American International University-Bangladesh (AIUB)

Flower Growth Classification Detection using CNN models & Ensemble method

Submitted By

17-34484-2	SHAFIUL ALAM FAYSAL
18-36283-1	MD SHOFIUL ALAM TANVIR
18-36318-1	MD RASHEDUL HASAN
18-37203-1	ASIF AHMED

Department of Computer Science

Faculty of Science & Technology (FST)

American International University-Bangladesh

December, 2022

Declaration

This thesis is composed of our original work, and contains no material previously published or written by another person except where due reference has been made in the text. We have clearly stated the contribution of others to our thesis as a whole, including statistical assistance, survey design, data analysis, significant technical procedures, professional editorial advice, financial support and any other original research work used or reported in our thesis. The content of our thesis is the result of work we have carried out since the commencement of Thesis. We acknowledge that copyright of all material contained in my thesis resides with the copyright holder(s) of that material. Where appropriate we have obtained copyright permission from the copyright holder to reproduce material in this thesis and have sought permission from co-authors for any jointly authored works included in the thesis.

SHAFIUL ALAM FAYSAL

ID: 17-34484-2

Department of Computer Science and Engineering

MD. SHOFIUL ALAM TANVIR

ID: 18-36283-1

Department of Computer Science and Engineering

MD. RASHEDUL HASAN

ID: 18-36318-1

Department of Computer Science and Engineering

ASIF AHMED

ID: 18-37203-1

Department of Computer Science and Engineering

Approval

This thesis titled “Flower classification using CNN models & Ensemble method” has been submitted to the following respected member of the board of examiners of the Faculty of Science and Technology impartial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering on dated and has been accepted satisfactorily.

SAZZAD HOSSAIN

Assistant Professor , Computer Science Supervisor

Department of Computer Science

American International University-Bangladesh

MD. MANZURUL HASAN

Associate Professor , Computer Science External

Department of Computer Science

American International University-Bangladesh

Dr. MD. ABDULLAH - AL - JUBAIR

Assistant Professor & Head-In-Charge
[Undergraduate Program]

Department of Computer Science

American International University-Bangladesh

PROF. Dr. DIP NANDI

Professor & Director, Computer Science

Faculty of Science and Technology

American International University-Bangladesh

MASHIOUR RAHMAN

Associate Dean

Faculty of Science and Technology

American International University-Bangladesh

Acknowledgement

First and foremost, we want to convey our gratitude to Allah for his kindness in allowing us to complete our thesis report on "**Flower Growth Classification Detection using CNN models & Ensemble method"**. We would like to thank the Faculty of Science & Technology (FST) for having thesis credit in the graduate program's curriculum and allowing us the opportunity to complete this. We are also grateful to AIUB's Faculty of Science and Technology and Office of Placement and Alumni for providing an opportunity for us to choose a field and complete a thesis there. We are thankful to our supervisor **Sazzad Hossain** sir, for his kind support, direction, useful supervision, guidelines and advice.

Table of Contents

List of Figures

List of Tables

Abstract

One of God’s most exquisite creations, flowers come in countless varieties of species and hues. Numerous flowers can be seen everywhere. There are currently around 352,000 different species of flowers. One flower genre has lots of species. Plumeria has many species, such as Candy stripe, Vera Cruz Ros, Sundance, Thailand red, and many more. As a result, there are a lot of flowers, but people can’t recognize which specific species and what is the current condition of the flower. To solve that problem, A strategy is suggested to solve the issue and identify the flower and its current status. Object recognition from a picture is currently quite promising with the use of machine learning algorithms, albeit there are still some difficulties. A method is put forth using CNN’s classification of a Plumeria flower data set.Our data set contains three classes of Plumeria flowers.A new data set using plumeria flowers was created for these different species of plumeria flowers. Some of our data was physically collected with the help of the smartphone, and some of The data was collected from different websites. We separated our whole data set into three different categories Bloom, pre-bloom, and post-bloom, and divided our data set for image training, validation, and testing. For training, we used 2991 flower images. For validating, we used 428 flower images. And for testing, a total of 854 flower images were used.The Convolutional Neural Network model (CNN) applied in this study are VGG-16 and VGG-19 and Inception V3.The main goal was to determine the state of a Plumeria flower. We train VGG-16 ,VGG-19 and Inception V3 model’s with our dataset.Then after training VGG-16, we get 81% of accuracy. For VGG-19 we got 80% and 81% for Inception V3. By using the data set of different species of Plumeria flower, we got our predicted result. It distinguished our results and research work from others. Till now most of the research work was done based on recognizing flower species or plants. But we tried to figure out the condition of a flower. We categorized them in three parts. So the flowers were detected as Pre Bloom, Bloom and Post Bloom. We have tried to make an algorithm which can recognize the state of a flower. Though there were many problems while training our data set as we didn't have enough memory to run our data set. Sometimes the Kernel crashes and our computer stops. After so many difficulties we achieved our predicted accuracy of the three stages of a flower. The most significant part of this research work was the recognition of flower states. When we examined our data set, there was about 81% of accuracy for identifying each of the flowers. It seemed that our algorithm worked perfectly so that it could analyze the stage of every flower.

Keywords:

CNN,VGG-16,VGG-19,Inception V3,Flower Classification.

Chapter 1: Introduction

1.1 Flower classification using CNN models and Ensemble methods

In our everyday life we see various types of flowers. There are millions of different species of flowers. Plumeria is a unique flower type that has different species like Candy Stripe, Vera Cruz Ros, Thailand Red, Sundance, and many more. As there is a big amount of this creation, it is nearly impossible for a normal person to recognize the species of a flower at a glance. Also, it is impossible for anyone to understand the state of a flower. Thus we proposed a solution to identify the state of a flower and categorized them into three criteria. We decided to use machine learning algorithms to identify a data set of Plumeria flowers. We used Convolutional Neural Network (CNN) models such as VGG-16, VGG-19, and Inception V3. Then we assembled them using the Ensemble method.Previously many works have been done to identify a flower. Many thesis papers have explained identifying many types of flowers. Specifically, we focused on the recognition of the status of a Plumeria flower. CNNs are quickly would become the standard technique in image classification.[T. Akiyama et el.,2019] VGG emerged as the 1st CNN to utilize 3*3 extremely small filters for each convolution operation.[V. Rezende et el.,2019] The authors used the CNN Inception V3 model with the help of transfer learning. Transfer learning methods were used to keep the last parameter of the model and remove the last layer of the present. The authors used oxford-17,102 datasets for this classification and achieved 95% of accuracy.[Xiaoling Xia et el.,2017]There are many different algorithms in CNN models. In image processing or identifying any object machine learning algorithms are widely used. The author used CNN for image classification of fruits. They used CNN, deep learning for fruit classification. They got 94.94\% accuracy With a 13-layer CNN model.[Naranjo Torres et el.,2020] A VGG-16 model was modified and updated using the Kaggle data set. It was pre trained in Imagenet for identifying different small data sets by avoiding overfitting. [Giraddi, S et el., 2020] To identify plant diseases, an examination was done on over 10 plant species. By using the XDB plant disease database they compared two VGG architectures. VGG-16 performed better than VGG-19 in the recognition of plant pathology. They found that a pre-trained CNN model with a depth equal to or smaller than VGG-16 can recognize disease-sensitive features more accurately.[Rezende, V. C et el.,2019]

Many researchers have published different types of papers regarding the recognition of any species. In our research work, we tried to discover the state of a Plumeria flower. So we used CNN models to train our data set and predict the highest average number. Then we used an ensemble method to those predictions to achieve higher accuracy. We used the amex Numpy function to generate an array's highest value. By picking up the highest values, it gave us the result of a flower-blooming state. As we have declared that our data was assembled in three categories, the result was in three states: Pre Bloom, Bloom, and Post Bloom. We trained our data set by changing different parameters. Our goal was to let people recognize or identify the state of any Plumeria flower. It will help the farmers to harvest different species of this flower. They can produce hybrid species with different colors and textures. Our research work will help people to further study and explore other types of flowers. We are ambitious to work on a mobile application that will identify any flower just by scanning through the camera.

1.2 Research Motivation:

Our main motive was not only to identify different Plumeria flowers but also the condition of their blooming session. There is much research work that has been done to identify trees and flowers of different species. But identifying a flower's blooming state has not been done yet. So we targeted to do this unique research to introduce people to the blooming states of a flower. Though we used some regular and popular Convolutional models in our data set training and segmentation, to identify the states we had to set a whole new algorithm along with different parameters. As there is no such work done by anyone, so we tried to do our research work based on this. We also tried some different criteria to show that our algorithm can recognize the life cycle of a flower. But unfortunately, we could not accommodate the data as it should be collected from the harvesters or farmers. So we moved forward with our existing data set and set up our research algorithm according to our target. This is how we were motivated to run the models and methods to achieve our success.

1.3 Research Objective:

A deep learning challenge is image classification. It takes an image class of an object as input and identifies it. There are many approaches for image classification. A convolutional neural network(CNN) is a subtype of neural network used for image classification or recognition. For our work, we chose CNN models because they have many hidden layers which reduce the dimensionality of an image without losing any image information. In CNN there are many hidden convolutional layers, pooling layers, and fully connected layers. To process images, we enter each image as 250x250x3 pixels RGB size. So for each image 187,500 neurons were provided.3x3 convolutional filters and ReLU activation function were used. On the pooling layer, we used 2x2 filters with max pooling and a stride of 2. After that, on a fully connected layer dropout, batch normalization layers finally used the softmax function to give us the result. Our main objective was the classification of flower images to recognize which state a flower belongs to Bloom , Pre Bloom, or Post Bloom .For this classification, we collected our own plumeria dataset. We applied the CNN Ensembled method. For those, we used some popular CNN models VGG-16, VGG-19, and Inception-V3. After assembling these models we got 81% accuracy and successfully categorized these flower images.

1.4 Research Question:

When we took the initiative in our research work, we didn't know how to start it. It seemed to be very difficult for us to decide the whole process. The following questions were aroused in our mind when we started our research.

1. Why did we choose flower classification?
2. Which model and method did we use?
3. How did we collect our data?
4. How was all data classified?
5. How did we process our data to get better results?
6. How did we reach our targeted accuracy?
7. How will people get benefitted?

1.5 Expected Research Outcome:

When we started our research work, we had to set some goals. Thus we predicted to have perfect accuracy, data training, model run and so on. Firstly, we tried to train our data set accurately by our own CNN algorithm, but unfortunately it didn't produce our expected result. So we had to use some popular Convolutional models such as VGG-16, VGG-19 and Inception V3. After that we had to put some effort into identifying the state of a flower by using the Ensemble method. It helped us to gather the result of those three models and pick the highest value. Though our expectation was to achieve 90% of accuracy, our models showed only 81% of it. Which is still a significant result. Our research will help farmers in their irrigation and also in the flower production industry to identify flower states easily. For our research we used only different classes of Plumeria flowers. But other flower states can easily be identified through our research. People will have to use different data sets to identify the varieties of flowers.

Chapter 2: Literature Review

2.1 Introduction

In this following section we'll discuss our research background information and related works. We use the CNN model to classify flower images. We used different types of CNN models to complete our work. Some popular CNN models VGG-16 model, the VGG-19 model, and the Inception -V3 model have been applied to classify the flower images.In the following part, we'll discuss related works about image classification.

2.2 Related Works

Deep learning has made significant progress in current times, with the help of deep learning techniques such as CNN achieving excellent results for large-scale image classification. The following study used a Convolutional neural network to classify flower photos. The CNN model applied in this research for classification contains 1 input layer, then 2 convolutional layers, 2 pooling layers, after that 2 fully connected layers, and finally 1 SoftMax layer. The activation function of ReLU. Four types of flowers are classified: roses, dandelions, tulips, and sunflowers. Each category had the same image size, which ranged from more than 600 to more than 800. The test set consisted of 200 pieces from each category, with the remaining used as the training set. The model can recognize various flower photos quickly and accurately.[Zhao Jiantao et el.,2021]

Ferhat Bozkurt (2021) recognized the flower species using the CNN transfer learning approach. Because flowers are so similar in shape and color, it is quite difficult to tell them apart. Network model VGG-16 model, VGG-19 model, SqueezeNet model, DenseNet-121 model, DenseNet-201 model, and InceptionResNetV2 model were the most popular pre-trained learning algorithms used for flower species classification. The Kaggle dataset was used, which contains 4317 photos of five different types of floral species. The flower species are dandelion, daisy, sunflower, rose, and tulip. There are a total of 764 daisies flower images, 1052 dandelion flower images, a total of 784 roses flower images, and a total of 733 sunflowers flower images. Finally, a total of 984 tulips flower images were included in the flower data. The photos in the dataset are not of great quality. Images in jpg format have a depth of 24 bits and a resolution of (320x240) pixels. CNN is made up of several layers, including an input layer for taking the image, several convolutional layers, several pooling layers, and finally a fully connected layer where a dropout layer is also applied. The pooling layer is applied to reduce feature map dimensions by averaging or maximizing subregion summarization.To improve the classification performance a fully connected layer is applied by the convolutional layers of the image features. To reduce network overfitting and improve training performance, the dropout layer eliminates some network connections from training. For the flower dataset, the InceptionResNetV2 model has the maximum accuracy (92.25%).[Ferhat Bozkurt et el.,2021]

CNN is used in this study to identify the medicinal plant. To classify the plants, a 3 layered CNN model is used. Data augmentation is a strategy for increasing the size of data. A total of 37 thousand pictures of ten different types of local medicinal plants are used, with a single predefined class dataset. There were 1671 raw photos of plant classes in total. 39% of the images were photographed on our own. while the remaining 61% of images were obtained from various sources. To create the feature map, they used 128 distinct types of weight filters. CNN architecture includes convolutional layers,

Max Pooling layers, Gaussian noise layer, Batch normalization layer, and finally dropout layer. The max pooling layer helped them to reduce the total amount of network parameters. Batch normalization was applied to make the network less sensitive toward any particular neuron of the weights. At last, a softmax output function is applied which will convert the outputs into probability-like values. As an optimizer Stochastic Gradient Descent (SGD) was utilized. As a loss function, the categorical cross-entropy loss was used. This CNN approach has an accuracy rating of 71.3%.[Raisa Akter et el.,2020]

Ping Lin and Yongming Chen (2018) used Deep Neural Network to recognize open-area Strawberry Flowers. To recognize strawberry flowers, 3 different types of image recognition algorithms were used based on region. Those three are CNN models. R-CNN is one of them, Another one is Fast R-CNN, and finally, the Faster R-CNN model is used. The strawberry blossom data was gathered by 4 cameras installed on a general ground vehicle. To anticipate the object bounding boxes and classification scores, the RPN approach is applied in the fully connected layer in a convolutional way. A precise deep learning model VGG19 framework is used to train the R-CNN-focused strawberry flower recognition. There are 47 layers in the network. There are 19 learnable weight layers in total: there are 16 convolutional layers and extra three fully connected layers exist in this model. The confusion matrix accuracy and recall were used to evaluate performance. The suggested approach yields a detection accuracy of 86.1%, which is at the cutting edge of the field.[Lin, Ping et el.,2018]

The following paper used deep learning TensorFlow to classify flowers. The CNN model was developed to forecast flower types based on flower photos and important attributes. Daisy, dandelion, sunflower, tulip, and rose flower photos were collected for the dataset. 633 for daisies, 898 for dandelions, 641 for roses, 699 for tulips, and 799 for sunflowers. The flowers have a fixed-size input of 224*224 RGB picture. CNN Model has numerous layers including an input layer, several hidden layers, and an output layer. During the convolutional layer or hidden layer, these layers were used: Convolutional 2D filters, Max pooling 2D filters, padding value, strides, kernel size, pool size, and an activation function Rectified linear unit(ReLu) are used in each layer. The top accuracy of the CNN model was 82%.[M. Rajalakshmi et el.,2021]

In this paper authors proposed the popular InceptionV3 model for classifying the flower images. Convolutional neural networks are an efficient recognition approach that has just been developed. This network eliminates complex image preparation, allowing users to input the original image directly. A deep neural network the InceptionV3 model is used for this classification.For this experiment, they used the both Oxford 17 and 102 flower images dataset. They applied a transfer learning technique that will retain the parameters of the previous layers but it will terminate the final layer of the InceptionV3 model. The model's final layer is trained by employing the backpropagation method, and The cross-entropy which is a loss function is used to change the weight parameter. The last layer has as many output nodes as there are classes in the flower dataset. The model's classification using the oxfords 17 flower dataset generates 95% of accuracy and while oxford 102 gave almost 94% of accuracy rate.[Xiaoling Xia et el.,2017]

The following study used a convolutional neural network to recognize flowers (CNN). This convolutional neural network(CNN) model is trained by initially providing it with a set of labeled flower pictures. Following that, the photos are transformed by a number of layers that contain a convolutional layer, activation function ReLU, pooling layer, and fully connected layers. Those images are taken in groups. The proposed approach requires batches with a fixed number of 32. This proposed model has trained over a hundred and fifty epochs. The image was divided into seventy-five percent for training and twenty-five percent for testing. The CNN model is trained using a subset of the Oxford 102 flowers image data. The stochastic gradient descent(SGD) optimizer is used, and for the loss calculation "categorical cross-entropy" is applied. The model's total accuracy was 90%. When a

real-time image of the hibiscus taken with a smartphone camera was fed into the model, a correct prediction with 98.46% accuracy was obtained.[Parvathy et el.,2020]

The authors of this study recommended employing multiple layered neural networks to identify flower photos based on information gathered during and after segmentation. The Oxford flower images data was utilized for flower image classification, and 7 classification methods that rely on gradient models, color, texture, and shape features including a segmentation algorithm were used. layers. When examined with different flower input images from multiple sources, the overall accuracy of the classification for shape, color, and texture-supervised active shape segmented flower images was 92%.[Inthiyaz Syed et el.,2018]

The authors of this research created an image-capturing approach used to observe flower plants. Every inspection contains 5 in-person images of the same participant taken from various positions (whole plant, each flower front view and each flower lateral view, each leaf top view and each leaf back view). The dataset's 101 species were chosen to primarily represent large plant species and their geographically spread species throughout Germany. The Flora Capture app was used to capture all of the photographs. They used the cutting-edge Inception-ResNet-v2 architecture. The accuracy of classifications for single views ranges from 77.4% for a whole flower plant and for a flower lateral view it shows 88.2% of accuracy.The best total accuracy (97.1%) of all combinations. Poa trivialis generated almost 60% of accuracy and Poa pratensis gave 70% of accuracy.[Rzanny Michael et el.,2019]

Rupinder Kaur, Anubha Jain, and Sarvesh Kumar use machine learning to classify sunflowers. Their proposed system is meant to work by distinguishing bloom from yield using a KNN technique. K nearest neighbor (KNN) was successfully used to separate six different types of sunflower photos. First, the district is generated by taking an information image, and these are then sent into the convolution layer. The usual rate of outcome is 88.52%, which is the best-order result. For each sunflower image, the usual level of each categorization is close to 84.52%.[Rupinder Kaur et el.,2022]

In the following research, flowers were successfully identified utilizing a convolutional neural network (CNN).The first stage detects the minimal bounding box around the blossom and localizes it.For localization, the flower area is segregated through the use of an FCN technique. The second phase involves training a CNN to correctly classify the various flower classifications. They use the VGG-16 model to initialize the proposed FCN. This VGG-16 model has a total of 5 convolutional blocks and 3 other fully connected layers are also used. Every convolutional block is consisting of 2 or 3 convolutional layers and every layer has an activation function which is ReLU. Toward the end of every convolutional phase, a max-pooling layer is applied to feature extraction of the image features, keeping the features representation and scaling independent. The suggested technique is quite accurate, with only 168 out of over 10,000 photos misclassified across all datasets.[Hiary H et el.,2018]

In the following study, an ensemble convolutional neural network was used to infer mode in a smartphone travel survey.Ensemble techniques use many classifications to generate more high accuracy than a single model. some of the most popular ensemble methods usually involve boosting, bagging, and stacking. They use "average voting," "majority voting," and "optimal weights" strategies to integrate the output of the convolutional models. This ensemble technique, which uses a random forest technique works as a meta-learner and produces almost 91.8% of accuracy. These hybrid methods of "majority voting" and "optimal weights" provide prediction estimation accuracy of around 89%, although "average voting" generates approximately 85% of accuracy.[Yazdizadeh Ali et el.,2019]

