DAYANANDA SAGAR UNIVERSITY

SCHOOL OF ENGINEERING

Department of Computer Science & Engineering (AIML)
Devarakaggalahalli, Harohalli Kanakapura Road,
Ramnagara - 562112
Karnataka, India

A
Special Topic-1
Report
On

"MEDICINAL PLANTS IDENTIFICATION USING NN"

Department of Computer Science Engineering (AI & ML)

SUBMITTED BY
Kasala Bhavana (ENG22AM0153)
Lakshya U Reddy (ENG22AM0169)
Nishat N Shahu (ENG22AM0184)
Tanya Gopal (ENG22AM0193)

Under the supervision of

Dr. Joshuva Arockia Dhanraj
Associate Professor
Dept. of AIML, SOE, DSU

2023-2024

DAYANANDA SAGAR UNIVERSITY

School of Engineering
Department of Computer Science & Engineering (AIML)
Devarakaggalahalli, Harohalli, Kanakapura Road,
Ramnagara – 562112
Karnataka, India

Department of Computer Science Engineering (AI & ML)

CERTIFICATE

This is to certify that Special Topic -1 (22AM2406) Project work titled **"MEDICINAL PLANTS IDENTIFICATION USING NN "** is carried out by **Kasala Bhavana (ENG22AM0153)**, **Lakshya U Reddy (ENG22AM0169), Nishat N Shahu (ENG22AM0184), Tanya Gopal (ENG22AM0193),** bonafide students of Bachelor of Technology in Computer Science and Engineering (AI&ML) at the School of Engineering, Dayananda Sagar University in partial fulfillment for the award of degree in Bachelor of Technology in Computer Science and Engineering (AI&ML),during the year 2023-2024.

Signature of Guide

Dr.Joshuva Arockia Dhanraj
Associate Professor
Dept. of AIML, SOE, DSU

Signature of Chairperson

Dr. Jayavrinda Vrindavanam, Ph.D ,
Professor and Chairperson,
Dept. of AIML, SOE, DSU

DECLARATION

We **Kasala Bhavana (ENG22AM0153), Lakshya U Reddy (ENG22AM0169), Nishat N Shahu (ENG22AM0184), Tanya Gopal (ENG22AM0193),** are students of fourth semester B. Tech in **Computer Science and Engineering (AI&ML)**, at School of Engineering, **Dayananda Sagar University**, hereby declare that the Special Topic-1 titled **"MEDICINAL PLANT IDENTIFICATION USING NN"** has been carried out by us and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Computer Science and Engineering (AI&ML)** during the academic year **2023-2024.**

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to School of Engineering & Technology, Dayananda Sagar University for providing us with a great opportunity to pursue our Bachelor's degree in this institution.

We would like to thank ***Dr. Udaya Kumar Reddy K R, Dean, School of Engineering & Technology, Dayananda Sagar University*** *for his constant encouragement and expert advice.*

It is a matter of immense pleasure to express our sincere thanks to ***Dr.Jayavrinda Vrindavanam, Department Chairman****,* ***Computer Science and Engineering (AI&ML)****,* ***Dayananda Sagar University,*** *for providing right academic guidance that made our task possible.*

We would like to thank our guide ***Dr. Joshuva Arockia Dhanraj, Associate Professor, Dept. of Computer Science and Engineering (AI&ML)****, for sparing his valuable time to extend help in every step of our Special Topic-1 work which paved the way for smooth progress and fruitful culmination of the research.*

We would like to thank our ***Special Topic-1 Coordinators Dr. Jayavrinda Vrindavanam, Professor, Dr. Joshuva Arockia Dhanraj, Associate Professor, and Dr. Mude Nagarjuna Naik, Assistant Professor*** *and all the staff members of Computer Science and Engineering (AI&ML) for their support.*

We are also grateful to our family and friends who provided us with every requirement throughout the course. We would like to thank one and all who directly or indirectly helped us in the Special Topic-1 work.

TABLE OF CONTENTS

LIST OF ABBREVIATION

NN	Neural Network
CNN	Central Neural Network

LIST OF FIGURES

ABSTRACT

India with a rich heritage of floral diversity, is well known for its medical plant wealth, but their identification is one of the major issues. A study published in the Journal of Ethnopharmacology estimated that approximately 70-80% of the Indian population depends on traditional medicine systems like ayurvedic medicines. Ayurvedic medicines have a vital role in preserving physical and mental health of human beings. Identification and classification of medicinal plants are essential for better treatment. Lack of experts in this field makes proper identification and classification of medicinal plants a tedious task. Hence, a fully automated system for medicinal plant classification is highly desirable. This project proposes a model that is designed to classify images of plants into various categories using Neural Network (NN), aiding in the accurate identification of medicinal plants. Our approach is to collect data, clean the data, build model, train and test the model and then deploy the model. The ultimate goal is to provide a reliable tool that can be used for educational purposes, botanical research, and potentially in medicinal plant discovery and application.

Keywords—Neural Network, Identification, Classification, Medicinal Plants.

CHAPTER 1

INTRODUCTION

CHAPTER 1 INTRODUCTION

Plants play a crucial role in preserving life and maintaining biodiversity on earth by facilitating air and water for living beings. Medicinal plants, one of the important class of plants, serve as medicine for many diseases. The knowledge about medicinal plants carried by generations must be preserved and protected. Computer vision, pattern recognition, and image processing technologies provide promising results for identification and classification of medicinal plants. Identifying a medicinal plant with required medicinal values is one of the major challenging tasks. Even though herbal medicine has no side effects, treatment using a wrongly identified medicinal plant may claim the life of a patient. Hence, a fully automated system to correctly classify the medicinal plants is inevitable at this point of time.

