

A Robust Deep Learning Architecture for Precise Identification and Classification of Medicinal Plant Leaves

*A Project Report submitted
in partial fulfillment of the requirements
for the award of the degree of*

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

By

- 1.Ayinaavilli Jyothi Surekha-21B01A0510
- 2.Challangi Lahari-21B01A0533
- 3.Chunduri Hari Chandra Bhargavi-21B01A0537
- 4.Durgam Rishitha-21B01A0546
- 5.Immadisetty Dhamarika Divya Sree-21B01A0561

Under the esteemed guidance of
Dr. V.V.R.Maheswara Rao M.Tech, Ph.D
Professor



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN(A)
(Approved by AICTE, Accredited by NBA & NAAC, Affiliated to JNTU Kakinada)
BHIMAVARAM – 534 202
2024 – 2025

SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN(A)
(Approved by AICTE, Accredited by NBA & NAAC, Affiliated to JNTU Kakinada)
BHIMAVARAM – 534 202

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



CERTIFICATE

This is to certify that the project entitled "A Robust Deep Learning Architecture Of Precise Identification and Classification of Medicinal Plant Leaves", is being submitted by A.Jyothi Surekha, Ch.Lahari, Ch.Bhargavi, D.Rishitha, I.D.Divya Sree bearing the Regd. No's. 21B01A0510, 21B01A0533, 21B01A0537, 21B01A0546, 21B01A0561 in partial fulfillment of the requirements for the award of the degree of "Bachelor of Technology in Computer Science & Engineering" is a record of Bonafide work carried out by her under my guidance and supervision during the academic year 2024-2025 and it has been found worthy of acceptance according to the requirements of the university.

Internal Guide

Head of the Department

External Examiner

ACKNOWLEDGEMENTS

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance has been a source of inspiration throughout this project. I take this opportunity to express our gratitude to all those who have helped me in this project.

We wish to place our deep sense of gratitude to **Sri. K. V. Vishnu Raju, Chairman of SVES**, for his constant support on each and every progressive work.

We wish to express our sincere thanks to **Dr. G. Srinivasa Rao, Principal of SVECW**, for being a source of inspiration and constant encouragement.

Our heartfelt appreciation to **Dr. P. Venkata Rama Raju, Vice-Principal of SVECW** for being a source of inspiration and constant encouragement.

We extend our profound thanks to **Dr P Srinivasa Raju, Our Director, Student Affairs, Admin, SVES-Bhimavaram** for being a source of inspiration and constant encouragement.

We are grateful to **Dr. P. Kiran Sree, Head of the Department of Computer Science & Engineering**, for his valuable pieces of advice in completing this project successfully.

Our special thanks to **Dr. V V R MaheswaraRao, Professor and guide** for his unflinching devotion and valuable suggestions throughout my project.

We are deeply indebted to **Dr. P. R. Sudha Rani, PRC** member for her valuable piece of advice in completing this project successfully.

Project Associates

A.Jyothi Surekha-21B01A0510

Ch.Lahari-21B01A0533

Ch.Bhargavi-21B01A0537

D.Rishitha-21B01A0546

I.D.Divya Sree-21B01A0561

ABSTRACT

Classifying medicinal plant leaves presents significant challenges due to the complexities of plant taxonomy and the subtle visual similarities between species. Traditional identification methods rely heavily on manual observation and expert knowledge, which are often time-consuming and prone to inaccuracies, particularly given the vast diversity of species. Environmental factors, such as lighting conditions, viewing angles, and background variations, further complicate accurate plant leaves identification. Additionally, distinguishing closely related species requires specialized botanical expertise, limiting non-experts' involvement in conservation and healthcare initiatives.

These challenges also hinder large-scale classification efforts, restricting the potential benefits of medicinal plant leaves for biodiversity conservation and healthcare innovation. Machine learning (ML) techniques automate plant leaves classification but face challenges, including generalization across diverse species and reliance on manual feature extraction, limiting accuracy and adaptability in complex datasets.

To address these challenges, this research employs a vision-based approach that leverages deep learning (DL) techniques for the accurate and rapid identification and classification of medicinal plant leaves. Specifically, Convolutional Neural Networks (CNNs) were chosen for their ability to automatically learn hierarchical features from images, thereby enhancing classification accuracy. After extensive training, validation, and testing, the model demonstrated impressive accuracy, even with complex datasets, establishing CNNs as highly effective tools for medicinal plant leaves classification.

This study highlights that identifying medicinal plant leaves enhances biodiversity and healthcare, using innovative image processing techniques to foster conservation and improve access to vital resources, supporting global health initiatives

Contents

S.No	Topic	Page No.
1. INTRODUCTION		02
2. SYSTEM ANALYSIS		06
2.1 Existing System		07
2.2 Proposing System		09
2.3 Feasibility Study		11
3. SYSTEM REQUIREMENTS SPECIFICATION		14
3.1 Software Requirements		15
3.2 Hardware Requirements		16
3.3 Functional Requirements		17
3.4 Non-Functional Requirements		20
4. SYSTEM DESIGN		21
4.1 Introduction		22
4.2 UML Diagrams		31
5. SYSTEM IMPLEMENTATION		36
5.1 Introduction		37
5.2 Project Modules		38
5.3 Algorithms		44
5.4 Screens		49
6.SYSTEM TESTING		60
6.1 Introduction		61
6.2 Testing Methods		63
6.3 Test Cases		65
7.CONCLUSION		69
8. BIBLOGRPAHY		72
9. APPENDIX		74
9.1 APPENDIX-A		75
9.2 APPENDIX-B		76

List of Figures

S. No.	Topic	Page No.
1.	InceptionV3 Model of Proposed System	09
2.	Architecture diagram of proposed system	23
3.	Medicinal leaf sample images	24
4.	Top 15 Species	25
5.	Steps in Image Pre-Processing	26
6.	Comparison table	30
7.	Use case Diagram of proposed System	32
8.	Sequence diagram of the proposed system	33
9.	Activity diagram of proposed system	34
10.	Class Diagram of Proposed System	35
11.	Project flowchart with different modules	38
12.	Rotation of images	40
13.	Workflow diagram of InceptionV3	45
14.	Workflow diagram of VGG16	46
15.	Workflow diagram of NASNetMobile	47
16.	Workflow diagram of DenseNet121	49
17.	Importing Required Libraries	49
18.	Downloading the dataset from Kaggle	49
19.	Identifying the top_15 leaves	50
20.	Converting the top_15 leaves into a dataset	50
21.	Loading and Preprocessing images	51
22.	Displaying Sample Images	51
23.	Loading the model	52
24.	Training Progress	52
25.	Training and Validation Accuracy, Loss	53
26.	Model Performance Over Epochs	53
27.	Correctly predicting the images with their uses	54
28.	Confusion Matrix	55
29.	Precision over epochs	65
30.	F1 score over epochs	66
31.	Recall over epochs	66
32.	Confusion Matrix of InceptionV3	67
33.	Confusion Matrix of VGG16	67
34.	Confusion Matrix DenseNet121	68
35.	Confusion Matrix NasnetMobile	68

INTRODUCTION

Medicinal plants have been an integral part of human civilization for centuries, serving as the primary source of remedies for various ailments. Even in modern medicine, these plants continue to play a significant role, with many pharmaceutical compounds derived from botanical sources. However, the accurate identification and classification of medicinal plant species present a considerable challenge due to the subtle morphological differences between species, environmental variations, and the specialized expertise required for precise classification. Traditional plant identification methods rely on manual observation and expert knowledge, which are often time-consuming, subjective, and prone to errors. Additionally, these methods require extensive training and experience, making them less accessible to non-experts and limiting their application on a large scale.

Our approach utilizes Convolutional Neural Networks (CNNs), specifically the Inception V3 model, to classify medicinal leaves with high accuracy. The system is designed to analyze leaf images, extract essential features, and predict their medicinal properties. By integrating a user-friendly interface, the project aims to make medicinal plant identification accessible to a wider audience, including researchers, farmers, and healthcare practitioners.

In recent years, advancements in deep learning have significantly enhanced the accuracy and efficiency of medical and botanical research. One such application is medicinal leaf recognition, which plays a crucial role in identifying plants with medicinal properties for pharmaceutical, healthcare, and agricultural industries. The accurate identification of medicinal plants is essential in preserving biodiversity, promoting herbal medicine, and ensuring the correct usage of plant-based remedies.

Role of Image Processing

Image Processing is a fundamental technique in computer vision that allows for the analysis and enhancement of digital images. In this project, image processing techniques are employed to preprocess leaf images, improving the efficiency of the deep learning model. Key image processing steps include:

- **Noise Reduction:** Removing unnecessary distortions to enhance clarity.
- **Segmentation:** Isolating the leaf from the background to focus on essential features.
- **Feature Extraction:** Identifying crucial leaf characteristics such as shape, texture, and vein patterns.
- **Contrast Adjustment & Normalization:** Ensuring consistency in image quality

By integrating Convolutional Neural Networks (CNN) with advanced image processing techniques, our project utilizes the Inception V3 model to accurately classify medicinal leaves. This approach enhances the reliability and efficiency of medicinal plant identification, contributing to fields like herbal medicine, agriculture, and botany research.

The necessity of developing efficient, scalable, and accurate plant classification systems has led to increased interest in artificial intelligence (AI) and computer vision techniques. Machine learning (ML) models have been extensively explored to address the challenges associated with traditional plant identification methods. While ML models improve efficiency, their reliance on handcrafted features and domain expertise limits their ability to generalize across diverse plant species.

The Objectives of the Proposed System are:

1. To develop a robust deep learning model using CNN.
2. Identify key features of medicinal leaves.
3. Provide insights into the medicinal properties and applications of identified leaves.
4. Implement a system that allows users to upload images and receive real time

Deep learning (DL) has revolutionized image-based classification tasks by automating feature extraction and enhancing accuracy. Among various deep learning techniques, Convolutional Neural Networks (CNNs) have proven to be highly effective for image classification due to their ability to learn hierarchical patterns from raw image data.

One of the critical challenges in plant leaf classification is the impact of environmental factors such as variations in lighting, background noise, and different angles of image capture. These variations significantly affect the accuracy of traditional machine learning models, making it difficult to achieve consistent results. However, CNNs are capable of overcoming these challenges by learning robust feature representations directly from image data. Through extensive training, validation, and testing, the developed CNN model demonstrates superior performance in accurately classifying medicinal plant leaves despite environmental inconsistencies.

Numerous models exist for medicinal plant classification, each offering unique advantages in terms of accuracy, efficiency, and feature extraction. These models

- * The AlexNet Model
- * The VGG16 Model
- * The ResNet Model
- * The MobileNet Model
- * The DenseNet Model
- * The Inception V3 Model.

The Inception V3 Model, a widely recognized model for efficient image classification, reducing computational costs with factorized convolutions.

The Inception V3 model, a deep learning architecture, is an advanced version of the Inception series, designed to optimize image classification tasks. It introduces key enhancements such as factorized convolutions, asymmetric convolutions, auxiliary classifiers, and label smoothing, making it highly efficient for complex visual recognition problems.

Technological Framework:

At the core of our project lies the **Inception V3 model**, a state-of-the-art convolutional neural network (CNN) known for its exceptional accuracy in image classification tasks. Leveraging deep learning techniques, our system efficiently classifies medicinal leaves by analyzing their distinct features. The model is trained on an extensive dataset, ensuring robustness and adaptability across various leaf species. The integration of TensorFlow and Keras facilitates smooth implementation and optimization of the neural network.

User-Centric Design:

Our project prioritizes accessibility and ease of use, incorporating a user-friendly interface designed to cater to a broad audience, including researchers, herbal practitioners, and the general public. The interface allows users to upload an image of a leaf and receive real-time classification results along with relevant medicinal properties. The seamless integration with Streamlit ensures a responsive and interactive experience, making the system intuitive even for non-technical users.

Real-World Applicability:

The real-world applicability of our system is a key focus, addressing the challenges faced in medicinal plant identification. Traditional identification methods can be time-consuming and require expert knowledge. Our AI-powered solution offers a quick, accurate, and scalable approach, aiding botanists, researchers, and healthcare professionals in identifying medicinal plants with ease.

This study explores the potential of deep learning-based image classification techniques for medicinal plant identification. By leveraging CNNs, the research aims to develop a reliable, scalable, and automated solution for plant leaf classification, addressing key challenges in traditional methods. The findings of this study pave the way for future advancements in AI-driven botanical research, fostering innovation in healthcare and environmental conservation.

In the upcoming sections, we will explore the design comparisons and improvements, providing insights into the evaluation of different CNN architectures such as Inception V3, VGG16, DenseNet121, and ResNet50 to highlight performance enhancements and model optimizations. Our goal is not only technological advancement but also the creation of a practical and impactful tool that contributes to medicinal research and healthcare innovations

2.SYSTEM ANALYSIS

SYSTEM ANALYSIS

2.1 Existing System

Machine Learning based Medicinal Leaf Recognition

Traditional medicinal plant identification relies on experts analyzing leaf structures, textures, and colors, which is time-consuming and prone to errors. Digital herbariums and taxonomy databases help, but manual comparison limits efficiency. Some mobile apps use image-matching, but accuracy is affected by lighting, background clutter, and plant growth stages. While deep learning offers promising alternatives, challenges like data quality, generalization, and adaptability to new species remain.

- 1. Lower Computational Cost:** Machine learning models require less processing power compared to deep learning, making them suitable for low-resource environments.
- 2. Easier to Interpret:** Feature-based models allow researchers to understand the importance of specific leaf characteristics in classification.
- 3. Faster Training Time:** Training machine learning models generally takes less time compared to deep learning, especially for small datasets.
- 4. Works Well with Small Datasets:** Machine learning models can perform well even with limited training data, whereas deep learning requires large datasets for optimal performance.

Drawbacks of existing system:

1. Dependence on Expertise

- Traditional methods for identifying medicinal plants rely heavily on experienced botanists, clinicians, or herbalists.
- Beginners or non-experts struggle with identification due to the need for specialized knowledge in botany.
- Variability in plant morphology due to environmental factors (e.g., season, climate, soil conditions) can make identification even more challenging.

2. Data Quality Dependence

- Machine learning models require diverse and high-quality datasets to perform well.
- Incomplete datasets with missing plant species or insufficient variations in lighting, angles, and backgrounds can result in poor model performance.
- Bias in the dataset (e.g., more samples of one plant species than others) can lead to skewed predictions, where the model favors more frequently seen species.

3. Overfitting Risks

- Overfitting occurs when a model learns the noise in the training data rather than the actual patterns, leading to poor performance on unseen data.
- This problem is more common in deep learning models, which have a high capacity to memorize training samples rather than generalizing features.
- Overly complex models require careful tuning, including techniques like regularization, dropout, and data augmentation to prevent overfitting.
- A lack of diverse testing data may mask overfitting issues, making real-world deployment unreliable.

4. Feature Extraction Challenges

- Traditional feature extraction methods, such as edge detection, texture analysis, and color histograms, may not be sufficient for distinguishing similar plant species.
- Deep learning models like CNNs automatically extract features, but they require significant computational resources and well-annotated data to learn effectively.
- Poor feature extraction can result in misleading classification, as the model may focus on irrelevant details rather than key botanical characteristics.
- Variations in leaf size, shape, venation, and surface texture across different growth stages add complexity to the feature extraction process.

5. Adaptability Issues

- Machine learning models struggle to identify new plant species that were not part of the training dataset, leading to limited real-world use.
- Models often fail when presented with variations such as hybrid species or genetically modified plants.
- Domain adaptation techniques, such as transfer learning, can help models generalize better, but they require additional training on new datasets.
- Continuous updating and retraining of the model with new species data is necessary for long-term accuracy and robustness.

2.2 Proposed System

The proposed system utilizes Inception V3, a deep learning-based CNN model, for medicinal plant identification. Inception V3 improves performance through several architectural enhancements, making it superior to traditional models like ResNet50 and MobileNet in terms of efficiency, feature extraction, and adaptability.

Instead of using large convolutional filters, Inception V3 employs factorized convolutions that break down larger convolutions into smaller ones (e.g., a 5×5 convolution is factorized into two 3×3 convolutions). This reduces computational complexity while maintaining or even improving accuracy.

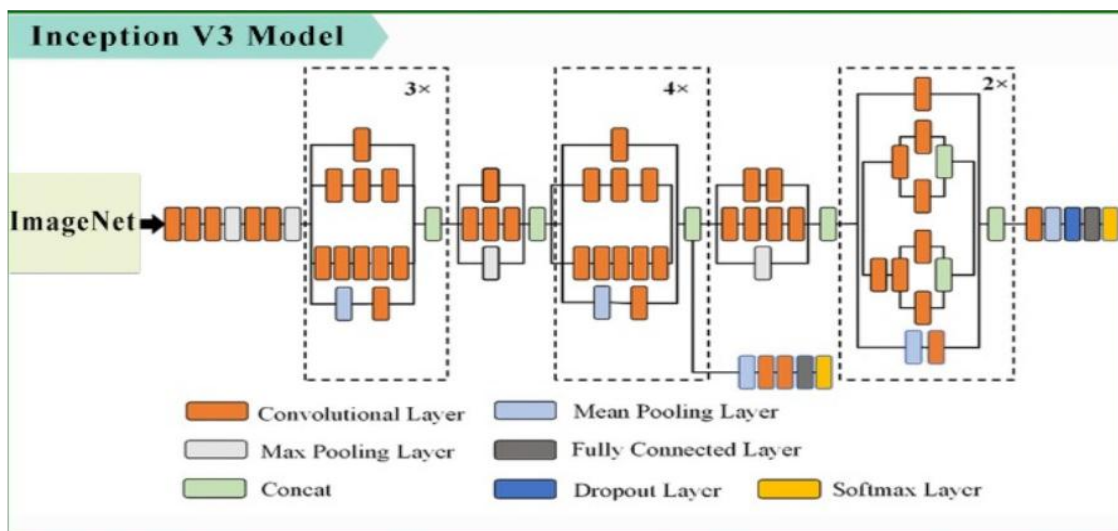


Fig.1:InceptionV3 Model of Proposed System

Our project focuses on the automated classification of medicinal plant leaves using deep learning, specifically Convolutional Neural Networks (CNNs). The model analyzes leaf images to accurately identify plant species, enhancing precision and generalization. Trained on a diverse dataset, it achieves high accuracy, supporting biodiversity conservation and healthcare applications. This research improves plant identification through advanced image processing, promoting accessibility and scientific advancements.

Advantages of Proposed System

1. Improved Accuracy

- Utilizes deep learning models, particularly CNNs, to enhance the precision of medicinal plant identification.
- Learns complex patterns from leaf images, reducing errors caused by

human judgment.

- o Ensures consistent classification, overcoming the limitations of subjective manual identification.
- o Models like Inceptionv3, VGG, ResNet and DenseNet significantly improve classification and accuracy due to their deep representations.

