School of Computer Science
Faculty of Science and Engineering
University of Nottingham Malaysia

UG FINAL YEAR DISSERTATION REPORT

Tropical Plant Classification and Identification with Convolutional Neural Networks (CNN)

Student Name: Lisa Ho Yen Xin

Student ID: 20297507

Supervisor Name: Dr Tissa Chandesa

Year: 2024

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE WITH ARTIFICIAL INTELLIGENCE (HONS)
THE UNIVERSITY OF NOTTINGHAM

Tropical Plant Classification and Identification with Convolutional Neural Networks (CNN)

Submitted in April 2024, in partial fulfillment of the conditions of the award of the degrees B.Sc.

- Lisa Ho Yen Xin -
School of Computer Science
Faculty of Science and Engineering
University of Nottingham
Malaysia

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature [illegible]

Date 30/04/2024

Acknowledgement

I would like to express my deepest gratitude to my supervisor, Dr. Tissa Chandesa, for his invaluable guidance, support, and insight throughout this project. His expertise and encouragement have been crucial in ensuring the success of this project.

I would also like to thank my family and friends for their continuous support and understanding. Their encouragement and belief in my abilities have been a constant source of motivation.

Abstract

This dissertation presents a comprehensive evaluation of the performance of various pre-trained Convolutional Neural Networks (CNN) models to determine their effectiveness in identifying tropical plant species. The study explores several image thresholding techniques, including Otsu's thresholding, Adaptive Mean thresholding, Adaptive Gaussian thresholding, and Rosin thresholding, for image segmentation. The CNN models examined in this research include VGG-19, ResNet-101, MobileNetV2, InceptionV3, and EfficientNetB0. The results of the study indicate that MobileNetV2 achieves the highest accuracy (81.25%) in tropical plant species identification.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Background and Motivation

Plant classification and identification are integral components of various fields, ranging from agriculture and ecology to conservation and medicine. Accurate plant species identification is essential for understanding ecosystems, managing crops, protecting biodiversity, and identifying possible medical applications. Despite its significance, accurately identifying plant species presents several challenges, particularly for individuals without a background in botany [1].

With approximately 390,000 to 420,000 plant species worldwide, of which only 270,200 have been identified and recognised by botanists [2], the task becomes labour-intensive, subjective, and time-consuming [3]. According to [4], leaves are a preferred choice for the recognition and classification of various plant species due to their flat and two-dimensional surfaces, which exhibit diverse characteristics like texture, colour, and shape. The diversity of plant species, with thousands of leaf variations exhibiting different physical traits that may change throughout the seasons, adds to the complexity of plant identification [3]. Additionally, some species possess outward characteristics similar to those within the same species, demanding a keen eye for detail and in-depth knowledge of plant taxonomy for accurate identifications [5].

To address these challenges and democratise plant knowledge, the combination of mobile technology and artificial intelligence has created new opportunities in the field of automated approaches, particularly machine learning and deep learning. Among deep learning algorithms, Convolutional Neural Networks (CNN) have been extensively studied for their superior performance in image recognition tasks, including the categorisation of numerous plant species. The properties of plant leaves are currently used in many automated plant recognition methods, establishing a foundation for trustworthy plant identification. This approach's ensuring validity and consistency have solidified its role in dependable plant species recognition over time [4].

This advancement has given rise to innovative applications that enable novices to learn about plant species independently and at their own pace [5]. Leveraging the power of artificial intelligence, these applications allow users to identify plants based on photographs captured via hand-held devices [6]. The effectiveness of these applications is further amplified by a vast collection of categorised and well-documented plant specimens, serving as a valuable botanical archive and reference database. Continuously enriched by contributions from researchers and botanists, this database ensures accurate and trustworthy results in plant identification [5].

1.2 Problem Statement

Existing plant identification and classification techniques often lack specificity and robustness when applied to tropical plants due to the remarkable biodiversity, continuous growth patterns, and fluctuating environmental conditions of tropical plant ecosystems [7]. With an estimated 40,000 to 53,000 species inhabiting tropical regions [8], there is a need to develop an effective CNN-based system tailored to the unique characteristics and complexities of tropical plant identification. This challenge is exacerbated by the lack of comprehensive datasets on tropical plants and issues related to image quality consistency in these regions [9].

1.3 Aim

The aim of this project is to evaluate the performance of CNN models in recognising tropical plant species.

1.4 Objectives

The proposed project aims to achieve the following objectives:

1. **Comparative Analysis of CNN Models**: Conduct a comprehensive comparative analysis of pre-trained CNN models, including VGG-19, ResNet-101, MobileNetV2, InceptionV3, and EfficientNetB0, to determine which model consistently provides the highest accuracy for tropical plant species identification.
2. **Thresholding-Based Image Segmentation**: Evaluate various thresholding techniques for image segmentation, including Otsu's thresholding, Adaptive Mean thresholding, Adaptive Gaussian thresholding, and Rosin thresholding, to determine the most effective method for segmenting tropical plant images.
3. **Web Application Development**: Develop a user-friendly web application that integrates the selected CNN model for tropical plant identification.

1.5 Project Overview

This report provides an overview of a project focused on advancing tropical plant species identification through the utilisation of CNN models. Acknowledging the challenges posed by the unique characteristics and biodiversity of tropical ecosystems, the project aims to evaluate and analyse the performance of five pre-trained CNN models, including VGG-19, ResNet-101, MobileNetV2, InceptionV3, and EfficientNetB0, and explore thresholding-based segmentation techniques. The primary objective is to identify the most effective CNN model for accurate and robust tropical plant species recognition.

2. Literature Review

2.1 Machine Learning and Deep Learning

Machine learning and deep learning are both branches of artificial intelligence, as illustrated in Figure 2.1. Any algorithm that can learn from data falls under the umbrella of machine learning. This includes both simpler statistically based algorithms and more complicated ones like neural networks. However, machine learning models are only capable of identifying patterns that are similar to training data [10].

Deep learning is a subset of machine learning that learns and makes predictions from large amounts of data by utilising neural networks. Neural networks are structured similarly to the human brain, allowing them to recognise complex patterns and relationships in the data [11]. Deep learning presents a key advantage in its ability to handle problems that conventional machine learning approaches struggle to address or consider impractical to solve. The accuracy of deep learning models increases as more data is gathered and analysed, allowing them to remain up-to-date and effective as new information becomes available [11].

Since deep learning models are relatively robust to noise and distortions in images, they can identify plant species from photographs obtained in real settings with up to 91.78% accuracy, as shown in a study by the Beijing Forestry University [12]. Due to their scalability, they can manage big datasets. This increases their accuracy and allows them to emerge as powerful tools in the field of plant identification. However, the process of developing models that autonomously adapt to new data poses significant computational challenges and requires extensive training, robust infrastructure, and access to large volumes of data to achieve accurate and reliable predictions [11].

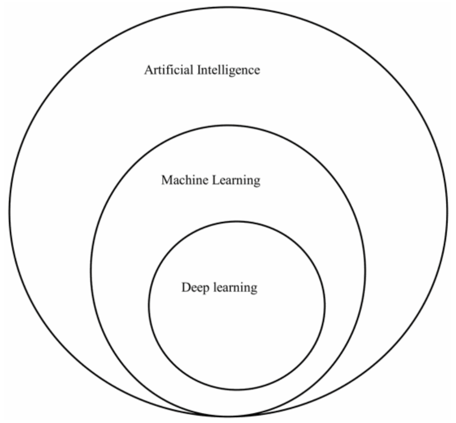


Figure 2.1 Artificial Intelligence Family

Among deep learning algorithms, CNNs have been extensively studied for their superior performance in image recognition tasks, including the categorisation of numerous plant species [13]. CNN models use convolutional layers to process and identify patterns within grid-like structured data [14], followed by the utilisation of one or more fully connected layers to categorise these identified features. This entire process is seamlessly integrated within a single framework that comprises multiple layers that are organised in a hierarchical manner [15].

CNN models possess notable advantages, including their capability to seamlessly integrate feature extraction and classification processes [15], as well as their aptitude for handling substantial volumes of data while establishing intelligent neural connections. Furthermore, CNN models often bypass the need for extensive preprocessing of image data [13].

2.2 Types of CNN models

An overview of CNN models well-suited for plant identification is presented in Table 2.1, including key details such as the year the model was introduced, the number of layers it comprises, and key distinguishing features. The highlighted CNN models include VGG-19, ResNet-101, MobileNetV2, InceptionV3, and EfficientNetB0.

The **VGG-19** network, created by the Visual Geometry Group at Oxford University in 2014, has 16 convolutional layers and three fully connected layers, all of which make use of Rectified Linear Unit (ReLU) activation function. VGG-19 stands out due to its stacked architecture, which has more convolutional layers than AlexNet [16]. It is well known for its capacity to extract complex features from images and has made significant contributions to the field of image classification.

Residual Network (ResNet) is renowned for its deep architecture, which occasionally has more than 100 layers. It introduced the idea of residual blocks, enabling the model to skip certain layers during training. This breakthrough made it possible to train much deeper CNNs and helped solve the vanishing gradient issue in very deep networks [16]. **ResNet-101**, which was introduced in 2015, has 101 layers and is widely used in various computer vision tasks, including image classification, object detection, and semantic segmentation [17].

InceptionV3 is a 48-layer CNN architecture that was developed by Google AI in 2015. The Inception module, a type of residual block that is designed to be effective and precise, is the foundation of InceptionV3. Inception modules can learn more complicated features than conventional CNN designs since they extract features from images using a range of kernel sizes [18].

MobileNetV2 was developed by Google AI in 2018. It is based on the inverted residual block (IR block) architecture and is a lightweight and efficient CNN architecture that is well-suited for deployment on mobile devices [19].

EfficientNet, developed by Google AI in 2019, is known for its ability of maintaining a fair balance between accuracy and efficiency. The network can learn more complicated features without sacrificing speed due to the compound scaling strategy, which scales the network's width, depth, and resolution in an even manner [20]. The base model, **EfficientNetB0**, strikes a fine balance that makes it suitable for various real-world applications.

Table 2.1 Types of CNN Models

Architecture	Year	Number of Layers	Key Features
VGG-19	2014	19	16 convolutional layers, 3 fully connected layers, ReLU activation, deep image feature extraction
ResNet-101	2015	101	Deep architecture, residual blocks
InceptionV3	2015	48	Inception modules, various kernel sizes
MobileNetV2	2018	Not specified	IR block
EfficientNetB0	2019	Not specified	Compound scaling strategy

2.3 Existing Plant Datasets

The project's primary focus is on tropical plant species. However, existing datasets often encompass a wide variety of plant types rather than being exclusive to tropical plants. To address this research need, several datasets are evaluated to curate a comprehensive dataset only containing tropical plant species, as shown in Table 2.2.

Table 2.2 Existing Plant Datasets

Datasets	Description
Pl@ntNet-300K	The Pl@ntNet-300K dataset contains 306,146 plant images covering 1,081 species. It contains images from the Pl@ntNet citizen observatory database. The images vary in quality and exhibit a strong class imbalance. This dataset is publicly available on the Zenodo open research platform [21].
iNaturalist	iNaturalist, an online social network of biodiversity observations, is a joint initiative of the California Academy of Sciences and National Geographic Society. It contains a very large collection of user-captured biodiversity images from multiple countries. It is also a free and accessible resource [22].
myDAUN (2017)	The myDAUN dataset contains 1,350 tropical shrub leaves images with a uniform background. It encompasses 45 distinct species with 30 leaf samples per species. The samples were collected in University of Malaya, Malaysia [23].
MepcoTropicLeaf (2021)	The MepcoTropicLeaf dataset contains 50 Indian Ayurvedic plant species from foothills of Western Ghats in Tamil Nadu [24].

Based on the evaluation in Table 2.2, iNaturalist is a suitable tool that meets the project's research need due to its vast collection of plant species for tropical regions. This diversity ensures a rich dataset for identifying a wide range of tropical plants. An advantage of using iNaturalist is that the photos are captured in real-world conditions, offering variations in lighting, angles, and backgrounds. This makes the dataset robust and more representative of real-world scenarios. Moreover, "Research Grade" observations are verified by a community of naturalists, enhancing the accuracy of plant labels in the photos and increasing the reliability of the dataset [22].

2.4 Thresholding-based Image Segmentation

Segmenting images poses challenges when dealing with harsh lighting conditions, shadows, and the similar colour tones shared by leaves and weeds. To reduce the background noise present in the dataset, several thresholding-based image segmentation techniques are reviewed.

Researchers conducted a study in 2022 where Otsu's thresholding was used to segment leaves and weeds. It was highlighted that the segmentation process was affected by shadows and harsh light [25]. Another study combined Otsu's thresholding with histogram equalisation for image enhancement, and morphological transformation to segment medicinal plant leaves [26].

In 2022, a study conducted in China implemented the Normalised Difference Vegetation Index (NDVI) to determine the extraction threshold of jujube trees and other ground objects [27]. A study conducted in 2019 focused on segmenting images by employing an adaptive thresholding method combined with the Normalised Difference Index (NDI) to distinguish between fruit and background [28].

2.5 Related Work

To efficiently categorise and identify plant species based on their distinctive traits and features, researchers have investigated a variety of methodologies throughout the years. Below is a brief review of several key approaches applied in the field of plant categorisation and identification.

2.5.1 Tropical Plants

The LifeCLEF Plant Identification Task 2019 focused on evaluating automated identification systems on a dataset of 10,000 plant species from data-deficient tropical regions. In the challenge, tropical flora was found to be inherently more difficult to identify compared to more generalist flora, resulting in significantly lower accuracy results [29].

A study conducted in 2022 applied EfficientNet-B0 on a mobile platform to classify Philippine herbal plants with an accuracy of 97.40% [30]. In 2020, a paper utilised ResNet with transfer learning and multi-stage visual data augmentation techniques, such as geometric transformation and distortion injection, to categorise rare and protected orchids in Indonesia. The proposed system showed convincing performance compared to existing methods [31].

