

PLANT DISEASE DETECTION USING DEEP LEARNING

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTERS OF TECHNOLOGY
IN
ARTIFICIAL INTELLIGENCE

Submitted by

Barsha Biswas

2K21/AFI/30

Under the supervision of

Dr. Rajesh Kumar Yadav



**DEPARTMENT OF COMPUTER SCIENCE &
ENGINEERING**

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi 110042

JUNE, 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, **Barsha Biswas**, Roll No – **2K21/AFI/30** student of M.Tech (Artificial Intelligence), hereby declare that the project Dissertation titled “**Apple Foliar Disease Detection using Deep Learning Technique**” which is submitted by me to the Department of Computer Science and Engineering, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi



Barsha Biswas

Date:

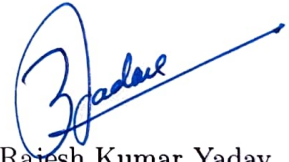
2K21/AFI/30

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled “**Apple Foliar Disease Detection using Deep Learning Technique**” which is submitted by Barsha Biswas, Roll No – 2K21/AFI/30, Department of Computer Science and Engineering, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi



Dr. Rajesh Kumar Yadav

Date:

Assistant Professor

Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I wish to express my sincerest gratitude to **Dr. Rajesh Kumar Yadav** for his continuous guidance and mentorship that he provided me during the project. He showed me the path to achieve my targets by explaining all the tasks to be done and explained to me the importance of this project as well as its industrial relevance. He was always ready to help me and clear my doubts regarding any hurdles in this project. Without his constant support and motivation, this project would not have been successful.

Place: Delhi


Barsha Biswas

Date:

2K21\AFI\30

Abstract

Agriculture, also known as Farming, is the science or practice of raising crops. And the whole world is dependent on it and around 38% of the world is dependent on it. So, due to this, its productivity rate should be high. The productivity rate of a plant is affected by the disease in a plant. So that's why plant disease should be detected at an early stage. For this, Farmers generally hire an agricultural expert who detects the disease using the naked eye and also they use instruments as well which are very expensive and which is not possible for all the farmers to afford it. There's another way to detect a plant disease, by using Artificial Intelligence(AI). Machine Learning(ML), Deep Learning(DL) which is a sub-branch of AI, is used in agriculture in order to detect disease in a plant.

So, in this work, a Dense-INC model is proposed which is based on Convolutional Neural Network(CNN) and it's inspired by DenseNet and InceptionNet. This model is trained on the "Plant Pathology 2020: FGVC7 dataset" and "Plant Pathology 2021: FGVC8 dataset".

The proposed model is first trained with 4 optimizers: Adam, Adadelta, Adagrad, and SGD with momentum and when I compare results then shows that Adagrad gives better results than other optimizers. To further evaluate the performance of the proposed model, the proposed model is further compared with two CNN-based models with Adagrad optimizer which are already been proposed. And the results show that the proposed model gives better results than two other CNN-based models and it's able to detect the disease with a low error rate.

Contents

Candidate's Declaration	i
Certificate	ii
Acknowledgement	iii
Abstract	iv
Content	vi
List of Tables	vii
List of Figures	ix
List of Symbols, Abbreviations	x
1 INTRODUCTION	1
2 PRIOR WORK	3
3 PRELIMINARY	12
3.1 Plant Disease	12
3.1.1 Classification of Plant Disease	13
3.2 Apple Foliar Disease	14
3.3 Convolutional Neural Network	16
3.4 InceptionNet	16
3.5 DenseNet	18
4 PROPOSED WORK	20
4.1 Problem Statement	20
4.2 Problem Solution	20
4.2.1 Data Preprocessing	20
4.2.2 Proposed Model	21
4.2.3 Optimizers Used	26
5 DATA EVALUATION	31
5.1 Datasets Used	31
6 EXPERIMENTS AND RESULTS	34
6.1 Performance Metrics	34
6.2 Analysis and Visualization of the Experimental Result	35
6.2.1 Performance on Plant Pathology 2020- FCVG7 dataset	35

6.2.2 Performance on Plant Pathology 2021- FCVG8 dataset	38
7 CONCLUSION AND FUTURE SCOPE	42
Bibliography	43
LIST OF PUBLICATIONS	51

List of Tables

2.1	Summarized review of literature papers	11
4.1	Detailed Architecture of Dense-INC architecture	22
5.1	Destribution of Datasets	31
6.1	Performance Evaluation of Plant Pathology 2020: FGCV7 dataset	38
6.2	Performance Evaluation of Plant Pathology 2021: FGCV8 dataset	40

List of Figures

1.1	Images of disease symptoms on apple leaves captured under different light conditions	
(a)	Indirect sunlight on leaf	
(b)	Direct sunlight on leaf	
(c)	Strong reflection on the leaf [9].	2
3.1	Five plants and the diseases that each of them carries[60]	13
3.2	Classification of plant diseases	14
3.3	Apple Leaves	
(a)	healthy leaves;	
(b)	Alternaria leaf spot;	
(c)	Brown spot;	
(d)	Mosaic;	
(e)	Grey spot;	
(f)	Rust.[61]	15
3.4	Schematic Diagram of CNN[65]	17
3.5	Schematic Diagram Of InceptionNet[69]	17
3.6	Schematic Diagram Of DenseNet[73]	18
4.1	Flowchart of Proposed Methodology	21
4.2	Architecture of Inception Module	24
4.3	Architecture of Dense Block	25
5.1	Sample Images of Plant Pathology 2020-FGCV7 Dataset[7]	32
5.2	Sample Images of Plant Pathology 2021-FGCV8 Dataset[8]	32
6.1	Performace Evaluation of the Proposed model using Adam Optimizer on Plant Pathology 2020-FCVG7 Dataset	36
6.2	Performace Evaluation of the Proposed model using Adadelta Optimizer on Plant Pathology 2020-FCVG7 Dataset	36
6.3	Performace Evaluation of the Proposed model using Adagrad Optimizer on Plant Pathology 2020-FCVG7 Dataset	37
6.4	Performace Evaluation of the Proposed model using SGD with momentum Optimizer on Plant Pathology 2020-FCVG7 Dataset	37
6.5	Performace Evaluation of the Proposed model using Adam Optimizer on Plant Pathology 2021-FCVG8 Dataset	38
6.6	Performace Evaluation of the Proposed model using Adadelta Optimizer on Plant Pathology 2021-FCVG8 Dataset	39
6.7	Performace Evaluation of the Proposed model using Adagrad Optimizer on Plant Pathology 2021-FCVG8 Dataset	39

6.8	Performance Evaluation of the Proposed model using SGD with momentum Optimizer on Plant Pathology 2021-FCVG8 Dataset	40
6.9	Comparison of Performance using Adagrad Optimizer	40

List of Symbols

<i>AI</i>	Artificial Intelligence
<i>APD</i>	Apple Plant Disease
<i>CAE</i>	Convolutional AutoEncoders
<i>CNN</i>	Convolutional Neural Network
<i>CV</i>	Computer Vision
<i>DCNN</i>	Deep Convolutional Neural Network
<i>DL</i>	Deep Learning
<i>DNN</i>	Deep Neural Network
<i>F – RCNN</i>	Faster Regions with Convolutional Neural Network
<i>GD</i>	Gradient Descent
<i>ML</i>	Machine Learning
<i>NLP</i>	Natural Language Processing
<i>NN</i>	Neural Network
<i>PD</i>	Plant Disease
<i>PDD</i>	Plant Disease Detection
<i>PNN</i>	Probabilistic Neural Network
<i>RNN</i>	Recurrent Neural Network
<i>ReLU</i>	Rectified Linear Unit
<i>R – CNN</i>	Regions with Convolutional Neural Network
<i>RMS</i>	Root Mean Square
<i>SGD</i>	Sochastic Gradient Descent
<i>SVM</i>	Support Vector Machine
<i>TTA</i>	Test-Time Augmentation

Chapter 1

INTRODUCTION

Agriculture[1] plays a vital role in the world economy and it's very important for the growth of the economy as well. Around 21% of the land is used for the production of apples. The apple plant can get easily affected by the disease. Diseases are caused by climate change, and by pathogens as well. Diseases that are commonly found on the apple plant which are caused by the pathogens are powdery mildew, rust, and scab.

All these diseases are found on the leaf of the apple plant. So, we can use ML[2], and the DL[3] model uses a leaf image in order to detect apple disease[4]. This method of using ML[2], and DL[3] algorithms is cheaper than hiring an expert. Hiring an expert is very expensive and also they don't give you accurate results all the time. So, Farmers can't afford to hire an expert but they can use AI-based models to do this task.

AI-based models learn, recognize patterns, and make decisions with minimal intervention from humans. It gives good results with a low error rate and also eliminates the possibility of human error. This helps farmers to gain insights into their crops and also use the data to increase overall production. DL[3] and ML[2], both are subfields of AI[5] and they both give phenomenal results. But these days, DL[3] based models are used more extensively than ML-based models for detecting PD[6] because DL[3] based models give better accuracy than ML-based models and also it does feature extraction and classification by themselves. But In ML[2], we have to do the feature extraction manually.

In DL[3], CNN has considered a revolutionary algorithm for the image classification task and the detection of PD[6] is the image classification task. Because it gives high accuracy.

In this work, the FGVC7 dataset from Plant Pathology 2020 Kaggle Competition and FGVC8 dataset[7] from Plant Pathology 2021 Kaggle Competition[8] are used. These two Kaggle competitions are supported by the "Cornell Initiative for Digital Agriculture"

In Fig. 1.1, we can see apple leaves with diseased parts and it can be detected by using deep learning techniques very easily.

Apple foliar Diseases[4] have been detected using the proposed CNN-based model and also it is compared with two CNN-based models which are already been suggested. Images are resized into 256x256.

This work involves the classification of images from the test dataset in order to determine whether the plants are healthy or not. The proposed model is used to identify different diseases.

This work contributes the following:

- This work reviews already proposed CNN models for the classification and identification of diseases. A Dense-INC model is proposed for this task. This proposed model is trained on two datasets: FGVC7[7] and FGVC8[8].

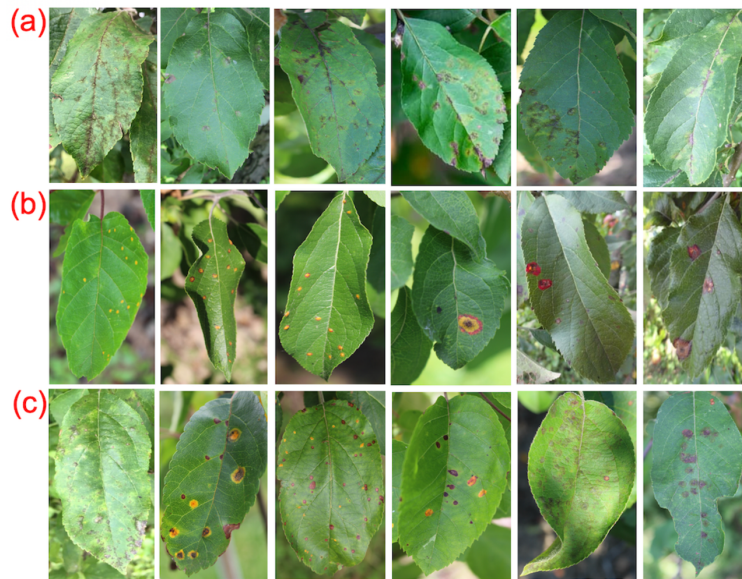


Figure 1.1: Images of disease symptoms on apple leaves captured under different light conditions

- (a) Indirect sunlight on leaf
- (b) Direct sunlight on leaf
- (c) Strong reflection on the leaf [9].

- The proposed Dense-INC model is inspired by DenseNet[10] and InceptionNet[11] and it consists of a Convolution Layer, Batch Normalization Layer, Activation Layer, Max-Pooling Layer, Global Average Pooling Layer, Dense Layer, Dropout Layer, Inception Block, Dense Block.
- Trained with 4 optimizers: Adam, Adadelta, Adagrad, and SGD with momentum and compared the results with each other. The learning rate[12] is set to 0.01 in each of the optimizers.
- Also compared the results with Two CNN-based models that are already proposed.
- And shows that the proposed model, the Dense-INC model works very well with Adagrad Optimizer which is trained on Plant Pathology 2021- FGCV8 Dataset[8].

In Chapter 2, Related Work will be discussed in which approaches are discussed which are already proposed to detect the disease in a plant. Then Preliminary will be discussed. The proposed Work will be outlined in Chapter 3. In Chapter 4 Data Evaluation will be explained. In Chapter 5, Experiments and Results will be explained, the Conclusion and Future Scope and then References.

