Plants Species Identification for Medicinal Plants Using Convolutional Neural Networks

Pragati Mahale [1], Aditya Shiledar [2], Sumedh Chavan [3], Shashwat Kale[4]
[1,2,3,4]All India Shri Shivji Memorial Society's Institute of Information Technology, Pune 411001, India

doi.org/10.64643/IJIRTV12I4-181074-229

Abstract— **Accurate identification of medicinal plant species is crucial in medicine, botany, pharmacology, and conservation biology. Identification through conventional methods largely relies on human judgment and visual inspection, which are time-consuming, labor-intensive, and prone to human error, especially when determining between morphologically similar species. To circumvent such limitations, the present research proposes a fast and automatic system for plant species identification based on Convolutional Neural Networks (CNNs). The model leverages the power of deep learning to extract distinctive features of plant leaf images with respect to shape, texture, and vein pattern for proper classification. A comprehensive dataset of high-resolution images of medicinal plants was used, with multiple light conditions and environmental changes to improve model generalization. Data augmentation techniques such as flipping, rotation, and color shifting were used to improve dataset robustness. Transfer learning and ensemble learning were utilized to reinforce the model further to improve accuracy and avoid overfitting. Attention mechanisms were also utilized to allow the network to focus on the informative regions of each image. The proposed CNN-based model yielded better classification performance, and performance metrics such as accuracy, precision, recall, and F1-score confirmed its applicability in practical situations. The model's performance was also compared to existing algorithms such as Faster R-CNN, SSD, and hybrid CNN-Random Forest models. The results affirm the applicability of the model in real-time applications, which include mobile plant identification apps and biodiversity monitoring. The research helps improve the classification of medicinal plants, promotes conservation, and streamlines the cultivation of sustainable herbal medicine systems.**

Index Terms— **Medicinal Plant Identification, Convolutional Neural Networks (CNN), Deep Learning, Image Classification, Plant Leaf Recognition, Data Augmentation, Transfer Learning.**

I. INTRODUCTION

Medicinal plants have served as the foundation of traditional medicine for thousands of years and continue to be integral to modern medicine, pharmacology, and conservation of biodiversity. They contain bioactive molecules that are essential for the identification of natural and synthetic drugs. Accurate identification of these plants is therefore imperative to avoid misuse, fatal misidentification, and enable conservation. However, with an estimated 420,000 flower plant species across the world—many of which share similar morphological features—identification remains a time-consuming, error-prone process requiring specialized taxonomic skills.

Conventional identification techniques usually involve morphological inspection by the naked eye of parameters such as leaf shape, venation type, flower orientation, and texture of the stem. Though good enough for masters, these techniques are not practical for mass or real-time application, especially where botanical proficiency may be scarce. Added to this is the morphological similarity between species and other environmental factors such as light, background, and growth stage.

The recent advances in artificial intelligence (AI) and computer vision, in specific, have made it possible to have automated recognition systems for plant species. Among them, the Convolutional Neural Networks (CNNs) have proven to be an effective deep learning framework having the capability to learn hierarchical features using image data. CNNs are best suited to handle light, angle, and texture variations, making them a suitable option for plant species classification through leaf images.

This research focuses on the development of a CNN-based model for automating medicinal plant recognition from images of leaves. The system, using data augmentation, transfer learning, and attention mechanisms, attempts to provide high accuracy, efficiency, and robustness across changing environmental conditions. The ultimate goal is to provide a scalable and dependable solution that can be applied in real-time applications, such as mobile applications, to help researchers, herbalists, farmers, and conservationists identify plant species correctly.

II. SYSTEM ARCHITECTURE

This section describes the systematic approach used in the design, development, and testing of a miniature obstacle detection and automatic stopping system for trains. The methodology includes hardware setup, software development, sensor calibration, and motor control implementation. Architecture for classifying medicinal plant species using Convolutional Neural Networks (CNN) is deployed to provide an end-to-end pipeline that takes an input dataset of plant images and provides an output predicted plant species. Automation and ease of the process are provided, even for experts, allowing them to classify plant species accurately and at high velocity.

As can be observed in the architecture diagram, the system follows a modular process from the input of the user to the output of the result. The core component of the system depends on the CNN model, which performs image preprocessing, feature extraction, training, and classification. The dataset is divided into the training set and the testing set (80%-20% split), which ensures the model gets effectively trained and tested. Each step is vital to high precision and reliability in different ambient environments.

