A Dissertation (MECS - 752) on

A COMPARATIVE ANALYSIS OF DEEP LEARNING MODELS FOR PLANT DISEASE IDENTIFICATION

Submitted in partial fulfilment of the requirement for the degree of

Master of Technology

In

Computer Science and Engineering

Submitted by

Megha Gupta

(Enrollment No.: 00716404819)

Under the guidance of

Dr. Nupur Prakash

(Professor)

USICT, GGSIPU

UNIVERSITY SCHOOL OF INFORMATION, COMMUNICATION & TECHNOLOGY

GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY

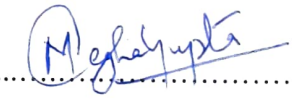
NEW DELHI – 110078

(2019 – 2021)

DECLARATION

This is to certify that the Dissertation (MECS - 752) entitled "**A Comparative Analysis of Deep Learning Models for Plant Disease Identification**" is a bonafide record of independent project work done by me under the supervision of **Prof. Nupur Prakash** and submitted to University School of Information, Communication & Technology, Guru Gobind Singh Indraprastha University in partial fulfilment for the award of the Degree of **Master of Technology** in **Computer science and Engineering**.

Dated: 28.6.2021



Megha Gupta

Enrollment no: 00716404819

CERTIFICATE

This is to certify that the Dissertation (MECS - 752) entitled "**A Comparative Analysis of Deep Learning Models for Plant Disease Identification**" is a bonafide record of independent project work done by "Megha Gupta" bearing enrollment number 00716404819 under my guidance and supervision.

To the best of my knowledge and believe work done by candidate has not been submitted for award of any other degree.

Dated: 28.6.2021

.....

Dr. Nupur Prakash
Professor
USICT, GGSIPU
New Delhi - 110078

ACKNOWLEGDEMENT

First of all, I am indebted to the **Guru Gobind Singh Indraprastha University** for giving me this opportunity to work at **University School of Information, Communication & Technology**, one of the best institutes in India.

I would like to express my sincere thanks to those who have contributed significantly to this report. It is a pleasure to extend the deep gratitude to my guide **Prof. Nupur Prakash** for her valuable guidance and support to continuously promote me for the progress of the report. I hereby present my sincere thanks to her for valuable suggestions towards my report, which helped me in making this report more efficient and user friendly.

I am very thankful to the authors of various publications to which I have been referring to. I express my sincere appreciation and thanks to all who have guided me directly or indirectly in this project.

MEGHA GUPTA
(00716404819)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Identification of plant diseases has been performed using machine learning and deep learning models on the datasets containing images of healthy and diseased plant leaves. The current study carries out an evaluation of some of the deep learning models based on convolutional neural network architectures for identification of plant diseases. Two publicly available datasets are considered for the study: 1) New Plant Diseases Dataset containing 87,900 leaf images taken in laboratory, under controlled conditions; 2) Cropped-PlantDoc dataset containing 8,872 leaf images taken in field conditions. The first dataset is an augmented version of the PlantVillage dataset (54,306 images), divided into train, valid and test set. It contains images of 12 healthy plants and 26 diseases of 14 different plants. As the images in the test set are less, all the images are combined into one folder and renamed as PlantDiseaseIdentification dataset. The second dataset contains images of 10 healthy plants and 17 diseases of 13 different plants. The dataset is augmented to 22,260 images, and renamed as Cropped-PlantDoc_Aug. The comparative study has been performed on both the datasets separately to analyze the high degree of accuracy achieved using ten CNN based models. The CNN models selected for the study presented in this project are AlexNet, ZFNet, VGGNet (four models), GoogLeNet, and ResNet (three models). They are trained using PyTorch, an open-source machine learning library, on Google Colaboratory. Mini-batch momentum based gradient descent is applied as the learning algorithm. All these models are trained on PlantDiseaseIdentification dataset from scratch. Using various evaluation metrics such as F1-score, accuracy and loss, it was found that GoogLeNet attained the highest F1-score and test accuracy of .996 and 99.59%, respectively. ResNet-50 achieved second highest F1-score of .994. The training on the Cropped-PlantDoc_Aug dataset is performed using transfer learning. Initially, the pre-trained models trained on PlantDiseaseIdentification dataset were used. However, the models did not perform well. The highest F1-score of .815 was achieved with GoogLeNet. As a next step, pre-trained models trained on ImageNet plus PlantDiseaseIdentification dataset were used. This resulted in improved F1-score of more than .85 for all model. The highest F1-score (.921), test accuracy (92.23%) and validation accuracy (92.67%) were obtained by ResNet-101, followed by ResNet-152 and ResNet-50.

CHAPTER 1

INTRODUCTION

Agriculture plays a vital role in every economy worldwide. The economic development of any country depends on its agricultural production. Traditionally, farmers follow ancestral farming patterns and practices especially in developing countries like India. A large section of human population is involved in farming as it is the basic need of human beings. Agriculture sector accounts for 14% of GDP in India.

Crop production involves taking care of all activities for better yield in all seasons. This involves soil analysis, type of seed used, major nutrient requirement, etc. However, a major challenge faced in crop production is in the form of plant diseases. Plant diseases can affect crop production both by reducing the quality and quantity of overall yield.

Main reasons for crop diseases are infections by insect pests, bacteria, fungi and viruses. It can affect the overall functional capacity of the plant resulting in reduced growth, less fruit production, more leaf falls, etc. The disease can spread from one crop to another. Sometimes viruses may get transferred with seeds from one place to another.

Traditionally, the identification of plant diseases is carried out by experts based on their knowledge and experience. However, finding an expert and contacting them is a tedious, time consuming and expensive task. It may sometimes take long time, making the eradication of the disease difficult. Now with technological advances, automatic disease detection is being attempted using image processing and computer vision techniques, machine learning techniques and deep learning techniques. Machine learning techniques like support vector machines (SVM), artificial neural networks (ANN), and random forest are most widely used for image classification.

In the present project, it is proposed to carry out a comparative analysis of various deep learning models for plant disease identification.

CHAPTER 2

PROBLEM STATEMENT

Plant health and food safety are closely linked. It is estimated that pests and diseases lead to loss of 20 – 40% of global food production constituting a threat to food security [1]. However, assessment of crop healthiness is not simple and requires a high level of expertise. The automatic identification of plant diseases by image identification using deep learning models has the potential to solve these problems. In this project, it is proposed to carry out a comparative analysis of various convolutional neural network architectures for plant disease identification.

CHAPTER 3

LITERATURE REVIEW

Plant diseases can be precisely and accurately recognized through images of plant leaves.

1. S. P. Mohanty *et al* (2016) [2] trained two deep convolutional neural network AlexNet and GoogleNet using a public dataset (PlantVillage dataset) of 54,306 images of diseased and healthy plant leaves collected under controlled conditions. The dataset consists of 14 crop species and 26 diseases. They found out that GoogleNet performed better than AlexNet and achieved an accuracy of 99.35% on a held-out test set.

2. H. Rahman *et al* (2017) [3] compared various machine learning approaches for identification of healthy and non-healthy plant leaves of cabbage, citrus and sorghum. They used 382 images of cabbage, 539 images of citrus and 262 images of sorghum as primary dataset. About 60% images were used for training and 40% were used for testing. Their results showed that the random forest classifier is the best machine learning method for classification of healthy and diseased plants.

3. H. B. Prajapati *et al* (2017) [4] presented a prototype system for detection and classification of rice diseases based on the images of infected rice plants. They considered three rice plant diseases namely Bacterial leaf blight, Brown spot, and Leaf smut. They collected the leaves from rice farm and prepared a dataset of images of rice plant leaves having a white background. Their system first removes the background from the image and used K-means clustering in order to extract the disease portions of the leaf image. They used Support Vector Machine (SVM) for multi-class classification and achieved 93.33% accuracy on training dataset and 73.33% accuracy on the test dataset.

4. K. P. Ferentinos (2018) [5] compared five convolutional neural networks architectures for the identification of plant diseases through simple leaves images of healthy or diseased plants. Models trained were AlexNet, AlexNetOWTBn, GoogleNet, Overfeat and VGG. These models were trained on 87,848 images, taken in both laboratory conditions and real conditions in cultivation fields. The data comprised 25 plant species in 58 distinct classes of [plant, disease] combinations, including some healthy plants. The data was divided into

two parts: train and test dataset in 80:20 ratio respectively. He found that the well-known CNN architecture VGG achieved error of 0.47% in the classification of 17,548 previously unseen by the model plant leaves images (testing set).

5. J. Boulent *et al* (2019) [6] surveyed 19 studies that relied on CNNs to automatically identify crop diseases. In six of these studies multi crop and multi disease models were trained while in the rest thirteen studies specialized approach focussed on a single crop. They identified the major issues and shortcomings of works in this research area.

6. P. Alagumariappan *et al* (2020) [7] developed a real time decision support system integrated with a camera sensor module for identification of plant diseases. They analysed the performance of two machine learning algorithms including Extreme Learning Machine (ELM) and Support Vector Machine (SVM) with linear and polynomial kernel. Their results demonstrated that the performance of ELM is better than SVM classifier.

7. J. Chen *et al* (2020) [8] studied transfer learning of the deep convolutional neural networks for the identification of plant leaf diseases. They considered using pre-trained model learned from the typical massive datasets, and then transferred to the specific task trained by their own data. They used VGGNet (VGG-19) pre-trained on ImageNet and achieved a validation accuracy of 91.83% on the dataset containing 500 rice images and 466 maize images (all images are diseased).

8. A. Sagar *et al* (2021) [9] studied five different CNN architectures, including VGG-16, ResNet-50, InceptionV3, InceptionResNet and DenseNet169 and performed a comparative study. They trained the networks on PlantVillage dataset containing 54,305 leaf images of diseased and healthy plants spanning over 14 plant species. They trained the models using transfer learning where they took the pre-trained version of the models and fine-tuned the last layers. They added four custom convolutional and max pooling layers on top of all the networks. Using the evaluation metrics such as F1-score, Precision, Recall and confusion metric, the best results were obtained using ResNet-50 with the accuracy of 0.982, precision of 0.94, recall of 0.94 and F1-score of 0.94.

CHAPTER 4

DATASET DESCRIPTION

4.1. Dataset 1: PlantDiseaseIdentification dataset

The ‘New Plant Diseases Dataset’ [10], which is an augmented version of publicly available PlantVillage dataset [11] is used in the comparative analysis. This dataset comprises of 87,867 images and is available on Kaggle platform. The dataset is already divided into training and validation sets in the ratio of 80:20. Apart from these images, another folder containing 33 images is available for testing. However, as the images for testing are less, all the images from training set, validation set and testing set are combined into one folder. This new folder is named as ‘**PlantDiseaseIdentification**’ which contains total 87,900 images. The total number of classes are 38 (Figure 4.1 and Table 4.1) for 14 unique plants. The total number of diseased plants are 26, as against 12 healthy plants. This new dataset is divided into three parts: training, validation and test set, in the ratio of 70:10:20. The data related to 14 unique plants is considered. These are Tomato, Bell pepper, Orange, Corn (maize), Grape, Potato, Strawberry, Peach, Apple, Squash, Blueberry, Cherry (including sour), Raspberry, and Soyabean. The number of images in each class are shown in Table 4.1.

Table 4.1: Provides number of images for each class of the 'PlantDiseaseIdentification' dataset.

Class	Scientific Name	No. of images
Tomato (Target spot)	*Corynespora cassiicola*	2284
Bell pepper (Bacterial spot)	*Xanthomonas campestris*	2391
Orange Haunglongbing (Citrus greening)	*Candidatus Liberibacter spp.*	2513
Tomato (Yellow leaf curl virus)	-	2457
Corn (maize) (Common rust)	*Puccinia sorghi*	2387
Corn (maize) healthy	-	2324
Tomato healthy	-	2411
Grape (Black Measles (Esca))	*Phaeomoniella aleophilum, Phaeomoniella chlamydospora*	2400
Potato (Late blight)	*Phytophthora infestans*	2424
Strawberry healthy	-	2280
Peach (Bacterial spot)	*Xanthomonas campestris*	2297
Tomato (Bacterial spot)	*Xanthomonas campestris pv. vesicatoria*	2127
Tomato (Tomato mosaic virus)	-	2238
Apple healthy	-	2510
Squash (Powdery mildew)	*Erysiphe cichoracearum*	2170
Corn (maize) (Northern leaf blight)	*Exserohilum turcicum*	2385
Tomato (Septoria leaf spot)	*Septoria lycopersici*	2181
Potato healthy	-	2282
Blueberry healthy	-	2270
Tomato (Two-spotted spider mite)	*Tetranychus urticae*	2176
Potato (Early blight)	*Alternaria solani*	2429
Grape healthy	-	2115
Apple (Apple scab)	*Venturia inaequalis*	2523
Tomato (Late blight)	*Phytophthora infestans*	2314
Grape (Leaf blight (Isariopsis leaf spot))	*Pseudocercospora vitis*	2152
Tomato (Leaf mold)	*Passalora fulva*	2352
Apple (Cedar rust)	*Gymnosporangium juniperi-virginianae*	2204
Cherry (including sour) healthy	-	2282
Cherry (including sour) (Powdery mildew)	*Podoshaera clandestine*	2104
Raspberry healthy	-	2226
Soybean healthy	-	2527
Strawberry (Leaf scorch)	*Diplocarpon earlianum*	2218
Tomato (Early blight)	*Alternaria solani*	2406
Peach healthy	-	2160
Corn (maize) (Gray leaf spot)	*Cercospora zeae-maydis*	2052
Grape (Black rot)	*Guignardia bidwellii*	2360
Apple (Black rot)	*Botryosphaeria obtuse*	2484
Bell pepper healthy	-	2485


Apple scab


Apple (Black rot)


Corn (Common rust)


Apple healthy


Grape (Black rot)


Grape healthy


Apple (Cedar rust)


Blueberry healthy


Cherry healthy


Cherry (Powdery mildew)


Corn (Gray leaf spot)


Corn healthy


Corn (Northern Leaf Blight)


Grape (Black measles (Esca))


Grape (Leaf blight)


Orange Haunglongbing (Citrus greening)


Peach (Bacterial spot)


Peach healthy


Bell pepper (Bacterial spot)


Bell pepper healthy


Potato (Early blight)


Potato healthy


Potato (Late blight)


Raspberry healthy


Soybean healthy

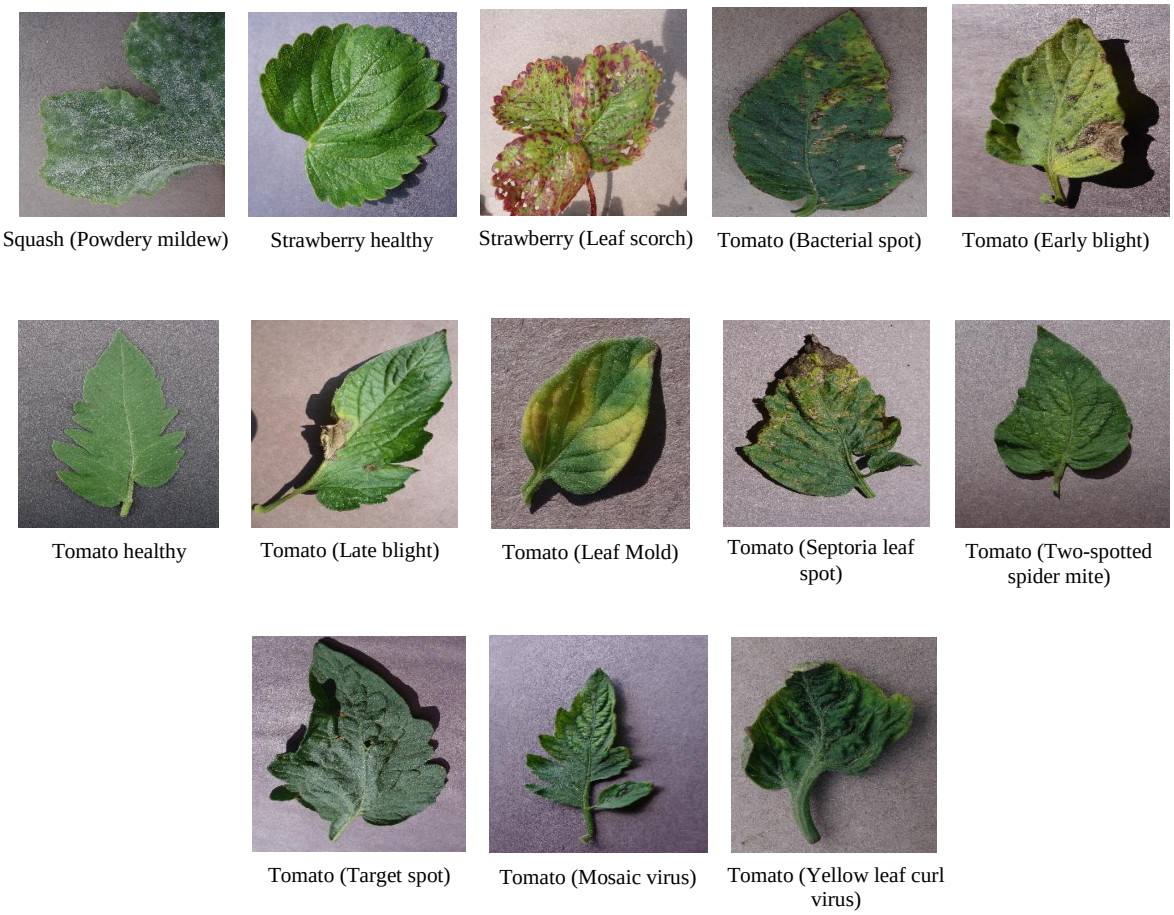


Figure 4.1: Example of leaf images from the 'PlantDiseaseIdentification' dataset

4.2. Dataset 2: Cropped-PlantDoc_Aug dataset

The New Plant Diseases Dataset contains images of plant leaves taken under controlled conditions i.e., in the laboratory under proper lightning and background. It does not contain leaf images taken directly from the field i.e., in uncontrolled conditions.