2. Fast and Efficient Identification

- o Automates plant classification, eliminating the time-consuming process of expert-based identification.
- o Enables real-time identification, making it suitable for field research and mobile applications.
- o Reduces dependency on human expertise, making plant identification accessible to a wider audience.

3. Automatic Feature Extraction

- o CNNs automatically extract key leaf features, such as shape, texture, venation, and color.
- o Eliminates the need for manual feature engineering, improving efficiency
- o Adapts to complex datasets, ensuring robust performance across diverse plant species.

4. Scalability and Adaptability

- o Can be continuously trained with new data to recognize additional plant species over time.
- o Adapts to real-world conditions such as variations in lighting, angles, and backgrounds.
- o Suitable for large-scale applications, including biodiversity research and agricultural studies.

5. Support for Biodiversity Conservation

- o Helps document and preserve rare and endangered medicinal plant species.
- o Contributes to ecological research by providing accurate plant classification data.

- Aids conservationists in identifying and monitoring plant populations Efficiently

6. Enhanced Global Healthcare Initiatives

- Assists in pharmaceutical research by providing reliable data on medicinal plant species.
- Promotes sustainable use of natural resources for healthcare and well-being.

2.3 Feasibility Study

A feasibility study is a high-level capsule version of the entire system analysis and design Process. The study begins by classifying the problem definition. Feasibility is to determine if it is worth doing. Once an acceptance problem is generated, the analyst develops a logical model of the system. A search for alternative is analyzed carefully. In other words, feasibility Study can be considered as preliminary investigation that helps the management to take decision about whether study of system should be feasible for development or not.

The main objective of a feasibility study is to define the problem scope and assess whether a feasible and appropriate solution exists. It helps in making informed decisions about system development and implementation.

The feasibility study for this project consists of two major aspects:

- Technical Feasibility
- Legal and Ethical Feasibility

1. Technical Feasibility

- **Image Processing Capabilities:** The system utilizes Inception V3 to extract shape, texture, and vein structure from medicinal leaf images, ensuring precise classification. It applies preprocessing techniques like noise reduction, segmentation, and contrast adjustment to enhance image quality.
- **Model Performance:** The deep learning model is trained on a large dataset of medicinal plants, enabling it to achieve high accuracy and generalization. However, training requires substantial computational power, making it dependent on high-performance GPUs or cloud resources. The model's accuracy is influenced by image quality, dataset size, and training parameters.
- **UI Development:** A user-friendly interface is crucial for enabling easy interaction and real-time classification. The system can be integrated into a

web-based or desktop application using frameworks like Streamlit. It ensures seamless input of images, displays classification results, and provides an intuitive user experience.

- **Scalability:** The system is designed to be scalable, allowing for the addition of new plant species with minimal modifications. However, to maintain accuracy, continuous dataset updates and model retraining are required.

2. Legal and Ethical Feasibility

- **Intellectual property:** The model uses pre-trained CNN architectures like Inception V3 and publicly available plant datasets, ensuring compliance with open-source licensing policies. However, careful review of dataset sources and third-party libraries is essential to avoid intellectual property infringements. Any proprietary datasets used must be properly licensed or developed in-house.
- **Ethical Considerations:** The system is designed to accurately classify medicinal plants, minimizing the risk of misidentification, which could lead to incorrect medicinal use. It ensures fairness by avoiding biases in training data, making classification results reliable for research, healthcare, and agriculture. Ethical concerns such as misuse of plant identification for illegal drug production must also be considered.
- **Transparency & Fairness:** The model should provide explainable classification results, allowing users to understand why a plant was classified a certain way. Periodic validation using expert-reviewed datasets ensures continuous improvement and credibility. The system should also include a feedback mechanism for users to report misclassifications, improving overall reliability.

The feasibility of this medicinal plant classification project is supported by advancements in deep learning, image processing, and hardware acceleration. Below are the key technical aspects ensuring the project's successful implementation:

1. Availability of Deep Learning Frameworks

- The project is implemented using TensorFlow and Keras, which provide pre-trained models, optimization tools, and efficient model-building capabilities.
- These frameworks allow the use of InceptionV3, a state-of-the-art CNN model, to

extract fine-grained features from medicinal plant leaves.

- TensorFlow provides GPU/TPU acceleration, reducing training time and improving performance.

2.Efficient Image Processing & Preprocessing

- OpenCV and TensorFlow's ImageDataGenerator are used for resizing, normalization, and data augmentation to improve model generalization.
- Feature extraction techniques, such as edge detection, vein structure analysis, and texture mapping, ensure accurate classification.
- The dataset undergoes contrast enhancement and noise reduction, making the model robust to different lighting conditions and backgrounds.

3.High-Performance Computing Support

- Training deep learning models requires significant computational power, which is made feasible through GPUs (CUDA-enabled) and TPUs (Tensor Processing Units).
- Google Colab and cloud-based platforms like AWS, Google Cloud AI, and Azure ML allow scalable training and deployment.
- Batch processing and parallel computing techniques are used to handle large datasets efficiently.

4.Use of Pretrained Models for Feature Extraction

- Instead of training from scratch, the project leverages InceptionV3, which is pretrained on ImageNet, significantly reducing training time while improving accuracy.
- Transfer learning helps extract meaningful leaf characteristics like color, vein patterns, and texture, making classification more effective.
- The model is fine-tuned with medicinal plant datasets to ensure it accurately distinguishes between multiple plant species.

3.SYSTEM REQUIREMENTS SPECIFICATION

SYSTEM REQUIREMENTS SPECIFICATION

3.1 Software Requirements

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected, or unexpected from client's point of view.

- Operating System: Windows 8 and above
- Dataset: Kaggle/Zip File
- Programming Language: Python
- Online Working Tool: Google Colaboratory

Python

- Python is the primary programming language used for developing the medicinal plant classification model.
- It provides powerful libraries like TensorFlow, Keras, OpenCV, NumPy, and Pandas for deep learning and image processing.
- Python allows easy integration with cloud platforms, databases, and APIs for deployment and scalability.

Google Colab

- Google Colab is a cloud-based Jupyter notebook that provides a free GPU/TPU environment for deep learning.
- It supports TensorFlow and Keras, enabling efficient model training without requiring high-end local hardware.
- Auto-save and collaboration features make it easier to work in teams and share research.
- It allows direct integration with Google Drive for data storage and loading datasets conveniently.

Deep Learning Frameworks

- TensorFlow & Keras are used for building and training the deep learning model.
- Pretrained models like InceptionV3 help in feature extraction and transfer learning,

improving classification accuracy.

- Image processing techniques like convolutional neural networks (CNNs), pooling layers, and activation functions help extract key features.
- Deep learning optimizers (Adam, SGD) and loss functions (categorical cross-entropy) ensure efficient model convergence.

3.2 Hardware Requirements

These hardware recommendations are based on traditional deployments and may grow or shrink depending on how active users are

- Processor: 2 X Intel Xeon Gold 5118 (2.3GHz, 12C, 10.4GT/s UPI, 16MB Cache)
- RAM: 128GB DDR4 RAM
- Storage: 1TB NVMe Class 40 SSD + 4TB SATA 7200rpm HDD
- Network Connectivity: Stable Internet
- Monitor: 24"Dell Ultrasharp

Processor:

- Dual Intel Xeon Gold 5118 processors ensure high computational power for deep learning tasks.
- With 12 cores per processor and 10.4GT/s UPI speed, it enables efficient multi-threaded execution.
- The 16MB cache enhances data retrieval speed, improving overall system performance.

Memory:

- High-speed DDR4 RAM ensures smooth handling of large datasets and deep learning models.
- Allows efficient parallel computing, essential for processing high-resolution medicinal plant images. Prevents memory bottlenecks, ensuring stable system performance

Storage:

- 1TB NVMe SSD provides ultra-fast read/write speeds, enhancing dataset loading and model execution.
- 4TB SATA HDD offers ample storage for datasets, model checkpoints, and research data.
- This combination balances speed (SSD) and capacity (HDD) for optimal data management.

Monitor: 24" Dell Ultrasharp

- High-resolution 24-inch display provides accurate visualization of medicinal plant images.
- Wide viewing angles and color accuracy help in assessing image quality and model outputs.
- Ideal for researchers analyzing leaf features and deep learning model performance.

3.3 Functional Requirements

Requirements, which are related to functional aspect of software fall into this category. They define functions and functionality within and from the software system. Here predicting personality of user and recommending suitable careers falls into functional requirements. The proposed project is divided into 6 modules, they are listed below.

1. Data Collection
2. Data Pre-processing
3. Feature Extraction
4. Choosing Right Model
5. Training the Model
6. Testing The Model
7. Validation and Deployment

1.Data Collection:

Data collection is the process of gathering and measuring information from various sources to train the model effectively.

- Collect a dataset of medicinal leaf images from sources like Kaggle and public repositories.
- Ensure diversity in the dataset, including different lighting conditions, angles, and leaf types.
- Augment the dataset with artificially generated variations to improve model robustness.

2.Data Pre-processing:

The dataset is collected from multiple sources, ensuring diverse medicinal plants.

- Images are preprocessed by resizing them to match the model's input size (299x299 pixels for InceptionV3).
- Normalization is applied by scaling pixel values between 0 and 1 to standardize input data.
- Data augmentation techniques such as flipping, rotation, and zooming help improve model generalization.
- Sample images from different classes are displayed to verify variability and consistency.

3.Feature Extraction

- Convolutional layers in InceptionV3 extract key leaf features like texture, vein patterns, and shape.
- Edge detection and color analysis techniques help distinguish plant species more effectively.
- Global Average Pooling (GAP) is used to reduce dimensionality while retaining important patterns.
- Transfer learning ensures pretrained feature maps from ImageNet are used for efficient classification.

4.Choosing the Right Model

- Multiple deep learning architectures are evaluated, including InceptionV3, ResNet50, DenseNet121, and MobileNet.
- InceptionV3 is chosen based on accuracy, efficiency, and feature extraction capabilities.
- The model is fine-tuned by adjusting hyperparameters, such as learning rate, batch size, and activation functions.

5.Train the Model

- The dataset is split into training (80%) and validation (20%) for effective learning.
- The model is trained using categorical cross-entropy loss and the Adam optimizer for efficient convergence.
- Early stopping is implemented to prevent overfitting and improve generalization.
- Accuracy and loss curves are monitored to assess the model's learning progress.

6.Test the Model

- The trained model is evaluated on an independent test dataset to check real-world performance.
- Performance metrics such as accuracy, precision, recall, and F1-score are calculated.
- A confusion matrix is generated to analyze misclassifications and identify areas for improvement.
- The model's ability to generalize across different lighting conditions and backgrounds is validated.

7.Validate and Deploy the Model

- The model is fine-tuned using cross-validation to ensure consistency across different datasets.
- A simple and user-friendly web interface is developed for easy plant identification.
- TensorFlow Lite is used for optimizing the model for mobile and real-time applications.
- The final model is deployed on Google Cloud, AWS, or a local server, enabling real-time medicinal plant classification.

3.4 Non-Functional Requirements

- **Scalability:** The system was designed to handle a growing dataset of medicinal leaf images efficiently. As more species and samples were added, the model successfully processed large-scale data without significant performance degradation.
- **Performance:** The model was optimized for fast and accurate predictions. After evaluating different architecture models, Inception V3 was chosen for its superior balance of accuracy and efficiency, ensuring minimal latency in classification tasks.
- **Interpretability:** The classification results were made explainable by integrating visualization graphs, allowing users to understand which parts of the leaf image contributed to the model's decision. This enhanced transparency and trust in the system.
- **Accuracy:** The deep learning model achieved high classification accuracy by training on a well-structured dataset. Data augmentation, hyperparameter tuning, and rigorous evaluation using accuracy, precision, recall, and F1-score ensured reliability in real-world applications.
- **Usability:** A Streamlit-based user interface was developed, providing a seamless experience for users to upload leaf images and receive instant classification results. The interface was designed with a simple, intuitive layout, ensuring ease of use for both technical and non-technical users.

4. SYSTEM DESIGN

SYSTEM DESIGN

4.1 Introduction

This section describes the System Design of our medicinal leaf recognition project, focusing on the methodology and architecture used to build and optimize the model. The goal of the system is to automate medicinal plant identification using deep learning, enabling accurate and efficient classification of leaf images.

To ensure the best possible performance, we experimented with four different convolutional neural network (CNN) architectures: Inception V3, VGG16, ResNet50, and DenseNet121. Each model was trained and tested on a curated dataset of medicinal plant images, with their performance evaluated based on accuracy, precision, recall, and inference time. After extensive testing, Inception V3 emerged as the best-performing model, achieving the highest accuracy and efficient feature extraction.

The system follows a structured workflow, including data collection, preprocessing, feature extraction, model training, and classification. Additionally, a user-friendly interface was implemented to allow users to upload leaf images and receive instant classification results. The next section presents the architectural design of the system, illustrating the key components and their interactions.

The system's performance evaluation is conducted using metrics like accuracy, precision, recall, and F1-score, along with confusion matrices to identify misclassifications. Experimental results show that InceptionV3 outperforms VGG16 and DenseNet, proving to be robust against variations in lighting, background, and viewing angles. The software implementation consists of multiple modules, including data collection, preprocessing, model training, evaluation, and a user interface for image upload and real-time classification.

Architectural Diagram of proposed System

The Architecture diagram below is a graphical representation of a set of concepts, that is a part of the architecture, including process, methods and flow of the project

The software design of our project follows a structured deep learning pipeline for medicinal plant classification. It begins with data collection, where images of medicinal plants are gathered from various sources, followed by data preprocessing, which involves resizing, normalization, and noise reduction to enhance image quality.

To improve model generalization, data augmentation techniques such as rotation, flipping, and contrast adjustments are applied before storing the processed data for model training.

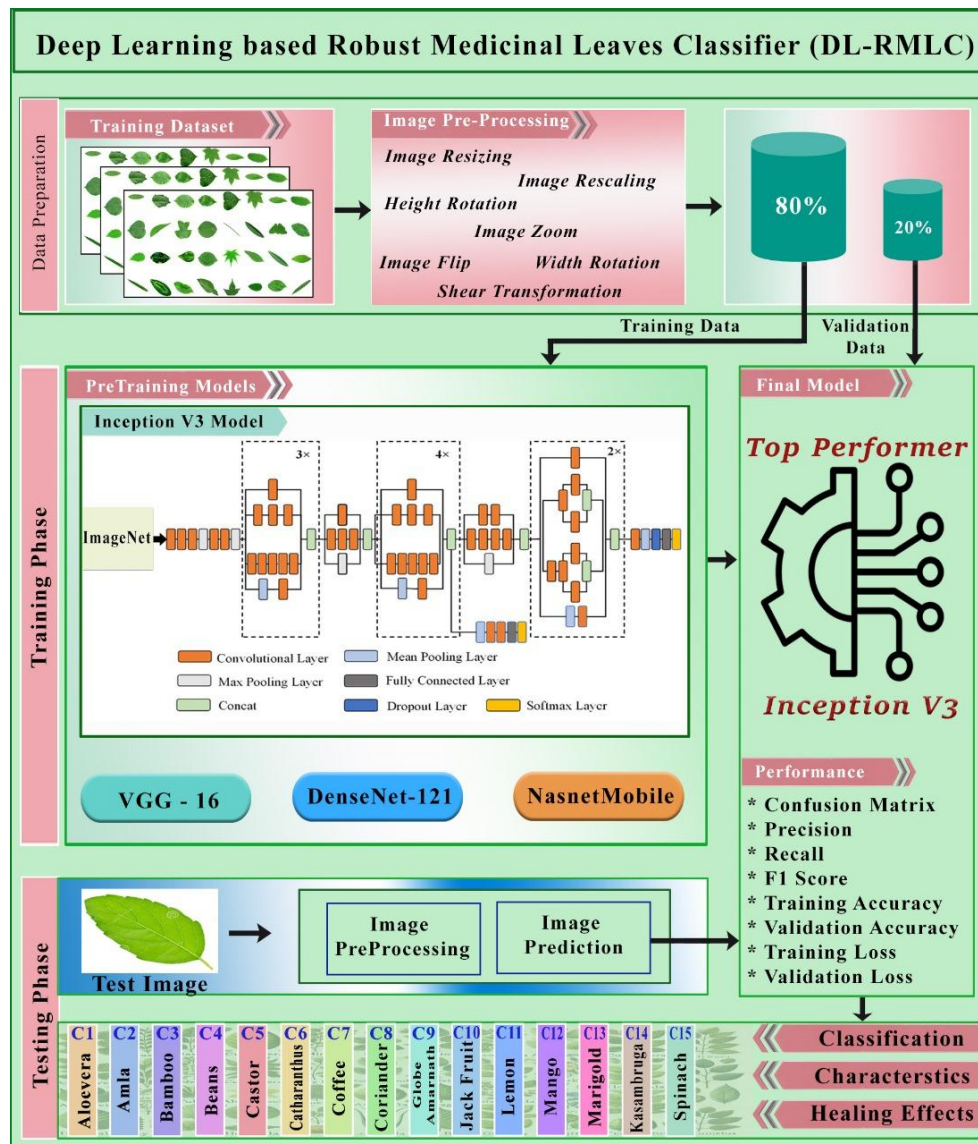


Fig.2 : Architecture diagram of proposed system

The above Fig.2 Architecture diagram shows a detailed explanation of the system design for the Deep Learning-Based Robust Medicinal Leaves Classifier (DL-RMLC). The system follows a structured approach comprising multiple phases: data processing, model training, and testing. The Inception V3 model has been identified as the top-performing deep learning model for classification, surpassing other models like VGG-16, DenseNet-121, and NasNetMobile in accuracy.

The System mainly consists of Three major phases:

STEP 1: Data Preparation and Pre-Processing

The first step in the system design is to prepare the dataset for training and testing the deep learning model. This includes data collection, preprocessing, and augmentation to ensure high-quality input data for model training.

1.Dataset Collection:

- A diverse dataset of medicinal leaf images is gathered from public datasets, such as Kaggle. It contains two main directories, medicinal leaf directory and medicinal plant directory.
- Since the project mainly focuses on the leaf directories, we need to remove plant directory or leave without considering it.
- The medicinal leaf directory dataset contains initially 80 different leaf folders containing more than 10,000 images in all the medicinal leaf folders.
- To optimize our model classification performance and high confidence we considered the top_15 species that achieved the highest classification accuracy using the Inceptionv3 model during initial testing.
- This approach helped us to refine the dataset by focusing on the well distinguished categories ensuring high reliable predictions.