2.5.2 Non-tropical Plants

In 2023, a paper compared deep learning algorithms such as ResNet50, MobileNet, and InceptionV3 for plant disease identification of common fruits and vegetables. The models achieved accuracy rates of 79.73%, 86.00%, and 93.33%, respectively [32]. A study conducted in 2021 based on the PlantVillage dataset for plant leaf disease classification showed that the B5 and B4 models of EfficientNet architecture achieved 99.91% and 99.97% respectively for accuracy [33]. In 2019, a paper compared VGG19, MobileNet, and MobileNetV2, to identify plant species in Hokkaido. MobileNetV2 achieved an average F1-score of 0.992, indicating high performance and practicality. The model is then integrated into a mobile application which showed a practical performance of 338.1 milliseconds per picture [34].

3. Description of Work

This dissertation focuses on evaluating CNN models for the identification of tropical plant species and explores thresholding techniques for segmenting tropical plant images. The study begins with a thorough review of existing literature on CNN models, plant datasets, and image thresholding techniques in the context of plant species identification. Based on this review, the research identifies VGG-19, ResNet-101, MobileNetV2, InceptionV3, and EfficientNetB0 as the CNN models to be evaluated. Simultaneously, the study investigates various image thresholding techniques suitable for segmenting tropical plant images, including Otsu’s thresholding, Adaptive Mean thresholding, Adaptive Gaussian thresholding, and Rosin thresholding.

The dataset utilised comprises images of tropical plant species sourced from diverse geographical locations. These images undergo preprocessing to enhance their quality and prepare them for input into the CNN models. Experiments are conducted on different datasets that are divided into training, validation, and testing sets. Each CNN model is trained on the training set and evaluated on the validation set. The model demonstrating the best performance on the testing set is selected as the optimal model and integrated into the web application developed for this study. Furthermore, segmentation experiments are conducted to assess the efficacy of various thresholding techniques in accurately extracting plant features from background elements.

4. Methodology

4.1 Implementation Details

The primary programming language used is Python due to its simple syntax and extensive library support for machine learning and deep learning tasks. Libraries such as Numpy, Pandas, Matplotlib, and Seaborn are useful for manipulating and analysing data. TensorFlow and its high-level API Keras provide efficient and user-friendly frameworks for building and training deep learning models. OpenCV was employed for image processing tasks, including data preprocessing and visualisation [35].

The project is run on Google Colab and JupyterLab. Google Colab is a free, cloud-based platform for running Python code in a Jupyter notebook environment [36]. JupyterLab offers a flexible and interactive workspace for coding, experimenting, and visualising results [37]. Due to Google Colab's limited RAM allocation per session, JupyterLab was used for areas requiring higher computational resources.

4.2 Dataset Preparation

The tropical plant dataset used in this project is generated from iNaturalist through the platform's export tool (see Figure 9.1 in Appendix A). "Research Grade" observations recorded before 20/12/2023 are narrowed down to countries known for their rich tropical biodiversity—Malaysia, Singapore, and Indonesia. The observations of the three countries, sorted in ascending order by genus, are combined into one CSV file, with each entry containing the scientific name of the plant and a corresponding image URL.

The combined dataset has a total of 105,632 images and 5,210 classes, with 1 to 1729 images per class. This makes the dataset highly imbalanced, which may lead to bias. Therefore, there is a need for data preprocessing. Classes with less than 10 images are removed, reducing the dataset to 95,557 images and 1,318 classes.

Given the constraints of limited computational resources, the decision is made to reduce the total dataset size to include only nine classes. Initially, the first nine classes, as listed in Table 4.1, are used in the experiment outlined in Section 5.1.2. Following the analysis of results (as discussed in Section 6.1), classes with less than 25 images are removed from the entire dataset. This refinement results in a dataset comprising 86,651 images across 747 classes. From this refined dataset, two smaller datasets are formed: one consisting of the first nine classes sorted in ascending order according to genus, and another composed of nine randomly chosen classes with a balanced number of images per class. The former dataset is used in the experiment detailed in Section 5.1.3, whereas the latter dataset is used in the experiment detailed in Section 5.1.6. The specific nine classes are listed in Tables 4.2 and 4.3 for reference.

Table 4.1 Initial Ascending Classes Dataset

No.	Name of Classes	Number of images
1.	Abelmoschus esculentus	75
2.	Abelmoschus moschatus	31
3.	Abroma augustum	37
4.	Abrus precatorius	20
5.	Acacia auriculiformis	153
6.	Acacia decurrens	13
7.	Acacia mangium	111
8.	Acalypha alopecuroidea	10
9.	Acalypha arvensis	10
Total		460

Table 4.2 New Ascending Classes Dataset

No.	Name of Classes	Number of images
1.	Abelmoschus esculentus	75
2.	Abelmoschus moschatus	31
3.	Abroma augustum	37
4.	Acacia auriculiformis	153
5.	Acacia mangium	111
6.	Acalypha hispida	151
7.	Acalypha indica	223
8.	Acalypha siamensis	93
9.	Acalypha wilkesiana	67
Total		941

Table 4.3 Random Classes Dataset

No.	Name of Classes	Number of images
1.	Amorphophallus paeoniifolius	106
2.	Cananga odorata	105
3.	Dactyloctenium aegyptium	103
4.	Ficus pumila	119
5.	Grona triflora	111
6.	Nerium oleander	112
7.	Syngonium podophyllum	101
8.	Talinum paniculatum	99
9.	Thunbergia laurifolia	101
Total		957

4.3 Data Augmentation

Based on the analysis in Section 6.1, data augmentation is applied to classes that have fewer than 75 images to ensure that each class in the dataset has an adequate number of images for effective model training. Augmentation parameters such as rotation range, width and height shift range, shear range, zoom range, horizontal flip, and fill mode are defined to introduce variability into the dataset. For each label that needs augmentation, all images corresponding to that label are selected and random transformations are applied to generate new augmented images. Each image undergoes augmentation three times to avoid over-augmentation. The augmented images and labels, as detailed in Table 4.3, are used in the experiments as described in Section 5.1.4 and Section 5.1.5.

Table 4.4 Augmented Dataset

No.	Name of Classes	Number of images
1.	Abelmoschus esculentus	75
2.	Abelmoschus moschatus	93
3.	Abroma augustum	111
4.	Acacia auriculiformis	153
5.	Acacia mangium	111

6.	Acalypha hispida	151
7.	Acalypha indica	223
8.	Acalypha siamensis	93
9.	Acalypha wilkesiana	201
Total		1211

4.4 Model Architecture

The CNN models are initialised with pre-trained weights obtained from the ImageNet dataset to leverage learned features. The top classification layer is excluded, and custom classification layers, including Flatten and Dense layers, are added to suit the specific task. Following the convolutional layers and max-pooling operations, the Flatten layer transforms the multi-dimensional feature maps into a one-dimensional vector. This flattened representation acts as the input for a Dense layer, where the number of units is determined by the number of classes in the classification task. The Dense layer employs a softmax activation function to produce class probabilities.

The VGG-19, ResNet-101, MobileNetV2, and EfficientNet-B0 models expect input images with dimensions of (224, 224, 3), while InceptionV3 expects input images with dimensions of (299, 299, 3). The output is a vector containing probabilities for each class, indicating the predicted class for the input image.

4.5 Model Parameters

4.5.1 Optimiser

Adam is chosen as the optimiser for model training due to its efficiency and advantages over other optimisers like stochastic gradient descent (SGD). It operates using only first-order gradients and minimal memory, calculating adaptive learning rates for each parameter by estimating the first and second moments of the gradients. This allows Adam to dynamically adjust learning rates based on parameter characteristics, leading to more effective and efficient optimisation. By combining the benefits of AdaGrad and RMSProp, Adam can automatically handle varying gradient magnitudes, prevent overshooting, address non-stationary objectives, and work with sparse gradients [38].

4.5.2 Loss Function

Categorical cross-entropy is a commonly used loss function in multi-class classification problems. The categorical cross-entropy loss calculation measures the difference between the true and predicted probabilities for each class, as represented by eq. (1).

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log\left(p_{ij}\right) \qquad \textbf{(1)}$$

N is the number of images, *C* is the number of classes, *y* is the true label, and *p* is the predicted probability of the true class. The loss is calculated for each image and averaged over the entire dataset [39].

4.6 Classification Performance Metrics

To evaluate the performance of the trained models, a confusion matrix and four primary quantitative metrics are used, including accuracy, precision, recall and F1-score.

4.6.1 Confusion Matrix

A confusion matrix is a table that visualises the performance of a classification model by comparing its predicted classes with the actual classes from a dataset. It displays the number of true positives, true negatives, false positives, and false negatives [39]. These values are useful for calculating quantitative metrics.

4.6.2 Accuracy

Accuracy, as defined in eq. (2), evaluates the overall effectiveness of a classification model by measuring the proportion of correctly classified instances relative to the total number of instances in the dataset.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions} \qquad \textbf{(2)}$$

However, in scenarios with imbalanced datasets, accuracy may not offer a reliable measure of the model’s performance due to its bias towards the majority class [39]. In such cases, additional metrics like precision, recall, and F1-score provide a more comprehensive evaluation.

4.6.3 Precision

Precision measures the proportion of true positive predictions out of all positive predictions generated by the model. Mathematically, as visualised in eq. (3), precision is defined as the number of true positive predictions divided by the sum of true positive predictions and false positive predictions [39].

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{3}$$

4.6.4 Recall

Recall, also known as sensitivity or True Positive Rate (TPR), measures a classification model's ability to correctly identify true positive instances out of all actual positive instances in the dataset. It is calculated by dividing the number of true positives by the sum of true positives and false negatives, as denoted in eq. (4).

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{4}$$

4.6.5 F1-score

F1-score assesses the overall performance of a model by calculating the harmonic mean of precision and recall, as depicted in eq. (5). A higher F1 score indicates that the model achieves a better balance between precision and recall [39].

$$F1 = 2 \cdot \frac{Precision\ \cdot Recall}{Precision + Recall} \tag{5}$$

4.7 Image Segmentation Process

The image segmentation process, as illustrated in Figure 4.1, comprises a series of processing steps aimed at achieving thresholding-based image segmentation. It includes steps such as converting images to grayscale, applying blurring techniques, enhancing contrast, and performing morphological operations. These steps prepare the image for segmentation by improving its quality and reducing noise. The segmentation process primarily relies on thresholding techniques to separate objects from the background. Once thresholding is applied, contour-based processing is used to extract and visualise the segmented regions.

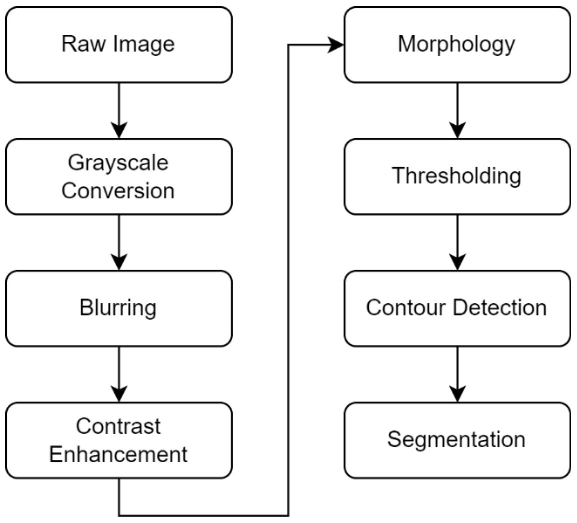


Figure 4.1 Image Segmentation Process

4.7.1 Image Blurring

The bilateral filter is a type of non-linear image filter used in image processing for noise reduction and edge preservation. Unlike traditional linear filters, which apply a fixed-weight kernel to each pixel, bilateral filtering considers both spatial proximity and intensity similarity when smoothing an image [40].

4.7.2 Contrast Enhancement

Contrast Limited Adaptive Histogram Equalisation (CLAHE) is a technique used to enhance the contrast of images, offering improvements over traditional histogram equalisation. In CLAHE, the image is divided into small regions called tiles, and histogram equalisation is applied to each tile individually. This approach enables localised contrast enhancement, allowing for different adjustments to contrast in various regions of the image, based on the characteristics of each tile [41].

A key feature of CLAHE is its contrast limiting mechanism. After equalising the histogram of each tile, the pixel values are clipped based on a specified contrast limit. This prevents over-amplification of noise in regions with large intensity variations [41]. By addressing this limitation, CLAHE ensures more effective contrast enhancement while preserving image quality.

4.7.3 Morphology

Morphology is a fundamental technique in image processing that involves the extraction of image components and the modification of shapes based on the spatial arrangement and intensity values of pixels. Among the various morphological operations, morphology closing combines dilation and erosion operations to effectively remove small noise or artifacts and connect disjointed parts of objects in an image. By smoothing out irregularities and eliminating jagged edges, morphology closing prepares the image for subsequent processing steps, such as thresholding, by providing a cleaner input [42]. The operation of morphological closing can be expressed using mathematical notation, as denoted in eq. (6).

$$(f \oplus B) \ominus B \tag{6}$$

f is the input image, B is the structuring element, $\oplus$ represents the dilation operation, and $\ominus$ represents the erosion operation.

4.7.4 Otsu's Thresholding

Otsu's thresholding is a histogram-based thresholding method that separates pixels into foreground and background based on their intensity values. It calculates the optimal threshold value by minimising the intra-class variance of pixel intensities, as represented in eq. (7). It is particularly effective when the histogram of pixel intensities exhibits a bimodal distribution, indicating clear distinctions between foreground and background intensities [42].

$$\sigma_B^2 = W_b \cdot W_f \left(\mu_b - \mu_f\right)^2 \tag{7}$$

σ_B^2 is the intra-class variance, W_b and W_f are the probabilities of the background and foreground classes, and μ_b and μ_f are the means of the background and foreground classes.

