PLANT DISEASE DETECTION USING ALEXNET MODEL

A PROJECT REPORT

Submitted by

Saumyakant Baral –

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

Electronics and Telecommunication Engineering

Department of Electronics and Telecommunication Engineering

C.V. RAMAN GLOBAL UNIVERSITY

BHUBANESWAR - ODISHA – 752054

May 2024

C.V. RAMAN GLOBAL UNIVERSITY

BHUBANESWAR – ODISHA – 752054

BONAFIDE CERTIFICATE

Certified that this project report “**Plant Disease Detection Using AlexNet Model**” is the bonafied work of “**Saumyakant Baral**” , “ carried out the project work under my supervision.

SIGNATURE
Mrs. Tapaswini Pattanaik
Asst. professor
Department of ECE

SIGNATURE
Supervisor

C.V. RAMAN GLOBAL UNIVERSITY
BHUBANESWAR – ODISHA – 752054

CERTIFICATE OF APPROVAL

This is to certify that we have examined the project entitled "**Plant Disease Detection Using AlexNet Model**" submitted by **Saumyakant Baral** . . . . C.V. Raman Global University, Bhubaneswar Odisha. We here by accord our approval of it as a major project carried out and presented in a manner required for its acceptance for the partial fulfillment for the **Bachelor Degree of Technology in Electronics and Communication Engineering** for which it has been submitted. This approval does not necessarily endorse or accept every statement made, opinion expressed or conclusions drawn as recorded in this major project, it only signifies the acceptance of the major project for the purpose it has been submitted.

Project Guide **External Examiner**

ACKNOWLEDGEMENT

I take this opportunity to express my deep sense of gratitude to my thesis supervisor, "**Mrs. Tapaswini pattanaik, Asst. Professor , Department of ECE**" for constant guidance and insightful comments during the course of work. I shall always cherish my association with him for his constant encouragement and freedom to thought and action that rendered to me throughout the work.

I also wish to express my sincere thanks to **Dr. Tusar Kanti Dash , Head of the Department** and all the faculty members of Electronics and Telecommunication Engineering Department for their cooperation and valuable suggestions in each stage of the thesis, which lead to successful completion of my work.

Finally, I deem it a great pleasure to thank one and all who helped me directly or indirectly in carrying out this thesis report.

TABLE OF CONTENTS

ABSTRACT

In the face of emerging threats to global food security, the imperative for effective plantdisease detection has reached unprecedented importance in sustaining agriculturaloperations. This research proposes a novel approach employing the AlexNet deep learning algorithm for the automated diagnosis of plant diseases based on leaf images. The inherent capability of AlexNet, rooted in its convolutional neural network architecture, facilitates precise differentiation between healthy and diseased plants. A diverse and comprehensive dataset encompassing various plant diseases ensures the model's adaptability to diverse agricultural settings.

The primary objective of this study is to assess the efficiency of AlexNet in comparisonto alternative approaches, utilizing key performance parameters such as accuracy, precision, recall, and F1 ratings. The results substantiate the superior accuracy of AlexNet over other models, affirming its efficacy in plant disease diagnosis. Moreover,this research extends its inquiry to explore the model's versatility across alternative plant species and disease types, underscoring its practical applicability in varied agricultural contexts.

Beyond contributing to the discourse on leveraging deep learning for plant disease diagnosis, this paper illuminates the utility of the AlexNet model in fostering smart agricultural practices and cultivation. The findings presented herein serve as a foundational step towards enhanced efforts in automated plant disease diagnosis, with the overarching goal of promoting sustainable agriculture and bolstering global food security.

The propose of this work is to present an enhanced approach for dictating leaf disease. The suggested system is built with Alex net and trained and tasted on a variety of tomatoleaf diseases. This model achieves 94.9% accuracy for classification and validation. in future model is implemented by increasing number of diseased classes as well as other plant diseases.

LIST OF FIGURES

LIST OF ABBREVIATIONS

Sl. No.	Abbreviations	Full Form
1.	AI	Artificial Intelligence
2.	CNN	Convolution Neural Network
3.	DL	Deep Learning
4.	ML	Machine Learning
5.	MAP	Mean Average Precision
6.	GPU	Graphical Processing Unit
7.	TPU	Tensor Processing Unit
8.	SVM	Support Vector Machines
9.	ReLU	Rectified Linear Unit
10.	RGB IMG	Red Gray Blue Image
11.	LRN	Local Response Normalization

CHAPTER – 1

1.1 INTRODUCTION

He world’s expanding population creates an existential problem for agriculture as its mainstay in the global supply of food. This leads to another major challenge that needs addressing which is the need to develop healthy and thriving plants because plant diseases could greatly affect production in agriculture. These diseases need proper and early detection because they determine crop yield, quality, and ultimately food security. It is worth noting that plant diseases pose a serious problem to a broader society as they contribute to massive economic losses; cause damage to the environment in general and to a particular area; disrupt national food supply chains and exacerbate global issues such as hunger and malnutrition. In traditional disease diagnosis methods, agronomists have to visually identify plants from different environments and then make a decision depending on this information. In a world of modern technology, it has become integral to implement state-of-the-art technologies such as deep learning algorithms. The use of deep neural networks like the AlexNet framework in detecting plant diseases could change how people approach plant diseases these days. Such models could come up with quick, precise, and expansive solutions revolutionizing farming activities.

The modern day demands more accurate means of detecting plant diseases hence, this paper is about using AlexNet deep learning methodology as a sophisticated detector. This research has been designed to enhance precision agriculture through the incorporation of some artificial intelligence concepts in a bid to mitigate the growing challenges of sustainable farming and global food security. every country needs farming to meet its requirements as well as to strengthen its economy. When crop plants are damaged by diseases, the country’s production and its economy are also affected. Because of data disparities, selecting an appropriate approach for image processing is always a difficult task. To produce good results, huge datasets necessitate advanced approaches such as CNN and large image datasets result in increased accuracy rates. Image processing is used to improve the quality of images to extract valuable information from them; as a result of this feature, image processing techniques are used in many areas of the medical and agricultural fields, such as colour processing, remote sensing, and pattern recognition.

Image processing techniques that are acceptable, effective, and dependable can be used to discover disease in plant leaves. Image processing can be used in a variety of fields, including biology, agriculture, medicine, engineering, computing. Computerized image processing techniques are critical for detecting and classifying plant diseases early before they cause widespread damage to entire crops. To address this, several DL, image processing, and ML techniques were being developed to detect and classify disease in

plants using images of plant leaves. DL technologies can help agricultural firms succeed. This research focuses on the comparative study of the performances, evaluation metrics, and results of numerous methodologies and methods previously used to detect and classify different forms of plant leaf diseases using image processing approaches. Accordingly, finding a reliable technique to apply is critical to increasing the yields of agricultural products.

1.2 Plant disease classification And Identification

Computer vision is a subdomain of AI that allows machines to counterfeit the human visual system and precisely draw out, inspect, and recognize real-world images in the same way that humans do. ML techniques have been used to detect and classify plant diseases, but with advancements in a subset of ML, DL, this area of research appears to have considerable potential in terms of increasing accuracy. Many developed DL architectures were used, along with various visualization techniques, to detect and classify plant disease symptoms accordingly. Medical diagnosis, espionage, satellite images, and agribusiness are just a few of the rapidly increasing industries that have already shown the benefits of computer vision-based technologies. Computer vision enabled systems can be used in agriculture to detect and classify plant diseases based on different features or symptoms that have been extracted. It uses a well-defined series of steps beginning with image acquisition and continuing with various image processing tasks such as scaling, filtering, segmentation, feature extraction, and selection, and finally, detection and classification are performed using ML or DL techniques.

1.3 factors responsible for plant disease

A wide range of agricultural diseases can arise at various stages of plant development and harm the plant's growth, which can have a negative impact on overall crop production. Plant diseases are caused by a variety of conditions at various phases of plant development. As summarized in, crop disease-causing variables are categorized into two: biotic factors and abiotic factors. Biotic factors such as viruses, fungi, bacteria, mites, and slugs emerge as a result of microbial infection in plants, whereas abiotic variables such as water, temperature, irradiation, and nutritional deprivation damage plant growth. Accordingly, some sample plant leaf images with different diseases from the Plant Village dataset and different images from other datasets showing healthy and diseased plant leaves have been included in the study and different images from other datasets showing healthy and diseased plant leaves have been summarized in the works of and accordingly.
Additionally, the detail computer vision-based techniques and processes including field crops, image acquisition, leaf image datasets, image preprocessing (test set, training set, and validation sets), data splitting, and performance assessment methods) for plant disease detection and classification have been clearly indicated in the work of. details of the factors responsible for plant diseases has been depicted in Fig. 1. Additionally, some sample plant leaf images with different diseases from the Plant Village dataset and different images from other datasets showing healthy and diseased plant leaves have been depicted in Fig. 2

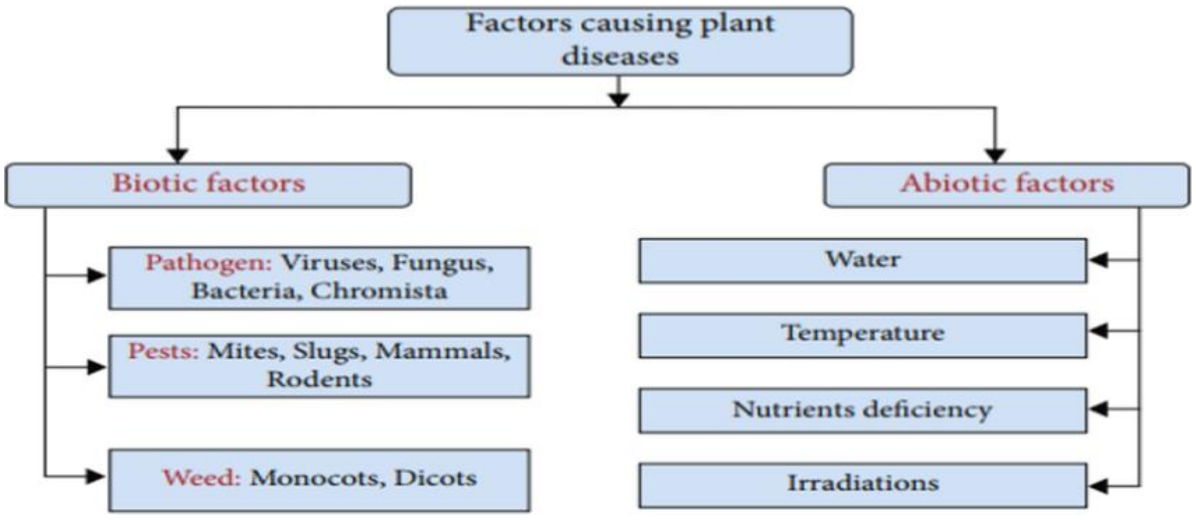

Factors causing plant diseases
Biotic factors
Abiotic factors
Pathogen: Viruses, Fungus, Bacteria, Chromista
Pests: Mites, Slugs, Mammals, Rodents
Weed: Monocots, Dicots
Water
Temperature
Nutrients deficiency
Irradiations