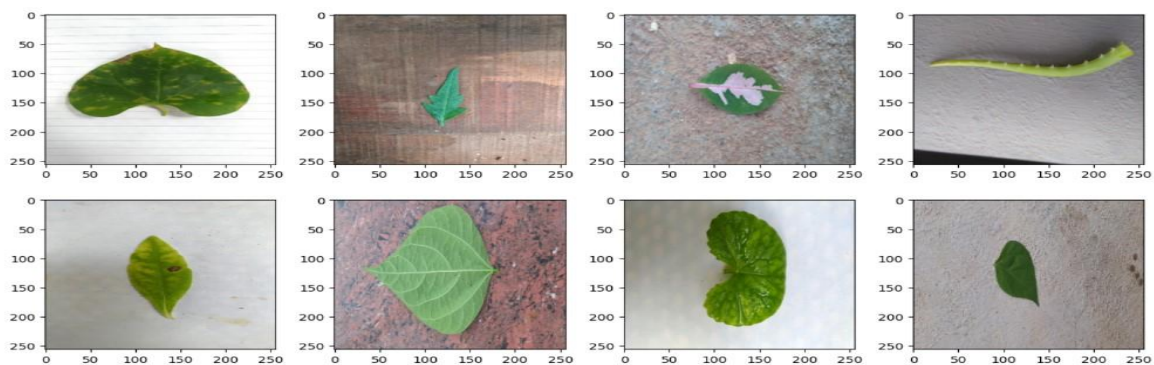


Fig.3: Medicinal leaf sample images

Medicinal Leaves Classification
















Local Name	Scientific Name	Number Of Images	Sample Images
Aloevera	Aloe vera	118	
Amla	Phyllanthus emblica	67	
Bamboo	Bambusa vulgaris	118	
Beans	Phaseolus vulgaris	97	
Castor	<u>Ricinus communis</u>	129	
Catharanthus	Catharanthus roseus	134	
Coffee	Coffea arabica	83	
Coriander	<u>Coriandrum sativum</u>	115	
Globe amaranth	Gomphrena globosa	81	
Jack Fruit	Artocarpus heterophyllus	110	
Lemon	Citrus x limon	123	
Mango	Mangifera indica	103	
Mangold	Tagetes	93	
Kasambruga	Kasambruga	48	
Spinach	Spinacia oleracea	149	

Fig.4:Top 15 Species

2. Image Pre-Processing:

The Preprocessing techniques are applied to enhance image quality and make the model robust to variations in lighting, orientation, and background. The following some steps followed:

1. Image Resizing: Standardizes all images to a fixed size (e.g., 224×224 pixels) for uniformity.
2. Image Rescaling: Normalizes pixel values to a range of 0 to 1 to speed up training and improve convergence.
3. Height & Width Rotation: Rotates images at different angles to help the model generalize across different leaf orientations.
4. Image Flipping: Flips images horizontally and vertically to increase dataset diversity.
5. Shear Transformation: Applies geometric transformations to simulate different perspectives of the leaf.
6. Image Zooming: Slightly zooms in and out on images to improve the model's robustness.

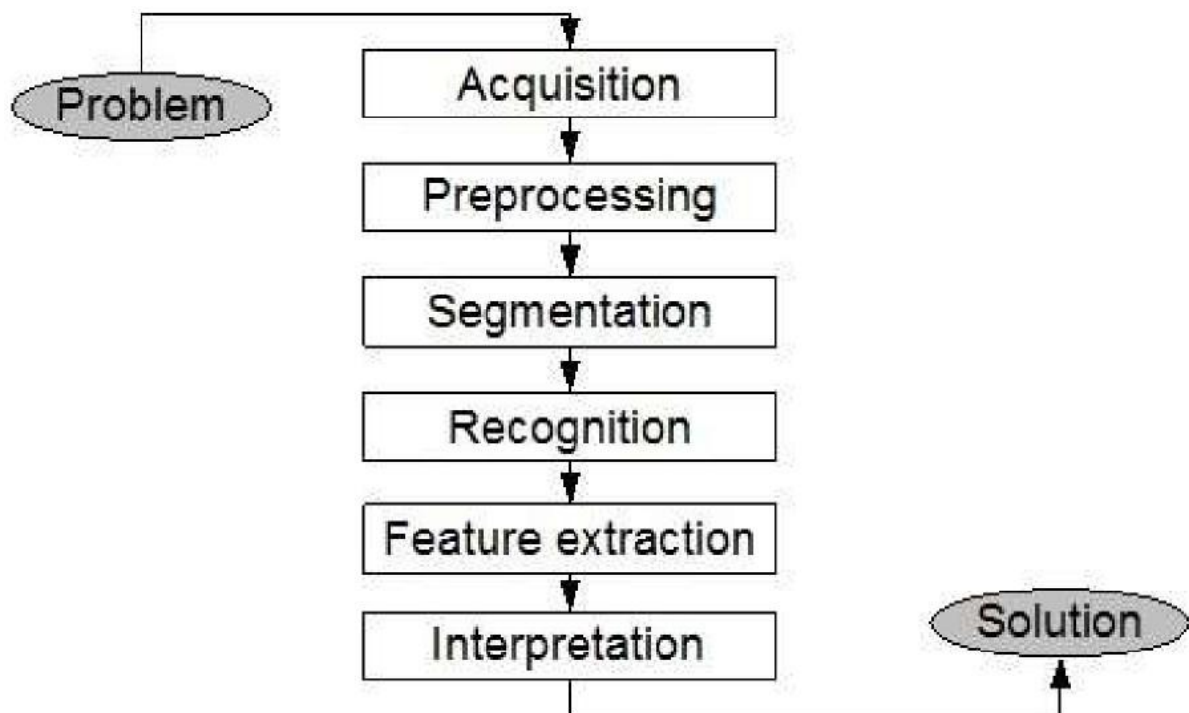


Fig.5: Steps in Image Pre-Processing

3.Data splitting:

- 80% of the dataset is used for training the deep learning model.
- 20% of the dataset is reserved for validation to evaluate the model's performance.

STEP-2 : Model Training and Validation:

After preprocessing, the image enters the feature extraction and classification phase, where the trained InceptionV3 model processes it through multiple convolutional layers to extract deep hierarchical features. The model leverages its Inception modules, which analyze different spatial scales within the image simultaneously, allowing it to detect intricate leaf structures such as venation patterns, texture variations, and edge contours.

Based on these extracted features, the model compares the input image with the patterns it has learned during training and predicts the most likely class of the leaf. This classification is performed through a fully connected neural network layer that assigns probability scores to each possible medicinal plant class, with the highest-scoring category being selected as the output.

Before training the model, different deep learning architecture models are considered. We trained our system with four different models to find out the top performing model. Same steps and process are followed for all the models to find best suitable model for our project. By examining the previous existing papers, we considered top 4 models that produced good results and inception v3 achieved highest accuracy than remaining models.

Below are the top-4 models considered:

- VGG16
- DenseNet121
- NasNetMobile
- Inception V3

❖ VGG16:

VGG16 is a deep convolutional neural network (CNN) that follows a simple, sequential architecture with 16 weight layers. It consists of multiple 3×3 convolutional layers, each followed by ReLU activation and max pooling layers. The architecture follows a strictly uniform pattern, stacking convolutional layers in increasing depth before down sampling through pooling. The fully connected layers at the end are responsible for classification.

Strengths:

- Straightforward architecture, making it easy to implement and modify.
- Performs well on image classification tasks due to deep feature extraction.
- Pre-trained versions are widely available, making transfer learning efficient.

Weaknesses:

- Computationally expensive due to a large number of parameters (~138 million).
- High memory consumption, making it inefficient for mobile and embedded applications.
- Susceptible to overfitting due to dense fully connected layers.

❖ **NasNetMobile:**

NASNetMobile is a Neural Architecture Search (NAS)-designed lightweight deep learning model optimized for mobile devices. Instead of manually designing layers, NASNet uses reinforcement learning to search for the best architecture, leading to highly efficient models. The network is composed of normal and reduction cells, where normal cells maintain feature dimensions while reduction cells down sample feature maps.

Strengths:

- Optimized for mobile and embedded systems, providing high accuracy with low latency.
- Fewer parameters (~4 million), making it lightweight compared to VGG16 and InceptionV3.
- Adaptable and scalable, allowing users to balance accuracy and efficiency.

Weaknesses:

- Requires specialized frameworks for implementation (e.g., TensorFlow NASNet models).
- Less generalizable compared to manually designed architectures like InceptionV3.
- Training from scratch is computationally expensive due to the architecture search process.

❖ **DenseNet121:**

DenseNet (Densely Connected Convolutional Network) is designed to maximize information flow between layers. Unlike traditional CNNs, each layer connects to every other layer in a feed-forward manner. This eliminates redundancy and allows feature reuse across the network. The architecture consists of dense blocks followed by transition layers.

Strengths:

- Feature reuse reduces the number of parameters while improving learning efficiency.
- Gradient flow is improved due to short connections, addressing the vanishing gradient problem.
- Requires fewer parameters than traditional deep networks, making it memory efficient.

Weaknesses:

- Computationally expensive due to numerous connections, leading to longer training times.
- Dense connectivity can lead to memory bottlenecks in large-scale models.
- More complex to implement compared to simpler architectures like VGG16.

Final Thoughts

Among these models, InceptionV3 stands out for its balance between accuracy and computational efficiency, making it a powerful choice for high-performance applications. VGG16, while simple and effective, is outdated in terms of efficiency. NASNetMobile is excellent for mobile devices but lacks flexibility in general-purpose applications. DenseNet, with its innovative connectivity, is great for optimizing feature learning but can be memory-intensive. Depending on the use case, each model offers unique advantages, with InceptionV3 excelling in large-scale image recognition tasks.

Inception V3 (Final Model - Top Performer)

Architecture:

- Uses Inception modules to extract features at multiple scales.
- Employs 1×1 convolutions for dimensionality reduction.
- Includes auxiliary classifiers to combat vanishing gradients.

Strengths:

- High accuracy in complex image classification tasks.
- More efficient than VGG-16 with fewer parameters.

The Inception V3 model is trained on the preprocessed dataset using Adam optimizer with a learning rate of 0.001.

Loss Function: Categorical Cross-Entropy is used since it is a multi-class classification problem.

▪ Performance Metrics Evaluated:

- Accuracy
- Precision

→ Recall

→ F1-score

→ Confusion Matrix Analysis

<u>MODEL</u>	<u>ACCURACY</u>	<u>Image Size</u>
InceptionV3	96%	256 x 256
Vgg16	67%	224 x 224
DenseNet121	93%	256 x 256
NasnetMobile	94%	224 x 224

Fig:6: Comparison table

STEP-3 :Testing Phase:

Once the InceptionV3 model is trained, it is deployed to classify new medicinal leaf images in real-time, enabling quick and accurate identification of plant species. The first step in this process is input image processing, where a user uploads an image of a medicinal leaf into the system. The uploaded image undergoes the same preprocessing steps that were applied to the training data, ensuring consistency in feature extraction.

This includes resizing the image to match the input dimensions required by the model (299×299 pixels for InceptionV3), normalizing pixel values to enhance contrast, and applying any necessary augmentations such as rotation or noise reduction. These steps help in standardizing input images, reducing variations due to lighting conditions, camera quality, or background distractions, ultimately improving classification accuracy.

Finally, the output and results are displayed to the user, providing detailed information about the identified medicinal leaf. The system presents the predicted leaf name, along with its known medicinal properties and healing effects, making it valuable for applications in herbal medicine, pharmacology, and traditional healing practices. Additionally, a confidence score is provided, indicating how certain the model is about its prediction. Higher confidence scores suggest greater reliability, whereas lower scores may prompt the user to verify the result or upload a clearer image. This real-time classification system enables researchers, herbalists, and healthcare professionals to efficiently identify medicinal plants, facilitating informed decisions about their potential therapeutic applications.

4.2 UML Diagrams

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

There are mainly two categories:

- Structure Diagrams
- Behavioral Diagram

Relationships in UML:

- Dependency
- Association
- Generalization
- Realization

1. Use Case Diagram of Proposed System:

A use case diagram in the Unified Modelling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical over view of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

The main purpose of designed use case diagram is to showcase all functions that performed by each actor. Roles of the actors in the system can be depicted as shown in figure 6. These diagrams are particularly useful in the design phase of software development, as they provide a clear and structured view of user interactions, making it easier to identify system requirements and functionalities.

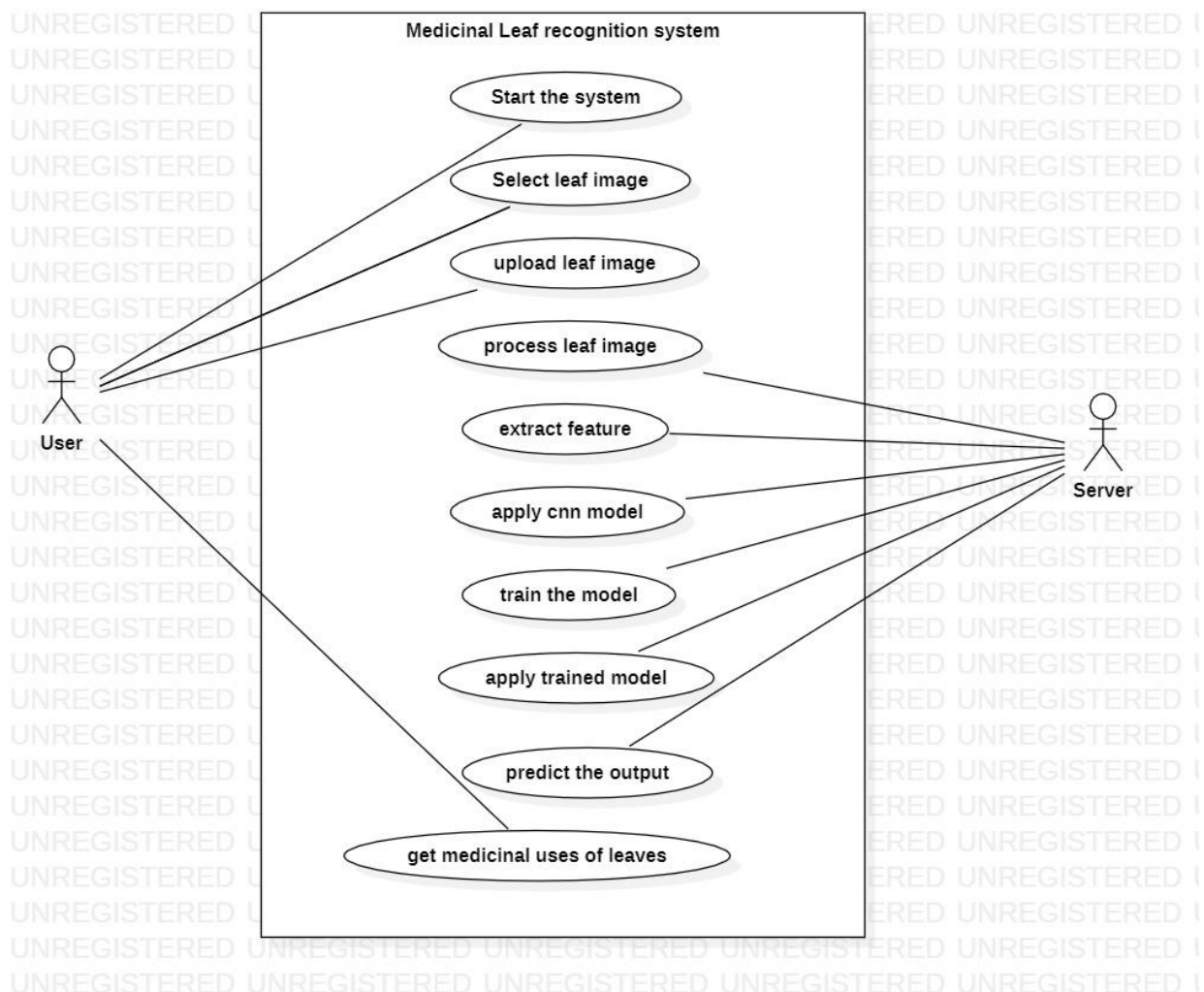


Fig.7: Use case Diagram of proposed System

2.Sequence Diagram

A sequence diagram in Unified Modelling Language (UML) diagram that illustrates the interaction between objects in a system over time. It represents how messages are exchanged between different components or actors in a sequential order. The diagram consists of lifelines, which represent objects or entities, and messages, which show the flow of communication between these entities. Sequence diagrams help in understanding system behavior by modeling the order of interactions and the logic behind different processes.

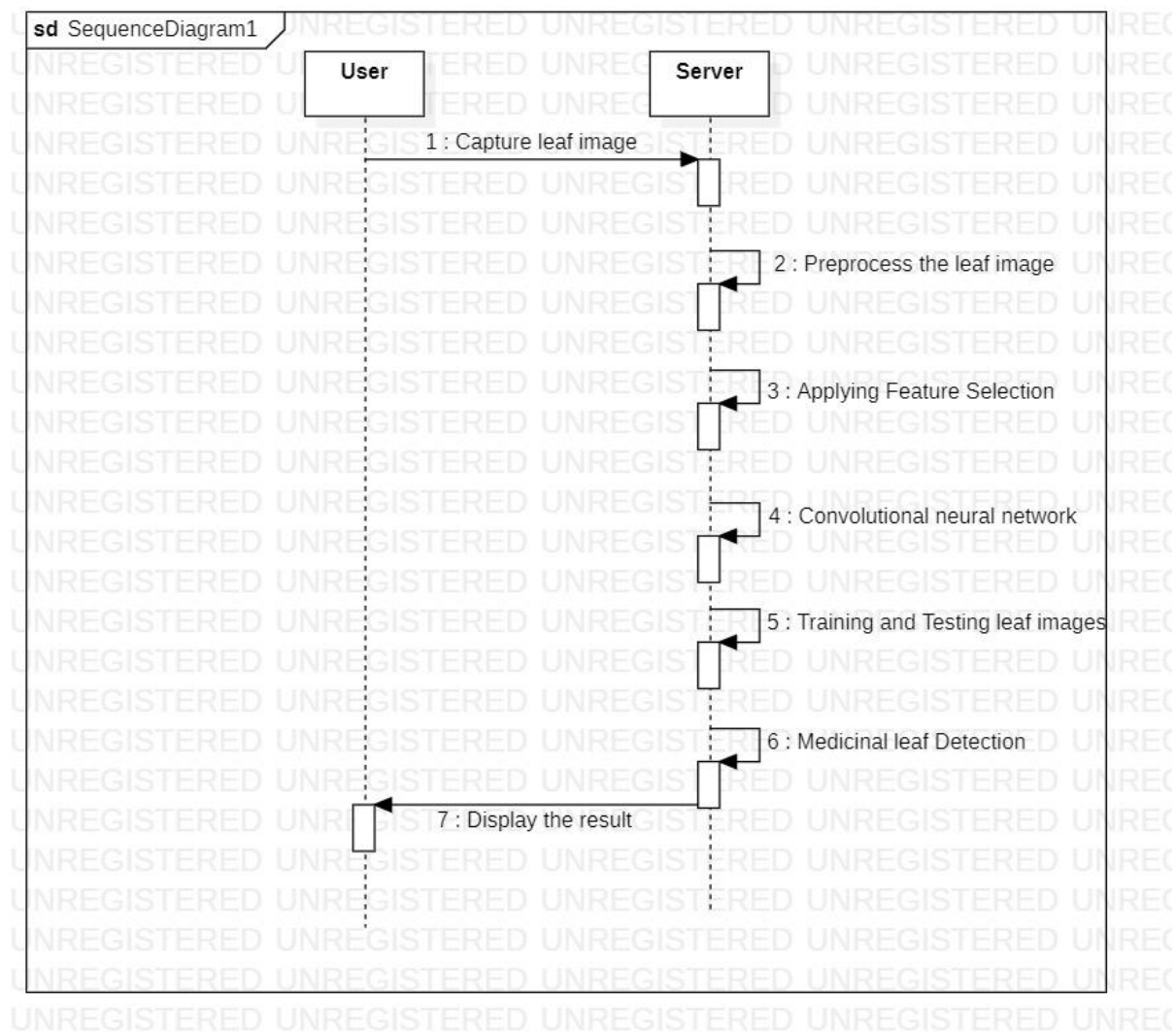


Fig.8: Sequence diagram of the proposed system

3.Activity Diagram:

An Activity Diagram is a type of UML (Unified Modeling Language) diagram that represents the flow of activities within a system. It visually depicts the sequence of actions, decisions, and parallel processes, making it useful for modeling workflows, business processes, and system functionalities. Activity diagrams use various symbols such as rectangles for actions, diamonds for decisions, and arrows for flow representation.

