CAN THO UNIVERSITY
COLLEGE OF INFORMATION AND COMMUNICATION TECHNOLOGY

GRADUATION THESIS
BACHELOR OF ENGINEERING IN
INFORMATION TECHNOLOGY
(HIGH-QUALITY PROGRAM)

PLANT IDENTIFICATION SYSTEM

Student: Nguyen Duy Khang
Student ID: B1910652
Class: 2019-2023 (K45)
Advisor: Associate Prof. Dr. Nguyen Thai Nghe

Can Tho, October 2023

CAN THO UNIVERSITY
COLLEGE OF INFORMATION AND COMMUNICATION TECHNOLOGY
DEPARTMENT OF INFORMATION TECHNOLOGY

GRADUATION THESIS
BACHELOR OF ENGINEERING IN
INFORMATION TECHNOLOGY
(HIGH-QUALITY PROGRAM)

PLANT IDENTIFICATION SYSTEM

Student: Nguyen Duy Khang
Student ID: B1910652
Class: 2019-2023 (K45)
Advisor: Associate Prof. Dr. Nguyen Thai Nghe

Can Tho, October 2023

ADVISOR'S COMMENTS

Can Tho, December 4th 2023

Advisor

EXAMINER'S FEEDBACK

Can Tho, December 4th 2023

Examiner

ACKNOWLEDGEMENT

I would like to sincerely thank my advisor Associate Prof. Dr. Nguyen Thai Nghe for his wholehearted guidance, assistance, and favorable conditions for me to complete my thesis. During the time of implementing the topic, the support and suggestions from lecturers are always the motivation for me to try and improve myself. Please send my lecturers my most sincere thanks.

Next, thank you very much to the lecturers at the College of Information and Communication Technology - Can Tho University, especially the teachers of the Faculty of Information Technology. Over the past 4 years, the lecturers have dedicated themselves to teaching and imparting knowledge as well as life experiences so that I can afford to complete the thesis well.

Last but not least, I would like to thank the members of the class 45 high quality Information Technology, and the members of the thesis topic group who enthusiastically helped and commented when I encountered difficulties.

Although I have tried my best to complete the thesis on time as expected, there are still many shortcomings inevitable. I look forward to receiving enthusiastic sympathy and suggestions from our lecturers and friends.

Can Tho, December 4th 2023

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

No.	Abbreviation	Origin word
1	CNN	Convolutional Neural Network
2	CPU	Central Processing Unit
3	GPU	Graphics Processing Unit
4	RAM	Random Access Memory
5	API	Application Programming Interface
6	GB	Gigabyte
7	RGB	Red, Green, Blue
8	2D	Two-dimensional
9	ID	Identification
10	No.	Number

TABLE OF CONTENTS

ABSTRACT

This research focuses on developing a plant recognition system using the deep learning model Inception V3 and deploying it on an iOS platform mobile application using the Swift programming language. To achieve this goal, I utilized the extensive PlantNet-300K dataset containing diverse plant images collected from various sources.

The Inception V3 model has proven to be a powerful tool for image recognition and classification. I trained this model on the PlantNet-300K dataset to create a highly accurate plant recognition model.

In addition to deploying the model in Swift, I have also built a user-friendly mobile application for plant recognition. This application allows users to capture or upload images of a plant species and returns results regarding the plant's name and detailed information.

The results of this research demonstrate the synergy between deep learning technology and mobile applications, making it easy for users to identify and learn about the plant species in their surroundings. This represents a significant step in exploring and preserving the planet's biodiversity.

This study contributes to the field of plant recognition and the development of practical applications for environmental education and nature conservation.

CHAPTER 1: INTRODUCTION

In this chapter, I will present an overview of the research situation of the plant classification project and the reason why I chose this topic along with the objectives, objects, and research scope of the topic. Finally, I list the contents of the study by period and some related studies.

1. The purpose of the study

The primary aim of this study is to develop an effective plant recognition system utilizing the Inception V3 deep learning model and deploying it on an iOS platform mobile application using the Swift programming language. By leveraging the PlantNet-300K dataset, I seek to create a robust model capable of accurately identifying various plant species. The goal is to provide a user-friendly tool that empowers individuals to quickly and accurately recognize plants in their environment, contributing to environmental awareness and conservation efforts.

2. The problem of the study

Identifying plant species can be a challenging task for individuals without expert botanical knowledge. Manual identification methods are often time-consuming and do not always yield accurate results. Additionally, with the growing need to conserve biodiversity, the need for effective and accessible plant identification tools is also increasing. This research addresses the need for a reliable and user-friendly plant identification system that leverages modern deep-learning techniques and is deployed on mobile phones, serving both enthusiast and expert in the field of botany and environmental conservation.

3. Related work

Currently, there is a substantial body of research on plant classification. The study conducted by Kaya et al. [1] used a convolutional neural network architecture with three layers, as well as VGG-16 and AlexNet networks, to classify plant species in four different datasets: Flavia, Swedish Leaf, UCI Leaf, and Plantvillage. For this, two types of transfer learning were employed. In the first method, the models were pre-trained on the ImageNet dataset and subsequently fine-tuned on each of the plant datasets. In the second method, three of the plant datasets were used for pre-training and the remaining for fine-tuning.

The study by Ahmad et al. [2] aims to detect and classify weeds in corn and soybean plantations. Detection was performed using YOLOv3, while the VGG16, ResNet50, and Inception V3 networks were used for classification. All models were pre-trained with ImageNet. The dataset used in the research was created from images

collected by the authors and images acquired through google searches, totaling 462 images distributed among four weed species.

The approach proposed by Pratondo et al. [3] used the VGG19 and Inception V3 networks, pre-trained with ImageNet, to classify two varieties of Curcuma. The dataset used in the research was constructed from photographs taken with a smartphone in markets, totaling 647 images. The study by Chen et al. [4] was developed for weed identification in cotton plantations. Several models were evaluated, and the ResNeXt101 model, pre-trained with the ImageNet dataset, showed the best performance. A dataset with 5187 images of 15 weed species was created, and the images were collected under natural light conditions.

Finally, in the study by Castro et al. [5], they propose a solution to address the common challenge of overfitting in deep neural networks through the Simultaneous Learning method, a regularization approach drawing on principles of Transfer Learning and Multi-task Learning. The test results on the PlantNet-300K dataset with the ResNet50 and Inception V3 models, using customized parameters λ and dropout, yield accuracy rates of 89.66 and 83.62, respectively.

4. Research Contents

The research content is divided into 5 main phases including:

Stage 1: Theoretical research

The content of theoretical research focuses on understanding the following documents:

- Basic knowledge of machine learning and neural networks.
- Research on the Inception V3 model, Convolutional Neural Network, and how to combine the Inception V3 model with Convolutional Neural Network.
- Research methods and processes for developing mobile applications on the iOS platform.

Stage 2: Studying the training data:

- Research available training data sets for plant classification problems.
- Explore PlantNet-300K dataset.

Stage 3: Building a model plant classification.

Stage 4: Experiment and evaluate the results.

Stage 5: Build a plant identification application on the iOS platform.

5. Outline

CHAPTER 1: INTRODUCTION: Introduce and give directions to research the plant classification problem.

CHAPTER 2: LITERATURE REVIEW: An overview of Inception V3 and Convolutional Neural Network.

CHAPTER 3: PLANT CLASSIFICATION MODEL: Overview of the system architecture to build.
CHAPTER 4: APPLICATION SYSTEM DESCRIPTION: Build a plant identification application on the iOS platform.
CHAPTER 5: TEST REVIEW: Description of test objectives, test scenarios, and test results.
CHAPTER 6: CONCLUSION: Provide research results, assessment, and direction.

CHAPTER 2: LITERATURE REVIEW

In this chapter, I present the theory of machine learning, Convolutional Neural Networks, Inception V3 model, model evaluation methods, and technologies used to build the system.

1. Machine learning

Currently, many problems can be applied with machine learning technology to solve such as human face recognition, natural language processing, intelligent robot development, etc. The essence of machine learning technology is training. Train the computer to become smarter in the direction that people want it by building computational models capable of self-improving parameters based on the data it encounters. To get a machine learning model that can be applied to real-world problems with high accuracy, I need to expose it and process through a lot of data. Doing the above is often referred to as training a machine learning model.

The process of training machine learning models can be divided into two forms: supervised learning and unsupervised learning.

- Supervised learning is a form of model training used in problems where people already know the answers, then using data sets with answers to put them into the model for processing. The parameters of the model will be automatically adjusted so that the output data processing results match the given answer.
- Unsupervised learning is often used in problems where the answer is unknown. A specific example is the classification problem, the model will be trained with a data set with no given answer, the goal of the problem is to classify the objects of that data set into classes based on the object characteristics.

2. Convolutional Neural Network

The Convolutional Neural Network [6] is commonly applied to image-processing tasks. The network's input consists of 2D matrices, where each value in the matrix typically represents a pixel. The neural network architecture can be summarized through an illustrative diagram, as shown in Figure 1.

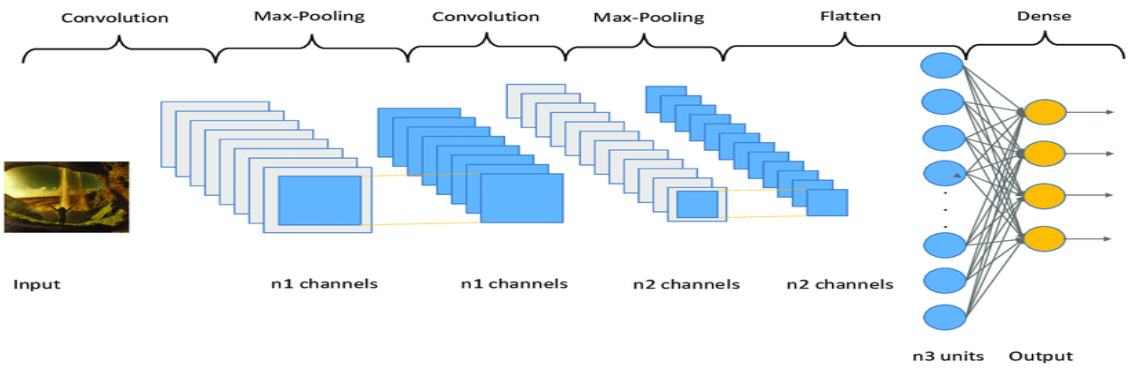

Figure 1. Convolutional Neural Network Architecture[1]

2.1. Convolutional layer

Filter (also known as Kernel) in a CNN is typically a small square matrix that acts as a sliding window over matrix-shaped data. Figure 2 below illustrates various filters in a neural network.