4.7.5 Adaptive Thresholding

Adaptive thresholding computes different threshold values for different parts of the image, unlike global thresholding methods, which apply a single threshold value to the entire image. In Adaptive Mean thresholding, a local threshold is calculated for each pixel based on the mean intensity of its neighbouring pixels [43]. This allows the threshold to adapt to local intensity changes, making it effective in handling illumination variations. On the other hand, Adaptive Gaussian thresholding calculates the local threshold for each pixel using a weighted sum of intensities in its neighbourhood, where the weights are determined by a Gaussian kernel [44]. This method assigns more weight to pixels closer to the centre of the kernel, resulting in a smoother transition between foreground and background regions compared to Adaptive Mean thresholding.

4.7.6 Rosin Thresholding

Rosin thresholding is an effective unimodal thresholding technique for dealing with histograms that exhibit a single prominent peak and do not follow a Gaussian distribution. By pinpointing the maximum deviation point between the histogram curve and the line drawn from the peak of the histogram to one of its endpoints, Rosin thresholding accurately determines the threshold value for image segmentation [45].

4.7.7 Contour Detection

Contour detection is the process of identifying continuous curves or outlines within an image that have distinct edges or boundaries [46]. It is applied after thresholding to precisely determine the boundaries of objects within the foreground of an image, since thresholding may result in regions with unclear or jagged boundaries due to noise or variations in lighting. Contour detection allows for a more accurate representation of object shapes by providing structured and well-defined boundaries for each object in the image.

4.7.8 Normalised Difference Index (NDI)

NDI is a spectral index commonly used in remote sensing and image processing applications to highlight certain features or characteristics within an image. By calculating the NDI, which represents the normalised difference between near-infrared (NIR) and red (RED) spectral bands, areas of interest such as vegetation can be determined due to their distinct spectral reflectance properties [47]. In practice, thresholding methods can utilise the computed NDI values as thresholds to segment the image into different regions or classes based on the desired characteristics. For instance, in vegetation detection applications, areas with high positive NDI values may correspond to healthy vegetation, while lower NDI values might indicate non-vegetated regions or other land cover types. By setting appropriate thresholds based on NDI values, thresholding algorithms can effectively distinguish between different features in the image, enabling precise segmentation and analysis. The formula eq. (8) for NDI is as follows:

$$NDI = \frac{NIR - RED}{NIR + RED} \quad \textbf{(8)}$$

4.8 Web Application Platform

Streamlit is chosen as the platform for developing the web application due to its reputation for simplicity and ease of use, particularly within the data science and machine learning communities. It allows the creation of interactive web applications directly from Python scripts and facilitates efficient development and debugging processes. Additionally, the Streamlit Community Cloud provides a user-friendly interface for hosting Streamlit web applications [48].

5. Experiments, Results and Discussion

In this section, the experimental methodologies used are outlined, and the findings of the conducted experiments, along with their implications, are presented. It covers the performance evaluation of CNN models, the effectiveness of various image segmentation techniques, and the integration of a model into a web application.

5.1 CNN Models Performance

5.1.1 Experiment Settings

Libraries such as pandas and NumPy are imported for data manipulation and analysis, while image processing tasks are handled by PIL, skimage, and OpenCV. Machine learning models are built and trained using libraries such as scikit-learn and TensorFlow, with Matplotlib and seaborn handling the visualisation of results and metrics.

The dataset is loaded from a CSV file containing image URLs and labels before being stored in NumPy arrays. Data augmentation techniques are applied using an *ImageDataGenerator* object from the Keras library. Before training, the images undergo resizing and normalisation, while the categorical labels are encoded into binary vectors using scikit-learn's *LabelBinariser* for multi-class classification. The dataset is then divided into 80% training, 10% validation, and 10% testing sets, with a fixed random seed for reproducibility.

The models are implemented using Keras and trained with categorical cross-entropy as the loss function, Adam as the optimiser, and accuracy as the metric for evaluation. The trained models are saved in the *SavedModel* format to ensure flexibility and portability. The trained models are evaluated on the testing set to assess their performance on unseen data based on performance metrics such as accuracy, precision, recall, and F1-score, which are calculated using functions from the modules in scikit-learn. Additionally, confusion matrices are generated to assess the models' overall performance and identify misclassified patterns.

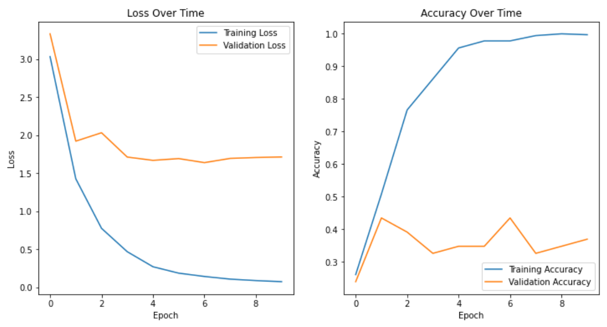
5.1.2 Experiment 1: Initial Dataset

In Experiment 1, the dataset comprises 460 images distributed across 9 classes, with training conducted over 10 epochs. Table 5.1 presents a summary of the performance of various CNN models trained on this dataset. Among these models, VGG-19 and MobileNetV2 achieve accuracy rates of 54.35%, demonstrating their effectiveness in the given task. However, EfficientNetB0 produces the least favourable results among the models, with an accuracy rate of 26.09%.

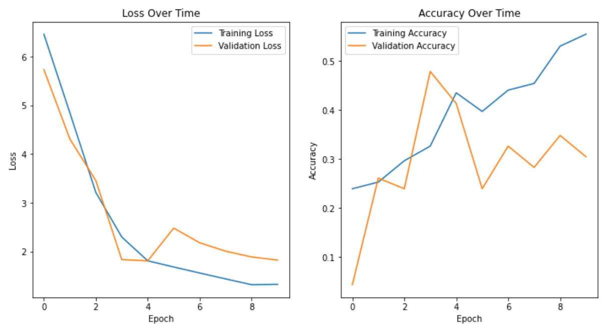
Table 5.1 Experiment 1: CNN Models Performance

CNN MODEL	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
VGG-19	**54.35**	19.34	25.46	21.98
ResNet-101	28.26	17.96	10.42	13.19
InceptionV3	41.30	14.91	17.36	16.04
MobileNetV2	**54.35**	29.26	36.11	32.32
EfficientNetB0	26.09	2.90	11.11	4.60

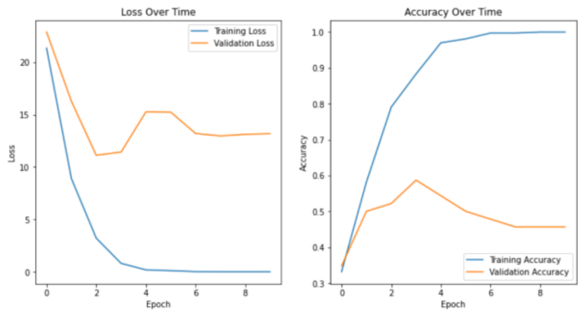
The loss and accuracy plots depicted in Figure 5.1 show that models achieve higher accuracy on the training sets compared to the validation sets. This suggests the models memorise the training data but fail to generalise well to new, unseen data. Since the models are trained on the reduced dataset, it may suggest that the models are too complex for the given dataset size.



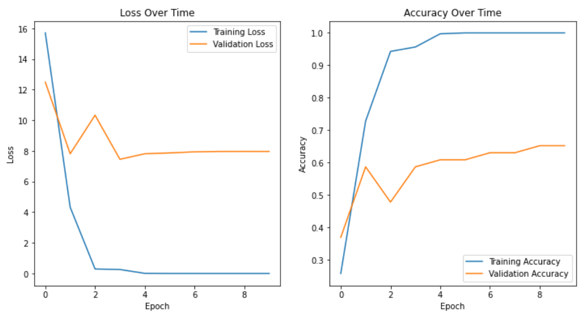
VGG-19



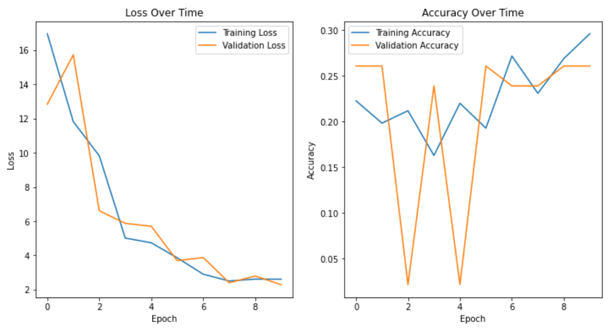
ResNet-101



InceptionV3



MobileNetV2



EfficientNetB0

Figure 5.1 Experiment 1: Loss and Accuracy Plots

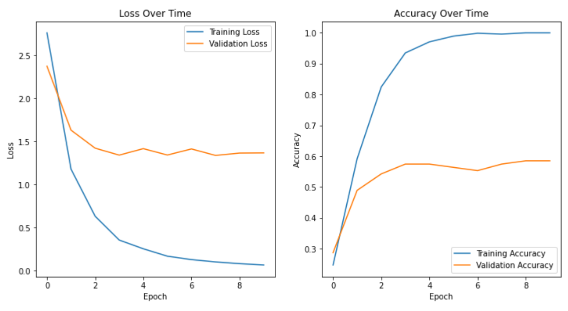
5.1.3 Experiment 2: Refined Dataset

Experiment 2 involves a dataset consisting of 941 images distributed across 9 classes, with training conducted over 10 epochs. Table 5.2 provides an overview of the performance of CNN models trained on this dataset. VGG-19 and MobileNetV2 showcase notable improvements in accuracy compared to Experiment 1, achieving rates of 70.53% and 74.74%, respectively. However, EfficientNetB0 continues to yield suboptimal results with an accuracy rate of 11.58%.

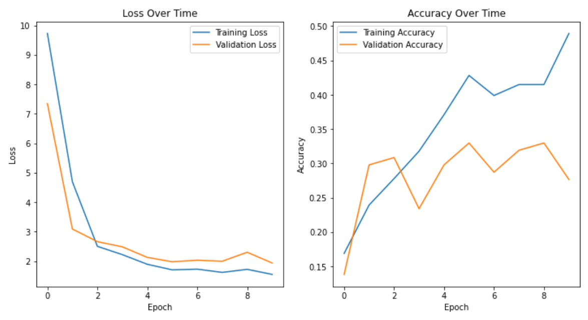
Table 5.2 Experiment 2: CNN Models Performance

CNN MODEL	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
VGG-19	**70.53**	67.57	58.03	62.44
ResNet-101	31.58	20.42	20.40	20.41
InceptionV3	**68.42**	81.94	58.01	67.93
MobileNetV2	**74.74**	83.99	66.43	74.18
EfficientNetB0	11.58	1.29	11.11	2.31

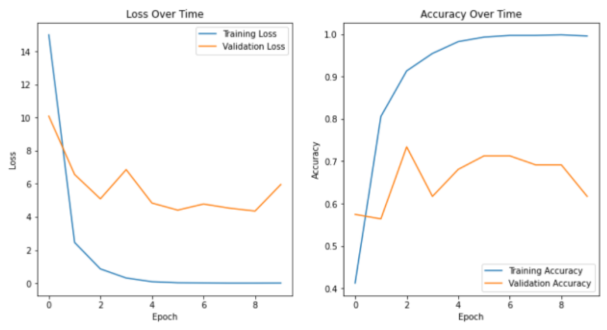
Figure 5.2 illustrates the loss and accuracy plots for Experiment 2. The models exhibit an improvement in validation accuracy compared to Experiment 1. ResNet-101 and EfficientNetB0 demonstrate similar patterns in both training and validation loss, indicating a gradual decrease in loss over the course of training.



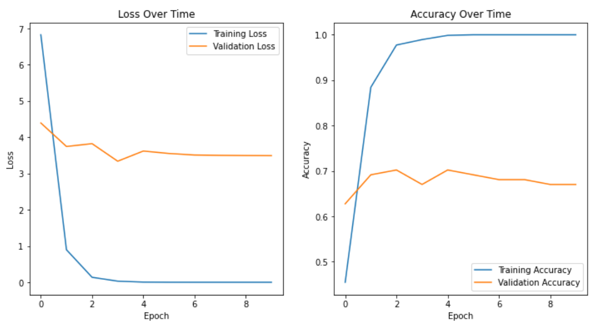
VGG-19



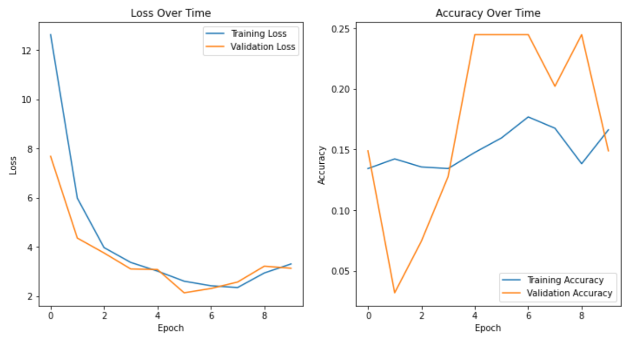
ResNet-101



InceptionV3



MobileNetV2



EfficientNetB0

Figure 5.2 Experiment 2: Loss and Accuracy Plots

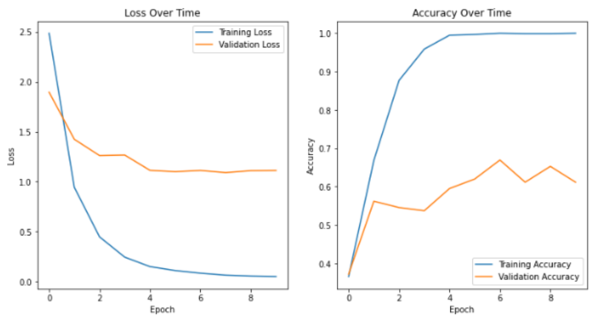
5.1.4 Experiment 3: Augmented Dataset

For Experiment 3, an augmented dataset was created by augmenting the dataset used in Experiment 2. This augmented dataset comprises 1211 images distributed across 9 classes and is used for training over 10 epochs. Table 5.3 provides an overview of the performance of CNN models trained on this augmented dataset. Notably, InceptionV3 outperforms VGG-19, achieving an accuracy rate of 74.59% compared to VGG-19's reduced accuracy of 66.39%. MobileNetV2 also shows improvement, with its accuracy increasing from 74.74% to 76.23%. However, ResNet-101 and EfficientNetB0 exhibit lower accuracies compared to Experiment 2, where the dataset was not augmented. This observation suggests that while data augmentation can enhance the model's generalisation ability by providing it with more diverse examples to learn from, it may also introduce some noise or inconsistencies that could impact performance.