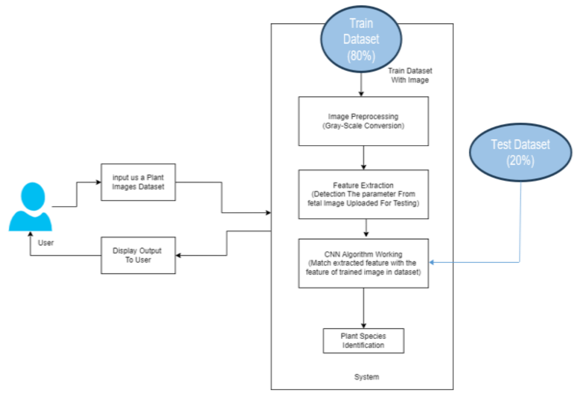

Train Dataset (80%)
Train Dataset With Image
Image Preprocessing (Gray-Scale Conversion)
Test Dataset (20%)
Feature Extraction (Detection The parameter From fetal Image Uploaded For Testing)
input us a Plant Images Dataset
User
Display Output To User
CNN Algorithm Working (Match extracted feature with the feature of trained image in dataset)
Plant Species Identification
System

Fig 1. System Architecture of Model

There are two streams in the architecture illustration:

User Interaction Layer, where users input plant images for identification. Processing and Classification Layer, where the critical actions such as image preprocessing, feature extraction, model training, and prediction occur.

In order to achieve correct model learning as well as model estimation, the dataset is divided into two sets:

Training Dataset (80%) – Trains the CNN model.

Testing Dataset (20%) – Tests model performance and generalization.

Each component in this architecture is dedicated to enhancing the accuracy, efficiency, and practical usability in real-world applications. A step-by-step discussion about each module.

2.1.1 User Input Module

Description: This module provides the end user with a facility to enter a dataset of plant images, specifically leaf images. These images are uploaded either in bulk or one by one based on the application interface.

Objective: To give the system visual information for various medicinal plant species.

2.1.2 Dataset Splitting

Training Dataset (80%):

Made up of labeled plant images for training the CNN model. It contains a diverse set of images with various lighting conditions, backgrounds, and growth stages.

Testing Dataset (20%):

Comprises unseen images to test the trained model's accuracy and generalization ability. It serves to verify performance based on measures like precision, recall, F1-score, and confusion matrix.

Purpose: Checks that the model is not only memorizing the training data but also does well on new, unseen data—preventing overfitting.

2.1.3 CNN Algorithm Operationalization

Training Phase:

The CNN is trained to identify patterns and features of images from labeled data. Methods such as transfer learning (e.g., using an existing pre-trained model like ResNet or VGG) can be utilized to enhance performance and shorten training time.

Testing Phase:

The model uses learned weights to make predictions from the test dataset. It compares the features extracted from the test images with the training data features.

2.2 Data Flow Diagrams/ER Diagrams:

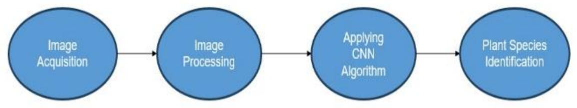


Fig.2 Data flow diagram for Plant Species Detection

This research proposes an enhanced methodology for the accurate identification of medicinal plant species utilizing deep learning techniques. The proposed approach incorporates several advancements over existing methods.

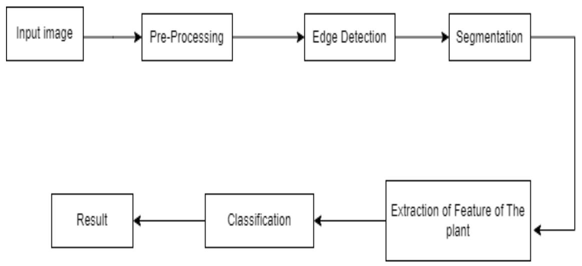


Fig.3 Work flow diagram for Plant Species Detection

The diagram illustrates the workflow for plant species identification using image processing techniques. It begins with an input image of the plant, which is subjected to pre-processing to enhance its quality, such as resizing or noise reduction. The next step is edge detection, where the system identifies the plant's structural outlines like leaves or stems. This is followed by segmentation, which isolates the plant from the background, ensuring that only relevant features are analyzed. Once segmented, the system proceeds to feature extraction, capturing essential attributes like shape, texture, and color. These features are then passed through a classification stage, where the plant is matched to a known species based on the extracted data. Finally, the result is displayed, identifying the plant species.

2.3 UML Design and Documentation

Unified Modeling Language (UML) serves as a standardized and visual modeling language for the analysis, design, and documentation of software systems and processes. It provides a common language that allows software engineers, designers, and stakeholders to understand, communicate, and visualize the various components and interactions within a system. UML offers a diverse set of diagrams, including use case, class, sequence, activity, and state diagrams, among others, each tailored to represent different aspects of a system. These diagrams aid in structuring, planning, and documenting software designs, enabling a clearer understanding of system architecture and functionalities. They serve as a crucial tool in the software development life cycle, facilitating communication between different teams.