Fig. 1 – factor responsible for plant diseases

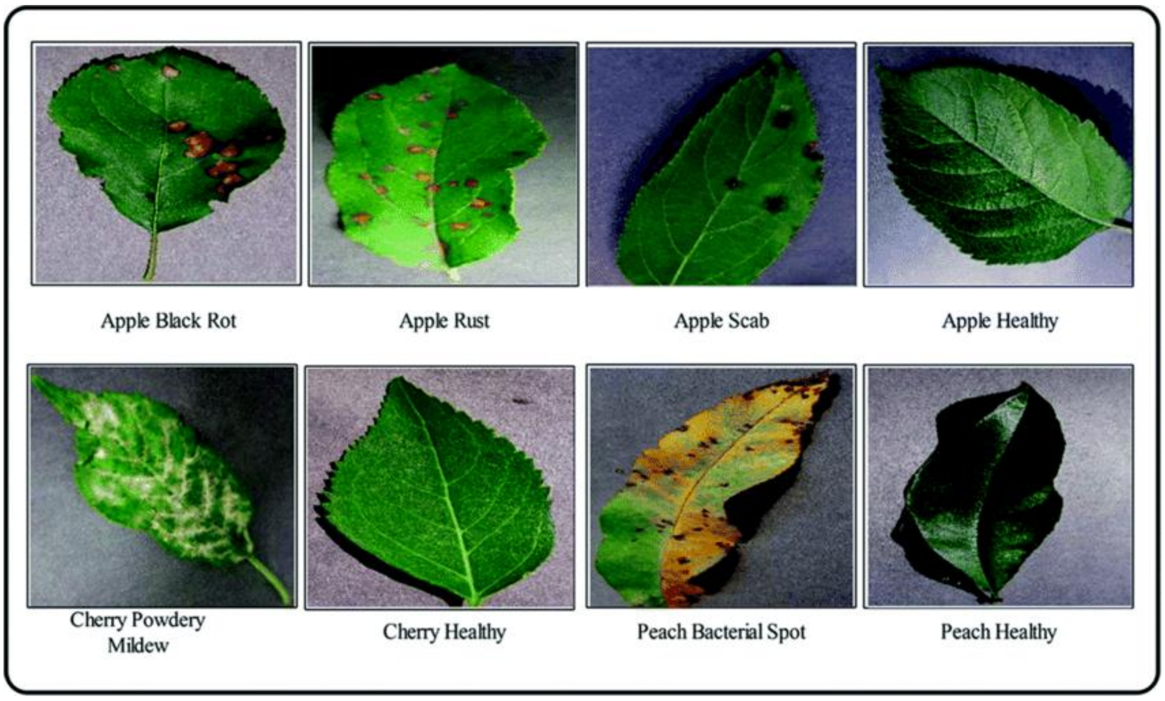


Fig. 2 - Some sample plant leaf images with different diseases from the kaggle dataset

1.4 DL and CNNs as powerful tools for image classification tasks

Representation Learning-DL, particularly CNNs, learns hierarchical representations from data, automatically capturing features at various levels of abstraction, making them highly effective for image classification. End-to-End Learning- CNNs integrate feature extraction and classification into a single model, optimizing performance jointly. This approach, unlike traditional methods, eliminates the need for separate feature extraction, leading to superior performance. Hierarchical Feature Extraction: CNNs mimic the hierarchical structure of the human visual system, extracting features from local to global levels. This hierarchical process allows CNNs to effectively capture both simple and complex features for accurate classification.

Parameter Sharing and Translation Invariance- CNNs share weights across spatial locations, enabling them to recognize patterns regardless of their position in the image. This mechanism provides robustness to translations, rotations, and other spatial transformations commonly encountered in images.

Scalability and Parallelization: DL frameworks, including CNNs, are scalable and can be parallelized, allowing efficient training on large datasets using GPUs and TPUs. This scalability is crucial for achieving state-of-the-art performance in image classification tasks.

Transfer Learning and Fine-Tuning: CNNs pre-trained on large-scale datasets can be fine-tuned for specific tasks, leveraging learned features and adapting them to new domains. This transfer learning approach enhances generalization and accelerates convergence, particularly useful with limited labelled data.

1.5 Mention of AlexNet As a pioneering CNN architecture.

Breakthrough in ImageNet Challenge-Alex Net's victory in the 2012 ImageNet Large Scale Visual Recognition Challenge showcased the effectiveness of deep learning for image classification, surpassing traditional computer vision methods by a significant margin. Architecture Innovation- AlexNet introduced a pioneering architecture comprising five convolutional layers followed by three fully connected layers, incorporating ReLU activation functions, local response normalization, dropout regularization, and overlapping max-pooling layers. This innovative design enabled effective learning of hierarchical features from raw images. Scalability and Parallelization-AlexNet was optimized to harness the computational power of GPUs, facilitating efficient training on large-scale datasets through parallelization. This scalability was instrumental in its success and paved the way for training deeper neural networks in subsequent years.

1.6 Summary

Alex Net's victory in the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) marked a watershed moment in the field of deep learning and computer vision. Its innovative architecture, comprising convolutional layers followed by fully connected layers with ReLU activation, local response normalization, dropout regularization, and overlapping max-pooling layers, demonstrated the effectiveness of deep learning for image classification tasks.

Alex Net's scalability, designed to leverage GPU parallelization, paved the way for training deeper networks on large-scale datasets efficiently. This breakthrough not only inspired further research into complex neural network architectures but also catalysed the widespread adoption of deep learning in academia and industry.

Ultimately, Alex Net's success highlighted the transformative potential of deep learning algorithms, leading to rapid advancements in computer vision research and solidifying deep learning as the dominant approach in various fields of artificial intelligence, including image recognition, natural language processing, and robotics.

CHAPTER – 2

LITERATURE REVIEW

2.1 Introduction

Plant diseases pose a significant threat to global food security by jeopardizing crop yield, quality, and sustainability. Timely and accurate detection of these diseases is crucial for effective disease management and mitigation of agricultural losses. Traditional methods of disease identification often rely on manual observation by trained experts, which can be time-consuming, labour-intensive, and subjective. In recent years, the emergence of deep learning techniques, particularly convolutional neural networks (CNNs), has revolutionized the field of computer vision and enabled automated image-based disease detection in plants. Among various CNN architectures, AlexNet stands out as a pioneering model that has demonstrated exceptional performance in image classification tasks. Its ability to learn intricate patterns and features from large-scale datasets, exemplified by its victory in the ImageNet Large Scale Visual Recognition Challenge in 2012, makes it a promising candidate for plant disease detection. This literature review aims to explore the utilization of the AlexNet model in the context of plant disease detection, examining existing studies, methodologies, challenges, and future directions in this rapidly evolving field. Through a comprehensive analysis of the current state-of-the-art, this review seeks to shed light on the potential of AlexNet-based approaches to revolutionize agricultural practices, enhance crop management strategies, and contribute to global efforts towards food security and sustainability.

2.2 Related Work

[1] presented various machine learning and deep learning techniques to identify plant disease detection by extracting features from the images, in machine learning. They have proposed classifiers such as Naïve Bayes, Decision Trees, K-Nearest Neighbour, Support Vector Machines (SVM), Random Forests and Multilayer Perceptron etc. and in deep learning, they have used Inception V4, VGG-16 and VGG-19. In Machine Learning, the RF classifier gained a precision of 78%, F1 score of 76.7%, CA (Classification Accuracy) of 76.8% and AUC (Area Under the Receiver Operating Characteristic Curve) of 92.8%. SGD gained a precision of 86.9%, F1 of 86.5%, CA of 86.5% and AUC of 91.2%. The SVM Classifier gets a precision of 87.3%, F1 of 87.1%, CA of 87% and AUC of 96.5%. In deep learning, VGG-19 gained a precision of87.7%, F1 of 87.4%, CA of 87.4% and AUC of 96.2%. Inception-V3 gained a precision of 89.2%, F1 of 89%, CA of 89% and AUC of 97% and in VGG-16, a precision of 89.6%, F1 of 89.5%, CA of 89.5% and AUC of 97% among which VGG-16 surpassed them. I InceptionV4, at Epoch 30, the training accuracy will be 99.9% and the validation accuracy will be 98% and in VGG-16, at Epoch 30, the training accuracy will be 80.7% and the validation accuracy will be 80.4%.

[2] presented a recent research review on advance deep learning models-based plant disease detection. The practical implications of their research include several important points. Firstly, they use of ML and DL

techniques for plant disease detection can enhance the accuracy and efficiency of disease detection compared to traditional manual methods, which can be beneficial for farmers and plant disease specialists to detect diseases at an early stage, preventing further spread and reducing the risk of crop losses. Secondly, developing generalizable models that can work for different plant species and diseases can save time and effort for researchers and practitioners, making it easier to detect and classify plant diseases in various settings. Thirdly, the availability of publicly accessible datasets for training and evaluating ML and DL models for plant disease detection can help researchers and practitioners develop more accurate and robust models, enhancing the performance of disease detection systems. Fourthly, the use of ML and DL techniques in plant disease detection can potentially reduce the need for manual labour and the cost of plant disease detection, which can be particularly useful for farmers and small-scale agricultural operations who may not have access to expensive equipment or specialized expertise. Lastly, the techniques and methodologies used in plant disease detection can also inform research and development in other fields.

[3] proposed a deep residual convolutional neural network (ResNet34) for the detection and classification of plant leaf diseases. They used a Plant Village dataset consisting of 15,200 images, covering 14 different crops and divided them into 38 classes in which 80-20 rule is applied for dividing the dataset, resulting in 12160 trained images and 3040 testing images, achieved 99.40% accuracy on the classification of diseases and improved precision of 96.51% and also compared the performance with other techniques such as SVM, K-NN, Decision Tree and Logistic Regression among them ResNet34 surpassed them.

[4] introduced an image-processing technique known as image segmentation. They used soft computing techniques for the detection of plant leaf diseases, for input data disease they took samples of plant leaves like beans leaf with bacterial disease, lemon leaf with sunburn disease, banana leaf with early scorch disease, fungal disease in beans leaf and rose with bacterial disease. They used the Minimum Distance Criterion with K-Mean Clustering which achieved an accuracy of 86.54%, the proposed algorithm achieved an accuracy of 93.63%, and finally, the SVM classifier with the proposed algorithm achieved a higher accuracy of 95.71% as compared to others.

