Harnessing the Power of Deep Neural Networks for Accurate Leaf Disease Identification

A dissertation submitted in
partial fulfilment for the degree of

Master of Technology

in

Computer Science

by

Avir Mondal
Roll no. - [**CS2209**]

under the supervision of

Dr. Ujjwal Bhattacharya
Associate Professor
Compute Vision and Pattern Recognition Unit

INDIAN STATISTICAL INSTITUTE, KOLKATA
June, 2024

CERTIFICATE

This is to certify that the dissertation titled “**Harnessing the Power of Deep Neural Networks for Accurate Leaf Disease Identification**” submitted by **Avir Mondal** to the Indian Statistical Institute, Kolkata, as part of the requirements for the Master of Technology in Computer Science degree, is a true record of the research work done by the candidate under my supervision. I confirm that the dissertation meets all the necessary requirements of this institute.

12 June,2024

[illegible]

Dr. Ujjwal Bhattacharya
Associate Professor
Compute Vision and Pattern Recognition Unit,
Indian Statistical Institute,
Kolkata-700108

Acknowledgement

I am deeply grateful to my advisor, **Dr. Ujjwal Bhattacharya**, from the **Computer Vision and Pattern Recognition Unit** at the **Indian Statistical Institute, Kolkata**. His constant guidance, support, and encouragement have been crucial throughout my research journey. Under his supervision, I have learned how to conduct thorough and meaningful research. His insightful ideas have been a constant source of motivation and inspiration.

I would like to thank all the research scholars of our laboratory for their valuable suggestions and discussions. I am also thankful to all my friends for their continuous help and support. I appreciate everyone who has contributed to my growth, even if I haven't mentioned them by name here.

Declaration

I, **Avir Mondal**, with Roll No. **CS2209**, declare that the dissertation titled **”Harnessing the Power of Deep Neural Networks for Accurate Leaf Disease Identification”** is my original work for the **M. Tech in Computer Science** degree at the **Indian Statistical Institute, Kolkata**.

I confirm that all the content in this report is my own work and any external sources have been properly cited. I understand that plagiarism or using others’ work without acknowledgment is taken very seriously and will have consequences.

Avir Mondal

Avir Mondal
Roll no.- CS2209

Abstract

Many countries around the world depends on agriculture, as it helps reduce poverty, increase national income, and improve food security. However, plant diseases often impact food crops, leading to significant annual losses and economic setbacks in agriculture. The best solution of the problem is to identify the plant disease as soon as possible so that necessary steps can take. Traditionally, humans have identified plant diseases visually, but this method is often slow, and also the number of domain experts are less. Recently, there has been significant progress in using deep learning to classify plant diseases. However, the main problem is to collect the sufficient annotated image data to train these models effectively for plant disease classification. Also the limited training data can negatively affect the performance of **CNN** models. To address this, we designed a **Deep Convolutional Generative Adversarial Network (DCGAN)** to overcome the issues of over-fitting and to increase the dataset sizes. Here we worked on the dataset called **DiaMOS Plant dataset**, consisting of 3006 images of pear leaves of four classes (3 diseases and one healthy class). The dataset was very imbalanced, so we used **DCGAN** on the minority classes separately to enhance the dataset. We developed some **CNN models** for classification and compared with some **Pre-trained models (VGG16, ResNet50, Inception V3)**. The results showed an average increment of classification accuracy.

Keywords: DCGAN, Plant disease classification, CNN, DiaMOS plant dataset

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Agriculture is very important for many countries. It helps reduce poverty, grow the economy, and make sure there is enough food. However, plant diseases are a big problem. They damage crops and cause huge losses every year, that impact on farmers and the economy. With the world population growing and the demand for food increasing, it is important to find faster and better ways to detect plant diseases. Finding these diseases early is very necessary to protecting crops.

Traditionally, the domain experts check plants for diseases by looking at them. But this is very slow method and also the number of good domain experts are very less.

New technology, especially in deep learning and computer vision, offers good ways to identify plant diseases quickly and accurately. This helps farmers take the necessary actions to protect their crops. Using these systems makes finding diseases faster and easier, which helps farmers grow more crops and better handle plant diseases.

1.2 Introduction

Plant diseases are a big problem for farmers because they can reduce the amount of crops produced and their quality. Traditional ways to find these diseases involve experts looking at plants, which takes a lot of time and can be inaccurate. New technology, like computer vision, machine learning and deep learning, offers faster and more accurate ways to detect diseases.

Convolutional Neural Networks (CNNs) are very good at classifying images. They have been used successfully in many fields, including plant disease detection. However, **CNNs** need a lot of balanced data to work well. If some classes of data are very less, the CNN might not perform accurately.

The **DiaMOS**[1] dataset is an example of an imbalanced dataset. It contains images of pear leaves classified into four groups: **healthy, curl, spot**, and **slug**. The **healthy** and **curl** leaf classes have fewer images, making it harder to train an accurate model. This study aims to balance the dataset using **DCGAN**[2] and then test the performance of a some proposed CNN classifiers and compared with them and some pre trained models**(VGG16[3], ResNet50**[4]**, Inception V3[5]**).

1.3 Problem Definition

The DiaMOS dataset presents an imbalanced distribution of classes, with the **healthy** leaf and **curl** leaf classes. This imbalance can result in biased models that do not accurately classify all types of leaves. The main goals of this work are to:

- **Data Augmentation using DCGAN[2]:** Utilize DCGAN to generate synthetic images for the minority healthy and curl leaf classes. This augmentation aims to balance the dataset, providing a equal distribution of samples across all classes.

- **Development and Evaluation of some CNN Classifiers:** Train some Convolutional Neural Network (CNN) classifiers on the upgraded dataset and evaluate its performance in classifying the four classes of pear leaves.

- **Comparison with Pretrained Models:** Compare the performance of the proposed CNN classifiers with some well-known pretrained models, including VGG16[3], ResNet50[4], and Inception V3[5].

By addressing the class imbalance through data augmentation and data generation, this works mainly focus on the improvement of the accuracy of plant disease classification systems, contributing to more effective and efficient disease management in agriculture.

Chapter 2

Related Work

This section talks about new ways to classify plant diseases. In one study[6], authors looked at diseases affecting 11 different plants and showed how to identify these diseases from leaf images using deep CNN models. In this paper, we focus on classifying the pear leaf diseases using deep CNN models.

Another study[7] explored factors that affect plant disease classification by analyzing the same plants and diseases under different conditions. They used five methods: MobileNetV2, EfficientB0, InceptionV2, ResNet50, and VGG16. The study found that model performance drops a lot when using representative datasets, but deep learning is still a good technique for classifying plant diseases.

In another work, researchers used the Convolutional Block Attention Module (CBAM) to improve CNN classification accuracy. This module can be added to the CNN architecture without much extra cost. They used several CNN models, including ResNet50, EfficientNetB0, VGG19, MobileNetV2 and InceptionV3, and for transfer learning in plant disease classification. The EfficientNetB0 and CBAM combination achieved 86.89% accuracy.

One study[8] looked into using ensemble learning to create a strong network for predicting four different pear leaf diseases. They tested several CNN models, including InceptionV3, EfficientNetB0, MobileNetV2, and VGG19.

Another study[9] used an improved YOLOv4-tiny model for detecting root crops with a single-board computer. The method could process up to 14 images of pixel 416×416 with 91% recall and 86% precision.

Balancing the data in deep learning is an important issue that has been researched a lot. It's crucial for good agricultural results, and it needs to be cost-effective, accurate, sensitive, and easy to use.

While data augmentation is useful for handling limited training data, it's not always easy to apply. Choosing the right data augmentation method requires knowing a lot about the specific task and domain. Also, setting the parameters for these methods can greatly affect the accuracy of the deep learning model.

Chapter 3

Dataset

3.1 Dataset Description

In this study, we use a field dataset known as the DiaMOS Plant[1] dataset to diagnose and monitor plant symptoms. DiaMOS Plant is an experimental dataset that includes images taken throughout an entire growing season of a pear tree, from February to July. This dataset aims to provide a comprehensive sample that captures the key cultural aspects of this plant. It is well-suited for applying machine learning and deep learning techniques for classification and detection tasks. The dataset comprises a total of 3505 images, with 499 images of fruits and 3006 images of leaves.

