LeafAI : Leaf Classification Application

Kajornsak Peerapathananont
Purichaya Leelerdsakulvong
Wiranchana Upapornpong

Bachelor of Engineering in Software Engineering
International College
King Mongkut's Institute of Technology Ladkrabang
Academic Year 2017
KMITL-2018-IC-B-003-004

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

COPYRIGHT 2018
INTERNATIONAL COLLEGE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Thesis - Academic Year 2017

Bachelor of Engineering in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

Title: LeafAI: Leaf Classification Application

Authors:

1. Kajornsak Peerapathananont Student ID: 57090006
2. Purichaya Leelerdsakulvong Student ID: 57090028
3. Wiranchana Upapornpong Student ID: 57090043

Approved for Submission

.....

(Dr. Ukrit Watchareeruetai)

Advisor

Date 15 June 2018

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Abstract

Plants are considered to be the the first living organisms on Earth. They provide essential key resources for environment and human life. One most important thing is that they are responsible for the presence of a gas needed for most organism, oxygen. Furthermore, they also provide products such as food, medicine, clothing, and fuels for human use. Even a single tree can create habitats for several organisms. As can be seen, plants play a dramatically crucial role and life would not be possible without plants.

Due to their importance, a comprehension of plant identities for maximizing the advantages from them are needed. This thesis proposes a system to classify leaf identification by using leaf taxonomy knowledge cooperating with the advancement in computer vision and artificial intelligence skills.

The system comprises of a classification model and a server-less mobile application in which the application accepts leaf images as inputs. The leaf images will be preprocessed into five leaf features: venation, base, apex, margin, and lamina using computer vision techniques. After this, the images will be fed to the classification model, which comprises of a five CNNs combined using a decision tree. Each CNN will classify one of the five leaf features and those five classifications will be fed into the decision tree and the decision tree's output is the leaf species. The whole model is pretrained and integrated in the mobile application. Which allows the user to take a new leaf image using their phone's cameras or pick an image from their phone's library.

Experiment results show that our proposed method is able to effectively classify the plant species in our dataset (testing on our test set, we were able to achieve 95.45% accuracy). Furthermore, as we have added a local plant to the dataset, we can see that the proposed method should be able to identify any plant species as long as our classification

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

model has been trained on it.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Acknowledgments

The completion of this project, LeafAI: Leaf Classification Application, could not have been completed without the support and assistance from several people.

First and foremost, we, the team members, wish to express our sincere appreciation toward our advisor, Dr. Ukrit Watchareeruetai. We are really grateful for his generous support. With his contribution, we gained crucial knowledge and valuable guidance which widen our project from various perspectives.

Our gratitude also goes to Asst.Prof.Dr. Chaiwat Nuthong, Dr. Isara Anantavrasilp, and Dr. Natthaphong Jungteerapanich for providing us continuous advice and encouragement, and giving us opportunity to present the project.

Last but not least, we would like to thank our third and fourth-year colleagues for the attentions, and feedbacks. Their cooperations have been helpful in many ways.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Contents

1 Introduction **1**

1.1 Motivation and problem description 1

1.2 Objective and scope 2

1.3 Thesis structure 3

2 Background knowledge **4**

2.1 Decision tree 4

2.2 Measuring impurity 5

2.2.1 Entropy 5

2.2.2 GINI index (GINI impurity) 6

2.2.3 Classification error 6

2.3 Information theory 7

2.3.1 Softmax classifer 8

2.4 Convolutional neural network 9

2.4.1 Convolution 9

2.4.2 Convolutional layer 10

2.4.3 Pooling layer 11

2.5 Approach to handling imbalanced datasets 11

2.5.1 Random under-sampling 11

2.5.2 Random over-sampling 11

2.5.3 Cluster-based over-sampling 11

2.5.4 Model penalization 11

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

3 Methodology **12**

3.1 Proposed method and system overview 12

3.2 Mobile application 15

3.2.1 Converting model to CoreML 15

3.2.2 Application overview 15

3.2.3 Application features 16

3.3 Preprocessing 19

3.3.1 Data reorientation 19

3.3.2 Preprocessing for apex and base 20

3.3.3 Preprocessing for lamina and margin 21

3.3.4 Preprocessing for venation 23

3.4 CNN training 26

3.4.1 Installation 26

3.4.2 Regularization 26

3.4.3 Weight initialisation 27

3.5 Decision tree 28

4 Experiments **29**

4.1 Flavia dataset 29

4.1.1 Morphological characteristics 29

4.1.2 Margin 31

4.1.3 Lamina 32

4.1.4 Base 33

4.1.5 Apex 34

4.1.6 Venation 35

4.2 Convolutional neural network - Architecture 37

4.3 Convolutional neural network - Margin 39

4.3.1 Objective 39

4.3.2 Setup 39

4.3.3 Results 39

This material is reserved for educational use only, not allowed for commercial use.

Forbidden to modify the content, and cite the document when use

4.3.4 Conclusion 41
4.4 Convolutional neural network - Lamina 41
4.4.1 Objective 41
4.4.2 Setup 41
4.4.3 Results 42
4.4.4 Conclusion 47
4.5 Convolutional neural network - Base 48
4.5.1 Objective 48
4.5.2 Setup 48
4.5.3 Results 49
4.5.4 Conclusion 50
4.6 Convolutional neural network - Apex 51
4.6.1 Experiment 1 : Segmented normal 51
4.6.2 Conclusion 53
4.7 Convolutional neural network - Venation 53
4.7.1 Objective 53
4.7.2 Results 54
4.7.3 Conclusion 57
4.8 Species classification 57
4.8.1 Conclusion 57
4.9 Final results 58
4.9.1 Local dataset 58
4.9.2 Convolutional neural network 58
4.9.3 Decision tree 59

5 Conclusion **60**
5.1 Future work 61
5.1.1 Mobile application 61
5.1.2 Classification model 61

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

List of Figures

2.1 An example of a decision tree 4
2.2 Architecture of CNN 9
2.3 Matrix convolution 10

3.1 Method overview 12
3.2 Process overview of the system 13
3.3 Base 14
3.4 Margin 14
3.5 Apex 14
3.6 Lamina 14
3.7 Venation 14
3.8 Application Overview 16
3.9 Home page 17
3.10 Info page 17
3.11 Choosing image source page 17
3.12 Confirm page 17
3.13 Result page 18
3.14 Preprocessing page 18
3.15 Angle for rotation 20
3.16 Rotated 20
3.17 Extraction of apex and base 20
3.18 Steps to get leaf shape image for lamina and margin 22
3.19 Steps to get the contour image 23

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

3.20 Input and output of venation preprocessing 24
3.21 Sobel edge detection for leaf A 24
3.22 Sobel edge detection for leaf B 25
3.23 Sobel edge detection for leaf C 25
3.24 Sobel edge detection for leaf D 25
3.25 Dropout application 26

4.1 Example inputs for margin CNN 39
4.2 Example segmented base image input 48
4.3 An example of flipping 48
4.4 Example segmented apex input 51
4.5 Example venation blocks input 53
4.6 Example sobel edge image input 54
4.7 Examples of Phut Phuket leaves 58

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

List of Tables

2.1 Transportation mode data 7
2.2 Information gain summarization 7

4.1 Dataset morphological characteristics 30
4.2 Margin class separation 31
4.3 Margin dataset 31
4.4 Lamina class separation 32
4.5 Lamina dataset 33
4.6 Base class separation 34
4.7 Base dataset 34
4.8 Apex class separation 35
4.9 Apex dataset 35
4.10 Venation class separation 36
4.11 Venation dataset 36
4.12 ConvNet architecture for all CNN 38
4.13 Full leaf image to classify margin (83.89%) 40
4.14 Black and white normal to classify margin (89.06%) 40
4.15 Outline image to classify margin (85.27%) 40
4.16 Different margin preprocessing methods and their accuracies 41
4.17 Lamina dataset with flip augmentation 42
4.18 Full leaf image to classify lamina (72.24%) 43
4.19 Black and white image to classify lamina (91.70%) 44
4.20 Black and white image to classify lamina with flip (94.65%) 45

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

4.21 Contour image to classify lamina (93.05%) 46
4.22 Different lamina preprocessing methods and their accuracies 47
4.23 Segmented normal to classify base (90.13%) 49
4.24 Segmented with flip to classify base (93.47 %) 49
4.25 Segmented with duplicate and flip to classify base (96.47 %) 50
4.26 Different base preprocessing methods and their accuracies 50
4.27 Segmented to classify apex (92.96 %) 52
4.28 Segmented with flip to classify apex (96.87 %) 52
4.29 Segmented with duplicates and flip to classify apex (95.57%) 52
4.30 Different apex preprocessing methods and their accuracies 53
4.31 Small segmented block to classify venation (12.9%) 55
4.32 Sobel egde detection to classify venation (91.29%) 56
4.33 Different venation preprocessing methods and their accuracies 57
4.34 Different venation preprocessing methods and their accuracies 58
4.35 CNN final accuracies 59

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Chapter 1

Introduction

This chapter includes the motivation behind this project as well as a description of the problem to be solved. It also describes the project's objectives and intended scope. Additionally, the proposed algorithm and system are explained. Lastly, this chapter concludes with a summary of the thesis's structure.

1.1 Motivation and problem description

It is undeniable for us to imagine what the world would be like without plants. Plants are believed to be the first living things born on Earth. They are the major source of the oxygen we breathe and almost everything in life relies on plants. Almost all products we consume, such as, food, clothing, medicine or fuels, all of them consists of plants as their parts. For instance, we cannot say that some of our favorite dish which contains only meat has nothing to do with plants, since the meat are supplied by most of the animals that has plants as their food. All that emphasizes how crucial plants are and it would be great if we are able to maximize their benefits. In order to take the most advantage from plants, comprehension of their identification is needed.

There are various ways to classify plants. Their parts, e.g., leaf, bark, flower, seed may be considered. However, leaves are relatively easy to find and obtain comparing to other parts. These reasons are why we are interested in the leaf classification and decided to focus on it as our project. To classify leaf by using plant taxonomy knowledge, there are

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

many classification systems (60+) [12] which were created by different people in different time. The most acceptable systems are molecular-based classification such as Angiosperm Phylogeny Group (APG) [1] system which classifies the plant by using its biological component (angiosperm). These systems require knowledge and technologies in genetic field. The second acceptable systems are visual-based classification which classify the plant by using its physical appearance. One of the components that has been used for classification is leaf.

Advancement in computer vision and artificial intelligence allows skills that would take a human years to learn, in this case visually identifying plants, to possibly be learned by machines in a matter of days or even hours, with the use of deep learning. This opens the possibility of enabling anybody with a smart phone to identify plants with a reasonable degree of accuracy, and allows such data collection and analysis to be widely available to non-experts. Deep learning [2] is a variation of machine learning, a field that examines computer algorithms that learn and improve on their own. There already exists numerous documentations of deep learning being able to successfully identify numerous plant types with high accuracy which encourage us to to further their work.

1.2 Objective and scope

The project's main objective is to implement an algorithm that is able to classify leaves. The algorithm will employ taxonomy, machine learning, and computer vision techniques. This algorithm will be used to train a model. We will also develop an iOS mobile application that will use this model. Computer vision will be used to prepare datasets for the training process as well as to prepare the input on the mobile application. To conclude our objectives are:

- Develop a new algorithm for leaf classification using taxonomy, machine learning, and computer vision techniques.
- Create, train, and test a model using this algorithm with public datasets.
- Gather and prepare a local dataset using plants around campus then use to further train our model.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

- Create a mobile application to use this model.
- Develop a computer vision algorithm to use on mobile phone application to process model input compatible with our trained model.
- Test application with local plants around campus.

Our project scope will be defined by these constraints:

- The input image must be of decent quality or taken in relatively good weather.
- The application can only be run on iOS 11 (Core ML support).

1.3 Thesis structure

- In Chapter 2, we describe the problems and review of related work.
- In Chapter 3, the methodology for overall system of our project is described here.
- In Chapter 4, the experiments and results of the project are provided.
- In Chapter 5, we will make a conclusion of our work and provide the future work of our project.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Chapter 2

Background knowledge

2.1 Decision tree

A decision tree (DT) is a simple classification model and a decision support tool that uses a tree-like graph. Each branch of the graph represents a possible decision and leaf nodes are the evaluation results. An example of a DT is as shown in Figure 2.1.