Therefore, the dataset containing field images is also used in the current analysis. The name of this dataset is PlantDoc dataset [12]. It is available on GitHub repository [13]. It contains 2,598 images of diseased and healthy plant leaves. The total number of classes are 27 spanning over 13 plant species (Figure 4.2 and Table 4.2). The number of diseased plants are 17 as against 10 healthy plants. In real scenarios, the image may have multiple leaves or a combination of diseased and healthy leaves. Therefore, in this dataset, an image contains more than one class i.e., all the leaves in the image has already been labelled with their particular classes. In order to crop the images for extracting only the leaves, the

authors of this dataset has provided the bounding box coordinates in the form of xml files (available on GitHub repository) for creating Cropped-PlantDoc dataset. These bounding box coordinates information is used in order to crop the images. After cropping 2,598 images, the total number of leaf images turned out to be 8,872 i.e., 8,872 bounding boxes. Since this Cropped-PlantDoc dataset is not a balanced dataset, I applied Common augmentation techniques such as flipping (Horizontal and Vertical flipping) and rotated the images at different angles (45º, 90º, 135º, 180º, 225º, 270º, 315º). After augmentation, the total number of images turned out to be 22,260 and the dataset is named as '**Cropped-PlantDoc_Aug'**. This augmented dataset is divided into three parts: training, validation and test set, in the ratio of 70:10:20. The data related to 13 unique plants is considered. These are Tomato, Bell pepper, Corn, Grape, Potato, Strawberry, Peach, Apple, Squash, Blueberry, Cherry, Raspberry, and Soyabean. The number of images in each class are shown in Table 4.2.

Table 4.2: Provides number of images for each class of the 'Cropped-PlantDoc_Aug' dataset.

Class	Scientific Name	No. of images
Tomato leaf (Bacterial spot)	*Xanthomonas campestris pv. vesicatoria*	840
Tomato leaf (Late blight)	*Phytophthora infestans*	880
Tomato leaf (Yellow virus)	-	829
Bell pepper leaf healthy	-	846
Corn (Leaf blight)	*Exserohilum turcicum*	834
Strawberry leaf healthy	-	842
Potato leaf (Early blight)	*Alternaria solani*	862
Grape leaf healthy	-	880
Corn (Gray leaf spot)	*Cercospora zeae-maydis*	780
Soyabean leaf healthy	-	798
Bell pepper (leaf spot)	*Xanthomonas campestris*	789
Apple (rust leaf)	*Gymnosporangium juniperi-virginianae*	890
Tomato (Septoria leaf spot)	*Septoria lycopersici*	866
Tomato (Mold leaf)	*Passalora fulva*	879
Corn (rust leaf)	*Puccinia sorghi*	889
Tomato leaf (Mosaic virus)	-	783
Tomato leaf (Late blight)	*Phytophthora infestans*	747
Peach leaf healthy	-	814
Grape leaf (Black rot)	*Guignardia bidwellii*	798
Tomato leaf (Early blight)	*Alternaria solani*	852
Apple leaf healthy	-	741
Cherry leaf healthy	-	717
Raspberry leaf healthy	-	856
Apple (Apple scab)	*Venturia inaequalis*	855
Blueberry leaf healthy	-	839
Tomato leaf healthy	-	792
Squash powdery mildew leaf	*Erysiphe cichoracearum*	762


Apple healthy


Apple (rust leaf)


Apple (Scab leaf)


Bell pepper healthy


Bell pepper (Leaf spot)

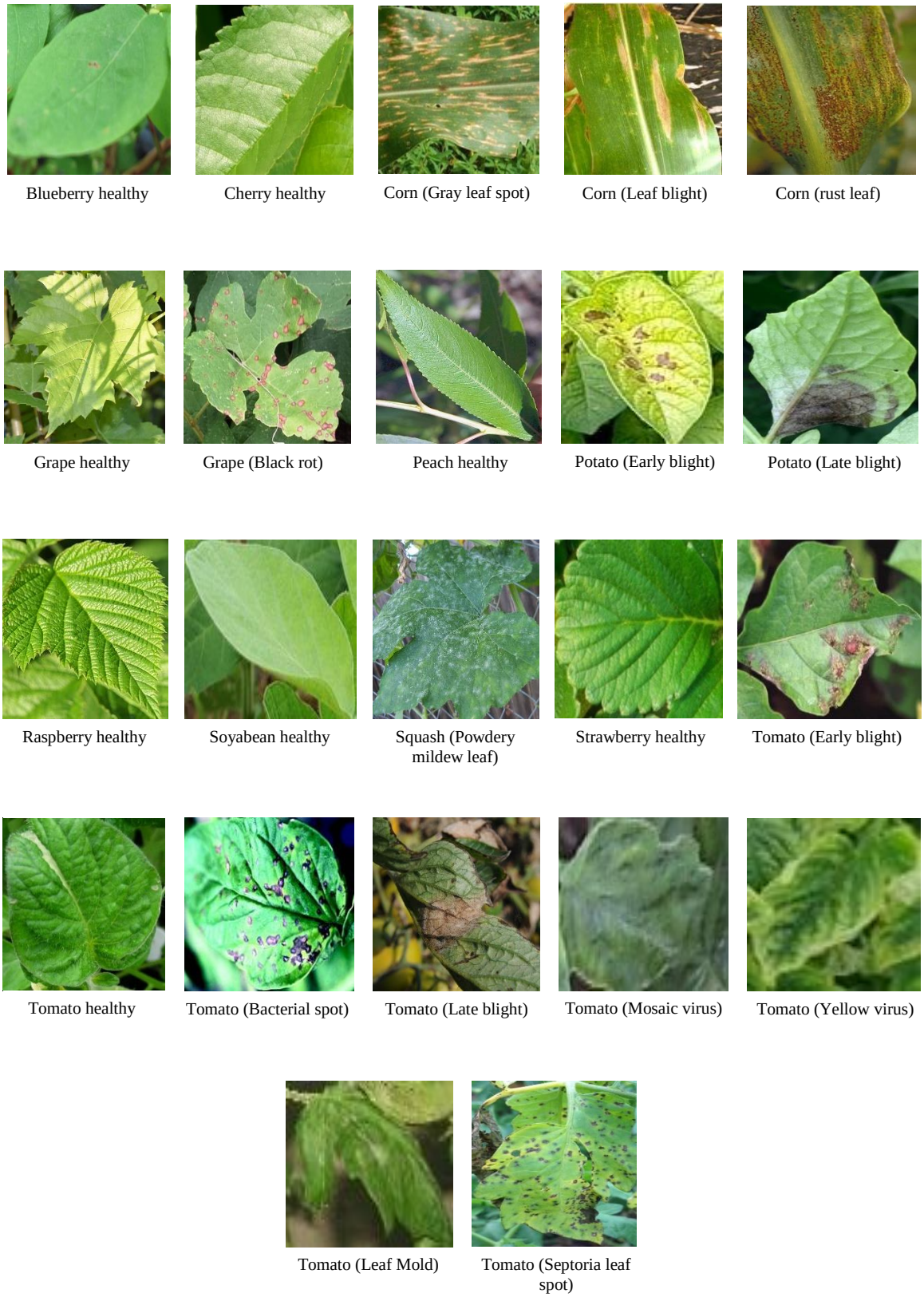


Figure 4.2: Example of leaf images from the 'Cropped-PlantDoc_Aug' dataset

CHAPTER 5

METHODOLOGY

5.1. Steps followed

1. The datasets 'PlantDiseaseIdentification dataset' and 'Cropped-PlantDoc_Aug' are pre-processed.
2. Standard and well-known CNN architectures (AlexNet, ZFNet, VGGNet (4 models), GoogLeNet, ResNet (3 models)) are trained, validated and tested on the PlantDiseaseIdentification dataset.
3. Mini-batch momentum based gradient descent is used as the learning algorithm.
4. Various evaluation metrics such as Accuracy, Loss, Precision, Recall and F1-Score, are used for comparing performance of these CNN models.
5. Transfer learning is performed on Cropped-PlantDoc_Aug dataset using the models that are trained on PlantDiseaseIdentification dataset.
6. Comparative evaluation of models trained on Cropped-PlantDoc_Aug dataset is done using the evaluation metrics such as Accuracy, Loss, Precision, Recall and F1-Score.

5.2. Basic CNN Components

A convolutional neural network, abbreviated as CNN or ConvNet falls under the category of deep neural networks i.e., networks having many non-linearities, but simultaneously have fewer parameters than fully connected feed-forward networks. CNNs are hence less prone to overfitting as compared to feed-forward neural network. The operation performed on the CNN is called as convolution operation. There are numerous applications where CNNs are applied such as image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, etc.

The basic architecture of CNN comprises of an input, convolution layer, pooling layer, fully connected layer, output layer [14].

Input: Input is an RGB Image of plant leaf.

Convolution layer: In this layer, convolution operation is performed on the given input. Convolution operation is a very simple mathematical operation of computing a weighted

average of all the previous neighbours to determine current value. Filters also called as kernels having dimensions (p x q) less than the dimensions of the input are applied on the input (such as image) in order to re-estimate the value of each pixel of interest by doing the weighted average of its neighbours (Figure 5.2). If the number of filters applied are *k*, then a feature map is generated having depth equal to *k*. Each cell of the feature map represents a neuron.

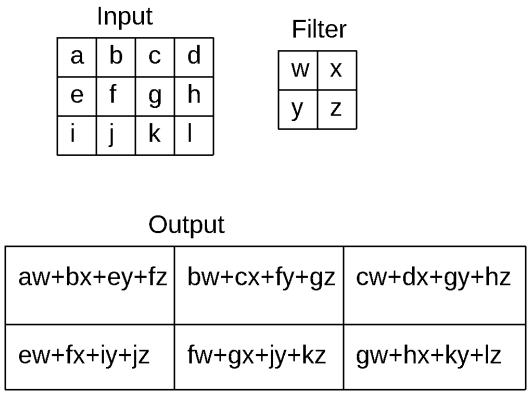


Figure 5.2: Convolution operation

Pooling layer: The task of this layer is to reduce the dimension of the output received from convolutional layer by doing average pooling, max pooling, etc.

Fully connected layer: Every neuron present in one layer of the network is linked to every other neuron present in another layer. It is similar to the conventional multi-layer perceptron model (MLP).

In the convolution layer, the input pixels or neurons are re-estimated by doing the weighted average of the neighbours of the particular pixel. Generally, the neighbourhood size is taken as a square size (for example: 3 × 3 or 5 × 5, etc.). In case of a fully connected layer, the entire layer is used as one neighbourhood. This is done by flattening the last convolutional layer.

Each feature map represents a particular feature of an input.

Advantage of CNN: The main advantage of CNN is that the same kernel is shared among all the neurons, therefore leading to fewer parameters. Also using CNNs, in one go, many different features can be extracted.

5.3. CNN Models

5.3.1. AlexNet

Alex Krizhevsky *et al* [15] proposed AlexNet in 2012. For tasks such as image recognition and classification, AlexNet is considered as the first innovative and creative model. The Table 5.3.1 highlights the layers, filter sizes and output sizes used in the network and thus represents the simple architectural design of AlexNet. Hardware limitations limited the learning capacity of deep CNN designs in the early 2000s, limiting them to tiny sizes. Training of AlexNet model was performed by using two NVIDIA GTX 580 GPUs in parallel. This was done in order to receive the advantage of the representational capacity of deep CNNs and also to overcome the hardware shortcomings.

Characteristics of AlexNet architecture:

- It was used in ImageNet challenge and was the winning network in the year 2012 (Top 5 error rate: 16.4).
- It is an 8-layer architecture (5 convolution layers and 3 fully connected layers).
- It is operated with 3-channel images (RGB images) that are (224x224x3) in size.
- Max pooling is used for subsampling.
- ReLU function is used as an activation function in order to address the problem of vanishing gradient to some extent. ReLU is called as Rectified Linear Unit. The first model to employ ReLU functionality was AlexNet.
- Large size filters (11x11 and 5x5) are used at the initial layers followed by 3x3 filter size for rest of the layers.
- 3 x 3 kernels are used for max pooling.
- 96 filters of size 11x11 with stride 4 are used in first conv layer, 256 filters of size 5x5 with stride 1 are used in second conv layer, 384 filters of size 3x3 with stride 1 are used in third and fourth conv layers, and 256 filters of size 3x3 with stride 1 are used in fifth conv layer.
- Batch normalisation follows each convolution layer.
- Dropout is also used in the first two fully-connected layers to reduce overfitting.
- Softmax as activation function is employed for output layer.
- Number of trainable parameters is 60 Million (M).
- Classified images into one of N classes, where N is the number of classes.

Table 5.3.1: Architecture of AlexNet

AlexNet Architecture			
Layer	No. of Filters (K)	Filter size	Output size
Input image			224 x 224 x 3
conv	96	11 x 11/4	55 x 55 x 96
maxpool		3 x 3/2	27 x 27 x 96
conv	256	5 x 5/1	27 x 27 x 256
maxpool		3 x 3/2	13 x 13 x 256
conv	384	3 x 3/1	13 x 13 x 384
conv	384	3 x 3/1	13 x 13 x 384
conv	256	3 x 3/1	13 x 13 x 256
maxpool		3 x 3/2	6 x 6 x 256
FC-4096			
FC-4096			
FC-1000			

5.3.2: ZFNet

ZFNet was proposed by Zeiler and Fergus [16] in 2013. The main reason behind the development of ZFNet model was to quantitatively visualize the performance of the AlexNet network. By interpreting the output feature maps after application of activations, they monitored the performance exhibited by the neurons. After doing visualization experiments on AlexNet, they showed that only a fraction of neurons was being active in the first and the second layers. Other neurons in the first and second layers of the network, on the other hand, were dead i.e., inactive. CNN topology was adjusted and parameter optimization was performed by the authors based on their findings. The learning capability and capacity of CNN were by reducing both the kernel size and stride in order to retain the maximum number of features in the first two convolutional layers of AlexNet.

Characteristics of ZFNet architecture:

- It was used in ImageNet challenge and was the winning network in the year 2013 (Top 5 error rate: 11.7).
- It is an 8-layer architecture (5 convolution layers and 3 fully connected layers). (Table 5.3.2)
- It is operated with 3-channel images (RGB images) that are (224x224x3) in size.
- Max pooling is used for subsampling.
- ReLU function is used as an activation function in order to address the problem of vanishing gradient to some extent. ReLU is called as Rectified Linear Unit.
- 96 filters of size 7x7 with stride 2 are used in the first convolutional layer instead of filter size 11x11 with stride 4 as used in AlexNet.
- 256 filters of size 5x5 with stride 2 are used in second convolutional layer.
- Third, fourth and fifth convolutional layers are similar to AlexNet in filter size and stride, only the number of filters are increased.
- In third convolutional layer, 512 filters are used instead of 384, in fourth conv layer 1024 filters are used instead of 384 and in fifth conv layer 512 filters are used instead of 256.
- Each convolution layer is followed by batch normalization.
- Dropout is also used in the first two fully-connected layers to reduce overfitting.
- Softmax as activation function is employed for output layer.
- Number of trainable parameters is 104 M.
- Classified images into one of N classes, where N is the number of classes.