CHAPTER– 3

BACKGROUND

3.1 Explanation of CNN

Convolutional Neural Networks (CNNs) are a class of deep neural networks particularly well-suited for tasks involving images and spatial data. They have revolutionized the field of computer vision and have become the go-to architecture for various image related tasks. Here's an explanation of CNNs**:**

Basic Structure: CNNs are composed of multiple layers, including convolutional layers, pooling layers and fully connected layers. Convolutional layers are the core building blocks of CNNs. They consist of filters (also called kernels) that convolve over the input image to extract features. Each filter learns to detect specific patterns or features, such as edges, textures, or shapes. Pooling layers down sample the feature maps generated by the convolutional layers, reducing the spatial dimensions while preserving the most important information. Max-pooling is a common pooling operation, where the maximum value within a region is retained.

Feature Learning: CNNs learn hierarchical representations of the input data. The early layers typically learn low-level features like edges and textures, while deeper layers learn higher-level features and patterns. Through the process of convolution and pooling, CNNs automatically learn to detect relevant features from raw data, eliminating the need for manual feature engineering.

Parameter Sharing: One key aspect of CNNs is parameter sharing. In convolutional layers, the same set of weights (filter) is applied across different spatial locations of the input image. This parameter sharing helps the model generalize well to new data and reduces the number of parameters, making CNNs more efficient.

Translation Invariance: CNNs exhibit translation invariance, meaning they can recognize patterns regardless of their position in the input image. This property is achieved through the use of convolutional filters that slide over the entire input image, enabling the model to detect features irrespective of their location.

Training: CNNs are typically trained using backpropagation and gradient descent optimization algorithms. During training, the model learns to minimize a loss function by adjusting its weights and biases to make accurate predictions. Large-scale labelled datasets are essential for training CNNs effectively. Common datasets used for training CNNs include ImageNet, CIFAR-10, and MNIST.

Applications: CNNs have a wide range of applications in computer vision, including image classification, object detection, segmentation, and image generation. They are used in various domains such as autonomous vehicles, medical imaging, satellite imagery analysis, and facial recognition systems. Convolutional Neural Networks (CNNs) are powerful deep learning models designed to process and analyse spatial data, particularly images. They learn hierarchical representations of features directly from raw data, enabling them to excel at a wide range of computer vision tasks.

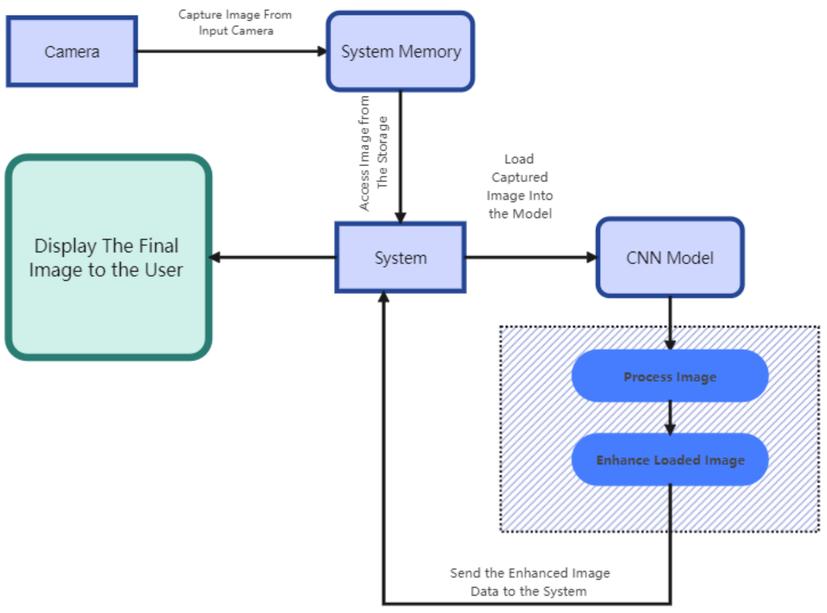

Camera
Capture Image From Input Camera
System Memory
Access Image from The Storage
Load Captured Image Into the Model
Display The Final Image to the User
System
CNN Model
Process Image
Enhance Loaded Image
Send the Enhanced Image Data to the System

Fig. 3- Block Diagram of CNN

3.2 How does CNN works ?

Convolutional Neural Networks are a class of deep learning neural networks, primarily used in image recognition and classification tasks. Here's a simplified overview of how they work:

Convolutional Layer: This is the core building block of CNNs. It applies convolution operation to the input data, typically an image. Convolution involves sliding a filter (also called a kernel) over the input and performing element-wise multiplication followed by summation. This process extracts features from the input data.

Activation Function: After the convolution operation, an activation function like ReLU (Rectified Linear Unit) is applied element-wise to the output of the convolutional layer. This introduces non-linearity into the network, enabling it to learn complex patterns in the data.

Pooling Layer: Pooling layers are used to reduce the spatial dimensions of the convolutional layer's output while retaining important information. Max pooling is a common pooling technique where the maximum value in each patch of the feature map is retained, effectively down sampling the input.

Fully Connected Layer: After several convolutional and pooling layers, the high-level reasoning in the neural network is done via fully connected layers. These layers take the features extracted by the convolutional layers and use them to classify the input into various classes.

Output Layer: The output layer produces the final classification or prediction. The activation function used in this layer depends on the task at hand. For example, for binary classification, a sigmoid function might be used, while for multi-class classification, a SoftMax function is common. During training, CNNs learn the filters or kernels that best extract features from the input data through a process called backpropagation, where the network adjusts its weights based on the error between the predicted output and the actual output.

CNNs have revolutionized many fields, especially computer vision, by achieving state of-the-art performance in tasks like image classification, object detection, and segmentation. Their ability to automatically learn features from raw data makes them powerful tools for various applications.

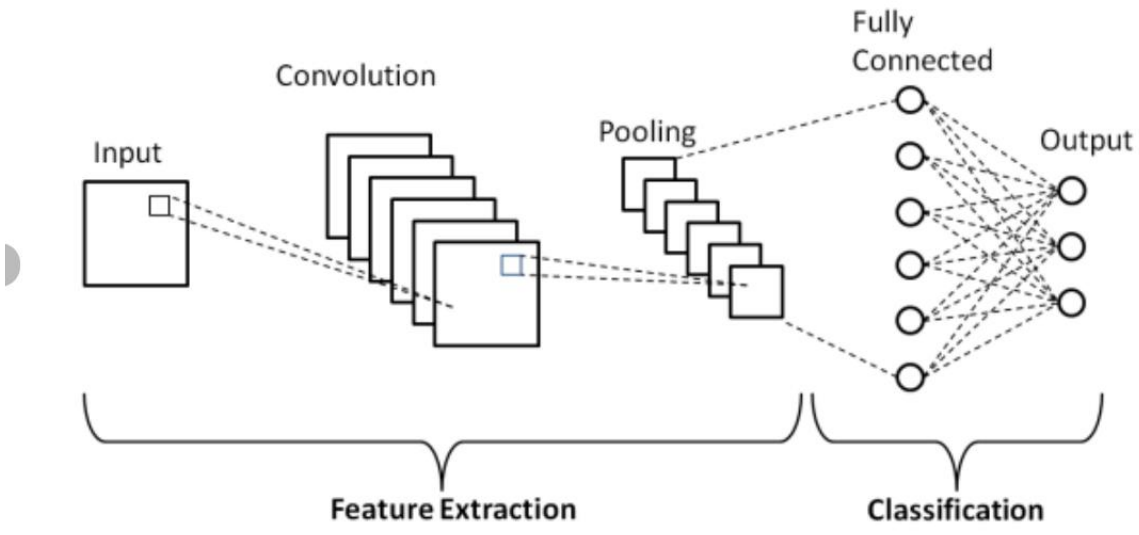


Fig. 4 – Semantic Diagram of CNN

3.3 Features of Transfer Learning

Transfer learning is a machine learning technique where a model trained on one task is repurposed or fine-tuned for another related task. Instead of starting the learning process from scratch, transfer learning leverages knowledge gained from solving one problem and applies it to a different but related problem. This approach is particularly useful when you have limited labelled data for the task at hand.

Pre-trained Model: Transfer learning typically starts with a pre-trained model that has been trained on a large dataset for a related task. For example, a model trained on a large dataset for image classification tasks, like ImageNet, can be used as a starting point for tasks like object detection or image segmentation.

Feature Extraction: In transfer learning, the early layers of the pre-trained model, which learn basic features like edges and textures, are often kept fixed (frozen), as these features are generally applicable to many tasks. The later layers, which learn higher level features specific to the original task, are often modified or removed.

Fine-tuning: After extracting features from the pre-trained model, additional layers are added or existing layers are modified to adapt the model to the new task. These added layers are typically trained using the new dataset, while the weights of the pretrained layers may be fine-tuned along with them. Fine-tuning allows the model to learn task- specific features while still benefiting from the knowledge gained during pretraining.

3.4 Transfer Learning Scenarios:

Domain Adaptation: When the source and target domains are different but related, transfer learning can help adapt the model's knowledge from the source domain to the target domain. For example, transferring knowledge from a model trained on photos to a model for satellite imagery.

Task Adaptation: When the tasks are different but share some underlying structure or features, transfer learning can be used to transfer knowledge between tasks. For example, using a model trained for sentiment analysis on product reviews to perform sentiment analysis on social media comments.

3.5 Benefits:

Reduced Training Time: Transfer learning can significantly reduce the time and computational resources required to train a model from scratch, especially when working with limited data. Improved Performance: By leveraging knowledge from pre-trained models, transfer learning often leads to better performance, especially in scenarios where labelled data is scarce. Overall, transfer learning is a powerful technique that allows models to beverage knowledge gained from solving one task to improve performance on related tasks, making it a valuable tool in machine learning practice

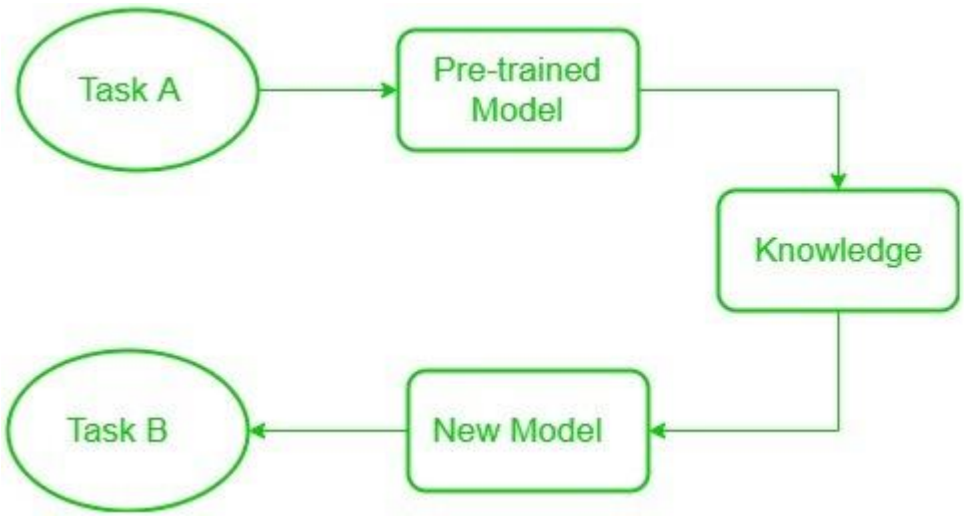


Fig. 5- Block Diagram of Transfer Learning

CHAPTER – 4

ALEXNET MODEL

4.1 Introduction

Plant diseases can significantly impact crop yield, quality, and overall agricultural productivity. Traditional methods of disease detection often rely on manual inspection, which is labour-intensive, time-consuming, and may not always be accurate. The integration of deep learning techniques, particularly convolutional neural networks (CNNs), has emerged as a promising approach to automate the detection of plant diseases from images. Among various CNN architectures, the AlexNet model has garnered significant attention and success since its introduction in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. AlexNet demonstrated remarkable performance in image classification tasks, significantly reducing error rates and paving.