Table: Dataset Description

DiaMOS Plant Dataset	**Description**
Plant	Pear
Data Source Location	Italy
Type of data	RGB Images
Annotation	csv, YOLO
ROI (Region of Interest)	leaf, fruit
Total size	3505 images (3006 leaf + 499 fruit)
Data Accessibility	`https://doi.org/10.5281/zenodo.5557313`
Accessed on	17 October 2021

Table 3.1: Description of the **DiaMOS Plant**[1] Dataset

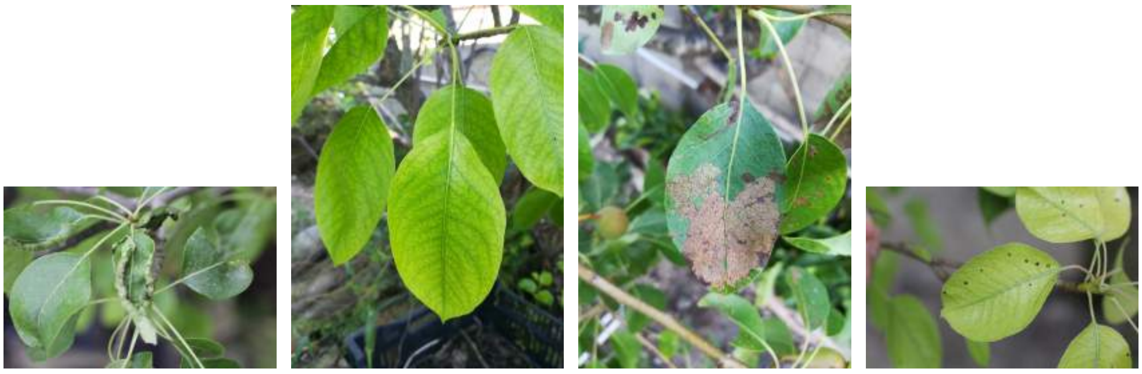
The images were collected using different devices, including a smartphone (Honor 6) and a DSLR camera (Canon EOS 60D). As a result, the images come in two resolutions: 2976 x 3968 and 3456 x 5184.

In our study, we mainly focus on classifying the pear leaf images. There are total 3006 leaf images of 4 different classes: healthy leaf, slug leaf, curl leaf, and spot leaf and the number of images are 43, 2025, 54 and 884 respectively.

Table: Distribution of Classes in Leaf Images

Leaves Images	**Leaf Symptoms**	**Size**
	Healthy	43
	Spot	884
	Curl	54
	Slug	2025

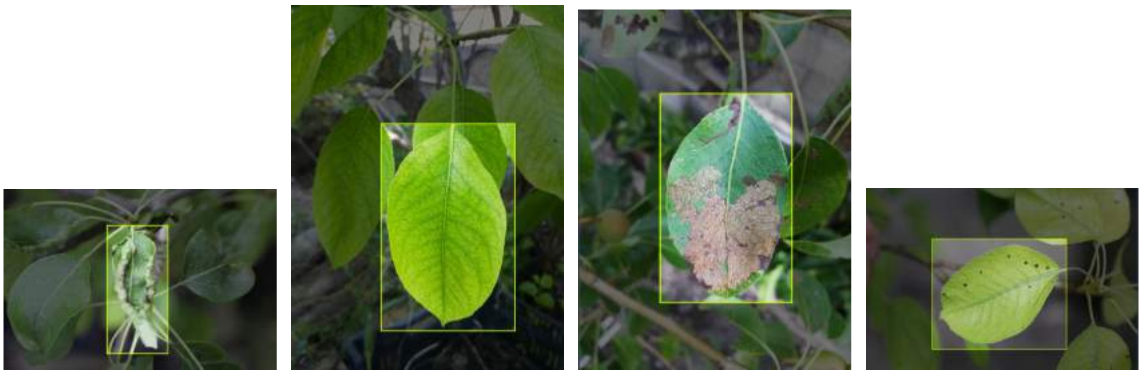
Table 3.2: Distribution of leaf images in the **DiaMOS Plant** Dataset



(a) (b) (c) (d)

Figure 3.1: Examples of pear leaves affected by different symptoms. (a)curl, (b)healthy, (c)slug, (d)spot

3.2 Dataset Preprocessing

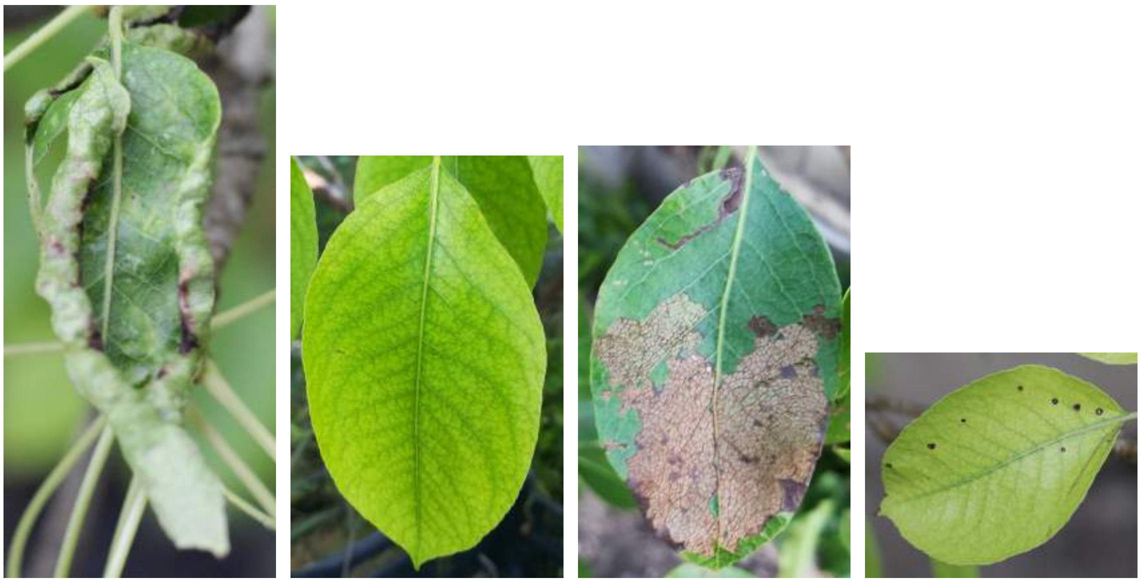


(a) (b) (c) (d)

Figure 3.2: Examples of pear leaves with the annotation (bounding box). (a)curl, (b)healthy, (c)slug, (d)spot

We have the dataset **DiaMOS Plant**[1] dataset, in which 4 classes are there named; curl, healthy, slug and spot containing 54, 43, 2025 and 884 leaf image respectively. All the images taken in real environment (not in the laboratory environment).

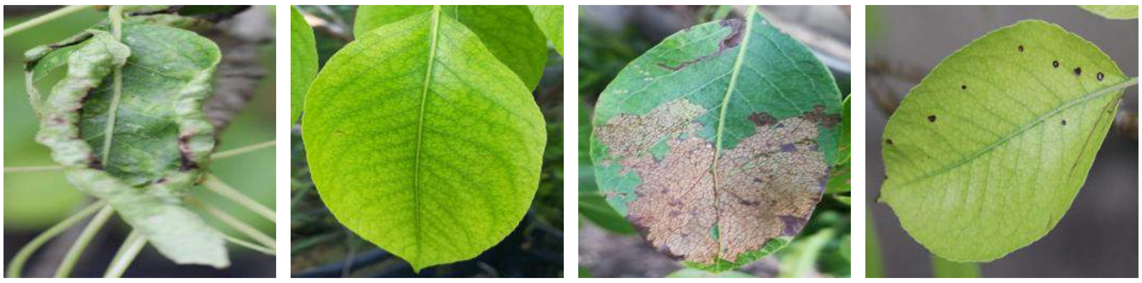
Also there was given an annotation file corresponding to each images. We first extracted the main leaf images according to the annotation file.



