PLANT DISEASE PREDICTION USING DEEP LEARNING

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Raghav Dalmia (191298)

Under the supervision of

Dr. Aman Sharma

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Plant Disease Prediction Using Deep Learning"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2022 to May 2023 under the supervision of **Dr. Aman Sharma,** Assistant Professor(Senior Grade), Computer Science & Engineering and Information Technology.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Raghav Dalmia, 191298.

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

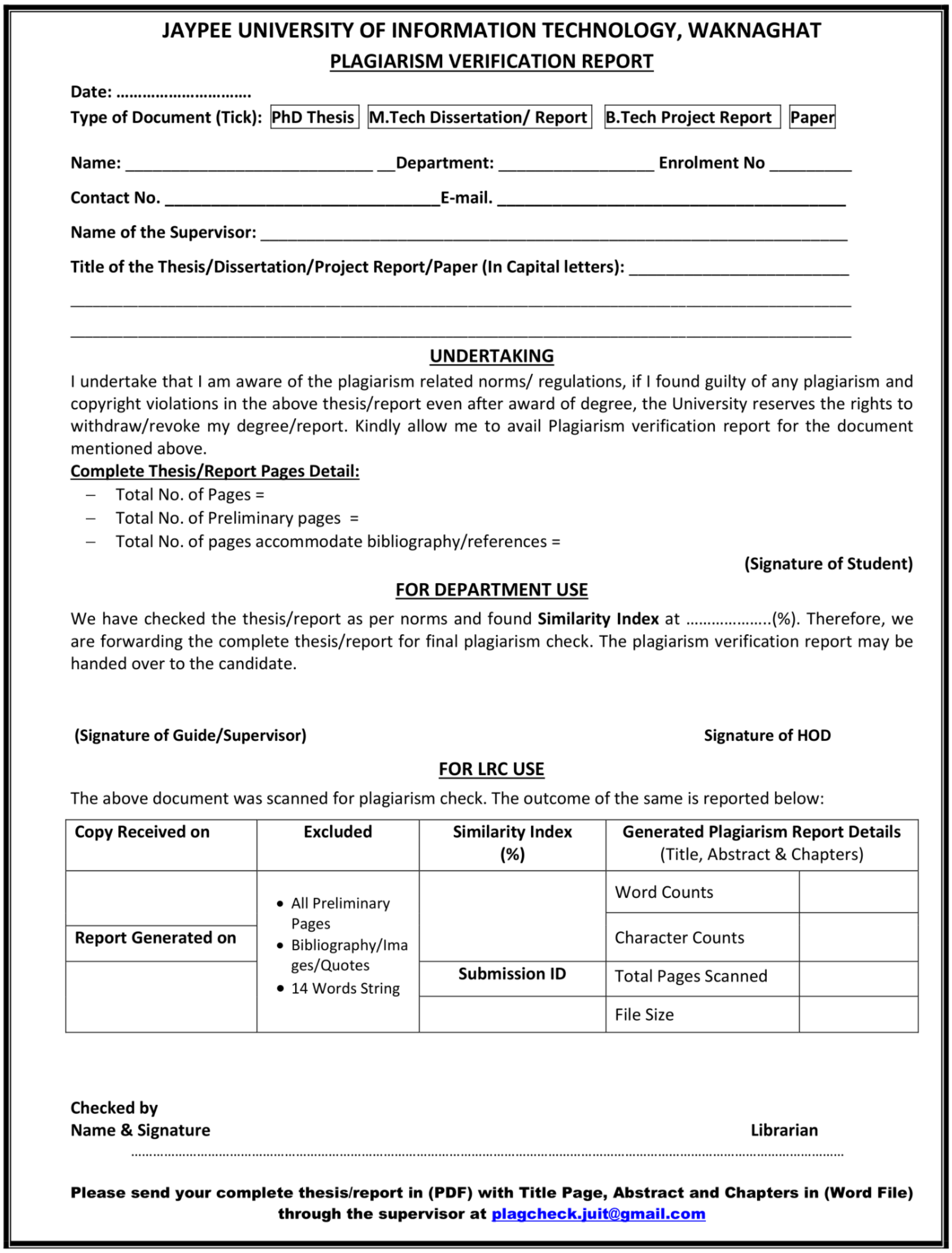
Dr. Aman Sharma

Assistant Professor(Senior Grade)

Computer Science & Engineering and Information Technology (CSE&IT)

Dated:

Plagiarism Certificate



JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ……………………….

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: ___________________________ __**Department:** _________________ **Enrolment No** _________

Contact No. ______________________________**E-mail.** ______________________________________

Name of the Supervisor: ____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): ________________________

____

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

– Total No. of Pages =
– Total No. of Preliminary pages =
– Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ……………….(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

………

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

Acknowledgement

I want to start by expressing my profound gratitude and compliments to the Almighty God. We are able to effectively complete the project work thanks to his divine blessings.

We want to express our sincere gratitude and debt of gratitude to Dr. Vivek Kumar Sehgal, Professor and Head of the Department of Computer Science and Information Technology at the Jaypee University of Information Technology, Waknaghat, for giving us the chance to work on this project. We are even more grateful to Dr. Aman Sharma, Assistant Professor (Senior Grade) as we were able to work under his guidance and were always supportive to us. Our supervisor's deep knowledge and great interest in the field of "Machine Learning" was the cherry on top to carry out this project. His constant patience, his scientific guidance, his constant encouragement, his constant energetic supervision, his constructive criticism, his valuable advice, his reading of many poor quality drafts and correcting them every step of the way, made this possible.

We also warmly welcome each individual who has directly or indirectly assisted us in making this project a success.

I would like to thank the various trained and untrained employees who have developed practical assistance and facilitated my company in this unique situation.

Finally, I must respectfully recognise my parent’s unwavering help and tolerance.

Regards

Project Group No. :
Student Name: Raghav Dalmia
Rollno.: 191298

Table Of Content

List Of Abbreviations

S. No	Abbreviation	Definition
1.	CNN	Convolutional Neural Network
2.	DL	Deep Learning
3.	VGG	Visual Geometry Group
4.	MLP	Multi-layer Perceptron

List Of Figures

List Of Graphs

List Of Tables

Abstract

Since plant diseases are one of the main factors affecting food production and reducing production losses, they must be swiftly identified and treated. India's agricultural sector employs close to 50% of the workforce, thus not having an appropriate solution would affect the livelihood of many people. Different deep learning algorithms have recently found usage in the diagnosis of plant diseases, offering a potent tool with highly accurate results. The objective of this study is to identify an ensemble-based solution by using several algorithms in the process of classifying and diagnosing plant diseases, describing trends, and emphasizing gaps and also comprises complete examination of the literature. The ensemble based solution is based on the top four performing deep learning algorithms using multi-layered perceptron as meta classifier. In this regard, we reviewed 15 studies from the previous three years that address problems with disease detection, dataset characteristics, researched crops, and pathogens in various ways. The proposed ensemble model achieved a maximum accuracy of 98.13% compared to the conventional architectures. For comparing the results, various performance metrics are used such as accuracy, loss, etc.

CHAPTER-1
INTRODUCTION

1.1. Introduction

Modern technology has given human society the capacity to supply sufficient meals to satisfy the call for extra than 7 billion people. However, a number of factors, such as climatic change, a loss in pollinators, plant diseases, and others continue to pose a danger to food security. Plant illnesses aren't the handiest danger to meals protection at the worldwide scale, however it can also have disastrous results for smallholder farmers whose livelihoods rely upon wholesome crops [1]. In the growing world, extra than eighty percent of the rural manufacturing is generated with the aid of using smallholder farmers, and reviews of yield lack of extra than 50% because of pests and illnesses are common. Additionally, the majority of the world's hungry people (50%) reside in homes with smallholder farmers, rendering this group particularly vulnerable to disruptions in food supply brought on by pathogens.

Technology in agriculture not only improves the precision of spotting plant diseases but also lessens the possibility of crop failure. Farmers are shifting to more accessible and inexpensive agricultural practices as the difficulty in the agriculture industry in terms of prices and the likelihood of crop failure due to natural conditions increases [2,3]. Machines that employ machine learning algorithms—which are quicker and more accurate than the human eye in predicting plant illnesses are being used to achieve this.

The Deep Learning (DL) technique is a subclass of Machine Learning (ML), which was introduced in 1943 as a way to systematically build a computer version that mimics the natural paths of humans. The evolution of this field of study, which may be divided into two historical periods—from 1943 to 2006 and from 2012 to the present—continuous. Many trends, including backpropagation, chain rule, Neocognitron, handwritten text reputation (LeNET architecture), and resolving the educational problem, had been discovered during the first phase. Modern algorithms/architectures, however, were developed for many applications in the second phase, including self-driving cars, the healthcare industry, text reputation, earthquake forecasts, marketing, finance, and image reputation. Among these architectures, AlexNet is regarded as a breakthrough in the field of deep learning (DL) because it was awarded the ImageNet project for product reputation known as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in the year 2012. Soon after, numerous architectures had been brought to conquer the loopholes discovered previously. Many comparative metrics are used to compare the outcomes of the performed architectures, such as Accuracy, Loss Function, Precision, Recall and F1 Score. Even the proposed ensemble architecture based on the blending is compared with literature survey to provide better comparison.

1.2. Problem Statement

Agriculture is one of the most significant economic sectors in India. India's agricultural sector employs close to 50% of the workforce. The largest producer of beans, rice, wheat, spices, and spice-related items is believed to be India. The quality of the items a farmer produces, which in turn depends on the development of their crops and the yields realized, determines the farmer's economic progress. Therefore, detecting plant diseases is crucial in agriculture. The environment of the farmer is impacted by illnesses that are particularly vulnerable to affect plant growth. Automated disease detection techniques are advantageous for detecting plant diseases at a very early stage. The leaves of the plant, for example, can display symptoms of a plant disease. It is time-consuming and expensive to manually diagnose plant diseases using photographs of the leaves. In order to automate the process of disease identification and categorization using leaf photos, a computational approach must be developed. Changes in disease control strategies present significant challenges for farmers.

The project's goal is to create a machine-learning system that can predict plant and agricultural diseases at an early stage with a greater accuracy rate for improved quality and quantity in order to solve this issue. The system may perform analysis of different plant variables using the images of leaves to discover correlations among different diseases and then create a prediction model by employing an ensemble method. This approach could offer a more effective and precise diagnostic for plant diseases, therefore, increasing the yield exponentially and better quality supply could be assured. Whereas increasing the model's accuracy, precision and loss while reducing false positives and false negatives is the major difficulty being faced.

1.3. Objectives

- Design, implementation, and evaluation of an image processing-based plant disease prediction system are the main objectives of this project.

- The foremost objective is to minimize the economic and aesthetic damage caused by plant diseases by using stack ensemble model.
- Test and validate the existing architecture.

- Compare the results of the proposed ensemble model with existing state-of-the-art literature using different metrics .
- By achieving these objectives, the main objective is to revolutionize the agriculture sector by early detection of plant disease. Thus, preventing mass destruction of crops.

1.4. Methodology

Corn(maize) plant dataset is used to perform the prediction. The dataset undergoes a variety of picture preprocessing operations, such as normalization, which projects image data pixels to a preset range, and data augmentation, which involves making modest changes to the existing data to boost diversity so that the model can process it more quickly. The dataset is then divided in a 3:1 ratio between training and testing data.

The initial data is then input into a variety of models in the suggested ensemble model. The estimation of each model's input and output, as well as the weights, are done by the meta classifier. All other models are eliminated, and only the best-performing ones are chosen. To increase accuracy and results, this stacking ensemble technique is used.

As shown in the table, the traditional deep learning architectures that performed really well on the dataset are Xception, Inception V3, ResNet 50 and CNN. Therefore, all four deep learning architectures are considered as input to the ensemble model as level 0 models. Multi-layer Perceptron is used as a level 1 classifier in the proposed model so as to improve the results. Fig4. shows our work, i.e. the Data is taken from Github which is deeply analyzed and the proposed model is trained with 75% of the total data.

1.5. Organization

The project report comprises 5 major sections.

Chapter 1: Introduction

The goal of this chapter is to address the problem description, suggest techniques or further study, and highlight the effectiveness of the suggested application. It also discusses the background and motivation behind the proposed architecture.

Chapter 2: Literature Survey

The review of the literature used to assess the theories studied and comprehended with cited sources is covered in this chapter.

Chapter 3: System Development

This chapter includes the things which led to the development of the project. This consists of the background of the Dataset, the preprocessing step, which prepares the dataset for further process. Further, the analysis of the proposed model is done with the help of an Algorithm, Flowchart and steps involved in ensemble model. Even the background of the used architectures is also mentioned.

Chapter 4: Performance Analysis

This chapter consists of comparing the results obtained and visualizing the dataset. Even the results snapshot is included. The comparative results are presented in the form of tables, figures and graphs.

Chapter 5: Conclusion

It consists of the opinion reached after performing all the necessary steps. It also consists of the future scope of the project and the application contribution.

CHAPTER-2
LITERATURE SURVEY

This chapter discusses many research papers and articles to establish how other scholars have handled the issue and used diverse methodologies. Plant diseases have been multiplying dramatically[4]. To forecast plant diseases and address the aforementioned issues, researchers have employed a range of strategies and algorithms over time. While some methods preferred online datasets, several approaches utilized live picture datasets.

Wagle and Harikrishnan [5], performed analysis over a dataset of various crops using SVM with linear kernel as well as radial function kernel and pre-trained AlexNet. Comparisons are based over different percentages of the dataset used for training. Islam, Shuvo et al. [6], compared four different Deep CNN Models of which Inception-ResNet-V2 performed the best. It was followed by Xception and then ResNet-101.

Picon, Alvarez-Gila et al. [7], devised a mobile capture device which can be used in real field conditions to classify crop diseases. Many improvements are made to enhance the results. Fenu and Malloci [8], collected field data mainly affected by three diseases. Image pre-processing and data augmentation is performed to train the data by six well known deep learning algorithms. VGG16 is one of the architectures used and it provided satisfactory results.

Sujatha, Chatterjee et al. [9], concluded that DL algorithms provided appreciable results compared to ML algorithms over a manually collected dataset of citrus leaves. Many metrics are considered such as Accuracy, Precision, Recall and AUC for the 6 models used. InceptionV3 provided the best results with 89% Accuracy, 89.2% Precision, 89% Recall and 97% AUC under the DL category whereas SVM provided best results under ML category with 87% Accuracy, 87.3% Precision, 87.1% Recall and 96.5% AUC. Bi, Wang et al. [10], devised an architecture that is cost effective and can easily be deployed on mobile phones. A dataset of 334 images were manually collected by agriculture experts and the results were compared with different architectures. MobileNet had the best handling time per image of 0.22sec.

Lu, Yi et al. [11], proposed the most simple architecture for plant disease prediction using 5 layers. The proposed architecture overpowers the results of BP Method, SVM and Particle Swarm Optimization (PSO) as well. Waheed, Goyal et al. [12], proposed an architecture based on categorical cross entropy as model's loss function and Adam optimizer as the model optimizer whereas for computational efficiency ReLU function is used. Influencing hyperparameters resulted in change in models' accuracy whereas, data augmentation helped in the generalization of the model.The proposed architecture was compared with various existing CNN architectures.

Jadhav, Udipi et al. [13], proposed pre-trained GoogleNet and AlexNet by modifying hyperparameters based on the dataset collected from soybean fields. GoogleNet outperformed other architectures with 98.75% Accuracy. Shrestha, Deepsikha, et al. [14], focussed on a single architecture with accuracy 88.80% with no overfitting. A balanced dataset of 200 images per class is used for analysis which is converted into a numpy array which is further multiplied to give an output that is used to extract features from the images. AlexNet is one of the most frequently used algorithms.

A comparison between AlexNet and VGG16 based on accuracy has been proposed by Rangaranjan, Purushothama et al [15]. Model performance was evaluated by hyperparameter tuning. Patil, Kumar et al. [16], proposed a model of plant disease prediction involving multiple algorithms with ANN having an accuracy of 90.79%.

Table 1. Survey of Pre-existing Architectures

S.No.	Author(s)	Approach	Crops	Performance Metrics	Limitations
1.	Wagle and Harikrishnan (2021) [5]	SVM with linear kernel and radial function kernel, AlexNet	Apple, Cherry, Corn, Grape, Peach, Pepper, Potato, Strawberry, Tomato	Accuracy	Comparison with other architectures isn't mentioned.
2.	Islam, Shuvo et al. (2021) [6]	Xception, Inception-ResNet-V2, ResNet-101, VGG19	Paddy	Accuracy, Precision, Recall, F1 Score	The small size of the dataset.
3.	Picon, Alvarez-Gila et al. (2018) [7]	ResNet50	Wheat	Accuracy	The model wasn't able to perform well enough.
4.	Fenu and Malloci (2021)	VGG-16, VGG-19,	Pear	Accuracy, Training	Not enough metrics were

	[8]	ResNet50, InceptionV3, MobileNetV2, EfficientNetB0		Time	considered for comparing the results.
5.	Sujatha, Chatterjee et al. (2021) [9]	VGG16, InceptionV3, VGG19, SVM, SGD, Random Forest	Citrus	Accuracy, Precision, Recall, AUC	The architecture wasn't able to perform well enough.
6.	Bi, Wang et al. (2020) [10]	MobileNet, InceptionV3, ResNet152	Apple	Accuracy	Other models won't be compatible with mobile applications.
7.	Lu, Yi et al. (2017) [11]	CNN, BP Method, SVM, Particle Swarm Optimization (PSO)	Rice	Accuracy	It didn't encode the position and orientation of the object.
8.	Waheed, Goyal et al. (2020) [12]	DenseNet, VGG19, Xception, NasNet, EfficientNet-B0	Corn	Accuracy, Training Time	Not enough metrics were considered for comparing the results.
9.	Jadhav, Udipi et al. (2020) [13]	GoogleNet and AlexNet	Soyabean	Accuracy	The small size of the dataset.
10.	Shrestha, Deepsikha, et al. (2020) [14]	CNN	Potato	Accuracy	There is no comparative analysis with other existing architectures.
11.	Rangaranjan, Purushothama et al. (2018) [15]	VGG16 and AlexNet	Tomato	Accuracy	Faced exploding gradient problem with some hyper parameters.
12.	Patil, Kumar et al. (2019) [16]	CNN, RNN, ANN, SVM, KNN	Tomato, Potato	Accuracy, Precision, F1 Score	Machine Learning models weren't able to perform well in comparison to DL

					architectures.

