applications of Python

Python is a flexible programming language with a wide range of applications in different sectors. Among the most widely used Python programmes are:

Web Development: With well-known web frameworks like Django and Flask that offer tools for developing online apps and APIs, Python is frequently used for web development.

Data Science: With packages like NumPy, Pandas, and Scikit-learn that offer tools for data manipulation, analysis, and machine learning, Python is one of the most widely used programming languages for data science.

Artificial Intelligence and Machine Learning: With packages like TensorFlow, PyTorch, and Keras that offer tools for deep learning, neural networks, and natural language processing, Python is a well-known language for artificial intelligence and machine learning.

Scientific Computing: With modules like SciPy and Matplotlib that offer tools for numerical computing, data visualization, and simulation, Python is widely used in scientific computing and research.

Desktop Application Development: Python can be used for desktop application development, with libraries such as PyQt and wxPython that provide tools for building graphical user interfaces.

Game Development: Python can be used for game development, with libraries such as Pygame that provide tools for building 2D games.

Automation and Scripting: Python is often used for automation and scripting, with tools such as Selenium and BeautifulSoup that provide instruments for test automation and web scraping.

Education: Python is frequently used in education, and its straightforward syntax makes it a great language to introduce new programmers to the fundamentals of programming.

These are just some of the many applications of Python, and its popularity continues to grow due to its simplicity, flexibility, and wide range of available libraries and frameworks.

6.5 Jupyter IDE

An interactive computing environment for programming, data research, and data visualization utilizing a variety of programming languages, such as Python, R, and Julia, is provided by the open-source web application known as Jupyter IDE.

The three primary programming languages that it supports—Julia, Python, and R—are the source of the term "Jupyter." A web-based interface offered by the Jupyter IDE enables users to write, run, and debug code in real-time as well as create and share interactive notebooks that include code, data, visualizations, and explanatory text.

Some of the key features of Jupyter IDE include:

Interactive computing: Jupyter provides an interactive computing environment that allows users to execute code in real-time and visualize the results immediately.

Multiple programming languages: Python, R, Julia, and other programming languages are among those supported by Jupyter.

Code sharing and collaboration: Jupyter allows users to share and collaborate on code and data using notebooks that can be easily exported to various formats, including HTML and PDF.

CHAPTER 7

OTHER CLASSIFIERS AND ITS DRAWBACKS

CHAPTER 7 OTHER CLASSIFIERS AND ITS DRAWBACKS

7.1 Different Classifiers

For the purpose of identifying leaf diseases, various classifier types are employed. The specifics of the leaf disease detection problem, such as the amount and complexity of the dataset, the required level of interpretability, and the computational resources available for training and inference, will determine which classifier should be used. Among the most typical ones are:

Decision Tree Classifier: A well-liked algorithm in machine learning for supervised learning is the decision tree classifier. Based on the characteristics of the incoming data, a tree-like structure of decisions and their potential repercussions is built. The tree is capable of making predictions on new, unlabeled data because it was trained on a labeled dataset. Recursively dividing the input space into smaller areas based on the characteristics of the input data is how decision tree classifiers operate.

Random Forest Classifier: An ensemble learning approach called a random forest classifier mixes various decision trees to increase classification accuracy. A majority vote of the individual tree classifications determines the final classification after each tree in the forest has been trained using a randomly chosen portion of the training data. Compared to single decision tree classifiers, the random forest classifier has a number of advantages. By merging the predictions of different trees, it lowers the possibility of over fitting and enhances generalization. Moreover, it is less sensitive to data noise and outliers.

Support Vector Machine (SVM) Classifier: A supervised learning algorithm that can be applied to classification tasks is the SVM. Finding the hyper plane that best divides the data into distinct classes is how SVM classifiers operate. Both linearly and non-linearly separable datasets can be used successfully with them. Binary and multi-class classification issues can both be solved with SVMs. In the instance of binary classification, SVM identifies the hyper plane with the greatest potential margin of separation between the two classes. SVM employs a number of approaches to split the multi-class classification problem into a number of binary classification problems.

Naive Bayes classifier: A straightforward and well-liked approach for classification tasks is the naive Bayes classifier. The Bayes theorem and the presumption of independence among the features (or variables) of the data serve as its foundations. Naive Bayes classifier remains a popular choice for classification tasks, particularly in situations where the data is high-dimensional, and the training data is limited. It is computationally efficient and can be trained and applied in real-time.