A convolutional network is applied to examine fruit photos throughout this article. The CNN design is made up of several stages or blocks that are made up of four basic elements: A kernel filter, a

convolution layer, a highly nonlinear learning rate, and a pooling layer or downsampling component are all included. Every stage intends to demonstrate attributes as feature maps, which are array pairs. The pooling layer minimizes the number of network parameters by shrinking the dimension of every convolutional outcome. For minor conversions, pooling techniques benefit in obtaining a consistent description of the input event.They created a CNN with a straightforward architecture that included the first layer as an input, then 3 convolutional layers, a flattened layer, and finally one fully connected layer, at the end an output layer. This model was trained over ten epochs with the help of an SGD optimizer with this speed. The overall final aggregate accuracy rate was 95.45% and the final average F1-score was almost 0.96.[Naranjo Torres et el.,2020]

In this research article, ConvNets Complexity was used for Deep Face identification. Some popular CNN techniques are the VGG16 model, VGG19 model, OverFeat model, Inception-v3 model, and ResNet50 model are among the Deep ConvNets structures studied for the purposes of this research. Following pre-processing, Every picture in the dataset is passed into the appropriate proposed approach, and features are extracted by removing the top of the fully connected layers. The "train test split" feature separates the database into training and testing data. During the training the image dataset "train test split" was specified to 0.2, This means approximately 80% of the entire dataset is used for training and 20% is allocated for testing. After the training, this GoogleNet ConvNets model with the help of another network model inception v3 provides the highest level of accuracy than the other 4 model architectures which is almost 98.46% of accuracy for ar face data. The yale b data it's generated with almost 97.94% of accuracy.[Raj, Mohan et el.,2019]

This study was conducted using a CNN architecture to detect blurred images. This CNN model has an input layer that takes images as input, multiple hidden layers in the middle of the model, and at the end an output layer that shows the result. These are hidden layers: several convolutional layers with activation layers like the ReLU activation function, multiple pooling layers this can be maximum or average, and at the end 2 or 3 fully-connected layers. Convolutional operations are executed on data by a convolutional layer. The dataset is thereafter transferred to the next level. A fully connected layer contains each of the prior layer's interconnections and performs higher data analysis. That Cnn architecture splits input pictures into 100 *100 *3 blocks prior to training and classifies them with 4 convolutional and several pooling layers. With just a 5% margin of error, the percentage of flower images contained in the training dataset varies from 70% to 95%. The additional images are equally distributed among testing and validation. They can increase the classification accuracy up 33.78% despite keeping an efficiency of 90.20%, based on the results.[Chen, C et el.,2019]

This study's authors studied the effectiveness of machine learning techniques for plant leaves infection. Researchers analyzed the accuracy of SVM, RF, SGD, and DL in identifying citrus diseases in plants. The disease classification performance achieved through research is quite impressive. Among them, SGD showed the highest result, almost 86.5% after SVM with 85% accuracy. [Sujatha, R et el.,2021]

To recognize tomato plant infections, S. Nandhini and K. Ashok Kumar (2021) used a CNN model. The goal of this research is to use CNN as a supporting technique to evaluate and extract information from pictures in order to immediately categorize the 4 kinds of tomato plant leaf disease. The system receives tomato leaf photos as input and returns an approximate label indicating the class of each disease image as output. Picture pixels are passed into the top layer of CNN, which would be the input layer. The 2nd layer is the Convolutional layer, which extracts features by performing a series of convolutional processes on the images. The Convolutional layer generates the extracted features.One of the most essential convolutional layer features is the stride size, feature map, and filter size. This classifier makes use of 2 classification algorithms: Vgg16 and InceptionV3. The average test prediction accuracy for the Vgg16 and Inception V3 reached 99.98% and 99.94%, respectively, after this approach was adjusted by using the ICRMBO method. This proposed technique increased

transparency, responsiveness, and concreteness, producing an overall accuracy of 99%.[S. Nandhini et el.,2021]

Inception-V3 and a CNN was used by the authors of the study below to categorize pneumonia from X-ray pictures. Inception-v3, ResNet50, and VGG16 are a few of the pre-trained CNN variants employed. By merging CNN, Inception-V3, VGG-16, and ResNet50, ensembles are produced. In addition to the standard assessment metrics, the performance of the pre-trained and ensemble deep learning models is measured. The CNN model consists of two dropout layers with a 0.5 dropout rate, two dense layers, three 2DConv, three max-pooling layers with a 2 2 pool size, three batch normalization layers, four activation layers with a ReLU activation function, and 2 activation layers. The first layer of the CNN model is made up of a 2DConv layer with 256 filters and a 3 3 kernel. The activation layer, which contains the ReLU activation function, comes after this 2DConv layer. The highest accuracy and recall scores, 99.29% and 99.73% were attained by Inception-V3 with CNN.[Mujahid, Muhammad et el.,2022]

The authors of this paper employed a hybrid deep-learning strategy to identify sunflower leaf disease. The stacking ensemble learning approach is used to combine VGG -16 and MobileNet, 2 classification transfer learning models, to create a hybrid model. The best performers are VGG -16 and MobileNet, with accuracy rates of 81% and 86%, respectively. Additionally, pre-trained networks operate best with small data sets, and since our data set is modest, these 2 pre-trained networks were picked for this task.[Arun Malik et el.,2022]

An ensemble of CNN was used in this study's research article for face validation. The convolutional neural network ensemble (ECNN) method is suggested, and data is collected using the CNN model VGG16, Inception-V3 model, and Xception model of ConvNets. Before the learning process, the structure of the selected models is uploaded. Ensemble approaches aggregate the result predictions of a "large" number of certified models by voting in order to improve classification accuracy. The percentage for every class label is provided by each of the train networks, which are numerous. The final classification is derived by averaging these probabilities. We employ the WebFace dataset, which includes 494,414 color photographs of 10,575 people with a variety of facial movements, lights, wrinkling, picture quality, locations, and partial occlusion. On the Online Faced database and the YouTube Face collection of data, the proposed ECNN technique beats state-of-the-art models with a Rank-5 precision of 97.12% and 100%, respectively.[Kalita, Jugal et el.,2019.

Chapter 3: Methodology

3.1 Introduction

We started our work based on some deep learning algorithms. We used VGG-16, VGG-19, and Inception V3 models. After that, we used the Ensemble method to figure out our final result of the state of a Plumeria flower. Our vision was to find three blooming sessions of a flower. We categorized into three different stages which are Pre Bloom, Bloom, and Post Bloom. We used our own dataset of Plumeria flowers. We trained our dataset by using three different deep-learning algorithms. By using the Ensemble method we tried to find our highest accuracy of the blooming state of a flower.

3.2 Model Framework

In our model architecture from figure after dataset collection we resized the data and make augmentation on it. Then we apply three popular convolutional neural networks. We train each of the models separately. During the VGG-16 model training, we applied a 3*3 convolutional layer with a size of 250*250*3 Relu activation function was used on the convolutional layer. A Max pooling layer was also used. For VGG-16 and VGG-19 training works as same. VGG-19 has three more convolutional layers. The softmax function is used to give us possibilities. Softmax is used at the end of both VGG models. The inception V3 model is made up of 42 layers which is a bit higher than the previous inception V1 and V2 models. But the efficiency of this model is really impressive. The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concatenations, dropouts, and fully connected layers. Batch normalization is used extensively throughout the model and applied to activation inputs. Loss is computed using Softmax. Finally, we ensembled those models and get the average performance accuracy. Lastly, we categorize flowers to which state it belongs.

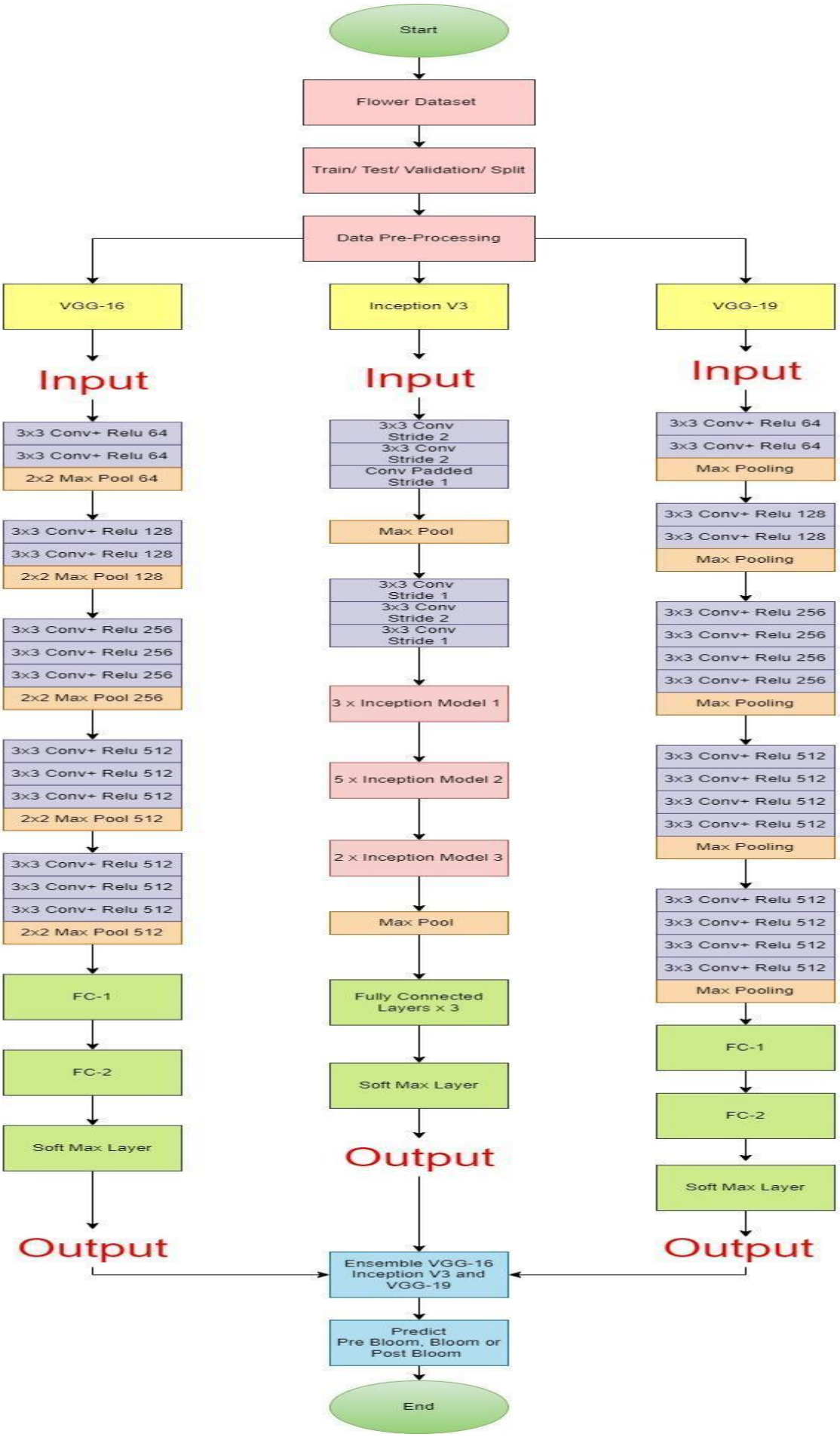


Figure 1: Framework Model.

3.3 Data Collection

Our data set contains three classes of Plumeria flowers. There are many kinds of Plumeria flowers. Such as Plumeria alba,Plumeria rubra,Plumeria pudica,Plumeria obtusa,Plumeria clusioides etc. A new data set using plumeria flowers was created for these different species of plumeria flowers. Some of our data was physically collected with the help of the smartphone, and some of The data was collected from different websites. We separated our whole data set into three different categories Bloom, pre-bloom, and post-bloom, and divided our data set for image training, validation, and testing. For training, we used 2991 flower images. For validating, we used 428 flower images. And for testing, a total of 854 flower images were used.

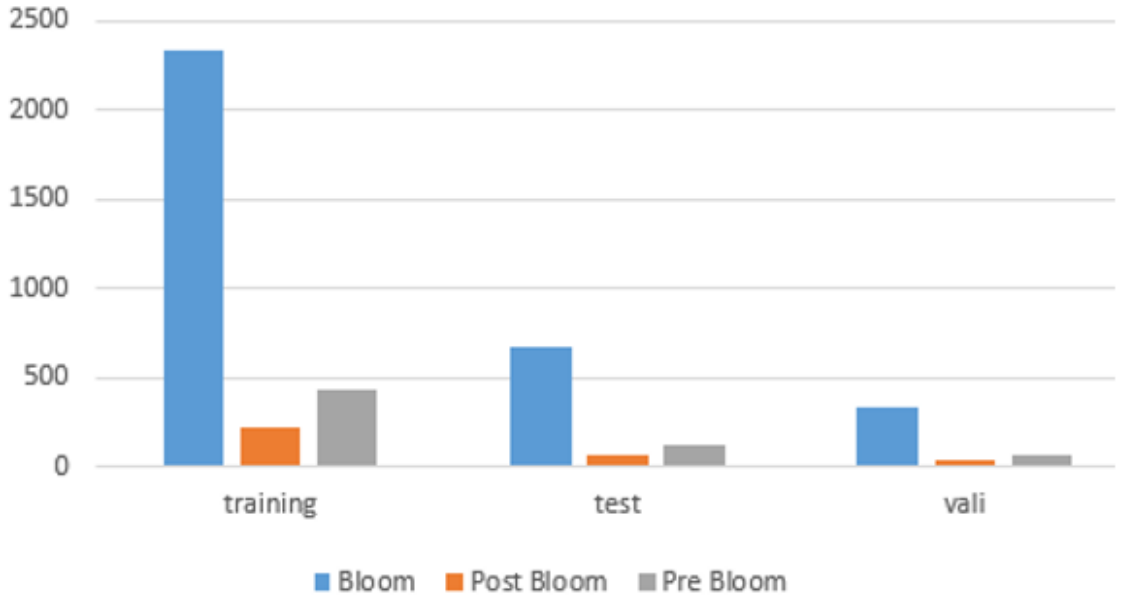


Figure 2: Data set graph distribution by classes.

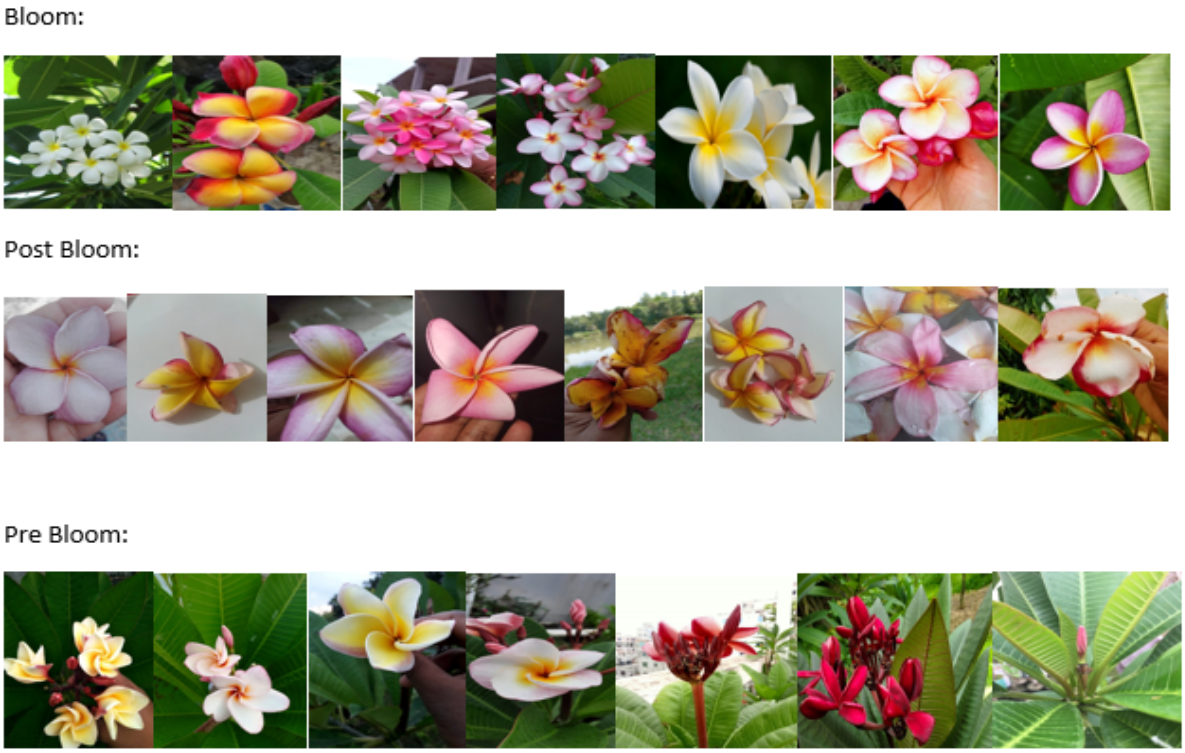


Figure 3: Sample Images of Plumeria Dataset

3.4 Data Preprocessing

To achieve better results data pre-processing is necessary. When using a raw dataset, the results are sometimes disappointing. So data pre-processing helps us to avoid this type of problem. Two types of pre-processing were used: Data Resizing and Data Augmentation.

Data Resizing: When we collected the data, it was fully scrambled. We formatted all pictures in JPG. We resized them to 250x250 pixels. All data was gathered according to RGB format. This process is mandatory to decode every picture's actual color. We collected different pictures of individual species of Plumeria flower to reduce abundance.

Data Augmentation: We tried Data augmentation to avoid over-fitting. We used eight types of data augmentation techniques to complete our process including Rotation (20), Horizontal flip (True), Vertical flip (True), Shear Range(0.2), Re-scale(1./255), Height-Shift Range(0.2), Zoom(0.2), Width-shift Range(0.2). These are different segments that were performed to achieve our maximum accuracy.

3.5 VGG -16

VGG - 16 is the first convolutional neural network that we used. It gives us better performance. In the VGG - 16 model, there is an input layer, convolutional layers, and fully connected layers.