Table 5.3.2: Architecture of ZFNet

ZFNet Architecture			
Layer	No. of Filters (K)	Filter size	Output size
Input image			224 x 224 x 3
conv	96	7 x 7/2	110 x 110 x 96
maxpool		3 x 3/2	55 x 55 x 96
conv	256	5 x 5/2	26 x 26 x 256
maxpool		3 x 3/2	13 x 13 x 256
conv	512	3 x 3/1	13 x 13 x 512
conv	1024	3 x 3/1	13 x 13 x 1024
conv	512	3 x 3/1	13 x 13 x 512
maxpool		3 x 3/2	6 x 6 x 512
FC-4096			
FC-4096			
FC-1000			

5.3.3. VGGNet

Simonyan and Zisserman [17] proposed VGGNet in 2014. Small size filters can improve the performance of CNNs, according to ZFNet, which is a frontline network in the 2013-ILSVRC competition. Based on these findings, VGG replaced the 11x11 and 5x5 filters (that were used in AlexNet model for first two convolutional layers respectively) with a stack of filters having dimensions of 3x3. Concurrent application of small size (3x3) filters was shown to generate the impact of the large size filter in an experiment (11x11, 7x7 and 5x5). By lowering the number of parameters, the usage of small size filters provides an additional benefit of low computing complexity. These discoveries ushered in a new research trend at CNN, which is to work with lower size filters. Architecture of VGG is shown in Table 5.3.3.

Characteristics of VGGNet architecture:

- VGG finished second in the 2014-ILSVRC competition (Top 5 error rate: 7.3), but gained notoriety for its simplicity, homogeneous topology, and enhanced depth.
- Four different ConvNet configurations (VGG11, VGG13, VGG16, VGG19) were proposed.
- The input to the model is a fixed-size 224 x 224 RGB image.
- It has same back to back multiple convolutional layers and some intermediate max pooling layer.
- Throughout the network the filter size is 3x3.
- The convolution stride is fixed to 1 pixel
- Padding of 1 is used for 3x3 conv layers in order to preserve the spatial resolution of the convoluted output.
- Max-pooling is performed over a 2 x 2 pixel window, with stride 2.
- Number of trainable parameters in VGG-11 is 133 M, in VGG-13 is 133 M, in VGG-16 is 138 M and in VGG-19 is 144 M.

Table 5.3.3: Architecture of VGG

ConvNet Configuration			
A	B	C	D
11 weight layers	13 weight layers	16 weight layers	19 weight layers
Input (224 x 224 RGB image)			
conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
Maxpool			
conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
Maxpool			
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
Maxpool			
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
Maxpool			
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
Maxpool			
FC-4096			
FC-4096			
FC-1000			

5.3.4. GoogLeNet

Christian Szegedy *et al* [18] proposed the GoogLeNet model in 2014. It is also called as InceptionNet. The GoogLeNet architecture's major goal was to achieve excellent accuracy at a low computing cost. It proposed the new notion of inception block in CNN, which uses the divide, transform, and merge principle to add multi-scale convolutional changes. Figure 5.3.4 represents the architecture of an inception block. This block has filters of various sizes (1x1, 3x3, and 5x5) that collect spatial data at various scales, including fine and coarse grain levels. GoogLeNet's use of the divide, transform, and merge concept aided in the solving of a problem involving the learning of various types of variations found in the same category of images with varying resolutions. The architecture of GoogLeNet is shown in Table 5.3.4.

Characteristics of GoogLeNet architecture:

- It was the winning network of ILSVRC14 (Top 5 error rate: 6.7)
- It is a 22 layers deep network.
- Inception module is the building block of GoogLeNet. Full architecture is a series of these Inception modules.
- Inception Module has 1x1 convolutions followed by 3x3 and 5x5 convolutions. Also, 3x3 max pooling followed by 1x1 convolution. Also, directly has 1x1 convolutions.
- Inception Module differ in number of filters across different layers.
- Each Inception layer counts for two convolutional layers.
- Number of trainable parameters is 6 M (approx.), hence 12x less parameters than AlexNet [15].
- It introduced 1x1 convolutions which reduces the number of computations.

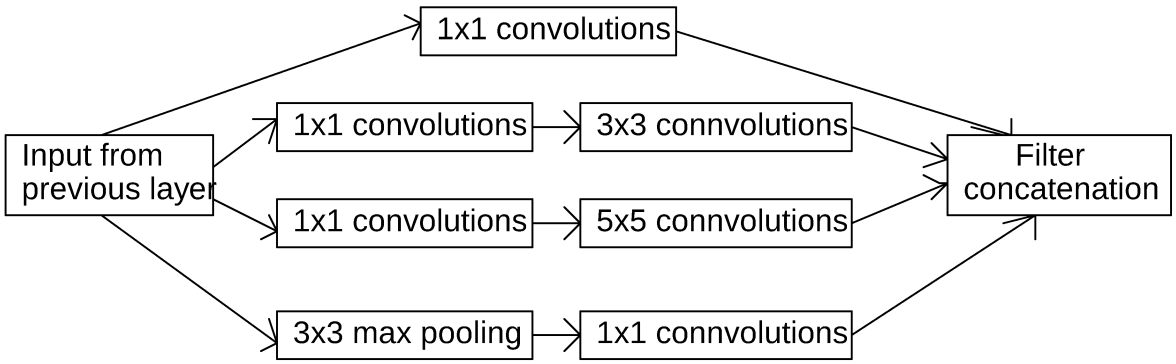


Figure 5.3.4: Inception Module

Table 5.3.4: GoogLeNet architecture

Layer	Filter size/stride	Output size	Depth	#1x1	#3x3 reduce	#3x3	#5x5 reduce	#5x5	Pool proj
Convolution	7 x 7/2	112 x 112 x 64	1						
Max pool	3 x 3/2	56 x 56 x 64	0						
Convolution	3 x 3/1	56 x 56 x 192	2		64	192			
Max pool	3 x 3/2	28 x 28 x 192	0						
Inception (3a)		28 x 28 x 256	2	64	96	128	16	32	32
Inception (3b)		28 x 28 x 480	2	128	128	192	32	96	64
Max pool	3 x 3/2	14 x 14 x 480	0						
Inception (4a)		14 x 14 x 512	2	192	96	208	16	48	64
Inception (4b)		14 x 14 x 512	2	160	112	224	24	64	64
Inception (4c)		14 x 14 x 512	2	128	128	256	24	64	64
Inception (4d)		14 x 14 x 528	2	112	144	288	32	64	64
Inception (4e)		14 x 14 x 832	2	256	160	320	32	128	128
Max pool	3 x 3/2	7 x 7 x 832	0						
Inception (5a)		7 x 7 x 832	2	256	160	320	32	128	128
Inception (5b)		7 x 7 x 1024	2	384	192	384	48	128	128
Avg pool	7 x 7/1	1 x 1 x 1024	0						
Dropout (40%)		1 x 1 x 1024	0						
linear		1 x 1 x 1000	1						

5.3.5. ResNet

Kaiming He *et al* [19] proposed ResNet in 2015. By establishing the concept of residual learning in CNNs and therefore devising an efficient methodology for deep network training, ResNet altered the CNN architectural race. They introduced residual connections (Figure 5.3.5). The architecture of ResNet is shown in Table 5.3.5.

Characteristics of ResNet architecture:

- Three different ResNet models were proposed (ResNet-50, ResNet-101, ResNet-152).
- ResNet-152 was used in ImageNet challenge and was the winning network in the year 2015 (Top 5 error rate: 3.57).

- Mainly 3x3 sized kernels are used for the convolutional layers. There are two basic design rules followed by convolutional layers: (1) the number of filters are same in the layers if the output feature map is of same size; and (2) in order to preserve the time complexity per layer, the number of filters is doubled when the feature map size is reduced to half.
- AlexNet and VGG models are 20 and 8 times less deep than ResNet-152 respectively, but still ResNet-152 shows less computational complexity than previously proposed networks i.e., AlexNet and VGG.
- After every three layers, the residual connection (shortcut connection) is added. Identity mapping is performed by these residual connections, and the output generated by these connections are just added to the outputs of the stacked layers as shown in Figure 5.3.5.
- Down-sampling is performed by conv3_1, con4_1, and conv5_1 with a stride of 2.
- Number of trainable parameters in ResNet-50 is 23 M (approx.), in ResNet-101 is 42 M (approx.), in ResNet-152 is 58 M (approx.).

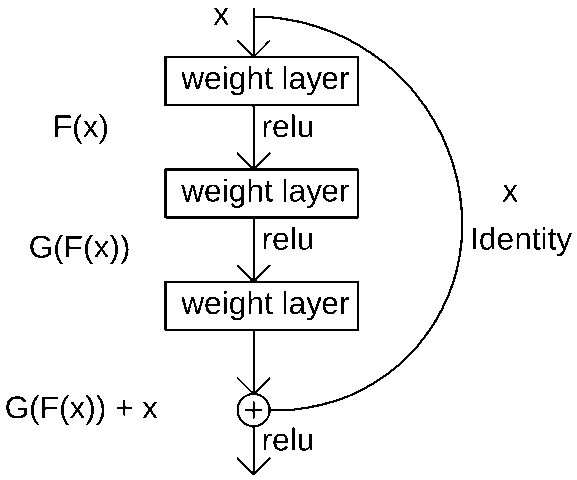


Figure 5.3.5: Residual Block

Table 5.3.5: ResNet architecture

Layer	50-layer	101-layer	152-layer	Output size
Conv1	7 x 7, 64, stride 2			112 x 112
Conv2_x	3 x 3 max pool, stride 2			56 x 56
	1 x 1, 64 3 x 3, 64 1 x 1, 256 × 3	1 x 1, 64 3 x 3, 64 1 x 1, 256 × 3	1 x 1, 64 3 x 3, 64 1 x 1, 256 × 3	
Conv3_x	1 x 1, 128 3 x 3, 128 1 x 1, 512 × 4	1 x 1, 128 3 x 3, 128 1 x 1, 512 × 4	1 x 1, 128 3 x 3, 128 1 x 1, 512 × 8	28 x 28
Conv4_x	1 x 1, 256 3 x 3, 256 1 x 1, 1024 × 6	1 x 1, 256 3 x 3, 256 1 x 1, 1024 × 23	1 x 1, 256 3 x 3, 256 1 x 1, 1024 × 36	14 x 14
Conv5_x	1 x 1, 512 3 x 3, 512 1 x 1, 2048 × 3	1 x 1, 512 3 x 3, 512 1 x 1, 2048 × 3	1 x 1, 512 3 x 3, 512 1 x 1, 2048 × 3	7 x 7
	Average pool, 1000-d fc			1 x 1

5.4. Transfer Learning

When the knowledge gained by the deep learning model for one task is used for gaining the knowledge by the same deep learning model for another similar type of task, then it is called transfer learning [20][21].

Formal definition of transfer learning:

"Given a source domain D_s, a corresponding source task T_s, as well as a target domain D_t and a target task T_t, the objective of transfer learning is to improve the performance of predictive function f_t for learning target task T_t by discover and transfer latent knowledge from D_s and T_s where $D_s \neq D_t$ or $T_s \neq T_t$. In most cases, the size of D_s is much larger than the size of D_t" [22].

Transfer learning is being frequently used with deep learning models. Two of the main reasons are [21]:

a) Insufficient data for learning a task: For neural networks to learn effectively, a lot of data is required which might be not available for a particular learning task.

b) Reduction in training time: Training time is reduced by a large extent using transfer learning.

5.4.1. Transfer Learning Approaches

Following are the two common approaches for implementing transfer learning [23][24]:

1. Develop Model Approach

In this approach, first the model is developed by training it from scratch on a large dataset. The model developed is used as a pre-trained model for the target learning task. This may involve using all or parts of the model.

2. Pre-trained Model Approach

In this approach, a pre-trained model is selected from the already available pre-trained models that are trained on large datasets. The selected pre-trained model is used for the target learning task. This may involve using all or parts of the model.

CHAPTER 6

IMPLEMENTATION

For implementation of this project the following steps are followed:

- The Google drive is mounted on Google Colaboratory notebook [25]. The zip file of both the datasets used are extracted from google drive.
- The Python libraries are imported.

6.1. Python libraries used

1) **NumPy**: It supports mathematical computations on arrays that are multidimensional in nature. Therefore, this python package is called as Numerical Python. It makes all the numerical and logical operations to carry out on array objects easily such as matrix multiplication, etc. It is a library containing many different functions for numerical computations, processing and data analysis on n-dimensional arrays.
2) **Matplotlib**: It is extensively used for visualization of data i.e., for making plots that depicts simulation of data. The data present in array objects is used by this popular library in order to generate 2D plots. It helps in data analysis (by using NumPy) that aids the understanding of the data for better computation.
3) **Pytorch**: It is a very popular library which is extensively used for machine learning and deep learning tasks by providing tensor computations through GPU. It is developed by Facebook's AI Research Lab (FAIR). It is primarily based upon Torch library. It is written in Python and hence very easy to understand and use. It is an open-source framework that contains functions for computational processing (such as automatic differentiation) of neural networks.

   There are two important features provided by PyTorch [26]:

   1. Tensor computations (like NumPy) with strong GPU (Graphics processing units).
   2. Autograd, that enables one to perform backpropagation in a functional manner (used to build large models and then do gradient descent through them).

   PyTorch tensor: It is a class within PyTorch framework defines as *torch.Tensor*. A tensor is similar to a NumPy array but since it uses GPU acceleration, it is called as PyTorch tensor.

Modules in PyTorch

1) **Autograd** module: The main task of this module is to perform automatic backpropagation i.e., gradient computation. All the operations such as linear combinations and activations or other functions performed are recorded (stored) during forward pass. After completion of forward pass, it modifies the weights and biases (parameters) by computing gradients using the information recorded during forward pass. This feature of PyTorch helps in saving time on one epoch.
2) **Optim** module: This module is defined as a class in PyTorch library as *torch.optim*. This class consists of different optimization learning algorithms that are used for the learning process of neural networks. is a module that implements various optimization algorithms used for building neural networks. Various optimizers supported are Stochastic Gradient Descent, Adam, Adagrad, Rmsprop, etc. Therefore, it is not required to code the entire learning algorithm.
3) **nn** module: This module helps to build a neural network. It is broken down into modular objects that share or have the same **Module** interface. Feedforward operation can be performed by just calling the *forward()* function and similarly, backpropagation can be done by just invoking *backward()* function. The raw autograd is a little too low-level for defining sophisticated network. *nn* (neural network) module together with PyTorch autograd makes the methods to effortlessly define the computational graphs and take gradients.

4) **DataLoader class**: It is used for iterating the datasets. It implements methods that can load a batch of data (in a main or sub-process) from the dataset and thus iterates over that dataset batch-wise.

5) **Mlflow** [27] is installed which is used for logging metrics and hyperparameters. Mlflow can be accessed and used openly without any barriers for logging all the machine learning or deep learning experiments, the trained models, the trained parameters and experiment results.

MLflow Tracking: This component of MLflow helps in tracking all the experiment runs. The values of the metrics and the parameters of the current run can be easily compared from the values of the metrics and the parameters of the previous run. It provides a great

user interface as each result can be visualized using graphs. It supports experiments done in different languages such as Python, R, Java, etc. The following information is recorded by each run:

1. Code Version
2. Start & End Time: Start and end time of the run.
3. Source: Name of the file to launch the run.
4. Parameters: Key-value pair string parameters.
5. Metrics: Key-value metrics, where the value is numeric.
6. Artifacts: Output file in any format. For example, images (PNGs), models, and data files.

6.2. Pre-Processing

The no. of unique plants, number of diseases and the number of images present for each class are calculated and plotted for both the datasets as shown in Figure 6.2(a) and Figure 6.2(b).