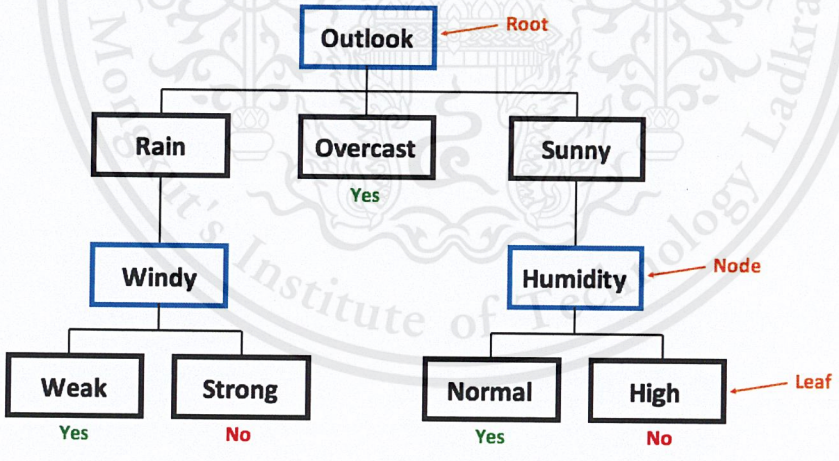


Figure 2.1: An example of a decision tree

Source: [5]

To construct a DT (using Iterative Dichotomiser 3 or ID3 [5] method), we have to begin at the set S as a root node and find the attribute that returns the highest value of information gain (IG); a variable that tells how significant a given attribute of the feature

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

vectors is which can also be understood as the measurement of the difference in entropy from before to after the set S is split on an attribute, to be the root node. In order to obtain the value of information gain, the entropy (impurity) or the measurement of the amount of uncertainty in the data set S is needed to be calculated. A branch with entropy of zero is a leaf node and the one with entropy more than zero needs further splitting. The formula of information gain is as follows:

$$IG = H(Parent) - \sum_{c \in Children} p(c) \times H(c), \tag{2.1}$$

where $H(Parent)$ is an entropy of parent (upper node), $H(c)$ is an entropy of children (lower node), and $p(c)$ is the probability of children.

When entropy is zero, this means that the sample is completely homogeneous. While when entropy is one, the sample is equally divided. The equation for entropy is as follows:

$$H(S) = \sum_{i=1}^{C} -p_i log_2 p_i, \tag{2.2}$$

where C is the number of classes and p_i is the probability of class i.

2.2 Measuring impurity

Given a data table that contains attributes and class of the attributes, we can measure homogeneity of the table based on the classes. A table is pure or homogenous if it contains a single class. If a data table contains several classes, then we the table is impure or heterogeneous. There are several ways to measure the degree of impurity. The most well known measurements of degree of impurity are entropy, GINI index, and classification error.

2.2.1 Entropy

Entropy is used to calculate the homogeneity of a sample in ID3 and C4.5 [8] algorithm. Based computations of entropy are similar to the GINI index computations.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

$$E = \sum_{i=1}^{C} -p_i log_2 p_i \tag{2.3}$$

2.2.2 GINI index (GINI impurity)

GINI index is used by the CART (classification and regression tree) [4], SLIQ algorithm [3], and SPRINT [7] algorithm. GINI impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

$$G = 1 - \sum_{i=1}^{C} p_i^2 \tag{2.4}$$

2.2.3 Classification error

Classification error measures classification error made by a node of a tree. The classification error rate of the training dataset should be approximately equal to the test dataset; if not, the model may be too particular for the training dataset and not general sufficient.

$$C = 1 - max\{p_i\} \tag{2.5}$$

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 2.1: Transportation mode data

Attributes				Classes
Gender	Car Ownership	Travel Cost($)/km	Income Level	Transportation Mode
Male	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Female	1	Cheap	Medium	Train
Female	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Male	0	Standard	Medium	Train
Female	1	Standard	Medium	Train
Female	1	Expensive	High	Car
Male	2	Expensive	Medium	Car
Female	2	Expensive	High	Car

Table 2.2: Information gain summarization

Gain	Gender	Car Ownership	Travel Cost/km	Income Level
Entropy	0.125	0.534	1.210	0.695
GINI Index	0.060	0.207	0.500	0.293
Classification Error	0.100	0.200	0.500	0.300

When computing impurity using the data in Table 2.1 by three different impurity measurement indices: entropy, GINI index and classification error, it can be seen that although they yield different numbers of impurity, but the information gain results are in the same direction as shown in Table 2.2.

2.3 Information theory

In order to improve a solution, we need to be able to express how good or bad the solution is at solving the problem. Information theory offers concrete, principled formalizations for these things we need to express. It gives us ways of measuring and expressing uncertainty.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

For example, classification in machine learning, when using the softmax classifier our model might say there is an 80% chance this image is a dog, and a 20% chance it is a cat. If the correct answer is dog. How good or bad is it that we only said there was an 80% chance it was a dog? How much better would it have been to say 85%?

2.3.1 Softmax classifer

The softmax classifier uses the softmax function which changes the raw class scores into normalized positive values that sum to one, so that the cross-entropy loss function can be applied. Softmax classifier provides probabilities for each class label

Entropy is the probability distribution of p. It is formally defined as the average amount of information produced by a stochastic source of data. If a variable X takes on the states $x_1, x_2, x_3, ..., x_n$, then entropy is defined as

$$H(p) = \sum_x p(x) \log_2 \left(\frac{1}{p(x)} \right) \tag{2.6}$$

Cross-entropy calculates the loss or difference between the ground truth and the predicted probabilities. It is defined as

$$H(p, q) = \sum_x q(x) \log_2 \left(\frac{1}{p(x)} \right) \tag{2.7}$$

where the true probability $p(x)$ is the ground true label, and the given distribution $q(x)$ is the predicted value of the current model.

The full softmax classifier loss is then defined as the average cross-entropy loss over the training examples and the regularization. Thus, the equation is defined as

$$L(w) = \frac{1}{N} \sum_{n=1}^{N} H(p_n, q_n) \tag{2.8}$$

where w is the set of weights, N is the number of examples, $q(x)$ is the predicted probability and $p(x)$ is the actual probability.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

2.4 Convolutional neural network

Convolution neuron network (CNN) is one of the deep architecture neural network that has learnable weights and biases. CNN is mostly used in image recognition. The name of Convolutional neural network comes from "Convolution operator" in linear filtering. CNN can adjust its weight to obtain a proper weight for detecting feature from the input. The structure of CNN can be described in Figure 2.2.

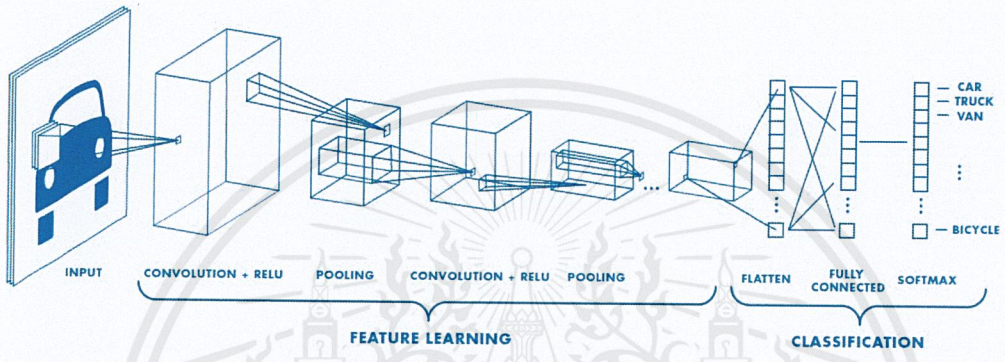


Figure 2.2: Architecture of CNN

Source: [11]

2.4.1 Convolution

Normally, in computer vision or digital signal processing, convolution operator is used for merging two signals or merging two matrices.

In this case, as shown in Figure 2.3, applying convolution filter (convolution kernel) to the input matrix will produce a new matrix with the weight of that kernel.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

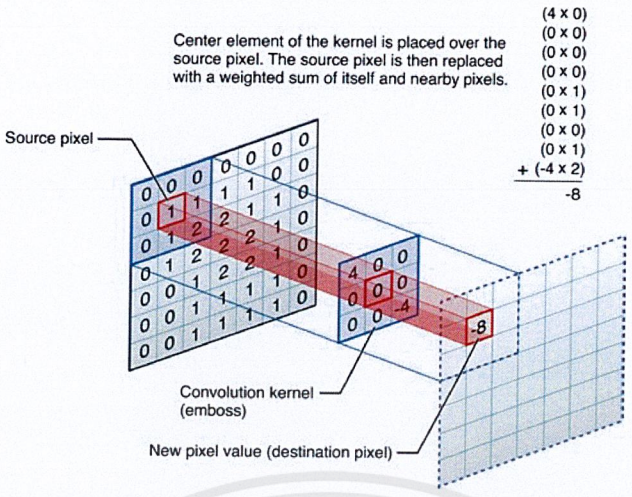


Figure 2.3: Matrix convolution

Source: [10]

2.4.2 Convolutional layer

Convolutional layer is a layer that CNN convolves kernel matrix with input matrix to produce an output matrix. Applying filters through the input matrix will obtain certain features.

Kernel

Kernel or Filter is a matrix that is applied to an input and produces the output corresponding to the kernel. To construct a Kernel or Filter, the size of the kernel must be specified to the system. Filter will be applied through the full depth of the input by sliding across the input and computing dot products.

Stride

Stride is one of hyper-parameter that can be specified in the convolution layer. The value of stride can be used to control the step of convolution operation. For example, when stride is equal to 1, this means the filter will be shifted by one pixel (or one column) on each operation.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

2.4.3 Pooling layer

Pooling layer is a layer that reduce (downsampling) the spatial size of the output from convolution layer to limit the number of parameters and avoiding overfitting. Normally, the most common form of filter in pooling layer is 2×2 filter with a stride of 2.

2.5 Approach to handling imbalanced datasets

2.5.1 Random under-sampling

Random under-sampling is an approach that eliminates majority class instances to handle the imbalance of the dataset. The method is done until the examples of both the majority and minority classes are balanced. The drawback of this method is that it may ignore necessary information which might be essential for creating rule classifiers.

2.5.2 Random over-sampling

Over-sampling aims to balance class distribution by randomly replicating minority class examples. Unlike under-sampling, this solution doesn't lead to information loss. However, by replicating the minority class samples, over-fitting problem may occur.

2.5.3 Cluster-based over-sampling

In this case, majority and minority class instances are oversampled such that all classes have the same size. This method applies K-means clustering algorithm on all classes independently. The main disadvantage of this algorithm is that it can increase the likelihood of over-fitting to the training data.

2.5.4 Model penalization

Model penalization is the method that biases the model by giving penalties cost on the minority class during training in order to pay more attention to them. The approach is desirable when the results are poor and the system is restricted to a specific algorithm.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Chapter 3

Methodology

3.1 Proposed method and system overview

Our system is a mobile application that runs without a server and, thus, does not need network connection. The mobile application contains a trained model that classifies leaf species. The trained model is integrated into the application using Core ML. Figure 3.1 illustrates the method overview Figure 3.2 shows the overall overview of the preprocessing.

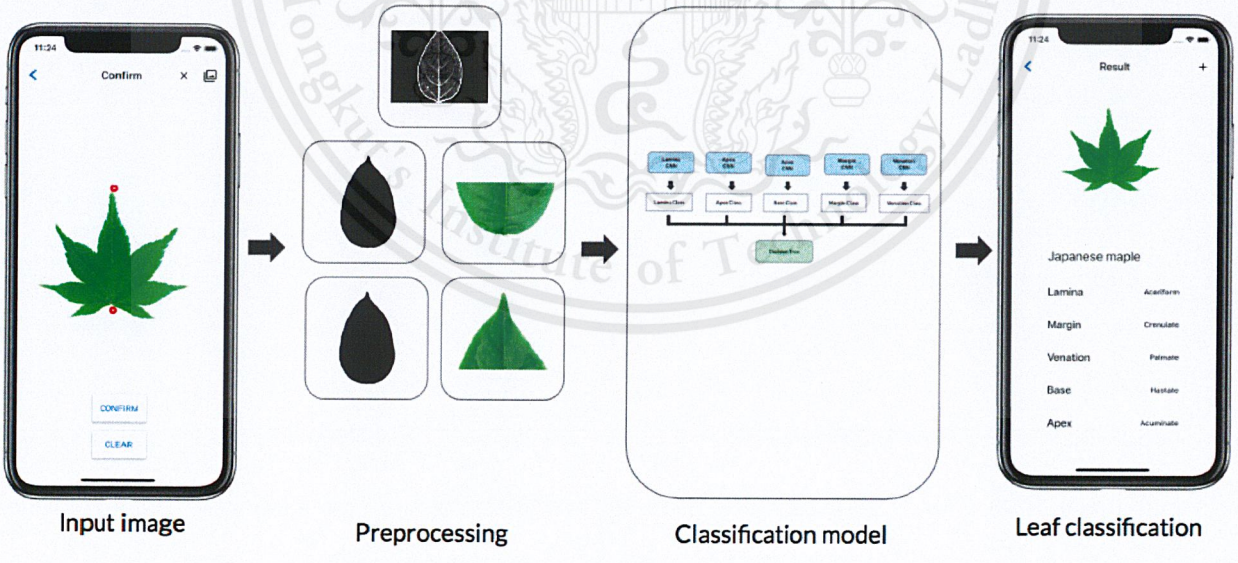


Figure 3.1: Method overview

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Once the user inputs an image, the image will be preprocessed five different ways to create five inputs for each of the neural networks. Each of the five inputs will be input into their respective CNNs which are feature specific neural networks (FSNN) to classify that specific feature. Those features will be used to classify the leaf species by the DT.

