Identification of Medicinal Plant Using Deep Learning

A major project report submitted in partial fulfillment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Yash Tyagi (201556) & Geetanjali Singh (201560)

Under the guidance & supervision of

Dr. Ekta Gandotra (Associate Professor)

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

CERTIFICATE

This is to certify that the work which is being presented in the project report titled **'Identification of Medicinal Plant Using Deep Learning'** in partial fulfillment of the requirements for the award of the degree of **B.Tech in Computer Science And Engineering / Information Technology** and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by **Yash Tyagi (201556) and Geetanjali Singh (201560)** during the period from August 2023 to May 2024 under the supervision of **Dr. Ekta Gandotra (Associate Professor)**, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat)

I also authenticate that I have carried out the above mentioned project work under the proficiency stream Cloud Computing.

Submitted by:
Yash Tyagi (201556)
Geetanjali Singh (201560)

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

Supervised by:
Dr. Ekta Gandotra
Associate Professor
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat

CANDIDATE'S DECLARATION

We hereby declare that the work presented in this report entitled **'Identification of Medicinal Plant Using Deep Learning'** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Ekta Gandotra** (Associate Professor, Department of Computer Science & Engineering and Information Technology).

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

(Student Signature with Date)
Yash Tyagi
201556

(Student Signature with Date)
Geetanjali Singh
201560

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

(Supervisor Signature with Date)

Dr. Ekta Gandotra
Associate Professor
Computer Science & Engineering / Information Technology
Dated:

ACKNOWLEDGEMENT

We are highly indebted to all the members of the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology for their steerage and constant supervision while providing necessary data related to the project and additionally for their support in finishing off the project. We would also like to express our gratitude towards Dr Ekta Gandotra, Associate Professor, as our project Supervisor, for her kind cooperation and encouragement which helped us in the completion of this project and for giving us such attention and time.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF GRAPHS

LIST OF ABBREVIATIONS

1. CNN: Convolutional Neural Network
2. VGG19: Visual Geometry Group 19
3. ResNet50: Residual Network
4. SVM: Support Vector Machine
5. DLNN: Deep Learning Neural Network
4. GPU: Graphical Processing Unit
5. CPU: Central Processing Unit
6. RAM: Random Access Memory
7. AI: Artificial Intelligent
8. ReLU: Rectified Linear Unit
9. RMSProp: Root Mean Square Propagation

ABSTRACT

The rich diversity of plant species, with the potential to unlock numerous health benefits, has been a source of fascination for herbalists, researchers, and healthcare workers. Their exploration aimed to study the utilization, therapeutic properties, and diverse applications of these herbal plants. However, the task of identifying plants continued to be demanding and challenging.

The existing solutions, such as manual identification and chemical analysis, proved ineffective and inaccurate. Manual labor was time-intensive and insufficient due to the complexity and diversity of flora. Traditional methods of identification were prone to human error, while chemical methods were limited by resources, posing serious health issues raises the need to automate the process of identification of medicinal plants.

Our project explores the deep learning branch of artificial intelligence and proposes an ensemble learning technique to quickly identify plants by analyzing the images of their leaves. The dataset consists of 45 classes. we used pre-trained neural networks like VGG19, ResNet50, InceptionV3, Xception, MobileNetV2, and DenseNet121 through a method called transfer learning. These models were already trained to extract the features and understand the images. Then the models were trained on a specific dataset by extracting the features from input images and the softmax layer connected to the dense layer was used as a classifier.

The proposed classifier was based on the stacking of the top three models and the output was used as the final classifier. The model proposed outperformed the pre-trained models like VGG19, ResNet50, InceptionV3, Xception, MobileNetV2, and DenseNet121 by achieving an accuracy of 98.98% on the test dataset, and a macro average of 99%, a weighted average of 99% of each class.

Chapter 1: Introduction

1.1 Introduction

The rich diversity of plant species with the potential to discover many health benefits has been a source of fascination for herbalists, researchers and healthcare workers to study their utilization, therapeutic properties and diverse applications. 80% of the world's population relied on traditional medicine for their primary health care needs and a major part of traditional therapy involved the use of plant extracts [1]. A total of 88 plant species from 44 families and 80 genera were identified as medicines and leaves were the most frequently used part [2].

The main difficulty was identifying more efficient and effective approaches compared to previous outdated and cost-effective approaches. So, to achieve our desired result we devised algorithms aimed at speeding up the process by conducting a cost-effective search for medicinal plants.

The convergence of computer programs with science opens a new chapter for exploration. This project aims to use deep learning algorithms to develop an efficient system for the automatic recognition of herbal plants to enable us to create reliable equipment to assist professionals in cataloguing these plants.

The main significance of this initiative is that field surveys are quite challenging in hilly regions due to rapid climate change so, cutting-edge technologies like deep-learning embedded with cloud computing can aid in the distribution of plants in these remote areas. Thus, our work fosters a cross-cultural exchange of knowledge by integrating technologies to promote mutual learning and reduce the risk of overharvesting. Also, we can help researchers by enabling them to focus on more modern aspects as models like deep learning can speed up the process by shifting the manual labour towards image identification and characterization of plants.

We utilized publicly available datasets from Kaggle and Mendeley. After applying preprocessing techniques to improve the quality of data, dense layers were introduced into our model. To further refine the models, we used the Adam optimizer and SoftMax activation function. Also, pre-trained models like VGG19, ResNet50, InceptionV3, Xception, MobileNetV2, and DenseNet121 were employed. Out of all the models, Densenet121 proved to be the best one with the highest accuracy of 97.81%. After this, the best-performing models were stacked together by ensemble techniques to make a new model.

The significance of this work lies in streamlining the process by making it more user-friendly and accessible to everyone who demands and needs it. We desire to cultivate and nurture a relationship between nature and technology through real-time identification. Thus, to sum up, we initiated this project to help local communities and scholars in fostering biodiversity preservation by intertwining technology.

1.2 Problem Statement

Medicinal plants have always played a major role in traditional systems of medicine to treat a wide range of ailments. However, accurate identification is a challenging task with significant uses in pharmacy, healthcare and research fields. We need to spot these plants to ensure safe utilization.

The lack of automatic detection leads to potential health risks and the diversity of medicinal flora is another major challenge, task become more rigorous as manual labour is an effective process which demands expertise in botany. So, we plan this project to develop a well-defined model to separate plants from images of leaves, flowers and all other distinct parts.

Another challenge includes large variability as these herbal plants can exhibit diversity in their visual characteristics like their development stage, illumination conditions and
influences from their surrounding environment, So, we need to generalize well across these variations. Another problem involves extracting relevant features like color, texture and shape from plant images that are discriminative for different plant species.

Also, there is a need to avoid the impact of the model's performance, especially on minority classes as some species could be more prevalent in the dataset than others, leading to class imbalance. Also, it is essential to fine-tune the models for specific tasks of identification. It might be difficult to distinguish between species that might have similar leaves so, we need to construct a model which can capture subtle differences and avoid misclassification. A lack of guided research with limited resources may worsen the situation leading to bad results.

By solving all the above problems, we plan to develop an application to aid in the authenticity of herbal products by employing those techniques which can help us to succeed in our mission to revolutionize the medicinal field to adapt to diverse growth stages and environmental conditions.

1.3 Objectives

1) To propose a customized model that can efficiently identify the different medicinal plant species using deep learning algorithms by intricate features from images and comparing the proposed model with different existing models.

2) Develop a user-friendly interface allowing users to capture plant images easily and provide real-time identification results that can be used for a better understanding of their properties and uses.

1.4 Significance and Motivation of the Project Work

This project is quite significant in several areas ranging from the conservation of flora and fauna to healthcare facilities. One of its major contributions lies in the accurate identification of plants which have medicinal and herbal value which is crucial for the preservation of biodiversity to value and protect various resources of the plants. While habitat is under threat, we seek to empower technology with tools that can swiftly aid the efforts of ecosystems.

Furthermore, we will address the inefficiencies in old methods which hinder the progress of botanical research by harnessing the capabilities of data science tools like deep learning and artificial intelligence to make the process faster and more efficient. This efficiency accelerates the pace of scientific discovery by encouraging more widespread engagement to foster a greater understanding of these medicinal plants and their wide range of applications.

The motivation driving this project is deeply behind its evolving design of technology. The integration of trainable parameters represents a more practical approach which is innovative. The project seeks to demonstrate how advancement in techniques and technologies can be used to address challenges in the real world showcasing the potential collaboration between science and technical efforts. The motivation aligns with the societal drive to leverage cutting-edge tools to enhance various aspects of human life including healthcare and environmental conservation diversity.

If we look towards a more practical level, we are motivated by the potential impact it could have on public health as correct and effective use of these medicinal plants is not only relevant to the scientific community but also holds the power to direct implications to healthcare practitioners and the general public. A dependable tool has the potential to open up new avenues to healthcare in remote areas. By facilitating a modern approach, we aim and aspire to make herbal practices more readily available to contribute towards improved outcomes.

Additionally, we are also motivated to bridge the gap between manual work and technical work through various patterns and features to understand and appreciate traditional wisdom related to medicinal plants. The integration of these knowledge domains can lead to a more comprehensive approach, acknowledging the strengths of modern technologies.

Global importance is a key behind doing this work as this holds universal importance with different cultures and regions relying on traditional practices. We try to provide a more standardized method that can be applied across diverse geographical contexts. We will contribute to a shared understanding of their value and the need for sustainable methods to maximize the resources which is another major motivation in doing this project.

In conclusion, the project is motivated by a combination of technological resources, impact on public health, bridging knowledge gaps and global relevance. It represents a concerted effort to make a meaningful contribution to measure the effectiveness of the herbal value of these plants on a global scale.

1.5 Organization of Project Report

The project report is organized in a standard format and is organized as follows:

Chapter 1: Introduction - Provides a summary of the research topic, including background, research questions, aims, and study importance.

Chapter 2: Literature Survey - Conducts a thorough examination of available literature on the project issue. The literature review analyses prior research studies, theories, and frameworks relating to the issue and finds gaps in the literature that the present study attempts to fill.

Chapter 3: System Development - Describes the project's research and research design. This chapter describes the study's data-gathering procedures, data processing methodologies, data analysis, and assessment methods.

Chapter 4: Testing - Presents and analyses the outcomes of the experiments carried out. This chapter offers a full analysis of the acquired data, a description of the experimental setup, and an evaluation of the findings produced from the different signal processing and feature engineering approaches.

Chapter 5: Results - Provides results and outcomes of the project work

Chapter 6: Conclusion and Future Scope - Provides a summary of the research findings, examines the study's relevance, and explains the research's contributions. This chapter also discusses the study's weaknesses and makes recommendations for further research.

Chapter 2: Literature Survey

2.1 Overview of Relevant Literature

Samreen Naeem et al [3] proposed the machine learning-based classification of medicinal species. In preprocessing, they cropped some parts of the leaf and transformed it into a grey-level format. They also performed a seed intensity-based edge/line detection utilizing the Sobel filter and drew observations for five regions. To optimize features, they employed a chi-square feature selection approach and selected fourteen optimized features. They achieved an accuracy classifier through perceptron which is multi-layered on six medicinal plant leaves which were 99.10% for Tulsi, 99.80% for Peppermint, 98.40% for Bael, 99.90% for Lemon balm, 98.40% for Catnip, and 99.20% for Stevia. Their study opened a new horizon in medicinal plant leaf classification.

J. Samuel Manoharan [4] proposed an algorithm majorly aimed at problems in datasets to improve the rate of detection. Their methodology is a two-step authenticated process to detect the medicinal properties of leaves. The proposed methodology improved the classification section for the detection rate to increase the production of Ayurveda provisions. This perfect automated system helped in identification without any human support in various enterprise sectors such as botanists, taxonomists, Ayurveda manufacturing companies, and Ayurveda practitioners.

Rahim Azadnia et al [5] present an automated system to recognize species of plants by analyzing the digital images of their leaves and extracting colour, shape, and texture features from the plant images. The trained model consists of 960 images of six groups of medicinal plants, with 160 images per group. the proposed system achieved an accuracy of 96.25% and can be used as a rapid and accurate identification tool for medicinal plants, which can be useful for researchers, farmers, and the pharmaceutical industry.

In this paper [6] Amgad Munner et al proposed an efficient and automatic classification system to recognize Malaysian herbs that would be used in medical or cooking areas. They also investigated various classifiers to build an efficient classifier. They employed two classifiers, namely Support Vector Machine (SVM) and Deep Learning Neural Network (DLNN) where SVM achieved 74.63% recognition accuracy, and DLNN achieved 93% recognition accuracy for both the experimental model and the developed mobile app.

Marsyita Hanafi et al [7] have done a review of recent studies on machine learning algorithms for plant classification using leaf images. The review includes the image processing methods used to detect leaves and extract important leaf features for some machine learning classifiers. These machine learning classifiers are categorized according to their performance when classifying leaf images based on typical plant features, namely shape, vein, texture, and a combination of multiple features. The leaf databases that are publicly available for automatic plant recognition are reviewed as well.

Paper [8] Raisa Akter et al propose a CNN-based Leaf Image Classification system for Bangladeshi Medicinal Plant Recognition. The system uses a three-layer convolutional neural network to extract high-level features for classification. The dataset of Bangladeshi medicinal plants was collected, preprocessed, and classified using a deep learning technique. The proposed architecture achieved a promising result of correctly identifying 71.3% of leaves, which is comparable to other existing methodologies. The paper concludes by mentioning the future direction of their work.

A. Kaya et al [9] demonstrate that transfer learning improved the overall performance of their models for plant classification. Two pre-trained CNNs for deep extraction of features and then applying classical machine learning algorithms for the classification model development. Transferred learning improved the overall performance of the deep learning model through the fine-tuning of pre-trained models providing the best classification performances compared to the other methods.

In paper [10] M. R. Dileep et al use a Deep Learning CNN model, to classify medicinal plants using leaf features such as shape, size, colour, and texture and a standard dataset for medicinal plants, commonly seen in Kerala. The proposed AyurLeaf model achieved a classification accuracy of 96.76%.

In paper [11] C. Sabarinathan et al focus on the use of a Convolutional Neural Network (CNN) to recognize medicinal plant leaves and showcase their uses. The authors propose that CNN can learn discriminative features to identify medicinal plant leaves, utilizing their shape qualities. The approach is compared against numerous prevalent handcrafted shape qualities, and the results on three large public leaf datasets show that the CNN prompts higher precision and consistency than the best in class.

In this paper [12] Amala Sabu et al present an automated system to recognize species of plants by analyzing the digital images of their leaves. The system uses traditional knowledge from older generations to identify medicinal plants. The proposed system contains four stages: Image acquisition, Image pre-processing, Feature Extraction, and Classification. The system has an average accuracy of 99.6%. The system follows supervised machine learning and is completed in two phases, training and testing.

S. H. Lee et al [13] use deep learning on raw leaf data to harvest discriminatory features from leaf images by learning and gaining intuition of the chosen features based on a DN approach and applying them as classifiers for plant identification. The arrangement of veins in a leaf is a better representative feature for plant classification than the outline shape and multi-level representation features are transformed hierarchically from lower-level to higher-level abstraction. The optimization capability can be further improved by going deeper into the network.

C. S. Chan et al [14] demonstrate the potential of CNNs for plant identification and to obtain a visual representation of which features are useful to differentiate a plant kingdom from different classes. CNN can provide better feature representation for leaf images compared to hand-crafted features with a performance of 99.5%. venation structure is an important feature in detection.

2.2 Key Gaps in the Literature

Some of the potential gaps are addressed here

- Diversity in flora is limited

  It appears that only a certain specific set of features was extracted from the dataset and it lacked variety in its species. A more detailed and comprehensive study is needed to widen the range ranging from regional to cultural variations.

- Inconsistent Evaluation Metrics

  To calculate the performance of the models applied, a lack of standardized evaluation metrics was observed with no facility for benchmarks.

- Robustness

  According to different conditions of the environment, the models were not robust to climatic variations such as lighting and background conditions.

- Adaptation across various geographical areas

  There is a lack of understanding of how various models are trained on the basics of regional datasets which can be transferred to become adaptable to different regions is another matter of concern.

- Compilation of traditional knowledge

  Collaborative efforts are not there to integrate traditional knowledge with machine learning tools which could be beneficial to local communities.

- Real-time processing with mobile app development

  The topic of deployment was only touched upon with no in-depth exploration of the same. Also, there are very few studies related to real-time detection.

- No interdisciplinary collaboration

Only technical aspects are focused on with no collaboration from experts in botany, or traditional medicine to improve the accuracy of the models.

- Scalability

  No challenges faced during measuring of scalability of models are discussed which can add value addition to large datasets.

- Transferability of models

  There is a gap in surveys in understanding the migration patterns of plant distribution where models need to be adjusted to additional influences on the characteristics of plants.

- Importance of feature representation

  In one of the papers, it is mentioned that CNNs provide feature extraction automatically which is less time-consuming compared to manual extraction. However, no detailed explanation is provided on features learned by CNNs to identify the significance of various species.

