



University of Fort Hare
Together in Excellence

Development of a Mobile Application to Auto-Detect Medicinal Plants Using an Artificial Intelligence Approach



University of Fort Hare

A dissertation

**submitted in fulfilment of the requirement for the degree of
Master of Science in
Computer Science**

**Faculty of Science and Agriculture
University of Fort Hare Alice
South Africa**

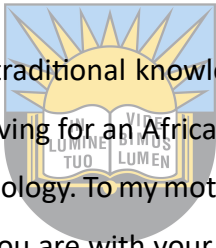
Supervisor: Mr. S. Ngwenya

Co-Supervisor: Mr. P. Sibandze

DECLARATION

At this moment, I declare that the work included in the thesis, Development of a Mobile Application to Auto-Detect Medicinal Plants Using an Artificial Intelligence Approach, is original work fulfilling a Master of Science in Computer Science degree. The research was done at the University of Fort Hare under the guidance of Mr. Sikhumbuzo Ngwenya and Mr. Phila Sibandze. I have yet to present the work covered in this thesis in any other institution.

DEDICATION



This thesis is dedicated to the African traditional knowledge holders, giving hope to traditional concepts that were part of the way of living for an African to be saved and passed on to the next generations with the assistance of technology. To my mother, I appreciate all the support you gave me. The constant question of how far you are with your studies and why you are so quiet about your schoolwork pushed me to realize that this body of work is not just about me; it's also a representative of where I come from, Enkosi Mamsukwini. To my wife, I am fortunate to have you in my life. Your push, support, and encouragement throughout this research are impressive. Whenever the research load was overwhelming, you picked me up and encouraged me to get up and try again. Enkosi mntu wam, ndiyakuthanda.

To my kids, never stop learning. I want you always to know where you come from and never be ashamed of the African in you. Your culture and customs are our wealth. Always be an explorer and set new boundaries for the difficult questions our people face. Utata uyanithanda, niziphathe kakuhle ngalolonke ixesha. To my ancestors, this research came as an answer to a prayer I sent. When I prayed for my career in computer science to advance, I was answered by a dream from my ancestors telling me to do my master's degree; I was also given the topic of ngamanyange entaba. Mr. Vuyani Faku, also known as uTata uNdabazithethwa, is a traditional healer who

trained me to become a traditional healer. Thank you for all the wisdom and knowledge you passed on to me. The constant reminder you gave me saying, "Zanomhlola, ndifuna wenze iiMasters," is finally happening. Ndiyabulela ndima, khoza, mvu emnyama, sonqalelo, sondise. Ndilithole leduna elizalwa nguMsondezi, ujola, uqengeba, ungwanya, uthole lomthwakazi, usomarhwarhwa, uZanomhlola elethongo igama. Ndiyacamagusha ndinqula umsebenzi wezandla zethongo eliphuma kumathongo okunyanga ubonakale, Camagu.



University of Fort Hare
Together in Excellence

ACKNOWLEDGEMENTS

I want to acknowledge my supervisors warmly, Mr. Sikhumbuzo Ngwenya and Mr. Phila Sibandze. Thank you for all the guidance that helped shape this body of work into what it is today. Thank you, Prof. K. Sibanda, for allowing me to explore this topic within the constraints of computer science. To Brighton Sekulane, thank you for your guidance in developing the application.



University of Fort Hare
Together in Excellence

Acronyms

AI – Artificial Intelligence

ANN - Artificial Neural Networks

CNN - Convolutional Neural Network

DB - Database

IOS – iPhone Operating system

KNN - K Nearest Neighbor

MATLAB – Matrix Laboratory

OS – Operating System

ResNets - Residual Networks

SQL – Structured Query Language



University of Fort Hare
Together in Excellence

ABSTRACT

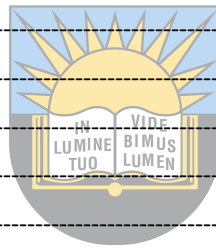
A mobile application assisting traditional healers in identifying medicinal plants is presented. Python programming language was used to build the mobile application that would auto-detect five medicinal plants (Mphepo, Mhlonyane, Hlaka-Hlakana, Mthoba & Cima mlilo) on a Spyder IDE version 5.4.0. The performances of PyTorch and TensorFlow lite Python libraries were tested using the Convolutional Neural Network Deep learning method. Five medicinal plants were part of the data set to be collected using a Samsung cellphone to capture images at different angles. Data sets were collected from their natural habitats at angles 30%, 60%, and 90% per plant species. The best-performing model was selected and presented to the mobile application. Offline capability was programmed into the mobile application, making it possible to operate the mobile application in areas with no network. A machine-learning database used for plant identification was created and stored on the device. On-device storage gave the mobile application the ability to perform plant identifications offline. The offline capability of mobile applications is crucial as the traditional healing community often picks up traditional medicine in jungles where telecommunications networks can be a challenge. Mobile applications achieved an identification result of 98% accuracy, with the 90% angle being the most successful. The most preferred time of the day was the afternoon.

Keywords

Artificial Intelligence in Botany, Medicinal Plant Identification, Mobile Application Development, Automated Plant Detection, AI-Driven Plant Recognition

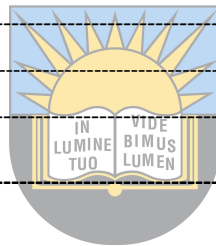
Table of Contents

DECLARATION	I
DEDICATION	I
ACKNOWLEDGEMENTS	III
ABSTRACT	V
LIST OF TABLES	X
LIST OF FIGURES	XI
CHAPTER ONE	1
1.1 INTRODUCTION	1
1.2 BACKGROUND	2
1.3 PROBLEM STATEMENT	4
1.4 AIM	5
1.5 OBJECTIVES	5
1.6 ETHICAL CONSIDERATIONS	6
1.7 EXPECTED OUTCOMES	6
1.8 CHAPTER OUTLINE	6
1.9 CONCLUSION	7
CHAPTER TWO	8
2. LITERATURE REVIEW	8
2.1 INTRODUCTION	8
2.2 AI TECHNIQUES IN PLANT IDENTIFICATION	9
2.3 IMAGE DATASETS AND PREPROCESSING	9
2.4 FEATURE EXTRACTION AND REPRESENTATION	10
2.5 MEDICINAL PLANT DATABASES	10
2.6 MOBILE APPLICATIONS FOR PLANT IDENTIFICATION	11
2.7 PERFORMANCE EVALUATION AND VALIDATION	11
2.8 DEVELOPMENT OF A MOBILE APPLICATION	12
2.8.1 Identification of medicinal plants	12
2.8.2 Software development	13
2.8.3 Waterfall methodology	14
2.8.4 Agile	16



University of Fort Hare
Together in Excellence

2.8.5	<i>Rapid application</i>	17
2.8.6	<i>Mobile application operating system</i>	19
2.8.6.1	Android	19
2.9	MACHINE LEARNING	22
2.9.1	<i>Nearest Neighbour</i>	23
2.9.2	<i>Naive Bayes</i>	24
2.9.2.1	Gaussian Naïve Bayes classifier	25
2.9.3	<i>Decision Trees</i>	26
2.9.3.1	Entropy	27
2.9.3.2	Information Gain	27
2.9.3.3	Gini Index	28
2.9.4	<i>Support Vector Machines (SVM)</i>	29
2.9.4.1	Linear	29
2.9.4.2	Gaussian Radial Basis Function	30
2.9.4.3	Polynomial Kernel	30
2.9.5	<i>Neural Networks</i>	31
2.9.5.1	Deep learning	32
2.10	SUMMARY	37
CHAPTER THREE		39
3.	RESEARCH METHODOLOGY AND DESIGN	39
3.1	INTRODUCTION	39
3.2	RESEARCH METHODOLOGY	39
3.2.1	<i>Mobile Application</i>	40
3.2.1.1	System Requirements	40
3.2.1.2	System specifications	41
3.2.1.3	System design	42
3.2.1.4	System development	43
3.2.1.5	System testing	44
3.2.1.6	System release	44
3.2.1.7	Machine learning	44
3.2.1.8	Data collection	45
3.2.1.9	Data processing	47
3.2.1.10	Models training	48
3.2.1.11	Python Development Libraries	48
3.2.1.12	Model storage	49
3.2.1.13	On-device	49
3.2.1.14	Cloud-based	49



University of Fort Hare
Together in Excellence

3.3	TESTING MOBILE APPLICATION	50
3.4	RESULTS	50
3.5	SUMMARY	50
CHAPTER FOUR		52
4.	SYSTEM DESIGN AND IMPLEMENTATION	52
4.1	INTRODUCTION	52
4.2	DEVELOPMENT ENVIRONMENT & SYSTEM ARCHITECTURE	52
4.2.1	<i>Mobile Application Development</i>	53
4.2.2	<i>Model Development Using Deep Learning</i>	53
4.2.3	<i>Model Deployment</i>	53
4.3	IMPLEMENTATION PROCESS	55
4.4	CODING STANDARDS AND GUIDELINES	55
4.5	USER INTERFACE DESIGN	56
4.5.1	<i>User registration and login</i>	56
4.5.1.1	Plant identifier	63
4.5.1.2	Assignments	67
4.5.2	<i>Model training & Model upload (Machine Learning)</i>	70
4.5.2.1	Deep learning	70
4.5.2.2	Image Resize	71
4.5.2.3	Tensorflow lite	77
4.5.2.4	Pytorch	77
4.5.3	<i>Model Performance, Evaluation and Validation</i>	79
4.5.4	<i>Correlation Analysis</i>	84
4.5.5	<i>Poor Performance Analysis</i>	84
4.6	SUMMARY	87
CHAPTER FIVE		88
5:	RESULTS AND DISCUSSIONS	88
5.1	INTRODUCTION	88
5.2	DESIGNING OF THE MEDICINAL PLANT DATABASE	88
5.3	DESIGNING THE MOBILE APPLICATION INTERFACE	89
5.3.1	<i>RAM performance</i>	91
5.3.2	<i>CPU performance</i>	93
5.3.3	<i>Application Security</i>	94
	<i>Figure 5-6: Offline login</i>	95

5.4 Developing an AI algorithm based on image recognition techniques to extract relevant features from the captured plant images.	95
5.4.1 Image angles and weather conditions test results.....	96
5.5 Evaluation of CNN	102
5.6 SUMMARY	103
CHAPTER SIX.....	104
6. CONCLUSION AND RECOMMENDATIONS.....	104
6.1 INTRODUCTION.....	104
6.2 RESEARCH OBJECTIVES ANALYSIS	104
6.3 RECOMMENDATIONS AND CONCLUSION	105
REFERENCES:	107
APPENDIX A	117



University of Fort Hare
Together in Excellence

LIST OF TABLES

<i>Table 2-1: Related works</i>	36
<i>Table 3-1: Data set organization</i>	45



University of Fort Hare
Together in Excellence

LIST OF FIGURES

Figure 2-1: Waterfall methodology (Source: Gechman, 2019)	14
Figure 2-2: Agile methodology.....	16
Figure 2-3: Rapid application	18
Figure 2-4: Android software stack (Source: Mesia Dhas and Shiny Angel, 2022)	20
Figure 2-5: Pl@ntNet architecture (Source: Affouard et al., 2017)	22
Figure 3-1: Design Science methodology processes.....	40
Figure 3-2: Mobile application system design	42



University of Fort Hare
Together in Excellence

CHAPTER ONE

1.1 Introduction

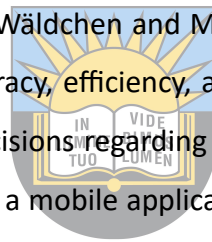
Identification and utilization of medicinal plants have been practiced for centuries across various cultures and regions (Jamshidi-Kia, Lorigooini, and Amini-Khoei, 2017). These plants play a crucial role in traditional medicine and are a valuable resource for healthcare practices worldwide (Jamshidi-Kia, Lorigooini, and Amini-Khoei, 2017). Accuracy in identifying medicinal plant species has been a challenging task requiring extensive botany knowledge and expertise. Traditional methods of plant identification rely on manual observation, comparing plant characteristics with botanical references, and consulting experts, which can be time-consuming, labor-intensive, and prone to errors (Liu et al., 2022). With rapid advancements in technology, particularly in artificial intelligence (AI), there is an opportunity to develop innovative solutions to address the challenges associated with medicinal plant identification (Shivaprakash et al., 2022).

The use of AI in plant identification has gained significant attention in recent years. AI algorithms, particularly those based on deep learning and convolutional neural networks (CNNs), have demonstrated remarkable capabilities in recognizing and classifying objects within images (Dhillon and Verma, 2020). These algorithms can be trained on vast datasets of plant images, enabling them to learn and identify specific plant species based on their visual characteristics (Barbedo, 2019).

While AI-based plant identification applications have been developed for general plant species, the domain of medicinal plants presents unique challenges (Khan *et al.*, 2021). Medicinal plants often exhibit specific characteristics and morphological features that distinguish them from other plant species (Salma, Kundu, and Mandal, 2017). Furthermore, accurately identifying medicinal plants is crucial for ensuring the safety and efficacy of herbal remedies and medicinal treatments (Bhardwaj, Verma, and Gupta, 2018).

Mobile applications equipped with AI algorithms have in recent years revolutionized the way plant identification is conducted, providing users with instant and reliable information on plant species, properties, and their potential medical applications (Chalmers, MacKenzie, and Carter, 2021). By leveraging AI technologies, such as image recognition and machine learning algorithms, a mobile application can be a valuable tool for botanists, herbalists, healthcare practitioners, and even the public. Therefore, there is a need to develop a specialized mobile application that focuses specifically on auto-detecting and identifying medicinal plants using an AI approach.

The research aimed to develop a mobile application that leverages AI algorithms to automatically detect and identify medicinal plants (Raji *et al.*, 2020). By combining image recognition techniques, machine learning algorithms, and a comprehensive database of medicinal plant species, the application anticipates providing users with a reliable and user-friendly tool for plant identification (Wäldchen and Mäder, 2018). The development of such an application can enhance the accuracy, efficiency, and accessibility of plant identification, enabling users to make informed decisions regarding using medicinal plants (Wäldchen and Mäder, 2018). For the impact of such a mobile application to be seen, a look at the research background is crucial.



University of Fort Hare
Together in Excellence

1.2 Background

African culture has always been closely related to nature, predominantly plants (Van Wyk and Prinsloo, 2018). There is vast knowledge about medicinal plants, which has made some African communities self-sufficient in most health-related issues (Ulasi *et al.*, 2020). There has always been an indigenous system of medicinal plants passed on from generation to generation (Demie, Negash, and Awas, 2018). Knowledge of medicinal plants needs to be adequately documented in African indigenous communities; it is, however, passed on orally from generation to generation (Buwa-Komoren *et al.*, 2019). Identifying the plant itself is crucial to knowing a medicinal plant's uses (Pengelly, 2020). Indigenous methods of identifying medicinal plants have become a specialty (Ozioma and Chinwe, 2019). As we exist in this digital era, only some people of African origin interact with nature, resulting in the loss of

indigenous knowledge and identification of medicinal plants (Howes *et al.*, 2020). Plant identification specialty mustn't be lost to African people as some healing methods centered on these plants will be lost. South African strategy for plant conservation has lost traditional knowledge and preservation, a critical number 13 strategic element of plant conservation in South Africa (Raimondo, 2018).

The following methods form part of the methods used by our ancestors to identify plants with medicinal benefits:

- Phenological structure of the plant (leaf shape, leaf color, stem shape, stem width, stem length, color of the flower produced by the plant).
- The smell of the plant.
- The taste of the plant (leaf).
- Location of the plant (does it grow near water, on the open field, or mountain).

The most effective criterion for identifying a plant is the first one listed above. The physical appearance of a plant is critical in plant identification. Other identification methods listed are used to confirm the identity of a plant. Even though the visual identification method is used in many cultures, the process differs (Sanmart'in, DeAraujo and Vasanthakumar, 2018). In Xhosa and most Nguni cultures, a traditional healer sometimes sends iThwasa to fetch traditional medicine from the forest or the medicine themselves (Molete, 2020). iThwasa is a person going through initiation to become a traditionally qualified iGqirha (traditional healer). Current methods of plant identification within traditional healing communities are working well but have come across a few challenges, such as:

- a) Knowledge of identifying medicinal plants is mostly with elders. When the custodian of the knowledge dies without passing it to the next generation, plant identification methods are lost.
- b) Only a few younger generation members are interested in the old ways of plant identification.

This research looked to go beyond just identifying medicinal plants; the intention was to create a mobile application that would be part of the routine of the community of amaGqirha. An application that would make it easy for iGqirha to send a task to ithwasa to fetch medicinal

plants and assist iGqirha in identifying medicinal plants. The mobile application was envisaged to assist amathwasa and amagqirha in identifying medicinal plants and other application users.

1.3 Problem Statement

Identifying medicinal plants is crucial in healthcare and traditional medicine (Süntar, 2020). However, accurately identifying medicinal plant species can be challenging, time-consuming, and prone to errors (Bock *et al.*, 2020). Traditional methods of plant identification rely on manual observation, consultation with experts, and reference to botanical literature, which can be inaccessible and overwhelming for non-experts (Strasser and Haklay, 2018). Additionally, misidentifying medicinal plants can lead to potential health risks and ineffective use of herbal remedies (Jamshidi-Kia, Lorigooini, and Amini-Khoei, 2017).

There is increased demand for medicinal plants coupled with the limited availability of trained botanists and herbalists, which hinders the accessibility and efficiency of plant identification (Demie, Negash and Awas, 2018). Innovative solutions that streamline and simplify identifying medicinal plants can enable users to make informed decisions regarding their utilization. Artificial intelligence (AI) and image recognition technologies offer a promising approach to address the challenges associated with medicinal plant identification (Abdollahi, 2022a). By harnessing the power of AI algorithms, a mobile application can analyze plant images, extract relevant features, and compare them with a comprehensive database of known medicinal plant species (Chattu, 2021). This approach has the potential to provide users with instant and accurate plant identification, supporting their efforts in herbal medicine, research, and conservation.

There is a lack of a user-friendly and efficient tool for auto-detecting and identifying medicinal plants using an AI approach (Adade Williams, Sikutshwa, and Shackleton, 2020). The existing methods for plant identification are often time-consuming, require specialized knowledge, and are limited in accessibility (Wäldchen *et al.*, 2018). Consequently, there is a need to develop a mobile application that leverages AI algorithms and image recognition techniques

to automate the process of medicinal plant identification, providing users with a reliable and user-friendly tool to identify and utilize medicinal plants safely and effectively.

Addressing this problem can empower botanists, herbalists, healthcare practitioners, and the public with an accessible and efficient solution for plant identification. It would enhance the accuracy and efficiency of medicinal plant identification, reduce the risk of misidentification, and facilitate the appropriate use of herbal remedies. Additionally, it would promote the conservation of medicinal plants by encouraging their identification and understanding, contributing to sustainable practices and the preservation of traditional knowledge (Adade Williams, Sikutshwa, and Shackleton, 2020).

1.4 Aim

This research aimed to develop an AI-powered mobile application that would auto-detect and classify medicinal plants.



1.5 Objectives

To respond to the aim, the following objectives were set:

University of Fort Hare
Together in Excellence

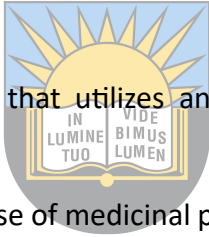
- a) Creating a comprehensive database of medicinal plant species, including information on their properties, traditional uses, and potential medical applications.
- b) Designing an intuitive and user-friendly mobile application interface for capturing plant images and submitting them for identification.
- c) Developing an AI algorithm based on image recognition techniques to extract relevant features from the captured plant images.
- d) Evaluate the performance of CNN developed in TensorFlow and PyTorch libraries on the in classification of different medicinal plant species.
- e) Evaluating performance and accuracy of the developed mobile application through rigour testing, comparing its results with manual identification and validation by experts. Rigour testing achieved by testing all the data set collection methods utilized.

1.6 Ethical Considerations

The research proposal was submitted to the University of Fort Hare research ethics committee with reference number/ registration number: NGW021SSIM01. Permissions to conduct the study were obtained from the University of Fort Hare. Participants were informed about the study, aims, and objectives. Participation was voluntary. Participants were asked to provide their consent before the study resumed. Mr. Onke Simandla ensured privacy and confidentiality. This research adhered to ethical guidelines during data collection, testing, and validation. Data is stored in a database accessible via the mobile application, which is user profile and password protected.

1.7 Expected Outcomes

The expected outcomes of this research include:

- 
- a) Developed a mobile application that utilizes an AI approach for auto-detecting and classifying medicinal plants.
 - b) Creating a comprehensive database of medicinal plant species which provides users with valuable information on plant properties, traditional uses, and potential medical applications.
 - c) Improved accuracy, efficiency, and accessibility of medicinal plant identification, enabling users to make informed decisions regarding medicinal plant utilization.
 - d) Validated developed mobile application through rigorous testing and evaluation

1.8 Chapter Outline

The research is structured as shown below:

Chapter 2: Literature Review:

The chapter covers a thorough literature review from previous research on plant identification, mobile application development, operating systems, intrusion detection mechanisms, and machine learning.

Chapter 3: Research Methodology:

This chapter explains the research methodology used to train and test the convolutional neural network model under different Python environments, which were TensorFlow and PyTorch and how these perform when deployed on a mobile application.

Chapter 4: Research Implementation:

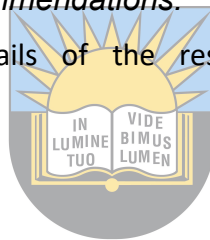
The chapter describes the implementation of the plant identification and mobile application models.

Chapter 5: Results and Discussion:

The chapter describes and discusses the study's findings from the research implementation.

Chapter 6: Conclusion and Recommendations:

The chapter summarizes the details of the research findings and covers related recommendations.



1.9 Conclusion

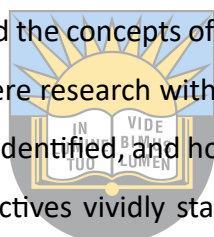
The proposed research aims to enhance the accuracy, efficiency, and accessibility of plant identification, benefiting traditional healers, botanists, herbalists, healthcare practitioners, and the public in their utilization of medicinal plants. The development of a mobile application to auto-detect medicinal plants using an AI approach has the potential to revolutionize the field of plant identification. This research presents a comprehensive plan to develop such an application, emphasizing the integration of AI algorithms, image recognition techniques, and a comprehensive medicinal plant database. The next chapter presents a review of related literature for this research.

CHAPTER TWO

2. LITERATURE REVIEW

2.1 Introduction

It is crucial to note the novelty of this work, and it is equally imperative to acknowledge that it is a path that other scholars have embarked upon in the past. Every scientific research builds on previous knowledge as a systematic investigation to spread new conclusions and establish facts. This chapter explores previous work on similar areas of study and reviews some existing research views behind it. It also seeks to help comprehend in-depth the fundamental concepts, research methods, and experimental techniques used in the field. It also brings an insight into how other scholars applied the concepts of interest in their research findings. This chapter also displays the settings where research within this field has been of interest. Gaps from the existing research have been identified, and how the current research, as indicated in the previous chapter under the objectives vividly stated within the scope of the indicated specific objectives as the building blocks of this research, will add value to the field about the subject in question.