(a) (b) (c) (d)

Figure 3.3: Examples of extracted pear leaves according to annotation. (a)curl, (b)healthy, (c)slug, (d)spot

Since the images sizes are not fixed for all the images, it can be problematic in work, so We resize all the images in $256 \times 256 \times 3$.



(a) (b) (c) (d)

Figure 3.4: Examples of extracted pear leaves according to annotation. (a)curl, (b)healthy, (c)slug, (d)spot

Figure 3.5 shows the histogram of the total image count for each plant disease class in the **DiaMOS** dataset.

Here we see that, the DiaMOS dataset has a limited number of records and it includes imbalanced data classes, which negatively affects plant disease classification

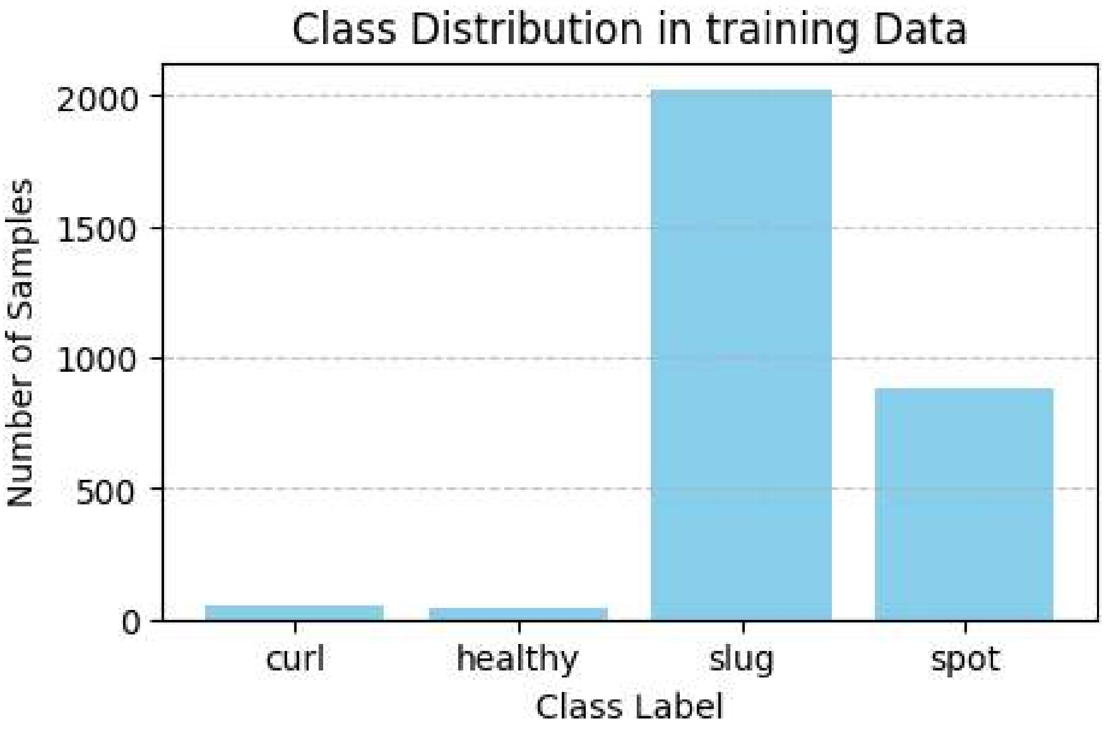


Figure 3.5: The distribution of images for each class

accuracy. Thus, our main focus is to expand the DiaMOS dataset and balance the number of records across all classes.

Here we see that 'curl' and 'healthy' classes data is very less. So we first augment these data (which include rotation, horizontal flip, vertical flip, noise addition, etc) and make it 5 times each to get variation in the dataset.

Then we built a DCGAN model and train the model separately on these 2 class dataset to generate the new images such that we can overcome the imbalanced problem.

Chapter 4

Data Generation

As we previously see that, the dataset is very much imbalanced, we apply **Deep Convolutional Generative Adversarial Networks (DCGAN)[2]** method to generate new synthetic images of the minority classes separately to balance the dataset.

4.1 DCGAN Model

A **Deep Convolutional Generative Adversarial Network (DCGAN)** is a type of neural network designed to generate realistic images. It consists mainly two models; a **generator** and a **discriminator**. Generator creates images from the random noise and the discriminator distinguished between real images and fake images.

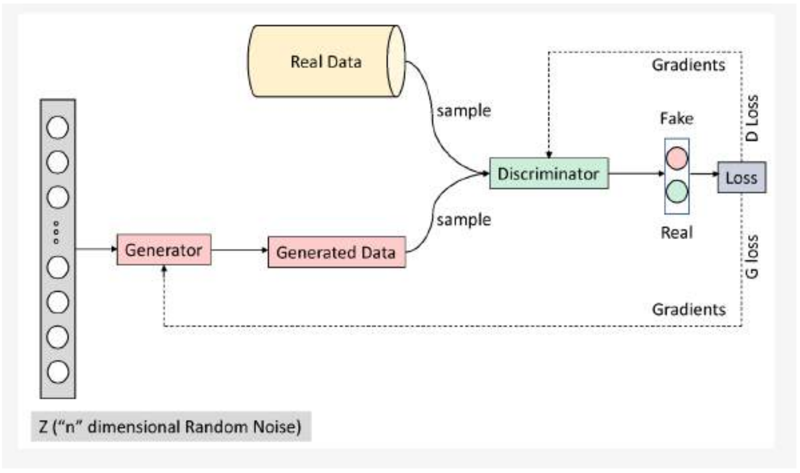


Figure 4.1: GAN Architecture

During training **Generator** trying to fool discriminator making the images as real as possible and a **discriminator** is trying to distinguished real and fake images properly. In this process generator produce highly realistic images over time. DC-

GANs are particularly useful for augmenting datasets, especially when dealing with imbalanced data, by generating additional synthetic images for minority classes.

4.1.1 Generator Model

The **Generator** in **DCGAN** takes a random noise vector and generates synthetic pear leaf images. Its architecture include:

- An input dense layer (taken input 128*1 vector from the latent space) followed by reshaping it into an initial shape of (64, 64, 256).
- It then applies three Conv2DTranspose layers to upsample the image, gradually increasing the spatial dimensions to (256, 256) while reducing the depth.
- Each Conv2DTranspose layer is followed by BatchNormalization and LeakyReLU activation to ensure stable and effective training.
- The final Conv2DTranspose layer uses a tanh activation to produce the output image with shape (256, 256, 3).

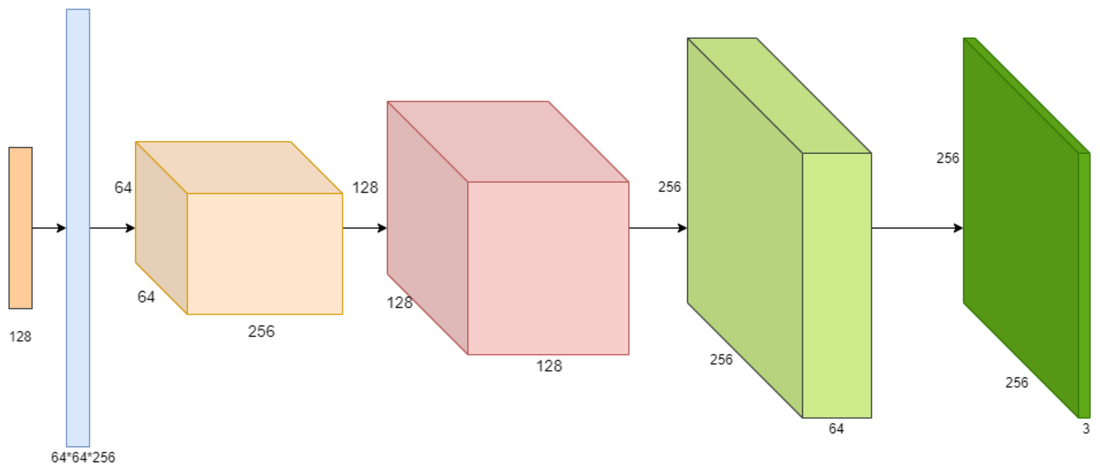


Figure 4.2: **Generator** Model where input taken as noise dimension 128 and output as an image of 256×256×3

4.1.2 Discriminator Model

The **Discriminator** in **DCGAN** evaluates the authenticity of pear leaf images. Its architecture include:

- The model consists of four convolutional layers, each followed by a LeakyReLU activation and dropout for regularization.
- BatchNormalization is applied after the second, third, and fourth convolutional layers to stabilize and accelerate the training.
- The final layer is a dense layer with a single unit, suitable for binary classification tasks.