1.1 Need for Identification

Identification and classification of medicinal plants play a crucial role in the preparation of ayurvedic medicines. Also, proper classification of medicinal plants is important for agronomist, botanist, ayurvedic medicinal practitioners, forest department officials and those who are involved in the preparation of ayurvedic medicines. But lack of expert taxonomists is a major issue in this area. Nowadays there is a potential class of customers who prefer ayurvedic medicines than other medicines. Features of leaf, flower, trunk, and branch are used by the taxonomists for classification purposes. Since the leaves are available on all seasons, it is the best choice for plant classification.

Computer vision and image processing methods can bridge the gap between the lack of expert taxonomists and potential requirement in identification and classification of the medicinal plants. Shape, color, and textures are the important spatial and morphological features used by the researchers to classify the plants. But the color feature is not a judicious feature for classification since it varies on seasons and also different stages of the same leaf will have a different color. Taxonomists use the variations in leaf characteristics as a tool for the classification of medicinal plants.

1.2 Social Impact

This project can empower individuals, especially those in regions with rich biodiversity and traditional medicinal practices, to accurately identify medicinal plants. This could enhance healthcare accessibility and self-sufficiency, particularly in rural or underserved communities where access to trained botanists or herbalists may be limited. Additionally, by promoting the sustainable use of medicinal plants, the project could contribute to biodiversity conservation and preservation of traditional knowledge systems.

Furthermore, improved plant identification can facilitate research in pharmacology and traditional medicine, potentially leading to the discovery of new medicinal compounds and treatments.

1.3 Scope

This project envisions a broad scope to develop a system for accurately identifying medicinal plants through neural network techniques. This involves collecting and preprocessing a dataset of plant images, selecting or designing appropriate neural network architectures, training and validating the model, integrating it into a user-friendly application or platform, testing its performance, and documenting the entire process comprehensively.

Chapter 2

PROBLEM DEFINITION

Chapter 2 PROBLEM DEFINITION

Problem:

Accurate identification of medicinal plant is essential for their use in medicine. Manual identification by experts is slow, error-prone, and not scalable, especially in areas with limited botanical expertise. How do we assist botanists, researchers, and the public in recognizing and utilizing these plants effectively.

Solution:

To address this problem, we have developed a deep learning-based model using Convolutional Neural Networks (CNNs) to automate the identification of medicinal plants from images. CNNs are particularly well-suited for image recognition tasks due to their ability to learn and extract hierarchical features from images.

Chapter 3

LITERATURE REVIEW

Chapter 3 Literature Review

Dileep M.R [1] proposed a AyurLeaf, a Deep Learning based Convolutional Neural Network (CNN) model, to classify medicinal plants using leaf features such as shape, size, color, texture etc . This research work also proposes a standard dataset for medicinal plants, commonly seen in various regions of Kerala, the state on southwestern coast of India. The proposed dataset contains leaf samples from 40 medicinal plants. A deep neural network inspired from Alexnet is utilised for the efficient feature extraction from the dataset. Finally, the classification is performed using Softmax and SVM classifiers. Three-fold cross-validation and five-fold cross-validation were specifically employed for training and testing.

Arunaggiri Pandian K [2], This paper explores the use of Convolutional Neural Network (CNN) based methodologies for identifying Indian leaf species, particularly focusing on medicinal plants found in rural areas. Three pre-trained CNN architectures, namely ResNet101, InceptionV3, and VGG16, trained on the ImageNet database, were selected for the study. Transfer learning technique was implemented by utilizing these pre-trained models and fine-tuning them for the Ayur Bharat dataset, consisting of 10 classes of medicinal plants with a total of 10,000 images. The performance of the CNN models was enhanced using the Canny edge detection method as a pre-processing technique. A comparison was made between the three architectures, both with and without pre-processing techniques. The InceptionV3 architecture, trained with the Canny edge detection pre-processing technique, achieved the best classification performance for the Ayur Bharat dataset.

C.Amuthalingeswaran [3], A Deep Neural Network (DNN) model was built for the identification of medicinal plants. The model was trained using approximately 8,000 images belonging to four different classes of medicinal plants. The trained model achieved a good accuracy rate of 85% when tested with images taken from open field land areas. The model was developed specifically for the efficient classification of medicinal plants, with an emphasis on distinguishing between different plant disease classes. The model successfully achieved an accuracy percentage of 85.15% in classifying medicinal plants and their disease classes

The research [4] aims to develop a vision-based smart method for recognizing herb plants using deep learning (DL) models. Six medicinal plants, including betel, curry, tulsi, mint, neem, and Indian beech, were selected from the Kaggle database. 500 images were collected for each medicinal plant. The collected data underwent resizing and augmentation processes to increase the sample size and improve the robustness of the DL model. The MobileNet DL model was chosen for fully automatic identification of medicinal leaves. The DL model was trained, validated, and tested to assess its effectiveness. Evaluation measures such as accuracy, precision, and recall were used to evaluate the model's performance.

Chapter 4

PROJECT DESCRIPTION

Chapter 4 Project Description

This project aims in the creation of an automated system for the recognition of medicinal plants through the utilization of Convolutional Neural Networks (CNNs). Herbal remedies are also derived from plants and have been more frequently used than any other system of medicine today. This may be a simplified approach to its classification but it must be stressed that correct identification of these plants is very important for their successful deployment and protection. Nonetheless, the conventional approaches of plant identification are tedious, require much time, and require expert help and as such cannot be implemented widely, especially in those areas where there is a shortage of botanical information.