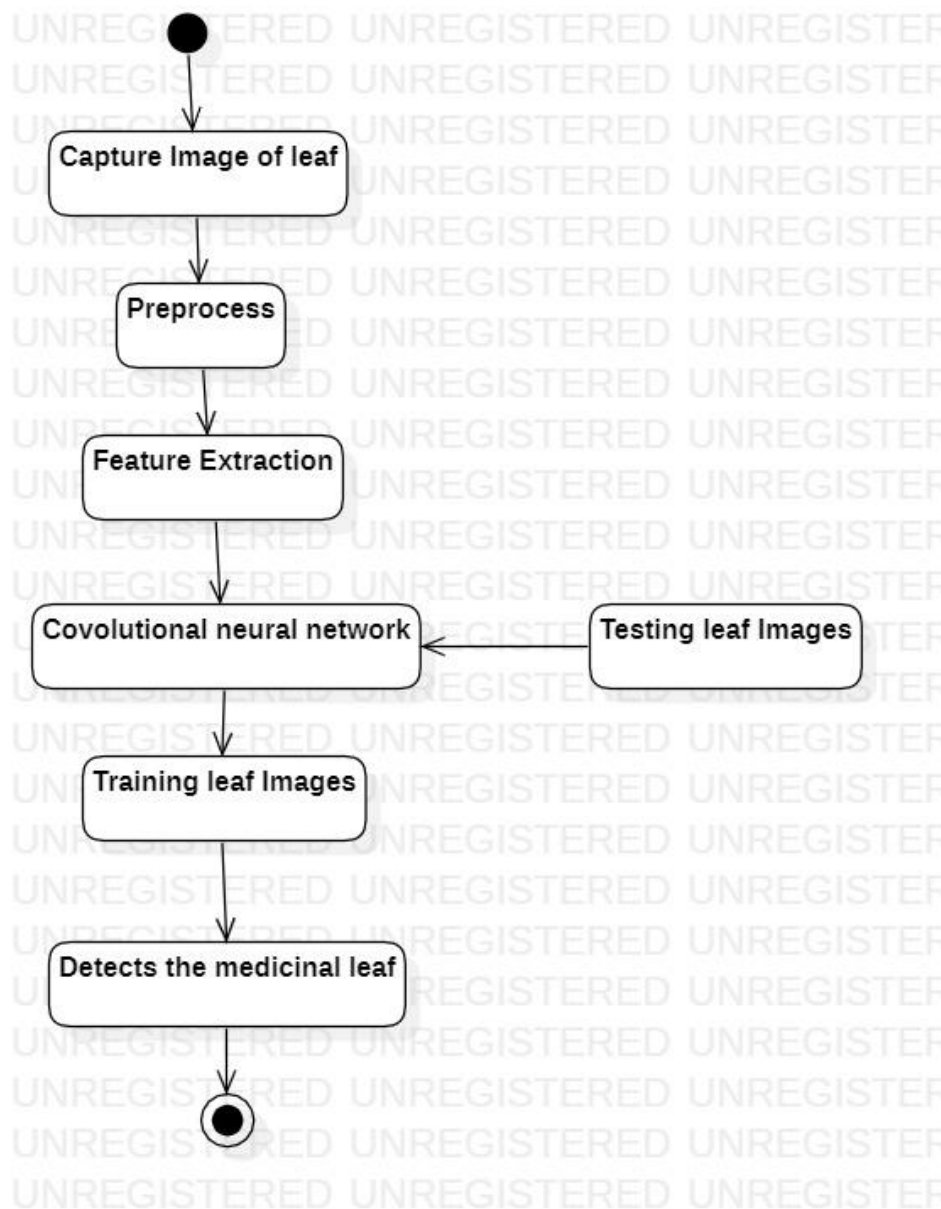


Fig.9: Activity diagram of proposed system

4. Class Diagram:

A class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information. The class diagram for proposed system is as shown in figure 9.

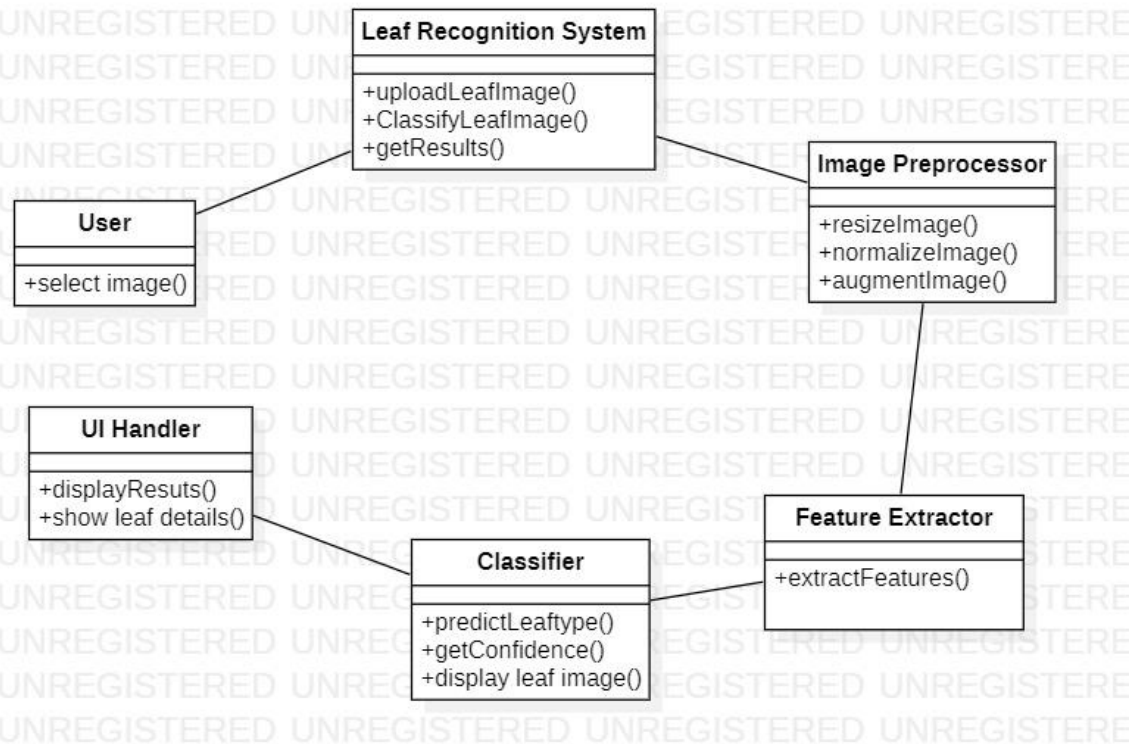


Fig.10: Class Diagram of Proposed System

5.SYSTEM IMPLEMENTATION

SYSTEM IMPLEMENTATION

5.1 Introduction:

The implementation of the Medicinal Leaf Recognition System using Deep Learning involves the integration of machine learning models, image processing techniques, and a user-friendly interface to classify medicinal leaves. This system leverages the Inception V3 model for feature extraction and classification, ensuring high accuracy in recognizing different leaf types.

The implementation is structured into multiple stages, including image preprocessing, feature extraction, classification, and result presentation. The system is designed as a standalone application, with leaf details hardcoded instead of using an external database. The front-end interface allows users to upload images and receive classification results, while the back-end processes the image and performs the recognition task.

This section details the step-by-step execution of each module, highlighting how the machine learning model, image processing techniques, and system architecture work together to achieve efficient medicinal leaf identification.

Integration of Core Components:

The integration of multiple technological components is a key step in the implementation of the medicinal plant leaf classification system. This includes the combination of deep learning models such as DenseNet121, VGG16, NasnetMobile, and InceptionV3 with advanced image processing techniques for efficient feature extraction and classification. Additionally, data augmentation techniques are incorporated to enhance the diversity of training data, improving the model's generalization capability. Ensuring seamless interaction between these components is crucial for achieving high accuracy and reliable performance.

5.2 Project Modules:

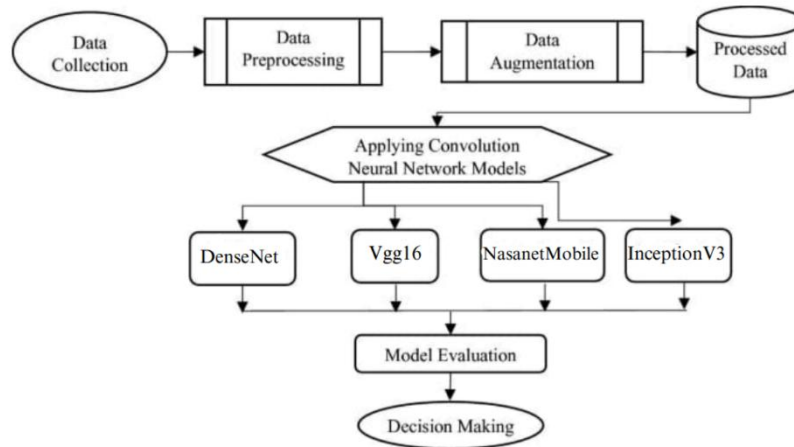


Fig.11: Project flowchart with different modules

1.Data Collection Module:

The data collection module forms the foundation of the medicinal plant leaf identification, dataset of medicinal plant leaves from various sources, including botanical gardens, research institutions, online plant databases, and real-world field captures using cameras or mobile devices. The dataset must include images taken under different lighting conditions, angles, and backgrounds to ensure the model generalizes well to real-world scenarios. Additionally, it is important to collect leaves in different stages of growth and under varying environmental conditions to improve classification robustness. Each image is labeled with its corresponding plant species to facilitate supervised learning. Proper organization of the dataset into training, validation, and testing sets is also a crucial part of this module to ensure effective model training and evaluation.

Initially the original dataset of our project is collected from the Kaggle source, Indian medicinal leaf dataset, which again contains two sub folders medicinal leaf dataset and medicinal plant dataset. Since the focus of our project is mainly on the leaf dataset, it is considered to remove or ignore the medicinal plant dataset. This medicinal leaf dataset contains 80 different leaves. Furthermore, we again converted the 80 leaf dataset into a small dataset containing 15 leaves based on the highest accuracy, according to the selected model of our project.

The quality and diversity of the collected data directly impact the accuracy and robustness of the classification model. Therefore, efforts are made to include high-resolution images from different geographic locations and varying growth stages of medicinal plants. Field-based collection using mobile applications allows botanists and researchers to capture real-time images with precise location tagging. Crowdsourcing initiatives encourage

contributions from local communities and herbal medicine practitioners, increasing the dataset's richness. Additionally, integrating spectral imaging data, such as infrared or ultraviolet scans, helps capture details invisible to the human eye, further enhancing classification accuracy. The use of drones for large-scale plant surveys in dense forests or agricultural lands is also explored to collect broader datasets.

2.Data Preprocessing Module:

Data pre-processing is a crucial data mining technique aimed at converting raw data into a comprehensible format. Real-world data often presents challenge such as incompleteness, inconsistency, lack of certain patterns, and errors. Data pre-processing is a reliable method for addressing these issues, preparing the raw data for subsequent analysis.

Once the dataset is collected, it undergoes preprocessing to ensure uniformity, enhance image quality, and eliminate unwanted noise that may interfere with classification. This module performs several critical operations, including image resizing to standard dimensions, color normalization to maintain consistency across images, and noise reduction techniques such as Gaussian blur or median filtering to remove background artifacts. Additionally, grayscale conversion may be applied to reduce computational complexity, allowing the model to focus on texture and shape features rather than color.

In some cases, edge detection algorithms like Canny Edge Detection or contour extraction techniques may be employed to highlight leaf veins and margins, which are essential for species differentiation. The primary goal of this module is to standardize the input images so that the classification model can learn effectively without being affected by unnecessary variations.

In preprocessing images undergo:

- Image rescaling
- Image resizing
- Image flipping
- Image rotation
- Shear transformations

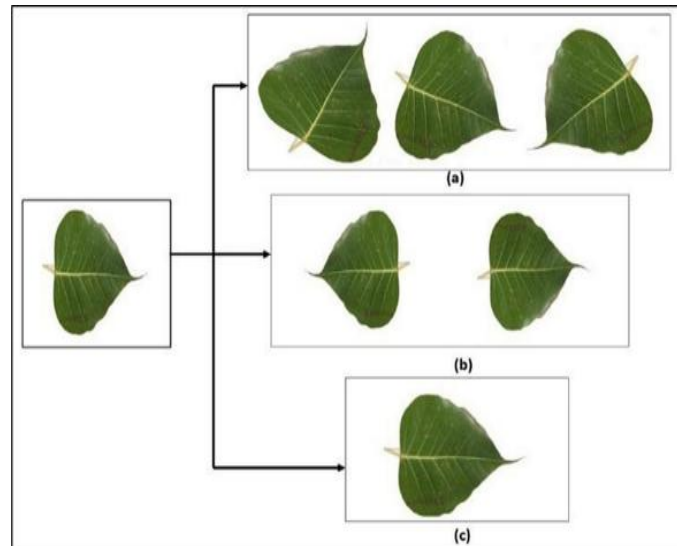


Fig.12: Rotation of images

3.Feature Extraction Module

The feature extraction module plays a crucial role in identifying distinguishing characteristics of medicinal plant leaves that help in accurate classification. This module processes each image to extract essential features such as shape, texture, venation patterns, and color histograms. Traditional feature extraction techniques like Histogram of Oriented Gradients (HOG), Local Binary Patterns, can be used to capture leaf details. In deep learning-based approaches, convolutional neural networks (CNNs) automatically learn hierarchical features from images, capturing both low-level details like edges and high-level patterns like leaf structures.

Feature extraction plays a crucial role in our medicinal leaf recognition system, as it helps in distinguishing different plant species based on their unique characteristics. Since we are using InceptionV3 as our deep learning model, feature extraction is primarily handled by its convolutional layers, which learn hierarchical representations of leaf structures.

Steps in Feature Extraction

1. Image Preprocessing

- The input leaf images are resized to a fixed dimension suitable for InceptionV3 (e.g., 299x299 pixels).
- Normalization is applied to scale pixel values between 0 and 1 to improve model efficiency.
- Data augmentation techniques (like rotation, flipping, and zooming) are used to enhance feature diversity.

2. Feature Extraction Using InceptionV3

- The pre-trained InceptionV3 model extracts high-dimensional features from the leaf images.
- We use intermediate layers (such as mixed layers) to capture detailed leaf patterns, including shape, texture, and vein structures.
- The final convolutional layer outputs a feature vector, which serves as the input for classification.

3. Dimensionality Reduction

- To remove redundant information and improve efficiency, techniques like Global Average Pooling (GAP) are used to reduce the feature map size.
- Principal Component Analysis (PCA) may be applied if needed for further dimensionality reduction.

4. Feature Representation

- Extracted features represent leaf characteristics such as:
 - Shape Features (captured through convolutional layers)
 - Texture Features (edge detection, vein structures)
 - Colour Features (leaf pigmentation captured via RGB channels)

5. Integration with Classification

- The extracted features are passed to the fully connected layers of the model, which maps them to the correct medicinal plant category using a SoftMax classifier.

Extracted features are stored in a structured format and passed to the classification model, where they serve as the basis for plant identification. The efficiency of this module directly impacts the system's ability to differentiate between similar-looking plant species.

4.Model Training Module

Once the features are extracted, the model training module takes over, using deep learning algorithms to learn patterns from the dataset. This module involves selecting appropriate models, such as **DenseNet, VGG16, NasNetMobile, and InceptionV3**, which are known for their high performance in image classification tasks. The training process consists of multiple iterations (epochs), where the model continuously updates its internal weights using backpropagation and an optimization algorithm like Adam.

During training, loss functions such as categorical cross-entropy measure the difference between the predicted and actual plant species, and the model gradually minimizes this loss. The dataset is split into training and validation sets to monitor performance, and techniques like dropout and batch normalization are applied to prevent overfitting. The success of this module determines how well the model generalizes to new leaf images.

To further refine the training process, an ensemble learning approach is explored, combining multiple deep learning models to achieve higher classification accuracy. Attention mechanisms, such as self-attention layers, help the model focus on the most critical leaf regions, improving species differentiation. The use of knowledge distillation enables lightweight models to learn from more complex architectures, allowing deployment on resource-constrained devices like mobile phones. Class imbalance handling techniques, such as focal loss and oversampling of underrepresented species, ensure that rare medicinal plants are accurately classified.

5. Model Evaluation Module

After training, the model must be evaluated to ensure it performs accurately on unseen data. The model evaluation module assesses performance using a separate test dataset, measuring key metrics such as accuracy, precision, recall, and F1-score. Confusion matrices and classification reports help analyze the model's strengths and weaknesses, identifying areas where misclassifications occur.

This module also involves comparing different deep learning architectures to determine which model offers the best trade-off between accuracy and computational efficiency. Additionally, techniques like cross-validation may be applied to further validate the model's performance across different subsets of data. If the model underperforms, hyperparameter tuning, data augmentation, or additional training may be required to improve classification accuracy.

6. Decision-Making and Classification Module

Once the best-performing model is selected, it is deployed in the decision-making and classification module, which is responsible for real-time plant identification. Users can upload an image of a medicinal plant leaf, and the system processes it, extracts relevant features, and classifies it into the most likely species. The classification output may include a confidence score indicating the reliability of the prediction, along with additional details about the identified plant, such as its medicinal properties, usage, and scientific classification. In cases where uncertainty is high, the system may provide a ranked list of possible species, allowing users to manually verify the result. This module plays a key role

in delivering meaningful insights to researchers, botanists, and healthcare professionals. To make classification results more actionable, confidence calibration techniques are applied to ensure probability scores reflect true reliability. In ambiguous cases, the system prompts for additional inputs, such as leaf vein images or multiple angles, to refine predictions. Post-classification feedback allows users to confirm or correct results, enabling continuous model improvement.