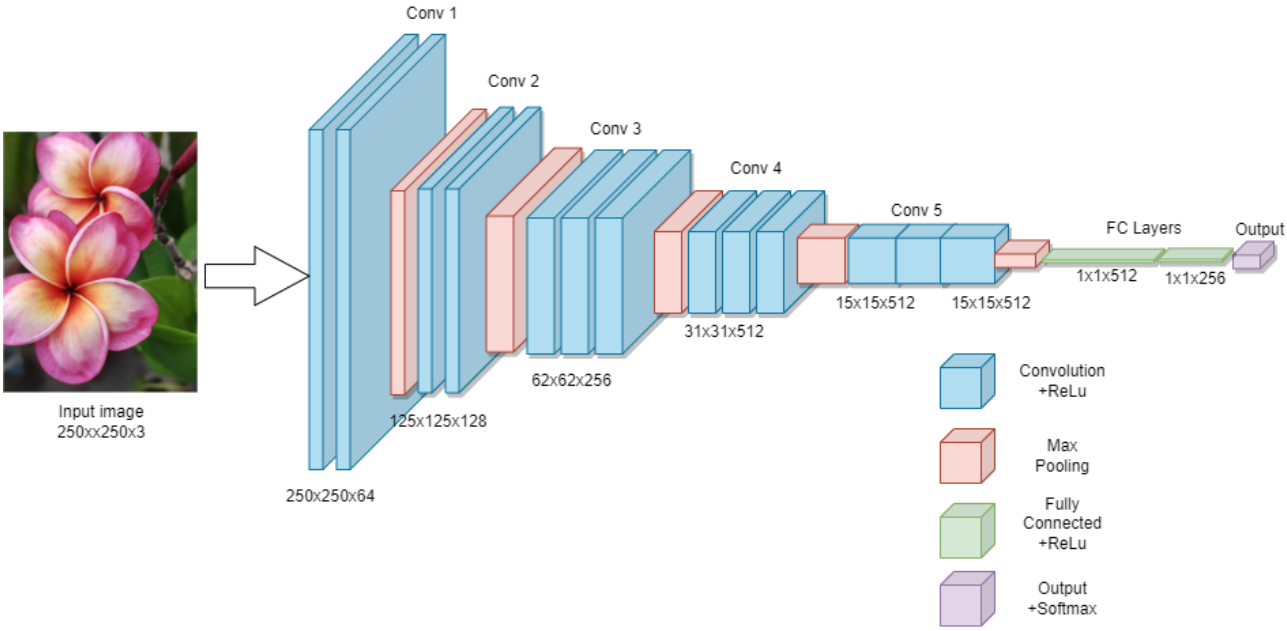


Figure 4:VGG-16 Model Architecture

1. Input layer: In this layer, 250× 250× 3 pixels size of flower images were given as input.

2. Convolution layers: In this layer, a total of 13 convolutional layers were used. The Max pooling layer is also used behind the 2, 4, 7, 10, and 13 layers. The kernel size was 3×3. The rectified linear activation function or ReLU was used as an activation function.

$$ReLU\,function: f(x) = \left\{ \begin{matrix} 0, x < 0 \\ x, x > 0 \end{matrix} \right\}$$

3. Fully connected layer: In this layer 2 fully connected layers were used. The number of neurons was 512,256. Batch normalization the addition of new layers to neural networks can speed up the training process, making learning more accessible and the model more stable. The dropout layer was also used to avoid overfitting, size was 0.1 Finally a softmax function was used to compare the probabilities and give the result. We used a 0.0000001 learning rate with Adam optimizer and sparse categorical cross-entropy loss.

$$Softmax: \sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K$$

Summary of VGG-16:

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 250, 250, 3)]	0
block1_conv1 (Conv2D)	(None, 250, 250, 64)	1792
block1_conv2 (Conv2D)	(None, 250, 250, 64)	36928
block1_pool (MaxPooling2D)	(None, 125, 125, 64)	0
block2_conv1 (Conv2D)	(None, 125, 125, 128)	73856
block2_conv2 (Conv2D)	(None, 125, 125, 128)	147584
block2_pool (MaxPooling2D)	(None, 62, 62, 128)	0
block3_conv1 (Conv2D)	(None, 62, 62, 256)	295168
block3_conv2 (Conv2D)	(None, 62, 62, 256)	590080
block3_conv3 (Conv2D)	(None, 62, 62, 256)	590080
block3_pool (MaxPooling2D)	(None, 31, 31, 256)	0
block4_conv1 (Conv2D)	(None, 31, 31, 512)	1180160
block4_conv2 (Conv2D)	(None, 31, 31, 512)	2359808
block4_conv3 (Conv2D)	(None, 31, 31, 512)	2359808
block4_pool (MaxPooling2D)	(None, 15, 15, 512)	0
block5_conv1 (Conv2D)	(None, 15, 15, 512)	2359808
block5_conv2 (Conv2D)	(None, 15, 15, 512)	2359808
block5_conv3 (Conv2D)	(None, 15, 15, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 512)	12845568
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 3)	771

Total params: 27,692,355
Trainable params: 12,977,667
Non-trainable params: 14,714,688

3.6 VGG -19

It's similar to VGG-16 and another version of the VGG model. Because of the use of numerous 3× 3 filters in each convolutional layer, it is a widely popular model for image classification. It has 3 more convolutional layers than VGG-16. The VGG-19 model has 19 layers with 16 convolution layers and 2 fully connected to classify the flower images into three categories.

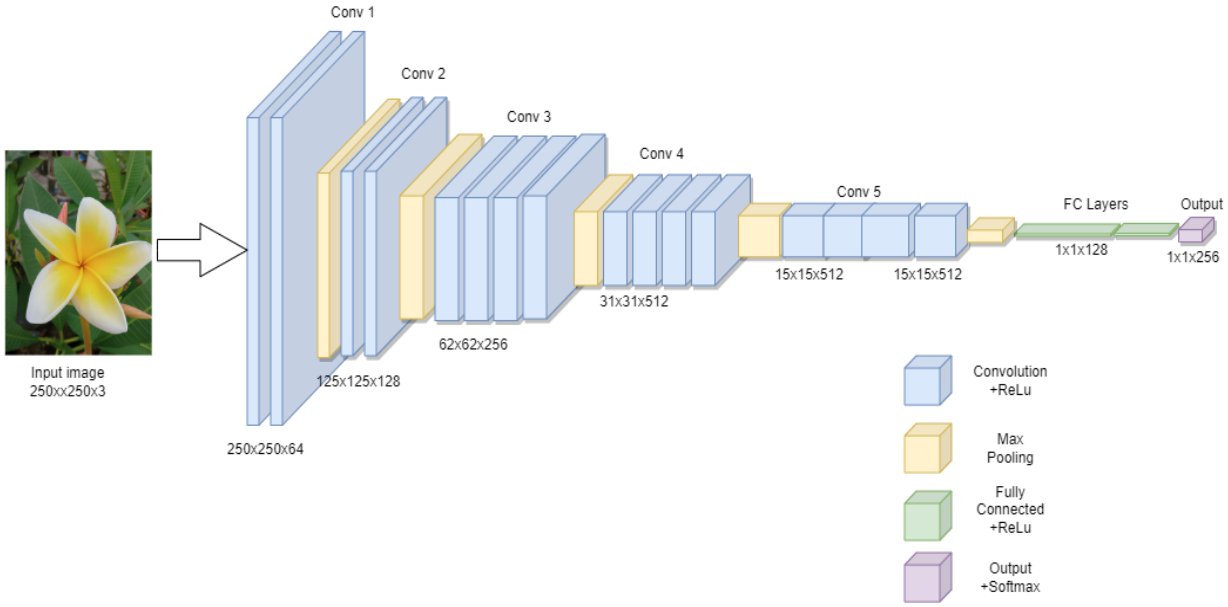


Figure 5: VGG-19 Model Architecture

1. Input layer: In this layer, 250× 250× 3 pixels RGB size of flower images were given as the input layer.
2. Convolutional layer: Convolution is used in this layer to extract features from the input image. The kernel works as a filter, extracting features from input data. Unusual features or pattern identifications that increase categorization efficiency. In the convolutional layer, we used the Conv2D layer for this Classification model. We used a total of 16 convolutional layers for this model. We used one MaxPooling2D layer for each of the convolutional layers. It helps us to reduce the image parameters. It basically chooses the maximum value of the pooling region. It gives us better results than average and min pooling. We used a total of 5 max-pooling layers for this model. We used the same padding which helps us to keep the dimension of the input and output matrix the same.
3. Fully connected layer: In this layer a total of 2 fully connected layers and 1 softmax layer are also used. Two dropout layers are used to avoid overfitting and reduce the complexity and dependency of the training set. Finally, a softmax layer is used. It gives multiple possibilities of results. This function calculates the total probabilities against each class group and gives the value for the specific class.

Summary of VGG-19:

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 250, 250, 3)]	0
block1_conv1 (Conv2D)	(None, 250, 250, 64)	1792
block1_conv2 (Conv2D)	(None, 250, 250, 64)	36928
block1_pool (MaxPooling2D)	(None, 125, 125, 64)	0
block2_conv1 (Conv2D)	(None, 125, 125, 128)	73856
block2_conv2 (Conv2D)	(None, 125, 125, 128)	147584
block2_pool (MaxPooling2D)	(None, 62, 62, 128)	0
block3_conv1 (Conv2D)	(None, 62, 62, 256)	295168
block3_conv2 (Conv2D)	(None, 62, 62, 256)	590080
block3_conv3 (Conv2D)	(None, 62, 62, 256)	590080
block3_conv4 (Conv2D)	(None, 62, 62, 256)	590080
block3_pool (MaxPooling2D)	(None, 31, 31, 256)	0
block4_conv1 (Conv2D)	(None, 31, 31, 512)	1180160
block4_conv2 (Conv2D)	(None, 31, 31, 512)	2359808
block4_conv3 (Conv2D)	(None, 31, 31, 512)	2359808
block4_conv4 (Conv2D)	(None, 31, 31, 512)	2359808
block4_pool (MaxPooling2D)	(None, 15, 15, 512)	0
block5_conv1 (Conv2D)	(None, 15, 15, 512)	2359808
block5_conv2 (Conv2D)	(None, 15, 15, 512)	2359808
block5_conv3 (Conv2D)	(None, 15, 15, 512)	2359808
block5_conv4 (Conv2D)	(None, 15, 15, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 128)	3211392
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 256)	33024
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 3)	771

Total params: 23,269,571
Trainable params: 3,245,187
Non-trainable params: 20,024,384

3.7 Inception -V3

Inception V3: Convolutional neural network's classification model includes Inception-v3.The Inception-V3 model is an updated form of the Inception series model. The Inception - V3 features a complex design with several simultaneous convolutional layers, although it only has a few parameters. Because there are fewer parameters utilized, the model may be implemented in one with fewer memory requirements and run faster. [17] Inception – V3 is the 48-layered convolutional neural network model. The image size of the flower was 250x250x3. Like other CNN models on the Inception V3 model, we used convolutional, max pool, and finally three fully connected(FC) layers. In the first FC layer, we used 512 neurons, and in the second fully connected(FC) layer, we used 256 neurons, and dropout was the same as 0.1 only. In the third fully connected layer, we used 128 neurons, and the dropout was 0.1.

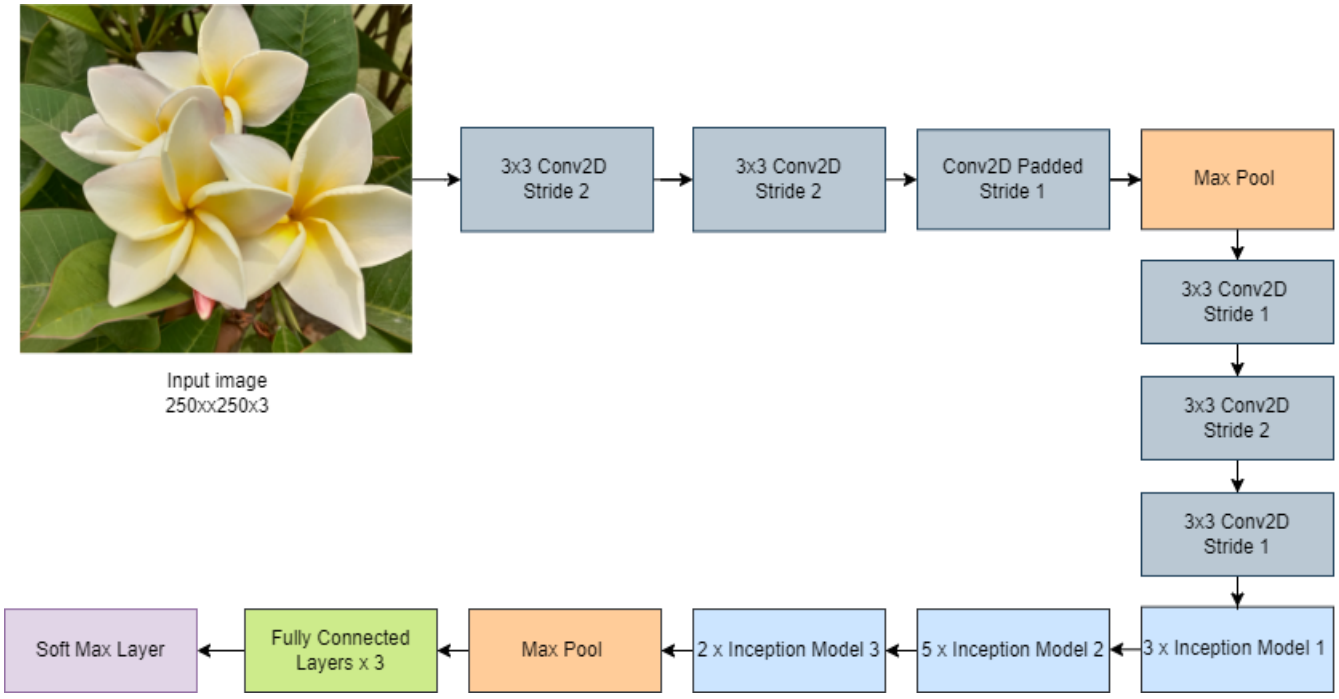


Figure 6: Inception-V3 Model Architecture

Summary of Inception-V3:

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 250, 250, 3)	0	
conv2d (Conv2D)	(None, 124, 124, 32)	864	input_1[0][0]
batch_normalization (BatchNorma	(None, 124, 124, 32)	96	conv2d[0][0]
activation (Activation)	(None, 124, 124, 32)	0	batch_normalization[0][0]
conv2d_1 (Conv2D)	(None, 122, 122, 32)	9216	activation[0][0]
batch_normalization_1 (BatchNor	(None, 122, 122, 32)	96	conv2d_1[0][0]
activation_1 (Activation)	(None, 122, 122, 32)	0	batch_normalization_1[0][0]
conv2d_2 (Conv2D)	(None, 122, 122, 64)	18432	activation_1[0][0]
batch_normalization_2 (BatchNor	(None, 122, 122, 64)	192	conv2d_2[0][0]
activation_2 (Activation)	(None, 122, 122, 64)	0	batch_normalization_2[0][0]
max_pooling2d (MaxPooling2D)	(None, 60, 60, 64)	0	activation_2[0][0]
conv2d_3 (Conv2D)	(None, 60, 60, 80)	5120	max_pooling2d[0][0]
batch_normalization_3 (BatchNor	(None, 60, 60, 80)	240	conv2d_3[0][0]
activation_3 (Activation)	(None, 60, 60, 80)	0	batch_normalization_3[0][0]
conv2d_4 (Conv2D)	(None, 58, 58, 192)	138240	activation_3[0][0]
batch_normalization_4 (BatchNor	(None, 58, 58, 192)	576	conv2d_4[0][0]
activation_4 (Activation)	(None, 58, 58, 192)	0	batch_normalization_4[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 28, 28, 192)	0	activation_4[0][0]
conv2d_8 (Conv2D)	(None, 28, 28, 64)	12288	max_pooling2d_1[0][0]
batch_normalization_8 (BatchNor	(None, 28, 28, 64)	192	conv2d_8[0][0]
activation_8 (Activation)	(None, 28, 28, 64)	0	batch_normalization_8[0][0]
conv2d_6 (Conv2D)	(None, 28, 28, 48)	9216	max_pooling2d_1[0][0]

conv2d_9 (Conv2D)	(None, 28, 28, 96)	55296	activation_8[0][0]
batch_normalization_6 (BatchNor	(None, 28, 28, 48)	144	conv2d_6[0][0]
batch_normalization_9 (BatchNor	(None, 28, 28, 96)	288	conv2d_9[0][0]
activation_6 (Activation)	(None, 28, 28, 48)	0	batch_normalization_6[0][0]
activation_9 (Activation)	(None, 28, 28, 96)	0	batch_normalization_9[0][0]
average_pooling2d (AveragePooli	(None, 28, 28, 192)	0	max_pooling2d_1[0][0]
conv2d_5 (Conv2D)	(None, 28, 28, 64)	12288	max_pooling2d_1[0][0]
conv2d_7 (Conv2D)	(None, 28, 28, 64)	76800	activation_6[0][0]
conv2d_10 (Conv2D)	(None, 28, 28, 96)	82944	activation_9[0][0]
conv2d_11 (Conv2D)	(None, 28, 28, 32)	6144	average_pooling2d[0][0]
batch_normalization_5 (BatchNor	(None, 28, 28, 64)	192	conv2d_5[0][0]
batch_normalization_7 (BatchNor	(None, 28, 28, 64)	192	conv2d_7[0][0]
batch_normalization_10 (BatchNo	(None, 28, 28, 96)	288	conv2d_10[0][0]
batch_normalization_11 (BatchNo	(None, 28, 28, 32)	96	conv2d_11[0][0]
activation_5 (Activation)	(None, 28, 28, 64)	0	batch_normalization_5[0][0]
activation_7 (Activation)	(None, 28, 28, 64)	0	batch_normalization_7[0][0]
activation_10 (Activation)	(None, 28, 28, 96)	0	batch_normalization_10[0][0]
activation_11 (Activation)	(None, 28, 28, 32)	0	batch_normalization_11[0][0]
mixed0 (Concatenate)	(None, 28, 28, 256)	0	activation_5[0][0] activation_7[0][0] activation_10[0][0] activation_11[0][0]
conv2d_15 (Conv2D)	(None, 28, 28, 64)	16384	mixed0[0][0]
batch_normalization_15 (BatchNo	(None, 28, 28, 64)	192	conv2d_15[0][0]
activation_15 (Activation)	(None, 28, 28, 64)	0	batch_normalization_15[0][0]
conv2d_13 (Conv2D)	(None, 28, 28, 48)	12288	mixed0[0][0]

conv2d_16 (Conv2D)	(None, 28, 28, 96)	55296	activation_15[0][0]
batch_normalization_13 (BatchNo	(None, 28, 28, 48)	144	conv2d_13[0][0]
batch_normalization_16 (BatchNo	(None, 28, 28, 96)	288	conv2d_16[0][0]
activation_13 (Activation)	(None, 28, 28, 48)	0	batch_normalization_13[0][0]
activation_16 (Activation)	(None, 28, 28, 96)	0	batch_normalization_16[0][0]
average_pooling2d_1 (AveragePoo	(None, 28, 28, 256)	0	mixed0[0][0]
conv2d_12 (Conv2D)	(None, 28, 28, 64)	16384	mixed0[0][0]
conv2d_14 (Conv2D)	(None, 28, 28, 64)	76800	activation_13[0][0]
conv2d_17 (Conv2D)	(None, 28, 28, 96)	82944	activation_16[0][0]
conv2d_18 (Conv2D)	(None, 28, 28, 64)	16384	average_pooling2d_1[0][0]
batch_normalization_12 (BatchNo	(None, 28, 28, 64)	192	conv2d_12[0][0]
batch_normalization_14 (BatchNo	(None, 28, 28, 64)	192	conv2d_14[0][0]
batch_normalization_17 (BatchNo	(None, 28, 28, 96)	288	conv2d_17[0][0]
batch_normalization_18 (BatchNo	(None, 28, 28, 64)	192	conv2d_18[0][0]
activation_12 (Activation)	(None, 28, 28, 64)	0	batch_normalization_12[0][0]
activation_14 (Activation)	(None, 28, 28, 64)	0	batch_normalization_14[0][0]
activation_17 (Activation)	(None, 28, 28, 96)	0	batch_normalization_17[0][0]
activation_18 (Activation)	(None, 28, 28, 64)	0	batch_normalization_18[0][0]
mixed1 (Concatenate)	(None, 28, 28, 288)	0	activation_12[0][0] activation_14[0][0] activation_17[0][0] activation_18[0][0]
conv2d_22 (Conv2D)	(None, 28, 28, 64)	18432	mixed1[0][0]
batch_normalization_22 (BatchNo	(None, 28, 28, 64)	192	conv2d_22[0][0]
activation_22 (Activation)	(None, 28, 28, 64)	0	batch_normalization_22[0][0]