Utilizing further development of deep learning technique, this project aims at establishing a CNN model which is trained with a medicinal plant images dataset to facilitate the determination of the sort of plant that is being analyzed with efficiency, accuracy, and within the shortest time possible. The first step of the project is data acquisition that involved downloading numerous images of the different medicinal plants and storing them in directories with appropriate labels. Image pre-processing procedures include image scaling, in which the images are rescaled into equivalent dimensions before being analyzed, and pixel scaling, in which the pixel intensity is adjusted to that which training improves the model's performance.

The model design center is the CNN composed of several convolutional layers to extract the feature map, pooling layers to reduce the dimensionality, and the dropout layers to avoid overfitting. The model is built with a suitable optimizer and loss function, and trained by the prepared and pre-processed data. Even during the course of training, the model is trained to recognize features unique to each plant species and cumulatively augments the precision across a number of epochs.

Finally, the training is conducted and after that, there is model validation in which a validation dataset is used to check the accuracy and reliability of the model. The trained model is then published for future prediction. The last step is to consider putting in practice the chosen model, for instance, on the Internet as a part of a website or an API to accept new pictures of plants and return the probable species.

Through the use of this CNN based approach, the project combats the difficulties which are associated with the plant identification techniques and presents an enhanced approach that has ranked chances of success when compared to human knowledge, not to mention the fact that it is more efficient and less time consuming than manual identification. By using this automated identification system, botanist, researcher, and even the general public would have easily identified medicinal plants as well as enhance the use of the existing and exploited plants on traditional and advance medicine.

4.1 Requirements

4.1.1 Hardware Requirements:

i.Computer with Sufficient Processing Power:

A computer with a high-performance CPU, and preferably a GPU, is recommended to handle the computational load of training CNNs.

Minimum 16GB RAM for handling large datasets and ensuring smooth model training.

ii. Storage:

Adequate storage space to store the dataset of images and trained models.

4.1.2. Software Requirements:

i. Operating System:

Windows, macOS, or Linux. Ubuntu (Linux) is often preferred for better support with machine learning frameworks.

ii. Python language:

Python 3.7 or higher, preferably 3.10.

iii. Machine Learning Libraries:

TensorFlow: For building and training the CNN model.

Keras: Used as an API within TensorFlow for easier model creation.

NumPy: For numerical operations.

OpenCV: For image processing tasks.

Scikit-learn: For performance metrics such as classification reports and confusion matrices.

Matplotlib and Seaborn: For data visualization.

Streamlit: for creating a web application for deployment.

iv. Development Environment:

Jupyter Notebook or any Python IDE (like PyCharm or VSCode) for writing and testing the code.

VS code streamlit environment (optional for deployment)

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 128, 32)	896
conv2d_1 (Conv2D)	(None, 126, 126, 32)	9,248
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_2 (Conv2D)	(None, 63, 63, 64)	18,496
conv2d_3 (Conv2D)	(None, 61, 61, 64)	36,928
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_4 (Conv2D)	(None, 30, 30, 128)	73,856
conv2d_5 (Conv2D)	(None, 28, 28, 128)	147,584
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 128)	0
conv2d_6 (Conv2D)	(None, 14, 14, 256)	295,168
conv2d_7 (Conv2D)	(None, 12, 12, 256)	590,080
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 256)	0
conv2d_8 (Conv2D)	(None, 6, 6, 512)	1,180,160
conv2d_9 (Conv2D)	(None, 4, 4, 512)	2,359,808
max_pooling2d_4 (MaxPooling2D)	(None, 2, 2, 512)	0
dropout (Dropout)	(None, 2, 2, 512)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 1500)	3,073,500
dropout_1 (Dropout)	(None, 1500)	0
dense_1 (Dense)	(None, 80)	120,080

Total params: 7,905,804 (30.16 MB)
Trainable params: 7,905,804 (30.16 MB)
Non-trainable params: 0 (0.00 B)

Figure 4.1: Model Summary

CHAPTER 5

METHODOLOGY

CHAPTER 5 METHODOLOGY

FLOW CHART:

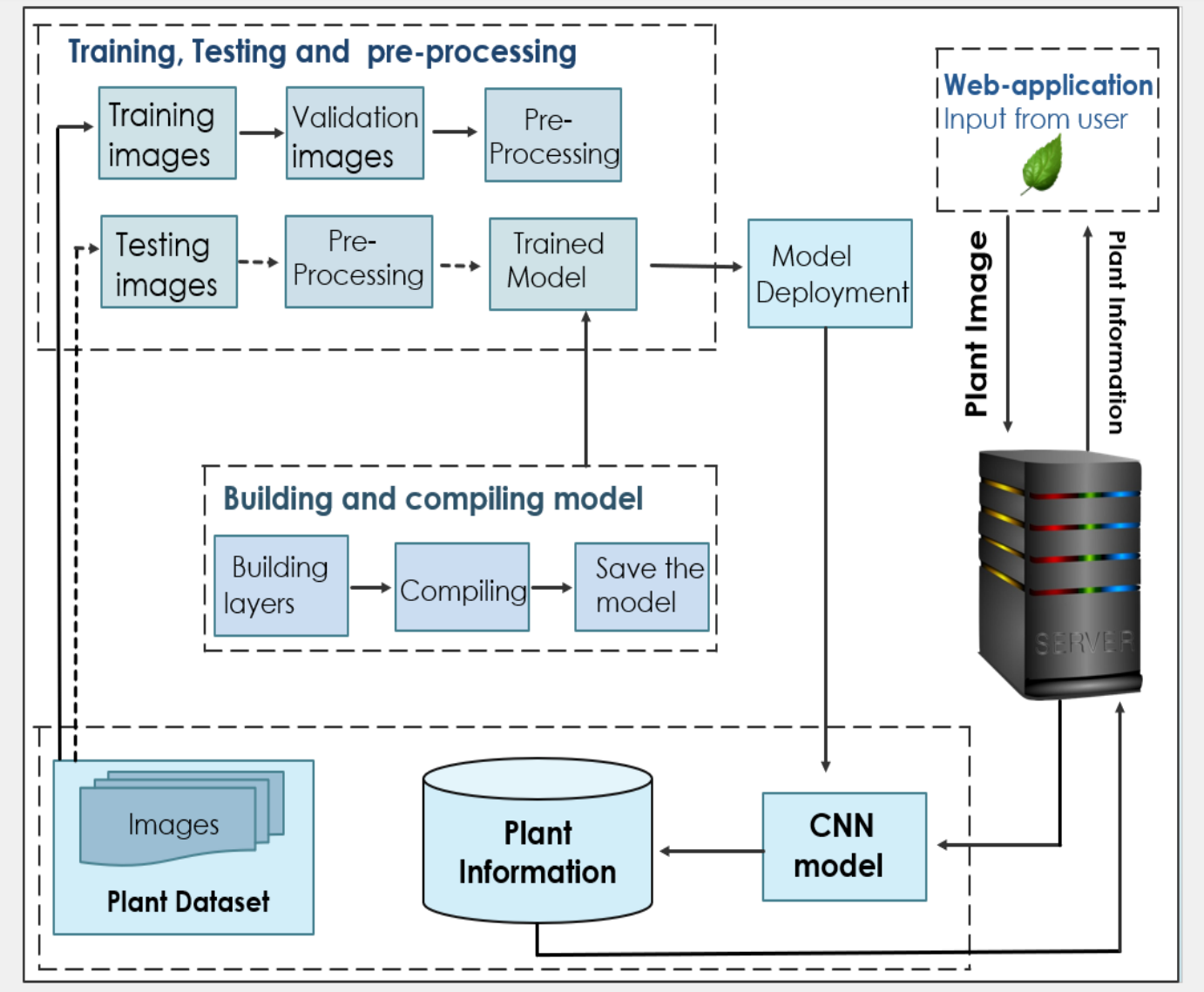

Figure 5.1: Flow Chart

The following are steps involved to develop a successful CNN model to identify and classify images: Data Collection, Preprocessing, Model training, Testing the model and deploy the model. All these key procedures necessary to construct the automated system for the identification of medicinal plants using CNNs are discussed in detail as follows:

1.Data Collection and Pre-Processing:

The initial procedure involves collection of images of different medicinal plants which are put into folder and sub-folder and labeled according to different species of the plants. These are then pre-processed into forms ready for training the CNN model using the given dataset. The images are then loaded using different functions in TensorFlow such as image_dataset_from_directory, then standardized into the 128*128 pixels then normalized. This helps to remove or transform some of the data inputs in a way that makes them more properly formatted and easily trainable, thus helping the model learn the desirable features faster and better.

```
training_set = tf.keras.utils.image_dataset_from_directory(
    'train',
    labels="inferred",
    label_mode="categorical",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True
)
```

Figure 5.2: Training Dataset

2.Model Building:

The primary part of the proposed approach is to build a CNN model. A sequential model using TensorFlow and Keras is built by convoluting layers that then extract feature maps of the images. Following these layers is the pooling layers in which the dimensions of the feature maps are made smaller in order to control the amount of overfitting and at the same time minimizing complexity. There are also dropout layers too in order to replenish further overfitting by determining each training input at random to zero out a given fraction of its units. The most recent layers of the model are the dense layers that determines the classification based on the features learned from the convolutional layers.

```
model.add(Conv2D(filters=64, kernel_size=3, padding='same', activation='relu'))
model.add(Conv2D(filters=64, kernel_size=3, activation='relu'))
model.add(MaxPool2D(pool_size=2, strides=2))
```

Figure 5.3: Part of the convolution layers

3.Model Compilation:

To set up the model after designing it, an optimizer (Adam), a loss function (categorical crossentropy), and accuracy as a metric are applied. This step optimizes the model for training purposes, where the algorithm is informed as to a plan or strategy on how it will be trained as well as the metrics by which its success will be evaluated.

```
model.compile(optimizer=tf.keras.optimizers.Adam(
    learning_rate=0.0001),loss='categorical_crossentropy',metrics=['accuracy'])
```

Figure 5.4: Compilation of model

4.Model Training:

The next thing to do is use the above pre-processed training data to train the model, and cross check its performance using a validation data set. The training process includes several epochs of training where the model gets exposed to input data and goes through several cycles of tuning its weights to give a minimal value to the loss function. The iterative process enable the model enhance the chances of its performance in recognizing medicinal plants in a real time by learning from a trained data set.

5.Model Evaluation:

When training is done the model is checked based on the validation set to make sure that the model has learned well enough to generalize to new data. This step involves evaluating of the model performance by means of such parameters as the validation loss and the accuracy of the given model, as well as the analysis of overfitting or underfitting of the model.