Chapter 2

PRIOR WORK

Classifying as well as detecting a disease in a plant is very important in the sector of agriculture, and DL[3] plays a key part in the process of detecting the PD[6]. Here, we'll review a few of the most notable works related to this field. Some DL[3] approaches that were proposed in recent years:

S.No.	Ref.	Methodology	Conclusion	Dataset Used
1.	[13]	In this paper, the deep learning approach which is used is CNN. The proposed CNN model is trained using an open dataset called PlantVillage[14] with 39 different classes. Before training, this dataset undergoes six types of data augmentation methods. This proposed model is compared with transfer learning[15], [16] approaches.	Compared to popular transfer learning approaches, data augmentation can improve model performance. With transfer learning[15], [16], deep learning models can be built with less training data, less time, and fewer computational costs.	PlantVillage[14]
2.	[17]	With the help of Google Net[18] Inception structure and Rainbow concatenation, Deep-CNN is proposed in this paper. There are five types of apple leaf diseases[4] that severely affect apple yield, including Alternaria leaf spot, Brown spot, Mosaic, Grey spot, and Rust. This proposed model is compared with transfer learning[15], [16] approaches.	With Rainbow concatenation and the Inception Structure of Google Net[18], the model gives better accuracy than other transfer learning approaches.	Apple Leaf Disease Dataset (ALDD)[19]

3.	[20]	Designed deep learning based on the NASNet[21] architecture. With the help of techniques such as differential learning rates, cyclical learning rates, and test-time augmentation (TTA), the model was fine-tuned.	With techniques such as differential learning rates, cyclical learning rates, and test-time augmentation (TTA), model accuracy is improved without reducing training efficiency. Although there are complex inter- and intra-class variations in the images of plant leaves, the classification of plant leaves as either diseased or healthy appears promising.	PlantVillage[14]
4.	[22]	CNN was developed to identify tomato diseases present on monitored tomato plants, compared CNN with F-RCNN[23], and used Transfer Learning[15], [16] for disease recognition on tomato plant leaves.	Before training, data augmentation is done to increase the dataset size and reduce overfitting. Transfer Learning[15], [16] is used as well. In comparison to F-RCNN[23], the proposed model gives better accuracy.	PlantVillage[14]
5.	[24]	Using transfer learning[15], [16], the INC-VGG model, which combines VGG19[25] with InceptionV3[26], was proposed. After the bottom layers of VGG19 models, three 3X3 ConvBN layers were applied, then two InceptionV3[26] layers, and lastly a Softmax[27] layer was applied.	This proposed model is a combination of two pretrained-model and it gives better accuracy than other pretrained models.	PlantVillage[14]

6.	[28]	proposed the DenseNet201 model which contains 201 layers. This proposed model is compared with transfer learning approaches.	As a result of this proposed model, the vanishing gradient problem has been solved, feature propagation has been improved, feature reuse has been promoted, and the number of parameters has been reduced. Overfitting occurs when networks have too many connections, which not only reduces their computation and parameter efficiency, but also makes them more susceptible to overfitting.	PlantVillage[14]
7.	[29]	Suggested a CNN model in which SVM[30] is used as a classifier, Mean Shift Algorithm[31] is used for segmentation, artificial computation is used for shape feature extraction. CNN is used to extract color features. This proposed model identifies 4 types of disease in rice plants.	This model gives high accuracy around 96.8% but takes too much time to train.	PlantVillage[14]
8.	[32]	Proposed a CNN model having 6 convolution layers and 3 pooling layers. It is applied to images of the leaves of mango plants.	This proposed model is simple and computationally efficient but gives low accuracy.	PlantVillage[14] and images from fields.
9.	[33]	The CNN model is proposed which consists of three convolutional layers, three max-pooling layers, and two fully connected layers. Nine different types of tomato PDs[6] are identified using this model.	This CNN model is Memory Efficient but it's giving low testing accuracy.	PlantVillage[14]
10.	[34]	Developed a Deep Siamese CNN. Photos of grape leaves were collected and sorted into four categories.	The proposed model handles the challenge of a small dataset. Although it gives high accuracy, it is not computationally efficient.	Photographs from the field.

11.	[35]	A combined model of CNN and CAE[36] is proposed.	The proposed model produces accurate results. Using a less dimensional input image means less training time, and the program automatically detects the important features without any human intervention. The orientations are not encoded and the input data is not spatially invariant	PlantVillage[14]
12.	[37]	An architecture based on CNN was proposed for the detection of nine different types of diseases on tomato plant leaves. By using conventional architecture such as AlexNet[38] and GoogleNet[18], they developed a classifier.	This proposed model eliminates the need to extract features from images in order to train models but it takes too much time to train.	PlantVillage[14]
13.	[39]	A Bayesian Learning[40] based DCNN has been proposed, which implements Bayesian learning[40] on top of a residual network. This model is trained to detect tomato, potato, and pepper bell disease.	The process of feature learning is efficient but it has a high computational time.	PlantVillage[14]
14.	[41]	Deployed a slight variation of CNN called LeNet[42], [43] which consists of the convolutional, activation, pooling and fully connected layers. This model is trained to detect tomato disease.	Data Augmentation is done before feeding into the model. It's giving around 95% of accuracy which is much better than other CNN or CNN based models.	PlantVillage[14]
15.	[44]	An image-processing techniques are used in this paper to detect PDs[6]. The paper used semi-supervised learning[45] techniques to detect disease of four classes and access images of 5000 healthy and diseased plant leaves from an open dataset.	It is an improved and advanced technology. As a result of advances in technology, devices can detect and recognize PDs[6] more easily. Also give much better accuracy and reduce the impact of diseases on harvests by recognizing them sooner and treating them with faster treatment	PlantVillage[14]

16.	[46]	Proposed Improved PNN[47] which is more robust than simple PNN[47] model. It aims to identify the health and infected disease based on the identification of featured regions. The proposed work is divided into two two parts: first is to find the features then second is for the classification task.	This proposed model can identify a disease from leaf images, seed images as well as root images. The work is applied on a random basis and collected leaf images of different plants from the web.	Images from the Web
17.	[48]	The authors developed a three-layer CNN system with an average accuracy of 94.9% after 40 epochs of training. This study used 800 leaf images of cucumber.	This model is prone to over-fitting because it uses only 800 leaf images. And also a simple model which uses only three layers.	Images captured from the farm.
18.	[49]	They studied the AlexNet[38] and GoogLeNet[18] DNN architectures and achieved a 99.55% classification accuracy on the PlantVillage[14] dataset using 50k sample images over 30 training epochs.	AlexNet[38] and GoogLeNet[18] DNN architectures give more accurate results as compared to simple CNN models.	PlantVillage[14]

19	[50]	<p>Proposes a DL[3] based multi-task prediction system for detecting both PD[6] and plant species in images. The proposed system consists of two main parts: the feature extraction network and the prediction network. The feature extraction network uses a pre-trained CNN model to extract features from input images, while the prediction network includes two branches for disease and species detection. The two branches share the feature extraction network, which allows for joint learning of both tasks and improves prediction accuracy.</p>	<p>The system was trained and evaluated on a dataset of 9 plant species and 10 different diseases. The paper also compares the proposed system with other state-of-the-art approaches and shows that it outperforms them in terms of accuracy. Overall, the paper presents a promising approach for multitask PD[6] and species detection using deep learning, which could have practical applications in agriculture and plant science.</p>	<p>Images captured from the farm</p>
20.	[51]	<p>Proposes a DL[3] based approach for automatic blight disease detection in potato and tomato plants. The proposed approach uses a DCNN model to classify plant images as healthy or diseased based on the presence of blight disease.</p>	<p>The model was trained and evaluated on a dataset of 11,000 plant images. The paper also compares the proposed approach with other state-of-the-art approaches and shows that it outperforms them in terms of accuracy and computational efficiency. Overall, the paper presents a promising approach for automatic blight disease detection in potato and tomato plants using DL[3], which could have practical applications in agriculture for early disease detection and management.</p>	<p>PlantVillage[14]</p>

21.	[52]	<p>The article discusses the use of deep learning algorithms for detecting PDs[6] using images. It provides an overview of DL[3] models, datasets, and pre-processing techniques used for training these models. The article highlights the importance of data augmentation, transfer learning[15], [16], and hyperparameter tuning for achieving high accuracy in PDD. It also discusses the challenges and future directions of image-based PDD using DL[3].</p>	<p>Overall, the article provides a comprehensive overview of the use of DL[3] algorithms for image-based PDD and highlights the potential for these algorithms to revolutionize agricultural production by enabling early and accurate detection of PDs[6].</p>	<p>All the datasets used in the task of PDD.</p>
22.	[53]	<p>The article discusses a performance-optimized DL[3] based approach for detecting PDs[6] in horticultural crops in New Zealand. The approach uses CNNs and transfer learning[15], [16] to achieve high accuracy in disease detection. The article describes the dataset used for training the CNN models, which includes images of diseased and healthy plants. The article also discusses the pre-processing techniques used to improve the performance of the models.</p>	<p>The results show that the proposed approach achieved high accuracy in detecting diseases in horticultural crops, which can help farmers identify and treat PDs[6] early, leading to better crop yield and quality. Overall, the article demonstrates the potential of DL[3] based approaches for improving PDD in horticultural crops.</p>	<p>PlantVillage[14].</p>

23.	[54]	<p>The article presents an approach for detecting PD[6] in cardamom using the EfficientNetV2[55] DL[3] model. The approach involves collecting a dataset of images of healthy and diseased cardamom plants, pre-processing the images, and training the EfficientNetV2[55] model on the dataset. The article also discusses the use of data augmentation and transfer learning[15], [16] techniques to improve the performance of the model.</p>	<p>The results show that the proposed approach achieved high accuracy in detecting diseases in cardamom plants, which can help farmers identify and treat PDs[6] early, leading to better crop yield and quality. Overall, the article demonstrates the potential of DL[3] based approaches for improving PDD in cardamom plants.</p>	PlantVillage[14].
24.	[56]	<p>The article presents an end-to-end DL[3] model for classifying corn leaf diseases using a dataset of images of healthy and diseased corn leaves. The proposed model uses a pre-trained CNN architecture and fine-tuning to achieve high accuracy in disease classification. The article also discusses the pre-processing techniques used to enhance the performance of the model, including image normalization and data augmentation.</p>	<p>The results show that the proposed model achieved high accuracy in classifying corn leaf diseases, which can help farmers identify and treat PDs[6] early, leading to better crop yield and quality. Overall, the article demonstrates the potential of DL[3] based approaches for improving PDD and classification in corn crops.</p>	PlantVillage[14]

25.	[57]	The article proposes a method for detecting APD[4] using leaf images through a CNN. The approach involves collecting a dataset of images of healthy and diseased apple leaves, pre-processing the images, and training the CNN model on the dataset. The article also discusses the use of data augmentation and transfer learning[15], [16] techniques to improve the performance of the model.	The results show that the proposed approach achieved high accuracy in detecting APDs[4], which can help farmers identify and treat PDs[6] early, leading to better crop yield and quality. Overall, the article demonstrates the potential of DL[3] based approaches for improving PDD in apple crops.	PlantVillage[14]
-----	------	--	--	------------------

Table 2.1: Summarized review of literature papers

The following are some of the key takeaways from the preceding table:

1. Data augmentation has shown to be a highly successful strategy for improving overall performance and making the model more resilient to noise.
2. Oversampling strategies for adjusting for uneven numbers of entities increase performance in classes with unequal numbers of entities.
3. For image classification tasks, DL[3] models are more accurate than machine learning models.
4. The classification accuracy improves as the number of photos per class used to train the models grows.
5. After achieving a particular high degree of accuracy, expanding the model's complexity to better suit the job yields decreasing returns.
6. When the model is trained for a small number of epochs (less than 20), the model frequently does not acquire enough features from the input to correctly categorize the test pictures.
7. Transfer learning[15], [16] is an excellent strategy for solving the PD[6] classification issue since it gives improved performance as well as reduced training time because these models have already been trained for greater performance on much bigger classification problems in the same category.
8. When pre-trained model weights are frozen, the models tend to be less accurate on the given task since they are more generalised.
9. The main advantage to use DL[3] Technique instead of ML[2] Technique is that DL[3] does Feature Extraction and Classification by itself but in ML[2] we do Feature Extraction Manually.

Chapter 3

PRELIMINARY

The combination of InceptionNet[11] and DenseNet[10] based CNN architecture was utilized to detect PD[6]. In the sections that follow, the design of these components is explained. Also mentioned below is thorough information regarding the techniques and Python libraries that we employed in our suggested implementation, such as Keras[58], TensorFlow[59], and others.

3.1 Plant Disease

PDs[6] are a group of disorders that affect the growth, development, and productivity of plants. These diseases can be caused by a variety of factors, including bacteria, fungi, viruses, nematodes, and environmental stressors such as drought, excess water, or extreme temperatures.

PDs[6] can have significant impacts on agricultural production and food security. They can reduce crop yields, lower crop quality, and cause economic losses for farmers and agricultural industries. In some cases, PDs[6] can even lead to the complete loss of a crop, which can have devastating consequences for farmers and communities that depend on agriculture for their livelihoods.

There are several types of PDs[6], including foliar diseases, root diseases, stem diseases, and fruit and seed diseases. Foliar diseases affect the leaves of plants, while root diseases affect the roots. Stem diseases can affect the stems, branches, and twigs of plants, while fruit and seed diseases affect the fruit or seed production of plants.

PDs[6] can be managed in several ways, including cultural, chemical, and biological methods. Cultural methods involve practices such as crop rotation, sanitation, and use of resistant cultivars to prevent the spread and development of PDs[6]. Chemical methods involve the use of pesticides and fungicides to control PDs[6], while biological methods involve the use of natural enemies, such as predatory insects or beneficial microbes, to control PDs[6].

Prevention is the best strategy for managing PDs[6]. Farmers and agricultural industries can prevent PDs[6] by implementing good agricultural practices, such as using disease-free seedlings, practicing crop rotation, and monitoring plants regularly for signs of disease. Early detection and diagnosis of PDs[6] are also critical for effective management and control.

In summary, PDs[6] are a significant challenge for agricultural production and food security. Effective management and control of PDs[6] require a combination of prevention, early detection and diagnosis, and appropriate management strategies tailored to the specific disease and crop being affected.

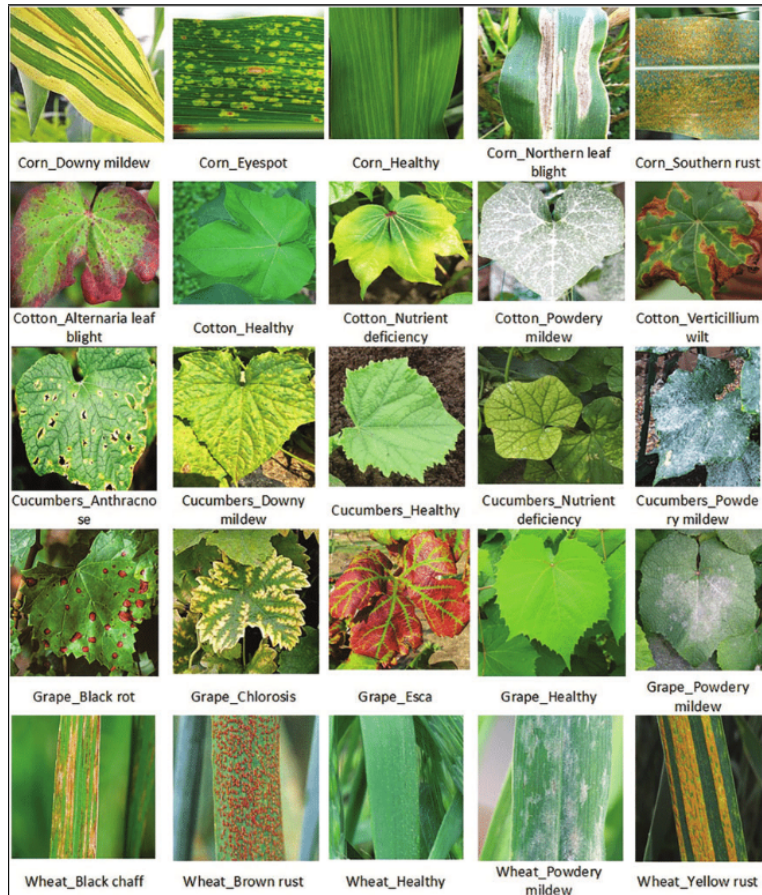


Figure 3.1: Five plants and the diseases that each of them carries[60]

3.1.1 Classification of Plant Disease

PDs[6] can be classified into several categories based on various criteria, including the type of pathogen causing the disease, the symptoms exhibited by the plant, and the mode of transmission of the disease. Here are some common classifications of PDs[6]:

1. Based on the type of pathogen:

- Bacterial diseases - caused by bacteria such as fire blight and crown gall.
- Nematode diseases - caused by microscopic worms that attack plant roots.
- Fungal diseases - caused by fungi such as powdery mildew, rusts, and blights.
- Viral diseases - caused by viruses such as mosaic viruses and leaf curl viruses.
- etc...