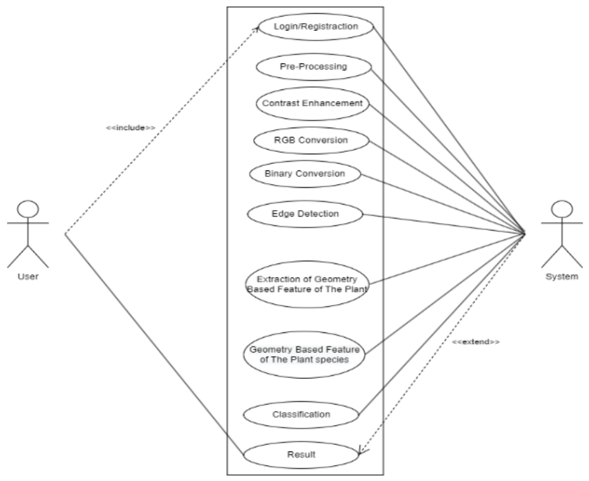


Fig 4 Use case diagram for plant species identification

2.4 System Thinking

The proposed medicinal plant identification system has a modular and integrated design philosophy where every module is a part of the overall workflow-based image classification logic. Rather than treating one operation such as preprocessing, feature extraction, or classification as an independent module, the system is designed as an integrated pipeline of dependent subsystems that are maximized for recurring real-time identification performance. Image acquisition and preprocessing form the foundation level of the system. Plant high-resolution images are acquired from diverse sources and standardized with regard to resizing, normalization, and optional conversion to grayscale. These processes ensure consistency within input data and enhance the quality of features learned during classification. The standardized input is subsequently fed to the CNN core structure, which automatically learns and identifies complex patterns such as leaf morphology, venation, texture, and shape through successive convolutional layers.The smart brain of the system is the learned CNN model. On receiving an input image, it processes the data through a series of convolutional and pooling operations, followed by fully connected operations. The softmax output layer gives a probabilistic output for a given plant species using the respective probability. This end-to-end deep learning pipeline eliminates the requirement of feature engineering by hand and enhances identification accuracy even under the condition of changing environmental factors such as varying light, background, or orientation of leaves.

For enabling real-time application, the system can be installed on field-deployable devices or mobile applications. This CNN can be integrated with smartphone cameras or field-deployable IoT devices to identify on-site plant species. GPS-tagging over this allows localization of the prediction event, and cloud integration in an optional way allows storage of identification history for ecological monitoring or agricultural guidance.

III. RESULTS

3.1 System Implementations

3.1.1 Implemented Libraries/Packages:

• Numpy

NumPy (Numerical Python) is a fundamental scientific computing library used to provide fast operations for large, multi-dimensional arrays and matrices. NumPy is applied in the plant species identification project to transform plant images into numerical arrays, normalize pixel values, and facilitate matrix operations while performing image preprocessing and CNN computation. It also acts as the underlying structure for data handling during feature extraction and model training.

• TensorFlow

TensorFlow is an open-source deep learning software developed by Google to construct, train, and deploy machine learning models. In this project, TensorFlow drives the model construction and training of the Convolutional Neural Network(CNN) to classify species. TensorFlow performs operations such as forward propagation.

• Skit-Learn

Scikit-learn is an extensive machine learning library containing functions for data splitting, model assessment, and performance measures. Scikit-learn is utilized in the project to divide the dataset into train and test sets, compute performance measures such as accuracy, precision, recall, and F1-score, and create confusion matrices. It also provides cross-validation to make sure the CNN model can generalize over unseen plant species.

• Keras

Keras is a deep learning neural network API implemented on top of TensorFlow for efficient prototyping and modular structure. In this project, Keras facilitates the definition of CNN structure in terms of layers, i.e., Conv2D, Max-Pooling, Dropout, and Dense. It also makes model compilation, training, and augmentation.

• Pandas

Pandas is a library of data manipulation and analysis that provides data structures such as Series and DataFrames. In the project, it is utilized to handle image labels, read and write classification outcomes. logs in order, and combine plant image data with metadata.

• MatPlotLib

Matplotlib is a general-purpose 2D plotting library for visualizing data and results. In this project, it is utilized to plot training and validation accuracy/loss curves, show the confusion matrix, and visualize sample predictions of plant images and their predicted and true labels. This helps with understanding model performance and communicating results well.

3.1.2 Implemented Functions:

1.Sequential()

Builds a linear stack of layers to build the CNN model..

2.classifier.add()

Adds a layer (for example, Convolution2D, MaxPooling2D, Flatten, Dense, Dropout) onto the CNN model in sequential order.

3.Convolution2D(filters=filters, kernel_size=kernel_size, subsample=kernel_size, input_shape=input_shape, activation=activation)

Adds a 2D convolutional layer with the specified number of filters and kernel size; enables the function. The first layer also requires the input shape.

4.MaxPooling2D(pool_size=(2,2))

Adds a max pooling layer which downsizes the feature maps by taking the maximum over a 2x2 window.

5.Flatten()

Maps 3D feature maps to a 1D feature vector that can be inputted into fully connected layers.

