Deep Learning-Based Plant Identification App: Leveraging EfficientNet Model and Modern Web Technologies

Dr. Sheetal Mapare
Vidyalankar Institute of Technology
Mumbai, India
sheetal.mapare@vit.edu.in

Arnav Gonge
Vidyalankar Institute of Technology
Mumbai, India
arnav.gonge@vit.edu.in

Siddhi Manche
Vidyalankar Institute of Technology
Mumbai, India
siddhi.manche@vit.edu.in

Vedant Patwardhan
Vidyalankar Institute of Technology
Mumbai, India
vedant.patwardhan@vit.edu.in

***Abstract*— To slow down the rate of decreasing biodiversity, the process of automated plant identification plays a crucial role. It is also one of the keys factors in agriculture, research, and ecology. In the following paper, we propose a recent approach to tackle plant identification engaging deep learning techniques and state-of-the-art web and mobile development frameworks. Our application leverages the EfficientNet model trained on the PlantCLEF 2016 dataset along with an Indian medicinal plant image dataset. The backend was created using FastAPI and containerized using Docker, while the frontend interface utilized AndroidStudio for development. After thorough testing and validation, we demonstrate the effectiveness of our application in plant identification tasks resulting in an 82.2% Top-1 Accuracy. Our findings provide a pillar in using the EfficientNet model for plant identification tasks.**

Keywords—Plant Identification, Deep Learning, EfficientNet

I. INTRODUCTION

With about 382,000 species of plants in the world, plant identification is a monotonous process requiring accurate domain knowledge. This process is of utmost importance as many plants provide medicinal properties that are vital for treatment as well as drug synthesis. Advancements in computer-vision, ML and DL have opened a pathway to automate the identification of plant species. Be that as it may, the effectiveness of these solutions varies due to the massive amount of plant species to identify. An effective solution would require huge amounts of high-quality training data and intensive compute resources.

Recent research has shown significant advancements in plant leaf disease detection methodologies. Guo et al. [1] proposed a comprehensive approach, integrating a deep learning algorithm—Region Proposal Network for leaf localization—and the Chan-Vese algorithm for symptom segmentation, alongside transfer learning for classification. Their work highlights the effectiveness of this combined approach in achieving accurate plant leaf disease detection. Building upon this, M.H. Saleem et al. [2] demonstrated that deep learning architectures such as ResNet, GoogLeNet, and LeNet consistently outperform traditional ML methods like SVM and Random Forest in this domain. Moreover, Mohanty et al. [3] showcased the practical applicability of DL models in plant disease detection through a mobile application-based solution. Collectively, these studies underscore the growing prominence of deep learning techniques in revolutionizing image-based plant disease detection.

The identification of medicinal plants holds significant importance in traditional medicinal practices, with over 80% of the global population relying on herbal remedies. Reference [4] proposes a methodology for the identification of medicinal plants based on the visual characteristics of leaves and flowers, as suggested by Jayalath et al. The methodology comprises four main steps: image enhancement, segmentation, feature extraction, and classification using Back Propagation Artificial Neural Networks (ANN). Notably, the study emphasizes the use of high-quality images collected through interviews with Ayurvedic doctors and botanical gardens, resulting in a dataset of 5000 images with uniform resolution and size. However, while the inclusion of flowers facilitates plant identification, the reliance solely on high-quality images may limit the scalability of the approach, considering the challenges associated with obtaining such images for all plant species.

On the other hand, reference [5] focuses on the utilization of deep learning, specifically convolutional neural networks (CNNs) like Densenet121, for the automated identification of medicinal plant leaves, as demonstrated by Rao et al. The study highlights the importance of TensorFlow and Keras in implementing the Densenet121 architecture, proving the significance of deep learning frameworks in modern plant identification research. By leveraging deep learning techniques, this approach offers a scalable solution for plant identification, potentially overcoming the limitations of image quality and availability encountered in traditional methods.
Combining the strengths of both approaches—visual feature extraction and deep learning—holds promise for advancing the identification of medicinal plants, crucial for traditional medicinal practices and biodiversity conservation efforts. However, despite the abundance of medicinal plants worldwide, their accurate identification and understanding of their uses remain challenging, amplified by the scarcity of high-quality data.

Acknowledging these challenges, our contributions aim to address these limitations and enhance the efficiency of plant identification methods. We have developed a model capable of accurately predicting over 588 species of plants, utilizing both visual features and DL techniques. Furthermore, to facilitate accessibility and usability, we have created an

intuitive application that enables users to run predictions on images directly from their own devices.