Integration with a medicinal plant database provides insights into traditional and scientific uses, bridging the gap between AI predictions and practical applications. Multi-label classification is explored for plants with hybrid characteristics, where a single leaf may belong to multiple known species or subspecies.

7. User Interface Module

To ensure accessibility and ease of use, the system is integrated into a user-friendly interface or an API. The user interface may be a web application, mobile app, or desktop software that allows users to upload images, view classification results, and explore additional plant-related information. The user interface (UI) of the medicinal plant leaf identification system is designed to be intuitive, responsive, and accessible to users from diverse backgrounds, including researchers, botanists, farmers, and herbal medicine practitioners.

The UI provides a seamless experience by allowing users to upload images of plant leaves. A clean and well-structured layout ensures that users can navigate through various features effortlessly, with clear icons and tooltips guiding them through the classification process. The UI displays classification results in an easy-to-understand format, presenting the identified plant species along with its availability regions, medicinal properties, and possible applications. Additionally, confidence scores for classification results are provided to help users gauge the reliability of the prediction.

8. Continuous Learning and Update Module

To maintain long-term accuracy and relevance, the system incorporates a continuous learning module that updates the classification model with new data over time. This module allows users to contribute new leaf images and report misclassifications, which are then reviewed and added to the training dataset. Periodic retraining ensures the model stays up to date with new plant species and environmental variations.

5.3 Algorithms

1.InceptionV3

Depth: Around 48 layers

Parameters: ~23 million

Input Size: Typically 299x299 pixels

Output: 1000 classes (can be modified for custom tasks).

Inception V3 is a powerful and efficient convolutional neural network (CNN) architecture developed by Google. It is an improved version of the original Inception (GoogLeNet) model and is widely used in image classification and computer vision tasks. The model was introduced as part of the "Rethinking the Inception Architecture for Computer Vision" paper. Inception V3 is designed to achieve high accuracy with optimized computational performance, making it suitable for both research and real-world applications.

Inception Modules:

Inception V3 uses advanced Inception modules that allow the network to capture features at multiple scales. Each module contains a combination of 1x1, 3x3, and 5x5 convolutions along with max-pooling, helping the model learn both fine and coarse features.

Factorization of Convolutions:

To reduce computational complexity, large convolutions (like 5x5 or 7x7) are factorized into smaller convolutions (e.g., 1x7 followed by 7x1). This makes the model faster and more efficient without compromising performance.

Applications in Our Project:

In our Medicinal Leaf Recognition system, we used Inception V3 as the core model due to its strong performance in extracting detailed and hierarchical features from complex leaf images. Its ability to learn multi-scale features allows better recognition of leaf shapes, patterns, and textures, which is crucial in identifying different medicinal plants accurately.

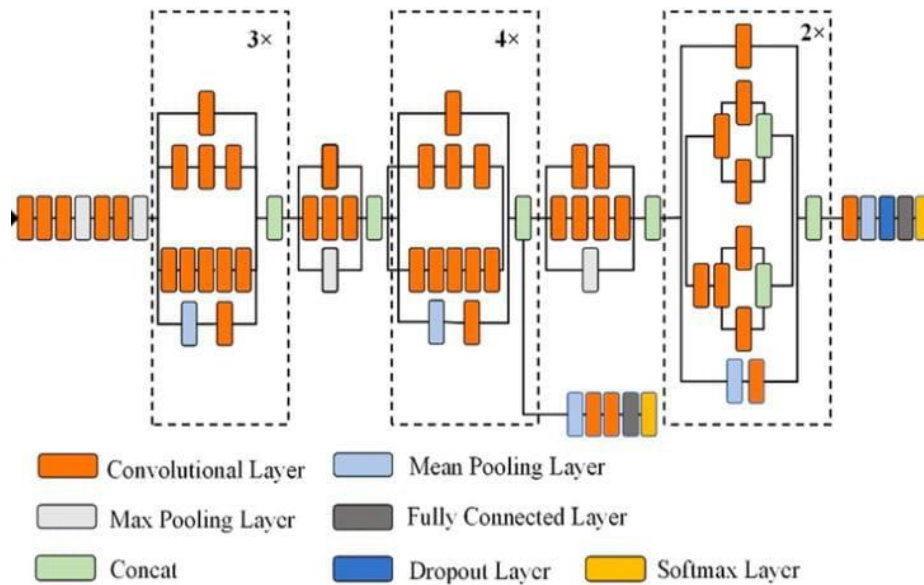


Fig.13:Workflow diagram of InceptionV3

2. VGG16

Depth:16 layers(13 convolutional + 3 fully connected)

Parameters: ~138 million

Input Size: Typically 224x224 pixels

Output: 1000 classes (can be modified for custom tasks).

VGG16 is a deep convolutional neural network architecture- developed by the Visual Geometry Group (VGG) at the University of Oxford. It gained popularity after performing remarkably well in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014.

The "16" in VGG16 refers to the number of weight layers (13 convolutional layers and 3 fully connected layers). The architecture is known for its simplicity and uniformity, using small 3x3 convolutional filters throughout the network, which makes it easy to understand and implement.

Key features of VGG16:

Deep Architecture with Simplicity: It uses a simple and consistent design of stacked 3x3 convolutional layers.

Max-Pooling Layers: Added after convolutional blocks to reduce spatial dimensions.

Fully Connected Layers: At the end of the network, it uses fully connected layers for classification.

ReLU Activation: Introduced after each convolution to add non-linearity.

VGG16 is widely used in various image classification and transfer learning tasks. Although it has more parameters and is computationally heavier than newer models, its strong performance and ease of use make it a popular baseline model in many computer vision projects.

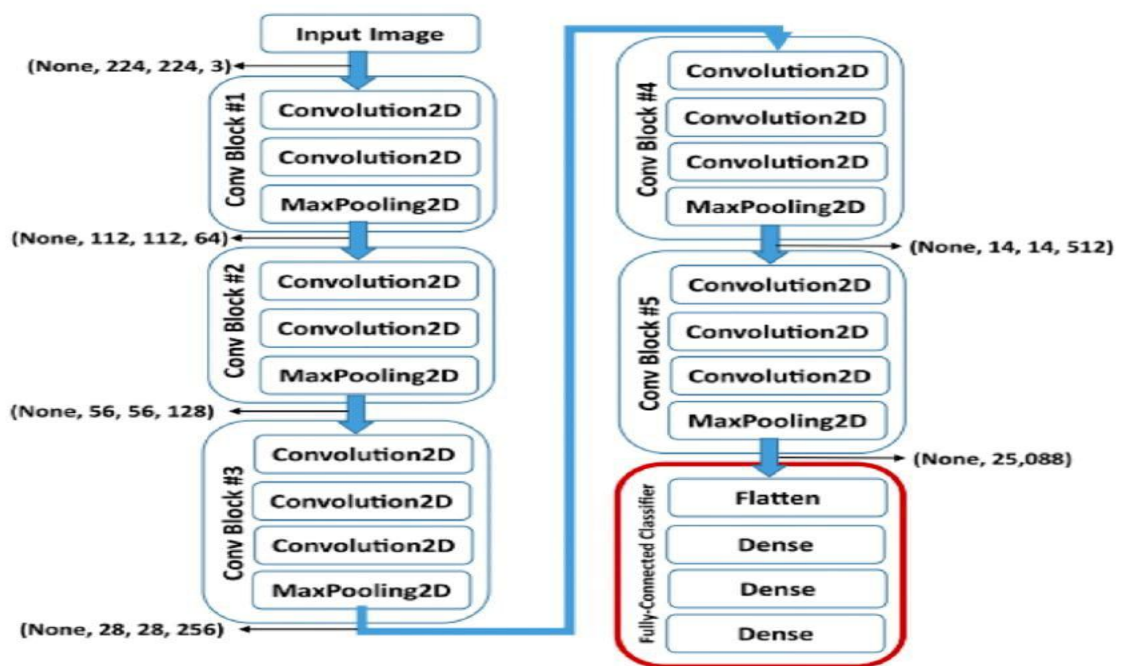


Fig.14:Workflow diagram of VGG16

3.NASNetMobile

Depth:~88 layers

Parameters: ~5.3 million

Input Size: Typically 224x224 pixels

Output: 1000 classes (can be modified for custom tasks).

NASNetMobile is a lightweight and efficient convolutional neural network architecture designed specifically for mobile and embedded devices. It is a part of the Neural Architecture Search Network (NASNet) family, developed by Google using AutoML, a technique where machine learning is used to design neural networks automatically.

The "NAS" in NASNet stands for Neural Architecture Search, where reinforcement learning is applied to find the best-performing convolutional cell structure. These optimized cells are then stacked to build a deep network, which ensures high accuracy with reduced computational cost.

NASNetMobile is a mobile-friendly version of NASNetLarge, optimized for speed and size without compromising much on accuracy. It is ideal for deployment on smartphones, tablets, and other devices with limited computing resources.

Key Features:

Efficient Architecture: Designed through automated search, balancing performance and resource usage.

Modular Design: Built using normal cells and reduction cells, making it scalable.

High Accuracy: Despite its smaller size, NASNetMobile provides competitive results on image classification tasks.

Mobile Optimization: Tailored for low-latency inference on mobile devices.

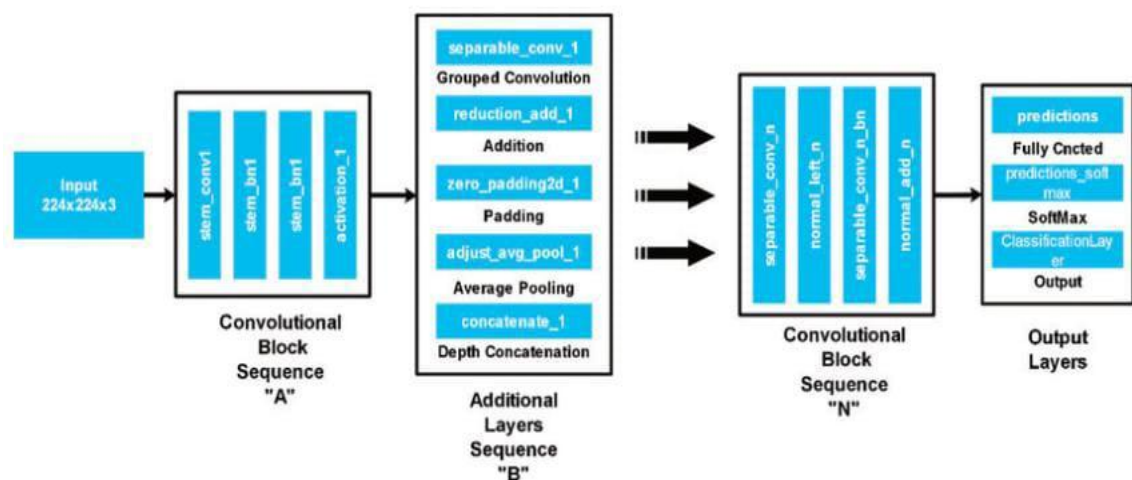


Fig.15:Workflow diagram of NASNetMobile

4.DenseNet121

Depth: ~121 layers

Parameters: ~8 million

Input Size: Typically 224x224 pixels

Output: 1000 classes (can be modified for custom tasks).

DenseNet121 is a popular convolutional neural network (CNN) architecture that is part of the DenseNet (Densely Connected Convolutional Networks) family. It was proposed by Gao Huang et al. in 2017. DenseNet121 is known for its efficiency and high accuracy in image classification and recognition tasks.

Unlike traditional CNNs, where each layer feeds only into the next one, DenseNet121 introduces direct connections between all layers. In this architecture, each layer receives inputs from all preceding layers and passes its own feature maps to all subsequent layers.

This dense connectivity helps in:

1. Improved Feature Propagation
2. Better Gradient Flow
3. Reduced Vanishing Gradient Problem
4. Fewer Parameters Compared to Traditional CNNs.

DenseNet121 consists of 121 layers and is composed of dense blocks, transition layers, and batch normalization. It is widely used in various computer vision tasks, including medical image analysis, object detection, and facial recognition. Overall, DenseNet121 offers a powerful balance between computational efficiency and high performance, making it a suitable choice for deep learning projects involving image classification or feature extraction.

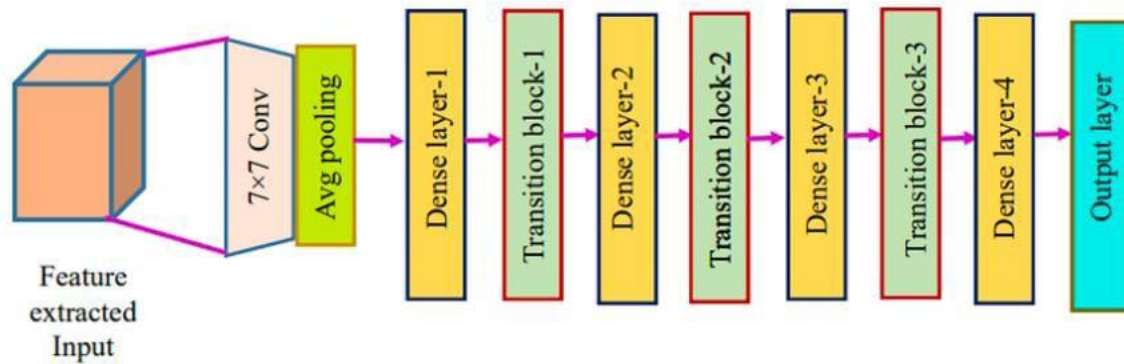


Fig.16:Workflow diagram of DenseNet121

5.2 Screens

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow_hub as hub
import tensorflow_datasets as tfds
from tensorflow.keras import layers

import numpy as np
import matplotlib.pyplot as plt
import tensorflow_hub as hub
import tensorflow_datasets as tfds
from tensorflow.keras import layers

from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf
from tensorflow import keras
import cv2
import os
```

Fig.17:Importing Required Libraries

```
!pip install kaggle

Requirement already satisfied: kaggle in /usr/local/lib/python3.11/dist-packages (1.6.17)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.11/dist-packages (from kaggle) (1.17.0)
Requirement already satisfied: certifi>=2023.7.22 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2025.1.31)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from kaggle) (4.67.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.11/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.3.0)
Requirement already satisfied: bleach in /usr/local/lib/python3.11/dist-packages (from kaggle) (6.2.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.11/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.11/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->kaggle) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->kaggle) (3.10)

[ ] #configuring the path of kaggle.json file
!mkdir -p ~/.kaggle

[ ] !cp kaggle.json ~/.kaggle/
cp: cannot stat 'kaggle.json': No such file or directory

[ ] !chmod 600 ~/.kaggle/kaggle.json
chmod: cannot access '/root/.kaggle/kaggle.json': No such file or directory

[ ] !kaggle datasets download -d aryashah2k/indian-medicinal-leaves-dataset
```

Fig.18: Downloading the dataset from Kaggle through API

```
[ ] from sklearn.metrics import classification_report
import numpy as np

# Generate predictions for the validation set
y_true = validation_generator.classes
y_pred = np.argmax(model.predict(validation_generator), axis=-1)

# Get class indices and names
class_indices = validation_generator.class_indices
class_names = list(class_indices.keys())

# Generate classification report
report = classification_report(y_true, y_pred, target_names=class_names, output_dict=True)

# Extract accuracy (precision) for each species
species_accuracies = {species: report[species]['precision'] for species in class_names}
sorted_species = sorted(species_accuracies.items(), key=lambda x: x[1], reverse=True)

# Select top 15 species
top_species = sorted_species[:15]
top_species_names = [species for species, _ in top_species]
```

/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its
self._warn_if_super_not_called()
42/42 272s 6s/step

Fig.19: Identifying the top_15 leaves from the dataset that has highest accuracy

```
[ ] import shutil
import os

# Path to the original dataset
original_dataset_dir = '/content/indian_medicinal_leaves/Indian Medicinal Leaves Image Datasets/Medicinal Leaf dataset'

filtered_dataset_dir = '/content/top_15_species'

if not os.path.exists(filtered_dataset_dir):
    os.makedirs(filtered_dataset_dir)

for species in top_species_names:
    src = os.path.join(original_dataset_dir, species)
    dst = os.path.join(filtered_dataset_dir, species)
    shutil.copytree(src, dst)

print(f"Filtered dataset with top 15 species is available at: {filtered_dataset_dir}")
```

Filtered dataset with top 15 species is available at: /content/top_15_species

```
[ ] import zipfile

zip_file_path = '/content/top_15_species-20250123T051752Z-001.zip'
extracted_dir = '/content/top_15_species'
```

Fig.20: Converting the top_15 leaves into a dataset


```
C:\Users> bharg > Downloads > Final_Project (1).ipynb > from google.colab import drive
+ Code + Markdown ... Select Kernel

def load_and_preprocess_images(dataset_path, image_size=(128, 128)):
    images = []
    labels = []

    # Iterate through class directories
    for class_dir in os.listdir(dataset_path):
        class_path = os.path.join(dataset_path, class_dir)
        class_label = int(class_dir) # Convert directory name to label if needed

    # Iterate through images in the class directory
    for image_file in os.listdir(class_path):
        image_path = os.path.join(class_path, image_file)

        # Read and preprocess the image
        image = cv2.imread(image_path)
        image = cv2.resize(image, image_size)
        image = image.astype(np.float32) / 255.0 # Normalize pixel values