Linear regression Classifier: The supervised learning process known as linear regression, which is used for regression analysis, forecasts a continuous output value based on one or more input features. Linear regression can be a useful classifier in some cases, its limitations in terms of predictive power, sensitivity to outliers, binary classification limitations, assumptions of homoscedasticity, and limited feature selection must be taken into account when considering its use in classification tasks.

K-means Classifier: K-means is a well-liked clustering algorithm that is applied in machine learning and data mining to group together comparable data points based on their characteristics. The algorithm divides the data into K clusters iteratively, where K is a user-specified parameter that indicates the desired number of clusters. This non-parametric technique categorizes data based on the k training examples that are closest to the samples in the feature space.

7.2 Drawbacks

Drawbacks of Decision Tree:

- a. **Over-fitting:** Whenever a model learns to fit training data too closely and is unable to generalize to new data, it is said to be over-fitting. Decision trees are prone to this.
- b. **Poor performance on imbalanced data:** For datasets with imbalanced classes, where one class has much less samples than the others, decision trees may have trouble performing adequately.
- c. **Instability:** Decision trees can be unstable, which means that even minor adjustments to the training set can result in significant alterations to the tree's structure.

Drawbacks of Random Forest:

- a. Complexity: Random Forest models can be quite complex, especially when many decision trees are used. This can make it difficult to interpret the model and explain the predictions.
- b. Memory and computation: Random Forest models can require a large amount of memory and computation, especially for large datasets with many features.
- c. Sensitivity to noise: Random Forest can be sensitive to noisy features, which can negatively affect the model's accuracy.

Drawbacks of SVM:

- a. Sensitivity to the choice of kernel: To move the input data into a higher dimensional space where the data may be more easily segregated, SVMs use kernels.
- b. Computationally expensive for large datasets: SVMs are relatively slow compared to other machine learning algorithms, especially for large datasets.
- c. Difficulty in choosing optimal hyper parameters: To get the best performance out of SVMs, a number of hyper parameters must be tweaked.

Drawbacks of Naive Bayes:

- a. Zero-frequency problem: This approach encounters the "zero-frequency problem," where it gives a categorical variable with zero probability if its category was not present in the training dataset but was present in the test data set.
- b. Naive Bayes classifier is a probabilistic model, and its predictions are based on probability estimates. These estimates may not always reflect the true probabilities of the data.
- c. Poor estimator: Naive Bayes classifier is known to be a poor estimator, meaning it may assign low probabilities to the correct class, especially when there are few training samples.

Drawbacks of Linear regression:

- a. Limited Predictive Power: The assumption underlying linear regression is that there is always a linear connection between the input features and the output variable. Poor performance on data sets with non-linear relationships may be the outcome of this.
- b. Binary Classification Limitations: Binary classification problems, where the outcome variable is either 0 or 1, do not lend themselves well to linear regression. While the model can produce continuous output values, thresholding these values to obtain binary classifications can result in inaccurate predictions.
- c. Limited Feature Selection: Linear regression assumes that all input features are linearly related to the output variable. This can limit the model's ability to capture complex interactions between variables and may result in suboptimal predictions.

Drawbacks of K-means:

- a. Sensitivity to initial centroid selection: Because the K-means algorithm is sensitive to the initial centroids chosen for each cluster, various initial centroids may provide different clustering outcomes.
- b. Cannot handle non-linearly separable data: K-means algorithm is designed to work only with linearly separable data, i.e., data points that can be separated by straight lines.
- c. Assumes equal variance and size of clusters: K-means algorithm assumes that the clusters have equal variance and size, which may not always be true in practice. This can lead to inaccurate clustering results in some cases.

Hence, CNN is "good" over all classifiers depends on the specific task and dataset being considered. Furthermore, the performance of a CNN can be highly dependent on its architecture, hyper parameters, and training data. In conclusion, CNNs are a strong and adaptable class of classifiers; the task and the practitioner's level of competence determine how effective they are.

CHAPTER 8

RESULTS

CHAPTER 8 RESULTS

Table 8.1 provides a summary of our CNN model's performance in identifying leaf diseases. The findings indicate that on the test dataset, our model had an overall accuracy of more than 90%.