CHAPTER-3
SYSTEM DEVELOPMENT

This chapter consists of the details about the dataset, the trends of the dataset. The dataset is visually presented in the form of graphs and tables. The proposed model is also described and the development leading to this.

3.1. Dataset

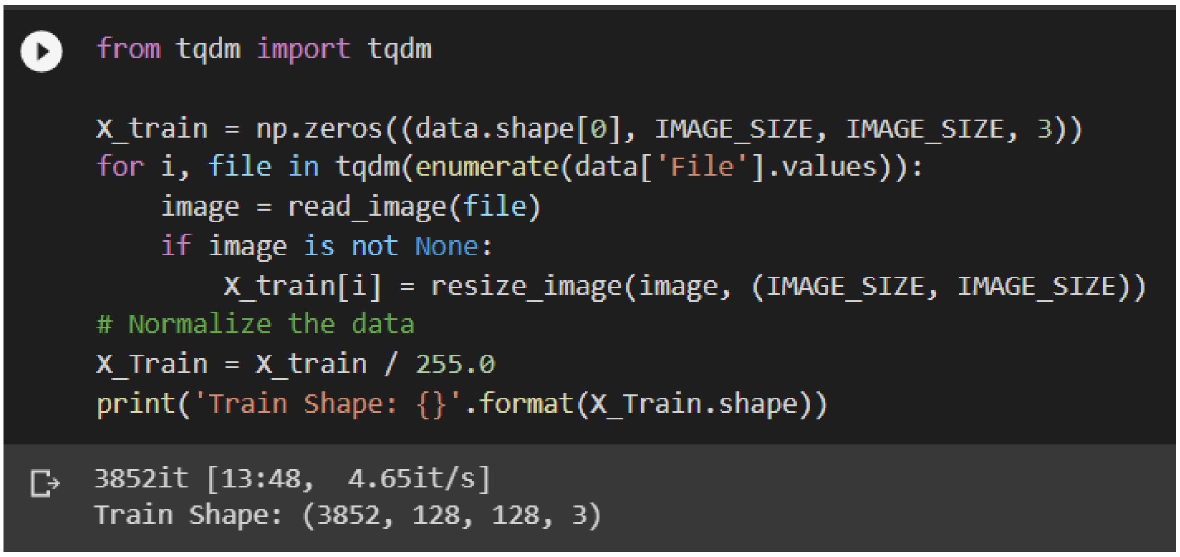
The dataset is based on the union of Github (PlantVillage-Dataset), extracted by spMohanty and Tairu O, Emmanuel's kaggle datasets[27,28]. The RGB version of the corn(maize) plant is used out of 14 different crops in the dataset. Corn(maize) crop consists of 4 classes, consisting of 3 unhealthy classes and 1 healthy class. The dataset consists of 3852 image instances with labels such as cercospora gray leaf spot, common rust, northern leaf blight and healthy. Many preparation operations were carried out in order to translate photos into machine-readable language. These processes included data augmentation—adding diversity to the current data by making modest changes—and normalization, which entailed projecting image data pixels to a preset range.

3.2. Data Preprocessing

Data preprocessing basically consists of the steps that must be followed to encode the data so that it can be easily analyzed by a machine. In this proposed model, normalization, data augmentation and standardization is performed. A technique called grayscale conversion is also present, which can be used according to the requirements of the architecture.

3.2.1. Normalization and Standardization

Projecting picture data pixels to a preset range, typically (0,1) or (-1,1), is referred to as normalization as stated in Figure 1. Standardization is a technique for preprocessing and resizing photographs to achieve uniform height and breadth. Rescale the data to have a mean of 0 and a standard deviation of 1 (unit variance).



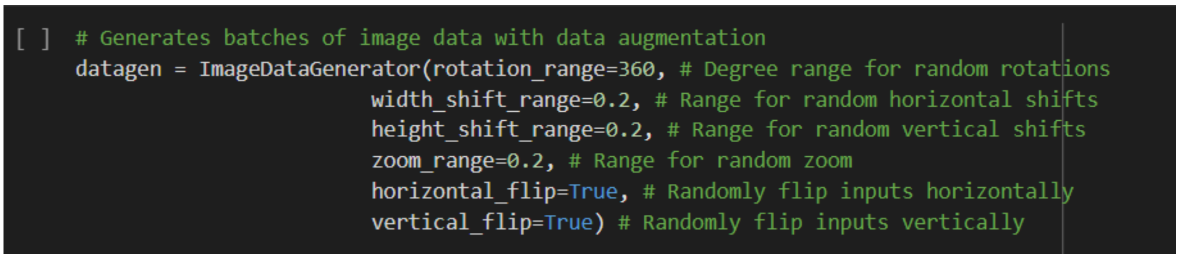
```
from tqdm import tqdm

X_train = np.zeros((data.shape[0], IMAGE_SIZE, IMAGE_SIZE, 3))
for i, file in tqdm(enumerate(data['File'].values)):
    image = read_image(file)
    if image is not None:
        X_train[i] = resize_image(image, (IMAGE_SIZE, IMAGE_SIZE))
# Normalize the data
X_Train = X_train / 255.0
print('Train Shape: {}'.format(X_Train.shape))
```

```
3852it [13:48,  4.65it/s]
Train Shape: (3852, 128, 128, 3)
```

Figure 1. Data Normalization

3.2.2. Data Augmentation

Data augmentation as mentioned in Figure 2 is the technique of adding small amounts of diversity to the existing data so that the model can handle it more quickly. Data may be enhanced using common techniques such as flipping it horizontally and vertically, rotating it, cropping it, shearing it, etc.



```
[ ] # Generates batches of image data with data augmentation
    datagen = ImageDataGenerator(rotation_range=360, # Degree range for random rotations
                                 width_shift_range=0.2, # Range for random horizontal shifts
                                 height_shift_range=0.2, # Range for random vertical shifts
                                 zoom_range=0.2, # Range for random zoom
                                 horizontal_flip=True, # Randomly flip inputs horizontally
                                 vertical_flip=True) # Randomly flip inputs vertically
```

Figure 2. Data Augmentation

3.3 Background and Preliminaries

The various deep learning models utilized for the suggested framework are described in this section. Before the final ensemble of the highest performing models, other models were tested. 4 models were ultimately chosen based on initial training accuracy metrics after 8 distinct models were trained on the training data set.

3.3.1 AlexNet [17]

Five convolutional layers and three fully connected layers make up the architecture of AlexNet as shown in Figure 3 and the tanh function is replaced by Rectified Linear Units

(ReLU). On the CIFAR-10 dataset, CNN with ReLU was 6 times quicker than CNN with Tanh and was able to reach a 25% error rate. By splitting the model's neurons over two of its GPUs, AlexNet enables multi-GPU training.

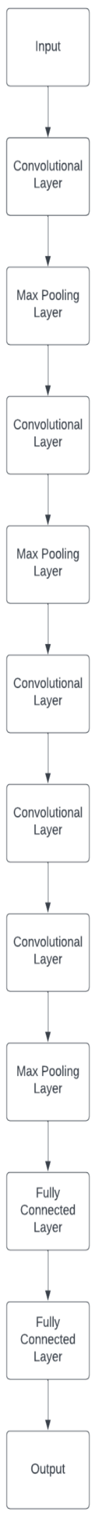


Figure 3. AlexNet Architecture

3.3.2 DenseNet121 [18]

DenseNet as shown in Figure 4 is a convolutional neural network architecture that has each layer directly connected to every other layer. DenseNet-121 has four DenseBlocks with 6, 12, 24, 16 layers respectively. DenseNet is specifically designed to ameliorate the accuracy loss caused by vanishing gradients in high-level neural networks. Simply said, because of the distance between the input and output layers, information disappears before it reaches its destination.

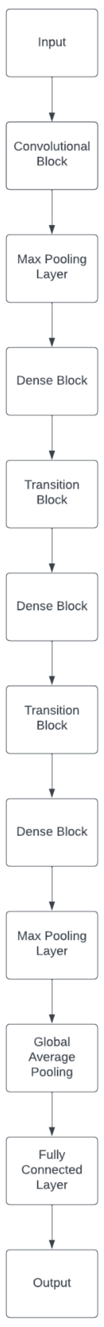


Figure 4. DenseNet121 Architecture

3.3.3 VGG16 [19]

VGG16 as shown in Figure 5 is a 16-layer deep convolutional neural network, i.e. there are 13 layers of convolution, 5 layers of max-pooling, and 3 layers of dense, for a total of 21 layers, but only 16 layers of weights. It is an object detection and classification algorithm that can classify 1000 images from 1000 different categories with 92.7% accuracy.

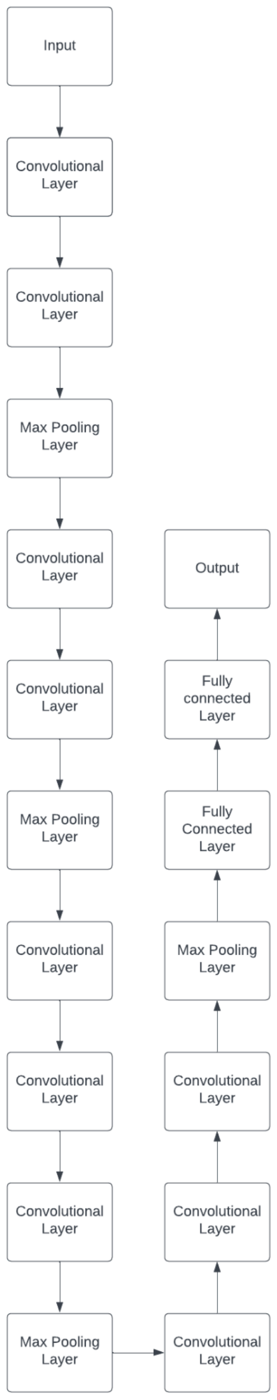


Figure 5. VGG16 Architecture

3.3.4 ResNet50 [20]

48 convolutional layers, one MaxPool layer, and one average pool layer makes up the architecture of ResNet50 as shown in Figure 6. A ResNet is a type of artificial neural network (ANN) that stacks residual blocks as mentioned in Figure 7 to form a network. It is based on bottleneck design, which reduces the high number of parameters and matrix multiplication.

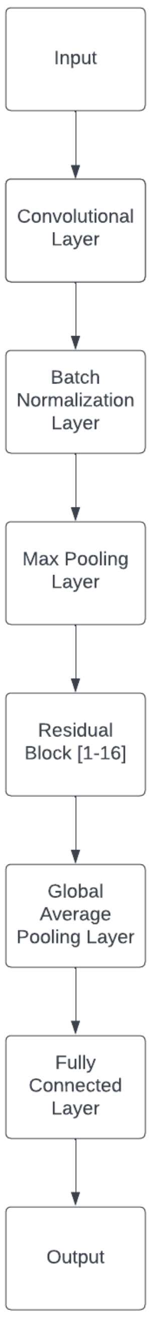


Figure 6. ResNet50 Architecture

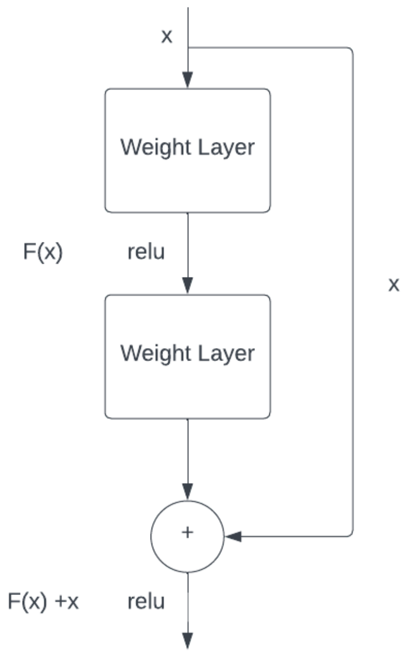


Figure 7. Residual Block

3.3.5 InceptionV3 [21]

InceptionV3 as shown in Figure 8 is an image recognition model proven to achieve over 78.1 accuracy on the ImageNet dataset. The model consists of symmetric and asymmetric building blocks such as convolutions, average pooling, max pooling, concatenation, dropout, and fully connected layers. Batch normalization is used extensively throughout the model and applied to the activation inputs whereas loss is calculated with softmax.

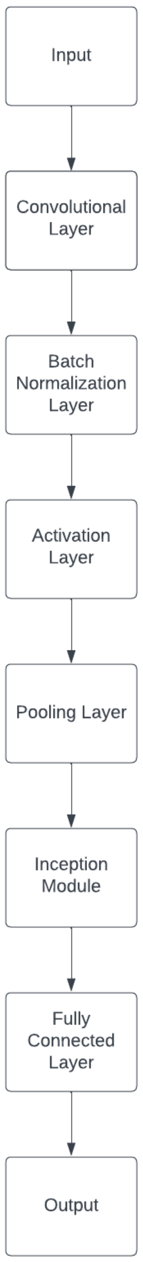


Figure 8. InceptionV3 Architecture

3.3.6 Xception [22]

Xception as shown in Figure 9 stands for Extreme version of Inception, consisting of 71 deep layers. It first applies a filter to each depth map and finally to depth to compress the input space with a 1X1 convolution. The architecture has 36 layers of convolutions that form the basis of the network's feature extraction.

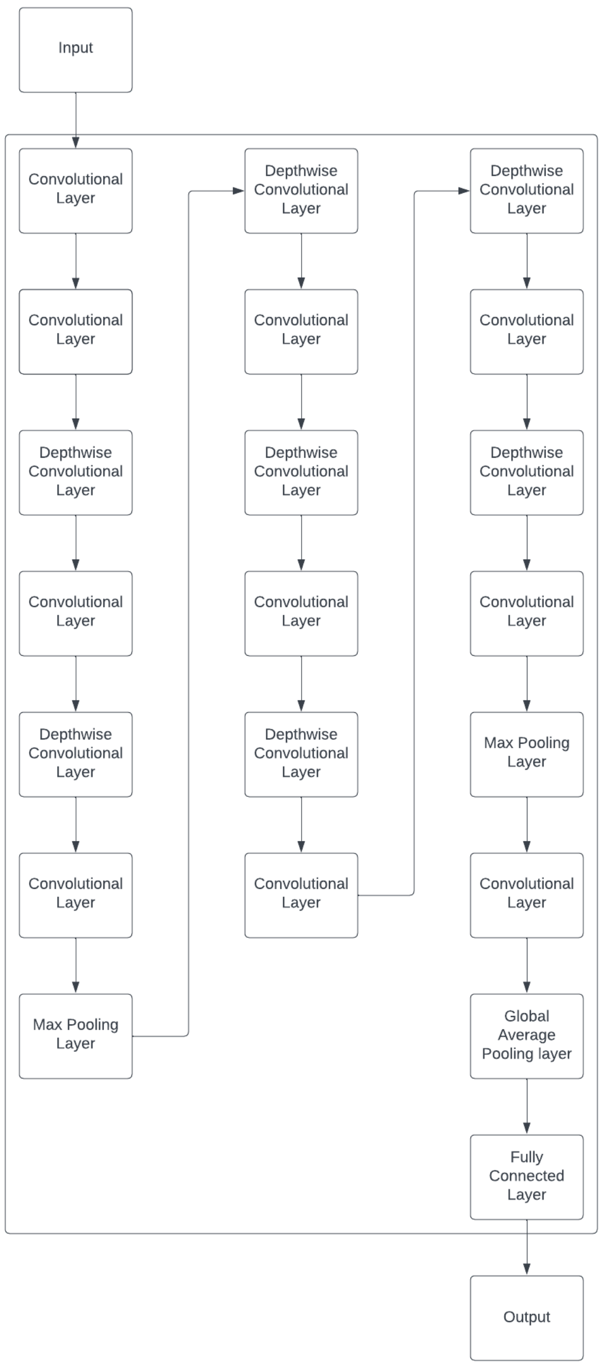


Figure 9. Xception Architecture consisting of Entry Flow, Middle Flow (8 Modules) and Exit Flow

3.3.7 MobileNet [23]

MobileNet consists of 28 layers as shown in Figure 10. The architecture is a network model that uses depthwise separable convolutions as the basic unit. Its depth separable fold has two layers: depthwise convolution and point convolution. As it is simple to implement on a mobile device, it is a cost-effective and efficient technique.

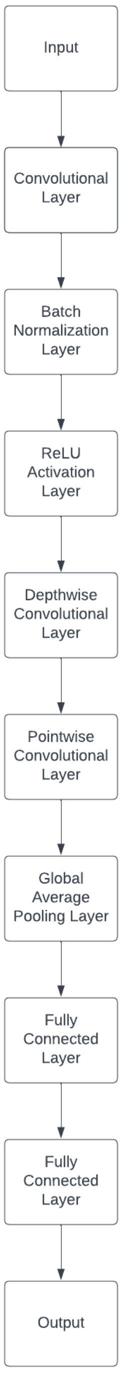

Input
Convolutional Layer
Batch Normalization Layer
ReLU Activation Layer
Depthwise Convolutional Layer
Pointwise Convolutional Layer
Global Average Pooling Layer
Fully Connected Layer
Fully Connected Layer
Output

Figure 10. MobileNet Architecture

3.3.8 CNN [24]

A type of deep learning model called CNN as shown in Figure 11 is used to analyze data in grid patterns, such as: Images inspired by the way the visual cortex of animals is organized, and created to automatically and adaptively learn the spatial hierarchy of features from lower-level to upper-level patterns. It requires little to no preprocessed data as compared to other deep learning algorithms.

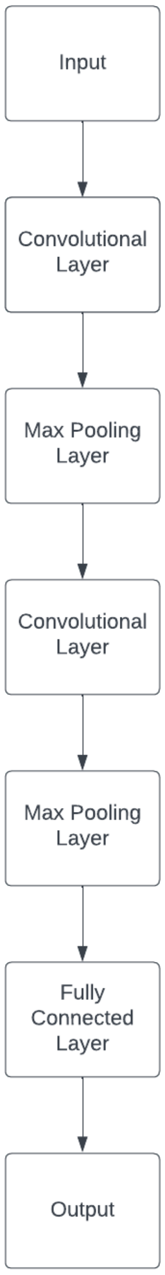


Figure 11. 5 Layer CNN Architecture

3.3.9. General Ensemble [25]

Ensemble models aggregate the results of various models to increase performance as a whole. A single model based on a single data sample can include biases, high variability, or plain mistakes that impair the validity of its analytical results in predictive modeling and other types of data analytics. The use of ensemble models, however, can lessen the consequences of such limits and improve the information available for making judgements. Bagging, stacking, and boosting are the three primary categories of ensemble learning techniques. Using numerous decision trees that have been fitted to different samples of the same dataset, bagging requires averaging the forecasts from each tree's predictions. Stacking is used to discover the best way to integrate the predictions when many unique architectures are fitted to the same data. Boosting involves sequentially adding ensemble members that correct the predictions supplied by prior models, creating a weighted average of the forecasts.