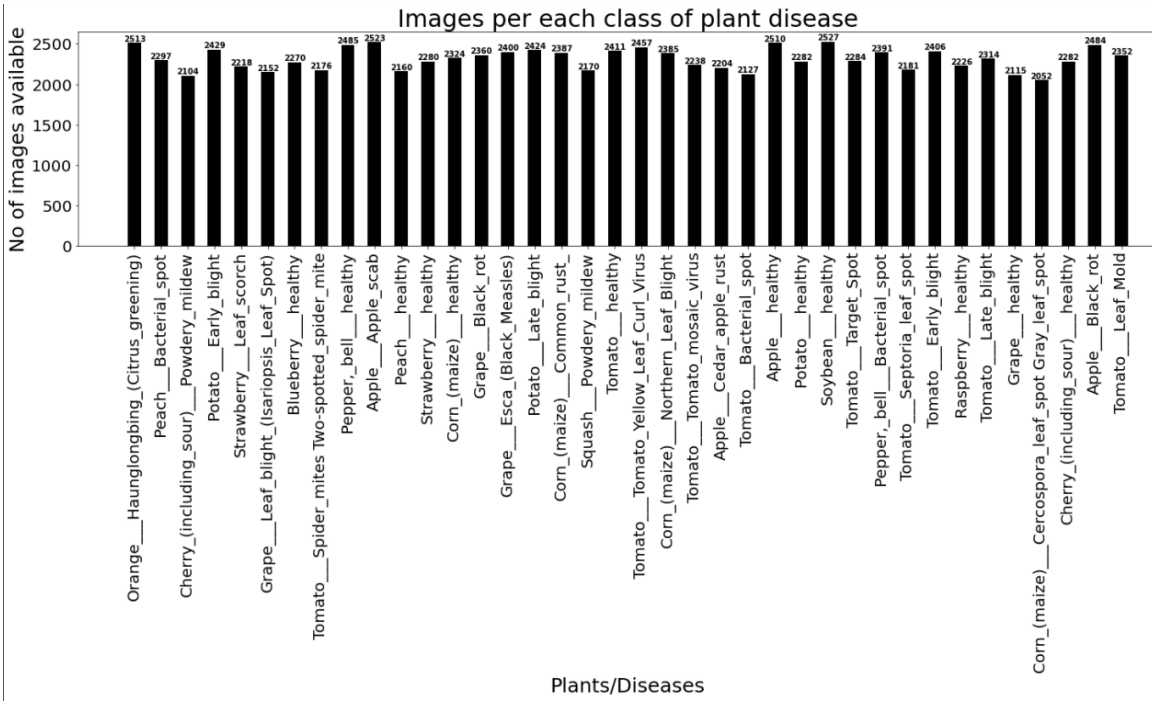


Figure 6.2(a): Number of images per class of the 'PlantDiseaseIdentification' dataset

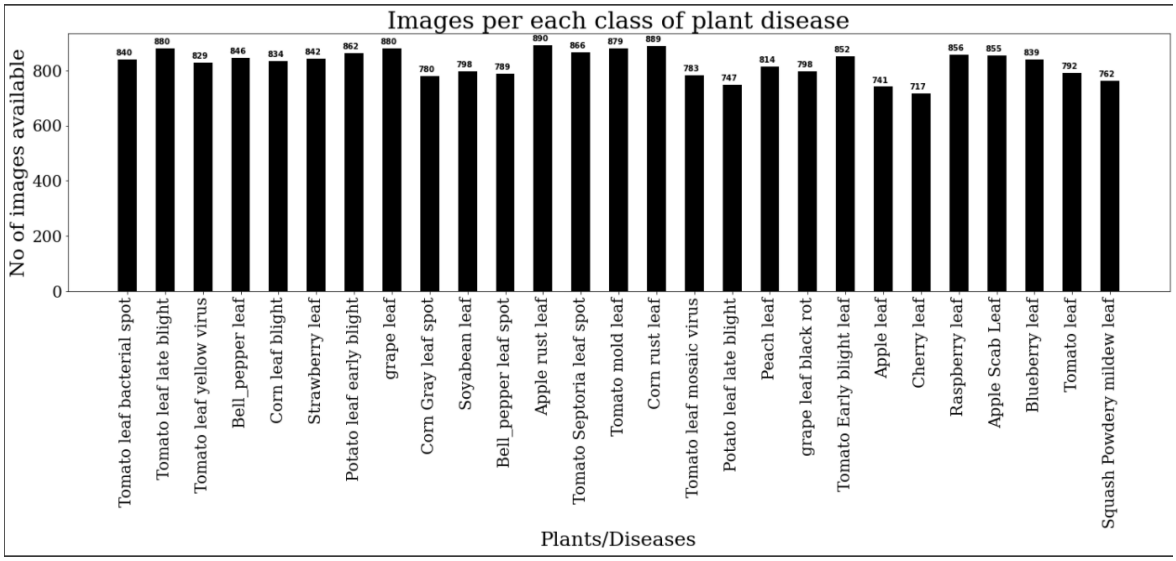


Figure 6.2(b): Number of images per class of the 'Cropped-PlantDoc_Aug' dataset

6.2.1. Pre-Processing for 'PlantDiseaseIdentification' dataset

1. The images are resized to 224 x 224 (from 256 x 256) using *Resize()* function that is available in *transforms* package of *torchvision* library. This is done as the models take 224 x 224 size images as input.
2. The Dataset is split into 3 parts: train (61,530), valid (8,790), test (17,580) sets.

6.2.2. Pre-Processing for 'Cropped-PlantDoc_Aug' dataset

1. In order to crop the images for extracting only the leaves, the authors of this dataset has provided the bounding box coordinates in the form of xml files (available on GitHub repository) for creating Cropped-PlantDoc dataset. These bounding box coordinates information is used in order to crop the images.
2. After cropping, the total number of leaf images turned out to be 8,872 i.e., 8,872 bounding boxes.
3. Common augmentation techniques are applied such as flipping (Horizontal and Vertical flipping) and rotating the images at different angles (45º, 90º, 135º, 180º, 225º, 270º, 315º). After augmentation, the total number of images turned out to be 22,260 and the dataset is named as '**Cropped-PlantDoc_Aug'**.

4. The images in the dataset are of different sizes. All the images are resized to 224 x 224 using *Resize()* function that is available in *transforms* package of *torchvision* library. This is done as the models take 224 x 224 size images as input.
5. This augmented dataset is divided into three parts: training, validation and test set, in the ratio of 70:10:20 i.e., train (15,582), valid (2,226), test (4,452) sets.

6.3. Training and Testing

1. The Architectures of AlexNet, ZFNet, VGGNet, GoogLeNet and ResNet are created using PyTorch.
2. Since this problem deals with classification, therefore cross entropy loss function is used.
3. The training loop is defined (fit function). All the models are trained by calling the fit function. Hyperparameters taken are: batch size = 32, learning rate = 0.001, momentum = 0.9, number of epochs, and weight decay.
4. After each epoch, training loss, training accuracy, validation loss and validation accuracy are calculated and logged. After training, test accuracy and test loss are calculated. Precision, recall and F1-score are also calculated.

6.4.Transfer Learning

6.4.1. Steps followed

1. The models trained on 'PlantDiseaseIdentification' dataset are used for training on 'Cropped-PlantDoc_Aug' dataset i.e. on the augmented version of the cropped PlantDoc dataset.
2. Training and validation Accuracy, training and validation loss, test accuracy and loss, as well as F1-score are calculated and tabulated for comparative analysis.
3. ImageNet weights are downloaded for each model.
4. Each model is trained on classification of 'PlantDiseaseIdentification' dataset (for 12 epochs).
5. The model weights are saved and these model weights are called as 'ImageNet + PlantDiseaseIdentification'.
6. The saved model weights are then loaded and each model is trained on 'Cropped-PlantDoc_Aug' dataset (augmented version of Cropped-PlantDoc dataset) separately.

CHAPTER 7

RESULTS

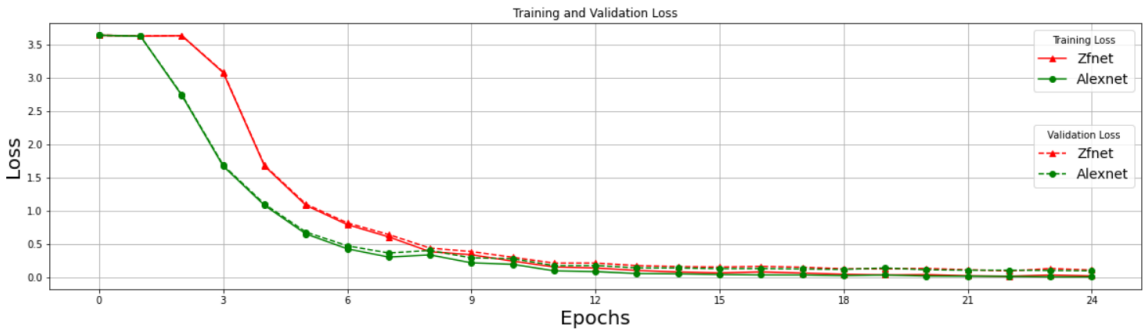
7.1. Performance of deep CNN based models trained on 'PlantDiseaseIdentification' dataset

7.1.1. Discussion

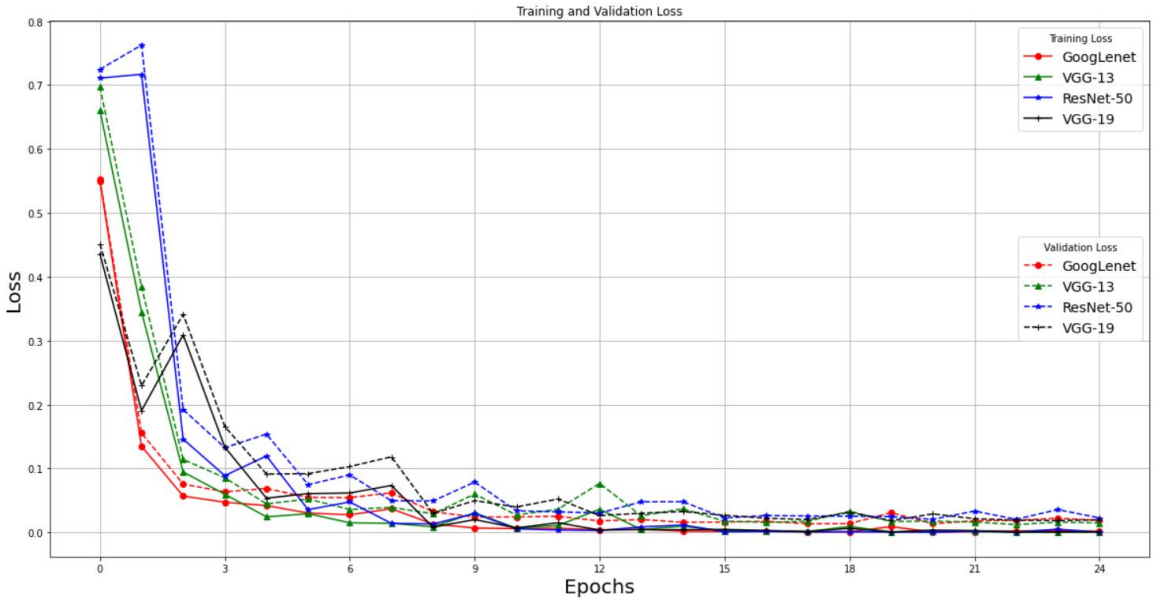
This section presents the comparative analysis of deep learning convolutional based neural network architectures in order to select the best model for the augmented version of the PlantVillage dataset i.e., for '*PlantDiseaseIdentification'* dataset. Training accuracy, validation accuracy, training loss, validation loss, testing accuracy, testing loss, precision, recall and F1-score were used for evaluating the results obtained. The performance metric, F1-score, is considered as the most important evaluation metric. It is the harmonic mean of Precision and recall. Therefore, for the task of identification of plant diseases, the model that achieved the highest F1-score was considered to be most suitable. It was observed that training the models for about 25 epochs, the training as well as the validation accuracy and loss were converged. The performances of deep learning architectures are represented by line graphs (Figure 7.1.1(a), Figure 7.1.1(b)). Table 7.1.1 shows the overall performance of deep CNN architectures.

Table 7.1.1: Performance of various deep learning architectures trained on 'PlantDiseaseIdentification' dataset.

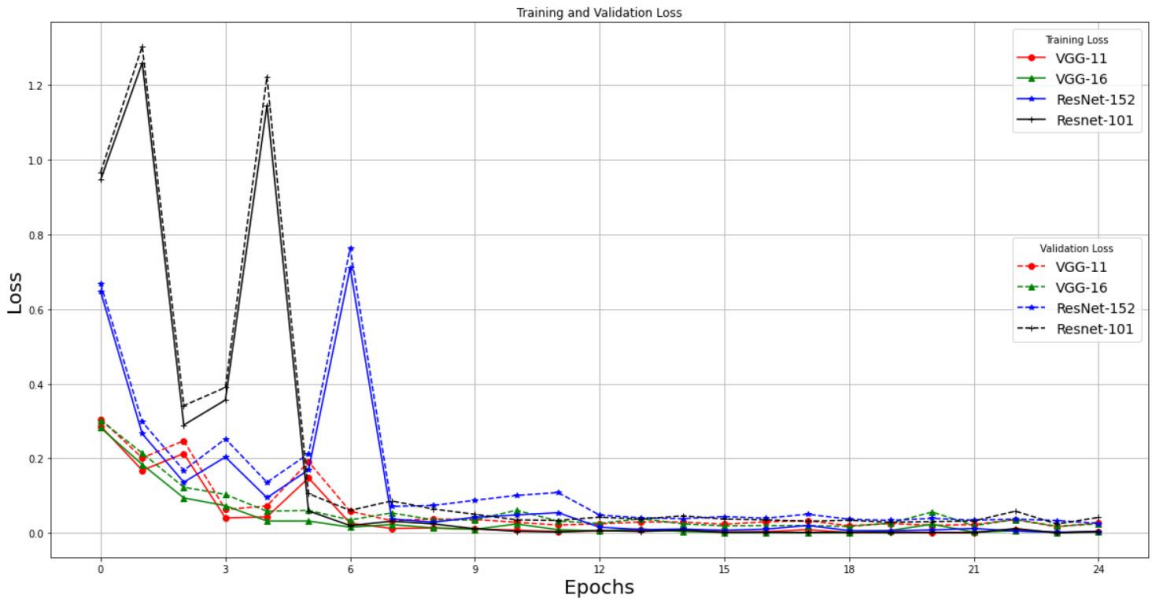
Deep Learning Architectures	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Precision	Recall	F1-score	Test Accuracy	Test Loss
AlexNet	0.9968	0.0110	0.9696	0.1019	0.968	0.967	0.967	0.9674	0.1128
ZFNet	0.9924	0.0250	0.9637	0.1139	0.962	0.961	0.961	0.9612	0.1269
VGG-11	0.9991	0.0026	0.9917	0.0277	0.992	0.991	0.992	0.9916	0.0326
VGG-13	0.9996	0.0010	**0.9950**	**0.0148**	0.994	0.994	**0.994**	0.9938	0.0199
VGG-16	0.9992	0.0022	0.9929	0.0244	0.993	0.993	0.993	0.9931	0.0247
VGG-19	0.9998	0.0006	0.9935	0.0197	0.994	0.994	**0.994**	0.9940	0.0200
GoogleNet	0.9995	0.0015	**0.9948**	**0.0188**	**0.996**	**0.996**	**0.996**	**0.9959**	**0.0136**
ResNet-50	**0.9999**	**0.0004**	0.9938	0.0226	0.994	0.994	**0.994**	0.9944	0.0197
ResNet-101	0.9987	0.0046	0.9885	0.0416	0.988	0.988	0.988	0.9878	0.0407
ResNet-152	0.9992	0.0031	0.9917	0.0257	0.993	0.993	0.993	0.9927	0.0250