Chapter 3: System Development

3.1 Requirements and Analysis

The requirements and analysis needed are mentioned below: -

3.1.1 Functional Requirements

- Image upload and its processing
    i. The software should be compatible enough to allow users to post images of medicinal plants for identification.
    ii. We must process uploaded images to extract relevant features.

- Plant identification
    i. The capability to recognize species based on images should be present in the model.
    ii. It should provide a probability level for each identification.

- Support for multiple species
    i. The system must support a diverse range of medicinal species.
    ii. It should be capable of expanding its database to include new plants when needed.
    iii. Adaptability to environmental variations.
    iv. The model should be capable of handling variations in various aspects like quality of image, lighting conditions and background effects.
    v. It should be able to adjust to different environmental settings such as climatic changes and seasons.

- Real-time processing
    i. The system must be able to provide real-time identification which is capable enough for effective use.

ii. The maximum acceptable processing time for an image to be identified is 10 seconds.

- Accuracy threshold
  i. The maximum acceptable accuracy level is 97%.

- Training mode
  i. The system should have a mode for retraining the model with new data to improve accuracy over time.

3.1.2 Non-Functional Requirements

- Performance
  i. The system should provide results within 3 seconds of image upload.

- Reliability
  i. The system should have a high rate of accuracy ranging (any value between 90% to 99.9%) under normal operating conditions.
  ii. The model should be stable to handle network issues without causing any data loss.

- Availability
  i. The system should be available for use at all times with maintenance periods informed to users well in advance.

- Scalability
  i. The system should be scalable to accommodate an increasing number of plant species in the database.
  ii. It should be able to handle a growing user base with the ability to scale resources as needed.
  iii.
- Compatibility
  i. The system should be compatible with a variety of browsers like Chrome, Firefox, Safari and Edge.

ii. The deployment should be on common frameworks like TensorFlow and PyTorch.

- Security
  i. User data should be encrypted during transition and storage.
  ii. User authentication and authorization mechanisms should be implemented to control access to sensitive features.

- Usability
  i. The user interface should be friendly enough for the user to navigate with minimal training.

- Maintainability
  i. The model should be well-documented to facilitate future updates and enhancements.
  ii. The codebase should follow industry best practices for maintainability and readability.

- Compliance
  i. The system should comply with relevant data protection regulations, ensuring the privacy and security of user data.
  ii. Ethical considerations related to conservation and biodiversity should be integrated into the system.

- Training Dataset
  i. We have used a dataset which includes information about various species and different growth stages to provide annotation.
  ii. Every image is labelled with the corresponding species for learning.

- Image resolution
  i. Minimum resolution of pixels to capture every minute detail for correct analysis.
  ii. Support for common image formats like JPEG, PNG and JPG.

- Data augmentation
    i. Data Augmented to enhance the model including variations in rotations and scaling.

- Validation and testing
    i. Separate models for performing the hold-out process to split data into training and testing to access generalization and performance on unseen data.

3.1.3 Hardware Requirements

- GPU(s):
    i. High-performance GPUs for efficient training like NVIDIA T4 x2 series or Tesla GPUs.

- CPU
    i. A multicore CPU for handling preprocessing tasks and managing data flow to support overall operations.

- RAM
    i. A good amount of RAM is required to facilitate smooth training.

- Storage
    i. Sufficient storage space is needed for model checkpoints or in case any immediate files are generated while running code.

- Networking
    i. A good internet speed is needed for the acquisition of data to update models.

3.1.4 Software Requirements

- Language
    i. Python is used for implementation.

- Deep Learning Frameworks
    i. Choose suitable deep learning frameworks like TensorFlow, and Keras for the development of models.

- Version Control
    i. Certain version control systems like Git are used to manage, collaborate with team members and track any changes done.

- Development Environment
    i. The integrated development environment used is Google Collab and Kaggle Notebook for the development and experimentation of code

3.2 Project Design and Architecture

The project design provides a high-level overview of how our project model for medicinal plant classification will be developed. Different methodologies are employed in three different phases. In phase I, which is the initial step we performed the collection of data from publicly available datasets on Kaggle and Mendeley. The species range from aloe vera, Amruthaballi, and neem to turmeric. This dataset is the basic foundation for further development.

Moving to the next phase, the dataset is divided into three sets with image preprocessing techniques used to improve the model. The core of this project is in the training and classification phase where dense layers are introduced with Adam optimizer and SoftMax activation function to redefine the model. We have selected the best model based on accuracy and performance metrics.

In the next stage, we have proposed. Here, we have integrated the selected model into an application to facilitate the user-friendly identification of herbal leaves. The integration of AI into a web-based interface is a key approach to creating a robust solution.

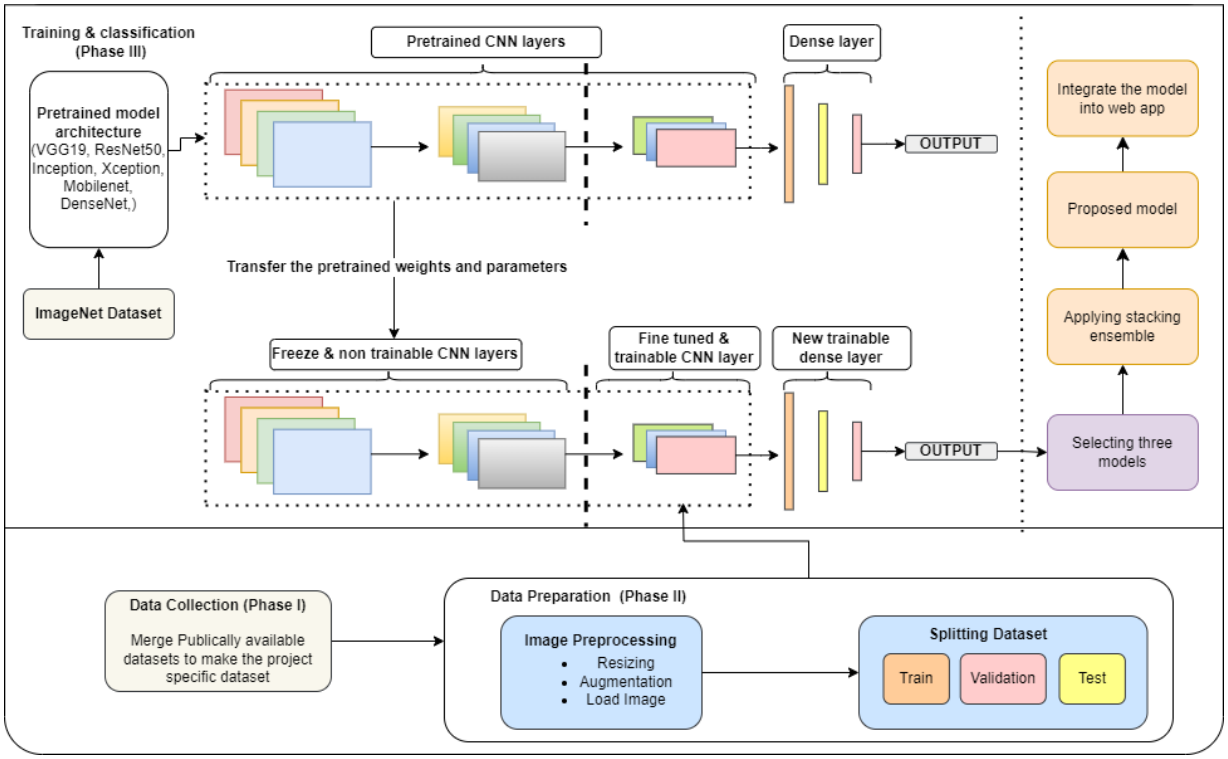


Figure 3.1: Project Architecture

3.3 Data Preparation

3.3.1 Data Collection

The dataset for this project is made by merging publicly available datasets on Kaggle (Indian medicinal leaves image dataset) and Mendeley (medicinal leaf dataset). The desired dataset focuses on 45 classes of different medicinal plants. The plants used were Aloe vera, Amla, Arive-Dantu, Basale, Betel, Bhrami, Coriender, Crape Jasmine, Curry, Drumstick, Fenugreek, Guava, Henna, Hibiscus, Honge, Indian Beech, Indian Mustard, Insulin, Jackfruit, Jamaica Cherry Gasagase, Jamun, Jasmine, Karanda, Lemon, Mango, Marigold, Mexican Mint, Mint, Neem, Oleander, Palak(Spinach), Papaya, Parijata, Peepal, Pomegranate, Rasna, Rose, Rose apple, Roxburgh fig, Sandalwood, Seetha Pala, Tamarind, Tulsi, Turmeric, Ashoka and several samples of classes are shown in Figure 3.2.

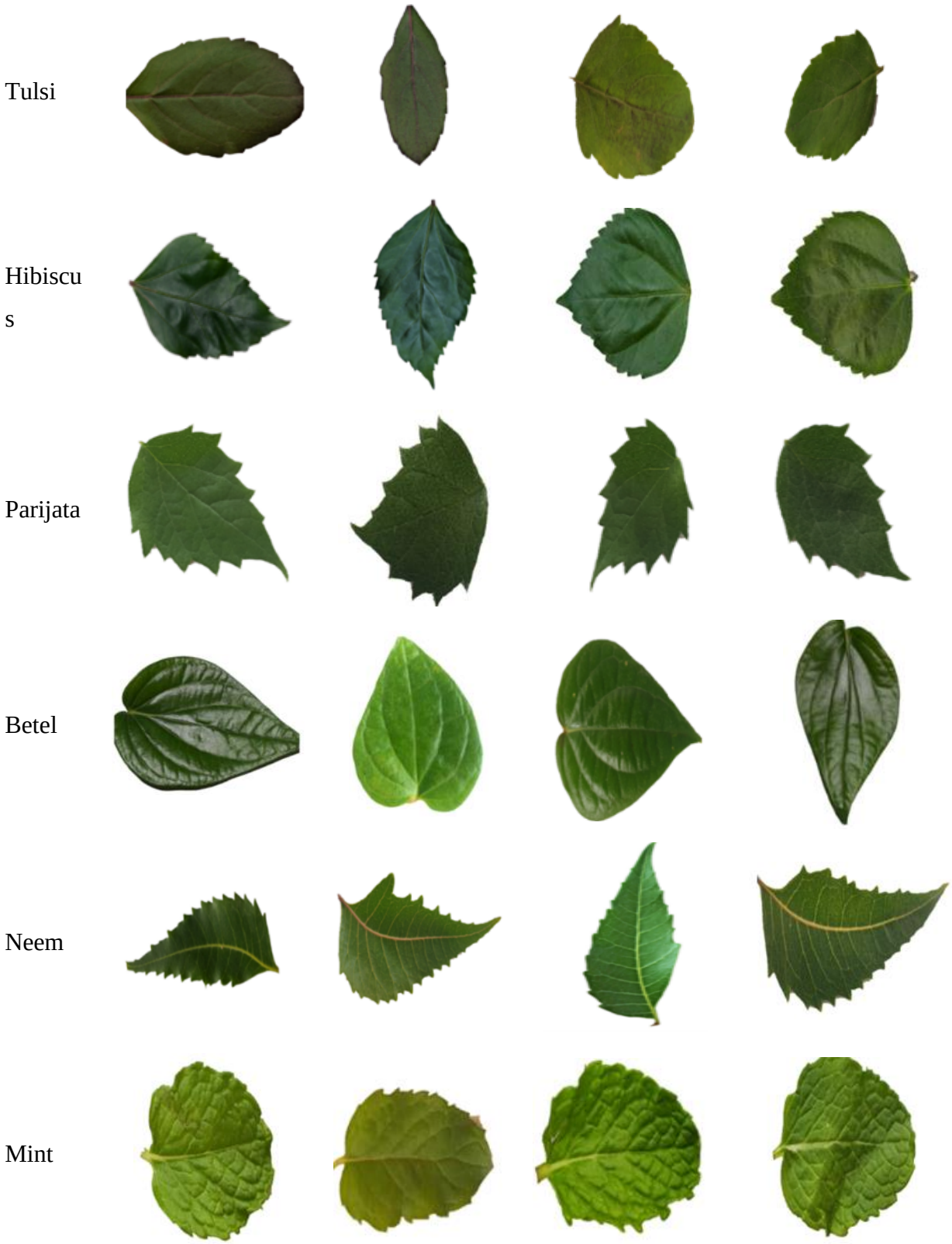


Fig 3.2: several classes from the dataset

3.3.2 Data Preprocessing

To achieve consistency in classification results and feature extraction, preprocessing is applied to the dataset.

In preparation of data first preprocessing was done on the data by resizing the images to 224 x 224 pixels to adjust input size and maintain uniformity, then the background of images was removed as it may dominate the analyzed image after that augmentation was done as the data was imbalanced by using various data augmentation techniques that were employed using an image processing library Kera's. These techniques included rotating the images at a fixed angle of 20 degrees, shifting the width and height by a random amount within a specified range (0.2), applying a shearing transformation with an angle of 0.2, and zooming the images by a factor of 0.2. Additionally, some images were mirrored horizontally for further diversification by having 500 images of each class. After data was prepared as needed, the dataset was split into training, validation, and testing in a ratio of 70%, 15%, and 15% respectively as given in Table 3.1.

Table 3.1: Number of images after splitting the dataset

Dataset	Number of Images
Training (70%)	15750
Validation (15%)	3375
Testing (15%)	3375

3.3.3 Training and classification

Convolutional Neural Networks (CNNs) are deep-learning algorithms designed to replicate the intelligence of the human brain. Unlike other machine learning algorithms, CNNs require minimal image pre-processing. They can analyze input images and assign weights and biases to different features or objects within the image, enabling them to distinguish between different images. These networks consist of multiple layers, with lower layers focused on identifying low-level features

and final layers dedicated to recognizing high-level features. In Figure 3, the visualization of feature maps from the VGG19 architecture illustrates how different blocks of the model emphasize distinct features. Some layers capture the foreground, background, lines, and more. Layers closer to the input layer capture finer details, while deeper layers exhibit less intricate feature maps as the model's depth increases. CNNs employed in this paper include VGG19, ResNet50, InceptionV3, Xception, MobileNetV2, and DenseNet121. The architecture is explained in detail in the sub-sections below.

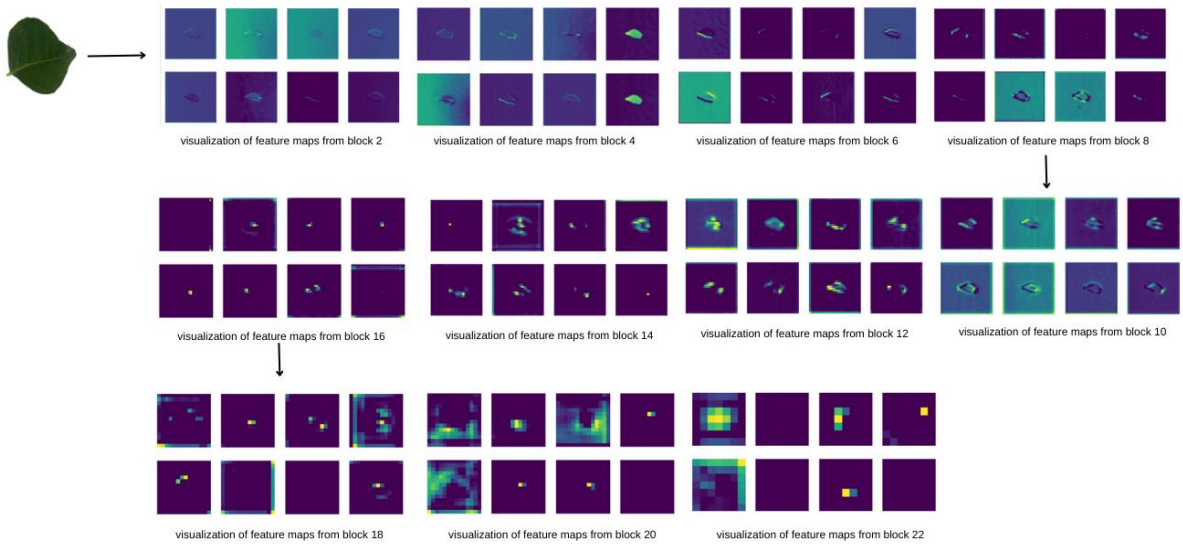


Fig 3.3: Feature Maps from the VGG19 CNN architecture

VGG19

VGG created by the Visual Geometry Group at Oxford in 2014, is a powerful neural network for recognizing images. It has a total of 19 layers out of which 16 are convolution layers with 3 fully connected layers and 1 SoftMax layer for classification [17]. It takes a 224 x 224 RGB image as input and the mean RGB value from each pixel is subtracted. The convolutional layers use 3 x 3 filters with a padding of 1 and a stride of 1. The number of filters increases from 64 to 512. The max pooling layers use 2 x 2 windows with a stride of 2. The average pooling layer reduces the feature map to 1 x 1. The fully connected layers have 4096, 4096, and 1000 units.