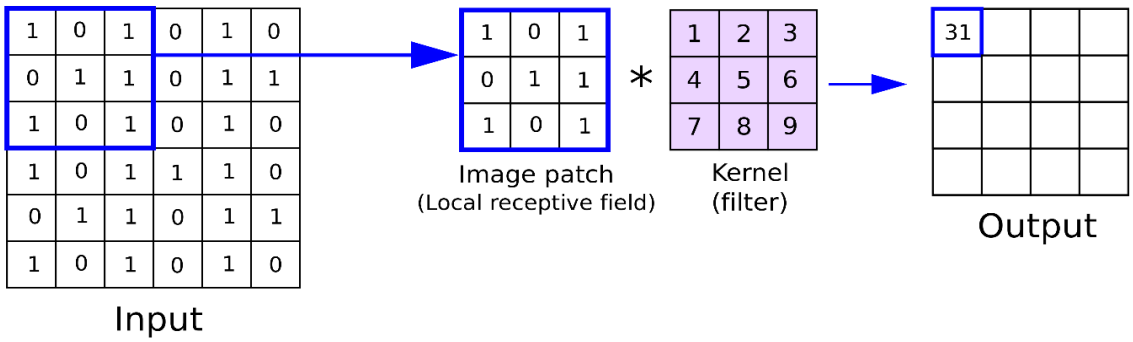

Figure 2. A Convolution Filter in Convolutional Neural Networks[2]

When a filter slides over an input matrix, it produces a new output matrix. Each position where the filter passes is calculated as illustrated in the diagram (each value in the matrix is multiplied by the corresponding value in the filter, and the sum of these multiplications results in a value in the corresponding position of the output matrix).

If the size of the input matrix is kept the same, the resulting output matrix will be smaller. To address this issue, one can add edges with values of 0 around the outside of the input matrix to keep the input and output sizes the same. This method is called padding.

[1] https://www.mdpi.com/1424-8220/20/4/1214/htm
[2] https://anhreynolds.com/blogs/cnn.html

A single convolutional layer can have multiple filters, and each filter will have different values representing a specific feature in the image. Passing an input matrix through a convolutional layer with multiple filters results in N output matrices, corresponding to the N filters in that layer. After each convolutional layer, I extract image features based on the set of filters used. The process of filters capturing diagonal features in an image is illustrated in Figure 3.

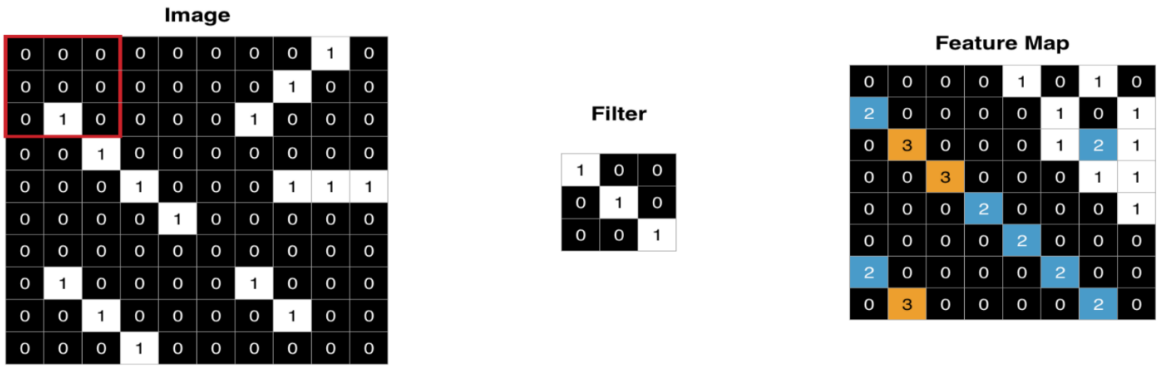

Figure 3. A Filter Capturing Diagonal Features in an Image[3]

2.2. Pooling layer

Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after the max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

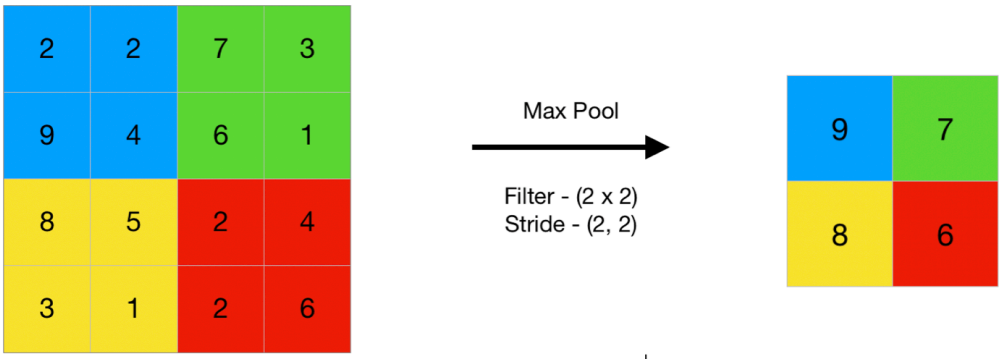

Figure 4. A pictorial example of max pooling[4]

Average pooling computes the average of the elements present in the region of the feature map covered by the filter. Thus, while max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in a patch.

[3] https://cs231n.github.io/convolutional-networks/
[4] https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

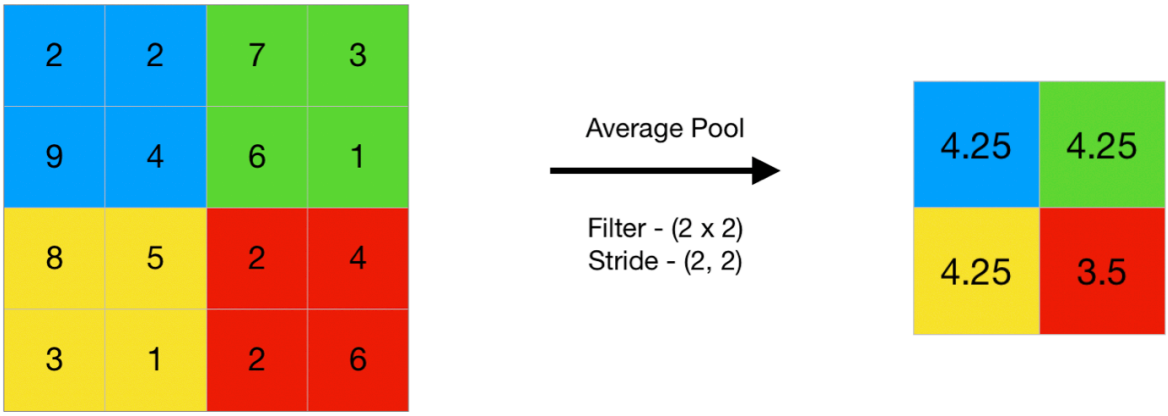

Figure 5. A pictorial example of average pooling[5]

2.3. Flatten layer

The flattened layer is responsible for transforming the input data from a two-dimensional matrix into one-dimensional data to feed into a neural network, typically for recognition and classification tasks. Typically, the input data for the flattened layer consists of matrices that have been processed by convolution and max pooling layers. These matrices carry information about the features of the image that have been extracted from previous layers.

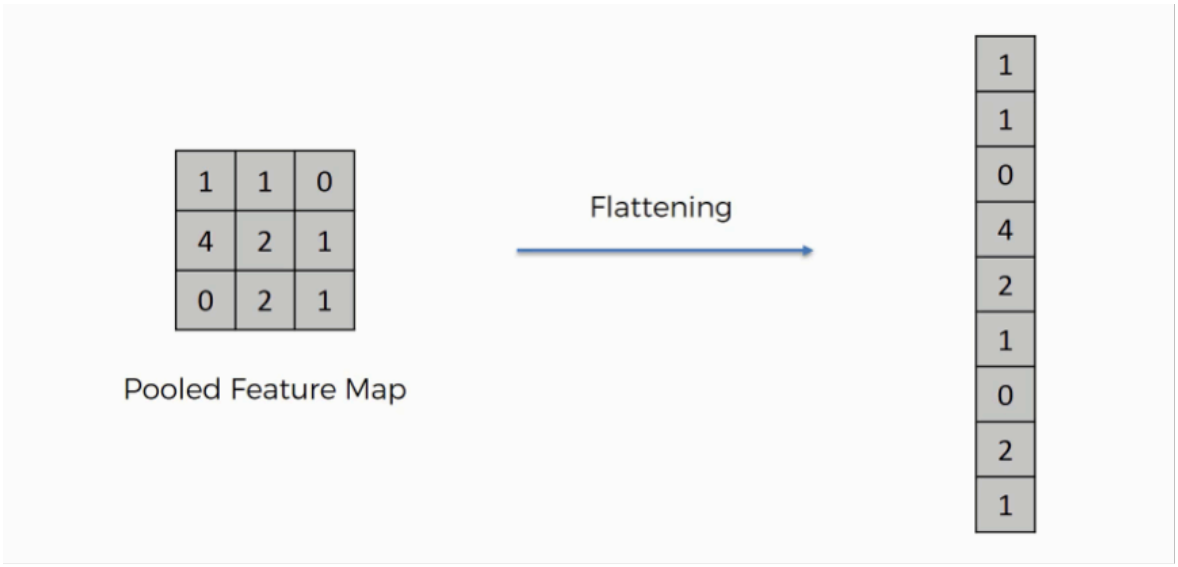

Figure 6. A pictorial example of flattening[6]

2.4. Dense layer

Dense layers, at their core, are basic neural networks where each neuron is connected to all the neurons in the layer before it. Dense layers are typically the final layers in a convolutional neural network, and their purpose is to process classification

[5] https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/
[6] https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening

tasks based on the features that have been extracted from the images in the preceding layers.

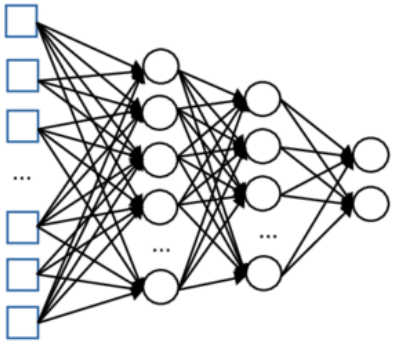

Figure 7. A pictorial example of a dense layer

3. Inception model

The Inception model is an architectural framework used to address image recognition and detection tasks. The model comes in multiple versions. In Inception V1 [7], the concept is based on the use of multiple filters with different sizes at the same level. As a result, instead of having deep layers, the model has parallel layers, which makes it "wider" rather than "deeper." The basic module of the Inception V1 model is comprised of four parallel layers: 1×1 convolution, 3×3 convolution, 5×5 convolution, and 3×3 max pooling. The basic module of the Inception V1 model is illustrated in Figure 8.