Plant species	Disease type	Accuracies of identifying leaf disease from the Reference Papers						Accuracy of identifying the leaf disease by CNN
		Random Forest	SVM	Decision Tree	K means	Linear Regression	Naives Bayes	
Apple	Cedar Rust, Black Rot, Scab	93%	90%	85%	79%	46%	61%	94%
Bell Pepper	Bacterial spot	90%	86%	84%	83%	54%	82%	94%
Corn	Common Rust, Gray Leafspot, Cercospora Leaf Spot	93%	86%	91%	84%	59%	87%	94%
Grape	Black Rot, Black Measles, Leaf Blight	86%	84%	78%	83%	70%	77%	97%
Peach	Bacterial spot	88%	87%	79%	89%	34%	88%	92%
Potato	Early Blight, Late Blight	92%	87%	91%	89%	16%	85%	96%
Tomato	Virus mosaic, bacterial spot, Leaf mold, late blight, and early blight	85%	81%	73%	74%	69%	45%	97%

Table 8.1: Comparison Table

From the table 8.1 given above various types of existing methods are compared to CNN, and we found that there is a difference between the accuracies of the existing methods and CNN. The CNN method shows an accuracy of about 90% on an average for every plant. This shows that CNN is the best accuracy providing methods when compared to above mentioned existing methods.

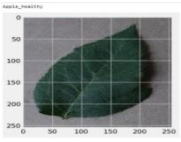
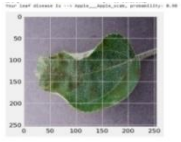
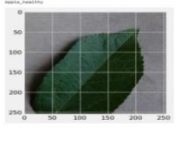

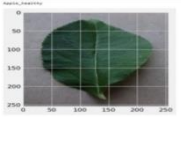
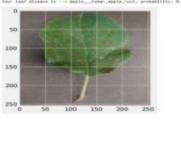
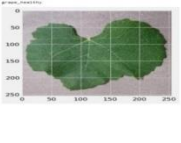
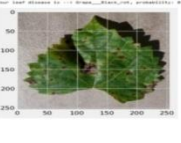
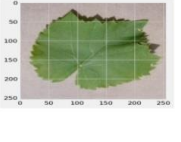
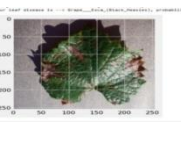
Plant Species	Healthy Leaf	Unhealthy Leaf	Disease name	Probability
Apple			Apple Scab	0.98
			Black Rot	0.94
			Cedar Apple Rust	0.99
Grape			Black Rot	0.97
			Black Measles	0.99

Table 8.2: Results of apple & grape

In the table 8.2 the apple and grape leaf diseases were described. The model was made to work upon the apple leaf diseases with accuracy of 94% Black rot and 99% of Apple scab whereas the grape leaf diseases with accuracy of 97% of Black rot and 99% of Black measles.

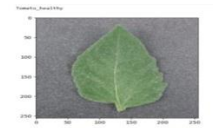
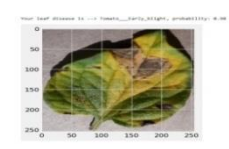







Plant Species	Healthy Leaf	Unhealthy Leaf	Disease name	Probability
Tomato			Early Blight	0.98
			Mosaic Virus	0.99
			Septoria Leaf Spot	0.97
			Leaf Mold	0.96
Pepper			Bacterial Spot	0.94

Table 8.3: Results of Tomato & Pepper

In the table 8.3 the tomato and pepper leaf diseases were described. The model was made to work upon the tomato leaf diseases with accuracy of 98% Early Blight, 99% of Mosaic Virus, 97% of Septoria Leaf spot and Leaf Mold of 96% whereas the pepper leaf diseases with accuracy of 94% of Bacterial Spot.