This model is designed to effectively distinguish between real and generated images by progressively reducing the spatial dimensions and increasing the depth through convolutional layers, followed by a dense layer for the final decision.

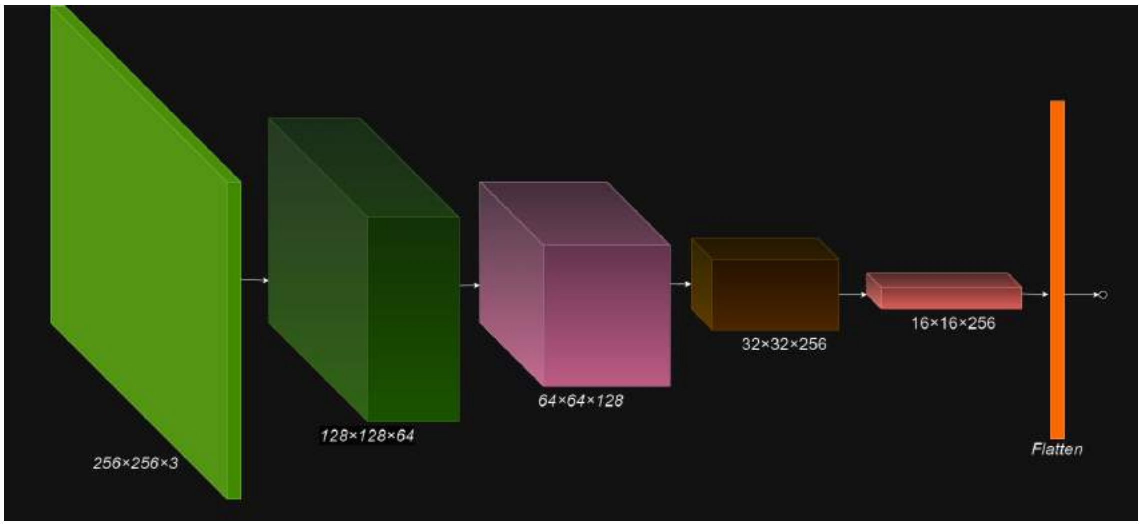


Figure 4.3: **Discriminator** Model to check the image is real or fake

4.1.3 Training Process

- The training of a **DCGAN** involves alternating between optimizing the **Generator** and the **Discriminator** in a minimax game. The Generator aims to fool the Discriminator, while the Discriminator strives to correctly classify real and fake images.
- During each iteration, the Discriminator is first updated to maximize its accuracy in distinguishing real from fake images.
- Next, the Generator is updated to minimize the Discriminator's ability to distinguish its outputs from real images.
- This adversarial process continues until the Generator produces images that are indistinguishable from real images.

4.2 Generated Data

Since there are two classes named '**curl**' and '**healthy**' has very very less images, so we train our **DCGAN** model on these two classes separately for **5000** epochs each and saved the models. Before training we resize all the images in $\mathbf{256 \times 256 \times 3}$.

After training using the saved models, we generate **1500** new synthetic images of size $\mathbf{256 \times 256 \times 3}$ for each of these two classes to made the dataset balanced.

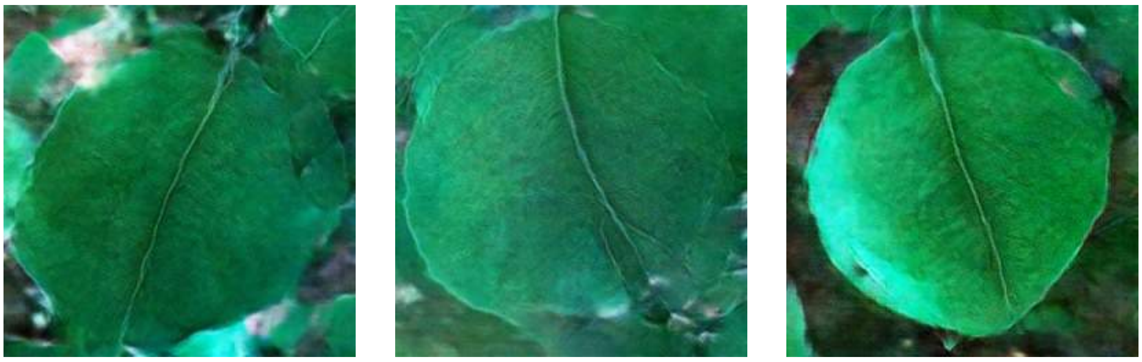


Figure 4.4: Examples of generated **'healthy'** pear leaves using our **DCGAN** model

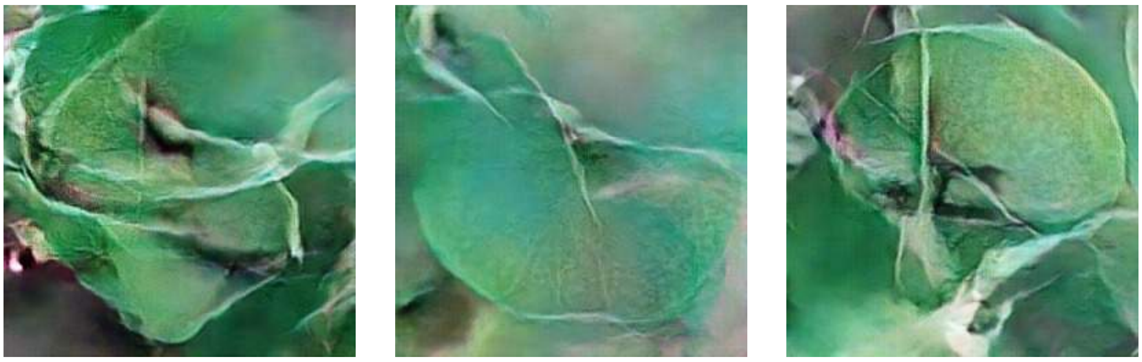


Figure 4.5: Examples of generated **'curl'** pear leaves using our **DCGAN** model

After data augmentation and data generation there are total 6394 leaf images of 4 different classes: healthy leaf, slug leaf, curl leaf, and spot leaf and the number of images are 1715, 2025, 1770 and 884 respectively.