3.3.10. Stacking Ensemble [26]

Stacking as mentioned in Figure 12 is an ensemble technique that makes better predictions for the future by merging different weak learners with Meta learners by ensembleing them in parallel. The fundamental advantage of stacking ensembles is that they can protect a variety of effective model's abilities to address classification and regression issues. The creation of a superior model that makes predictions that surpass all other models is also beneficial.

Final model can be considered as being placed on top of the intermediate models after it is trained using the intermediate predictions. By employing such techniques, we may enhance our abilities and consistently produce models that are superior to the intermediate models.

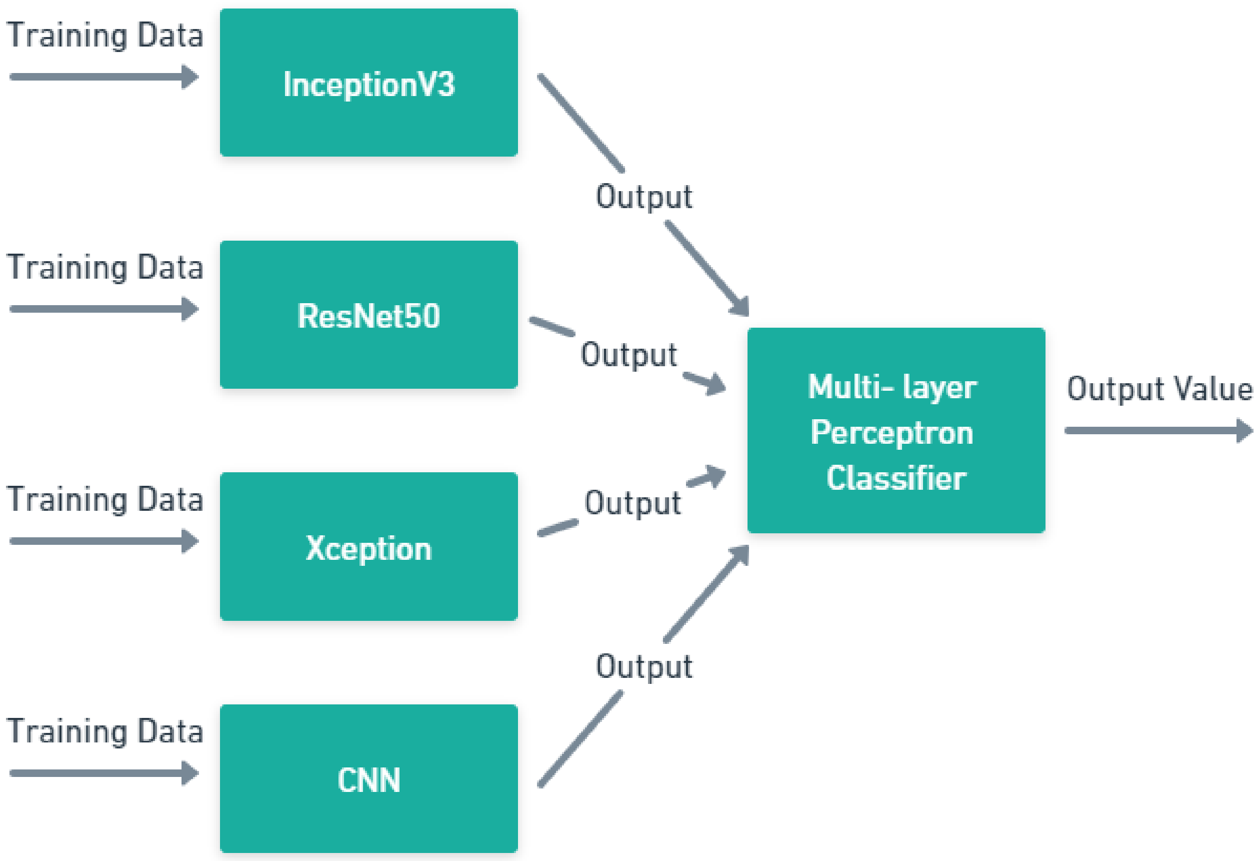


Figure 12. Proposed Stacking Ensemble Architecture

3.4. Proposed Ensemble Model

To decide how to integrate predictions from two or more underlying deep learning algorithms, stacking, often referred to as Super Learning in ensemble approaches, employs meta-learning algorithms. The benefit of stacking is that it enables the employment of various efficient models for classification or regression tasks, resulting in predictions that outperform those of the individual models in the ensemble. Algorithm 1 illustrates the stacking algorithm, while Table 2 defines the variables.

```
Input: I = ${(x_1, y_1), (x_2, y_2), (x_3, y_3), ... , (x_N, y_N)}$                //given Dataset
Output: 0,1,2,3                                   // Different plant disease and healthy class

Phase 1: Data Pre-processing
I' = S(I)                                         // S( ) function for    standardization
I'' = N(I')                                          // N( ) function for Normalization
I1 = DA(I'')                                      // DA( ) function for Data Augmentation

Phase 2: Training the Model
$D_t$ = D{t=1,2,3, … T}                              //set of Deep learning ensemble classifiers
W = Meta Learning Classifier
X = 75% dataset for training, X ∈ I1.
Y = 25% dataset for testing, Y ∈ I1.

For t = 1,2,3, … T:
$h_t = D(X_t)$;                                                //Training base learners
end.
$X'$ = Φ                                                              //A new Dataset
For l = 1,2,3, … L:
  For t = 1,2,3, … T:
    $z_{lt} = h_t(x_l)$;                                   //Classifying the training examples
end;
$X' = X' \cup \{((z_{lt}, z_{lt}, ... , z_{lt}), y_l)\}$;                                      //A new Dataset
end;
$h' = W(X')$;                                              //Train Meta-Learner Classifier

Phase 3: Testing the model

$H(x) = h'(h_1(x), h_2(x), ... , h_T(x),)$                        // Training of testing Dataset
Result = H classifies Y
```

Algorithm 1. Proposed Ensemble model for Plant Disease Prediction

Table 2. Symbols utilized in Algorithm 1

S.No.	**Symbols**	**Meaning**
1.	I	Existing Dataset
2.	D	Deep Learning Algorithms
3.	X	Training Set
4.	Y	Test Set
5.	W	Meta Classifier
6.	X'	New Training Dataset
7.	h	Trained Learners at level-0
8.	h'	Trained Learner at Level-1
9.	H	Stacking Model

Phase 1

In the proposed ensemble model, the original data is fed to numerous data preprocessing functions, consisting of standardization, normalization and data augmentation. These processes help the machine to easily analyze the image dataset.

Phase 2

Then the preprocessed data is first fed into a variety of models. The meta classifier is responsible for the estimation of both input and output of each model, as well as for the weights. All other models are eliminated, and only the best-performing ones are chosen. To increase accuracy and outcomes, this stacking ensemble technique is used.

As shown in Table 4, the traditional deep learning architectures that performed really well on the dataset are Xception, Inception V3, ResNet 50 and CNN. Therefore, all four deep learning architectures are considered as input to the ensemble model as level 0 models. Multi-layer Perceptron is used as a level 1 meta classifier in the proposed model so as to improve the results.

Phase 3

This phase revolves around testing the remaining 25% of the dataset with the proposed architecture and also comparing the results using different performance metrics.

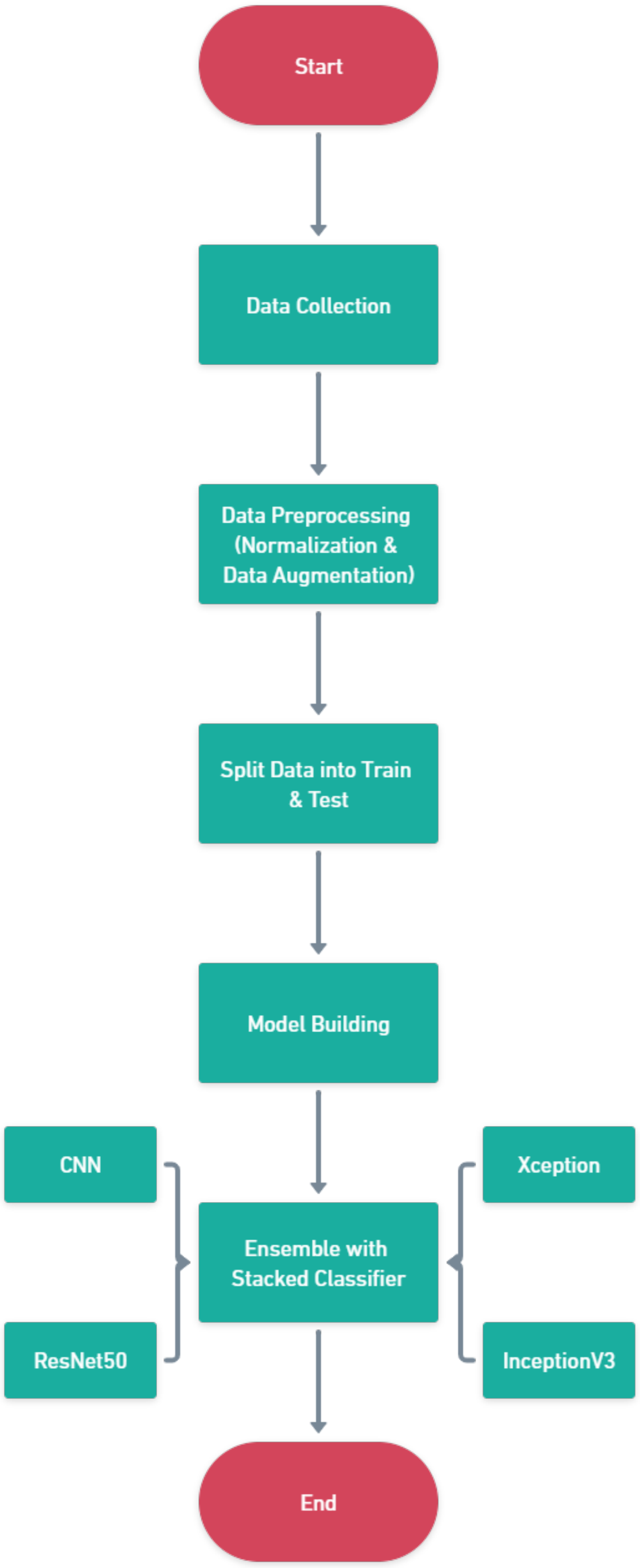

Start
Data Collection
Data Preprocessing (Normalization & Data Augmentation)
Split Data into Train & Test
Model Building
CNN
ResNet50
Ensemble with Stacked Classifier
Xception
InceptionV3
End

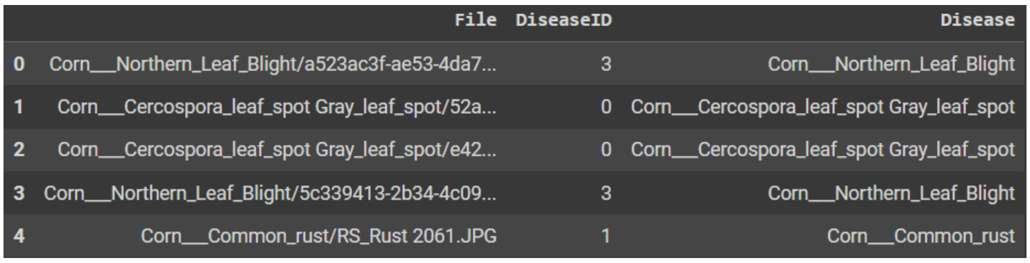
Figure 13. Flowchart of the proposed methodology

Figure 13 depicts our suggested approach, in which data from Kaggle and Github is thoroughly preprocessed before being used in the proposed model, which is trained on 75% of the total data using a variety of deep learning architectures. The top 4 architectures are used as the level 0 of the stacking architecture whereas multi-layer perceptron is at level 1. The different comparison metrics are used to perform a comparative analysis using the 25% of the remaining dataset.

CHAPTER-4
PERFORMANCE ANALYSIS

4.1 Data Visualization

This section consists of the visual description of the dataset in the form of graphs and images. The foremost step for data visualization was to convert the random images into a Dataframe for easier analysis as mentioned in Figure 14. Figure 15 shows the bar graph consisting of the frequency of different classes of the dataset. Table 3 represents the number of image instances in each class. Whereas Figure 16 represents 10 random samples of each class for visual identification of diseases. Figure 17 shows the random sample of images after image pre-processing is performed.



	File	DiseaseID	Disease
0	Corn___Northern_Leaf_Blight/a523ac3f-ae53-4da7...	3	Corn___Northern_Leaf_Blight
1	Corn___Cercospora_leaf_spot Gray_leaf_spot/52a...	0	Corn___Cercospora_leaf_spot Gray_leaf_spot
2	Corn___Cercospora_leaf_spot Gray_leaf_spot/e42...	0	Corn___Cercospora_leaf_spot Gray_leaf_spot
3	Corn___Northern_Leaf_Blight/5c339413-2b34-4c09...	3	Corn___Northern_Leaf_Blight
4	Corn___Common_rust/RS_Rust 2061.JPG	1	Corn___Common_rust

Figure 14. Images converted into a Dataframe

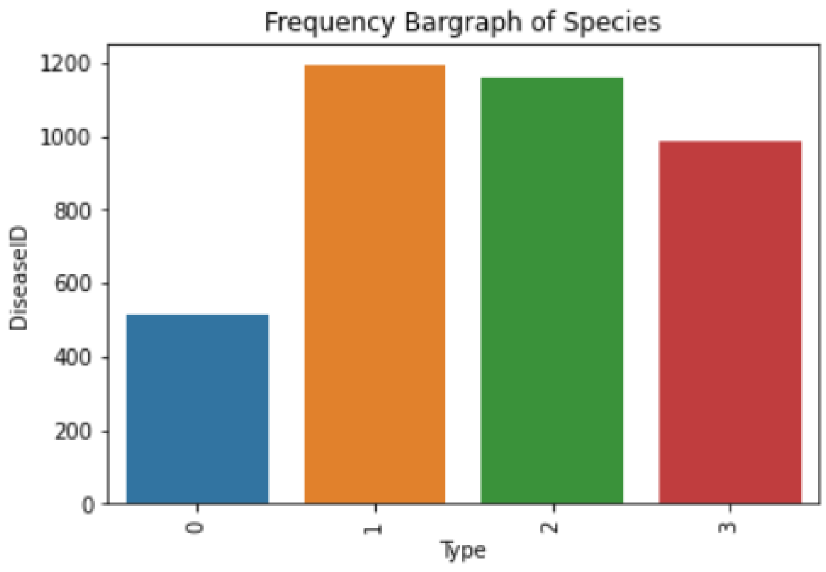


Figure 15. Frequency of Species

Table 3. Constituents of Dataset

Disease	Number of Images
Cercospora Gray Leaf Spot	513
Common Rust	1192
Northern Leaf Blight	985
Healthy	1162

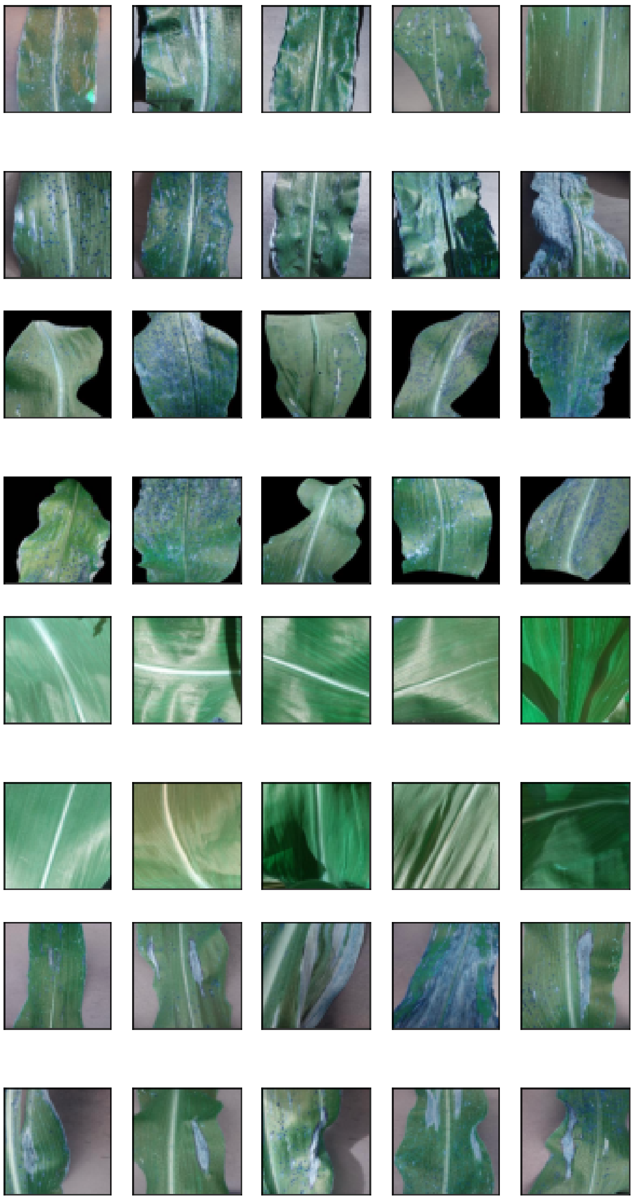


Figure 16. Random Sample of Diseased and Healthy Leaves

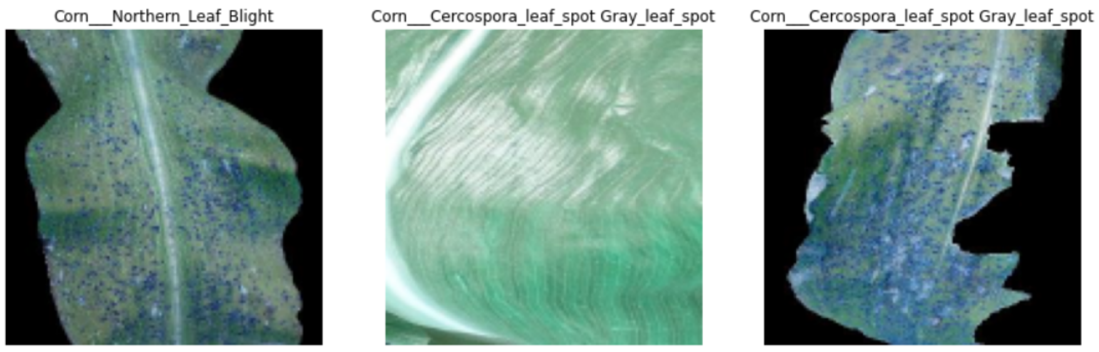

Corn__Northern_Leaf_Blight
Corn__Cercospora_leaf_spot Gray_leaf_spot
Corn__Cercospora_leaf_spot Gray_leaf_spot

Figure 17. Random Sample after Preprocessing

4.2. Model Selection

Choosing the best model may be difficult, especially when you have to take into account a number of different things including data quantity, complexity, and performance. A great deal of time was spent in developing multiple baseline architectures to ensure the selection of the optimal model. The baseline architectures included Xception, VGG16, MobileNet, DenseNet121, AlexNet, 5 layered CNN, ResNet50 and InceptionV3. Evaluation was made on the basis of accuracies for the architectures mentioned above and picked the architectures with highest accuracies and results are compiled in Table 4.