University of Fort Hare
Together in Excellence

The first field to be examined is software development, which has been used extensively to convert manual and automated processes (Zhu et al., 2020). Automation ensures a quicker and more efficient workflow as most processes need limited human intervention compared to the manual processes replaced by automated processes (Khatib et al., 2021).

The second field looked at is machine learning, which replicates the human brain when processing real-life situations (Marcus, 2018). Machine learning helps introduce human-like thinking to machines and systems (Marcus, 2018).

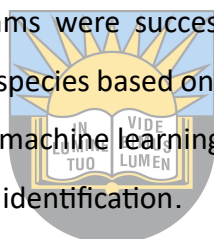
Therefore, previous research in mobile application development must be examined. The literature review describes methodologies and research that cover similar works to the current study on identifying medicinal plants.

Identification of medicinal plants has been a subject of interest in various fields, including botany, ethnobotany, pharmacology, and traditional medicine (Süntar, 2020). With the advent

of artificial intelligence (AI) and its application in image recognition, a growing body of research has explored the potential of AI-based approaches for automating the identification of plant species, including medicinal plants (Süntar, 2020). The literature review highlights vital findings and advancements in developing mobile applications that utilize AI for auto-detecting and identifying medicinal plants.

2.2 AI Techniques in Plant Identification

Researchers explored various AI techniques for plant identification, focusing on machine learning algorithms, particularly deep learning methods such as convolutional neural networks (CNNs) (Wäldchen and Mäder, 2018). CNNs displayed remarkable capabilities in image recognition tasks, enabling accurate identification and classification of objects within images (Giger, 2018). These algorithms were successfully applied to plant identification, including recognizing medicinal plant species based on their visual features (Abdollahi, 2022). Discussed in deep learning under the machine learning section later in the chapter, are some researches that utilized CNN in image identification.



University of Fort Hare
Together in Excellence

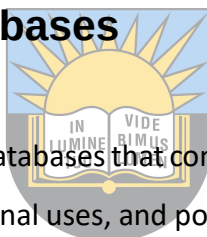
2.3 Image Datasets and Preprocessing

The availability of high-quality image datasets plays a critical role in training AI algorithms for plant identification (Tian *et al.*, 2020). Researchers have compiled and curated large-scale datasets of plant images, including medicinal plants, to train AI models (Singh, 2018). Preprocessing techniques, such as image augmentation, normalization, and feature extraction, have enhanced the accuracy and generalization of AI models (Castiglioni *et al.*, 2021). It is crucial to have a large data set when training a deep learning model, the data helps in avoiding overfitting. The large data set gives the machine learning models the ability to capture complicated patterns, and differences in the data while supporting more complex models. Such large data sets require a workstation with high specifications as the memory and the central processing unit will be working intensively in training the models.

2.4 Feature Extraction and Representation

Extracting relevant features from plant images is crucial for accurate identification (Muneer and Fati, 2020). Researchers explored various approaches, such as shape analysis, texture analysis, color histograms, and deep feature representations, to capture the distinctive characteristics of medicinal plants (Lee *et al.*, 2017). Feature extraction techniques aim to transform raw image data into informative representations that can be used by AI algorithms for classification (Bi, Xue, and Zhang, 2020). The deep learning method CNN is regarded as a very effective model in feature extraction. In research about tumor feature extraction using CNN, it was concluded that convolutional neural networks display very high accuracy, sensitivity and specificity in image feature extraction, pattern recognition, and classification.

2.5 Medicinal Plant Databases



The development of comprehensive databases that contain information about medicinal plant species, their properties, their traditional uses, and potential medical applications is essential for accurate plant identification (Anand *et al.*, 2019). These databases are references for AI algorithms and provide users with valuable information about identified plants (Jha *et al.*, 2019). Researchers curated and expanded existing databases to include various medicinal plant species, enhancing the effectiveness and usefulness of AI-based plant identification applications (Noor *et al.*, 2022). There are online databases that were available for a study of medicinal plants on platforms like Pacific Data Hub, Mendely Data, Science Direct, and many others. Online data sets make it easier to analyze data, should the data set for research already exist. The challenge with this research was there no data set that matched the requirements for this research. Data set requirements for this research were:

- a) Plants captured in their natural habitats
- b) The entire plant considered, not just a leaf or flower
- c) Plant images captured in different angles (30%, 60% & 90%)
- d) Plant images captured on different times of the day (morning, midday and afternoon)

- e) The data set requirements made it difficult to find an online data set. That forced a collection of the data set for this research to be done.

2.6 Mobile Applications for Plant Identification

The utilization of mobile applications for plant identification has gained significant attention due to their widespread availability and accessibility (Nowak *et al.*, 2020). Several mobile applications that employ AI algorithms for auto-detecting and identifying plant species, including medicinal plants, have been developed. These applications allow users to capture plant images using their smartphones and receive instant identification results and additional information about the identified plants (Boho *et al.*, 2020). One of such mobile applications is known as PictureThis, which allows you to capture a plant image or upload an already captured image. PictureThis displayed only the results of the identified plant image with no accuracy percentage displayed and other similarities. Another mobile application that was looked at is known as Pl@ntNet, which also allows for an image upload or snapshot of the plant image. Pl@ntNet displays the identified plant image with the identification percentage, also displayed are other plants that Pl@ntNet deems similar to the plant being identified. Pl@ntNet allows for identification selection of different parts of the plant image (e.g. bark, leaf, flower, fruit, and other which must have some of the other identification aspects for identification to be possible). This makes Pl@ntNet in identification of the entire plant as done in this research.

2.7 Performance Evaluation and Validation

Performance evaluation of AI-based plant identification applications is crucial to assess their accuracy, robustness, and usability (Tian *et al.*, 2020). Researchers have conducted rigorous testing, comparing the results of AI algorithms with those of experts who manually identified them. Metrics such as precision, recall, and F1 score have been used to evaluate the performance of AI models, and user feedback has been used to assess the usability and effectiveness of mobile applications (Souza *et al.*, 2022).

Overall, the literature demonstrates that the development of a mobile application to auto-detect medicinal plants using an AI approach has the potential to revolutionize plant identification practices. The utilization of AI algorithms, coupled with comprehensive plant databases and user-friendly mobile interfaces, can enhance the accuracy, efficiency, and accessibility of plant identification, supporting the utilization of medicinal plants in healthcare practices, research, and conservation efforts. However, further research is needed to address challenges such as plant species diversity, image quality variations, and the inclusion of traditional knowledge in medicinal plant databases (Antonelli *et al.*, 2020).

Validation in machine learning differs from model to model. The supervised learning models such as Logistic and Linear regression need the model to be checked for overfitting and underfitting as validation requirements. The support vector machine (SVM), decision trees and random forests validation requirements are as follows: split data set into training and test data sets. The SVM, decision trees and random forests are then trained on the training data set and tested on the test data set. The unsupervised learning model's validation requirements differ depending on the data set and requirements. Models such as anomaly detection usually require ROC curves and precision-recall curves to measure performance. Neural networks models use a similar validation model to SVM, decision trees, and random forests where data sets are split into training data and test data. K-fold cross-validation is also utilized in the neural networks models where data sets are randomly selected into k groups which are approximately equal in size.

2.8 Development of a Mobile Application

2.8.1 Identification of medicinal plants

Identification of medicinal plants has been part of people's day-to-day lives, especially for herbalists, specifically in Africa (Tembo *et al.*, 2021). The main features used are the plant's leaves, color, and texture (Mostajer Kheirkhah and Asghari, 2019). With the dawn of civilization, humankind gradually migrated towards the use of pharmaceutical drugs abandoning indigenous healing methods (Skrbo *et al.*, 2022). Nevertheless, the majority of

traditional healers still rely on medicinal plants to cure their patients (Tesfahuneygn and Gebreegziabher, 2019).

In 2019, the Covid-19 pandemic prompted even ordinary individuals to familiarize themselves with these approaches, as the world sought remedies from nature to effectively handle the pandemic (Iheanacho, 2022). The science of healing directed the masses to the properties of herbs to help manage illness. Plants such as African wormwood (Mhlonyane) were wildly recommended; Mhlonyane have been growing in people's backyards for years; however, with the dawn of modern medicine, they were abandoned (Howes et al., 2020). COVID-19 then brought people back to primitive ways of healing, and people were obliged to know about these and their benefits (Pieroni et al., 2020). There is growing interest in natural plant-based remedies as a source of commercial products (Najmi et al., 2022).

South Africa boasts of these medicinal plants; unfortunately, these have not been widely explored (Sadgrove, 2020). Around 80% of South Africans use traditional medicines to meet their primary healthcare needs; however, only a few South African medicinal plants have been exploited to their full potential in commercialization (Van Wyk and Prinsloo, 2018). Many other medicinal plants are at people's disposal, but many need help with them and their uses (Eshete and Molla, 2021). As the globe advances and many people become more technologically inclined, a system must be developed to help people identify these medicinal plants using the resources they are exposed to. This way, modern meets primitive towards a common goal. Using the gadgets at their disposal, people can use the software to identify the plants and learn their properties and benefits. Science couldn't be more instrumental.

2.8.2 Software development

Developing a mobile application that identifies medicinal plants involves deciding which software development methodology to use. Gechman (2019) advised that choosing a software development methodology is crucial for successfully delivering a project. Therefore, the following methods were considered when building the mobile application for the research.

2.8.3 Waterfall methodology

Waterfall methodology was the first model to be introduced in software process modeling in 1970 by Dr, Winston, Royce by outlining the linear and sequential approach to software development (Senarath, 2021). It is structured so that all activities of a particular step are connected to the next step. This approach is more suited for project management, where all costs have already been planned, and there is no space for more funds or time to deliver the project (Aroral, 2021). The method needs proper planning for each development step as it can't be repeated once it has been completed (Senarath, 2021).

The methodology becomes very difficult to use as any modifications required after development starts cannot be included in that software release. Another software version will have to be released to accommodate whatever modification is required (Gechman, 2019). The waterfall architecture is shown in Figure 2-1.

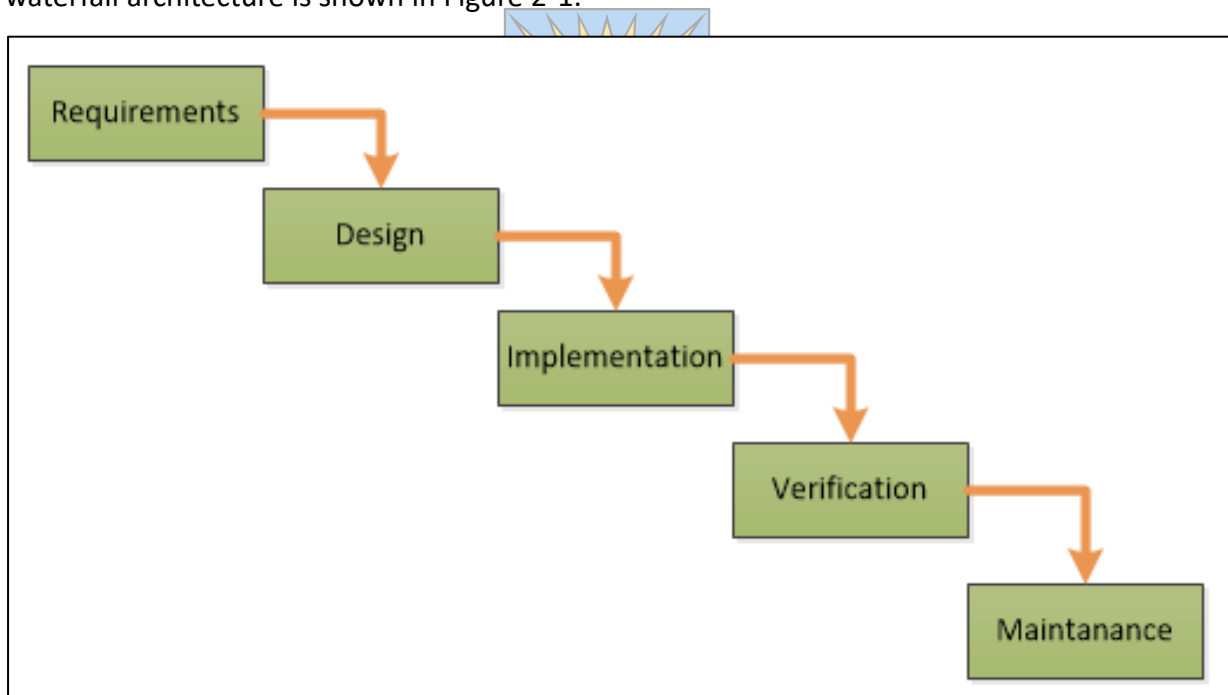


Figure 2-1: Waterfall methodology (Source: Gechman, 2019)

The following steps represents the waterfall methodology:

- a) **Step 1: Requirements** in software development refer to the necessary factors, components and even functions to be in place before development can take place. The process involves people who require a system and meeting software development teams to determine system requirements. In some organizations, business analysts

mainly collect system requirements. The analyst drafts a solution for the business requirement, sometimes involving software development (Dutta, 2022).

- b) **Step 2: Design** is creating a solution document for system requirements, developers and system analysts design a software solution. The design process creates solutions such as system architecture, how the user interface will look, database design, data flow throughout the system, and logical diagram design (Senarath, 2021).
- c) **Step 3: Implementation** is when the system's design is finalized, the system's building begins. Implementation occurs when the programming according to the system design occurs, and the programming code gets converted into a user application (Kramer, 2018).
- d) **Step 4: Verification** this stage of the waterfall model is where the implemented system is tested (Kramer, 2018). This is the phase where the system is tested to see if it meets user requirements (Kramer, 2018). The test also focuses on system glitches and bugs. This is the phase where the system is deemed successful (Kramer, 2018). If the system verification is successful, the system will be released; if not, the project will fail (Kramer, 2018).
- e) **Step 5: Maintenance** is when the software undergoes changes and assume different versions improving system performance and addressing errors picked up during the verification process (Senarath, 2021). This phase enhances the system's quality and reliability (Senarath, 2021).
- f) One of the studies that adopted the waterfall methodology was during the software development by Sujana et al., (2021) on plant identification and data processing. The system aimed to provide plant reports and create a plant database. The plant information was manually inserted into an SQL database (Sujana et al., 2021). System users could log on and capture plant data into the database. The system was successfully developed using the waterfall method and worked as intended; all the

plant information added to the database was done successfully, and reports were generated (Sujana et al., 2021).

2.8.4 Agile

The Agile methodology was developed based on an Agile manifesto created by 14 leading figures in the software industry (Al-Saqqqa, Sawalha, and Abdelnabi, 2020). Manifesto describes different software development approaches that work and those that do not work (Senarath, 2021). So, in this methodology, the software is built in iterations, which assists with continuously testing the software in the phases in which it is built (Al-Saqqqa, Sawalha, and Abdelnabi, 2020). Identifying software issues such as bugs and software changes is quickly dealt with in this methodology, and all the testing and software iterations happen before the final software release (Saeed et al., 2019). The agile method works well in flexible or unstable environments where system requirements or design can change (Senarath, 2021).

The following are different agile methodologies: crystal, extreme programming, feature-driven development, and scrum (Kumar, Maheshwary, and Malche, 2019). Architecture is helpful for software development, where the software needs constant changes and testing (Kumar, Maheshwary, and Malche, 2019). The Agile methodological architecture is illustrated in Figure 2-2:

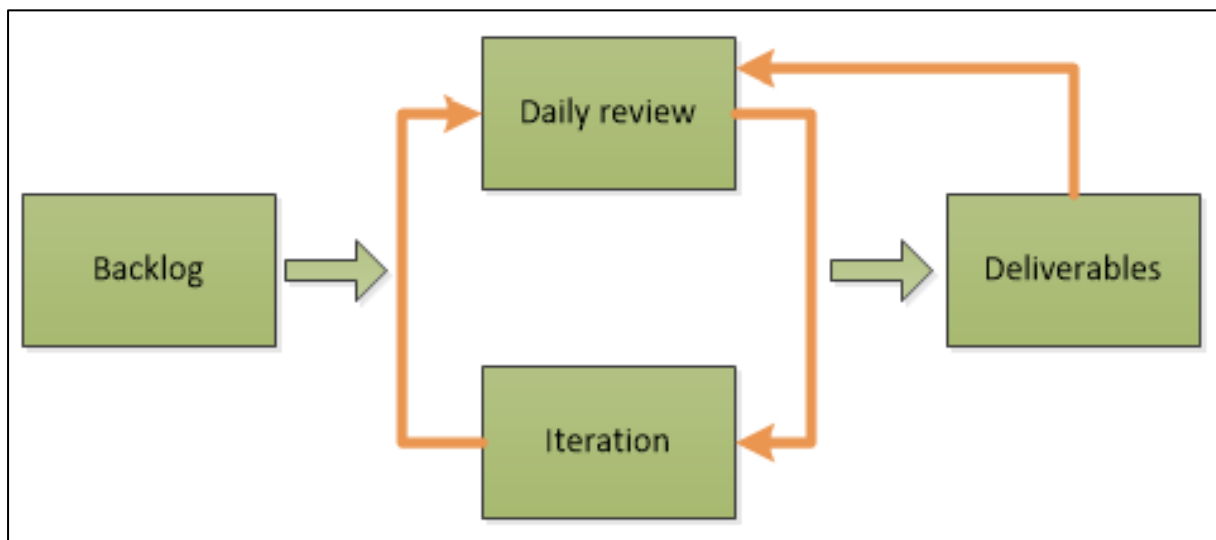


Figure 2-2: Agile methodology

The following steps represent the Agile methodology:

- a) **Step 1: Backlog**—all the system requirements converted into system designs for implementation (Jiménez, Afonso, and Fernandes, 2020). The agile methodology allows for arranging backlogs into groups of manageable deliverables (Jiménez, Afonso, and Fernandes, 2020). The software can be developed and tested in steps; hence, the input value is backlogged (Senarath, 2021).
- b) **Step 2: Iteration** - involves developing the software and making any required changes to the system. Iteration is where the system's coding will occur (Kumar, Maheshwary, and Malche, 2019).
- c) **Step 3: Daily review**—is frequent assessment of the software during development. The review is done until the system satisfies the requirements (Al-Saqqqa, Sawalha, and Abdelnabi, 2020).
- d) **Step 4: Deliverables** is where the backlog has been delivered (Kurniawan, Shidiq, and Sutoyo, 2020). A backlog could be the login process of the system; therefore, a deliverable, in this case, would be a system login page where users can log on (Kurniawan, Shidiq, and Sutoyo, 2020). Agile allows for the capability to move to the subsequent backlog after a successful deliverable until the software has been successfully delivered. One of the success stories in which the agile methodology was used is the development of Spotify. Agile methodology assisted the developers of the Spotify application after trying the SCRUM methodology, which did not succeed because of its very restrictive nature (Salameh and Bass, 2022). SCRUM methodology was replaced by agile, which proved to be a perfect fit for the agile environment of music and different streaming platforms (Salameh and Bass, 2022).

2.8.5 Rapid application

The methodology has four phases: requirements planning, user design, construction, and cutover (Yen and Davis, 2019). Rapid application is most suited for fast-paced development

projects with limited funds, as this methodology's system requirements remain the same. Still, the design and construction are repeated until the user/system owners are satisfied (Yen and Davis, 2019). The rapid application methodology arranges requirements into prototypes that can be improved gradually until the system is delivered (Dogiparthi, 2019). The architecture for rapid application is shown in Figure 2-3 below:

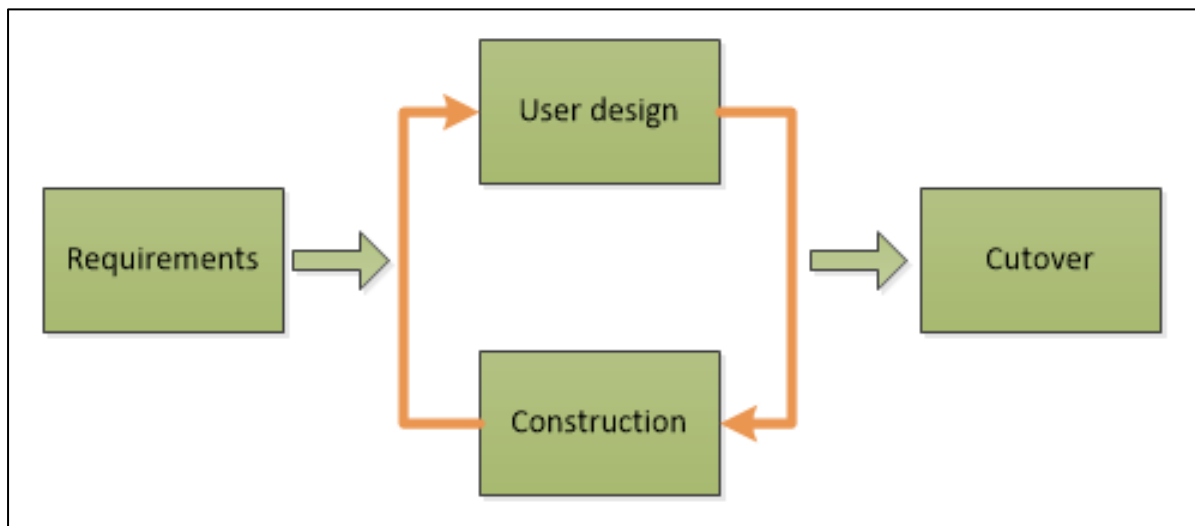


Figure 2-3: Rapid application

Rapid application methodology is defined by following steps:

- a) **Step 1: Requirements** are collected on platforms like focus groups and conferences (Dogiparthi, 2019). Not much time is spent collecting system requirements (Dogiparthi, 2019).
- b) **Step 2: User design** is when the requirements have been gathered, and a design is created in accordance with them. (Yen and Davis, 2019).
- c) **Step 3: Construction** – The software is coded in prototypes (Dogiparthi, 2019). The prototypes are created using the user design and tested frequently by the users (Dogiparthi, 2019).
- d) **Step 4: Cutover** is when the software is delivered to the users when the final prototype is accepted as meeting system requirements (Dogiparthi, 2019). Maulana et al. (2021) developed an application in Indonesia for students to interact with tutors on different

learning programs offered. The application was successfully built and included an admin account for monitoring all the learning and system-related issues (Maulana et al., 2021). The admin account can load and edit learning material on the application (Maulana et al., 2021).