Table: Distribution of Classes in Leaf Images after annotation and data generation using DCGAN[2]

Leaves Images	**Leaf Symptoms**	**Size**
	Healthy	1715
	Spot	884
	Curl	1770
	Slug	2025

Table 4.1: Distribution of leaf images after augmentation and data generation using DCGAN

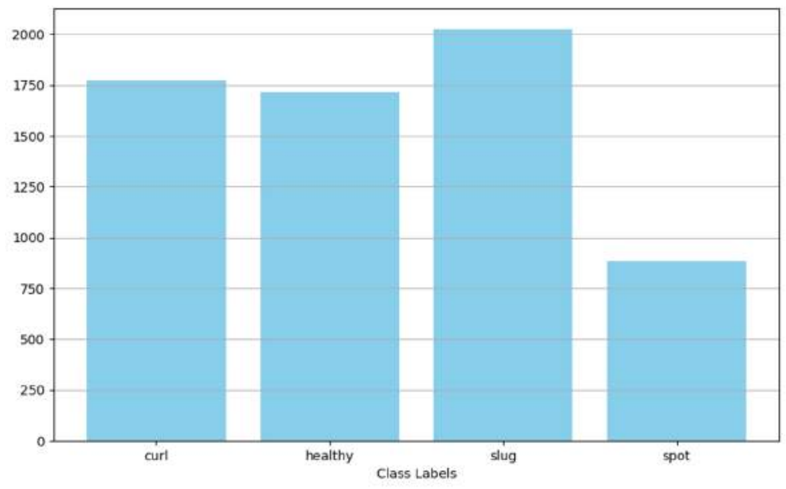
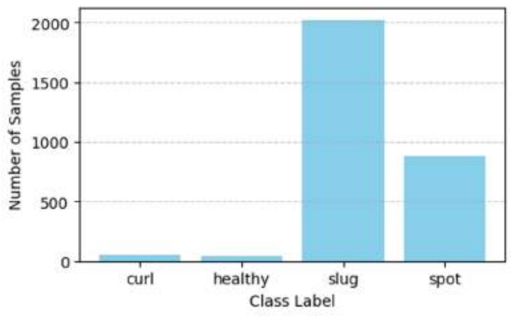


Figure 4.6: Distribution of new upgraded dataset

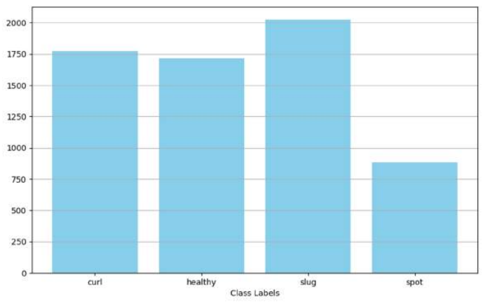
Chapter 5

Methodology

In our research, we worked on both the original pear leaf dataset and the upgraded dataset after augmentation and data generation using DCGAN to enhance the classification of pear leaf images.



(Original dataset)



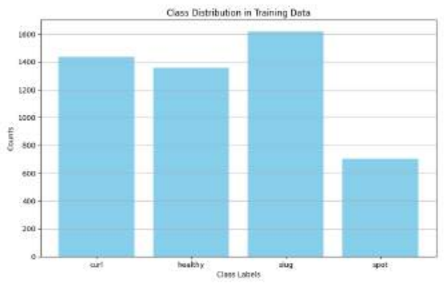
(Upgraded dataset)

Figure 5.1: Distribution of original and upgraded datasets

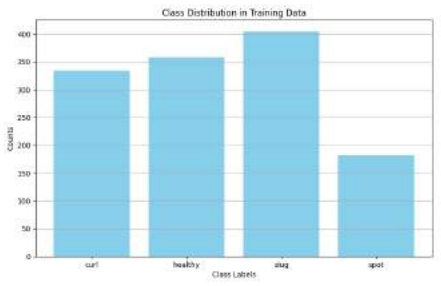
We utilized three pre-trained models (**VGG16**[3], **ResNet50**[4] and **Inception V3**) along with our proposed CNN models to perform the classification tasks. We compare the models performance in the both dataset mention above and observe the enhancement of the performance in original dataset to upgraded dataset.

5.1 Dataset Splitting

To conduct the study, we divided the dataset into training and test sets in a 8:2 ratio. Before training, we preprocessed the data to ensure compatibility with CNN networks. All images were resized to 224×224 pixels with three color channels and reshaped to fit the network's expected input dimensions. We also scaled the pixel values to the [0,1] range and converted the images into float32 arrays. Then for training we divide the training set into train and validation set in 8:2 ratio.



(Training data)



(Test data)

Figure 5.2: Distribution of training and test datasets

5.2 Some Pre-trained Models

Here in our research work we trained two pre trained models; VGG16 and ResNet50 to check the performance on the original and the upgraded datasets.

VGG16

[3] VGG16 is a convolutional neural network model proposed by the Visual Geometry Group (VGG) at the University of Oxford. Known for its simplicity and depth, VGG16 consists of 16 layers, including 13 convolutional layers followed by 3 fully connected layers. It uses small 3x3 convolution filters throughout the network, which allows it to capture fine features in images. VGG16 is widely used for image classification and has achieved excellent performance on benchmark datasets like ImageNet.

ResNet50

[4] ResNet50 is a deep residual network introduced by Microsoft Research. It consists of 50 layers, utilizing a unique architecture with residual blocks that allow the network to learn residual functions. This design helps to train much deeper networks compared to traditional architectures. ResNet50 has become a standard for image classification tasks due to its high accuracy and efficiency, and it has been instrumental in advancing the field of deep learning.

Inception V3

[5] Inception V3 is a type of deep learning model used for recognizing images. Developed by Google, it looks at images in different ways to understand them better. It uses smaller pieces to process images quickly and accurately. This model is good

at figuring out what's in a picture and is often used because it's both powerful and efficient.

5.3 Classification Models

In this study, we propose some Convolutional Neural Network (CNN) architecture to classify pear leaf diseases effectively. The model is designed to classify the leaf images into four classes: curl, healthy, slug, and spot.

Here we consider 4 different CNN models with different number of convolution layers. Train all the models and see the performance of the models on the test set and compared.

Model-1

We can summarized the model as:

- Here, we used a Sequential Convolutional Neural Network (CNN) model for classifying images.

- The model starts with a convolution layer that has 32 filters, followed by a layer that normalizes the data.

- A pooling layer that reduces the size of the data, and a dropout layer that helps prevent overfitting.

- This pattern is repeated with increasing complexity: first with 64 filters, then with 128 filters, and finally another layer with 64 filters. Each convolution layer is followed by normalization, pooling, and dropout.

- After these layers, the model is flattened, which means converting the data into a one-dimensional array. Then, the data passes through three dense (fully connected) layers with 64, 128, and 64 units, respectively.

- Each dense layer uses ReLU activation to introduce non-linearity and L2 regularization to prevent overfitting.

- The last layer of the model is a dense layer with 4 units and softmax activation, which is used for classifying the images into one of four categories.

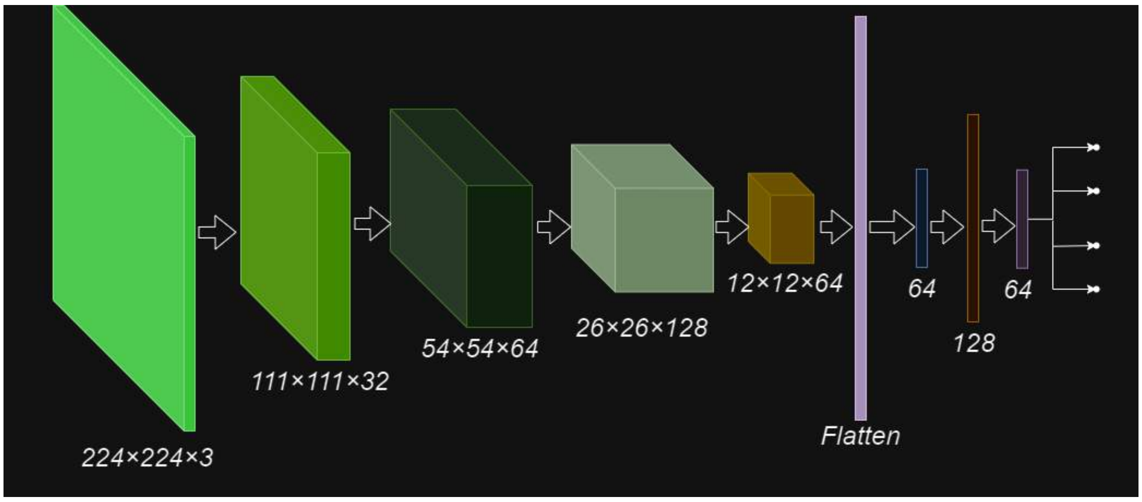


Figure 5.3: Model-1 Architecture where input taken as 224×224×3 size image and output saying the corresponding class labels

Model-2

In Model 2, we extend the architecture of Model-1 by adding an extra convolutional layer with 32 channels while keeping all other components the same. This additional layer allows for capturing more complex features in the image data.