Table 4. Top 4 performing architectures being used as Level-0 learners

S.No.	Algorithms	Accuracy
1.	Xception	97.61%
2.	InceptionV3	97.51%
3.	ResNet50	95.33%
4.	CNN	94.49%
5.	AlexNet	93.21%
6.	DenseNet121	90.97%
7.	MobileNet	87.95%
8.	VGG16	84.53%

Stacking focuses on examining the space of several models that are used to address the same problem, which sets it apart from these ensemble strategies. It attempts to learn the optimal way to integrate input predictions to produce superior output predictions by taking into account the outputs of heterogeneous sub-models as input. Therefore, in this case Multi-Layered Perceptron Classifier is used as a Level 1 Meta Classifier.

4.3. Results and Discussion

This section contains the outcomes of our suggested approach. Various performance metrics are used to evaluate architecture performance based on the confusion matrix. The comparisons are made with pre-existing architectures as shown in Graph 4.
The proposed architecture overpowers the traditional deep learning techniques with an accuracy of 98.13%. The new architecture proved to be more efficient than others for predicting diseases in corn plants. The proposed model is a stacking ensemble model which uses really powerful techniques like CNN, Xception etc. to build the final results. The effective utilization of the hybrid model accounts for the proposed approach's superior performance.

4.4. Performance Metrics

- TP : True Positive: Predicted value correctly predicted as actual positive
- FP: Predicted values incorrectly predicted an actual positive. i.e., Negative values predicted as positive
- FN: False Negative: Positive values predicted as negative
- TN: True Negative: Predicted values correctly predicted as an actual negative

Accuracy

This is one of the most popular performance metrics for determining percentage of correct outcomes. Total number of correct outcomes by the total number of outcomes gives out to be accuracy.

$$Accuracy = \frac{Number\ of\ correct\ Predictions}{Total\ number\ of\ Predictions}$$

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

Precision

The number of true positives by the total number of positively predicted values is called precision.

$$Precision = \frac{TP}{TP + FP}$$

Recall

It is a feature of a classification model that identifies all data points in a particular class.

$$Recall = \frac{TP}{TP + FN}$$

Loss Function

It is a function that compares the target output value and the predicted output value. If the prediction is significantly wrong, the loss function will return a higher number whereas if they are fairly good, a lower number is returned.

Loss Function = MSE = $\frac{1}{n}\sum_{n}^{1} (Y_i - \hat{Y}_i)^2$

ROC Curve

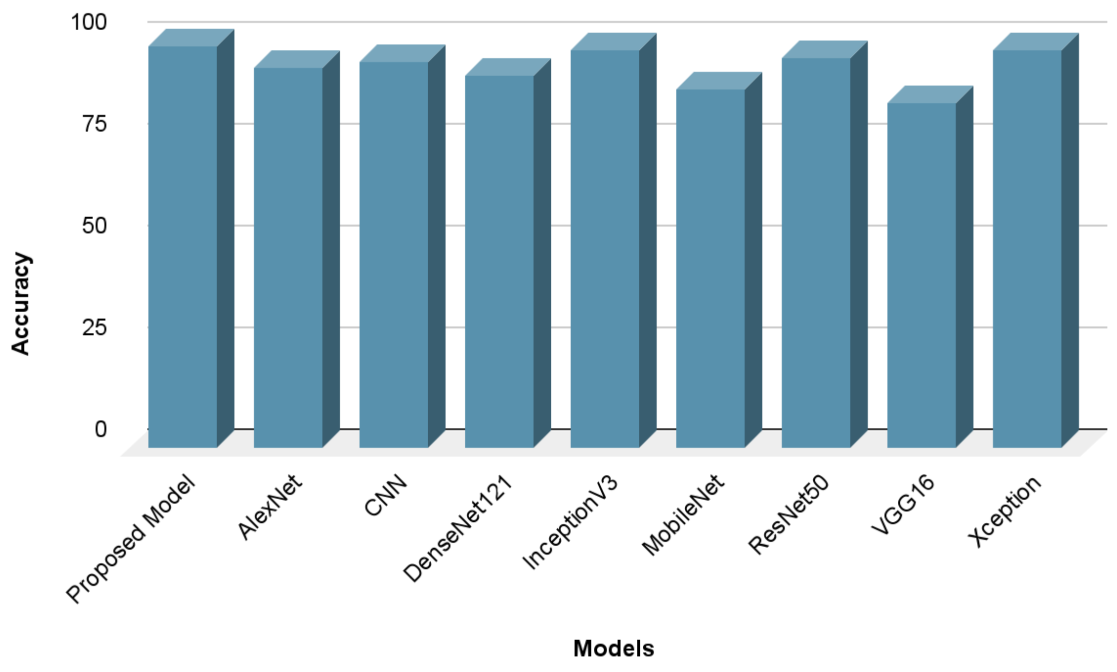
ROC curves are visual depictions of the relationship between a test's specificity and sensitivity. It is produced by plotting the fraction of true positives out of the total actual positives versus the fraction of false positives out of the total actual negatives.

$X - Axis\text{: } 1 - Specificity = \frac{FP}{FP + TN}$

$Y - Axis\text{: } Sensitivity = \frac{TP}{TP + FN}$

4.5. Comparison with traditional Deep Learning Architectures

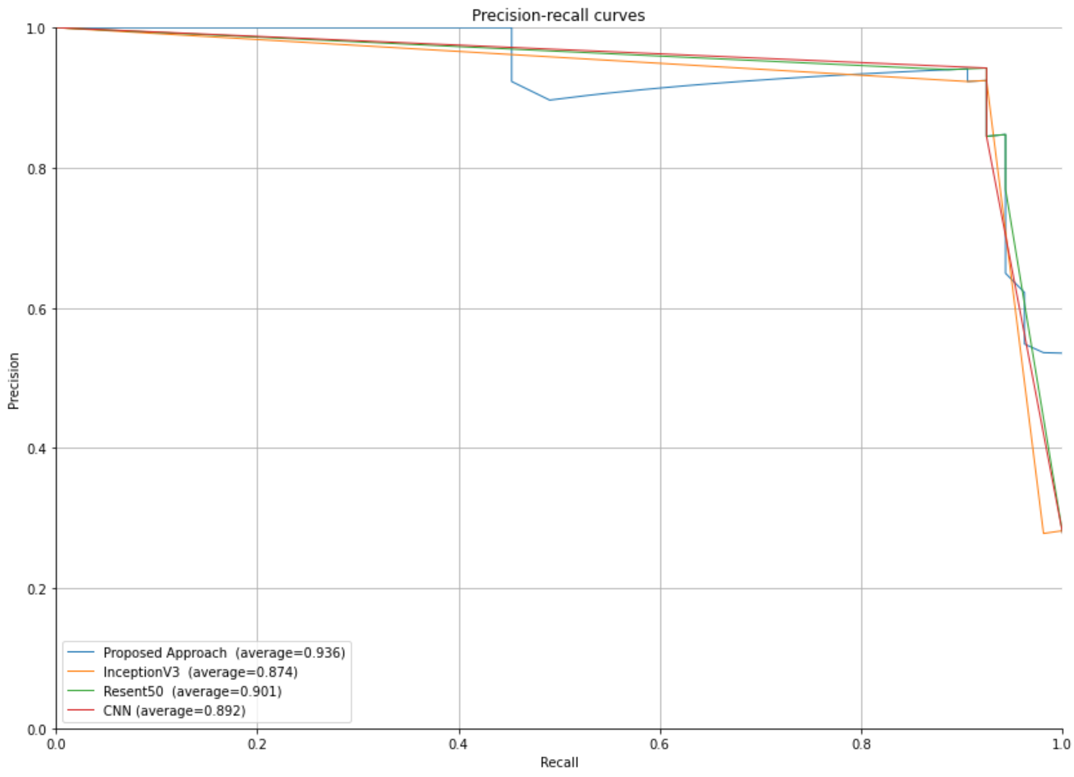
Various conventional deep learning architectures were implemented on the dataset before applying the proposed model. The deep learning model with the highest accuracy score was then selected for use in the ensemble approach. As shown in the Table 4, the architectures which gave the highest accuracies were CNN, Xception, ResNet50 and InceptionV3. These architectures were used as Level-0 models for the proposed approach whereas using multi-layered perceptron as the meta classifier. Graph 2 and 3 shows the precision-recall and ROC Curves of the proposed approach. Figure 19 shows the compilation of all the loss function graphs and accuracy graphs for training and testing dataset. Table 5 displays the trends for the proposed model and traditional architecture for Accuracy, Precision, Loss function, and ROC Curve.



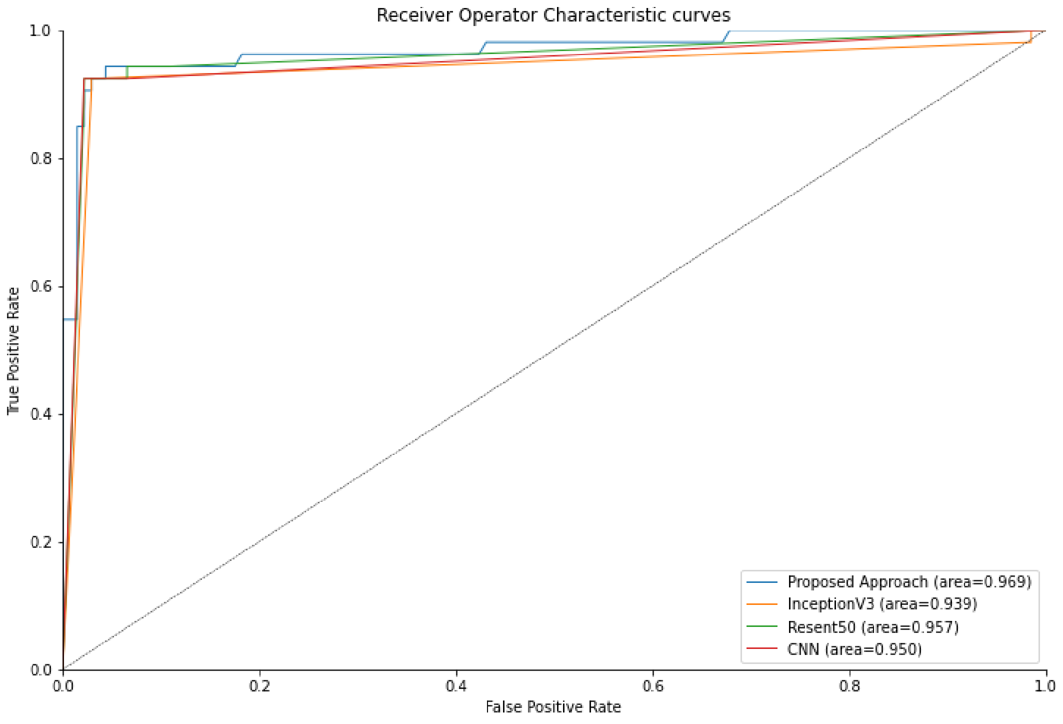
Graph 1. Accuracy based comparison of Proposed Ensemble and Existing Literature

Table 5. Comparison of Proposed Architecture and different Architectures on the basis of Different Metrics

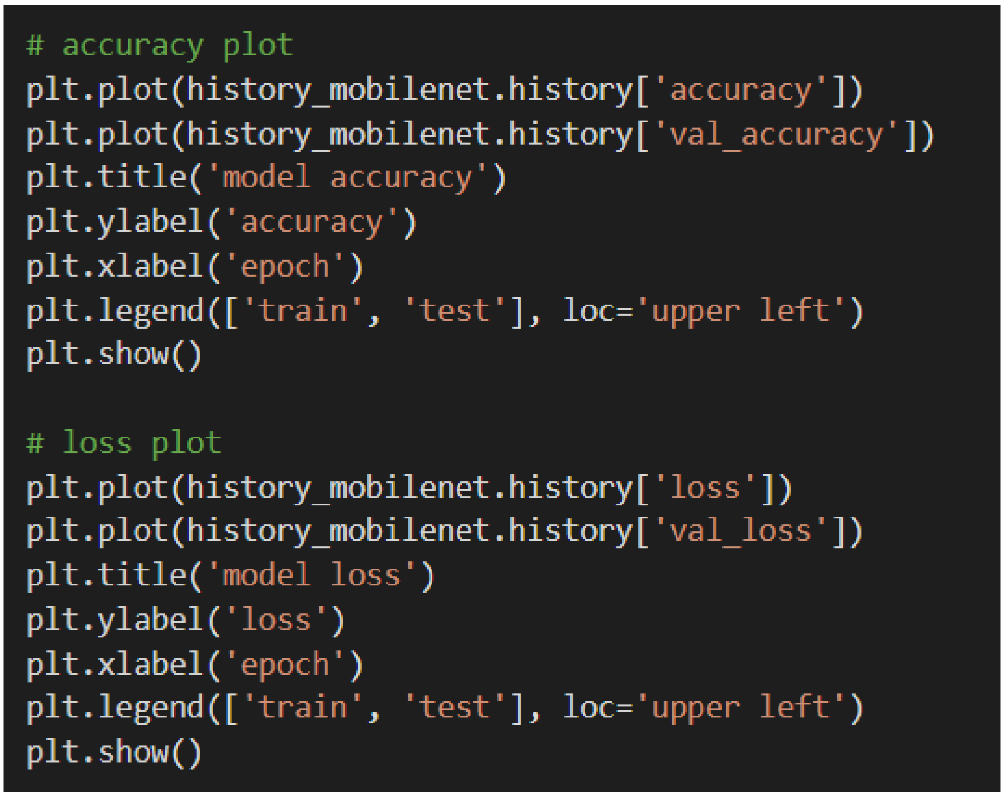
S.No.	Algorithm	Accuracy	Precision	Loss	ROC
1.	**Proposed Approach**	**98.13**	**93.62**	**3.26**	**96.91**
2.	AlexNet	93.21	91.54	9.23	92.82
3.	CNN	94.49	89.21	7.36	95.00
4.	DenseNet121	90.97	89.17	13.86	92.16
5.	InceptionV3	97.51	87.46	3.41	93.95
6.	MobileNet	87.95	88.73	18.47	91.76
7.	ResNet50	95.33	90.14	6.66	95.92
8.	VGG16	84.53	87.31	19.25	90.89
9.	Xception	97.61	86.98	4.07	91.43