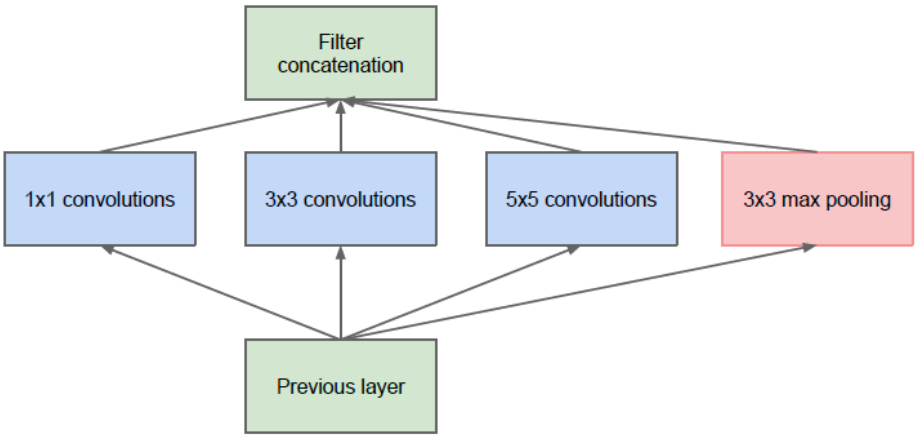

Figure 8. The initial form (Naïve Form) of the Inception V1 model[7]

One of the limitations of this Naïve Form is that the 5×5 convolution layer is computationally expensive, which means it requires more time and computational

[7] https://iq.opengenus.org/inception-v3-model-architecture/

resources. To address this, the authors added a 1×1 convolution layer before each convolution layer.

The Inception V3 [8] model is an advanced and optimized version of the V1 model. Inception V3 employs several techniques to optimize the network for better adaptability. It consists of a total of 42 layers and achieves lower error rates compared to its predecessors. The key modifications in the Inception V3 model include:

- Splitting convolution layers into smaller convolution layers.
- Splitting Convolution layers into asymmetric convolution layers.
- Incorporating additional auxiliary classifiers.
- Efficiently reducing data size.

The model's architecture is built incrementally, as follows:

- Initial data transformation (Factorized Convolutions): This reduces computation effectively by reducing the number of parameters related to the network.
- Reducing the size of convolution layers (Smaller convolutions): For instance, replacing a 5×5 filter with two 3×3 filters instead to reduce the number of parameters.
- Reducing asymmetric parameters (Asymmetric convolutions): A 3×3 convolution layer can be replaced by a 1×3 layer followed by a 3×1 layer.
- Auxiliary classifiers: In Inception V3, auxiliary classifiers are applied for regularization, reducing the risk of model overfitting.

Data size reduction: Data size reduction is typically achieved through pooling operations.

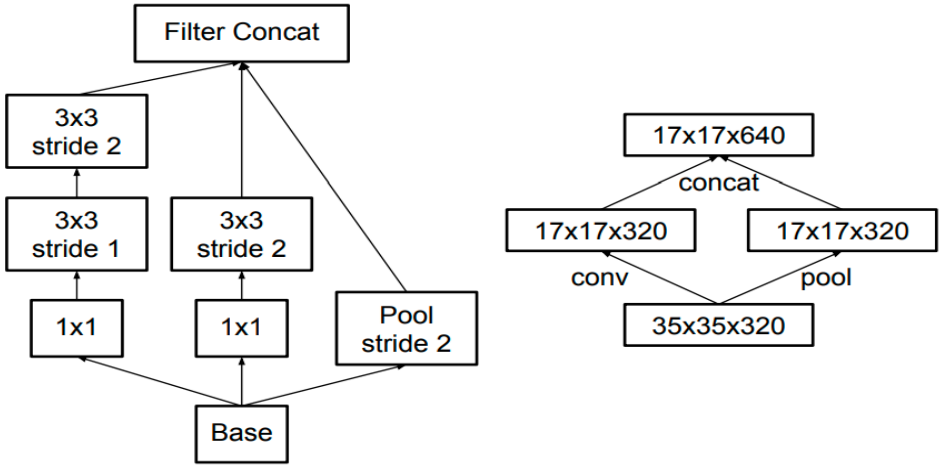

Figure 9. Data size reduction process[8]

[8] https://blog.paperspace.com/popular-deep-learning-architectures-resnet-inceptionv3-squeezenet/

All these concepts are integrated into the final architecture. Deep learning techniques require a standard training dataset to train, creating a large number of values and large parameter sets for the model to classify the problem's features effectively. Inception V3 is an image recognition model that has been demonstrated to achieve high accuracy on the ImageNet dataset. The architecture of the model is depicted in Figure 10.

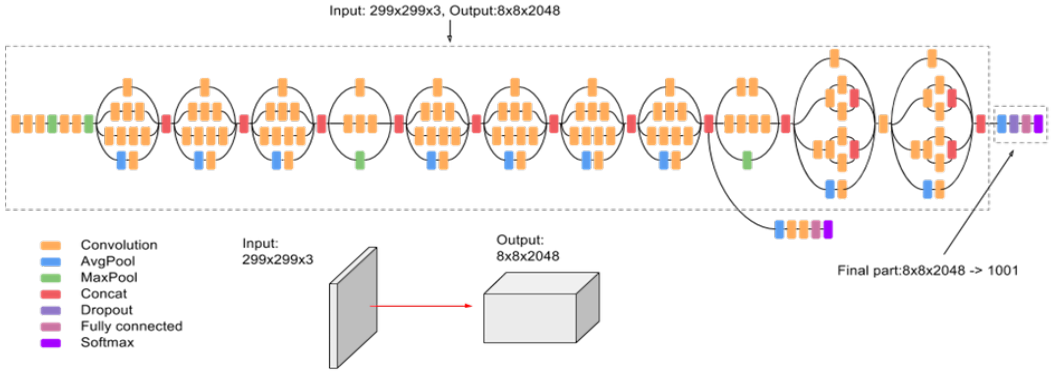

Figure 10. The architecture of the Inception V3 model[9]

The model is constructed with specialized blocks, both symmetric and asymmetric, consisting of convolution, average pooling, max pooling, concat, dropout, and fully connected layers. The output size of each module matches the input size of the subsequent module. The model offers improved accuracy and reduced computational cost compared to the previous versions.

4. Model evaluation method

Accuracy [9] is a fundamental metric for evaluating the performance of a classification model. It is calculated by dividing the number of correct predictions by the total number of predictions. In the context of a multi-class classification model, accuracy can be calculated as formula (1).

$$accuracy = \frac{correct\ predictions}{all\ predictions} \quad (1)$$

5. System application building technology

5.1. Swift

Swift [10] is an object-oriented programming language developed by Apple and used to develop applications for Apple platforms, including iOS, macOS, watchOS, tvOS, and Linux. Swift is a new programming language, first released in 2014. Swift was designed to replace Objective-C, Apple's traditional programming language.

[9] https://cloud.google.com/tpu/docs/inception-v3-advanced

To develop iOS applications with Swift, developers need to use Xcode, an IDE developed by Apple. Xcode provides several tools and features to help developers build iOS applications with Swift, including:

- SwiftUI: SwiftUI is a framework that allows developers to create iOS user interfaces using Swift code.
- UIKit: UIKit is a framework that allows developers to create iOS user interfaces using Objective-C or Swift code.
- Cocoa Touch: Cocoa Touch is a framework that provides basic features and functionality for iOS applications, including networking, audio, location, and device access.

5.2. Core ML Tools

Core ML Tools [11] is a set of tools and libraries developed by Apple for developing, training, and deploying machine learning models on iOS, macOS, and other Apple devices. One of the important features of Core ML Tools is the ability to convert machine learning models from other popular formats such as TensorFlow, Keras, PyTorch, Scikit-learn, ONNX, and Caffe to the Core ML format. This allows deploying its models on Apple devices, improving performance and security. Core ML Tools supports a variety of machine learning models, including image classification models, time series prediction models, object detection models, and many other types of models. This makes it suitable for a wide range of applications and use cases.

5.3. Core ML

Core ML [12] provides a unified representation for all models. Your app uses Core ML APIs and user data to make predictions and to train or fine-tune models, all on a person’s device. Core ML optimizes on-device performance by leveraging the CPU, GPU, and Neural Engine while minimizing its memory footprint and power consumption. Running a model strictly on a person’s device removes any need for a network connection, which helps keep a person’s data private and your app responsive.

5.4. Core Data

Core Data [13] is a powerful Apple framework used for data management in iOS and macOS applications. Designed to provide an efficient way to interact with and store data, Core Data offers a straightforward interface for working with databases, supporting features such as object relationships, change tracking, and automatic data synchronization. It alleviates the burden on developers when handling data and creates a stable environment for application development.

5.5. WikipediaKit

WikipediaKit [14] is a Swift API client framework that is developed to help interact with Wikipedia through iOS or macOS applications. It was developed to support plant classification applications in querying information from Wikipedia. The framework provides tools and application programming interfaces (APIs) that allow the application to search, retrieve, and display information about plant species.

CHAPTER 3: PLANT CLASSIFICATION MODEL

In this chapter, I present an overview of the plant classification model and the dataset used in training the model. The process of training, using, and evaluating the model is then described.

1. Dataset

The dataset used to train the plant classification model is PlantNet-300K [15]. PlantNet-300K is a recently released dataset designed to represent real-life ambiguity. It was proposed as a strong candidate for the evaluation of set-valued classification and uncertainty estimation. The dataset was constructed from the PlantNet project [16], a large-scale collection of plant images collected from a large set of citizen scientists. PlantNet-300K has 306,146 images containing 1,081 different classes and is divided into 80% training set, 10% validation set, and 10% test set. Since some plant species are common while others are very rare, the dataset naturally has a high-class imbalance with 80% of classes only representing 11% of the total number of images. Additionally, the dataset inherently contains ambiguity and uncertainty, such as images collected under sub-optimal conditions, different species producing visually similar flowers that are challenging to distinguish, and images captured from different perspectives. The dataset is structured as shown in Figure 11, where each directory contains images associated with a class name specified in the "plantnet300K_species_names.json" file.

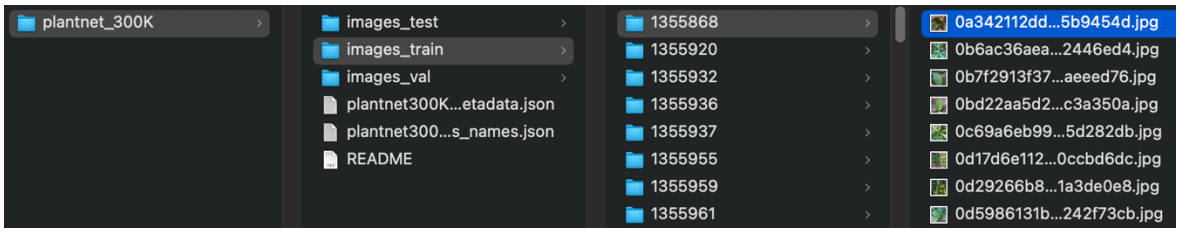

Figure 11. PlantNet-300K dataset structure

2. Dataset preprocessing

To be able to put data into training, I must go through data processing steps before I can use it. The steps are as follows:

Step 1 (dataset processing): replace the entire folder name with the class name provided in the file "plantnet300K_species_names.json" in which there are 2 or 3 folders with the same class name and are listed in Table 1.