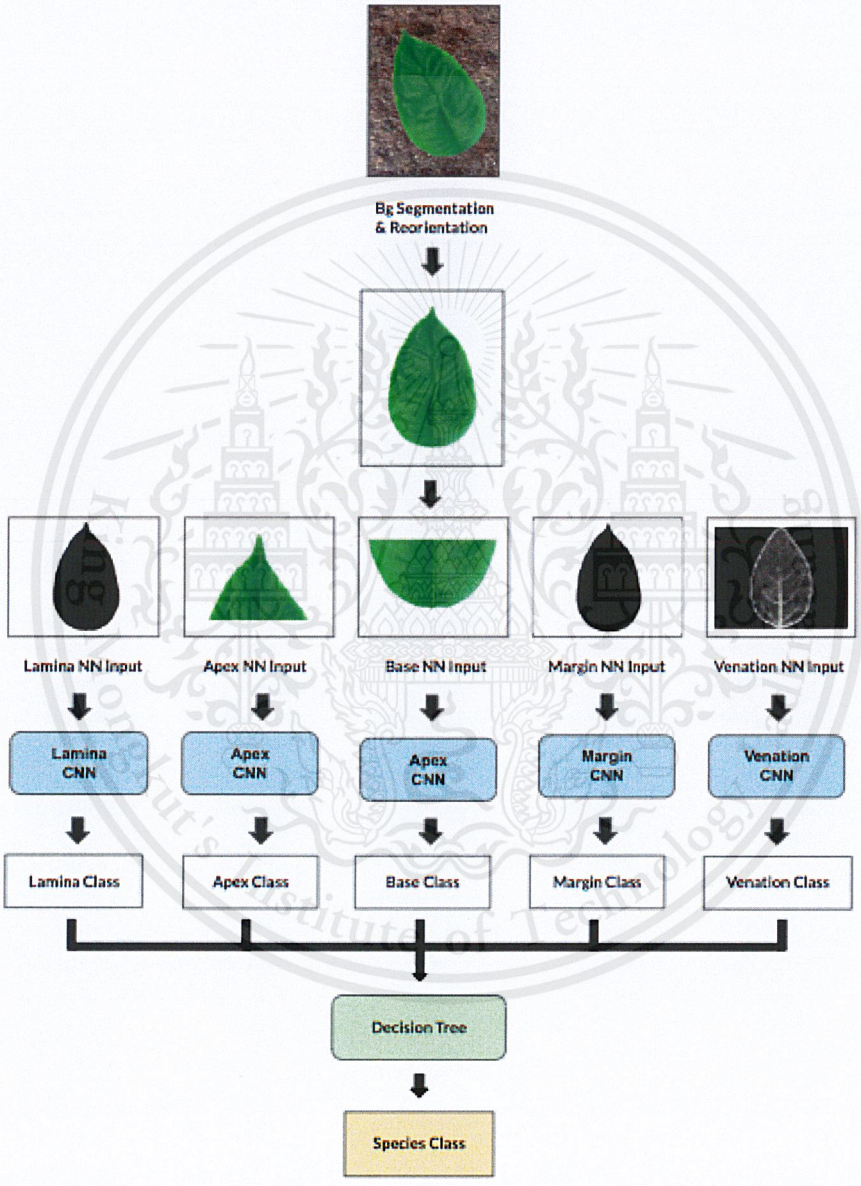


Figure 3.2: Process overview of the system

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

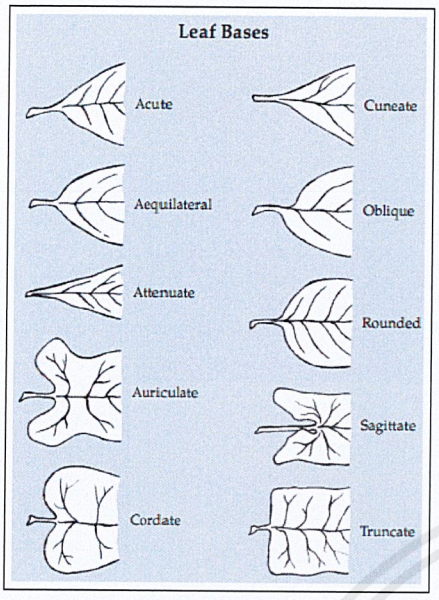


Figure 3.3: Base

Source: [9]

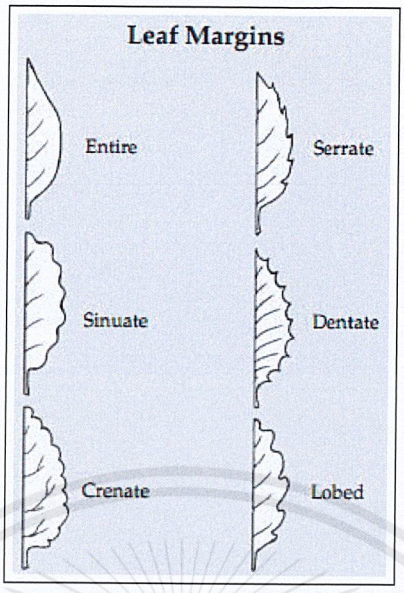


Figure 3.4: Margin

Source: [9]

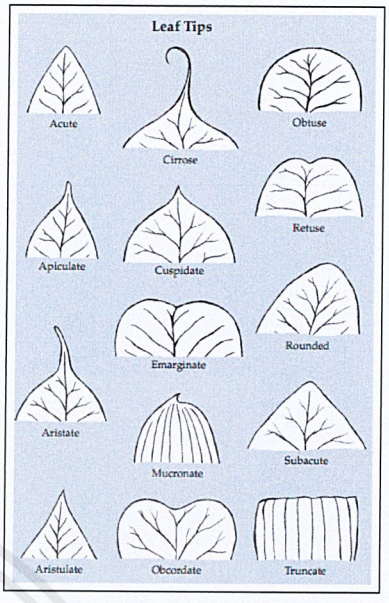


Figure 3.5: Apex

Source: [9]

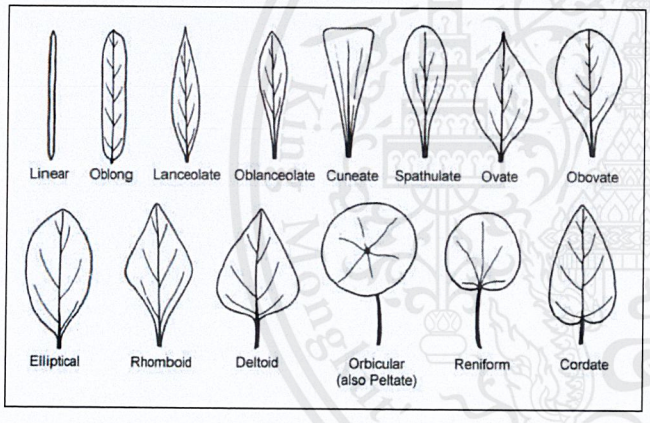


Figure 3.6: Lamina

Source: [9]

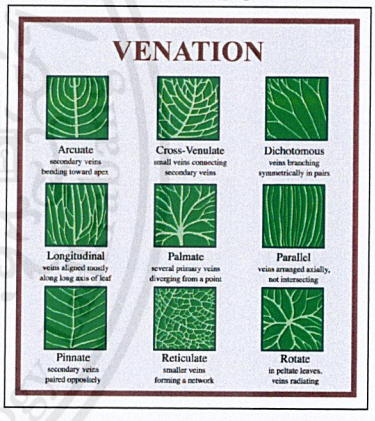


Figure 3.7: Venation

Source: [9]

- Venation - The arrangement of the veins on the leaf
- Apex - The shape of the tip of the leaf
- Base - The shape of the bottom of the leaf
- Lamina - The shape of the leaf as a whole
- Margin - The type of the edge of the leaf

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

We chose these five features because they cover the characteristics that are unique to species' leaves, thus they should provide sufficient information for classification. Furthermore, these features are currently used by botanists to identify leaf species. The input for each FSNN will be preprocessed in a way to further help the NN extract efficient features. This is so that we can enhance the information that is important to that feature, while also removing the unnecessary information.

The DT will be configured using the ID3 method, this is because the data that the DT will be handling is just the different classes of each feature, which are discrete. The DT's input are the classification from the five CNNs which are the five leaf characteristics.

3.2 Mobile application

Since the project is planned to be server-less application, the pre-processing and post-processing will be implemented on mobile device as well. The application will be implemented in iOS Application with Swift language.

3.2.1 Converting model to CoreML

For supporting Machine Learning application in iOS Application, Our Keras models have to be converted into CoreML model which is latest iOS Framework for working on Machine Learning in iOS Devices.

To convert model, Apple does provide coremltools which is a Python package for converting existing model such as Caffe model, Keras model, Scikit-learn model, and etc. into supported model in CoreML.

3.2.2 Application overview

Mobile application contains the pretrained classification model that have been converted using the CoreML library. It also has the preprocessing methods that convert the original input image into the inputs for the CNNs. Figure 3.8 illustrates an overview of the whole processing and classification process.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

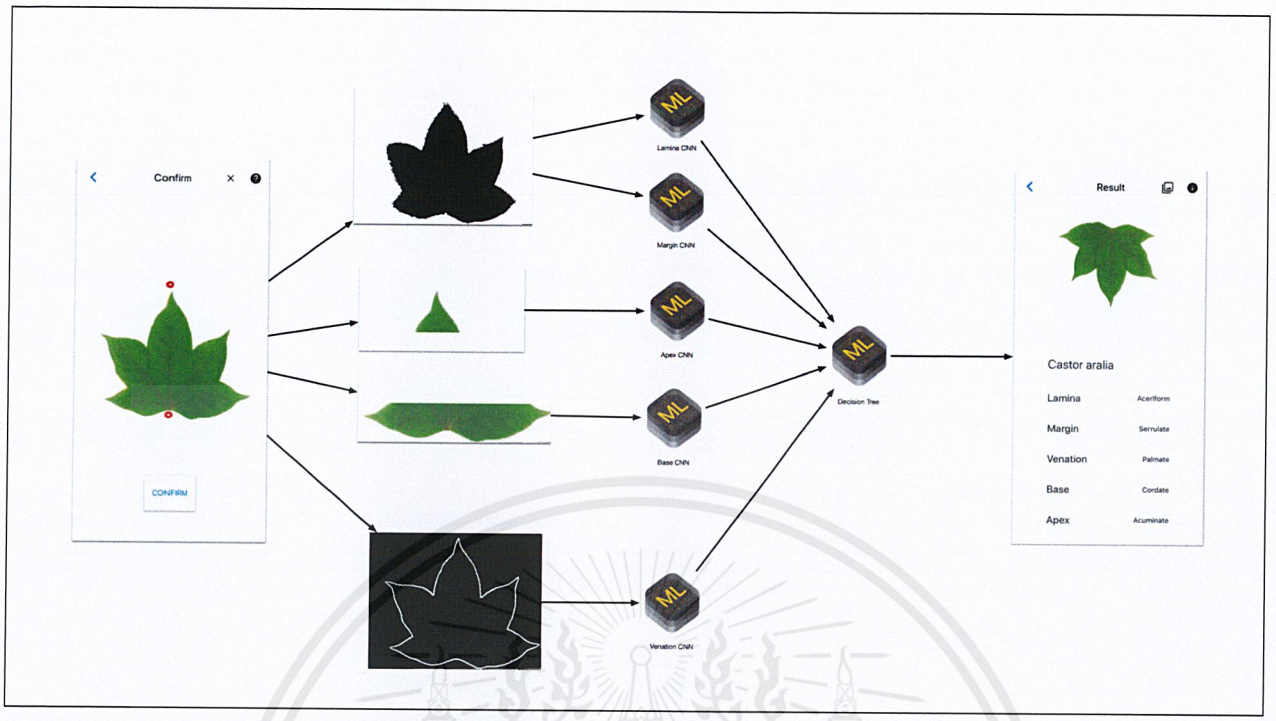


Figure 3.8: Application Overview

3.2.3 Application features

Figure 3.9 shows the home page of the application, where users can view the dataset that our application can classify. By tapping on one of the leaf cards, user can view more detailed information about that species as in Figure 3.10.