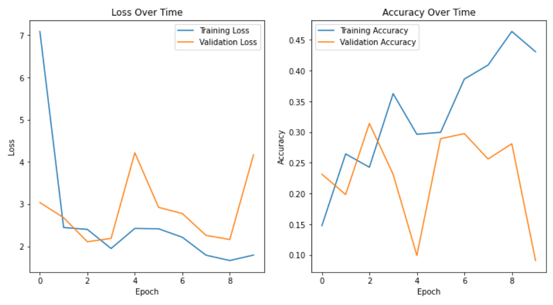
Table 5.3 Experiment 3: CNN Models Performance

CNN MODEL	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
VGG-19	**66.39**	58.14	60.34	59.22
ResNet-101	12.30	23.43	12.83	16.58
InceptionV3	**74.59**	73.01	66.62	69.67
MobileNetV2	**76.23**	79.10	74.39	76.67
EfficientNetB0	9.84	1.09	11.11	1.99

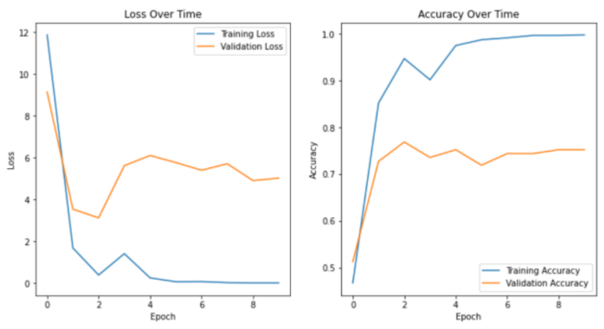
Figure 5.3 shows the loss and accuracy plots for the models in Experiment 3. A decrease in both training and validation losses is observed, indicating that the models are progressively improving their performance over the training epochs. Additionally, there is a clear improvement in validation accuracy compared to previous experiments, suggesting that the models are generalising better to unseen data. This improvement is particularly significant given the absence of data augmentation in Experiment 2.



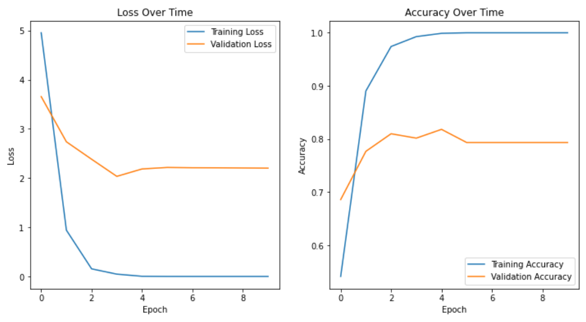
VGG-19



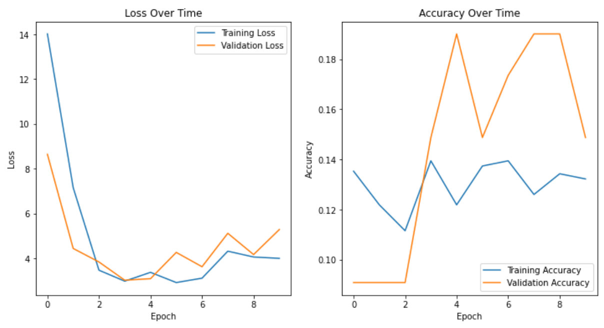
ResNet-101



InceptionV3



MobileNetV2



EfficientNetB0

Figure 5.3 Experiment 3: Loss and Accuracy Plots

5.1.5 Experiment 4: Augmented Dataset with Increased Epochs

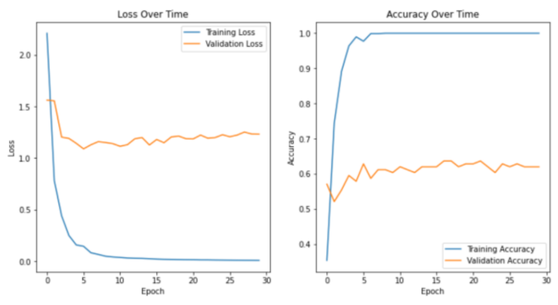
Experiment 4 is designed to investigate the effect of prolonged training on the performance of CNN models using the augmented dataset introduced in Experiment 3. The dataset maintains the same number of images and classes but extends the training duration to 30 epochs. This setup allows for a direct comparison of model performance under different training durations while keeping other experimental conditions constant.

The performance of CNN models on the augmented dataset with increased epochs is summarised in Table 5.4. InceptionV3 achieves an accuracy rate of 73.77%, which is slightly lower compared to the previous accuracy of 74.59% in Experiment 3. MobileNetV2 maintains its high accuracy, with a rate of 77.05%. VGG-19 shows improved performance compared to Experiment 3, but still falls short of the accuracy achieved in Experiment 2. Despite improvements compared to Experiment 3, the performance of ResNet-101 and EfficientNetB0 still falls short of the levels observed with InceptionV3 and MobileNetV2. EfficientNetB0 continues to struggle with low accuracy and other performance metrics, indicating its limited effectiveness in this experimental setup, even with an increased number of epochs.

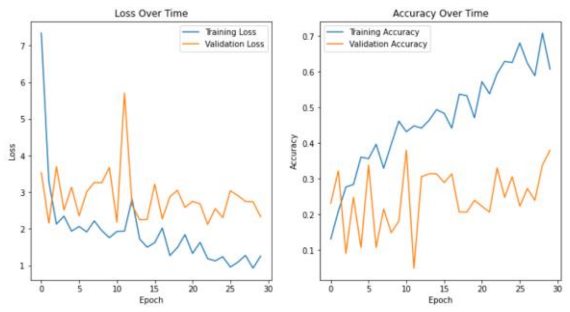
Table 5.4 Experiment 4: CNN Models Performance

CNN MODEL	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
VGG-19	**69.67**	63.93	63.15	63.54
ResNet-101	30.33	37.28	29.26	32.78
InceptionV3	**73.77**	74.79	73.77	73.50
MobileNetV2	**77.05**	74.07	71.43	72.73
EfficientNetB0	11.48	1.28	11.11	2.29

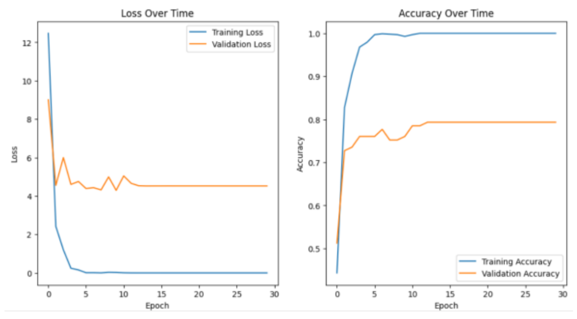
Based on Figure 5.4, InceptionV3 experiences a stagnation in both validation loss and accuracy from the 12th epoch onwards. This suggests that the model may have reached its optimal performance and further training may not significantly improve its results. On the other hand, MobileNetV2 shows a gradual improvement in validation accuracy, albeit at a slower pace from the 11th epoch onwards. This suggests that the model continues to learn and refine its performance even after several epochs.



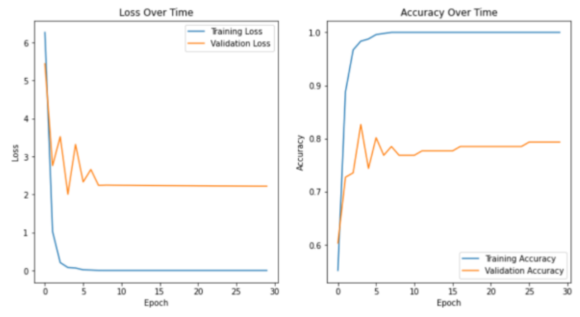
VGG-19



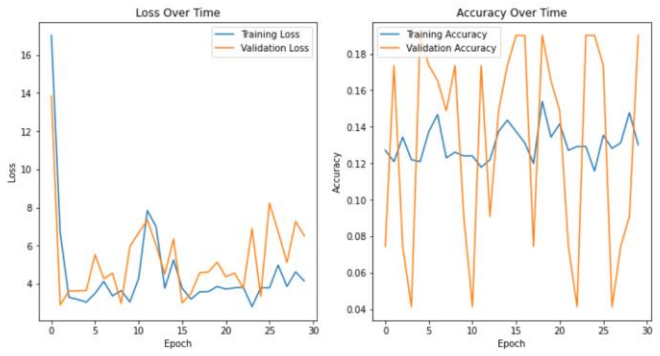
ResNet-101



InceptionV3



MobileNetV2



EfficientNetB0

Figure 5.4 Experiment 4: Loss and Accuracy Plots

5.1.6 Experiment 5: Random Classes Dataset

Experiment 5 introduces a dataset comprising random classes to assess the CNN models' performance when trained on a dataset with distinct plant genera. The dataset consists of 957 images distributed across 9 randomly selected classes and is trained over 10 epochs. Performance evaluation is conducted using the same metrics as in previous experiments.

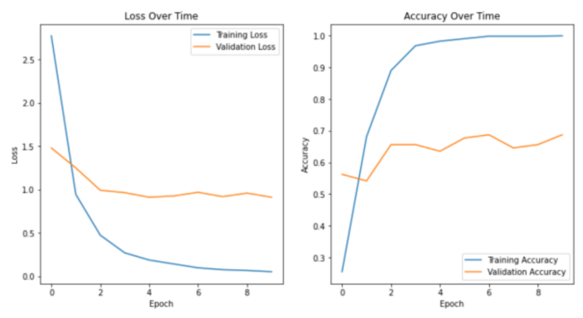
Table 5.5 summarises the performance of CNN models on the random classes dataset. InceptionV3 and MobileNetV2 demonstrate the highest accuracy rates of 79.17% and 81.25%, respectively. Both models also exhibit high precision, recall, and F1-scores, indicating their robust performance across various settings. VGG-19 shows slightly better performance compared to Experiment 3, where the dataset comprises similar plant genera, but performs worse than in Experiment 4, where the epochs were increased.

Table 5.5 Experiment 5: CNN Models Performance

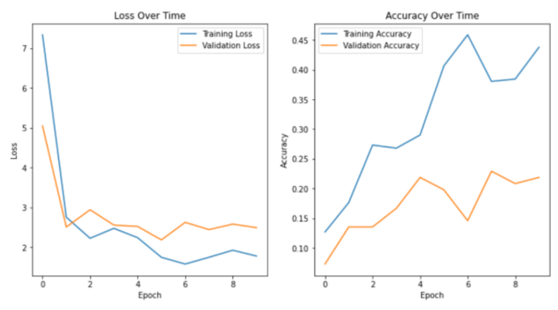
CNN MODEL	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
VGG-19	**67.71**	69.62	67.43	68.51
ResNet-101	33.33	32.86	27.95	31.02
InceptionV3	**79.17**	78.21	75.56	76.86
MobileNetV2	**81.25**	81.90	81.66	81.78
EfficientNetB0	18.75	2.08	11.11	3.51

Figure 5.5 depicts slight fluctuating trends in validation accuracy. Despite these fluctuations, the average validation accuracy across all epochs surpasses that of previous experiments. This

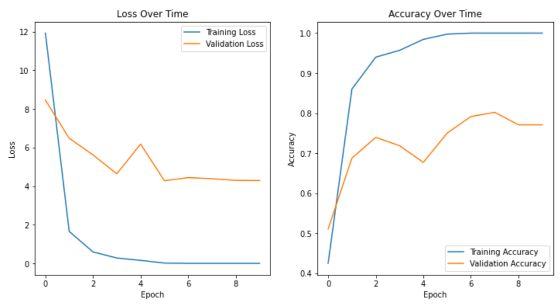
indicates that although the models may encounter temporary setbacks in performance during certain epochs, they consistently achieve higher overall accuracy compared to earlier experiments.



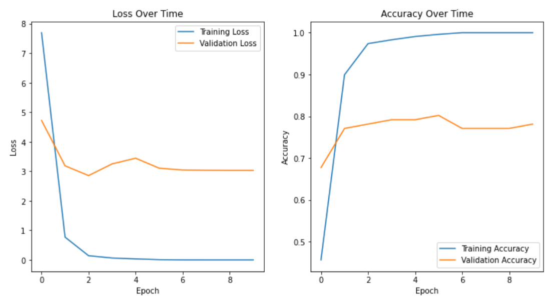
VGG-19



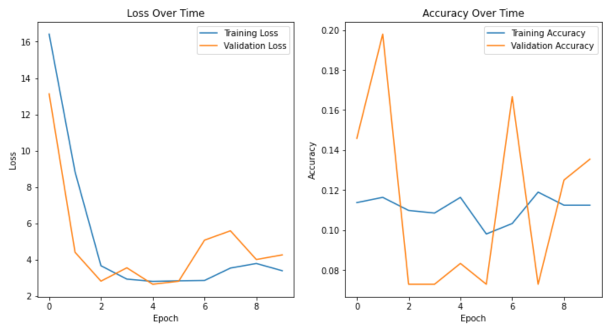
ResNet-101



InceptionV3



MobileNetV2



EfficientNetB0

Figure 5.5 Experiment 5: Loss and Accuracy Plots

Overall, MobileNetV2 emerges as the top-performing model across all experiments, demonstrating its reliability and consistency across various datasets and experimental conditions.