The subsequent structure remains identical to Model-1, including flattening the data, passing through dense layers with ReLU activation and L2 regularization, and ending with a softmax output layer for classification. The training process, optimizer, learning rate, loss function, and evaluation metric are consistent with Model-1.

Model-3

In Model-3, we extend the architecture of Model-1 by adding 2 extra convolutional layer with 128 channels and 32 channels while keeping all other components the same. This additional layer allows for capturing more complex features in the image data.

Model-4

In Model-4, we extend the architecture of Model-1 by adding 3 extra convolutional layer with 128 channels, 32 channels and 16 channels while keeping all other components the same. This additional layer allows for capturing more complex features in the image data.

The models are compiled with the Adam[10] optimizer, using a learning rate of 0.001. The loss function is categorical cross-entropy[11], appropriate for multi-class

Table 5.1: Hyperparameters and Model Details

Hyperparameter	**Value/Description**
Input Size	224 x 224 x 3 (Height x Width x Channels)
Number of Classes	4 (curl, healthy, slug, spot)
Batch Size	16
Epochs	200
Learning Rate	0.0001
Optimizer	Adam
Loss Function	Categorical Cross-Entropy
Metrics	**Accuracy**, Precision, Recall, F1 score
Activation Functions	ReLU, Softmax

classification, and for evaluation here we use accuracy, precision, recall, F1 score and the ROC curve.

Chapter 6

Experimental Results and Observations

6.1 Environmental Setup

The models are developed using our institute's GPU and various free cloud-based Python development environments such as Kaggle Notebooks and Google Colaboratory. For this project, we utilized several libraries: TensorFlow, an open-source library for numerical computations and machine learning; Keras, which is used for building neural networks; NumPy for data analysis and mathematical operations; Matplotlib for graph visualization; etc.

6.2 Evaluation Metrics

In this study, we evaluate the performance of the models using several standard metrics: **accuracy, precision, recall, and F1 score, ROC-curve**. These metrics give a comprehensive understanding of the model's performance as a result we can properly understand the model is good or not.

We can't tell the models performance only seeing the accuracy of the model because in the case of imbalance data , a dumb model also can give the good accuracy. That's why we checking the precision, recall, F1 score and the ROC-curve

6.2.1 Accuracy

Accuracy is the ratio of the correctly predicted outcome to the total outcomes. It is a simple and commonly used metric to measure the overall performance of the model.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.1}$$

where:

- TP (True Positive): The number of correctly predicted positive instances.
- TN (True Negative): The number of correctly predicted negative instances.
- FP (False Positive): The number of incorrectly predicted positive instances.
- FN (False Negative): The number of incorrectly predicted negative instances.

6.2.2 Precision

Precision, also known as positive predictive value, is the ratio of correctly predicted positive instances to the total predicted positive instances. It provides insight into the accuracy of the positive predictions made by the model.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6.2}$$

6.2.3 Recall

Recall, or sensitivity, is the ratio of correctly predicted positive instances to the total actual positive instances. It indicates how well the model can identify positive instances.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6.3}$$

6.2.4 F1 Score

The F1 score is the harmonic mean of precision and recall. It provides a single metric that balances both precision and recall, especially useful in cases of imbalanced datasets.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.4}$$

6.2.5 ROC Curve

The Receiver Operating Characteristic (ROC) curve is a graphical representation that illustrates the ability of a binary classifier system. The curve is plotted with the True Positive Rate (TPR) or (recall) on the y-axis and the False Positive Rate (FPR) on the x-axis.

The Area Under the ROC Curve (AUC-ROC) is a scalar value that indicates the performance of the model. A model with an AUC of 0.5 indicates that the model has no discriminative ability, and an AUC of 1.0 indicates perfect classification.

$$\text{TPR or (Recall)} = \frac{TP}{TP + FN} \tag{6.5}$$

$$\text{FPR} = \frac{FP}{FP + TN} \tag{6.6}$$

The ROC curve provides an aggregate measure of performance across all possible classification thresholds, making it a useful tool for evaluating the trade-offs between sensitivity and specificity.

6.2.6 Confusion Matrix

The confusion matrix is a table that is used to describe the performance of a classification model on a set of test data for which the true values are known. It allows visualization of the performance of an algorithm.

The matrix itself is straightforward but provides a powerful insight into the types of errors being made by the classifier:

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Table 6.1: Confusion Matrix

The confusion matrix provides insight into not only the accuracy of the model but also the types of errors it is making, thus helping to fine-tune the model further.

These metrics together offer a robust evaluation of the model's performance, addressing both the accuracy of the predictions and the model's capability to correctly classify both positive and negative instances.

6.3 Results and Discussion

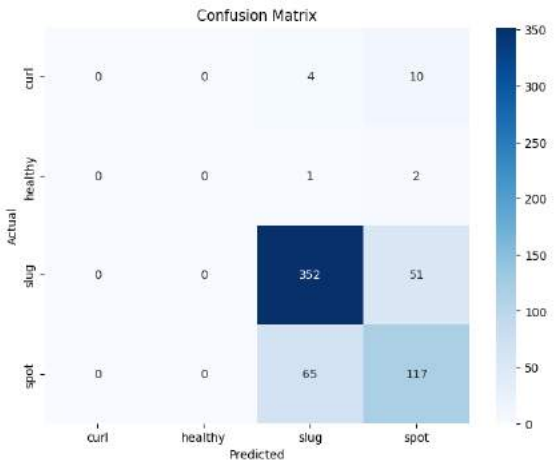
In this section, we compare results from two different approaches:

- first, we described about the experiments that we did on the **original pear leaf dataset DiaMOS Plant**[1] dataset with the pre trained models VGG16 and Resnet50 models, also with proposed models.

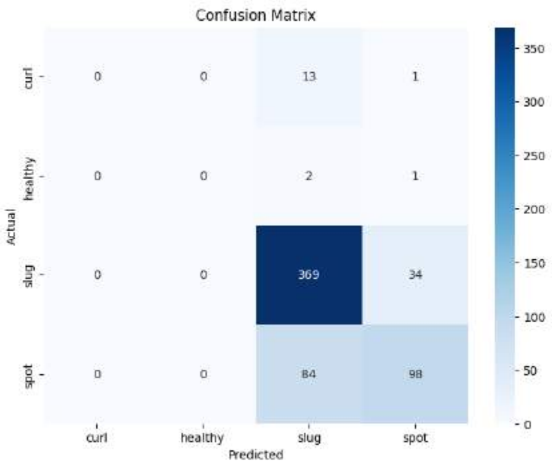
- and second, apply these models to the **expanded DiaMOS** dataset that includes additional images generated by **DCGAN**.

For both the approaches, we analyse the following parameters: Accuracy, Precision, Recall, F1 score, etc.

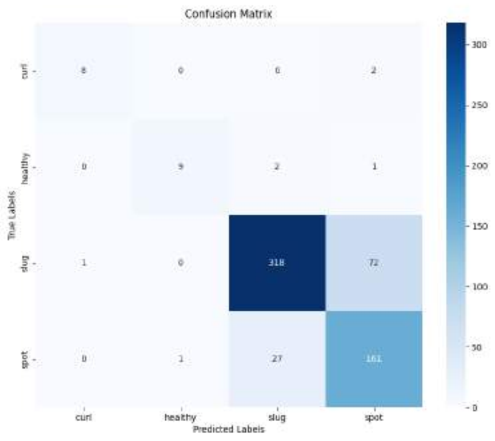
Firstly , we train the models over the **original DiaMOS** dataset. Since the data was imbalanced therefore obviously the model performance will be not so good. As we see from the confusion matrices **(Figure 6.1)** that in the pre trained model the minority class data is fully null, it may be for overfitting because the pre trained models are heavy and the number of data is very less.