2.8.6 Mobile application operating system

The mobile application was developed on an Android OS platform. The operating system was chosen as it is an open source and a flexible system to create (Alwalah, Alshhry, and Elbadri, 2019). Android is a Java platform that provides a customized virtual machine (Alwalah, Alshhry, and Elbadri, 2019). The Android platform has excellent developer tools like Java and Eclipse IDE (Alwalah, Alshhry, and Elbadri, 2019). Compared to one of its rivals, which is IOS, where the Apple-owned OS has a development platform that needs to be defined correctly, and the IOS uses objective-C, which is not easy to use (Alwalah, Alshhry and Elbadri, 2019)

2.8.6.1 Android



Google developed an open-source mobile operating system built on a modified Linux Kernel called Android (Sarkar et al., 2019). Android software stack containing Java-written applications run on a virtual machine (Mesia Dhas and Shiny Angel, 2022). The Android software is stacked according to the following layers shown in Figure 2-4:

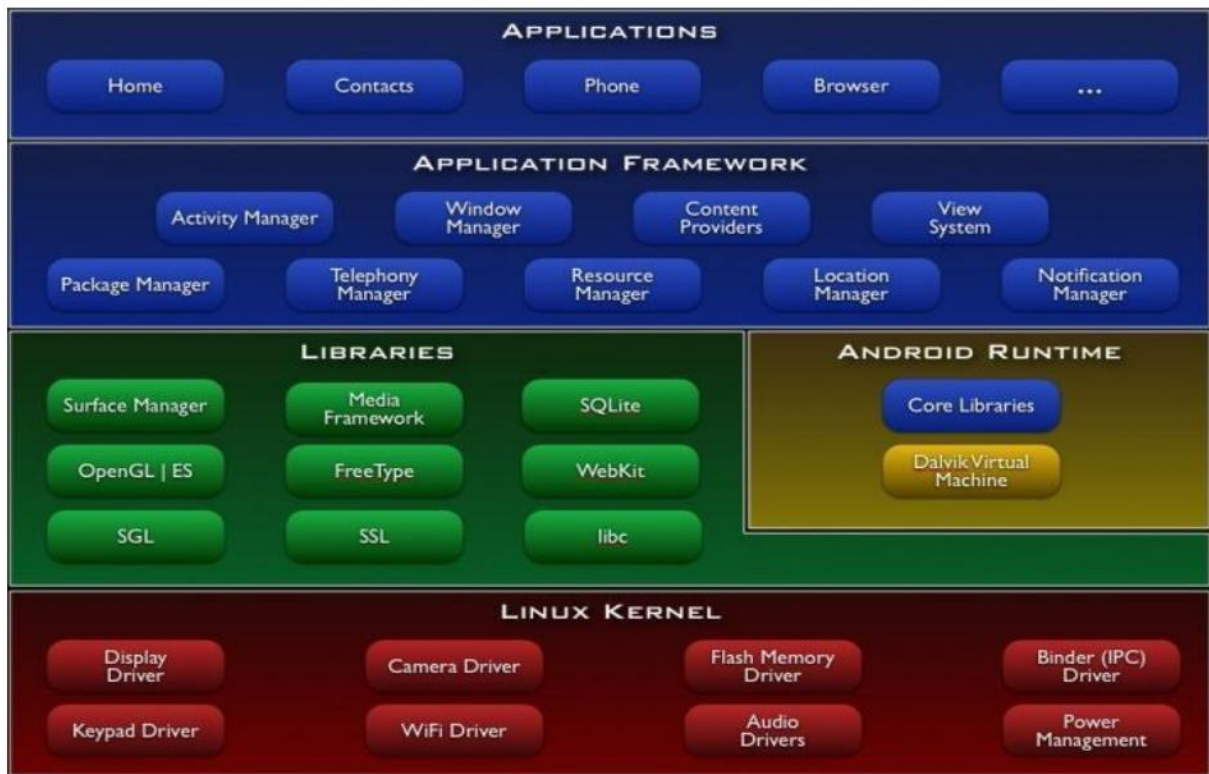


Figure 2-4: Android software stack (Source: Mesia Dhas and Shiny Angel, 2022)

The operating system stack of Android written in Java, C++, C, and XML is as follows: Applications, Application Framework, Android runtime, Native libraries, and Linux Kernel (Xenakis Christos, 2022). As the mobile application is built to be hosted on a smartphone, there is a need to look closer at the smartphone hosting of a mobile application that contains scientific data. Some challenges have been found by Khan et al. (2022) in the scheduling part of smartphones.

As smartphones improve, manufacturers put a lot of work into battery life-saving (Balaji, Nathani, and Santhakumar, 2019). These power-saving initiatives interfere with scheduled tasks that must run at certain times. González-Pérez et al. (2022) discovered some work developers did to try and counter the interference of smartphone operation systems on mobile application scheduled tasks. Initiatives like “Don’t kill my app” and “AutoStarter” show the extent of the challenge faced by developers when programming on a smartphone.

Although the mobile application developed does not have a scheduled task that has to run continuously in the background collecting data, a challenge is faced when a smartphone is in

maximum power saving mode and a task that needs to be sent from the mobile application. That task only reaches the intended recipient in time, as smartphones stop most applications in maximum power mode. The application created in this current research is no different and is affected by these power-saving initiatives by the manufacturers of smartphones. González-Pérez et al., (2022) suggest that using alarms is a better way of dealing with the issue of having a task that needs to be carried out on the mobile application. Using alarms does free up some battery life as the application will not be running all the time (in the foreground or the background of the smartphone) while the task is still to be carried out. Even though the research done by González-Pérez et al. (2022) dives deeper into mobile applications that might need to run multiple tasks within a short space of time (like within a minute), this research rarely has various functions that are run within a minute. Research by González-Pérez et al., (2022) is relevant to work done as there could be cases where a superuser sends tasks to multiple users simultaneously. The mobile application created needed to be adaptive under those circumstances to ensure the continuous working of the mobile application.



Prasandika et al. (2021) developed a mobile application called Aayu, which is a plant-identifying application. The application users capture an image of the plant using a mobile phone camera, and the application analyzes the image using the CNN (Convolutional Neural Networks) methodology. Therefore, the application displays the identification results (Prasandika et al., 2021).

The front end of the Auyu mobile application was developed using React Native (an open-source platform for creating mobile applications), the Java programming language, the Integrated Development Environment (IDE), and Android Studio (Prasandika et al., 2021). The back end was developed using Java and Java Spring Boot (which is powerful and flexible for developing database transactions) (Prasandika et al., 2021). In a research done by Affouard et al., (2017), a mobile application called Pl@ntNet was developed. This application is a repository for botanical information with plant identification done across three platforms: An Android application, an IOS application (which was created when the research paper was released), and a web application. The architecture of the Pl@ntNet mobile application is represented in Figure 2-5 below.

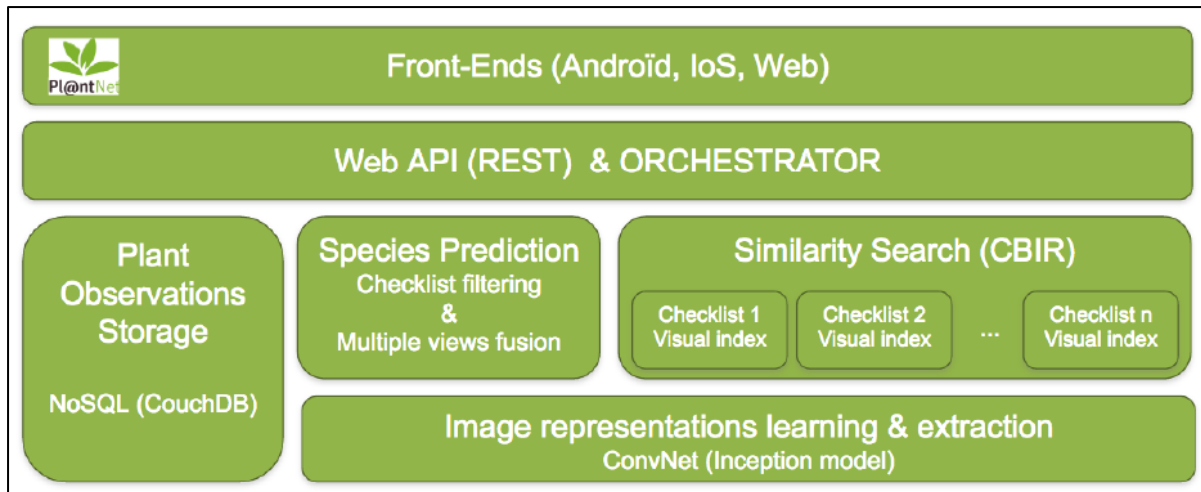


Figure 2-5: Pl@ntNet architecture (Source: Affouard et al., 2017)

Plant observation storage – a database of plant images captured by system users. An algorithm is used to identify the plant; if the identification percentage exceeds a certain threshold, it is stored in the NoSQL document called CouchDB (Affouard et al., 2017). Image representation learning and extraction—this is the mobile application training using machine learning algorithms (covered in the next section) to identify plants. The mobile application Pl@ntNet was trained in a supervised method using a convolutional neural network (CNN) (Affouard et al., 2017).

University of Fort Hare
Together in Excellence

2.9 Machine learning

Machine learning focuses on teaching machines the processes behind human decision-making in certain situations (Carvalho, Pereira, and Cardoso, 2019). When teaching machines human behavior and decision-making, data that needs to be analyzed by machines for decision-making is arranged so that machines can start learning their patterns (De Laat, 2018). This gave birth to supervised and unsupervised machine learning (Rashidi et al., 2021).

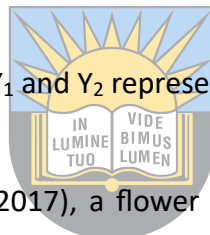
Supervised machine learning is where data used to train machines is labeled (Bagherzadeh and Asil, 2019). Unsupervised machine learning uses unlabeled data (Bagherzadeh and Asil, 2019). Machine learning has different algorithms for image analysis (Batta, 2018). Some of those algorithms will be discussed, and the impact these algorithms had in other research conducted in the field of mobile application creation for plant identification will be examined.

2.9.1 Nearest Neighbour

The nearest Neighbor method, also known as K Nearest Neighbor (KNN), where K represents the number of nearest neighbors included in decision-making, is an algorithm that saves all available data points and classifies new data points according to the classification of its neighbors (Bansal, Goyal and Choudhary, 2022). KNN method is applied by selecting the K number of neighbors, calculating the Euclidean distance using Equation 1 of K number of neighbors, taking K nearest number as per calculated Euclidean distance, and count several data points in each category from neighbors, then finally assign data point to the category which has the most number of data points (Abu Alfeilat et al., 2019).

$$\text{Euclidean distance} = \sqrt{(X_2 - X_1)^2} + \sqrt{(Y_2 - Y_1)^2} \quad \text{Equation 1}$$

Where X_1 represents, X_2 , represents, Y_1 and Y_2 represents.... respectively



In a research by Gogul and Kumar (2017), a flower was used to identify a plant using 28 different plant species, with 80 pictures of each species (Gogul and Kumar, 2017). The methodology used to analyze the plant images were captured using a smartphone embedded with a convolutional neural networks (CNN) mobile application. The classifiers used in this research are a combination of multiple classifiers. Using one classifier might affect the image quality and thus jeopardize the identification accuracy of the Computer Vision system the researcher is building (Gogul and Kumar, 2017).

Classifiers used were OverFeat CNN, Bagging trees, Linear Discriminant Analysis, Gaussian Naïve Bayes, KNearest Neighbor, Logistic Regression, Decision Trees, Random Forests, and Stochastic Gradient Boosting (Gogul and Kumar, 2017). This system works by taking an image of a plant by smartphone, making sure the flower part of the plant is the focus of the image (Gogul and Kumar, 2017). The image is then sent through the CNN algorithm for classification and identification. Firebase is an image storage platform (Gogul and Kumar, 2017). The mobile application talks to a server where the CNN algorithms are hosted. Java is used to develop

mobile applications, and Python is used to create the CNN subsystem (Gogul and Kumar, 2017).

The test done in this research was divided into three parts. Firstly, OverFeat CNN was used for plant feature extraction (Gogul and Kumar, 2017). Secondly, Bagging trees, Linear Discriminant Analysis, Gaussian Naïve Bayes, KNearest Neighbor, Logistic Regression, Decision Trees, Random Forests, and Stochastic Gradient Boosting are used as classifiers (Gogul and Kumar, 2017). Thirdly, images are randomly selected and inserted into the network for identification purposes (Gogul and Kumar, 2017). The results for the different classifiers are shown in figure 2-6 below.

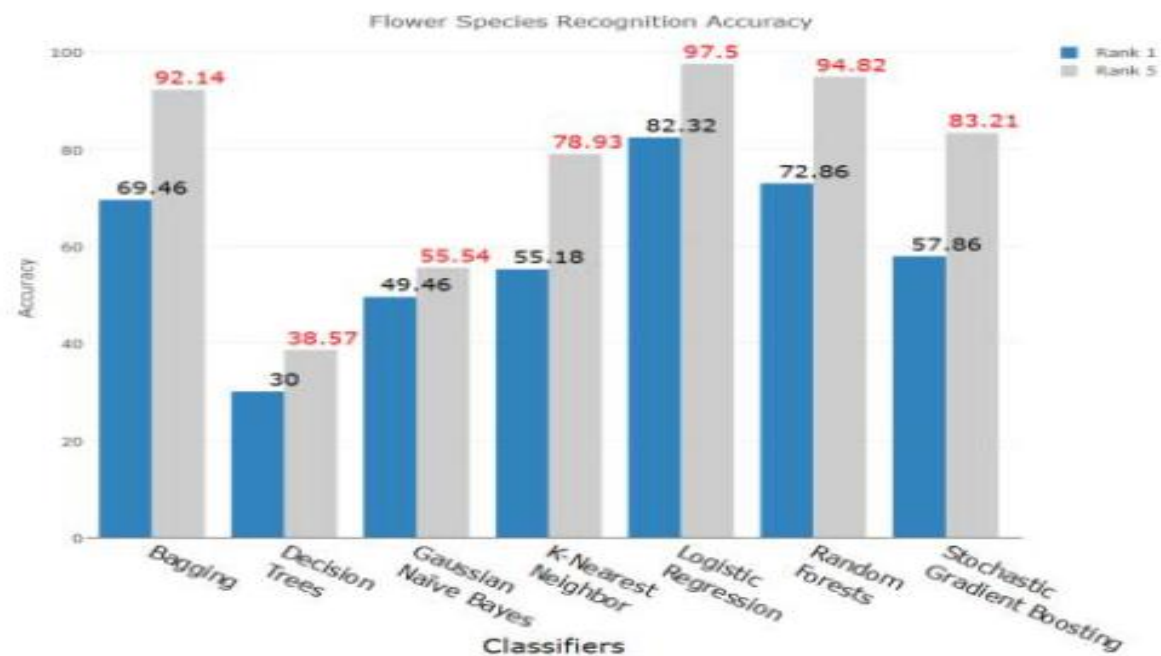


Figure 2-6: Performance Results of the flower detection system developed by Gogul and Kumar, (2017)

2.9.2 Naïve Bayes

Naïve Bayes is a set of classification algorithms based fundamentally on the independence and equality of each classified feature (Wickramasinghe and Kalutarage, 2021). Bayes Theorem is the influence behind the creation of the Naïve Bayes algorithms represented by Equation 2 (Mwamba et al., 2020). The theorem is based on probabilities, where an event is based on an already occurred event.

$$P(A|B) = P(B|A)P(A) / P(B)$$

Equation 2

Where A and B are the events, A is the probable event that the theorem looks for based on the already-known event B (Ojo, Olanrewaju, and Folorunso, 2017). Although there are different types of Naïve Bayes classifiers, the focus will only be on the Gaussian Naïve Bayes classifier.

2.9.2.1 Gaussian Naïve Bayes classifier

This classifier deals with continuous probable values for each feature, distributed by the Gaussian distribution (also known as the normal distribution) calculated using Equation 3 (Gemaque et al., 2020).

$$P(X|Y = c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(x-\mu_c)^2}{2\sigma_c^2}}$$

Equation 3



Where X and Y are the events (as explained under Naïve Bayes above), sigma and mu are the variances (Ojo, Olanrewaju, and Folorunso, 2017).

University of Fort Hare
Together in Excellence

In research by Archana et al. (2022), rice plant disease is identified using a support vector machine-based probabilistic neural network (NSVMBPNN) as an image classifier. The background noise was removed from the images using a modified K-means algorithm (Archana et al., 2022). The image classifier was then compared to other classifiers (Naïve Bayes and Probabilistic neural network) for performance (Archana et al., 2022). There were four categories in which the images were divided Bacterial leaf blight, Brown spot, Healthy leaf, and Rice blast. Every plant category had 125 leaf images, 75% of which were selected randomly to be the training dataset, and the remaining 25% were for testing purposes. Figure 2-7 shows the results from the research done by Archana et al., (2022).

Performance metrics	Bacterial leaf blight	Brown spot	Healthy leaf	Rice blast
NSVMBPNN	95.20	97.60	99.20	98.40
Naïve Bayes	90.40	92.00	93.60	92.00
PNN	95.10	93.63	98.20	95.23
SVM	88.00	96.00	97.60	92.80

Figure 2-7: The performance of different classification methods in detecting different rice diseases

Even though the Naïve Bayes classifier was not the best-performing classifier in this research, it is worthwhile to see how the classifier fares compared to other classifiers. Archana et al., (2022) demonstrated different results the classifiers can produce; even though the research doesn't explain the weather conditions the images were captured in (to see if all the images were captured under the same conditions), it was described in the research that except for 16 images found on the Indian Rice Research Institute (IRRI) website, all the other images were captured using a Canon EOS 3000D camera device. The researcher did not define the device to capture the 16 photos (Archana et al., 2022).



2.9.3 Decision Trees

University of Fort Hare
Together in Excellence

The algorithm analyzes data and arranges data features into a flow chart that assists with making the desired choice from the provided data using the training datasets (Bansal, Goyal, and Choudhary, 2022). The decision tree consists of the following: root node, branches, and leaf node (Lee, Cheang, and Moslehpour, 2022). The root node is the main aim the algorithm is trying to achieve; the root is at the top of the tree (Alzubi, Nayyar, and Kumar, 2018). Branches represent different options available in decision-making from the root node (Banerjee et al., 2019).

A leaf node is at the end of each branch, representing the possible outcome of that branch (Anticipatory and Improvement, 2020). There are two types of leaf nodes: the square leaf node and the circle leaf node (Ibrahim and Shiba, 2019). The square leaf node represents a possible decision to be made and the circle leaf node represents a chance event or an

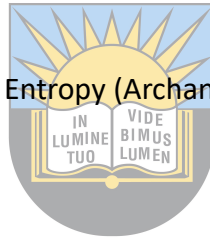
unknown outcome (Świechowski et al., 2023). The decision tree is designed so that information flows from the top downwards; as you move downwards, the information impurity increases (Lee, Cheang, and Moslehpour, 2022). Splitting measures are used to solve such issues; some are Entropy, Information Gain, and Gini Index (Malhotra and Singh, 2021).

2.9.3.1 Entropy

Entropy seeks to measure inaccuracy or impurity in data (Rama Devi and Abualkibash, 2019). In Entropy, the value 0 means that there is no impurity in the data (there is data from the same class), and the value 1 means that the data has impurity (there is data from different classes) (Rama Devi and Abualkibash 2019) as represented by Equation 4:

$$Entropy = -\sum_{i,j,k=0}^{G-1} P(i,j,k) * \log (P(i,j,k)) \quad \text{Equation 4}$$

Where p is the probability function of Entropy (Archana et al., 2022).



2.9.3.2 Information Gain

University of Fort Hare
Together in Excellence

The splitting measure is dependent on Entropy. As impurity in the data is discovered, the information gain platform is defined (Gajowniczek and Ząbkowski, 2021). Information gain is the amount of information a feature has about a particular class (Cunningham and Delany, 2021). The order of attributes in a decision tree node is determined by information gain (Mienye, Sun, and Wang, 2019) represented by Equation 5 below.

$$Gain = E_{parent} - E_{children} \quad \text{Equation 5}$$

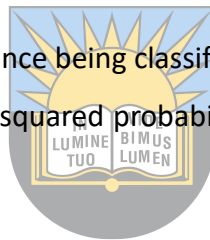
E_{parent} represents the Entropy of the parent node, and $E_{children}$ represents the average Entropy of the child nodes (Miller et al., 2020).

2.9.3.3 Gini Index

The Gini index determines the probability of a randomly selected feature that is not classified correctly (Naheed et al., 2020). The Gini Index varies between 0 and 1, where 0 is the purity of classification, and 1 is the random distribution substance of various classes (Díez-Sanmartín and Cabezuelo, 2020). A value of 0.5 represents an equal distribution of substances across different classes (Naheed et al., 2020). When creating a decision tree, the features with the lowest value of the Gini index will be the most preferred (Bansal, Goyal, and Choudhary, 2022). The Gini Index is represented as follows:

$$Gini\ Index = 1 - \sum_{i=1}^n (P_i)^2 \quad \text{Equation 6}$$

Where P_i is the probability of a substance being classified for a specific class, the Gini Index is defined by deducting the sum of the squared probabilities of each class from one (Agbokou and Mensah, 2023).



Research has utilized the decision tree as a methodology. MATLAB is used in research by Xue et al., (2019) where an application is developed to run on a smartphone or tablet. The plant leaves are used to analyze and apply the methodologies Xue et al., (2019) researched on scanned plant leaves. Twenty medicinal plant species were selected from a medicinal plant site for research. Samples without plant decays were collected from the site, measured with VIS/NRI (visible/near-infrared), and stored in zip-lock bags. These plant samples were then stored in a cooler bag with ice to avoid dehydration and transported to a laboratory. In the lab, plant leaf samples were scanned using a Brother MFC-660CN (Brother Industries, Ltd., Japan) scanner and analyzed using the code written in MATLAB (Xue et al., 2019).

Several machine learning classifiers (decision trees, discriminant analysis, logistic regression, support vector machines, nearest neighbor classifiers, and ensemble classifiers) were tested with artificial neural networks (ANN). ANN became the best machine-learning algorithm for this research (Xue et al., 2019). Twelve parameters were derived from the plant leaves

morphometric characteristics that were analyzed using MATLAB Neural Network Toolbox™ 10; this formed part of model A (Xue et al., 2019). Model B was based on the same machine learning algorithms where 401 wavelengths obtained by VIS/NRI were used as input data (Xue et al., 2019). Both models analyzed 20 plant species. Model A achieved an accuracy of 98.3%, and Model B achieved 92.5% accuracy (Xue et al., 2019).

2.9.4 Support Vector Machines (SVM)

SVM is a machine-learning method for image classification, outlier detection, and regression (Saranya et al., 2020). SVM creates a boundary between data points called a hyperplane (Cao, Liu, and He, 2020). Hyperplane helps SVM in decision-making when classifying data (Cao, Liu, and He, 2020). Data to be analyzed in SVM has two classes, and a decision boundary (hyperplane) is created between two classes (Bansal, Goyal, and Choudhary, 2022). All the data (two classes) separated by the hyperplane have some features that are like classified data. Multiple hyperplanes could exist between data points, and SVM could choose the best one for data classification (Cervantes et al., 2020). There are two types of SVM, Simple SVM (linear regression and classification) and Kernel SVM (non-linear data) (Ghosh, Dasgupta, and Swetapadma, 2019). SVM algorithms are helpful in cases with small data with many features (Cervantes et al., 2020). SVM has different kernel functions that can be specified for the decision function (Cervantes et al., 2020). These kernel functions are:

2.9.4.1 Linear

Linear function is useful for text classification (Deng et al., 2019). The linear function works well with many features (Deng et al., 2019). Equation 7 illustrates the linear kernel function (which describes the decision boundary of the SVM):

$$\text{Linear kernel: } f(X) = w^T \cdot X + b \quad \text{Equation 7}$$

Where X is the data to be classified, w is the weight vector, and b is the linear coefficient estimated from data used to train the algorithm (Margotti, 2023).