4.2 Architecture of ALEX Net

AlexNet is a convolutional neural network architecture that played a pivotal role in popularizing deep learning, particularly convolutional neural networks (CNNs), in computer vision tasks. It was developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton and won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 by a significant margin.

Input Layer: The network takes an input image of size 227x227x3 (RGB images).

Convolutional Layers: AlexNet consists of five convolutional layers, each followed by a max-pooling layer. The first convolutional layer has 96 kernels of size 11x11x3 with a stride of 4.The second convolutional layer has 256 kernels of size 5x5x48.The third convolutional layer has 384 kernels of size 3x3x256.The fourth and fifth convolutional layers have 384 and 256 kernels of size 3x3x192 and 3x3x192, respectively. All convolutional layers use the Rectified Linear Unit (ReLU) activation function.

Max Pooling Layers: After each convolutional layer, there is a max-pooling layer to down sample the feature maps. Max pooling is applied with a stride of 2.

Fully Connected Layers: AlexNet has three fully connected layers, also known as dense layers. The first two fully connected layers consist of 4096 neurons each, followed by ReLU activation and dropout to reduce overfitting. The third fully connected layer (output layer) consists of 1000 neurons, corresponding to the 1000 classes in the ImageNet dataset at the time. It uses SoftMax activation for classification.

Dropout: Dropout is applied to the first two fully connected layers with a dropout rate of 0.5 to prevent overfitting during training.

Normalization: Local Response Normalization (LRN) is applied after the first and second convolutional layers to normalize the responses and improve generalization.

Output Layer: The output layer predicts the probabilities of each class using SoftMax activation. AlexNet significantly advanced the field of computer vision and deep learning by demonstrating the effectiveness of deep CNNs for image classification tasks. Its success paved the way for deeper and more complex neural network architectures, leading to the development of modern deep learning models.

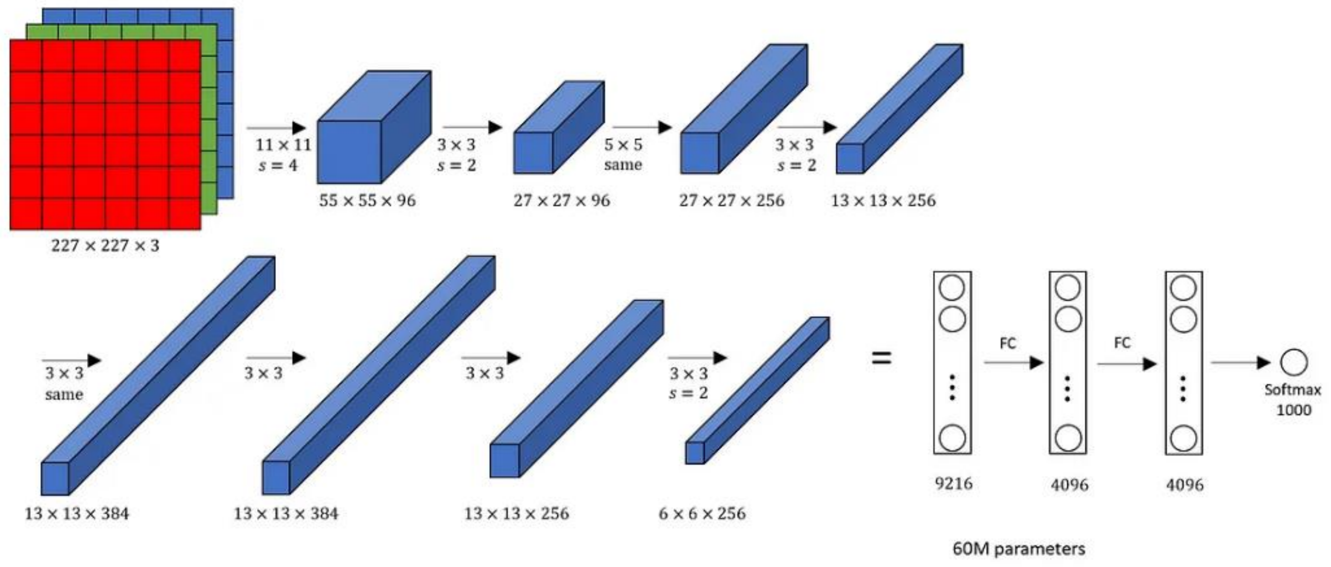


Fig. 6 - Architecture Of Alex Net

4.3 Activation Function

AlexNet primarily uses the Rectified Linear Unit (ReLU) activation function after each convolutional and fully connected layer. ReLU introduces non-linearity to the model by replacing negative values with zero, allowing the network to learn complex patterns and improving training speed. ReLU activation has several advantages, including faster convergence during training and reduced likelihood of the vanishing gradient problem.

ReLU Activation: Rectified Linear Unit (ReLU) activation functions are used after each convolutional and fully connected layer. ReLU introduces non-linearity, enabling the network to model more complex relationships in the data.

Local Response Normalization: Local Response Normalization (LRN) is implemented to enhance generalization by normalizing the output of a neuron relative to its neighbours. This creates a form of lateral inhibition, making the network more robust to variations in input data.

SoftMax Activation: The final layer of AlexNet uses the SoftMax activation function to convert the model's raw output into class probabilities. This allows the network to provide a probability distribution over the possible classes for each input image.

4.4 Parameters of Alex Net

Convolutional Layers: Number of Kernels: Each convolutional layer in AlexNet consists of multiple kernels (also called filters). The number of kernels varies across layers, starting from 96 in the first layer and increasing to 256, 384, 384, and finally 256 in subsequent layers. Kernel Size: The size of the kernels also varies across layers. For example, the first layer uses 11x11x3 kernels, while the subsequent layers use smaller kernel sizes such as 5x5x48, 3x3x256, etc.

Stride: The stride determines how much the kernel shifts during convolution. In AlexNet, the stride is 4 for the first convolutional layer and 1 for the rest.

Max Pooling Layers : Pooling Size: After each convolutional layer, max-pooling is applied with a pooling size of 3x3 and a stride of 2.

Fully Connected Layers : AlexNet has three fully connected layers: The first two layers have 4096 neurons each. The third layer (output layer) has 1000 neurons, corresponding to the number of classes in the ImageNet dataset at the time. Each fully connected layer has its set of weights, contributing to a significant number of parameters.

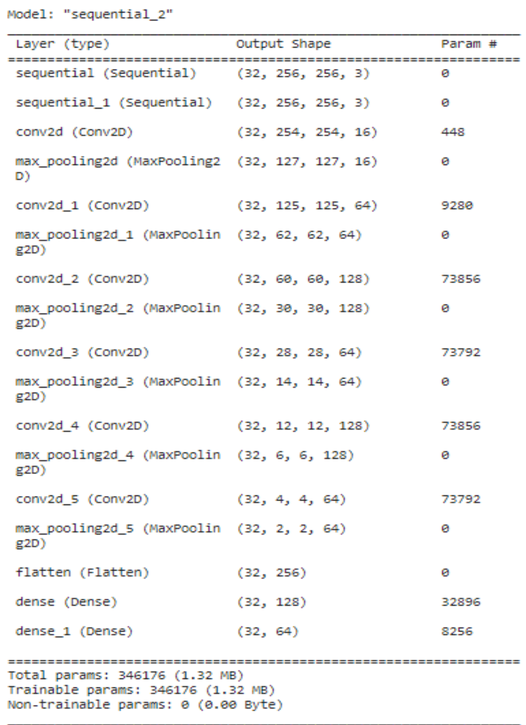
Dropout: Dropout is applied with a dropout rate of 0.5 to the first two fully connected layers. Dropout helps prevent overfitting by randomly dropping neurons during training.

Local Response Normalization (LRN): LRN is applied after the first and second convolutional layers to normalize the responses. It helps improve generalization by enhancing the contrast between different feature maps.

Total Parameters:

The total number of parameters in AlexNet can be calculated by summing the parameters from each layer:

Adding up the parameters from all layers gives the total number of parameters in AlexNet.



```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (32, 256, 256, 3)         0

 sequential_1 (Sequential)   (32, 256, 256, 3)         0

 conv2d (Conv2D)             (32, 254, 254, 16)        448

 max_pooling2d (MaxPooling2  (32, 127, 127, 16)        0
 D)

 conv2d_1 (Conv2D)           (32, 125, 125, 64)        9280

 max_pooling2d_1 (MaxPoolin  (32, 62, 62, 64)          0
 g2D)

 conv2d_2 (Conv2D)           (32, 60, 60, 128)         73856

 max_pooling2d_2 (MaxPoolin  (32, 30, 30, 128)         0
 g2D)

 conv2d_3 (Conv2D)           (32, 28, 28, 64)          73792

 max_pooling2d_3 (MaxPoolin  (32, 14, 14, 64)          0
 g2D)

 conv2d_4 (Conv2D)           (32, 12, 12, 128)         73856

 max_pooling2d_4 (MaxPoolin  (32, 6, 6, 128)           0
 g2D)

 conv2d_5 (Conv2D)           (32, 4, 4, 64)            73792

 max_pooling2d_5 (MaxPoolin  (32, 2, 2, 64)            0
 g2D)

 flatten (Flatten)           (32, 256)                 0

 dense (Dense)               (32, 128)                 32896

 dense_1 (Dense)             (32, 64)                  8256

=================================================================
Total params: 346176 (1.32 MB)
Trainable params: 346176 (1.32 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig. 7 - Parameters of AlexNet

4.5 Significance

Breakthrough in Deep Learning: AlexNet demonstrated the power of deep convolutional neural networks (CNNs) for image classification tasks. By significantly reducing the error rates on the ImageNet dataset, it sparked widespread interest and investment in deep learning research.

Architecture Innovation: AlexNet introduced several architectural innovations, such as using multiple convolutional layers and ReLU activation functions, which proved crucial for achieving superior performance.

GPU Acceleration: Training AlexNet required substantial computational resources. Its success catalysed the adoption of Graphics Processing Units (GPUs) for deep learning tasks due to their ability to parallelize computations, leading to significant speedups in model training.

Impact on Computer Vision: The breakthrough performance of AlexNet on the ImageNet dataset marked a significant milestone in computer vision research. It inspired subsequent developments in image classification, object detection, segmentation, and other computer vision tasks, leading to the development of more advanced architectures like VGG, GoogleNet, and ResNet.

Overall, AlexNet revolutionized the field of deep learning and computer vision, paving the way for numerous advancements and applications in artificial intelligence. Its architectural principles and training methodologies continue to influence state-of-the-art deep learning models today.

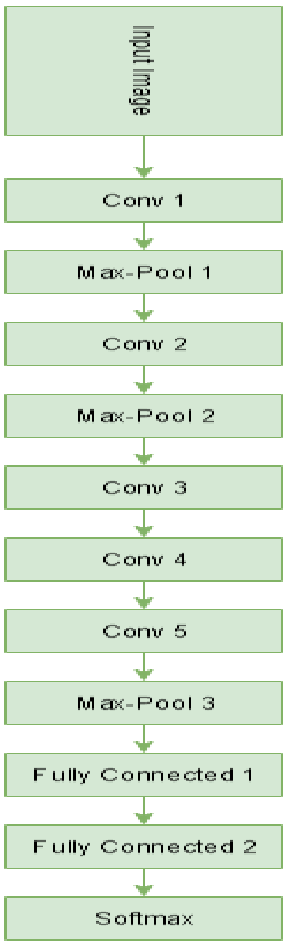


Fig. 8 - Layers Of AlexNet

CHAPTER – 5

SIMULATION AND DATASET

5.1 Data Collection and Preprocessing

Description about Dataset

This dataset is created using offline augmentation from the original dataset. This dataset consists of about 87K RGB images of healthy and diseased crop leaves which is categorized into 38 different classes. A new directory containing 33 test images is created later for prediction purpose.

Data set link - *https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset/data*

Modelling The Dataset

It is advisable to use GPU instead of CPU when dealing with images dataset because CPUs are generalized for general purpose and GPUs are optimized for training deep learning models as they can process multiple computations simultaneously. They have a large number of cores, which allows for better computation of multiple parallel processes. Additionally, computations in deep learning need to handle huge amounts of data, this makes a GPU's memory bandwidth most suitable.

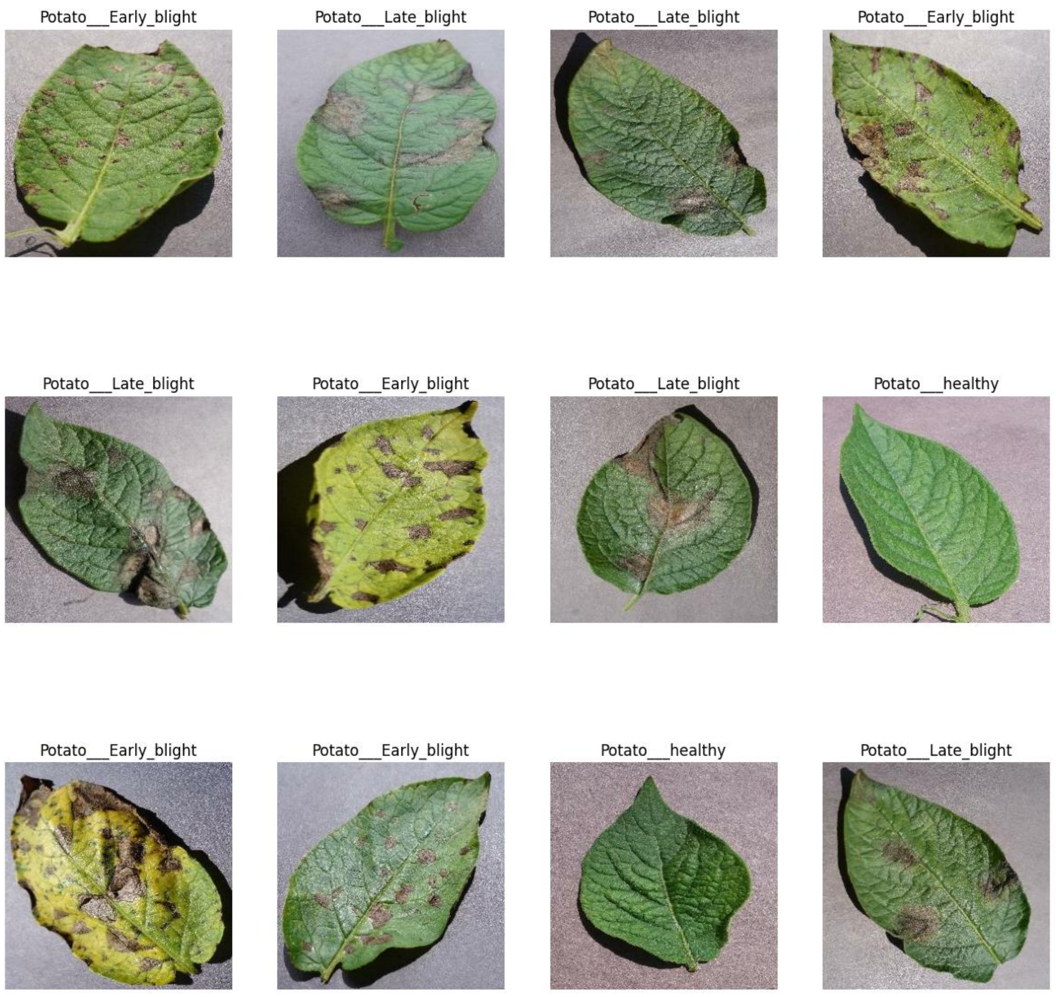


Fig. 9- Samples of Dataset

Pre-processing the data set

Preprocessing a dataset for plant disease detection involves several steps tailored to image data and the specific requirements of the detection task. Here's a detailed outline of preprocessing steps for plant disease detection datasets:

1. Data Cleaning: Handle Missing Values: Check for missing values in metadata and labels, if applicable, and decide on an appropriate strategy for handling them. Remove Duplicates: Identify and remove duplicate samples to prevent bias in the dataset.

2. Image Preprocessing: Image Resizing: Resize images to a uniform size suitable for model input. Common sizes for deep learning models are 224x224 or 256x256 pixels. Image Normalization: Normalize pixel values to a common scale (e.g., [0, 1] or [-1, 1]) to improve model convergence and performance. You can divide pixel values by 255 for normalization. Image Augmentation: Apply transformations such as rotation, flipping, cropping, and zooming to augment the dataset. Augmentation increases dataset diversity and helps improve model generalization. Libraries like OpenCV or TensorFlow's Image Data Generator can be used for augmentation.

3. Label Encoding: Encode categorical labels (e.g., disease classes) into numerical representations suitable for machine learning models. This may involve one-hot encoding or label encoding.

4. Train-Test Split: Split the dataset into training and test sets to evaluate model performance. A common split is 80-20 or 70-30 for training and testing, respectively. Optionally, create a validation set for hyperparameter tuning if using techniques like cross-validation.

5. Data Balancing: If the dataset is imbalanced (i.e., certain classes are underrepresented), apply techniques such as oversampling, under sampling, or data augmentation to balance the class distribution. This helps prevent the model from being biased towards the majority class.

6. Data Standardization : Standardize numerical features to have a mean of 0 and a standard deviation of 1. While this step is not typically applied to image data directly, it may be relevant if additional numerical features are included in the dataset.

7. Feature Extraction : Extract meaningful features from the images using techniques such as edge detection, texture analysis, or feature descriptors. This can help reduce the dimensionality of the dataset and improve model performance, especially if working with limited computational resources.

8. Metadata Integration : If available, integrate metadata such as plant species, environmental conditions, and disease severity into the dataset. This additional information can enhance the model's understanding of the data and improve its predictive accuracy. By carefully preprocessing the dataset, you can ensure that it is well-prepared for training machine learning models for plant disease detection. The specific preprocessing steps may vary depending on the characteristics of the dataset and the requirements of the machine learning model being used.

5.2. Transfer learning Alex Net

Transfer learning with AlexNet involves leveraging the pre-trained weights of AlexNet, originally trained on a large dataset such as ImageNet, and fine-tuning it on a new dataset for a specific task, such as plant disease detection. Here's how you can perform transfer learning with AlexNet on a plant disease dataset:

Load Pre-trained AlexNet Model: Begin by loading the pre-trained AlexNet model along with its weights. You can use deep learning frameworks like TensorFlow or PyTorch to load the pre-trained model.

Replace the Output Layer: Since the original AlexNet model was trained for image classification with 1000 classes, you need to replace the output layer with a new output layer suitable for your plant disease detection task. The new output layer should have the number of neurons equal to the number of classes in your plant disease dataset.

Freeze Pre-trained Layers: Optionally, you can choose to freeze the weights of the pre- trained layers to prevent them from being updated during training. Freezing the pretrained layers can be beneficial when you have a small dataset to prevent overfitting and reduce training time.

Fine-tuning: Train the modified AlexNet model on your plant disease dataset. During training, backpropagation updates the weights of the new output layer and potentially the weights of the last few layers if you choose not to freeze them. The learning rate for fine-tuning is typically set to a smaller value compared to training from scratch.

Validation and Hyperparameter Tuning: Validate the performance of the fine-tuned model on a validation set and tune hyperparameters such as learning rate, batch size, and dropout rate if necessary. This step helps optimize the model's performance and generalization ability.

Evaluate on Test Set: Finally, evaluate the fine-tuned AlexNet model on a held-out test set to assess its performance in detecting plant diseases. Calculate metrics such as accuracy, precision, recall, and F1-score to quantify the model's effectiveness.

Transfer learning with AlexNet allows you to benefit from the feature extraction capabilities of the pre-trained model while adapting it to the specifics of your plant disease detection task. By fine-tuning the model on your dataset, you can achieve improved performance compared to training a model from scratch, especially when working with limited labelled data.

5.3 Adapt the fully connected layers in the plant disease dataset

Adapting the fully connected layers of AlexNet for a specific number of classes in a plant disease dataset involves replacing the original output layer of AlexNet with a new output layer that matches the number of classes in your dataset. Here's how you can adapt the fully connected layers:

Replace Output Layer: Remove the original output layer of AlexNet, which was designed for ImageNet's 1000 classes. Add a new fully connected layer with the number of neurons equal to the number of classes in your plant disease dataset. you can add a SoftMax activation function to the new output layer if you're performing multi-class classification. If you're performing binary classification, you can use a sigmoid activation function instead.

Fine-tune the Fully Connected Layers: During fine-tuning, only the weights of the new output layer and potentially the weights of the preceding fully connected layers are updated. You can choose to freeze the weights of the other layers (e.g., convolutional layers) to prevent them from being updated during training, especially if you have a small dataset to prevent overfitting.

The original output layer of AlexNet is replaced with a new fully connected layer with 10 neurons, corresponding to the number of classes in the plant disease dataset. Additionally, a SoftMax activation function is added to the new output layer to perform multi-class classification. Finally, the model is ready for fine-tuning on the plant disease dataset.

5.4. Training and process

Details of training process

The training process for fine-tuning AlexNet on a plant disease dataset involves several key components, including the choice of optimizer, learning rate scheduling, and batch size.

1. Optimizer Choice: The optimizer is responsible for updating the weights of the neural network during training based on the gradients computed with respect to the loss function. Common optimizers used for fine-tuning deep learning models include. Stochastic Gradient Descent (SGD): A basic optimizer that updates weights based on the average gradient of the loss function with respect to the parameters. Adam: An adaptive optimization algorithm that combines the advantages of SGD with momentum and adaptive learning rates. RMSprop: Another adaptive optimization algorithm that divides the learning rate by an exponentially decaying average of squared gradients. The choice of optimizer depends on factors such as convergence speed, stability, and performance on the specific dataset.

2. Learning Rate Scheduling: Learning rate scheduling involves adjusting the learning rate over time during training to improve convergence and performance. Common learning rate scheduling strategies include: Fixed Learning Rate: Keep the learning rate constant throughout training. Step Decay: Decrease the learning rate by a factor at specific epochs or after a certain number of iterations. Exponential Decay: Reduce the learning rate exponentially over time. Adaptive Learning Rate: Adjust the learning rate dynamically based on the model's performance, such as using learning rate schedules like ReduceLROnPlateau. Learning rate scheduling helps prevent the model from getting stuck in local minima and allows for more effective exploration of the loss landscape.

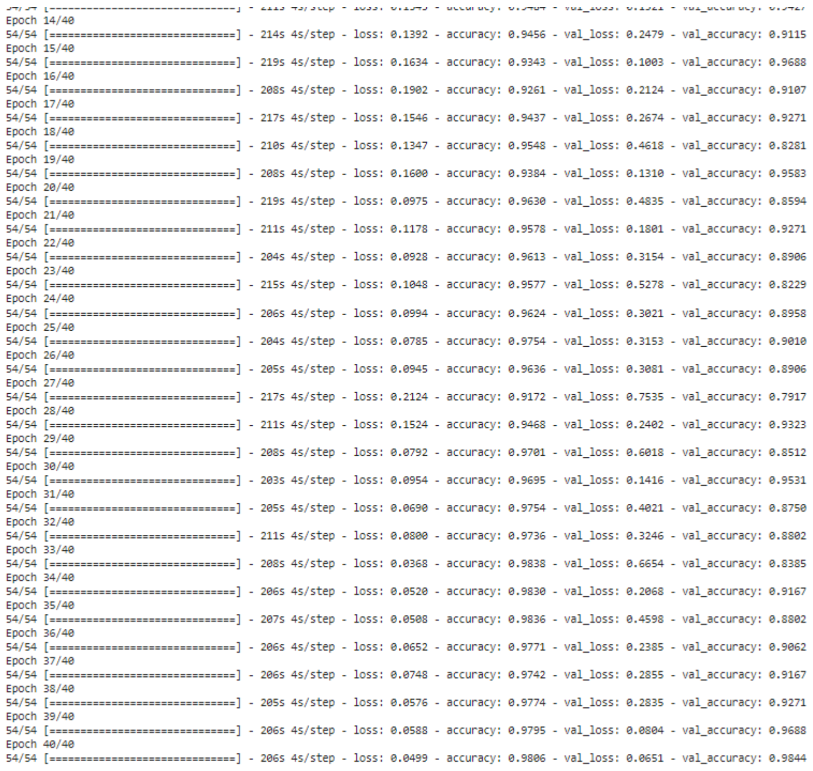
3. Batch Size: Batch size refers to the number of samples processed by the model in each iteration during training. Choosing an appropriate batch size depends on factors such as the size of the dataset, computational resources, and model architecture. Common batch sizes range from 16 to 128, with larger batch sizes generally leading to faster convergence but requiring more memory. It's important to balance the batch size with the available memory and computational resources to ensure efficient training.

5.5 Monitoring Training Dataset

Monitoring the training progress of a model during training involves tracking various metrics such as loss curves, accuracy, and validation performance. These metrics provide insights into how well the model is learning and generalizing from the training data. Here's how you can monitor these aspects during the training process:

1.Loss Curves: Loss curves show the trend of the loss function (e.g., cross-entropy loss) over epochs or iterations during training. Plotting loss curves helps visualize how the model's loss decreases over time, indicating improvement in learning. Both training loss and validation loss can be plotted to assess over fitting. A decreasing training loss with a stable or slightly increasing validation loss indicates good model performance.

2. Accuracy Metrics: Accuracy metrics measure the model's performance in terms of correct predictions compared to the ground truth labels. Calculate accuracy on both the training and validation sets to assess the model's performance during training and its ability to generalize to unseen data. Other metrics such as precision, recall, and F1-score can also provide additional insights into the model's performance, especially for imbalanced datasets or multi-class classification problems.