conv2d_20 (Conv2D)	(None, 28, 28, 48)	13824	mixed1[0][0]
conv2d_23 (Conv2D)	(None, 28, 28, 96)	55296	activation_22[0][0]
batch_normalization_20 (BatchNo	(None, 28, 28, 48)	144	conv2d_20[0][0]
batch_normalization_23 (BatchNo	(None, 28, 28, 96)	288	conv2d_23[0][0]
activation_20 (Activation)	(None, 28, 28, 48)	0	batch_normalization_20[0][0]
activation_23 (Activation)	(None, 28, 28, 96)	0	batch_normalization_23[0][0]
average_pooling2d_2 (AveragePoo	(None, 28, 28, 288)	0	mixed1[0][0]
conv2d_19 (Conv2D)	(None, 28, 28, 64)	18432	mixed1[0][0]
conv2d_21 (Conv2D)	(None, 28, 28, 64)	76800	activation_20[0][0]
conv2d_24 (Conv2D)	(None, 28, 28, 96)	82944	activation_23[0][0]
conv2d_25 (Conv2D)	(None, 28, 28, 64)	18432	average_pooling2d_2[0][0]
batch_normalization_19 (BatchNo	(None, 28, 28, 64)	192	conv2d_19[0][0]
batch_normalization_21 (BatchNo	(None, 28, 28, 64)	192	conv2d_21[0][0]
batch_normalization_24 (BatchNo	(None, 28, 28, 96)	288	conv2d_24[0][0]
batch_normalization_25 (BatchNo	(None, 28, 28, 64)	192	conv2d_25[0][0]
activation_19 (Activation)	(None, 28, 28, 64)	0	batch_normalization_19[0][0]
activation_21 (Activation)	(None, 28, 28, 64)	0	batch_normalization_21[0][0]
activation_24 (Activation)	(None, 28, 28, 96)	0	batch_normalization_24[0][0]
activation_25 (Activation)	(None, 28, 28, 64)	0	batch_normalization_25[0][0]
mixed2 (Concatenate)	(None, 28, 28, 288)	0	activation_19[0][0] activation_21[0][0] activation_24[0][0] activation_25[0][0]
conv2d_27 (Conv2D)	(None, 28, 28, 64)	18432	mixed2[0][0]
batch_normalization_27 (BatchNo	(None, 28, 28, 64)	192	conv2d_27[0][0]

activation_27 (Activation)	(None, 28, 28, 64)	0	batch_normalization_27[0][0]
conv2d_28 (Conv2D)	(None, 28, 28, 96)	55296	activation_27[0][0]
batch_normalization_28 (BatchNo	(None, 28, 28, 96)	288	conv2d_28[0][0]
activation_28 (Activation)	(None, 28, 28, 96)	0	batch_normalization_28[0][0]
conv2d_26 (Conv2D)	(None, 13, 13, 384)	995328	mixed2[0][0]
conv2d_29 (Conv2D)	(None, 13, 13, 96)	82944	activation_28[0][0]
batch_normalization_26 (BatchNo	(None, 13, 13, 384)	1152	conv2d_26[0][0]
batch_normalization_29 (BatchNo	(None, 13, 13, 96)	288	conv2d_29[0][0]
activation_26 (Activation)	(None, 13, 13, 384)	0	batch_normalization_26[0][0]
activation_29 (Activation)	(None, 13, 13, 96)	0	batch_normalization_29[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 13, 13, 288)	0	mixed2[0][0]
mixed3 (Concatenate)	(None, 13, 13, 768)	0	activation_26[0][0] activation_29[0][0] max_pooling2d_2[0][0]
conv2d_34 (Conv2D)	(None, 13, 13, 128)	98304	mixed3[0][0]
batch_normalization_34 (BatchNo	(None, 13, 13, 128)	384	conv2d_34[0][0]
activation_34 (Activation)	(None, 13, 13, 128)	0	batch_normalization_34[0][0]
conv2d_35 (Conv2D)	(None, 13, 13, 128)	114688	activation_34[0][0]
batch_normalization_35 (BatchNo	(None, 13, 13, 128)	384	conv2d_35[0][0]
activation_35 (Activation)	(None, 13, 13, 128)	0	batch_normalization_35[0][0]
conv2d_31 (Conv2D)	(None, 13, 13, 128)	98304	mixed3[0][0]
conv2d_36 (Conv2D)	(None, 13, 13, 128)	114688	activation_35[0][0]
batch_normalization_31 (BatchNo	(None, 13, 13, 128)	384	conv2d_31[0][0]
batch_normalization_36 (BatchNo	(None, 13, 13, 128)	384	conv2d_36[0][0]
activation_31 (Activation)	(None, 13, 13, 128)	0	batch_normalization_31[0][0]

activation_36 (Activation)	(None, 13, 13, 128)	0	batch_normalization_36[0][0]
conv2d_32 (Conv2D)	(None, 13, 13, 128)	114688	activation_31[0][0]
conv2d_37 (Conv2D)	(None, 13, 13, 128)	114688	activation_36[0][0]
batch_normalization_32 (BatchNo	(None, 13, 13, 128)	384	conv2d_32[0][0]
batch_normalization_37 (BatchNo	(None, 13, 13, 128)	384	conv2d_37[0][0]
activation_32 (Activation)	(None, 13, 13, 128)	0	batch_normalization_32[0][0]
activation_37 (Activation)	(None, 13, 13, 128)	0	batch_normalization_37[0][0]
average_pooling2d_3 (AveragePoo	(None, 13, 13, 768)	0	mixed3[0][0]
conv2d_30 (Conv2D)	(None, 13, 13, 192)	147456	mixed3[0][0]
conv2d_33 (Conv2D)	(None, 13, 13, 192)	172032	activation_32[0][0]
conv2d_38 (Conv2D)	(None, 13, 13, 192)	172032	activation_37[0][0]
conv2d_39 (Conv2D)	(None, 13, 13, 192)	147456	average_pooling2d_3[0][0]
batch_normalization_30 (BatchNo	(None, 13, 13, 192)	576	conv2d_30[0][0]
batch_normalization_33 (BatchNo	(None, 13, 13, 192)	576	conv2d_33[0][0]
batch_normalization_38 (BatchNo	(None, 13, 13, 192)	576	conv2d_38[0][0]
batch_normalization_39 (BatchNo	(None, 13, 13, 192)	576	conv2d_39[0][0]
activation_30 (Activation)	(None, 13, 13, 192)	0	batch_normalization_30[0][0]
activation_33 (Activation)	(None, 13, 13, 192)	0	batch_normalization_33[0][0]
activation_38 (Activation)	(None, 13, 13, 192)	0	batch_normalization_38[0][0]
activation_39 (Activation)	(None, 13, 13, 192)	0	batch_normalization_39[0][0]
mixed4 (Concatenate)	(None, 13, 13, 768)	0	activation_30[0][0] activation_33[0][0] activation_38[0][0] activation_39[0][0]
conv2d_44 (Conv2D)	(None, 13, 13, 160)	122880	mixed4[0][0]

batch_normalization_44 (BatchNo	(None, 13, 13, 160)	480	conv2d_44[0][0]
activation_44 (Activation)	(None, 13, 13, 160)	0	batch_normalization_44[0][0]
conv2d_45 (Conv2D)	(None, 13, 13, 160)	179200	activation_44[0][0]
batch_normalization_45 (BatchNo	(None, 13, 13, 160)	480	conv2d_45[0][0]
activation_45 (Activation)	(None, 13, 13, 160)	0	batch_normalization_45[0][0]
conv2d_41 (Conv2D)	(None, 13, 13, 160)	122880	mixed4[0][0]
conv2d_46 (Conv2D)	(None, 13, 13, 160)	179200	activation_45[0][0]
batch_normalization_41 (BatchNo	(None, 13, 13, 160)	480	conv2d_41[0][0]
batch_normalization_46 (BatchNo	(None, 13, 13, 160)	480	conv2d_46[0][0]
activation_41 (Activation)	(None, 13, 13, 160)	0	batch_normalization_41[0][0]
activation_46 (Activation)	(None, 13, 13, 160)	0	batch_normalization_46[0][0]
conv2d_42 (Conv2D)	(None, 13, 13, 160)	179200	activation_41[0][0]
conv2d_47 (Conv2D)	(None, 13, 13, 160)	179200	activation_46[0][0]
batch_normalization_42 (BatchNo	(None, 13, 13, 160)	480	conv2d_42[0][0]
batch_normalization_47 (BatchNo	(None, 13, 13, 160)	480	conv2d_47[0][0]
activation_42 (Activation)	(None, 13, 13, 160)	0	batch_normalization_42[0][0]
activation_47 (Activation)	(None, 13, 13, 160)	0	batch_normalization_47[0][0]
average_pooling2d_4 (AveragePoo	(None, 13, 13, 768)	0	mixed4[0][0]
conv2d_40 (Conv2D)	(None, 13, 13, 192)	147456	mixed4[0][0]
conv2d_43 (Conv2D)	(None, 13, 13, 192)	215040	activation_42[0][0]
conv2d_48 (Conv2D)	(None, 13, 13, 192)	215040	activation_47[0][0]
conv2d_49 (Conv2D)	(None, 13, 13, 192)	147456	average_pooling2d_4[0][0]
batch_normalization_40 (BatchNo	(None, 13, 13, 192)	576	conv2d_40[0][0]
batch_normalization_43 (BatchNo	(None, 13, 13, 192)	576	conv2d_43[0][0]

batch_normalization_48 (BatchNo	(None, 13, 13, 192)	576	conv2d_48[0][0]
batch_normalization_49 (BatchNo	(None, 13, 13, 192)	576	conv2d_49[0][0]
activation_40 (Activation)	(None, 13, 13, 192)	0	batch_normalization_40[0][0]
activation_43 (Activation)	(None, 13, 13, 192)	0	batch_normalization_43[0][0]
activation_48 (Activation)	(None, 13, 13, 192)	0	batch_normalization_48[0][0]
activation_49 (Activation)	(None, 13, 13, 192)	0	batch_normalization_49[0][0]
mixed5 (Concatenate)	(None, 13, 13, 768)	0	activation_40[0][0] activation_43[0][0] activation_48[0][0] activation_49[0][0]
conv2d_54 (Conv2D)	(None, 13, 13, 160)	122880	mixed5[0][0]
batch_normalization_54 (BatchNo	(None, 13, 13, 160)	480	conv2d_54[0][0]
activation_54 (Activation)	(None, 13, 13, 160)	0	batch_normalization_54[0][0]
conv2d_55 (Conv2D)	(None, 13, 13, 160)	179200	activation_54[0][0]
batch_normalization_55 (BatchNo	(None, 13, 13, 160)	480	conv2d_55[0][0]
activation_55 (Activation)	(None, 13, 13, 160)	0	batch_normalization_55[0][0]
conv2d_51 (Conv2D)	(None, 13, 13, 160)	122880	mixed5[0][0]
conv2d_56 (Conv2D)	(None, 13, 13, 160)	179200	activation_55[0][0]
batch_normalization_51 (BatchNo	(None, 13, 13, 160)	480	conv2d_51[0][0]
batch_normalization_56 (BatchNo	(None, 13, 13, 160)	480	conv2d_56[0][0]
activation_51 (Activation)	(None, 13, 13, 160)	0	batch_normalization_51[0][0]
activation_56 (Activation)	(None, 13, 13, 160)	0	batch_normalization_56[0][0]
conv2d_52 (Conv2D)	(None, 13, 13, 160)	179200	activation_51[0][0]
conv2d_57 (Conv2D)	(None, 13, 13, 160)	179200	activation_56[0][0]
batch_normalization_52 (BatchNo	(None, 13, 13, 160)	480	conv2d_52[0][0]

batch_normalization_57 (BatchNo	(None, 13, 13, 160)	480	conv2d_57[0][0]
activation_52 (Activation)	(None, 13, 13, 160)	0	batch_normalization_52[0][0]
activation_57 (Activation)	(None, 13, 13, 160)	0	batch_normalization_57[0][0]
average_pooling2d_5 (AveragePoo	(None, 13, 13, 768)	0	mixed5[0][0]
conv2d_50 (Conv2D)	(None, 13, 13, 192)	147456	mixed5[0][0]
conv2d_53 (Conv2D)	(None, 13, 13, 192)	215040	activation_52[0][0]
conv2d_58 (Conv2D)	(None, 13, 13, 192)	215040	activation_57[0][0]
conv2d_59 (Conv2D)	(None, 13, 13, 192)	147456	average_pooling2d_5[0][0]
batch_normalization_50 (BatchNo	(None, 13, 13, 192)	576	conv2d_50[0][0]
batch_normalization_53 (BatchNo	(None, 13, 13, 192)	576	conv2d_53[0][0]
batch_normalization_58 (BatchNo	(None, 13, 13, 192)	576	conv2d_58[0][0]
batch_normalization_59 (BatchNo	(None, 13, 13, 192)	576	conv2d_59[0][0]
activation_50 (Activation)	(None, 13, 13, 192)	0	batch_normalization_50[0][0]
activation_53 (Activation)	(None, 13, 13, 192)	0	batch_normalization_53[0][0]
activation_58 (Activation)	(None, 13, 13, 192)	0	batch_normalization_58[0][0]
activation_59 (Activation)	(None, 13, 13, 192)	0	batch_normalization_59[0][0]
mixed6 (Concatenate)	(None, 13, 13, 768)	0	activation_50[0][0] activation_53[0][0] activation_58[0][0] activation_59[0][0]
conv2d_64 (Conv2D)	(None, 13, 13, 192)	147456	mixed6[0][0]
batch_normalization_64 (BatchNo	(None, 13, 13, 192)	576	conv2d_64[0][0]
activation_64 (Activation)	(None, 13, 13, 192)	0	batch_normalization_64[0][0]
conv2d_65 (Conv2D)	(None, 13, 13, 192)	258048	activation_64[0][0]
batch_normalization_65 (BatchNo	(None, 13, 13, 192)	576	conv2d_65[0][0]

activation_65 (Activation)	(None, 13, 13, 192)	0	batch_normalization_65[0][0]
conv2d_61 (Conv2D)	(None, 13, 13, 192)	147456	mixed6[0][0]
conv2d_66 (Conv2D)	(None, 13, 13, 192)	258048	activation_65[0][0]
batch_normalization_61 (BatchNo	(None, 13, 13, 192)	576	conv2d_61[0][0]
batch_normalization_66 (BatchNo	(None, 13, 13, 192)	576	conv2d_66[0][0]
activation_61 (Activation)	(None, 13, 13, 192)	0	batch_normalization_61[0][0]
activation_66 (Activation)	(None, 13, 13, 192)	0	batch_normalization_66[0][0]
conv2d_62 (Conv2D)	(None, 13, 13, 192)	258048	activation_61[0][0]
conv2d_67 (Conv2D)	(None, 13, 13, 192)	258048	activation_66[0][0]
batch_normalization_62 (BatchNo	(None, 13, 13, 192)	576	conv2d_62[0][0]
batch_normalization_67 (BatchNo	(None, 13, 13, 192)	576	conv2d_67[0][0]
activation_62 (Activation)	(None, 13, 13, 192)	0	batch_normalization_62[0][0]
activation_67 (Activation)	(None, 13, 13, 192)	0	batch_normalization_67[0][0]
average_pooling2d_6 (AveragePoo	(None, 13, 13, 768)	0	mixed6[0][0]
conv2d_60 (Conv2D)	(None, 13, 13, 192)	147456	mixed6[0][0]
conv2d_63 (Conv2D)	(None, 13, 13, 192)	258048	activation_62[0][0]
conv2d_68 (Conv2D)	(None, 13, 13, 192)	258048	activation_67[0][0]
conv2d_69 (Conv2D)	(None, 13, 13, 192)	147456	average_pooling2d_6[0][0]
batch_normalization_60 (BatchNo	(None, 13, 13, 192)	576	conv2d_60[0][0]
batch_normalization_63 (BatchNo	(None, 13, 13, 192)	576	conv2d_63[0][0]
batch_normalization_68 (BatchNo	(None, 13, 13, 192)	576	conv2d_68[0][0]
batch_normalization_69 (BatchNo	(None, 13, 13, 192)	576	conv2d_69[0][0]
activation_60 (Activation)	(None, 13, 13, 192)	0	batch_normalization_60[0][0]
activation_63 (Activation)	(None, 13, 13, 192)	0	batch_normalization_63[0][0]

activation_68 (Activation)	(None, 13, 13, 192)	0	batch_normalization_68[0][0]
activation_69 (Activation)	(None, 13, 13, 192)	0	batch_normalization_69[0][0]
mixed7 (Concatenate)	(None, 13, 13, 768)	0	activation_60[0][0] activation_63[0][0] activation_68[0][0] activation_69[0][0]
conv2d_72 (Conv2D)	(None, 13, 13, 192)	147456	mixed7[0][0]
batch_normalization_72 (BatchNo	(None, 13, 13, 192)	576	conv2d_72[0][0]
activation_72 (Activation)	(None, 13, 13, 192)	0	batch_normalization_72[0][0]
conv2d_73 (Conv2D)	(None, 13, 13, 192)	258048	activation_72[0][0]
batch_normalization_73 (BatchNo	(None, 13, 13, 192)	576	conv2d_73[0][0]
activation_73 (Activation)	(None, 13, 13, 192)	0	batch_normalization_73[0][0]
conv2d_70 (Conv2D)	(None, 13, 13, 192)	147456	mixed7[0][0]
conv2d_74 (Conv2D)	(None, 13, 13, 192)	258048	activation_73[0][0]
batch_normalization_70 (BatchNo	(None, 13, 13, 192)	576	conv2d_70[0][0]
batch_normalization_74 (BatchNo	(None, 13, 13, 192)	576	conv2d_74[0][0]
activation_70 (Activation)	(None, 13, 13, 192)	0	batch_normalization_70[0][0]
activation_74 (Activation)	(None, 13, 13, 192)	0	batch_normalization_74[0][0]
conv2d_71 (Conv2D)	(None, 6, 6, 320)	552960	activation_70[0][0]
conv2d_75 (Conv2D)	(None, 6, 6, 192)	331776	activation_74[0][0]
batch_normalization_71 (BatchNo	(None, 6, 6, 320)	960	conv2d_71[0][0]
batch_normalization_75 (BatchNo	(None, 6, 6, 192)	576	conv2d_75[0][0]
activation_71 (Activation)	(None, 6, 6, 320)	0	batch_normalization_71[0][0]
activation_75 (Activation)	(None, 6, 6, 192)	0	batch_normalization_75[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 768)	0	mixed7[0][0]

mixed8 (Concatenate)	(None, 6, 6, 1280)	0	activation_71[0][0] activation_75[0][0] max_pooling2d_3[0][0]
conv2d_80 (Conv2D)	(None, 6, 6, 448)	573440	mixed8[0][0]
batch_normalization_80 (BatchNo	(None, 6, 6, 448)	1344	conv2d_80[0][0]
activation_80 (Activation)	(None, 6, 6, 448)	0	batch_normalization_80[0][0]
conv2d_77 (Conv2D)	(None, 6, 6, 384)	491520	mixed8[0][0]
conv2d_81 (Conv2D)	(None, 6, 6, 384)	1548288	activation_80[0][0]
batch_normalization_77 (BatchNo	(None, 6, 6, 384)	1152	conv2d_77[0][0]
batch_normalization_81 (BatchNo	(None, 6, 6, 384)	1152	conv2d_81[0][0]
activation_77 (Activation)	(None, 6, 6, 384)	0	batch_normalization_77[0][0]
activation_81 (Activation)	(None, 6, 6, 384)	0	batch_normalization_81[0][0]
conv2d_78 (Conv2D)	(None, 6, 6, 384)	442368	activation_77[0][0]
conv2d_79 (Conv2D)	(None, 6, 6, 384)	442368	activation_77[0][0]
conv2d_82 (Conv2D)	(None, 6, 6, 384)	442368	activation_81[0][0]
conv2d_83 (Conv2D)	(None, 6, 6, 384)	442368	activation_81[0][0]
average_pooling2d_7 (AveragePoo	(None, 6, 6, 1280)	0	mixed8[0][0]
conv2d_76 (Conv2D)	(None, 6, 6, 320)	409600	mixed8[0][0]
batch_normalization_78 (BatchNo	(None, 6, 6, 384)	1152	conv2d_78[0][0]
batch_normalization_79 (BatchNo	(None, 6, 6, 384)	1152	conv2d_79[0][0]
batch_normalization_82 (BatchNo	(None, 6, 6, 384)	1152	conv2d_82[0][0]
batch_normalization_83 (BatchNo	(None, 6, 6, 384)	1152	conv2d_83[0][0]
conv2d_84 (Conv2D)	(None, 6, 6, 192)	245760	average_pooling2d_7[0][0]
batch_normalization_76 (BatchNo	(None, 6, 6, 320)	960	conv2d_76[0][0]
activation_78 (Activation)	(None, 6, 6, 384)	0	batch_normalization_78[0][0]