2.9.4.2 Gaussian Radial Basis Function

The Gaussian Radial Basis function (Equation 8) is helpful for non-linear data and is one of the most potent kernels (Ghosh, Dasgupta, and Swetapadma, 2019).

$$\text{Gaussian Radial Basis: } f(X_1, X_2) = \exp(-\gamma * ||X_1 - X_2||^2) \quad \text{Equation 8}$$

Where $||X_1 - X_2||$ is the dot product between features, γ shows how much a single point has on other data points around it (Hong, Lin, and Chen, 2019).

2.9.4.3 Polynomial Kernel

The Polynomial Kernel function (Equation 9) is utilized for image classification (Chandra and Bedi, 2021).

$$\text{Polynomial Kerne: } K(x_1, x_2) = (x_1 * x_2 + 1)^d \quad \text{Equation 9}$$

Where x_1 and x_2 represent the vectors, d is the degree of the polynomial (Hamzah and Othman, 2021).

Some research has used SVM as the methodology. The following research utilized many plant features for plant identification purposes. Features are color, size, texture, petal-type, petal count, disk flower, corona, aestivation of flower, and flower class (Patel and Patel, 2019). With so many plant features included in the research by Patel and Patel (2019), it is only fair to expect a high identification accuracy rate.

Identification accuracy was at 76.92% (Patel and Patel, 2019). The accuracy was achieved by analyzing 102 plant species comprising 25000 pictures (Patel and Patel, 2019). The research utilized two classification methods for plant analysis purposes: Multiple Kernel Labelling (MKL) and Support Vector Machine (SVM) (Patel and Patel, 2019). The methodology used to analyze the plant images was arranged: dataset, preprocessing, feature extraction, flower

classification, and classification output (Patel and Patel, 2019). A programming platform, MATLAB, is used to test the plant identification ability achieved by applying the methodologies analyzed by Patel and Patel (2019).

There is a success story on research conducted by Barré et al., (2017) where a mobile application developed for tree identification by identifying its leaves called Leafsnap. Mobile applications had been installed on nearly 1 million IOS devices (by the time of the journal publication) (Barré et al., 2017). This system works by the system user uploading an image of the plant leaf into the application; the user then tries to identify the leaf through an existing database (Barré et al., 2017). The database comprises professionally collected data by botanists (Barré et al., 2017). The system will identify plant leaves using a trained SVM methodology and give results on plant identity (Barré et al., 2017). If the plant is not part of the database, then there is an option where the user can add a plant leaf image and identify what plant it is (Barré et al., 2017).



2.9.5 Neural Networks

University of Fort Hare

Together in Excellence

Neural networks were developed to mimic the human brain neurons signaling to each other (Shao, Zhao, and Liu, 2021). The neural networks are also known as artificial neural networks (ANN) or simulated neural networks (SNN) (Shao, Zhao, and Liu, 2021). ANN comprises node layers: the input layer, the hidden layer (this could be more than one depending on the data being analyzed), and the output layer (Inbavel and Prakasam, 2022). The nodes, also known as artificial nodes, have weight and a threshold associated with them (Datilo, Ismail, and Dare, 2019). The node is only activated when the value of the node is above the specified threshold, passing data to the next layer of the ANN (da Silva, Elnabarawy, and Wunsch II, 2019). The data interaction between the nodes is shown in Figure 2-8 below.

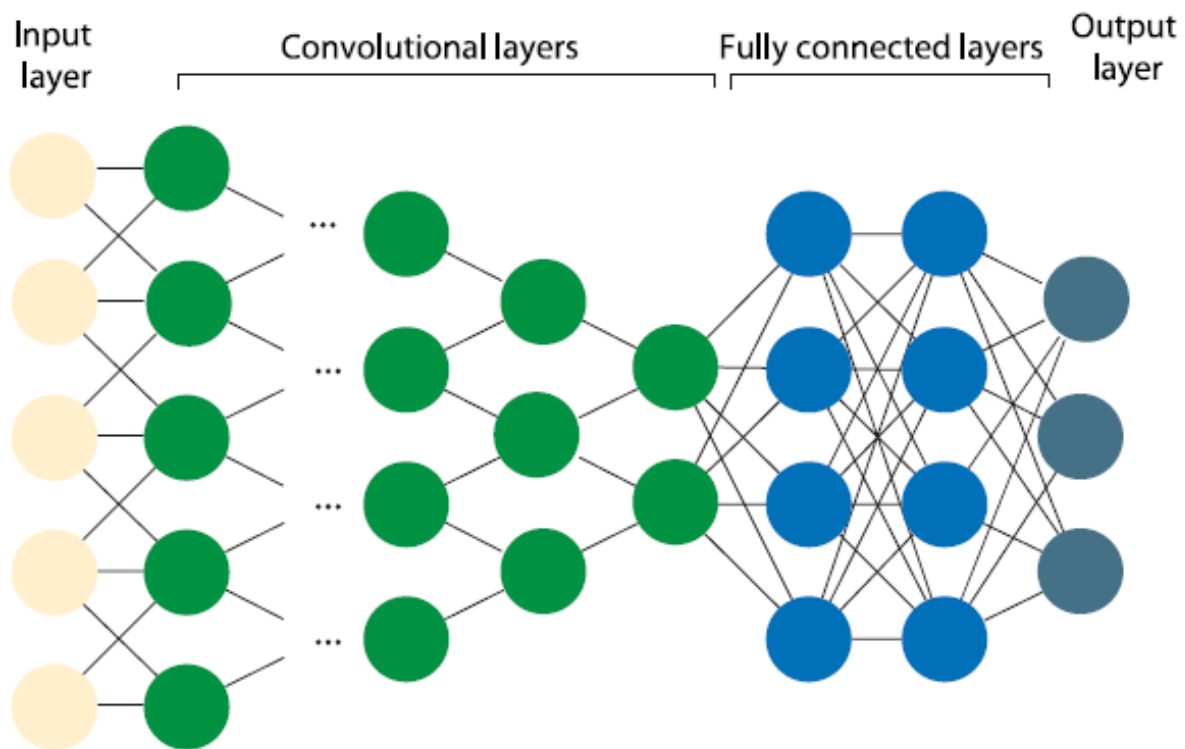


Figure 2-8: Neural Network (Source: Wäldchen and Mäder, 2018).

To set the threshold for the layers, Equation 10 was used:

$$\sum w_i x_i + \text{bias} = w_1 x_1 + w_2 x_2 + w_3 x_3 + \text{bias} \quad \text{Equation 10}$$

The output of the node is determined by the following:

$$\text{output} = f(x) = 1 \text{ if } \sum w_1 x_1 + b \geq 0; 0 \text{ if } \sum w_1 x_1 + b < 0$$

2.9.5.1 Deep learning

Deep learning and neural networks are closely related (Santos et al., 2021). A neural network that consists of three layers (including input and output) is a primary neural network; however, a neural network that consists of more than three layers (including input and output) is called a deep learning algorithm, where the deep refers to three or more layers (Pieroni et al., 2020).

In a study done by Shah, Nasika, and Otterpohl (2021) for the agriculture sector, a weed-detecting robot was conceptualized and developed (software). Even though this current research does not focus on robot creation, there are many similarities between the research done by Shah, Nasika, and Otterpohl (2021) and this one. The similarities are software development and plant identification (machine learning model training, on-field usage of the proposed solution). It is worthwhile to see how Shah, Nasika, and Otterpohl (2021) approached this research.

Research is conducted in rice fields of Kashmir in India (Shah, Nasika and Otterpohl, 2021). Onshape design software designs the mobile robot platform (Shah, Nasika, and Otterpohl, 2021). The robot is conceptualized to operate between rows of rice fields and detect weeds while in the early stages of development (Shah, Nasika, and Otterpohl, 2021). Hardware aspects of the robot won't be discussed as they have no relevance in research, while software aspects have some interest that needs to be checked. Shah, Nasika, and Otterpohl (2021) suggested that C++ or Python be used for robot software with OpenCV libraries for system intelligence (as it is widely used for object identification, facial recognition, and tracking objects).



University of Fort Hare

The dataset for research consists of 200 images of three weed species taken under different conditions not explained by Shah, Nasika, and Otterpohl (2021). Python was used to build and train a neural network model that analyzed the dataset with TensorFlow object-detecting API (Shah, Nasika, and Otterpohl, 2021). A structured approach was followed by the researcher where the process of training the TensorFlow algorithm was divided into preparing the workspace, annotating images (Labelling was used), generating TF record file formats input files, configure-train-optimize model, exporting the inference graph for testing (Shah, Nasika and Otterpohl, 2021). The trained model detected 97% identification accuracy on more enormous plant weeds, 88% - 99% for medium-sized plants, and 98% on smaller plants identification rate (Shah, Nasika, and Otterpohl, 2021). The collected images were taken from their natural habitat without removing them (Shah, Nasika, and Otterpohl, 2021). This point is crucial as it is how the data in this research was collected (in its natural habitat, taking a picture of its surroundings). Most researchers focused on plant leaves or flowers; Shah,

Nasika, and Otterpohl (2021) worked on the plant the way it is where it is (in its natural habitat).

In another study conducted by Cui and Qu (2021), no mobile application was created for image processing. Still, Pytorch's deep learning framework was utilized to construct a convolutional neural network for fire recognition for images and videos. The photos to test the build model were collected from the internet and median filtering was used to cut out image noise (Cui and Qu, 2021). Cui and Qu (2021) need to discuss the results of the tests done on the research; glimpses of the experiment showed that the Pytorch framework can detect fire from most images on the internet.

A common trend around plant identification has developed in all research: dataset collection. There are only two ways to create a dataset: collect it from the field or get images from another source. Thompson and Brink, (2020) got their dataset from an iNaturalist platform. This dataset type is risky as the quality of the images taken is not controlled by the researcher. The image quality might affect plant identification accuracy, which happened to Thompson and Brink, (2020) with an accuracy of 66.15%. The researcher needed an equal number of protea images as the only dataset available was the one on the platform (Thompson and Brink, 2020). Even though the convolutional neural network (CNN) machine learning algorithm was applied to 41 different protea species, there was no equal and fair chance given to the CNN algorithm to be trained on protea identification. Even though there was no mobile application or computer-aided system created by Thompson and Brink (2020), plant identification methods give guidance on some of the challenges that are being faced in the field of plant identification using machine learning methods. Work done by Thompson and Brink (2020) gave researchers more reason to collect datasets in the field so that quality and quantity control of the dataset is required for machine learning algorithm training.

One of the key drivers of current research was to use a mobile plant identification application in a real-life situation where research is tested to solve real-life issues. Research conducted by Wang et al., (2020) falls into that category where a mobile application is created for students interested in plants. The research tested an m-learning solution to see if putting a learning experience on a smartphone can improve learners' learning capability and capacity. Even

though Wang et al., (2020) did not explain the type of smartphone OS used for research (Android, IOS, etc.), the kind of programming language used to build the mobile application, and how the mobile application was set up (system architecture). That does not take away the relevance of their study to the current one where a mobile application was tested against a real-life situation. The tests conducted by Wang et al., (2020) had to be set up expressly for this research as m-learning is not an approved concept yet. That is where the two types of research differ. With this research, the current situation of iGqirha sending umkhwetha to fetch traditional is an existing process that is allowed under the Traditional Health Practitioners Act 22 of 2007 of South Africa. This means that the research was conducted on the existing process that does not need approval (the process of sending umkhwetha to fetch traditional medicine), and the research can be adopted as the solution to the conventional healing community.

Plant images taken from their natural environment showing all the plant features (not just the leaves or flowers) are the basis of the dataset collected by Sun et al., (2017) in a research conducted to try and identify big forest trees. The researcher has yet to develop a mobile application for this research, but what is interesting is the ability to train a machine-learning model to identify a big tree. Collecting the images for this research took work, as the details needed by the machine learning algorithms might be compromised by the need to capture the complete picture of the tree.

Images were captured from different angles (some level to the ground, some facing up to the tree, and some standing on top of some unidentified object) (Sun et al., 2017). Capturing different ground levels helped me understand different angles to plant images that can be captured, improving the chances of a machine learning algorithm identifying a plant. Even though Sun et al., (2017) used a method of collecting data because research was done on big trees, a similar process of collecting data where different angles were used to capture plant images in their natural environment was adopted. Although most medicinal plants being researched do not grow to become big trees (only Mthoba grows to be a medium-sized tree), different angles were utilized to capture the medicinal plant images. The dataset for the research done by Sun et al., (2017) consists of 100 plant species containing 100 images of each species. The deep Residual Network (ResNets) method was used (Sun et al., 2017). According

to Sun et al., (2017), ResNets have shown overwhelming accuracy on large-scale images. ResNets have residual building blocks that can skip a layer of building blocks that are added during the training phase of the building block (Sun et al., 2017). Ignoring a layer that is not useful to the building block helps the residual building block only use functional layers in plant identification, improving the ability to identify a plant as more layers are added (Sun et al., 2017). 80% of the images collected were used to train the ResNets method and 20% for testing (Sun et al., 2017). Sun et al., (2017) achieved an identification accuracy of 91.78% in their research.

Mobile application creation for plant identification is a field in which some researchers are more interested. Prasandika et al., (2021) showed no difference in this case where a mobile application that identifies plants using plant leaves images uploaded from the smartphone is created. The developers used Python as the preferred programming language, and TensorFlow was used for deep learning methods using CNN (Prasandika et al., 2021). Amazon web services were used as a platform where the application was deployed using Amazon Elastic Compute Cloud (EC2) to deploy the application (Prasandika et al., 2021).

After analysis, the application would identify plant images, and identification results would show plant leaf names (Prasandika et al., 2021). Mobile applications would further display information about which plant leaf belongs to (Prasandika et al., 2021). Mobile applications do not have a way of dealing with identified leaves, and plant leaves remain unidentified. A gap in research by Prasandika et al., (2021) addressed in current research by making sure all the plant images put through the mobile application have some identification (from the plant names to unknown plants) where mobile application user can label/name an unknown plant (that new plant record of the unknown plant is stored on the database).

Table 2-1: Related works

Paper	Mobile Application	Software development language	Data set	Algorithm used	Evaluation matrix	Outcome
(Wang et al., 2020)	Yes	-	Eight plant species, 150 leave images per plant species	Deep neural networks (DNN)	Leaf identification	The test results were 95.25% identification accuracy.
(Barré et al., 2017)	Yes	-	184 tree species	Nearest neighbors	Tree identification	The researcher developed a mobile application called Leafsnap, which had about 1 million

						downloads after the paper was published.
(Prasandika et al., 2021)	Yes	Python	Ten leaf images per plant species	-	Leaf identification	The test results were 90% identification accuracy.
(Affouard et al., 2017)	Yes	-	10 000 plant species, 332 000 plant images	Convolutional Neural Network (CNN)	Flower & Leaf Identification	This is another successful mobile application called PI@ntNet which has been downloaded more than 3 million times in over 170 countries.
(Ganesan, 2022)	Yes	-	Flexible data sets	All the machine-learning models	Plant identification	This study shows the relationship between machine learning and mobile applications. The researcher does not create a mobile application but shows the relationship between machine learning algorithms and mobile applications.

Table 2-1 shows related work done in plant identification while creating a mobile application. There is a gap in research where most researchers only focus on plant leaves or flowers of the plant species. Datasets used in different research already conducted in similar work include data collected from other sources; there needs to be a clear gap in data collection from the field of the complete plant species. Not just flowers and leaves of plant species. An open-source programming language Python has been used to develop mobile applications, but there is also a gap in using open-source machine learning algorithms in conjunction with mobile application creation. You do find that some researchers have used open-source machine learning algorithms but have yet to create a mobile application to make practical use of the plant identification solution created (Zeroual et al., 2019). Open-source machine learning algorithms can develop frequently tested models depending on the model's platform (Ganesan, 2022). There are plenty of open-source development environments that were available during the implementation of this research, platforms such as Vercel, Budibase, Vs Code, Gitlab and others.

2.10 Summary

Chapter Two addressed two parts of this research: mobile application development and machine learning. Some of the mobile application methods discussed are waterfall, agile and

rapid. The methods discussed have pros and cons. The waterfall model is suitable for having well-structured application development steps as once you have passed a specific step in the development model you cannot go back to it, the same advantage can also be a disadvantage in situations where certain parts of the development processes have been missed, like requirements. Those missed parts would have to wait for future software releases to be corrected. Some methods discussed in the machine learning section are nearest neighbor, decision trees, support vector machines, and deep learning methods. These methods shed some light on researchers' models of similar plant identification paths. The next chapter covers methodology and design where the approach and design of the mobile application development are discussed.



University of Fort Hare
Together in Excellence

CHAPTER THREE

3. RESEARCH METHODOLOGY AND DESIGN

3.1 Introduction

An in-depth literature review was conducted to understand gaps in this field and how they can be addressed in this research. The literature review assisted in selecting the methodology used in this study and how to implement it. Chapter three explains the research methodology, and design. The design of mobile applications also included a feature of traditional healers and their trainees' ability to use mobile applications. The mobile application will consist of a plant identification process relevant to research in the conventional healing community. The methodical approach used while designing the mobile application and machine learning algorithms is discussed displaying how the research design was conducted.

3.2 Research Methodology

Addressing the research objectives, the study needed to be divided into two sections: mobile application development and machine learning. The mobile application addresses the development of a medicinal plant-detecting application whose primary end users are the traditional healers. The chapter covers issues such as coding languages, coding platforms, and mobile device operating systems are discussed and designed. The machine learning side addresses the ability to identify a plant species, involving data collection, analysis of data, and training machine learning models.

This nature of research requires careful consideration when selecting a research methodology as the research carries two steps (i.e., mobile application development and machine learning), which could be subjects on their own. The research methodology depends on the data collection required to address the research topic (Snyder, 2019). The type of data needed for this research is twofold: how the application will look and the collection of medicinal plant images from their natural environment.

Displayed in Figure 3-1 below is the design science methodology process followed in the research which was adopted from Snyder, 2019. System requirements were collected for a system design that will be the front end of the mobile application. The researcher collected system information as he is iGqirha and perfectly placed to understand the needs of a traditional healing community. The process between coding and system design was allowed to be a constant handshake (back and forth) as some requirements needed some adaptation. The CNN machine learning classification method was selected amongst other machine learning models that will be discussed.

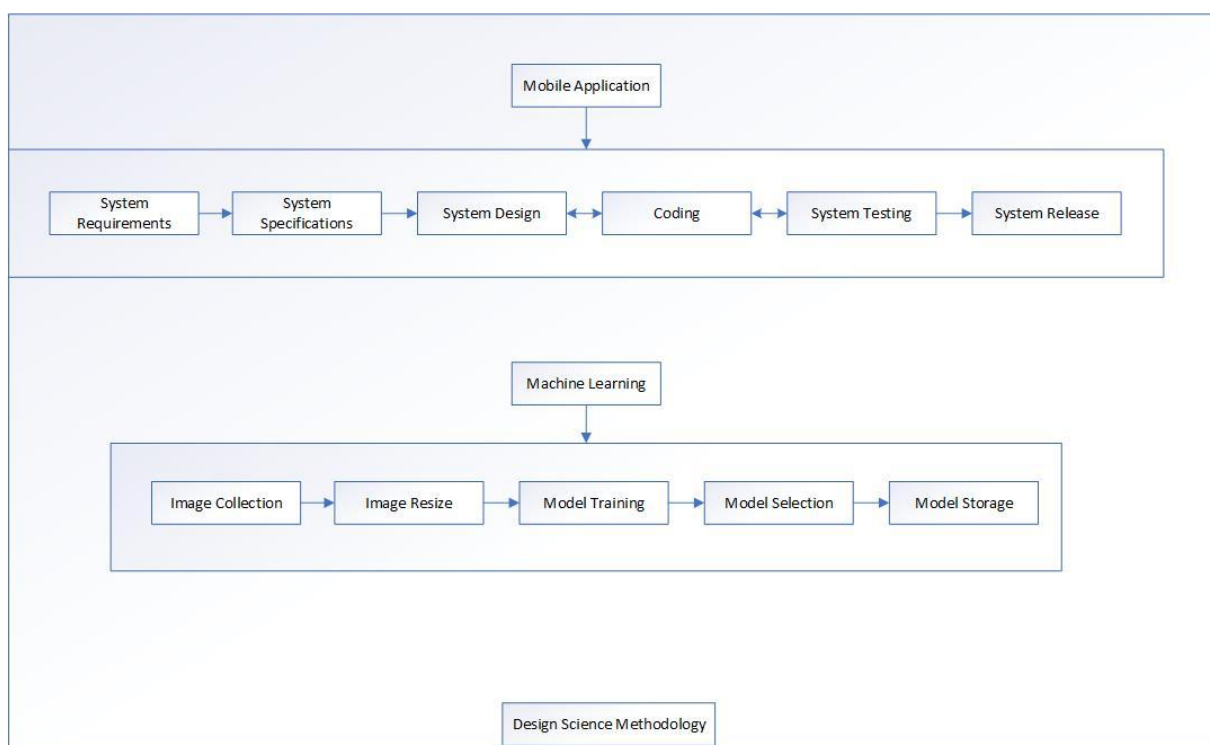


Figure 3-1: Design Science methodology processes

3.2.1 Mobile Application

The design science methodology process is discussed in this section, where rationale behind mobile application & machine learning design structure are covered.

3.2.1.1 System Requirements

The delivered product was pictured and worked backward to collect the system requirements. This meant asking, "What did we want to see at the end of the research?" The answer is

iGqirha, which uses a mobile application installed on a mobile device to identify medicinal plants. The answer already includes two critical aspects of system requirements: mobile application and mobile device.

Addressing mobile application requirements first, the answer to the question "What outcome did the research seek?" guided us through mobile application requirements at a high level. We now know there was an application to be created for iGqirha for plant identification. This means that before you can identify a plant, you need to see the plant; in technological terms, a database of these plants needs to be kept so that iGqirha can identify a plant species. Plant identity information in the database can be recalled for comparison. Each system user, i.e., needed to have a separate profile so that all the plants identified or to be identified could be kept separate from other users, which means that each system user needed to have login details on the system.



3.2.1.2 System specifications

System requirements guide specifications (Nikolakis, Maratos, and Makris, 2019). Coding for mobile applications has been established, the approach needs to start at the end to address the question, "Which mobile device operating systems will mobile applications run on?" Answering the question sheds some light on the coding language selected. The operating systems tested for mobile applications were Android and IOS. Programming languages that were looked at for system specification were C# and Python. These languages were selected for their flexibility in implementing mobile applications across platforms (Wang and Wu, 2019). Eventually, Python was chosen as the second language. As shown in the literature review chapter, Python has made strides in mobile application and machine learning for the plant identification field. The decision taken to use Python was based on the capabilities I have on Python as a programmer as I do not have exposure to C#. There is some evidence by researchers like Zeroual et al., (2019) who used Python successfully for mobile applications and machine learning.

