Sugarcane Disease Prediction Using Deep Learning

Submitted in partial fulfillment of the requirements of the degree

BACHELOR OF ENGINEERING IN **COMPUTER ENGINEERING**

By

Akshara Sarode	**120A1004**
Yash Patil	**120A1079**
Sufiyan Posharkar	**120A1083**
Sanskar Unkule	**120A1118**

Name of the Mentor

Dr. Aparna Bannore

Department of Computer Engineering

SIES GRADUATE SCHOOL OF TECHNOLOGY

NERUL, NAVI MUMBAI – 400706

ACADEMIC YEAR

2022 – 2023

CERTIFICATE

This is to certify that the Mini Project entitled **“Sugarcane Disease Prediction Using Deep Learning”** is a bonafide work of submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **“Bachelor of Engineering”** in **“Computer Engineering”.**

Akshara Sarode	**120A1004**
Yash Patil	**120A1079**
Sufyain Posharkar	**120A1083**
Sanskar Unkule	**120A1118**

(Dr. Aparna Bannore)
Mentor

(Dr. Aparna Bannore)
Head of Department

(Dr. Atul Kemkar)
Principal

Mini Project Approval

This Mini Project entitled "**Sugarcane Disease Prediction Using Deep Learning"** by is approved for the degree of **Bachelor of Engineering** in **Computer Engineering.**

Akshara Sarode	**120A1004**
Yash Patil	**120A1079**
Sufyain Posharkar	**120A1083**
Sanskar Unkule	**120A1118**

Examiners

1..
(Internal Examiner Name & Sign)

2..
(External Examiner name & Sign)

Date: Place:

Contents

Sr. No.	Topic	Page No.
1.	**Chapter 1: Introduction** – Introduction – Motivation – Problem Statement & Objectives – Organization of the Report	 1 2 2 3
2.	**Chapter 2: Literature Survey** – Survey of Existing System – Limitation Existing system or Research gap	 4 6
3.	**Chapter 3: Proposed System** – Introduction – Architecture/ Framework – Algorithm and Process Design – Details of Hardware & Software – Experiment and Results for Validation and Verification	 7 8 9 13 13
4.	**Chapter 4: Conclusion** – Conclusion – Future work	 21 22
5.	**References**	23

ABSTRACT

Sugarcane is an important cash crop worldwide, and disease outbreaks can cause significant economic losses. Early detection and timely management of sugarcane diseases are crucial for maintaining crop yield and quality. In this project, we propose a deep learning-based approach for sugarcane disease prediction using transfer learning and Adam optimizer. We utilized pre-trained models, including AlexNet, VGG16, VGG19, MobileNet, DenseNet, ResNet50, and EfficientNet, to train our model on sugarcane disease images, including Red Rust, Red Rot, Bacterial Blight, and Healthy Crops. The proposed model achieved high accuracy and performance, with ResNet101 outperforming other models. The project demonstrates the effectiveness of transfer learning in improving the classification of sugarcane diseases. The developed system can assist farmers in early detection and timely treatment of sugarcane diseases, thus increasing crop yield and reducing losses.

Keywords: Deep Learning, CNN, Transfer Learning, Alexnet, VGG 16, Adam’s Optimizer, epoch.

Acknowledgement

We would like to express our thanks to the people who have helped us the most throughout our project. We are grateful to our guide **Dr. Aparna Bannore** and coordinator **Prof. Sunil Punjabi** for nonstop support for the project.

A special thanks goes to each other who worked together as a team in completing the project, where we all exchanged our own interesting ideas, thoughts and made it possible to complete our project with all accurate information. We also wish to thank our parents for their personal support and attention who inspired me to go my own way.

We would also like to extend our sincere gratitude to our Principal **Dr. Atul Kemkar** and our Head of the Department **Dr. Aparna Bannore** for their continuous support and encouragement.

We also would like to thank our other faculty members for providing us with all the required resources and references for the project.

List of Abbreviations

SR NO.	ABBREVATIONS	FULL FORM
1.	CNN	Convolutional Neural Network
2.	DL	Deep Learning
3.	VGG	Visual Geometry Group
4.	DETR	Detection Transformer
5.	SVM	Support Vector Machine
6.	KNN	K-Nearest Neighbor
7.	YOLO	You Only Look Once
8.	R-CNN	Region Convolutional Neural Network

List of Figures

SR NO.	LIST OF FIGURES	PAGE NO.
1.	Architecture of the system	8
2.	Data Augmentation	9
3.	Transfer Learning	10
4.	Alexnet	10
5.	Vgg 16	11
6.	ResNet 101	11
7.	MobileNet	12
8.	EfficientNet	12
9.	Alexnet Parameters	13
10.	Alexnet epoch	14
11.	Densenet	14
12.	DenseNet Accuracy	14
13.	Inceptionv3	15
14.	MobileNet	15
15.	ResNet50 Model	15
16.	ResNet101 Leaf	16
17.	ResNet101 Diseases	16
18.	ResNet101 Model Accuracy	16
19.	ResNet101 Model Loss	17
20.	VGG 16	17
21.	Vgg19 Model Accuracy	18
22.	EfficientNet Model Accuracy	18
23.	Rust Disease Prediction	19
24.	Rot Disease Prediction	19
25.	Model Accuracies Comparison	20
26.	Comparison of performance for different DL models	20

Chapter 1: Introduction

1.1 Introduction

Sugarcane is one of the world's most important crops, which accounts for a significant portion of the world's sugar supply. However, sugarcane cultivation is vulnerable to several kinds of diseases, which can have a substantial influence on crop yield and quality. Traditional disease control measures, which can have a substantial influence on agricultural output and quality. Traditional disease detection and diagnosis techniques are frequently time-consuming and labor-intensive, necessitating professional knowledge and training. As a result, there is a rising interest in creating automated methods for detecting sugarcane disease utilizing modern technologies such as deep learning.

However, detecting sugarcane diseases in person would be difficult due to the variety of diseases observed in sugarcane, like Red Rot, Red Rust, and Bacterial Blight, each of which have been researched in this research project. While there would probably be numerous indications of other forms of diseases, it would be difficult for a man to detect such diseases in a vast area of crops, and by the time such diseases were identified, it might be too late for the farmer to dispose of impacted crops. So, technology would be beneficial for them to identify diseases in time by first analyzing how healthy crops look and how crops look when infected with diseases such as red rot, red rust, and Bacterial Blight, and then using transfer learning and various CNN algorithms to predict whether the crop has been infected by the following diseases using image datasets captured from the fields.

Deep learning and transfer learning can help improve the accuracy and efficiency of sugarcane disease detection in this situation. Deep learning algorithms can be trained to recognize patterns and diagnose disease symptoms with high accuracy by employing enormous data sets of annotated photos. In turn, transfer learning allows for the reuse of previously trained models, decreasing the requirement for vast amounts of training data and speeding up the construction of illness prediction models.

1.2 Motivation

Sugarcane is an important crop that is cultivated across India. Sugarcane cultivation has significance in India since it is the second largest agro-based business after cotton. The crop occupies around 50 lakh hectares, while the industry's yearly output may reach Rs. 80000 crores. However, sugarcane crops are vulnerable to a variety of diseases, which may have a major impact on harvest quality and yield. The traditional approach of disease prediction may be more time-consuming and difficult for farmers and other visual experts, which may also prove to be expensive and frequently prone to human error and may also lead to financial losses for the farmers. In addition, spotting subtle disease symptoms in the early stages might be difficult, despite the fact that early detection of diseases is essential for halting disease transmission. The motivation for this effort originated from the potential significance of deep learning and transfer learning in this context.

1.3 Problem Statement and Objectives

Sugarcane diseases can have a major negative influence on crop yield and quality, resulting in financial losses for farmers and the sugar industry. Traditional disease prediction techniques rely on manual inspection by specialists, which can be time-consuming and subject to human error. Therefore, a reliable, automated approach is required for the early identification and diagnosis of illnesses affecting sugarcane.