User can choose to take a new image from their mobile phone camera or from their photos library as shown in Figure 3.11. Then they must tap the base and apex points, after that they can confirm the points as shown in Figure 3.12. After classification by the system, user will be shown the result as in Figure 3.13. User can also view the inputs for the CNNs as shown in Figure 3.14.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

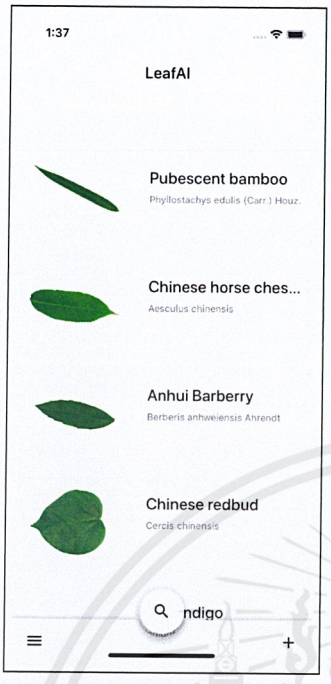


Figure 3.9: Home page

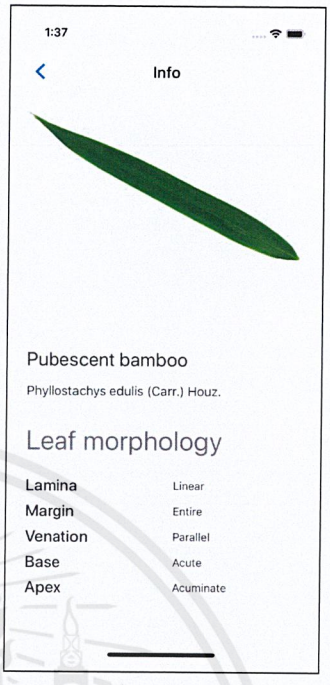


Figure 3.10: Info page

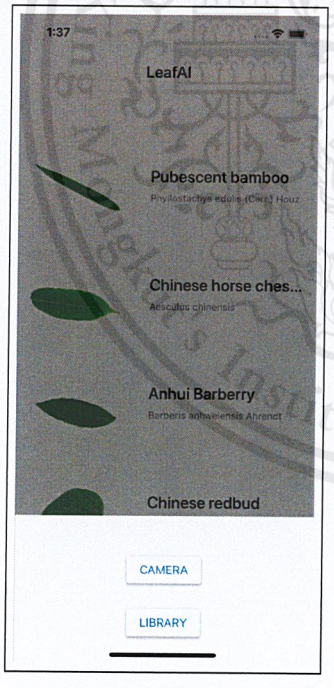


Figure 3.11: Choosing image source page

Figure 3.12: Confirm page

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

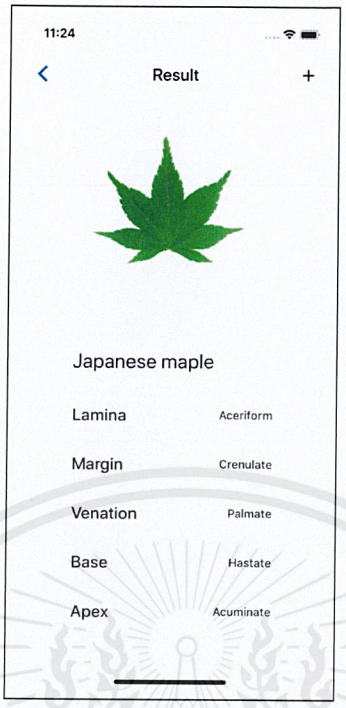


Figure 3.13: Result page

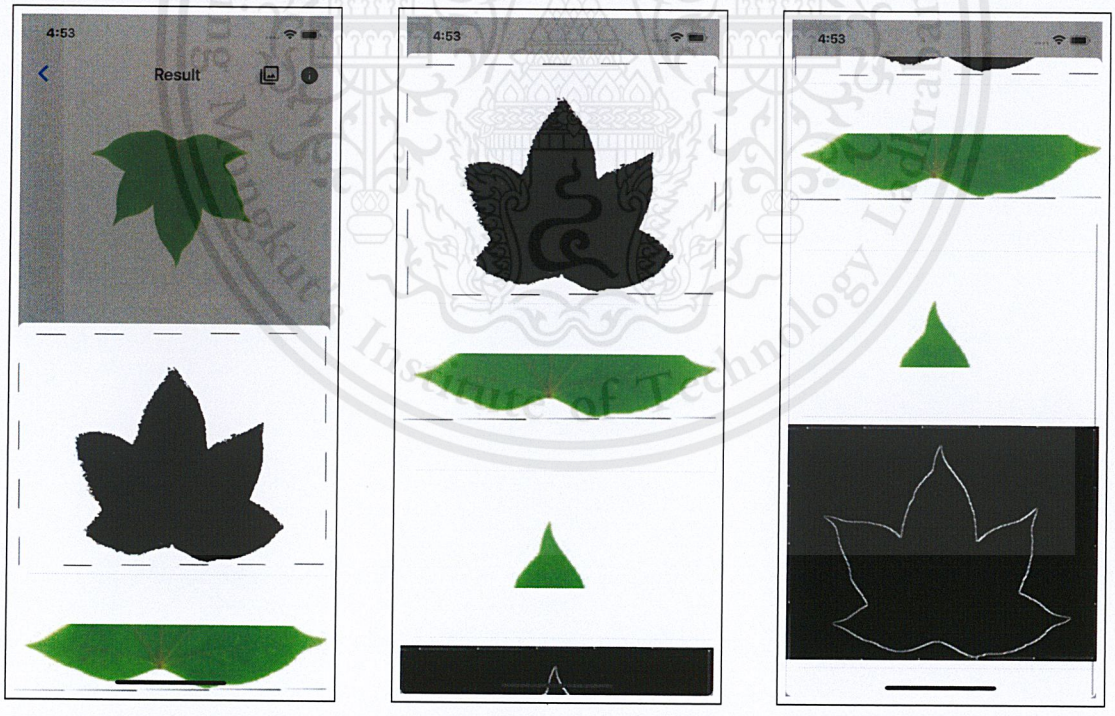


Figure 3.14: Preprocessing page

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

3.3 Preprocessing

The input to our model is a single image captured through the mobile phone camera. Before the image can be used as input to the model it must first be preprocessed to enhance the information appropriate for each of the five features that the model will classify. Several preprocessing methods were implemented and tested to find the most suitable method from them. The results of those tests are mentioned in Chapter 4. The next subsections describe those methods and how they were implemented. The chosen methods are mentioned in Section 3.1.

3.3.1 Data reorientation

In order to avoid orientation problems in some representations of the leaf such as the centroid contour distance signature, we decided to reorient all images of the dataset.

The reorientation procedure is described below :

Step 1 : Given the apex and base points, calculate the angle θ for rotation

Step 2 : Find region of interest (ROI) using contour of the leaf

Step 3 : Crop image to get ROI only

Step 4: Rotate ROI according to θ

Step 5: Rescale ROI to fit 800×600 pixels

Step 6: Normalize image by adding a white border

The apex and base points, angle θ for rotation and the output image after reoriented are shown in Figures 3.15 and 3.16, respectively.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

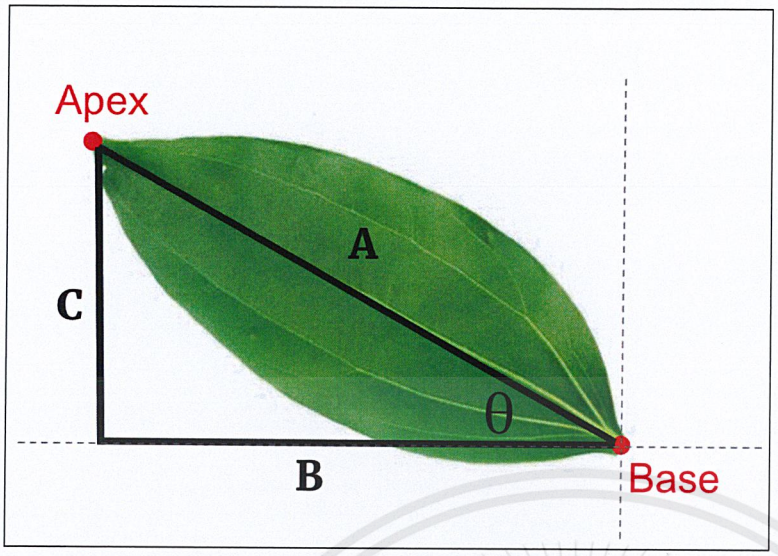


Figure 3.15: Angle for rotation

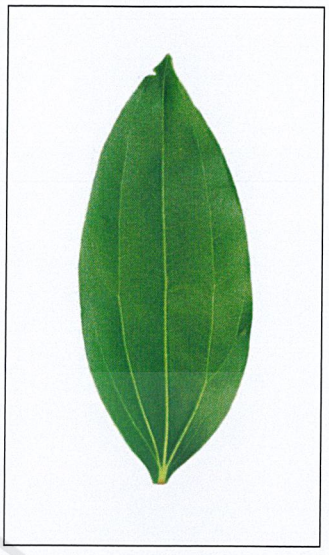


Figure 3.16: Rotated

3.3.2 Preprocessing for apex and base

Since we already know the orientation of the leaf, we can simply cut the top and bottom 25% area of the leaf for the apex and base, respectively. This method is also used by botanists to describe the apex and base area.

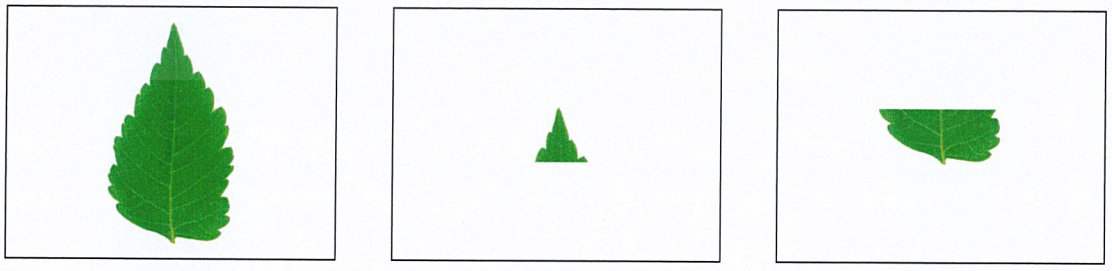
Following is the reorientation procedure:

Step 1 : Find ROI using leaf contour

Step 2 : Crop top and bottom 25% of ROI for apex and base, respectively

Step 3 : Normalize the two images by adding a white border

Figure 3.17 shows the input image and the outputs, which are the segmented apex and base, respectively.



(a) Original (b) Apex (c) Base

Figure 3.17: Extraction of apex and base

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

3.3.3 Preprocessing for lamina and margin

Several preprocessing techniques were researched and implemented to find the one most suitable for classify the lamina and margin. In the end the leaf shape image achieve the highest accuracy.

Black and white image

These are the steps implemented to get the leaf shape image for both the lamina and margin. Figure 3.18 shows the steps being applied and their results.

Step 1: Convert to grayscale

Image color is not considered and a grayscale image would be more proper and easier for further processes.

Step 2: Invert to negative

To make the white part of the image to be the main information

Step 3: Blur

Use Gaussian smoothing to improve contour finding

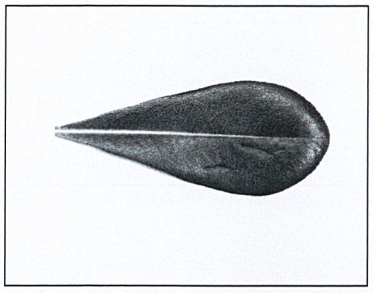
Step 4: Apply image thresholding

Convert to binary image for contour finding

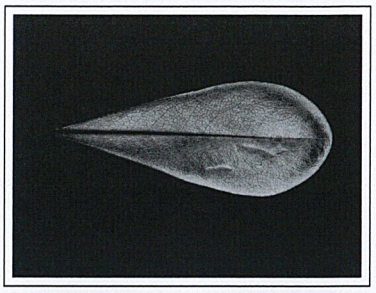
Step 5: Morphological closing

Apply morphological closing in case of any improper thresholding

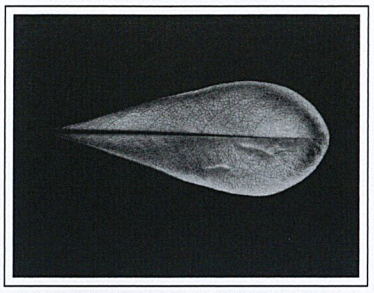
This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use



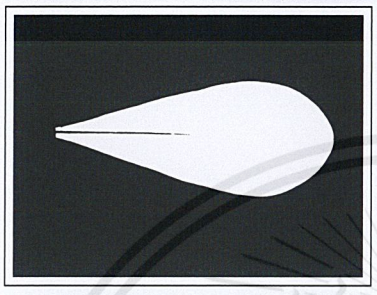
(a) Convert to grayscale



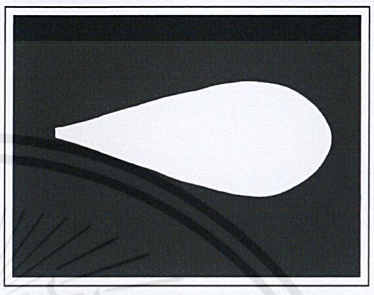
(b) Invert the grayscale



(c) Gaussian smoothing



(d) Thresholding



(e) Morphological closing

Figure 3.18: Steps to get leaf shape image for lamina and margin

Contour image

These are the steps implemented to get the contour image. Figure 3.19 shows the steps being applied and their results.

Step 1: Convert to grayscale

Image color is not considered and a grayscale image would be more proper and easier for further processes.