No.	Class	Id
1	Lactuca_virosa	1355868, 1432783
2	Pelargonium_peltatum	1355959, 1394455

3	Pelargonium_zonale	1355978, 1435714
4	Tradescantia_zebrina	1356076, 1398178, 1422105
5	Asystasia_gangetica	1356279, 1405641
6	Nymphaea_nouchali	1356309, 1409296
7	Nephrolepis_cordifolia	1356420, 1389294
8	Alliaria_petiolata	1358150, 1392475
9	Lavandula_canariensis	1358755, 1560405
10	Pancratium_maritimum	1360978, 1421011
11	Cirsium_palustre	1361759, 1503981
12	Freesia_refracta	1362582, 1397556
13	Tradescantia_pallida	1362927, 1408774
14	Schefflera_actinophylla	1362954, 1399783, 1408657
15	Melilotus_officinalis	1363750, 1393792
16	Duchesnea_indica	1364000, 1411834
17	Guizotia_abyssinica	1364112, 1392658
18	Calendula_arvensis	1364161, 1390637, 1432786
19	Gynura_aurantiaca	1373530, 1409191
20	Schkuhria_pinnata	1374466, 1423286
21	Acalypha_hispida	1375797, 1408688
22	Mussaenda_philippica	1379556, 1420496
23	Vanilla_planifolia	1387687, 1404481, 1419807
24	Acacia_saligna	1389567, 1568914
25	Dryopteris_carthusiana	1391805, 1411827
26	Dryopteris_cristata	1391807, 1411829
27	Hebe_andersonii	1392689, 1646149
28	Kniphofia_uvaria	1393393, 1420863
29	Lactuca_alpina	1393414, 1513142
30	Lamium_maculatum	1393425, 1412418
31	Limnanthes_douglasii	1393537, 1419352
32	Sasa_palmata	1394714, 1420092
33	Pyracantha_coccinea	1394994, 1413391
34	Sedum_kamtschaticum	1396156, 1418545
35	Metasequoia_glyptostroboides	1396708, 1434712
36	Tradescantia_x_andersoniana	1396823, 1497630
37	Cymbalaria_muralis	1397364, 1711787
38	Fragaria_virginiana	1397613, 1420364

39	Sedum_palmeri	1398128, 1421021
40	Dryopteris_erythrosora	1398196, 1418655
41	Lupinus_nootkatensis	1398526, 1580985
42	Barbarea_vulgaris	1400100, 1414276
43	Peperomia_obtusifolia	1402921, 1409216
44	Alocasia_cucullata	1408037, 1442930
45	Alocasia_macrorrhizos	1408045, 1421107
46	Fragaria_x_ananassa	1408092, 1420365, 1667445
47	Acacia_auriculiformis	1408227, 1567995
48	Acacia_podalyriifolia	1408594, 1568799
49	Peperomia_argyreia	1409214, 1419923
50	Zamia_furfuracea	1409283, 1420896
51	Fittonia_albivenis	1409642, 1418146
52	Selenicereus_anthonyanus	1410024, 1488171
53	Pelargonium_x_hortorum	1419115, 1550692
54	Liriope_muscari	1419334, 1434011
55	Mazus_pumilus	1424003, 1434584
56	Melampodium_divaricatum	1424005, 1514627
57	Tagetes_lemmonii	1438868, 1522375

Table 1. Folders with the same class name

After renaming the folder to the class name, the remaining data has 1,019 different classes stored in the file "labels.txt" and 3 data for training, validation, and testing with the following number of images: 243,916, 31,118, 31,112 is shown in Figure 12.

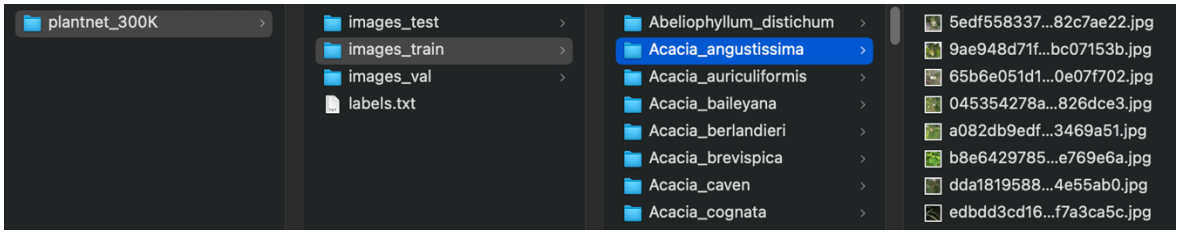

Figure 12. PlantNet-300K dataset structure after processing

Step 2 (image processing): read the entire image and resize the image in 3 training, validation, and test files to the size of 360 x 360. Image data is standardized by scaling pixel values to help bring them to a suitable format for inclusion in neural network shape. Figure 13 below shows an example image after being processed.

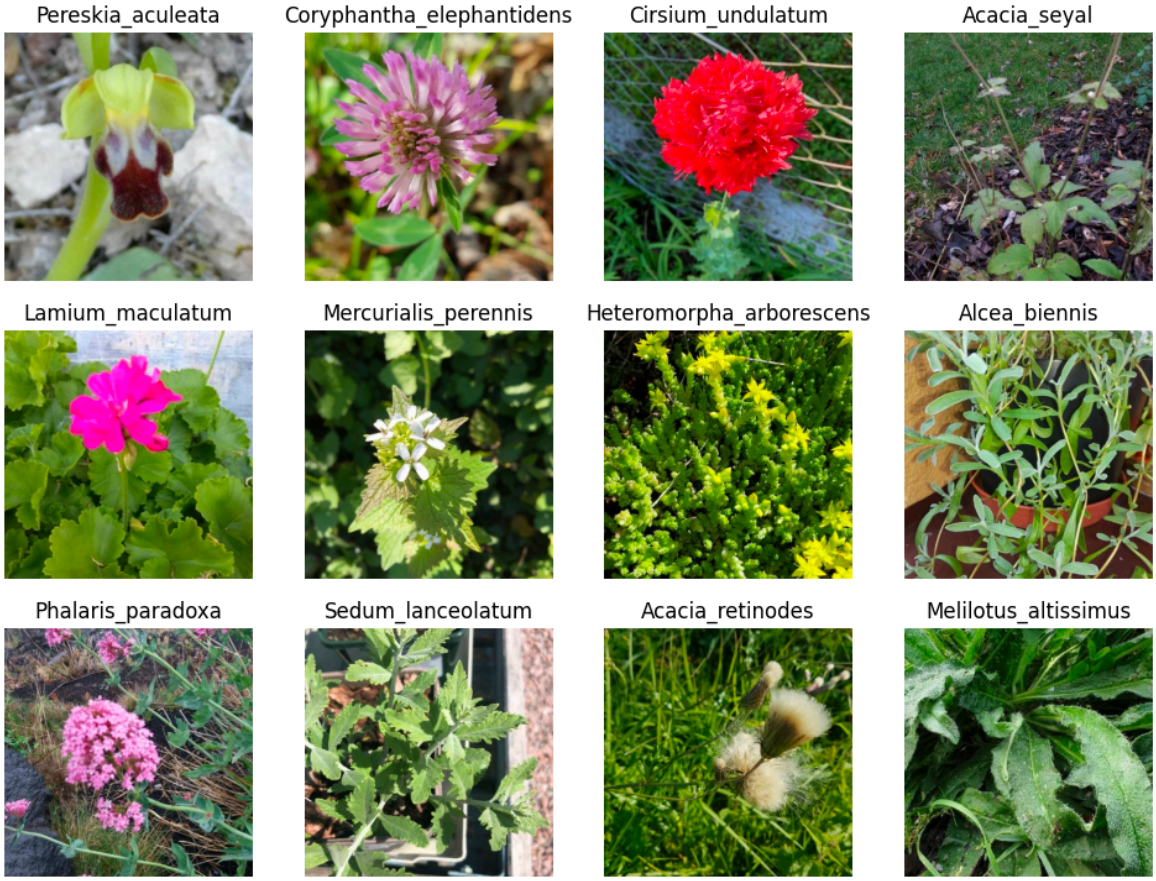

Figure 13. The example images after being processed

3. Plant classification overview

Transfer Learning is a technique that aims to extract knowledge from a source task to a target task rather than performing concurrent learning, as in Multi-task Learning [17]. This technique usually involves training a model on a large labeled dataset (source task) and then fine- tuning it on a smaller, specific dataset (target task), transferring part of the acquired knowledge from one base to another. Transfer Learning is often used to speed up the training and convergence of models with many parameters, using pre-trained weights on the larger dataset.

To build a plant classification model, I utilized the Inception V3 model as a pre-trained base on the ImageNet dataset to learn features from images. Subsequently, I employed GlobalAveragePooling2D, BatchNormalization, and dense layers to extract information from the learned features. Finally, a dense layer with a softmax activation function was used to make predictions for each plant class. The architecture of the plant classification model is illustrated in Figure 14.

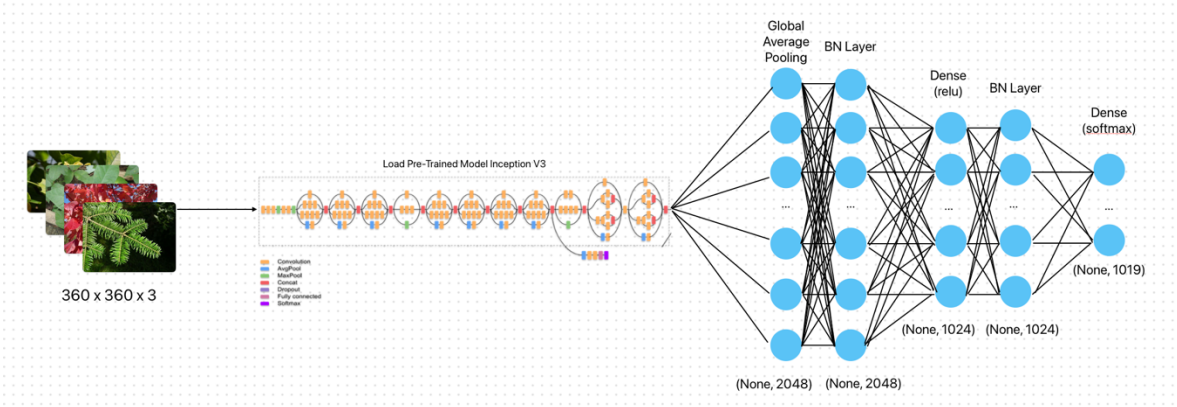

Figure 14. The plant classification model using pre-trained Inception V3

The input image has dimensions of 360x360 and is fed into the Inception V3 model through the model's input layer. Inception V3 extracts features from the input image through convolutional and pooling layers. After passing through the layers of Inception V3, the output is then passed through a GlobalAveragePooling2D layer. This layer converts each feature matrix into a single value by taking the average of all values in the matrix. BatchNormalization is applied after the GlobalAveragePooling2D layer to normalize the output and stabilize the learning process. Next is a dense layer with 1024 units and a ReLU activation function. This helps learn complex representations from the features extracted by Inception V3. Subsequently, a Batch Normalization layer is applied to normalize the output before it is fed into the final dense layer. The last layer is a dense layer with softmax activation to perform classification for the classes in the dataset, with a total of 1019 classes.