3.2.1.3 System design

System specifications provided a clear direction for combining mobile application systems. Figure 3-2 below represents aspects of mobile applications. This is the first time the application has started to take shape, and some system processes are visually represented.

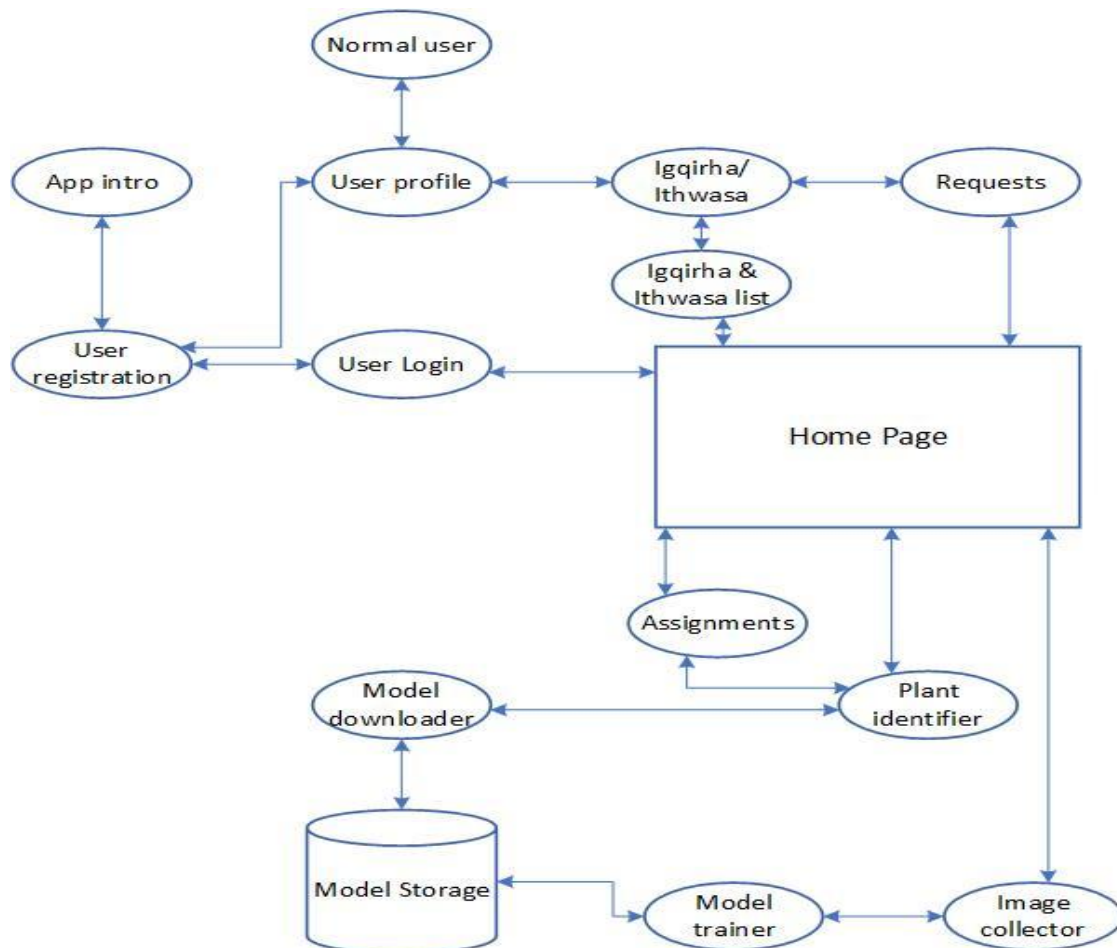
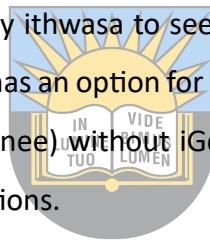


Figure 3-2: Mobile application system design

Figure 3-2 depicts one mobile application system design as the first point of contact for all aspiring application users. The introduction provides a summary of the application and exposes would-be users to some terms and conditions applicable to using the application. The following process after introduction is the user login step, where already registered users can log in. First-time users will have a chance to view and evaluate the application before registering. When registering, the user selects the type of user: iGqirha, ithwasa, or regular user. The difference between the three user profiles is crucial as the information you can share with iGqirha is more complex and of higher value than the information you can share with

ithwasa. The average user gets to see the medicinal plant database and identify plants; there are no assignments or additional information an average user can see about a plant. After the user is registered and logged in, a home page with options to go to different mobile application processes is displayed.

The first-time plant identifier is used, the machine learning model (to be discussed under machine learning later in the same chapter) is downloaded from a storage it was stored on and installed on the mobile application is required. When a machine learning model is installed, the application is ready for plant identification. The assignments process is also associated with the plant identifier process. The process works when iGqirha sends a task to ithwasa, requesting them to pick up a particular plant species. The request will be created and retrieved on the application under the process assignment. When ithwasa has received the assignment and is next to the plant suspected to be the plant required, the plant identifier process will have to be called upon by ithwasa to see if the plant they see is the plant they were sent to pick up. The application has an option for ithwasa to request to be under iGqirha. Traditionally, there is no ithwasa (trainee) without iGqirha (trainer); the application tried to capture this aspect of traditional relations.



University of Fort Hare

There is an ability to add new plant species to the database. The process is represented under the image collector where GPS coordinates or addresses of new plant species can be captured and represented on a field map under the process field map. Existing plants on trained machine learning models can be captured and entered in the database and represented on the field map when GPS coordinates are provided. Already existing plants captured on the image collector are added to the machine learning model installed into the device, increasing the training dataset and identification accuracy.

3.2.1.4 System development

System development of mobile applications involves design, as previously discussed. Python was chosen for this development instead of C# and the dotNet framework. The system was

developed in phases and tested, i.e., the login aspect would be designed and tested to see if it was working before moving to another mobile application process. Breaking down application processes to see if they work separately is in line with design science methodology (Muratovski, 2021).

3.2.1.5 System testing

When all mobile application processes were developed, they were tested separately (e.g., the login process) to ensure they worked. After confirming separate system processes described in the system design section above, all processes are integrated into the mobile application system, which is tested for its overall performance (i.e., from installation to identifying plant species).



3.2.1.6 System release

Mobile application systems can only be released after system testing has been confirmed. System testing was needed to ensure that the system operated as intended, which was for amaGqirha to be able to use this application for medicinal plant identification.

3.2.1.7 Machine learning

The section displays how design science methodology assisted in addressing the machine learning aspect. Research methodology helps shape the data collection, influencing the research outcome (Vihar, 2021). Discussed below are different aspects of the machine learning methodology for this research:

3.2.1.8 Data collection

The data collected were images of medicinal plants presented in a machine-learning algorithm. The overharvesting of medicinal plants is becoming a problem in conserving some of these plant species (Raimondo, 2018). IGqirha needed to avoid adding to this problem. The research had to find a way to conserve medicinal plants. A decision was made to take plant images from their natural habitat. Plant images focused on the complete plant structure instead of just plant leaves, as most of the literature covered in chapter two.

Data collection was divided into Morning, Middy, and Afternoon. The decision was to cover different light shades and weather conditions as much as possible. Adding to other parts of the day, plant images were taken at three different positions: 90°, 60°, and 30°. There were 600 plant images taken per different position, which means that there were 1800 plant images taken for the morning session of the day chosen. The same method was repeated throughout the day, meaning that 5400 plant images were taken per plant species by the end of the day. Figure 3-3 below shows how the datasets were arranged as captured.

Table 3-1: Data set organization

A	B	C	D	E	F	G
1 Name of data capture:	Onke Simandla					
2 Date:		16-Mar-22	20-Mar-22			
3 Weather:	Minimum - Maximum	13°C - 23°C	17°C - 25°C			
4	Cloud Cover	Partly cloudy	Sunny			
5 Coordinates	GPS Latitude Ref	South latitude	South latitude			
6	GPS Latitude	Shown on the table below	Shown on the table below			
7	GPS Longitude Ref	East longitude	East longitude			
8	GPS Longitude	Shown on the table below	Shown on the table below			
9						
10 Plant Name	Photo ID	Coordinate	Angle	Date	Time of the day	Weather Discription
11 Cima Mlilo	20220316_112124.jpg	32°38'10.4308"S 26°52'36.3054"E	30°	3/16/22, 11:21:24 AM	Morning	Overcast
12 Cima Mlilo	20220316_112124(0).jpg	32°38'10.4308"S 26°52'36.3054"E	30°	3/16/22, 11:21:24 AM	Morning	Overcast
13 Cima Mlilo	20220316_112124(1).jpg	32°38'10.4308"S 26°52'36.3054"E	30°	3/16/22, 11:21:24 AM	Morning	Overcast
14 Cima Mlilo	20220316_112125.jpg	32°38'10.4308"S 26°52'36.3054"E	30°	3/16/22, 11:21:25 AM	Morning	Overcast
15 Cima Mlilo	20220316_112125(0).jpg	32°38'10.4308"S 26°52'36.3054"E	30°	3/16/22, 11:21:25 AM	Morning	Overcast
16 Cima Mlilo	20220316_112125(1).jpg	32°38'10.4308"S 26°52'36.3054"E	30°	3/16/22, 11:21:25 AM	Morning	Overcast
17 Cima Mlilo	20220316_112126.jpg	32°38'10.4308"S 26°52'36.3054"E	30°	3/16/22, 11:21:26 AM	Morning	Overcast
18 Cima Mlilo	20220316_112126(0).jpg	32°38'10.4308"S 26°52'36.3054"E	30°	3/16/22, 11:21:27 AM	Morning	Overcast
19 Cima Mlilo	20220316_112126(1).jpg	32°38'10.4308"S 26°52'36.3054"E	30°	3/16/22, 11:21:27 AM	Morning	Overcast
20 Cima Mlilo	20220316_112127.jpg	32°38'10.4308"S 26°52'36.3054"E	30°	3/16/22, 11:21:29 AM	Morning	Overcast
21 Cima Mlilo	20220316_112128.jpg	32°38'10.4308"S 26°52'36.3054"E	30°	3/16/22, 11:21:29 AM	Morning	Overcast
22 Cima Mlilo	20220316_112130.jpg	32°38'10.4308"S 26°52'36.3054"E	30°	3/16/22, 11:21:30 AM	Morning	Overcast

The methods that were used to collect the plant images were as follows:

Used a Samsung A9 smartphone camera to take plant images with following specification:

- Main camera Quad
- 24 MP, f/1.7, 27mm (wide), 1/2.8", 0.9µm, PDAF

- 10 MP f/2.4, 52mm (telephoto), 1/3.9", 1.0 μ m, AF, 2x optical zoom
- 8 MP, f/2.4, 12mm (ultrawide), 1/4.0", 1.12 μ m
- 5 MP, f/2.2, (depth)

Took images on three different day periods (morning, midday, and afternoon). Morning time was from 08:00 am to 11:59 am CAT. Midday time was from 12:00 pm to 2:59 pm CAT. Afternoon time was from 3:00 pm to 5:59 pm CAT. The images were captured at any time between the stipulated periods for the different periods of the day.

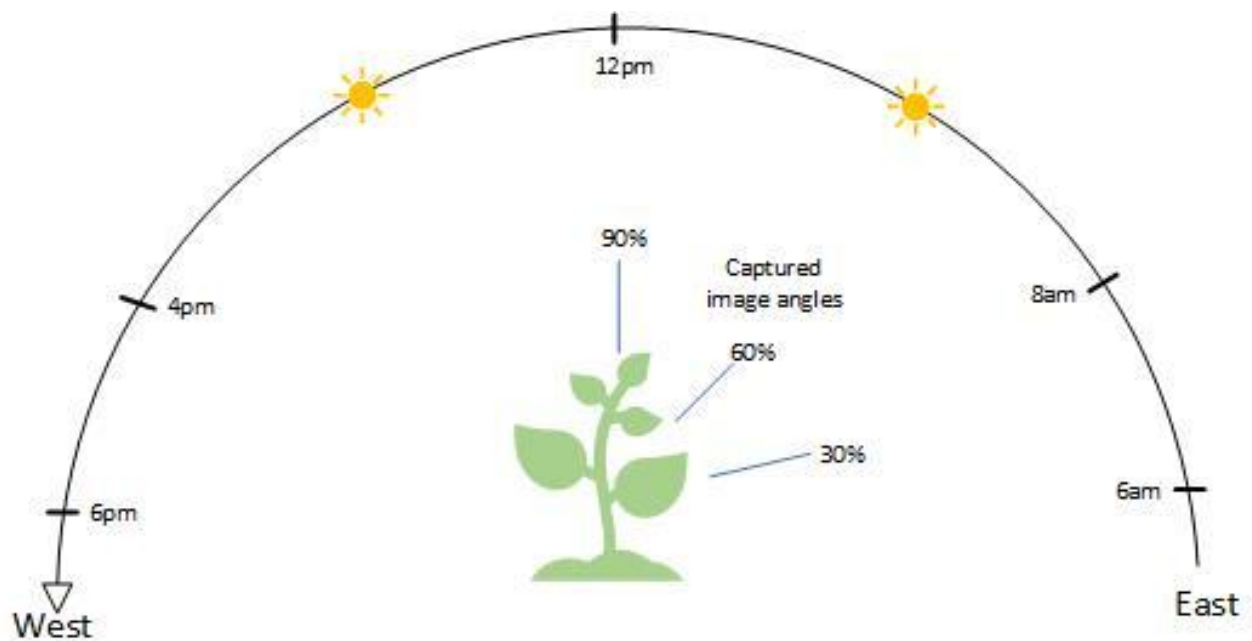


Figure 3-3: Image capturing and time of the day during data set capturing

- Five different medicinal plants, Mhlonyane, Hlaka-Hlakane, Mthoba, Cima Mlilo, and Mpepho, were captured on images.
- For each period (morning, midday, and afternoon), the images were captured from 3 approximate angles: 90 degrees, 60 degrees, and 30 degrees. There were 600 pictures of each approximate angle taken.
- The weather conditions during the image capture were also recorded.

3.2.1.9 Data processing

When collected, research data was taken through a process of preparation for model training. The process involved ensuring the image size of all the collected data was the same. Image resize was applied uniformly across all photos. The data processing method was crucial before the machine learning model was trained. Data processing steps determined the quality of data used to train the model. Some research has been done where background filtering was applied to the dataset, ensuring data in the machine learning model is only of required plant species and not any background noise (Patel and Patel, 2019). Plant filter solutions that reduced background noise in a picture were not used as data was collected from its natural habitat, so it was crucial to see if the model could identify the plant amid its natural surroundings. In traditional ways of plant identification, a plant is not picked up from its natural habitat, and surrounding plants are also part of the identification method as sometimes, when someone is defining a particular plant, they will use surrounding plants like "the plant you are looking for likes to be around the grass in open spaces in the forest ."Figure 3-4 below shows how data processing fits into machine learning methodology. Data processing is only represented by Image Resize.

University of Fort Hare
Together in Excellence

The image resize process was planned to follow the steps shown in Figure 3-4 below. Data collected needs to be arranged into different plant species and segments of the day. When all collected data has been arranged, the source of collected plant images was selected using the image resizer. A destination where resized images were placed was specified as the same destination from which the machine learning model got the training data. A new image size (320 x 240 pixels) was selected for images to have a uniform size.

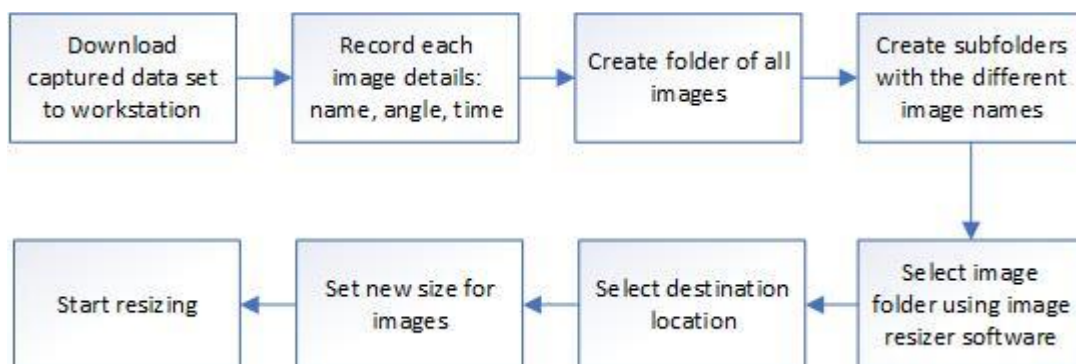


Figure 3-4: Image resize methodology

3.2.1.10 Models training

Resized images were used to train machine learning models. The resized images were sorted into training, validation, and test data. The model was initialized; image data was then loaded into the model to start training. When training was done, the trained model was saved. TensorFlow and PyTorch were the models that were trained for testing between them. The model methodology shown in Figure 3-5 below was applied separately to both selected machine learning models.

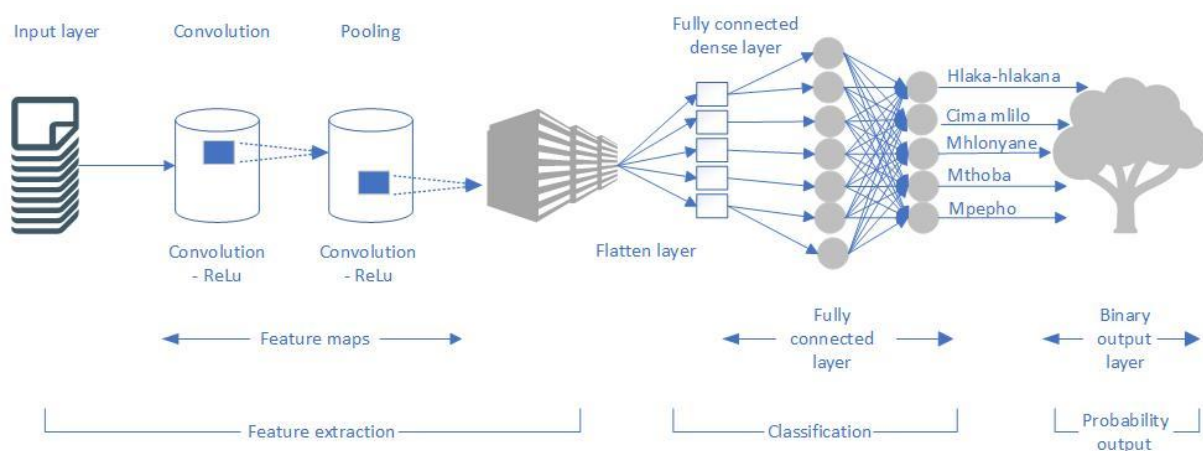


Figure 3-5: Machine learning model training

University of Fort Hare
Together in Excellence

3.2.1.11 Python Development Libraries

Selected models (Tensorflow and Pytorch) were tested for plant identification, and a model that achieved higher plant identification was chosen for the mobile application. Identification results were determined from a test covered in detail in the results chapter. The selected models used CNN for feature and plant identification, which was shown in the literature chapter as better performing machine learning algorithm in image identification.

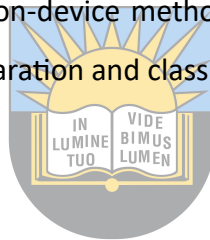
3.2.1.12 Model storage

The better performing model that was loaded to the mobile application and stored with access to the mobile application. The chosen machine learning model has a better identification rate. There are two possible approaches to be used in the storage of trained machine learning model discussed below:

3.2.1.13 On-device

The approach involved saving a trained model and running classification on the device.

- The advantage is that no network access is required, so that the app can work offline.
- The disadvantage is that the on-device method is resource-heavy in storage for the trained model. All image preparation and classification happen locally.



3.2.1.14 Cloud-based

University of Fort Hare
Together in Excellence

- An approach involves using a cloud-based server to host trained models running image classification and only returning results.
- The advantage is most of the heavy work and processing is not handled by the user's device but by a hosted server.
- The disadvantage is that mandatory internet and server access must be paid for under the hosting service.

The selected option was the one-device method, as there was a considerable possibility of mobile application users in settings with limited or no network when using the plant identification process on mobile applications.

3.3 Testing Mobile Application

A mobile application created for medicinal plant identification was to be tested for identification accuracy with different plant samples from those used for training machine learning models. Real-time testing from captured snapshots was to be used for experimentation purposes. The performance of the mobile application when it performed the application tasks like plant identification was tested. Central processing unit (CPU) and memory utilization of that mobile application process were measured and tested against other plant identification mobile applications like Pl@ntNet on similar processes.

3.4 Results

Findings from mobile application experimentation were to be presented, displaying accuracy achieved by mobile application. Mobile applications were tested for performance ability by measuring specific mobile application processes' central processing unit (CPU) and memory utilization. A measurement of how much CPU and memory resources were utilized by these processes was covered:

- Application launch
- Plant identification launch
- Plant identification

University of Fort Hare
Together in Excellence

A comparison of CPU and memory performance with similar mobile applications was to be made and presented.

3.5 Summary

Chapter three revealed the design of the mobile application, giving a visual aid to the research's aim. Design decisions, such as hosting the database on-device, were discussed. A summary of methodology and design was also covered. An analysis of the application's usability was to be covered to measure mobile application performance. Implementation of

the method and design of the mobile application and machine learning using design science methodology is described in chapter four.



University of Fort Hare
Together in Excellence

CHAPTER FOUR

4. SYSTEM DESIGN AND IMPLEMENTATION

4.1 Introduction

Chapter four describes the implementation of research by taking a closer look at how the mobile application was developed and how machine learning models were trained and tested. The mobile application development approach is explained, as are the machine learning concepts behind the trained models. The design science research method was used when the mobile application was developed, the use of this system development principle was influenced by the flexible nature of agile as discussed by Senarath, 2021. There was a need for designing, implementing, testing and go back to the designing or implementation phase of the system design method depending on the challenges picked up on the mobile application development phase. The design science method assisted in making the development process of the mobile application run smoothly in the flexible and unstable environment as Senarath, 2021 discussed in his research about the agile methodology.

4.2 Development Environment & System Architecture

The development tools and the environment utilised during the mobile application system architecture development are discussed in this section.

4.2.1 Mobile Application Development

The mobile application was developed on a workstation (Figure 4.2) with Android Studio IDE installed. The technical specifications of the workstation are listed on figure 4-1. Android Studio was utilized to code the mobile application, which included coding in Python, testing, and debugging the code, which was then saved in the workstation as an APK file.

4.2.2 Model Development Using Deep Learning

The model was developed using deep learning CNN using TensorFlow and PyTorch libraries. Preloaded training data was processed using the image resizer within the Spider IDE. A program was written on the image resizer using Python programming language, where height and width variables were defined. The shape attribute was used to assist in obtaining the image height and width required. An image variable was defined and pointed to the folder that has the images, the shape attribute was then used to define the width and the height as `image.shape(0)` multiplied by a defined scale and the total number converted to an integer. The width and the height of the images was set at 320x240 pixels for all images. Resized images were ingested into the model for training, and after training, the model was uploaded to a Firebase database accessible online.