Graph 2. Precision-Recall Curve

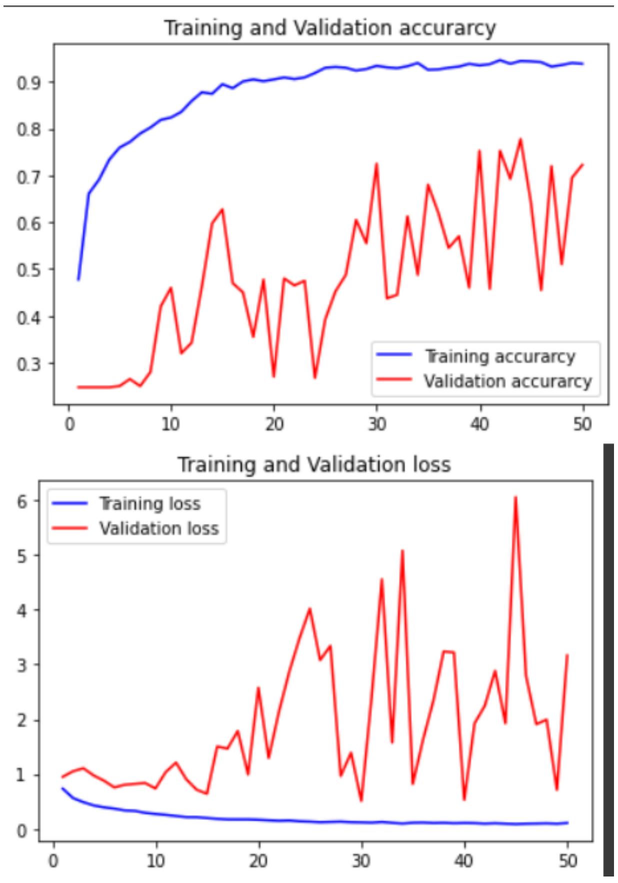


Graph 3. ROC Curve

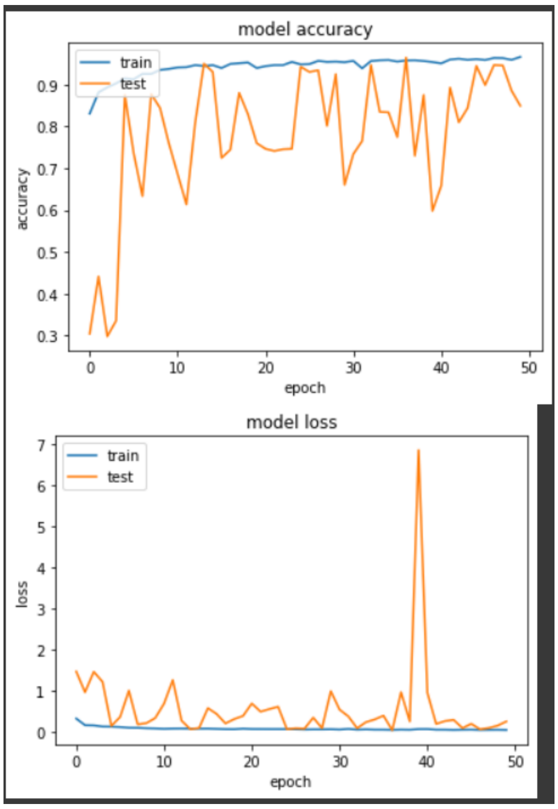


```
# accuracy plot
plt.plot(history_mobilenet.history['accuracy'])
plt.plot(history_mobilenet.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# loss plot
plt.plot(history_mobilenet.history['loss'])
plt.plot(history_mobilenet.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

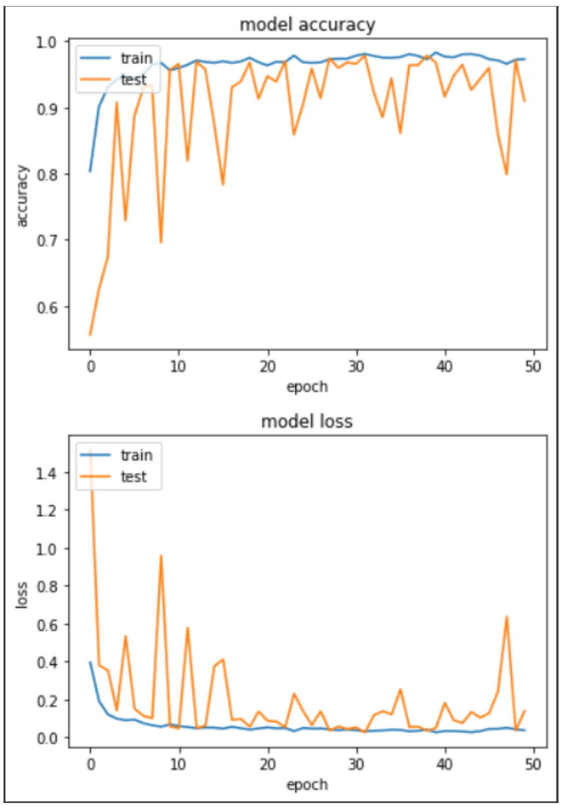
Figure 18. Code used to print Accuracy and Loss Graphs



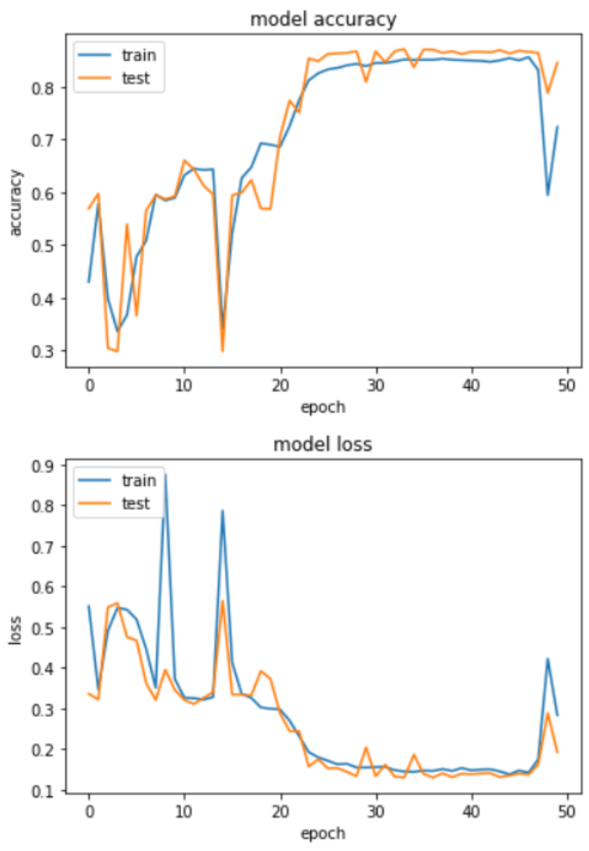
(a) AlexNet



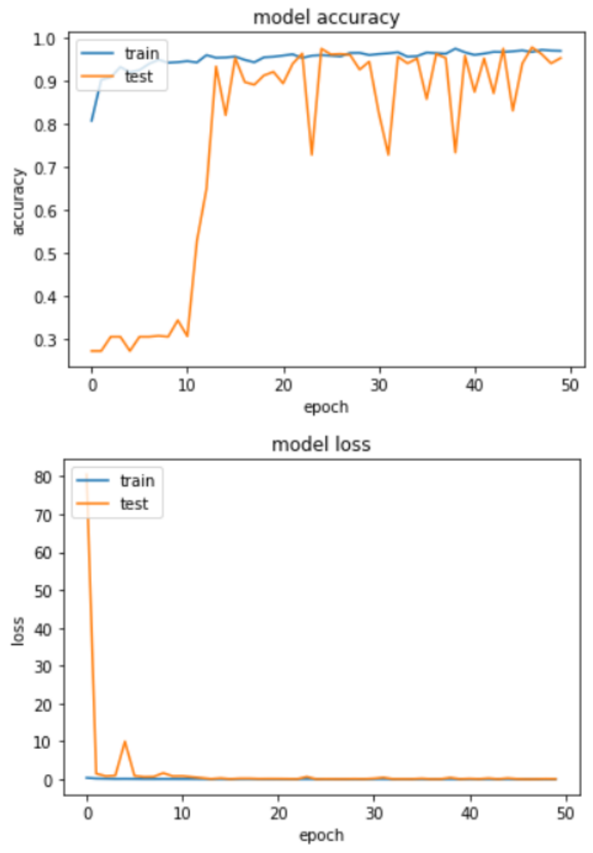
(b) 5 Layered CNN



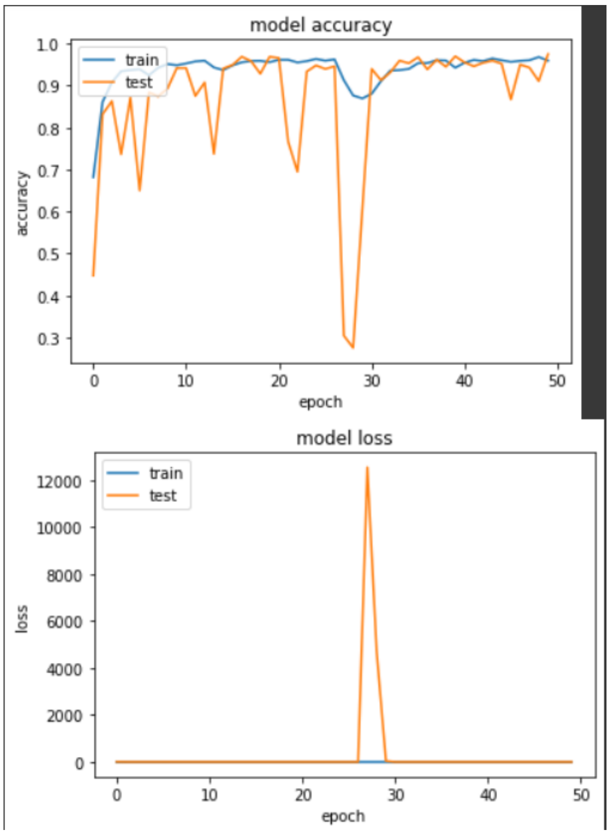
(c) DenseNet121



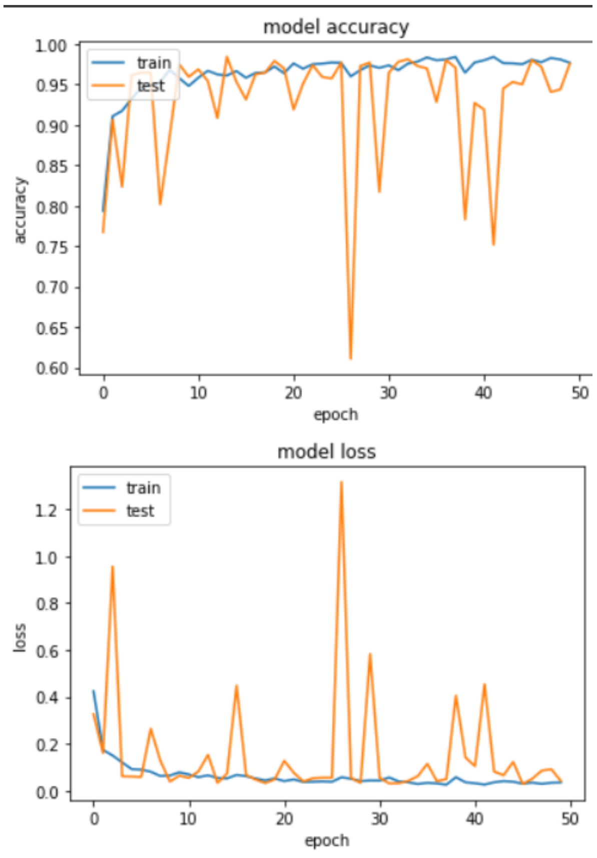
(d) VGG16



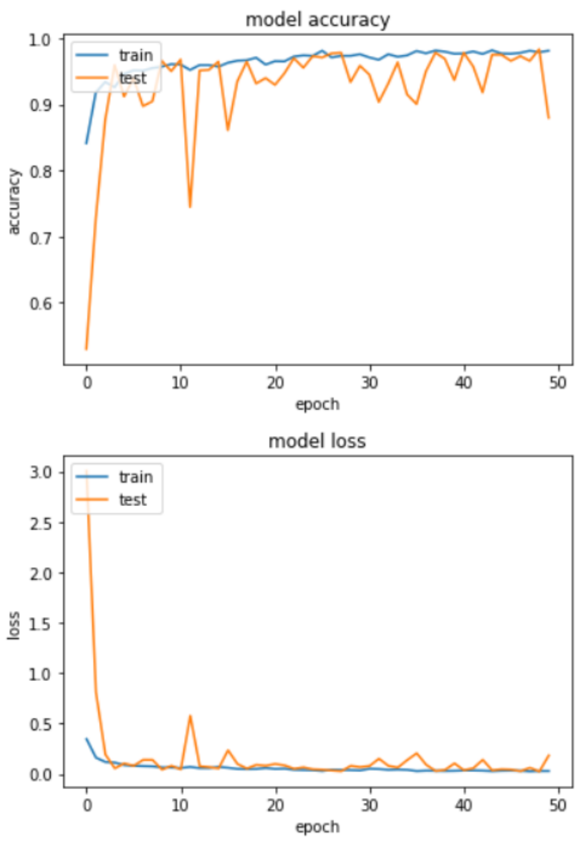
(e) ResNet50



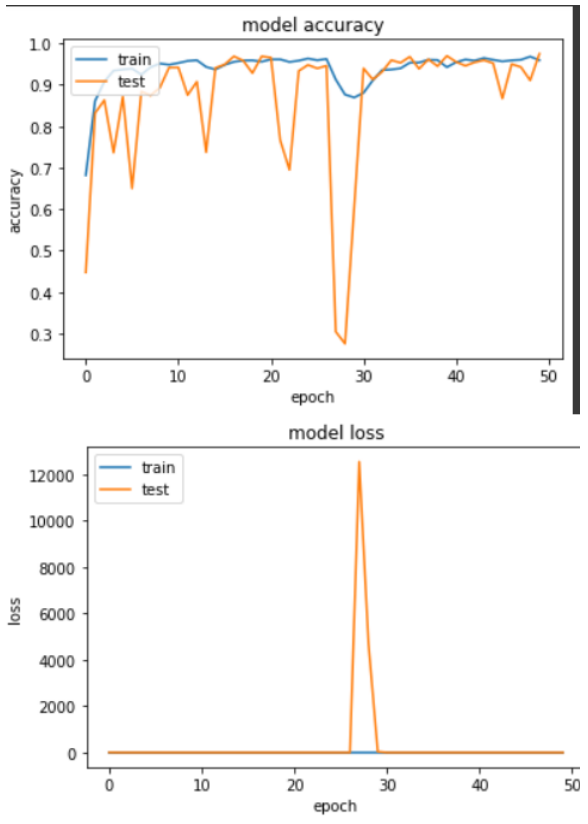
(f) InceptionV3



(g) Xception

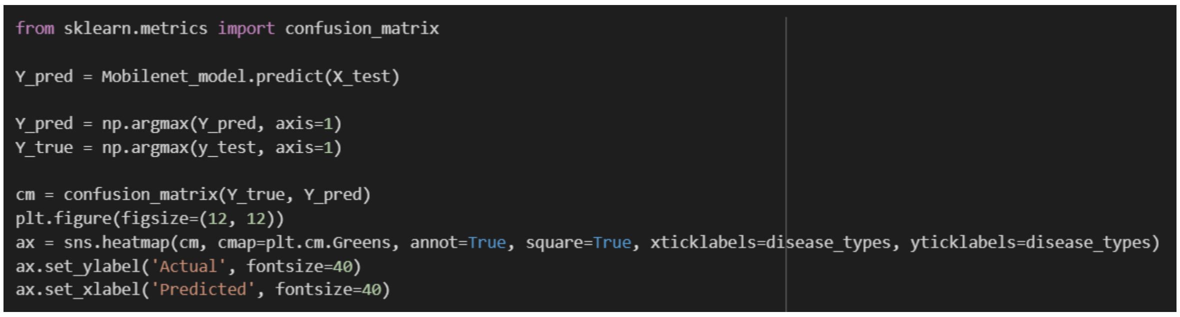


(h) MobileNet



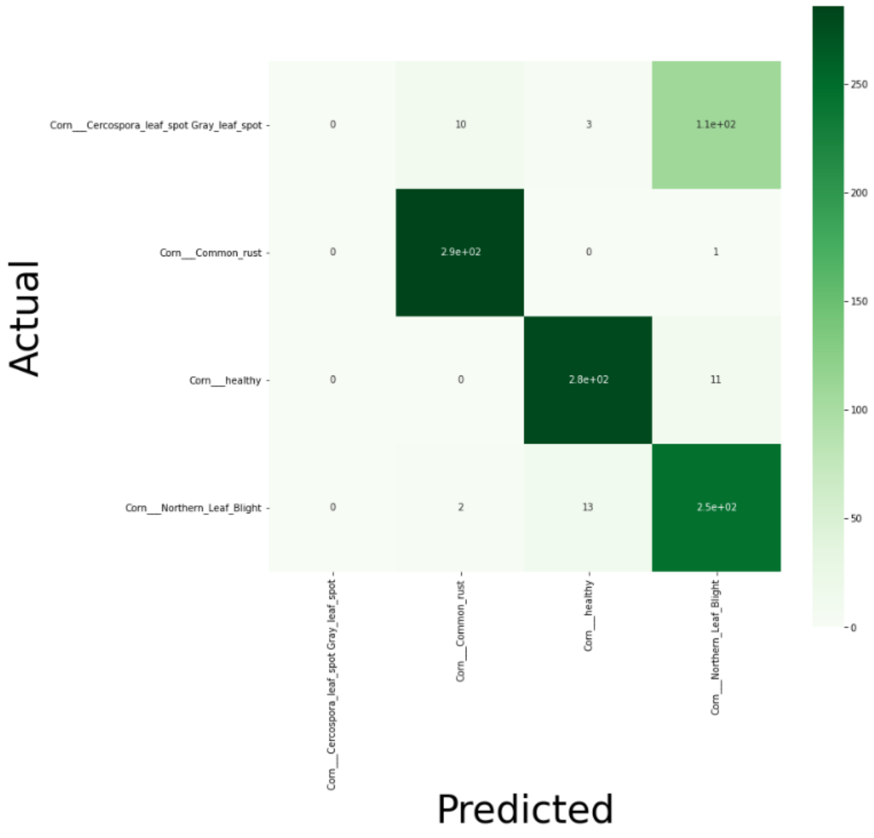
(i) Proposed Ensemble Model

Figure 19. Model Accuracy and Loss Graphs for each epoch

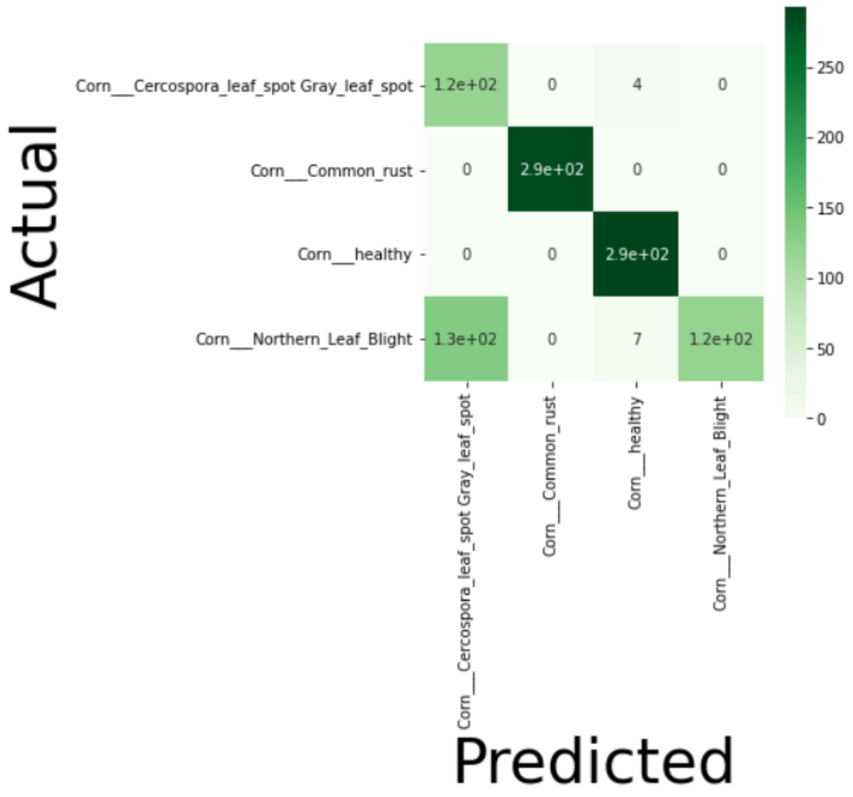