6.Dense(units, activation)

Adds a densely (fully connected) connected layer with units of the specified number and activation function.

7.Dropout(rate)

Includes dropout regularization that disables randomly some of the neurons during training to prevent overfitting.

8.Classifier.compile(optimizer=optimizer, loss=loss, metrics=metrics)

Tunes the model during training by setting optimizer, loss function, and metrics to monitor.

9.ImageDataGenerator(rescale=rescale, shear_range=shear_range, zoom_range=zoom_range, horizontal_flip=horizontal_flip)

3.1.3 Implemented Algorithms:

Step 1: Loading and Preprocessing the Data

The dataset of images is loaded with the flow_from_directory function, which automatically annotates images according to their directory structure. Real-time data augmentation is performed on training images to improve generalization and avoid overfitting. These involve operations such as rescaling pixel values, applying random shear transformations, zoom, and horizontal flipping. Test images are also rescaled to normalize pixel values between 0 and 1, ensuring dataset consistency.

Step 2: CNN Model Architecture Design

A sequential CNN model is built based on a stack of convolutional layers. ReLU activation functions are employed along with small 1x1 filters to learn hierarchical visual features from the input images. Every convolutional layer has a max pooling layer following it to decrease the spatial sizes of the feature maps and prevent overfitting. The resultant pooled feature maps are flattened in order to transform the 3D tensor into a 1D feature vector for input to the dense layers.

Step 3: Fully Connected Layers

Following flattening, the model has a dense layer with 256 neurons and ReLU activation for extracting intricate patterns from the features. In order to avoid overfitting, a dropout layer with dropout rate 50% is used. An output layer with 16 neurons and soft-max activation for multi-class classification where each neuron represents a unique plant species is then added.

Step 4: Build the Model

The CNN model is built with the Stochastic Gradient Descent (SGD) optimizer and learning rate 0.01 to fine-tune the learning process. Categorical cross entropy is chosen as the loss function, which is appropriate for problems involving multi-class classification. Accuracy is chosen as the main performance indicator to track the model during training as well as validation.

Step 5: Training the CNN

The model is learned by batches of training images that have been augmented, and performance is also tested on an independent test dataset at each epoch. The model is trained for 20 epochs with step numbers per epoch and validation steps to allow complete learning. The iterative training process helps optimize the weights of the model and enhance accuracy over time.

Step 6: Model Evaluation

Finally, after training, the performance of the model is checked on the test dataset to analyze how well it can generalize. The model is also tested on the training set to verify if the model is underfitting or overfitting by comparing its performance on training and test data.

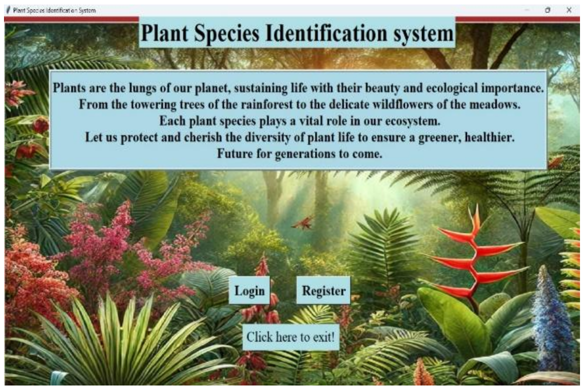
Step 7: Saving the Model

The trained CNN model is then saved on disk in the .h5 format, allowing reuse or deployment without the requirement to retrain. The serialized state keeps the architecture, weights, and the state of the optimizer.

Step 8: Visualization

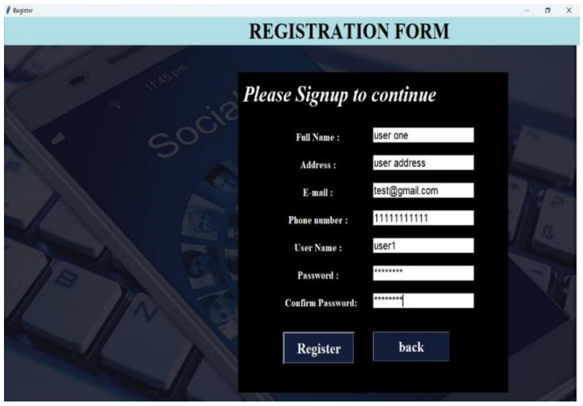
The training and validation accuracy and loss values are plotted vs. epochs to visualize the learning process. The plots assist in model performance trend diagnosis, e.g., convergence, overfitting, or underfitting. The created plots are exported as image files for subsequent analysis or reporting.

3.1.4 Graphics User Interface Screenshots:



Plants are the lungs of our planet, sustaining life with their beauty and ecological importance.
From the towering trees of the rainforest to the delicate wildflowers of the meadows.
Each plant species plays a vital role in our ecosystem.
Let us protect and cherish the diversity of plant life to ensure a greener, healthier.
Future for generations to come.