```
Epoch 14/40
54/54 [==============================] - 214s 4s/step - loss: 0.1392 - accuracy: 0.9456 - val_loss: 0.2479 - val_accuracy: 0.9115
Epoch 15/40
54/54 [==============================] - 219s 4s/step - loss: 0.1634 - accuracy: 0.9343 - val_loss: 0.1003 - val_accuracy: 0.9688
Epoch 16/40
54/54 [==============================] - 208s 4s/step - loss: 0.1902 - accuracy: 0.9261 - val_loss: 0.2124 - val_accuracy: 0.9107
Epoch 17/40
54/54 [==============================] - 217s 4s/step - loss: 0.1546 - accuracy: 0.9437 - val_loss: 0.2674 - val_accuracy: 0.9271
Epoch 18/40
54/54 [==============================] - 210s 4s/step - loss: 0.1347 - accuracy: 0.9548 - val_loss: 0.4618 - val_accuracy: 0.8281
Epoch 19/40
54/54 [==============================] - 208s 4s/step - loss: 0.1600 - accuracy: 0.9384 - val_loss: 0.1310 - val_accuracy: 0.9583
Epoch 20/40
54/54 [==============================] - 219s 4s/step - loss: 0.0975 - accuracy: 0.9630 - val_loss: 0.4835 - val_accuracy: 0.8594
Epoch 21/40
54/54 [==============================] - 211s 4s/step - loss: 0.1178 - accuracy: 0.9578 - val_loss: 0.1801 - val_accuracy: 0.9271
Epoch 22/40
54/54 [==============================] - 204s 4s/step - loss: 0.0928 - accuracy: 0.9613 - val_loss: 0.3154 - val_accuracy: 0.8906
Epoch 23/40
54/54 [==============================] - 215s 4s/step - loss: 0.1048 - accuracy: 0.9577 - val_loss: 0.5278 - val_accuracy: 0.8229
Epoch 24/40
54/54 [==============================] - 206s 4s/step - loss: 0.0994 - accuracy: 0.9624 - val_loss: 0.3021 - val_accuracy: 0.8958
Epoch 25/40
54/54 [==============================] - 204s 4s/step - loss: 0.0785 - accuracy: 0.9754 - val_loss: 0.3153 - val_accuracy: 0.9010
Epoch 26/40
54/54 [==============================] - 205s 4s/step - loss: 0.0945 - accuracy: 0.9636 - val_loss: 0.3081 - val_accuracy: 0.8906
Epoch 27/40
54/54 [==============================] - 217s 4s/step - loss: 0.2124 - accuracy: 0.9172 - val_loss: 0.7535 - val_accuracy: 0.7917
Epoch 28/40
54/54 [==============================] - 211s 4s/step - loss: 0.1524 - accuracy: 0.9468 - val_loss: 0.2402 - val_accuracy: 0.9323
Epoch 29/40
54/54 [==============================] - 208s 4s/step - loss: 0.0792 - accuracy: 0.9701 - val_loss: 0.6018 - val_accuracy: 0.8512
Epoch 30/40
54/54 [==============================] - 203s 4s/step - loss: 0.0954 - accuracy: 0.9695 - val_loss: 0.1416 - val_accuracy: 0.9531
Epoch 31/40
54/54 [==============================] - 205s 4s/step - loss: 0.0690 - accuracy: 0.9754 - val_loss: 0.4021 - val_accuracy: 0.8750
Epoch 32/40
54/54 [==============================] - 211s 4s/step - loss: 0.0800 - accuracy: 0.9736 - val_loss: 0.3246 - val_accuracy: 0.8802
Epoch 33/40
54/54 [==============================] - 208s 4s/step - loss: 0.0368 - accuracy: 0.9838 - val_loss: 0.6654 - val_accuracy: 0.8385
Epoch 34/40
54/54 [==============================] - 206s 4s/step - loss: 0.0520 - accuracy: 0.9830 - val_loss: 0.2068 - val_accuracy: 0.9167
Epoch 35/40
54/54 [==============================] - 207s 4s/step - loss: 0.0508 - accuracy: 0.9836 - val_loss: 0.4598 - val_accuracy: 0.8802
Epoch 36/40
54/54 [==============================] - 206s 4s/step - loss: 0.0652 - accuracy: 0.9771 - val_loss: 0.2385 - val_accuracy: 0.9062
Epoch 37/40
54/54 [==============================] - 206s 4s/step - loss: 0.0748 - accuracy: 0.9742 - val_loss: 0.2855 - val_accuracy: 0.9167
Epoch 38/40
54/54 [==============================] - 205s 4s/step - loss: 0.0576 - accuracy: 0.9774 - val_loss: 0.2835 - val_accuracy: 0.9271
Epoch 39/40
54/54 [==============================] - 206s 4s/step - loss: 0.0588 - accuracy: 0.9795 - val_loss: 0.0804 - val_accuracy: 0.9688
Epoch 40/40
54/54 [==============================] - 206s 4s/step - loss: 0.0499 - accuracy: 0.9806 - val_loss: 0.0651 - val_accuracy: 0.9844
```

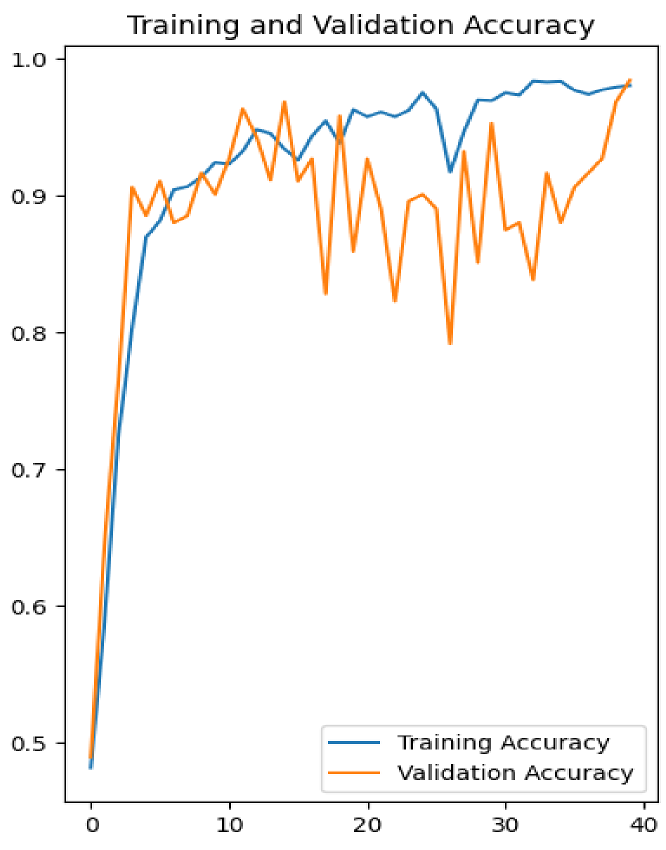
Fig.10- Loss, Accuracy, Validation Loss and Validation Accuracy of Trained Dataset

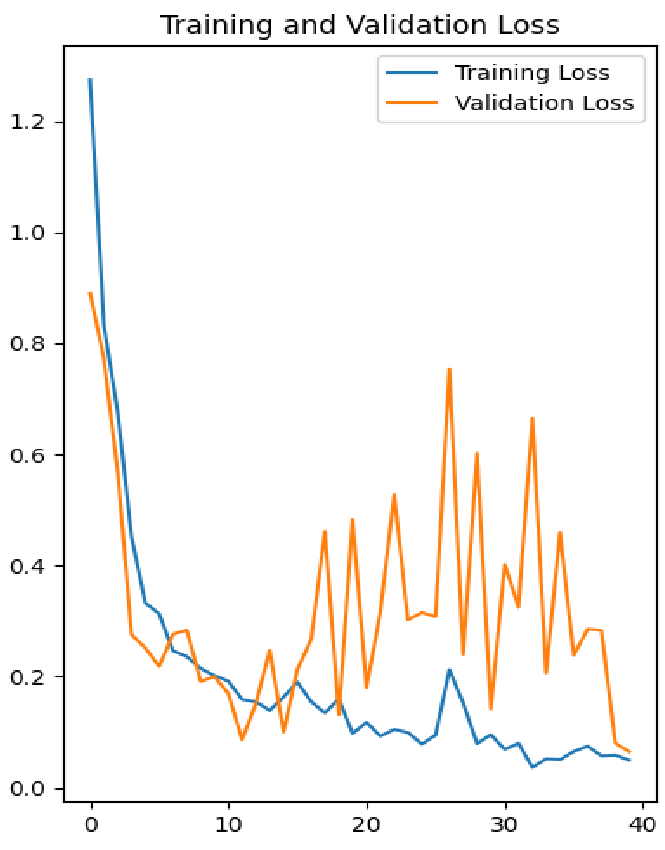
3. Validation Performance: During training, evaluate the model's performance on a validation set at regular intervals (e.g., after each epoch). Monitor metrics such as accuracy, loss, precision, recall, and F1score on the validation set to assess the model's generalization ability. Early stopping can be employed based on validation performance to prevent overfitting. Stop training if the validation performance starts deteriorating or stops improving.

CHAPTER – 6

6.1. RESULTS

1. AlexNet Model





Co-ordinates for Training and Validation Accuracy
X-axis = Epoch
Y-axis = Accuracy

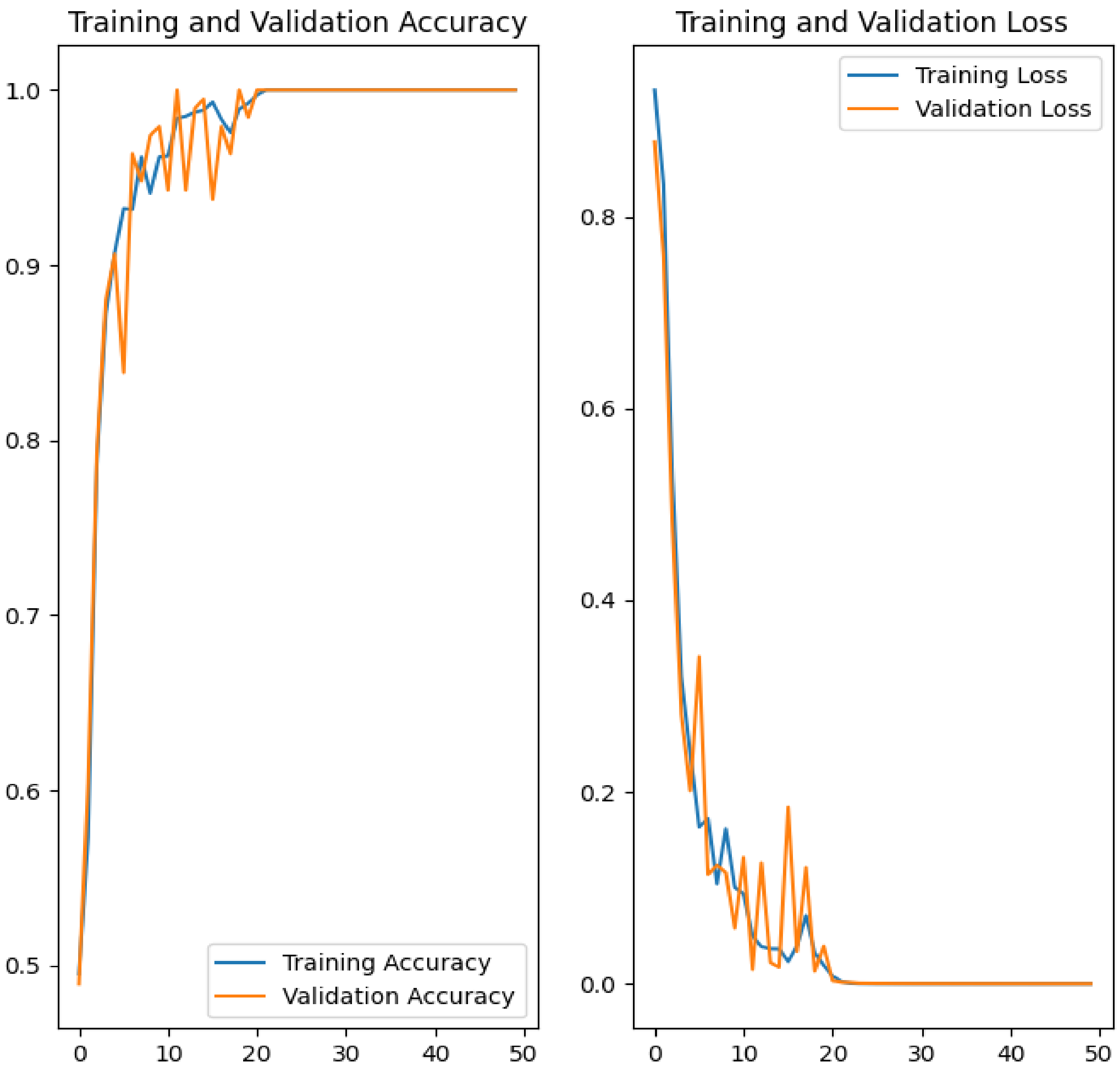
Co-ordinates for Training and Validation Loss
X-axis = Epoch
Y-axis = Loss

Fig.11 – Graph of Training and Validation Of Accuracy and Loss using Alexnet

The above curve indicates that the model is overfitting, and can't generalise on new data. In particular, the model performs well on training data, but poorly on the new data in the validation set. At a point, the validation loss decreases but starts to increase again.

A notable reason for this occurrence is that the model may be too complex for the data or that the model was trained for a long period. In this case, training can be halted when, the loss is low and stable, this is usually known as "early stopping". Early stopping is one of the many approaches used to prevent overfitting.

2. ResNet Model



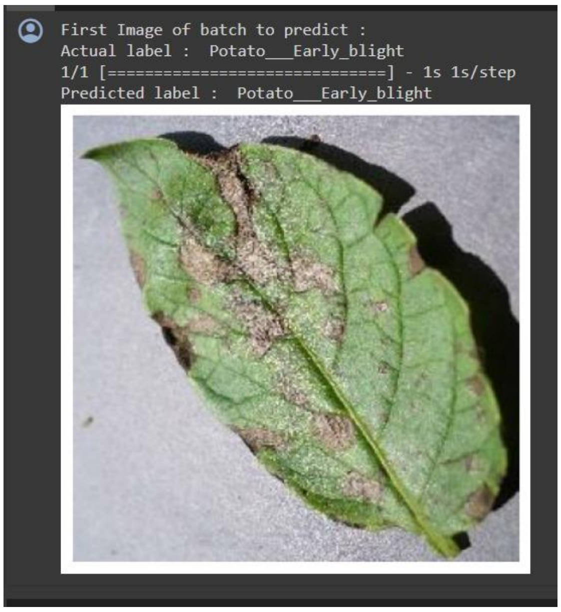
Co-ordinates for Training and Validation Accuracy
X-axis = Epoch
Y-axis = Accuracy

Co-ordinates for Training and Validation Loss
X-axis = Epoch
Y-axis = Loss

Fig.12 – Graph of Training and Validation Of Accuracy and Loss using ResNet

This indicates an optimal fit, i.e., a model that doesn’t overfit and underfit.

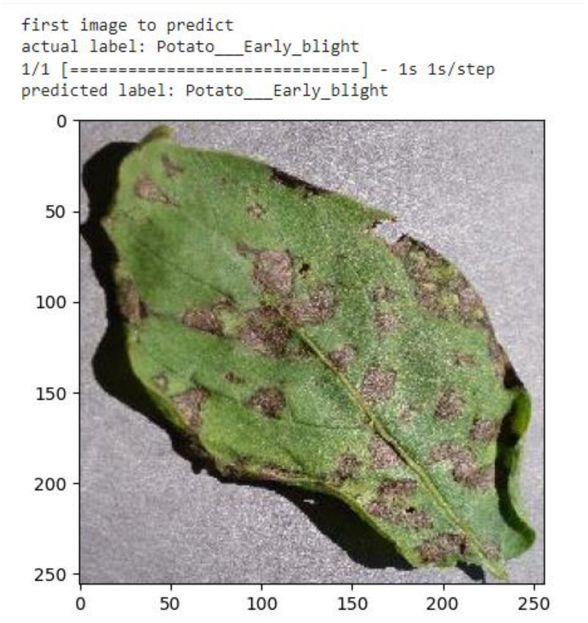
6.2. Understanding the result