5.2 Image Segmentation Results

The segmentation results obtained from applying various thresholding techniques such as Otsu's thresholding, Adaptive Mean thresholding, Adaptive Gaussian thresholding, and Rosin thresholding are presented. In each thresholding method, three experiments are conducted to explore distinct segmentation strategies. The first experiment involves applying normal thresholding and treating the entire image as a single entity. In the second experiment, the image is divided into four segments, and thresholding is independently applied to each before recombining them. The third experiment integrates NDI into the thresholding process to highlight specific regions of interest, such as vegetation.

5.2.1 Image Segmentation Implementation

The segmentation process involves multiple image processing techniques implemented using the OpenCV library, as visualised in Figure 5.6. After loading the raw images using the PIL library and storing them in NumPy arrays, preprocessing steps include resizing, normalisation, and blurring using a bilateral filter. The bilateral filter is applied with a *d* value of 9, which determines the size of the pixel neighbourhood considered during filtering. Additionally, *sigmaColor* and *sigmaSpace* are set to 75, controlling colour similarity and spatial proximity, respectively. Contrast enhancement is then performed using CLAHE, with a *clipLimit* value of 0.01 limiting the maximum contrast enhancement allowed in each tile of the image. The *tileGridSize* of (8, 8) specifies the size of the grid used for histogram equalisation, affecting the localised contrast adjustments. After applying morphology closing with a kernel size of (5, 5) to refine the image by smoothing and closing gaps between object boundaries, thresholding is performed.

After thresholding, contour detection is performed by using the *cv2.RETR_EXTERNAL* flag to retrieve only the outer contours outlining objects in the image. The *cv2.CHAIN_APPROX_NONE* parameter stores all contour points without simplifying or approximating their shapes. The detected contours are then overlaid onto a copy of the original image and drawn in green with a thickness of 3 pixels. To segment the image, an empty mask of the same size as the original image is initialised as a single-channel array of zeros. This mask is filled with white pixels (pixel value 255) within the interiors of the contours, while the background remains black (pixel value 0). This filled contour mask is then applied to the original image, preserving only the areas where the mask is non-zero and effectively segmenting the original image. Finally, the segmented image is converted to an 8-bit unsigned integer representation using the scikit-image library, enabling further analysis and visualisation.

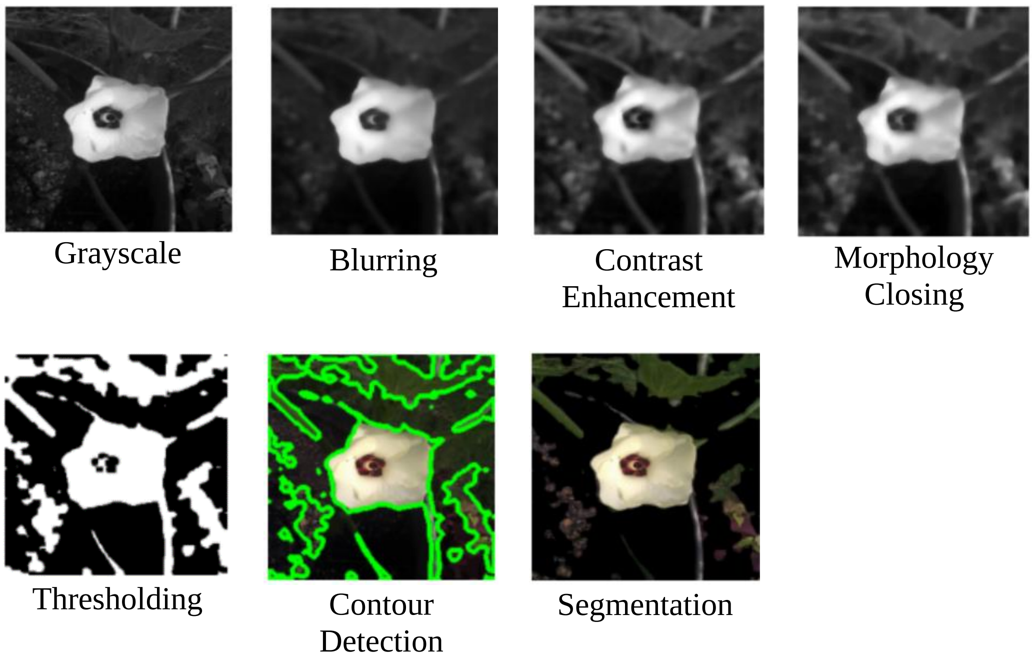


Figure 5.6 Image Segmentation Process

5.2.2 Otsu’s Thresholding

Otsu’s thresholding algorithm is applied using the *cv2.threshold()* function with the *cv2.THRESH_BINARY* + *cv2.THRESH_OTSU* flags. This method automatically calculates an optimal threshold value based on the image histogram. Following thresholding, contour filtering is performed to refine the results. The area threshold, set to 1000, determines the minimum contour area required for inclusion, while the proximity threshold of 100 controls the proximity of contours to a specified centre point. Contours with areas below the area threshold or located farther than the proximity threshold defined are excluded.

The results of the Otsu’s thresholding experiments are presented in Figures 5.7 to 5.9. When applied to the entire image, Otsu’s thresholding demonstrates the most promising results compared to the other two experiments, although it struggles to accurately differentiate foreground and background for some images due to their complex background. Otsu combined with NDI yields some notably well-segmented images, with certain images showing perfectly extracted regions, such as the green parts of the plant (e.g., leaves). However, flowers are not detected in some instances.

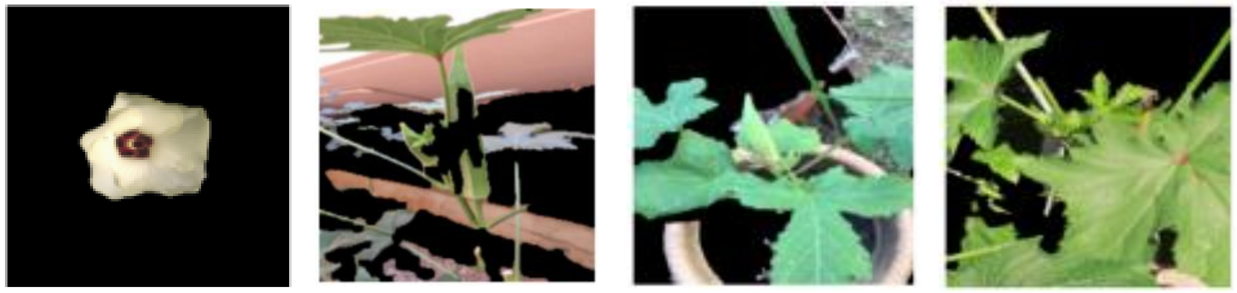


Figure 5.7 Normal Otsu

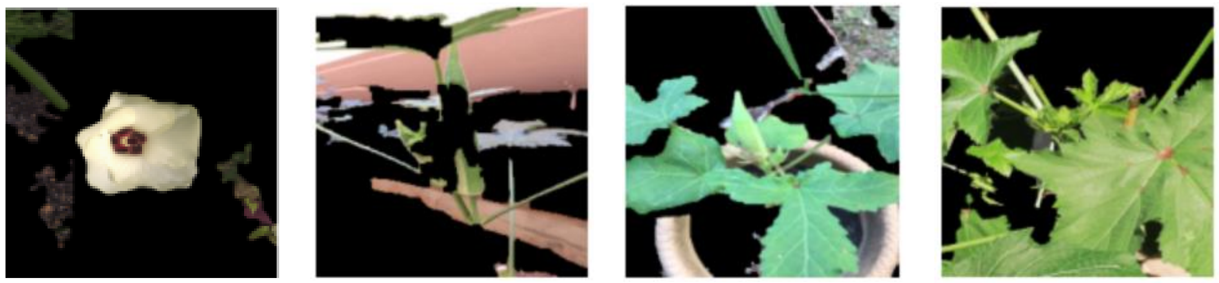


Figure 5.8 Otsu with image splitting

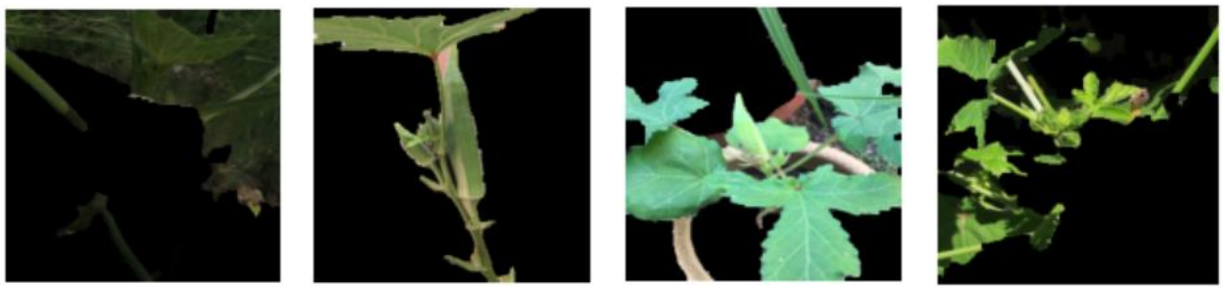


Figure 5.9 Otsu with NDI

5.2.3 Adaptive Mean Thresholding

The Adaptive Mean thresholding method is applied using the *cv2.adaptiveThreshold()* function provided by the OpenCV library. *cv2.ADAPTIVE_THRESH_MEAN_C* indicates the method for adaptive thresholding using the mean of the neighbourhood area while *cv2.THRESH_BINARY* specifies the binary thresholding type where pixel values are thresholded based on a binary criterion. A local mean value for each pixel is calculated based on its neighbouring pixels, with the neighbourhood size set as 199. A constant offset of 5 is then subtracted from the calculated mean to fine-tune the thresholding result.

The results of the Adaptive Mean thresholding experiments are presented in Figures 5.10 to 5.12. Adaptive Mean thresholding demonstrates better performance compared to Otsu's method in accurately differentiating foreground and background objects. When incorporating NDI, Adaptive Mean thresholding proves to be more effective in the detection of flowers as compared to Otsu.

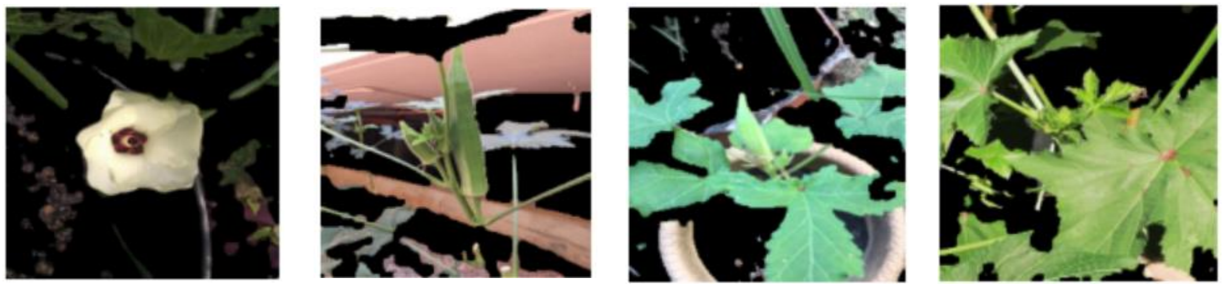


Figure 5.10 Normal Adaptive Mean

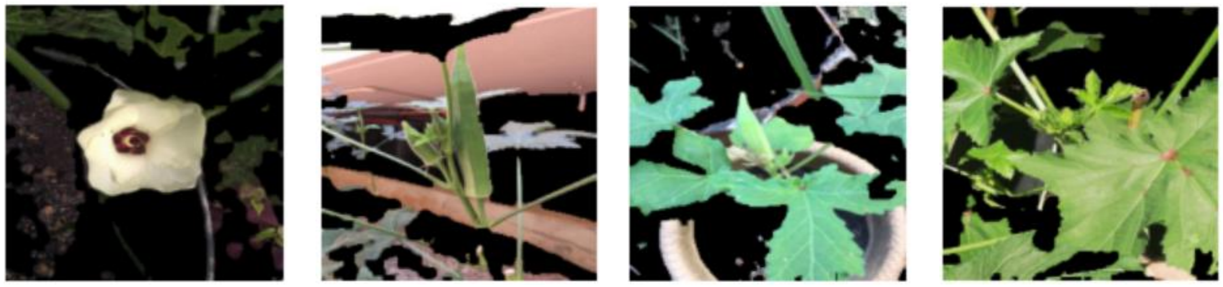


Figure 5.11 Adaptive Mean with image splitting

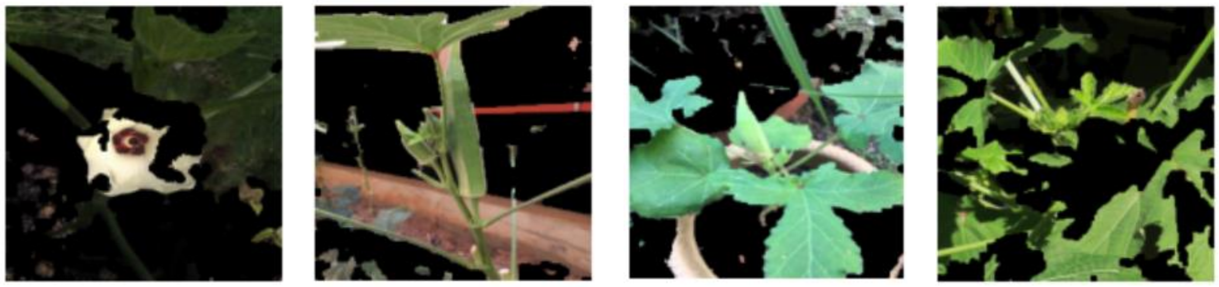


Figure 5.12 Adaptive Mean with NDI

5.2.4 Adaptive Gaussian Threshold

The Adaptive Gaussian thresholding method is applied using the same function and binary thresholding type as Adaptive Mean. The method for adaptive thresholding is *cv2.ADAPTIVE_THRESH_GAUSSIAN_C*, which uses the Gaussian-weighted sum of the neighbourhood area. The neighbourhood size set as 299 and the constant offset is set to 5.