Fig 6. Home Page



Full Name :	user one
Address :	user address
E-mail :	test@gmail.com
Phone number :	11111111111
User Name :	user1
Password :	********
Confirm Password:	********

Fig 7. Registration page

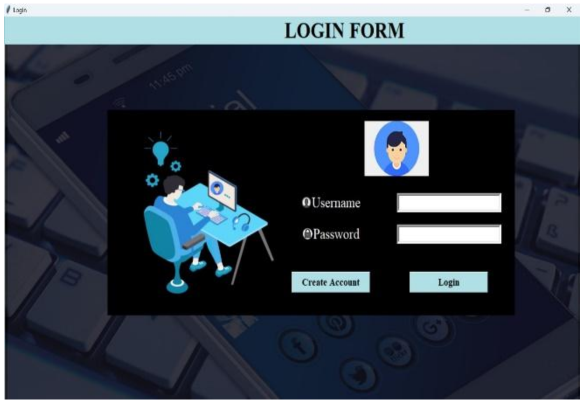


Fig 8. Login Page

The Plant Species Identification System starts with an aesthetically appealing home page that emphasizes the significance of plants in our environment. The home page consists of a forest background comprising lush green plants with a motivational quote urging protection and preservation of the plant diversity. There are three buttons: Login, Register, and Exit, leading the user to continue with the application.

If the user is new, he/she clicks on the Register button, which redirects to the Registration Form. There, they are asked to enter their personal details such as full name, address, email ID, phone number, preferred username, and password. The form has a compact design with a dark background and light blue highlights to make it easier to read and maintain focus. When the information is filled in, the user can click on Register to send the form or Back to go back to the home screen.

On registration, the user proceeds to the Login Form, where they need to enter their username and password. The login page features a friendly interface with icons for the input fields and a simple design. If the user is not yet registered, they can click on Create Account in order to go to the registration form. Once logged in successfully, the user is taken to the main functional interface of the system where they can upload a picture of a plant leaf and the trained CNN model would predict the species with special emphasis on medicinal plants.

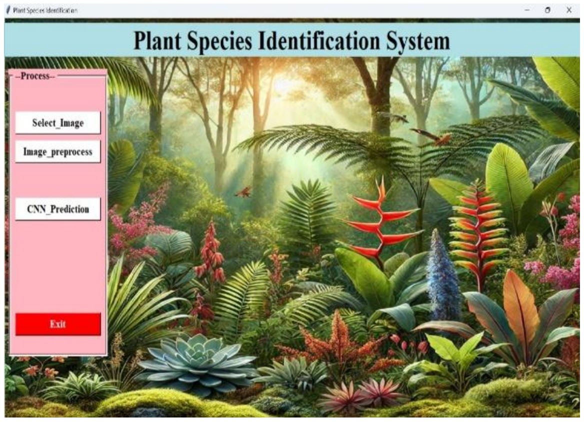


Fig 9. Species Identification module

The interface presented here is the graphical user interface (GUI) of a Plant Species Identification System, developed with Python's Tkinter library. It is made to help users identify plant species via an easy three-step approach with image selection, preprocessing, and prediction via a Convolutional Neural Network (CNN).

A prominent, easily readable heading labeled Plant Species Identification System centered on a light blue background at the interface top provides users instant context regarding the purpose of the application. The interface has a high-quality picture of a dense forest as a background, which makes it more visually appealing and in line with the theme of plant identification.

On the left-hand side of the screen, there is a pink background panel named "Process--", that helps the user through the application's main functionality. The first of these buttons is "Select_Image", to which the user can browse and select a picture of a plant from their device. After choosing an image, the user may press the "Image_preprocess" button, which will preprocess the chosen image by applying necessary operations like resizing and normalization to get it ready for analysis. tHE button invokes the pre-trained CNN model to examine the preprocessed image and determine the plant species. There is also an "Exit" button located at the bottom of the panel that allows the user to close down the application. In total, this GUI is a simple and graphically interesting platform for identification of plant species that allows users to interface with a deep learning model in an intuitive and accessible way.

IV. SYSTEM TESTING

The following series of screenshots display the graphical user interface (GUI) of a "Plant Species Identification System" and illustrate the end-to-end process of system testing for disease identification in plant leaves. The interface is developed using a pleasantly designed interface with a natural background, matching the botanical subject matter of the application.