In addition to predictive capabilities, our application incorporates a valuable feedback feature. This feature allows users to suggest the correct identification of a plant if initially misidentified, contributing to the continuous improvement and refinement of our model. By integrating user feedback and continuously updating our dataset, we strive to enhance the accuracy and reliability of our plant species recognition system, thereby supporting the sustainable utilization of plants having medicinal properties and the preservation of biodiversity.

II. METHODOLOGY

A. System Architecture

The system architecture involves deploying a plant species prediction model on an AWS EC2 instance, facilitated by Docker containerization [6]. The DL model is packaged into a Docker container for efficient deployment and management on the EC2 backend. Users interact with an Android app frontend to upload plant images for identification. These images are processed by the Dockerized model running on the EC2 instance, which provides prediction services via API endpoints. Predictions, along with user history and plant information, are stored in an AWS RDS database. Additionally, AWS S3 is utilized for image storage. This architecture ensures seamless integration, scalability, and ease of deployment, allowing for efficient identification.

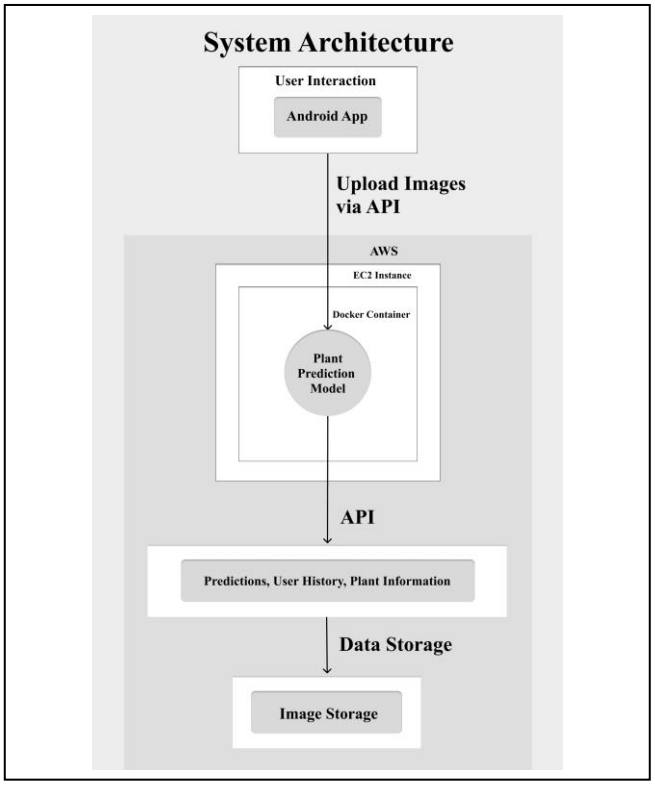

Fig. 1. Proposed System Architecture

B. Datasets

1) PlantCLEF 2015:

Aiming to develop an image-based plant identification system, PlantCLEF 2015 [7] is a dataset designed to test systems and methodologies on a massive scale that can be adjusted to real-world scenarios. The dataset was created using a community platform that allowed participation in 2011 and had thousands of participants. PlantCLEF 2015 is a collection of carefully chosen photos from various cameras, contributors, locations, seasons, and unique plants. More specifically, the PlantCLEF 2015 dataset consists of 113,205 images from 41,794 observations of 1000 different kinds of ferns, plants, and trees that are found throughout Western Europe. Every image has a binomen linked with it and corresponds to one of the seven viewing categories ("whole plant", "fruit", "leaf", "flower", "stem", "branch", and "leaf scan") in the meta-data. For our model, we utilized the PlantCLEF 2015 dataset as the primary dataset. Specifically, we focused on images labeled with the leaf and leaf scan tags from a smaller set of over 553 plant species. This subset was chosen to ensure a diverse representation of plants suitable for training our model. The dataset comprised a total of 19,305 training images and 4,956 validation images. By selecting images from these specific categories and species, we aimed to train our model to accurately identify plant species on the basis of leaf characteristics.