    # Append the image and label to lists
    images.append(image)
    labels.append(class_label)
```

Fig.21: Loading and Preprocessing images

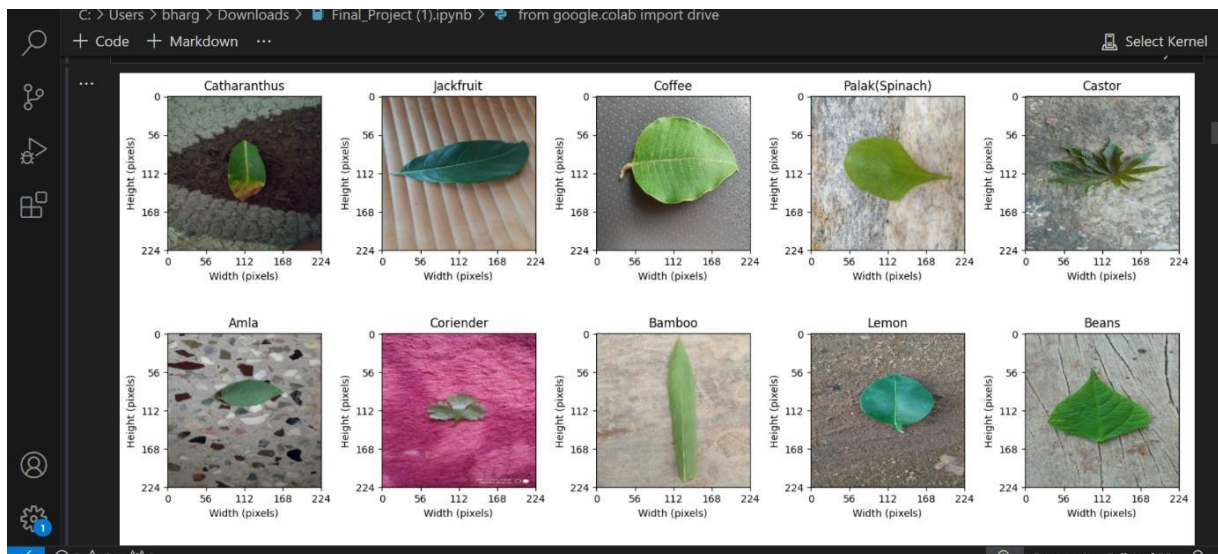
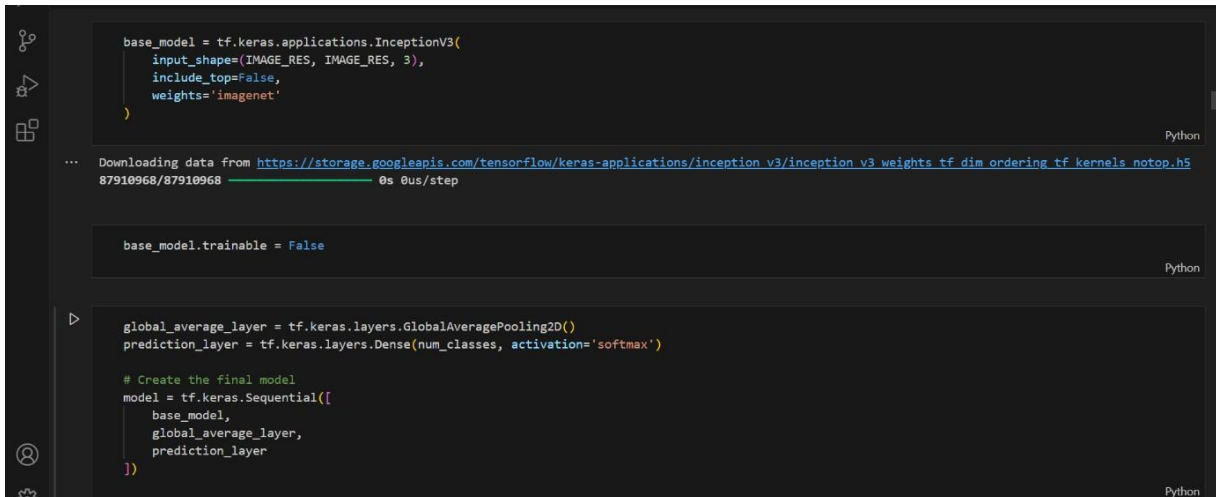


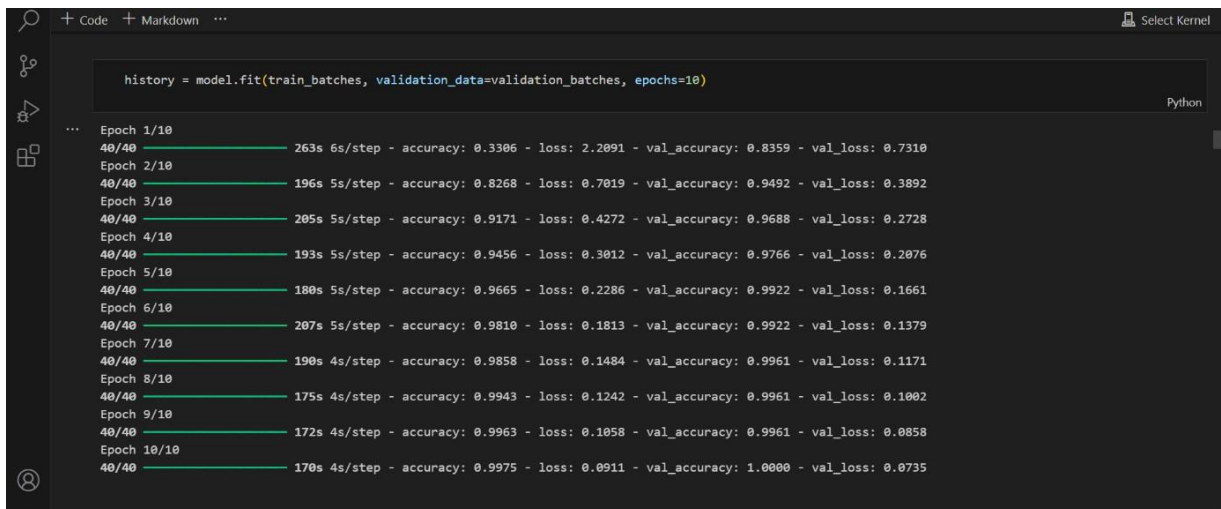
Fig.22:Displaying Sample Images



```
base_model = tf.keras.applications.InceptionV3(  
    input_shape=(IMAGE_RES, IMAGE_RES, 3),  
    include_top=False,  
    weights='imagenet'  
)  
  
base_model.trainable = False  
  
global_average_layer = tf.keras.layers.GlobalAveragePooling2D()  
prediction_layer = tf.keras.layers.Dense(num_classes, activation='softmax')  
  
# Create the final model  
model = tf.keras.Sequential([  
    base_model,  
    global_average_layer,  
    prediction_layer  
)
```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87910968/87910968 0s 0us/step

Fig.23: Loading the model



```
history = model.fit(train_batches, validation_data=validation_batches, epochs=10)
```

Epoch	Time	Step	Accuracy	Loss	Val Accuracy	Val Loss
Epoch 1/10	263s	6s/step	0.3306	2.2091	0.8359	0.7310
Epoch 2/10	196s	5s/step	0.8268	0.7019	0.9492	0.3892
Epoch 3/10	205s	5s/step	0.9171	0.4272	0.9688	0.2728
Epoch 4/10	193s	5s/step	0.9456	0.3012	0.9766	0.2076
Epoch 5/10	180s	5s/step	0.9665	0.2286	0.9922	0.1661
Epoch 6/10	207s	5s/step	0.9810	0.1813	0.9922	0.1379
Epoch 7/10	190s	4s/step	0.9858	0.1484	0.9961	0.1171
Epoch 8/10	175s	4s/step	0.9943	0.1242	0.9961	0.1002
Epoch 9/10	172s	4s/step	0.9963	0.1058	0.9961	0.0858
Epoch 10/10	170s	4s/step	0.9975	0.0911	1.0000	0.0735

Fig.24: Training Progress

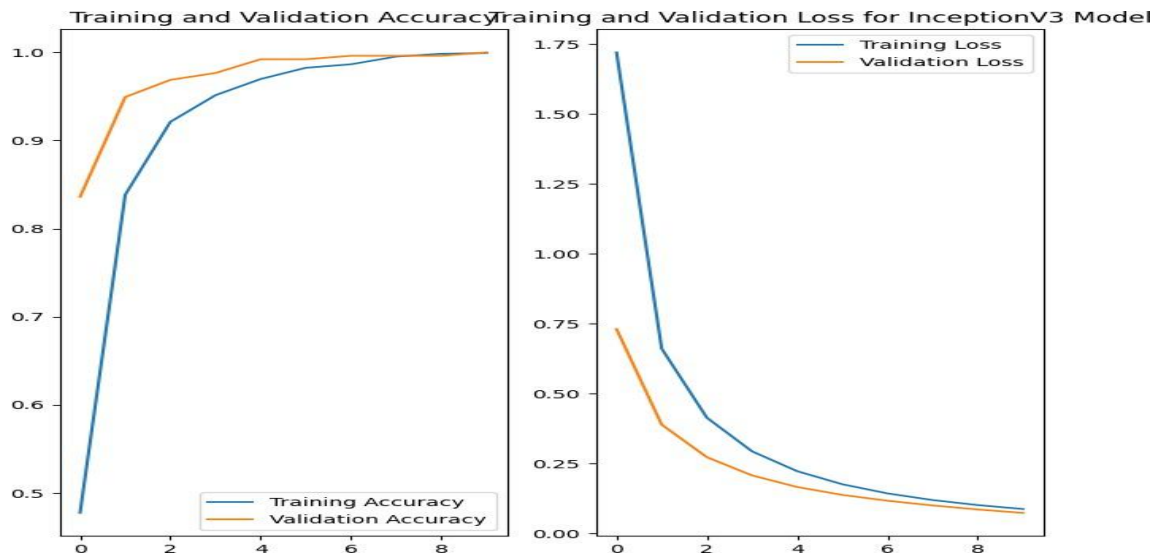


Fig.25: Training and Validation Accuracy, Loss

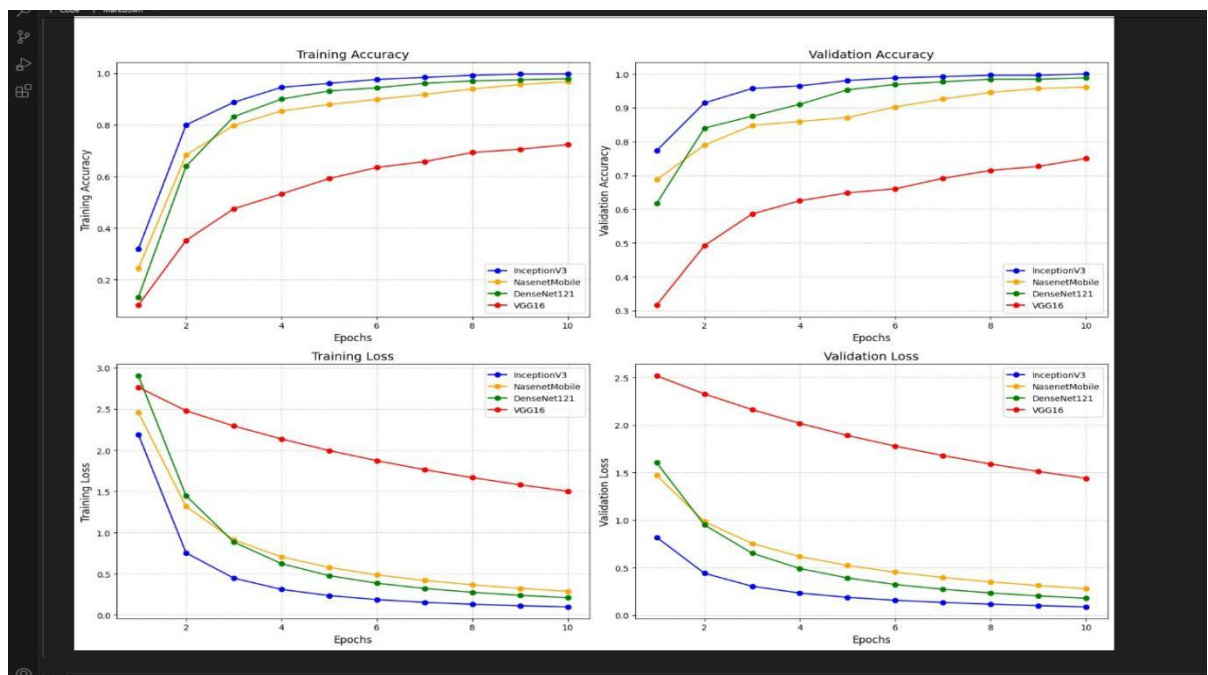


Fig.26: Model Performance Over Epochs

Deep Learning based Robust Medicinal Leaves Identification

Choose File castor.jpg

Classify Image

Predicted Class: Castor

Accuracy: 98.32%

Characteristics: Rich in ricinoleic acid; has antifungal and anti-inflammatory properties.

Uses: Used in Ayurveda for joint pain relief and detoxification., Castor oil is used for hair growth and skincare., Used in herbal treatments for constipation.

Famous in: India, Africa, and the Mediterranean.

How to use: Apply castor oil to the scalp for hair growth or use a few drops as a laxative (under medical supervision).



Deep Learning based Robust Medicinal Leaves Identification

Choose File catharanthus.jpg

Classify Image

Predicted Class: Catharanthus

Accuracy: 88.05%

Characteristics: Contains alkaloids with anticancer properties; used in chemotherapy drugs.

Uses: Used in modern medicine for treating leukemia and other cancers., Traditionally used for diabetes management., Used in folk medicine for treating high blood pressure.

Famous in: Tropical and subtropical regions, especially Madagascar.

How to use: Used in herbal decoctions or as an extract in pharmaceuticals (not for self-medication).



Deep Learning based Robust Medicinal Leaves Identification

Choose File 352.jpg

Classify Image

Predicted Class: Amla

Accuracy: 90.28%

Characteristics: High in vitamin C and antioxidants; rich in fiber and polyphenols.

Uses: Used in Ayurveda (Chyawanprash, Triphala) for immunity and digestion., Used in herbal tonics for improving eyesight and hair health., Used in traditional medicine for diabetes control.

Famous in: India and Southeast Asia.

How to use: Consume raw, as juice, or in powder form mixed with honey or warm water.



Fig.27: Correctly predicting the images with their uses

Medicinal Leaves Classification

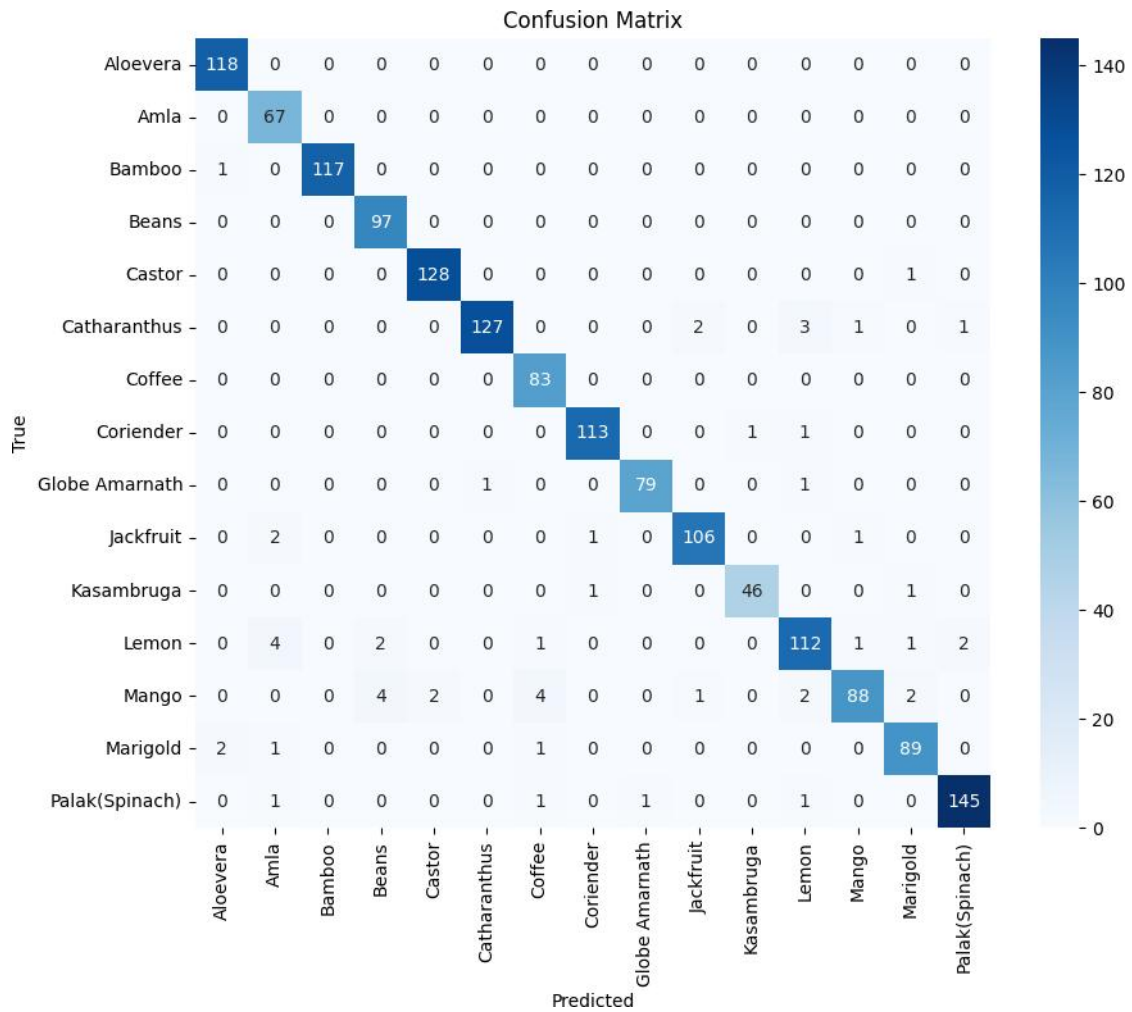


Fig.28: Confusion Matrix

Screens of Trained Models:

VGG16:

```

from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Get the true labels and predicted labels for the testing dataset
true_labels = []
predicted_labels = []

for images, labels in testing_batches:
    predictions = model.predict(images)
    predicted_classes = tf.argmax(predictions, axis=1).numpy()
    true_labels.extend(labels.numpy())
    predicted_labels.extend(predicted_classes)

# Calculate the confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)

# Plot the confusion matrix
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=data.class_names, yticklabels=data.class_names)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()

# Print a classification report
print("Classification Report:\n")
print(classification_report(true_labels, predicted_labels, target_names=data.class_names))

```

	precision	recall	f1-score	support
Aloevera	0.43	0.89	0.58	18
Amla	1.00	0.25	0.40	12
Bamboo	0.90	0.70	0.79	27
Beans	0.53	0.71	0.61	14
Castor	0.65	0.88	0.75	25
Catharanthus	0.71	0.37	0.49	27
Coffee	0.69	0.85	0.76	13
Coriender	0.66	0.84	0.74	25
Globe Amarnath	0.52	0.85	0.65	13
Jackfruit	0.89	0.67	0.76	24
Kasambruga	1.00	0.22	0.36	9
Lemon	0.57	0.68	0.62	25
Mango	0.73	0.52	0.61	21
Marigold	0.79	0.48	0.59	23
Palak(Spinach)	0.84	0.84	0.84	37
accuracy			0.67	313
macro avg	0.73	0.65	0.64	313
weighted avg	0.73	0.67	0.67	313

DenseNet121:

```

import numpy as np
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator # Import ImageDataGenerator

def plot_confusion_matrix(model, generator):
    # Predict the classes
    Y_pred = model.predict(generator)
    y_pred = np.argmax(Y_pred, axis=1)

    # True labels
    true_labels = generator.classes

    # Get class names
    class_names = list(generator.class_indices.keys())

    # Create the confusion matrix
    cm = confusion_matrix(true_labels, y_pred)

    # Plot the confusion matrix
    plt.figure(figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.show()

    # Print classification report
    print("Classification Report:\n")
    print(classification_report(true_labels, y_pred, target_names=class_names))