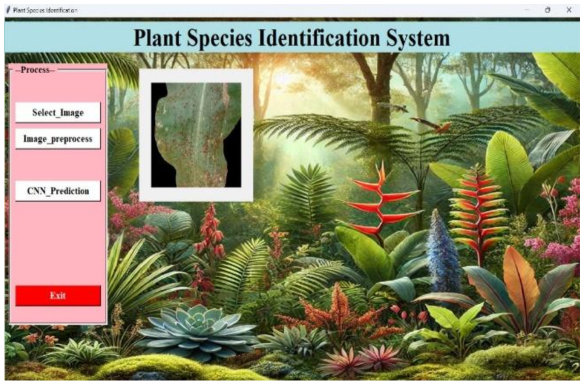


Fig 10. Image selection

In the initial screenshot, the user interface presents a title banner named "Plant Species Identification System," and a sidebar named "--Process--" consisting of buttons named "Select Image," "Image Preprocess," "CNN Prediction," and "Exit." At this point, a plant leaf photo (with symptoms of disease) is uploaded using the "Select Image" button. This photo is used as the input to the identification system.

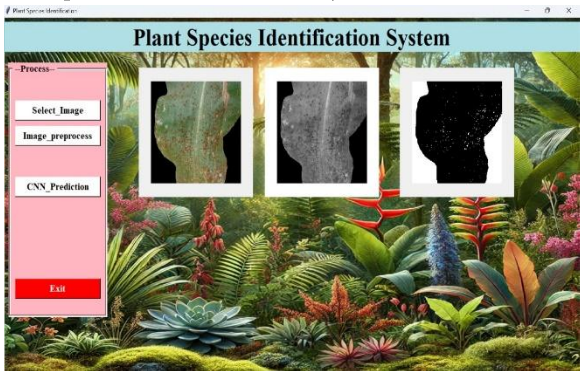


Fig 11. Image preprocessing

The second screenshot demonstrates the application of the "Image Preprocess" function. In this case, three images appear side by side: the source image, a grayscale transformed image, and a binary segmented image. This preprocessing is necessary to heighten the feature extraction process so that the CNN (Convolutional Neural Network) model can give attention to salient patterns and textures that play a role in precise disease prediction.

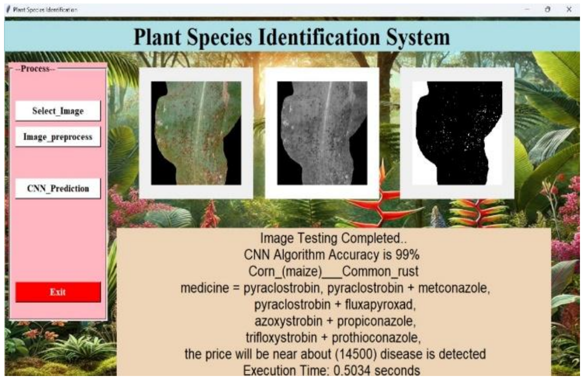

Plant Species Identification
Plant Species Identification System
--Process--
Select_Image
Image_preprocess
CNN_Prediction
Exit

Image Testing Completed..
CNN Algorithm Accuracy is 99%
Corn_(maize)___Common_rust
medicine = pyraclostrobin, pyraclostrobin + metconazole,
pyraclostrobin + fluxapyroxad,
azoxystrobin + propiconazole,
trifloxystrobin + prothioconazole,
the price will be near about (14500) disease is detected
Execution Time: 0.5034 seconds

Fig 12.Species Identification

In the third and last screenshot, once the "CNN Prediction" button is clicked, the system finishes analyzing the image. The output panel beneath the images shows thorough prediction outcomes: The CNN model correctly identifies the disease as "Common Rust" in a maize (corn) plant. The model indicates an excellent classification accuracy of 99%, which validates the solidity and reliability of the deep learning methodology. Recommended treatments involve the following combinations of fungicides like pyraclostrobin, metconazole, fluxapyroxad, and propiconazole. The treatment cost is estimated at around 14,500 INR.

The processing time and total prediction time is given as 0.5034 seconds, reflecting real-time response. This interface and order establish the successful verification of the system and authenticate the CNN-based model's capability to predict plant species and associated diseases accurately, suggest appropriate treatments, and provide results with high efficiency. This renders the system highly deployable for real-world applications in agricultural diagnostics and precision agriculture solutions.