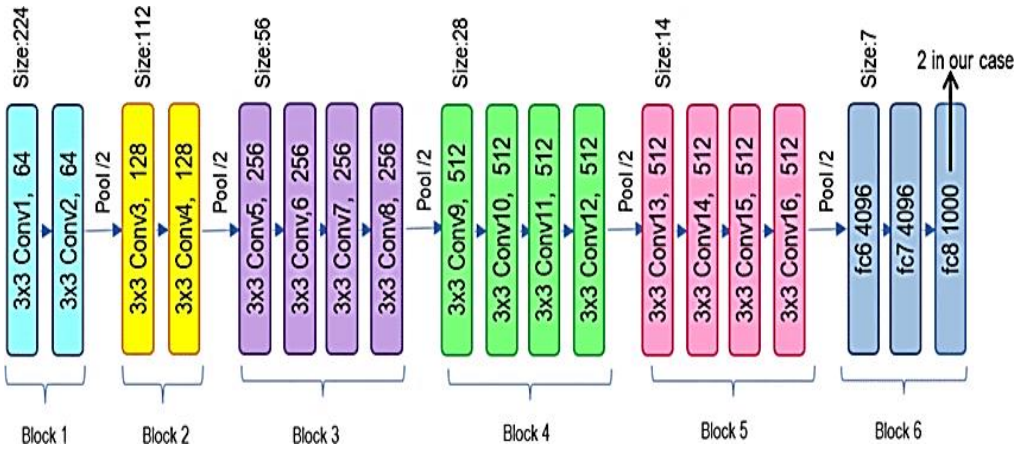


Fig 3.4: VGG19 model architecture

ResNet50

ResNet50 developed by Microsoft in 2015, is a robust convolutional neural network designed for image recognition and is a variation of the ResNet model. It contains 50 layers with 48 convolution layers [18], 1 max pooling, and an average pooling layer for classification. It is based on residual network architecture that uses skip connections to avoid the vanishing gradient problem. ResNet50 architectural design can be broken down into different components: Convolutional Layers, Identity Block, Convolutional Block, and Fully Connected Layers. Each component plays an essential role in processing visual information within the network. The Convolutional Layers are tasked with feature extraction from the input image, while the Identity Block and Convolutional Block facilitate efficient data flow through the network. Finally, the Fully Connected Layers perform the final classification. The building blocks of the residual network, add more layers to the network without increasing the complexity. The identity block is used when the input and output dimensions are the same, and the convolutional block is used when the input and output dimensions are different. The fully connected layers are used for the final classification of the image, using a SoftMax function[24]. Works on 224 x 224 pixels RGB images.

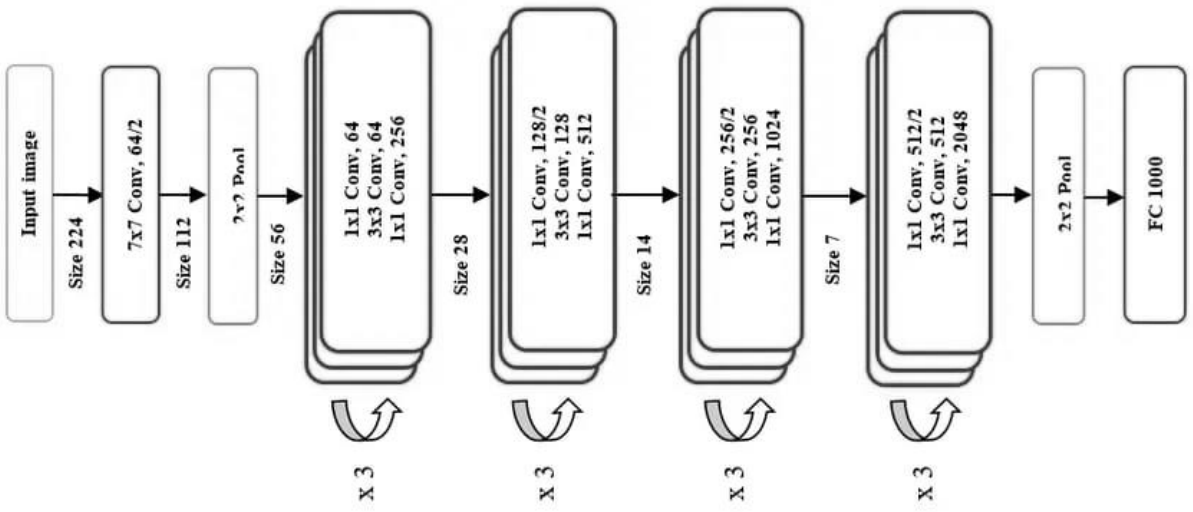


Figure 3.5: ResNet50 model architecture

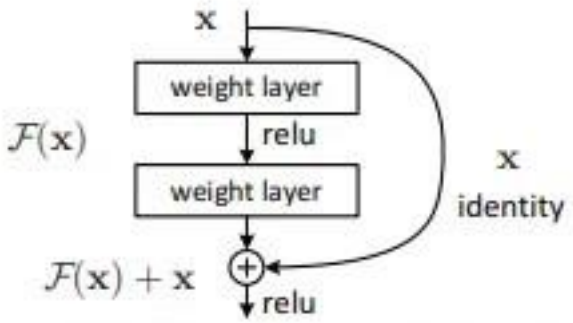


Fig 3.6: Residual block

InceptionV3

InceptionV3 developed by Google in 2015[16], has 48 layers using the inception module, which is a block of layers that performs multiple parallel operations on the input. The Inception module allows the network to capture different types of features at different scales and levels of abstraction. The network architecture includes 48 convolutional layers, 3 fully connected layers, and 1 SoftMax layer. It also uses an auxiliary classifier to provide additional supervision to the lower layers. Works on 299 x 299 pixels RGB images. It performs the final classification using a SoftMax function. Its architecture allows to extract features of images with high accuracy.

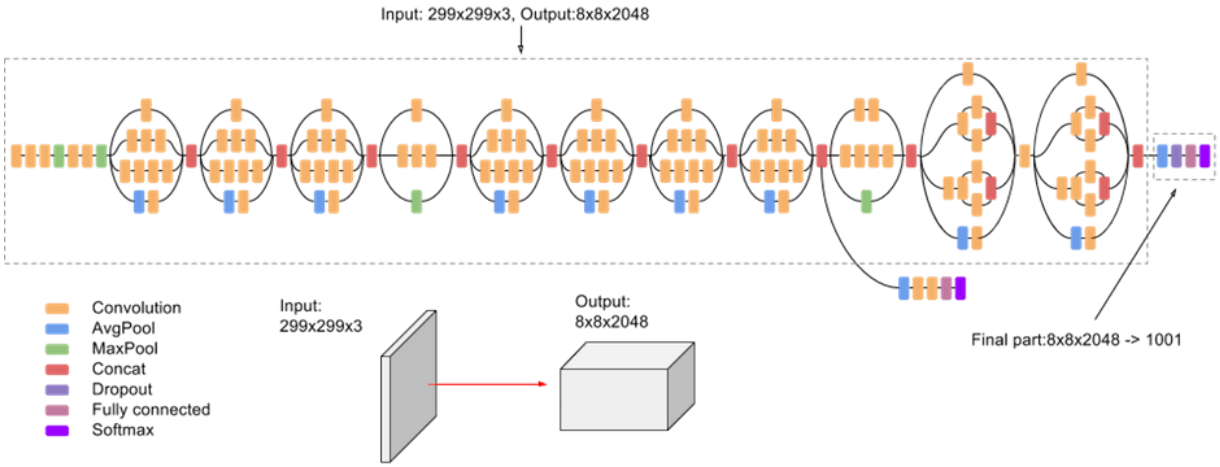


Fig 3.7: Inceptionv3 model architecture

Xception

Xception developed in 2017[19], is a highly effective convolutional neural network and is also based on an inception module but by using separable convolutions instead of regular convolutions. Depth wise separable convolutions are convolutions that split input to different channels and apply a spatial convolution separately to every channel followed by a pointwise convolution to combine all the channels. This reduces the number of parameters and computations [23]. The architecture of the Xception model consists of 36 convolutional layers, divided into four parts: the entry flow, the middle flow, the exit flow, and the classifier. It also uses residual connections to preserve the information. It takes a 299 x 299 RGB image as input.

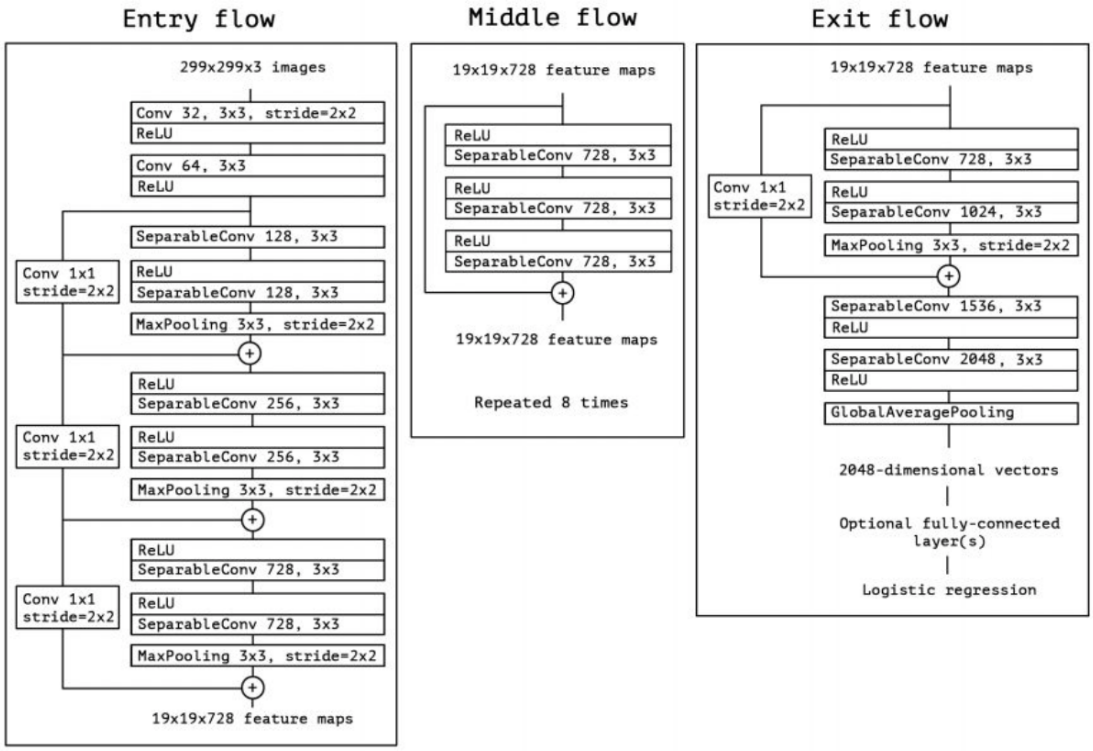


Fig 3.8: Xception model architecture

MobileNetV2

MobileNetV2 is a convolutional neural network architecture that tries to work well on mobile devices, there are two types of blocks used to extract features as sources of nonlinearity: [20]. one is the residual block with stride 1. Another is the block with stride 2 for downsizing the first layer is 1×1 convolution with ReLU. The second layer is depth wise convolution. The third layer is again a 1×1 convolution but without any nonlinearity [23]. The overall structure of MobileNetV2 consists initially of a full convolution layer with 32 filters, followed by an additional 19 bottleneck layers.

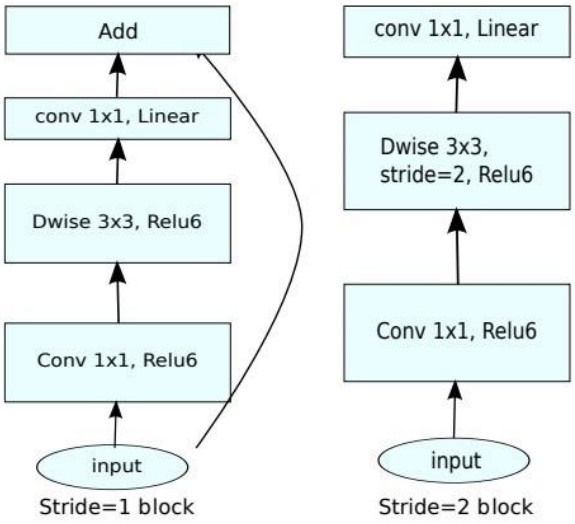


Fig 3.9: MobileNetV2 model architecture

DenseNet121

Dense Nets are divided into Dense Blocks, where the dimensions of the feature maps remain constant within a block, but the number of filters changes between them and these layers are called Transition Layers and down sampling is applied by using batch normalization, 1x1 convolution and 2x2 pooling layers so each layer has access to previous feature maps. Dense Nets layers are very narrow and they just add a small set of new feature-maps. Each layer has direct access to the gradients from the loss function and the original input image [21].

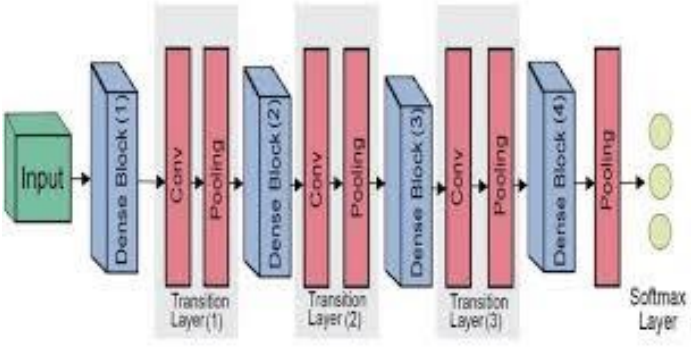


Fig 3.10: DenseNet121 model architecture

Transfer Learning

Deep learning aims to function as a human mind. When a child is taught how dogs and cats look and form a mental representation, then the child can recognize these animals in the future. Similarly, deep learning uses the same principle. Training the neural network model requires time and processing power to capture the accurate weights as required by the model. Transfer learning is used to solve this problem, which is the next step in deep learning as a pre-trained model. Transfer learning skips the requirement of finding the weights and moves to the training step of the model on a new dataset according to the requirement.

Ensemble Learning

Ensemble learning improves classifier performance by utilizing methods such as bagging, boosting, and stacking. Ensembles can consist of either homogenous or heterogeneous classes. In homogenous ensembles, a single base classifier is trained on various datasets, whereas in heterogeneous ensembles, multiple classifiers are trained on the same dataset. The ensemble makes predictions based on averaging, weighted averaging, and voting on the outputs from the base classifiers. In our project identification of medicinal plants using deep learning [22], we employed a heterogeneous ensemble approach with stacking to achieve the outcome.