```
from sklearn.metrics import confusion_matrix

Y_pred = Mobilenet_model.predict(X_test)

Y_pred = np.argmax(Y_pred, axis=1)
Y_true = np.argmax(y_test, axis=1)

cm = confusion_matrix(Y_true, Y_pred)
plt.figure(figsize=(12, 12))
ax = sns.heatmap(cm, cmap=plt.cm.Greens, annot=True, square=True, xticklabels=disease_types, yticklabels=disease_types)
ax.set_ylabel('Actual', fontsize=40)
ax.set_xlabel('Predicted', fontsize=40)
```

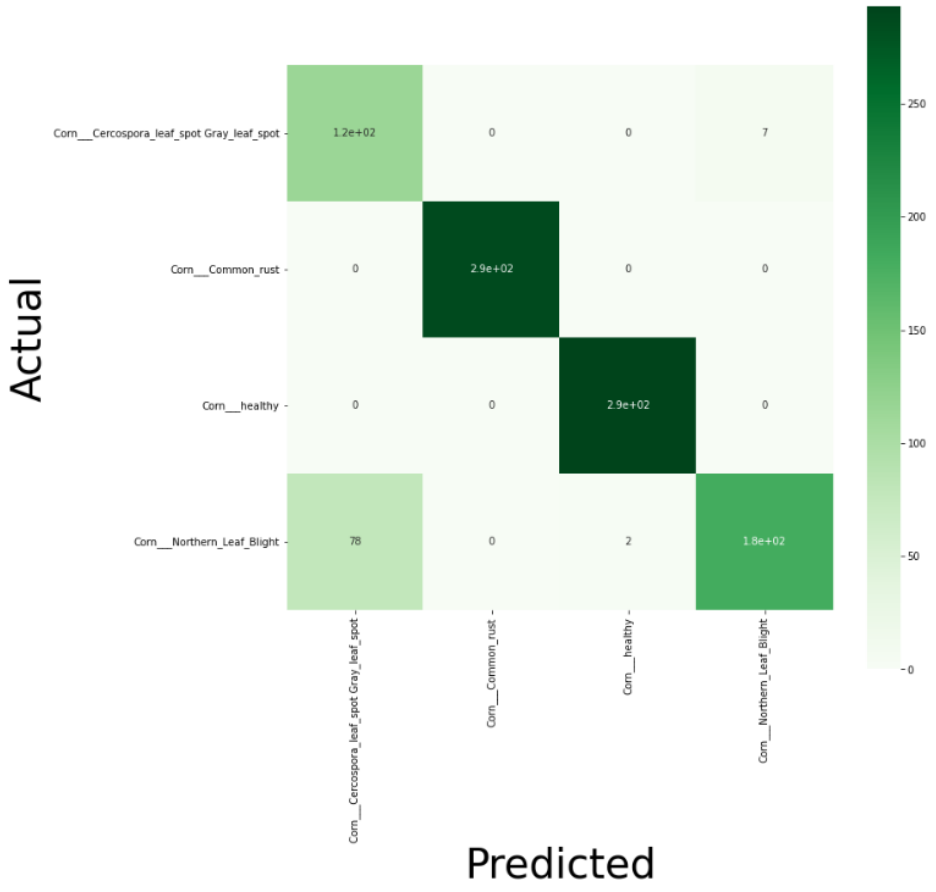
Figure 20. Code used to print Confusion Matrix



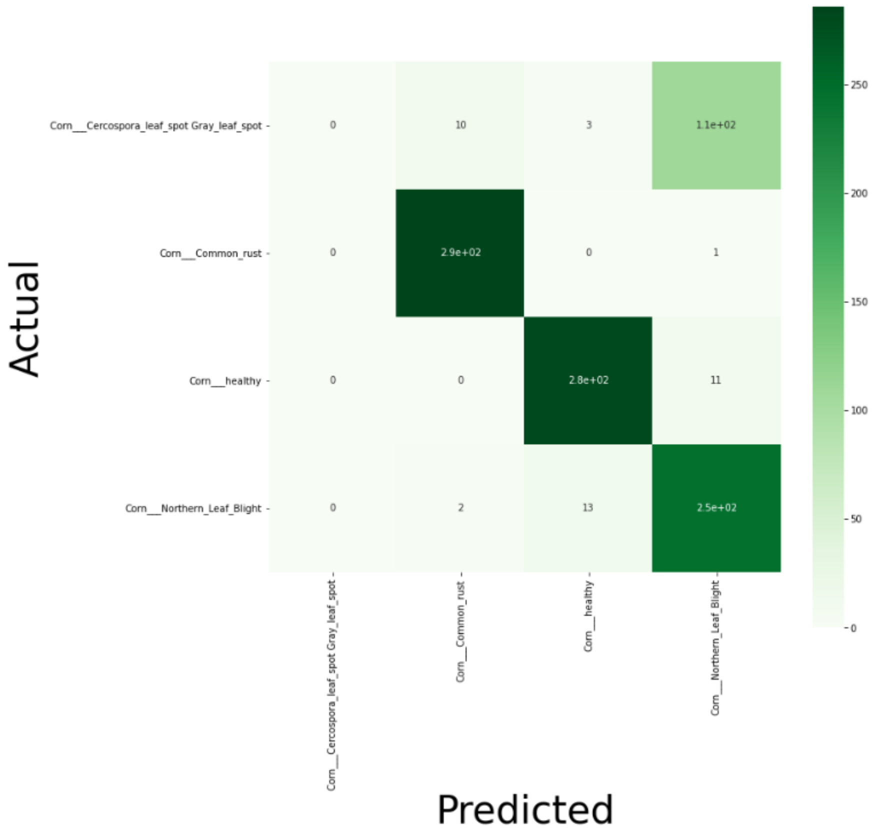
(a) AlexNet



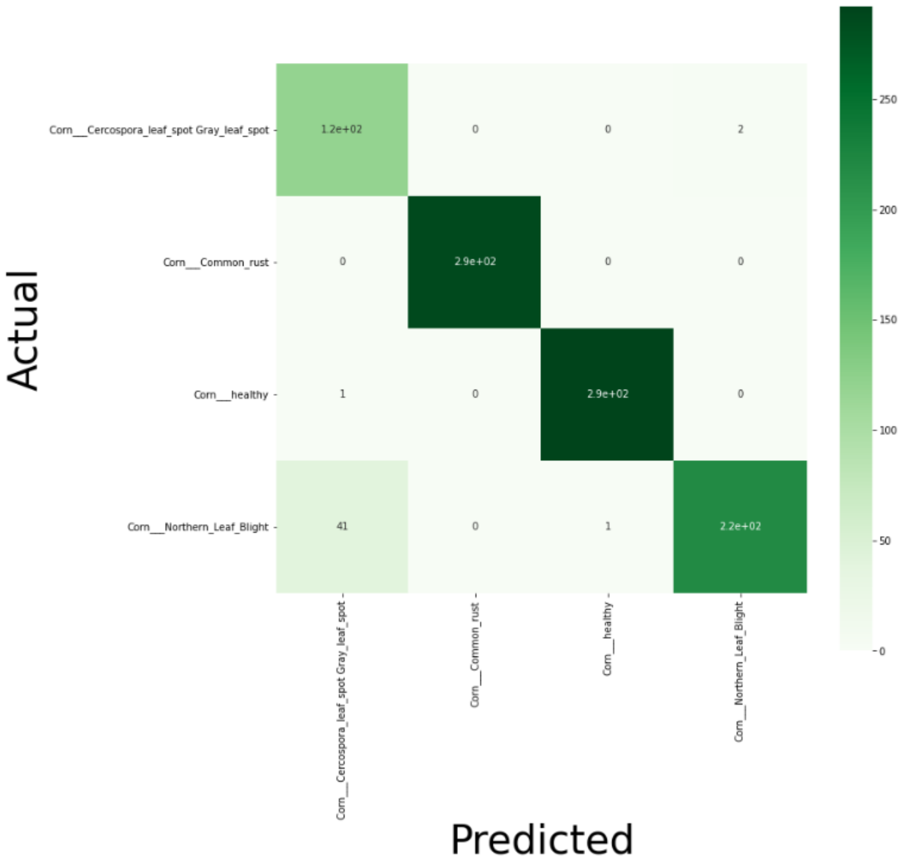
(b) 5 Layered CNN



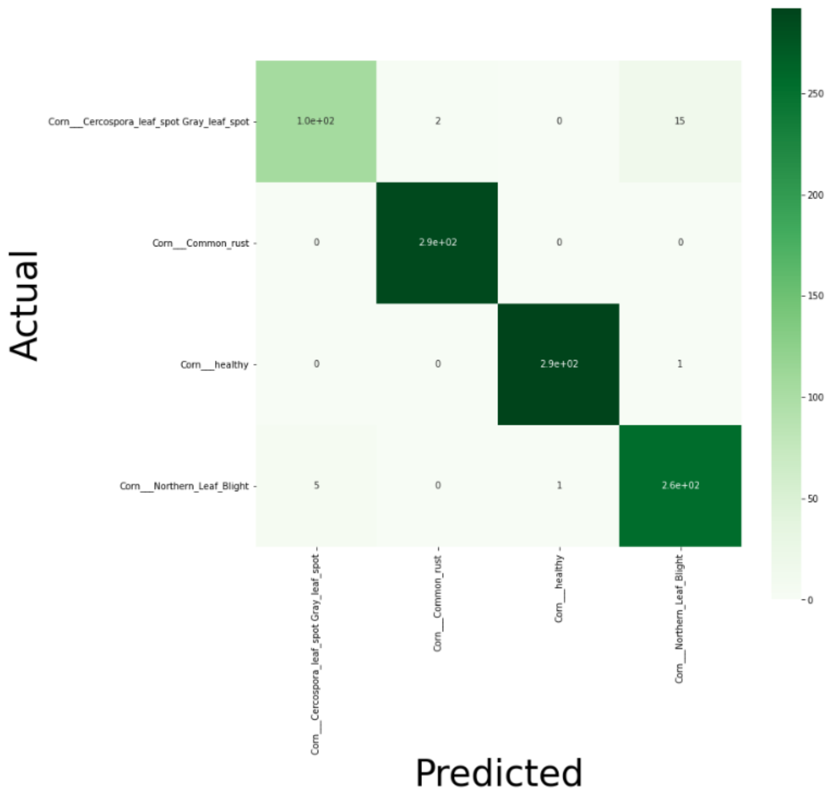
(c) DenseNet121



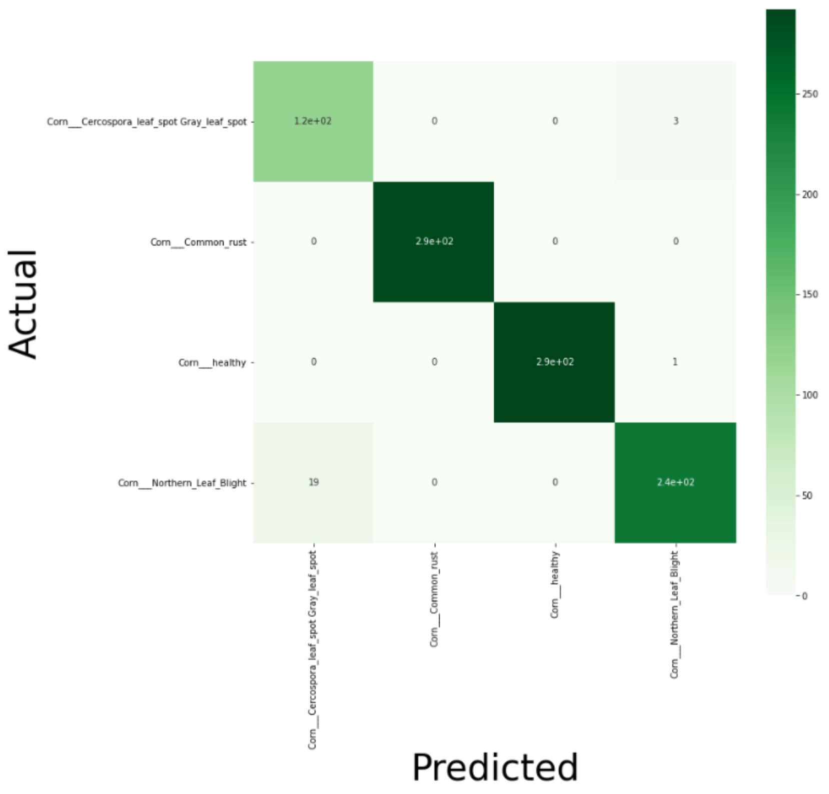
(d) VGG16



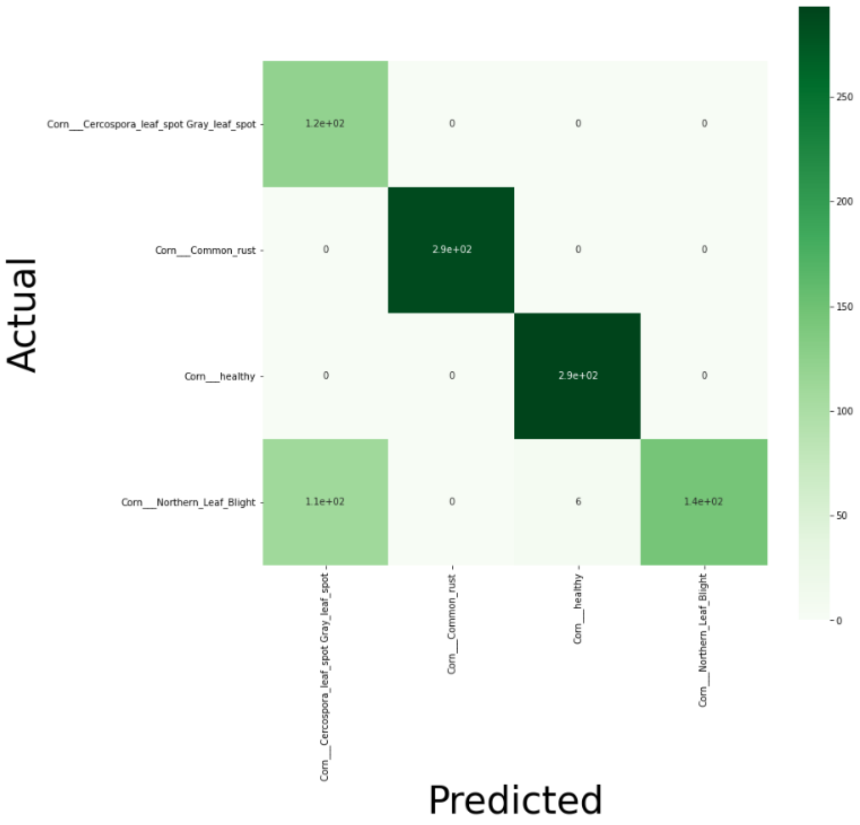
(e) ResNet50



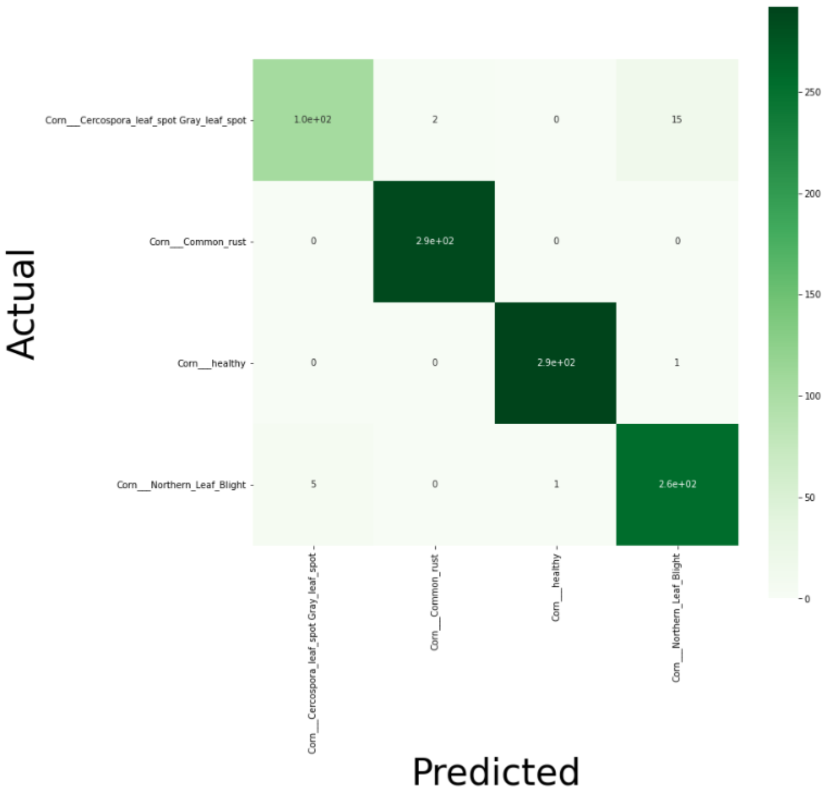
(f) InceptionV3



(g) Xception

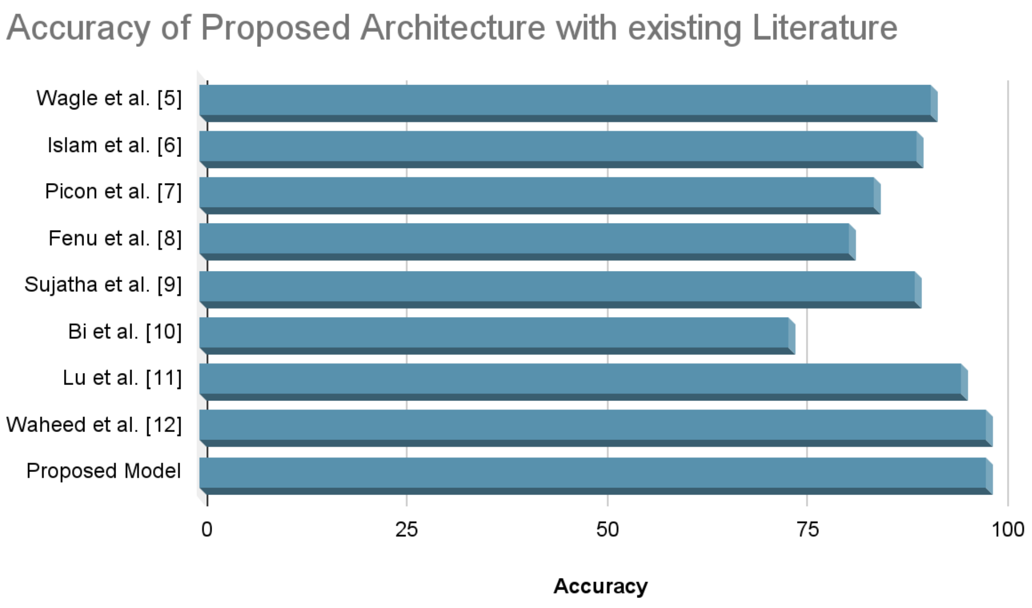


(h) MobileNet



(i) Proposed Ensemble Model

Figure 21. Confusion Matrix for Respective Models



Graph 4. Comparison of Accuracy with Literature Survey

CHAPTER-5
CONCLUSION

5.1 Conclusion

Plant disease prediction is an important activity to ensure protection of crops from devastation as plant diseases and pests have a significant impact on food security and the environment. The proposed framework can contribute as a preventive measure to classify the presence of disease in a plant leaf sample. The model was trained on healthy leaf samples and four crop(maize) plant diseases. As an outcome, the proposed model obtained a maximum accuracy of 98.13%. The proposed model is better than conventional models because it is based on a stack-based ensemble approach using the four best-performing conventional models as base models. It also performs significantly better than other state-of-art literature, mentioned in the results portion of the report.

5.2 Future Scope

- In future we would like to explore this domain even further, expanding to other crops and applying transformers to improve the results even further.
- A mobile application could also be developed to maximize the reach at minimal cost. This would involve the selection of crops in which the individual wants to examine the disease and then uploading the images of the crop.
- By gathering data from surrounding government fields and sectors and creating our own dataset, the model can be enhanced, leading to a more optimized model.
- K Fold Validation technique can be used in addition to the blending technique when ensembling, which might produce better results.

5.3 Applications Contribution

It is suggested to use an ensemble learning strategy based on stacking to boost classifier diversity. It employs a meta-learning technique to determine the most efficient approach to aggregate the predictions from one or more underlying deep learning architectures. Eight architectures were chosen as the basic models, and the meta model was then supplied with them. In comparison to all other current state-of-the-art architectures, the final model outperformed them all with an accuracy of 98.13%. Xception, Inception V3, ResNet 50 and 5 layer CNN architectures are stacked to form the ensemble model. The stacked ensemble model is compared with existing Deep Learning Architectures on the basis of ROC curve, Precision, Accuracy, Loss Function and also compared with the literature discussed earlier.

References

[1] Too, E. C., Yujian, L., Njuki, S., & Yingchun, L. (2019). A comparative study of fine-tuning deep learning models for plant disease identification. *Computers and Electronics in Agriculture*, *161*, 272–279. https://doi.org/10.1016/j.compag.2018.03.032

[2] Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, *7*(September). https://doi.org/10.3389/fpls.2016.01419

[3] Jayakumar, D., Elakkiya, A., Rajmohan, R., & Ramkumar, M. O. (2020, July 3). Automatic Prediction and Classification of Diseases in Melons using Stacked RNN based Deep Learning Model. *2020 International Conference on System, Computation, Automation and Networking, ICSCAN 2020*. https://doi.org/10.1109/ICSCAN49426.2020.9262414

[4] Turkoglu, M., Hanbay, D., & Sengur, A. (2022). Multi-model LSTM-based convolutional neural networks for detection of apple diseases and pests. *Journal of Ambient Intelligence and Humanized Computing*, *13*(7), 3335–3345. https://doi.org/10.1007/s12652-019-01591-w

[5] Wagle, S. A., & Harikrishnan, R. (2021). Comparison of plant leaf classification using modified alexnet and support vector machine. *Traitement Du Signal*, *38*(1), 79–87. https://doi.org/10.18280/TS.380108

[6] Islam, M. A., Rahman Shuvo, N., Shamsojjaman, M., Hasan, S., Hossain, S., & Khatun, T. (2021). An Automated Convolutional Neural Network Based Approach for Paddy Leaf Disease Detection. *International Journal of Advanced Computer Science and Applications*, *12*(1), 280–288. https://doi.org/10.14569/IJACSA.2021.0120134

[7] Picon, A., Alvarez-Gila, A., Seitz, M., Ortiz-Barredo, A., Echazarra, J., & Johannes, A. (2019). Deep convolutional neural networks for mobile capture device-based crop disease classification in the wild. *Computers and Electronics in Agriculture*, *161*, 280–290. https://doi.org/10.1016/j.compag.2018.04.002

[8] Fenu, G., & Malloci, F. M. (2021). Using Multioutput Learning to Diagnose Plant Disease and Stress Severity. *Complexity*, *2021*. https://doi.org/10.1155/2021/6663442

[9] Sujatha, R., Chatterjee, J. M., Jhanjhi, N. Z., & Brohi, S. N. (2021). Performance of deep learning vs machine learning in plant leaf disease detection. *Microprocessors and Microsystems*, *80*. https://doi.org/10.1016/j.micpro.2020.103615

[10] Bi, C., Wang, J., Duan, Y., Fu, B., Kang, J. R., & Shi, Y. (2022). MobileNet Based Apple Leaf Diseases Identification. *Mobile Networks and Applications*, *27*(1), 172–180. https://doi.org/10.1007/s11036-020-01640-1

[11] Lu, Y., Yi, S., Zeng, N., Liu, Y., & Zhang, Y. (2017). Identification of rice diseases using deep convolutional neural networks. *Neurocomputing*, *267*, 378–384. https://doi.org/10.1016/j.neucom.2017.06.02

[12] Waheed, A., Goyal, M., Gupta, D., Khanna, A., Hassanien, A. E., & Pandey, H. M. (2020). An optimized dense convolutional neural network model for disease recognition and classification in corn leaf. *Computers and Electronics in Agriculture*, *175*. https://doi.org/10.1016/j.compag.2020.105456

[13] Jadhav, S. B., Udupi, V. R., & Patil, S. B. (2021). Identification of plant diseases using convolutional neural networks. *International Journal of Information Technology (Singapore)*, *13*(6), 2461–2470. https://doi.org/10.1007/s41870-020-00437-5

[14] Shrestha, G., Deepsikha, Das, M., & Dey, N. (2020). Plant Disease Detection Using CNN. *Proceedings of 2020 IEEE Applied Signal Processing Conference, ASPCON 2020*, 109–113. https://doi.org/10.1109/ASPCON49795.2020.9276722

[15] Rangarajan, A. K., Purushothaman, R., & Ramesh, A. (2018). Tomato crop disease classification using pre-trained deep learning algorithm. *Procedia Computer Science*, *133*, 1040–1047. https://doi.org/10.1016/j.procs.2018.07.070

[16] Patil, R. R., Kumar, S., & Rani, R. (2022). Comparison of Artificial Intelligence Algorithms in Plant Disease Prediction. *Revue d'Intelligence Artificielle*, *36*(2), 185–193. https://doi.org/10.18280/ria.360202

[17] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet Classification with Deep Convolutional Neural Networks*. http://code.google.com/p/cuda-convnet/

[18] Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2016). *Densely Connected Convolutional Networks*. http://arxiv.org/abs/1608.06993

[19] Simonyan, K., & Zisserman, A. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. http://arxiv.org/abs/1409.1556

[20] He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition*. http://arxiv.org/abs/1512.03385

[21] Szegedy, C., Vanhoucke, V., Ioffe, S., & Shlens, J. (2015). *Rethinking the Inception Architecture for Computer Vision*.

[22] Chollet, F. (2016). *Xception: Deep Learning with Depthwise Separable Convolutions*. http://arxiv.org/abs/1610.02357

[23] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. http://arxiv.org/abs/1704.04861

[24] Ballard, D., Lecun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). *Backpropagation Applied to Handwritten Zip Code Recognition*.

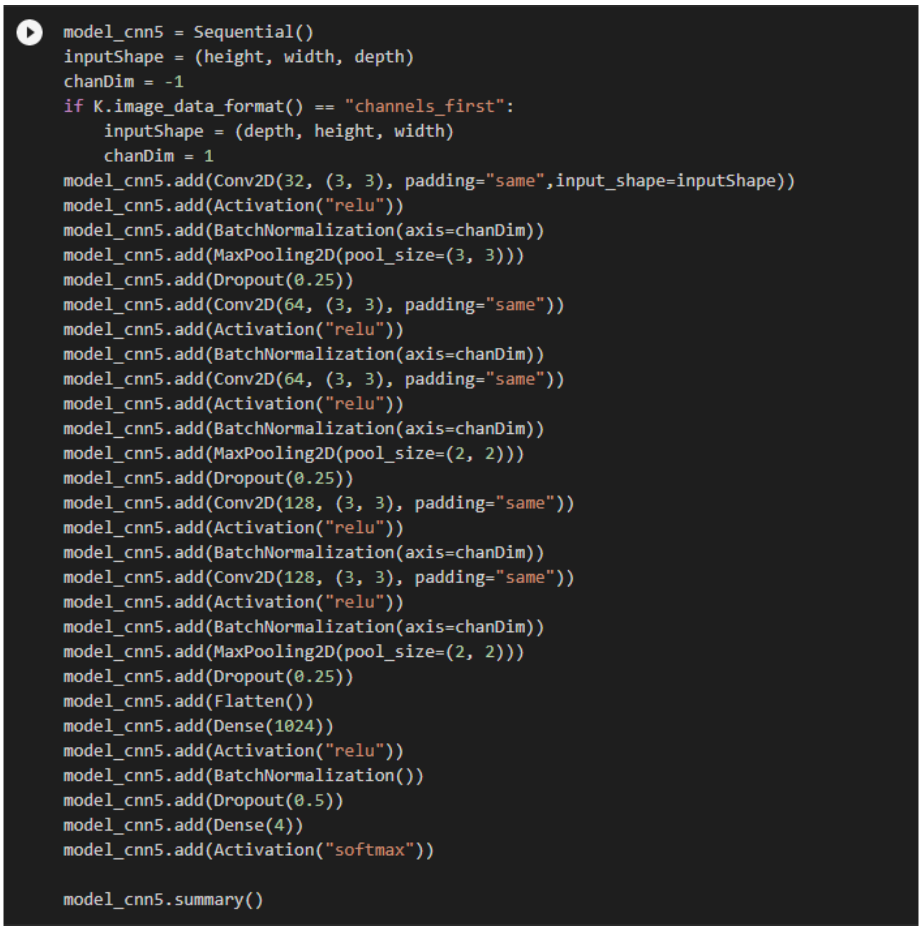
[25] Opitz, D., & Maclin, R. (1999). Popular Ensemble Methods: An Empirical Study. In *Journal of Artificial Intelligence Research* (Vol. 11).

[26] Ian H. W., Eibe F. and Mark A. H., "*Data Mining: practical machine learning tools and techniques*". 3rd ed., Morgan Kaufmann, 2011.

[27] spMohanty, "PlantVillage Dataset" (2016). Distributed by Github. https://github.com/spMohanty/PlantVillage-Dataset

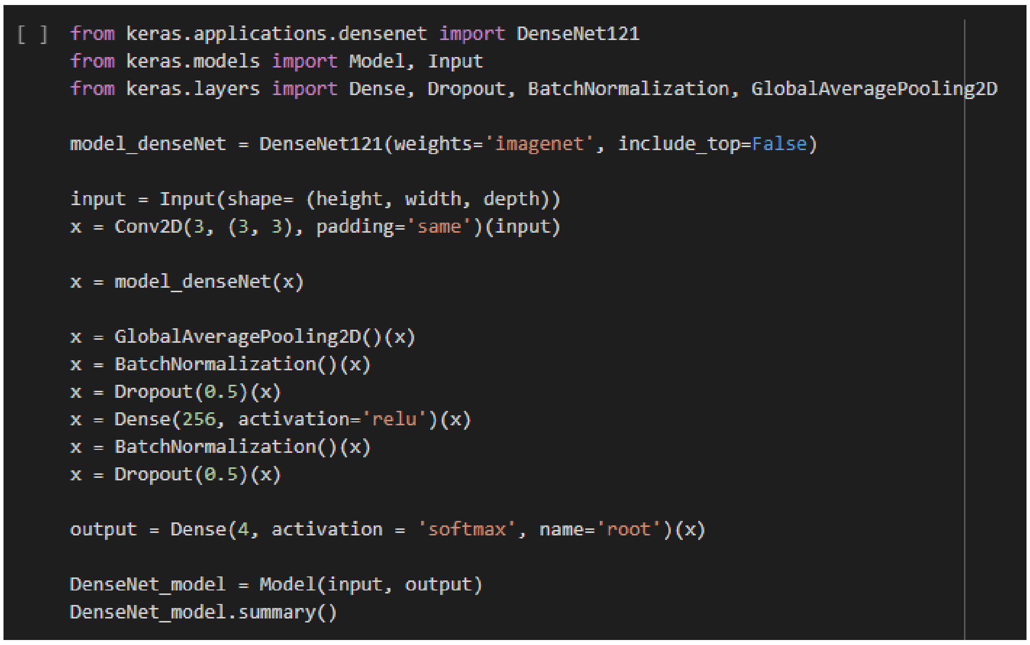
[28] Emmanuel, "PlantVillage Dataset" (2018). Distributed by Kaggle. https://www.kaggle.com/datasets/emmarex/plantdisease

Appendices



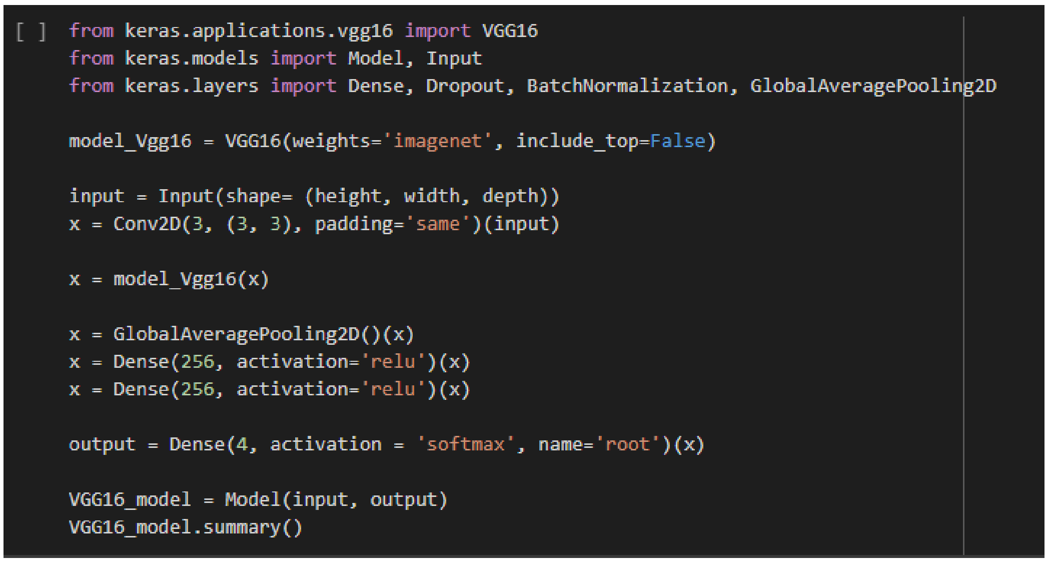
```
model_cnn5 = Sequential()
inputShape = (height, width, depth)
chanDim = -1
if K.image_data_format() == "channels_first":
    inputShape = (depth, height, width)
    chanDim = 1
model_cnn5.add(Conv2D(32, (3, 3), padding="same",input_shape=inputShape))
model_cnn5.add(Activation("relu"))
model_cnn5.add(BatchNormalization(axis=chanDim))
model_cnn5.add(MaxPooling2D(pool_size=(3, 3)))
model_cnn5.add(Dropout(0.25))
model_cnn5.add(Conv2D(64, (3, 3), padding="same"))
model_cnn5.add(Activation("relu"))
model_cnn5.add(BatchNormalization(axis=chanDim))
model_cnn5.add(Conv2D(64, (3, 3), padding="same"))
model_cnn5.add(Activation("relu"))
model_cnn5.add(BatchNormalization(axis=chanDim))
model_cnn5.add(MaxPooling2D(pool_size=(2, 2)))
model_cnn5.add(Dropout(0.25))
model_cnn5.add(Conv2D(128, (3, 3), padding="same"))
model_cnn5.add(Activation("relu"))
model_cnn5.add(BatchNormalization(axis=chanDim))
model_cnn5.add(Conv2D(128, (3, 3), padding="same"))
model_cnn5.add(Activation("relu"))
model_cnn5.add(BatchNormalization(axis=chanDim))
model_cnn5.add(MaxPooling2D(pool_size=(2, 2)))
model_cnn5.add(Dropout(0.25))
model_cnn5.add(Flatten())
model_cnn5.add(Dense(1024))
model_cnn5.add(Activation("relu"))
model_cnn5.add(BatchNormalization())
model_cnn5.add(Dropout(0.5))
model_cnn5.add(Dense(4))
model_cnn5.add(Activation("softmax"))