Based on the results of the Adaptive Gaussian thresholding experiments, as presented in Figures 5.13 to 5.15, it is observed that the application of Adaptive Gaussian thresholding with NDI integration slightly outperforms the results obtained from Adaptive Mean with NDI integration.

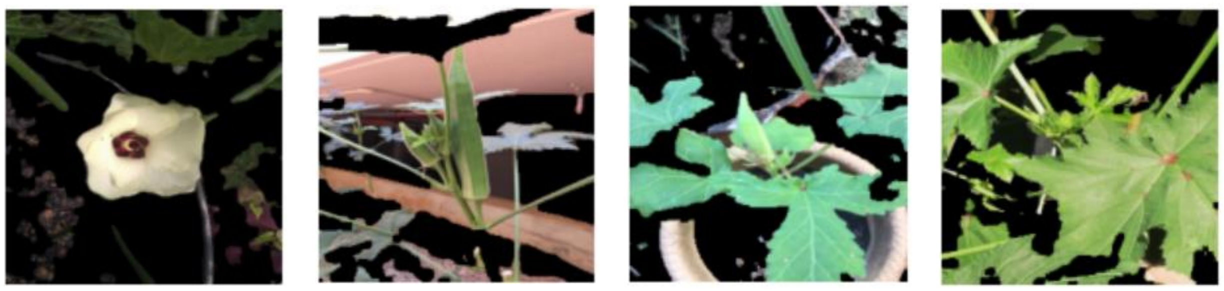


Figure 5.13 Normal Adaptive Gaussian

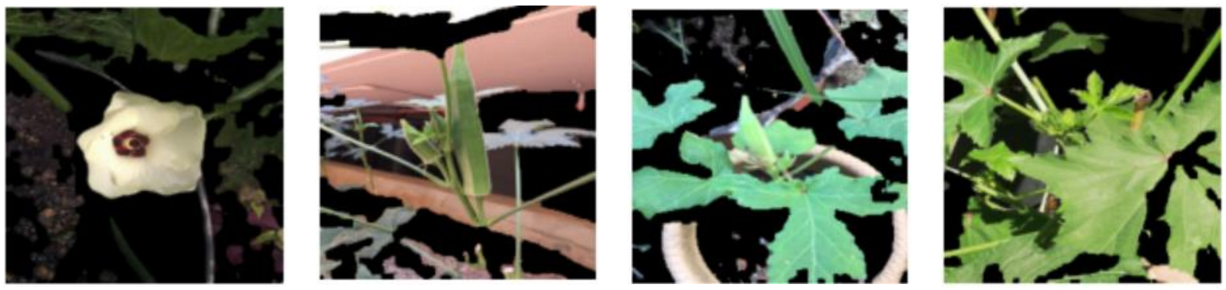


Figure 5.14 Adaptive Gaussian with image splitting

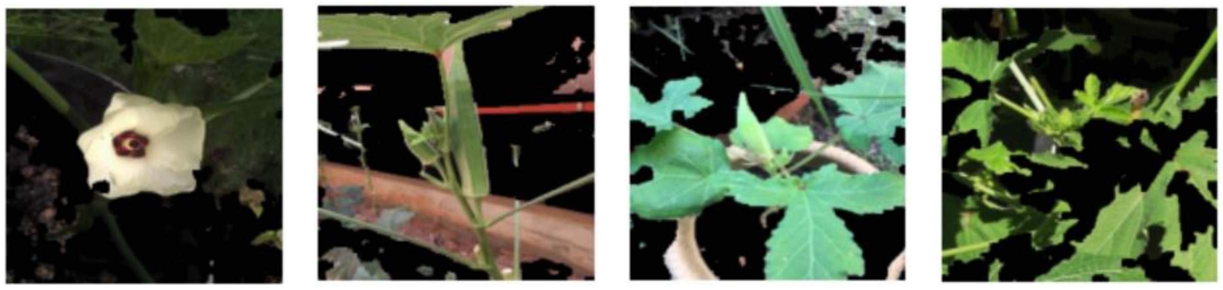


Figure 5.15 Adaptive Gaussian with NDI

5.2.5 Rosin Thresholding

Rosin thresholding iteratively calculates an optimal threshold value based on the image histogram. The convergence threshold *k*, set to 1, determines the stopping criterion for the iterative process and ensures convergence within a specified tolerance. This iterative approach refines the threshold value until convergence is achieved, effectively separating the foreground from the background in the image. Similar to the previous two adaptive thresholding algorithms, the same function and binary thresholding type are used to generate a binary image based on the computed threshold.

The segmentation results of the Rosin thresholding algorithm, depicted in Figures 5.16 to 5.18, exhibit similar outcomes to those observed in previous thresholding methods.

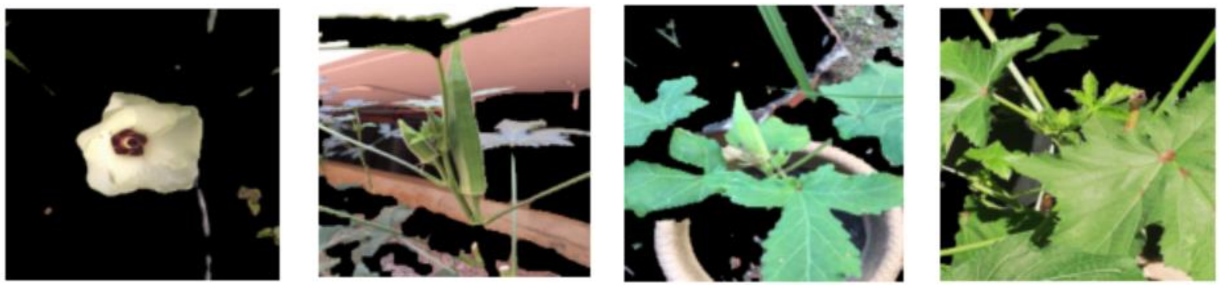


Figure 5.16 Normal Rosin

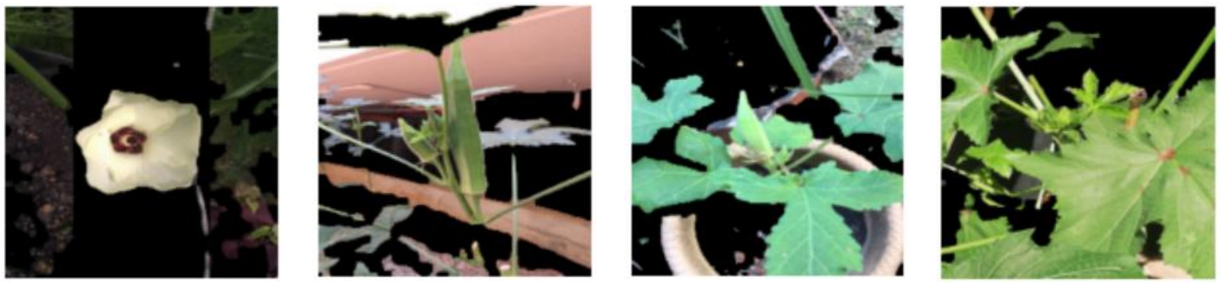


Figure 5.17 Rosin with image splitting

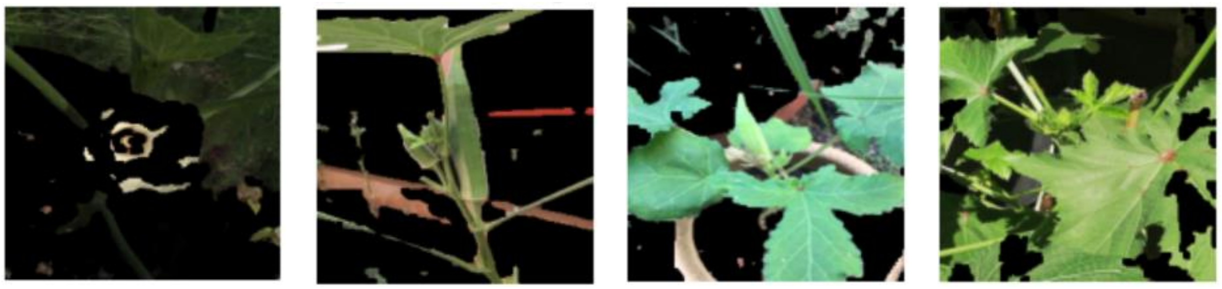


Figure 5.18 Rosin with NDI

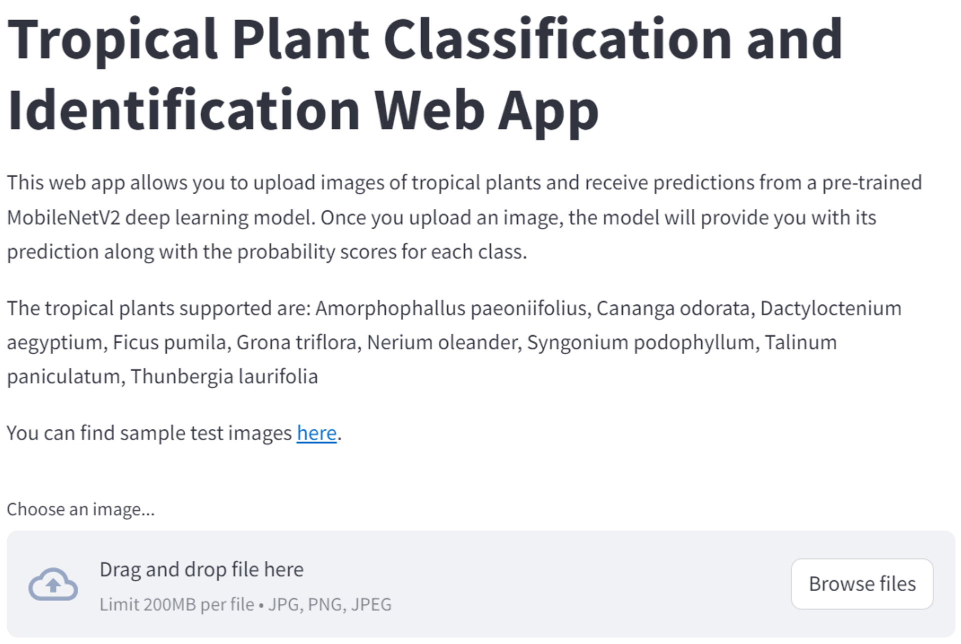
5.3 Web Application Integration

The model with the best performance on the testing set, MobileNetV2, is integrated into a web application developed using Streamlit. Initially, the model and corresponding class labels are loaded into the web application. Upon user interaction, uploaded images undergo preprocessing, including resizing to 224x224 pixels and normalisation, to meet the input requirements of the MobileNetV2 model. The preprocessed images, converted into NumPy arrays, are fed into the MobileNetV2 model for inference.

During the inference process, the model predicts the most likely label for the uploaded image along with probability scores for each class. These predictions are then presented to the user within the web application's interface, displaying the predicted label and a tabulated view of the predicted probabilities for each class. This enhances the interpretability of the results, with probabilities rounded to two decimal points for clarity.

For deployment, the application is hosted on the Streamlit Community Cloud platform. This enables efficient utilisation of the model for image classification tasks, enhancing user interaction and accessibility. The deployed application can be accessed via the following link: https://plantapp-hfylh2.streamlit.app.

The user interface design of the web application and the model’s predicted output are presented in Figure 5.19 and Figure 5.20.


Tropical Plant Classification and Identification Web App

This web app allows you to upload images of tropical plants and receive predictions from a pre-trained MobileNetV2 deep learning model. Once you upload an image, the model will provide you with its prediction along with the probability scores for each class.

The tropical plants supported are: Amorphophallus paeoniifolius, Cananga odorata, Dactyloctenium aegyptium, Ficus pumila, Grona triflora, Nerium oleander, Syngonium podophyllum, Talinum paniculatum, Thunbergia laurifolia

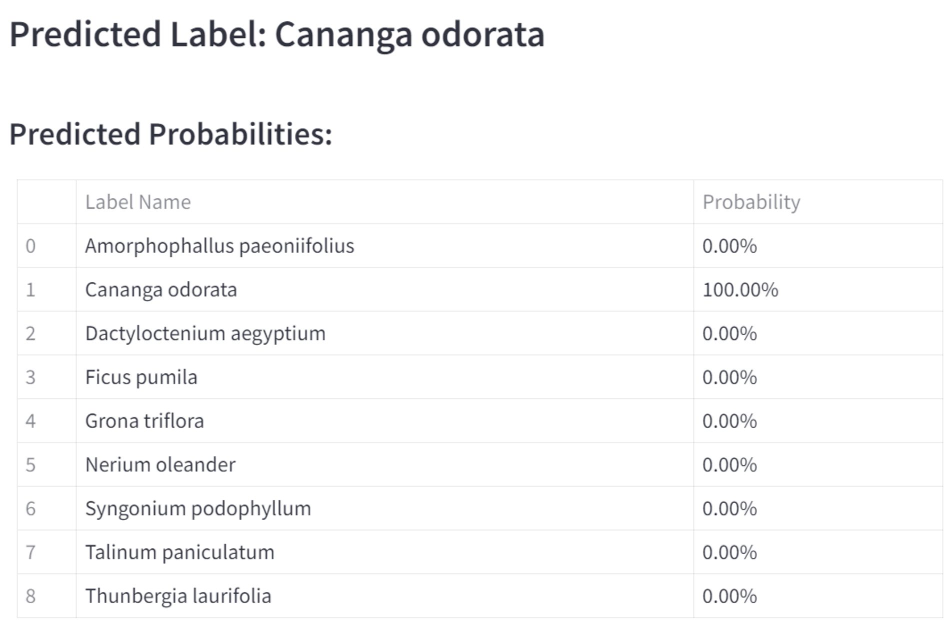
You can find sample test images here.

Choose an image...