4. Model training

In this job, the models have been deployed using the TensorFlow library. The experiments utilized the feature extraction blocks of the Inception V3 models, specifically their convolutional parts. The original dense layers were removed, and the dense layer employed 1024 neurons were added. Adam was chosen as the optimizer and the remaining parameters of the model are shown in Table 2. The model was trained using the "fit()" method provided by Keras, supplying the necessary parameters and conducting the training process.

No.	Parameter	Value	Description
1	input_shape	(360, 360, 3)	Shape of the input images
2	batch_size	64	The parameter determining the number of samples to be processed in one training iteration

3	learning_rate	$1x10^{-4}$	The hyperparameter that governs the extent of adjustments to the model in response to the estimated error during each update of the model weights.
4	rescale	1./255	Rescaling factors for the pixel values of the images
5	dense_units	1024	Number of units in the dense layer
6	num_classes	1019	Number of classes in the classification task
7	epochs	5	The hyperparameter determines the number of times I pass all the data in the training dataset through the neural network

Table 2. Parameters of the plant classification model

To train the model, I used a personal computer due to the relatively large dataset. I created a file named "train_script_plantnet_300k.ipynb" to record the entire Python commands used for model training. All the output messages will be displayed below the execution line, allowing me to observe and monitor the training progress. To expedite the training process, I utilized a GPU. The hardware configuration of the machine includes 8GB of RAM, a 256 GB hard drive, and an Apple M1 CPU with integrated GPU.

5. Result evaluation

During the training process, Keras' algorithm saves models that achieve the best results based on the model's loss, as depicted in Figure 15. The best model in training is at epoch 3, with the accuracy is 0.7824 evaluated on the validation set and saved as "plant_model_epoch04.h5".

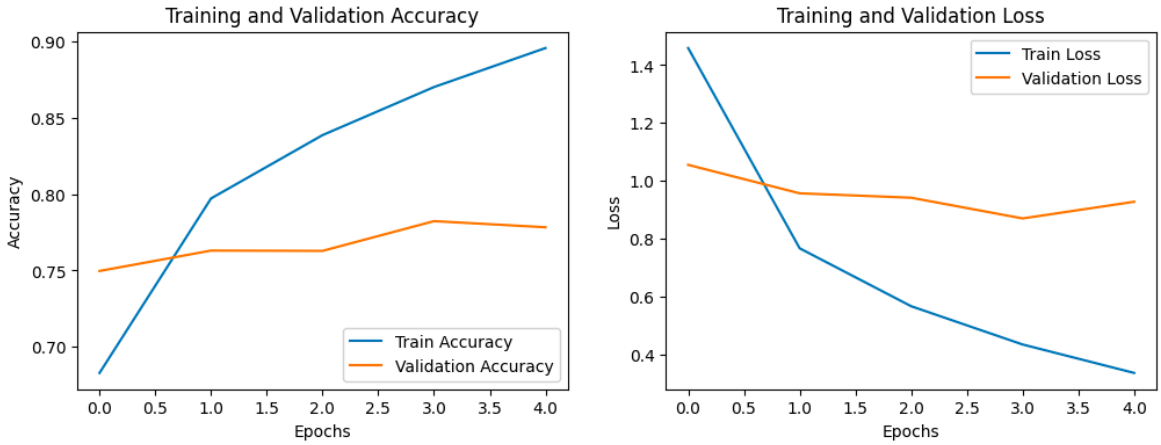

Figure 15. Plant classification model training process

After obtaining the best model, I proceeded to test it on the test set and received a result of 0.7798. The actual result illustrated in Figure 16 is 12 images randomly taken from the test data set.

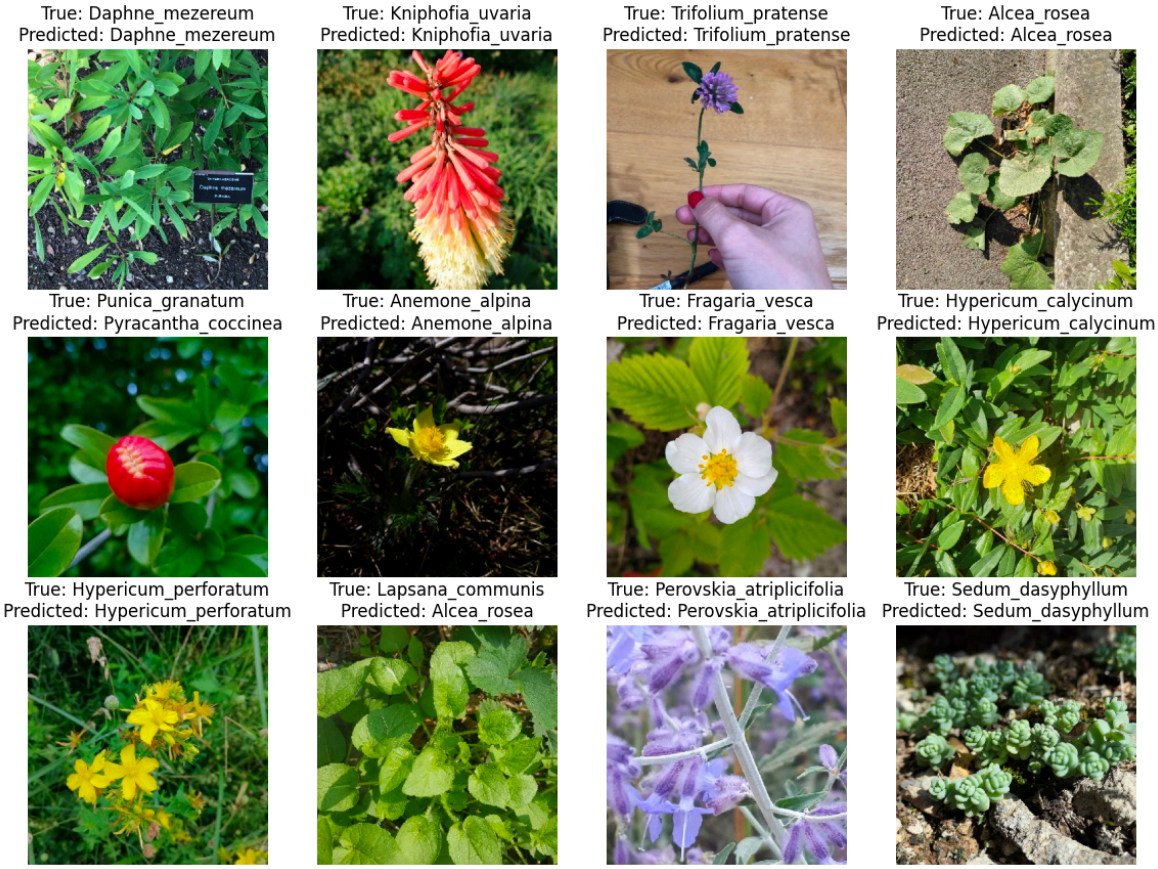

Figure 16. Results of the plant classification model

CHAPTER 4: APPLICATION SYSTEM DESCRIPTION

In this chapter, I describe the structures and operations of the system, which consist of two parts: converting a plant classification model from TensorFlow to Core ML and implementing the model in an iOS mobile platform application. After providing an overview of the model conversion, I will present details about the operation and functionalities within the application.

1. Converting a plant classification model from TensorFlow to Core ML

As outlined in the objectives, in addition to developing a plant classification model, I simultaneously developed an application for the model on the iOS mobile platform. This application enables users to easily utilize the model for plant classification. The Core ML model can run directly on the iOS platform without the need for a network connection, enhancing the mobility and performance of the application. To perform the conversion from TensorFlow to Core ML, I utilized Core ML Tools and provided the necessary data in Table 3.

No.	Parameter	Value	Description
1	model	plant_model_epoch04.h5	The plant classification model from the .h5 file was trained using TensorFlow
2	labels	labels.txt	The text file contains 1019 labels of the model
3	shape	1, 360, 360, 3	Shape of the input images. The number of images is 1, the image size is 360 x 360, and the number of color channels of the image is 3 (RGB)
4	scale	1/255	Rescaling factor for the pixel values of the images.
5	bias	0, 0, 0	This parameter defines the value value (offset) that is added to the image input before feeding it to the model. In this case, no value is added (all are 0)

Table 3. Parameters for converting from TensorFlow to Core ML

The TensorFlow model after conversion to a Core ML model is saved with the name "plant_model.mlpackage" and general information about the model is depicted in Figure 17.