2. Based on the symptoms:

- Leaf diseases - affecting the leaves of the plant.
- Stem and trunk diseases - affecting the stems or trunks of the plant.
- Root diseases - affecting the roots of the plant.
- Fruit and flower diseases - affecting the fruits or flowers of the plant.

- etc...

3. Based on the mode of transmission:

- Airborne diseases - transmitted through the air, such as powdery mildew.
- Soilborne diseases - transmitted through the soil, such as root rot.
- Waterborne diseases - transmitted through water, such as downy mildew.
- Insect-borne diseases - transmitted by insects, such as citrus greening disease
- etc...

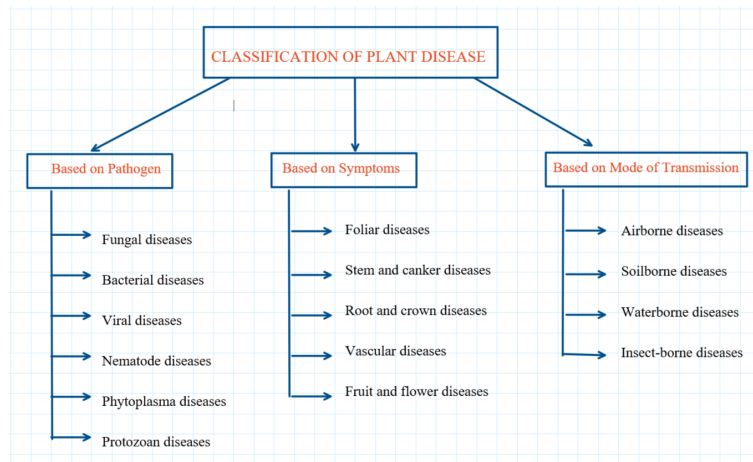


Figure 3.2: Classification of plant diseases

Understanding the classification of PDs[6] can help farmers and gardeners identify and control PDs[6] more effectively. By recognizing the type of pathogen causing the disease, the symptoms exhibited by the plant, and the mode of transmission of the disease, appropriate management strategies can be implemented to prevent or control the spread of the disease.

3.2 Apple Foliar Disease

Apple foliar diseases are PDs[6] that affect the leaves of apple trees. These diseases can have a significant impact on the health and productivity of apple trees, as well as the quality of the fruit produced.

Here are some common foliar diseases that affect apple trees:

1. Apple scab - caused by the fungus *Venturia inaequalis*, apple scab is a common and destructive disease of apple trees. It affects the leaves, fruit, and twigs of the tree, causing dark spots and lesions on the leaves and fruit, and can ultimately lead to defoliation and reduced yield.
2. Cedar apple rust - caused by the fungus *Gymnosporangium juniperi-virginianae*, cedar apple rust is a fungal disease that affects both cedar trees and apple trees. It causes bright orange spots on the leaves of the apple tree, and can lead to defoliation and reduced fruit quality.

3. Powdery mildew - caused by various fungi, powdery mildew is a fungal disease that affects many different plants, including apple trees. It appears as a white, powdery coating on the leaves and can cause distorted growth and reduced fruit quality.
4. Fire blight - caused by the bacteria *Erwinia amylovora*, fire blight is a bacterial disease that affects many different plants, including apple trees. It causes the leaves and shoots to turn brown and wilt, and can lead to defoliation and reduced fruit quality.
5. Black rot - caused by the fungus *Botryosphaeria obtusa*, black rot is a fungal disease that affects the leaves, fruit, and twigs of the apple tree. It causes black spots on the leaves and fruit, and can lead to defoliation and reduced yield.

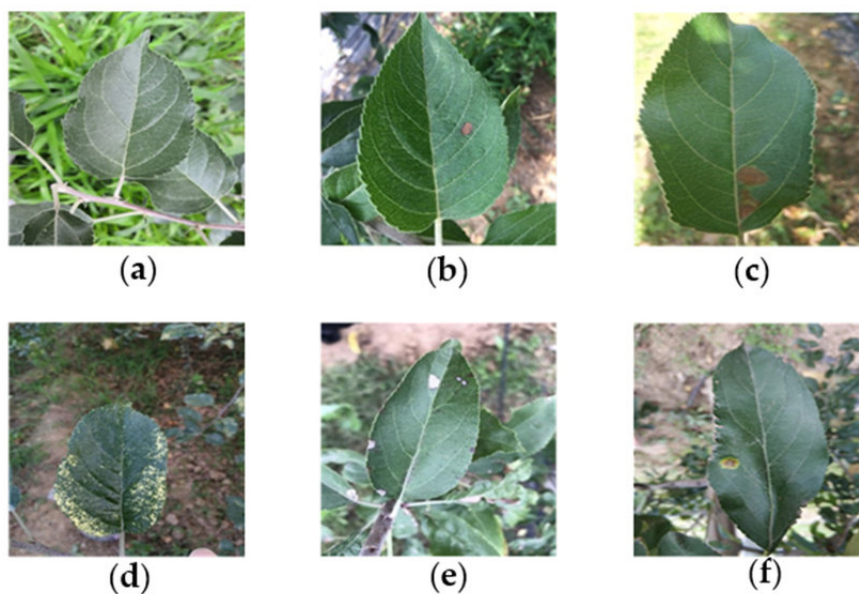


Figure 3.3: Apple Leaves

- (a) healthy leaves;
- (b) Alternaria leaf spot;
- (c) Brown spot;
- (d) Mosaic;
- (e) Grey spot;
- (f) Rust.[61]

Prevention and control of apple foliar diseases include:

- Planting disease-resistant varieties
- Proper pruning to increase air circulation and sunlight penetration
- Removing infected leaves and fruit from the tree and the ground
- Fungicide applications, especially during periods of high humidity and rain
- Avoiding overhead irrigation, which can spread fungal spores

Regular inspection and monitoring of apple trees for signs of foliar diseases is crucial for early detection and prompt treatment.

3.3 Convolutional Neural Network

A CNN is a type of ANN[62], [63] commonly used for image and video recognition and analysis. CNNs are inspired by the structure and function of the visual cortex in animals and can automatically learn and extract features from images and other types of visual data.

The key building blocks of a CNN are convolutional layers, pooling layers, and fully connected layers. The convolutional layers apply a set of filters or kernels to the input image or feature map, which convolves the image and produces a new set of feature maps that capture local patterns and structures. The pooling layers then downsample the feature maps to reduce the dimensionality and improve the computational efficiency of the network. Finally, the fully connected layers use the flattened feature maps as input and perform classification or regression tasks based on the learned features.

CNNs are trained using a supervised learning[64] approach, where the network is fed a large set of labeled training examples and adjusts its weights and biases to minimize the error or loss function. This process is typically performed using backpropagation, which computes the gradients of the loss function with respect to the weights and biases and updates them accordingly.

One of the advantages of CNNs is their ability to automatically learn and extract features from images and other types of visual data, without the need for explicit feature engineering. This allows them to perform well on a variety of tasks, including object detection, image segmentation, and image classification.

In recent years, CNNs have achieved state-of-the-art performance on many CV[66] tasks, including object detection and image classification. They have also been used in many other domains, such as NLP[67], speech recognition, and even music generation.

Overall, CNNs are a powerful and versatile type of NN that have revolutionized CV[66] and many other fields. They are widely used in industry and academia for a variety of applications and continue to be an active area of research and development.

3.4 InceptionNet

InceptionNet is a DCNN architecture for image classification and object recognition tasks. It was developed by a team of researchers at Google and was first introduced in 2014.

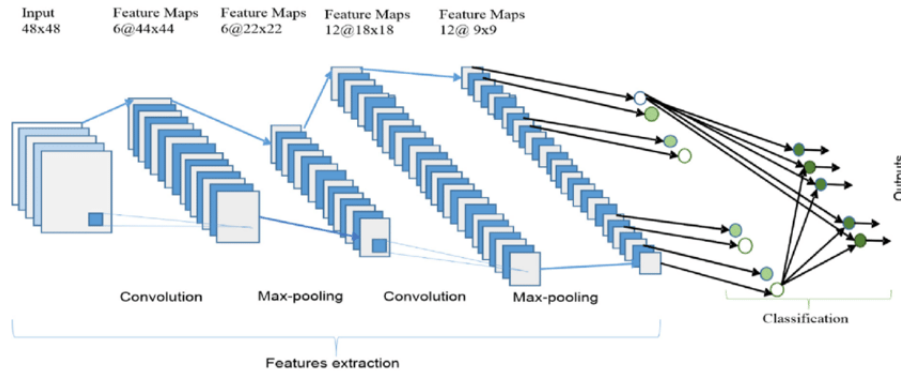


Figure 3.4: Schematic Diagram of CNN[65]

InceptionNet was designed to address the challenge of creating deeper and more complex NNs[68] while maintaining computational efficiency and avoiding overfitting.

The main innovation of InceptionNet is the use of inception modules, which are multiple convolutional layers with different filter sizes and pooling operations stacked together in parallel. These modules allow the network to capture both fine-grained and coarse-grained features at different scales and resolutions, making it more effective at recognizing objects with different shapes and sizes.

Another key feature of InceptionNet is the use of 1×1 convolutions, which help reduce the dimensionality of the feature maps and improve computational efficiency. These convolutions allow the network to combine information from different channels in a more efficient manner and reduce the number of parameters needed in the network.

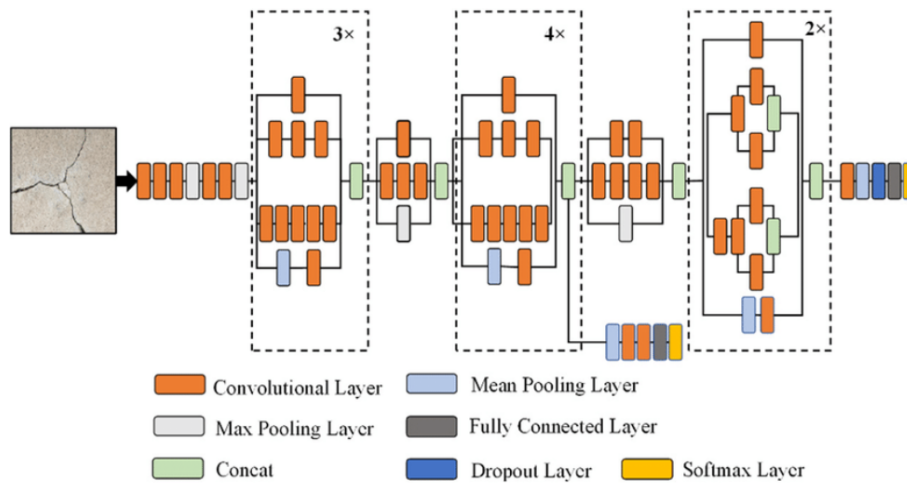


Figure 3.5: Schematic Diagram Of InceptionNet[69]

InceptionNet has achieved state-of-the-art performance on several benchmark image classification tasks, including the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014 and 2015. It has also been used for other tasks such as object detection, image segmentation, and visual question answering.

InceptionNet has inspired several variations and improvements, such as Inception V2[70], Inception V3[26], and Inception-ResNet[71], which combine the inception mod-

ules with residual connections for improved performance. These variations have achieved even better performance on various tasks and have become widely used in industry and academia.

Overall, InceptionNet is a powerful and efficient DL[3] architecture that has significantly advanced the state-of-the-art in image classification and other CV[66] tasks. Its success has inspired further research and development in the field of DL[3] and has contributed to the rapid progress of AI[5].

3.5 DenseNet

DenseNet (Densely Connected Convolutional Networks) is a DL[3] architecture for image recognition tasks that was introduced by a team of researchers at Facebook AI Research in 2017. DenseNet is designed to address the challenges of training very DCNNs by encouraging feature reuse and reducing the number of parameters in the network.

The main idea behind DenseNet is to connect every layer to every other layer in a feedforward manner. In traditional CNNs, each layer takes the output of the previous layer as its input. In DenseNet, however, each layer takes the feature maps of all preceding layers as its input. This allows the network to reuse features more efficiently and reduces the number of parameters needed to train the network.

DenseNet is built up of dense blocks, which are composed of multiple convolutional layers with batch normalization and ReLU[72] activation functions. Each dense block connects to the preceding block by concatenating the feature maps of all preceding layers. The dense blocks are separated by transition layers, which reduce the dimensionality of the feature maps and control the number of channels.

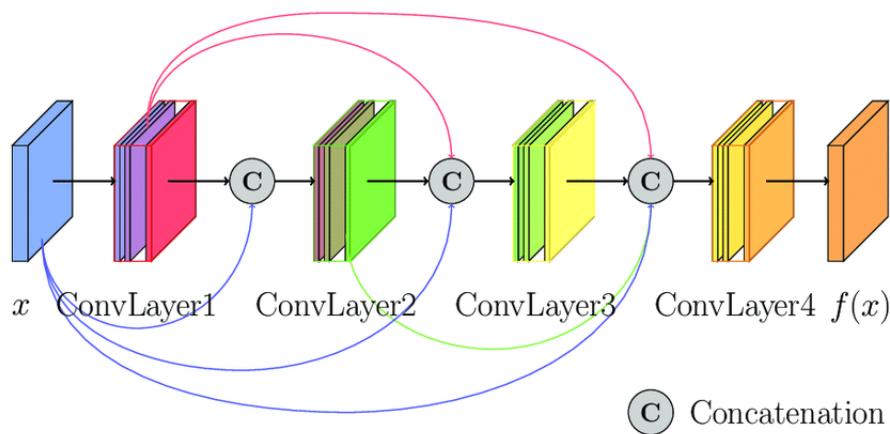


Figure 3.6: Schematic Diagram Of DenseNet[73]

DenseNet has achieved state-of-the-art performance on several benchmark image recognition tasks, including the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2017. It has also been used for other tasks such as object detection and segmentation.