TABLE I. IMAGES EXTRACTED FROM PLANTCLEF 2015 DATASET

Images	Total	Leaf	Leaf Scan
Train	19305	9664	9641
Validation	4956	2414	2542

2) Indian Medicinal Plant Dataset:

To support the development of our mobile application tailored for the Indian Subcontinent, the availability of a comprehensive dataset of Indian medicinal plant species was important. This dataset [8] comprising of 5,945 images, sourced from various regions of Karnataka and Kerala, was utilized in our research. These images capture the diverse characteristics of forty plant species, including variations in resolution, illumination, background, and seasonal conditions Captured in real-time using smartphones, these images provide valuable insights for researchers interested in algorithmic models based on image processing, machine learning, and deep learning.

TABLE II. IMAGES EXTRACTED FROM INDIAN MEDICINAL PLANT DATASET

Images	Total
Train	4756
Validation	1189

C. Model Training

We employed the EfficientNetV2S model [9] for our image classification task, utilizing pre-trained weights from the ImageNet dataset [10] as the base for transfer learning. To adapt the model to our specific task, we added a dropout layer with a dropout rate of 0.2 to prevent overfitting. Subsequently, we appended a new softmax layer to the model to perform classification. During training, we employed a staged approach to fine-tune the model. Initially, we froze all layers except the newly added top layer and trained the model for 50 epochs. Following this, we gradually unfroze the last 156 layers of the network while keeping the batch normalization layers frozen. This fine-tuning strategy allows the model to learn task-specific features while retaining the learned representations from the pre-trained weights. The model was trained for an additional 50 epochs after unfreezing the specified layers. This approach aims to strike a balance between leveraging pre-trained features and adapting the model to our specific classification task.

1) Focal Loss:

The focal loss function, pioneered by Lin et al. [11], has emerged as a potent solution for handling class imbalance in classification tasks, particularly in computer vision applications. This loss function dynamically adjusts the contribution of individual training examples based on their predicted probabilities, effectively down weighting the influence of well-classified instances, and focusing more on those that are harder to classify. By introducing a modulating factor controlled by a parameter known as gamma, the focal loss provides a flexible mechanism to mitigate the effects of class imbalance and improve the performance of machine learning models.

In our research, we employed the focal loss, specifically the CategoricalFocalCrossentropy variant, to address class imbalance in our dataset. Notably, we calculated class weights (alpha values) based on the class frequencies in the dataset. Each class was assigned a unique alpha value proportional to its class count, allowing the model to prioritize learning from minority classes effectively. This approach aimed to rectify the imbalance in class representation and improve the model's ability to generalize across all classes.

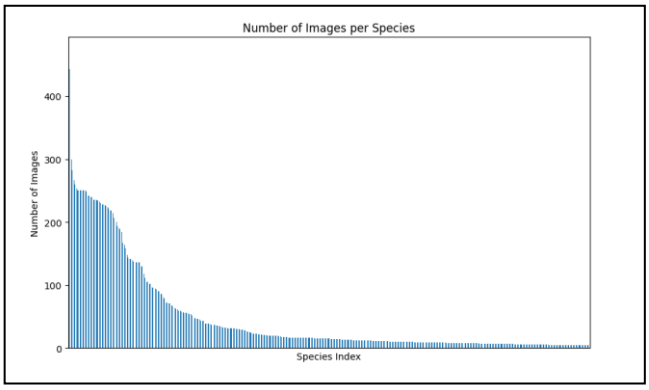

Fig. 2. Class counts for our merged datasets. Class indices are not displayed on the x-axis for conciseness.

2) Optimizer and Learning Rate:

In our research, we implemented a Cyclical Learning Rate (CLR) scheduler to optimize the training process of our model. CLR, proposed by Smith [12], is a learning rate policy that cyclically adjusts the learning rate between two boundaries over a predefined number of iterations. This dynamic adjustment enables the model to explore a wider range of learning rates during training, potentially leading to faster convergence and improved generalization performance.

The TriangularCyclicalLearningRate scheduler, inspired by CLR, operates by oscillating the learning rate between an initial value and a maximal value in a triangular fashion over a specified number of steps. This cyclic variation in the learning rate allows the model to effectively navigate the optimization landscape and discover better solutions. For our experiments, we configured the scheduler with a starting learning rate of 10^{-3} and a maximum learning rate of 10^{-2} for training only the softmax layer. We set the step size to 1504, determining the frequency of iterations within each cycle. After unfreezing the layers, we set the initial and maximal learning rates to 10^{-4} and 10^{-5} respectively. For this iteration the step size was set to 6016. By utilizing this cyclical learning rate strategy, we aimed to enhance the training dynamics of our model and potentially achieve better convergence and generalization.