3.2.2 Evaluation Measures

The evaluation measures used are:

Accuracy

Accuracy measures how often the model predicts the correct class out of all the predictions.

$$Accuracy = \frac{number\ of\ correct\ prediction}{total\ number\ of\ prediction}$$

Precision

Precision measures how many correctly predicted classes actually are true.

$$Precision = TP\ (TP + FP)$$

Recall

It measures How often the model predicts the correct class when the class is true.

$$Recall = TP\ (TP + FN)$$

F1 Score

A balance between precision and recall, calculated by their harmonic mean.

$$F1\ Score = 2 * precision * recall(precision + recall)$$

3.4 Implementation

The code for training and testing this project was done on the Kaggle notebook. Some of the main code snippets are:

```
class_folders = os.listdir(dataset_path)

images_per_row = 5
num_rows = math.ceil(len(class_folders) / images_per_row)

fig, axs = plt.subplots(num_rows, images_per_row, figsize=(15, 15))

for i, class_folder in enumerate(class_folders):
    class_folder_path = os.path.join(dataset_path, class_folder)
    image_files = [f for f in os.listdir(class_folder_path) if f.endswith('.jpg')]
    if image_files:
        first_image_path = os.path.join(class_folder_path; image_files[0])

        img = mpimg.imread(first_image_path)

        row = i // images_per_row
        col = i % images_per_row

        axs[row, col].imshow(img)
        axs[row, col].set_title(class_folder)
        axs[row, col].axis('off')

plt.tight_layout()
plt.show()
```

Fig 3.11: Loading the images to display

In figure 3.11 images are input into the system to be visible on the console.

```
def rename_and_resize_images(directory, output_directory, target_size=(224, 224)):
    for root, dirs, files in os.walk(directory):
        for folder in dirs:
            folder_path = os.path.join(root, folder)
            output_folder = os.path.join(output_directory, os.path.relpath(folder_path, directory))
            os.makedirs(output_folder, exist_ok=True)

            images = os.listdir(folder_path)
            idx = 1
            for img_name in sorted(images):
                img_ext = os.path.splitext(img_name)[1]
                new_name = f"{folder}_{idx:03d}{img_ext}"
                old_path = os.path.join(folder_path, img_name)
                new_path = os.path.join(output_folder, new_name)
                if old_path != new_path:
                    while os.path.exists(new_path):
                        idx += 1
                        new_name = f"{folder}_{idx:03d}{img_ext}"
                        new_path = os.path.join(output_folder, new_name)
                    try:
                        img = Image.open(old_path)
                        # Attempt to resize the image if it's not corrupt
                        resized_img = img.resize(target_size)
                        resized_img.save(new_path)
                    except Exception as e:
                        print(f"Skipping {old_path} - Error occurred: {str(e)}")
                        continue
                idx += 1

output_dataset_path = '/kaggle/working/medicinal_leaf_dataset'
rename_and_resize_images(dataset_path, output_dataset_path)
```

Fig 3.12: Rename and resizing the images

In figure 3.12 all the images are resized to 224 x 224 from their default sizes.

```
def get_dataset_partitions_tf(ds, train_split=0.7, val_split=0.15, test_split=0.15, shuffle=True, shuffle_size=10000):
    assert (train_split + test_split + val_split) == 1

    ds_size = len(ds)

    if shuffle:
        ds = ds.shuffle(shuffle_size, seed=12)

    train_size = int(train_split * ds_size)
    val_size = int(val_split * ds_size)

    train_ds = ds.take(train_size)
    val_ds = ds.skip(train_size).take(val_size)
    test_ds = ds.skip(train_size).skip(val_size)

    return train_ds, val_ds, test_ds
```

Fig 3.13: Splitting the dataset

In this figure 3.13 the ‘get_dataset_partitions_tf’ function used to split dataset into training, validation and testing into equal proportions equal to 1.

```
feature_extractor = VGG19(input_shape=(224,224,3),
                          include_top=False,
                          weights="imagenet")

for layer in feature_extractor.layers:
    layer.trainable=False

import tensorflow as tf
from tensorflow.keras import layers, models

with mirrored_strategy.scope():
    model_vgg19 = keras.Sequential([
        resize_and_rescale,
        layers.BatchNormalization(),
        feature_extractor,
        layers.BatchNormalization(),
        layers.Flatten(),
        layers.Dense(256, activation='relu'),
        layers.Dropout(0.2),
        layers.Dense(45, activation = 'softmax')
        ])

    model_vgg19.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate = 1e-4),
        loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
        metrics=['accuracy']
    )

    model_vgg19.build(input_shape = (None, 224, 224,3))

earlystopping = tf.keras.callbacks.EarlyStopping(monitor = 'val_accuracy',
                                mode = 'max' ,
                                patience = 7,
                                verbose = 1)
callback_list = [earlystopping]

history_vgg19=model_vgg19.fit(train_ds,
                        validation_data=val_ds,
                        epochs = 20,
                        callbacks = callback_list,
                        verbose = 1)

model_vgg19.save("/kaggle/working/vgg19.keras")
```

Fig 3.14: Implementation of VGG19

In figure 3.14 VGG19 is used as feature extractor for image classification by freezing front layers. After this, Adam optimizer used with cross-entropy loss.

```
feature_extractor = ResNet50(input_shape=(224,224,3),
                         include_top=False,
                         weights="imagenet")

for layer in feature_extractor.layers:
    layer.trainable=False

with mirrored_strategy.scope():
    model_resnet = keras.Sequential([
        resize_and_rescale,
        layers.BatchNormalization(),
        feature_extractor,
        layers.BatchNormalization(),
        layers.Flatten(),
        layers.Dense(256, activation='relu'),
        layers.Dropout(0.2),
        layers.Dense(45, activation = 'softmax')
        ])

    model_resnet.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate = 1e-4),
        loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
        metrics=['accuracy']
    )

    model_resnet.build(input_shape = (None, 224, 224,3))

earlystopping = tf.keras.callbacks.EarlyStopping(monitor = 'val_accuracy',
                              mode = 'max' ,
                              patience = 7,
                              verbose = 1)


callback_list = [earlystopping]

history_resnet=model_resnet.fit(train_ds,
                        validation_data=val_ds,
                        epochs = 20,
                        callbacks = callback_list,
                        verbose = 1)

model_resnet.save("/kaggle/working/resnet50.keras")
```

Fig 3.15: Implementation of ResNet50

In figure 3.15 Resnet 50 model used for classification by freezing layers and adding fully connected layers on top.

```
feature_extractor = InceptionV3(input_shape=(224,224,3),
                        include_top=False,
                        weights="imagenet")
for layer in feature_extractor.layers:
    layer.trainable=False

with mirrored_strategy.scope():
    model_inception = keras.Sequential([
        resize_and_rescale,
        layers.BatchNormalization(),
        feature_extractor,
        layers.BatchNormalization(),
        layers.Flatten(),
        layers.Dense(256, activation='relu'),
        layers.Dropout(0.2),
        layers.Dense(45, activation = 'softmax')
        ])

    model_inception.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate = 1e-4),
        loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
        metrics=['accuracy']
    )

    model_inception.build(input_shape = (None, 299, 299,3))

earlystopping = tf.keras.callbacks.EarlyStopping(monitor = 'val_accuracy',
                            mode = 'max' ,
                            patience = 7,
                            verbose = 1)

callback_list = [earlystopping]

history_inception=model_inception.fit(train_ds,
                        validation_data=val_ds,
                        epochs = 20,
                        callbacks = callback_list,
                        verbose = 1)

model_inception.save("/kaggle/working/inception.keras")
```

Fig 3.16: Implementation of InceptionV3

In figure 3.16 Inception V3 model used from ‘tensorflow.keras.applications’. The rest of implementation remains same.

```
feature_extractor = Xception(input_shape=(224,224,3),
                         include_top=False,
                         weights="imagenet")

for layer in feature_extractor.layers:
    layer.trainable=False

with mirrored_strategy.scope():
    model_xception = keras.Sequential([
        resize_and_rescale,
        layers.BatchNormalization(),
        feature_extractor,
        layers.BatchNormalization(),
        layers.Flatten(),
        layers.Dense(256, activation='relu'),
        layers.Dropout(0.2),
        layers.Dense(45, activation = 'softmax')
        ])

    model_xception.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate = 1e-4),
        loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
        metrics=['accuracy']
    )

    model_xception.build(input_shape = (None, 299, 299,3))

earlystopping = tf.keras.callbacks.EarlyStopping(monitor = 'val_accuracy',
                                mode = 'max' ,
                                patience = 7,
                                verbose = 1)

callback_list = [earlystopping]

history_xception=model_xception.fit(train_ds,
                        validation_data=val_ds,
                        epochs = 20,
                        callbacks = callback_list,
                        verbose = 1)

model_xception.save("/kaggle/working/xception.keras")
```

Fig 3.17: Implementation of Xception

In figure 3.17 Xception model used and rest part remains unchanged as usage of other models and compilation process also follow similar pattern.

```
feature_extractor = MobileNetV2(input_shape=(224,224,3),
                        include_top=False,
                        weights="imagenet")

for layer in feature_extractor.layers:
    layer.trainable=False

with mirrored_strategy.scope():
    model_mobilenet = keras.Sequential([
        resize_and_rescale,
        layers.BatchNormalization(),
        feature_extractor,
        layers.BatchNormalization(),
        layers.Flatten(),
        layers.Dense(256, activation='relu'),
        layers.Dropout(0.2),
        layers.Dense(80, activation = 'softmax')
        ])

    model_mobilenet.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate = 1e-4),
        loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
        metrics=['accuracy']
    )

    model_mobilenet.build(input_shape = (None, 224, 224,3))

earlystopping = tf.keras.callbacks.EarlyStopping(monitor = 'val_accuracy',
                            mode = 'max' ,
                            patience = 7,
                            verbose = 1)

callback_list = [earlystopping]

history_mobilenet=model_mobilenet.fit(train_ds,
                        validation_data=val_ds,
                        epochs = 20,
                        callbacks = callback_list,
                        verbose = 1)

model_mobilenet.save("/kaggle/working/mobilenetv2.keras")
```

Fig 3.18: Implementation of MobilenetV2

In figure 3.18 MobilenetV2 used as it is suitable for mobile devices and helps in user-interface design as well.

```
feature_extractor = DenseNet121(input_shape=(224,224,3),
                        include_top=False,
                        weights="imagenet")

for layer in feature_extractor.layers:
    layer.trainable=False

with mirrored_strategy.scope():
    model_densenet = keras.Sequential([
        resize_and_rescale,
        layers.BatchNormalization(),
        feature_extractor,
        layers.BatchNormalization(),
        layers.Flatten(),
        layers.Dense(256, activation='relu'),
        layers.Dropout(0.2),
        layers.Dense(80, activation = 'softmax')
        ])

    model_densenet.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate = 1e-4),
        loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
        metrics=['accuracy']
    )

    model_densenet.build(input_shape = (None, 224, 224,3))

earlystopping = tf.keras.callbacks.EarlyStopping(monitor = 'val_accuracy',
                            mode = 'max' ,
                            patience = 7,
                            verbose = 1)


callback_list = [earlystopping]

history_densenet=model_densenet.fit(train_ds,
                        validation_data=val_ds,
                        epochs = 20,
                        callbacks = callback_list,
                        verbose = 1)

model.save("/kaggle/working/denesenet121.keras")
```

Fig 3.19: Implementation of DenseNet121

In figure 3.19 DenseNet121 models used for feature extractions with the implementation part remaining same.

```
import matplotlib.pyplot as plt

# loss and accuracy graph
acc = history_resnet.history['accuracy']
val_acc = history_resnet.history['val_accuracy']

loss = history_resnet.history['loss']
val_loss = history_resnet.history['val_loss']

epochs_range = range(1, len(acc) + 1)  # Corrected the range

plt.figure(figsize=(9, 3))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

Fig 3.20: Plotting the training and validation accuracy and loss graph

In this figure 3.20 accuracy and loss data from training history object ‘history_exception’ is achieved.

```
import matplotlib.pyplot as plt
import numpy as np

# Evaluate the model on the test dataset
evaluation_results_mobilenet = model_mobilenet_test.evaluate(test_ds)

# Extract loss and accuracy from the evaluation results
loss, accuracy = evaluation_results_mobilenet

# Print the loss and accuracy
print(f"Loss: {loss}")S
print(f"Accuracy: {accuracy}")


plt.figure(figsize=(3, 3))
# Visualize the results using a bar chart
labels = ['Loss', 'Accuracy']
values = [loss, accuracy]

# Set the width of the bars
bar_width = 0.3  # Change this value as needed

plt.bar(labels, values, color=['red', 'green'], width=bar_width)
plt.title('Model Evaluation Results on Test Dataset')
plt.xlabel('Metrics')
plt.ylabel('Values')
plt.show()
```

Fig 3.21: Plotting the testing accuracy and loss graph

```python
def predict(model, img):
    img_array = tf.keras.preprocessing.image.img_to_array(images[i].numpy())
    img_array = tf.expand_dims(img_array, 0)

    predictions = model.predict(img_array, verbose = 0)

    predicted_class = class_names[np.argmax(predictions[0])]
    confidence = round(100 * (np.max(predictions[0])), 2)
    return predicted_class, confidence
```

```python
true_labels = []
predicted_labels = []

for images, labels in test_ds:
    for i in range(len(labels)):
        true_labels.append(labels[i].numpy())

        predicted_class, _ = predict(model_mobilenet_test, images[i].numpy())
        predicted_labels.append(predicted_class)

true_labels = np.array(true_labels)
predicted_labels = np.array(predicted_labels)
```

```python
report = classification_report(true_labels, predicted_labels)
table = PrettyTable()
table.field_names = ["Classification Report", "Class Names"]

table.add_row([report, '\n'.join(f"{i}: {class_name}" for i, class_name in enumerate(class_names))])

table.align["Classification Report"] = "l"
table.align["Class Names"] = "l"

print(table)
```

Fig 3.22: Predicting and plotting the Classification report

In this figure 3.22 classification report is obtained and plotted.

```
app = Flask(__name__)

# allowed file extensions for image uploads
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}

# Path to the folder where uploaded images will be stored
UPLOAD_FOLDER = 'uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

model = tf.keras.models.load_model('custom_leaf_vmodel.h5')

# check if a file has an allowed extension
def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
        ALLOWED_EXTENSIONS

# route to render the upload form
@app.route('/')
def upload_form():
    return render_template('u.html')
```

```
# route to process the uploaded image
@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return redirect(request.url)
    file = request.files['file']
    if file.filename == '':
        return redirect(request.url)
    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        # Process the uploaded image with the model
        image = tf.keras.preprocessing.image.load_img(os.path.join(app
            .config['UPLOAD_FOLDER'], filename), target_size=(224, 224))
        image = tf.keras.preprocessing.image.img_to_array(image)
        image = np.expand_dims(image, axis=0)
        predictions = model.predict(image)
        # Get the predicted class
        predicted_class = np.argmax(predictions[0])
```

```
        # map the class to the plant name based on your model's output
        plant_names = ['Aloevera', 'Amla', 'Amruthaballi', 'Arali',
            'Astma_weed', 'Badipala', 'Balloon_Vine', 'Bamboo', 'Betel',
            'Bhrami', 'Bringaraja', 'Castor', 'Chakte', 'Chilly', 'Coriender',
            'Curry', 'Doddpathre', 'Drumstick', 'Ekka', 'Eucalyptus',
            'Gasagase', 'Ginger', 'Globe Amarnath', 'Guava', 'Henna',
            'Hibiscus', 'Honge', 'Jackfruit', 'Jasmine', 'Kambajala',
            'Kasambruga', 'Kohlrabi', 'Lantana', 'Lemon', 'Malabar_Nut',
            'Mango', 'Marigold', 'Mint', 'Neem', 'Nerale', 'Onion', 'Palak
            (Spinach)', 'Papaya', 'Parijatha', 'Pomoegranate', 'Pumpkin',
            'Raddish', 'Rose', 'Sampige', 'Seethapala', 'Tamarind', 'Tomato',
            'Tulsi', 'Turmeric', 'ashoka', 'camphor']
        plant_name = plant_names[predicted_class]
        return render_template('res.html', plant_name=plant_name, filename
            =filename)
    else:
        return redirect(request.url)