activation_79 (Activation)	(None, 6, 6, 384)	0	batch_normalization_79[0][0]
activation_82 (Activation)	(None, 6, 6, 384)	0	batch_normalization_82[0][0]
activation_83 (Activation)	(None, 6, 6, 384)	0	batch_normalization_83[0][0]
batch_normalization_84 (BatchNo	(None, 6, 6, 192)	576	conv2d_84[0][0]
activation_76 (Activation)	(None, 6, 6, 320)	0	batch_normalization_76[0][0]
mixed9_0 (Concatenate)	(None, 6, 6, 768)	0	activation_78[0][0] activation_79[0][0]
concatenate (Concatenate)	(None, 6, 6, 768)	0	activation_82[0][0] activation_83[0][0]
activation_84 (Activation)	(None, 6, 6, 192)	0	batch_normalization_84[0][0]
mixed9 (Concatenate)	(None, 6, 6, 2048)	0	activation_76[0][0] mixed9_0[0][0] concatenate[0][0] activation_84[0][0]
conv2d_89 (Conv2D)	(None, 6, 6, 448)	917504	mixed9[0][0]
batch_normalization_89 (BatchNo	(None, 6, 6, 448)	1344	conv2d_89[0][0]
activation_89 (Activation)	(None, 6, 6, 448)	0	batch_normalization_89[0][0]
conv2d_86 (Conv2D)	(None, 6, 6, 384)	786432	mixed9[0][0]
conv2d_90 (Conv2D)	(None, 6, 6, 384)	1548288	activation_89[0][0]
batch_normalization_86 (BatchNo	(None, 6, 6, 384)	1152	conv2d_86[0][0]
batch_normalization_90 (BatchNo	(None, 6, 6, 384)	1152	conv2d_90[0][0]
activation_86 (Activation)	(None, 6, 6, 384)	0	batch_normalization_86[0][0]
activation_90 (Activation)	(None, 6, 6, 384)	0	batch_normalization_90[0][0]
conv2d_87 (Conv2D)	(None, 6, 6, 384)	442368	activation_86[0][0]
conv2d_88 (Conv2D)	(None, 6, 6, 384)	442368	activation_86[0][0]
conv2d_91 (Conv2D)	(None, 6, 6, 384)	442368	activation_90[0][0]
conv2d_92 (Conv2D)	(None, 6, 6, 384)	442368	activation_90[0][0]

average_pooling2d_8 (AveragePoo	(None, 6, 6, 2048)	0	mixed9[0][0]
conv2d_85 (Conv2D)	(None, 6, 6, 320)	655360	mixed9[0][0]
batch_normalization_87 (BatchNo	(None, 6, 6, 384)	1152	conv2d_87[0][0]
batch_normalization_88 (BatchNo	(None, 6, 6, 384)	1152	conv2d_88[0][0]
batch_normalization_91 (BatchNo	(None, 6, 6, 384)	1152	conv2d_91[0][0]
batch_normalization_92 (BatchNo	(None, 6, 6, 384)	1152	conv2d_92[0][0]
conv2d_93 (Conv2D)	(None, 6, 6, 192)	393216	average_pooling2d_8[0][0]
batch_normalization_85 (BatchNo	(None, 6, 6, 320)	960	conv2d_85[0][0]
activation_87 (Activation)	(None, 6, 6, 384)	0	batch_normalization_87[0][0]
activation_88 (Activation)	(None, 6, 6, 384)	0	batch_normalization_88[0][0]
activation_91 (Activation)	(None, 6, 6, 384)	0	batch_normalization_91[0][0]
activation_92 (Activation)	(None, 6, 6, 384)	0	batch_normalization_92[0][0]
batch_normalization_93 (BatchNo	(None, 6, 6, 192)	576	conv2d_93[0][0]
activation_85 (Activation)	(None, 6, 6, 320)	0	batch_normalization_85[0][0]
mixed9_1 (Concatenate)	(None, 6, 6, 768)	0	activation_87[0][0] activation_88[0][0]
concatenate_1 (Concatenate)	(None, 6, 6, 768)	0	activation_91[0][0] activation_92[0][0]
activation_93 (Activation)	(None, 6, 6, 192)	0	batch_normalization_93[0][0]
mixed10 (Concatenate)	(None, 6, 6, 2048)	0	activation_85[0][0] mixed9_1[0][0] concatenate_1[0][0] activation_93[0][0]
flatten (Flatten)	(None, 73728)	0	mixed10[0][0]
dense (Dense)	(None, 512)	37749248	flatten[0][0]
dropout (Dropout)	(None, 512)	0	dense[0][0]
dense_1 (Dense)	(None, 256)	131328	dropout[0][0]
dropout_1 (Dropout)	(None, 256)	0	dense_1[0][0]
dense_2 (Dense)	(None, 128)	32896	dropout_1[0][0]
dropout_2 (Dropout)	(None, 128)	0	dense_2[0][0]
dense_3 (Dense)	(None, 3)	387	dropout_2[0][0]

Total params: 59,716,643
Trainable params: 37,913,859
Non-trainable params: 21,802,784

3.8 Ensemble Method

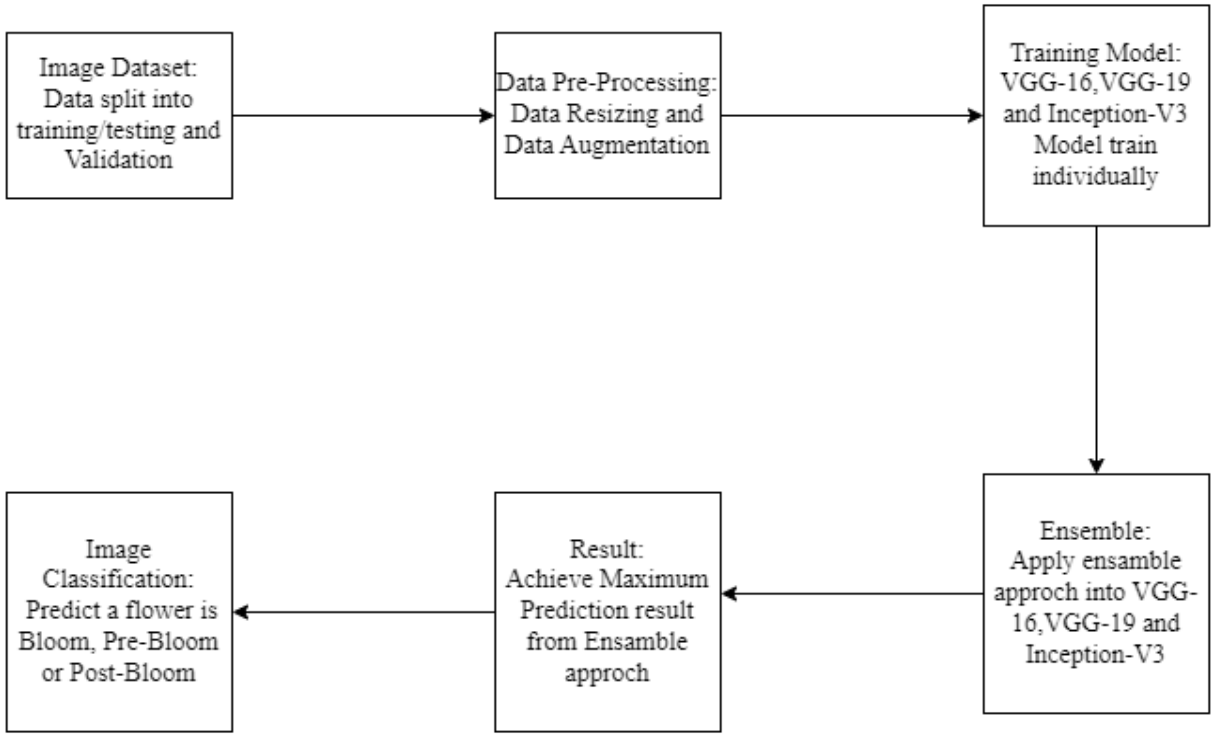


Figure 7: Ensemble method procedure

In the Ensemble method we usually train the same data set using different models. For each model, we set a prediction which is combined to make our final result. The ensemble method is a complex machine learning method that improves overall performance by combining other models.[Rajayogi, J et el.,2019] We used VGG-16, VGG-19, and Inception V3 models to boost our classification accuracy. These three models individually predicted the result of those data sets. After combining these results, an average accuracy prediction was set to determine the state of a Plumeria flower. We saved each model in hdf5 file formats. The amax Numpy function was used to generate an array's maximum value. To determine the state of a flower condition, this function picked the highest value.

Model Name	Total Parameters	NON- Trainable Parameters	Trainable Parameters
VGG-16	27,692,355	14,714,688	12,977,667
VGG-19	23,269,571	20,024,384	3,245,187
INCEPTION-V3	59,716,643	21,802,784	37,913,859
	Total=110,678,569	Total=56,541,856	Total=54,136.713

Table 1: Model Parameters

3.9 Training the models

We separated our whole data set into three different categories Bloom, pre-bloom, and post-bloom, and divided our data set for image training, validation, and testing. For training, we used 2991 flower images. For validating, we used 428 flower images. And for testing, a total of 854 flower images were used. After separating our data we preprocessed data. We updated our model multiple times and adapted the optimizer, learning rate, and loss function to maximize accuracy and minimize loss. We used Adam as an optimizer and for the loss function, we used sparse_categorical_crossentropy. Our minimum learning rate is 1e-7. For less memory, we use 1 batch size and quicker for training our model.

3.10 Project Time Estimation

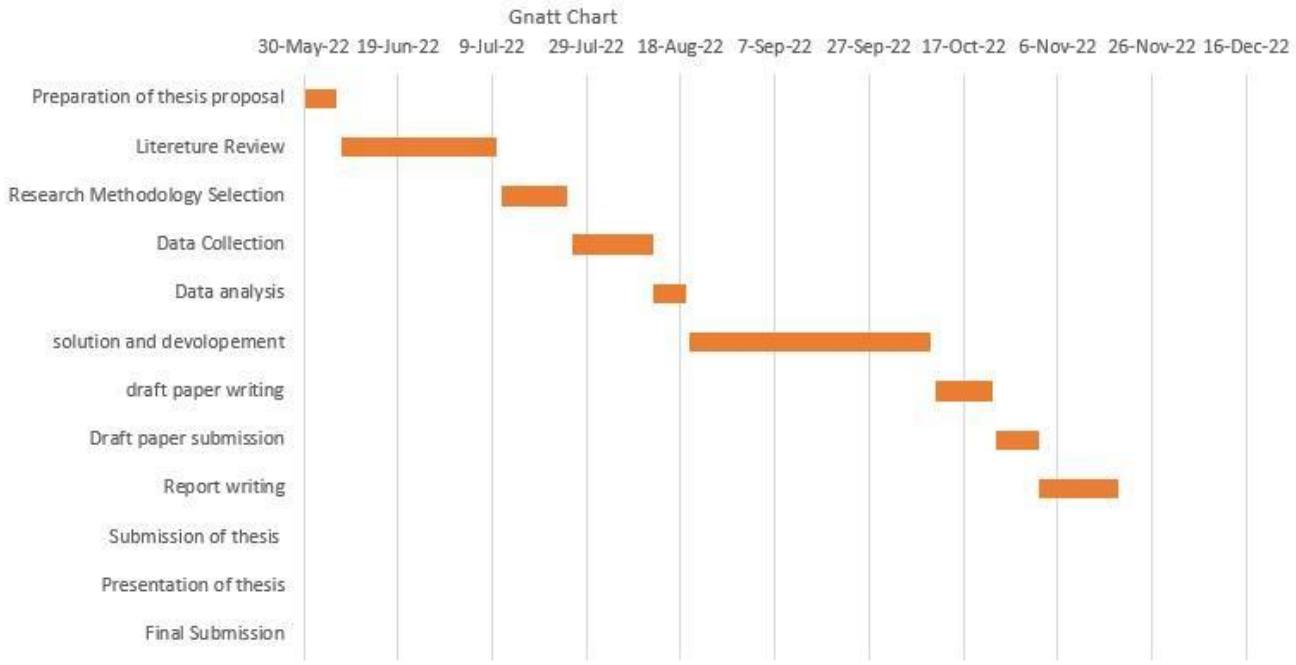


Figure 8: Project Time Estimation

3.11 Project Cost Estimation

	Estimated Rate Per Unit/Month	Quantity/ Unit	Total Cost (BDT)
1. Project Team	0.0	1	0.0
			Total = 0.0
2. Office Cost	0.0	0	0.0
			Total = 0.0
3. Hardware Cost			
Server	0.0	0	0.0
PC’s	0.0	4	0.0
			Total = 0.0
4. Networking Cost			
Cabling and other networks	0.0	4	0.0
Internet	1,500 * 7 (Month)	4	42000.0

			Total = 42,000.0
5. Meeting Cost			
Transportation	300* 7 (Month)	4	8400.0
Meal	500*7 (Month)	4	14,000.0
			Total= 22,400.0
6. Software Cost			
User License	300 * 7(Month)	4	8400
Database	0.0	4	0.0
Operating System	0.0	4	0.0
			Total= 8400.0
7. Thesis Book Printing			

Cost			
Page Printing	10.0(per page)		5000.0
Cover page	1,000.0	1	1000.0
Binding	600.0	1	600.0
			Total=6500.0
		Grand Total	**79,300.0**

Table 2: Cost Estimation.

Chapter 4: Results and Findings

4.1 VGG-16 Result and Analysis

Model	Training accuracy	Validation accuracy	Number of epochs
VGG-16	94	81	30

Table 3: VGG-16 Result

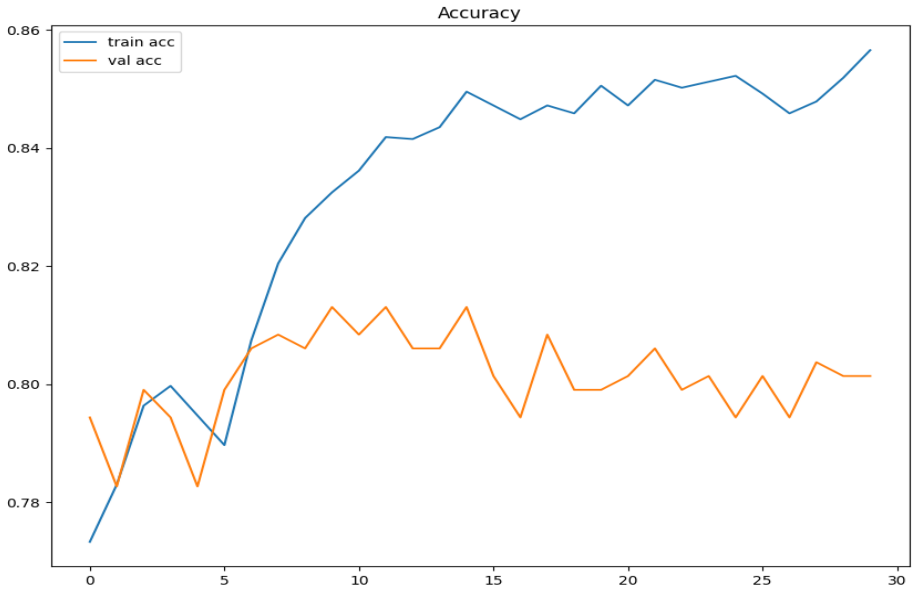


Figure 9: VGG-16 model's Accuracy graph

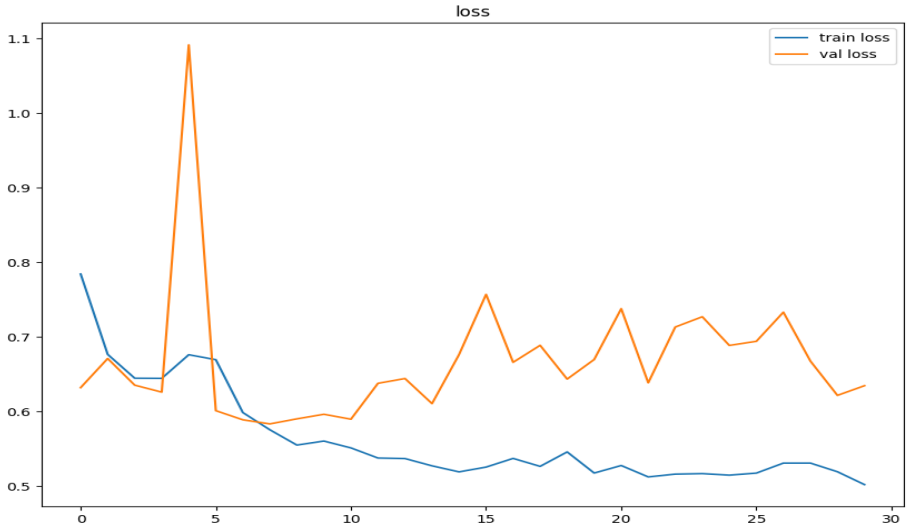


Figure 10: Loss of VGG-16

For our dataset, VGG16 performed well and produced 81 percent validation accuracy. In Fig:9, we see that there is a gap between the accuracy of training and the accuracy of validation. It makes low noise and smoother over training time. Fig:10 indicates VGG16's training loss heading to the lower loss.

4.2 VGG-19 Result and Analysis

Model	Training accuracy	Validation accuracy	Number of epochs
VGG-19	85	80	30

Table 4: VGG-19 Result

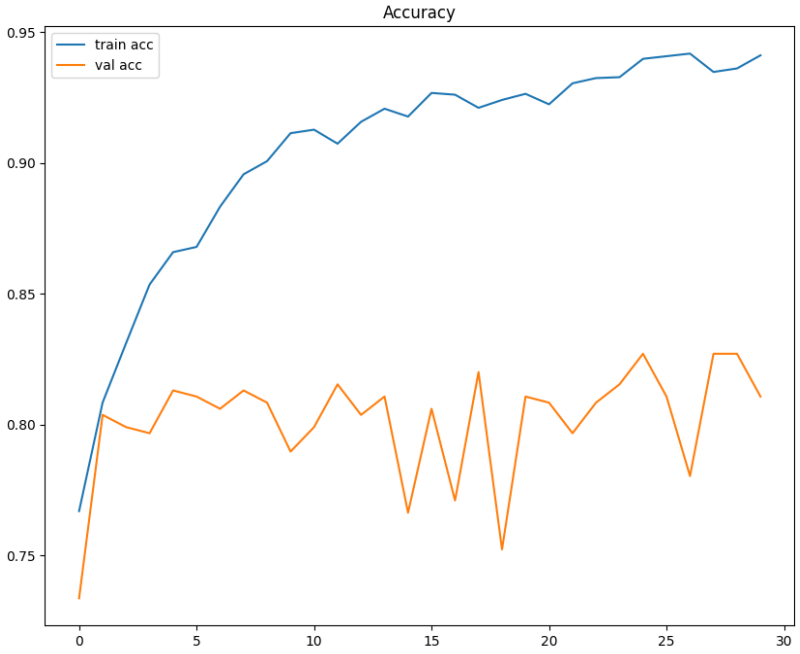


Figure 11: VGG-19 model's Accuracy graph

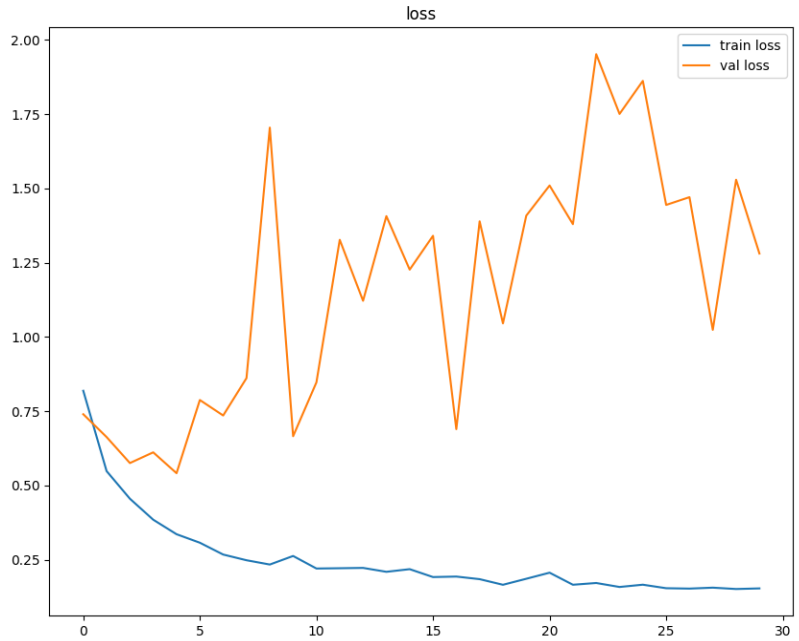


Figure 12: Loss of VGG-19

For our dataset, our VGG16 model performed well and produced a validation precision of 97 percent. We show in Fig 11. That accuracy of training and accuracy of validation are satisfactory because it allows a little bit of noise and prevents overfitting successfully. Fig: 12 shows that the loss of training is significantly decreasing.