4.2.3 Model Deployment

The Firebase database is uploaded to the mobile application by initiating a machine learning model loader on the application during initial installation. The database carried the trained machine learning model that was used in the plant identification process of the mobile application. The Firebase database assisted with transportation of the trained machine model from the workstation to the mobile application. Future enhancements of the database are made easy to implement via the firebase database, making it easy to release database related updates to the mobile application users wherever they are where there is cellular network

coverage. The plant identification process was ready when the model was loaded onto the mobile application. The mobile application system was put together using components displayed in Figure 4-2 below. The application was coded on a Dell Latitude 5480 workstation where Spyder IDE version 5.4.0 was loaded. The specifications for the workstation are as follows:

Device specifications

Latitude 5480

Device name	NO-F1965-L
Processor	Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz 2.90 GHz
Installed RAM	16,0 GB (15,9 GB usable)
Device ID	9475b18f-c58e-40fa-9afe-d47259e1e390
Product ID	00329-10330-00000-AA221
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Figure 4-1: Workstation specification



The mobile application was built using Python programming on Android Studio (version: Dolphin 2021 .3.1 Patch 1). Android Studio is a programming platform used to build mobile applications by coding. The mobile application was deployed onto a Samsung A9 Android device, and regular testing was done after completing parts of the development schedule. The Android application layer hosts mobile applications built for research. In the application framework layer, components like location manager, notification manager, resource manager, and activity manager are some of the classes used in Android applications to build mobile applications (Mesia Dhas and Shiny Angel, 2022).

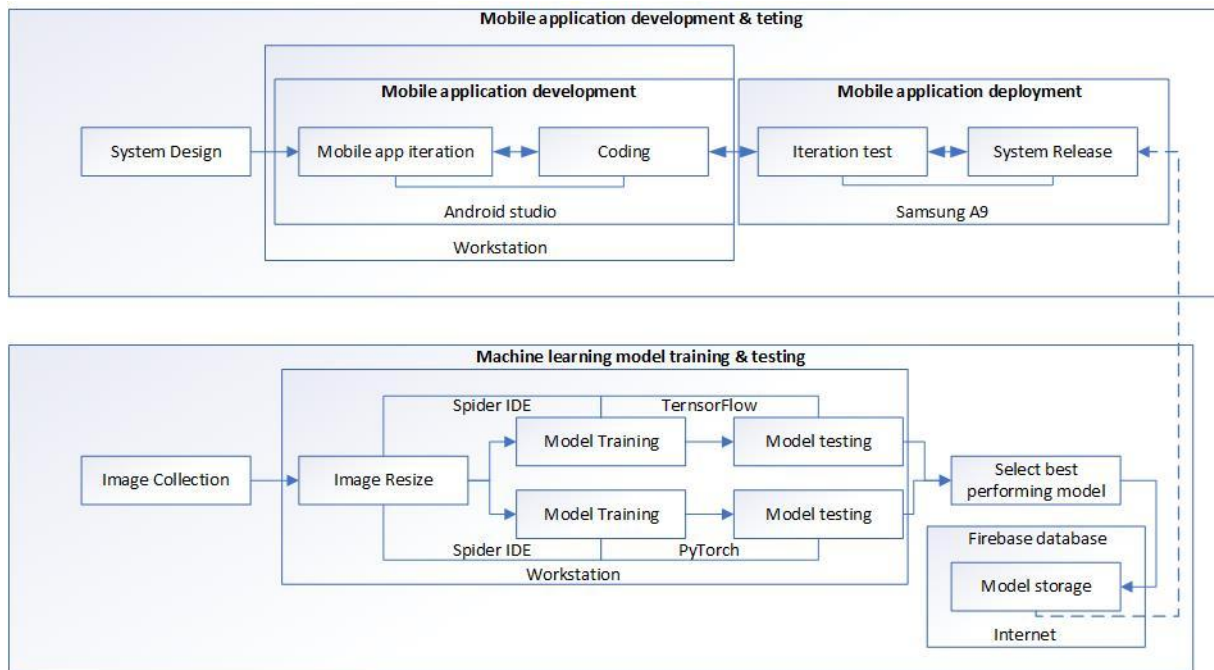


Figure 4-2: System architecture



4.3 Implementation Process

The development of mobile applications was scheduled into manageable sections for ease of development (Aldayel and Alnafjan, 2017). The schedule was divided into user registration and login, model training and model upload, and medicinal plant identifier.

4.4 Coding Standards and Guidelines

The section was vital in coding as it set rules of how coding was done by the coding team (Elliott, 2018). There were coding standards and guidelines that were followed during the application development and machine learning model training. Variable naming was the first coding standard that was followed, where self-explainable variables had to be created (Dalla Palma *et al.*, 2020). Self-explainable variables assisted in having an easy to understand code (Dalla Palma *et al.*, 2020). Commenting on certain parts of the code was done assisting in explaining the code, making it easier to follow the code (Hu *et al.*, 2018). For the code to be

easy to read similar indentation was used for functions and classes (Lu *et al.*, 2020). The maximum line length was set to 79 characters (Hu *et al.*, 2018). New packages were imported on separate lines ensuring easy understanding and identification of the packages. (Hu *et al.*, 2018). There were space requirements that were implemented, where there was a space between variables, binary operators, comparisons, assignment, augmented assignment, and booleans (Rivers and Koedinger, 2017). The coding standards assisted in creating readable and presentable code for mobile applications.

4.5 User Interface Design

The front end section of the mobile application and the processes that users will interact with are discussed in this section.

4.5.1 User registration and login



The mobile application was built to introduce users to the application's uses. After the introduction, the user is guided to a login screen where first-time users must register (Figure 4-4), and already-registered users can log in (Figure 4-6). The methodology followed for user registration development is depicted in Figure 4-3 below.

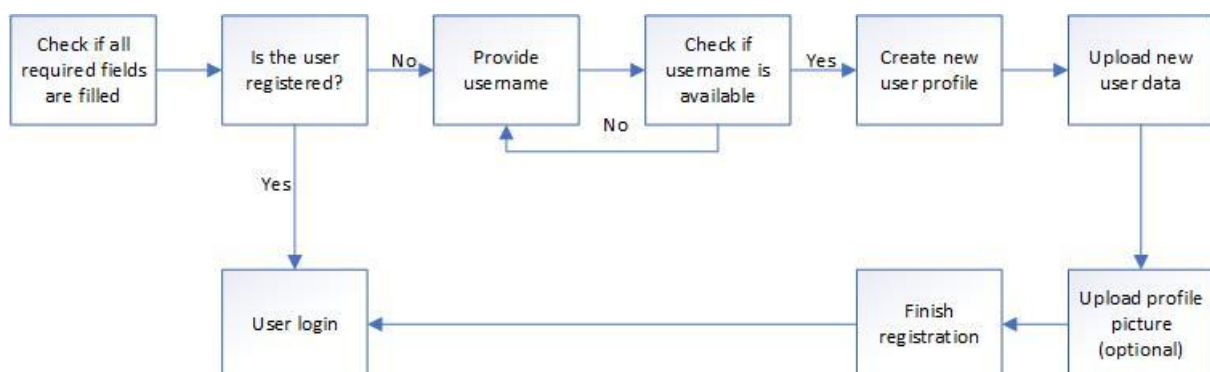


Figure 4-3: User registration and login architecture

Part of the coding for user registration is displayed below:

```
//Create a new user profile
mAuth.createUserWithEmailAndPassword(getEmail(), getPassword())

//Upload new user data
FirebaseUser fbUser = mAuth.getCurrentUser();

Map<String, Object> user = new HashMap<>();
user.put("account_type", binding.accountTypeSpinner.getValue());
user.put("username", getEmail());
user.put("first_name", binding.nameEditText.getValue().trim());
user.put("last_name", binding.surnameEditText.getValue().trim());
user.put("dob", dob);
user.put("gender", binding.genderTypeSpinner.getValue());
user.put("province", binding.provinceTypeSpinner.getValue());
user.put("hide_profile", false);

db.collection("users").document(fbUser.getUid()).set(user)
```

Snippet 4-1: Create a new user profile

The above code displays *the user.put()* command is used for the user to input details required for the user registration process. These commands are used after a username and password are created for a user profile using the following command:

```
mAuth.createUserWithEmailAndPassword(getEmail(), getPassword())
```

Mobile applications enable users to upload a profile picture; the command for profile picture upload is as follows:

```
Upload profile picture
final StorageReference ref =
storage.getReference().child("users").child(mAuth.getCurrentUser().getUid()).child("profile_pic.jpg");
UploadTask = ref.putFile(profilePicUri);

Finish Registration
startActivity(new Intent(RegisterActivity.this, MainActivity.class));
finish();
```

Snippet 4-2: Upload profile picture

Command *UploadTask = ref.putFile(profilePicUri)* is used to upload the selected profile picture, which will be displayed the next time the user logs in to the application. Figures 4-4 display user and registration screenshots from the mobile application.

The image displays two versions of a mobile application's registration screen. Both screens have a purple header with the text 'Register New Account'. The left screen features a blue circular profile icon placeholder at the top. Below it is a dropdown menu for 'Select Account Type'. The form includes fields for 'Email', 'Password', 'Confirm Password', 'First Name', 'Last Name', and 'Date of Birth'. The right screen shows a similar form but with additional fields for 'First Name', 'Last Name', 'Date of Birth', 'Gender', and 'Province'. A 'SET' button is next to the 'Date of Birth' field. At the bottom of the right screen is a purple 'SUBMIT REGISTRATION' button. Both screens show a status bar at the top with the time 15:11 and battery level 32%.

Figure 4-4: User registration showing login details (email and password)

The user selects the type of user they are (Sangoma, iThwasa, ordinary user) from a drop-down list shown in Figure 4-5 below. The reason behind these types of users is that in the traditional healing community, there are the following types of categories: iGqirha/Sangoma (this is a traditionally qualified healer) and iThwasa (a traditional healer trainee). The idea behind ordinary users is to enable people who are not traditional healers interested in medicinal plants to view plant databases of the mobile application.

Figure 4-5: Select user type

When a user is registered, the screenshot shown in Figure 4-7 below will be the welcome screen where the user will need to log in before proceeding to the contents of the mobile application. The code for checking if a user is already registered is displayed below:



```
//Check if the username is available
db.collection("users")
  .whereEqualTo("username", getEmail())
  .get()
  .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
    @Override
    public void onComplete(@NonNull Task<QuerySnapshot> task) {
      if (task.isSuccessful()) {
        if(task.getResult().size()>0){
          showToast("Email/Username already taken!");
          loadingDialog.dismiss();
        }else{
          loadingDialog.setMessage("Creating Account...");
          registerNewUser();
        }
      } else {
        showToast("Something went wrong...");
        loadingDialog.dismiss();
      }
    }
  });
```

Snippet 4-3: Check if username is available

A database `db.collection("users")` is called upon where `onComplete` is created, which has a statement that helps check whether the username presented belongs to an already registered user. Figure 4-6 for a registered user is displayed below:

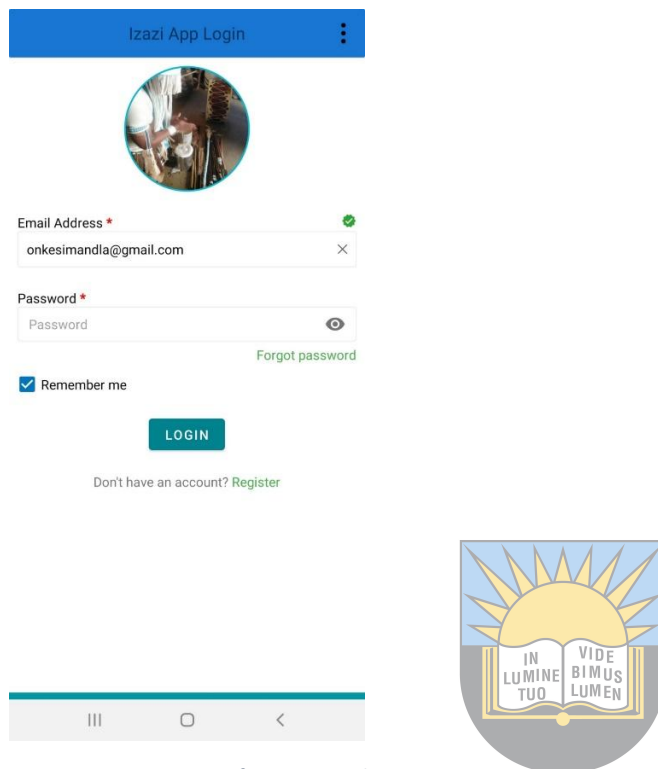


Figure 4-6: Login screen for registered user

Throughout the registration process, there were some asterisks which stand for mandatory fields displayed in some fields where the user must fill in the required information. The code to check if the user has captured all necessary information is displayed below:

```
//Check if all required fields have been filled
private boolean allFieldsComplete() {
    if(binding.accountTypeSpinner.getSpinner().getSelectedItemPosition()==0){
        showToast("Please select account type!");
        return false;
    }

    if(!isEmailValid()){
        showToast("Email/Username is invalid!");
        return false;
    }

    if(!isPasswordValid()){
        return false;
    }

    if(binding.nameEditText.getValue().trim().equals("")){
        showToast("First name cannot be empty!");
    }
}
```

```

    return false;
}

if(binding.surnameEditText.getValue().trim().equals("")){
    showToast("Last name cannot be empty!");
    return false;
}

if(binding.dobEditText.getValue().trim().equals("")){
    showToast("Date of birth not set!");
    return false;
}

if(binding.genderTypeSpinner.getSpinner().getSelectedItemPosition()==0){
    showToast("Please select gender!");
    return false;
}

if(binding.provinceTypeSpinner.getSpinner().getSelectedItemPosition()==0){
    showToast("Please select province!");
    return false;
}

return true;
}

```

Snippet 4-4: Check if required fields are filled



An "if" statement is used to check if all mandatory fields are filled. If the "if" statement is false, a command `showToast ("Please select gender!")` will be executed. An error message "Please select gender!" will be displayed, guiding the user to select the necessary field previously not selected. The Logon process for the mobile application Izazi also includes using fingerprints to log on to the application, depending on a smartphone with fingerprint capability. The fingerprint option was added as an extra layer of security when logging on to the mobile application, as displayed in Figure 4-7 below.

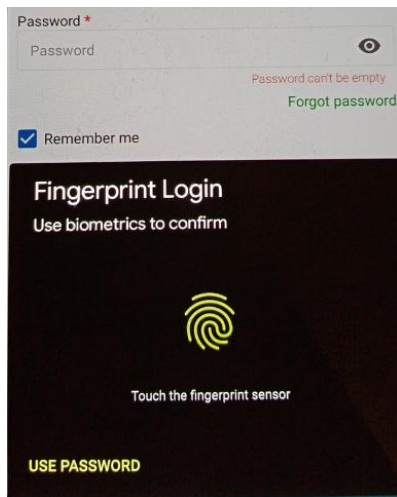


Figure 4-7: Fingerprint login

The code for checking if the smartphone has biometric authentication is shown below:

```
//Check if biometric authentication is available on the phone
BiometricManager biometricManager = BiometricManager.from(getActivity());
switch (biometricManager.canAuthenticate()) {
    case BiometricManager.BIOMETRIC_SUCCESS:
        Log.d("MY_APP_TAG", "App can authenticate using biometrics.");
}
```

Snippet 4-5: Check if biometric authentication is available

The above code is run to check if the smartphone has fingerprint capability; if the capability is there, the below code is executed:

```
//Fetch fingerprint login user setting
fingerprintLogin = Prefs.getBoolean("fingerprint_login",false);

if(fingerprintLogin){//If true login using fingerprint
    binding.fingerprintCard.setVisibility(View.VISIBLE);
    binding.textViewFingerprint.setVisibility(View.VISIBLE);
}
```

Snippet 4-6: Fetch user fingerprint login setting

Another part of the mobile application was added where the application can check for internet access on the smartphone. If there is no internet access, the mobile application goes offline. Even with Offline mode, it is possible with the login process and crucial as some medicinal plants are found in remote areas without an internet connection. The code below shows how Python checks internet connectivity:


```
//Function to check if internet access is available on the device
public static boolean isInternetAvailable(Context context) {

    //Get an instance of ConnectivityManager
    ConnectivityManager cm
    (ConnectivityManager)context.getSystemService(Context.CONNECTIVITY_SERVICE);

    //Get an instance of NetworkInfo
    NetworkInfo activeNetwork = cm.getActiveNetworkInfo();

    //Return true if internet access is available and false if not,
    return activeNetwork != null && activeNetwork.isConnectedOrConnecting();
}
```

Snippet 4-7: Check device internet availability

Function `isInternetAvailable(Context context)` is within a public class so it can be accessed anywhere within the program. Before a code that needs to establish an internet connection is run, the `isInternetAvailable(Context context)` function is always executed to check for connectivity. The login process is also called this internet connectivity function. Firstly, the code was set up to check if the user can log in via the Firebase database, where user profile information is kept. If a user cannot log in via Firebase, the code was structured to check if the user has logged into a mobile device at least once. It was crucial to check as users' login details get downloaded on their smartphones when they login to the Izazi mobile application for the first time. User credentials on smartphones are then used to authenticate users during the loss of internet connectivity.

4.5.1.1 Plant identifier

When a user successfully logged onto the mobile application, they were directed to a menu which has Plant identifier as one of the options in the menu where a user can identify plants; see Figure 4-8.

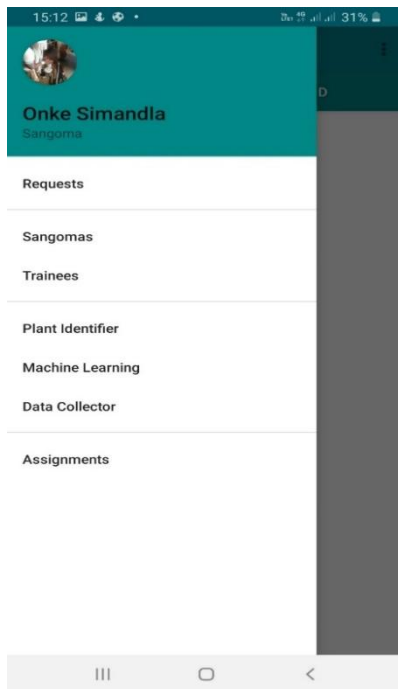


Figure 4-8: Mobile application menu

The plant identifier process needed a machine learning model to do plant identification, which will be covered under the machine learning section. Steps taken on the mobile application before asking the plant to be identified are covered. The code behind plant image identification is discussed below. Plant identification is divided into two parts: identify plants from a gallery or camera. Plants identified from the gallery are from plant images saved on the gallery smartphone, and identification made from the camera are plant images taken from plant locations. The code shown below is for pictures taken from a gallery of smartphones.

```
//Get the model file and model labels file
File model = new File(getExternalFilesDir("ML"), Prefs.getString("model", ""));
File label = new File(getExternalFilesDir("ML"), LABEL_PATH);

//Initialize Image Classifier object
classifier = TensorFlowImageClassifier.create(model, label, INPUT_SIZE, QUANT);

//Get an image from the gallery
ImagePicker.with(IdentifierActivity.this)
    .cropSquare() //Crop image(Optional), Check Customization for more option
    .galleryOnly()
    .compress(1024) //Final image size will be less than 1 MB(Optional)
    .maxResultSize(320, 240) //Final image resolution will be less than 1080 x 1080(Optional)
```

```
.start();
```

Snippet 4-8: Get model files and labels

The plant image was taken from the smartphone gallery using the command `ImagePicker.with(IdentifierActivity.this)` and the command `.galleryOnly()`, which points the plant identification process to the smartphone gallery. Plant images were cropped for space purposes on smartphones, as there could be plenty of plant images that need to be identified locally on the device. Command `.cropSquare()` gave code the ability to instruct the plant identifier process to crop an image to a size specified using command `.maxResultSize(320,240)`, which takes just 320 x 240 pixels of the image from the gallery for identification. The following code is for plant identification done via smartphone camera:

```
//Get an image from the camera
ImagePicker.with(IdentifierActivity.this)
.cropSquare() //Crop image(Optional), Check Customization for more option
.compress(1024) //Final image size will be less than 1 MB(Optional)
.maxResultSize(320, 240) //Final image resolution will be less than 1080 x 1080(Optional)
.start();
```



University of Fort Hare
Together in Excellence

```
//Get image location data
profilePicUri = data.getData();
try {
    //Get the image as a bitmap
    Bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), profilePicUri);
    //Display the image to the user
    binding.imagePlant.setImageBitmap(bitmap);

    //Create a resized bitmap
    bitmap = Bitmap.createScaledBitmap(bitmap, INPUT_SIZE, INPUT_SIZE, false);

    //Use a prepared classifier to identify an image and return the results
    final List<Classifier.Recognition> results = classifier.recognizeImage(bitmap);

    //Clear previous results
    itemFastItemAdapter.clear();
```

```

//Show user number of results
binding.textViewResults.setText("Results (" +String.valueOf(results.size())+"");

//Go through all the results
for (Classifier. Recognition s:results) {
    //Prepare the result item
    ResultsAdapterItem itemAdapter2 = new ResultsAdapterItem(this);
    //Get confidence in identification as a percentage
    int confidence = (int)(s.getConfidence()*100);
    //Display results to a user
    itemAdapter2.setResult(capitalizeString(s.getTitle()));
    itemAdapter2.setConfidence(String.valueOf(confidence)+"%");
    itemFastItemAdapter.add(itemAdapter2);
}

} catch (IOException e) {
    e.printStackTrace();
}

```

Snippet 4-9: Get image from camera



The difference between code for taking an image from a gallery and a picture from a camera is the command `.galleryOnly()` in a section where code instructs mobile devices where to get an image. The command was not included in the section where a plant image was supposed to be taken from a camera, making it possible for the plant identifier process to use another available method for getting images i.e. the camera.

The plant's geographical location was also captured using the command `Bitmap bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), profilePicUri);` where plant location was captured for future display on the map shown on the application.

As shown in Figure 4-9 below, you can select one of two options to access all the plant identifier options discussed above with code.

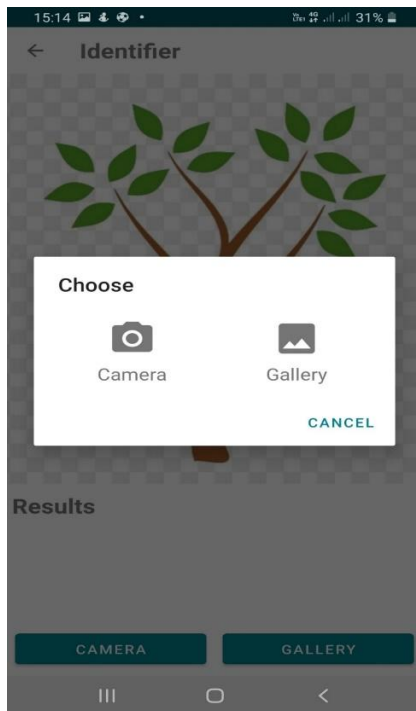
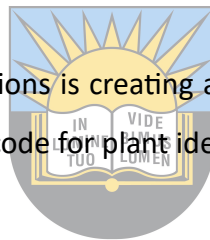


Figure 4-9: Plant identifier options

Another capability of mobile applications is creating assignments. The section below covers how assignments were coded. More code for plant identifiers is in Appendix A.