if __name__ == '__main__':
    app.run(debug=True)
```

Fig 3.23: User Interface code

In this figure 3.23 user interface code is implemented to provide real time output.

3.5 Key Challenges

While developing this project, we came across various challenges. We need to address these issues to ensure the accuracy and reliability of the developed models. Some of the potential key challenges and the strategies to address them are as follows: -

1) The constraints of the dataset are limited and imbalanced. A more representative dataset is made by implementing data augmentation techniques to expand and update the dataset with balanced classes.

2) There is a need to inculcate those practices which are cultural and ethical to ensure transparency. We prioritized these strategies by developing and maintaining ethical guidelines for all the AI tools.

3) VGG 19 and Resnet 50 models are overfitted during model training. Techniques like dropout regularization and early stopping are used to prevent overfitting and generalize models.

4) There is difficulty in overcoming the complexity of deep learning models applied by end-users. So, models like Inception V3, and Xception are used to provide more clearer analysis to overcome the black-box nature of the models.

Chapter 4: Testing

4.1 Testing strategy

In order to obtain precise results in identification of medicinal plants, we employed a testing strategy on different models. The models used are VGG 19, ResNet50, InceptionV3, Xception, mobileNetV2 and DenseNet121. We utilize the strength of these models by extracting useful features from data.

We further elaborated our research by proposing our own customized model using ensemble learning techniques. We preferred this technique as it produces high performance output with diverse algorithms. After various experimentation, this method proved to be suitable due to its robust nature and adapted itself to wide range of different species of plants.

Further, we identified the top three models from our proposed models and employed stacking on them. We utilized this technique as this approach helps to predict multiple models from models to improve its performance.

To overcome the limitation of individual models we came up with the idea of selection of top performing models through stacking to produce advancement in domain of medicinal field.

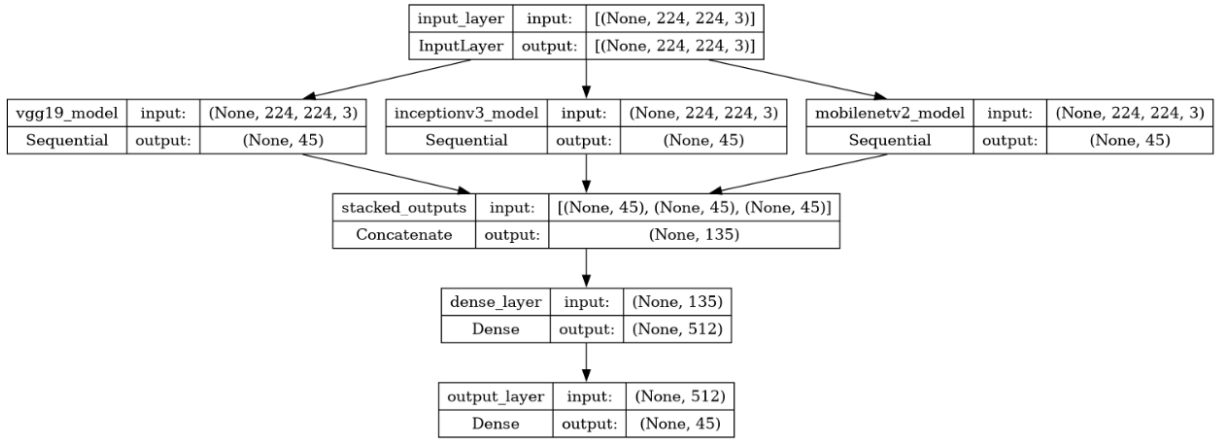


Fig 4.1: Architecture of Proposed Model

In this figure 4.1, we have demonstrated the architecture of stacking model. After stacking is applied on the models which are VGG 19, Inception V3 and MobileNet V2, dense layer is introduced on the stacked outputs using activation function relu and learning rate of 0.001. The number of neurons used in dense layer are 512 and total predicted classes are 45. After this the output is obtained on the stacked model.

To provide a user interface with database management we will develop a mobile application. This application will be simple to use based on user-feedback. Various deep learning tools will be integrated for real-time processing.

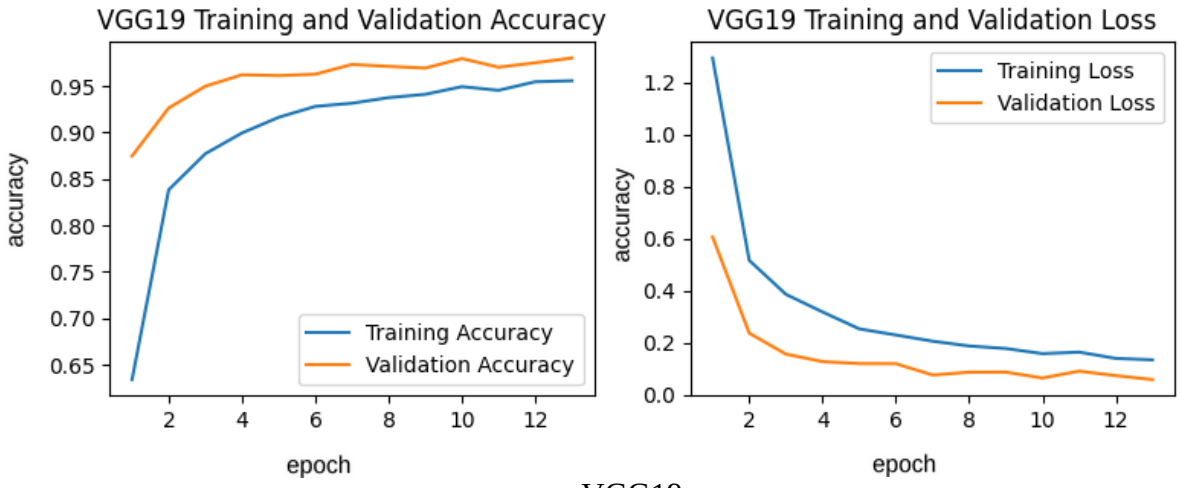
At present, we have developed a basic structure for the user-interface. We have used flask and designed the basic front-end of the interface. The user will upload the image and processing will be done of the models which we have trained. Initially, we have setup a backend application using TensorFlow.

Chapter 5: Results and Evaluation

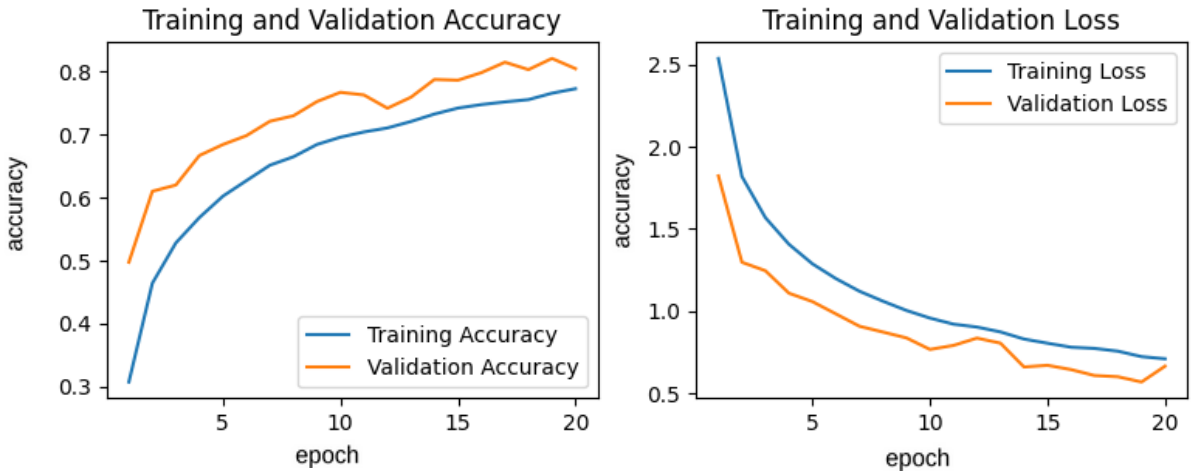
In this chapter, we have discussed the results achieved in identifying herbs and measuring the performance of the application.

The evaluation of VGG 19, ResNet50, InceptionV3, Xception, mobileNetV2 and DenseNet121 models trained with twenty epochs and with learning rate of 0.001. ResNet50 exhibited low-quality performance with an accuracy of 81.89% only to generalize the data. So, it is not suitable to solve complex problems.

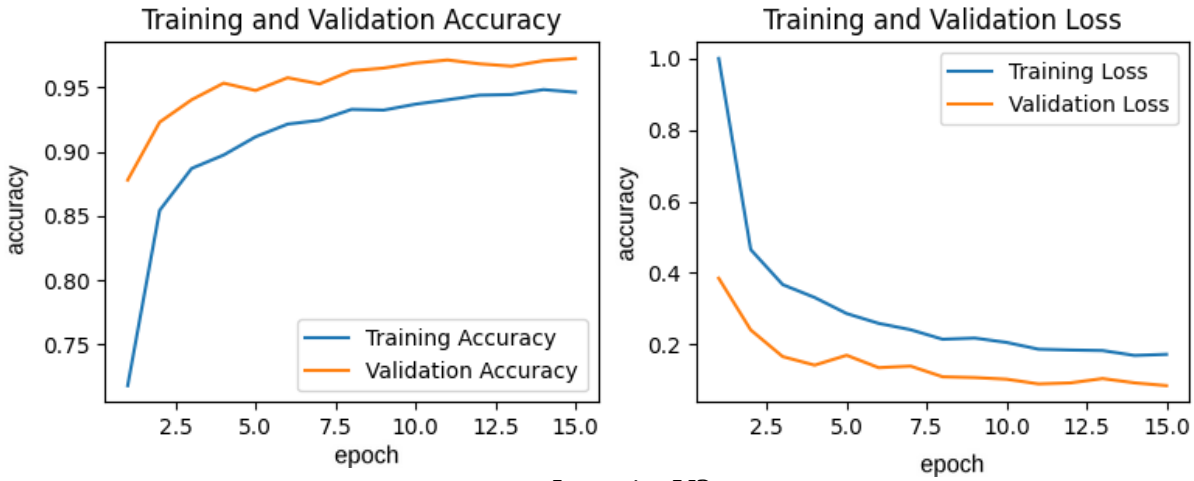
However, DenseNet121 showed an accuracy of 97.81% with a reduction in loss from 1.6% to 0.2%. It is fit to quickly learn from the training set with superior performance. Also, VGG19, Inception V3, Xception and mobileNetV2 are good enough for demonstration as they showed an accuracy of 96.16%, 96.18%, 96.75% and 97.03% respectively by the end of training. Hence, ResNet50 has some issues grasping high accuracy remaining five models are well-suited to display a high degree of success.



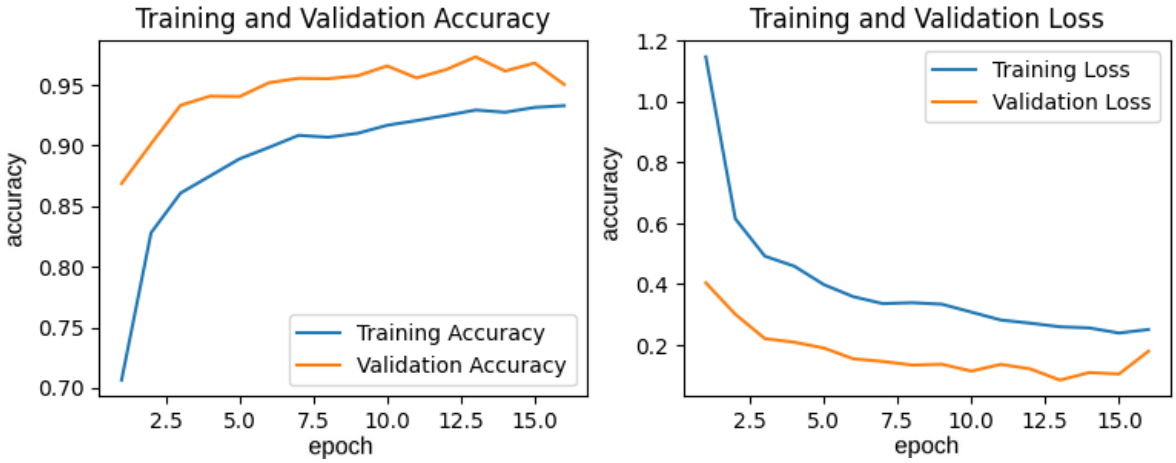
VGG19



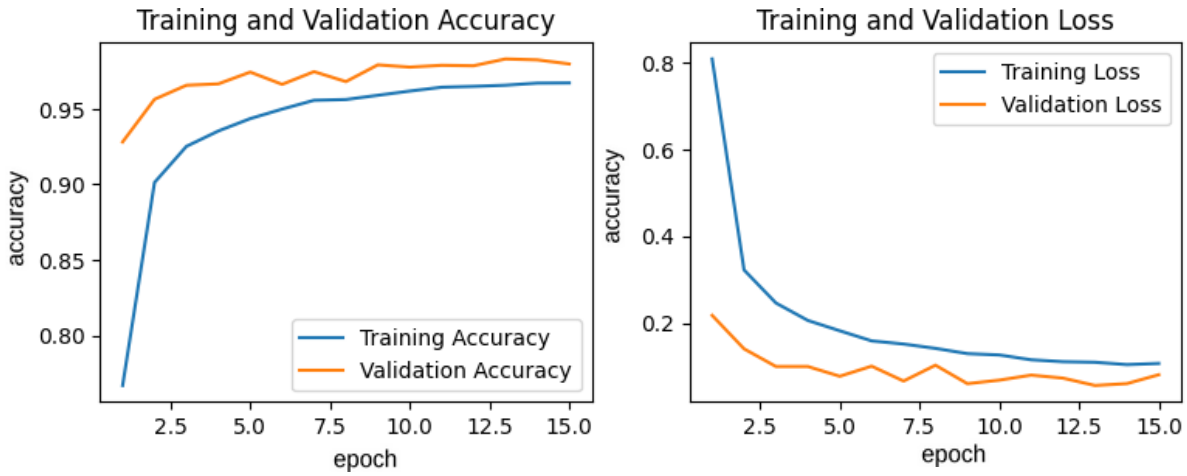
ResNet50



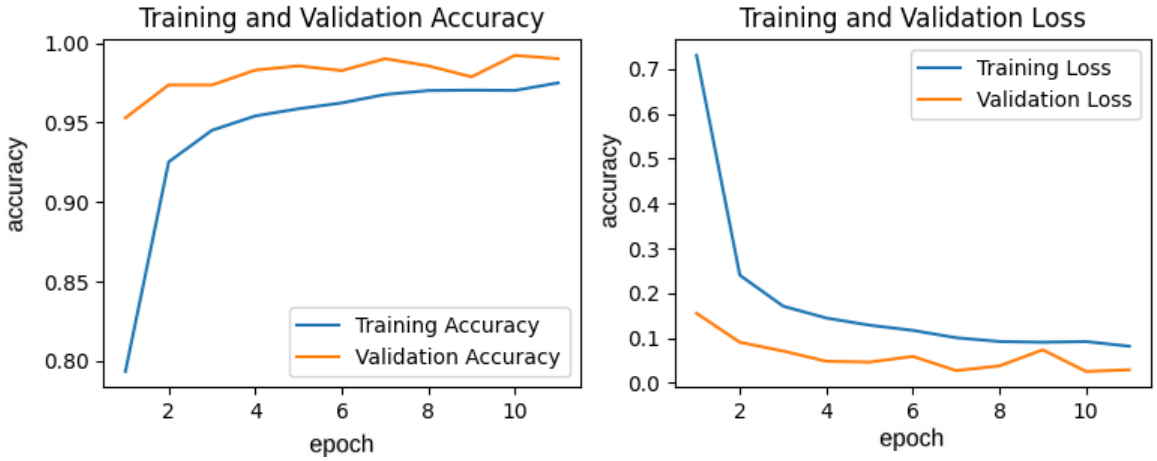
InceptionV3



Xception



MobileNetV2



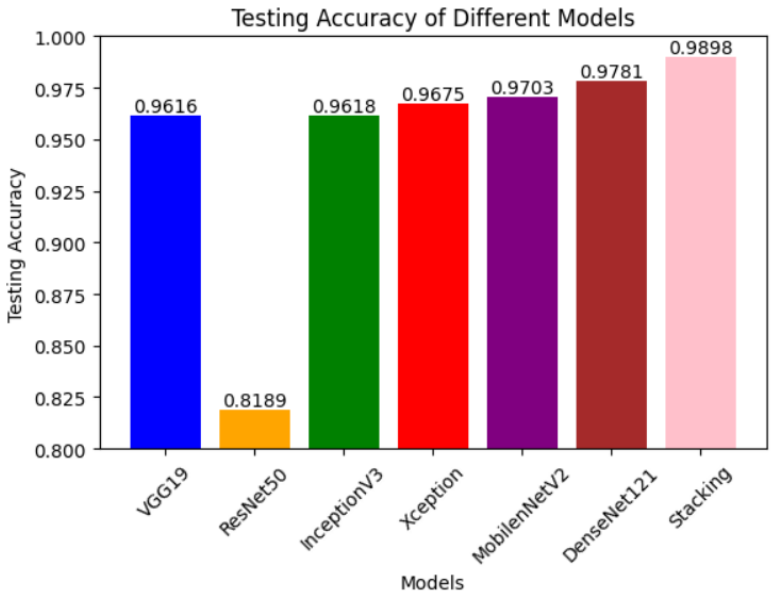
DenseNet121

Graph 5.1: Comparison of testing and validation accuracy and loss of different models

This shows the effectiveness of VGG19, Inception V3, Xception and mobileNetV2 architectures in learning and classifying the dataset, with DenseNet121 demonstrating exceptional performance among the models trained.

Table 5.1: Training, validation and testing accuracy and loss

Models (lr=0.0001)	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy	Testing Loss	Testing Accuracy
VGG19	0.1352	0.9557	0.0595	0.9802	0.0892	0.9616
ResNet50	0.7097	0.7728	0.6648	0.8047	0.5707	0.8189
InceptionV3	0.1722	0.9464	0.0846	0.9727	0.1245	0.9618
Xception	0.2511	0.9329	0.1797	0.9504	0.1121	0.9675
MobilenNetV2	0.1065	0.9673	0.0804	0.9799	0.0564	0.9703
DenseNet121	0.0815	0.9749	0.0289	0.9871	0.0412	0.9781
Stacking	0.0817	0.9850	0.0275	0.9900	0.0234	0.9898



Graph 5.2: Comparison of testing accuracies of different models