One of the advantages of DenseNet is its ability to reduce the number of parameters needed for training, which can lead to faster training times and better generalization performance. This makes DenseNet particularly useful for tasks where computational resources are limited.

Overall, DenseNet is a powerful DL[3] architecture that has significantly advanced the state-of-the-art in image recognition tasks. Its success has inspired further research and development in the field of DL[3] and has contributed to the rapid progress of AI[5].

Chapter 4

PROPOSED WORK

4.1 Problem Statement

As we know, agriculture[1] is the practice of raising a crop and the whole world population is dependent on it. So, due to this, its productivity rate should be high. The productivity rate of a plant is affected by the disease in a plant. So that's why PD[6] should be detected at an early stage. PD[6] detection involves developing an accurate and efficient system for identifying diseases that affect plants. PDs[6] can be caused by various factors such as bacteria, fungi, viruses, and environmental conditions, and can have a significant impact on crop yield and quality. Early detection and diagnosis of PDs[6] are critical to prevent the spread of the disease and minimize crop damage.

Traditionally, PDD has relied on visual inspection by trained experts, which can be time-consuming, costly, and subject to human error. With advances in technology, automated systems for PD[6] detection using ML[2] and CV[66] techniques have been developed. These systems typically involve capturing images of plants, extracting relevant features, and using ML[2] algorithms to classify the images as healthy or diseased. The goal is to develop accurate and efficient systems for PDD that can be used by farmers and agricultural experts to monitor and manage crop health.

4.2 Problem Solution

In this section, we will discuss the proposed Dense-INC model that is proposed in this work. In this section, Data Preprocessing is described in the first subsection, then the Proposed model is explained and then Optimizers used are explained in the last subsection.

Overall workflow of the proposed solution is given in Fig. 4.1.

4.2.1 Data Preprocessing

Here, two datasets[7], [8] are used in the evaluation process of the proposed model. The details of these datasets are present in the methodology section. In the cases if the train data set is very large, we have chosen to trim the train dataset to max 1000 samples per class in order to reduce training time. All the images were resized to 256*256 pixels and Data Augmentation is done as well before feeding to the model. Rescaling, rotation, width_shift, height_shift, horizontal_flip, zoom, shear and vertical_flip are done before training. It is done to avoid the problem of overfitting.

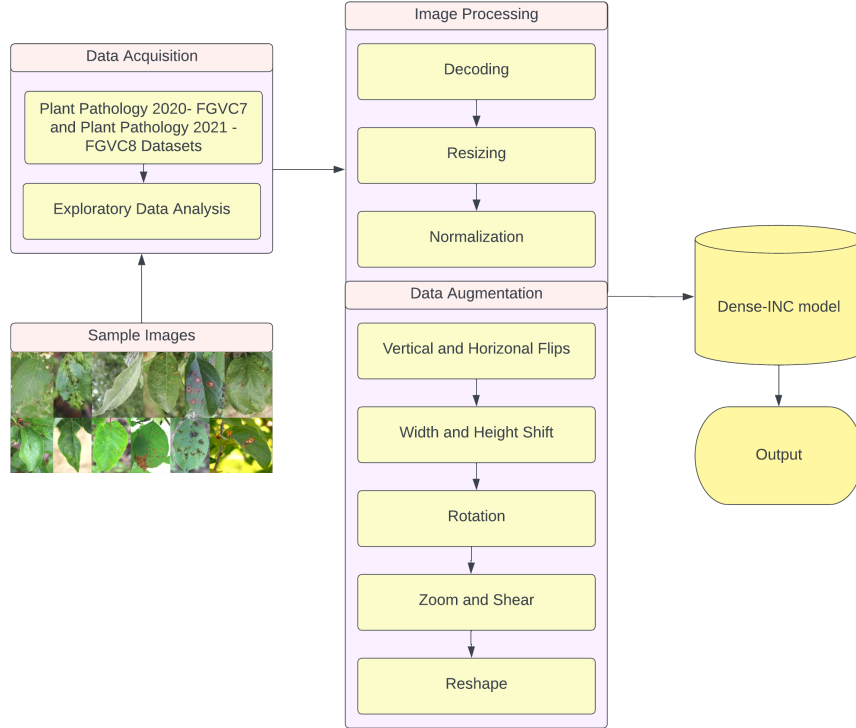


Figure 4.1: Flowchart of Proposed Methodology

4.2.2 Proposed Model

CNN based model called Dense-INC has been proposed which contains Convolution Layer, Batch Normalization Layer, Activation Layer, Max-Pooling Layer, Global Average Pooling Layer, Dense Layer, Dropout Layer, Inception Block and Dense Block.

Here Inception Block is inspired by InceptionNet[74] and Dense Block is Inspired by DenseNet[10]. And in this proposed architecture, ReLU[72] and Softmax[27] activation function is used.

Dense-INC architecture is given in Table 4.1.

In Dense-INC model, there are 2 Convolution Layers, 3 Batch Normalization Layers, 2 Max-Pooling Layers, 2 Activation Layers(ReLU[72]), 1 Global Average Pooling Layer, 4 Dense Layers(3 layers having ReLU[72] activation function and 1 layer having Softmax[27] activation function), 1 Dropout Layers having dropout rate of 0.5, 2 Inception Module and 2 Dense Module.

S.No.	Name	Activations	Total Learnables
1.	Image_Input	256x256x3	0
2.	conv2d	256x256x64	1792
3.	batch_normalization	256x256x64	265
4.	activation	256x256x64	0
5.	max_pooling2d	128x128x64	0
6.	inception_module	128x128x16	0
7.	dense_module	128x128x54	0

8.	conv2d_1	128x128x32	156704
9.	batch_normalization_1	128x128x32	128
10.	activation_1	128x128x32	0
11.	max_pooling2d_1	64x64x32	0
12.	inception_module_1	64x64x80	0
13.	dense_module_1	64x64x272	0
14.	global_average_pooling2d	272	0
15.	dense	9216	2515968
16.	dense_1	4096	37752832
17.	dense_2	1024	4195328
18.	dense_3	128	131200
19.	dropout	128	0
20.	batch_normalization_3	128	512
21.	dense_4	<ul style="list-style-type: none"> • 4 (Plant Pathology 2020) • 12 (Plant Pathology 2021) 	<ul style="list-style-type: none"> • 516 • 1548
	Total params:		<ul style="list-style-type: none"> • 46,254,604 (Plant Pathology 2020) • 46,255,636 (Plant Pathology 2021)

Table 4.1: Detailed Architecture of Dense-INC architecture

Detailed Explanation of Each Layer is given below:

1. Input: Defines the shape of the input tensor, which is (input_image, input_image, 3) for this model.
2. Conv2D: Performs 2D convolution on the input tensor with a specified number of filters (64 in the first layer), kernel size of 3x3, and padding of 'same', which means the output feature maps will have the same spatial dimensions as the input.
3. BatchNormalization: Normalizes the outputs of the previous layer to ensure the mean activation is close to 0 and the activation standard deviation is close to 1.
4. Activation: Applies the ReLU[72] activation function to the output of the previous layer.
5. MaxPooling2D: Downsamples the input tensor along the spatial dimensions by taking the maximum value within a specified pool size (default is 2x2) and strides (default is pool size).
6. inception_module: A custom layer that applies the inception module that takes the output of the previous layer as input and returns a feature map with increased depth through concatenation of different convolutional filters.
7. dense_block: A custom layer that applies the dense block that takes the output of the previous layer as input and returns a feature map with increased depth through concatenation of previous feature maps.
8. GlobalAveragePooling2D: Takes the average of each feature map in the previous layer along the spatial dimensions to reduce the tensor to a vector of length equal to the number of filters.
9. Dense: A fully connected layer that takes the vector output of the previous layer and applies a specified number of neurons (9216, 4096, 1024, and 128 in this model) with ReLU[72] activation.
10. Dropout: Randomly sets a fraction of input units to 0 at each update during training to prevent overfitting.
11. BatchNormalization: Applies batch normalization to the output of the previous layer.
12. Dense: A fully connected layer that takes the output of the previous layer and applies a Softmax[27] activation to classify the input into one of 4 possible classes(in case of Plant Pathology 2020-FGCV7[7]) or 12 possible classes(in case of Plant Pathology 2021-FGCV8[8]) [as defined by the number of neurons in the output layer].

Two Modules that are defined in this work:

- Inception Module
- Dense Block

1. Inception Module

Inception Module is inspired by InceptionNet[74]. InceptionNet[74] is a DCNN architecture that uses inception modules, which allow for efficient information processing and feature extraction at multiple scales. Inception Module in the proposed work contains 3 Convolution Layers of kernel size 1x1 with activation function ReLU[72] and also the padding is same, 3 Convolution Layers of kernel size 3x3 with activation function ReLU[72] and also the padding is same, 1 Convolution Layers of kernel size 5x5 with activation function ReLU[72] and also the padding is same and 1 Max-Pooling Layer having poolsize of 3, strides of 1x1, the padding is same and the Concatenation Layer to merge some Layers.. The architecture of InceptionNet is given in Fig.3.5.

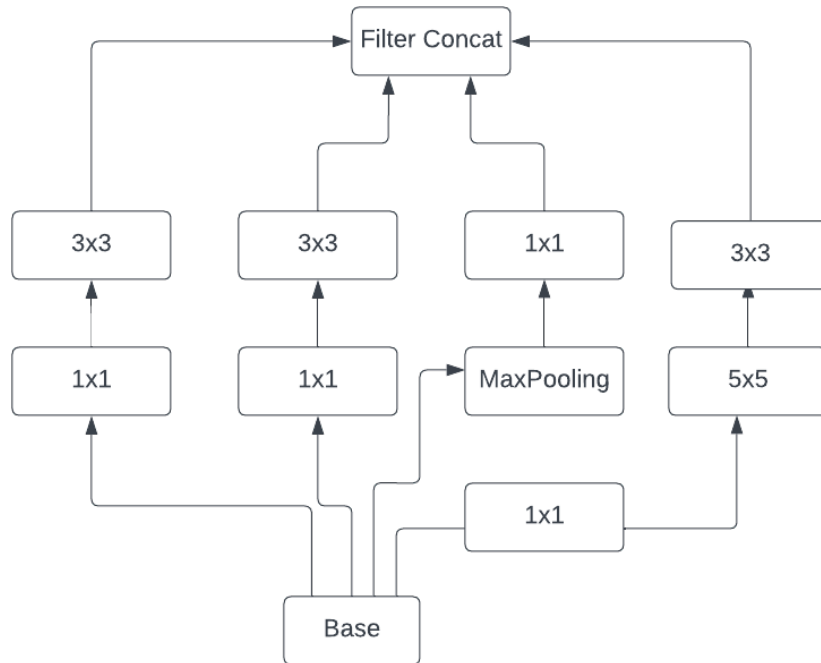


Figure 4.2: Architecture of Inception Module

In Fig. 4.2, The Inception Module takes an input tensor of shape (h, w, c) and applies a series of operations to it to produce an output tensor of shape $(h, w, 4 * \text{filter_}3 \times 3)$.

The module consists of four branches of convolutional layers: a 1x1 convolution branch, a 3x3 convolution branch, a max pooling and 1x1 convolution branch, and a 1x1 convolution followed by 5x5 convolution and then 3x3 convolution branch. The output of each branch is concatenated along the channel axis to produce the final output.

Fig. 4.3 shows the flow of data through the Inception module, starting with the input tensor and ending with the output tensor. The operations performed at each layer are shown, along with the shape of the tensors at each stage. The final output tensor is the concatenation of the output tensors from each branch along the channel axis.

2. Dense Block

Dense Module is Inspired from DenseNet[10]. DenseNet[10] is a deep CNN architecture that connects each layer to every other layer in a feed-forward fashion, leading to feature reuse, parameter efficiency, and improved gradient flow. DenseNet[10] consists of 4 Layers: Batch Normalization Layer, Activation Layer, Convolution Layers of kernel size 3x3 with the same padding and the Concatenation Layer to merge all the Layers. The architecture of DenseNet[10] is given in Fig. 3.6.

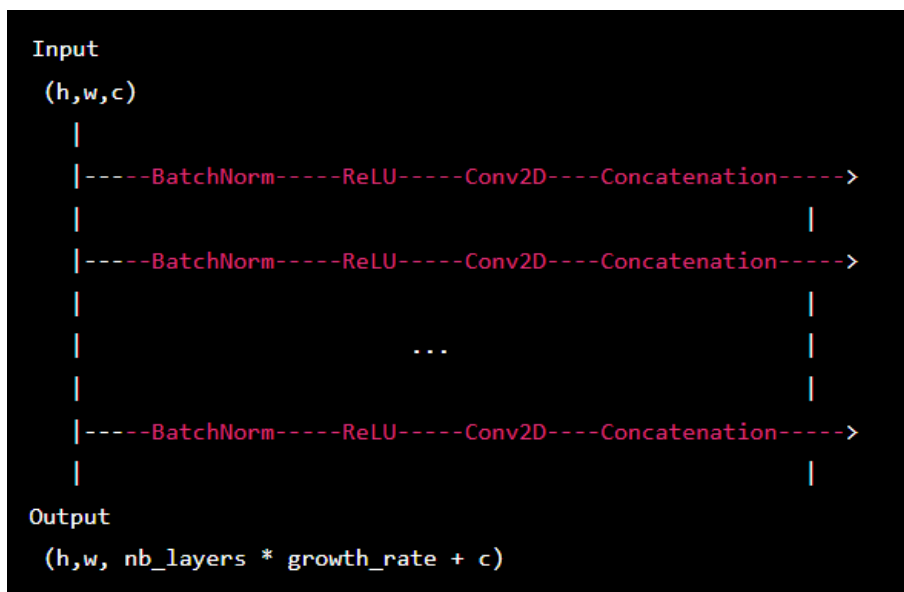


Figure 4.3: Architecture of Dense Block

In Fig. 4.3, The Dense Module takes an input tensor of shape (h, w, c) and applies a series of operations to it to produce an output tensor of shape $(h, w, nb_layers * growth_rate + c)$.