6.Model Saving and Deployment:

Once the models have achieved the desired performance, they can be stored in a file for use in future predictions. This saved model can then be deployed in a production environment. Images

are then passed through the model for prediction and the process is repeated as more images are processed from the saved model through deployment. This could be either through web-based applications, APIs, Mobile applications or through cloud-based services. The employed strategy enables users to input photos of plants with the possibility of getting the species identification in a short time effectively and easily.

```python
#Tensorflow Model Prediction
def model_prediction(test_image):
    model = tf.keras.models.load_model(r"C:\Users\nishu\OneDrive\Desktop\plant dataset\trained_model.keras")
    image = tf.keras.preprocessing.image.load_img(test_image,target_size=(128,128))
    input_arr = tf.keras.preprocessing.image.img_to_array(image)
    input_arr = np.array([input_arr]) #convert single image to batch
    predictions = model.predict(input_arr)
    return np.argmax(predictions) #return index of max element
```

Figure 5.5: Deploying the model

Thus, by using the presented detailed approach, the work accomplishes the proposed objective of using CNNs to develop a system for the automation of the process of identifying medicinal plants, which can solve the problems of traditional methods and meet the needs of botanist and general population.

CHAPTER 6

RESULT AND ANALYSIS

CHAPTER 6 RESULT AND ANALYSIS

The results of this project demonstrate the effectiveness of using Convolutional Neural Networks (CNNs) for the automated identification of medicinal plants. The evaluation metrics and visualizations confirm that the model has learned to accurately classify various species of medicinal plants from images. Below are the key results from the model training, evaluation, and testing phases.

6.1 Training and Validation Accuracy Result:

Training Accuracy: The model showed a steady increase in accuracy over the epochs, indicating that it was learning the distinguishing features of different plant species effectively. Training Accuracy: 93.8

Validation Accuracy: The validation accuracy also improved consistently, demonstrating that the model generalizes well to unseen data. Validation Accuracy: 98.6

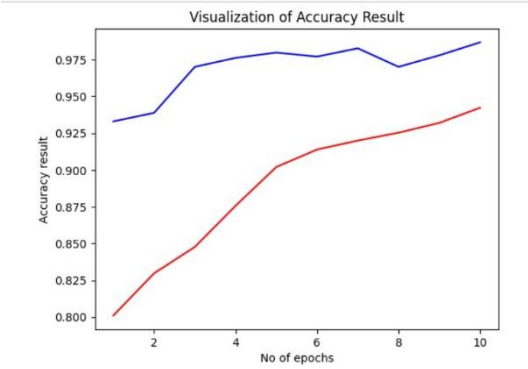

Figure 6.1: Training and Validation Accuracy Graph

6.2 Classification Report:

The classification report provides key metrics such as precision, recall, F1-score, and support for each class in your dataset. Precision is the ratio of correctly predicted positive observations to the total predicted positives. Recall is the ratio of correctly predicted positive observations to the all observations in actual class. The F1 score is the harmonic mean of precision and recall. It combines precision and recall into a single metric by giving each an equal weight.

```
print(classification_report(Y_true,predicted_categories,target_names = class_name))
```

	precision	recall	f1-score	support
Aloevera	1.00	1.00	1.00	118
Amla	0.97	1.00	0.99	67
Amruthaballi	1.00	0.99	0.99	90
Arali	1.00	1.00	1.00	89
Astma_weed	0.95	0.99	0.97	82
Badipala	1.00	0.97	0.99	76
Balloon_Vine	1.00	0.97	0.98	61
Bamboo	0.97	0.97	0.97	118
Beans	0.95	1.00	0.97	97
Betel	0.99	1.00	1.00	114
Bhrami	0.99	0.98	0.99	104
Bringaraja	0.96	1.00	0.98	73
Caricature	1.00	1.00	1.00	76
Castor	1.00	0.99	1.00	129
Catharanthus	0.99	0.97	0.98	134
Chakte	1.00	0.99	0.99	68
Chilly	1.00	1.00	1.00	69
Citron lime (herelikai)	0.91	1.00	0.95	99
Coffee	1.00	0.98	0.99	83
Common rue(naagdalli)	0.99	0.99	0.99	67
Coriender	0.99	0.98	0.99	115
Curry	0.99	0.97	0.98	168
Doddpathre	1.00	0.98	0.99	142
Drumstick	1.00	0.98	0.99	56
Ekka	1.00	1.00	1.00	81
Eucalyptus	0.95	1.00	0.98	80
Ganigale	0.99	1.00	0.99	75
Ganike	0.98	1.00	0.99	63
Gasagase	1.00	0.99	0.99	79
Ginger	1.00	0.99	0.99	82
Globe Amarnath	0.93	1.00	0.96	81
Guava	1.00	0.98	0.99	128
Henna	1.00	0.99	0.99	80
Hibiscus	1.00	0.94	0.97	118
Honge	0.98	1.00	0.99	113
Insulin	1.00	0.96	0.98	89
Jackfruit	1.00	0.97	0.99	110
Jasmine	1.00	0.98	0.99	49
Kambajala	1.00	1.00	1.00	59
Kasambruga	0.98	1.00	0.99	48
Kohlrabi	1.00	1.00	1.00	73
Lantana	1.00	0.97	0.99	76
Lemon	0.99	1.00	1.00	123
Lemongrass	1.00	1.00	1.00	8
Malabar_Nut	1.00	1.00	1.00	51
Malabar_Spinach	1.00	0.95	0.97	79
Mango	1.00	0.99	1.00	103
Marigold	1.00	1.00	1.00	93
Mint	1.00	0.96	0.98	135
Neem	1.00	0.98	0.99	132
Nelavembu	0.98	1.00	0.99	90
Nerale	1.00	0.92	0.96	62
Nooni	0.97	1.00	0.99	72
Onion	1.00	0.97	0.98	92
Padri	1.00	1.00	1.00	73
Palak(Spinach)	1.00	1.00	1.00	149
Papaya	1.00	0.99	1.00	135
Parijatha	1.00	1.00	1.00	66
Pea	1.00	1.00	1.00	47
Pepper	1.00	0.88	0.93	8
Pomoegranate	1.00	1.00	1.00	75
Pumpkin	0.99	0.99	0.99	92
Raddish	1.00	1.00	1.00	40
Rose	1.00	0.98	0.99	106
Sampige	1.00	1.00	1.00	61
Sapota	0.98	0.98	0.98	44
Seethaashoka	0.98	1.00	0.99	47
Seethapala	0.92	1.00	0.96	114
Spinach1	0.99	1.00	0.99	67
Tamarind	1.00	0.98	0.99	176
Taro	1.00	0.97	0.99	69
Tecoma	0.97	1.00	0.99	69
Thumbe	0.95	1.00	0.97	74
Tomato	1.00	1.00	1.00	62
Tulsi	0.96	0.99	0.97	177
Turmeric	0.97	1.00	0.99	39
ashoka	1.00	1.00	1.00	81
camphor	1.00	0.98	0.99	66
kamakasturi	0.88	1.00	0.94	67
kepala	0.99	1.00	0.99	76
accuracy			0.99	6899
macro avg	0.99	0.99	0.99	6899
weighted avg	0.99	0.99	0.99	6899