VGG16



ResNet50

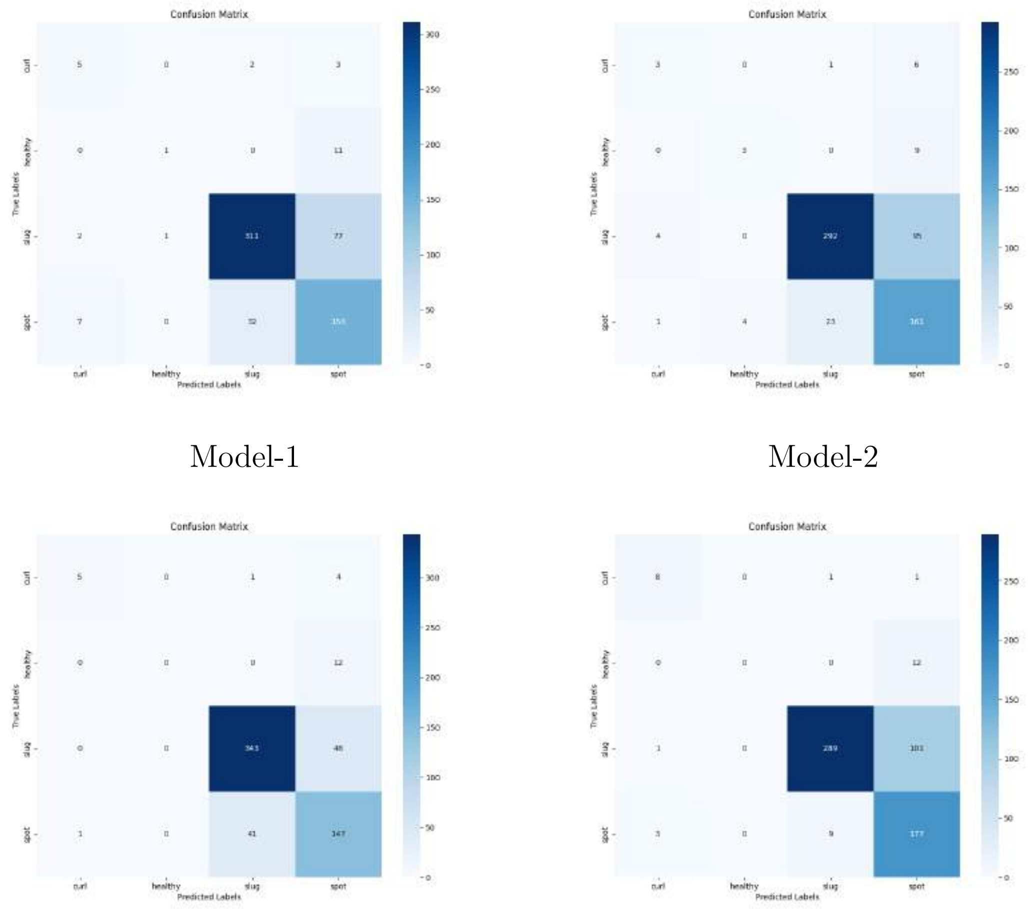


Inception V3

Figure 6.1: Confusion matrices for the original **DiaMOS** dataset of **pre-trained models**

From the **Table 6.2** we observe that for the pre trained models; **VGG16** and **ResNet50** give the accuracy around **77%** where as the **Inception V3** gives **82%**. And for proposed models; **Model-3** give the best performance among them which is approximate same as the **Inception V3** model. Here we also see the precision , recall and F1 score .

Then Secondly, we train the same models on the **extended DiaMOS dataset** which was made by adding some synthetic minority class leaf images using **DCGAN**



Model-1 Model-2

Model-3 Model-4

Figure 6.2: Confusion matrices for the original **DiaMOS** dataset of proposed models

architecture.Here the dataset is not so skewed , so it is natural that the model performance will be increased.

From the confusion matrices **(Figure 6.3 & 6.5)** and from the **Table 6.3** we observe that the models classify the dataset in a good manner. And the performance of proposed models as well as the pre trained models increase.

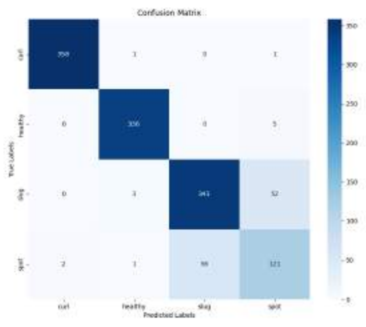
Moreover, after expanding the dataset using the **DCGAN** architecture, we see that the classification accuracy enhanced approx 10% for proposed models as well as pre trained models. And also increase the other performance measure as well, like, precision, recall, F1 score.

Here comparing all our models, we observe that **Model-3** give the best performance among them. It also give the better result as compare to **VGG16** and **ResNet50** models and give approx same performance as **Inception V3**.

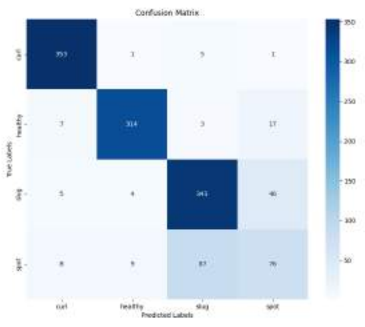
In addition, we compared proposed **model-3** with the other existing pre trained models with respect to the **complexity** of the models. And we observed that the **model-3** give approx same performance with existing pre trained models (VGG16,

Table 6.2: The results obtained from different models using the original DiaMOS dataset

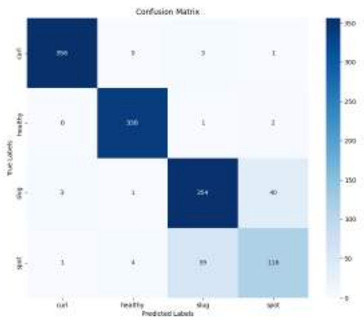
Architecture	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
VGG19	77.91	75	78	77
ResNet50	77.57	75	78	75
Inception V3	**82.39**	**84**	**82**	**83**
Model-1	77.57	80	78	78
Model-2	76.25	80	76	77
Model-3	**82.23**	**81**	**82**	**82**
Model-4	78.74	83	79	79



VGG16



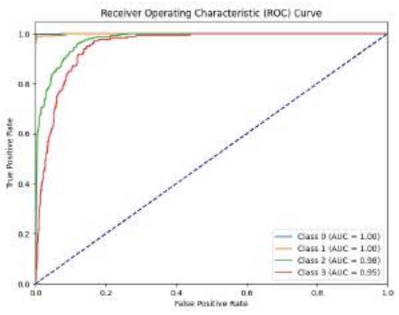
ResNet50



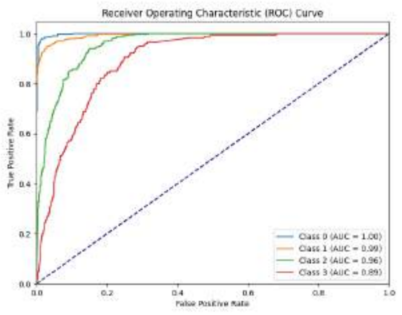
Inception V3

Figure 6.3: Confusion matrix results for the extended **DiaMOS** dataset

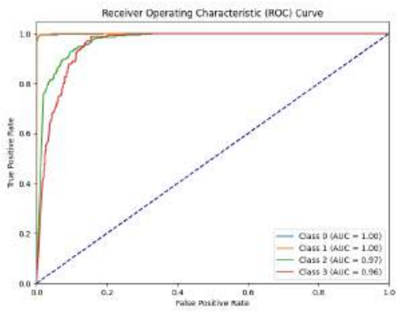
ResNet50, Inception V3) with a very less memory and less time. Also the number of parameters in **Model-3** is very less as compare to the existing pre trained models, so it can be fit in a small device also.