4.3 Inception- V3 Result and Analysis

Model	Training accuracy	Validation accuracy	Number of epochs
Inception- V3	82	81	30

Table 5: Inception-V3 Result

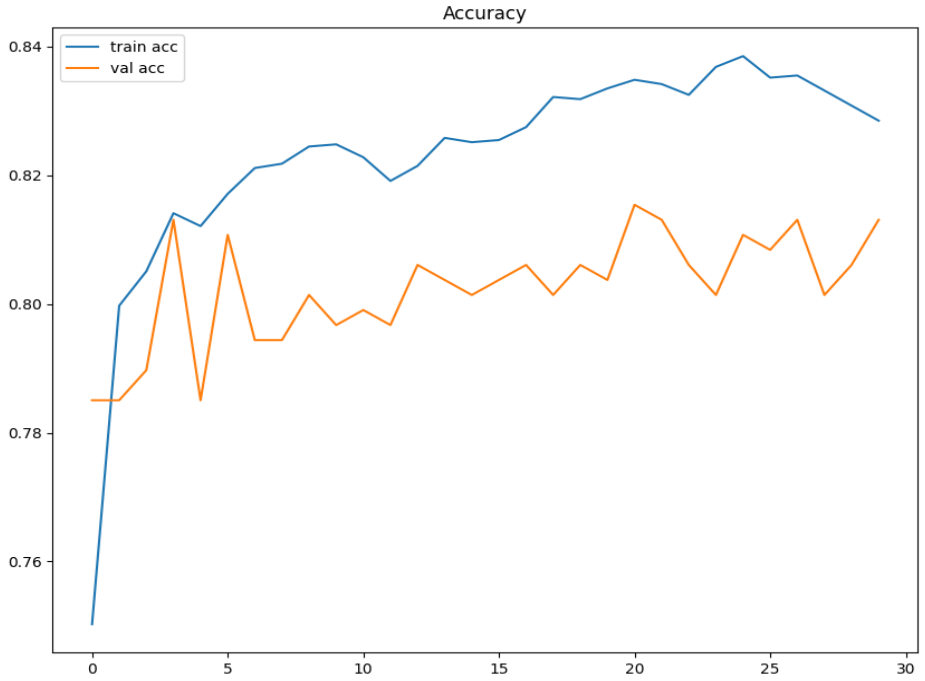


Figure 13: Inception-V3 model's Accuracy graph

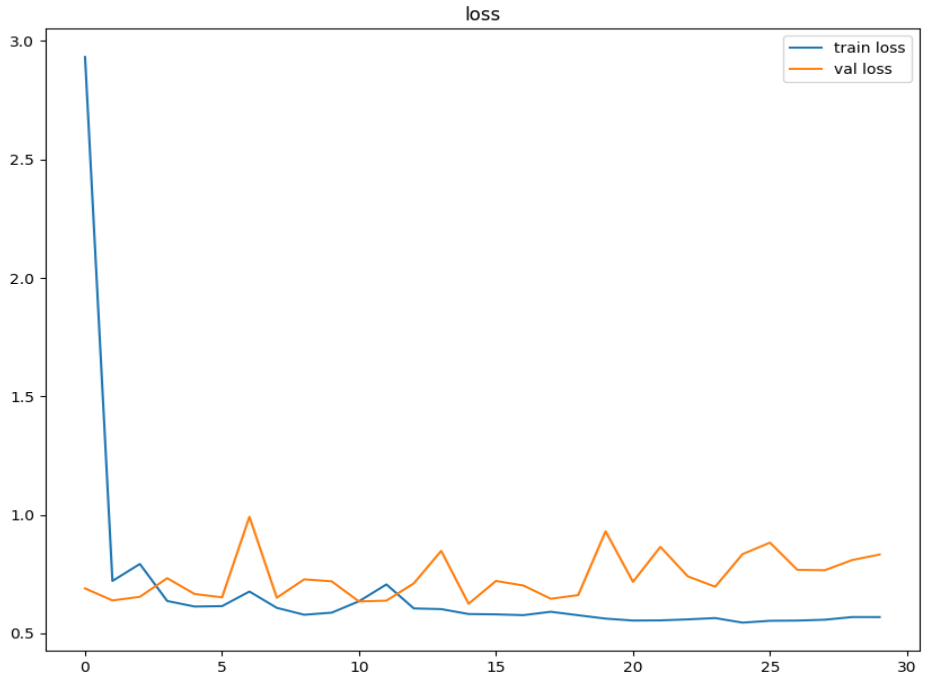
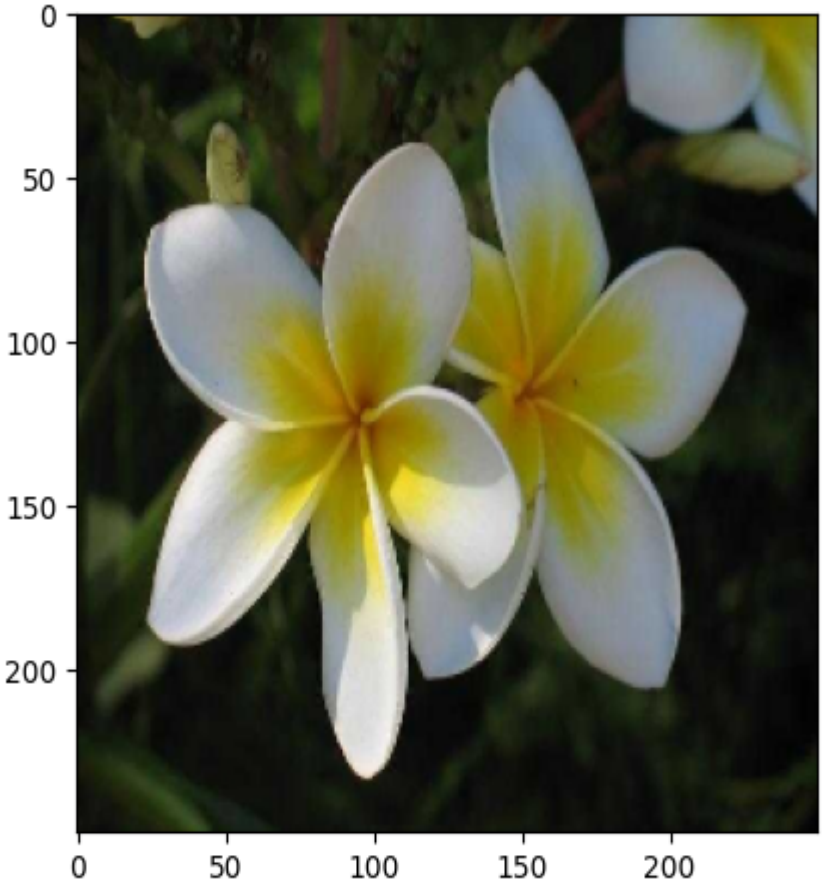


Figure 14:Loss of Inception- V3

For our dataset, Inception3 gives a reasonable validation precision of 81 percent. But we can note in Fig:13 that training accuracy is good and that training accuracy and validation accuracy are slowly growing. Accuracy is not fulfilled slowly but according to the validation. Train loss is gradually decreasing in 14.

4.4 Flower state prediction

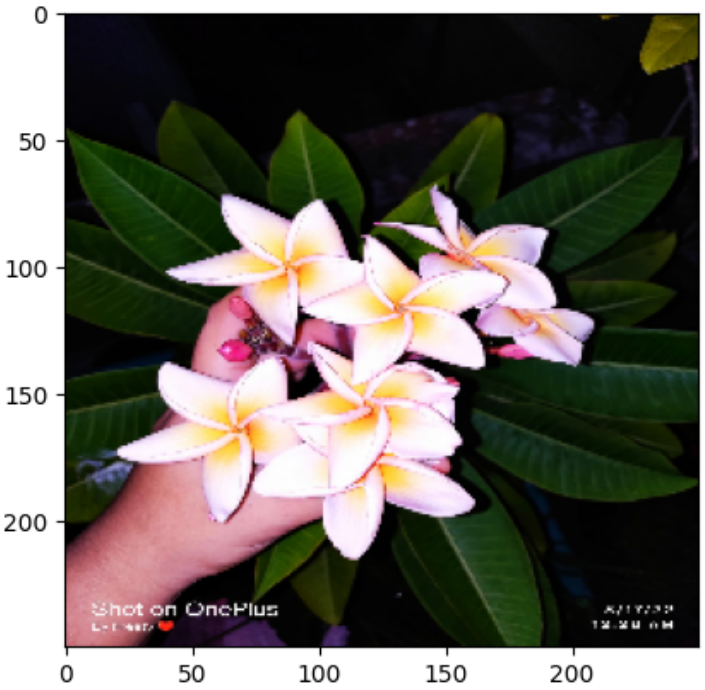
We ensembled VGG-16, VGG-19, and Inception V3, we got 81% accuracy from VGG-16,80% from VGG-19, and 80% from inception v3.finally our ensemble method gives us almost 81% accuracy. Based on this performance, we tried to predict the plumeria flower state.




```
Out[40]: 'Bloom'
```

Figure 15: Predicted Bloom

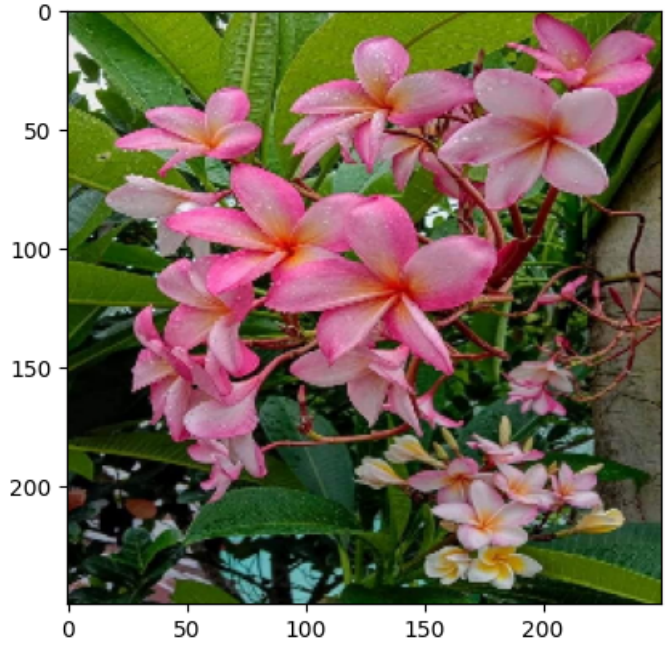
Our Ensemble method correctly predicted this flower is blooming. During prediction, this picture produced three arrays for model 1. These are 0.85719335, 0.07755615and 0.06525049. Model 2 also made three arrays, these are 9.9997365e-01, 2.3759167e-08 and 2.6395877e-05. Model 3 also gives us three arrays, which are 0.84740424, 0.08406211 and 0.06853368. Then we used the amax function to get the highest value which is 0.99997365. Similarly,fig.15 correctly predicts bloom.




```
Out[45]: 'Bloom'
```

Figure 16: Predicted Bloom

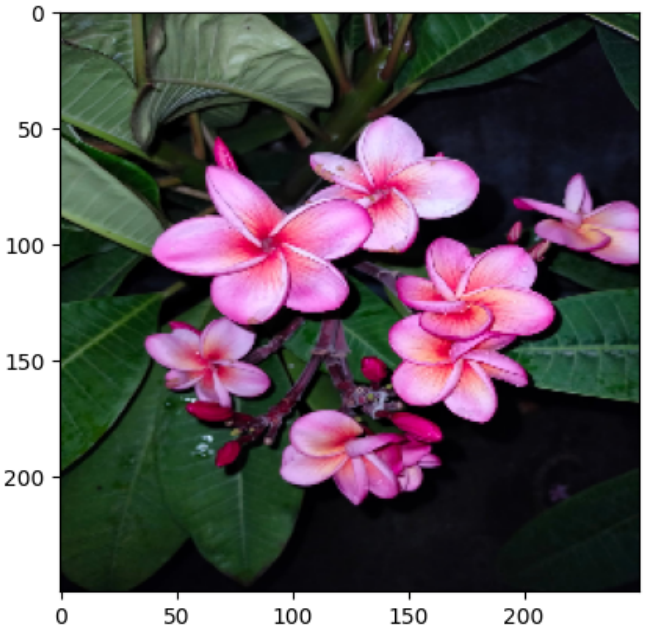
Our Ensemble method correctly predicted this flower is blooming. During prediction, this picture produced three arrays for model 1. These are 0.85719335, 0.07755615 and 0.06525049. Model 2 also made three arrays, these are 9.9968207e-01, 2.4118574e-06 and 3.1557604e-04. Model 3 also gives us three arrays, which are 0.84740424, 0.08406211, 0.06853368. Then we used the amax function to get the highest value which is 0.99968207. Similarly,fig.16 correctly predicts bloom.




```
Out[36]: 'Bloom'
```

Figure 17: Predicted Bloom

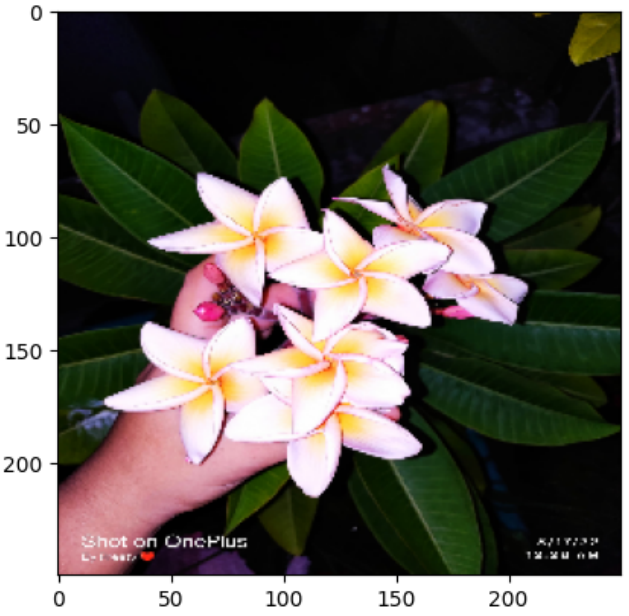
Our Ensemble method correctly predicted this flower is blooming. During prediction, this picture produced three arrays for model 1. These are 0.85719335, 0.07755615, 0.06525049. Model 2 also made three arrays, these are 9.9887437e-01, 1.6145277e-05, 1.1094540e-03. Model 3 also gives us three arrays, which are 0.84740424, 0.08406211, 0.06853368. Then we used the amax function to get the highest value which is 0.99887437. Similarly, fig.17 correctly predicts bloom.




```
Out[42]: 'Bloom'
```

Figure 18: Predicted Bloom

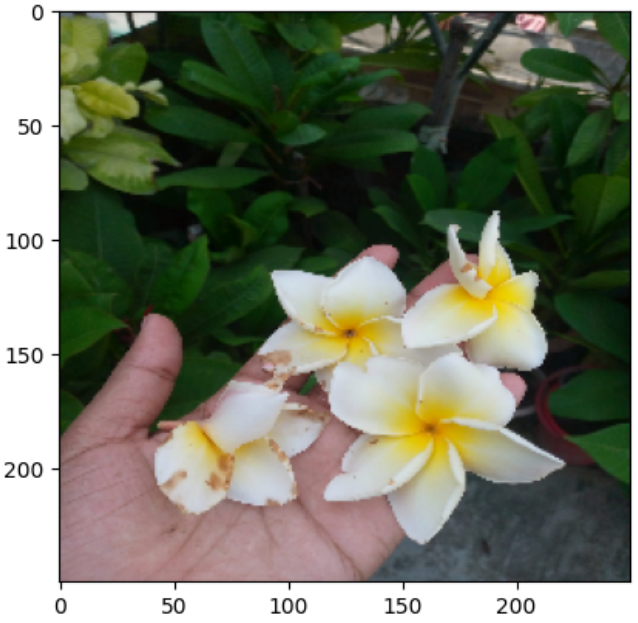
Our Ensemble method correctly predicted this flower is blooming. During prediction, this picture produced three arrays for model 1. These are 0.85719335, 0.07755615, 0.06525049. Model 2 also made three arrays, these are 9.9999583e-01, 1.1975727e-09, 4.1353082e-06. Model 3 also gives us three arrays, which are 0.84740424, 0.08406211, 0.06853368. Then we used the amax function to get the highest value which is 0.9999958. Similarly, fig.18 correctly predicts bloom.




```
Out[45]: 'Bloom'
```

Figure 19: Predicted Bloom

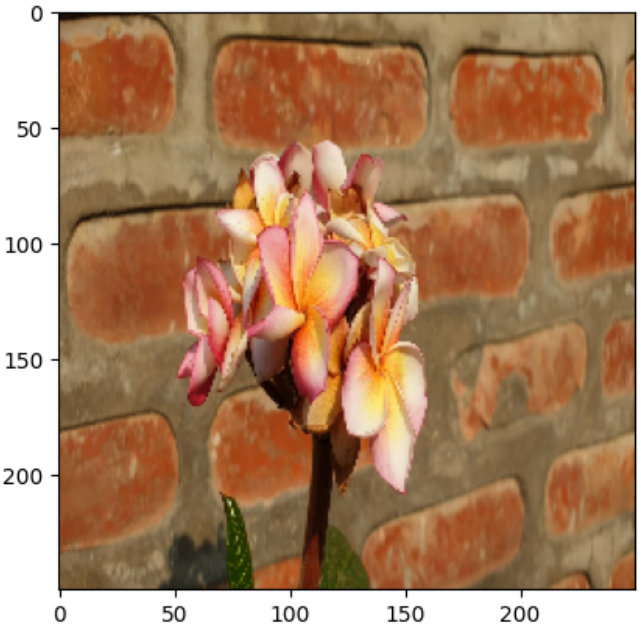
Our Ensemble method correctly predicted this flower is blooming. During prediction, this picture produced three arrays for model 1. These are 0.85719335, 0.07755615, 0.06525049. Model 2 also made three arrays, these are 0.9889044 , 0.00170918, 0.00938633. Model 3 also gives us three arrays, which are 0.84740424, 0.08406211, 0.06853368. Then we used the amax function to get the highest value which is 0.9889044. Similarly, fig.19 correctly predicts bloom.




```
Out[38]: 'Post Bloom'
```

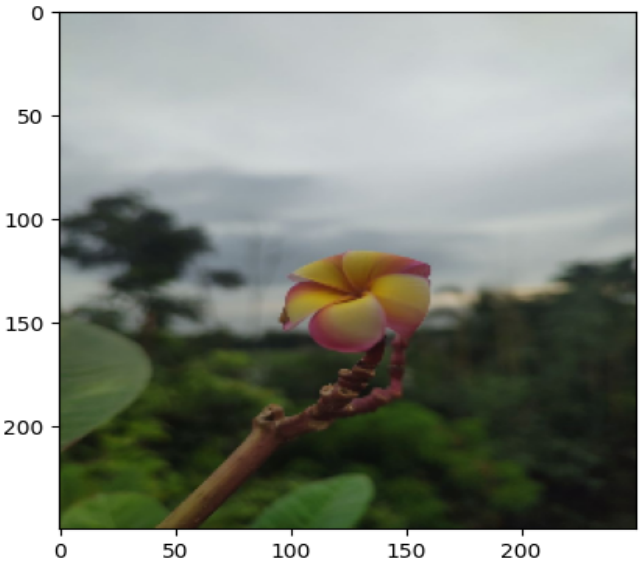
Figure 20:Predicted Post-Bloom

Like earlier Bloom predictions, during our Ensemble approach, Post-Bloom prediction generates three arrays from each model. We got a total of nine arrays from three models. These arrays are 0.85719335, 0.85719335, 0.07755615 and 0.06525049.1.3386421e-09, 1.0000000e+00 and 1.0606028e-16,0.84740424, 0.08406211and 0.06853368. After applying the amax function, our ensemble method correctly predicts this flower (fig.21) as post-bloom. Similarly, other flowers are also predicted as post-bloom.