Objectives of the proposed system are:

- Evaluate the performance of a pre-trained deep learning model for transfer learning on sugarcane disease image datasets.
- Create and improve a deep learning model for predicting sugarcane disease using transfer learning and the Adam's optimizer.
- Evaluate the accuracy of the proposed model with existing machine learning-based methods for predicting sugarcane disease.
- Demonstrate whether the suggested approach is effective in detecting and diagnosing sugarcane diseases early, with the potential to increase sugarcane farming's efficacy and output.

1.4 Organization of the report

This report is broken down into chapters. The report starts with Chapter 1, which gives the brief of the sugarcane disease prediction using deep learning, also the motivation of the project and problem statement and objectives of choosing this topic as our project.

As it moves on to Chapter 2, it analyzes the explanation of different techniques and models that are used in similar projects. Also it discusses the strengths and limitations of the existing models of the project.

In chapter 3 of this report. Proposed system, it gives a detailed description of the dataset used in the project. The proposed system also gives an overview of deep learning techniques which are used in this project. Detailed explanation of the models used, including VGG16, VGG19, MobileNet, ResNet50, DenseNet, InceptionV3, and EfficientNet. Finally, the presentation of the experimental results are obtained, also the performance of the models is also evaluated.

Everything concludes at Chapter 4 with all the summary of the project and its outcomes. Also there is a discussion of the limitations and future scope of the project. And the report ends with conclusion and final remarks.

Chapter 2: Literature Survey

2.1 Survey of existing system

Various machine learning and deep learning models and approaches have been used in earlier attempts to build a sugarcane disease prediction model, and each attempt served a particular goal.

Militante et al. [1] developed a method in which the author classified photos of sugarcane leaves into four groups using CNN, which is separated into seven different categories: healthy, red rot, leaf scald, and false smut. In order to train and evaluate CNN, they collected a dataset of 4,800 photos of sugarcane leaves. During training with 60 iterations, the model correctly classified photos of sugarcane leaves with an accuracy of 96.6% and a validation accuracy of 95%.

Militante et al., 2019, [2] proposed a technique in which they have used an adaptive aspect of deep learning which uses CNN's predictions to update the dataset used for training to improve its accuracy over time as it is exposed to more images. The model is able to detect multiple diseases at same time. The highest validation accuracy of 95.40% obtained over 40 epochs using VGG Net model, Le Net model achieves the rate of 93.65% and Strided Net model achieves 90.10%.

RAJESH and Gowri [3] The following model employs SVM (Support Vector Machine) for rainfall prediction and also makes use of Naive Bayes in order to forecast sugarcane production. SVM, however, provides the greatest outcome with greater accuracy for disease identification. Naive Bayes predicts crop disease with an accuracy of 61.21%, KNN predicts it with an accuracy of 83.64%, SVM predicts it with an accuracy of 83.64%, and Random Forest predicts it with an accuracy of 82.62%. It is a web-based model that makes predictions based on historical data.

Malik and Shadab [4] The suggested model has used a dataset of five sugarcane plant diseases from various Karnataka locations. On photos gathered from internet sources, models that trained to their highest accuracy of 93.40% and 76.40% were successful. Faster R-CNN and YOLO are two separate object detection methods that are employed. On the test set, their average precision score was 58.13% for the two of them.

Amarasingam and Narmilan [5] In the suggested model, deep learning algorithms are utilized to detect white leaf disease in crops. The author used an unmanned aerial vehicle to acquire RGB imagery of sugarcane. Detection Transformer, YOLOv5, YOLOR, DETR, and Faster R-CNN are all used in the suggested model. Yolov5 beat other models, according to the data, reaching precision of 0.95 and accuracy of 95%, 92%, 93%, and 79%. The detection performance of DETR is the worst for metric values of 77%, 69%, 77%, and 41% accuracy.

2.2 Limitation of Existing and Research Gap

Militante et al. [1] The 96x96 resolution would present some issues while analyzing and training the system in this case, which would eventually lead to some inaccuracy. The model is not being used for huge farms.

Militante et al., 2019, [2] It would take more time to predict and train the model because the model just uses a few CNN algorithms and does not use many other CNN algorithms or optimizers.

RAJESH and Gowri [3] SVM and Naive Bayes are frequently used for disease prediction, but this approach is useless since the model must first be able to identify the disease; prediction should then be the following step. SVM and Naive Bayes only train for small to medium large datasets. Moreover, it is less accurate at predicting and detecting disease.

Malik and Shadab [4] It would not be appropriate to trust some of the photographs because they were downloaded from the internet, and since object detection algorithms are utilized, it is possible that the model will occasionally identify the wrong thing. For the test set, the precision score is too low.

Amarasingam and Narmilan [5] Since the photographs were captured by an unmanned aerial vehicle (UAV), it's likely that the model might mistake some other crop leaves for sugarcane leaves. Additionally, because YOLOv5 is employed, the model only addresses one specific condition, which implies that training would require a significant amount of time.

Chapter 3: Proposed System

3.1 Introduction

Introducing a model which can predict and detect disease in sugarcane crops will help farmers improve crop yield and quality more quickly, safeguarding them from financial and other losses in the long run. It will additionally help the sugar industry through assisting it in increasing the quantity and quality of its sugarcane products. This system uses transfer learning and the Adam optimizer to predict sugarcane disease using deep learning. In order to effectively identify the type of disease, our system will make use of the strength of deep learning algorithms to automatically learn the pertinent features and patterns from photos of infected sugarcane leaves.

The suggested method makes use of picture datasets related to sugarcane disease and a pre-trained deep learning model for transfer learning. To enhance the model's performance for predicting sugarcane disease, the Adam optimizer is used for tuning. The system's performance is compared to that of other machine learning-based methods using a sizable and varied dataset of photos depicting sugarcane disease. The suggested method offers precise and timely predictions for disease management, which has the potential to revolutionize sugarcane disease detection and prevention.

Following concepts are used in this proposed system:

- **Transfer Learning [6]:** a method for reducing the quantity of training data and time by using a pre-trained deep learning model as the basis for training a new model on a related task.
- **Deep Learning [8]:** A subset of machine learning use multi-layered artificial neural networks to automatically identify the pertinent features and patterns in input data.
- **Convolutional Neural Network (CNN) [9]:** A specific class of neural network created especially for problems involving picture classification. For automatic feature extraction from the input images, CNN's use convolutional and pooling layers.
- **Adam optimizer [12]:** a popular optimization approach for deep learning model training. It enhances the effectiveness and speed of the training process by adjusting the learning rate of the model based on the gradient of the loss function.