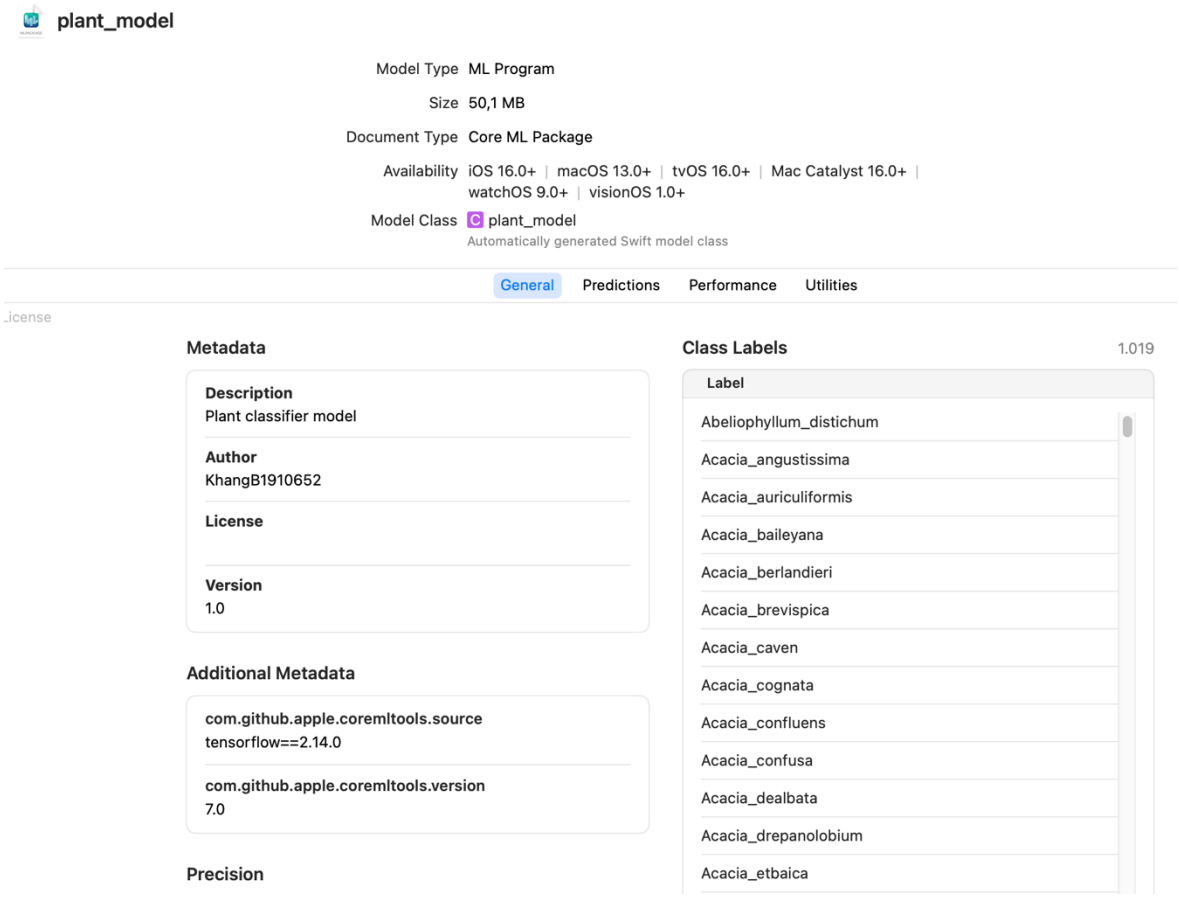

Figure 17. General information about the Core ML model for plant classification

2. Functional requirements

2.1. Upload image

2.1.1. Select a photo

Function name: Select a photo	**Function code**: ND001
Actor: User	**Priority level:** Normal
Description: Allows users to upload 1 to the library.	
Pre-conditions: - The user's device has granted permission to access the app's photo library. - Make sure the image file format is correct.	
Operation process: The user accesses the application, clicks the "Classify plant" button in the middle of the screen, and selects "Select a photo" from the bottom sheet window that appears. The user can then choose any photo from the library.	

Result: Display the image library for photo

Table 4. Describe the select a photo function

2.1.2. Take a photo

Function name: Take a photo	**Function code**: ND002
Actor: User	**Classification:** Medium
Description: Allows users to take any photo and the system will automatically upload it.	
Pre-conditions: The user's device has granted camera access permission to the application.	
Operation process: The user accesses the application, clicks the "Classify plant" button in the middle of the screen, and selects "Take a photo" from the bottom sheet window that appears. After that, the user can take any photo from the camera.	
Result: Display the camera interface.	

Table 5. Describe the take a photo function

2.1.3. Live preview

Function name: Live preview	**Function code**: ND003
Actor: User	**Classification:** Medium
Description: Allows users to view plant species prediction results directly from the camera.	
Pre-conditions: The user's device has granted camera access permission to the application.	
Operation process: The user accesses the application, clicks the "Classify plant" button in the middle of the screen, and selects "Live Preview" from the bottom sheet window that appears.	
Result: The live preview screen that appears will provide continuous prediction results based on images scanned from the camera.	

Table 6. Describe the live preview function

2.2. Classify plant

Function name: Classify plant	**Function code**: ND004
Actor: User	**Classification:** Medium
Description: Providing predictions after the user uploads an image.	
Pre-conditions: The user has successfully uploaded an image.	
Operation process: The user accesses the application, and clicks the "History List" button on the screen.	

Result: A plant species prediction result will be returned.

Table 7. Describe the classified plant function

2.3. View Wikipedia info

Function name: View Wikipedia info	**Function code**: ND005
Actor: User	**Classification:** Medium
Description: Allows users to view predicted plant species information through Wikipedia.	
Pre-conditions: - The user has performed the function select a photo or take a photo and the prediction results are returned. - The user's device is connected to the Internet.	
Operation process: The user clicks the "View Wikipedia Info" button in the middle of the prediction screen.	
Result: If there is an Internet connection, it will return information about predicted plant species from Wikipedia in English and Vietnamese.	

Table 8. Describe the view Wikipedia info function

2.4. View history list

Function name: View history list	**Function code**: ND006
Actor: User	**Classification:** Medium
Description: Allows users to review the entire prediction history.	
Pre-conditions: The user's device has granted memory access permission to the application.	
Operation process: The user accesses the application, and clicks the "History List" button on the screen.	
Result: The predicted plant species history page will be displayed.	

Table 9. Describe the view history list function

2.5. View my plant list

Function name: View my plant list	**Function code**: ND007
Actor: User	**Classification:** Medium
Description: Allows users to review all plant species that have been retrieved from Wikipedia.	
Preconditions: - Users have previously retrieved information about tree species from the view Wikipedia info function. - The user's device has granted memory access permission to the application.	

Processing: The user accesses the application, and clicks the "My plant" button on the screen.
Result: My plant page will be displayed.

Table 10. Describe the view my plant list function

3. Non-functional requirements

3.1. Enforcement requirements

The application must be compatible with various iOS devices, including both iPhones and iPads. It should perform well on devices with different configurations, ensuring fast response times. The response time for handling requests should not exceed 10 seconds.

3.2. Security requirements

- The user's images when using the application will be stored locally on the user's device to enhance security.
- The application works without affecting the operating system and other applications.
- The application does not contain viruses, malware, or junk files.

4. Use case diagram

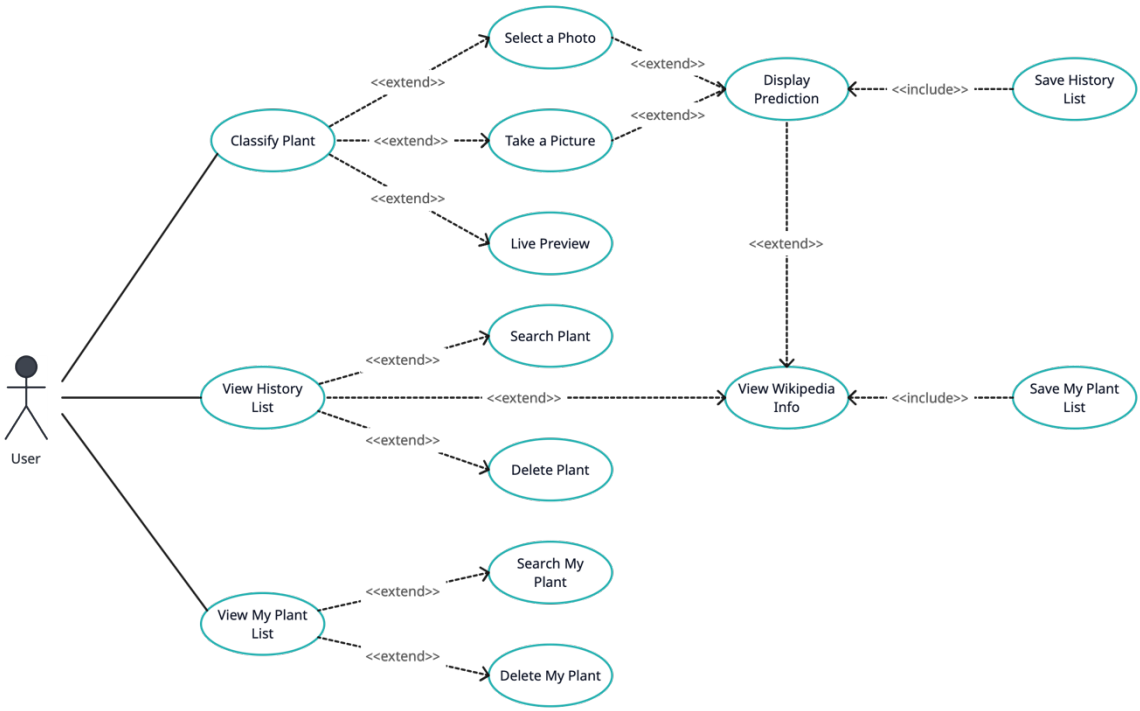

Figure 18. Use case diagram

5. Database (Core Data Model)

HistoryPlant stores the history list, and SavedPlant stores my plant list.

Attribute	Type	Description
id	UUID	Plant code.
name	String	Name of the predicted plant.
imageData	Binary Data	Photos of the predicted plant.
date	Date	Time when performing plant prediction.

Table 11. HistoryPlant entity

Attribute	Type	Description
id	UUID	Plant code.
nameEn	String	English names of the plant.
nameVi	String	Vietnamese names of the plant.
imageData	Binary Data	Photo of plant from Wikipedia.
descriptionEn	String	English description of plant from Wikipedia.
descriptionVi	String	Vietnamese description of plant from Wikipedia.
date	Date	Time saved.

Table 12. SavedPlant entity

6. Plant classification application on the iOS platform

The homepage features a tab view that displays several plants in my plant list and three buttons: "Classify Plant," "View History," and "View My Plant".

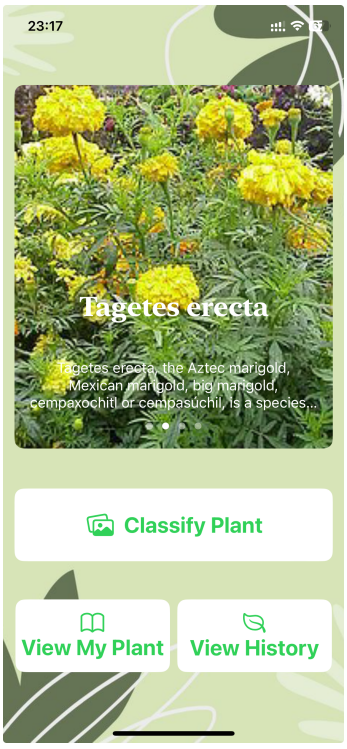

Figure 19. Home page screen

"Classify Plant" offers three options: selecting a photo from the library, taking a photo using the camera, and using live preview to see real-time predictions.

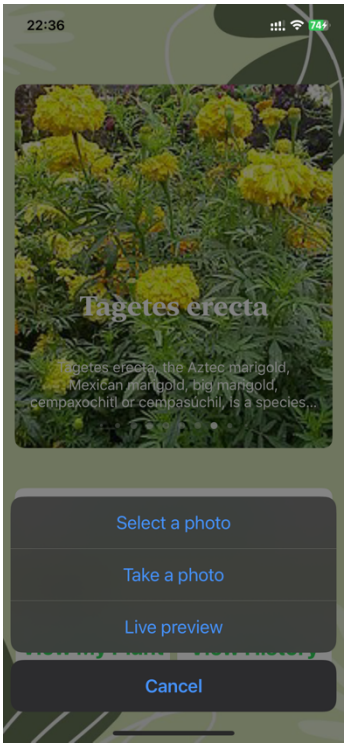

Figure 20. Classify Plant screen

"Select a photo", "Take a photo, "Live preview" is illustrated in Figure 21.