D. Backend Server

Our system is built using FastAPI, a modern web framework for building APIs in Python. It utilizes TensorFlow for machine learning functionalities, including model loading and predictions. The system architecture includes AWS services such as S3 for image storage and RDS for database management.

We implement endpoints for user registration, login, image classification, feedback submission, and retrieval of user history. These endpoints handle requests from clients and interact with the database and model for processing. Our backend system is containerized using Docker for easy deployment and scalability. The Docker container encapsulates the entire backend application along with its dependencies, ensuring consistency across different environments. Subsequently, the Dockerized backend is deployed on an AWS EC2 instance, offering a robust and scalable infrastructure for serving API requests. EC2 instances provide flexibility in terms of computing power and storage capacity, enabling our system to adeptly handle varying workloads. Additionally, the images from user history are stored in S3, while all user-related information resides in the RDS, ensuring efficient management and retrieval of data. Overall, our system provides a user-friendly interface for users to classify medicinal plant species using images captured from smartphones, with the capability to store user interactions for further analysis and improvement.

E. Database

The MySQL database schema facilitates various functionalities essential for the application's operation. It

includes user management capabilities, allowing for the registration and authentication of users. Additionally, the database stores user interaction history, enabling the tracking of user activities and predictions made by the application. Feedback collection from users is also supported, providing a mechanism for users to submit their opinions or comments regarding the application's performance or content.
Moreover, the schema includes tables for storing information about plant species, such as their names, common names, and uses, which is integral for the application's core functionality of plant species identification and information retrieval. When the model runs a prediction on the image, the class index is queried from the database to retrieve additional information about the predicted plant species, enhancing the user's experience by providing comprehensive details about the identified plants.

F. Mobile Frontend

For the mobile frontend of our plant identification application, we utilized Flutter, a framework for building cross-platform mobile applications using the Dart programming language. Flutter offers a set of pre-built widgets and tools that allows to develop visually appealing and responsive user interfaces. The user interface (UI) of our mobile application was designed to be interactive and user-friendly, ensuring a seamless experience for users.

We employed Flutter's widget library to provide easy access to key functionalities such as image upload, prediction interface and feedback submission. The feature allows users to either capture real-time photos using their mobile camera or select an image from their gallery. This functionality leverages Flutter's image picker plugin to provide a seamless image selection experience.

Once users upload an image, the mobile application communicates with the backend server to process the image using our trained model. The prediction results, including the top predicted plant species and their corresponding confidence scores, are then displayed to the user within the application's interface. We utilized Flutter's text and layout widgets to present this information in a clear and visually appealing manner.

In addition to identifying plants, our mobile application includes a feedback feature that allows users to provide corrections or additional information if a plant is misidentified. Users can easily submit feedback directly from the application, contributing to the continuous improvement of our plant identification model. This functionality was implemented using Flutter's form widgets and HTTP client for communication with the backend server.

To enhance user engagement and provide a personalized experience, our mobile application includes a user history feature that maintains a record of past plant identification requests and their corresponding results. This feature enables users to review their previous interactions with the application. Flutter's local storage capabilities were utilized to store and retrieve user history data within the application.

Overall, the mobile frontend of our plant identification application offers a seamless and intuitive user experience, leveraging Flutter's capabilities to deliver a responsive and visually appealing interface. By combining efficient backend processing with a user-friendly frontend, our application empowers users to identify plant species accurately and contribute to the refinement of our model through feedback submissions.