The block consists of a series of nb_layers layers, each of which performs batch normalization, ReLU[72] activation, a 3x3 convolution with $growth_rate$ filters, and concatenation with the input tensor. The output of each layer is fed as input to the next layer, and the input tensor is updated to be the output of the last layer. The final output tensor is the concatenation of all the intermediate feature maps along the channel axis.

The Fig. 4.3 shows the flow of data through the dense module, starting with the input tensor and ending with the output tensor. The operations performed at each layer are shown, along with the shape of the tensors at each stage. The Concatenation block takes the outputs from all the previous Convolutional blocks and concatenates them along the channel axis.

Here, Kernel Initializer is used. In DL[3], a kernel initializer is a method used to initialize the weights of the kernels or filters used in convolutional layers of a NN[68]. The weights of these kernels are important as they determine the output of the layer and thus, the overall performance of the network.

There are several different types of kernel initializers available, each with its own strengths and weaknesses. Some of the most commonly used kernel initializers include: Random normal, Random uniform, Glorot uniform, He normal, LeCun uniform.

Choosing the right kernel initializer can have a significant impact on the performance of a NN[68], especially in the early stages of training. By setting the weights of the kernels to appropriate values, we can help ensure that the network learns useful features and achieves high accuracy on the task at hand.

In this proposed Dense-INC model, two kernel initializers are used: LeCun-uniform initializer and another one is He-uniform initializer.

The LeCun-uniform kernel initializer and the He-uniform kernel initializer are two commonly used methods for initializing the weights of convolutional kernels in NNs[68].

- The **LeCun-uniform kernel initializer** is named after Yann LeCun, and is designed to initialize the weights of kernels in a way that helps to prevent gradients from vanishing or exploding during training. The initializer uses a uniform distribution to randomly initialize the weights of each kernel, with a range determined by the formula $\sqrt{3/\text{fan_in}}$, where `fan_in` is the number of input channels to the layer. This initialization method is commonly used in convolutional layers of NNs[68], and has been shown to work well in practice for a wide range of tasks.
- The **He-uniform kernel initializer**, on the other hand, is named after Kaiming He, and is a modified version of the LeCun-uniform initializer that is specifically designed for use with ReLU[72] activation functions. The He-uniform initializer sets the range of the uniform distribution used to initialize the weights to $\sqrt{6/\text{fan_in}}$, where `fan_in` is the number of input channels to the layer. This initialization method is preferred when using ReLU[72] activation functions because it helps to prevent the "dying ReLU" problem, where a large number of neurons in the network can become permanently "dead" (i.e. outputting zero) due to a zero gradient during training.

Both the LeCun-uniform and He-uniform kernel initializers have been shown to work well in practice for a variety of DL[3] tasks, and are widely used in state-of-the-art NN architectures. The choice of which initializer to use may depend on the specific requirements of the task at hand, and the characteristics of the dataset being used for training.

In this work, LeCun-uniform Kernel Initializer is used in convolution layers during defining Inception Module and Dense Module and He-uniform Kernel Initializer is used in convolution layers during defining the overall model.

4.2.3 Optimizers Used

In this proposed work, 4 optimizers are used to check which optimizer gives the better results with the proposed Dense-INC model. Optimizers are: Adam, Adadelta, Adagrad, SGD with momentum. Optimizers in DL[3] are algorithms used to update the weights and biases of a NN[68] during training in order to minimize the loss function. The choice of optimizer can have a significant impact on the performance of the model, and there

are several different optimizers available to choose from. Here's a brief overview of each optimizer used in this work:

1. **Adam:** Adam (Adaptive Moment Estimation) optimizer is a GD optimization algorithm used in DL[3] for optimizing the parameters of a NN[68]. It is an extension of the SGD algorithm and was proposed by Kingma and Ba in their paper "Adam: A Method for Stochastic Optimization" in 2014[75].

Adam optimizer calculates adaptive learning rates for each parameter of the NN[68] using the first and second moments of the gradients. Specifically, it maintains two moving averages of the gradient: the first moment (mean) and the second moment (variance). These moments are calculated for each parameter during training and are used to adjust the learning rate of each parameter.

The update rule for the Adam optimizer can be expressed mathematically as follows:

- Calculate the gradient of the objective function with respect to the model parameters:

$$g_t = \nabla_{\theta} J(\theta_t)$$

where g_t is the gradient at time step t , θ_t is the model parameters at time step t , and $J(\theta_t)$ is the objective function at time step t .

- Update the exponential moving average of the first moment of the gradient:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

where m_t is the first moment estimate at time step t , β_1 is a hyperparameter that controls the decay rate of the moving average, and m_0 is initialized to 0.

- Update the exponential moving average of the second moment of the gradient:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

where v_t is the second moment estimate at time step t , β_2 is a hyperparameter that controls the decay rate of the moving average, and v_0 is initialized to 0.

- Compute the bias-corrected first and second moment estimates:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

where t is the current time step.

- Update the model parameters:

$$\theta_{t+1} = \theta_t - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

where α is the learning rate, ϵ is a small constant added for numerical stability, and $\sqrt{\cdot}$ denotes the square root operation.

The Adam optimizer has several advantages over the traditional GD algorithms such as faster convergence, better performance on sparse gradients, and robustness to noisy gradients. Additionally, it requires less memory and computation compared to other optimization algorithms such as Adagrad and RMSprop.

In summary, the Adam optimizer is a widely used optimization algorithm in DL[3] that efficiently updates the parameters of a NN[68] by calculating adaptive learning rates based on the first and second moments of the gradients.

2. **Adadelta:** Adadelta is a gradient-based optimization algorithm that is commonly used in DL[3] for updating the model's weights during training. It is an extension of the Adagrad optimizer and was introduced by Matthew Zeiler in his paper "ADADELTA: An Adaptive Learning Rate Method" in 2012[76].

The Adadelta algorithm is designed to address some of the drawbacks of other optimization methods such as Adagrad, which can suffer from a diminishing learning rate problem that can lead to slow convergence. Adadelta addresses this issue by adapting the learning rate[12] on a per-parameter basis and by keeping a moving estimate of the second moment of the gradient.

At each time step during training, Adadelta calculates a running average of the second moments of the gradients and uses this average to adjust the learning rate for each weight. The learning rate[12] is scaled by the ratio of the RMS value of the gradient to the RMS value of the parameter updates.

The update rule for Adadelta can be expressed mathematically as follows:

- Calculate the gradient of the objective function with respect to the model parameters:

$$g_t = \nabla_{\theta} J(\theta_t)$$

where g_t is the gradient at time step t , θ_t is the model parameters at time step t , and $J(\theta_t)$ is the objective function at time step t .

- Compute the exponential moving average of the squared gradients:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma)g_t^2$$

where $E[g^2]_t$ is the moving average of the squared gradients at time step t , γ is the decay rate, and $E[g^2]_0$ is initialized to 0.

- Compute the update:

$$\Delta\theta_t = -\frac{\text{RMS}[\Delta\theta]_{t-1}}{\text{RMS}[g]_t} g_t$$

where $\text{RMS}[g]_t = \sqrt{E[g^2]_t + \epsilon}$ is the root-mean-square (RMS) of the gradient at time step t , and $\text{RMS}[\Delta\theta]_{t-1} = \sqrt{E[\Delta\theta^2]_{t-1} + \epsilon}$ is the RMS of the update at time step $t - 1$.

- Compute the moving average of the squared updates:

$$E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1 - \gamma)\Delta\theta_t^2$$

Adadelta has several advantages over other optimization algorithms such as Adagrad and SGD, including faster convergence and more stable learning rates, which leads to

better generalization performance. Additionally, Adadelta does not require manual tuning of hyperparameters such as the learning rate or the momentum term.

In summary, Adadelta is a widely used optimization algorithm in DL[3] that adapts the learning rate[12] on a per-parameter basis using a moving average of the second moments of the gradients. This leads to faster convergence and more stable learning rates, making it a popular choice for training deep NNs[68].

3. **Adagrad:** Adagrad is a gradient-based optimization algorithm that is commonly used in DL[3] for updating the model’s weights during training. It was introduced by Duchi et al. in their paper ”Adaptive Subgradient Methods for Online Learning and Stochastic Optimization” in 2011[77].

The Adagrad algorithm is designed to adaptively adjust the learning rate of each weight in the network based on the historical gradient information. It does this by keeping a running sum of the squared gradients for each weight, which is then used to scale the learning rate for that weight.

The update rule for Adagrad can be expressed mathematically as follows:

- (a) Calculate the gradient of the objective function with respect to the model parameters:

$$g_t = \nabla_{\theta} J(\theta_t)$$

where g_t is the gradient at time step t , θ_t is the model parameters at time step t , and $J(\theta_t)$ is the objective function at time step t .

- (b) Compute the sum of the squared gradients:

$$G_t = G_{t-1} + g_t^2$$

where G_t is the sum of the squared gradients at time step t , G_0 is initialized to 0.

- (c) Compute the update:

$$\Delta\theta_t = -\frac{\alpha}{\sqrt{G_t + \epsilon}} g_t$$

where α is the learning rate, ϵ is a small constant added for numerical stability, and $\sqrt{}$ denotes the square root operation.

Adagrad has several advantages over other optimization algorithms such as SGD and its variants, including faster convergence and more stable learning rates. However, it also has some drawbacks, such as a diminishing learning rate problem that can lead to slow convergence and difficulty in adapting to changes in the optimization landscape.

To address these drawbacks, several variants of Adagrad have been proposed, such as Adadelta and RMSProp, which adapt the learning rate[12] in a more sophisticated way and incorporate momentum to improve convergence speed. Nonetheless, Adagrad remains a popular choice for many DL[3] tasks, particularly in settings where the optimization landscape is relatively stable and the gradients are not too noisy.

4. **SGD with momentum:** SGD with momentum is an optimization algorithm that is commonly used in DL[3] to train NNs[68]. It was first introduced by Rumelhart et al. in their classic book "Parallel Distributed Processing" in 1986[78].

The SGD with momentum algorithm works by adding a momentum term to the standard SGD update rule. The momentum term is a running average of the past gradients, which helps to smooth out the updates and prevent oscillations in the optimization process.

The update rule for SGD with momentum can be expressed mathematically as follows:

- (a) Calculate the gradient of the objective function with respect to the model parameters:

$$g_t = \nabla_{\theta} J(\theta_t)$$

where g_t is the gradient at time step t , θ_t is the model parameters at time step t , and $J(\theta_t)$ is the objective function at time step t .

- (b) Compute the momentum:

$$v_t = \gamma v_{t-1} + \alpha g_t$$

where v_t is the momentum at time step t , γ is the momentum coefficient, and v_0 is initialized to 0.

- (c) Update the model parameters:

$$\theta_{t+1} = \theta_t - v_t$$

The SGD with momentum algorithm has several advantages over standard SGD, including faster convergence and improved generalization performance. By smoothing out the updates and reducing oscillations, SGD with momentum can find better optima and escape from local minima more easily. It can also help to avoid getting stuck in plateaus or saddle points, which are common in high-dimensional optimization problems.

Overall, SGD with momentum is a popular choice for training deep NNs[68] and has been widely used in many state-of-the-art models in computer vision, NLP[67], and other domains.

Chapter 5

DATA EVALUATION

For using the suggested algorithm, we have employed 2 datasets which are called Plant Pathology 2020: FGCV7[7] and Plant Pathology 2021: FGCV8[8]. We utilized python 3.0 on an Intel i5 9th-generation CPU with 12 GB memory and NVIDIA K80 GPU with 12GB memory to acquire the experimental results for our suggested technique. Now, we have reviewed the comprehensive description of datasets that we have compared in the following section.

5.1 Datasets Used

In this part, we have done an overview of the details of the datasets. We have given the dataset to detect a particular disease in an apple plant. We utilized the 2 genuine datasets in Table 2: Plant Pathology 2020: FGCV7 and Plant Pathology 2021:FGCV8[8]. Now, we have discussed the structure and configuration of all datasets in Table 5.1.

Dataset Name	Training Images	Testing Images	Total Images
Plant Pathology 2020: FGCV7[7]	14,707	4,008	18,715
Plant Pathology 2021:FGCV8[8]	14,906	3,726	18,632

Table 5.1: Distribution of Datasets

1. Plant Pathology 2020: FGCV7

The Plant Pathology 2020 - FGVC7[7] dataset is a subset of images of leaves with different diseases such as apple scab, cedar apple rust, and healthy leaves. It was made available to the Kaggle community for the 'Plant Pathology Challenge'; part of the Fine-Grained Visual Categorization (FGVC) workshop at CVPR 2020 (Computer Vision and Pattern Recognition). The objective of this challenge was to train a model using images of training dataset to accurately classify a given image from testing dataset into different diseased category or a healthy leaf.

The distribution of these categories in the dataset is as follows:

- Healthy: 5162 images (27.6%)
- Multiple Diseases: 1420 images (7.6%)
- Rust: 3166 images (16.9%)



Figure 5.1: Sample Images of Plant Pathology 2020-FGCV7 Dataset[7]

The sample images of Plant Pathology 2020-FGCV7[7] dataset are given in Fig. 5.1. Note that some images may have more than one label if the leaf shows symptoms of multiple diseases or conditions. The "Other" category refers to images that were not classified as any of the four main categories, and it includes images with unclear or ambiguous labels.

It is important to note that the distribution of classes in the dataset is imbalanced, with the "Healthy" class being the most represented and the "Multiple Diseases" class being the least represented. This can make it challenging to train machine learning models that are accurate across all classes, and researchers and developers may need to use techniques such as data augmentation and class balancing to improve model performance.

2. Plant Pathology 2021: FGCV8

Plant Pathology 2021 - FGVC8[8] is a Kaggle competition launched on March 15, 2021, and closed on May 27, 2021. The competition was aimed at developing a model that can accurately classify a given image of a plant leaf into different diseased categories or a healthy leaf. The dataset provided for this competition consists of 18632 labeled apple tree leaf images.