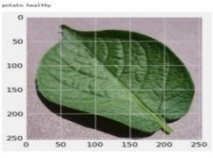
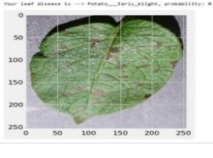
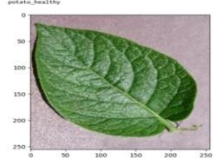

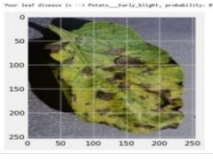
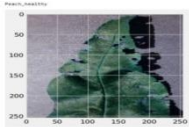
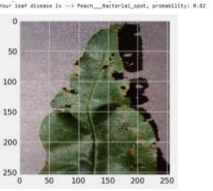
Plant Species	Healthy Leaf	Unhealthy Leaf	Disease name	Probability
Potato			Early Blight	0.67
		 leaf is unhealthy and the disease is Potato_LateBlight probability: 0.96617 Potato_LateBlight	Late Blight	0.96
		 Your leaf disease is --> Potato_EarlyBlight, probability: 0.99	Early Blight	0.99
Peach		 Your leaf disease is --> Peach_BacterialSpot, probability: 0.92	Bacterial Spot	0.92

Table 8.4: Results of Potato & Peach

In the table 8.4 the potato and peach leaf diseases were described. The model was made to work upon the potato leaf diseases with accuracy of 67% Early Blight, 96% of Late Blight and 99% of Early Blight whereas the peach leaf diseases with accuracy of 92% of Bacterial Spot.

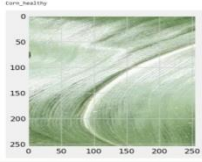
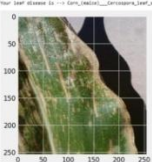
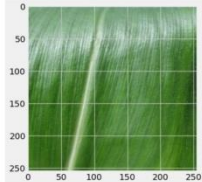
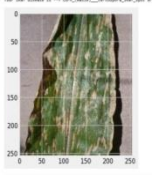
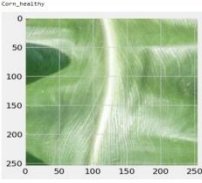
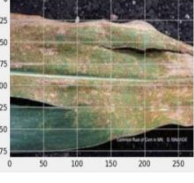
Plant Species	Healthy Leaf	Unhealthy Leaf	Disease name	Probability
Corn			Cercospora Leaf Spot	0.79
			Cercospora Leaf Spot	0.89
			Common Rust	0.94

Table 8.5: Results of corn

In the table 8.5 the corn leaf diseases were described. The model was made to work upon the corn leaf diseases with accuracy of 79% Cercospora Leaf Spot, 89% of Cercospora Leaf Spot and 94% of Common Rust. Since the accuracy percentage is high, the model can effectively predict maize leaf diseases. Our model has the best accuracy on the classes of Cedar Apple Rust, Black Measles of Grape, Early Blight of Potato, and Mosaic Virus of Tomato, according to the results with 99%, while the accuracy on the Cercospora Leaf Spot of Corn class is relatively lower at 79%.

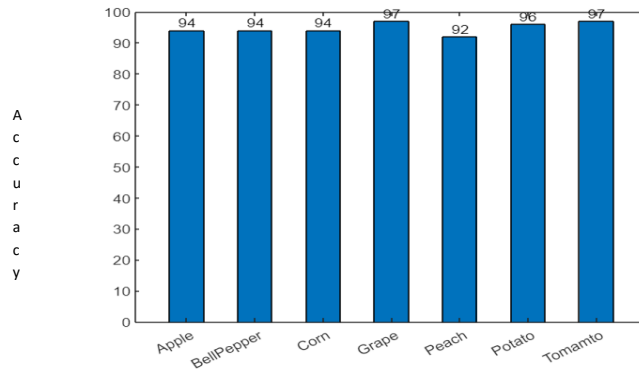


Figure 8.1: Accuracy of Disease with respect to each plant

This graph 8.1 shows that on x-axis the name of the plant species and on y-axis the accuracy.

Name(accuracy)=apple(94%),bellpepper(94%),corn(94%),grape(97%),peach(92%),

Potato (96%), tomato (97%)

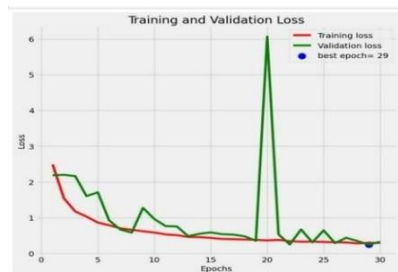


Figure 8.2: Training & validation loss



Figure 8.3: Training & validation accuracy

On the x-axis are Epochs, while on the y-axis is Loss. The red line represents training loss, while the green line represents validation loss. And at 29, the best epoch starts.