VGG16



ResNet50



Inception V3

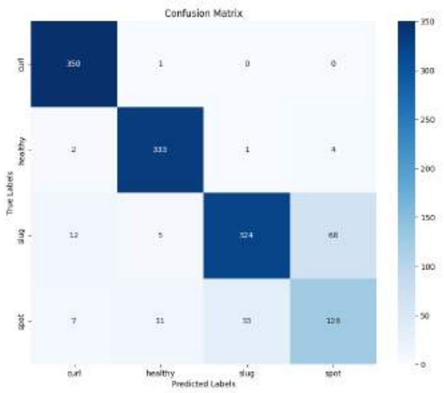
Figure 6.4: ROC curve for the extended **DiaMOS** dataset

Table 6.3: The results obtained from 3 different models on the extended dataset using DCGAN

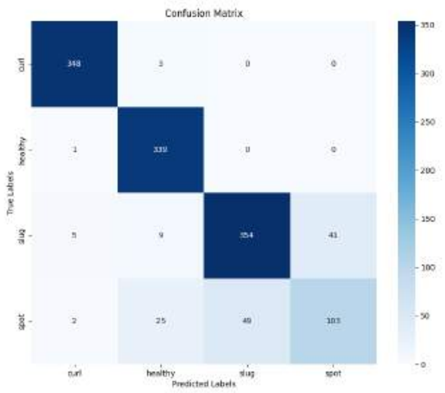
Architecture	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
VGG19	90.54	91	91	91
ResNet50	84.91	84	85	84
Inception V3	**91.01**	**91**	**91**	**91**
Model-1	88.74	89	89	89
Model-2	89.44	89	89	89
Model-3	**91.95**	**92**	**92**	**92**
Model-4	88.51	91	89	89

Table 6.4: Complexity comparison of proposed model and existing models.

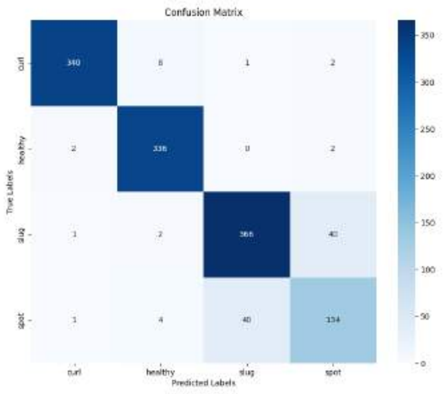
Architecture	**Accuracy (%)**	**Size (MB)**	**Parameter(M)**	**Time(s)**
VGG16	90.54	56.13	14.71	11
ResNet50	84.91	89.98	23.59	16
Inception V3	91.01	83.17	21.80	15
Model-1	88.74	2.96	0.77	7
Model-2	89.44	0.99	0.25	5
Model-3	**91.95**	**1.35**	**0.35**	**6**
Model-4	88.51	3.04	0.79	7



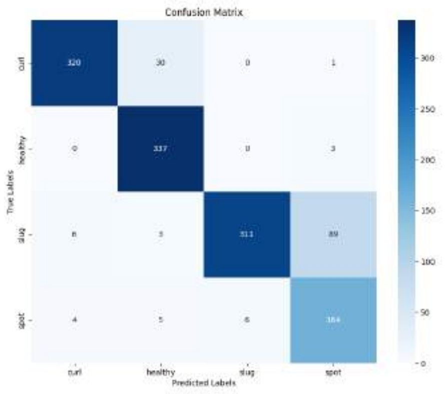
Model-1



Model-2

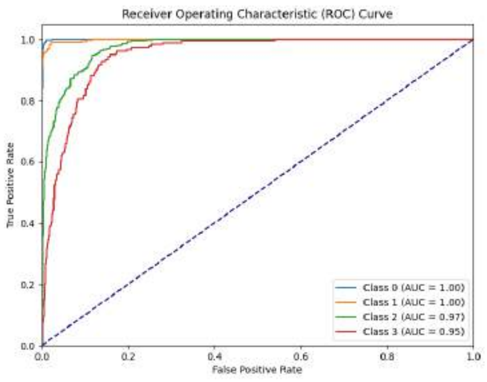


Model-3

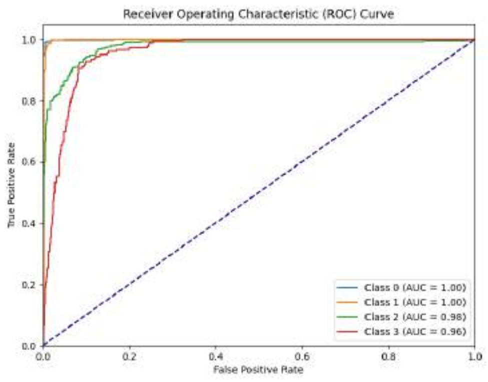


Model-4

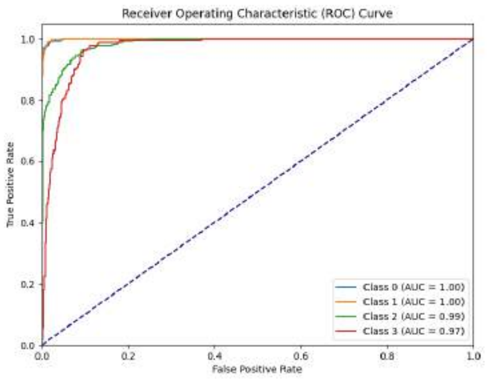
Figure 6.5: Confusion matrix for proposed models on the extended **DiaMOS** dataset



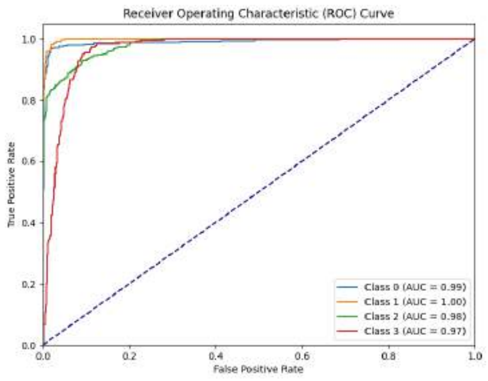
Model-1



Model-2



Model-3



Model-4

Figure 6.6: ROC curve for proposed models on the extended **DiaMOS** dataset

Chapter 7

Conclusions and Future Work

In this research, first we built a **DCGAN** model and using it we generate some synthetic image data. In the dataset there were 2 classes '**curl**' and '**healthy**' which has very less data which made all the dataset imbalanced. therefore we utilize the **DCGAN** approach on these two classes to generate $256 \times 256 \times 3$ pixels leaf images using a minimal data source to make the dataset more or less balanced.

After a series of experiments, we see that the new dataset using the **DCGAN** architecture has improved the accuracy of pear disease classification. Moreover, recall, precision, and F1-score are also increase as well on the new dataset.

Also in this work, we develop some **CNN** models to identify and classify 4 class classification. And compared with some pre-trained models (**VGG16, ResNert50, Inception V3**).

The resulting metrics describes that our models are giving approximate same results as compare to these pre-trained models but the running time is very less. Also the number of parameters in our models are very less as compare to them so it can be fit in a small device also.

In future research, we plan to explore more advanced deep learning methods to further enhance classification accuracy, recall, precesion, F1 score, etc. Also plan to explore some technique that gives better results in less time and also in less memory.

Bibliography

[1] G. Fenu and F. M. Malloci, “Diamos plant: A dataset for diagnosis and monitoring plant disease,” *Agronomy*, vol. 11, no. 11, 2021.

[2] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” 2016.

[3] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015.

[5] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” 2015.

[6] D. Joseph, P. Pawar, and R. Pramanik, “Intelligent plant disease diagnosis using convolutional neural network: a review,” *Multimedia Tools and Applications*, vol. 82, pp. 1–67, 10 2022.

[7] G. Fenu and F. M. Malloci, “Evaluating impacts between laboratory and field-collected datasets for plant disease classification,” *Agronomy*, vol. 12, no. 10, 2022.

[8] G. Fenu and F. M. Malloci, “Classification of pear leaf diseases based on ensemble convolutional neural networks,” *AgriEngineering*, vol. 5, no. 1, pp. 141–152, 2023.

[9] A. Osipov, V. Shumaev, A. Ekielski, T. Gataullin, S. Suvorov, S. Mishurov, and S. Gataullin, “Identification and classification of mechanical damage during continuous harvesting of root crops using computer vision methods,” *IEEE Access*, vol. 10, pp. 1–1, 01 2022.

[10] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017.

[11] Z. Zhang and M. R. Sabuncu, “Generalized cross entropy loss for training deep neural networks with noisy labels,” 2018.