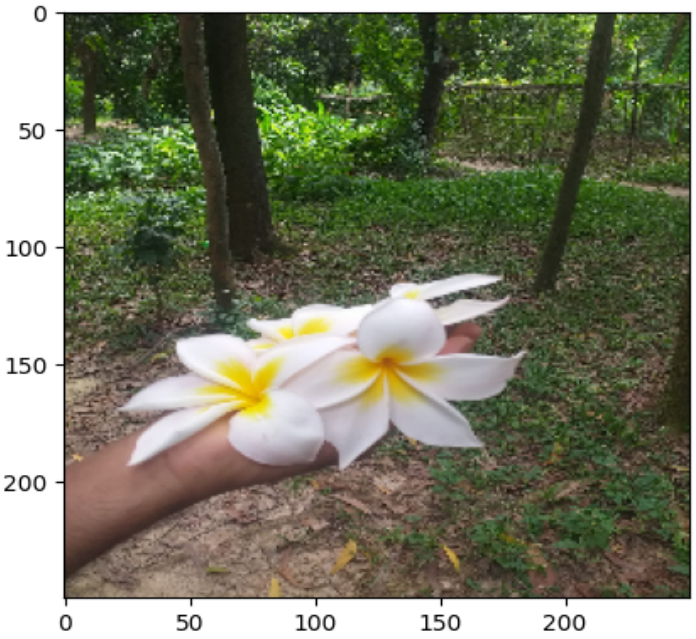

```
Out[62]: 'Post Bloom'
```

Figure 21: Predicted Post-Bloom



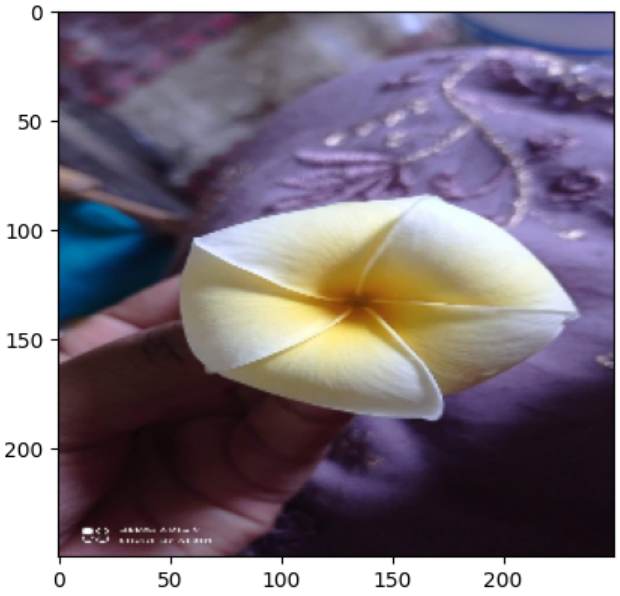

```
Out[87]: 'Post Bloom'
```

Figure 22: Predicted Post-Bloom



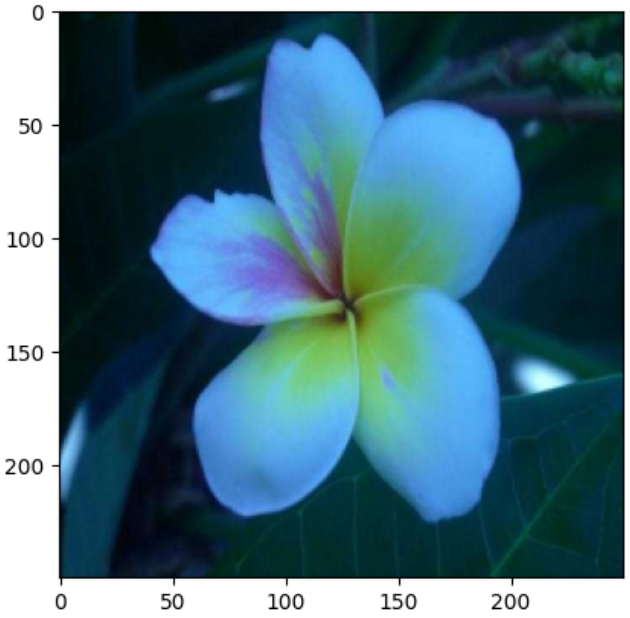

```
Out[189]: 'Post Bloom'
```

Figure 23: Predicted Post-Bloom



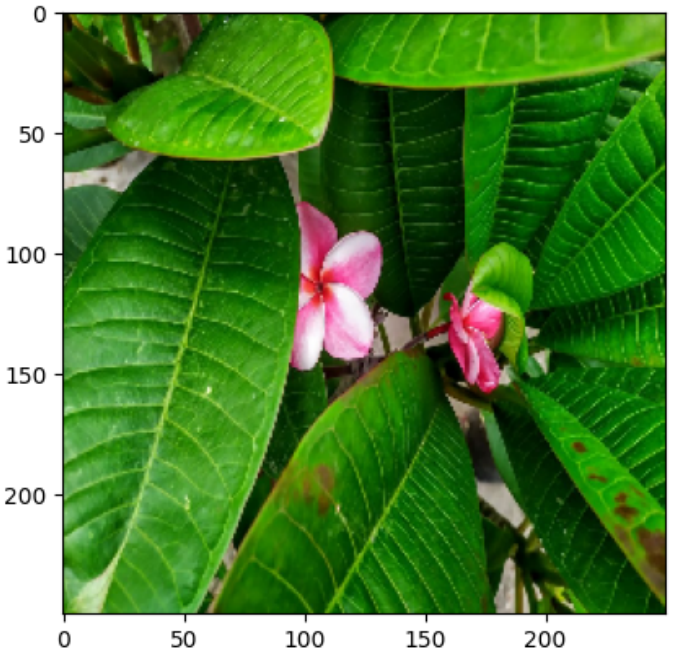

```
Out[131]: 'Post Bloom'
```

Figure 24: Predicted Post-Bloom



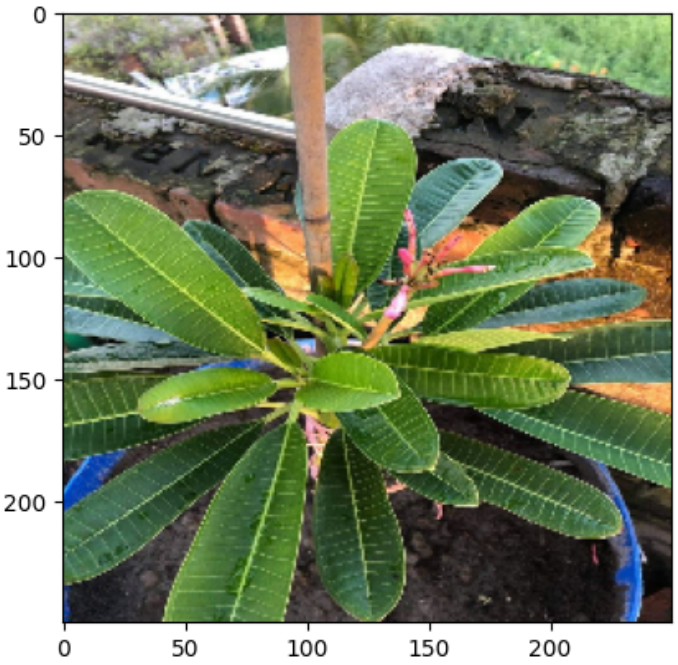
Out[162]: 'Post Bloom'

Figure 25: Predicted Post-Bloom



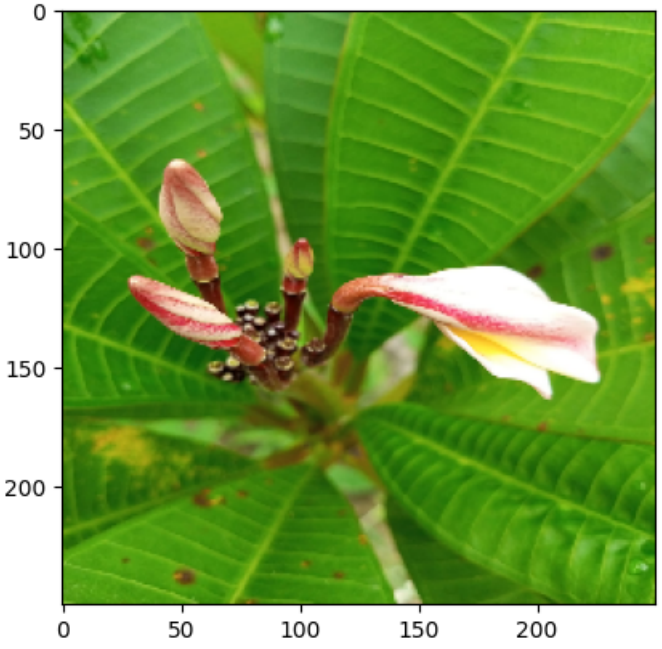
Out[78]: 'Pre Bloom'

Figure 26: Predicted Pre Bloom



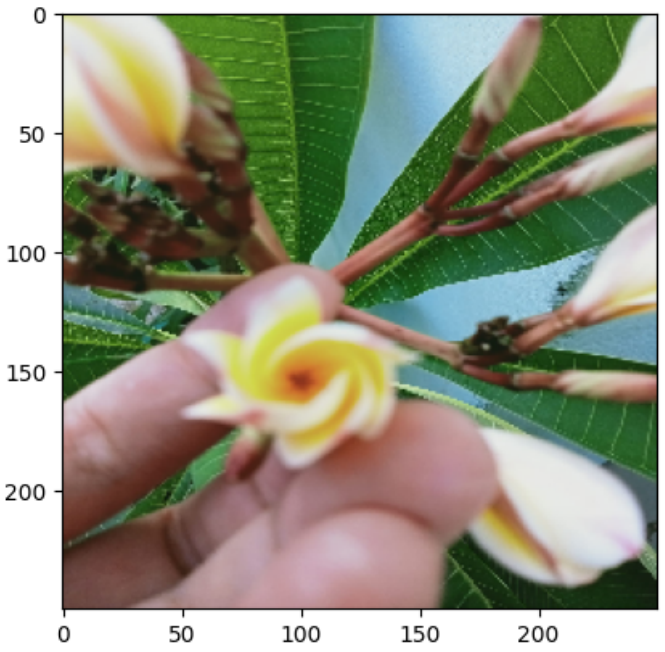

```
Out[79]: 'Pre Bloom'
```

Figure 27:. Predicted Pre Bloom




```
Out[90]: 'Pre Bloom'
```

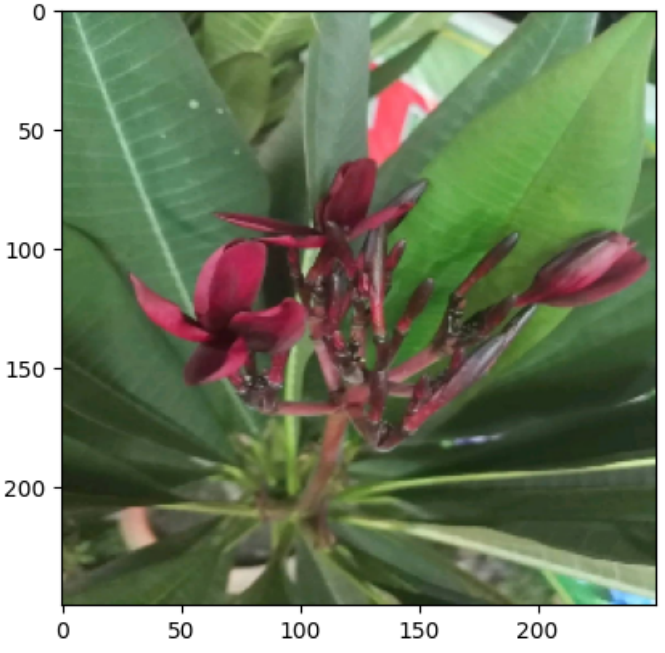
Figure 28: Predicted Pre Bloom



Out[98]: 'Pre Bloom'

Figure 29: Predicted Pre Bloom

Like previous Bloom predictions, during our Ensemble approach, Pre-bloom prediction also generates three arrays from each of the models. We got a total of nine arrays from three models. These arrays are 1.5615928e-05, 2.2710054e-11 and 9.9998438e-01,8.3960512e-12, 1.5426237e-27 and 1.0000000e+00,1.8306604e-27, 0.0000000e+00 and 1.0000000e+00. After applying the amax function, our ensemble method correctly predicts this flower(fig 17) as pre-bloom. Similarly, other flowers also generated a total of nine arrays from three different models. After applying the amax function, fig 29 also correctly predicted Pre Bloom.



Out[102]: 'Pre Bloom'

Figure 30: Predicted Pre Bloom

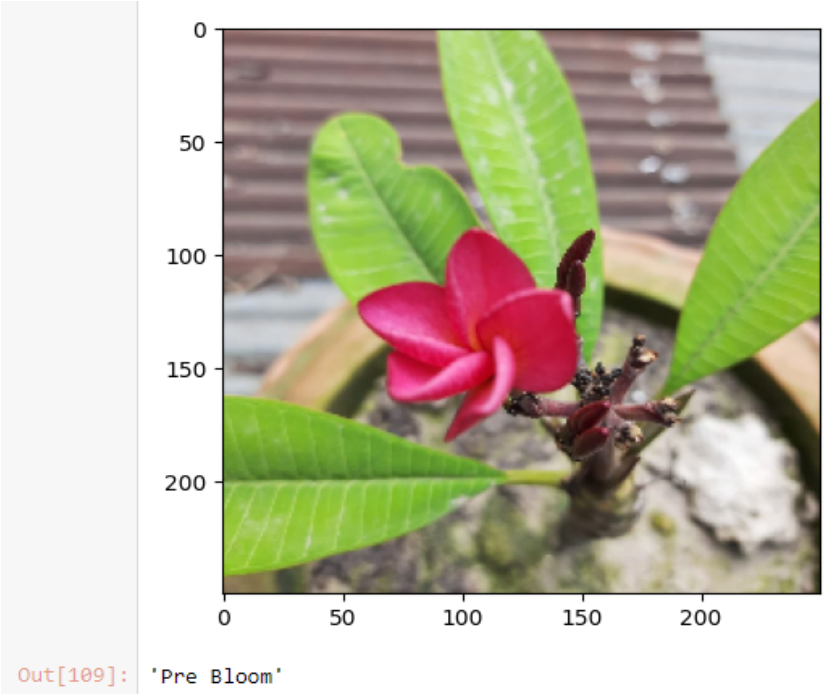
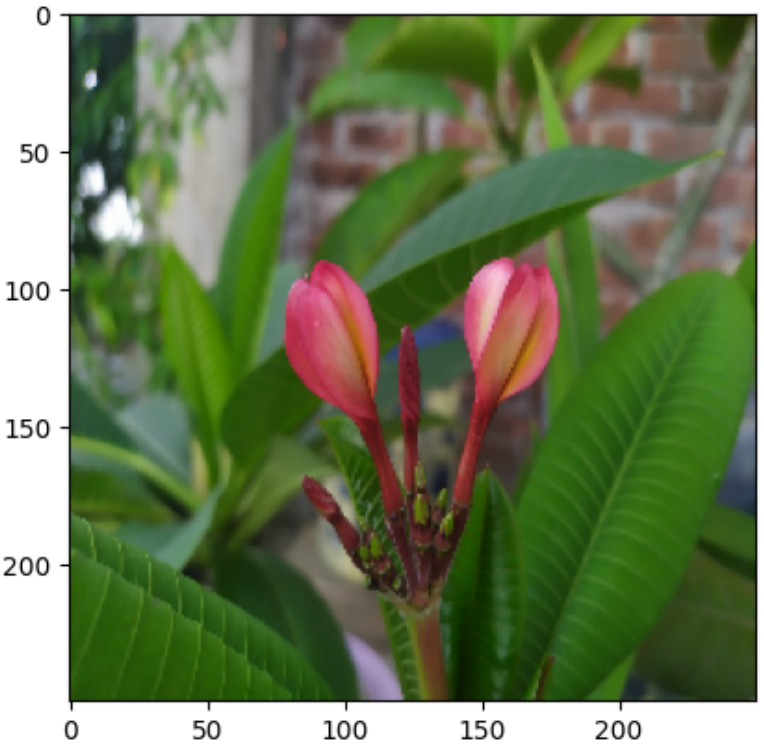


Figure 31: Predicted Pre Bloom

Like previous Bloom predictions, during our Ensemble approach, Pre-bloom prediction also generates three arrays from each of the models. We got a total of nine arrays from three models. These arrays are 0.26223204, 0.00397305 and 0.73379487,1.0856831e-03, 3.8107078e-08 and 9.9891424e-01,1.0688014e-02, 2.0018363e-09 and 9.8931199e-01. After applying the amax function, our ensemble method correctly predicts this flower(fig 17) as pre-bloom. Similarly, other flowers also generated a total of nine arrays from three different models. After applying the amax function, fig 31 also correctly predicted it as pre-bloom.Although we Ensembled 3 models and got 81% accuracy. For prediction we got the result from them and we only took the highest accuracy, we still have a gap of 19%. Because of the gap we got some errors when predicting. Here are some examples,