Drag and drop file here

Limit 200MB per file • JPG, PNG, JPEG

Figure 5.19 Web Application User Interface



Predicted Label: Cananga odorata

Predicted Probabilities:

	Label Name	Probability
0	Amorphophallus paeoniifolius	0.00%
1	Cananga odorata	100.00%
2	Dactyloctenium aegyptium	0.00%
3	Ficus pumila	0.00%
4	Grona triflora	0.00%
5	Nerium oleander	0.00%
6	Syngonium podophyllum	0.00%
7	Talinum paniculatum	0.00%
8	Thunbergia laurifolia	0.00%

Figure 5.20 Web Application Model Predicted Output

6. Analysis and Insights

6.1 Relationship Between Class Size and Model Performance

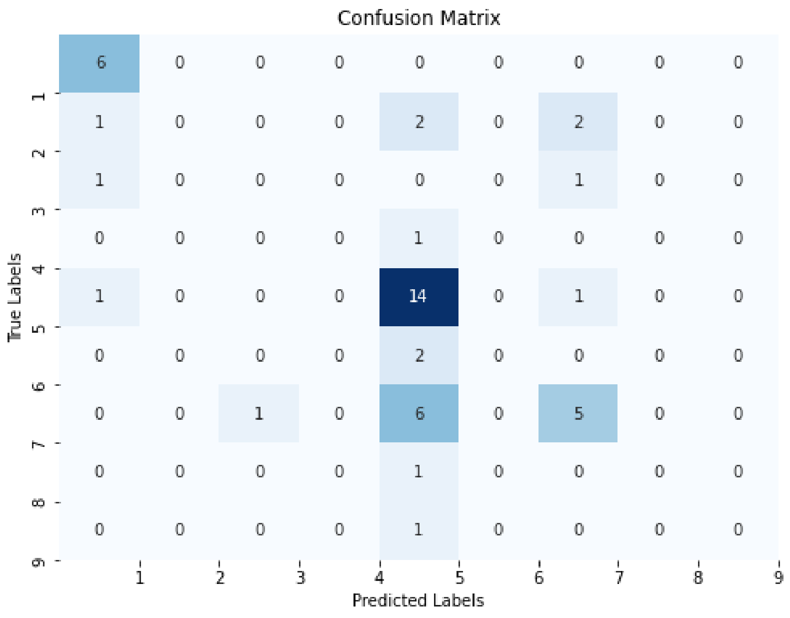


Figure 6.1 Confusion Matrix for VGG-19 from Experiment 1

Based on the confusion matrix of Experiment 1's VGG-19 model in Figure 6.1, a thorough examination of the model's performance across the nine classes reveals a correlation between class size and accurate predictions. Among the nine classes, two classes, each containing equal or more than 75 images, demonstrated a high accuracy rate exceeding 80%. This observation suggests that a larger class size contributes to better model performance. In contrast, classes with smaller sample sizes, each containing fewer than 40 images, demonstrate zero correct predictions. This underscores the limitations of small class sizes, emphasising the need for an adequate number of images per class for effective model training.

Additionally, despite having over 100 images, Class 7 exhibits a relatively low accuracy rate of 40%. It is observed that the model frequently misclassifies Class 7 as Class 5. Further investigation reveals that both Class 5 and Class 7 belong to the same genus, indicating potential challenges in distinguishing between closely related classes. The implications of such relationships between classes from the same genera are elaborated upon in Section 6.2.

In summary, the analysis indicates that a minimum class size of at least 75 images is beneficial for training robust models. Larger class sizes contribute to improved model generalisation, reducing the risk of limited representation and poor performance in smaller classes. To address this issue, data augmentation techniques should be employed for classes with fewer images, ensuring a minimum of 75 images per class. To control the extent of augmentation and maintain dataset balance, each image should only undergo augmentation a maximum of three times. Consequently, classes with fewer than 25 images should be excluded to maintain dataset integrity and prevent over-augmentation.

6.2 Relationship Between Plant Genera Similarity and Model Performance

In Section 6.1, the analysis highlights a misclassification between two classes, which are Acacia auriculiformis and Acacia mangium, respectively, despite having an adequate number of images for model training. This misclassification can be attributed to the similarity between the two classes, which belong to the same genus. While they share similarities in appearance, such as leaf shape and tree structure, they differ in flower colour, with Acacia auriculiformis having yellow flowers and Acacia mangium having white flowers, as shown in Figure 6.2 and Figure 6.3.

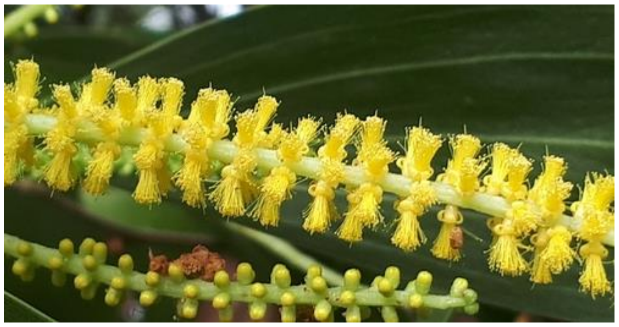


Figure 6.2 Acacia auriculiformis

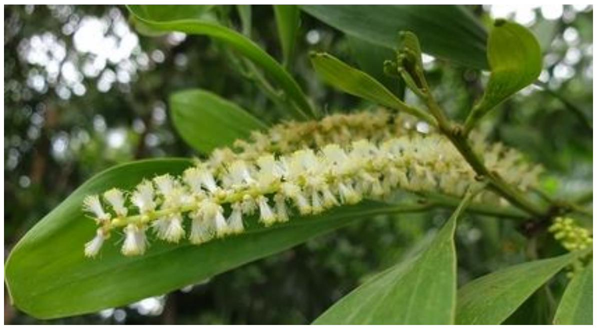


Figure 6.3 Acacia mangium

This observation suggests that classes from the same genus pose challenges for the model due to their close resemblance. Despite efforts to distinguish between these classes based on features like flower colour, the model struggles to differentiate them accurately, leading to misclassifications. This highlights the importance of considering plant genera similarity in model training and suggests the need for additional features or techniques to improve classification accuracy for closely related classes.

6.3 Analysis of Thresholding-Based Image Segmentation for Plant Images

The challenges encountered in thresholding-based image segmentation, particularly with plant images, stem from the complexity of plant anatomy and the variability of image characteristics.

The intricate nature of plant anatomy, including variations in colour, texture, and shape, presents difficulties for thresholding methods to determine plant regions accurately. Moreover, occlusions, overlapping structures, and irregular backgrounds, such as dense foliage or intricate plant architectures, further complicate the segmentation process. These factors collectively contribute to incomplete or inaccurate segmentations, where thresholding methods struggle to distinguish foreground from background, resulting in segmentation errors.

To address these issues, alternative segmentation methods tailored for plant images can be explored. Machine learning-based approaches, such as deep learning-based segmentation networks like U-Net and Mask R-CNN, offer more flexibility in capturing complex image features and may potentially yield better segmentation results [49]. These models can be trained on annotated plant image datasets to learn to segment plants accurately. Additionally, algorithms like GrabCut, which iteratively refine segmentation boundaries based on user inputs, provide a more precise delineation of object boundaries, offering an alternative to traditional thresholding techniques [50].

To quantitatively assess the performance of segmentation methods and identify areas for improvement, several performance metrics can be considered, such as Intersection over Union (IoU), dice coefficient, precision and recall, and F1-score. IoU and dice coefficient measure the overlap between the segmented region and the ground truth region, providing insights into segmentation accuracy. Precision and recall evaluate the trade-off between correctly segmented regions and missed regions, while the F1-score provides a balanced measure of segmentation accuracy [51].

7. Contributions and Future Works

7.1 Summary of Contributions

The key contributions of this dissertation include a comprehensive exploration of CNN models' performance, an in-depth investigation of thresholding-based image segmentation techniques, integration of a model into a web application, and the curation of a dataset specific to tropical plants. Through a series of experiments, the study evaluates the performance of CNN models trained on different datasets, providing valuable insights into how dataset characteristics and augmentation techniques influence model performance. Challenges such as the relationship between class size, plant genera similarity, and model performance are highlighted, as well as issues related to small class sizes and similar plant genera. Additionally, the effectiveness of thresholding-based image segmentation methods for plant images is analysed, with limitations identified and solutions proposed for further exploration and improvement.

Overall, the project successfully achieves its aims and objectives, contributing valuable knowledge and insights to the fields of plant image analysis and machine learning.

7.2 Future Work

Several key areas can be explored to further enhance the effectiveness and scope of the study. Since models tend to perform better with larger datasets, future work involves fully utilising all the data in the curated tropical plant dataset as compared to the current nine classes. With advancements in computational resources, such as more powerful supercomputers and increased RAM, processing larger datasets becomes more feasible, potentially leading to improved model performance. Furthermore, more advanced augmentation techniques, such as Generative Adversarial Network (GAN) architectures, can be explored to further diversify and enhance the realism of generated images.

Another potential direction for future work is to explore strategies for overcoming overfitting in models. Dropout layers can be implemented to introduce randomness by randomly deactivating neurons during training and preventing the network from relying too heavily on specific neurons and features. Additionally, weight decay, also known as L2 regularisation, can impose penalties on large weights in the network, encouraging the model to prioritise smaller weights and smoother decision boundaries. Furthermore, early stopping can serve as a proactive measure against overfitting by terminating the training process when the model’s performance on a separate validation dataset begins to deteriorate. By monitoring the model’s performance on the validation set, early stopping prevents the model from becoming overly specialised to the training data and facilitates the learning of more generalisable features [52].

To address challenges related to similarities between genera, incorporating additional features beyond visual characteristics, such as texture patterns or leaf venation, can provide supplementary information to improve classification accuracy. Fine-tuning the model architecture or adjusting training parameters specifically for classes with high similarity may also optimise the model’s ability to discern between them.

Additionally, extending the functionality of the web application to mobile platforms can enhance accessibility and usability. Optimising the user interface and implementing features tailored for mobile use will improve the overall user experience and reach a wider audience.

7.3 Reflection

Reflecting on the project, several challenges were encountered that influenced the direction and outcomes of the study. One of the primary challenges was the gathering of datasets, particularly the lack of datasets specific to tropical plants. The process of curating a suitable dataset was time-consuming and required extensive effort, leading to delays in experimentation.

Resource constraints presented another significant obstacle, particularly regarding computational resources for model training. Limited access to high-performance computing infrastructure and computational resources resulted in prolonged training times and restricted the exploration of larger and more complex models. These constraints not only impacted the efficiency of the research process but also restricted the scope and scale of the experiments conducted.

However, while the project encountered obstacles and constraints, all the project’s aims and objectives were achieved, leading to the successful completion of the project.

8. Bibliography

[1] P. G. M. Sobha and P. A. Thomas, “Deep Learning for Plant Species Classification Survey,” in *2019 International Conference on Advances in Computing, Communication and Control (ICAC3)*, IEEE, Dec. 2019, pp. 1–6. doi: 10.1109/ICAC347590.2019.9036796.

[2] P. Kumar Thella and V. Ulagamuthalvi, “A Brief Review on Methods for Classification of Medical Plant Images,” in *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, IEEE, Apr. 2021, pp. 1330–1333. doi: 10.1109/ICCMC51019.2021.9418380.

[3] L. Chang and X. Ding, “Research on classification and recognition method of plant leaves based on deep learning,” in *2022 International Symposium on Advances in Informatics, Electronics and Education (ISAIEE)*, IEEE, Dec. 2022, pp. 152–155. doi: 10.1109/ISAIEE57420.2022.00039.

[4] K. Suwais, K. Alheeti, and D. Al_Dosary, “A Review on Classification Methods for Plants Leaves Recognition,” *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 2, 2022, doi: 10.14569/IJACSA.2022.0130211.

[5] V. K. Gajjar, A. K. Nambisan, and K. L. Kosbar, “Plant Identification in a Combined-Imbalanced Leaf Dataset,” *IEEE Access*, vol. 10, pp. 37882–37891, 2022, doi: 10.1109/ACCESS.2022.3165583.

[6] A. Nayak, S. Chakraborty, and D. K. Swain, “Application of smartphone-image processing and transfer learning for rice disease and nutrient deficiency detection,” *Smart Agricultural Technology*, vol. 4, p. 100195, Aug. 2023, doi: 10.1016/j.atech.2023.100195.

[7] R. T. Corlett, “Plant diversity in a changing world: Status, trends, and conservation needs,” *Plant Divers*, vol. 38, no. 1, pp. 10–16, Feb. 2016, doi: 10.1016/j.pld.2016.01.001.

[8] J. W. F. Slik *et al.*, “An estimate of the number of tropical tree species,” *Proceedings of the National Academy of Sciences*, vol. 112, no. 24, pp. 7472–7477, Jun. 2015, doi: 10.1073/pnas.1423147112.

[9] W. K. Cornwell, W. D. Pearse, R. L. Dalrymple, and A. E. Zanne, “What we (don’t) know about global plant diversity,” *Ecography*, vol. 42, no. 11, pp. 1819–1831, Nov. 2019, doi: 10.1111/ecog.04481.

[10] S. R. and Girija, “Introduction to Deep Learning and its related case studies,” in *2022 International Interdisciplinary Humanitarian Conference for Sustainability*

(IIHC), IEEE, Nov. 2022, pp. 1097–1103. doi: 10.1109/IIHC55949.2022.10060089.

[11] M. Reda, R. Suwwan, S. Alkafri, Y. Rashed, and T. Shanableh, “AgroAId: A Mobile App System for Visual Classification of Plant Species and Diseases Using Deep Learning and TensorFlow Lite,” *Informatics*, vol. 9, no. 3, p. 55, Jul. 2022, doi: 10.3390/informatics9030055.

[12] Y. Sun, Y. Liu, G. Wang, and H. Zhang, “Deep Learning for Plant Identification in Natural Environment,” *Comput Intell Neurosci*, vol. 2017, pp. 1–6, 2017, doi: 10.1155/2017/7361042.