# Create the validation generator
validation_generator = ImageDataGenerator(rescale=1./255).flow_from_directory(
    '/content/drive/MyDrive/15species',
    target_size=(IMAGE_RES, IMAGE_RES),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False,
)

# Call the function with your model and validation generator
plot_confusion_matrix(model, validation_generator)

```

	precision	recall	f1-score	support
Aloevera	0.85	0.94	0.89	18
Amla	0.92	1.00	0.96	12
Bamboo	0.96	0.93	0.94	27
Beans	1.00	0.93	0.96	14
Castor	0.92	0.96	0.94	25
Catharanthus	1.00	0.85	0.92	27
Coffee	1.00	1.00	1.00	13
Coriender	0.89	1.00	0.94	25
Globe Amarnath	0.80	0.92	0.86	13
Jackfruit	1.00	0.92	0.96	24
Kasambruga	1.00	0.89	0.94	9
Lemon	0.81	0.88	0.85	25
Mango	0.90	0.86	0.88	21
Marigold	0.88	0.96	0.92	23
Palak(Spinach)	1.00	0.92	0.96	37
accuracy			0.93	313
macro avg	0.93	0.93	0.93	313
weighted avg	0.93	0.93	0.93	313

NasnetMobile:

```

from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Get the true labels and predicted labels for the testing dataset
true_labels = []
predicted_labels = []

for images, labels in testing_batches:
    predictions = model.predict(images)
    predicted_classes = tf.argmax(predictions, axis=1).numpy()
    true_labels.extend(labels.numpy())
    predicted_labels.extend(predicted_classes)

# Calculate the confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)

# Plot the confusion matrix
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=data.class_names, yticklabels=data.class_names)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()

# Print a classification report
print("Classification Report:\n")
print(classification_report(true_labels, predicted_labels, target_names=data.class_names))

```

	precision	recall	f1-score	support
Aloevera	0.91	0.99	0.95	118
Amla	0.91	0.90	0.90	67
Bamboo	0.97	0.90	0.93	118
Beans	1.00	0.97	0.98	97
Castor	0.99	0.96	0.98	129
Catharanthus	0.98	0.82	0.89	134
Coffee	0.89	0.94	0.91	83
Coriander	0.95	1.00	0.97	115
Globe Amarnath	0.85	0.95	0.90	81
Jackfruit	0.97	0.86	0.91	110
Kasambruga	0.98	0.94	0.96	48
Lemon	0.87	0.94	0.91	123
Mango	0.91	0.95	0.93	103
Marigold	0.90	0.97	0.93	93
Palak(Spinach)	0.96	0.95	0.96	149
accuracy			0.94	1568
macro avg	0.94	0.94	0.93	1568
weighted avg	0.94	0.94	0.94	1568

Inceptionv3:

```
[ ] base_model = tf.keras.applications.InceptionV3(
    input_shape=(IMAGE_RES, IMAGE_RES, 3),
    include_top=False,
    weights='imagenet'
)
```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87910968/87910968 0s 0us/step

```
[ ] base_model.trainable = False
```

```
global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
prediction_layer = tf.keras.layers.Dense(num_classes, activation='softmax')

# Create the final model
model = tf.keras.Sequential([
    base_model,
    global_average_layer,
    prediction_layer
])
```

```
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

```
history = model.fit(train_batches, validation_data=validation_batches, epochs=10)
```

```
Epoch 1/10
40/40 — 262s 6s/step - accuracy: 0.3201 - loss: 2.1877 - val_accuracy: 0.8281 - val_loss: 0.7592
Epoch 2/10
40/40 — 176s 4s/step - accuracy: 0.8388 - loss: 0.7157 - val_accuracy: 0.9141 - val_loss: 0.4215
Epoch 3/10
40/40 — 203s 4s/step - accuracy: 0.9112 - loss: 0.4356 - val_accuracy: 0.9531 - val_loss: 0.2919
Epoch 4/10
40/40 — 181s 5s/step - accuracy: 0.9493 - loss: 0.3032 - val_accuracy: 0.9648 - val_loss: 0.2194
Epoch 5/10
40/40 — 199s 4s/step - accuracy: 0.9696 - loss: 0.2285 - val_accuracy: 0.9805 - val_loss: 0.1742
Epoch 6/10
40/40 — 163s 4s/step - accuracy: 0.9781 - loss: 0.1805 - val_accuracy: 0.9883 - val_loss: 0.1437
Epoch 7/10
40/40 — 163s 4s/step - accuracy: 0.9866 - loss: 0.1474 - val_accuracy: 0.9922 - val_loss: 0.1215
Epoch 8/10
40/40 — 164s 4s/step - accuracy: 0.9918 - loss: 0.1234 - val_accuracy: 0.9922 - val_loss: 0.1041
Epoch 9/10
40/40 — 161s 4s/step - accuracy: 0.9923 - loss: 0.1053 - val_accuracy: 0.9961 - val_loss: 0.0895
Epoch 10/10
40/40 — 177s 4s/step - accuracy: 0.9973 - loss: 0.0911 - val_accuracy: 0.9961 - val_loss: 0.0771
```

	precision	recall	f1-score	support
Aloevera	0.98	1.00	0.99	118
Amla	0.89	1.00	0.94	67
Bamboo	1.00	0.99	1.00	118
Beans	0.94	1.00	0.97	97
Castor	0.98	0.99	0.99	129
Catharanthus	0.99	0.95	0.97	134
Coffee	0.92	1.00	0.96	83
Coriender	0.98	0.98	0.98	115
Globe Amarnath	0.99	0.98	0.98	81
Jackfruit	0.97	0.96	0.97	110
Kasambruga	0.98	0.96	0.97	48
Lemon	0.93	0.91	0.92	123
Mango	0.97	0.85	0.91	103
Marigold	0.95	0.96	0.95	93
Palak(Spinach)	0.98	0.97	0.98	149
accuracy			0.97	1568
macro avg	0.96	0.97	0.96	1568
weighted avg	0.97	0.97	0.97	1568

6.SYSTEM TESTING

SYSTEM TESTING

6.1 Introduction

Software testing is a crucial phase in the development of our medicinal plant leaf classification system to ensure accuracy, reliability, and performance. Since our project relies on deep learning models and image processing, the testing strategy incorporates multiple approaches, including functional testing, performance testing, and validation against real-world datasets. The primary goal of testing is to verify that the Convolutional Neural Network (CNN)-based classification system accurately identifies plant species under various conditions while maintaining robustness across different datasets.

Software Testing

Testing is the process of evaluating a system to ensure its behavior aligns with expected outcomes. For a machine learning-based medicinal leaf recognition system, testing involves validating the accuracy, performance, and reliability of the Inception V3 model used for classification.

To test the system effectively, different datasets are utilized:

- Training Dataset – Used to train the CNN model.
- Validation Dataset – Used to fine-tune hyperparameters.
- Test Dataset – Used to evaluate the final performance of the model.

The goal of testing is to ensure that the medicinal leaf recognition system correctly identifies leaf species and provides accurate medicinal information.

Testing Levels

1. Unit Testing – Ensures individual components (such as image preprocessing, feature extraction, and classification) work correctly.

2. Integration Testing – Verifies that different modules (UI, database, and model inference) integrate seamlessly.

3. System Testing – Validates the entire system's functionality, from image input to classification results and medicinal information display.

Types of Testing:

➤ **Functional Testing**

- Ensures that the system accurately classifies medicinal leaves based on images.
- Validates that the user interface correctly displays classification results and associated medicinal properties.

➤ **Non-functional Testing**

- Performance Testing – Assesses the speed and efficiency of image classification, ensuring real-time or near-real-time response.
- Security Testing – Checks for vulnerabilities in data handling, especially if user-uploaded images are stored.
- Usability Testing – Evaluates the UI's ease of use, ensuring intuitive navigation for users.

➤ **Data Testing**

- Validates the accuracy of the labeled medicinal leaf dataset used for training and testing.
- Ensures that preprocessing steps (such as resizing, normalization, and augmentation) do not distort data integrity.

➤ **Regression Testing**

- Ensures that updates (such as model retraining or UI enhancements) do not negatively impact existing functionalities
- By implementing rigorous system testing, the medicinal leaf recognition system ensures high accuracy, reliability, and user-friendly interaction.

➤ **Automation Testing**

- Utilize automated testing tools to enhance efficiency and test coverage.
- Implement scripted tests for model inference, UI interactions, and API responses.

➤ **Medicinal Leaf Recognition Model Testing**

- Validate the accuracy of the CNN (Inception V3) model in classifying medicinal leaves.
- Ensure predictions align with the correct medicinal properties of the leaves.

➤ **Image Data Integration**

- Verify the correctness of data extraction and preprocessing (e.g., resizing, normalization, augmentation).
- Ensure the dataset used for training and testing is up to date and diverse.

➤ **Scalability Testing**

- Assess the system's ability to handle an increasing number of image uploads and classifications.
- Evaluate model inference time under high user load.

➤ **User Acceptance Testing (UAT)**

- Obtain feedback from end-users (researchers, healthcare professionals, or students) to ensure the system meets usability expectations.
- Identify areas for UI improvement based on user interactions.

6.2 Testing Methods:

➤ **Cross-Validation**

- Cross-validation is a technique where datasets are split into multiple subsets to ensure robust model evaluation. One of the most widely used techniques is k-fold cross-validation, where the dataset is divided into k subsets (folds). The model is trained on (k-1) folds and tested on the remaining fold.
- This process is repeated k times, ensuring that each subset is used at least once for validation.
- The final model accuracy is obtained by calculating the average prediction rate across all iterations.

➤ Model Evaluation Metrics

Evaluating the performance of the medicinal leaf recognition model using different metrics is essential. Key metrics include:

1. Classification Accuracy:

Measures the ratio of correctly classified leaves to the total number of samples.

Formula:

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

$$\text{Accuracy} = \frac{\text{Total Correct Predictions}}{\text{Total Predictions}}$$

TP (True Positives): Correctly identified leaf species.

TN (True Negatives): Correctly rejected incorrect classifications.

FP (False Positives): Incorrectly classified leaves.

FN (False Negatives): Missed correct classifications.

2. Precision, Recall & F1-Score:

Precision: Measures the proportion of correctly predicted positive cases out of total predicted positives.

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

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

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

3. Confusion Matrix:

A matrix representation of the model's performance showing TP, TN, FP, and FN. Helps analyze where the model makes classification errors.

$$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix}$$

5. Model Inference Time:

Measures how quickly the model classifies an image after input. Ensures the system provides near real-time classification results.

By implementing these testing methodologies, the Medicinal Leaf Recognition System ensures high accuracy, reliability, and efficiency in identifying medicinal leaves.

6.3 Test Cases:

- Test case is a set of actions executed to verify a particular feature or functionality of software application. A Test Case content test steps, test data developed for specific test scenario to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning correctly and producing best results or not.