(a) AlexNet, ZFNet

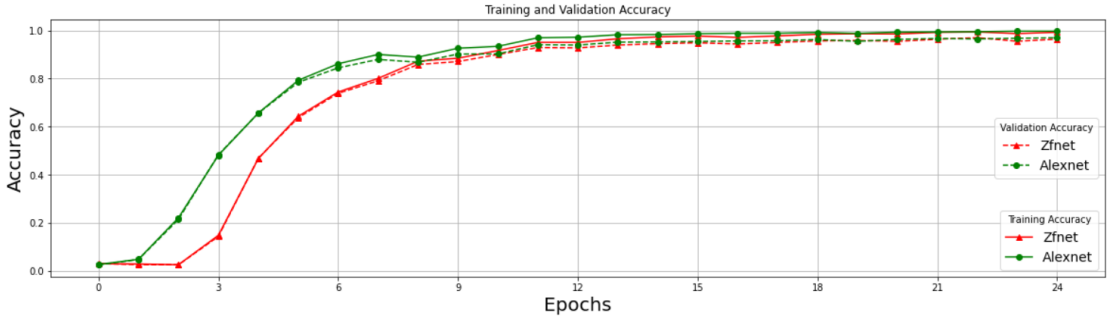


(b) GoogLeNet, VGG-13, ResNet-50, VGG-19

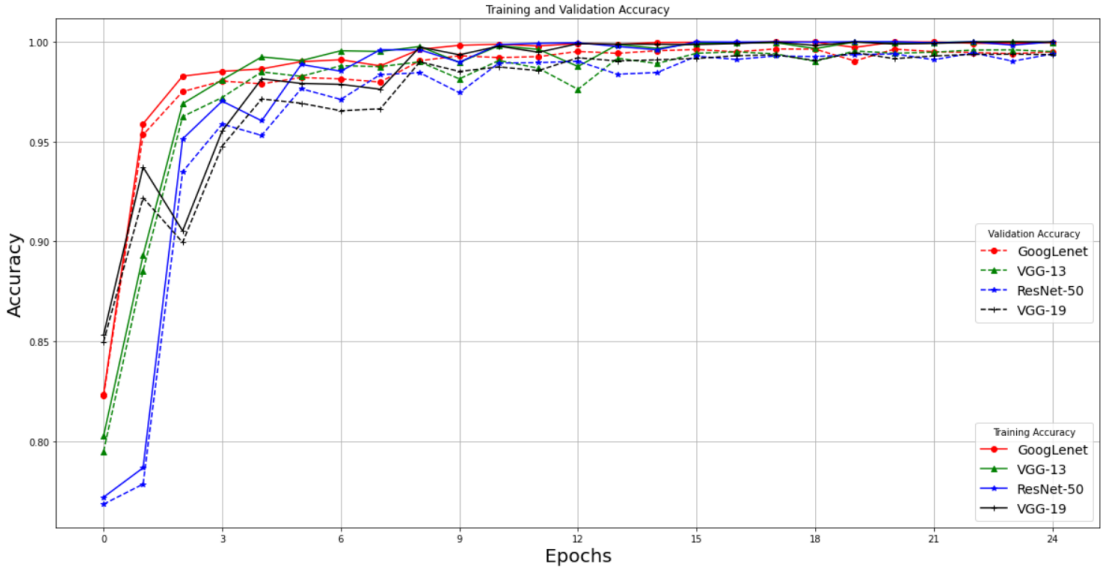


(c) VGG-11, VGG-16, ResNet-152, ResNet-101

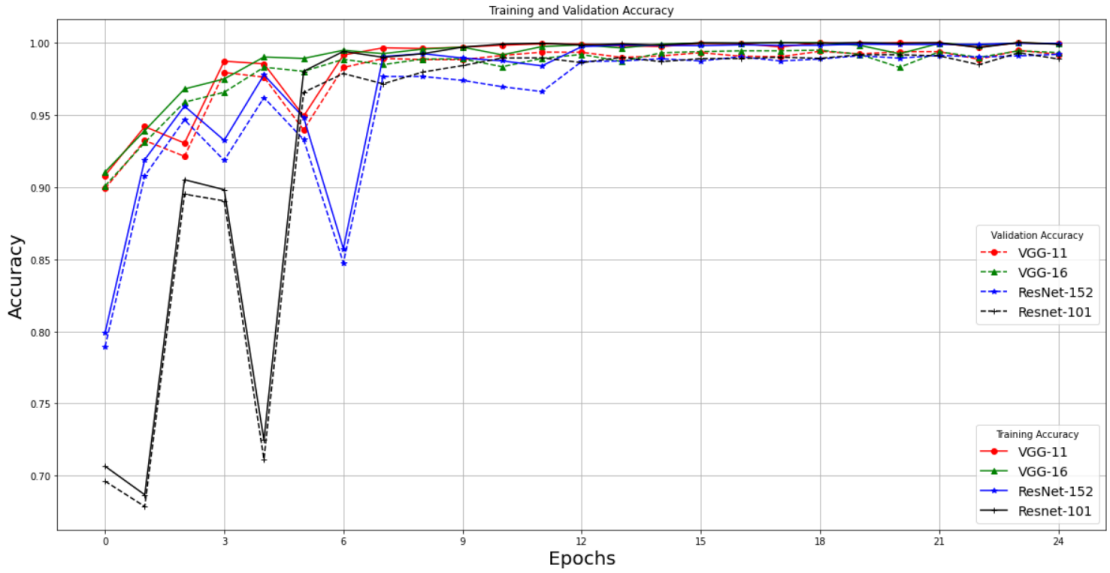
Figure 7.1.1(a): Training on PlantDiseaseIdentification dataset - Training loss and Validation loss after each epoch of (a) AlexNet, ZFNet, (b) GoogLeNet, VGG-13, ResNet-50, VGG-19, (c) VGG-11, VGG-16, ResNet-152, ResNet-10



(a) AlexNet, ZFNet



(b) GoogLeNet, VGG-13, ResNet-50, VGG-19



(c) VGG-11, VGG-16, ResNet-152, ResNet-101

Figure 7.1.1(b): Training on PlantDiseaseIdentification dataset - Training and Validation accuracy after each epoch of (a) AlexNet, ZFNet, (b) GoogLeNet, VGG-13, ResNet-50, VGG-19, (c) VGG-11, VGG-16, ResNet-152, ResNet-101

7.1.2. Comparative Analysis

As presented in Figure 7.1.1(a) and Figure 7.1.1(b), the performance of deep CNN architectures indicates that overfitting and underfitting has not occurred. Since the dataset used is balanced unlike PlantVillage dataset, the models achieved higher accuracy and low loss rate after training them for 25 epochs. Overall, for the purpose of comparative study ten different CNN architectures were considered. These are AlexNet, ZFNet, VGG-11, VGG-13, VGG-16, VGG-19, GoogleNet, ResNet-50, ResNet-101, ResNet-152. From Table 7.1.1, Figure 7.1.1(a) and Figure 7.1.1(b), a few observations were made:

- The GoogLeNet model attained the highest F1-score, precision, recall, test accuracy and lowest test loss among all the models. It achieved second highest validation accuracy i.e., 0.0002 value less than the highest validation accuracy attained by VGG13 model and second lowest validation loss i.e., 0.004 value more than the lowest validation loss attained by again VGG-13 model. Therefore, for the augmented version of the PlantVillage dataset, GoogLeNet can be considered as the best CNN architecture among all the models used for doing analysis. It implies that

the concept of adding Inception blocks in the architecture is useful for obtaining higher identification results. Moreover, this model has least number of trainable parameters i.e., 6 M among all the models considered. Therefore, the training is faster for this model as compared to others.

- VGG-13, VGG-19 and ResNet-50 attained the second highest F1-score, precision and recall. VGG-16 and ResNet-152 attained the third highest F1-score, precision and recall i.e., 0.001 value less than the second highest F1-score, precision and recall. ResNet-50 attained third highest validation accuracy followed by VGG-19, VGG-16, ResNet-152, VGG-11 and ResNet-101. ResNet-152 and VGG-11 models attained same validation accuracy. VGG-19 attained the third lowest validation loss followed by ResNet-50, VGG-16, ResNet-152 and VGG-11. ResNet-50 achieved the second highest test accuracy and second lowest test loss followed by VGG-19, VGG-13, VGG-16, ResNet-152 and VGG-11. Therefore, all these models gave comparable results. As ResNet-50 has 23 M trainable parameters that is around 6x times less than the number of trainable parameters for all the four models of VGG analyzed. Since ResNet-101 and ResNet-152 have more number of convolution layers, and therefore has more number of trainable parameters as compared to ResNet-50, which in turn increased the training time. Therefore, ResNet-50 can be considered as second best model for 'PlantDiseaseIdentification' dataset.
- VGG-13, VGG-19 and ResNet-50 attained comparable results. However, the number of trainable parameters in VGG-13 is less than VGG-19. Therefore, VGG-13 can be considered as third best model for 'PlantDiseaseIdentification' dataset.
- ZFNet attained the least F1-score, precision and recall among all the models analyzed, followed by AlexNet. Both the models took five to six more epochs to converge as compared to other models analyzed as shown in the Figure 7.1.1(a) and Figure 7.1.1(b). They attained low validation and test accuracy and high validation and test loss among all the models.

7.2. Performance of pre-trained (trained on PlantDiseaseIdentification dataset) deep CNN based models trained on 'Cropped-PlantDoc_Aug' dataset using transfer learning

The 'PlantDiseaseIdentification' dataset contains leaf images taken in laboratory under controlled conditions. It does not contain leaf images taken directly from the field. For

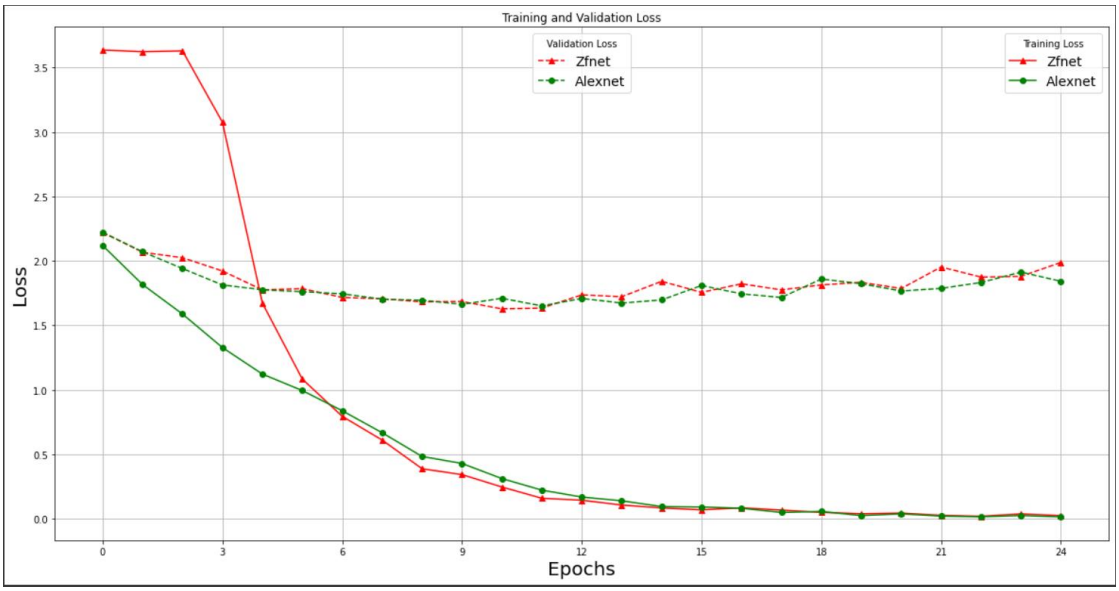
comparative analysis on dataset containing field images, '*Cropped-PlantDoc_Aug*' dataset is considered. Since, the number of images in Cropped-PlantDoc_Aug dataset is 22,260 which is less, therefore the pre-trained models i.e. models trained on PlantDiseaseIdentification dataset are used for training on field images.

7.2.1. Discussion

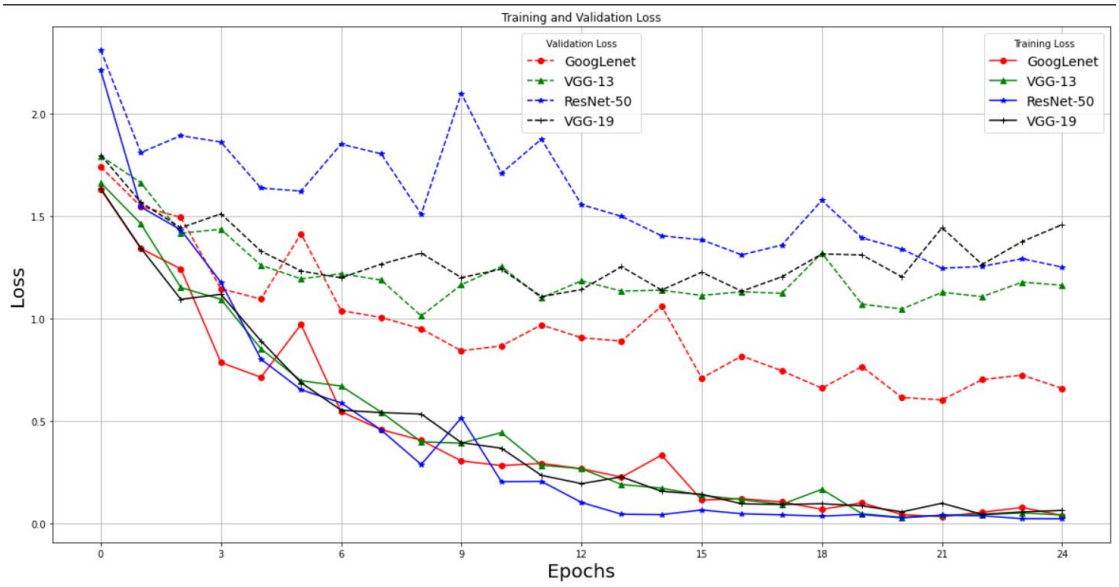
This section presents the comparative analysis of deep learning convolutional based neural network architectures in order to select the best model for the augmented version of the 'Cropped-PlantDoc' dataset i.e., '*Cropped-PlantDoc_Aug'* dataset. Training accuracy, validation accuracy, training loss, validation loss, testing accuracy, testing loss, precision, recall and F1-score were used for evaluating the results obtained. The model that achieved the highest F1-score was considered to be most suitable. The models were trained for about 25 epochs. In order to avoid overfitting, weight decay hyperparameter is applied having value 0.0005. The performances of deep learning architectures are represented by line graphs (Figure 7.2.1(a), Figure 7.2.1(b)). Table 7.2.1 shows the overall performance of deep CNN architectures.

Table 7.2.1: Performance of various deep learning architectures trained on 'Cropped-PlantDoc_Aug' dataset using transfer learning (i.e. pre-trained weights obtained by training the models on PlantDiseaseIdentification are applied as initial weights)

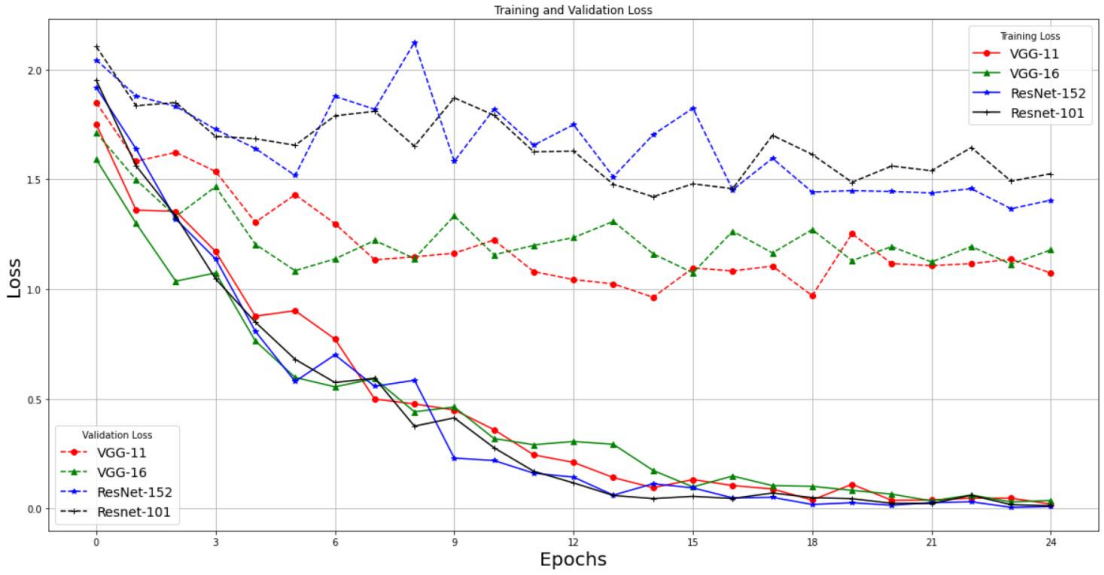
Deep Learning Architectures	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Precision	Recall	F1-score	Test Accuracy	Test Loss
AlexNet	0.9975	0.0160	0.5328	1.8422	0.542	0.538	0.532	0.5381	1.9085
ZFNet	0.9951	0.0273	0.5261	1.9885	0.547	0.534	0.528	0.5357	2.0561
VGG-11	0.9958	0.0195	0.7324	1.0728	0.743	0.740	0.733	0.7337	1.0794
VGG-13	0.9878	0.0395	0.7180	1.1621	0.734	0.712	0.710	0.7100	1.1722
VGG-16	0.9887	0.0360	0.7261	1.1781	0.732	0.716	0.715	0.7181	1.1934
VGG-19	0.9802	0.0619	0.6846	1.4582	0.712	0.697	0.693	0.6964	1.4067
GoogleNet	0.9899	0.0373	**0.8203**	**0.6579**	**0.823**	**0.817**	**0.815**	**0.8170**	**0.6717**
ResNet-50	0.9984	0.0209	0.7157	1.2504	0.701	0.700	0.696	0.7000	1.2115
ResNet-101	0.9968	**0.0120**	0.6557	1.5254	0.662	0.656	0.651	0.6547	1.5383
ResNet-152	0.9978	0.0087	0.6706	1.4049	0.688	0.680	0.678	0.6781	1.3797