model_cnn5.summary()
```

Code 1. CNN Architecture from scratch



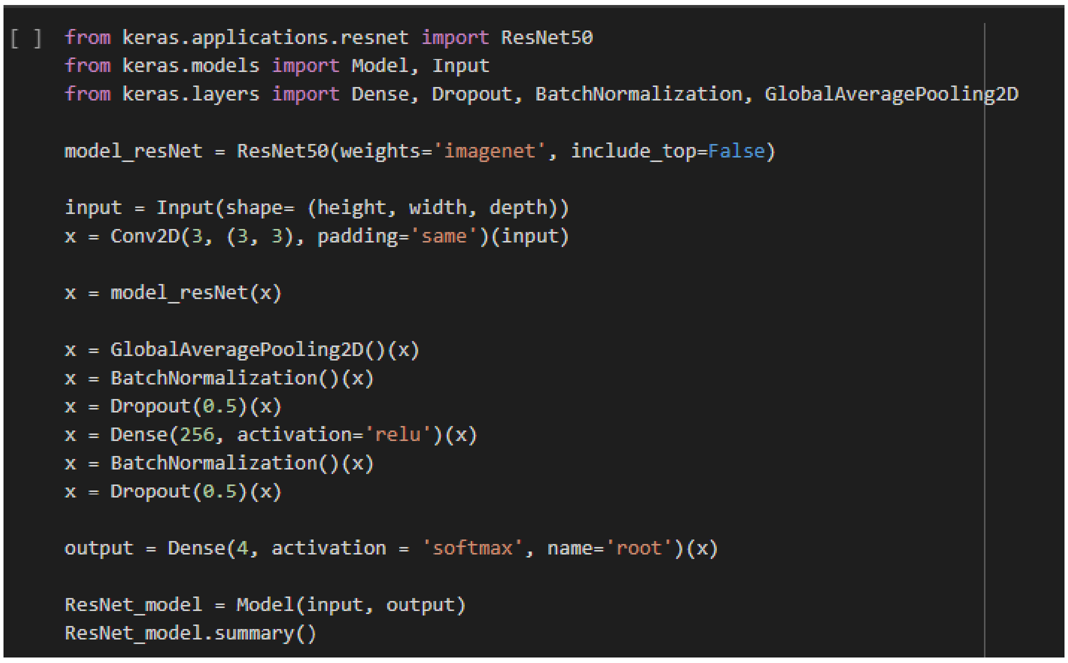
```
from keras.applications.densenet import DenseNet121
from keras.models import Model, Input
from keras.layers import Dense, Dropout, BatchNormalization, GlobalAveragePooling2D

model_denseNet = DenseNet121(weights='imagenet', include_top=False)

input = Input(shape= (height, width, depth))
x = Conv2D(3, (3, 3), padding='same')(input)

x = model_denseNet(x)

x = GlobalAveragePooling2D()(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
x = Dense(256, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)

output = Dense(4, activation = 'softmax', name='root')(x)