On the x-axis Epochs and the y-axis precision, this is shown in Fig. 8.3. The red line represents training accuracy, whereas the green line represents validation accuracy. And at 22, the best epoch starts.

Accuracy and loss are crucial measures for measuring how well machine learning models perform during training and validation. Loss is a statistic used to compare the output actually produced to the output anticipated by the model. Improved model performance is shown by a reduced loss value. The model's accuracy, on the other hand, is an assessment of how well it can classify the incoming data. The model is operating more effectively when the accuracy value is greater. The validation loss may start to climb and the validation accuracy may start to decline, whilst the training loss may continue to decline and the training accuracy may continue to rise, if the training data have caused the model to be over fit. In order to ensure that the model is not over-fitting and is doing well on both the training and validation data, it is crucial to keep an eye on the training and validation loss as well as accuracy.

CONCLUSION

A number of academic projects and commercial products have been created in recent years as a result of the increased interest in using CNNs for the identification of leaf diseases. These CNN applications for leaf disease identification have shown remarkable results and hold the promise of revolutionizing the agricultural industry. First gather and preprocess the dataset of leaf image data in this process, then divide it into training and validation sets. Using the training set, create the CNN architecture, train it, and then evaluate its performance using the validation set. Fine-tune the model and evaluate it on a separate test set. Once satisfied with the model's performance, deploy it for real-world leaf disease detection. Python offers several powerful libraries such as TensorFlow, Keras, PyTorch, and scikit-learn, that can help to implement the leaf disease detection using CNN efficiently. This application of computer vision can significantly contribute to the development of sustainable agriculture practices and can help farmers make better decisions for crop management.

Convolutional Neural Networks (CNN)-based leaf disease detection is a useful use of computer vision in agriculture, to sum up. We can precisely identify plant diseases with CNN's assistance, enabling farmers to take the necessary precautions to stop the spread of disease and boost crop productivity. In conclusion, using CNNs to detect leaf diseases is a key use of computer vision and machine learning in agriculture, with the potential to increase agricultural output and sustainability. Hence CNN is the best from all other algorithm because it has an accuracy of above 90% whereas the mentioned algorithms have accuracy below 90%.

REFERENCES

1. Sachin D. Khirade, A. B. Patil, "Plant Disease Detection Using Image Processing",2015 International Conference on Computing Communication Control and Automation.
2. Raditya Rifqi Rayhan, Muhammad Nurul Puji, WindaAstuti,"Development of Automatic Tomato Plant Diseases Detection System based on Convolutional Neural Network", Second Asia Pacific International Conference on Industrial Engineering and Operations Management Surakarta, Indonesia, September 14-16, 2021.
3. Thanjai Vadivel & R. Suguna (2022) Automatic recognition of tomato leaf disease using fast enhanced learning with image processing, Act a Agriculture Scandinavica, Section B Soil & Plant Science, 72:1, 312-324, DOI:10.1080/09064710.2021.1976266.
4. Tahmina Tashrif Mim¹, Md. Helal Sheikh², Rokhsana Akter Shampa³, Md. Shamim Reza⁴and Md. Sanzidul Islam, "Leaves Diseases Detection of Tomato Using Image Processing", 1,2,3,4 Dept. of Computer Science and Engineering, Daffodil International University, Dhaka, Bangladesh 5 Dept. of SWE, Daffodil International University, Dhaka, Bangladesh.
5. Sagar Vetall¹, R.S. Khule², "Tomato Plant Disease Detection using Image Processing", Department of Electronics and Telecommunication, Matoshri College of Engineering & Research Centre, Eklahare, Nashik, Savitribai Phule University of Pune, India^{1,2}
6. Mrunalini R. et al., An application of K-means clustering and artificial intelligence in pattern recognition for crop diseases, 2011.
7. S.Raj Kumar , S.Sowrirajan," Automatic Leaf Disease Detection and Classification using Hybrid Features and Supervised Classifier", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 5, Issue 6,2016..
8. A.-K. Mahlein, T. Rumpf, P. Welke et al., "Development of spectral indices for detecting and identifying plant diseases," Remote Sensing of Environment, vol. 128, pp. 21–30, 2013. View at Publisher · View at Google Scholar · View at Scopus.