The distribution of these categories in the dataset is as follows:

- Healthy: 4826 images (25.9%)
- Complex: 821 images (4.4%)
- Rust: 6226 images (33.4%)
- Powdery Mildew: 4759 images (25.5%)

The sample images of Plant Pathology 2021-FGCV8[8] dataset are given in Fig. 5.2.



Figure 5.2: Sample Images of Plant Pathology 2021-FGCV8 Dataset[8]

Note that some images may have more than one label if the leaf shows symptoms of multiple diseases or conditions.

It is important to note that the distribution of classes in the dataset is also imbalanced, with the "Complex" class being the least represented. This can make it challenging to train machine learning models that are accurate across all classes, and researchers and developers may need to use techniques such as data augmentation and class balancing to improve model performance.

Chapter 6

EXPERIMENTS AND RESULTS

In this chapter, I'm going to discuss about the performance of the proposed model. Here, two datasets are used in the evaluation process of the proposed model. The details of these datasets are present in the Data Evaluation section. Here in this work, the proposed model has been trained with 4 optimizers and compared the results with each other. Four optimizers used: Adam, Adadelta, Adagrad and SGD with momentum. These optimizers are explained in the Chapter: Proposed Work. Learning rate[12] set to 0.01 in each of the optimizers. And also compared the results with two CNN-based models which are already proposed.

6.1 Performance Metrics

Here, 3 performance metrics are used to compare the results: Accuracy, Precision and Recall. These are the metrics used to evaluate the performance of a classification model, including CNN-based models for plant disease detection.

1. **Accuracy:** Accuracy refers to the degree to which a measurement, calculation, or prediction is correct or exact. It is usually expressed as a percentage or a ratio that represents the number of correct predictions or measurements divided by the total number of predictions or measurements made. In other words, accuracy measures how well a model or a system is performing in terms of its ability to correctly identify or classify data.

For example, if a model correctly classifies 90 out of 100 images, its accuracy would be 90%. However, it's important to note that accuracy alone may not always be the most meaningful metric, especially in cases where the distribution of data is imbalanced or when certain types of errors are more costly than others. In such cases, other metrics such as precision, recall may be more appropriate to evaluate the performance of a model or a system.

$$Accuracy = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \quad (6.1)$$

2. **Precision:** Precision is a metric used to evaluate the performance of a classification model or a system in terms of its ability to make precise predictions. It is defined as the ratio of true positive predictions to the total number of positive predictions made by the model. In other words, precision measures the proportion of the positive predictions that are actually correct.

For example, suppose a binary classification model predicts that out of 100 samples, 80 are positive and 20 are negative. If the model correctly identifies 70 of the positive samples as positive and misclassifies 10 of them as negative, then the precision of the model would be $70/(70+10) = 0.875$ or 87.5%.

$$Precision = \frac{\text{Number of Correctly Predicted Positive Instances}}{\text{Number of Total Positive Predictions you made}} \quad (6.2)$$

3. **Recall:** Recall, also known as sensitivity or true positive rate, is a metric used to evaluate the performance of a classification model or a system in terms of its ability to correctly identify positive samples. It is defined as the ratio of true positive predictions to the total number of actual positive samples in the data. In other words, recall measures the proportion of the positive samples that are correctly identified by the model.

For example, suppose a binary classification model predicts that out of 100 samples, 80 are positive and 20 are negative. If the model correctly identifies 70 of the positive samples as positive and misses 10 of them, then the recall of the model would be $70/(70+10) = 0.875$ or 87.5%.

$$Recall = \frac{\text{Number of Correctly Predicted Positive Instances}}{\text{Number of Total Positive Instances in a Dataset}} \quad (6.3)$$

In the context of plant disease detection, accuracy, precision, and recall metrics can be used to evaluate the performance of the CNN-based model in identifying healthy and diseased plants, as well as in identifying the specific disease affecting the plant. These metrics can help in fine-tuning the model and improving its performance by identifying the areas where the model is performing poorly.

6.2 Analysis and Visualization of the Experimental Result

In this section, I'm going to discuss about the performance of the proposed method on the two datasets which is discussed in the Data Evaluation Chapter. Here, we use Accuracy, Precision and Recall to evaluate the performance of the proposed model.

Now, we deployed the provided method on the supplied two datasets in which we have compared Accuracy, Precision and Recall with the 4 optimizers to check which optimizer gives the better results with the proposed Dense-INC model. Optimizers are: Adam, Adadelta, Adagrad, SGD with momentum. These optimizers are discussed in the Proposed Work Chapter.

6.2.1 Performance on Plant Pathology 2020- FCVG7 dataset

1. Using Adam Optimizer

On this dataset, we ran our model for 50 epochs with Adam Optimizer, and got the highest accuracy of 0.8870, precision of 0.9009, recall of 0.8759 and the lowest loss of 0.3208. All the performance metrics for the course are as shown in Fig. 6.1.

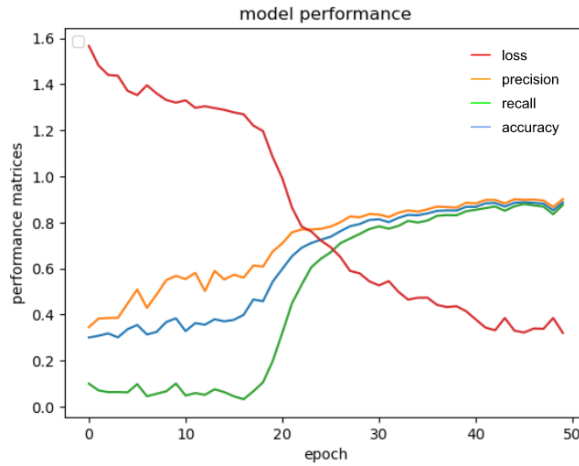


Figure 6.1: Performance Evaluation of the Proposed model using Adam Optimizer on Plant Pathology 2020-FCVG7 Dataset

2. Using Adadelata Optimizer

On this dataset, we ran our model for 50 epochs with Adadelata Optimizer, and got the highest accuracy of 0.6261, precision of 0.7275, recall of 0.4591 and the lowest loss of 0.9249. All the performance metrics for the course are as shown in Fig. 6.2.

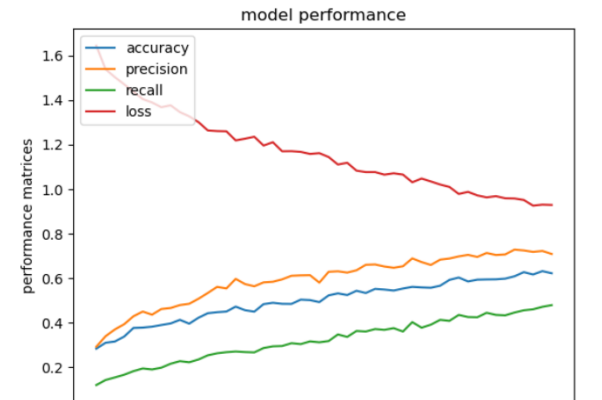


Figure 6.2: Performance Evaluation of the Proposed model using Adadelata Optimizer on Plant Pathology 2020-FCVG7 Dataset

3. Using Adagrad Optimizer

On this dataset, we ran our model for 50 epochs with Adagrad Optimizer, and got the highest accuracy of 0.9082, precision of 0.9234, recall of 0.8885 and the lowest loss of 0.2758. All the performance metrics for the course are as shown in Fig. 6.3.

4. Using SGD with momentum Optimizer

On this dataset, we ran our model for 50 epochs with SGD with momentum Optimizer, and got the highest accuracy of 0.8537, precision of 0.8871, recall of 0.8169 and the lowest loss of 0.4186. All the performance metrics for the course are as shown in Fig. 6.4.

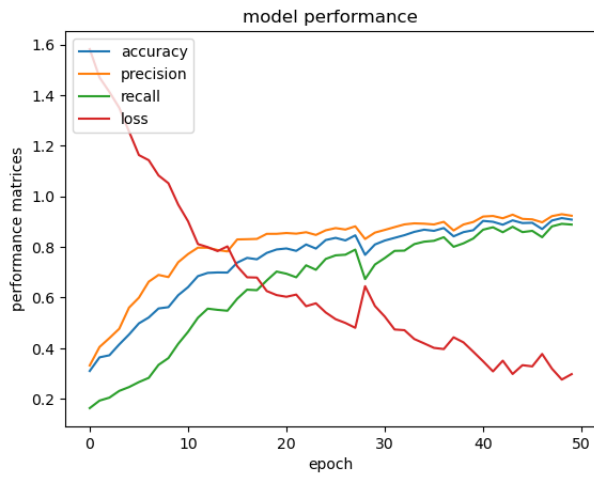


Figure 6.3: Performace Evaluation of the Proposed model using Adagrad Optimizer on Plant Pathology 2020-FCVG7 Dataset

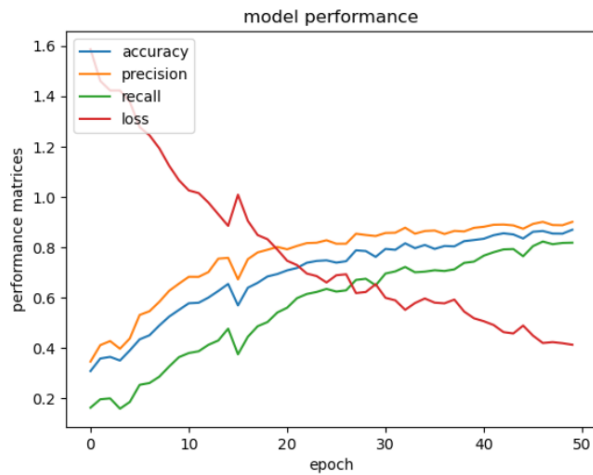


Figure 6.4: Performace Evaluation of the Proposed model using SGD with momentum Optimizer on Plant Pathology 2020-FCVG7 Dataset

Summarization of the results is given in Table 6.1.

In Table 6.1, It is shown that Adagrad optimizer gives better results with a loss of 0.27 and Adadelata optimizer gives worst results with a loss of 0.92.

Optimizers	Accuracy	Precision	Recall
Adam (in %)	88.70%	90.09%	87.59%
Adadelta (in %)	62.61%	72.75%	45.91%
Adagrad (in %)	90.82%	92.34%	88.85%
SGD with momentum (in %)	85.37%	88.71%	81.69%

Table 6.1: Performance Evaluation of Plant Pathology 2020: FGCv7 dataset

6.2.2 Performance on Plant Pathology 2021- FCvG8 dataset

1. Using Adam Optimizer

On this dataset, we ran our model for 50 epochs with Adam Optimizer, and got the highest accuracy of 0.8895, precision of 0.9023, recall of 0.8829 and the lowest loss of 0.3143. All the performance metrics for the course are as shown in Fig. 6.5.

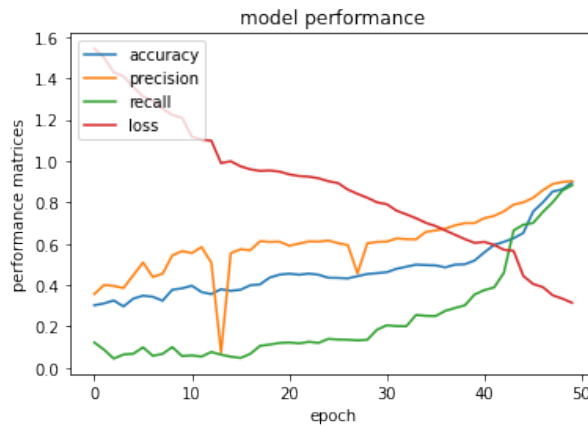


Figure 6.5: Performace Evaluation of the Proposed model using Adam Optimizer on Plant Pathology 2021-FCvG8 Dataset

2. Using Adadelta Optimizer

On this dataset, we ran our model for 50 epochs with Adadelta Optimizer, and got the highest accuracy of 0.6356, precision of 0.7123, recall of 0.5029 and the lowest loss of 0.645. All the performance metrics for the course are as shown in Fig. 6.6.

3. Using Adagrad Optimizer

On this dataset, we ran our model for 50 epochs with Adagrad Optimizer, and got the highest accuracy of 0.9256, precision of 0.9523, recall of 0.9029 and the lowest loss of 0.2543. All the performance metrics for the course are as shown in Fig. 6.7.

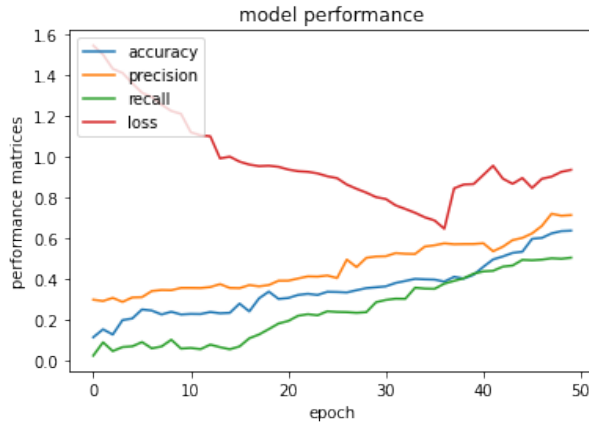


Figure 6.6: Performace Evaluation of the Proposed model using Adadelta Optimizer on Plant Pathology 2021-FCVG8 Dataset

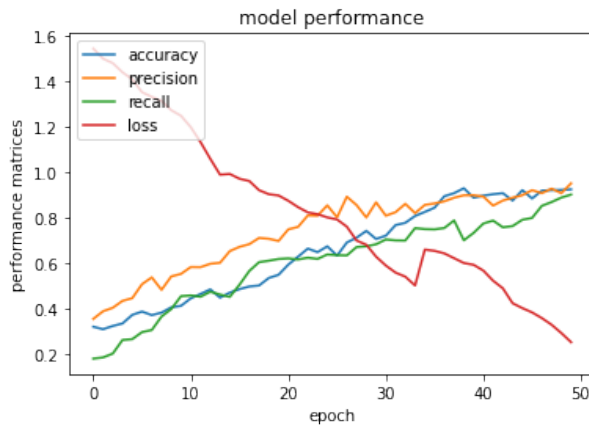


Figure 6.7: Performace Evaluation of the Proposed model using Adagrad Optimizer on Plant Pathology 2021-FCVG8 Dataset

4. Using SGD with momentum Optimizer

On this dataset, we ran our model for 50 epochs with SGD with momentum Optimizer, and got the highest accuracy of 0.8656, precision of 0.9023, recall of 0.8176 and the lowest loss of 0.3502. All the performance metrics for the course are as shown in Fig. 6.8.