DenseNet_model = Model(input, output)
DenseNet_model.summary()
```

Code 2. Pre-trained DenseNet121



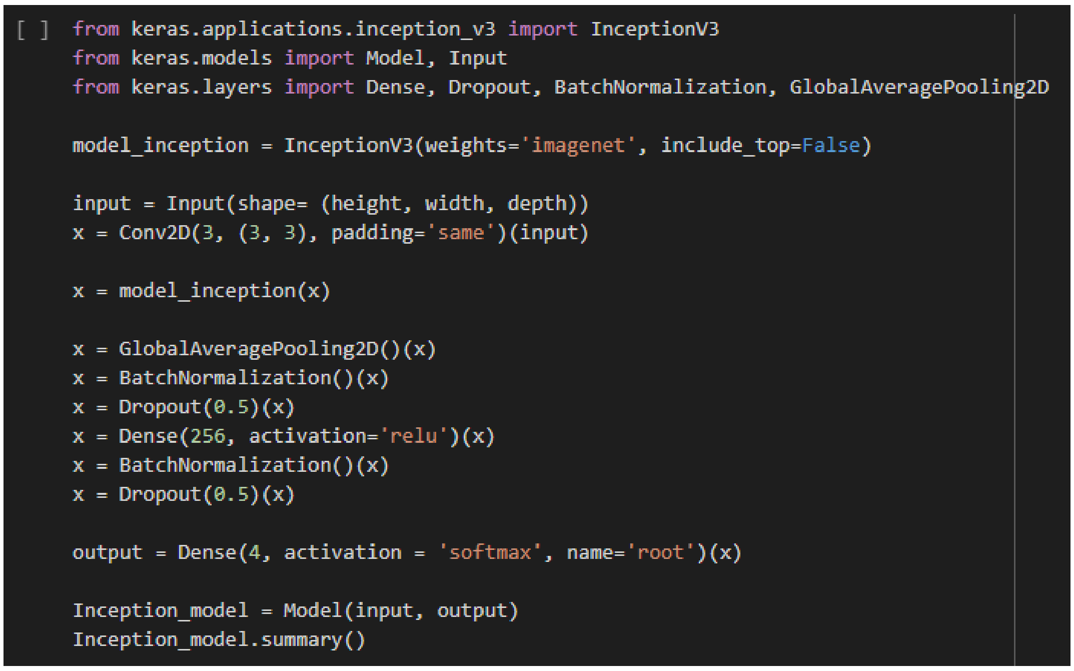
```
[ ]  from keras.applications.vgg16 import VGG16
     from keras.models import Model, Input
     from keras.layers import Dense, Dropout, BatchNormalization, GlobalAveragePooling2D

     model_Vgg16 = VGG16(weights='imagenet', include_top=False)

     input = Input(shape= (height, width, depth))
     x = Conv2D(3, (3, 3), padding='same')(input)

     x = model_Vgg16(x)

     x = GlobalAveragePooling2D()(x)
     x = Dense(256, activation='relu')(x)
     x = Dense(256, activation='relu')(x)

     output = Dense(4, activation = 'softmax', name='root')(x)

     VGG16_model = Model(input, output)
     VGG16_model.summary()
```

Code 3. Pre-trained VGG16



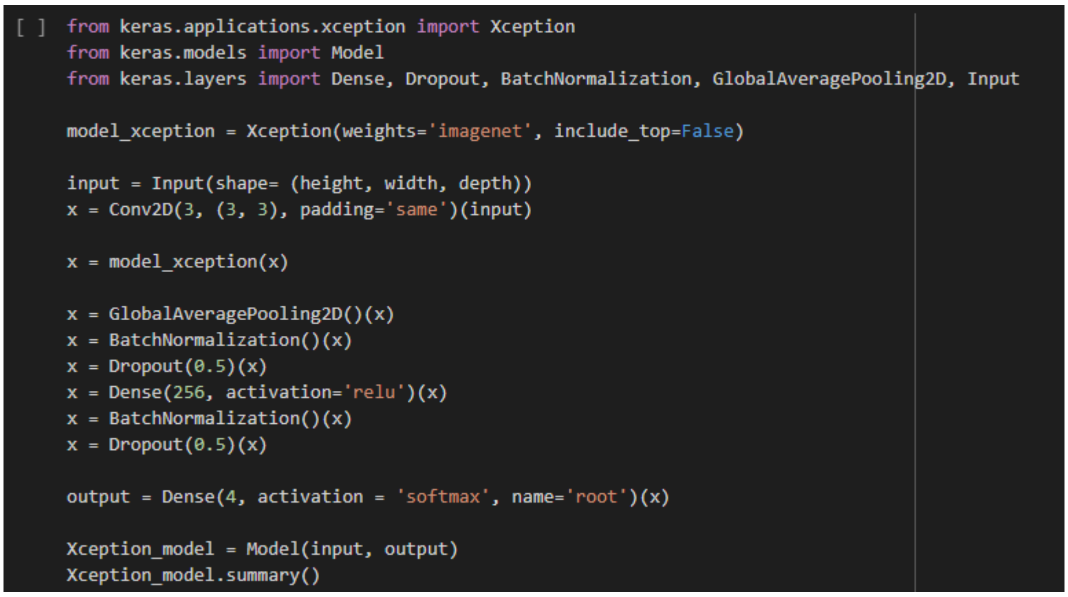
```
[ ]  from keras.applications.resnet import ResNet50
     from keras.models import Model, Input
     from keras.layers import Dense, Dropout, BatchNormalization, GlobalAveragePooling2D

     model_resNet = ResNet50(weights='imagenet', include_top=False)

     input = Input(shape= (height, width, depth))
     x = Conv2D(3, (3, 3), padding='same')(input)

     x = model_resNet(x)

     x = GlobalAveragePooling2D()(x)
     x = BatchNormalization()(x)
     x = Dropout(0.5)(x)
     x = Dense(256, activation='relu')(x)
     x = BatchNormalization()(x)
     x = Dropout(0.5)(x)

     output = Dense(4, activation = 'softmax', name='root')(x)

     ResNet_model = Model(input, output)
     ResNet_model.summary()
```

Code 4. Pre-trained ResNet50



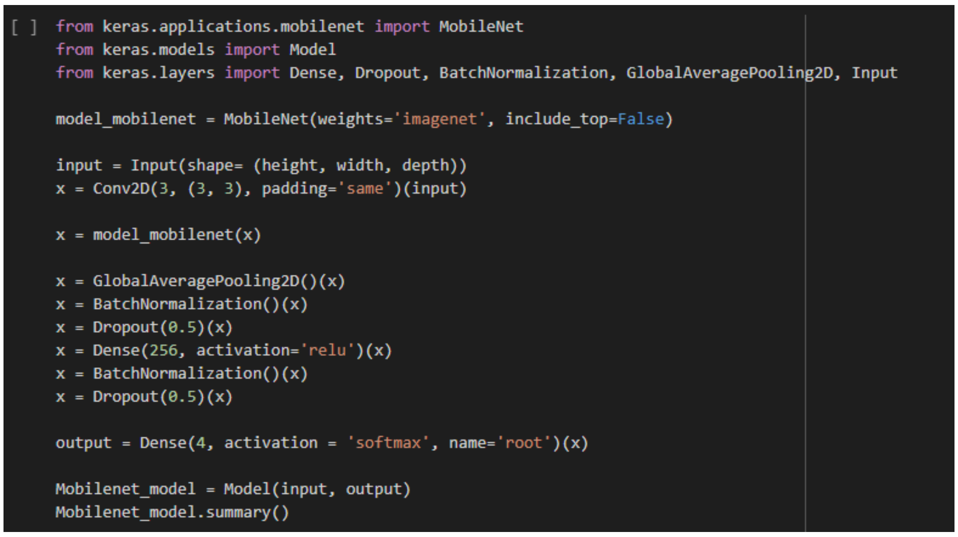
```
[ ]  from keras.applications.inception_v3 import InceptionV3
     from keras.models import Model, Input
     from keras.layers import Dense, Dropout, BatchNormalization, GlobalAveragePooling2D

     model_inception = InceptionV3(weights='imagenet', include_top=False)

     input = Input(shape= (height, width, depth))
     x = Conv2D(3, (3, 3), padding='same')(input)

     x = model_inception(x)

     x = GlobalAveragePooling2D()(x)
     x = BatchNormalization()(x)
     x = Dropout(0.5)(x)
     x = Dense(256, activation='relu')(x)
     x = BatchNormalization()(x)
     x = Dropout(0.5)(x)

     output = Dense(4, activation = 'softmax', name='root')(x)

     Inception_model = Model(input, output)
     Inception_model.summary()
```

Code 5. Pre-trained InceptionV3



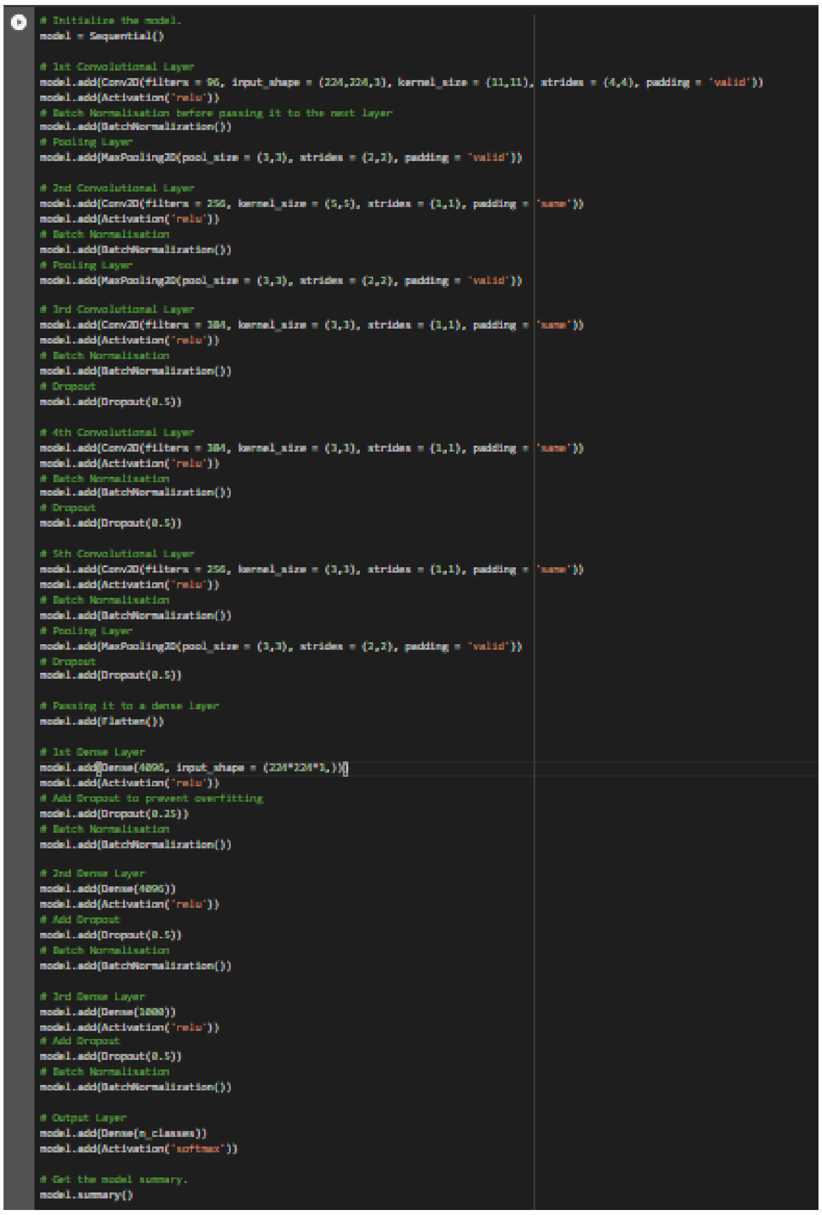
```
[ ]  from keras.applications.xception import Xception
     from keras.models import Model
     from keras.layers import Dense, Dropout, BatchNormalization, GlobalAveragePooling2D, Input

     model_xception = Xception(weights='imagenet', include_top=False)

     input = Input(shape= (height, width, depth))
     x = Conv2D(3, (3, 3), padding='same')(input)

     x = model_xception(x)

     x = GlobalAveragePooling2D()(x)
     x = BatchNormalization()(x)
     x = Dropout(0.5)(x)
     x = Dense(256, activation='relu')(x)
     x = BatchNormalization()(x)
     x = Dropout(0.5)(x)

     output = Dense(4, activation = 'softmax', name='root')(x)

     Xception_model = Model(input, output)
     Xception_model.summary()
```

Code 6. Pre-trained Xception



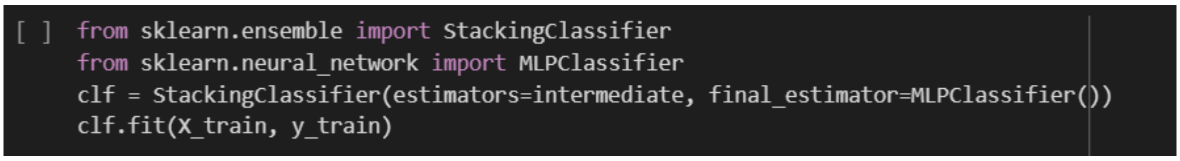
```python
from keras.applications.mobilenet import MobileNet
from keras.models import Model
from keras.layers import Dense, Dropout, BatchNormalization, GlobalAveragePooling2D, Input

model_mobilenet = MobileNet(weights='imagenet', include_top=False)

input = Input(shape= (height, width, depth))
x = Conv2D(3, (3, 3), padding='same')(input)

x = model_mobilenet(x)

x = GlobalAveragePooling2D()(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
x = Dense(256, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)

output = Dense(4, activation = 'softmax', name='root')(x)

Mobilenet_model = Model(input, output)
Mobilenet_model.summary()
```

Code 7. Pre-trained MobileNet



```python
# Initialize the model.
model = Sequential()

# 1st Convolutional Layer
model.add(Conv2D(filters = 96, input_shape = (224,224,3), kernel_size = (11,11), strides = (4,4), padding = 'valid'))
model.add(Activation('relu'))
# Batch Normalisation before passing it to the next layer
model.add(BatchNormalization())
# Pooling Layer
model.add(MaxPooling2D(pool_size = (3,3), strides = (2,2), padding = 'valid'))

# 2nd Convolutional Layer
model.add(Conv2D(filters = 256, kernel_size = (5,5), strides = (1,1), padding = 'same'))
model.add(Activation('relu'))
# Batch Normalisation
model.add(BatchNormalization())
# Pooling Layer
model.add(MaxPooling2D(pool_size = (3,3), strides = (2,2), padding = 'valid'))

# 3rd Convolutional Layer
model.add(Conv2D(filters = 384, kernel_size = (3,3), strides = (1,1), padding = 'same'))
model.add(Activation('relu'))
# Batch Normalisation
model.add(BatchNormalization())
# Dropout
model.add(Dropout(0.5))

# 4th Convolutional Layer
model.add(Conv2D(filters = 384, kernel_size = (3,3), strides = (1,1), padding = 'same'))
model.add(Activation('relu'))
# Batch Normalisation
model.add(BatchNormalization())
# Dropout
model.add(Dropout(0.5))

# 5th Convolutional Layer
model.add(Conv2D(filters = 256, kernel_size = (3,3), strides = (1,1), padding = 'same'))
model.add(Activation('relu'))
# Batch Normalisation
model.add(BatchNormalization())
# Pooling Layer
model.add(MaxPooling2D(pool_size = (3,3), strides = (2,2), padding = 'valid'))
# Dropout
model.add(Dropout(0.5))

# Passing it to a dense layer
model.add(Flatten())

# 1st Dense Layer
model.add(Dense(4096, input_shape = (224*224*3,)))
model.add(Activation('relu'))
# Add Dropout to prevent overfitting
model.add(Dropout(0.25))
# Batch Normalisation
model.add(BatchNormalization())

# 2nd Dense Layer
model.add(Dense(4096))
model.add(Activation('relu'))
# Add Dropout
model.add(Dropout(0.5))
# Batch Normalisation
model.add(BatchNormalization())

# 3rd Dense Layer
model.add(Dense(1000))
model.add(Activation('relu'))
# Add Dropout
model.add(Dropout(0.5))
# Batch Normalisation
model.add(BatchNormalization())

# Output Layer
model.add(Dense(n_classes))
model.add(Activation('softmax'))

# Get the model summary.
model.summary()
```

Code 8. AlexNet Architecture from Scratch



```
[ ] from sklearn.ensemble import StackingClassifier
    from sklearn.neural_network import MLPClassifier
    clf = StackingClassifier(estimators=intermediate, final_estimator=MLPClassifier())
    clf.fit(X_train, y_train)
```

Code 9. Proposed Ensemble Model