Classification reports of all the models applied individually are shown in table 5.1. This report consists of accuracy, macro average and weighted average including evaluation metrics of precision, recall, f1-score and support.

Classification Report					Class Names
	precision	recall	f1-score	support	0: Aloevera
					1: Amla
0	0.97	0.96	0.97	73	2: Arive-Dantu
1	0.90	0.70	0.79	88	3: Basale
2	1.00	1.00	1.00	80	4: Betel
3	1.00	0.99	0.99	72	5: Bhrami
4	0.99	1.00	0.99	76	6: Coriender
5	0.99	0.95	0.97	79	7: Crape_Jasmine
6	0.99	0.99	0.99	70	8: Curry
7	1.00	0.96	0.98	78	9: Drumstick
8	1.00	1.00	1.00	83	10: Fenugreek
9	1.00	1.00	1.00	83	11: Guava
10	1.00	1.00	1.00	66	12: Henna
11	1.00	1.00	1.00	72	13: Hibiscus
12	0.89	0.80	0.84	79	14: Honge
13	1.00	1.00	1.00	74	15: Indian_Beech
14	0.79	1.00	0.88	65	16: Indian_Mustard
15	1.00	1.00	1.00	70	17: Insulin
16	1.00	1.00	1.00	72	18: Jackfruit
17	0.90	0.99	0.94	74	19: Jamaica_Cherry-Gasagase
18	0.98	1.00	0.99	84	20: Jamun
19	1.00	1.00	1.00	72	21: Jasmine
20	1.00	0.99	0.99	75	22: Karanda
21	0.99	0.99	0.99	96	23: Lemon
22	1.00	1.00	1.00	81	24: Mango
23	0.99	0.99	0.99	85	25: Marigold
24	0.97	1.00	0.99	77	26: Mexican_Mint
25	0.92	0.97	0.94	86	27: Mint
26	1.00	1.00	1.00	80	28: Neem
27	0.99	1.00	0.99	73	29: Oleander
28	1.00	1.00	1.00	86	30: Palak(Spinach)
29	1.00	0.98	0.99	83	31: Papaya
30	0.97	0.99	0.98	68	32: Parijata
31	0.99	1.00	0.99	76	33: Peepal
32	1.00	1.00	1.00	80	34: Pomegranate
33	1.00	1.00	1.00	59	35: Rasna
34	0.98	1.00	0.99	95	36: Rose
35	1.00	1.00	1.00	67	37: Rose_apple
36	0.91	0.80	0.85	66	38: Roxburgh_fig
37	1.00	1.00	1.00	85	39: Sandalwood
38	0.99	1.00	0.99	84	40: Seethapala
39	1.00	0.99	0.99	72	41: Tamarind
40	0.75	0.81	0.78	73	42: Tulsi
41	0.96	1.00	0.98	67	43: Turmeric
42	1.00	0.99	0.99	76	44: ashoka
43	0.97	0.99	0.98	76	
44	0.96	0.95	0.96	80	
accuracy			0.97	3456	
macro avg	0.97	0.97	0.97	3456	
weighted avg	0.97	0.97	0.97	3456	

VGG19

Classification Report					Class Names
	precision	recall	f1-score	support	0: Aloevera
					1: Amla
0	0.91	0.90	0.90	79	2: Arive-Dantu
1	0.68	0.50	0.58	88	3: Basale
2	0.80	0.88	0.83	80	4: Betel
3	0.80	1.00	0.89	74	5: Bhrami
4	0.94	0.97	0.95	76	6: Coriender
5	0.64	0.88	0.74	82	7: Crape_Jasmine
6	0.94	0.87	0.90	75	8: Curry
7	0.61	0.89	0.72	54	9: Drumstick
8	0.90	0.74	0.81	73	10: Fenugreek
9	0.76	0.73	0.75	75	11: Guava
10	0.98	0.71	0.83	84	12: Henna
11	0.82	0.96	0.89	80	13: Hibiscus
12	0.65	0.44	0.53	68	14: Honge
13	0.94	0.91	0.93	70	15: Indian_Beech
14	0.63	0.76	0.69	66	16: Indian_Mustard
15	0.90	0.97	0.93	64	17: Insulin
16	0.98	0.99	0.98	82	18: Jackfruit
17	0.64	0.63	0.63	78	19: Jamaica_Cherry-Gasagase
18	0.76	0.93	0.84	71	20: Jamun
19	0.95	0.88	0.91	66	21: Jasmine
20	0.90	0.72	0.80	75	22: Karanda
21	0.95	0.69	0.80	80	23: Lemon
22	0.67	0.97	0.79	69	24: Mango
23	0.83	0.96	0.89	73	25: Marigold
24	0.97	0.72	0.83	83	26: Mexican_Mint
25	0.74	0.77	0.76	91	27: Mint
26	1.00	0.87	0.93	68	28: Neem
27	0.97	0.90	0.93	86	29: Oleander
28	0.99	1.00	0.99	76	30: Palak(Spinach)
29	0.80	0.85	0.83	81	31: Papaya
30	0.84	0.88	0.86	60	32: Parijata
31	0.91	0.93	0.92	87	33: Peepal
32	0.96	0.96	0.96	80	34: Pomegranate
33	0.99	0.97	0.98	73	35: Rasna
34	0.89	0.68	0.78	73	36: Rose
35	0.98	0.81	0.89	79	37: Rose_apple
36	0.68	0.69	0.69	94	38: Roxburgh_fig
37	0.73	1.00	0.85	85	39: Sandalwood
38	0.92	0.76	0.83	95	40: Seethapala
39	0.95	0.71	0.81	75	41: Tamarind
40	0.51	0.60	0.55	81	42: Tulsi
41	0.86	0.85	0.85	79	43: Turmeric
42	0.78	1.00	0.88	83	44: ashoka
43	0.82	0.71	0.76	72	
44	0.78	0.70	0.74	73	
accuracy			0.83	3456	
macro avg	0.84	0.83	0.83	3456	
weighted avg	0.84	0.83	0.82	3456	

ResNet50

Classification Report					Class Names
	precision	recall	f1-score	support	0: Aloevera
					1: Amla
0	0.99	0.92	0.95	79	2: Arive-Dantu
1	0.84	0.82	0.83	88	3: Basale
2	1.00	0.96	0.98	80	4: Betel
3	1.00	1.00	1.00	74	5: Bhrami
4	1.00	0.99	0.99	76	6: Coriender
5	0.99	0.96	0.98	82	7: Crape_Jasmine
6	0.99	1.00	0.99	75	8: Curry
7	1.00	1.00	1.00	54	9: Drumstick
8	1.00	0.97	0.99	73	10: Fenugreek
9	1.00	0.97	0.99	75	11: Guava
10	1.00	1.00	1.00	84	12: Henna
11	1.00	1.00	1.00	80	13: Hibiscus
12	0.85	0.88	0.86	68	14: Honge
13	1.00	0.99	0.99	70	15: Indian_Beech
14	0.93	0.86	0.90	66	16: Indian_Mustard
15	0.91	1.00	0.96	64	17: Insulin
16	1.00	0.99	0.99	82	18: Jackfruit
17	0.95	0.91	0.93	78	19: Jamaica_Cherry-Gasagase
18	0.95	1.00	0.97	71	20: Jamun
19	1.00	1.00	1.00	66	21: Jasmine
20	1.00	1.00	1.00	75	22: Karanda
21	0.99	1.00	0.99	80	23: Lemon
22	0.97	0.96	0.96	69	24: Mango
23	1.00	0.97	0.99	73	25: Marigold
24	1.00	0.89	0.94	83	26: Mexican_Mint
25	0.87	0.88	0.87	91	27: Mint
26	1.00	1.00	1.00	68	28: Neem
27	0.99	0.99	0.99	86	29: Oleander
28	1.00	1.00	1.00	76	30: Palak(Spinach)
29	0.79	1.00	0.88	81	31: Papaya
30	0.88	1.00	0.94	60	32: Parijata
31	0.98	0.97	0.97	87	33: Peepal
32	1.00	1.00	1.00	80	34: Pomegranate
33	0.99	1.00	0.99	73	35: Rasna
34	0.95	1.00	0.97	73	36: Rose
35	0.99	1.00	0.99	79	37: Rose_apple
36	0.90	0.95	0.92	94	38: Roxburgh_fig
37	1.00	0.84	0.91	85	39: Sandalwood
38	1.00	0.98	0.99	95	40: Seethapala
39	0.97	0.96	0.97	75	41: Tamarind
40	0.83	0.93	0.88	81	42: Tulsi
41	1.00	0.91	0.95	79	43: Turmeric
42	0.99	0.99	0.99	83	44: ashoka
43	0.95	1.00	0.97	72	
44	1.00	0.92	0.96	73	
accuracy			0.96	3456	
macro avg	0.96	0.96	0.96	3456	
weighted avg	0.96	0.96	0.96	3456	

InceptionV3

Classification Report					Class Names
	precision	recall	f1-score	support	0: Aloevera
					1: Amla
0	0.96	0.97	0.97	79	2: Arive-Dantu
1	0.79	0.88	0.83	88	3: Basale
2	1.00	0.94	0.97	80	4: Betel
3	1.00	0.99	0.99	74	5: Bhrami
4	1.00	0.93	0.97	76	6: Coriender
5	0.98	0.96	0.97	82	7: Crape_Jasmine
6	0.99	1.00	0.99	75	8: Curry
7	0.92	1.00	0.96	54	9: Drumstick
8	0.99	1.00	0.99	73	10: Fenugreek