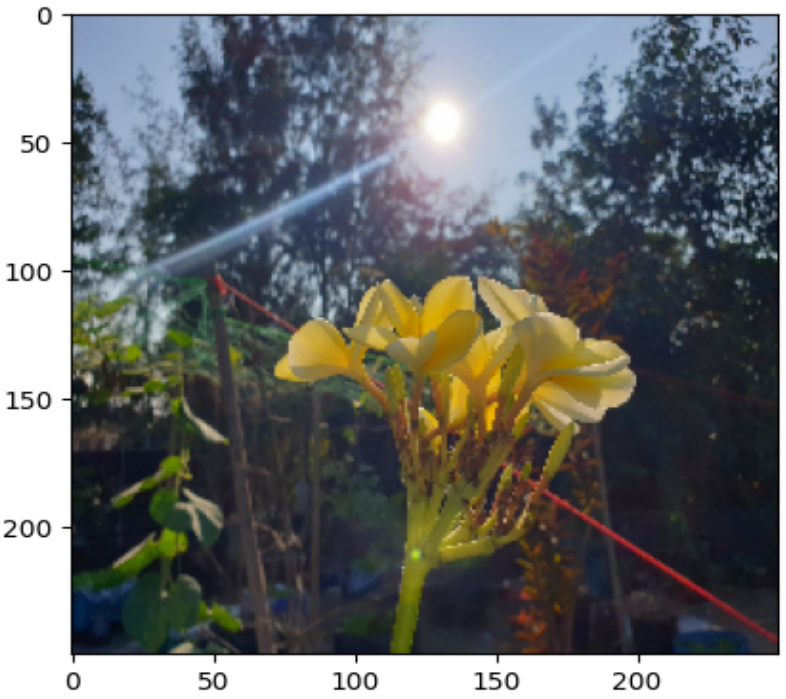

```
Out[186]: 'Bloom'
```

Figure 32: Should be pre bloom



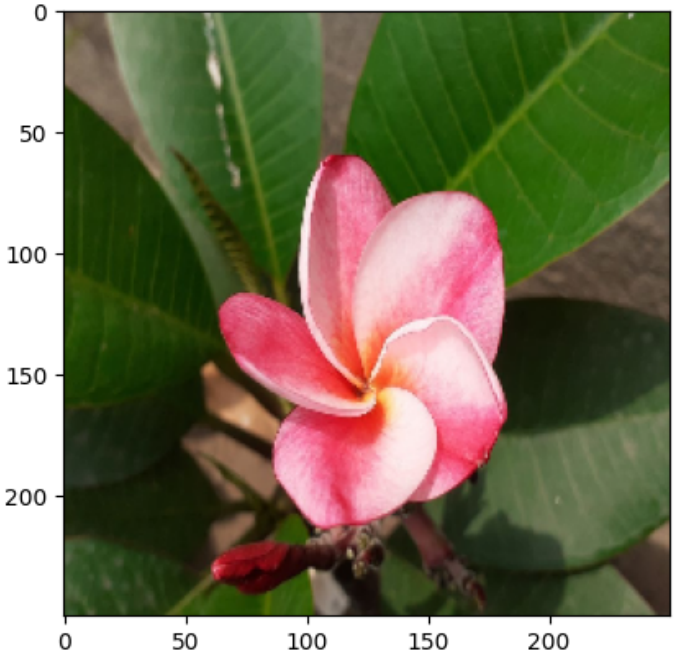

```
Out[157]: 'Bloom'
```

Figure 33:Should be pre bloom



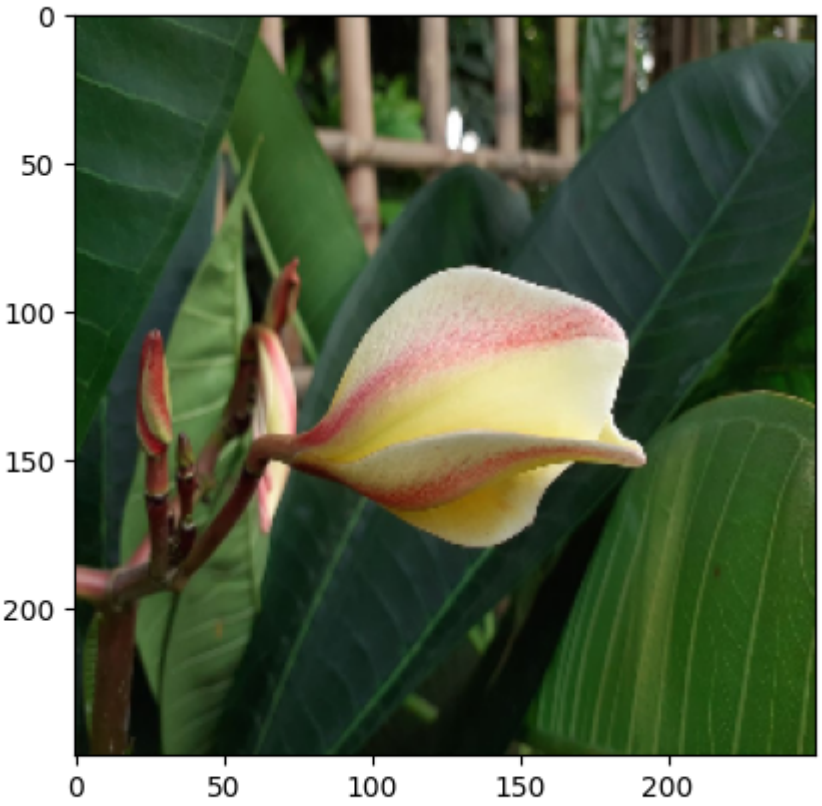

```
Out[196]:  'Bloom'
```

Figure 34:Should be pre bloom



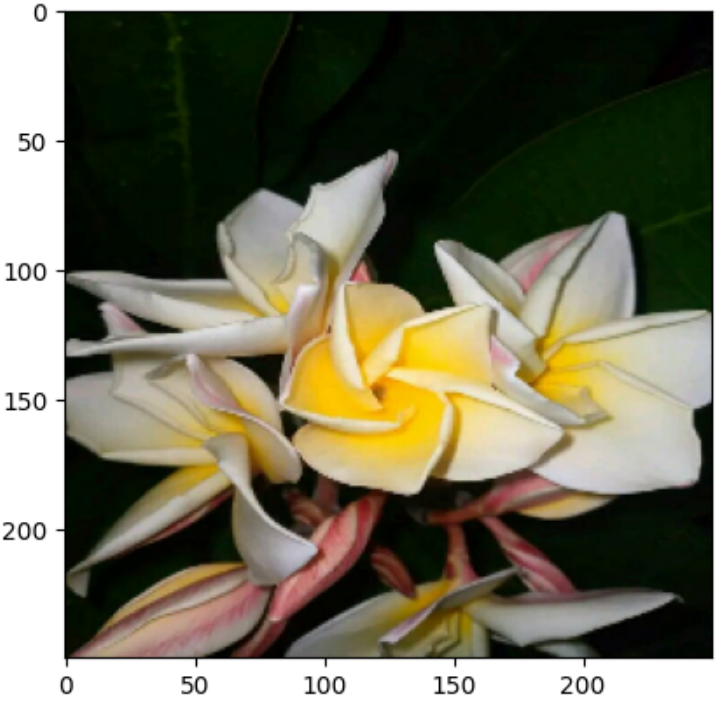

```
Out[205]:  'Bloom'
```

Figure 35:Should be pre bloom



Out[224]: 'Bloom'

Figure 36:Should be pre bloom

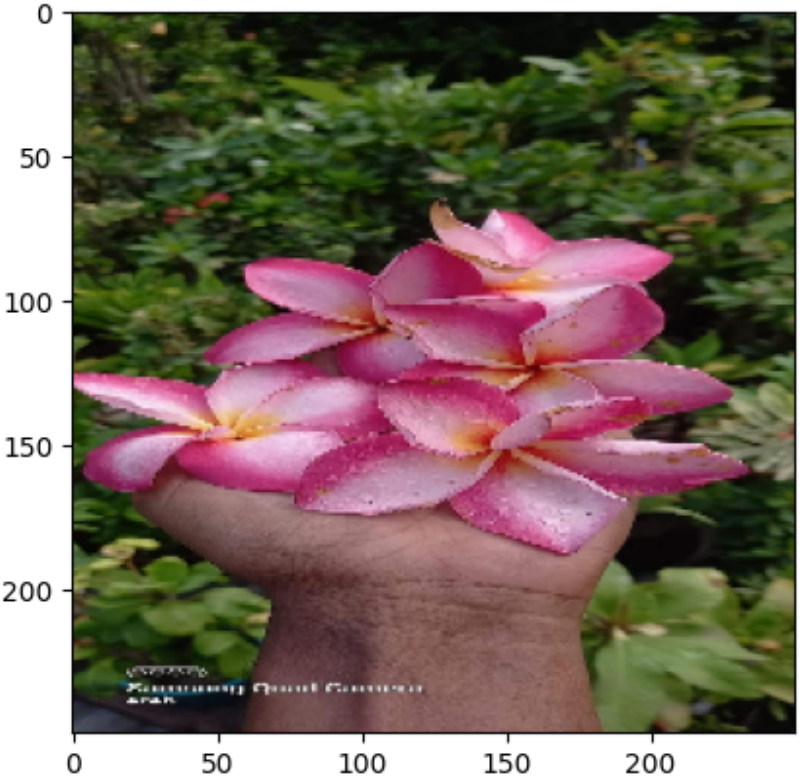


Out[122]: 'Pre Bloom'

Figure 37:Should be post bloom

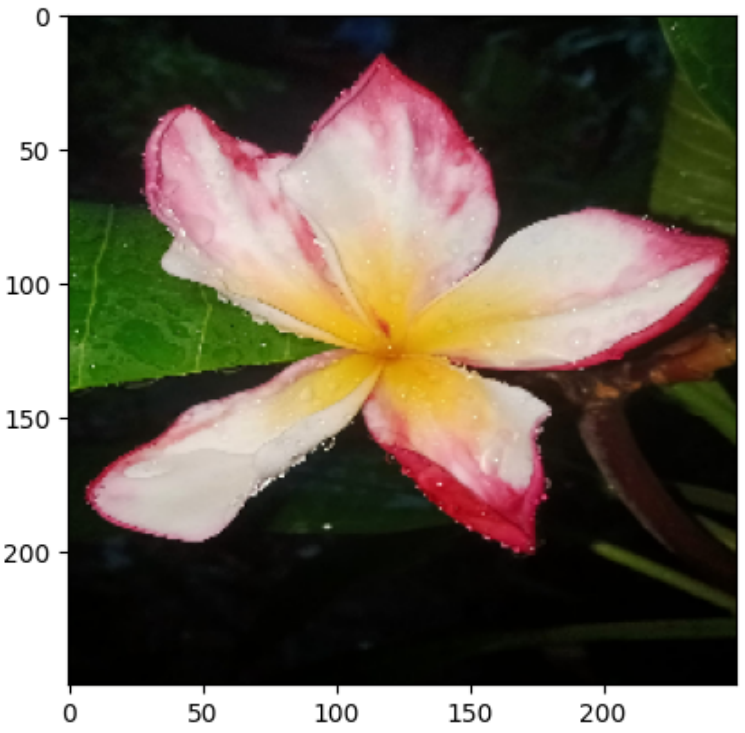

```
Out[105]:  'Pre Bloom'
```

Figure 38:Should be bloom



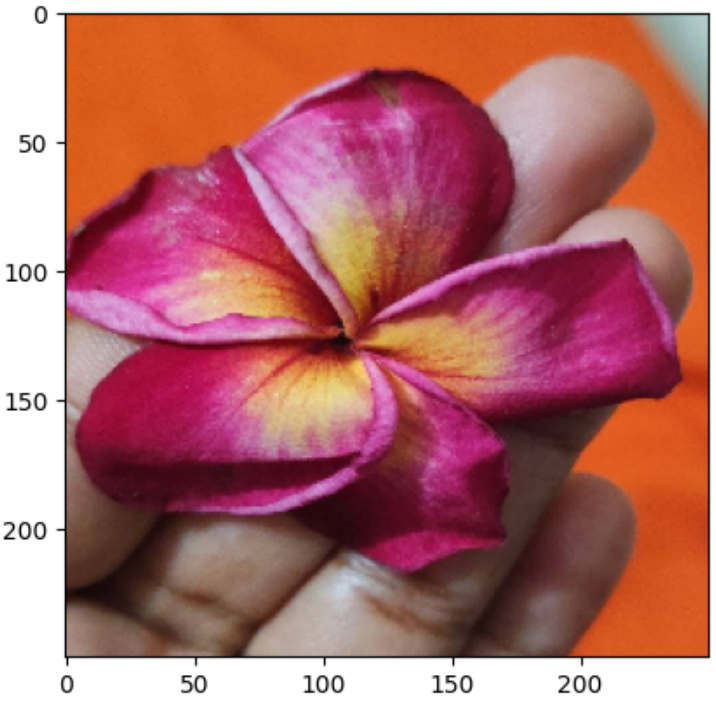

```
Out[158]:  'Bloom'
```

Figure 39:Should be post bloom



Out[137]: 'Bloom'

Figure 40:Should be post bloom



Out[209]: 'Bloom'

Figure 41:Should be post bloom

In Fig.(32-36)it is clearly in ‘Pre Bloom’ state but our method has predicted it as ‘Bloom’.
Fig 38 it predicted ‘Pre Bloom’ but it is in ‘Bloom’ state.
Fig (37,39-41) it predicted ‘Bloom’ but it is in ‘Post Bloom’ state.

Chapter 5: Discussion

Our main purpose was to find the state of a Plumeria flower. By using a complete data set of flowers, we trained them in three different CNN models which are VGG-16, VGG-19 and Inception V3. After that we used the Ensemble method to combine our predictions and found our final result. Our algorithm and structural diagram can recognize the flower states. It gives us about 81% of accuracy and correction. Till now many researchers have implemented varieties of algorithms to recognize a flower species. But we tried deep learning in this criteria to recognize the condition of a flower. Thus it is a bit of a different study.

We faced various unexpected issues like the data train gave poor results or prediction values were so low in number. As we had to fix them and repeat the whole process. All the time we had to check the predictions by changing the parameters. It took a lot of time to achieve our expected accuracy.

Chapter 6: Conclusion

As we have discussed earlier, we are trying to determine the state of a flower. So in our research work we have used three Convolutional layers and ensemble methods to recognize our data. We had an accuracy of 81%. Our target was fulfilled when we achieved our highest success. By changing different parameters we tried to acquire this result.

References

T. Akiyama, Y. Kobayashi, Y. Sasaki, K. Sasaki, T. Kawaguchi and J. Kishigami, "Mobile Leaf Identification System using CNN applied to plants in Hokkaido," 2019 IEEE 8th Global Conference on Consumer Electronics (GCCE), 2019, pp. 324-325,https://doi.org/10.1109/GCCE46687.2019.9015298.

V. Rezende, M. Costa, A. Santos and R. C. L. de Oliveira, "Image Processing with Convolutional Neural Networks for Classification of Plant Diseases," 2019 8th Brazilian Conference on Intelligent Systems (BRACIS), 2019, pp. 705-710, https://doi.org/10.1109/BRACIS.2019.00128.

Xiaoling Xia, Cui Xu \& Bing Nan, "Inception-v3 for flower classification," 2017 2nd International Conference on Image, Vision and Computing (ICIVC), 2017, pp. 783-787 https://doi.org/10.1109/ICIVC.2017.7984661

Naranjo Torres, José , Mora, Marco , Hernández García, Ruber \& Barrientos, Ricardo , Fredes, Claudio, Valenzuela Keller, Andres. (2020). A Review of Convolutional Neural Network Applied to Fruit Image Processing. Applied Sciences. 10. 3443. 10.3390/app10103443.https://doi.org/10.3390/app10103443

Giraddi, S., Seeri, S., Hiremath, P. S., & G.N, J. (2020). Flower Classification using Deep Learning models. 2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE). https://doi.org/10.1109/icstcee49637.2020.9277041

Rezende, V. C., Costa, M., Santos, A., & de Oliveira, R. C. L. (2019). Image Processing with Convolutional Neural Networks for Classification of Plant Diseases. 2019 8th Brazilian Conference on Intelligent Systems (BRACIS). https://doi.org/10.1109/bracis.2019.00128

Zhao Jiantao, Chu Shumin. ``Research on Flower Image Classification Algorithm Based on Convolutional Neural Network" Journal of Physics: Conference Series, August 2021. https://doi.org/10.1088/1742-6596/1994/1/012034

Bozkurt, F. (2021). A Study on CNN Based Transfer Learning for Recognition of Flower Species. European Journal of Science and Technology, (32), 883-890. https://doi.org/10.31590/ejosat.1039632

Raisa Akter;Md Imran Hosen; (2020). CNN-based Leaf Image Classification for Bangladeshi Medicinal Plant Recognition. 2020 Emerging Technology in Computing, Communication, and Electronics (ETCCE) https://doi.org/10.1109/ETCCE51779.2020.9350900

Lin, Ping \& Chen, Yongming; (2018). "Detection of Strawberry Flowers in Outdoor Field by Deep Neural Network." 482-486.10.1109 ICIVC.2018.8492793. https://doi.org/10.1109/ICIVC.2018.8492793

M. Rajalakshmi,Dr. R.Kavitha,Dr. S. Karpagavalli, "Flower Classification Based On Deep Learning Using Tensorflow" Nat. Volatiles and Essent. Oils, 2021; 8(6): 5125-5131.

Parvathy and N. Rao and Shahistha Bai and Naeema Nazer and J Prof.Anju, FLOWER RECOGNITION SYSTEM USING CNN. International Research Journal of Engineering and Technology (IRJET). Volume: 07 Issue: 06 June 2020

Inthiyaz, Syed \& Madhav, B.\& Prasad, Ch. (2018)."Flower image classification with basket of features and multi-layered artificial neural networks." International Journal of Engineering and

Technology(UAE). 7. 642-650 http://dx.doi.org/10.14419/ijet.v7i1.1.10795

Rzanny, Michael \& Mäder, Patrick \& Deggelmann, Alice and Minqian, Chen and Wäldchen, Jana. (2019). Flowers, leaves or both? How to obtain suitable images for automated plant identification. Plant Methods. 15-77. https://doi.org/10.1186/s13007-019-0462-4

Rupinder Kaur, Anubha Jain, Sarvesh Kumar,Optimization classification of sunflower recognition through machine learning, Materials Today: Proceedings, Volume 51, Part 1,2022,Pages 207-211,ISSN 2214-7853. https://doi.org/10.1016/j.matpr.2021.05.182

Hiary, H., Saadeh, H., Saadeh, M., \& Yaqub, M. (2018). Flower classification using deep convolutional neural networks. IET Computer Vision, 12(6), 855–862. https://doi.org/10.1049/iet-cvi.2017.0155

Yazdizadeh, Ali, Patterson, Zachary \& Farooq, Bilal. (2019). Ensemble Convolutional Neural Networks for Mode Inference in Smartphone Travel Survey. IEEE Transactions on Intelligent Transportation Systems. 21. 1-8. 10.1109/TITS.2019.2918923 https://doi.org/10.1109/TITS.2019.2918923

Raj, Mohan \& Chakkaravarthy, Sibi \& Vaidehi, V.. (2019). Ensemble of Convolutional Neural Networks for Face Recognition: IC3 2018.http://dx.doi.org/10.1007/978-981-10-7895-8_25

Chen, C., Yan, Q., Li, M., \& Tong, J. (2019, July). Classification of blurred flowers using convolutional neural networks. In Proceedings of the 2019 3rd International Conference on Deep Learning Technologies (pp. 71-74). https://doi.org/10.1145/3342999.3343006

Sujatha, R.; Chatterjee, Jyotir Moy; Jhanjhi, NZ; Brohi, Sarfraz Nawaz (2021). Performance of deep learning vs machine learning in plant leaf disease detection. Microprocessors and Microsystems, 80(), 103615–. https://doi.org/10.1016/j.micpro.2020.103615

Nandhini, S. Kulandasamy, Ashokkumar. (2021). Improved crossover based monarch butterfly optimization for tomato leaf disease classification using convolutional neural network. Multimedia Tools and Applications. 80. 1-28. .https://doi.org/10.1007/s11042-021-10599-4

Mujahid, Muhammad \& Rustam, Furqan \& Álvarez, Roberto \& Mazón, Juan \& De la Torre Díez, Isabel \& Ashraf, Imran. (2022). Pneumonia Classification from X-ray Images with Inception-V3 and Convolutional Neural Network. Diagnostics. 12. 1280. https://doi.org/10.3390/diagnostics12051280.

Arun Malik, Gayatri Vaidya, Vishal Jagota, Sathyapriya Eswaran, Akash Sirohi, Isha Batra, Manik Rakhra, Evans Asenso, "Design and Evaluation of a Hybrid Technique for Detecting Sunflower Leaf Disease Using Deep Learning Approach", Journal of Food Quality, vol. 2022, Article ID 9211700, 12 pages, 2022.https://doi.org/10.1155/2022/9211700

Kalita, Jugal; Balas, Valentina Emilia; Borah, Samarjeet; Pradhan, Ratika (2019). [Advances in Intelligent Systems and Computing] Recent Developments in Machine Learning and Data Analytics Volume 740 (IC3 2018) || Ensemble of Convolutional Neural Networks for Face Recognition. , 10.1007/978-981-13-1280-9(Chapter 43), 467–477 http://dx.doi.org/10.1007/978-981-13-1280-9_43

Rajayogi, J R; Manjunath, G; Shobha, G (2019). [IEEE 2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS) - Bengaluru,

India (2019.12.20-2019.12.21)] 2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS) - Indian Food Image Classification with Transfer Learning. , (), 1–4. https://doi.org/10.1109/csitss47250.2019.90310

Appendix

ML= Machine Learning
DL = Deep Learning
ReLU = Rectified linear unit
VGG = Visual Geometry Group
CNN = Convolutional neural network
SGD = Stochastic Gradient Descent
Conv2D = Convolutional 2D
FC = Fully connected layer
ECNN= Ensemble Convolutional neural network
RGB =Red, Green, Blue

Here is our project code:

https://github.com/Faysal179038/Flower-classification-using-CNN-models-Ensemble-method