Step 2: Invert to negative

To make the white part of the image to be the main information

Step 3: Blur

Use Gaussian smoothing to improve contour finding

Step 4: Apply image thresholding

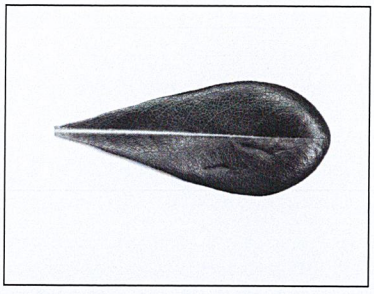
Convert to binary image for contour finding

Step 5: Morphological closing

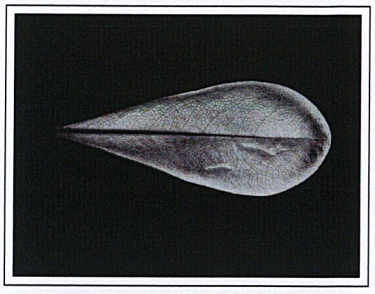
Apply morphological closing in case of any improper thresholding

Step 6: Find contours

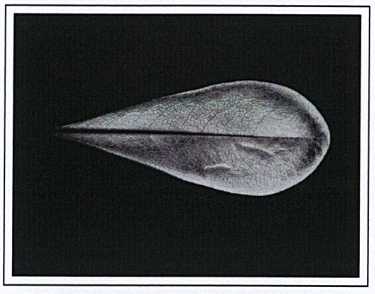
This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use



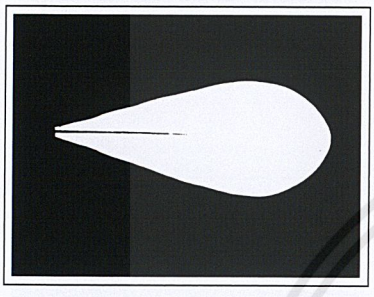
(a) Convert to grayscale to find contour



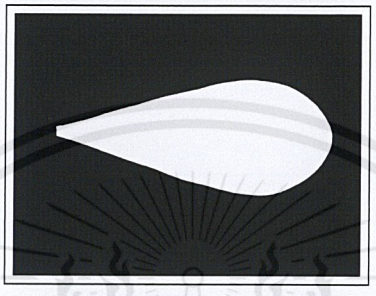
(b) Negative to find contour



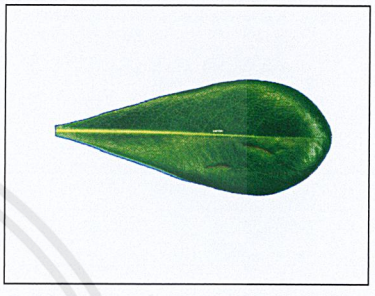
(c) Gaussian smoothing to find contour



(d) Thresholding to find contour



(e) Closing to find contour



(f) Contour that is found

Figure 3.19: Steps to get the contour image

3.3.4 Preprocessing for venation

Small segmented blocks

The idea of the method is to divide a leaf image into square blocks. Only the blocks that contain non-background region are accepted. The blocks then will be used as the inputs for the classification model. By using the blocks, the resolution of the input image are decreased while the necessary data is still maintained. The following are steps for the venation preprocessing. Figures 3.20 shows the input and output of the process.

Step 1: Find the contour of the leaf.

Step 2: Pick the upper and lower bound of coordinates (x,y) from the contour.

Step 3: Divide the input image into square blocks within the boundaries found in Step 2. The size of the blocks of each leaf image depends on the area of the leaf.

Step 4: Detect the background region of each block and only use the blocks that contain non-background region.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

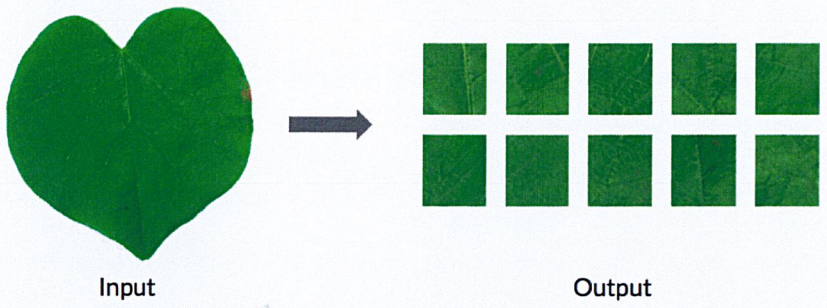


Figure 3.20: Input and output of venation preprocessing

Sobel edge detection

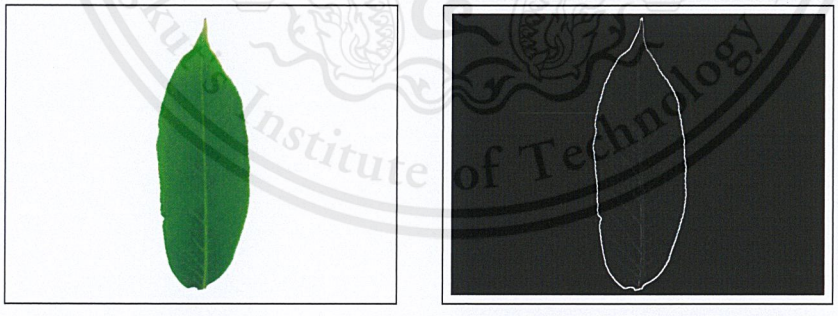
In this section, venation preprocessing using Sobel edge detection is implemented. The input and output image sizes are 800 × 600 pixels. The steps for computing Sobel filter are as follows

Step 1: Apply Sobel filter

Step 2: Perform scaling, calculating the absolute, and converting image into CV_8U

Step 3: Add weight to achieve some smoothing

Figures 3.21 - 3.24 show different types of leaves and their results after applying Sobel edge detection.

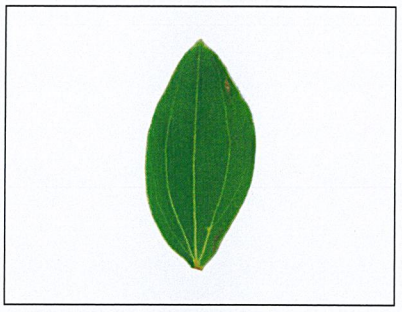


(a) Image A before applying Sobel edge detection

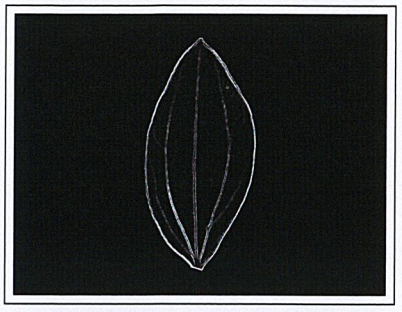
(b) Image A after applying Sobel edge detection

Figure 3.21: Sobel edge detection for leaf A

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use



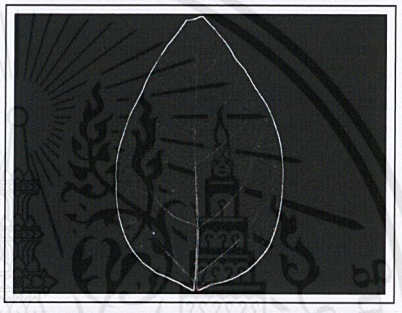
(a) Image B before applying Sobel edge detection



(b) Image B after applying Sobel edge detection

Figure 3.22: Sobel edge detection for leaf B

(a) Image C before applying Sobel edge detection



(b) Image C after applying Sobel edge detection

Figure 3.23: Sobel edge detection for leaf C

(a) Image D before applying Sobel edge detection

(b) Image D after applying Sobel edge detection

Figure 3.24: Sobel edge detection for leaf D

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

3.4 CNN training

3.4.1 Installation

Working set up for CNN training: GeForce GTX 1070, Windows 10, CUDA 9.0, CUDNN 7.0.5 for CUDA 9.0, tensorflow-gpu 1.5.0, and Keras 2.1.5.

3.4.2 Regularization

In order to counter the overfitting problem research on the following regularisation techniques were made and some were integrated in later CNN models which yielded much better accuracies (results in Chapter 4).

Dropout

In a single training iteration, all neurons will have a probability p to be completely eliminated from the network. This forces the neural network to cope with failures and not to rely on the existence of a particular neuron (or set of neurons), but relying more on a general agreement of several neurons within a layer. This is a very simple technique that works quite well already for combating overfiting on its own, without introducing further regularisers. An illustration is shown in Figure 3.25.

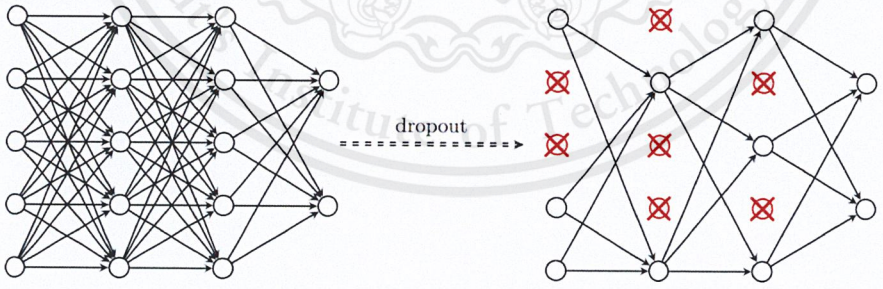


Figure 3.25: Dropout application

Source: [6]

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

L_2 Regularisation

A common underlying cause for overfitting is that our model is too complex (in terms of parameter count) for the problem and training set size at hand. A regulariser aims to decrease complexity of the model while maintaining parameter count the same. L_2 regularisation (a.k.a weight decay) does so by penalising weights with large magnitudes, by minimising their L_2 norm, using a hyperparameter λ to specify the relative importance of minimising the norm to minimising the loss on the training set. Introducing this regulariser effectively adds a cost of $\frac{\lambda}{2}\sum_{i=1}^{k}|w_i|$ to the loss function.

Early Stopping

This is simply to stop training once the validation loss has not decreased for a given number of epochs. This decreases training time.

3.4.3 Weight initialisation

The initial weights of various layers within the network can greatly affect its learning rate. For example, initialising all weights to zero would hinder learning as no weights would initially be active. Initialising the weights in an appropriate way can be influential to how easily the network learns from the training set as it effectively preselects the initial position of the model parameters with respect to the loss function to optimise.

Xavier (a.k.a Glorot) Initialisation

Makes it easier for a signal to pass through the layer during forward as well as backward propagation, for a linear activation. It draws weights from a probability distribution with variance equal to $Var(W) = \frac{2}{n_{in}+n_{out}}$, where n_{in} and n_{out} are the numbers of neurons in the previous and next layer, respectively.

He Initialisation

This is a version of the Xavier initialisation suitable for ReLU activations.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

3.5 Decision tree

The data we used to train our DT consists of the classifications made by the five CNNs on the training set with the plant species as the label. We used the Scikit-learn python machine learning library which contains the DictVectorizer class which transforms lists of feature-value mappings to vectors. After vectorization, our DT contains 5 features per feature vector.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Chapter 4

Experiments

4.1 Flavia dataset

The Flavia dataset consists of 1907 images and 32 plants species. First, each class is split into the number of classes for each feature, then, the full dataset is split into train and test datasets. The train dataset, which is 80% of the full dataset, is used to train the CNN, while the test dataset, which is 20% of the full dataset, is used to validate the trained CNN model.

4.1.1 Morphological characteristics

Table 4.1 contains the morphological characteristics of the dataset we used. The feature types were identified manually by comparing the dataset images to illustrated samples.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.1: Dataset morphological characteristics

Class No.	Lamina	Base	Apex	Margin	Venation
1	Linear	Acute	Acuminate	Entire	Parallel
2	Lanceolate	Acute	Apiculate	Serrulate	Anastomosing
3	Cordate	Cordate	Acuminate	Entire	Palmate
4	Elliptic	Obtuse	Obtuse	Entire	Pinnate
5	Aceriform	Hastate	Acuminate	Double serrate	Palmate
6	Oblanceolate	Attenuate	Acuminate	Entire	Camptodromous
7	Aceriform	Cordate	Acuminate	Serrulate	Palmate
8	Ovate	Obtuse	Acute	Crenate	Pinnate
9	Lanceolate	Acute	Acuminate	Entire	Three-veined from base
10	Lanceolate	Acute	Mucronate	Spinose	Rectipennate
11	Ovate	Obtuse	Acuminate	Serrulate	Anastomosing
12	Obovate	Attenuate	Round	Entire	Pinnate
13	Ovate	Obtuse	Acuminate	Entire	Anastomosing
14	Ovate	Acute	Acuminate	Entire	Brochidodromous
15	Lanceolate	Acute	Acuminate	Crenate	Camptodromous
16	Lanceolate	Acute	Apiculate	Serrate	Anastomosing
17	Acicular	Attenuate	Acute	Entire	Parallel
18	Flabellate	Truncate	Bilobed	Sinuate	Flagellate
19	Ovate	Obtuse	Round	Entire	Brochidodromous
20	Linear	Attenuate	Acute	Entire	Pinnate
21	Linear	Attenuate	Acute	Entire	Parallel
22	Ovate	Acute	Acuminate	Serrulate	Anastomosing
23	Lanceolate	Acute	Acute	Sinuate	Brochidodromous
24	Falcate	Oblique	Acute	Serrate	Pinnate
25	Lanceolate	Acute	Acute	Crenate	Anastomosing
26	Oblong	Acute	Acute	Entire	Brochidodromous
27	Aceriform	Cordate	Acuminate	Serrate	Pedate
28	Ovate	Oblique	Mucronate	Spinose	Pedate
29	Oblanceolate	Acute	Obtuse	Entire	Brochidodromous
30	Deltoid	Cordate	Acute	Crenate	Cladodromous
31	Tulip	Cordate	Emarginate	Entire	Pedate
32	Lanceolate	Attenuate	Acute	Crenulate	Brochidodromous

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

4.1.2 Margin

Leaf margin feature can be divided into eight classes so that dataset is separated into eight classes as shown in Table 4.2. Table 4.3 shows the amount of images in each margin class.