9	0.96	0.96	0.96	75	11: Guava
10	1.00	0.96	0.98	84	12: Henna
11	1.00	0.97	0.99	80	13: Hibiscus
12	0.91	0.87	0.89	68	14: Honge
13	1.00	0.99	0.99	70	15: Indian_Beech
14	0.90	0.94	0.92	66	16: Indian_Mustard
15	0.90	1.00	0.95	64	17: Insulin
16	1.00	1.00	1.00	82	18: Jackfruit
17	0.93	0.87	0.90	78	19: Jamaica_Cherry-Gasagase
18	1.00	0.99	0.99	71	20: Jamun
19	0.99	1.00	0.99	66	21: Jasmine
20	0.99	1.00	0.99	75	22: Karanda
21	1.00	1.00	1.00	80	23: Lemon
22	0.97	0.99	0.98	69	24: Mango
23	0.97	1.00	0.99	73	25: Marigold
24	0.98	0.98	0.98	83	26: Mexican_Mint
25	0.95	0.86	0.90	91	27: Mint
26	1.00	1.00	1.00	68	28: Neem
27	1.00	1.00	1.00	86	29: Oleander
28	0.99	0.99	0.99	76	30: Palak(Spinach)
29	0.96	1.00	0.98	81	31: Papaya
30	0.92	0.98	0.95	60	32: Parijata
31	1.00	0.98	0.99	87	33: Peepal
32	1.00	0.99	0.99	80	34: Pomegranate
33	1.00	1.00	1.00	73	35: Rasna
34	0.97	0.97	0.97	73	36: Rose
35	1.00	0.99	0.99	79	37: Rose_apple
36	0.94	0.89	0.92	94	38: Roxburgh_fig
37	1.00	0.99	0.99	85	39: Sandalwood
38	0.99	0.97	0.98	95	40: Seethapala
39	0.99	1.00	0.99	75	41: Tamarind
40	0.84	0.94	0.89	81	42: Tulsi
41	0.97	0.96	0.97	79	43: Turmeric
42	0.99	0.99	0.99	83	44: ashoka
43	0.97	0.99	0.98	72	
44	0.97	0.95	0.96	73	
accuracy			0.97	3456	
macro avg	0.97	0.97	0.97	3456	
weighted avg	0.97	0.97	0.97	3456	

Xception

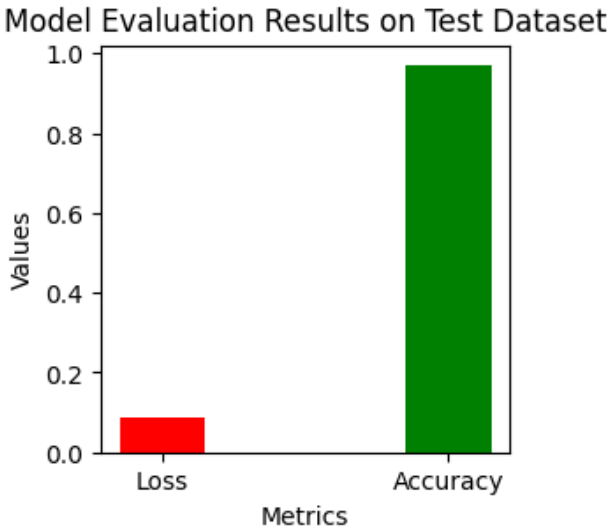
Classification Report					Class Names
	precision	recall	f1-score	support	0: Aloevera
					1: Amla
0	0.99	0.95	0.97	79	2: Arive-Dantu
1	0.93	0.84	0.88	88	3: Basale
2	1.00	1.00	1.00	80	4: Betel
3	1.00	1.00	1.00	74	5: Bhrami
4	1.00	1.00	1.00	76	6: Coriender
5	0.96	0.99	0.98	82	7: Crape_Jasmine
6	1.00	1.00	1.00	75	8: Curry
7	1.00	1.00	1.00	54	9: Drumstick
8	1.00	1.00	1.00	73	10: Fenugreek
9	0.97	1.00	0.99	75	11: Guava
10	1.00	0.98	0.99	84	12: Henna
11	0.99	1.00	0.99	80	13: Hibiscus
12	0.97	0.90	0.93	68	14: Honge
13	1.00	1.00	1.00	70	15: Indian_Beech
14	0.93	1.00	0.96	66	16: Indian_Mustard
15	1.00	1.00	1.00	64	17: Insulin
16	1.00	1.00	1.00	82	18: Jackfruit
17	0.99	0.95	0.97	78	19: Jamaica_Cherry-Gasagase
18	0.99	1.00	0.99	71	20: Jamun
19	1.00	1.00	1.00	66	21: Jasmine
20	1.00	1.00	1.00	75	22: Karanda
21	0.99	1.00	0.99	80	23: Lemon
22	1.00	0.99	0.99	69	24: Mango
23	1.00	1.00	1.00	73	25: Marigold
24	1.00	0.99	0.99	83	26: Mexican_Mint
25	0.94	0.93	0.94	91	27: Mint
26	1.00	1.00	1.00	68	28: Neem
27	1.00	1.00	1.00	86	29: Oleander
28	1.00	1.00	1.00	76	30: Palak(Spinach)
29	0.99	1.00	0.99	81	31: Papaya
30	0.98	1.00	0.99	60	32: Parijata
31	1.00	1.00	1.00	87	33: Peepal
32	1.00	1.00	1.00	80	34: Pomegranate
33	1.00	1.00	1.00	73	35: Rasna
34	1.00	1.00	1.00	73	36: Rose
35	1.00	1.00	1.00	79	37: Rose_apple
36	0.96	0.84	0.90	94	38: Roxburgh_fig
37	1.00	1.00	1.00	85	39: Sandalwood
38	1.00	0.99	0.99	95	40: Seethapala
39	1.00	0.99	0.99	75	41: Tamarind
40	0.73	0.95	0.82	81	42: Tulsi
41	0.95	0.96	0.96	79	43: Turmeric
42	1.00	1.00	1.00	83	44: ashoka
43	0.99	1.00	0.99	72	
44	0.99	0.95	0.97	73	
accuracy			0.98	3456	
macro avg	0.98	0.98	0.98	3456	
weighted avg	0.98	0.98	0.98	3456	

MobileNetV2

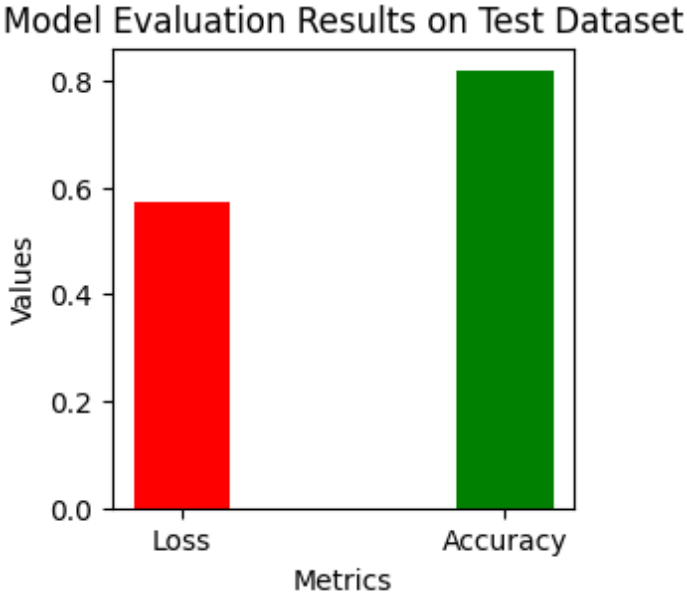
Classification Report					Class Names
	precision	recall	f1-score	support	0: Aloevera
					1: Amla
0	0.99	0.99	0.99	79	2: Arive-Dantu
1	0.93	0.89	0.91	88	3: Basale
2	1.00	1.00	1.00	80	4: Betel
3	1.00	1.00	1.00	74	5: Bhrami
4	1.00	0.99	0.99	76	6: Coriender
5	0.99	0.96	0.98	82	7: Crape_Jasmine
6	1.00	1.00	1.00	75	8: Curry
7	1.00	1.00	1.00	54	9: Drumstick
8	1.00	1.00	1.00	73	10: Fenugreek
9	1.00	1.00	1.00	75	11: Guava
10	1.00	1.00	1.00	84	12: Henna
11	1.00	1.00	1.00	80	13: Hibiscus
12	0.98	0.96	0.97	68	14: Honge
13	1.00	1.00	1.00	70	15: Indian_Beech
14	1.00	0.97	0.98	66	16: Indian_Mustard
15	1.00	1.00	1.00	64	17: Insulin
16	1.00	1.00	1.00	82	18: Jackfruit
17	0.94	0.95	0.94	78	19: Jamaica_Cherry-Gasagase
18	1.00	1.00	1.00	71	20: Jamun
19	1.00	1.00	1.00	66	21: Jasmine
20	1.00	1.00	1.00	75	22: Karanda
21	1.00	1.00	1.00	80	23: Lemon
22	1.00	1.00	1.00	69	24: Mango
23	1.00	1.00	1.00	73	25: Marigold
24	1.00	1.00	1.00	83	26: Mexican_Mint
25	0.97	0.99	0.98	91	27: Mint
26	1.00	1.00	1.00	68	28: Neem
27	1.00	1.00	1.00	86	29: Oleander
28	1.00	1.00	1.00	76	30: Palak(Spinach)
29	0.99	1.00	0.99	81	31: Papaya
30	1.00	1.00	1.00	60	32: Parijata
31	1.00	0.99	0.99	87	33: Peepal
32	1.00	1.00	1.00	80	34: Pomegranate
33	1.00	1.00	1.00	73	35: Rasna
34	1.00	1.00	1.00	73	36: Rose
35	1.00	1.00	1.00	79	37: Rose_apple
36	0.93	0.94	0.93	94	38: Roxburgh_fig
37	1.00	0.99	0.99	85	39: Sandalwood
38	0.99	1.00	0.99	95	40: Seethapala
39	1.00	1.00	1.00	75	41: Tamarind
40	0.85	1.00	0.92	81	42: Tulsi
41	1.00	0.95	0.97	79	43: Turmeric
42	1.00	1.00	1.00	83	44: ashoka
43	1.00	0.99	0.99	72	
44	0.99	0.96	0.97	73	
accuracy			0.99	3456	
macro avg	0.99	0.99	0.99	3456	
weighted avg	0.99	0.99	0.99	3456	

DenseNet121

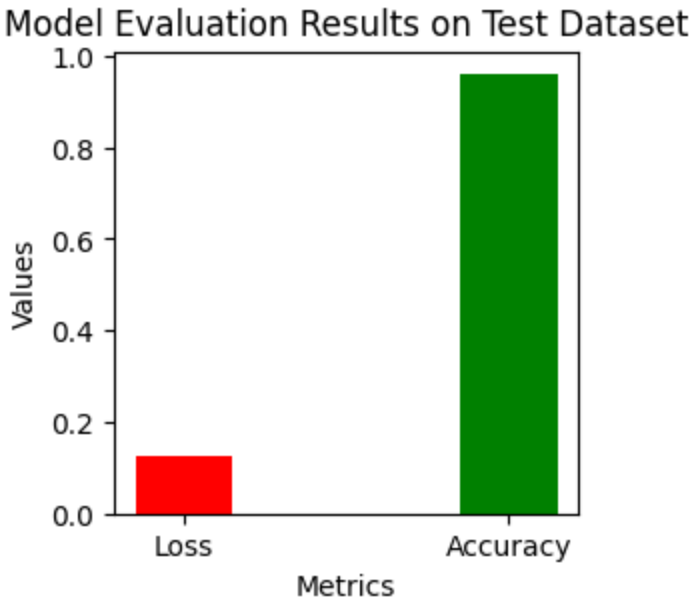
Fig 5.1: Classification report for different models



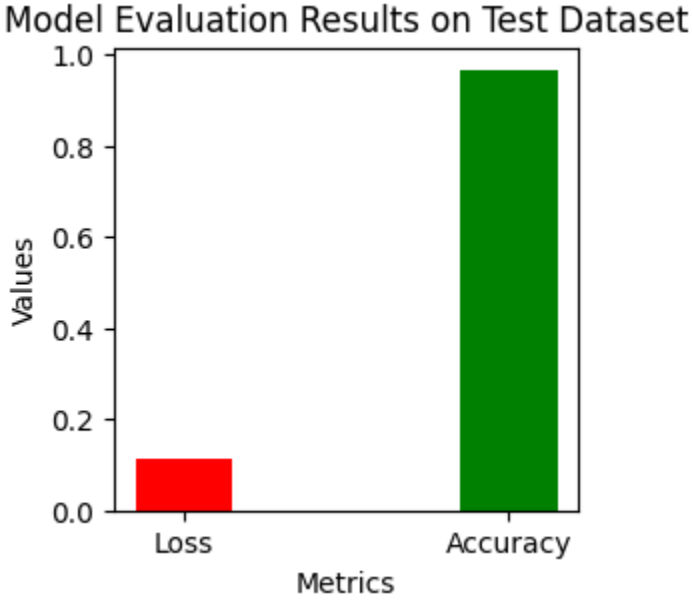
VGG19



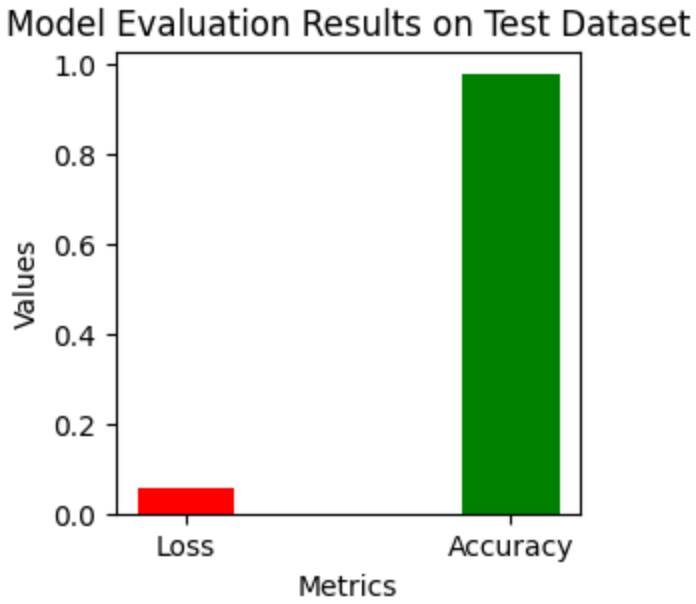
ResNet50



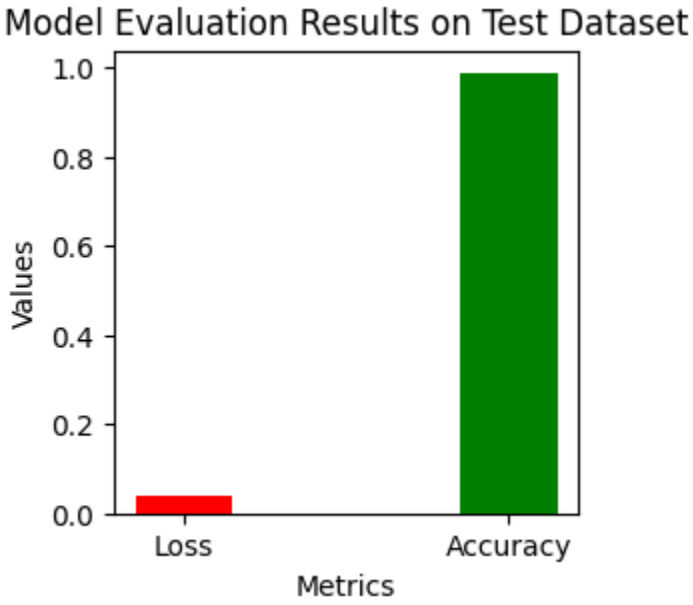
InceptionV3



Xception



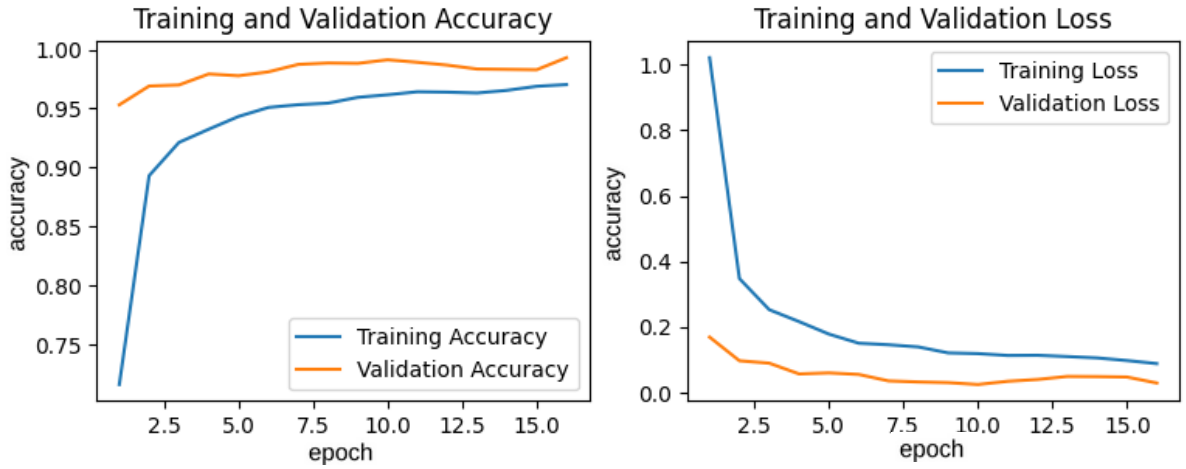
MobileNetV2



DenseNet121

Graph 5.3: Testing loss and accuracy graph of models

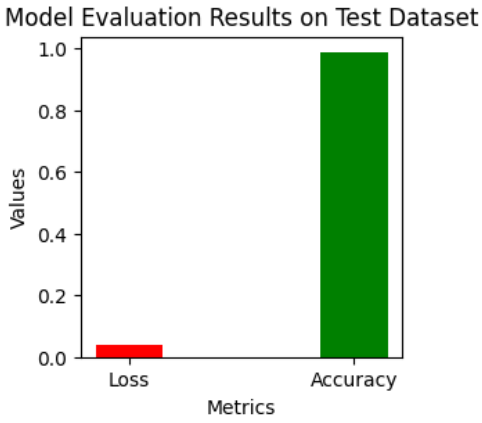
The fig demonstrates the traning accuracy and validation accuracy alongwith training loss and validation of stacked model applied. The training accuracy is 98.94% and training loss is reduced from 10.21% to 0.88%. The validation accuracy is 99% and validation loss is reduced from 1.6% to 0.29%.



Graph 5.4: Training and Validation accuracy and loss of stacked models

The testing accuracy is 98.94% and testing loss is 0.29%