```
First Image of batch to predict :
Actual label :  Potato___Early_blight
1/1 [==============================] - 1s 1s/step
Predicted label :  Potato___Early_blight
```

Fig. 13- Potato Early Blight using AlexNet Model (Actual = Predicted)

The total number of images in Potato Disease leaf dataset is about 2152; Hence, acquiring an accuracy of more than 96% and validation loss of 0.53% at 40 Epochs.



```
first image to predict
actual label: Potato___Early_blight
1/1 [==============================] - 1s 1s/step
predicted label: Potato___Early_blight
```

Fig. 14- Potato Early Blight using ResNet Model (Actual = Predicted)

The total number of images in Potato Disease leaf dataset is about 352; Hence, acquiring an accuracy of more than 98% and validation loss of 10.08% at 50 Epochs.

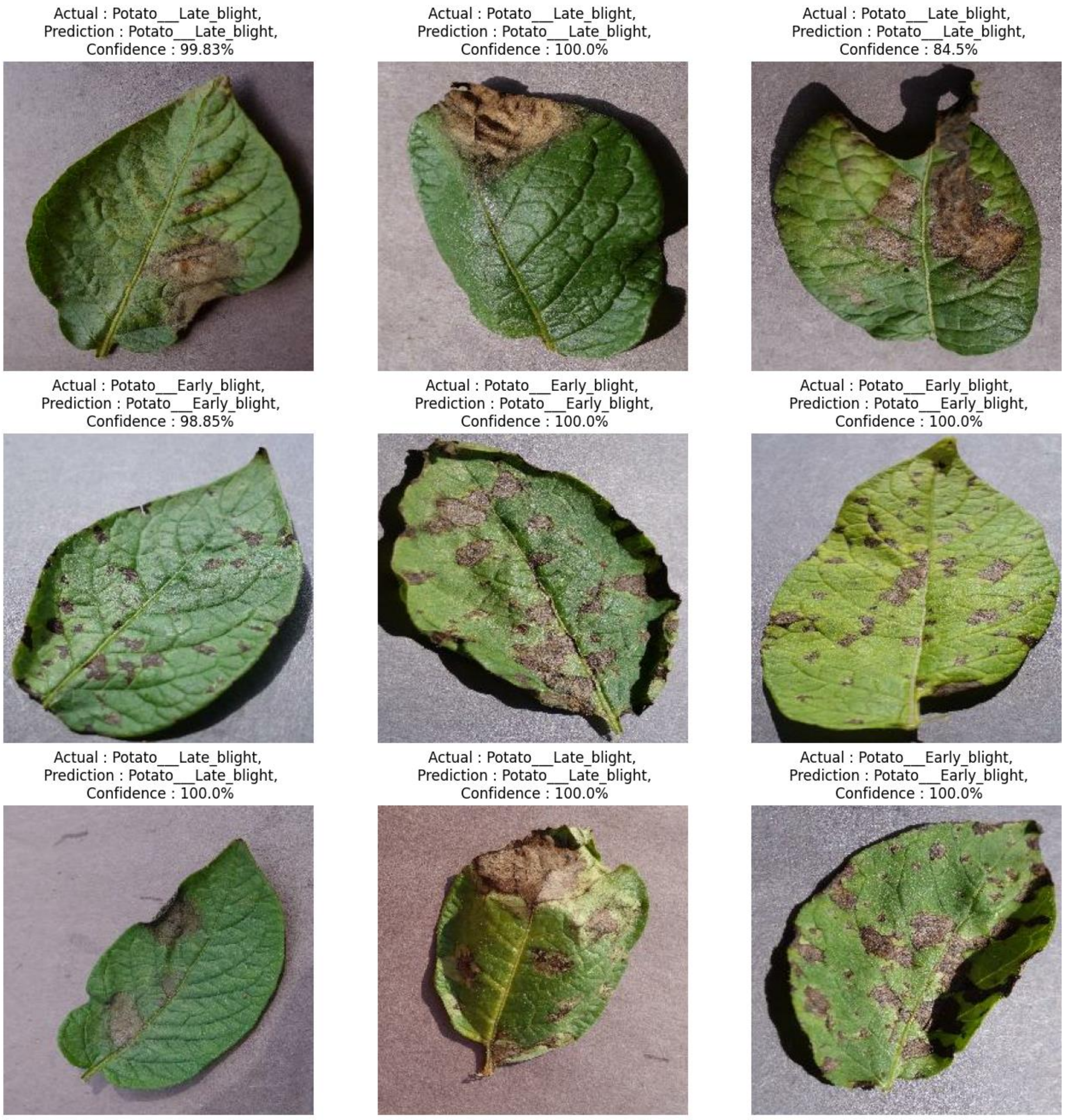

Actual : Potato___Late_blight,
Prediction : Potato___Late_blight,
Confidence : 99.83%
Actual : Potato___Late_blight,
Prediction : Potato___Late_blight,
Confidence : 100.0%
Actual : Potato___Late_blight,
Prediction : Potato___Late_blight,
Confidence : 84.5%
Actual : Potato___Early_blight,
Prediction : Potato___Early_blight,
Confidence : 98.85%
Actual : Potato___Early_blight,
Prediction : Potato___Early_blight,
Confidence : 100.0%
Actual : Potato___Early_blight,
Prediction : Potato___Early_blight,
Confidence : 100.0%
Actual : Potato___Late_blight,
Prediction : Potato___Late_blight,
Confidence : 100.0%
Actual : Potato___Late_blight,
Prediction : Potato___Late_blight,
Confidence : 100.0%
Actual : Potato___Early_blight,
Prediction : Potato___Early_blight,
Confidence : 100.0%

Fig. 15 – Final Outputs of dataset with 100% confidence

6.3. Discussion

Plant Disease Classification is an important procedure to be followed in this new normal of Agriculture Sector. The testing set created from the Plant Disease dataset contains 2252 images and has been used for the model evaluation in this project. For the detection, our model achieved a Mean Average Precision (MAP) of 96.50% at 40epochs on the test dataset. The loss values and loss classifier values at different epochs from 5 till 40.

When comparing the performance of a trained model for plant disease detection with other methods or models, it's essential to consider factors such as accuracy, computational efficiency, generalization ability, and interpretability. Here are some methods or models that are commonly compared in the context of plant disease detection:

Deep Learning Models: Convolutional Neural Networks (CNNs), such as AlexNet, ResNet, have shown superior performance in plant disease detection tasks due to their ability to learn hierarchical features directly from raw image data. These models typically outperform traditional machine learning algorithms in terms of accuracy but may require more computational resources and larger datasets for training.

Transfer Learning: Transfer learning involves leveraging pre-trained deep learning models, such as those trained on ImageNet, and fine-tuning them for plant disease detection tasks. Transfer learning can be more effective than training from scratch when dealing with limited labelled data and can lead to faster convergence and better generalization.

Future Work

In future work, it would be beneficial to train AlexNet models with a larger dataset containing more diseases and crops. An increase in dataset size can be achieved by adding more images as data points to enable the network to identify and classify a wider range of diseases and plant species.

The increasing use of cameras and improvement in their quality means that accurate diagnoses using smartphones is only a matter of time.

Furthermore, models can be trained using additional data such as panorama views of land areas, aerial photos, and images of different stages of different diseases. Additionally, the effect of image rotation on the network's performance could be explored.

Moreover, this work can be continued in future with further enhancements. Different sort of classification algorithms can be applied over this project for better results.

Conclusion

Hence from the above proposed method we have presented model for automatic classification of plant disease classification on Alex Net and CNN. The proposed model was based on Convolutional Neural Network architecture. We deployed Alex Net, a convolutional neural network, as the encoder to encode an image into a compact representation as the graphical features.

After that, a prediction model CNN was selected as the decoder to classify the plant diseases. Meanwhile, we utilized the soft attention model with CNN such that the learning can be focused on a particular part of the image to improve the performance. The whole model is fully trainable by using the stochastic gradient descent that makes the training process easier. The experimental evaluations indicate that the proposed model can classify the plant disease and pests for images automatically.

Reference

[1] C Jackulin, S. Murugavalli. A comprehensive review on detection of plant disease using machine learning and deep learning approaches. Measurement: Sensors (2022).

[2] Muhammad Shoaib, Babar Shah, Shaker EI-Sappagh, Akhtar Ali, Asad Ullah, Fayadh Alenezi, Tsanko Gechev, Tariq Hussain and Farman Ali. An advanced deep learning models-based plant disease detection: A review of recent research. Frontiers in Plant Science (2023).

[3] Vinod Kumar, Hirik Arora, Harsh, Jatin Sisodia. Resnet-based approach for Detection and Classification of Plant Leaf Diseases. IEEE Xplore (2020).

[4] Vijai Singh, A.K. Mishra. Detection of plant leaf diseases using image segmentation and soft computing techniques. Information processing in agriculture (2017).

[5] LILI LI1 , SHUJUAN ZHANG 2 , AND BIN WANG. Plant Disease Detection and Classification by Deep Learning—A Review. IEEE Access(2021)

[6] M. Islam, Anh Dinh, K. Wahid and P. Bhowmik, "Detection of potato diseases using image segmentation and multiclass support vector machine," 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), Windsor, ON, Canada, 2017, pp. 1-4, doi: 10.1109/CCECE.2017.7946594.

[7] Ranjan, Malvika, et al. "Detection and classification of leaf disease using artificial neural network." International Journal of Technical Research and Applications 3.3 (2015): 331-333.

[8] L. Liu and G. Zhou, "Extraction of the Rice Leaf Disease Image Based on BP Neural Network," 2009 International Conference on Computational Intelligence and Software Engineering, Wuhan, China, 2009, pp. 1-3, doi: 10.1109/CISE.2009.5363225.

[9] Ramcharan, Amanda, et al. "A mobile-based deep learning model for cassava disease diagnosis." Frontiers in plant science (2019): 272.

[10] H. Hong, J. Lin and F. Huang, "Tomato Disease Detection and Classification by Deep Learning," 2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), Fuzhou, China, 2020, pp. 25-29, doi:

10.1109/ICBAIE49996.2020.00012.

[11] D. Das, M. Singh, S. S. Mohanty and S. Chakravarty, "Leaf Disease Detection using Support Vector Machine," 2020 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 2020, pp. 1036-1040, doi:

10.1109/ICCSP48568.2020.9182128.

[12] Singh, Swati, et al. "Deep learning based automated detection of diseases from apple leaf images." CMC-Computers, Materials & Continua 71.1 (2022): 1849-1866.

[13] Bao, W., Yang, X., Liang, D., Hu, G., & Yang, X. (2021). Lightweight convolutional neural network model for field wheat ear disease identification. Computers and Electronics in Agriculture, 189, 106367.

[14] Storey, G., Meng, Q., & Li, B. (2022). Leaf disease segmentation and detection in apple orchards for precise smart spraying in sustainable agriculture. Sustainability, 14(3), 1458.