Table 4.2: Margin class separation

Margin class	**Leaf species classes**
Crenate	8, 5, 25, 30
Crenulate	32
Double serrate	5
Entire	1, 3, 4, 6, 9, 12, 13, 14, 17, 19, 20, 21, 26, 29, 31
Serrate	24, 27, 16
Serrulate	2, 7, 11, 22
Sinutate	18, 23
Spinose	10, 28

Table 4.3: Margin dataset

Margin class	**No. of train images**	**No. of test images**	**Full No. of images**
Crenate	186	47	233
Crenulate	44	12	56
Double serrate	44	12	56
Entire	741	186	927
Serrate	139	35	174
Serrulate	176	44	220
Sinutate	93	24	117
Spinose	96	24	120
Total	1519	384	1907

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

4.1.3 Lamina

The lamina feature consists of 14 classes so that dataset is separated into 14 classes as shown in Table 4.4. Table 4.5 shows the amount of images in each lamina class and dataset.

Table 4.4: Lamina class separation

Lamina class	**Leaf species classes**
Aceriform	5, 7, 27
Acicular	17
Cordate	3
Deltoid	30
Elliptic	10, 14, 15, 23, 26
Falcate	24
Flabellate	18
Lanceolate	9, 22, 25, 32
Linear	1, 20, 21
Oblanceolate	6, 29
Oblong	2, 16
Ovate	4, 8, 11, 13, 19, 28
Spathulate	12
Tulip	31

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.5: Lamina dataset

Lamina class	**No. of train images**	**No. of test images**	**Full No. of images**
Aceriform	128	33	161
Acicular	61	16	77
Cordate	57	15	72
Deltoid	51	13	64
Elliptic	237	60	297
Falcate	52	13	65
Flabellate	49	13	62
Lanceolate	176	44	220
Linear	148	37	185
Oblanceolate	95	24	119
Oblong	95	24	119
Ovate	280	70	350
Spathulate	50	13	63
Tulip	42	11	53
Total	1521	386	1907

4.1.4 Base

The base feature consists of seven classes so that dataset is separated into seven classes as shown in Table 4.6. Table 4.7 shows the amount of images in each base class and dataset.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.6: Base class separation

Base class	**Leaf Species Classes**
Acute	1, 2, 9, 10, 14, 15, 16, 22, 23, 25, 26, 29
Cordate	3, 7, 27, 30, 31
Obtuse	4, 8, 11, 19
Oblique	24, 28
Hastate	5
Attenuate	6, 12, 13, 17, 20, 21, 32
Truncate	18

Table 4.7: Base dataset

Base class	**No. of train images**	**No. of test images**	**Full No. of images**
Acute	556	140	696
Cordate	235	59	294
Obtuse	194	49	243
Oblique	96	24	120
Hastate	44	12	56
Attenuate	348	88	436
Truncate	49	13	62
Total	1522	385	1907

4.1.5 Apex

The apex feature consists of eight classes so that dataset is separated into eight classes as shown in Table 4.8. Table 4.9 shows the amount of images in each apex class and dataset.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.8: Apex class separation

Apex class	Leaf species classes
Acuminate	1, 3, 5, 6, 7, 9, 11, 13, 14, 15, 22, 27
Apiculate	2, 16
Obtuse	4, 29
Acute	8, 17, 20, 21, 23, 24, 25, 26, 30, 32
Mucronate	10, 28
Round	12, 19
Bilobed	18
Emarginate	31

Table 4.9: Apex dataset

Apex class	No. of train images	No. of test images	Full No. of images
Acuminate	552	139	691
Apiculate	95	24	119
Obtuse	104	26	130
Acute	486	122	608
Mucronate	96	24	120
Round	99	25	124
Bilobed	49	13	62
Emarginate	42	11	53
Total	1523	384	1907

4.1.6 Venation

The venation feature consists of eleven classes so that dataset is separated into eleven classes as shown in Table 4.10. Table 4.11 shows the amount of images in each margin class and dataset.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.10: Venation class separation

Venation class	**Leaf species classes**
Anastomosing	2, 11, 13, 16, 22, 25
Brochidodromous	14, 19, 23, 26, 29, 32
Camptodromous	6, 15
Cladodromous	30
Flagellate	18
Palmate	3, 5, 7
Parallel	1, 17, 21
Pedate	27, 28, 31
Pinnate	4, 8, 12, 20, 24
Rectipennate	10
three-veined	9

Table 4.11: Venation dataset

Venation class	**No. of train images**	**No. of test images**	**Full No. of images**
Anastomosing	264	66	330
Brochidodromous	276	70	346
Camptodromous	97	25	122
Cladodromous	51	13	64
Flagellate	49	13	62
Palmate	144	36	180
Parallel	156	40	196
Pedate	128	33	161
Pinnate	260	66	326
Rectipennate	52	13	65
Three-veined	44	11	55
Total	1521	386	1907

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

4.2 Convolutional neural network - Architecture

The ConvNet architecture shown in Table 4.12 is the final architecture that is used for all five CNNs in the final project. Where N_C denotes the number of classes there are for that feature. The final softmax layer uses Glorot initialization. All the conv2D layers use He initialization as explained in Section 3.4.2 and L_2 regularisation with λ of 0.0001. The training period is run for a maximum of 150 epochs, but utilizes early stopping with a patience value of five and uses a batch size of 32.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.12: ConvNet architecture for all CNN

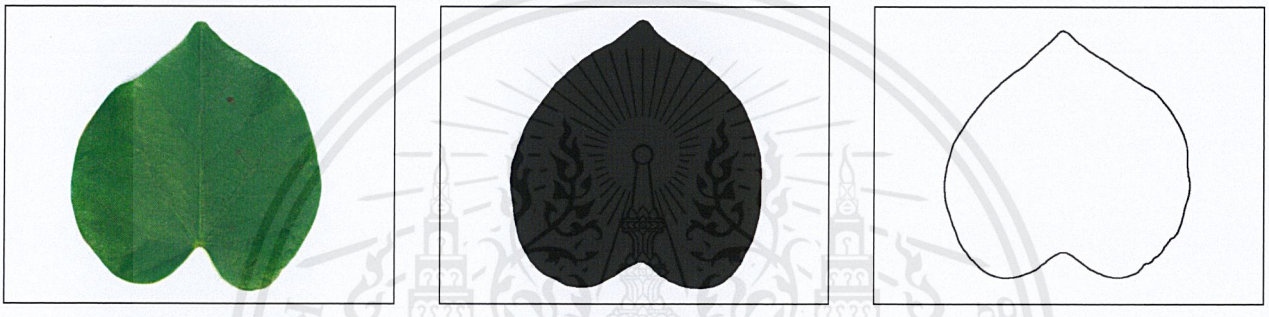
Layer	Filter size-stride	Number of filter	Activation	Dropout rate	Output shape
conv2d	(3×3)-1	32	ReLU	-	300×400
conv2d	(3×3)-1	32	ReLU	-	300×400
maxpooling	(2×2)-2	32	-	-	150×200
conv2d	(3×3)-1	64	ReLU	-	150×200
conv2d	(3×3)-1	64	ReLU	-	150×200
maxpooling	(2×2)-2	64	-	-	75×100
conv2d	(3×3)-1	128	ReLU	-	75×100
conv2d	(3×3)-1	128	ReLU	-	75×100
dropout	-	-	-	20	75×100
maxpooling	(2×2)-2	128	-	-	37×50
conv2d	(3×3)-1	256	ReLU	-	37×50
conv2d	(3×3)-1	256	ReLU	-	37×50
maxpooling	(2×2)-2	256	-	-	18×25
conv2d	(3×3)-1	512	ReLU	-	18×25
conv2d	(3×3)-1	512	ReLU	-	18×25
maxpooling	(2×2)-2	512	-	-	9×12
dropout	-	-	-	25	75×100
fc	-	1	ReLU	-	1×512
dropout	-	-	-	25	75×100
fc	-	1	Softmax	-	1× N_C

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

4.3 Convolutional neural network - Margin

4.3.1 Objective

Test the ability of using full leaf image, black and white leaf shape image, and leaf contour as input in classifying margin feature by measuring the accuracy. In order to find the best preprocessing method, of the three, for margin. Figure 4.1 shows examples of these inputs. The accuracy of the trained model is measured by letting the model classify the test dataset shown in Table 4.3 and calculating the percentage of correct classifications.



(a) Full leaf image (b) Leaf shape image (c) Outline image

Figure 4.1: Example inputs for margin CNN

4.3.2 Setup

The dataset is split according to Table 4.2. The model that received the highest accuracy uses the Conv2D architecture explained in Section 4.2

4.3.3 Results

Tables 4.13 - 4.15 show the accuracy results of the preprocessing methods shown in Figure 4.1.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.13: Full leaf image to classify margin (83.89%)

Actual / Predicted	Crenate	Crenulate	Double Serrate	Entire	Serrate	Serrulate	Sinuate	Spinose	Recall
Crenate	26	0	7	0	0	0	0	0	78
Crenulate	0	12	0	0	0	0	0	0	100
Double Serrate	15	0	5	0	1	0	0	0	0.95
Entire	6	0	0	183	0	2	0	1	95
Serrate	0	0	0	0	22	2	0	0	91
Serrulate	0	0	0	2	0	32	0	1	91
Sinuate	0	0	0	0	1	0	23	0	95
Spinose	0	0	0	0	0	0	0	20	100
Precision	63	100	41	98	91	94	100	90	

Table 4.14: Black and white normal to classify margin (89.06%)

Actual / Predicted	Crenate	Crenulate	Double Serrate	Entire	Serrate	Serrulate	Sinuate	Spinose	Recall
Crenate	30	0	11	2	2	0	0	0	66
Crenulate	0	12	0	0	0	0	0	0	100
Double Serrate	15	0	1	0	0	0	0	0	0.62
Entire	0	0	0	180	0	2	0	1	98
Serrate	1	0	0	2	31	0	0	0	91
Serrulate	1	0	0	2	0	42	0	1	91
Sinuate	0	0	0	0	1	0	24	0	96
Spinose	0	0	0	0	0	0	0	22	100
Precision	63	100	0.83	96	91	95	100	91	

Table 4.15: Outline image to classify margin (85.27%)

Actual / Predicted	Crenate	Crenulate	Double Serrate	Entire	Serrate	Serrulate	Sinuate	Spinose	Recall
Crenate	29	0	11	2	2	0	0	0	65
Crenulate	0	12	0	0	0	0	0	0	100
Double Serrate	12	0	1	0	0	0	0	0	0.76
Entire	0	0	0	177	0	2	0	1	98
Serrate	1	0	0	3	29	0	0	0	87
Serrulate	1	0	0	3	1	40	0	1	86
Sinuate	0	0	0	0	1	0	22	0	95
Spinose	0	0	0	1	0	0	0	20	95
Precision	67	100	0.83	92	87	95	100	90	

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

4.3.4 Conclusion

The preprocessing method that received the highest accuracy is the black and white image, at first it was hypothesized that the outline image would be the best, but it was less than 5% more efficient than the full leaf image at classifying margin type. It seems that the large contrast between the two regions allowed the model to be better at classifying the margin. Table 4.16 shows different methods and their accuracies.

Table 4.16: Different margin preprocessing methods and their accuracies

	Accuracy
Full leaf	83.89%
Black and White	**89.27%**
Outline	85.27%

4.4 Convolutional neural network - Lamina

4.4.1 Objective

Test the ability of using full leaf image, black and white leaf shape image, and leaf contour as input in classifying lamina feature by measuring the accuracy. In order to find the best preprocessing method, of the three, for lamina. Figure 4.1 shows examples of these inputs. The accuracy of the trained model is measured by letting the model classify the test dataset shown in Table 4.5 and calculating the percentage of correct classifications.

Dataset augmentation was also tested to see if this can improve the accuracy achieved in classifying features. Table 4.17 shows a summary of horizontally flipped dataset.