Figure 6.2: Classification Report: Precision, Recall, F1 score

6.3 Confusion Matrix:

The confusion matrix provides a visual representation of the model's performance by showing the number of correct and incorrect predictions for each class. The confusion matrix indicates that most predictions are correct, with few misclassifications, demonstrating the model's high accuracy.

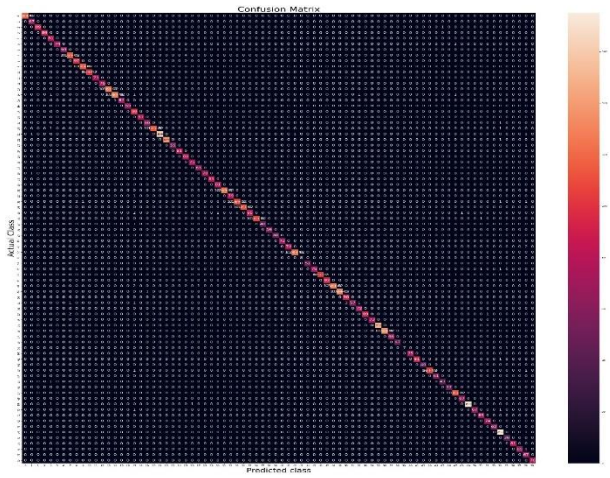

Figure 6.3: Confusion Matrix

6.4 Deployment Testing:

Finally, the model was deployed and tested with images to ensure it performs well. An example image was used to verify the model's prediction capabilities shown below:

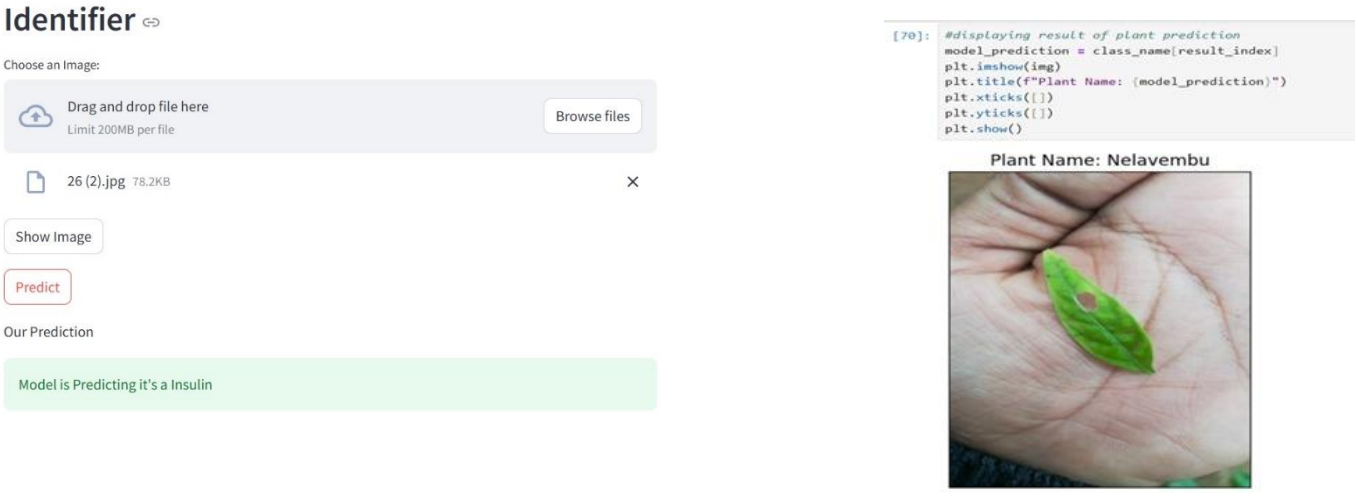

Figure 6.4: Model Prediction Outputs

CONCLUSION

In conclusion, we have successfully developed a CNN model to identify medicinal plants from images. Our model demonstrates good accuracy and can be used for practical applications in various fields. This project demonstrates the power of CNNs in automating the identification of medicinal plants, offering a scalable and reliable alternative to traditional methods. The developed system can significantly enhance the accessibility and accuracy of plant identification, contributing to both traditional medicine and modern pharmacological research. This automated approach not only saves time and effort but also democratizes access to botanical expertise, making it a valuable tool for various stakeholders. Future work could involve expanding the dataset, improving the model architecture, and deploying the model in real-world scenarios.