```
54/54 [==============================] - 233s 139ms/step - loss: 0.0299 - accuracy: 0.9893
Loss: 0.02990795485675335
Accuracy: 0.9892939925193787
```



Graph 5.5: Testing accuracy and loss of stacked models

The classification report of individual classes with their accuracy, macro average and weighted average. The f1-score accuracy is 99%. The macro average and weighted average of precision , recall and f1-score is same and equal to 99%.

Classification Report					Class Names
	precision	recall	f1-score	support	0: Aloevera
					1: Amla
0	0.99	0.99	0.99	79	2: Arive-Dantu
1	0.93	0.89	0.91	88	3: Basale
2	1.00	1.00	1.00	80	4: Betel
3	1.00	1.00	1.00	74	5: Bhrami
4	1.00	0.99	0.99	76	6: Coriender
5	0.99	0.96	0.98	82	7: Crape_Jasmine
6	1.00	1.00	1.00	75	8: Curry
7	1.00	1.00	1.00	54	9: Drumstick
8	1.00	1.00	1.00	73	10: Fenugreek
9	1.00	1.00	1.00	75	11: Guava
10	1.00	1.00	1.00	84	12: Henna
11	1.00	1.00	1.00	80	13: Hibiscus
12	0.98	0.96	0.97	68	14: Honge
13	1.00	1.00	1.00	70	15: Indian_Beech
14	1.00	0.97	0.98	66	16: Indian_Mustard
15	1.00	1.00	1.00	64	17: Insulin
16	1.00	1.00	1.00	82	18: Jackfruit
17	0.94	0.95	0.94	78	19: Jamaica_Cherry-Gasagase
18	1.00	1.00	1.00	71	20: Jamun
19	1.00	1.00	1.00	66	21: Jasmine
20	1.00	1.00	1.00	75	22: Karanda
21	1.00	1.00	1.00	80	23: Lemon
22	1.00	1.00	1.00	69	24: Mango
23	1.00	1.00	1.00	73	25: Marigold
24	1.00	1.00	1.00	83	26: Mexican_Mint
25	0.97	0.99	0.98	91	27: Mint
26	1.00	1.00	1.00	68	28: Neem
27	1.00	1.00	1.00	86	29: Oleander
28	1.00	1.00	1.00	76	30: Palak(Spinach)
29	0.99	1.00	0.99	81	31: Papaya
30	1.00	1.00	1.00	60	32: Parijata
31	1.00	0.99	0.99	87	33: Peepal
32	1.00	1.00	1.00	80	34: Pomegranate
33	1.00	1.00	1.00	73	35: Rasna
34	1.00	1.00	1.00	73	36: Rose
35	1.00	1.00	1.00	79	37: Rose_apple
36	0.93	0.94	0.93	94	38: Roxburgh_fig
37	1.00	0.99	0.99	85	39: Sandalwood
38	0.99	1.00	0.99	95	40: Seethapala
39	1.00	1.00	1.00	75	41: Tamarind
40	0.85	1.00	0.92	81	42: Tulsi
41	1.00	0.95	0.97	79	43: Turmeric
42	1.00	1.00	1.00	83	44: ashoka
43	1.00	0.99	0.99	72	
44	0.99	0.96	0.97	73	
accuracy			0.99	3456	
macro avg	0.99	0.99	0.99	3456	
weighted avg	0.99	0.99	0.99	3456	

Fig 5.2: Classification report of stacked models

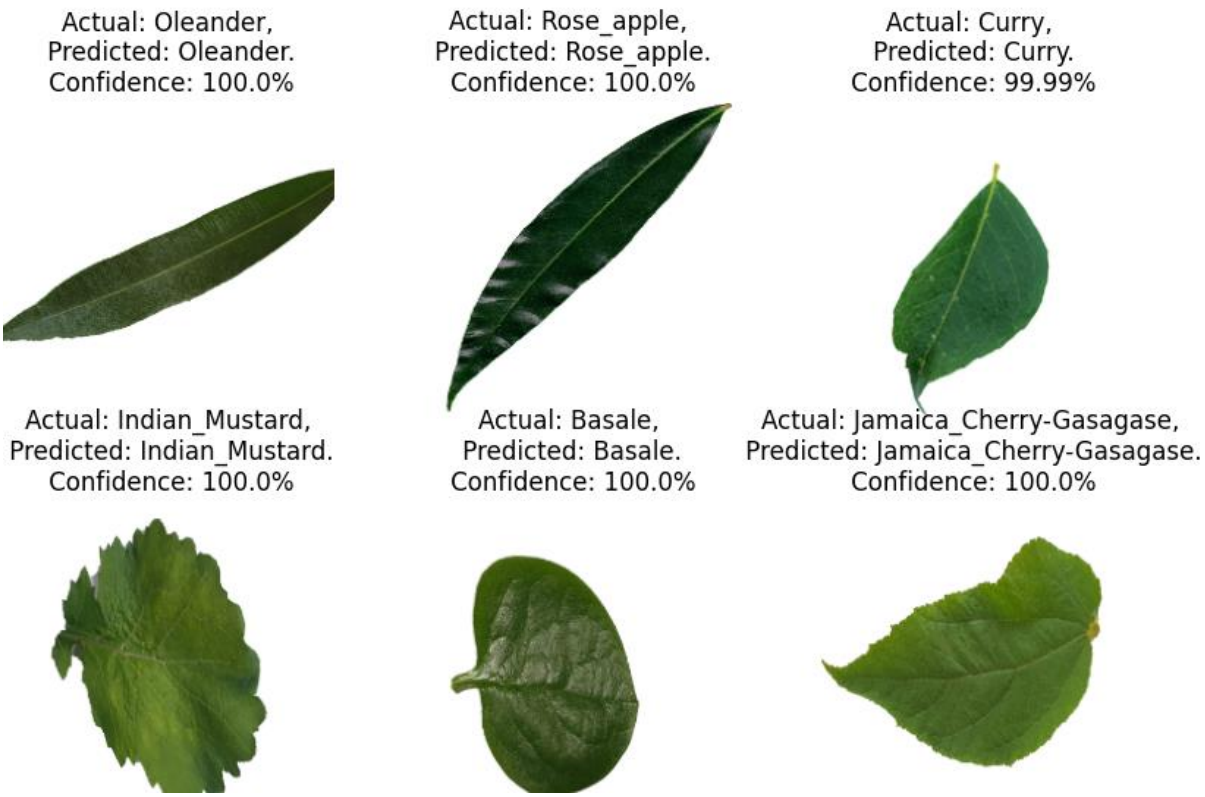


Fig 5.3: Actual v/s predicted classes of stacked models

Finally, we made the basic structure for user-interface using front-end web development tools and flask tool for deployment.

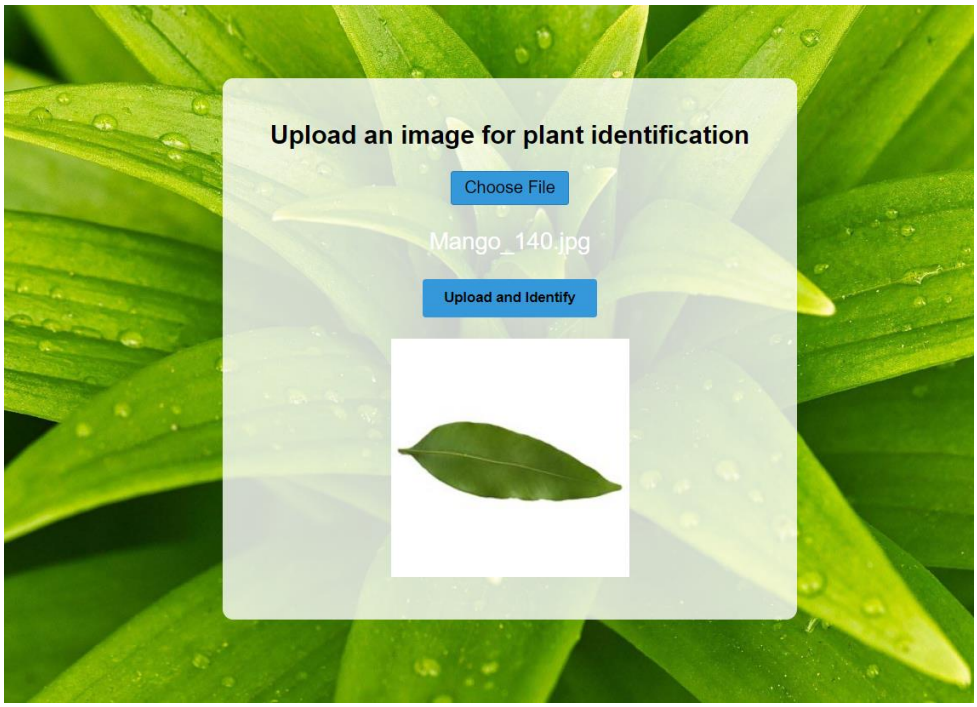


Fig 5.4: User interface of uploading the image

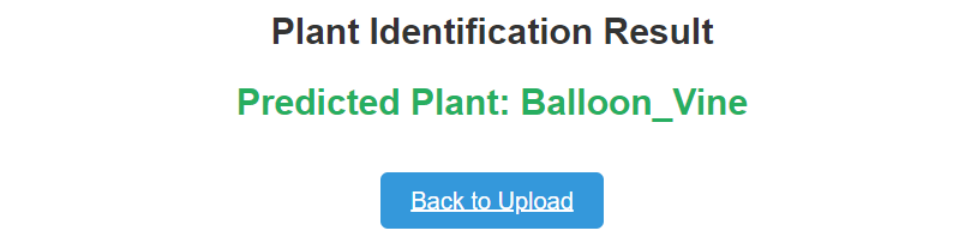


Fig 5.5: Result of uploaded the image

Chapter 6: Conclusions and Future Scope

6.1 Conclusion

The project has found its way to have significant effects in the detection of plants which have herbal significance for the application of deep learning tools. By implementing these cutting-edge methods, we obtained the desired results with potential levels of applications in pharmacy.

The core findings are achieved through models to identify various features and patterns which reflect their robustness and high efficiency. We also achieved a high level of precision and recall by doing continuous experimentation with our work which indicates that our model is quite reliable in separating closely related species.

The proposed classifier was based on the stacking of the top three models and the output was used as the final classifier. The model proposed outperformed the pre-trained models like VGG19, ResNet50, InceptionV3, Xception, MobileNetV2, and DenseNet121 by achieving an accuracy of 98.98% on the test dataset, and a macro average of 99%, a weighted average of 99% of each class.

The project is quite adaptable to a wide range of plant images ranging from high-resolution images to varied climatic conditions. This characteristic suggests that our work is fit to solve real-world problems.

Although we achieved many good results still there are many areas where our work needs to be improved and re-defined. One significant limitation is that our dataset faces challenges during generalization across various geographical regions. We need a more extensive set of data which can provide more diversity with a higher range of image resolutions.

The computational resources require a more practical approach with a high range of scalability to become more feasible for deployment. We plan to address and resolve all these issues in our future work.

In spite of the above-mentioned limitations, this project makes a valuable contribution to field of technology and science. It succeeds in algorithmic innovations which involves development and optimization of networks and architecture in such a way that we push the boundaries of tech world by helping researches in medicinal field to cure various diseases in an organic manner. Advance data handling and preprocessing methods were adopted to broaden the field of science by addressing the challenges of identification of various species by separating them from one another on basics of their color, texture, and healing properties.

The combination of cross-world collaboration between science and technology contributes towards the integration of various tech equipment in the biological domain to solve problems in a holistic manner.

Transparency in decision-making is ensured through ethical AI considerations. It contributes in levelling up education for researchers, practitioners, students and teachers through human-computer interaction by addressing issues in picking up plants which have high medicinal values through user experience.

In conclusion, the process of identification of medicinal plants advances various domains of botany by utilizing resources from the field of computer science and technology. Our work can be improved by using more advanced tools in the near future and by broadening our horizon of knowledge to contribute towards the healthcare domain in an effective manner which can help in the ailment of incurable diseases without the use of any harmful chemicals.

6.2 Future Scope

We need to work on more of the project in future as technology continues to evolve. Future work may involve the compilation of sensor-based data focusing on text description of plant uses, chemical composition, data fusion with emerging AI trends.

We plan to enhance confidence level in prediction by using probabilistic modelling techniques to produce practical applications from user feedback. Also, we plan to engage efforts on global scale from botany experts to enrich training data with less time consumption.

We can further optimize real-time application through cloud computing and mobile application to enhance accessibility by focusing on handy processing of application. A more dynamic approach can be made by incorporating blockchain for data integrity in near future to provide a secure framework for recording and validating images.

References

[1] https://ccrumresin/ViewData/Multiple?mid=1640

[2] S. Sachar and A. Kumar, “Deep
ensemble learning for automatic medicinal leaf identification,” International Journal of Information Technology, vol. 14, no. 6, pp. 3089–3097, Aug. 2022, doi 101007/s41870-022-01055-z.

[3] Naeem, S. et al. (2021) ‘The classification of medicinal plant leaves based on multispectral and texture feature using machine learning approach’, Agronomy, 11(2), p. 263. doi:10.3390/agronomy11020263.

[4] Manoharan J, S. (2021) ‘Flawless detection of herbal plant leaf by machine learning classifier through two-stage authentication procedure’, June 2021, 3(2), pp. 125–139. doi:10.36548/jaicn.2021.2.005.

[5] Azadnia, R. and Kheiralipour, K. (2021) ‘Recognition of leaves of different medicinal plant species using a robust image processing algorithm and artificial neural networks classifier’, Journal of Applied Research on Medicinal and Aromatic Plants, 25, p. 100327. doi: 10.1016/j.jarmap.2021.100327.

[6] Muneer, A. and Fati, S.M. (2020) ‘Efficient and automated herbs classification approach based on shape and texture features using Deep Learning’, IEEE Access, 8, pp. 196747–196764. doi:10.1109/access.2020.3034033.

[7] Pushpanathan, K. et al. (2020) ‘Machine learning in Medicinal plants recognition: A Review’, Artificial Intelligence Review, 54(1), pp. 305–327. doi:10.1007/s10462-020-09847-0.

[8] Akter, R. and Hosen, M.I. (2020) ‘CNN-based leaf image classification for Bangladeshi medicinal plant recognition’, 2020 Emerging Technology in Computing, Communication and Electronics (ETCCE) [Preprint]. doi:10.1109/etcce51779.2020.9350900.

[9] A. Kaya, A. S. Keceli, C. Catal, H. Y. Yalic, H. Temucin, and B. Tekinerdogan, “Analysis of transfer learning for deep neural network based plant classification models,” Computers and Electronics in Agriculture, vol. 158, pp. 20–29, Mar. 2019, doi: https://doi.org/10.1016/j.compag.2019.01.041.

[10] M. R. Dileep and P. N. Pournami, "AyurLeaf: A Deep Learning Approach for Classification of Medicinal Plants," TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), Kochi, India, 2019, pp. 321-325, doi: 10.1109/TENCON.2019.8929394.

[11] Sabarinathan, C., Hota, A., Raj, A., Dubey, V.K. and Ethirajulu, V., 2018. Medicinal plant leaf recognition and show medicinal uses using convolutional neural network. Int. J. Glob. Eng, 1(3), pp.120-127.

[12] Sabu, A., Sreekumar, K. and Nair, R.R. (2017) ‘Recognition of Ayurvedic medicinal plants from leaves: A computer vision approach’, 2017 Fourth International Conference on Image Information Processing (ICIIP) [Preprint]. doi:10.1109/iciip.2017.8313782.

[13] S. H. Lee, C. S. Chan, S. J. Mayo, and P. Remagnino, “How deep learning extracts and learns leaf features for plant classification,” Pattern Recognition, vol. 71, pp. 1–13, Nov. 2017, doi: https://doi.org/10.1016/j.patcog.2017.05.015.

[14] S. H. Lee, C. S. Chan, P. Wilkin and P. Remagnino, "Deep-plant: Plant identification with convolutional neural networks," 2015 IEEE International Conference on Image Processing (ICIP), Quebec City, QC, Canada, 2015, pp. 452-456, doi: 10.1109/ICIP.2015.7350839.

[15] Akhtar, Z. (2021) Xception: Deep learning with depth-wise separable convolutions, OpenGenus IQ: Computing Expertise & Legacy. Available at: https://iq.opengenus.org/xception-model/ (Accessed: 30 November 2023).

[16] T, A.N. (2021) Inception V3 model architecture, OpenGenus IQ: Computing Expertise & Legacy. Available at: https://iq.opengenus.org/inception-v3-model-architecture/ (Accessed: 30 November 2023).

[17] VGG-19 architecture [39]. VGG-19 has 16 convolution layers grouped into... Available at: https://www.researchgate.net/figure/VGG-19-Architecture-39-VGG-19-has-16-convolution-layers-grouped-into-5-blocks-After_fig5_359771670 (Accessed: 30 November 2023).

[18] Schematic overview of the personalized VGG-19 network ... - researchgate. Available at: https://www.researchgate.net/figure/Schematic-overview-of-the-personalized-VGG-19-network-architecture-with-description-of_fig4_340554560 (Accessed: 30 November 2023).

[19] Papers with code - mobilenetv2 explained (no date) Explained | Papers With Code. Available at: https://paperswithcode.com/method/mobilenetv2#:~:text=MobileNetV2%20is%20a%20convolutional%20neural,are%20between%20the%20bottleneck%20layers (Accessed: 13 May 2024).

[20] Tsang, S.-H. (2019) Review: MOBILENETV2 - light weight model (image classification), Medium. Available at: https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c (Accessed: 13 May 2024).

[21] Ruiz, P. (2018) Understanding and visualizing DenseNets, Medium. Available at: https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a (Accessed: 13 May 2024).

[22] Mulugeta, A.K., Sharma, D.P. and Mesfin, A.H. (2023) Deep learning for medicinal plant species classification and recognition: A systematic review, Frontiers. Available at:

https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2023.1286088/full (Accessed: 14 May 2024).

[23] Sonia, S.V., N, D. and S, D.R. (2023) 'Medicinal plants classification by Visual Characteristics of leaves using CNN', 2023 Second International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT) [Preprint]. doi:10.1109/iceeict56924.2023.10157410.

[24] Kaya, A. et al. (2019) 'Analysis of Transfer Learning for deep neural network-based plant classification models', Computers and Electronics in Agriculture, 158, pp. 20–29. doi: 10.1016/j.compag.2019.01.041.

A

ORIGINALITY REPORT

16%	13%	12%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.researchgate.net Internet Source	3%
2	ir.juit.ac.in:8080 Internet Source	2%
3	www.ncbi.nlm.nih.gov Internet Source	1%
4	Submitted to Liverpool John Moores University Student Paper	1%
5	Amala Sabu, K Sreekumar, Rahul R Nair. "Recognition of ayurvedic medicinal plants from leaves: A computer vision approach", 2017 Fourth International Conference on Image Information Processing (ICIIP), 2017 Publication	1%
6	Shobha Tyagi, Anupama Sharma, Neha Batra, Pronika Chawla, Pinki Sagar, Neha Tyagi. "Study and Identification Analysis of Health and Diseases of Medicinal and Aromatic Plants Through Machine Learning Using	1%

Random Forest and Convolutional Neural Networks", 2023 3rd International Conference on Advancement in Electronics & Communication Engineering (AECE), 2023
Publication

7	digitalcommons.du.edu Internet Source	<1 %
8	technodocbox.com Internet Source	<1 %
9	www.ir.juit.ac.in:8080 Internet Source	<1 %
10	Raisa Akter, Md Imran Hosen. "CNN-based Leaf Image Classification for Bangladeshi Medicinal Plant Recognition", 2020 Emerging Technology in Computing, Communication and Electronics (ETCCE), 2020 Publication	<1 %
11	Aydin Kaya, Ali Seydi Keceli, Cagatay Catal, Hamdi Yalin Yalic, Huseyin Temucin, Bedir Tekinerdogan. "Analysis of transfer learning for deep neural network based plant classification models", Computers and Electronics in Agriculture, 2019 Publication	<1 %
12	research.ijcaonline.org Internet Source	<1 %

13	Internet Source	<1%
14	www.mdpi.com Internet Source	<1%
15	Submitted to Higher Education Commission Pakistan Student Paper	<1%
16	assets.researchsquare.com Internet Source	<1%
17	"Proceedings of Data Analytics and Management", Springer Science and Business Media LLC, 2023 Publication	<1%
18	"Inventive Computation and Information Technologies", Springer Science and Business Media LLC, 2023 Publication	<1%
19	"Advanced Computational and Communication Paradigms", Springer Science and Business Media LLC, 2023 Publication	<1%
20	towardsdatascience.com Internet Source	<1%
21	tojqi.net Internet Source	<1%
	Submitted to Deakin University	

22	Student Paper	<1%
23	dspace.bracu.ac.bd:8080 Internet Source	<1%
24	pubag.nal.usda.gov Internet Source	<1%
25	Adibaru Kiflie Mulugeta, Durga Prasad Sharma, Abebe Haile Mesfin. "Deep learning for medicinal plant species classification and recognition: a systematic review", Frontiers in Plant Science, 2024 Publication	<1%
26	Submitted to Berlin School of Business and Innovation Student Paper	<1%
27	Submitted to University of the West Indies Student Paper	<1%
28	link.springer.com Internet Source	<1%
29	Submitted to Green University Of Bangladesh Student Paper	<1%
30	Submitted to Macao Polytechnic Institute Student Paper	<1%
31	Submitted to Panipat Institute of Engineering & Technology Student Paper	<1%

32	Samreen Naeem, Aqib Ali, Christophe Chesneau, Muhammad H. Tahir, Farrukh Jamal, Rehan Ahmad Khan Sherwani, Mahmood Ul Hassan. "The Classification of Medicinal Plant Leaves Based on Multispectral and Texture Feature Using Machine Learning Approach", Agronomy, 2021 Publication	<1 %
33	Sue Han Lee, Chee Seng Chan, Simon Joseph Mayo, Paolo Remagnino. "How Deep Learning Extracts and Learns Leaf Features for Plant Classification", Pattern Recognition, 2017 Publication	<1 %
34	Submitted to University of Bedfordshire Student Paper	<1 %
35	Veena Tapale, Rutuja Rekhawar, Arpita Savagonmath, Sainath Habre, Savita Adhav. "Phyto Leafy Recognition Using Deep Learning", 2023 7th International Conference On Computing, Communication, Control And Automation (ICCUBEA), 2023 Publication	<1 %
36	"Computer Vision and Image Processing", Springer Science and Business Media LLC, 2021 Publication	<1 %

37	Submitted to lcba Student Paper	<1%
38	dalspace.library.dal.ca Internet Source	<1%
39	www.frontiersin.org Internet Source	<1%
40	zdocs.ro Internet Source	<1%

Exclude quotes	Off	Exclude matches	< 10 words
Exclude bibliography	On		

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: …….

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

Name: _____ __**Department:** _____ **Enrolment No** _____

Contact No. _____**E-mail.** _____

Name of the Supervisor: _____

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

_____

_____

UNDERTAKING

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

Complete Thesis/Report Pages Detail:

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

(Signature of Student)

FOR DEPARTMENT USE

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

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

FOR LRC USE

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

<table>
<tr><th>Copy Received on</th><th>Excluded</th><th>Similarity Index (%)</th><th colspan="2">Generated Plagiarism Report Details (Title, Abstract & Chapters)</th></tr>
<tr><td rowspan="2"></td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td rowspan="2">Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

……

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