(a) AlexNet, ZFNet

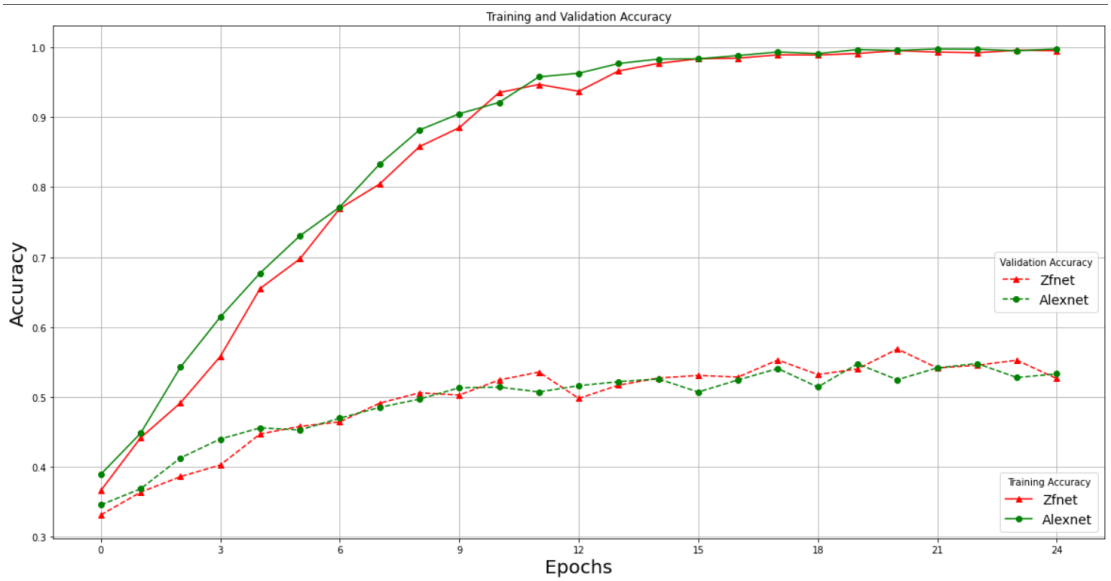


(b) GoogLeNet, VGG-13, ResNet-50, VGG-19

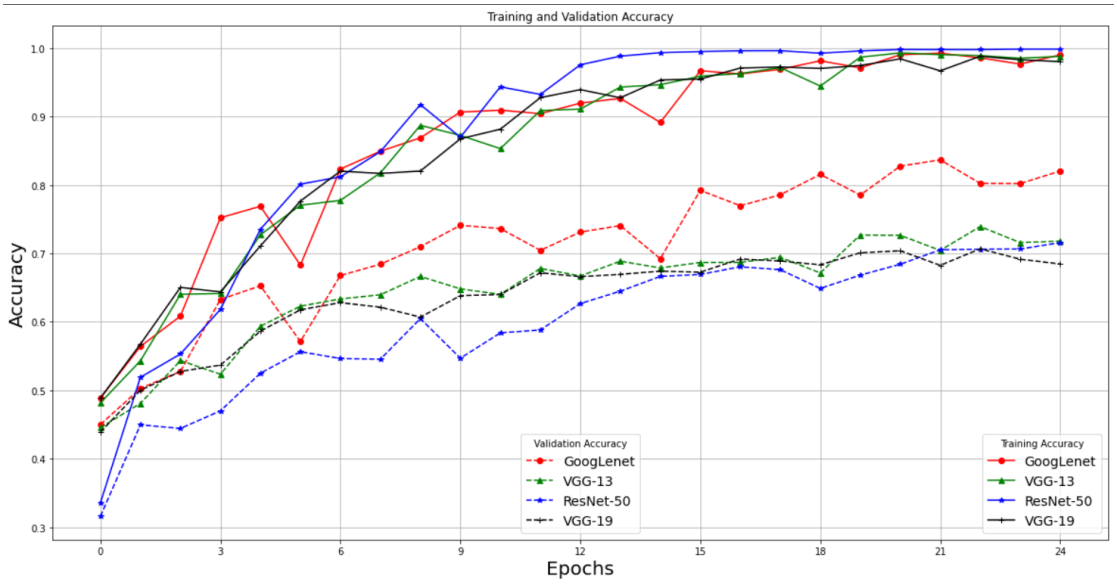


(c) VGG-11, VGG-16, ResNet-152, ResNet-101

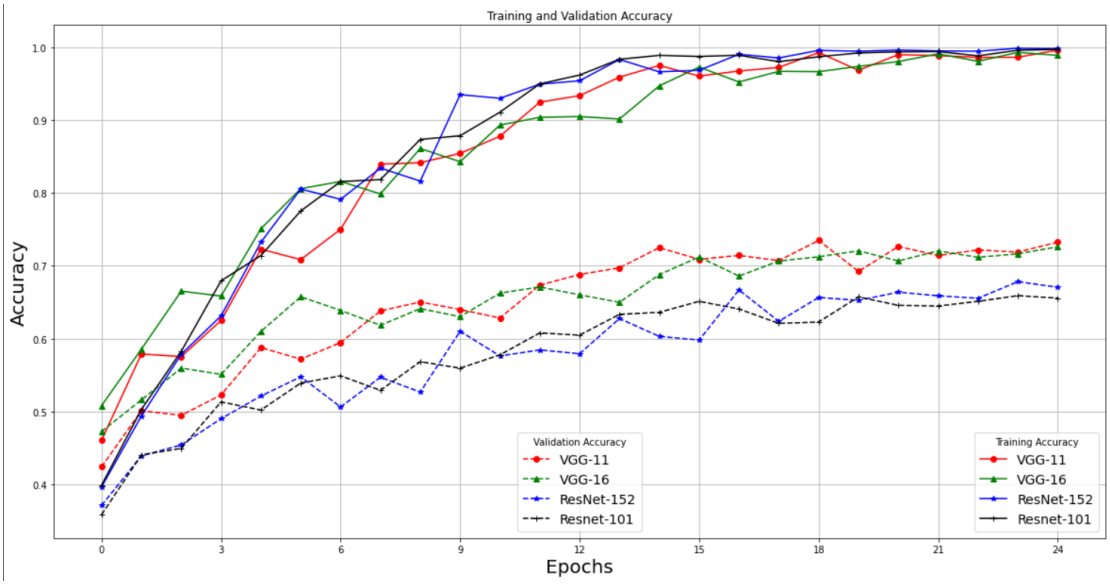
Figure 7.2.1(a): Transfer learning on Cropped-PlantDoc_Aug dataset using pre-trained models trained on PlantDiseaseIdentification dataset - Training and Validation loss after each epoch of (a) AlexNet, ZFNet, (b) GoogLeNet, VGG-13, ResNet-50, VGG-19, (c) VGG-11, VGG-16, ResNet-152, ResNet-101



(a) AlexNet, ZFNet



(b) GoogLeNet, VGG-13, ResNet-50, VGG-19



(c) VGG-11, VGG-16, ResNet-152, ResNet-101

Figure 7.2.1(b): Transfer learning on Cropped-PlantDoc_Aug dataset using pre-trained models trained on PlantDiseaseIdentification dataset - Training and Validation Accuracy after each epoch of (a) AlexNet, ZFNet, (b) GoogLeNet, VGG-13, ResNet-50, VGG-19, (c) VGG-11, VGG-16, ResNet-152, ResNet-101

7.2.2. Comparative Analysis

As presented in Figure 7.2.1(a) and Figure 7.2.1(b), the performance of deep CNN architectures indicates that slight overfitting has occurred while training AlexNet and

ZFNet models. The training accuracy achieved by all the models is more than 98%, whereas validation accuracy did not go beyond 72% except GoogLeNet that achieved 82.03% of validation accuracy. From Table 7.2.1, Figure 7.2.1(a) and Figure 7.2.1(b), a few observations were made:

- The GoogLeNet model attained the highest F1-score, precision, recall, test accuracy and lowest test loss among all the models. Also, the highest validation accuracy and lowest validation loss is achieved by it. GoogLeNet outperformed among other models. Therefore, for the augmented version of the Cropped-PlantDoc dataset, GoogLeNet can be considered as the most suitable CNN architecture among all the models used for doing analysis when pre-trained on PlantDiseaseIdentification dataset. It implies that the concept of adding Inception blocks in the architecture is useful for obtaining higher identification results. Moreover, this model has least number of trainable parameters i.e., 6 M among all the models considered. Therefore, the training is faster for this model as compared to others.
- The second highest F1-score and test accuracy was obtained by VGG-11, followed by VGG-16, VGG-13, ResNet-50 and VGG-19. VGG-11 attained the second highest validation accuracy.
- The validation accuracy of ResNet-50 and of all the models of VGG is in the range 71 to 74 percent. ResNet-101 and ResNet-162 achieved validation accuracy of 65 percent and 67 percent respectively.
- The AlexNet and ZFNet depicted poor performance. The validation accuracy attained by then was below 54 percent. This might be because of less number of weight layers in the AlexNet and ZFNet models.

7.3. Performance of deep CNN based models trained on 'PlantDiseaseIdentification' dataset using transfer learning

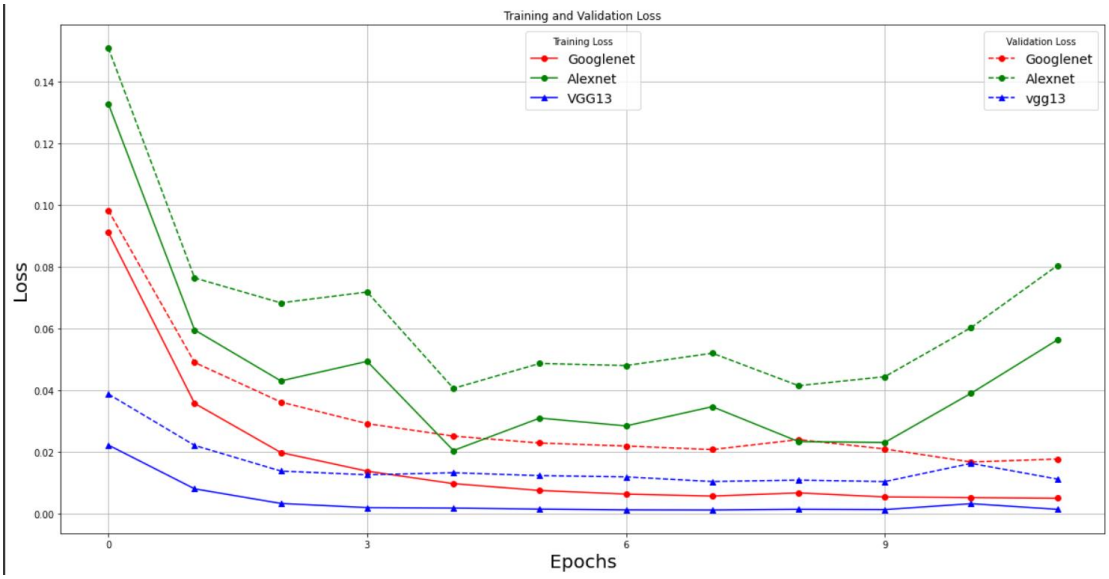
7.3.1. Discussion

This section presents the comparative analysis of deep learning convolutional based neural network architectures trained on augmented version of the PlantVillage dataset i.e., '*PlantDiseaseIdentification'* using transfer learning. The pre-trained ImageNet weights of all the models (except ZFNet) were downloaded from PyTorch library and these weights are used as initial weights for the models. The pre-trained ImageNet weights are not available in PyTorch. Hence, ZFNet is excluded from this analysis.

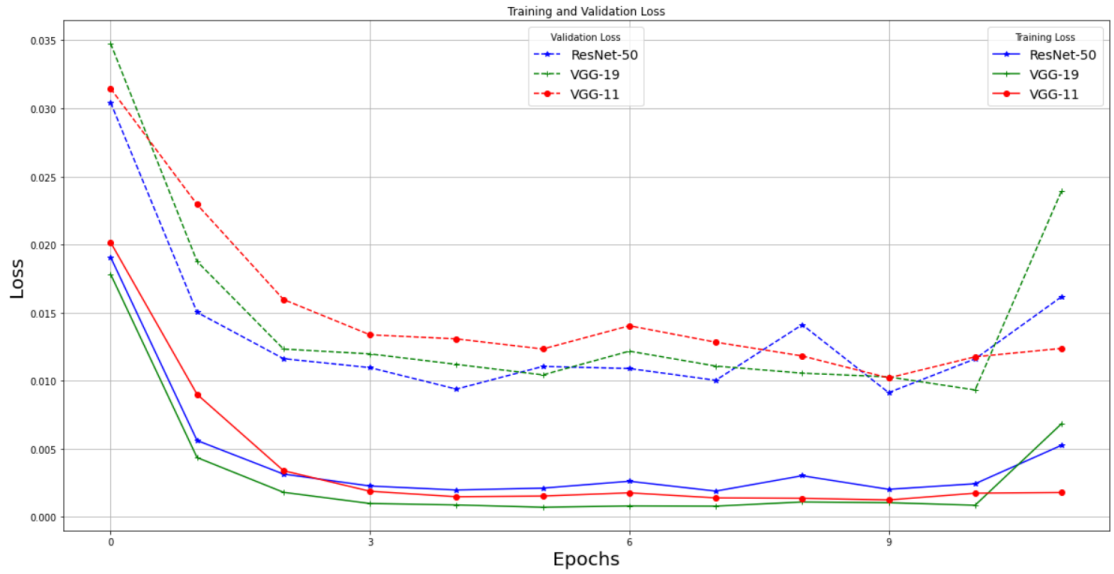
Each pre-trained model (trained on ImageNet) is trained on the augmented version of PlantVillage dataset first. Training accuracy, validation accuracy, training loss, validation loss, testing accuracy, testing loss, precision, recall and F1-score are used for evaluating the results obtained. It is observed that training the models for about 12 epochs, the training as well as the validation accuracy and loss are converged. To prevent overfitting, weight decay of 0.003 is applied. The performances of deep learning architectures are represented by line graphs (Figure 7.3.1(a), Figure 7.3.1(b)). Table 7.3.1 shows the overall performance of deep CNN architectures.