3.2 Architecture and Framework

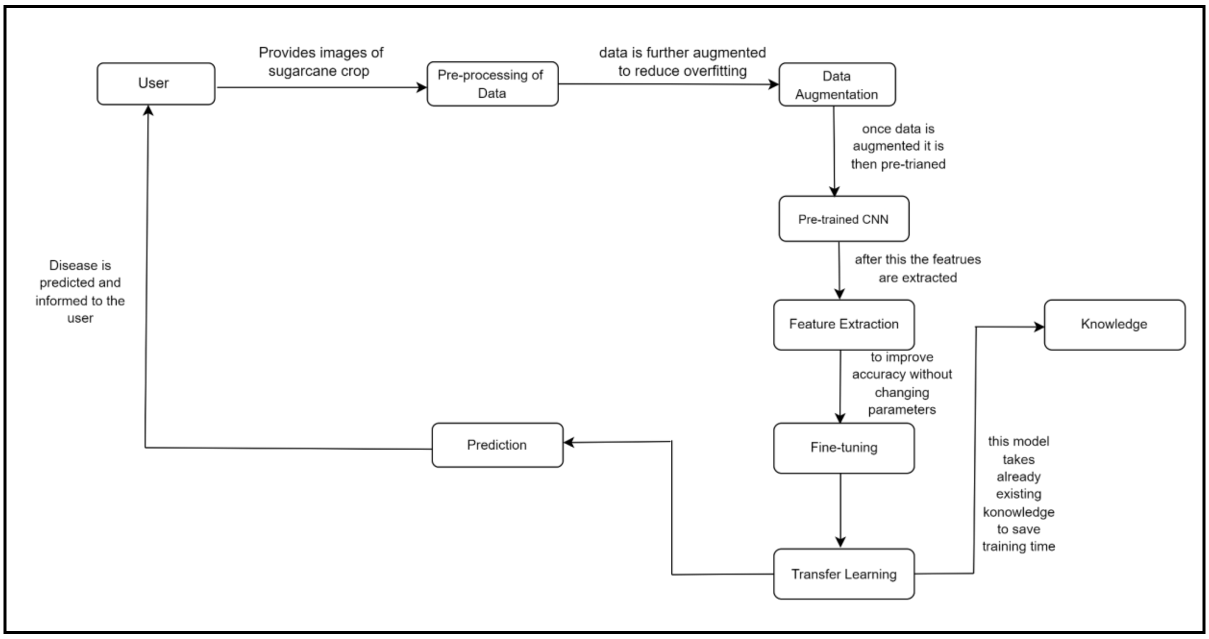

User
Provides images of sugarcane crop
Pre-processing of Data
data is further augmented to reduce overfitting
Data Augmentation
once data is augmented it is then pre-trianed
Pre-trained CNN
after this the featrues are extracted
Feature Extraction
to improve accuracy without changing parameters
Fine-tuning
Transfer Learning
this model takes already existing konowledge to save training time
Knowledge
Prediction
Disease is predicted and informed to the user

Fig 3.1 Block diagram of the system.

User: The system receives the user's input during the first phase. Red, green, and blue are the three-color channels present in the RGB format of the photographs.

Preprocessing: Preprocessing is done on the input images to get rid of any noise or artifacts that could hinder the performance of the deep learning models.

Data Augmentation: Data augmentation techniques are used to create new images from the original ones by applying various transformations, such as rotations, flips, and zooms, to increase the diversity of the training data.

Pre-trained Convolutional Neural Network: To extract characteristics from the input photos, a pre-trained CNN model is utilized. The CNN has been trained on a sizable image dataset, and the feature representations it has discovered can typically be used for a variety of computer vision tasks.

Feature Extraction: In order to create a more condensed representation of the input images, the features recovered from the CNN are subsequently passed through a number of fully connected layers.

Transfer Learning: The associated layers of the sugarcane disease prediction model are initialized using the pre-trained CNN model weights. Compared to training the model from scratch, this enables faster training and improved performance.

Knowledge: In order to reduce the training time of the dataset, the transfer learning model uses knowledge that is already stored in the database of prior datasets.

Prediction: Utilizing the learned model to make predictions on fresh input photos is what the architecture's final stage entails.

TensorFlow and Keras are just two examples of deep learning frameworks that can be used to build the proposed system. An illustration of a potential framework for the system is as follows:

TensorFlow: TensorFlow: TensorFlow is a Google-developed, open-source deep learning framework. For creating and training deep neural networks, it offers a complete set of tools and APIs, including pre-built models for transfer learning and optimization techniques like Adam optimizer. Additionally, TensorFlow lacks support for distributed training across multiple GPUs or CPUs, which would make training quicker and more effective.

The suggested method uses TensorFlow and uses the Adam optimizer for training and pre-built models like Inception or ResNet50 for transfer learning. Additionally, TensorFlow offers tools for data augmentation and image preprocessing, making it simpler to get the dataset ready for training. The developed model can then be deployed and tested on fresh photos.

3.3 Algorithm and Process Design

Data Augmentation [5]: Data Augmentation encompasses a suite of techniques that enhance the size and quality of training datasets such that better Deep Learning models can be built using them.

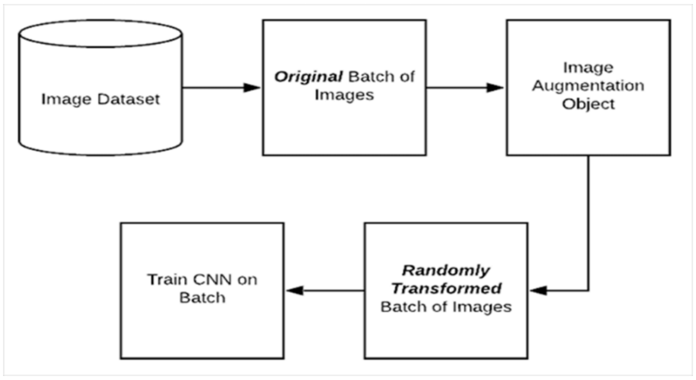


Image Dataset
Original Batch of Images
Image Augmentation Object
Randomly Transformed Batch of Images
Train CNN on Batch

Fig 3.3.1 Data Augmentation

Transfer Learning [6]: Transfer Learning is a concept which enables trained models to share their knowledge and help improve the outcomes. Transfer Learning, thus, is a framework of extending the capabilities of known systems to understand and share knowledge among tasks. The figure highlights the siloed traditional approach where we train a separate model for each task. We will get into the details in the later sections, meanwhile as shown in the figure, the transfer learning setup (as compared to traditional approach) leverages knowledge from source tasks to improve upon the learning on target task.

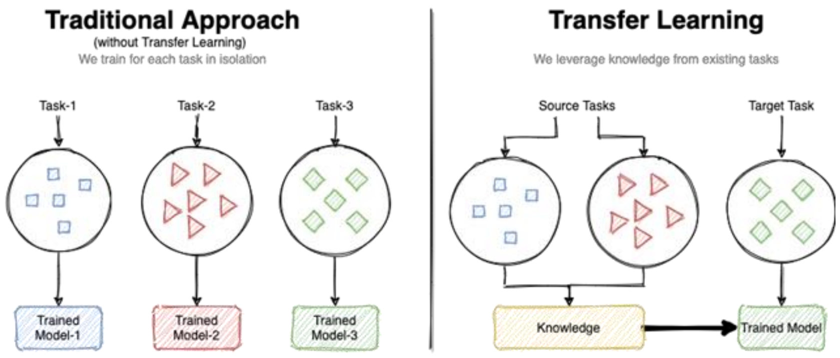


Fig 3.3.2 Transfer Learning

AlexNet [7]: AlexNet consists of eight layers, including five convolutional layers and three fully connected layers. The network uses the rectified linear unit (ReLU) activation function to introduce nonlinearity and prevent the vanishing gradient problem. The network also employs local response normalization (LRN) to improve generalization performance.

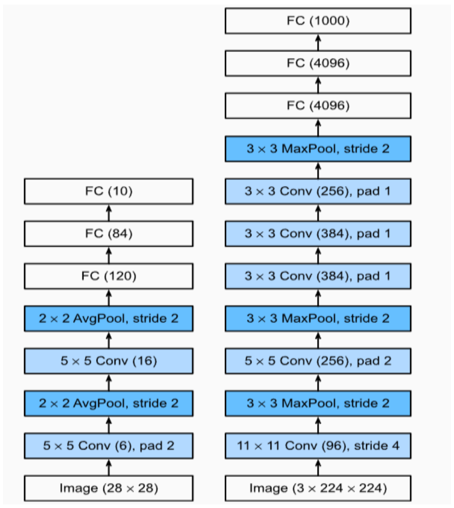


Fig 3.3.3 Alexnet

VGG 16[8]: The VGG model stands for the Visual Geometry Group from Oxford. The model was very simple and had a greater depth than AlexNet. The paper had two models with 16- and 19-layers depth. All the CNN layers were using 3 by 3 filters with stride and a pad of size 1 and a max pooling size of 2 with stride 2.