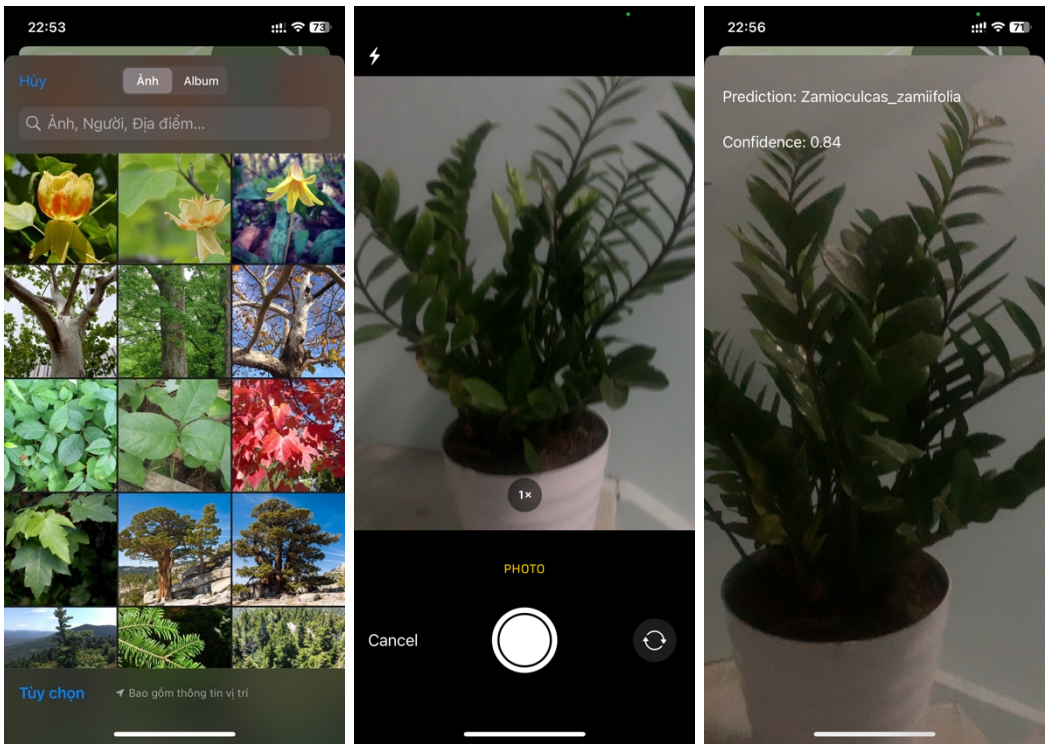

Figure 21. Select a photo screen, take a photo screen, live preview screen

The page displays predictions for the image, whether selected from the library or captured using the camera.

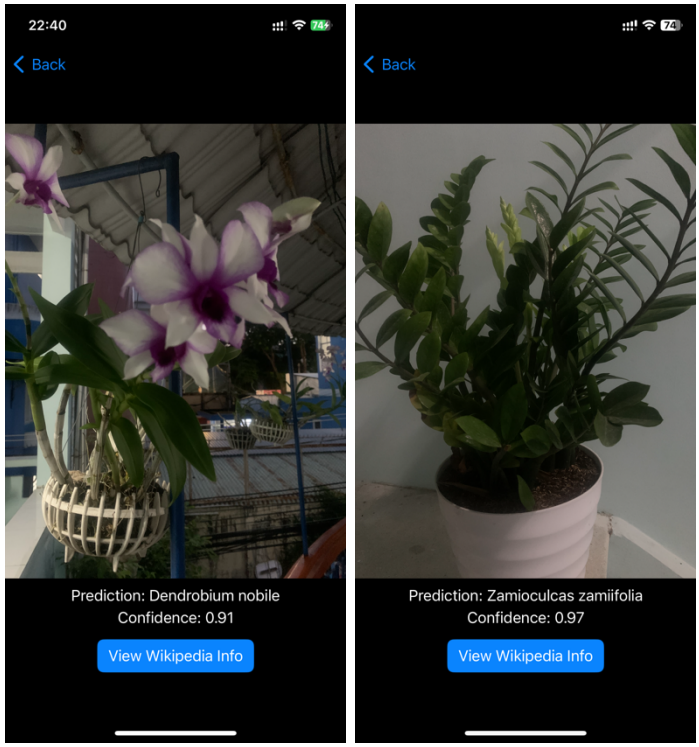

Figure 22. Prediction screen

The page shows information about the plant species from Wikipedia, comprising two sections in both English and Vietnamese.

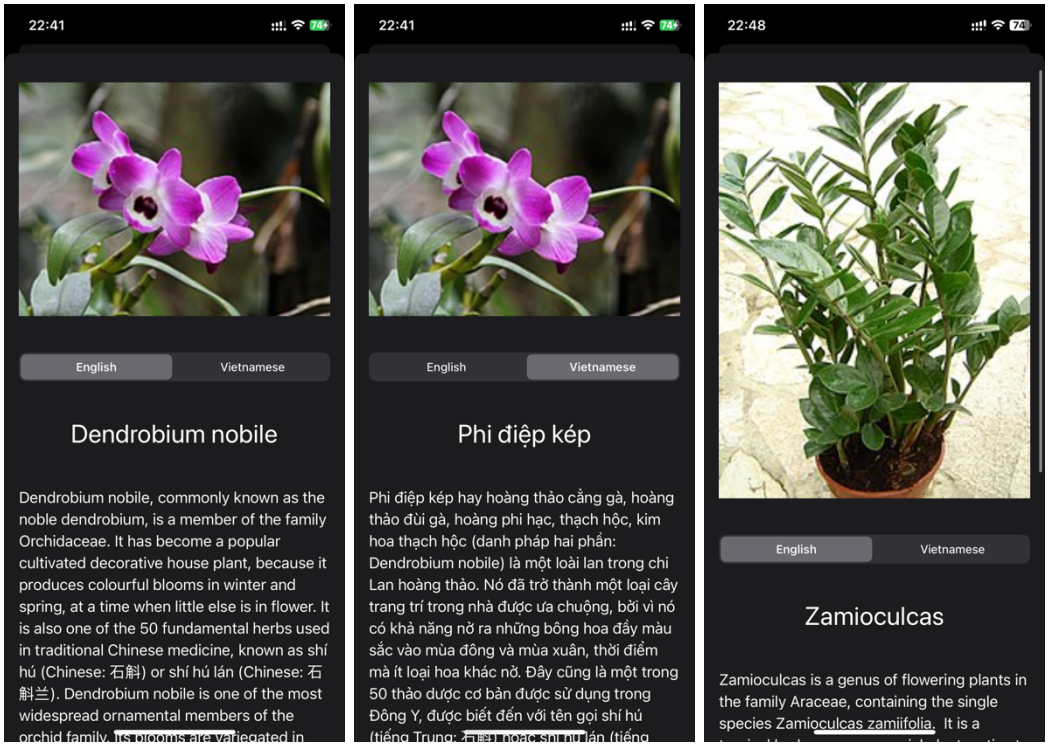

Figure 23. Wikipedia screen

The user history page allows searching by the name of a previously predicted plant species. Clicking "Show more" reveals information from Wikipedia if the device is connected to the internet or swiping left displays the option to delete.

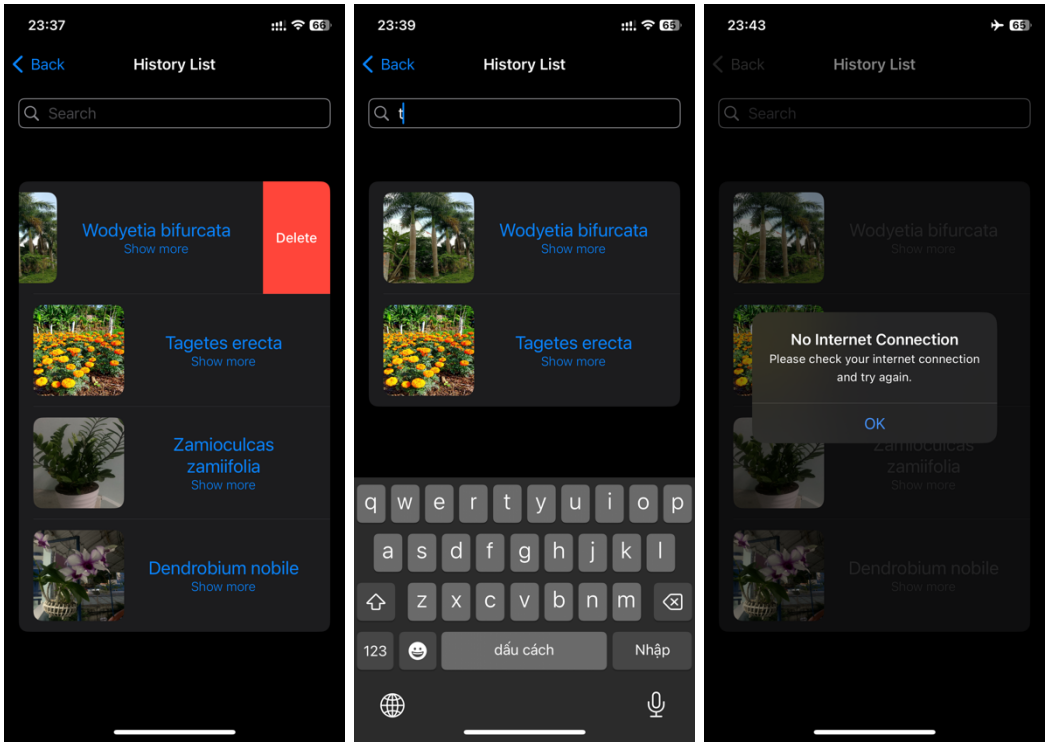

Figure 24. History screen

On my plant page, users have the ability to access Wikipedia information offline.

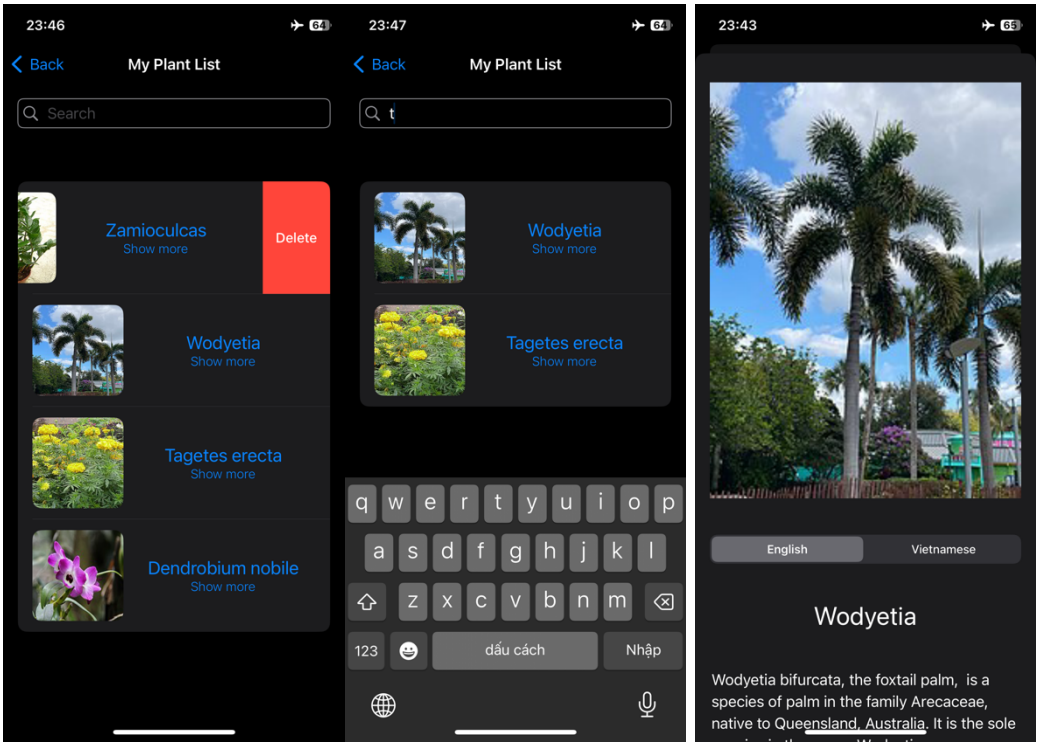

Figure 25. My plant screen

CHAPTER 5: TEST REVIEW

To check for potential errors that may occur during the application deployment and ensure that the system operates according to the defined goals, I focus on two types of testing: functional testing and non-functional testing. Among these, functional testing is given higher and more rigorous priority.