Table 7.3.1: Performance of various deep learning architectures on 'PlantDiseaseIdentification' dataset (transfer learning using ImageNet weights)

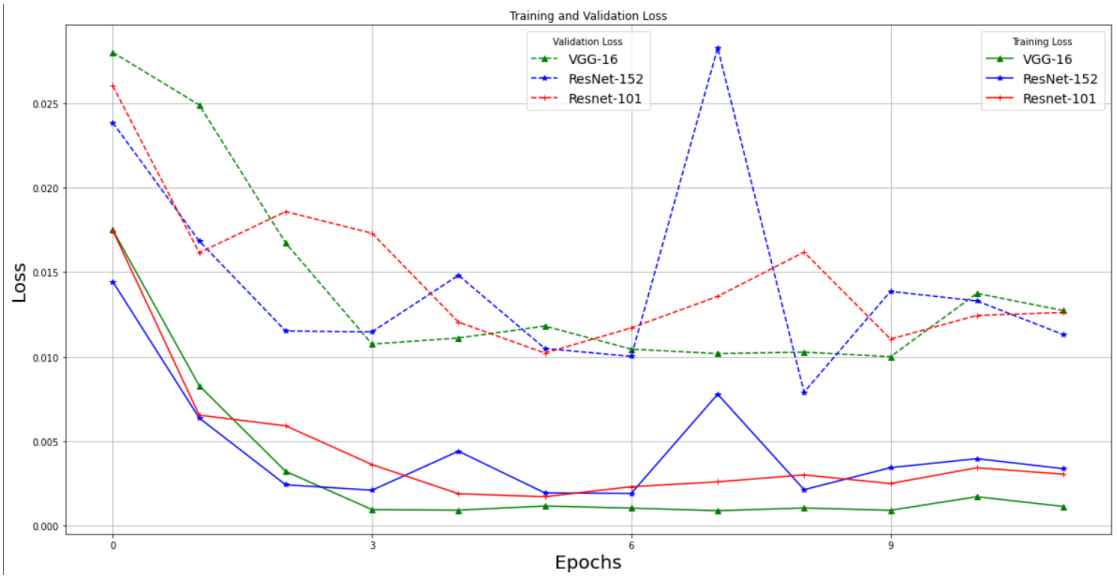
Deep Learning Architectures	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Precision	Recall	F1-score	Test Accuracy	Test Loss
AlexNet	0.9825	0.0563	0.9749	0.0804	0.976	0.974	0.974	0.9743	0.0823
VGG-11	0.9999	0.0017	0.9969	0.0123	0.997	0.997	0.997	0.9969	0.0126
VGG-13	0.9999	0.0015	0.9970	0.0113	0.997	0.997	0.997	0.9970	0.0119
VGG-16	0.9999	0.0011	0.9969	0.0127	0.997	0.997	0.997	0.9973	0.0125
VGG-19	0.9981	0.0068	0.9934	0.0239	0.994	0.993	0.993	0.9933	0.0219
GoogleNet	0.9999	0.0051	0.9965	0.0178	0.996	0.996	0.996	0.9962	0.0169
ResNet-50	0.9993	0.0052	0.9964	0.0161	0.995	0.995	0.995	0.9952	0.0163
ResNet-101	0.9999	0.0030	0.9976	0.0126	0.998	0.998	**0.998**	0.9976	0.0106
ResNet-152	0.9997	0.0033	0.9976	0.0112	0.998	0.998	**0.998**	0.9978	0.0097



(a) AlexNet, VGG-13, GoogLeNet

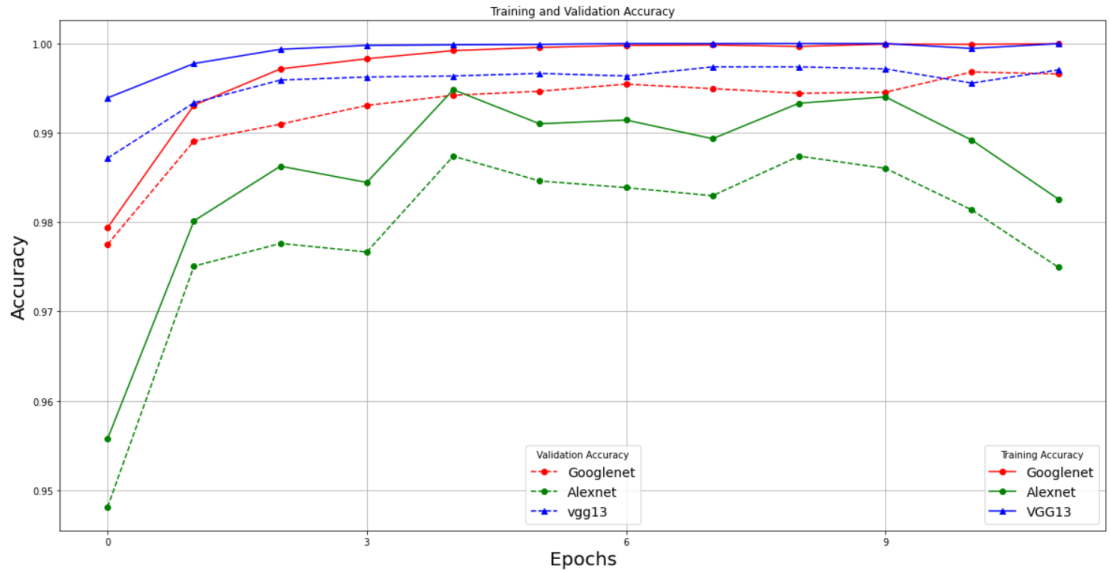


(b) VGG-11, VGG-19, ResNet-50

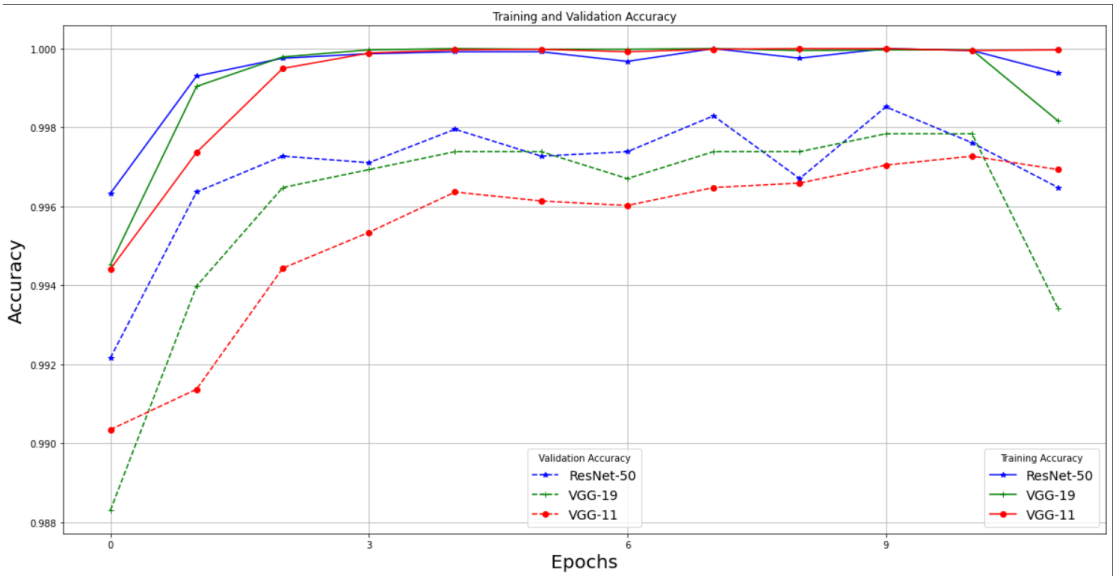


(c) VGG-16, ResNet-101, ResNet-152

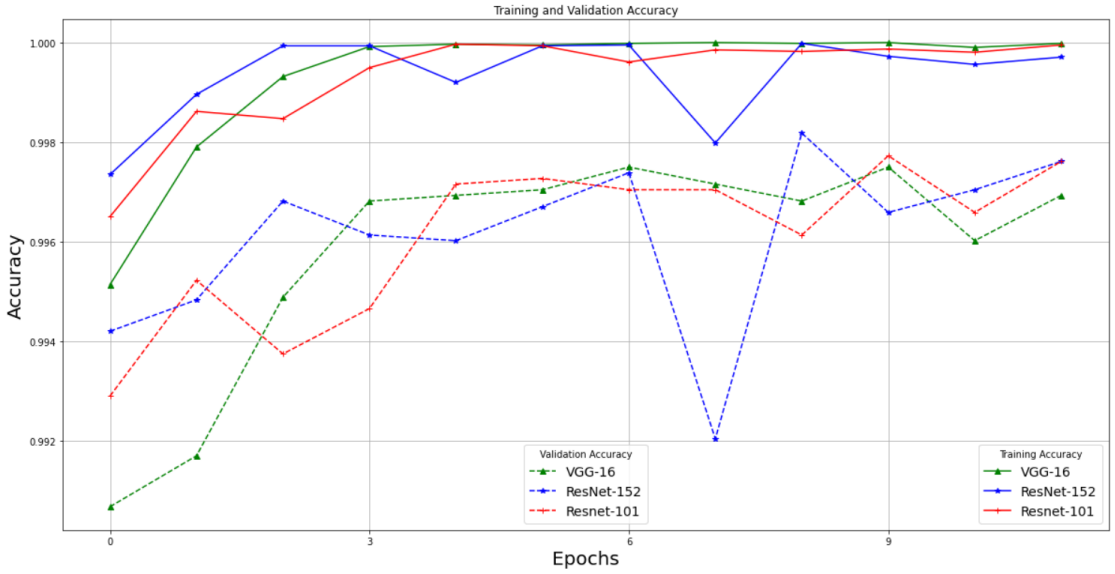
Figure 7.3.1(a): Transfer learning on PlantDiseaseIdentification dataset using pre-trained models trained on ImageNet dataset - Training and Validation loss after each epoch of (a) AlexNet, VGG-13, GoogLeNet, (b) VGG-11, VGG-19, ResNet-50, (c) VGG-16, ResNet-101, ResNet-152



(a) AlexNet, VGG-13, GoogLeNet



(b) VGG-11, VGG-19, ResNet-50



(c) VGG-16, ResNet-101, ResNet-152

Figure 7.3.1(b): Transfer learning on PlantDiseaseIdentification dataset using pre-trained models trained on ImageNet dataset - Training and Validation accuracy after each epoch of (a) AlexNet, VGG-13, GoogLeNet, (b) VGG-11, VGG-19, ResNet-50, (c) VGG-16, ResNet-101, ResNet-152

7.3.2. Comparative Analysis

As presented in Figure 7.3.1(a) and Figure 7.3.1(b), the performance of deep CNN architectures indicates that overfitting and underfitting has not occurred. All the models achieved higher accuracy and low loss rate after training them for 12 epochs. From Table 7.3.1, Figure 7.3.1(a) and Figure 7.3.1(b), a few observations were made:

- The results obtained are comparable to the results that are obtained earlier (without transfer learning) as presented in Table 7.1.1.
- Except AlexNet, the F1-score achieved by all the models was more than .99 (range: .993 for VGG-19 to .998 for ResNet-101 and ResNet-152). The F1-score achieved with AlexNet was .974.
- The validation accuracy and test accuracy were also above 99% for all the models except AlexNet.

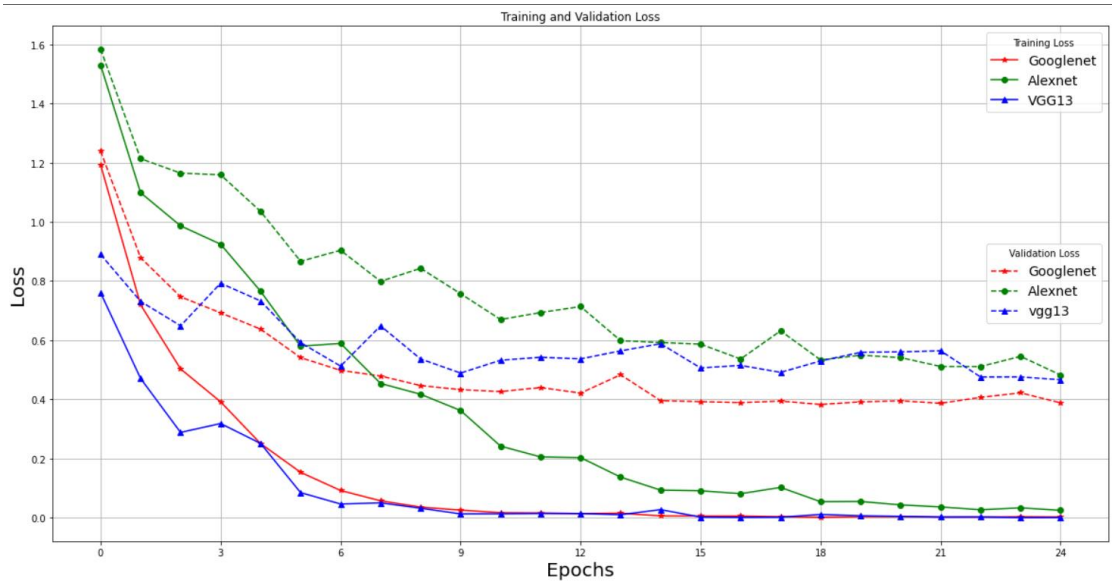
7.4. Performance of deep CNN based models trained on 'Cropped-PlantDoc_Aug' dataset using transfer learning

7.4.1. Discussion

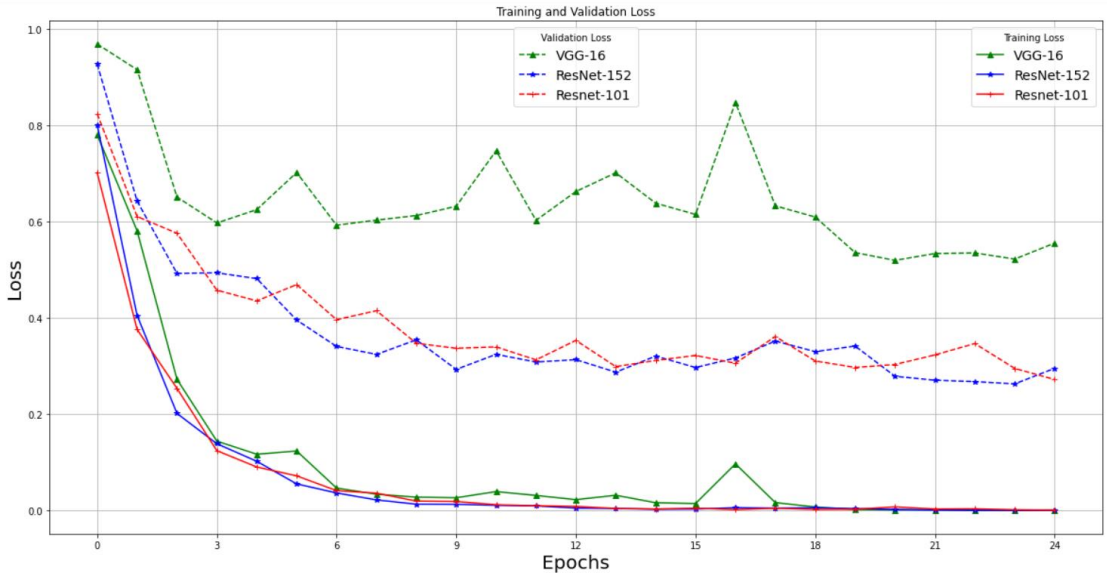
This section presents the comparative analysis of deep learning convolutional based neural network architectures trained on the augmented version of the Cropped-PlantDoc dataset i.e., '*Cropped-PlantDoc_Aug'* dataset using transfer learning. Training accuracy, validation accuracy, training loss, validation loss, testing accuracy, testing loss, precision, recall and F1-score were used for evaluating the results obtained. It was observed that training the models for about 25 epochs the training as well as the validation accuracy and loss were converged. To avoid overfitting, weight decay of 0.0005 was used. The performances of deep learning architectures are represented by line graphs (Figure 7.4.1(a), Figure 7.4.1(b)). Table 7.4.1 shows the overall performance of deep CNN architectures.

Table 7.4.1: Performance of various deep learning architectures on Cropped-PlantDoc_Aug dataset (transfer learning using ImageNet + PlantDiseaseIdentification)

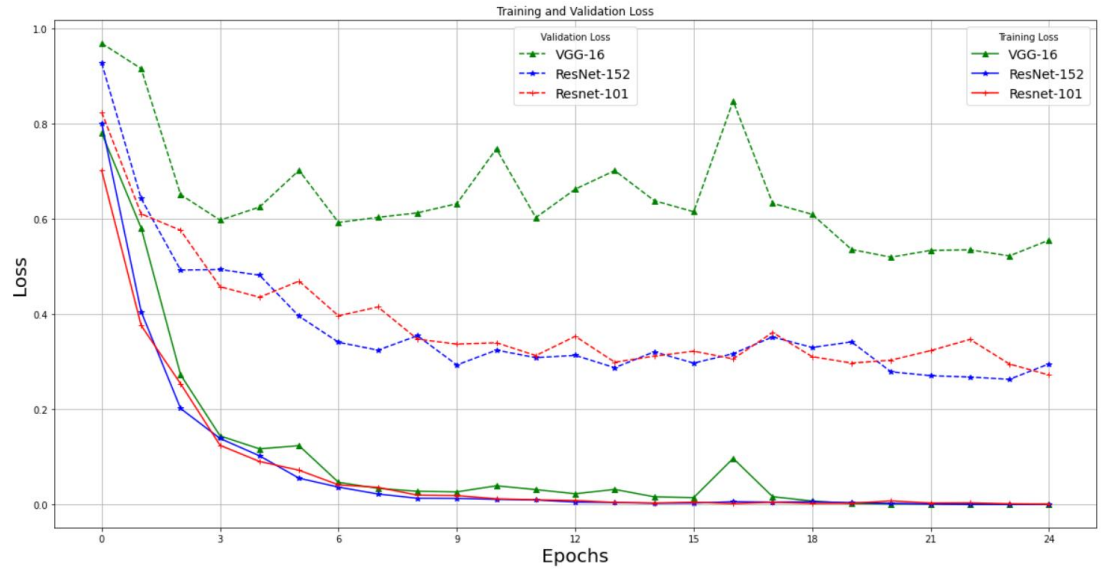
Deep Learning Architectures	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Precision	Recall	F1-score	Test Accuracy	Test Loss
AlexNet	0.9949	0.0251	0.8585	0.4825	0.856	0.853	0.853	0.8535	0.5380
VGG-11	0.9973	0.0096	0.8585	0.5545	0.866	0.858	0.859	0.8627	0.5706
VGG-13	0.9997	0.0009	0.8777	0.4662	0.879	0.879	0.878	0.8808	0.4917
VGG-16	0.9998	0.0013	0.8819	0.5549	0.869	0.868	0.867	0.8691	0.5955
VGG-19	0.9996	0.0018	0.8656	0.5708	0.850	0.847	0.846	0.8493	0.6326
GoogLeNet	0.9998	0.0028	0.8879	0.3882	0.879	0.876	0.876	0.8783	0.4322
ResNet-50	0.9998	0.0011	0.9194	0.2856	0.914	0.912	0.912	0.9142	0.3100
ResNet-101	0.9997	0.0008	**0.9267**	**0.2719**	**0.923**	**0.921**	**0.921**	**0.9223**	**0.2955**
ResNet-152	0.9999	0.0004	0.9182	0.2951	0.920	0.920	0.919	0.9207	**0.2807**