Summarization of the results is given in Table 6.2.

In Table 6.2, It is shown that Adagrad optimizer gives better results with a loss of 0.25 and Adadelta optimizer gives worst results with a loss of 0.64.

In Table 6.2, It is shown that Adagrad optimizer gives better results with a loss of 0.25 and Adadelta optimizer gives the worst results with a loss of 0.64.

According to Table 6.1 and 6.2, we can say that the proposed model gives better results with the Plant Pathology 2021- FGCV8[8] dataset as compared to Plant Pathology 2020-FGCV7[7] dataset. And the proposed model trained using Adagrad Optimizer in both the datasets.

So to further analyze the result, the proposed model has been compared with two

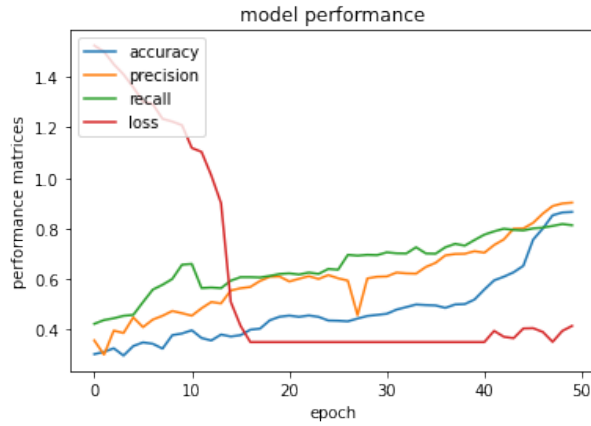


Figure 6.8: Performance Evaluation of the Proposed model using SGD with momentum Optimizer on Plant Pathology 2021-FCVG8 Dataset

Optimizers	Accuracy	Precision	Recall
Adam (in %)	88.95%	90.23%	88.29%
Adadelta (in %)	63.56%	71.23%	50.29%
Adagrad (in %)	92.56%	95.23%	90.29%
SGD with momentum (in %)	86.56%	90.23%	81.76%

Table 6.2: Performance Evaluation of Plant Pathology 2021: FGCV8 dataset

CNN based models which are already suggested: Model1[79] and Model2[17]. These two models have been trained with the Adagrad Optimizer and trained on Plant Pathology 2021- FGCV8[8] dataset.

Results are given in Fig. 6.9.

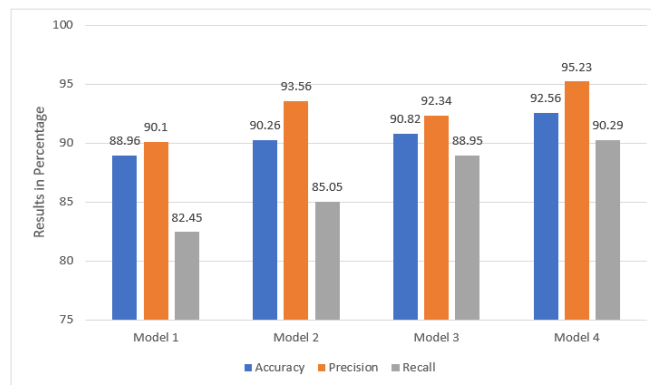


Figure 6.9: Comparison of Performance using Adagrad Optimizer

In Fig. 6.9, as it's already mentioned Model1[79] and Model2[17] are the CNN-based models and Model3 is the proposed model, Dense-INC model which is trained on Plant Pathology 2020- FGCV7[7] dataset and Model4 is the proposed model, Dense-INC model which is trained on Plant Pathology 2021- FGCV8[8] dataset. All the Models are trained using Adagrad Optimizer.

So, I can conclude that Model4 which is the proposed Dense-INC model which gives better results as compared to other models with the accuracy of 92.56%, Precision of 95.23% and Recall of 90.29%.

Chapter 7

CONCLUSION AND FUTURE SCOPE

In recent years, the use of CNNs for plant disease detection has gained increasing attention in the field of agriculture. In this report, we have explored the use of a CNN-based model for the detection of plant diseases.

The results of our study demonstrate that the proposed Dense-INC model is effective in detecting plant diseases, with high levels of accuracy achieved in the classification of healthy and diseased plants, as well as in identifying the specific disease affecting the plant. The model was trained on a large dataset of plant images, which allowed it to learn the features and patterns associated with different plant diseases. The performance of the model was evaluated using various metrics such as accuracy, precision and recall, and the results showed that the proposed model achieved high levels of accuracy in all metrics.

One of the main advantages of the Dense-INC model is its ability to learn features directly from the image data, which eliminates the need for hand-crafted features. Moreover, the use of deep learning techniques, such as CNNs, allows for the analysis of complex data sets with high dimensional inputs, such as plant images. Additionally, the model can be used for real-time detection of plant diseases, which can help farmers take preventive measures before the disease spreads.

However, there are some limitations to the use of the Dense-INC model for plant disease detection. One of the main challenges is the lack of large and diverse datasets. Collecting and labeling a large dataset of plant images can be time-consuming and expensive, especially for rare diseases or in regions where access to plant samples is limited. Moreover, some diseases can have similar symptoms, making it difficult for the model to differentiate between them.

In general, it can be said that the use of the model for plant disease detection shows great potential for improving the accuracy and efficiency of plant disease detection. The development of such models can provide significant benefits to the agricultural industry, by reducing the risk of crop losses and improving food security.

The use of CNNs for plant disease detection has shown promising results in recent years. However, there is still a lot of scope for future research to improve the accuracy and efficiency of the Dense-INC model for plant disease detection.

In this report, we discuss some of the potential areas for future research.

- **Improving the dataset:** One of the main limitations of CNN-based models for plant disease detection is the lack of large and diverse datasets. Future research can focus on creating larger and more diverse datasets, especially for rare diseases or in regions where access to plant samples is limited. Moreover, the dataset can be

augmented by applying various techniques such as rotation, flipping, and zooming to increase the diversity of the dataset.

- **Improving the dataset:** One of the main limitations of CNN-based models for plant disease detection is the lack of large and diverse datasets. Future research can focus on creating larger and more diverse datasets, especially for rare diseases or in regions where access to plant samples is limited. Moreover, the dataset can be augmented by applying various techniques such as rotation, flipping, and zooming to increase the diversity of the dataset.
- **Exploring transfer learning:** Transfer learning involves fine-tuning a pre-trained CNN on a smaller dataset of plant images. This approach can overcome the limitations of small datasets and improve the accuracy of the model. Future research can explore the use of transfer learning for plant disease detection and evaluate its performance.
- **Developing mobile applications:** The development of user-friendly mobile applications based on CNN-based models can help farmers detect plant diseases quickly and accurately. Future research can focus on developing such applications that are easy to use, accessible, and provide real-time information to the farmers.
- **Exploring the use of multiple modalities:** CNN-based models can be trained on multiple modalities such as visible light images, hyperspectral images, and thermal images. Future research can explore the use of multiple modalities to improve the accuracy of the model.
- **Incorporating contextual information:** Incorporating contextual information: Plant diseases can be affected by various environmental factors such as temperature, humidity, and soil conditions. Future research can explore the incorporation of contextual information into the CNN-based models to improve the accuracy of the model.
- **Explainability and interpretability:** CNN-based models are often considered black boxes, and it is challenging to explain the reasoning behind the model's predictions. Future research can focus on developing methods for explaining the model's predictions and making the model more interpretable.

In conclusion, there is still a lot of scope for future research to improve the accuracy and efficiency of Dense-INC model for plant disease detection. The development of such models can provide significant benefits to the agricultural industry by reducing the risk of crop losses and improving food security. Future research can focus on improving the dataset, exploring transfer learning, developing mobile applications, incorporating multiple modalities, and improving the explainability and interpretability of the models.

Bibliography

- [1] K. V. Flannery, “The Origins of Agriculture,” <https://doi.org/10.1146/annurev.an.02.100173.001415>, vol. 2, no. 1, pp. 271–310, Nov. 2003, doi: 10.1146/ANNUREV.AN.02.100173.001415.
- [2] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science* (1979), vol. 349, no. 6245, pp. 255–260, Jul. 2015, doi: 10.1126/SCIENCE.AAA8415.
- [3] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553. Nature Publishing Group, pp. 436–444, May 27, 2015. doi: 10.1038/nature14539.
- [4] “Apple: Diseases and Symptoms — Vikaspedia.” <https://vikaspedia.in/agriculture/crop-production/integrated-pest-managment/ipm-for-fruit-crops/ipm-strategies-for-apple/apple-diseases-and-symptoms> (accessed Apr. 06, 2023).
- [5] J. Mccarthy, “WHAT IS ARTIFICIAL INTELLIGENCE?,” 2007, Accessed: Mar. 31, 2023. [Online]. Available: <http://www-formal.stanford.edu/jmc/>
- [6] “plant disease - Symptoms and signs — Britannica.” <https://www.britannica.com/science/plant-disease/Symptoms-and-signs> (accessed May 25, 2022).
- [7] “Plant Pathology 2020 - FGVC7 — Kaggle.” <https://www.kaggle.com/c/plant-pathology-2020-fgvc7/overview> (accessed Mar. 31, 2023).
- [8] “Plant Pathology 2021 - FGVC8 — Kaggle.” <https://www.kaggle.com/c/plant-pathology-2021-fgvc8> (accessed Dec. 14, 2022).
- [9] “[PDF] The Plant Pathology 2020 challenge dataset to classify foliar disease of apples — Semantic Scholar.” <https://www.semanticscholar.org/paper/The-Plant-Pathology-2020-challenge-dataset-to-of-Thapa-Snavely/17005a1bd4189707f17d8bef9a0909c9399f7171> (accessed Mar. 18, 2023).
- [10] Y. Zhu and S. Newsam, “DenseNet for dense flow,” *Proceedings - International Conference on Image Processing, ICIP*, vol. 2017-September, pp. 790–794, Feb. 2018, doi: 10.1109/ICIP.2017.8296389.
- [11] P. Hao, J. H. Zhai, and S. F. Zhang, “A simple and effective method for image classification,” *Proceedings of 2017 International Conference on Machine Learning and Cybernetics, ICMLC 2017*, vol. 1, pp. 230–235, Nov. 2017, doi: 10.1109/ICMLC.2017.8107769.

- [12] “Learning rate - Wikipedia.” https://en.wikipedia.org/wiki/Learning_rate (accessed Apr. 06, 2023).
- [13] G. Geetharamani and A. P. J., “Identification of plant leaf diseases using a nine-layer deep convolutional neural network,” *Computers and Electrical Engineering*, vol. 76, pp. 323–338, Jun. 2019, doi: 10.1016/j.compeleceng.2019.04.011.
- [14] “GitHub - spMohanty/PlantVillage-Dataset: Dataset of diseased plant leaf images and corresponding labels.” <https://github.com/spMohanty/PlantVillage-Dataset> (accessed May 25, 2022).
- [15] S. Panigrahi, A. Nanda, and T. Swarnkar, “A Survey on Transfer Learning,” *Smart Innovation, Systems and Technologies*, vol. 194, no. 10, pp. 781–789, 2021, doi: 10.1007/978-981-15-5971-6_83.
- [16] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010. doi: 10.1109/TKDE.2009.191.
- [17] P. Jiang, Y. Chen, B. Liu, D. He, and C. Liang, “Real-Time Detection of Apple Leaf Diseases Using Deep Learning Approach Based on Improved Convolutional Neural Networks,” *IEEE Access*, vol. 7, pp. 59069–59080, 2019, doi: 10.1109/ACCESS.2019.2914929.
- [18] “Deep Learning: GoogLeNet Explained — by Richmond Alake — Towards Data Science.” <https://towardsdatascience.com/deep-learning-googlenet-explained-de8861c82765> (accessed May 26, 2022).
- [19] “PlantPathology Apple Dataset — Kaggle.” <https://www.kaggle.com/datasets/piantic/plantpathology-apple-dataset> (accessed Apr. 04, 2023).
- [20] A. Adedoja, P. A. Owolawi, and T. Mapayi, “Deep learning based on NASNet for plant disease recognition using leave images,” *icABCD 2019 - 2nd International Conference on Advances in Big Data, Computing and Data Communication Systems*, Aug. 2019, doi: 10.1109/ICABCD.2019.8851029.
- [21] “Review: NASNet — Neural Architecture Search Network (Image Classification) — by Sik-Ho Tsang — Medium.” <https://sh-tsang.medium.com/review-nasnet-neural-architecture-search-network-image-classification-23139ea0425d> (accessed Apr. 06, 2023).
- [22] R. G. De Luna, E. P. Dadios, and A. A. Bandala, “Automated Image Capturing System for Deep Learning-based Tomato Plant Leaf Disease Detection and Recognition,” *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, vol. 2018-October, pp. 1414–1419, Feb. 2019, doi: 10.1109/TENCON.2018.8650088.
- [23] “Faster R-CNN Explained for Object Detection Tasks — Paperspace Blog.” <https://blog.paperspace.com/faster-r-cnn-explained-object-detection/> (accessed Apr. 06, 2023).