1. Objectives

- Ensure the final product meets user requirements.
- Serve the application testing team within the testing scope and schedule.
- Review for maintenance and future development.

2. Testing content

Testing content includes:

- Detailed testing plan
- Test process management

3. Testing plan details

In the test plan, priority will be given to testing the mandatory functions, which play a critical role throughout the entire application. This will be presented through test cases.

	macOS	**iOS**
Device	MacBook Air	iPhone Xs
CPU	Apple M1	Apple A12
RAM	8 GB	4 GB
Storage	256 GB	64 GB
OS version	macOS Sonoma 14.1.1	iOS 17.1.1
Software	Xcode	Developer mode

Table 13. Testing environment

4. Testing functions

Below is a table of functions that need to be tested on mobile platforms, along with the expected results for each test case.

No.	**Function**	**Description**	**Test Cases**	**Expected Results**
1	Upload image	User uploads images for plant classification.	Select a photo.	Display the image library for photo selection.
			Take a photo.	Display the camera interface.

			Live preview.	The live preview screen that appears will provide continuous prediction results based on images scanned from the camera.
2	Classify plant	Providing predictions after the user uploads an image.	Successful classification.	Display image and prediction results on the screen.
			Testing predictions for the same image at different upload times to check for consistency.	Display identical prediction results.
3	View Wikipedia info	Allows users to view predicted plant species information through Wikipedia.	The device is connected to the internet.	Display information retrieved from Wikipedia in both English and Vietnamese.
			The device is not connected to the internet.	Display error message for no internet connection.
4	View history list	Allows users to review the entire prediction history.	Search by name	Display one or multiple results found with the word in the name.
			Delete	Successful deletion with a refreshed list display.
			Shows more when the device	Display one or multiple results

			is not connected to the internet.	found with the word in the name.
			Shows more when the device is connected to the internet.	Display error message for no internet connection.
5	View my plant list	Allows users to review all plant species that have been retrieved from Wikipedia.	Search by name	Display one or multiple results found with the word in the name.
			Delete	Successful deletion with a refreshed list display.
			Shows more when the device is not connected to the internet.	Display information retrieved from Wikipedia in both English and Vietnamese.
			Shows more when the device device is connected to the internet.	Display information retrieved from Wikipedia in both English and Vietnamese.

Table 14. Testing functions

5. Testing cases

5.1. Upload image

Test case code: TC001					
Test case ID	**Scenario**	**Input Data**	**Expected Result**	**Actual Result**	**Evaluation**
TC001-1	Select a photo.	Open the device's photo library.	Display the image library for photo selection.	Display as expected result.	Success.

TC001-2	Take a photo.	Use the device's camera.	Display the camera interface.	Display as expected result.	Success.
TC001-3	Live preview.	Use the device's camera.	The live preview screen that appears will provide continuous prediction results based on images scanned from the camera.	Display as expected result.	Success.

Table 15. Image upload testing

5.2. Classify plant

Test case code: TC002					
Test case ID	**Scenario**	**Input Data**	**Expected Result**	**Actual Result**	**Evaluation**
TC002-1	Successful classification.	Photo upload successful.	Display image and prediction results on the screen.	Display as expected result.	Success.
TC002-2	Testing predictions for the same image at different upload times to check for consistency.	In the same image, the first upload and the second upload, which are 5 minutes apart, display different results.	Display identical prediction results.	Display as expected result.	Success.

Table 16. Classification plant testing

5.3. View Wikipedia info

Test case code: TC003					
Test case ID	**Scenario**	**Input Data**	**Expected Result**	**Actual Result**	**Evaluation**
TC003-1	The device is connected to the internet.	The name of the successfully predicted plant species.	Display information retrieved from Wikipedia in both English and Vietnamese.	Display as expected result.	Success.
TC003-2	The device is not connected to the internet.	The name of the successfully predicted plant species.	Display error message for no internet connection.	Display as expected result.	Success.

Table 17. Wikipedia information viewing testing

5.4. View history list

Test case code: TC004					
Test case ID	**Scenario**	**Input Data**	**Expected Result**	**Actual Result**	**Evaluation**
TC004-1	Search by name	Input from the keyboard.	Display one or multiple results found with the word in the name.	Display as expected result.	Success.
TC004-2	Delete	One species in the list.	Successful deletion with a refreshed list display.	Display as expected result.	Success.
TC004-3	Shows more when the device is not connected to the internet.	One species in the list.	Display one or multiple results found with the word in the name.	Display as expected result.	Success.

TC004-4	Shows more when the device device is connected to the internet.	One species in the list.	Display error message for no internet connection.	Display as expected result.	Success.

Table 18. History list testing

5.5. View my plant list

Test case code: TC005					
Test case ID	**Scenario**	**Input Data**	**Expected Result**	**Actual Result**	**Evaluation**
TC005-1	Search by name	Input from the keyboard.	Display one or multiple results found with the word in the name.	Display as expected result.	Success.
TC005-2	Delete	One species in the list.	Successful deletion with a refreshed list display.	Display as expected result.	Success.
TC005-3	Shows more when the device is not connected to the internet.	One species in the list.	Display information retrieved from Wikipedia in both English and Vietnamese.	Display as expected result.	Success.
TC005-4	Show more when the device device is connected to the internet.	One species in the list.	Display information retrieved from Wikipedia in both English and Vietnamese.	Display as expected result.	Success.

Table 19. My plant list testing

CHAPTER 6: CONCLUSION

In this chapter, I summarize the results I have achieved in the process of researching this topic. At the same time, I also make some suggestions to develop and improve this research in the future.

1. Result achieved

After the implementation of the project, the program has completed the following results:

- Machine learning theory.
- Research on relevant papers related to image classification, especially plant classification.
- Inception V3 model theory.
- PlantNet-300K training dataset.
- Methods for building, training, and evaluating machine learning models.
- iOS application development technology.
- Methods for building applications on the iOS platform.
- Application testing methods.

During the project implementation, I integrated a plant classification model using machine learning technology into a practical application by developing it on the iOS mobile platform. The application can be utilized for educational purposes and provides quick assistance in classifying various plant species using a handheld mobile device.

2. Development

During the research, I derived several recommendations:

- Improving the accuracy of the model can be achieved by utilizing more advanced machine learning models such as Inception V4, Inception Resnet V2. However, fine-tuning numerous parameters is necessary to prevent the model from overfitting due to complex datasets.
- To enhance the diversity of plant species, incorporating the PlantCLEF dataset with over 10,000 species can be considered.
- Exploring alternative technologies like TensorFlow Lite, React Native, or Flutter could be beneficial for building the application, providing compatibility across various operating systems.
- Considering the use of a paid version of Google Colab for improved model training.

To apply the research results to real-life scenarios, the software needs to undergo more in-depth testing and be deployed to the app store on mobile platforms, especially iOS, to ensure easy access for everyone.

REFERENCES

[1] Kaya, A., Keceli, A. S., Catal, C., Yalic, H. Y., Temucin, H., & Tekinerdogan, B, "Analysis of transfer learning for deep neural network based plant classification models," in *Computers and electronics in agriculture, 158, 20-29*, 2019.

[2] Ahmad, A., Saraswat, D., Aggarwal, V., Etienne, A., & Hancock, B, "Performance of deep learning models for classifying and detecting common weeds in corn and soybean production systems," in *Computers and Electronics in Agriculture, 184, 106081*, 2021.

[3] Pratondo, Agus, Elfahmi Elfahmi, and Astri Novianty, "Classification of Curcuma longa and Curcuma zanthorrhiza using transfer learning," in *PeerJ Computer Science, 8, e1168*, 2022.

[4] Chen, D., Lu, Y., Li, Z., & Young, S, "Performance evaluation of deep transfer learning on multi-class identification of common weed species in cotton production systems," in *Computers and Electronics in Agriculture, 198, 107091*, 2022.

[5] Castro, P. H. N., Fortuna, G. C., de Queiroz, R. A. B., & Moreira, G. J. P, "Regularization Through Simultaneous Learning: A Case Study for Hop Classification," in *arXiv preprint arXiv:2305.13447*, 2023.

[6] Wang, Sun-Chong, and Sun-Chong Wang, "Artificial neural network," in *Interdisciplinary computing in java programming, 81-100*, 2003.

[7] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9)*, 2015.

[8] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826)*, 2016.

[9] "Accuracy," [Online]. Available: https://www.kdnuggets.com/2018/06/right-metric-evaluating-machine-learning-models-2.html.

[10] "Swift," [Online]. Available: https://developer.apple.com/swift/.

[11] "Core ML Tools," [Online]. Available: https://apple.github.io/coremltools/docs-guides/source/overview-coremltools.html.

[12] "Core ML," [Online]. Available: https://developer.apple.com/documentation/coreml.

[13] "Core Data," [Online]. Available: https://developer.apple.com/documentation/coredata/.

[14] "WikipediaKit," [Online]. Available: https://github.com/Raureif/WikipediaKit.

[15] Garcin, C., Joly, A., Bonnet, P., Lombardo, J. C., Affouard, A., Chouet, M., ... & Salmon, J, "Pl@ ntNet-300K: a plant image dataset with high label ambiguity and a long-tailed distribution," in *NeurIPS 2021-35th Conference on Neural Information Processing Systems*, 2021, December.

[16] Affouard, A., Goëau, H., Bonnet, P., Lombardo, J. C., & Joly, A, "Pl@ ntnet app in the era of deep learning," in *ICLR: International Conference on Learning Representations*, 2017, April.

[17] PAN, Sinno Jialin; YANG, Qiang, "A survey on transfer learning," in *IEEE Transactions on knowledge and data engineering, 22(10), 1345-1359*, 2009.