REFERENCES

[1] M.R. Dileep; P.N. Pournami," **AyurLeaf: A Deep Learning Approach for Classification of Medicinal Plants**", TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), doi: 10.1109/TENCON.2019.8929394

[2] Arunaggiri Pandian K; Sai Kumar T S, **"Identification of Indian Medicinal Plants from Leaves using Transfer Learning Approach",** 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), **DOI:** 10.1109/ICOEI51242.2021.9452917

[3] C.Amuthalingeswaran, Mr. M.Sivakumar , Dr. P.Renuga, S.Alexpandi, S.Santhana Hari, "**IDENTIFICATION OF MEDICINAL PLANT'S AND THEIR USAGE BY USING DEEP LEARNING**" Proceedings of the Third International Conference on Trends in Electronics and Informatics (ICOEI 2019) IEEE Xplore Part Number: CFP19J32-ART; ISBN: 978-1-5386-9439-8

[4] S. Kavitha, T. Satish Kumar, E. Naresh, Vijay H. Kalmani, Kalyan Devappa Bamane & Piyush Kumar Pareek," **Medicinal Plant Identification in Real-Time Using Deep Learning",** Published on :07 December 2023,volume(5) article number 73(2024).

APPENDIX:

Importing Libraries

```
import tensorflow as tf
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

Data preprocessing

Training Image Preprocessing

```
training_set = tf.keras.utils.image_dataset_from_directory(
    'train',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False,
)
validation_set = tf.keras.utils.image_dataset_from_directory(
    'valid',
    labels="inferred",
    label_mode="categorical",
class_names=None,
```

```
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False,
)
for x,y in training_set:
    print(x,x.shape)
    print(y,y.shape)
    break
```

Building Model

```
from tensorflow.keras.layers import Dense,Conv2D,MaxPool2D,Flatten,Dropout
from tensorflow.keras.models import Sequential
model = Sequential( )
```

Building Convolution Layer

```
model.add(Conv2D(filters=32,kernel_size=3,padding='same',activation='relu',input_shape=[128,128,3]))
model.add(Conv2D(filters=32,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))

model.add(Conv2D(filters=64,kernel_size=3,padding='same',activation='relu'))

model.add(Conv2D(filters=64,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
```

```
model.add(Conv2D(filters=128,kernel_size=3,padding='same',activation='relu'))
model.add(Conv2D(filters=128,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))

model.add(Conv2D(filters=256,kernel_size=3,padding='same',activation='relu'))
model.add(Conv2D(filters=256,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))

model.add(Conv2D(filters=512,kernel_size=3,padding='same',activation='relu'))
model.add(Conv2D(filters=512,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))

model.add(Dropout(0.25)) #to avoid overfitting

model.add(Flatten())

model.add(Dense(units=1500,activation='relu'))

model.add(Dropout(0.4))

model.add(Dense(units=80,activation='softmax'))
```

Compiling the Model

```
model.compile(optimizer=tf.keras.optimizers.Adam(
        learning_rate=0.0001),loss='categorical_crossentropy',metrics=['accuracy'])
model.summary( )
```

Model Training

```
training_history = model.fit(x=training_set,validation_data=validation_set,epochs=10)
```

Model Evaluation

```
train_loss,train_acc = model.evaluate(training_set)
print(train_loss,train_acc)

val_loss,val_acc = model.evaluate(validation_set)
print(val_loss,val_acc)
```

Saving the model

```
model.save("trained_model.keras")

#Recording history in json
import json
with open("training_hist.json","w") as f:
    json.dump(training_history.history,f)
```

Accuracy Visulization

```
training_history.history['accuracy']

training_history.history['val_accuracy']

epochs = [i for i in range(1,11)]
epochs
plt.plot(epochs,training_history.history['accuracy'],color='red',label='Training Accuracy')
plt.plot(epochs,training_history.history['val_accuracy'],color='blue',label='Validation Accuracy')
plt.xlabel("No of epochs")
plt.ylabel("Accuracy result")
plt. title("Visualization of Accuracy Result")
plt.show()
```

Some other metrics

```
class_name = validation_set.class_names
class_name
test_set = tf.keras.utils.image_dataset_from_directory(
    'valid',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=False,
    seed=None,

validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False
)
```

```
y_pred = model.predict(test_set)
y_pred,y_pred.shape

predicted_categories = tf.argmax(y_pred,axis=1)
predicted_categories
true_categories = tf.concat([y for x,y in test_set],axis=0)
true_categories
Y_true = tf.argmax(true_categories,axis=1)
Y_true
```

Classification metrix

```
from sklearn.metrics import classification_report
print(classification_report(Y_true,predicted_categories,target_names = class_name))
```

Confusion Matrix

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_true,predicted_categories)
cm.shape
```

Visualization of confusion matrix

```
plt.figure(figsize=(40,40))
sns.heatmap(cm,annot=True,annot_kws={'size':16})
plt.xlabel('Predicted class',fontsize=30)
plt.ylabel('Actual Class',fontsize=30)
plt.title('Confusion Matrix',fontsize=34)
plt.show( )
```