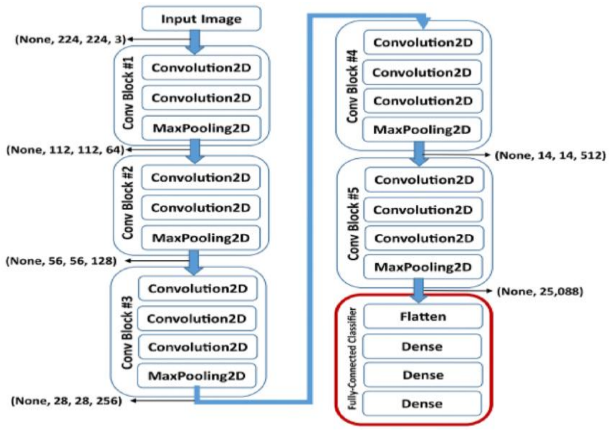


Fig 3.3.4 Vgg 16

ResNet 101: ResNet 101 is a CNN architecture developed by Microsoft Research, it consists of 101 layers, including convolutional layers, pooling layers, and fully connected layers. The key innovation of the ResNet architecture is the use of residual connections, which allow information to be passed directly from one layer to another, bypassing several layers in between. This helps to prevent the vanishing gradient problem and makes it possible to train very deep neural networks.

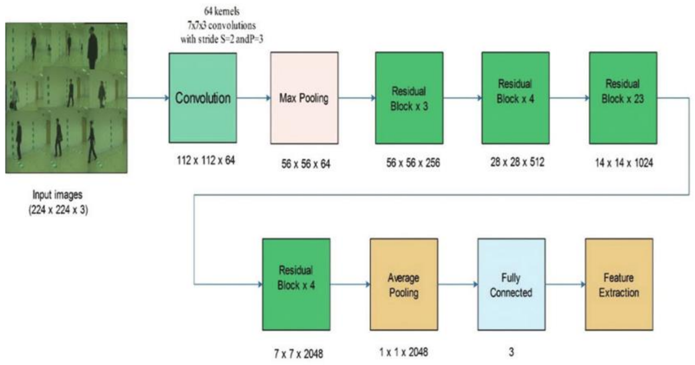


Fig 3.3.5 ResNet 101

MobileNet [9]: MobileNet is a type of convolutional neural network designed for mobile and embedded vision applications. They are based on a streamlined architecture that uses depth wise separable convolutions to build lightweight deep neural networks that can have low latency for mobile and embedded devices. Overall MobileNet is a powerful and efficient neural network architecture that enables real-time processing of images on resource-limited devices, making it an ideal choice for various computer vision applications such as object detection, image classification, and semantic segmentation on mobile devices.

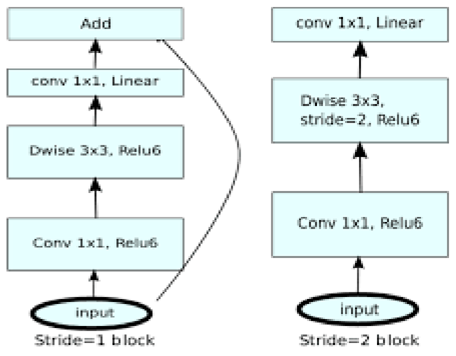


Fig 3.3.6 MobileNet

EfficientNet[10]: EfficientNet achieves its efficiency by using a combination of scaling techniques, including compound scaling, which scales the network depth, width, and resolution in a principled way, and a new type of mobile inverted bottleneck convolutional block. EfficientNet models have achieved models have achieved state-of-the-art results on a wide range of computer vision tasks, including image classification, object detection, and segmentation.

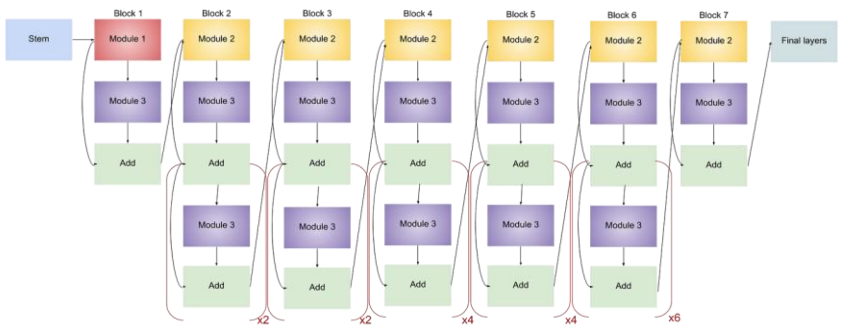


Fig 3.3.7 EfficientNet

3.4 Details of Hardware and Software

Software Used:

To build and operate the model, we utilized Google Collab

Python version 3.11.2

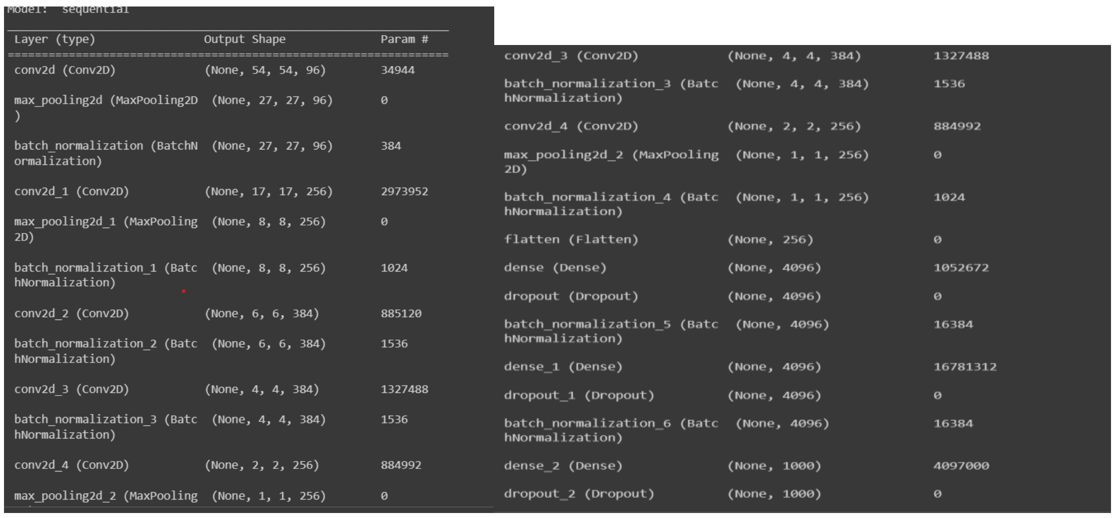
Libraries Used:

To implement various models and algorithms in the proposed model we have used some external libraries.

1. NumPy: used for numerical computing and array manipulation.
2. Matplotlib: used for data visualization.
3. Scikit-learn: used for data preprocessing, splitting the dataset, and evaluation of the model.
4. Keras: used for building and training the deep learning model.
5. Pandas: used for data manipulation and analysis.

3.5 Experiment and Results

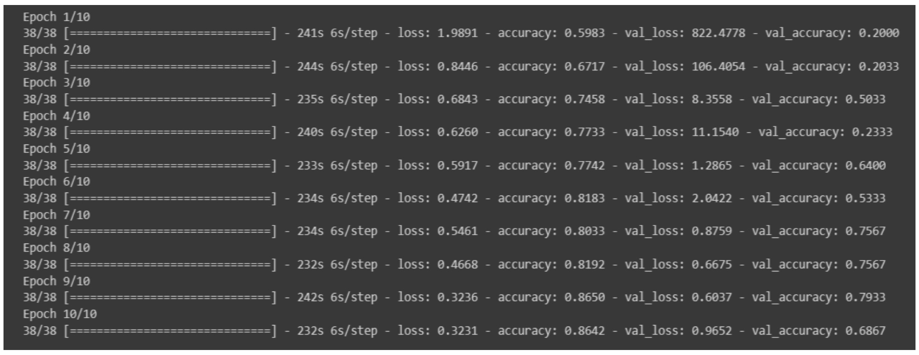
AlexNet:



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 54, 54, 96)        34944

 max_pooling2d (MaxPooling2D  (None, 27, 27, 96)       0
 )

 batch_normalization (BatchN  (None, 27, 27, 96)       384
 ormalization)

 conv2d_1 (Conv2D)           (None, 17, 17, 256)       2973952

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 256)        0
 2D)

 batch_normalization_1 (Batc  (None, 8, 8, 256)        1024
 hNormalization)

 conv2d_2 (Conv2D)           (None, 6, 6, 384)         885120

 batch_normalization_2 (Batc  (None, 6, 6, 384)        1536
 hNormalization)

 conv2d_3 (Conv2D)           (None, 4, 4, 384)         1327488

 batch_normalization_3 (Batc  (None, 4, 4, 384)        1536
 hNormalization)

 conv2d_4 (Conv2D)           (None, 2, 2, 256)         884992

 max_pooling2d_2 (MaxPooling  (None, 1, 1, 256)        0
```

```
 conv2d_3 (Conv2D)           (None, 4, 4, 384)         1327488

 batch_normalization_3 (Batc  (None, 4, 4, 384)        1536
 hNormalization)

 conv2d_4 (Conv2D)           (None, 2, 2, 256)         884992

 max_pooling2d_2 (MaxPooling  (None, 1, 1, 256)        0
 2D)

 batch_normalization_4 (Batc  (None, 1, 1, 256)        1024
 hNormalization)

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 4096)              1052672

 dropout (Dropout)           (None, 4096)              0

 batch_normalization_5 (Batc  (None, 4096)             16384
 hNormalization)

 dense_1 (Dense)             (None, 4096)              16781312

 dropout_1 (Dropout)         (None, 4096)              0

 batch_normalization_6 (Batc  (None, 4096)             16384
 hNormalization)

 dense_2 (Dense)             (None, 1000)              4097000

 dropout_2 (Dropout)         (None, 1000)              0
```

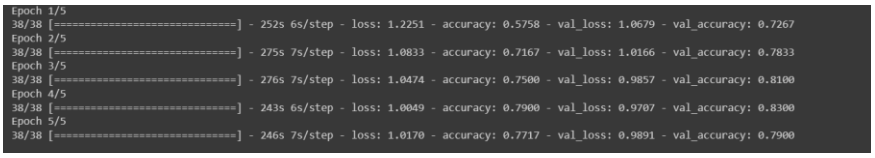
Fig 3.5.1 Alexnet Parameters



```
Epoch 1/10
38/38 [==============================] - 241s 6s/step - loss: 1.9891 - accuracy: 0.5983 - val_loss: 822.4778 - val_accuracy: 0.2000
Epoch 2/10
38/38 [==============================] - 244s 6s/step - loss: 0.8446 - accuracy: 0.6717 - val_loss: 106.4054 - val_accuracy: 0.2033
Epoch 3/10
38/38 [==============================] - 235s 6s/step - loss: 0.6843 - accuracy: 0.7458 - val_loss: 8.3558 - val_accuracy: 0.5033
Epoch 4/10
38/38 [==============================] - 240s 6s/step - loss: 0.6260 - accuracy: 0.7733 - val_loss: 11.1540 - val_accuracy: 0.2333
Epoch 5/10
38/38 [==============================] - 233s 6s/step - loss: 0.5917 - accuracy: 0.7742 - val_loss: 1.2865 - val_accuracy: 0.6400
Epoch 6/10
38/38 [==============================] - 234s 6s/step - loss: 0.4742 - accuracy: 0.8183 - val_loss: 2.0422 - val_accuracy: 0.5333
Epoch 7/10
38/38 [==============================] - 234s 6s/step - loss: 0.5461 - accuracy: 0.8033 - val_loss: 0.8759 - val_accuracy: 0.7567
Epoch 8/10
38/38 [==============================] - 232s 6s/step - loss: 0.4668 - accuracy: 0.8192 - val_loss: 0.6675 - val_accuracy: 0.7567
Epoch 9/10
38/38 [==============================] - 242s 6s/step - loss: 0.3236 - accuracy: 0.8650 - val_loss: 0.6037 - val_accuracy: 0.7933
Epoch 10/10
38/38 [==============================] - 232s 6s/step - loss: 0.3231 - accuracy: 0.8642 - val_loss: 0.9652 - val_accuracy: 0.6867
```

Fig 3.5.2 Alexnet epoch

Densenet:



```
Epoch 1/5
38/38 [==============================] - 252s 6s/step - loss: 1.2251 - accuracy: 0.5758 - val_loss: 1.0679 - val_accuracy: 0.7267
Epoch 2/5
38/38 [==============================] - 275s 7s/step - loss: 1.0833 - accuracy: 0.7167 - val_loss: 1.0166 - val_accuracy: 0.7833
Epoch 3/5
38/38 [==============================] - 276s 7s/step - loss: 1.0474 - accuracy: 0.7500 - val_loss: 0.9857 - val_accuracy: 0.8100
Epoch 4/5
38/38 [==============================] - 243s 6s/step - loss: 1.0049 - accuracy: 0.7900 - val_loss: 0.9707 - val_accuracy: 0.8300
Epoch 5/5
38/38 [==============================] - 246s 7s/step - loss: 1.0170 - accuracy: 0.7717 - val_loss: 0.9891 - val_accuracy: 0.7900
```

Fig 3.5.3 Densenet

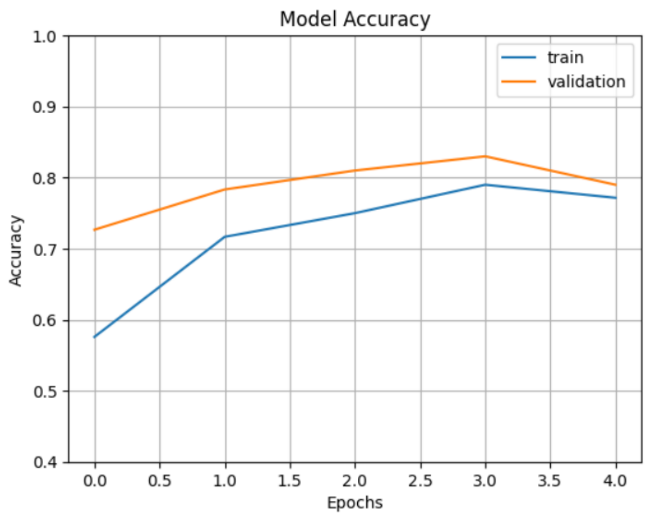
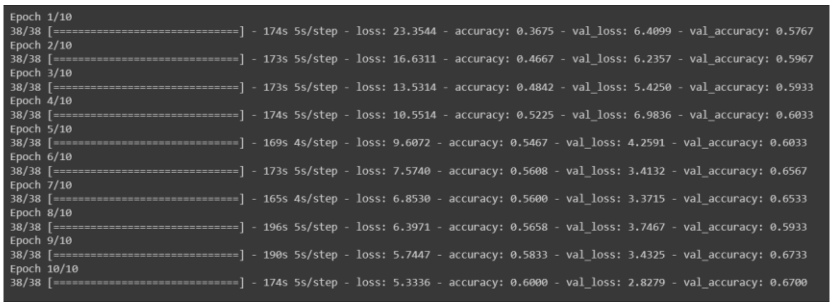


Fig 3.5.4 DenseNet Accuracy

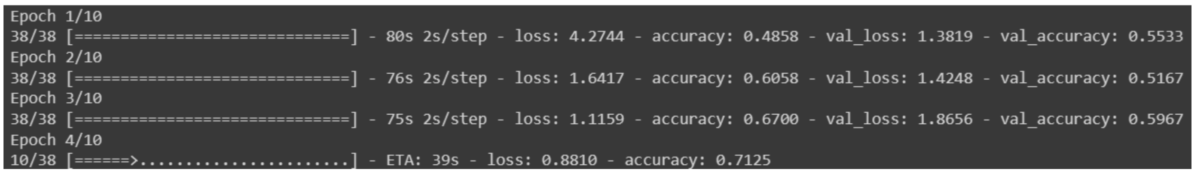
Inceptionv3:



```
Epoch 1/10
38/38 [==============================] - 174s 5s/step - loss: 23.3544 - accuracy: 0.3675 - val_loss: 6.4099 - val_accuracy: 0.5767
Epoch 2/10
38/38 [==============================] - 173s 5s/step - loss: 16.6311 - accuracy: 0.4667 - val_loss: 6.2357 - val_accuracy: 0.5967
Epoch 3/10
38/38 [==============================] - 173s 5s/step - loss: 13.5314 - accuracy: 0.4842 - val_loss: 5.4250 - val_accuracy: 0.5933
Epoch 4/10
38/38 [==============================] - 174s 5s/step - loss: 10.5514 - accuracy: 0.5225 - val_loss: 6.9836 - val_accuracy: 0.6033
Epoch 5/10
38/38 [==============================] - 169s 4s/step - loss: 9.6072 - accuracy: 0.5467 - val_loss: 4.2591 - val_accuracy: 0.6033
Epoch 6/10
38/38 [==============================] - 173s 5s/step - loss: 7.5740 - accuracy: 0.5608 - val_loss: 3.4132 - val_accuracy: 0.6567
Epoch 7/10
38/38 [==============================] - 165s 4s/step - loss: 6.8530 - accuracy: 0.5600 - val_loss: 3.3715 - val_accuracy: 0.6533
Epoch 8/10
38/38 [==============================] - 196s 5s/step - loss: 6.3971 - accuracy: 0.5658 - val_loss: 3.7467 - val_accuracy: 0.5933
Epoch 9/10
38/38 [==============================] - 190s 5s/step - loss: 5.7447 - accuracy: 0.5833 - val_loss: 3.4325 - val_accuracy: 0.6733
Epoch 10/10
38/38 [==============================] - 174s 5s/step - loss: 5.3336 - accuracy: 0.6000 - val_loss: 2.8279 - val_accuracy: 0.6700
```

Fig 3.5.5 Inceptionv3

MobileNet:



```
Epoch 1/10
38/38 [==============================] - 80s 2s/step - loss: 4.2744 - accuracy: 0.4858 - val_loss: 1.3819 - val_accuracy: 0.5533
Epoch 2/10
38/38 [==============================] - 76s 2s/step - loss: 1.6417 - accuracy: 0.6058 - val_loss: 1.4248 - val_accuracy: 0.5167
Epoch 3/10
38/38 [==============================] - 75s 2s/step - loss: 1.1159 - accuracy: 0.6700 - val_loss: 1.8656 - val_accuracy: 0.5967
Epoch 4/10
10/38 [======>.......................] - ETA: 39s - loss: 0.8810 - accuracy: 0.7125
```

Fig 3.5.6 MobileNet

ResNet50:

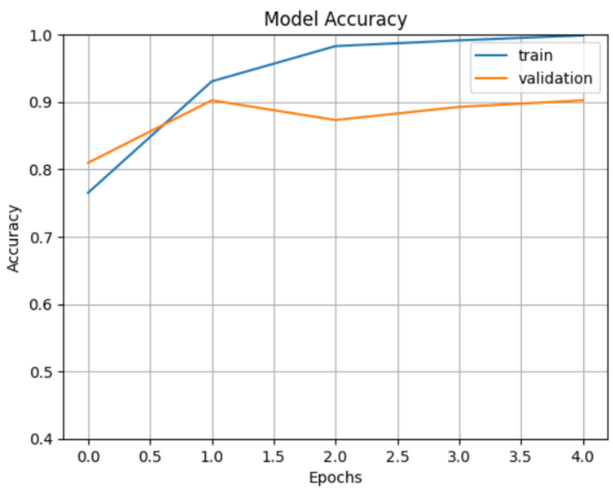


Fig 3.5.7 ResNet50 Model

ResNet101:

```
/content/drive/MyDrive/dummy_sugarcane/Healthy/0.jpg
```

Fig 3.5.8 ResNet101 Leaf

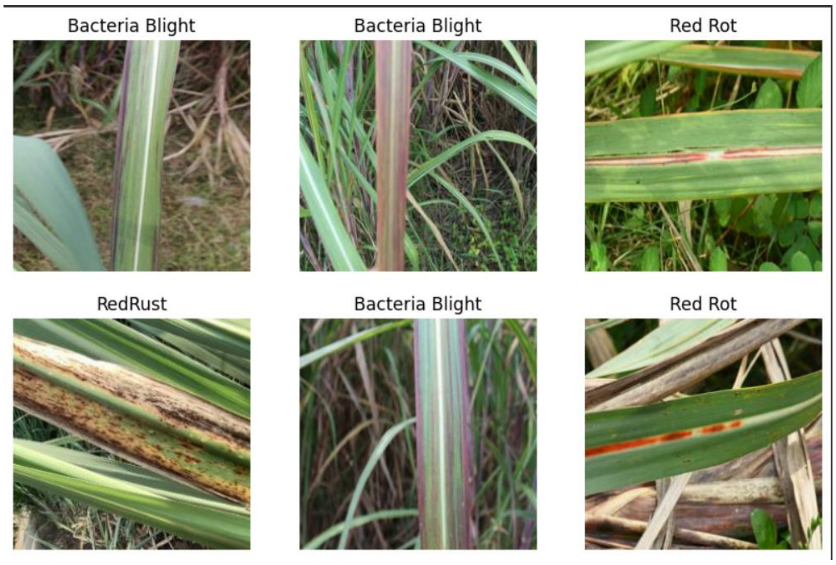


Fig 3.5.9 ResNet101 Diseases

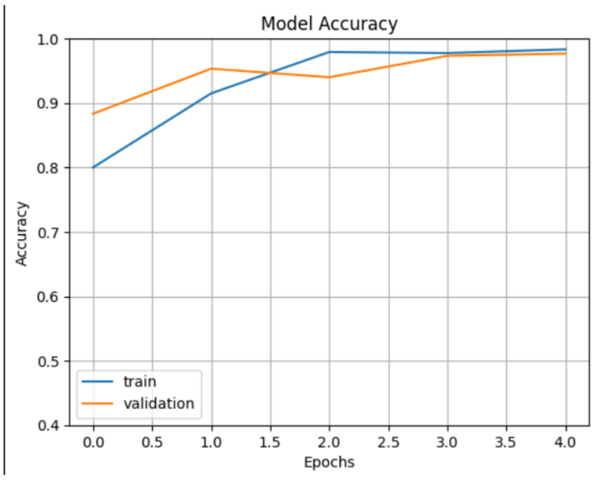


Fig 3.5.10: ResNet101 Model Accuracy

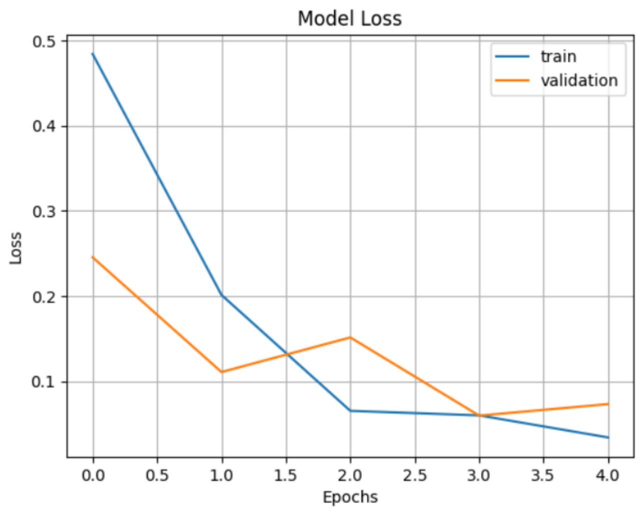
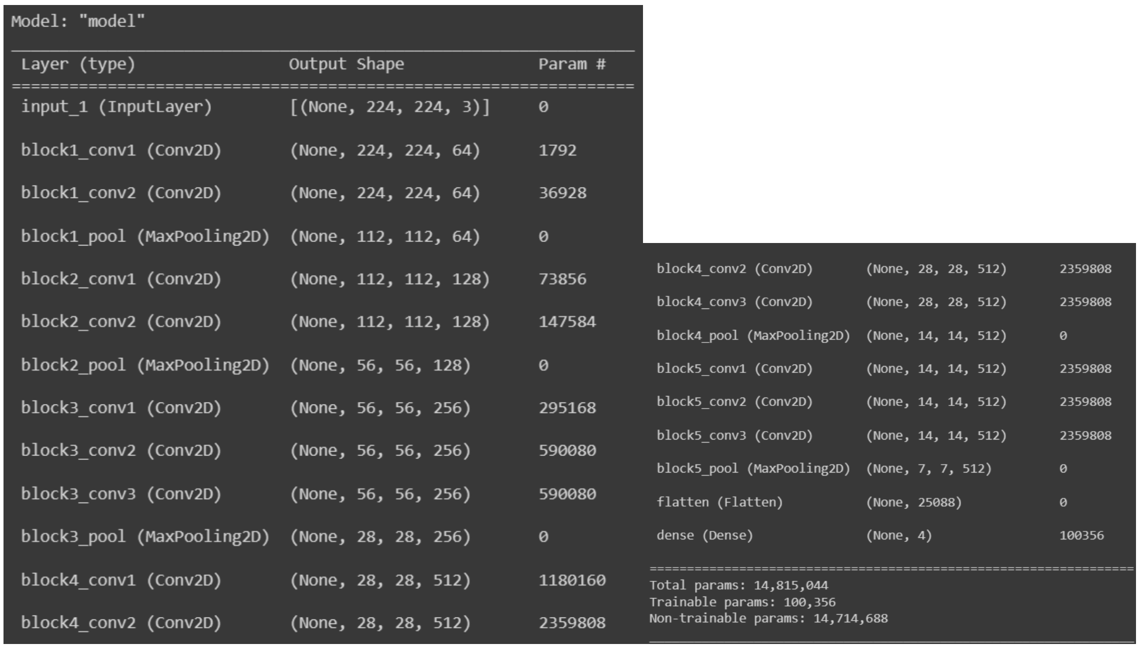


Fig 3.5.11 ResNet101 Model Loss

Vgg 16:


```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
```

```
 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 4)                 100356

=================================================================
Total params: 14,815,044
Trainable params: 100,356
Non-trainable params: 14,714,688
_________________________________________________________________
```
Fig 3.5.12 VGG 16

Vgg19:

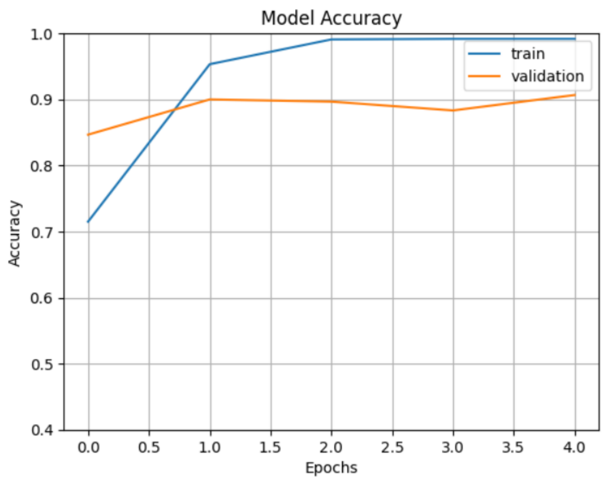


Fig 3.5.13 Vgg19 Model Accuracy

EfficientNet:

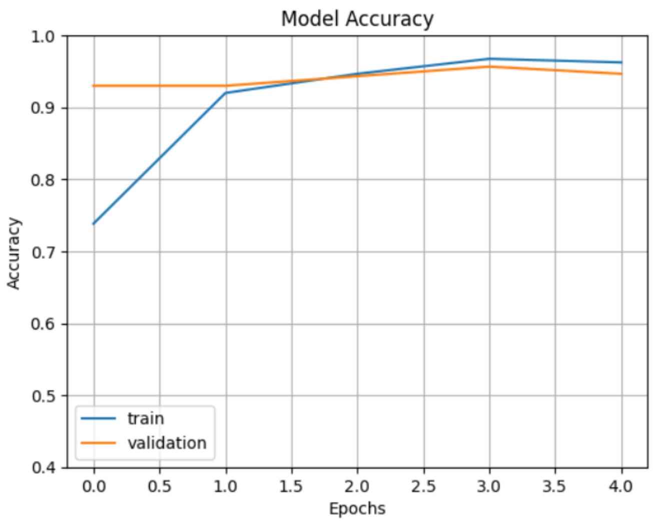
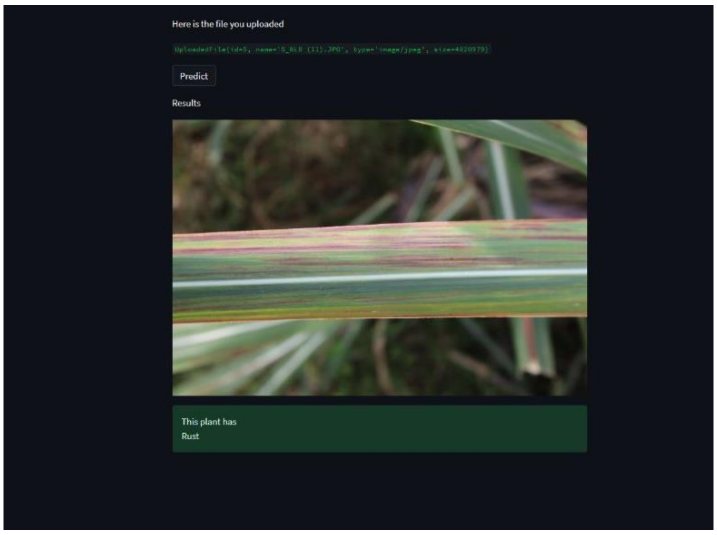


Fig 3.5.14 EfficientNet Model Accuracy



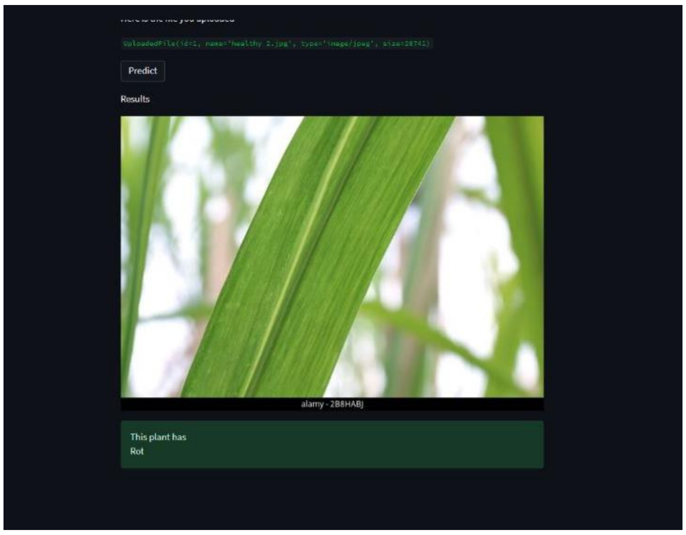
Here is the file you uploaded