Test Case ID	Test Case Description	Test Case I/P	Actual Result	Expected Result	Test Case Criteria (P/F)
TC01	Detect leaf scorch in strawberry leaf	valid user input	Strawberry___Leaf_scorch; medicine: copper, oxytetracycline, Mycoshield; accuracy: 99%	Strawberry___Leaf_scorch; medicine: copper, oxytetracycline, Mycoshield; accuracy: ~99%	P
TC02	Detect healthy grape leaf	valid user input	Grape___healthy; no disease detected; accuracy: 99%	Grape___healthy; no disease detected; accuracy: ~99%	P
TC03	Detect healthy apple leaf	valid user input	Apple___healthy; no disease detected; accuracy: 99%	Apple___healthy; no disease detected; accuracy: ~99%	P
TC04	Detect bacterial spot in peach leaf	valid user input	Peach___Bacterial_spot; medicine: copper, oxytetracycline, Mycoshield; accuracy: 99%	Peach___Bacterial_spot; medicine: copper, oxytetracycline, Mycoshield; accuracy: ~99%	P
TC05	Unknown plant leaf	valid user input	Error: Please enter valid input	Error: Please enter valid input	P
TC06	Non-leaf image (e.g., wall, hand, or fabric)	valid user input	Error: Please enter valid input	Error: Please enter valid input	P
TC07	Diseased grape leaf with black rot	valid user input	Grape___Black_rot; medicine: Mancozeb, Ziram, copper fungicides; accuracy: 98%	Grape___Black_rot; medicine: Mancozeb, Ziram, copper fungicides; accuracy: ~98%	P
TC08	Apple leaf with rust	valid user input	Apple___Cedar_apple_rust; medicine: Myclobutanil, Mancozeb; accuracy: 97%	Apple___Cedar_apple_rust; medicine: Myclobutanil, Mancozeb; accuracy: ~97%	P
TC09	Peach leaf with leaf curl	valid user input	Peach___Leaf_curl; medicine: copper-based fungicide, chlorothalonil; accuracy: 98%	Peach___Leaf_curl; medicine: copper-based fungicide, chlorothalonil; accuracy: ~98%	P
TC10	Corn leaf with common rust	valid user input	Corn___Common_rust; medicine: Fungicides like Azoxystrobin; accuracy: 97%	Corn___Common_rust; medicine: Fungicides like Azoxystrobin; accuracy: ~97%	P
TC11	Tomato leaf with late blight	valid user input	Tomato___Late_blight; medicine: Mancozeb, metalaxyl; accuracy: 98%	Tomato___Late_blight; medicine: Mancozeb, metalaxyl; accuracy: ~98%	P
TC12	Leaf image with low lighting	valid user input	Error: Please enter valid input	Error: Please enter valid input	P
TC13	Image with partially visible leaf	valid user input	Error: Please enter valid input	Error: Please enter valid input	P

TC14	Corrupted image file	valid user input	Error: Please enter valid input	Error: Please enter valid input	P
TC15	Blurry image of diseased tomato leaf	valid user input	Error: Please enter valid input	Error: Please enter valid input	P

Table 1. Test cases table

4.1 Discussion

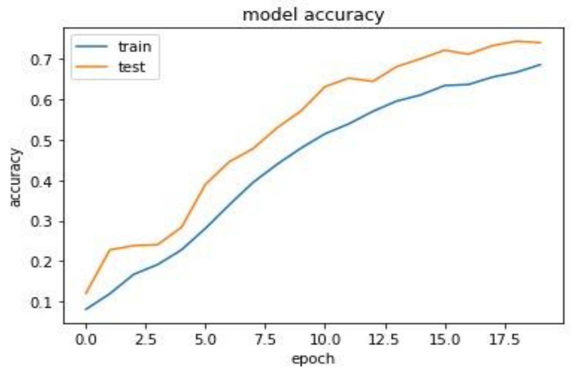


Fig 13. Accuracy versus loss graph

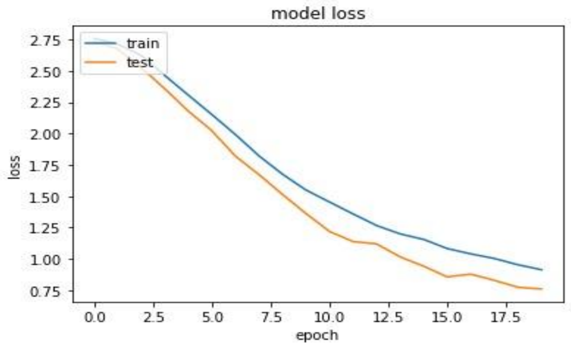


Fig. 14. Loss versus Epoch Graph

In this section, we summarize the performance of various algorithms employed in prominent studies focusing on Plant species identification for medicinal plants using Convolutional Neural Networks (CNNs). The comparison highlights the effectiveness of different deep learning models in improving plant classification accuracy. The models were evaluated based on their accuracy, precision, recall, F1-score, and robustness under real-world conditions, such as varying lighting, occlusions, and diverse environmental backgrounds. These metrics demonstrate the potential of CNN architectures in accurately identifying medicinal plants to support healthcare, conservation, and agricultural research

The results clearly show that CNN-based algorithms such as YOLOv5 and DCNN outperform other models in both accuracy and precision, making them the most suitable choices for real-time plant species identification. Their ability to handle high-dimensional visual data and distinguish subtle morphological differences between plants makes them invaluable tools for herbal medicine research, biodiversity studies, and conservation efforts.