PART - II

LOADING AND TESTING THE MODEL

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
```

Load Model

```
model = tf.keras.models.load_model('trained_model.keras')
model.summary()
```

Visualizing single test image

```
import cv2
import numpy as np
from PIL import Image
# Load the image
img = Image.open("test/1556.jpg")
# Convert the image data to a numpy array
img_array = np.array(img)
# Convert the image data to float
img_float = img_array.astype(float)

image_path = "test/1556.jpg"
#reading image
```

```
img = cv2.imread(image_path)
#to get actual image color [Convert bgr to rgb]

img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
#displaying image
plt.imshow(img)
plt.title("Test Image")
plt.xticks([])
plt.yticks([])
plt.show()
```

Testing the Model

```
image = tf.keras.preprocessing.image.load_img(image_path,target_size=(128,128))
input_arr = tf.keras.preprocessing.image.img_to_array(image)
input_arr = np.array([input_arr]) #convert single image to batch
print(input_arr.shape)

prediction = model.predict(input_arr)
prediction,prediction.shape

result_index = np.argmax(prediction)
result_index

class_name = [‘ # Write/Paste all your class names #’]

#displaying result of plant prediction
model_prediction = class_name[result_index]
plt.imshow(img)
plt.title(f"Plant Name: {model_prediction}")
plt.xticks([])
plt.yticks([])
plt.show( )
```

DEPLOYING THE MODEL AND DEVELOPING WEB APPLICATION USING STREMLIT

```
import streamlit as st
import tensorflow as tf
from tensorflow.keras.models import load_model
import numpy as np
from PIL import Image
import cv2

#Tensorflow Model Prediction
def model_prediction(test_image):
    model = tf.keras.models.load_model(r"C:\Users\nishu\OneDrive\Desktop\plant
dataset\trained_model.keras")
    image = tf.keras.preprocessing.image.load_img(test_image,target_size=(128,128))
    input_arr = tf.keras.preprocessing.image.img_to_array(image)
    input_arr = np.array([input_arr]) #convert single image to batch
    predictions = model.predict(input_arr)
    return np.argmax(predictions) #return index of max element
```

Web Application

```
#Sidebar
```

```
st.sidebar.title("Dashboard")
app_mode = st.sidebar.selectbox("Select Page",["Home","About","Plant Identifier"])

#homepage
if(app_mode=="Home"):
    st.header("**MEDICINAL PLANT IDENTIFIER**")
    img = Image.open("home_pg.jpeg")
    st.image(img,width=500,channels='RGB')
    st.markdown("""
    Welcome to Medicinal Plant Identifier System! 🌿🔍
    Our mission is to help in identifying medicinal plant efficiently. Just Upload an image of a plant, and our system will identify what kind of medicinal plant it is.!
    ### How It Works
    1. **Upload Image:** Go to the **Plant Identifier** page and upload an image of a plant you want to identify.
    2. **Analysis:** Our system will process the image using algorithms to identify the possible plant.
    3. **Results:** View the results and recommendations of the plant.

    ### Why Choose Us?
    - **Accuracy:** Our system utilizes machine learning techniques for accurate Identification.
    - **User-Friendly:** Simple and intuitive interface for seamless user experience.
    - **Fast and Efficient:** Receive results in seconds, allowing for quick decision-making.

    ### Get Started
    Click on the **Plant Identifier** page in the sidebar to upload an image and know different kinds of medicinal plants.!

    ### About Us
    Learn more about the project, our team, and our goals on the **About** page.
    """)

#About Project
elif(app_mode=="About"):
    st.header("About")
    st.markdown("""
                ### Our Team
                "Hello and welcome to Plant Identifier!
                We are students from the AI-ML department who have create an application that can recognize medicinal plants from images.
                Meet our awesome team: """)

    img1 = Image.open("bhav.jpeg")
    st.image(img1,width=180,caption = 'kasala Bhavana', channels='RGB')
    img2 = Image.open("lak.jpeg")
    st.image(img2,width=250,caption = 'Lakshya U Reddy', channels='RGB')
    img3 = Image.open("nish.jpeg")
    st.image(img3,width=220,caption = 'Nishat N Shahu', channels='RGB')
    img4 = Image.open("tan.jpeg")
   st.image(img4,width=310,caption = 'Tanya Gopal', channels='RGB')


    st.markdown("""We hope you enjoy using our application and learning about different plants. Thank you for checking out Plant Identifier!""")
    st.markdown("""
```

```
            #### About Dataset
            This dataset is recreated using offline augmentation from the original dataset.The original dataset can be found from ncbi library.
            This dataset consists of about 6K plus rgb images of medicinal leaves which is categorized into 80 different classes.The total dataset is divided into training and validation set preserving the directory structure.
            A new directory containing 2K plus test images is created later for prediction purpose.
            #### Content
            1. train (6899 images)
            2. test (2068 images)
            3. validation (6899 images)
            """)

#Prediction Page
elif(app_mode=="Plant Identifier"):
    st.header("Identifier")
    test_image = st.file_uploader("Choose an Image:")
    if(st.button("Show Image")):
        st.image(test_image,use_column_width=True)
    #Predict button
    if(st.button("Predict")):
        st.write("Our Prediction")
        result_index = model_prediction(test_image)
        #Reading Labels
        class_name =  [' # Name all your classes # ‘]
        st.success("Model is Predicting it's a {}".format(class_name[result_index]))
```