4.5.1.2 Assignments

University of Fort Hare
Together in Excellence

This section was one of the crucial sections of research as it helps adapt mobile applications to the traditional healing community. A mobile application was set up to send an assignment to iThwasa (trainee user), and a user was tasked to pick up a particular medicinal plant. Figure 4-10 below depicts how the assignment looks on mobile applications:

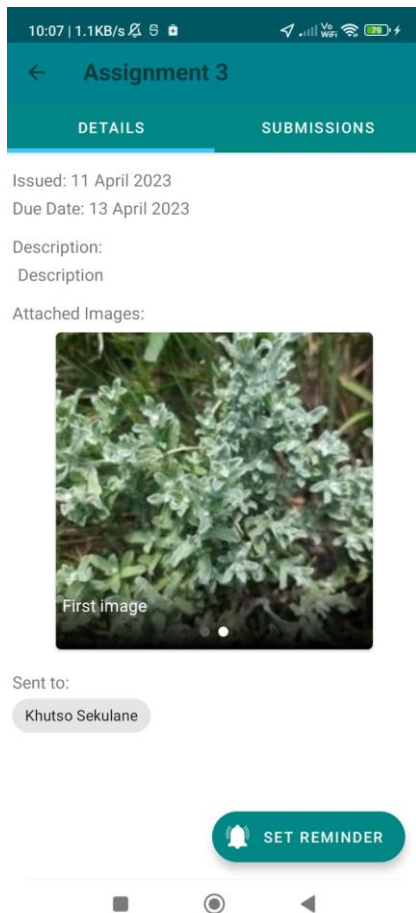


Figure 4-10: iThwasa assignment



University of Fort Hare

Figure 4-10 depicts the following details: the date the assignment was issued, the due date of the assignment, a description of the assignment where the user initiated, a plant image also shown for users who might not know what the plant looks like. The assignment also has a set reminder option; the reminder was coded to be created on a smartphone calendar instead of done on a mobile application. The reason behind the creation of reminders outside of mobile applications comes from González-Pérez et al., (2022) where frequently occurring events that need to be monitored on a device or mobile application can be monitored by creating tasks outside of a mobile application. The set alarm tab saves mobile application resources by setting reminders for users to pick up a particular medicinal plant on the smartphone. The code for creating assignment reminders opens the device calendar; it is up to the user to fill in the necessary information for the remainder. The code below shows how the set remainder process works on the mobile application:

```
//Listener for "Set Reminder" button press
```

```

binding.reminderFab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        // Create an intent to add a new calendar event
        Intent intent = new Intent(Intent.ACTION_INSERT)
            .setData(CalendarContract.Events.CONTENT_URI)
            .putExtra(CalendarContract.Events.TITLE, title+" (Submission)")
            .putExtra(CalendarContract.Events.DESRIPTION,
binding.textViewDescription.getText().toString())
            .putExtra(CalendarContract.Events.EVENT_LOCATION, "In-app submission.")
            .putExtra(CalendarContract.Events.ALL_DAY, true)
            .putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME, dueDate);

        // Start the calendar app to add the event
        startActivity(intent);

    }
});

```



Snippet 4-10: Set reminder listener

University of Fort Hare
Together in Excellence

The codes discussed above made mobile application functional from the user login process until the user needs to identify a plant image. The section on mobile applications only catered to the functionality of mobile applications, as discussed. A part where a mobile application must identify a plant from an input image can only be possible by including machine learning in a mobile application. The section below discusses the plant identification process, in which a mobile application was needed to identify the plant image taken from the gallery or the camera.

4.5.2 Model training & Model upload (Machine Learning)

The machine learning section continues from the mobile application section, as the machine learning part was developed after the mobile application. The training of the two selected models (TensorFlow lite, Pytorch) will be discussed. The selection of models used for mobile applications will also be covered. Some tests between machine learning models were used to determine models suitable for the application.

Two selected machine learning methods used deep learning algorithms. Deep learning is an algorithm under unsupervised learning where a machine can learn its patterns in an unlabeled dataset (Ganesan, 2022). The approach is based on the design science methodology followed in this research, where collected data was labeled and presented to a deep learning algorithm for model training. Even though datasets were labeled, machine learning models could find similarities within the presented labeled dataset. They are playing with how plant identification is done in the traditional healing community. If one is unsure of how a particular plant looks like, the person will say the plant slightly looks like a specific plant you might know, making it possible for you to picture the plant. The following sections will discuss deep learning, Tensorflow lite, and Pytorch.

4.5.2.1 Deep learning

Machine learning technique Deep learning is designed to teach machines to do what humans do. The deep learning technique learns how the human brain functions, analyzes certain human decisions and mimics them to machines. Deep learning needs large datasets for it to work correctly. A large dataset requirement is the reason behind the collection of so many plant images, as discussed in the previous chapter in the data collection section. Deep learning is architecturally built on neural networks where input is presented to the input layer. The input layer is connected to hidden layers; many deep networks have hidden layers (Adak, 2021). The neural network type used by Tensorflow lite and Pytorch for research is

convolutional neural networks (CNN). CNN is ideally placed to analyze and process images as 2D data. CNN has 2D convolutional layers that make analysis of images not tricky. CNN architecture is shown in Figure 4-11 below, where convolution, rectified linear units, and pooling are used within the hidden layer to analyze plant images that are input to the algorithm (Adak, 2021).

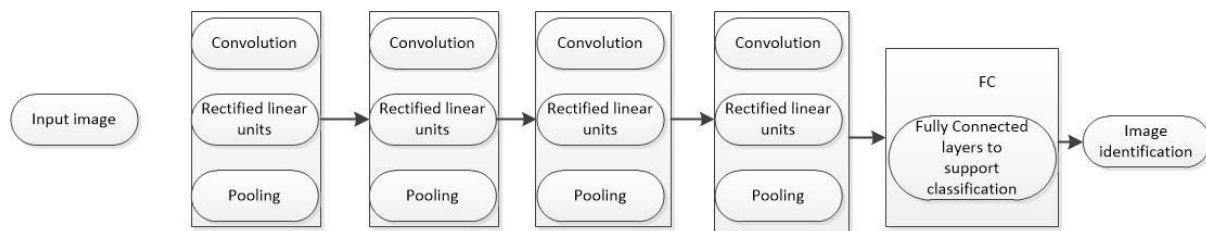
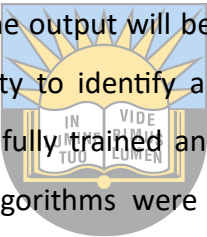


Figure 4-11: CNN architecture(Adak, 2021)

After the plant image passes through hidden layers, it is sent to a fully connected layer to support classification (Adak, 2021). The output will be derived from a fully connected layer, and the algorithm will gain the ability to identify and classify images according to their interpretation. The algorithm will be fully trained and ready to be used for identification. However, before we discuss how algorithms were trained, collected datasets must be prepared by image resizing.


 University of Fort Hare
 Together in Excellence

4.5.2.2 Image Resize

Collected images were resized into smaller sizes to reduce model training time and resources required for the training model. A Python code for image resizing was written on Spyder IDE, shown below. Spyder has an image resizer model to prepare data sets for algorithm training. Figure 4-12 depicts data from the image resizer and how the model maker is dumped into a folder in the workstation where the Firebase storage database will read and upload the trained model.

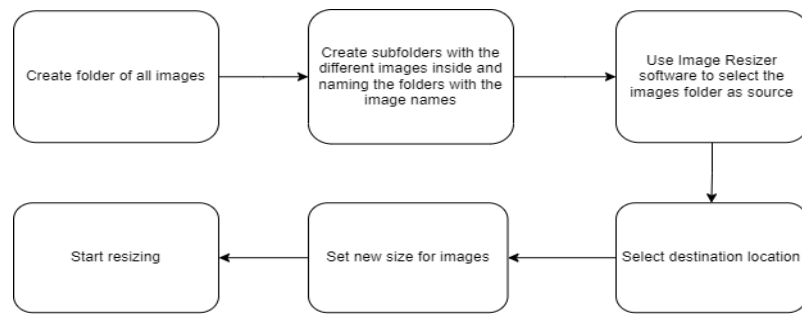


Figure 4-12: Image resizing methodology

Firestore database then presents the trained model to a mobile application built by Android studio (version Dolphin 2021 .3.1), where a mobile application can download and read the trained model. The code below shows the process of image resizing:

```

from PyQt5 import QtWidgets, uic, QtCore, QtGui
from PyQt5.QtWidgets import QMessageBox
import sys
from PIL import Image
from pathlib import Path
import glob
from threading import Thread

sourceFolder = ""
destinationFolder = ""

length = 0
width = 0

class Ui(QtWidgets.QMainWindow):
    def __init__(self):
        super(Ui, self).__init__()
        uic.loadUi('UI/Main Page.ui', self)

        self.pushButtonSelectSource.clicked.connect(self.getFolderSource)
        self.pushButtonSelectDestination.clicked.connect(self.getFolderDestination)
        self.pushButtonResize.clicked.connect(self.StartResize)

```

```

def getFolderSource(self):
    global sourceFolder

    sourceFolder = QtWidgets.QFileDialog.getExistingDirectory(self, 'Select Source Folder', "")
    self.lineEditSourceFolder.setText(sourceFolder)

    imageList = glob.glob(sourceFolder+"/*")

    self.labelSource.setText("Source (" +str(len(imageList))+")")

def getFolderDestination(self):
    global destinationFolder

    destinationFolder = QtWidgets.QFileDialog.getExistingDirectory(self, 'Open Destination Folder', "")
    self.lineEditDestinationFolder.setText(destinationFolder)

```

Snippet 4-13: Image resizing

The code above shows the image resizer model being instructed to get images that will be used for model training using commands such as `def getFolderSource(self)`. The image resizer model is directed to follow a specific path entered on the image resize window, as shown in Figure 4-13 below.



University of Fort Hare
Together in Excellence

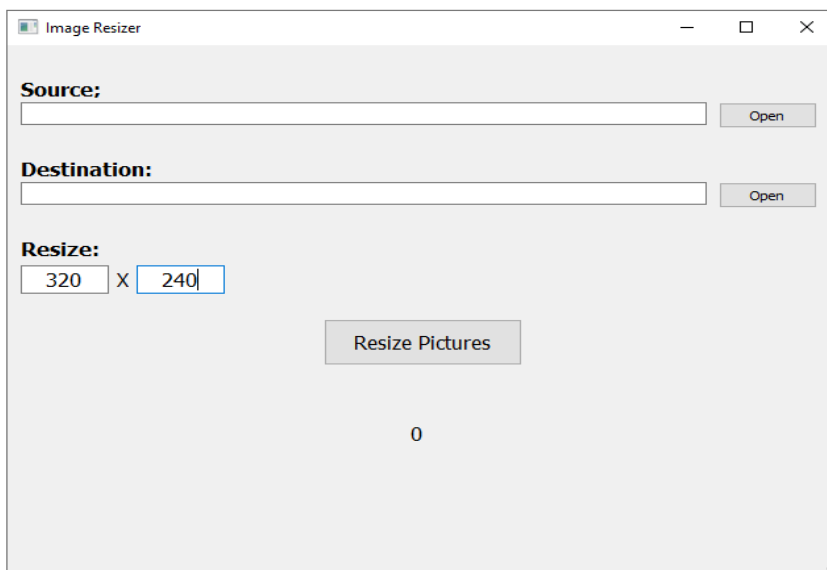


Figure 4-13: Image resizer

Model is also shown where to put resized images on code with command `getFolderDestination(self)` where code instructs model to put resized images to the path shown on destination part of image resizer window with code:

```
destinationFolder = QtWidgets.QFileDialog.getExistingDirectory(self, 'Open Destination Folder', "")
```

`self.lineEditDestinationFolder.setText(destinationFolder)`, part of the above code.

Image resize is started by executing the command `def StartResize(self):` shown below, where the length and width of all images will be inputted as text (as shown in figure 4-13 above), a command that makes input taken as text converted into an integer is `length = int(self.lineEditLength.text())` and `width = int(self.lineEditWidth.text())` as shown on the code below.

```
def StartResize(self):
    global length, width

    length = int(self.lineEditLength.text())
    width = int(self.lineEditWidth.text())

    """d = MyServer()
    d.setDaemon(True)
    d.start()"""
```



University of Fort Hare
Together in Excellence

```
imageList = glob.glob(sourceFolder+"/*")
imageListSize = len(imageList)

count = 0

for x in range(0,imageListSize):
    #print(imageList[x])

    # Opens an image in RGB mode
    im = Image.open(str(imageList[x]))

    imName = Path(im.filename).name
```

```

        newsize = (length, width)
        im1 = im.resize(newsize)
        #im1.show()
        im1.save(destinationFolder+"/"+imName)
        im1.close()

    self.UpdateDestinationText(str(x+1))

    self.UpdateStatusText(str(x+1)+" of "+str(imageListSize)+" Resized")
    count=count+1

self.UpdateStatusText("Complete! \n"+str(count)+" Images resized.")

def UpdateDestinationText(self,text):
    global length, width

    self.labelDestination.setText("Destination (" +text+ ")")

def UpdateStatusText(self,text):

    self.labelStatus.setText(text)

```



University of Fort Hare
Together in Excellence

Snippet 4-14: Image resize set length

New image size is recorded on the image resize model using the command `newsize = (length, width)` applied in all photos on the specified source folder. Command `self.UpdateDestinationText(str(x+1))` is used to update all sizes of images in all subfolders found in the specified destination folder:

```

class MyServer(Thread):
    isAlive = False
    def run(self):
        global sourceFolder, destinationFolder, length, width, window
        self.isAlive = True

    imageList = glob.glob(sourceFolder+"/*")

```

```

imageListSize = len(imageList)

count = 0

for x in range(0,imageListSize):
    #print(imageList[x])

    # Opens an image in RGB mode
    im = Image.open(str(imageList[x]))

    imName = Path(im.filename).name

    newsize = (length, width)
    im1 = im.resize(newsize)
    #im1.show()
    im1.save(destinationFolder+"/"+imName)
    im1.close()

    window.UpdateDestinationText(str(x+1))

    window.UpdateStatusText(str(x+1)+" of "+str(imageListSize)+" Resized")
    count=count+1

print("Stopped")
window.UpdateStatusText("Complete! \n"+str(count)+" Images resized.")

def kill(self):
    self.isAlive = False
    self.soc.close()

app = QtWidgets.QApplication(sys.argv)
window = Ui()
window.show()
app.exec_()

```



University of Fort Hare
Together in Excellence

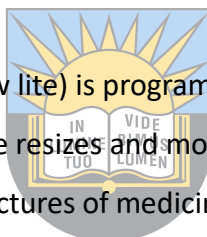
Snippet 4-15: Update all image sizes

When all available images in the source folder are resized, the command `def kill(self):` is executed to ensure that the image resize process stops, and the code terminates after it is executed. When image resizing is done, a dataset can be inputted into the TensorFlow and PyTorch algorithms for training.

4.5.2.3 Tensorflow lite

Tensorflow lite allows plant image recognition to be performed on mobile devices instead of going to the cloud when doing an image identification on the mobile device (Zeroual et al., 2019). The ability to make plant identification locally on mobile devices made the Tensorflow lite method one of the suitable models for research, as fields where these medicinal plants are found can have a network challenge as some of them are remote.

The deep learning method (Tensorflow lite) is programmed into Python code on a Spyder IDE version 5.4.0 application, where image resizes and model training codes are written. The first step was to go to the field to collect pictures of medicinal plants, which were later resized and used in model training. The following section shows how collected datasets were processed through image resizing and model training.



University of Fort Hare
Together in Excellence

4.5.2.4 Pytorch

Pytorch is another machine learning model trained and tested against Tensorflow to see which model best analyzes and uses collected datasets. Like Tensorflow, Pytorch is a deep neural network library that creates and defines models (Novac et al., 2022). Pytorch model training begins by loading the resized images from their respective folders and creating a .csv file that contains each plant name with its corresponding index starting from 0. If there are three types of plants, you will have the following indexes: "0,1,2", 0 being type A, B, and C being the other plant types, respectively. When a model has been trained, it will output an index, which is

used to find out which plant type it corresponds to. Coding for the Pytorch model is done on Spyder IDE. The code for training Pytorch is displayed below:

```
# Train model
for epoch in range(num_epochs):
    losses = []

    for batch_idx, (data, targets) in enumerate(train_loader):
        # Get data to Cuda if possible
        data = data.to(device=device)
        targets = targets.to(device=device)

        # forward
        scores = model(data)
        loss = criterion(scores, targets)

        #Append losses
        losses.append(loss.item())

        # backward
        optimizer.zero_grad()
        loss.backward()

        # gradient descent or Adam step
        optimizer.step()

#Export model to a PyTorch script file
model_scripted = torch.jit.script(model)

#Save the model as type .pt
model_scripted.save('pytorch_model1.pt')
```



University of Fort Hare
Together in Excellence

Snippet 4-16: Train model

A loop is created with code `for batch_idx, (data, targets) in enumerate(train_loader):`. Python is instructed to go through plant images one at a time updating the Pytorch model with plant image features for that specific index. Plant image features are converted to weights; the weight gets updated by executing part of the code that shows `scores = model(data)` where the variable scores are the current weight. The `loss = criterion(scores, targets)` is a variable that has plant features weight stored and updated by executing code `losses.append(loss.item())`. Weight gets updated by executing a loop until all plant images of the same index are finished. The index's total weight is used for plant image identification, where an image that needs to be identified gets converted into a weight. That weight is compared to existing weights on the model database identifying the index. An index is then interpreted into the plant species it

belongs to and is used to determine the plant. The trained model is saved as a .pt file and uploaded to a mobile application for plant identification. The architecture for Pytorch model training is shown in Figure 4-14 below:

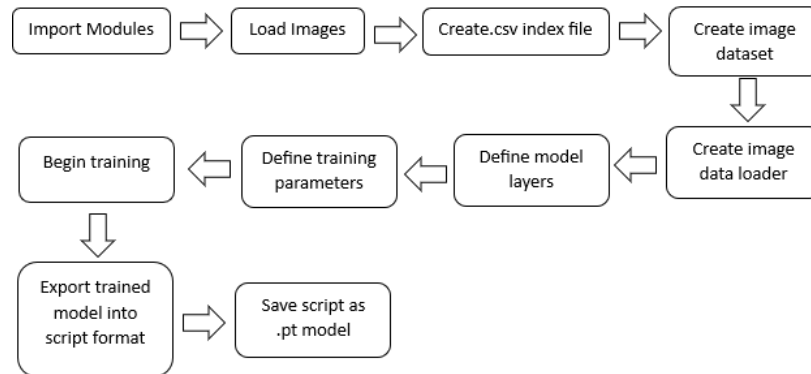


Figure 4-14: Pytorch model training architecture

4.5.3 Model Performance, Evaluation and Validation

Before the evaluation of the two selected models of the research (Ternsoflow and Pytorch) which utilize the CNN algorithm, it was crucial to prove the superiority of the deep learning algorithms by testing the CNN algorithm with other discussed machine learning algorithms. SVM and Random Forest were tested against the CNN algorithm using the same dataset that was collected. Figure 4-15 below displays the accuracy results.

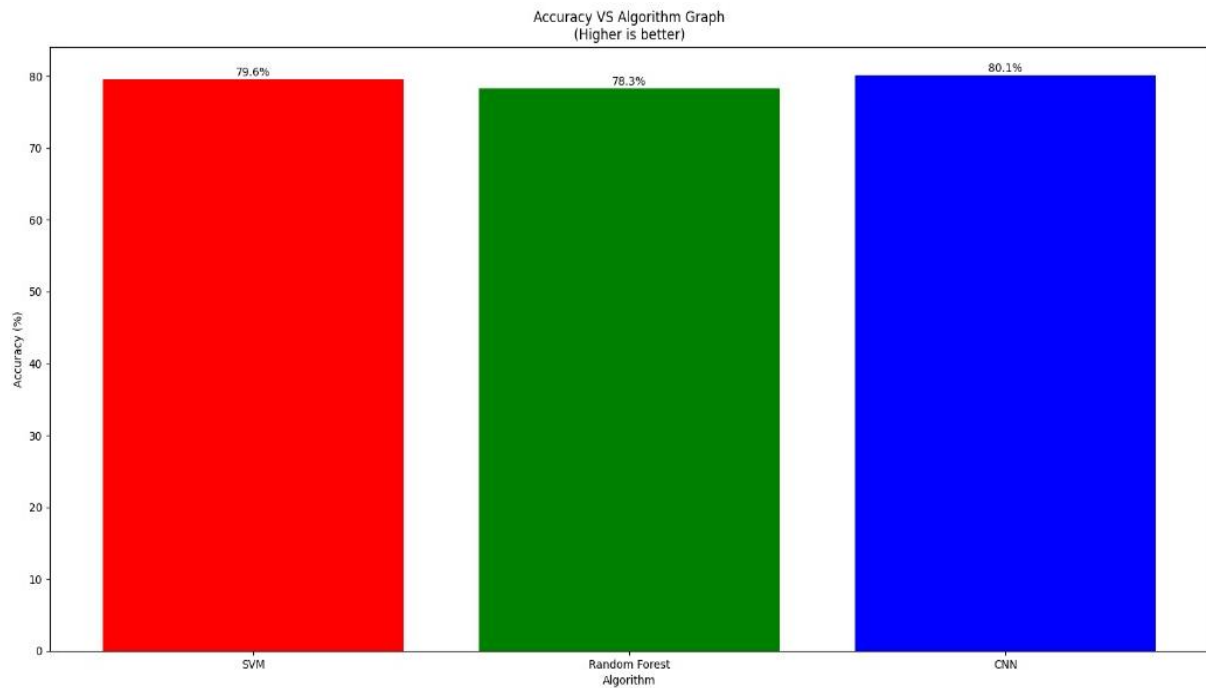


Figure 4-15: Accuracy performances of SVM, Random Forest and CNN

The three models are closely matched in accuracy with CNN being the best performing model at 80.1% accuracy, SVM second placed with 79.6% and lastly Random Forest with 79.3%. The next aspect that was tested between the three models was precision, which is crucial in image identification when combined with accuracy. Figure 4-16 below display the precision results.