```
UploadedFile(id=5, name='S_BLB (11).JPG', type='image/jpeg', size=4820579)
```

Results

This plant has
Rust

Fig 3.5.15 Rust Disease Prediction

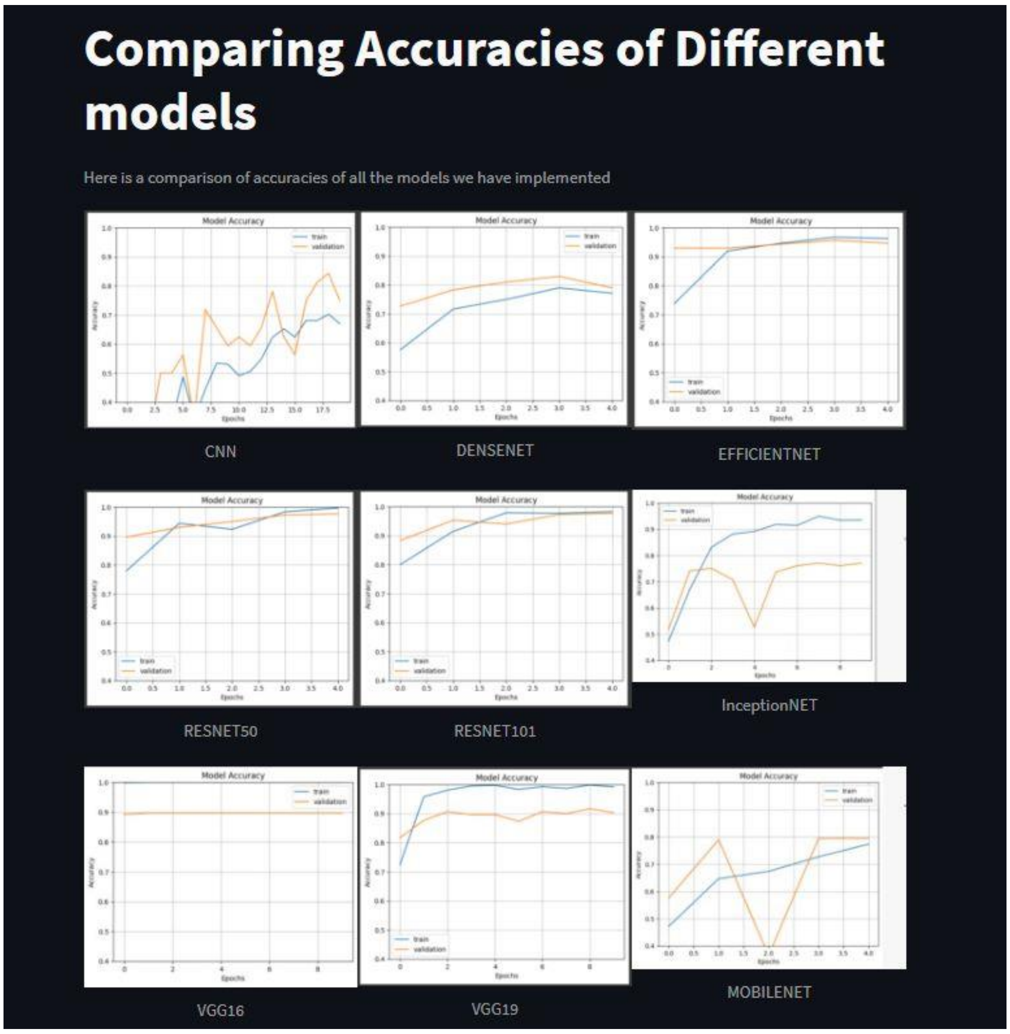


```
UploadedFile(id=1, name='healthy 2.jpg', type='image/jpeg', size=28741)
```

Results

This plant has
Rot

Fig 3.5.16 Rot Disease Prediction



Comparing Accuracies of Different models

Here is a comparison of accuracies of all the models we have implemented

Fig 3.5.17 Model Accuracies Comparison

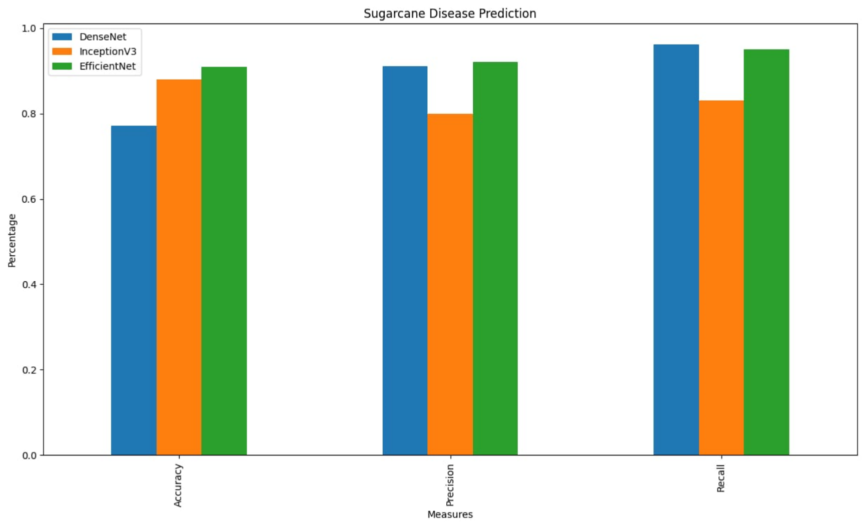


Fig 3.5.18 Comparison of performance for different DL models

Chapter 4: Conclusion

4.1 Conclusion

In conclusion, the proposed sugarcane disease prediction system using deep learning and transfer learning has shown promising results in detecting and classifying sugarcane diseases. The use of a pre-trained model and transfer learning technique has helped in achieving higher accuracy with limited training data. The Adam optimizer has also played a crucial role in improving the training process and reducing the loss.

The system has been tested on three major sugarcane diseases, namely Red Rust, Red Rot, and Bacterial Blight, as well as healthy crop images.

The future scope of the system includes incorporating additional sugarcane diseases, expanding the dataset with more images, and improving the interpretability of the model. The system can also be integrated with other agricultural systems to provide a comprehensive solution for sugarcane cultivation.

Overall, the sugarcane disease prediction system can be a valuable tool for sugarcane farmers and researches to identify and mitigate the impact of sugarcane diseases, leading to better yields and a more sustainable sugarcane industry

4.2 Future Scope

In future, we plan to add certain things which would improve the sugarcane disease prediction model that will allow the users to get more accurate results.

Expansion of dataset: The current model is trained on a limited dataset of four classes(Red Rust, Red Rot, Bacterial Blight and Healthy Crop). The system can be improved by expanding the dataset to include other sugarcane diseases and pest infestations.

Model Optimization: The system can be further optimized by exploring different deep learning architectures and algorithms to achieve higher accuracy and reduce false positives.

Real-time disease detection: The system can be deployed on a mobile platform and integrated with a camera to enable real time detection of sugarcane diseases in the field.

Multi-spectral imaging: multi-spectral imaging can be used to capture images at different wavelengths, which can provide more detailed information about the health of sugarcane crops. This can be used to develop a more accurate and robust disease detection system.

Integration with IoT devices: The system can be integrated with IoT devices such as sensors and weather stations to collect real-time data on environmental factors that may affect the growth and health of sugarcane crops.

Collaboration with experts: Collaborating with domain experts such as agronomists and plant pathologists can provide valuable insights and help improve the accuracy and effectiveness of the system.

References

[1] Militante, Sammy V., Bobby D. Gerardo, and Ruji P. Medina. "Sugarcane disease recognition using deep learning." *2019 IEEE Eurasia conference on IOT, communication and engineering (ECICE)*. IEEE, 2019.

[2] Militante, Sammy V., and Bobby D. Gerardo. "Detecting sugarcane diseases through adaptive deep learning models of convolutional neural network." *2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*. IEEE, 2019.

[3] RAJESH, TR, and V. Gowri. "Enhanced Approach for Disease Prediction in Sugarcane Crop with the support of advanced machine learning strategies." *Annals of the Romanian Society for Cell Biology* (2021): 16805-16814.

[4] Malik, Hashmat Shadab, et al. "Disease recognition in sugarcane crop using deep learning." *Advances in Artificial Intelligence and Data Engineering: Select Proceedings of AIDE 2019*. Springer Singapore, 2021.

[5] Amarasingam, Narmilan, et al. "Detection of White Leaf Disease in Sugarcane Crops Using UAV-Derived RGB Imagery with Existing Deep Learning Models." *Remote Sensing* 14.23 (2022): 6137.

[6] Shorten, Connor, and Taghi M. Khoshgoftaar. "A survey on image data augmentation for deep learning." *Journal of big data* 6.1 (2019): 1-48.

[7] Sarkar, Raghav Bali Dipanjan. "Transfer Learning in Action." (2021): 69-76.

[8] "Dive into Deep Learning — Dive into Deep Learning 1.0.0-beta0 documentation." https://d2l.ai/chapter_convolutional-modern/alexnet.html.

[9] Shanmugamani, Rajalingappaa. *Deep Learning for Computer Vision: Expert techniques to train advanced neural networks using TensorFlow and Keras*. Packt Publishing Ltd, 2018.

[10] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." *arXiv preprint arXiv:1704.04861* (2017).

[11] Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." *International conference on machine learning*. PMLR, 2019.

[12] Zhang, Zijun. "Improved adam optimizer for deep neural networks." *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)*. IEEE, 2018.