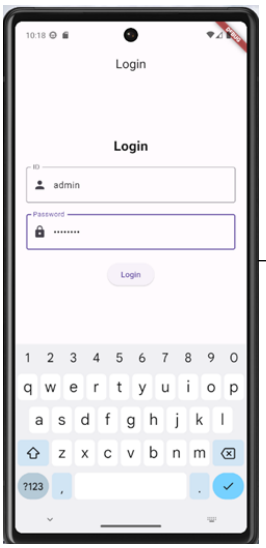

Fig. 3. Login Page

Fig. 4. Image Selection Page

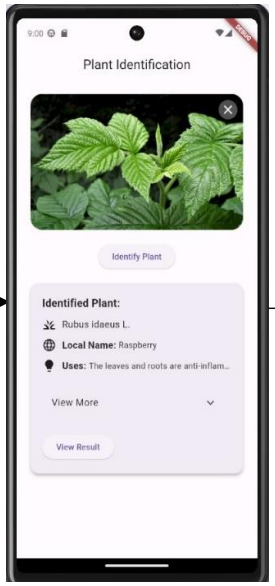

Fig. 5. Plant Identification

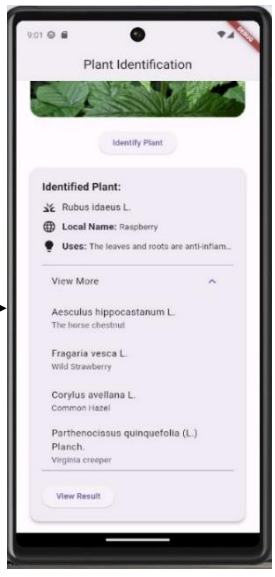

Fig. 6. Detailed information of identified plants

III. EXPERIMENTS AND RESULTS

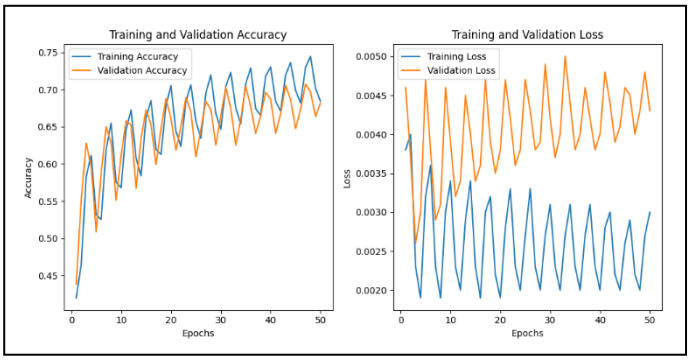

Fig. 7. Training and Validation Loss Trends (Softmax Layer)

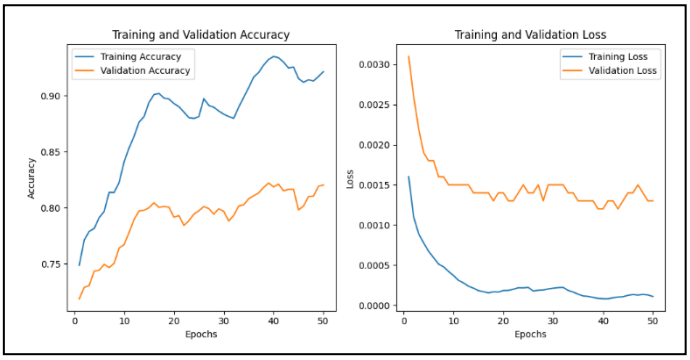

Fig. 8. Training and Validation Loss Trends (After Unfreezing 156 Layers)

A. *Initial Training Phase*

During the initial training phase, focusing on training only the softmax layer for the first 50 epochs, the model exhibited a noticeable improvement in performance. The training accuracy steadily increased to approximately 74%, indicating effective learning from the training dataset. Simultaneously, the validation accuracy, although slightly lower, reached a commendable 71%, demonstrating the model's ability to generalize well to unseen data. The corresponding loss metrics showed consistent convergence, with the training loss decreasing to around 0.0022 and the validation loss stabilizing at approximately 0.0040.

B. *Unfreezing Layers*

Subsequently, as the last 156 layers were unfrozen while keeping batch normalization layers frozen, the model underwent further refinement to enhance its predictive capabilities. Over the subsequent 50 epochs, the model's accuracy experienced a notable boost, reaching approximately 92% on the training dataset and 82% on the validation dataset. This improvement suggests that unfreezing additional layers facilitated the model in capturing more intricate patterns and nuances present in the data. Moreover, the loss metrics continued to decline, with the training loss diminishing to approximately 0.0001 and the validation loss converging around 0.0013.

C. *Precision, Recall, F1 Score, and Support:*

Evaluation of precision, recall, F1 score, and support across all classes revealed promising results. The weighted average precision (1), recall (2), and F1 score (3) were computed to be 0.84, 0.84, and 0.83, respectively, indicating a balanced performance across different classes. The cumulative support (4), representing the number of samples, was found to be 6145, affirming the model's robustness and reliability across a diverse range of medicinal plant species.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (1)$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (2)$$

$$F1\ Score = 2 \times \frac{Precision\ x\ Recall}{Precision + Recall} \quad (3)$$

$$Support = Total\ no.\ samples \quad (4)$$

IV. Conclusion

In conclusion, our research presents a comprehensive approach to leveraging machine learning for the classification of medicinal plants using mobile-captured images. Through the creation and implementation of a sophisticated model architecture, we have demonstrated promising results in accurately identifying various plant species, which holds major implications for fields such as agriculture, pharmacology, and botanical research. Our system, built on FastAPI and TensorFlow, offers a user-friendly interface for seamless interaction, while AWS services provide robust backend infrastructure for scalability and reliability.