(a) AlexNet, VGG-13, GoogLeNet

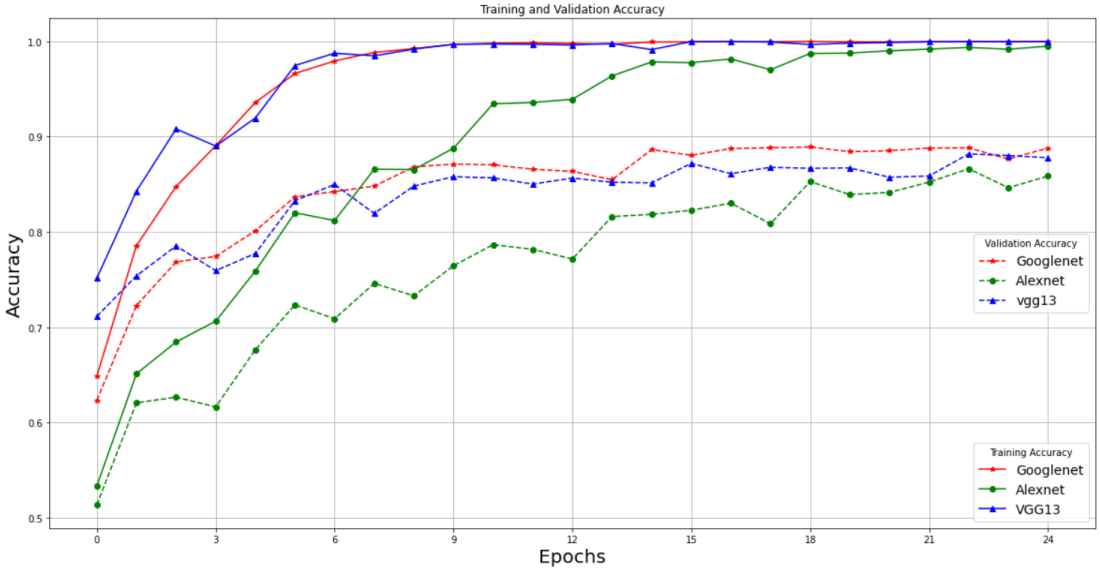


(b) VGG-11, VGG-19, ResNet-50

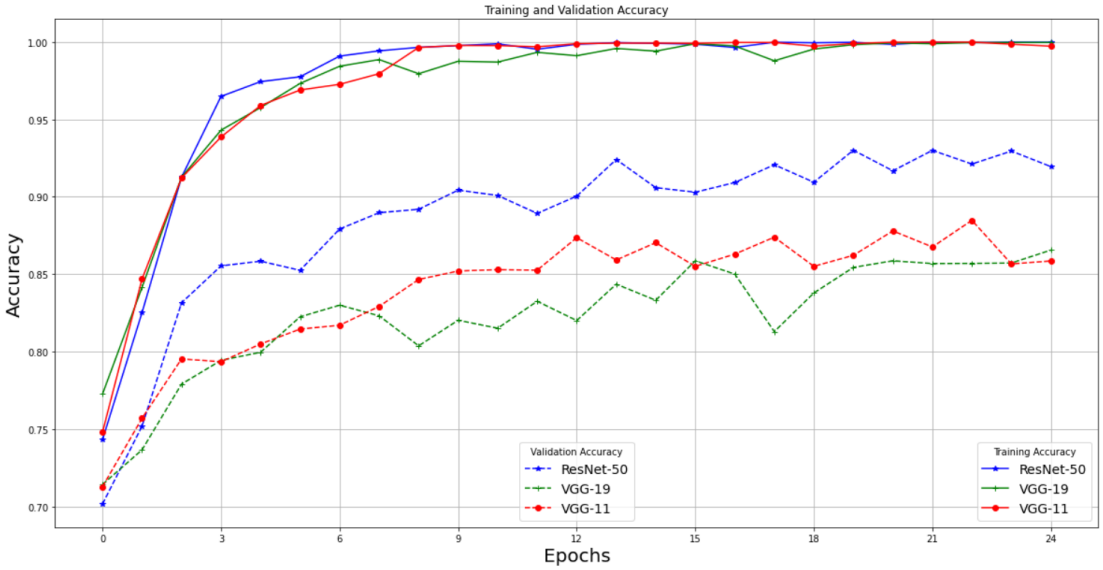


(c) VGG-16, ResNet-101, ResNet-152

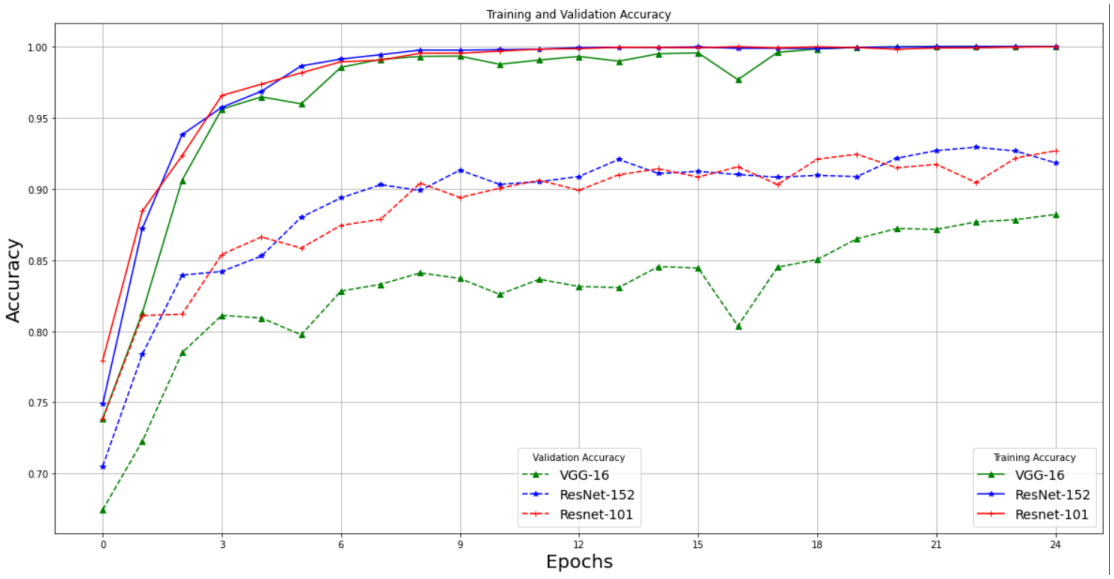
Figure 7.4.1(a): Transfer learning on Cropped-PlantDoc_Aug dataset using pre-trained models trained on ImageNet plus PlantDiseaseIdentification dataset - Training and Validation loss after each epoch of (a) AlexNet, VGG-13, GoogLeNet, (b) VGG-11, VGG-19, ResNet-50, (c) VGG-16, ResNet-101, ResNet-152



(a) AlexNet, VGG-13, GoogLeNet



(b) VGG-11, VGG-19, ResNet-50



(c) VGG-16, ResNet-101, ResNet-152

Figure 7.4.1(b): Transfer learning on Cropped-PlantDoc_Aug dataset using pre-trained models trained on ImageNet plus PlantDiseaseIdentification dataset - Training and Validation accuracy after each epoch of (a) AlexNet, VGG-13, GoogLeNet, (b) VGG-11, VGG-19, ResNet-50, (c) VGG-16, ResNet-101, ResNet-152

7.4.2. Comparative analysis

As presented in Figure 7.4.1(a) and Figure 7.4.1(b), the performance of deep CNN architectures indicates that overfitting is reduced to some extent. It seems that using ImageNet and PlantDiseaseIdentification pre-trained weights, the models achieved higher accuracy and low loss rate after training them for 25 epochs. From Table 7.4.1, Figure 7.4.1(a) and Figure 7.4.1(b), a few observations are made:

- The highest F1-score is achieved by ResNet-101, followed by ResNet-152 and ResNet-50. By applying ImageNet pre-trained weights, the F1-score of ResNet models is increased by about 20-25 value.
- Also, ResNet-101 achieved highest validation and test accuracy and lowest validation loss. The validation accuracy and test accuracy also increased by about 20-25 percent.
- The F1-score as well as validation and test accuracy also increased for other models. Performance of GoogLeNet model is improved by 6 percent, whereas for VGG models it improved by 15 percent (for VGG-19), 16 percent (for VGG-13 and VGG-16), and 13 percent (for VGG-11).

- There is a major improvement in the performance of AlexNet model. The F1-score, validation accuracy as well as test accuracy improved by 32 percent. It seems that using models that are already trained on a large dataset like ImageNet, the models tend to perform better.
- Performing transfer learning by applying only PlantDiseaseIdentification pre-trained weights, GoogLeNet achieved the highest F1-score. But after performing transfer learning by applying ImageNet pre-trained weights as well as PlantDiseaseIdentification pre-trained weights, ResNet outperformed GoogLeNet and achieved the highest F1-score among all the models.
- Therefore, for the augmented version of Cropped-PlantDoc dataset, ResNet-101 can be considered as the most suitable model followed by ResNet-152 and ResNet-50.

CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1. Conclusion

In this project, a comparative analysis has been performed among various standard well known deep learning models for plant disease identification task. For this task, comparative analysis was performed separately on two datasets: 1) PlantDiseaseIdentification dataset containing 87,900 leaf images taken in laboratory under controlled conditions, 2) Cropped-PlantDoc_Aug dataset containing 22,260 leaf images taken in field conditions. Except AlexNet and ZFNet, all the CNN models achieved more than .99 F1-score for PlantDiseaseIdentification dataset. The highest F1-score, test accuracy and lowest test loss was attained by GoogLeNet, followed by ResNet-50 and VGG-13. For performing comparative analysis on field images, transfer learning was performed on Cropped-PlantDoc_Aug dataset by using pre-trained models trained on ImageNet plus PlantDiseaseIdentification dataset. All the models achieved F1-score of more than .85. The highest F1-score, test accuracy and validation accuracy was obtained by ResNet-101, followed by ResNet-152 and ResNet-50.

8.2. Future Work

In future the performance of other standard and modified versions of convolutional models can also be analysed for plant disease identification task. In order to further improve the validation accuracy of models for field images, regularization methods and hyperparameter tuning can be performed. Various deep learning based optimizers such as Rmsprop, Adam, Adagrad and Adadelta can also be used to enhance the performance of the models.

CHAPTER 9

PAPER COMMUNICATED TO THE CONFERENCE

Paper entitled ‘Comparison of Deep CNN Models for Plant Disease Identification’ is communicated to ICIASF 2021: 15. International Conference on Intelligent agriculture and Smart Farming. The conference to be held in online mode on September 09-10, 2021 in Tokyo, Japan.

Conference Code: 21JP09ICIASF

The paper communicated is under review.

REFERENCES

[1] IPPC Secretariat, "Plant Health and Food Security," *Food and Agriculture Organization of the United Nation, International Plant Protection Convention*, (2017). Pamphlet I7829EN/1/09.17.

[2] S. P. Mohanty, D. P. Hughes and M. Salathé, "Using Deep Learning for Image-Based Plant Disease Detection," in *Frontiers in Plant Science*, vol. 7, 22 Sept 2016. DOI: *10.3389/fpls.2016.01419*

[3] H. Rahman et al., "A comparative analysis of machine learning approaches for plant disease identification," in *Advancements in Life Sciences – International Quarterly Journal of Biological Sciences*, pp. 120-126, Aug 2017.

[4] H. B. Prajapati, J. P. Shah and V. K. Dabhi, "Detection and Classification of Rice Plant Diseases," in *Intelligent Decision Technologies, IOS Press*, pp. 357-373, 29 Aug 2017. DOI: *10.3233/IDT-170301*.

[5] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," in *Computers and Electronics in Agriculture*, pp. 311-318, 2018. DOI: *https://doi.org/10.1016/j.compag.2018.01.009*.

[6] J. Boulent, S. Foucher, J. Théau and P. St-Charles, "Convolutional Neural Networks for the Automatic Identification of Plant Diseases," in *Front. Plant Sci.*, 23 July 2019.

[7] P. Alagumariappan *et al.*, "Intelligent Plant Disease Identification System Using Machine Learning," in *Eng. Proc.*, 14 Nov 2020.

[8] J. Chen, J. Chen D. Zhang, Y. Sun and Y. A. Nanehkaran, "Using deep transfer learning for image-based plant disease identification," in *Computers and Electronics in Agriculture*, 2020. DOI: *https://doi.org/10.1016/j.compag.2020.105393*.

[9] A. Sagar, D. Jacob, "On Using Transfer Learning For Plant Disease Detection," *bioRXiv preprint*, 25 May, 2021.
DOI: https://doi.org/10.1101/2020.05.22.110957.

[10] S. Bhattarai, "New Plant Disease Dataset," *Kaggle.com,* 2018. [Online]. Available: https://www.kaggle.com/vipoooool/new-plant-diseases-dataset. [Accessed: 21- Apr- 2021].

[11] D. P. Hughes and M. Salathé, “An open access repository of images on plant health to enable the development of mobile disease diagnostics,” *arXiv:1511.08060*, 2015.

[12] D. Singh *et al.*, “PlantDoc: A Dataset for Visual Plant Disease Detection,” *arXiv:1911.10317v1 [cs.CV]*, 23 Nov 2019.

[13] "pratikkayal/PlantDoc-Object-Detection-Dataset", *GitHub*, 16 Oct, 2019. [Online]. Available: https://github.com/pratikkayal/PlantDoc-Object-Detection-Dataset. [Accessed: 20- May- 2021].

[14] A. Khan, A. Sohail, U. Zahoora and A. S. Qureshi, “A Survey of the Recent Architectures of Deep Convolutional Neural Networks,” in *Artificial Intelligence Review*, 21 Apr 2020. DOI: *https://doi.org/10.1007/s10462-020-09825-6*.

[15] A. Krizhevsky, I. Sutskever and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *NIPS*, 2012.

[16] M. D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” *arXiv:1311.2901v3 [cs.CV]*, 28 Nov 2013.

[17] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv:1409.21556v6 [cs.CV]*, 10 Apr 2015.

[18] C. Szegedy *et al.*, “Going deeper with convolutions,” *arXiv:1409.4842v1 [cs.CV]*, 17 Sep 2014.

[19] K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition,” *arXiv:1512.03385v1 [cs.CV]*, 10 Dec 2015.

[20] D. Sarkar, “A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning”, *Medium*, 15 Nov, 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a. [Accessed: 12- Jun- 2021].

[21] S. Kostadinov, “What Is Deep Transfer Learning and Why Is It Becoming So Popular?”, *Medium*, 16 Nov, 2019. [Online]. Available: https://towardsdatascience.com/what-is-deep-transfer-learning-and-why-is-it-becoming-so-popular-91acdcc2717a. [Accessed: 13- Jun- 2021].

[22] C. Tan *et al.*, “A Survey on Deep Transfer Learning,” *arXiv:1808.01974v1 [cs.LG]*, 6 Aug 2018.

[23] J. Brownlee, “A Gentle Introduction to Transfer Learning for Deep Learning”, *Machine Learning Mastery*, 16 Sept, 2019. [Online]. Available: https://machinelearningmastery.com/transfer-learning-for-deep-learning/. [Accessed: 15- Jun- 2021].

[24] S. Ruder, “Transfer Learning – Machine Learning’s Next Frontier”, 21 Mar 2017. [Online]. Available: https://ruder.io/transfer-learning/. [Accessed: 16- Jun- 2021].

[25] “Google Colaboratory”, *Colab.research.google.com*. [Online]. Available: https://colab.research.google.com/notebooks/intro.ipynb?utm_source=scs-index#recent=true. [Accessed: 01- May- 2021].

[26] “PyTorch documentation – PyTorch 1.8.1 documentation,” *PyTorch.org*. [Online]. Available: https://pytorch.org/docs/stable/index.html. [Accessed: 03- May- 2021].

[27] “Quickstart - MLflow 1.17.0 documentation,” Mlflow.org. [Online]. Available: https://www.mlflow.org/docs/latest/quickstart.html. [Accessed: 08- May- 2021].