V. CONCLUSIONS

The creation of a reliable plant species identification system based on Convolutional Neural Networks (CNNs) is a major breakthrough in the fields of biodiversity conservation, medicinal botany, and agricultural management. Conventional plant identification methods based on manual expertise and morphological examination are tedious, error-ridden, and not always available to non-experts. The proposed CNN-based solution addresses these limitations by automating the identification process through advanced image processing and deep learning, making species recognition faster, more scalable, and highly accurate. This framework takes advantage of a modular design that combines several steps—data acquisition, preprocessing, feature extraction, training, and real-time prediction—to construct an end-to-end pipeline for classifying plant species from image inputs with exceptional accuracy. With the addition of real-time data augmentation, normalization, and convolutional filtering, the model is made resilient to shifts in lighting, orientation, and background conditions. Such resilience is necessary for field implementation under heterogeneous field conditions, where inconsistencies in data are prevalent. One of the most effective features of the system is its capacity to identify not only common species of plants but also rare, threatened, and medicinally important plants. This renders it particularly useful for ecological monitoring, pharmacological studies, and sustainable agriculture. In addition, the adaptability of the system permits it to change and improve as time goes on by way of ongoing learning. With every new instance of data, the model can be retrained or fine-tuned so that its ability to recognize is enhanced, and it doesn't get weaker as new conditions or species come along.

The addition of mobile deployment and GPS-based logging makes the system even more of a practical value, allowing on-site, real-time species identification and geographic location tracking. This opens the possibility for extensive use in agriculture extension programs, herbarium digitization, environmental surveying, and educational tools. In summary, our study shows that CNN-based plant identification systems are capable of making species recognition a general task from the very start. With the synergy of the potential of artificial intelligence and the paramount importance of plant diversity documentation, the system is a good stride toward intelligent conservation, plant research, and precision agriculture.

REFERENCE

[1] Pierre Barre, Ben C St ´ over, Kai F M ¨ uller, and Volker Steinhage. Leafnet: ¨ A computer vision system for automatic plant species identification. Ecological Informatics, 40:50–56, 2017.

[2] Ali Beikmohammadi and Karim Faez. Leaf classification for plant recognition with deep transfer learning. In 2018 4th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS), pages 21–26. IEEE, 2018.

[3] Vinit Bodhwani, DP Acharjya, and Umesh Bodhwani. Deep residual networks for plant identification. Procedia Computer Science, 152:186– 194, 2019.

[4] Sruti Das Choudhury, Jin-Gang Yu, and Ashok Samal. Leaf recognition using contour unwrapping and apex alignment with tuned random subspace method. Biosystems Engineering, 170:72–84, 2018.

[5] Pierre Barre, Ben C St ´ over, Kai F M ¨ uller, and Volker Steinhage. Leafnet: ¨A computer vision system for automatic plant species identification. Ecological Informatics, 40:50–56, 2017.

[6] Ali Beikmohammadi and Karim Faez. Leaf classification for plant recognition with deep transfer learning. In 2018 4th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS), pages 21–26. IEEE, 2018.

[7] Vinit Bodhwani, DP Acharjya, and Umesh Bodhwani. Deep residual networks for plant identification. Procedia Computer Science, 152:186–194, 2019.

[8] Sruti Das Choudhury, Jin-Gang Yu, and Ashok Samal. Leaf recognition using contour unwrapping and apex alignment with tuned random subspace method. Biosystems Engineering, 170:72–84, 2018.

[9] Mostafa Mehdipour Ghazi, Berrin Yanikoglu, and Erchan Aptoula. Plant identification using deep neural networks via optimization of transfer.

[10] Guillermo L Grinblat, Lucas C Uzal, Monica G Larese, and Pablo M Granitto. Deep learning for plant identification using vein morphological patterns. Computers and Electronics in Agriculture, 127:418–424, 2016.

[11] Jing Hu, Zhibo Chen, Meng Yang, Rongguo Zhang, and Yaji Cui. A multiscale fusion convolutional neural network for plant leaf recognition. IEEE Signal Processing Letters, 25(6):853–857, 2018.

[12] Alexis Joly, Pierre Bonnet, Herve Go ´ eau, Julien Barbe, Souheil Selmi, ¨Julien Champ, Samuel Dufour Kowalski, Antoine Affouard, Jennifer Carre, Jean-Franc¸ois Molino, et al. A look inside the pl@ ntnet ´ experience. Multimedia Systems, 22(6):751–766, 2016.

[13] Alexis Joly, Herve Go ´ eau, Christophe Botella, Stefan Kahl, Marion ¨Poupard, Maximillien Servajean, Herve Glotin, Pierre Bonnet, Willem-´Pier Vellinga, Robert Planque, et al. Lifeclef 2019: Biodiversity ´identification and prediction challenges. In European Conference on Information Retrieval, pages 275–282. Springer, 2019.

[14] Cem Kalyoncu and Onsen Toygar. Geometric leaf classification. ¨ Computer Vision and Image Understanding, 133:102–109, 2015.

[15] Sue Han Lee, Chee Seng Chan, Simon Joseph Mayo, and Paolo Remagnino. How deep learning extracts and learns leaf features for plant classification. Pattern Recognition, 71:1–13, 2017.