The evaluation of our model's performance showcases commendable accuracy and generalization capabilities, particularly evident during the initial training phase focusing on the softmax layer and subsequent fine-tuning of additional layers. Precision, recall, and F1 score metrics further attest to the balanced performance across different plant species, underscoring the model's robustness and reliability.

V. Future work

Moving forward, our research suggests several areas for future exploration and enhancement:

A. Expansion of Dataset:

Acquiring a more extensive and diverse dataset, including images spanning various plant ages and growth stages, is essential for improving model generalization and accuracy.

B. Integration of Plant Age Information:

Incorporating metadata on plant age into the dataset and training process can facilitate better classification accuracy, enabling the model to adapt to variations in plant morphology over time.

C. Exploration of Transfer Learning Techniques:

Further investigation into transfer learning methods, particularly utilizing pre-trained models on larger datasets like ImageNet, could accelerate model training and enhance classification performance.

D. Crowdsourced Data Collection:

Engaging citizen scientists in crowdsourced data collection initiatives can help gather a broader range of images from different geographical regions, enriching the dataset and improving model robustness.

E. Integration of Multi-Sensor Data:

Exploring the integration of multi-sensor data, such as spectral or environmental information, alongside image data, could provide valuable context for plant classification, enhancing model interpretability and resilience to environmental variability.

References

[1] Y. Guo, J. Zhang, C. Yin, X. Hu, Y. Zou, Z. Xue, and W. Wang, "Plant Disease Identification Based on Deep Learning Algorithm in Smart Farming," Discrete Dynamics in Nature and Society, vol. 2020, p. 2479172, Aug. 2020, doi: 10.1155/2020/2479172.

[2] M. H. Saleem, J. Potgieter, and K. Mahmood Arif, "Plant Disease Detection and Classification by Deep Learning," Plants (Basel), vol. 8, no. 11, p. 468, Oct. 2019, doi: 10.3390/plants8110468.

[3] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using Deep Learning for Image-Based Plant Disease Detection," Front. Plant Sci., vol. 7, pp. 1419, May 2016, doi: 10.3389/fpls.2016.01419.

[4] D. Jayalath, D. Nadeeshan, G. Amarawansha, H. Jayasuriya, and D. Nawinna, Identification of Medicinal Plants by Visual Characteristics of Leaves and Flowers, Dec. 2019. doi: 10.1109/ICIIS47346.2019.9063275.

[5] R. Upendar Rao, M. Sai Lahari, K. Pavana Sri, K. Yaminee Srujana, and D. Yaswanth, "Paper Id: IJRASET41190," IJRASET, Apr. 03, 2022. doi: 10.22214/ijraset.2022.41190.

[6] M. Openja, F. Majidi, F. Khomh, B. Chembakottu, and H. Li, "Studying the Practices of Deploying Machine Learning Projects on Docker," in Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering (EASE '22). Association for Computing Machinery, New York, NY, USA, 2022, pp. 190–200. doi: 10.1145/3530019.3530039.

[7] A. Joly et al., "LifeCLEF 2015: Multimedia Life Species Identification Challenges," in Experimental IR Meets Multilinguality, Multimodality, and Interaction, J. Mothe et al. (Eds.), Springer International Publishing, Cham, 2015, pp. 462-483. DOI: 10.1007/978-3-319-24027-5_46.

[8] B. R. Pushpa and Shobha Rani, "Indian Medicinal Leaves Image Datasets," Mendeley Data, Version 3, 2023, doi: 10.17632/748f8jkphb.3.

[9] M. Tan and Q. V. Le, "EfficientNetV2: Smaller Models and Faster Training," arXiv:2104.00298 [cs.CV], 2021.

[10] J. Deng et al., "ImageNet: A large-scale hierarchical image database," in Proceedings of the Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009, IEEE, Piscatway, NJ, USA, 2009, pp. 248–255.

[11] Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2980-2988. doi:10.1109/ICCV.2017.324

[12] Smith, L. N. (2017). Cyclical Learning Rates for Training Neural Networks. Proceedings of the 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), 464-472. doi:10.1109/WACV.2017.58