9. Arivazhagan S, Newlin Shebiah R, Ananthi S, Vishnu Varthini S. Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features. *Agric Eng Int CIGR* 2013; 15(1):211–7.
10. Dhaygude Sanjay B, Kumbhar Nitin P. Agricultural plant leaf disease detection using image processing. *Int J Adv Res Electr Electron InstrumEng* 2013;2(1).
11. Pham TN, Van Tran L, Dao SVT. 2020. Early disease classification of mango leaves using feed-forward neural network and hybrid metaheuristic feature selection. *IEEE Access*.8:189960– 189973
12. M. Hasan, S. Ullah, M. Khan and K. Khurshid, "COMPARATIVE ANALYSIS OF SVM, ANN AND CNN FOR CLASSIFYING VEGETATION SPECIES USING HYPERSPECTRAL THERMAL INFRARED DATA", *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. -213, pp. 1861-1868, 2019. Available: 10.5194/isprs archives-xlii2-w13-1861-2019
13. Y. Oo and N. Htun, "Plant Leaf Disease Detection and Classification using Image Processing", *International Journal of Research and Engineering*, vol. 5, no. 9, pp. 516-523, 2018. Available:10.21276/ijre.2018.5.9.4
14. Padmavathi and K. Thangadurai, "Implementation of RGB and Grayscale Images in Plant Leaves Disease Detection–Comparative Study", *Indian Journal of Science and Technology*, vol. 9, no. 6, 2016. Available: 10.17485/ijst/2016/v9i6/77739.
15. S. Jadhav, "Convolutional Neural Networks for Leaf Image-Based Plant Disease Classification", *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 8, no. 4, p. 328, 2019. Available:10.11591/ijai.v8.i4.pp328-341.
16. S. Sankaran, A. Mishra, R. Ehsani and C. Davis, "A review of advanced techniques for detecting plant diseases", *Computers and Electronics in Agriculture*, vol. 72, no. 1, pp. 1-13, 2010. Available: 10.1016/j.compag.2010.02.007.
17. Isleib, "Signs and symptoms of plant disease: Is it fungal, viral or bacterial?" *MSU Extension*, 2012. [Online]. Available: https://www.canr.msu.edu/news/signs_and_symptoms_of_plant_disease_is_it_fungal_viral_or_bacterial. [Accessed: 11- Apr- 2020].

18. P. Pawara, E. Okafor, L. Schomaker and M. Wiering, "Data Augmentation for Plant Classification", 2017. Available: https://www.researchgate.net/publication/319990090_Data_Augmentation_for_Plant_Classification. [Accessed 11 April 2020].
19. M. Ji, L. Zhang and Q. Wu, "Automatic grape leaf diseases identification via United Model based on multiple convolutional neural networks", *Information Processing in Agriculture*, 2019. Available: 10.1016/j.inpa.2019.10.003.
20. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei L. (2009). "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. (IEEE).
21. C. Buche, S. Walleign and M. Polceanu, "Soybean Plant Disease Identification Using Convolutional Neural Network", 2018. [Accessed 22 April 2020].
22. P. Sharma, Y. Berwal and W. Ghai, "Performance analysis of deep learning CNN models for disease detection in plants using image segmentation", *Information Processing in Agriculture*, 2019. Available: 10.1016/j.inpa.2019.11.001.
23. S. Mohanty, D. Hughes and M. Salathé, "Using Deep Learning for Image-Based Plant Disease Detection", *Frontiers in Plant Science*, vol. 7, 2016. Available: 10.3389/fpls.2016.01419.
24. S. Lee, H. Goëau, P. Bonnet and A. Joly, "New perspectives on plant disease characterization based on deep learning", *Computers and Electronics in Agriculture*, vol. 170, p. 105220, 2020. Available: 10.1016/j.compag.2020.105220.
25. K. Lin, L. Gong, Y. Huang, C. Liu and J. Pan, "Deep Learning Based Segmentation and Quantification of Cucumber Powdery Mildew Using Convolutional Neural Network", *Frontiers in Plant Science*, vol. 10, 2019. Available: 10.3389/fpls.2019.00155.