4.4.2 Setup

The dataset is split according to Table 4.4. To test the ability of augmenting the dataset in increasing accuracy, the dataset was also horizontally flipped. The model that received the highest accuracy uses the Conv2D architecture explained in Section 4.2

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.17: Lamina dataset with flip augmentation

Lamina Class	**No. of train images**	**No. of test images**	**Full No. of images**
Aceriform	267	65	322
Acicular	123	31	154
Cordate	115	29	144
Deltoid	102	26	128
Elliptic	475	119	594
Falcate	104	26	130
Flabellate	99	25	124
Lanceolate	352	88	440
Linear	296	74	370
Oblanceolate	190	48	238
Oblong	190	48	238
Ovate	560	140	700
Spathulate	100	26	126
Tulip	84	22	106
Total	3047	767	3814

4.4.3 Results

Tables 4.18 - 4.21 show the accuracy results of the preprocessing methods shown in Figure 4.1, as well as, the augmenting of the dataset.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.18: Full leaf image to classify lamina (72.24%)

Predicted \ Actual	Aceriform	Acicular	Cordate	Deltoid	Elliptic	Falcate	Flabellate	Lanceolate	Linear	Oblanceolate	Oblong	Ovate	Spathulate	Tulip	Recall
Aceriform	33	0	0	0	0	0	0	0	0	0	0	0	0	0	100
Acicular	0	16	0	0	0	0	0	0	0	0	0	0	0	0	100
Cordate	0	0	15	5	0	0	0	0	0	0	0	0	0	3	65
Deltoid	0	0	0	8	0	0	0	8	0	0	0	0	0	0	50
Elliptic	0	0	0	0	59	12	0	24	6	19	11	0	0	0	45
Falcate	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Flabellate	0	0	0	0	0	0	13	0	0	0	0	0	0	0	100
Lanceolate	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
Linear	0	0	0	0	0	1	0	0	31	0	0	0	0	0	96
Oblanceolate	0	0	0	0	0	0	0	1	0	5	0	1	0	0	71
Oblong	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0
Ovate	0	0	0	0	1	0	0	4	0	0	13	70	0	2	77
Spathulate	0	0	0	0	0	0	0	0	0	0	0	0	12	0	100
Tulip	0	0	0	0	0	0	0	0	0	0	0	0	0	6	100
Precision	100	100	100	61	98	0	100	0	83	20	0	95	85	54	

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.19: Black and white image to classify lamina (91.70%)

Predicted \ Actual	Aceriform	Acicular	Cordate	Deltoid	Elliptic	Falcate	Flabellate	Lanceolate	Linear	Oblanceolate	Oblong	Ovate	Spathulate	Tulip	Recall
Aceriform	33	0	0	0	0	0	0	0	0	0	0	0	0	0	100
Acicular	0	16	0	0	0	0	0	0	0	0	0	0	0	0	100
Cordate	0	0	15	0	0	0	0	0	0	0	0	0	0	0	100
Deltoid	0	0	0	13	0	0	0	8	0	0	0	0	0	0	81
Elliptic	0	0	0	0	49	0	0	1	0	1	1	1	0	0	92
Falcate	0	0	0	0	0	12	0	0	0	0	0	0	0	0	100
Flabellate	0	0	0	0	0	0	12	0	0	0	2	0	0	0	85
Lanceolate	0	0	0	0	4	1	0	31	0	0	0	2	0	0	91
Linear	0	0	0	0	1	0	0	0	37	0	0	0	0	0	97
Oblanceolate	0	0	0	0	1	0	0	1	0	23	0	1	0	0	88
Oblong	0	0	0	0	4	0	0	0	0	0	23	0	2	0	79
Ovate	0	0	0	0	1	0	0	3	0	0	0	66	0	0	94
Spathulate	0	0	0	0	0	0	0	0	0	0	0	0	13	0	100
Tulip	0	0	0	0	0	0	0	0	0	0	0	0	0	22	100
Precision	100	100	100	100	81	92	100	70	100	95	88	94	86	100	

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.20: Black and white image to classify lamina with flip (94.65%)

Predicted \ Actual	Aceriform	Acicular	Cordate	Deltoid	Elliptic	Falcate	Flabellate	Lanceolate	Linear	Oblanceolate	Oblong	Ovate	Spathulate	Tulip	Recall
Aceriform	65	0	0	0	0	0	0	0	0	0	0	0	0	0	100
Acicular	0	31	0	0	0	0	0	0	0	0	0	0	0	0	100
Cordate	0	0	29	0	0	0	0	0	0	0	0	0	0	0	100
Deltoid	0	0	0	26	0	0	0	0	0	0	0	0	0	0	100
Elliptic	0	0	0	0	110	0	0	7	1	1	4	2	0	0	88
Falcate	0	0	0	0	0	22	0	2	0	0	0	0	0	0	98
Flabellate	0	0	0	0	0	0	25	0	0	0	0	0	0	0	100
Lanceolate	0	0	0	0	1	3	0	74	0	0	2	1	0	0	91
Linear	0	0	0	0	0	1	0	0	73	0	0	0	0	0	98
Oblanceolate	0	0	0	0	5	0	0	1	0	47	0	0	0	0	88
Oblong	0	0	0	0	1	0	0	0	0	0	41	0	2	0	93
Ovate	0	0	0	0	2	0	0	4	0	0	0	137	0	0	95
Spathulate	0	0	0	0	0	0	0	0	0	0	0	0	24	0	100
Tulip	0	0	0	0	0	0	0	0	0	0	0	0	0	22	100
Precision	100	100	100	100	92	89	100	84	98	97	87	99	92	100	

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.21: Contour image to classify lamina (93.05%)

Actual / Predicted	Aceriform	Acicular	Cordate	Deltoid	Elliptic	Falcate	Flabellate	Lanceolate	Linear	Oblanceolate	Oblong	Ovate	Spathulate	Tulip	Recall
Aceriform	33	0	0	0	0	0	0	0	0	0	0	0	0	0	100
Acicular	0	16	0	0	0	0	0	0	0	0	0	0	0	0	100
Cordate	0	0	15	1	0	0	0	0	0	0	0	0	0	0	93
Deltoid	0	0	0	12	0	0	0	0	0	0	0	0	0	0	100
Elliptic	0	0	0	0	54	0	0	4	1	1	1	2	0	0	85
Falcate	0	0	0	0	1	11	0	0	0	0	0	0	0	0	99
Flabellate	0	0	0	0	0	0	13	0	0	0	0	0	0	0	100
Lanceolate	0	0	0	0	1	2	0	36	0	0	2	1	0	0	85
Linear	0	0	0	0	1	1	0	0	37	0	0	0	0	0	94
Oblanceolate	0	0	0	0	2	0	0	0	0	24	0	1	0	0	88
Oblong	0	0	0	0	0	0	0	2	0	0	21	0	0	0	94
Ovate	0	0	0	0	1	0	0	1	0	0	0	66	0	0	97
Spathulate	0	0	0	0	0	0	0	1	0	0	0	0	13	0	92
Tulip	0	0	0	0	0	0	0	0	0	0	0	0	0	11	100
Precision	100	100	100	92	90	78	100	80	97	96	87	94	100	100	

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

4.4.4 Conclusion

The experiment that achieved the highest accuracy is the model trained with black and white image that was augmented to contain vertically flipped leaf images, however the improvement was less than 1% so it is not quite worth it to augment that dataset. Using black and white images and using outline images received almost the same amount of accuracy this is probably because the shape of the leave can be seen almost same by the CNN using both methods. Table 4.22 shows different lamina preprocessing methods and their accuracies.

Table 4.22: Different lamina preprocessing methods and their accuracies

	Accuracy
Full leaf normal	72.24%
Black and white normal	91.70%
Black and white with flip	**94.65%**
Outline normal	93.05%

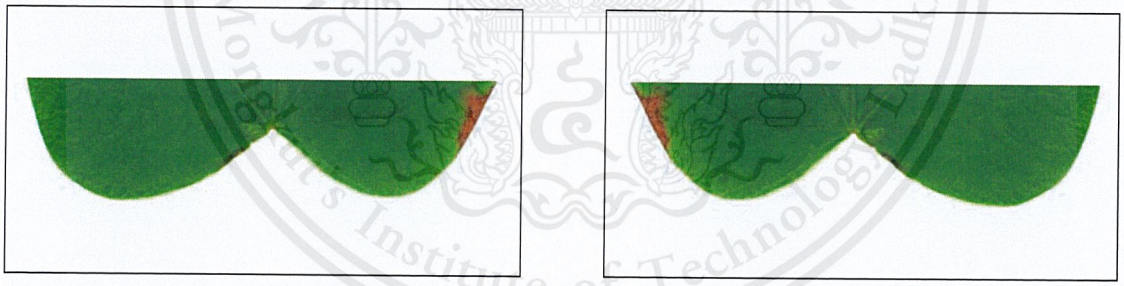
This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

4.5 Convolutional neural network - Base

4.5.1 Objective

Test the ability of using segmented base area image and dataset augmentation in classifying base feature. Figure 4.2 shows an example of a segmented base image input and Figure 4.3 shows an example of horizontal flip. The accuracy of the trained model is measured by letting the model classify the test dataset shown in Table 4.7 and calculating the percentage of correct classification.

Figure 4.2: Example segmented base image input



(a) Original image (b) After augmentation

Figure 4.3: An example of flipping

4.5.2 Setup

The dataset is split according to Table 4.6. The model that received the highest accuracy uses the Conv2D architecture explained in Section 4.2

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

4.5.3 Results

Table 4.23: Segmented normal to classify base (90.13%)

Predicted \ Actual	Acute	Attenuate	Cordate	Hastate	Oblique	Obtuse	Truncate	Recall
Acute	122	5	0	0	4	4	0	90
Attenuate	17	82	0	0	0	4	0	79
Cordate	0	0	57	0	0	0	0	100
Hastate	0	0	2	12	0	0	0	85
Oblique	0	0	0	0	20	0	0	100
Obtuse	0	0	0	0	0	41	0	100
Truncate	0	0	0	0	0	0	13	100
Precision	87	94	96	100	83	83	100	

Table 4.24: Segmented with flip to classify base (93.47 %)

Predicted \ Actual	Acute	Attenuate	Cordate	Hastate	Oblique	Obtuse	Truncate	Recall
Acute	259	8	0	0	4	4	0	94
Attenuate	15	163	0	0	0	4	0	89
Cordate	0	0	118	0	0	0	0	100
Hastate	0	0	0	23	0	0	0	100
Oblique	0	0	0	0	40	0	0	100
Obtuse	4	4	0	0	4	90	0	88
Truncate	0	0	0	0	0	0	25	100
Precision	93	93	100	100	83	99	100	

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.25: Segmented with duplicate and flip to classify base (96.47 %)

Actual / Predicted	Acute	Attenuate	Cordate	Hastate	Oblique	Obtuse	Truncate	Recall
Acute	269	1	0	0	2	1	0	98
Attenuate	3	164	0	0	0	0	0	98
Cordate	0	0	118	1	0	0	0	98
Hastate	0	0	0	22	0	0	0	98
Oblique	0	0	0	0	44	0	0	98
Obtuse	4	4	0	0	2	97	0	98
Truncate	0	0	0	0	0	0	25	98
Precision	97	97	100	95	91	98	100	

4.5.4 Conclusion

Augmenting the dataset to contain vertically flipped images was able to increase accuracy by almost 3%, which is not a large amount but if the dataset itself was originally larger maybe this would make a more significant improvement. Table 4.26 shows different base preprocessing methods and their accuracies.

Table 4.26: Different base preprocessing methods and their accuracies

	Accuracy
Segmented normal	90.13%
Segmented with flip	93.73%
Segmented with flip and dup	**96.47%**

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

4.6 Convolutional neural network - Apex

4.6.1 Experiment 1 : Segmented normal

Objective

Test the ability of using segmented base in input in classifying apex feature. Figure 4.4 shows an example of a segmented apex image input. The accuracy of the trained model is measured by letting the model classify the test dataset shown in Table 4.9 and calculating the percentage of correct classification.

Figure 4.4: Example segmented apex input

Setup

The dataset is split according to Table 4.8. The model that received the highest accuracy uses the Conv2D architecture explained in Section 4.2

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.27: Segmented to classify apex (92.96 %)

Actual / Predicted	Acuminate	Acute	Apiculate	Bilobed	Emarginate	Mucronate	Obtuse	Round	Recall
Acuminate	134	7	5	0	0	1	0	0	99
Acute	0	112	1	0	0	1	2	0	96
Apiculate	0	0	18	0	0	0	0	0	100
Bilobed	0	0	0	13	0	0	0	0	100
Emarginate	0	0	0	0	11	0	0	0	100
Mucronate	0	1	0	0	0	22	0	0	95
Obtuse	0	1	0	0	0	0	22	0	95
Round	1	1	0	0	0	0	2	25	86
Precision	99	99	75	100	100	99	84	100	

Table 4.28: Segmented with flip to classify apex (96.87 %)

Actual / Predicted	Acuminate	Acute	Apiculate	Bilobed	Emarginate	Mucronate	Obtuse	Round	Recall
Acuminate	132	1	0	0	0	0	1	0	98
Acute	7	120	1	0	0	1	0	0	93
Apiculate	0	0	23	0	0	0	0	0	100
Bilobed	0	0	0	13	0	0	0	0	100
Emarginate	0	0	0	0	11	0	0	0	100
Mucronate	0	0	0	0	0	23	0	0	100
Obtuse	0	1	0	0	0	0	25	0	96
Round	0	0	0	0	0	0	0	25	100
Precision	94	98	95	100	100	95	96	100	

Table 4.29: Segmented with duplicates and flip to classify apex (95.57%)

Actual / Predicted	Acuminate	Acute	Apiculate	Bilobed	Emarginate	Mucronate	Obtuse	Round	Recall
Acuminate	127	1	0	0	0	0	0	0	99
Acute	9	119	0	0	0	0	0	0	92
Apiculate	0	0	24	0	0	0	0	0	100
Bilobed	0	0	0	13	0	0	0	0	100
Emarginate	0	0	0	0	11	0	0	0	100
Mucronate	0	0	0	0	0	23	0	0	100
Obtuse	0	1	0	0	0	0	25	0	96
Round	1	1	0	0	0	0	1	25	89
Precision	92	97	100	100	100	95	96	100	

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

4.6.2 Conclusion

Both the augmentation methods were able to make some differences, but the flip was able to make quite a significant improvement of around 5%. It is worth the time to augment the dataset of segmented apex images to classify apex. Table 4.30 shows different apex preprocessing methods and their accuracies.