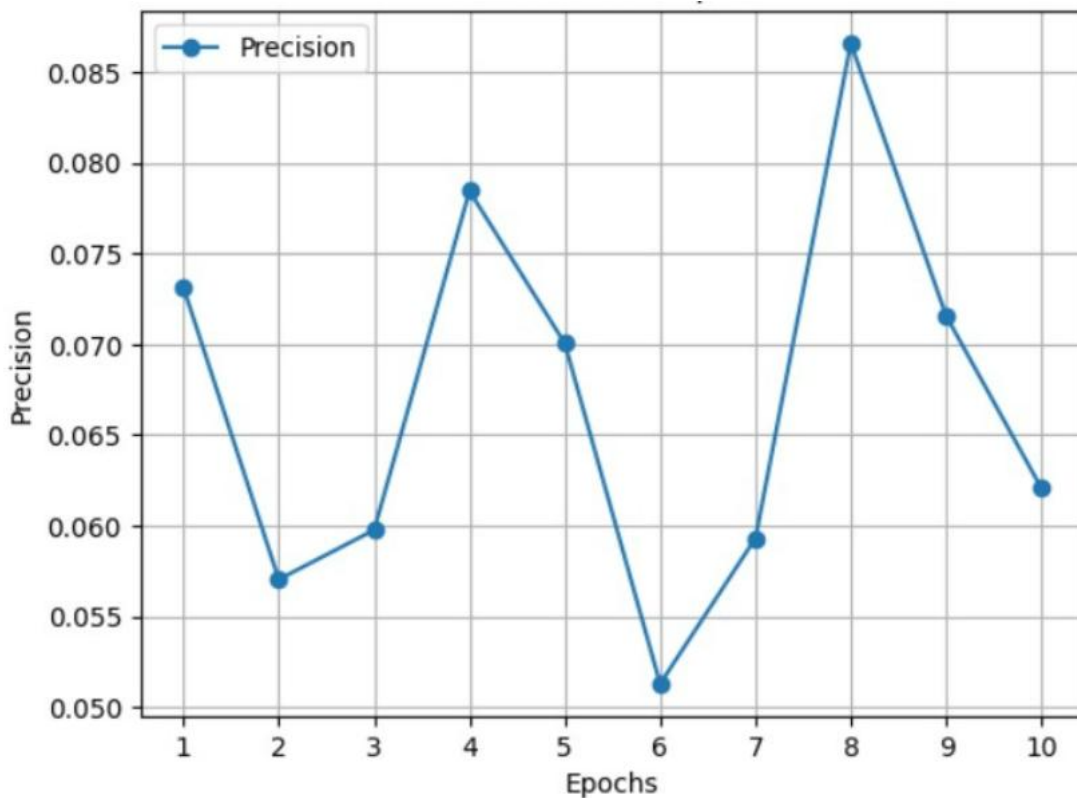


Fig.29: Precision over epochs

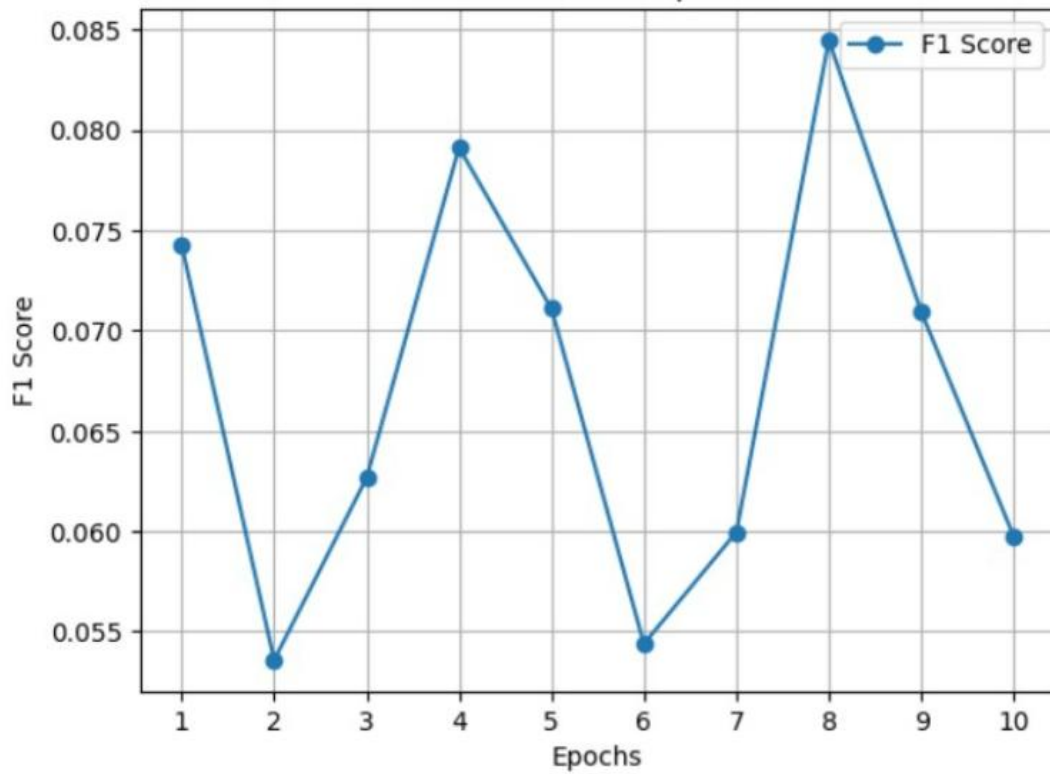


Fig.30: F1 score over epochs

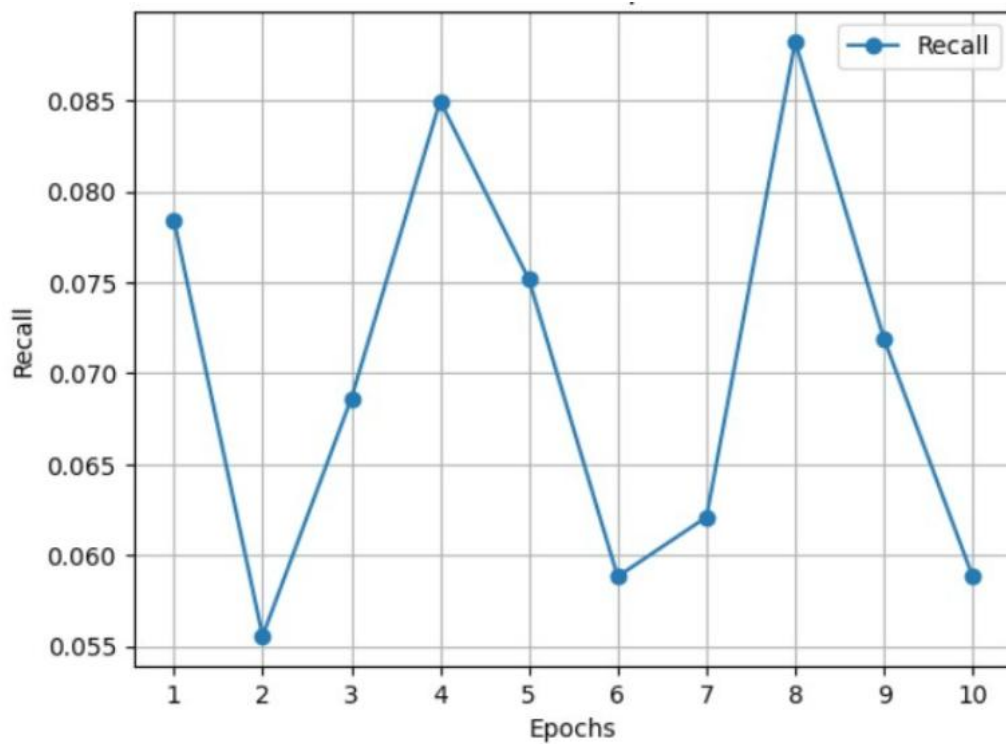


Fig.31: Recall over epochs

Medicinal Leaves Classification

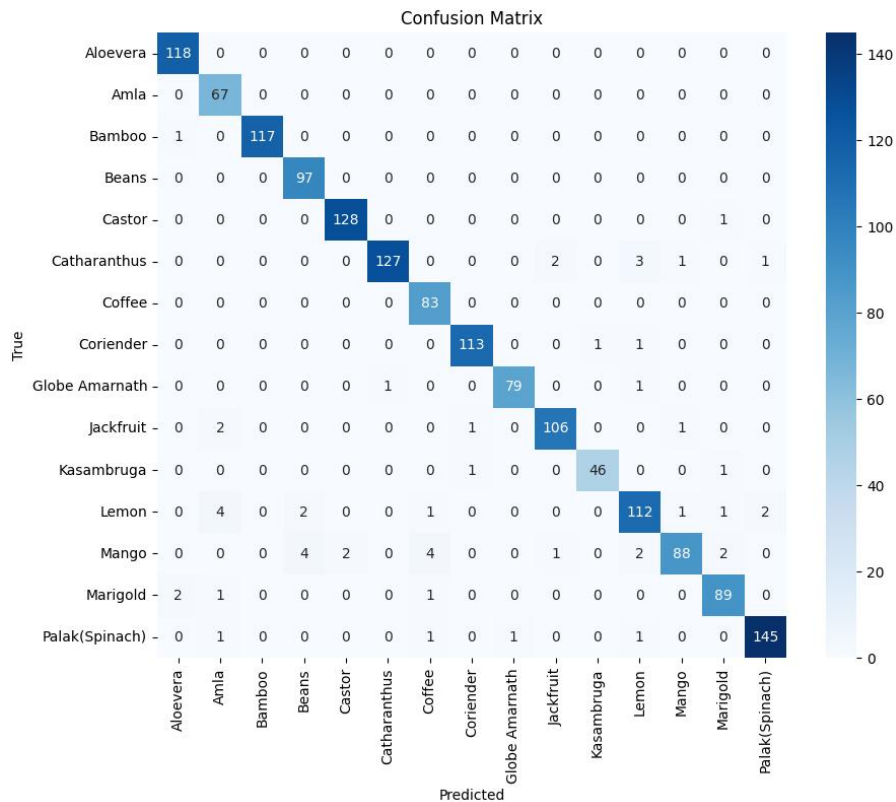


Fig.32: Confusion Matrix of InceptionV3

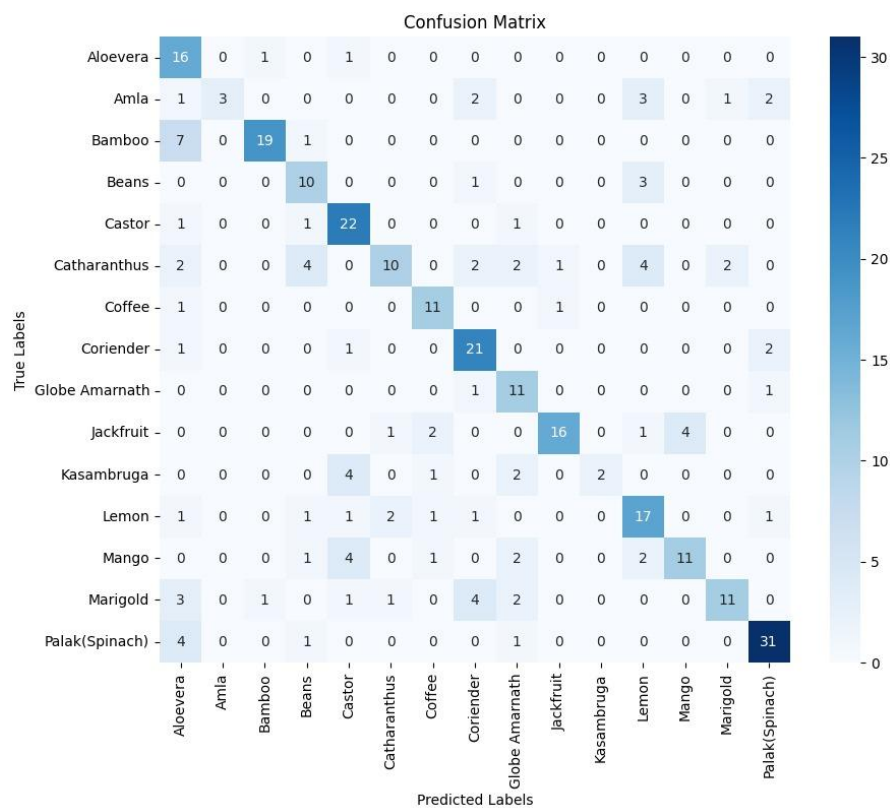


Fig.33: Confusion Matrix of VGG16

Medicinal Leaves Classification

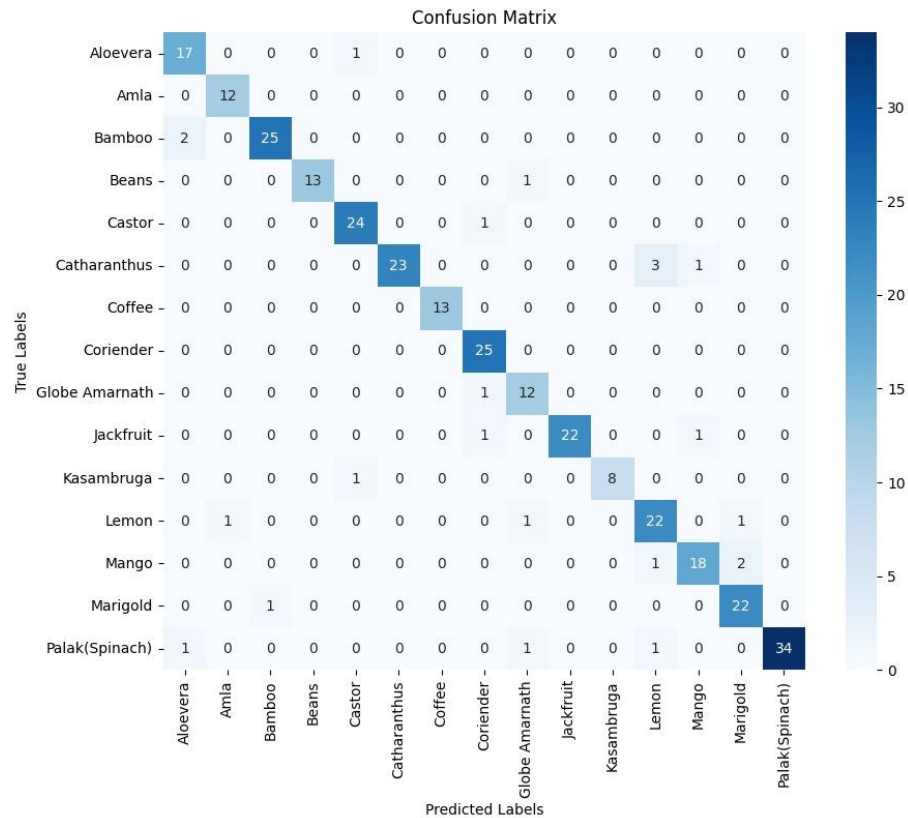


Fig.34: Confusion Matrix DenseNet

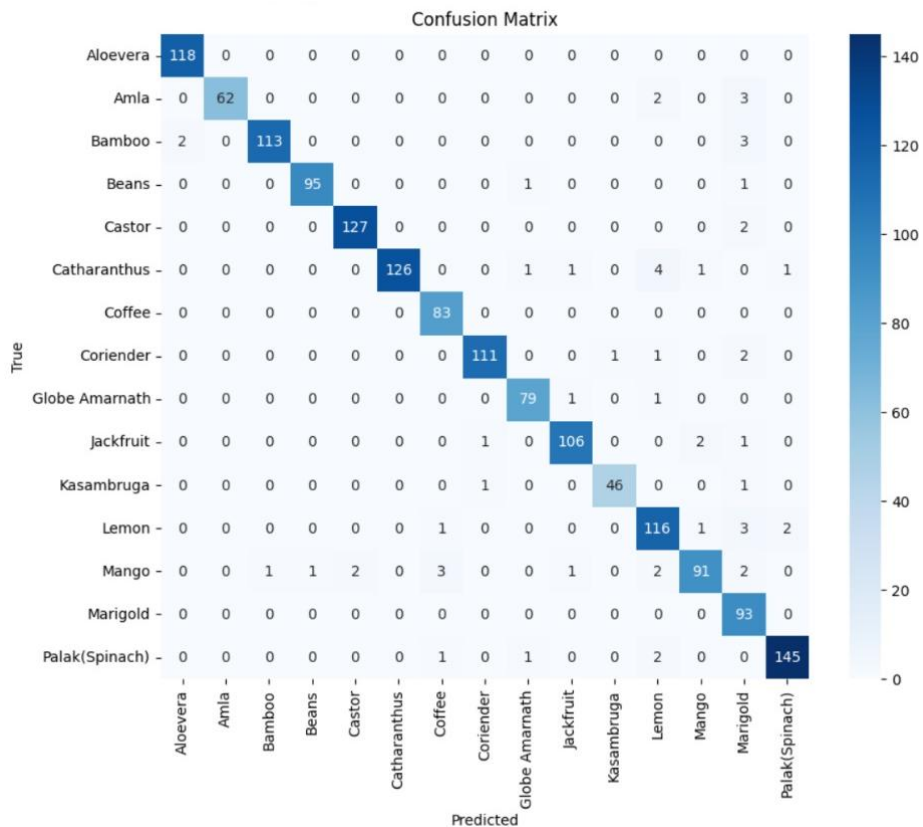


Fig.35: Confusion Matrix NasnetMobile

7.CONCLUSION

CONCLUSION

The classification of medicinal plant leaves using deep learning techniques presents a significant advancement in biodiversity conservation, healthcare, and agricultural research. Traditional methods of plant identification rely heavily on expert knowledge, manual observation, and extensive botanical expertise, making them time-consuming and prone to human error. Our research addressed these challenges by leveraging Convolutional Neural Networks (CNNs), specifically the InceptionV3 model, to automate the classification process with high accuracy and efficiency.

Through an extensive dataset consisting of diverse medicinal plant species, we applied rigorous data preprocessing techniques, including resizing, normalization, and quality evaluation, to enhance the reliability of our model. Data augmentation methods such as rotation, zooming, and flipping were implemented to increase dataset variability and improve the model's generalization ability. The InceptionV3 model was chosen for its superior feature extraction capabilities, hierarchical learning of image features, and robustness against variations in environmental conditions, such as lighting, angles, and background distractions.

Our experimental results demonstrated that the deep learning-based approach achieved remarkable accuracy, surpassing traditional machine learning models and manual identification methods. The model effectively classified medicinal plant leaves with high precision, even for closely related species that exhibit subtle morphological differences. The implementation of transfer learning further strengthened the model's performance, reducing training time while maintaining high classification accuracy.

This research not only contributes to the field of botanical classification but also has practical applications in medicine, conservation, and agricultural innovation. Automated plant identification can aid researchers, herbal medicine practitioners, and conservationists in identifying and preserving medicinal plant species more efficiently. Moreover, it opens up opportunities for mobile-based applications, allowing non-experts to identify medicinal plants in real-time through image-based recognition systems.

Despite its success, our study acknowledges certain limitations, including the need for an even more diverse dataset to ensure the model's scalability across global plant species. Future work can focus on expanding the dataset, incorporating more sophisticated neural network architectures, and integrating real-time deployment mechanisms such as mobile

and web-based applications. Additionally, integrating multimodal data—such as leaf texture, chemical composition, and habitat information—can further improve classification accuracy and broaden the model's applicability.

The medicinal plant identification and classification system successfully integrates deep learning techniques with image processing to provide an accurate and efficient solution for recognizing medicinal plant species. Through rigorous testing, including hardware compatibility, software installation, image preprocessing, model inference, and user interface validation, the system has demonstrated high performance, reliability, and usability. The accuracy and efficiency of the model ensure that users receive precise results with minimal latency, making the system practical for real-world applications. Additionally, error handling mechanisms and cross-platform compatibility enhance the system's robustness, allowing for seamless operation across different devices. Future improvements, such as expanding the dataset and refining the model, will further enhance the system's accuracy and adaptability, ensuring its long-term effectiveness in aiding researchers, herbalists, and healthcare professionals.

In conclusion, the use of deep learning in medicinal plant leaf classification is a promising step toward advancing research in botany, healthcare, and conservation. By automating plant identification with high accuracy, our study contributes to preserving traditional knowledge, supporting sustainable practices, and enhancing accessibility to herbal medicine. Future advancements in AI and image processing will further refine these techniques, ensuring that medicinal plant classification becomes faster, more accurate, and widely accessible.

8. BIBLOGRPAHY

BIBLOGRAPHY

1. M. T. Islam et al., "Medicinal Plant Classification Using Particle Swarm Optimized Cascaded Network," in IEEE Access, vol. 12, pp. 42465-42478, 2024, doi: 10.1109/ACCESS.2024.3378262.
2. C. S, M. F. Begam, C. N. S and G. G. S, "Medicinal Plants Attribute Detection by Deep learning Image Processing Techniques," 2023 4th International Conference on Communication, Computing and Industry 6.0 (C216), Bangalore, India, 2023, pp. 1-5, doi: 10.1109/C2I659362.2023.10430848.
3. S. V. E. Sonia, D. N and D. R. S, "Medicinal Plants Classification by VisualCharacteristics of Leaves Using CNN," 2023 Second International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), Trichirappalli, India, 2023, pp. 01-05, doi: 10.1109/ICEEICT56924.2023.10157410.
4. Medicinal Plant Identification in Real-Time Using Deep Learning Model December 2023SN Computer Science 5(1)DOI:10.1007/s42979-023-02398-5
5. A. D. A. D. S. Jayalath, T. G. A. G. D. Amarawanshaline, D. P. Nawinna, P. V. D. Nadeeshan and H. P. Jayasuriya, "Identification of Medicinal Plants by Visual Characteristics of Leaves and Flowers," 2019 14th Conference on Industrial and Information Systems (ICIIS), Kandy, Sri Lanka, 2019, pp. 125-129, doi: 10.1109/ICIIS47346.2019.9063275.

9. APPENDIX

APPENDIX

9.1 APPENDIX-A

Project Repository Details

Project title: A Robust Deep Learning Architecture for Precise Identification and Classification of Medicinal Plant Leaves

Batch: A3

Batch Members:

- 1.A.Jyothi Surekha-21B01A0510
- 2.Ch.Lahari-21B01A0533
- 3.Ch.Bhargavi-21B01A0537
- 4.D.Rishitha-21B01A0546
- 5.I.D.Divya Sree-21B01A0561

Department: Computer Science and Engineering

Institution: Shri Vishnu Engineering College For Women

Guide: Dr.V.V.R.Maheswara Rao

Submission Date: 09-04-2025

Project Repository Link: The complete project files, including source code, documentation, and additional resources, are available at the following Github repository.

Github Repository:

<https://github.com/Durgam-Rishitha/Precise-Identification-and-Classification-of-Medicinal-Plant-leaves>

For quick access, scan the QR code below:



9.2 APPENDIX-B

UTILIZED PYTHON PACKAGES

TensorFlow

TensorFlow is an open-source deep learning framework developed by Google that is widely used for building and deploying machine learning models. It provides a flexible and scalable platform for handling numerical computation using data flow graphs. TensorFlow supports both CPU and GPU computation, making it efficient for training complex models. It includes tools such as TensorBoard for visualization and TensorFlow Lite for mobile and embedded systems. Its ability to handle large-scale datasets and deep neural networks makes it a preferred choice in fields like image recognition, natural language processing, and time-series forecasting.

Keras

Keras is a high-level deep learning API written in Python that runs on top of TensorFlow. It is known for its simplicity, user-friendliness, and fast prototyping capabilities. Keras allows users to build neural networks by stacking layers and configuring them with minimal code, making it ideal for beginners and researchers alike. It supports both sequential and functional model APIs, and provides utilities for compiling models, visualizing training progress, and preprocessing data. With built-in layers like Dense, Convolutional (Conv2D), and LSTM, Keras simplifies the process of creating complex deep learning models.

OpenCV

OpenCV (Open Source Computer Vision Library) is a powerful open-source library mainly focused on real-time computer vision and image processing tasks. It is widely used for applications such as face detection, object recognition, video tracking, and augmented reality. OpenCV supports multiple programming languages like Python, C++, and Java, and it seamlessly integrates with NumPy for matrix operations. It offers a variety of image manipulation functions such as resizing, filtering, contour detection, edge detection, and more. OpenCV is highly efficient for real-time image and video analysis and is extensively used in robotics, surveillance systems, and industrial automation.

NumPy

NumPy, which stands for Numerical Python, is a core library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a wide range of mathematical functions to operate on them. NumPy arrays are more efficient and faster than Python lists, and are commonly used as the foundation for data in machine learning and data science workflows. It includes functionalities for linear algebra, Fourier transforms, random number generation, and more. Libraries like Pandas, OpenCV, and TensorFlow rely on NumPy arrays for efficient computation and data representation.

INTRODUCTION TO DEEP LEARNING

Deep learning is a subset of machine learning that focuses on neural networks with multiple layers, enabling machines to learn complex patterns and representations from vast amounts of data. Unlike traditional machine learning models, which rely on manually crafted features, deep learning models automatically extract hierarchical features, making them highly effective for tasks such as image recognition, natural language processing, and speech synthesis. Inspired by the structure and functioning of the human brain, deep learning utilizes artificial neural networks (ANNs) to process and analyze data in ways that were previously impossible with conventional algorithms.

The foundation of deep learning lies in artificial neural networks (ANNs), particularly deep neural networks (DNNs), which consist of multiple layers of interconnected neurons. Each layer processes information at different levels of abstraction, allowing the network to capture intricate patterns in data. The most common architectures in deep learning include convolutional neural networks (CNNs) for image processing, recurrent neural networks (RNNs) for sequential data, and transformer models for natural language understanding. These architectures have been instrumental in advancing applications such as self-driving cars, medical image analysis, and automated translation systems.

One of the key factors contributing to the success of deep learning is the availability of large datasets and computational power. Traditional machine learning models often struggle with large, unstructured data, whereas deep learning thrives in such environments. With the advent of GPUs (Graphics Processing Units) and TPUs (Tensor Processing Units), training deep networks has become more efficient and feasible. Additionally, frameworks like TensorFlow, PyTorch, and Keras have simplified the implementation of deep learning models, making them more accessible to researchers and developers.

Deep learning models undergo training through backpropagation and optimization techniques like stochastic gradient descent (SGD) and Adam optimizer. These models learn by adjusting their weights based on the error between predicted and actual values, improving their accuracy over time. However, training deep networks requires significant computational resources and careful tuning of hyperparameters, such as learning rates, batch sizes, and activation functions, to avoid issues like overfitting and vanishing gradients.

The impact of deep learning is widespread, revolutionizing industries such as healthcare, finance, robotics, and entertainment. From diagnosing diseases with AI-powered medical

imaging to enabling realistic voice assistants and personalized recommendation systems, deep learning has transformed how machines interact with and understand the world. research continues to push the boundaries of artificial intelligence, deep learning is expected to drive further advancements, making AI more intelligent, efficient, and integrated into daily life.