- [24] J. Chen, J. Chen, D. Zhang, Y. Sun, and Y. A. Nanehkaran, “Using deep transfer learning for image-based plant disease identification,” *Comput Electron Agric*, vol. 173, p. 105393, Jun. 2020, doi: 10.1016/J.COMPAG.2020.105393.
- [25] S. Mascarenhas and M. Agarwal, “A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification,” *Proceedings of IEEE International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications, CENTCON 2021*, pp. 96–99, 2021, doi: 10.1109/CENTCON52345.2021.9687944.
- [26] “Inception V3 CNN Architecture Explained . — by Anas BRITAL — Medium.” <https://medium.com/@AnasBrital98/inception-v3-cnn-architecture-explained-691cfb7bba08> (accessed Apr. 06, 2023).
- [27] M. Wang, S. Lu, D. Zhu, J. Lin, and Z. Wang, “A High-Speed and Low-Complexity Architecture for Softmax Function in Deep Learning,” *2018 IEEE Asia Pacific Conference on Circuits and Systems, APCCAS 2018*, pp. 223–226, Jan. 2019, doi: 10.1109/APCCAS.2018.8605654.
- [28] V. Tiwari, R. C. Joshi, and M. K. Dutta, “Dense convolutional neural networks based multiclass plant disease detection and classification using leaf images,” *Ecol Inform*, vol. 63, no. March, p. 101289, 2021, doi: 10.1016/j.ecoinf.2021.101289.
- [29] F. Jiang, Y. Lu, Y. Chen, D. Cai, and G. Li, “Image recognition of four rice leaf diseases based on deep learning and support vector machine,” *Comput Electron Agric*, vol. 179, Dec. 2020, doi: 10.1016/J.COMPAG.2020.105824.
- [30] D. A. Pisner and D. M. Schnyer, *Support vector machine*. Elsevier Inc., 2019. doi: 10.1016/B978-0-12-815739-8.00006-7.
- [31] D. Demirović, “An Implementation of the Mean Shift Algorithm,” *Image Processing On Line*, vol. 9, pp. 251–268, Sep. 2019, doi: 10.5201/IPOL.2019.255.
- [32] U. P. Singh, S. S. Chouhan, S. Jain, and S. Jain, “Multilayer Convolution Neural Network for the Classification of Mango Leaves Infected by Anthracnose Disease,” *IEEE Access*, vol. 7, pp. 43721–43729, 2019, doi: 10.1109/ACCESS.2019.2907383.
- [33] M. Agarwal, A. Singh, S. Arjaria, A. Sinha, and S. Gupta, “ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network,” *Procedia Comput Sci*, vol. 167, no. 2019, pp. 293–301, 2020, doi: 10.1016/j.procs.2020.03.225.
- [34] A. Smetanin, A. Uzhinskiy, G. Ososkov, P. Goncharov, and A. Nechaevskiy, “Deep learning methods for the plant disease detection platform,” *AIP Conf Proc*, vol. 2377, no. October, 2021, doi: 10.1063/5.0068797.
- [35] P. Bedi and P. Gole, “Plant disease detection using hybrid model based on convolutional autoencoder and convolutional neural network,” *Artificial Intelligence in Agriculture*, vol. 5, pp. 90–101, 2021, doi: 10.1016/j.aiia.2021.05.002.
- [36] Y. Zhang, “A Better Autoencoder for Image: Convolutional Autoencoder”.

- [37] M. Brahim, M. Arsenovic, S. Laraba, S. Sladojevic, K. Boukhalifa, and A. Moussaoui, “Deep Learning for Plant Diseases: Detection and Saliency Map Visualisation,” no. June, pp. 93–117, 2018, doi: 10.1007/978-3-319-90403-0_6.
- [38] M. Z. Alom et al., “The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches,” 2018, [Online]. Available: <http://arxiv.org/abs/1803.01164>
- [39] G. Sachdeva, P. Singh, and P. Kaur, “Plant leaf disease classification using deep Convolutional neural network with Bayesian learning,” *Mater Today Proc*, vol. 45, pp. 5584–5590, 2021, doi: 10.1016/j.matpr.2021.02.312.
- [40] Read, “Bayesian Learning”.
- [41] P. Tm, A. Pranathi, K. Saiashritha, N. B. Chittaragi, and S. G. Koolagudi, “Tomato Leaf Disease Detection Using Convolutional Neural Networks,” 2018 11th International Conference on Contemporary Computing, IC3 2018, Nov. 2018, doi: 10.1109/IC3.2018.8530532.
- [42] “LeNet - Wikipedia.” <https://en.wikipedia.org/wiki/LeNet> (accessed Apr. 06, 2023).
- [43] Y. LeCun et al., “Handwritten digit recognition with a back-propagation network,” *Adv Neural Inf Process Syst*, vol. 2, pp. 396–404, 1990, Accessed: Apr. 06, 2023. [Online]. Available: <http://yann.lecun.com/exdb/publis/pdf/lecun-90c.pdf>
- [44] V. Suma, R. A. Shetty, R. F. Tated, S. Rohan, and T. S. Pujar, “CNN based Leaf Disease Identification and Remedy Recommendation System,” *Proceedings of the 3rd International Conference on Electronics and Communication and Aerospace Technology, ICECA 2019*, pp. 395–399, Jun. 2019, doi: 10.1109/ICECA.2019.8821872.
- [45] M. F. A. Hady and F. Schwenker, “Semi-supervised Learning,” *Intelligent Systems Reference Library*, vol. 49, pp. 215–239, 2013, doi: 10.1007/978-3-642-36657-4_7/COVER.
- [46] P. Soni and R. Chahar, “A segmentation improved robust PNN model for disease identification in different leaf images,” *1st IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems, ICPEICES 2016*, Feb. 2017, doi: 10.1109/ICPEICES.2016.7853301.
- [47] D. F. Specht, “Probabilistic neural networks,” *Neural Networks*, vol. 3, no. 1, pp. 109–118, Jan. 1990, doi: 10.1016/0893-6080(90)90049-Q.
- [48] Y. Kawasaki, H. Uga, S. Kagiwada, and H. Iyatomi, “Basic study of automated diagnosis of viral plant diseases using convolutional neural networks,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9475, pp. 638–645, 2015, doi: 10.1007/978-3-319-27863-6_59/COVER.
- [49] S. P. Mohanty, D. P. Hughes, and M. Salathé, “Using deep learning for image-based plant disease detection,” *Front Plant Sci*, vol. 7, no. September, p. 1419, Sep. 2016, doi: 10.3389/FPLS.2016.01419/BIBTEX.

- [50] A. S. Keceli, A. Kaya, C. Catal, and B. Tekinerdogan, “Deep learning-based multi-task prediction system for plant disease and species detection,” *Ecol Inform*, vol. 69, p. 101679, Jul. 2022, doi: 10.1016/J.ECOINF.2022.101679.
- [51] A. O. Anim-Ayeko, C. Schillaci, and A. Lipani, “Automatic blight disease detection in potato (*Solanum tuberosum* L.) and tomato (*Solanum lycopersicum*, L. 1753) plants using deep learning,” *Smart Agricultural Technology*, vol. 4, p. 100178, Aug. 2023, doi: 10.1016/J.ATECH.2023.100178.
- [52] A. V. Panchal, S. C. Patel, K. Bagyalakshmi, P. Kumar, I. R. Khan, and M. Soni, “Image-based Plant Diseases Detection using Deep Learning,” *Mater Today Proc*, Aug. 2021, doi: 10.1016/J.MATPR.2021.07.281.
- [53] M. H. Saleem, J. Potgieter, and K. M. Arif, “A Performance-Optimized Deep Learning-Based Plant Disease Detection Approach for Horticultural Crops of New Zealand,” *IEEE Access*, vol. 10, pp. 89798–89822, 2022, doi: 10.1109/ACCESS.2022.3201104.
- [54] C. K. Sunil, C. D. Jaidhar, and N. Patil, “Cardamom Plant Disease Detection Approach Using EfficientNetV2,” *IEEE Access*, vol. 10, pp. 789–804, 2022, doi: 10.1109/ACCESS.2021.3138920.
- [55] M. Tan and Q. V Le, “EfficientNetV2: Smaller Models and Faster Training.” PMLR, pp. 10096–10106, Jul. 01, 2021. Accessed: Apr. 04, 2023. [Online]. Available: <https://proceedings.mlr.press/v139/tan21a.html>
- [56] H. Amin, A. Darwish, A. E. Hassanien, and M. Soliman, “End-to-End Deep Learning Model for Corn Leaf Disease Classification,” *IEEE Access*, vol. 10, pp. 31103–31115, 2022, doi: 10.1109/ACCESS.2022.3159678.
- [57] V. K. Vishnoi, K. Kumar, B. Kumar, S. Mohan, and A. A. Khan, “Detection of Apple Plant Diseases Using Leaf Images Through Convolutional Neural Network,” *IEEE Access*, vol. 11, pp. 6594–6609, 2023, doi: 10.1109/ACCESS.2022.3232917.
- [58] “Keras: Deep Learning for humans.” <https://keras.io/> (accessed Mar. 31, 2023).
- [59] “TensorFlow.” https://www.tensorflow.org/gclid=Cj0KCQjwiZqhBhCJARIsACHHEH_bXjIpYxfmoPdyXU66tPd3k61RZuKyqpRwSuKvD1G8NxXkgugHZcsaAnDGEALw_wcB (accessed Mar. 31, 2023).
- [60] “This figure illustrates the five plants and the diseases that each of... — Download Scientific Diagram.” https://www.researchgate.net/figure/This-figure-illustrates-the-five-plants-and-the-diseases-that-each-of-them-carries_fig3_357495598 (accessed Apr. 05, 2023).
- [61] “Healthy apple leaves and five common disease types: (a) healthy leaves;... — Download Scientific Diagram.” https://www.researchgate.net/figure/Healthy-apple-leaves-and-five-common-disease-types-a-healthy-leaves-b-Alternaria_fig1_352247462 (accessed Apr. 03, 2023).

- [62] A. K. Jain, J. Mao, and K. M. Mohiuddin, “Artificial neural networks: A tutorial,” *Computer* (Long Beach Calif), vol. 29, no. 3, pp. 31–44, Mar. 1996, doi: 10.1109/2.485891.
- [63] “Artificial neural network - Wikipedia.” https://en.wikipedia.org/wiki/Artificial_neural_network (accessed Apr. 06, 2023).
- [64] P. Cunningham, M. Cord, and S. J. Delany, “Supervised learning,” *Cognitive Technologies*, pp. 21–49, 2008, doi: 10.1007/978-3-540-75171-7_2/COVER.
- [65] “The overall architecture of the Convolutional Neural Network (CNN)... — Download Scientific Diagram.” https://www.researchgate.net/figure/The-overall-architecture-of-the-Convolutional-Neural-Network-CNN-includes-an-input_fig4_331540139 (accessed Apr. 04, 2023).
- [66] R. A. Jarvis, “A Perspective on Range Finding Techniques for Computer Vision,” *IEEE Trans Pattern Anal Mach Intell*, vol. PAMI-5, no. 2, pp. 122–139, 1983, doi: 10.1109/TPAMI.1983.4767365.
- [67] K. R. Chowdhary, “Natural Language Processing,” *Fundamentals of Artificial Intelligence*, pp. 603–649, 2020, doi: 10.1007/978-81-322-3972-7_19.
- [68] C. M. Bishop, “Neural networks and their applications,” *Review of Scientific Instruments*, vol. 65, no. 6, p. 1803, Jun. 1998, doi: 10.1063/1.1144830.
- [69] “The architecture of Inception-V3 model. — Download Scientific Diagram.” https://www.researchgate.net/figure/The-architecture-of-Inception-V3-model_fig5_349717475 (accessed Mar. 17, 2023).
- [70] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, pp. 448–456, Feb. 2015, Accessed: Apr. 07, 2023. [Online]. Available: <https://arxiv.org/abs/1502.03167v3>
- [71] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning,” *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pp. 4278–4284, Feb. 2016, doi: 10.1609/aaai.v31i1.11231.
- [72] A. M. Fred Agarap, “Deep Learning using Rectified Linear Units (ReLU),” Mar. 2018, Accessed: Apr. 04, 2023. [Online]. Available: <https://arxiv.org/abs/1803.08375v2>
- [73] “Example of the DenseNet model. — Download Scientific Diagram.” https://www.researchgate.net/figure/Example-of-the-DenseNet-model_fig2_337919457 (accessed Mar. 17, 2023).
- [74] J. Zhong, C. Pun, and S. Member, “An End-to-End Dense-InceptionNet for Image Copy-Move Forgery Detection,” vol. 15, pp. 2134–2146, 2020.
- [75] D. P. Kingma and J. L. Ba, “Adam: A Method for Stochastic Optimization,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Dec. 2014, Accessed: Apr. 07, 2023. [Online]. Available: <https://arxiv.org/abs/1412.6980v9>

- [76] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” Dec. 2012, Accessed: Apr. 07, 2023. [Online]. Available: <https://arxiv.org/abs/1212.5701v1>
- [77] DuchiJohn, HazanElad, and SingerYoram, “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,” *The Journal of Machine Learning Research*, Jul. 2011, doi: 10.5555/1953048.2021068.
- [78] “Parallel Distributed Processing.” <https://mitpress.mit.edu/9780262680530/parallel-distributed-processing/> (accessed Apr. 07, 2023).
- [79] S. P. Singh, K. Pritamdas, K. J. Devi, and S. D. Devi, “Custom Convolutional Neural Network for Detection and Classification of Rice Plant Diseases,” *Procedia Comput Sci*, vol. 218, pp. 2026–2040, 2023, doi: 10.1016/j.procs.2023.01.179.

LIST OF PUBLICATIONS

1. Barsha Biswas, Rajesh Kumar Yadav, “A Review of Convolutional Neural Network-Based Approaches for Disease Detection in Plants”. Accepted and presented at the International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT 2023).
Indexed by Scopus.
Paper Id: IDCIoT172



2. Barsha Biswas, Rajesh Kumar Yadav, “Multilayer Convolutional Neural Network Based Approach to Detect Apple Foliar Disease”. Accepted and presented at the International Conference on Innovation in Technology (INOCON 2023).
Indexed by Scopus.
Paper Id: INOCON499



● **20% Overall Similarity**

Top sources found in the following databases:

- 11% Internet database
- 10% Publications database
- Crossref database
- Crossref Posted Content database
- 16% Submitted Works database

TOP SOURCES

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	dspace.dtu.ac.in:8080 Internet	3%
2	Barsha Biswas, Rajesh Kumar Yadav. "Multilayer Convolutional Neural ... Crossref	<1%
3	S. Kim, D.J. Park, D.E. Chang. "RAPIDO: a rejuvenating adaptive PID-typ... Crossref	<1%
4	doctorpenguin.com Internet	<1%
5	mdpi.com Internet	<1%
6	Anju Yadav, Udit Thakur, Rahul Saxena, Vipin Pal, Vikrant Bhateja, Jerry... Crossref	<1%
7	internationaljournalssrg.org Internet	<1%
8	"Intelligent Computing Theories and Application", Springer Science and... Crossref	<1%