Table 4.30: Different apex preprocessing methods and their accuracies

	Accuracy
Segmented normal	92.96%
Segmented with flip	**96.87%**
Segmented with flip and dup	95.57%

4.7 Convolutional neural network - Venation

4.7.1 Objective

Test the ability of input of segmented images of the leaf area image with resolution 65×65 pixels explained in Section 3.2.4. Figure 4.5 shows an example of blocks input. Another method we tested is the sobel edge image an example is shown in Figure 4.6.

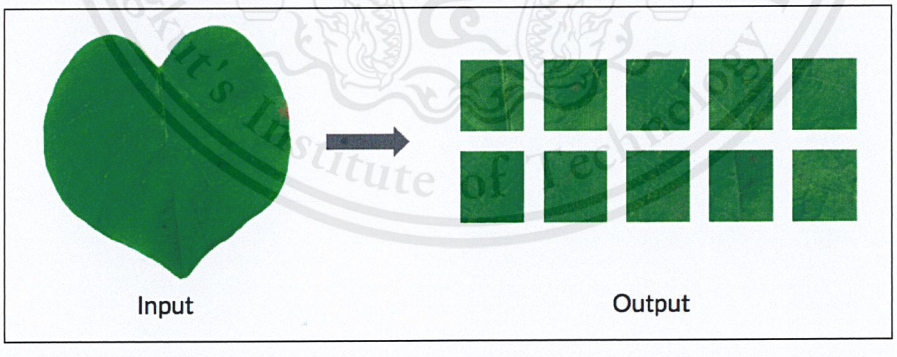


Figure 4.5: Example venation blocks input

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

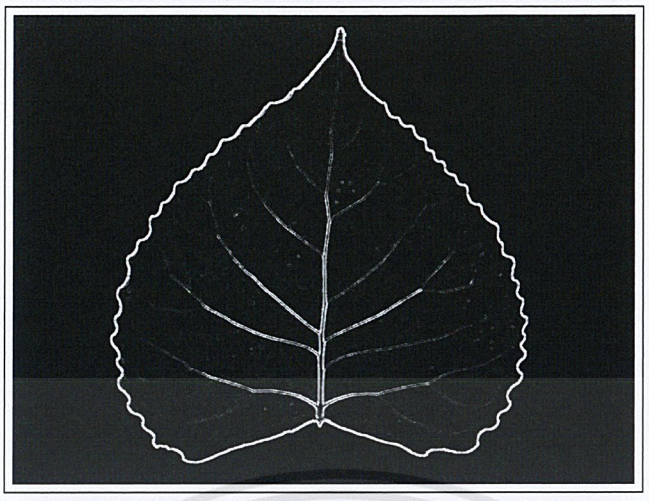


Figure 4.6: Example sobel edge image input

Setup

The dataset is split according to Table 4.10. The model that received the highest accuracy uses the Conv2D architecture explained in Section 4.2

4.7.2 Results

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.31: Small segmented block to classify venation (12.9%)

Predicted \ Actual	Anastomosing	Brochidromous	Camptodromous	Cladodromous	Flagellate	Palmate	Parallel	Pedate	Pinnate	Rectipennate	Three-veined	Recall
Anastomosing	28	16	2	2	0	0	0	1	3	0	0	50
Brochidromous	401	546	220	92	106	246	139	257	389	129	109	20
Camptodromous	0	0	22	0	0	0	0	0	0	0	0	100
Cladodromous	0	0	0	0	0	0	0	0	0	0	0	0
Flagellate	0	0	0	0	0	0	0	0	1	0	0	0
Palmate	11	4	8	10	0	3	0	11	12	0	3	0
Parallel	0	0	0	0	0	0	36	0	0	0	0	100
Pedate	0	0	0	0	0	0	0	0	0	0	0	0
Pinnate	240	140	17	34	10	31	8	35	239	10	15	30
Rectipennate	0	0	0	0	0	0	0	0	0	0	0	0
Three-veined	0	0	0	0	0	0	0	0	0	0	0	0
Precision	0	77	0	0	0	0	19	0	37	0	0	

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.32: Sobel egde detection to classify venation (91.29%)

Actual / Predicted	Anastomosing	Brochidromous	Camptodromous	Cladodromous	Flagellate	Palmate	Parallel	Pedate	Pinnate	Rectipennate	Three-veined	Recall
Anastomosing	65	1	0	0	0	0	0	0	0	0	0	98
Brochidromous	0	67	0	0	0	0	0	0	0	0	0	100
Camptodromous	0	0	25	0	0	0	0	0	0	0	0	100
Cladodromous	0	0	0	13	0	0	0	0	0	0	0	100
Flagellate	0	0	0	0	13	0	0	0	0	0	0	100
Palmate	0	0	0	0	0	34	0	2	0	0	0	94
Parallel	0	0	0	0	0	0	40	0	0	0	0	100
Pedate	0	0	0	0	0	0	0	33	0	0	0	100
Pinnate	2	0	0	0	0	0	4	0	60	0	0	90
Rectipennate	0	0	0	0	0	0	1	0	0	12	0	92
Three-veined	0	1	1	0	0	0	0	0	0	0	9	81
Precision	97	97	86	100	100	100	88	94	100	100	100	

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

4.7.3 Conclusion

Sobel edge detection was able to capture the characteristics of the venation much better than the small segmented blocks. This is probably because the whole shape of the venation is needed to differentiate between venation classes. Table 4.34 shows different venation preprocessing methods and their accuracies.

Table 4.33: Different venation preprocessing methods and their accuracies

	Accuracy
Segmented blocks	12.9%
Sobel	**91.29%**

4.8 Species classification

CNNs combined with decision tree

The decision tree classifies the leaf species, thus it is trained on the classifications made by the five CNNs on the training set. The classification process is done by combining the feature classifications from the five CNNs and inputting it into the trained decision tree.

In total there are 3,041 training samples (this dataset has been horizontally flipped and does not include the local plant species). This method achieved the highest accuracy of 92.77%

Competitor CNN

In order to compare our classification method, we trained a competitor CNN that uses an input of a full leaf image, an example is shown in Figure 4.1a, to classify the leaf species. This CNN uses the same architecture used for all our feature-specific CNNs, which is detailed in Table 4.12. This method achieved an accuracy of 97.02%

4.8.1 Conclusion

Another method we used to compare is the one used by the Flavia dataset's own creator which uses numerical features and a probabilistic neural network (PNN).

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.34: Different venation preprocessing methods and their accuracies

	Accuracy
CNNs combined with decision tree	**92.77%**
Competitor CNN	97.02%
PNN	90%

4.9 Final results

4.9.1 Local dataset

We added one local plant species so that we can test our model and mobile application live. We added 40 images to the dataset. Figure 4.7 show examples of the local plant that we added which is the Phut Phuket plant. Its lamina, base, apex, margin, and venation are Lanceolate, Acute, Acuminate, Entire, and Brochidodromous, respectively.

Figure 4.7: Examples of Phut Phuket leaves

4.9.2 Convolutional neural network

The preprocessing method that achieved the highest accuracy was chosen as the method for that feature. After finalizing the preprocessing methods, the five CNNs were retrained using those methods on a dataset that contained the new local plant species mentioned in Section 4.9.1. Since the experiment results show that horizontally flipping dataset images to increase dataset information always improved accuracy, this augmentation technique was applied to the whole dataset. Table 4.35 are the accuracies of the CNNs that was included in the final project.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Table 4.35: CNN final accuracies

Feature Name	Accuracy in classifying that feature
Lamina	97.33%
Margin	90.60%
Base	95.18%
Apex	86.69%
Venation	97.97%

4.9.3 Decision tree

After the final CNNs were combined with the decision tree, the overall accuracy of the classification model when tested with our testing set is 95.45%.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Chapter 5

Conclusion

This thesis proposes a plant species classification system that focuses on the plants' leaves. The system is a server-less mobile application that allows the user to take photos or pick an image through it and the system will classify the leaf species for the user. The system composes of two main components the mobile application and the classification model it contains.

The thesis's main contribution is the classification model which comprises of five convolutional neural networks (CNN) connected by a decision tree. Each CNN classifies a single feature of the leaf. The five features are lamina, margin, apex, base and venation. Each feature-specific CNN requires a different type of input so that each input allows the CNN to efficiently classify that feature type. The inputs are a result of processing the original leaf input image using computer vision. The different processing methods used are explained in Section 3.3. Several processing techniques were implemented and compared to find the one that yielded the highest accuracy. Those results are documented in Chapter 4.

Then the classification from each CNN will be used by the decision tree to classify the leaf species. The decision tree helps to determine the shortest path to the classification. Additionally, the tree structure is similar to the dichotomous key, a tool used by botanists to identify plants and other items in the natural world. Because of this similarity, our method removes the black-box nature of deep learning and makes the classification process easier to understand and allows for this method to support any number of plant species. In the end, our method achieved a satisfying accuracy of 95.45%.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

5.1 Future work

5.1.1 Mobile application

Currently, the system requires the user to choose the apex and base points to allow the system to rotate the leaf correctly, and so there is the possibility of user error. In the future, the system could require zero input from the user and use computer vision to locate the apex and base points of the leaf automatically to improve user experience.

Suggestions were made to create a platform to allow users to make reports if wrong classifications were made so that information can be gathered to allow improvements to the classification model. Also a way for users to suggest new plant species if they found one that is not in the dataset in order to increase plant database.

5.1.2 Classification model

If the dataset was updated, for example, by adding more leaf species. The classification model will need to be retrained from the beginning, so we think that further research can be made on transfer learning to allow shorter retraining for the CNNs and decision tree.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

Bibliography

[1] P.K. Bhattacharyya and K. Bhattacharyya, "Comparison of the angiosperm phylogeny group classification (2009) with that of the system of Takhtajan (2009) and a note on the chase and reveal (2009), Haston etal. (2009), and Mabberley (2008)," *Academia*, URL: https://www.academia.edu/3097032/comparion_of_the_angiosperm_phylogeny_group_classification_2009_with_that_of_the_system_of_Takhtajan_2009_and_a_note_on_the_chase_and_reveal_2009_Haston_etal._2009_and_Mabberley_2008_, 2017. [Online; accessed December 13,2017].

[2] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2004.

[3] Z. Huacheng and X. Wu, "Improvement of SLIQ algorithm and its application in evaluation," *Third International Conference on Genetic and Evolutionary Computing*, pp.77-80, 2009.

[4] K. Mehmed, *Decision Trees and Decision Rules*, 1st edition, Wiley-IEEE Press, 2011.

[5] T. Mitchell. *Machine Learning* , 1st edition, McGraw-Hill, New York, NY, USA, 1997.

[6] V. Petar, "Deep learning for complete beginners: convolutional neural networks with keras," *Cambridgespark*, URL: https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html, 2017 [Online; accessed March 12,2018].

[7] C. Qingyun, "Research on incremental decision tree algorithm," *Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology*, pp.303-306, 2011.

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use

[8] J. Ross Quinlan. *C4.5: programs for machine learning* , 1st edition, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 1993.

[9] -, "Leaf margins, tips, and bases," *Lifeofplant*, URL: http://lifeofplant.blogspot.com/2011/03/leaf-margins-tips-and-bases.html, 2002.

[10] Apple, "vImage Programming Guide," *Performing Convolution Operations*, URL: https://developer.apple.com/library/content/documentation/Performance/Conceptual/vImage/Introduction/Introduction.html, 2016. [Online; accessed December 13,2017].

[11] Mathworks, "Convolutional neural network" *MATLAB & Simulink*, URL: https://www.mathworks.com/discovery/convolutional-neural-network.html [Online; accessed December 13,2017].

[12] -, "List of systems of plant taxonomy," *List of systems of plant taxonomy*, URL: https://en.wikipedia.org/wiki/List_of_systems_of_plant_taxonomy, 2017. [Online; accessed December 13,2017].

This material is reserved for educational use only, not allowed for commercial use.
Forbidden to modify the content, and cite the document when use