[13] R. Swathika, S. Srinidhi., N. Radha, and K. Sowmya., “Disease Identification in paddy leaves using CNN based Deep Learning,” in *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, IEEE, Feb. 2021, pp. 1004–1008. doi: 10.1109/ICICV50876.2021.9388557.

[14] M. F. X. Cham, R. Tanone, and H. A. T. Riadi, “Identification of Rice Leaf Disease Using Convolutional Neural Network Based on Android Mobile Platform,” in *2021 2nd International Conference on Innovative and Creative Information Technology (ICITech)*, IEEE, Sep. 2021, pp. 140–144. doi: 10.1109/ICITech50181.2021.9590188.

[15] T. Akiyama, Y. Kobayashi, Y. Sasaki, K. Sasaki, T. Kawaguchi, and J. Kishigami, “Mobile Leaf Identification System using CNN applied to plants in Hokkaido,” in *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*, IEEE, Oct. 2019, pp. 324–325. doi: 10.1109/GCCE46687.2019.9015298.

[16] M. Gehlot and M. L. Saini, “Analysis of Different CNN Architectures for Tomato Leaf Disease Classification,” in *2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, IEEE, Dec. 2020, pp. 1–6. doi: 10.1109/ICRAIE51050.2020.9358279.

[17] A. P. K, S. K. T S, T. A. S, and P. A, “Identification of Indian Medicinal Plants from Leaves using Transfer Learning Approach,” in *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*, 2021, pp. 980–987. doi: 10.1109/ICOEI51242.2021.9452917.

[18] P. Sivakumar, N. S. R. Mohan, P. Kavya, and P. V. S. Teja, “Leaf Disease Identification: Enhanced Cotton Leaf Disease Identification Using Deep CNN Models,” in *2021 IEEE International Conference on Intelligent Systems, Smart and Green Technologies (ICISSGT)*, 2021, pp. 22–26. doi: 10.1109/ICISSGT52025.2021.00016.

[19] A. Hussain, B. Barua, A. Osman, R. Abozariba, and A. T. Asyhari, “Performance of MobileNetV3 Transfer Learning on Handheld Device-based Real-Time Tree Species Identification,” in *2021 26th International Conference on Automation and Computing (ICAC)*, 2021, pp. 1–6. doi: 10.23919/ICAC50006.2021.9594222.

[20] P. G. Shambharkar and S. Sharma, “Plant Disease Detection And Prevention Using Deep Learning,” in *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2023, pp. 1832–1838. doi: 10.1109/ICACCS57279.2023.10112733.

[21] C. Garcin *et al.*, “Pl@ntNet-300K: a plant image dataset with high label ambiguity and a long-tailed distribution,” in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [Online]. Available: https://openreview.net/forum?id=eLYinD0TtIt

[22] “iNaturalist,” iNaturalist. Accessed: Dec. 20, 2023. [Online]. Available: https://www.inaturalist.org

[23] M. Murat, S.-W. Chang, A. Abu, H. J. Yap, and K.-T. Yong, “Automated classification of tropical shrub species: a hybrid of leaf shape and machine learning approach,” *PeerJ*, vol. 5, p. e3792, Sep. 2017, doi: 10.7717/peerj.3792.

[24] R. Ahila Priyadharshini, S. Arivazhagan, and M. Arun, “Ayurvedic Medicinal Plants Identification: A Comparative Study on Feature Extraction Methods,” 2021, pp. 268–280. doi: 10.1007/978-981-16-1092-9_23.

[25] J. M. Gumiran, A. F. Fajardo, and R. P. Medina, “Enhanced Threshold-based Segmentation for Maize Plantation,” in *2022 7th International Conference on Communication, Image and Signal Processing (CCISP)*, 2022, pp. 284–290. doi: 10.1109/CCISP55629.2022.9974289.

[26] Z. Yuan *et al.*, “Image Recognition Study on Leaf Characteristics of Medicinal Plants Based on Otsu Dynamic Threshold Segmentation Algorithm,” in *2022 IEEE 2nd International Conference on Mobile Networks and Wireless Communications (ICMNWC)*, 2022, pp. 1–7. doi: 10.1109/ICMNWC56175.2022.10031938.

[27] Y. Wang, L. Wang, N. Tuerxun, L. Luo, C. Han, and J. Zheng, “Extraction of Jujube Planting Areas in Sentinel-2 Image Based on NDVI Threshold—a case study of Ruoqiang County,” in *2022 29th International Conference on Geoinformatics*, 2022, pp. 1–6. doi: 10.1109/Geoinformatics57846.2022.9963828.

[28] E. Zemmour, P. Kurtser, and Y. Edan, “Automatic Parameter Tuning for Adaptive Thresholding in Fruit Detection,” *Sensors*, vol. 19, no. 9, 2019, doi: 10.3390/s19092130.

[29] H. Goëau, P. Bonnet, and A. Joly, "Overview of LifeCLEF Plant Identification Task 2019: diving into Data Deficient Tropical Countries," in *Conference and Labs of the Evaluation Forum*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:198488668

[30] J. P. C. Mirandilla, C. B. Bating, M. K. Cabatuan, and J. A. C. Jose, "Classification of Philippine Herbal Medicine Plant Using EfficientNet on Mobile Platform," in *TENCON 2022 - 2022 IEEE Region 10 Conference (TENCON)*, 2022, pp. 1–6. doi: 10.1109/TENCON55691.2022.9977715.

[31] D. S. Dewantara, R. Hidayat, H. Susanto, and A. M. Arymurthy, "CNN with Multi Stage Image Data Augmentation Methods for Indonesia Rare and Protected Orchids Classification," in *2020 International Conference on Computer Science and Its Application in Agriculture (ICOSICA)*, 2020, pp. 1–5. doi: 10.1109/ICOSICA49951.2020.9243174.

[32] S. Prajapati, S. Qureshi, Y. Rao, S. Nadkarni, M. Retharekar, and A. Avhad, "Plant Disease Identification Using Deep Learning," in *2023 4th International Conference for Emerging Technology (INCET)*, 2023, pp. 1–5. doi: 10.1109/INCET57972.2023.10170463.

[33] Ü. Atila, M. Uçar, K. Akyol, and E. Uçar, "Plant leaf disease classification using EfficientNet deep learning model," *Ecol Inform*, vol. 61, p. 101182, Mar. 2021, doi: 10.1016/j.ecoinf.2020.101182.

[34] T. Akiyama, Y. Kobayashi, Y. Sasaki, K. Sasaki, T. Kawaguchi, and J. Kishigami, "Mobile Leaf Identification System using CNN applied to plants in Hokkaido," in *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*, 2019, pp. 324–325. doi: 10.1109/GCCE46687.2019.9015298.

[35] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed. O'Reilly Media, Inc., 2019.

[36] T. Pessoa, R. Medeiros, T. Nepomuceno, G.-B. Bian, V. H. C. Albuquerque, and P. P. Filho, "Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications," *IEEE Access*, vol. PP, p. 1, Apr. 2018, doi: 10.1109/ACCESS.2018.2874767.

[37] "JupyterLab," GitHub. Accessed: Apr. 19, 2024. [Online]. Available: https://github.com/jupyterlab/jupyterlab

[38] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations*, Apr. 2014.

[39] J. Terven, D.-M. Cordova-Esparza, A. Ramirez-Pedraza, and E. Chávez Urbiola, “Loss Functions and Metrics in Deep Learning. A Review.” Apr. 2023.

[40] B. Patil and V. Burkpalli, “Segmentation of cotton leaf images using a modified chan vese method,” *Multimed Tools Appl*, vol. 81, pp. 1–19, Apr. 2022, doi: 10.1007/s11042-022-12436-8.

[41] K. Zuiderveld, “VIII.5. - Contrast Limited Adaptive Histogram Equalization,” in *Graphics Gems*, P. S. Heckbert, Ed., Academic Press, 1994, pp. 474–485. doi: https://doi.org/10.1016/B978-0-12-336156-1.50061-6.

[42] P. D. Sunil Bhutada Nakerakanti Yashwanth and K. Shekar, “Opening and closing in morphological image processing,” *World Journal Of Advanced Research and Reviews*, vol. 14, pp. 687–695, Apr. 2022, doi: 10.30574/wjarr.2022.14.3.0576.

[43] “OpenCV: Image Thresholding,” Opencv.org. Accessed: Apr. 20, 2024. [Online]. Available: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html

[44] N. A. Rehman and F. Haroon, “Adaptive Gaussian and Double Thresholding for Contour Detection and Character Recognition of Two-Dimensional Area Using Computer Vision,” *Engineering Proceedings*, vol. 32, no. 1, 2023, doi: 10.3390/engproc2023032023.

[45] P. Rosin, “Unimodal thresholding,” *Pattern Recognit*, vol. 34, pp. 2083–2096, Apr. 2001, doi: 10.1016/S0031-3203(00)00136-9.

[46] X.-Y. Gong, H. Su, D. Xu, Z.-T. Zhang, F. Shen, and H.-B. Yang, “An Overview of Contour Detection Approaches,” *Machine Intelligence Research*, vol. 15, no. 6, pp. 656–672, 2018, doi: 10.1007/s11633-018-1117-z.

[47] T. Borowik, N. Pettorelli, L. Sönnichsen, and B. Jedrzejewska, “Normalized difference vegetation index (NDVI) as a predictor of forage availability for ungulates in forest and field habitats,” *Eur J Wildl Res*, vol. 59, Apr. 2013, doi: 10.1007/s10344-013-0720-0.

[48] “Streamlit,” Github. Accessed: Apr. 19, 2024. [Online]. Available: https://github.com/streamlit/streamlit

[49] Y.-J. La, D. Seo, J. Kang, M. Kim, T.-W. Yoo, and I.-S. Oh, “Deep Learning-Based Segmentation of Intertwined Fruit Trees for Agricultural Tasks,” *Agriculture*, vol. 13, no. 11, 2023, doi: 10.3390/agriculture13112097.

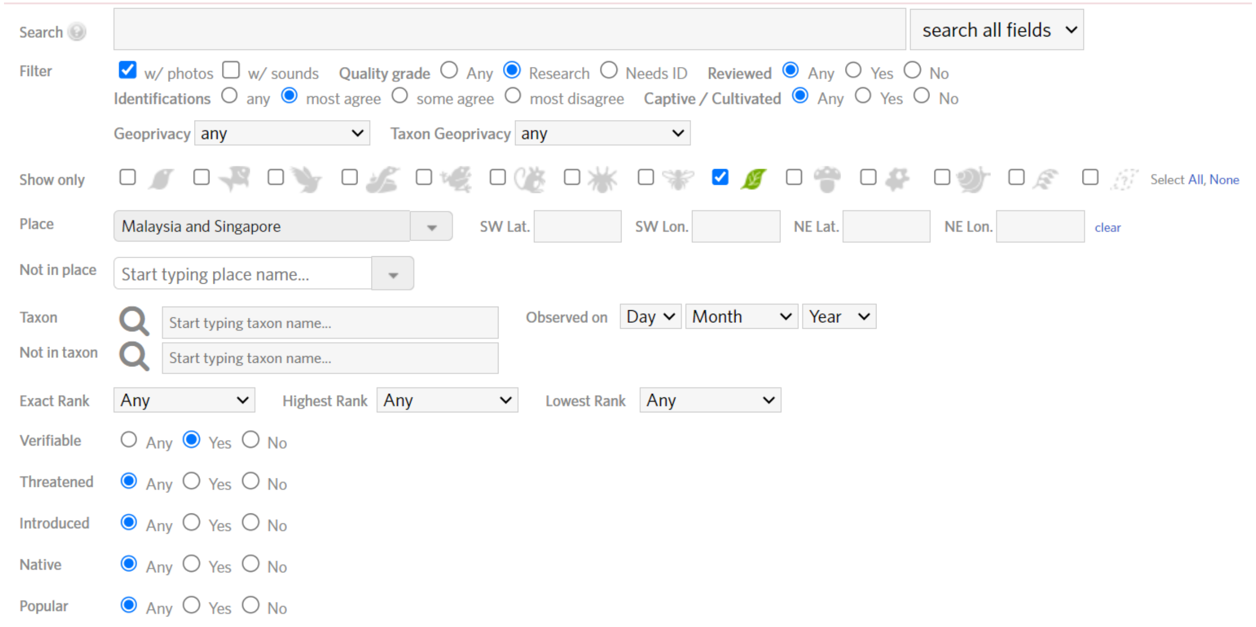
[50] Z. Wang, Y. Lv, R. Wu, and Y. Zhang, “Review of GrabCut in Image Processing,” *Mathematics*, vol. 11, no. 8, 2023, doi: 10.3390/math11081965.

[51] İ. Ataş, “Performance Evaluation of Jaccard-Dice Coefficient on Building Segmentation from High Resolution Satellite Images,” Apr. 2023, doi: 10.17694/bajece.1212563.

[52] J. Kasche and F. Nordström, “Regularization Methods in Neural Networks.” p. 31, 2020.

9. Appendices

Appendix A. iNaturalist Export Tool



Search | search all fields

Filter: [✓ w/ photos] w/ sounds Quality grade Any [✓ Research] Needs ID Reviewed [✓ Any] Yes No
Identifications any [✓ most agree] some agree most disagree Captive / Cultivated [✓ Any] Yes No

Geoprivacy any Taxon Geoprivacy any

Show only ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☑ ☐ ☐ ☐ ☐ ☐ Select All, None

Place Malaysia and Singapore SW Lat. SW Lon. NE Lat. NE Lon. clear

Not in place Start typing place name...

Taxon Start typing taxon name... Observed on Day Month Year

Not in taxon Start typing taxon name...

Exact Rank Any Highest Rank Any Lowest Rank Any

Verifiable Any [✓ Yes] No

Threatened [✓ Any] Yes No

Introduced [✓ Any] Yes No

Native [✓ Any] Yes No

Popular [✓ Any] Yes No

Figure 9.1 iNaturalist Export Tool