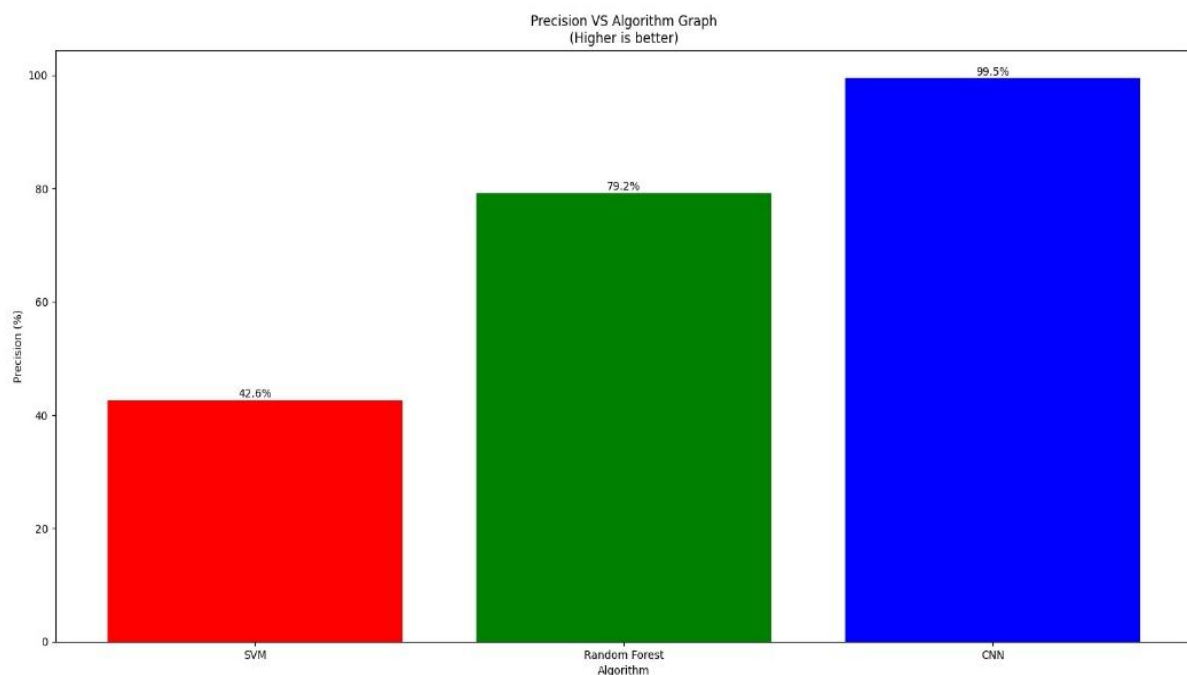


Figure 4-16: Precision performances of SVM, Random Forest and CNN

CNN achieved precision results of 99.5%, which far outperformed the second placed Random Forest algorithm which achieved 79.2% precision. SVM was last placed with results of 42.6%. The precision results support the decision for the research to utilize CNN as the algorithm for the mobile application.

After proving the performance of CNN, it is crucial to discuss a test between two trained deep learning models, Pytorch and TensorFlow that was conducted. Testing these models assisted in selecting the model suitable for the mobile application Izazi. The test was conducted by taking the first plant images for every angle captured at every time of the day (these images were put aside for testing purposes) and analyzing them for plant identification results for each model.

Evaluation of the two models was based on several critical statistical analyses to guide the selection of the final model deployed in the mobile application. A descriptive statistical assessment was performed for both models to provide an overview of how they were performed for each plant under different conditions. An evaluation test was conducted by analyzing the performance of the models for each plant species. Figure 4-17 shows the results of the analysis. Second and third assessments were performed to evaluate the performance of the models at different times of the day (Figure 4-18) and different angles (Figure 4-19).

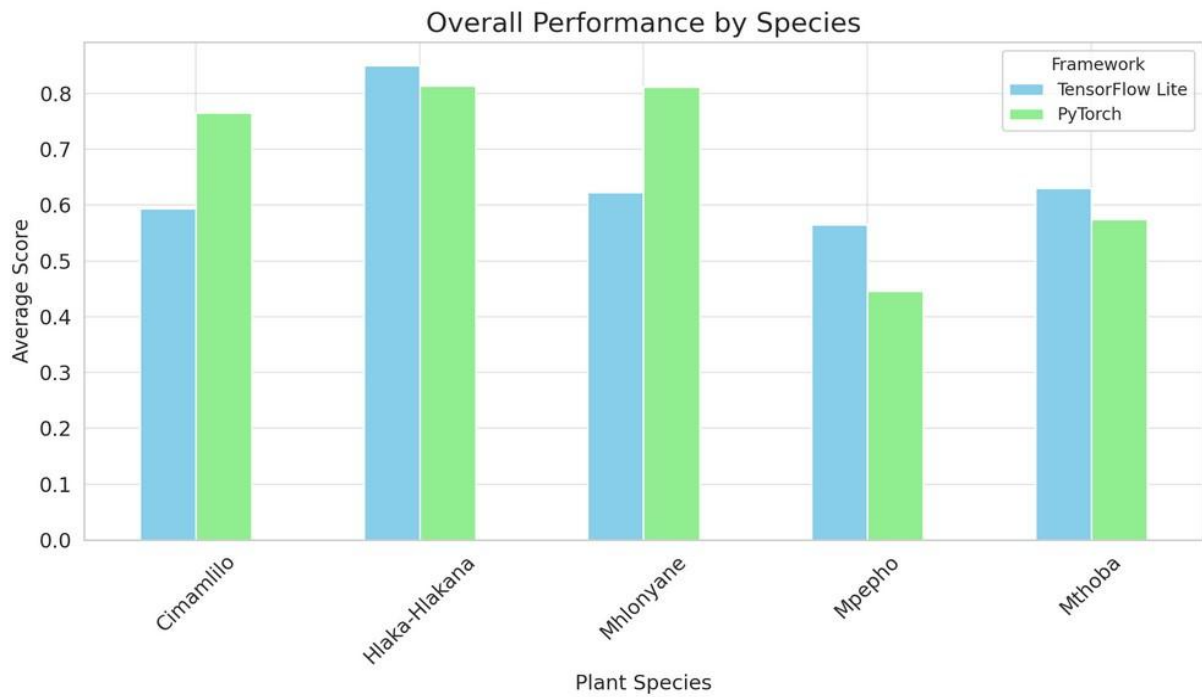


Figure 4-17: Overall performances of the models on all the plant species

The overall performance by time of the day for the different machine learning models is displayed in figure 4-18 below:

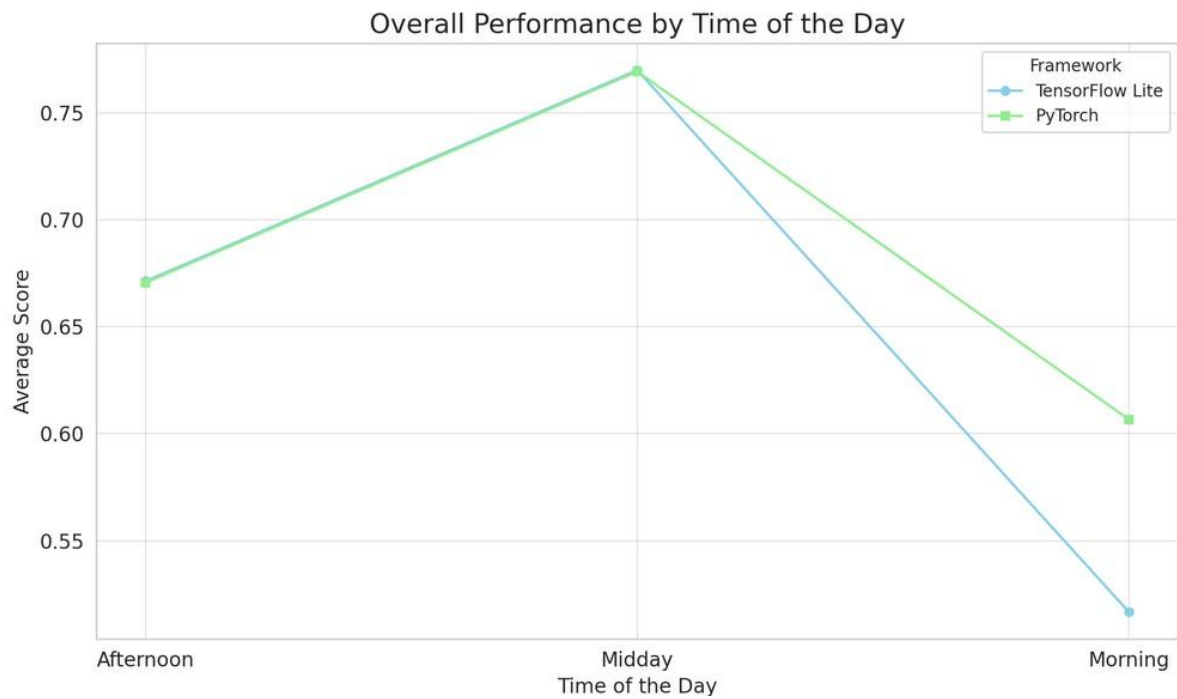


Figure 4-18: Overall performance of the models at different times of day.

The two machine models were then tested on their overall performance on image angles on selected test data sets, as displayed on figure 4-19 below:

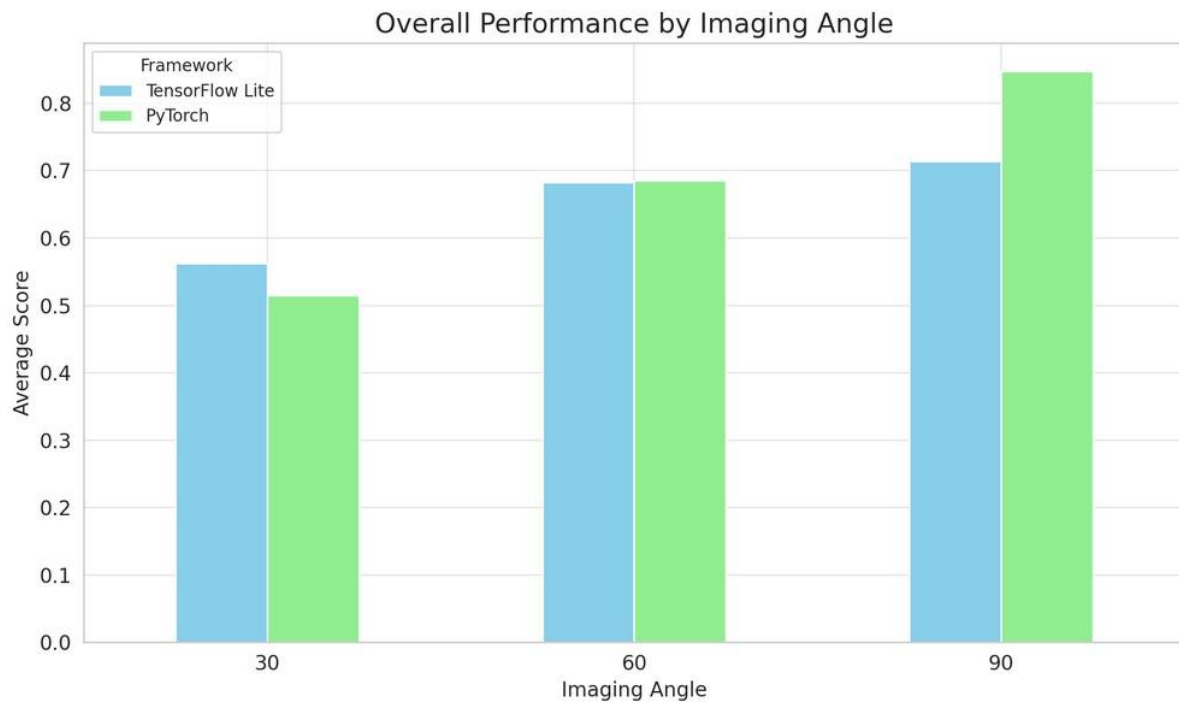


Figure 4-19: Overall performance of models at different angles.

Results indicated that both classification techniques varied across the variables.

Overall performance by day, the TensorFlow model had the highest detection accuracy over PyTorch when classifying Hlaka-Hlakana, with 80% accuracy. However, this was a 4% better detection than PyTorch, which achieved an 81% accuracy. The lowest detection rates in this category were achieved when detecting Mpepho, with 57% and 45% from TensorFlow and PyTorch, respectively. The overall performance of models over the day was relatively comparable in the morning and midday periods. However, PyTorch showed a better detection rate of 61% in the afternoon than that of TensorFlow (52%). A similar performance pattern was achieved when evaluating the performance of the models based on imaging angles. Differences in detection accuracy between the two models were insignificant, except for PyTorch, which achieved the highest detection rate of 85% at an angle of 90°.

4.5.4 Correlation Analysis

A correlation analysis between the two models was also performed for different plant species under various conditions. Results are presented in the correlation matrix in Figure 4-20, indicating that both model frameworks are closely related, with a medium coefficient of 64%. This upbeat but not vital correlation suggests that one model positively identifies a plant. Similar outcomes were achieved using other models. Nonetheless, the correlation could be better, which means there are circumstances in which neither model produces identical results. This could be due to various factors, such as the imaging architecture and optimization used by each model.

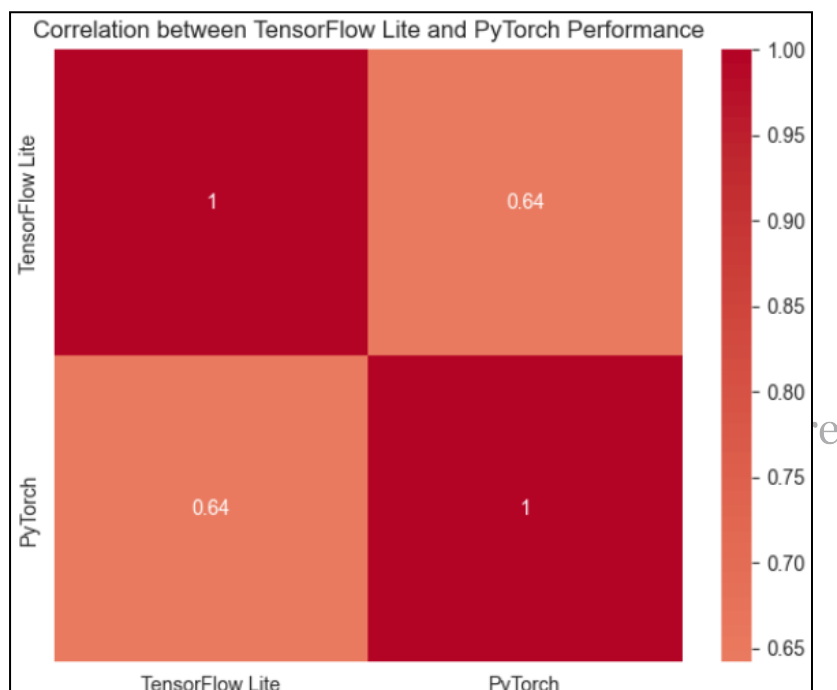


Figure 4-20: Correlation matrix for the performance of the two models

4.5.5 Poor Performance Analysis

Analysis was performed to understand better conditions under which both models did not optimally classify the plants, and the underperformance threshold was set at 50%. Figures 4-21 to 4-24 show graphs of poor performance for both models for plants with a classification of less than 50%.

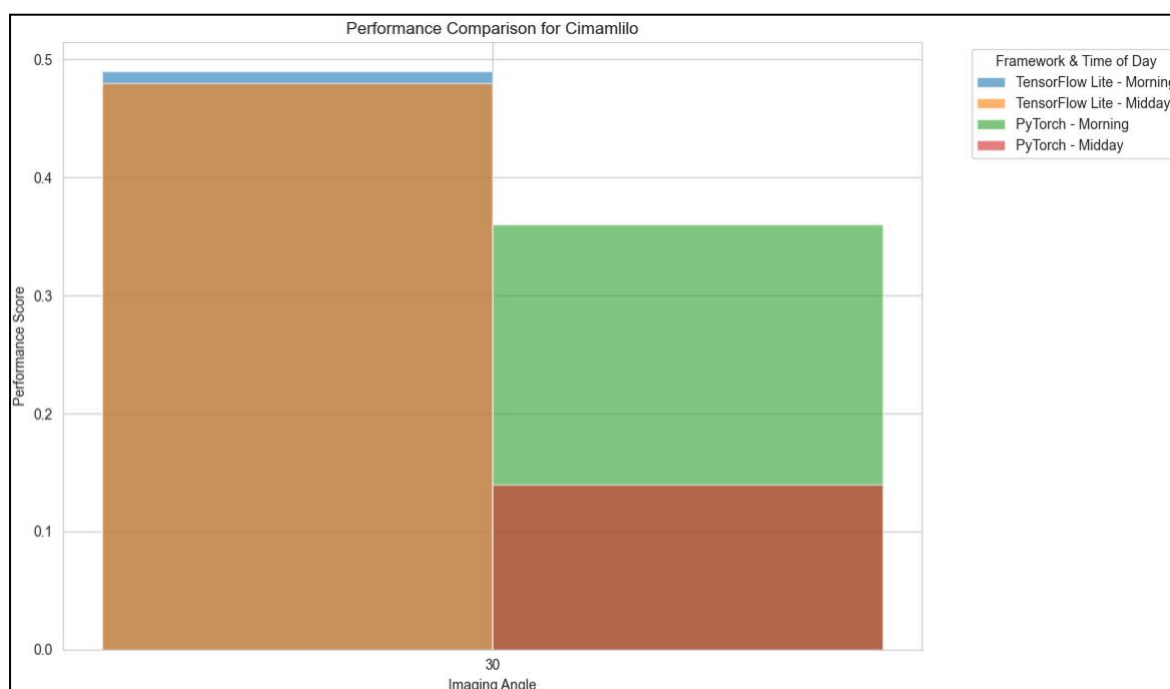


Figure 4-21: Underperformance comparison for Cimamilo for TensorFlow and PyTorch.

The performance results of the medicinal plant Mpepho on different image angles and different time of the day for the two machine learning algorithms are displayed below:

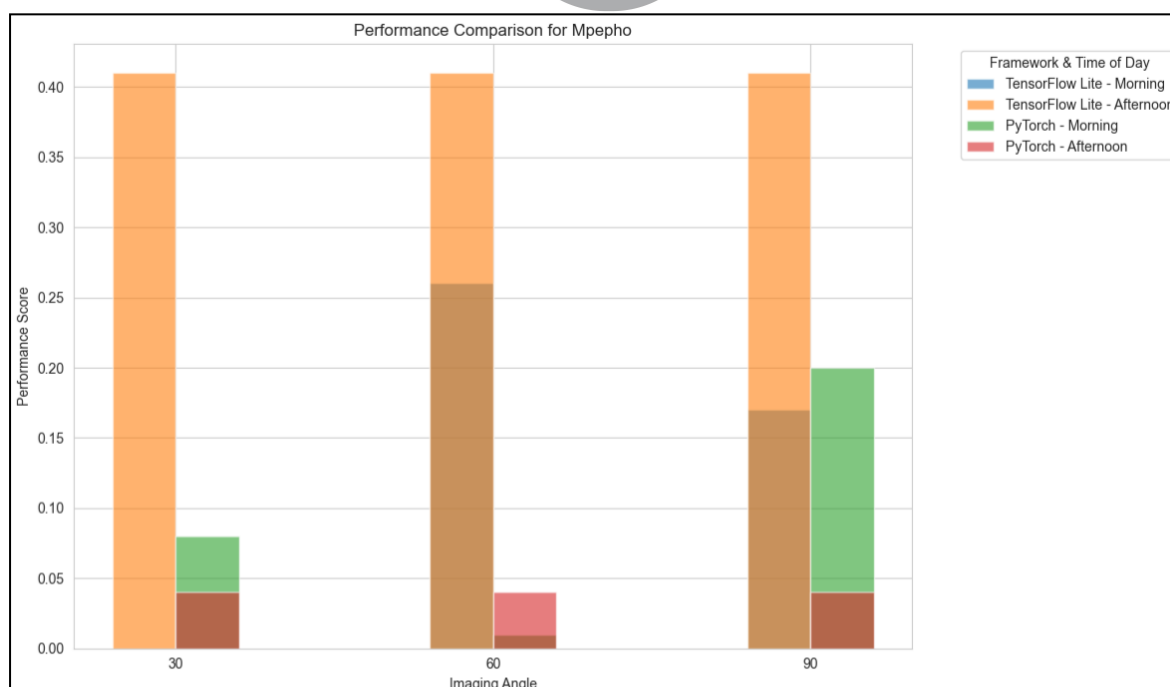


Figure 4-22: Underperformance comparison of Mpepho for TensorFlow and PyTorch.

The under performance results of the plant Mthoba are displayed below:

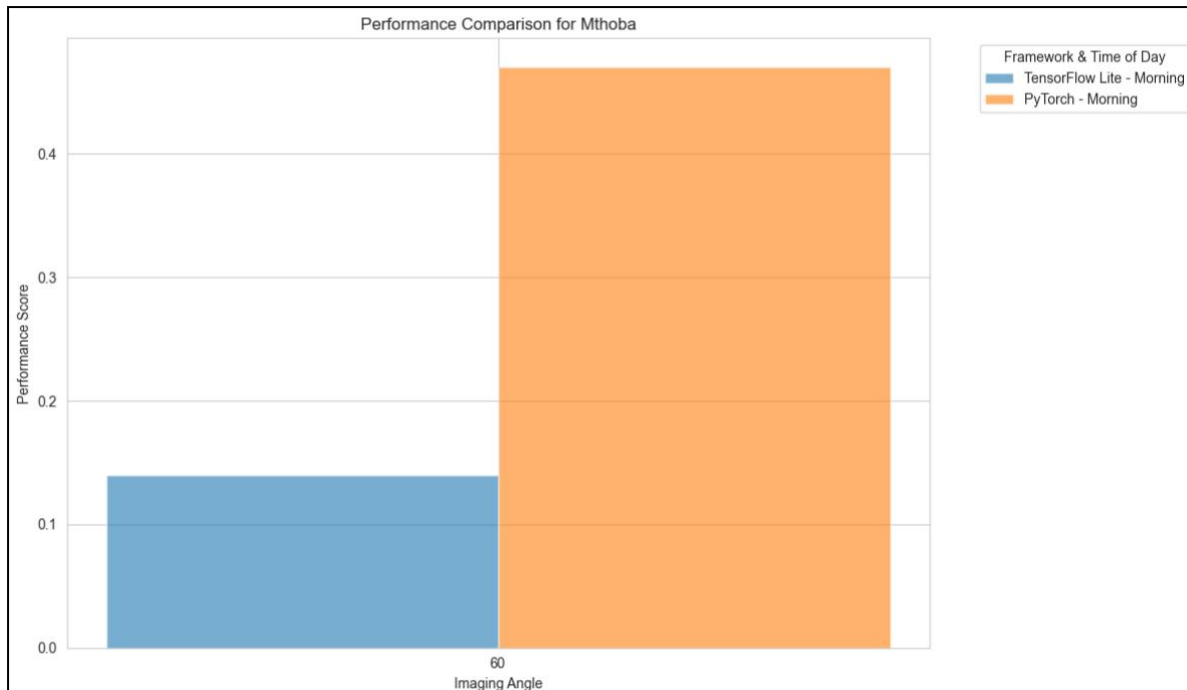


Figure 4-23: Underperformance comparison for Mthoba for TensorFlow and PyTorch.



Mhlonyane had some results displaying under performance as displayed in figure 4-24 below:

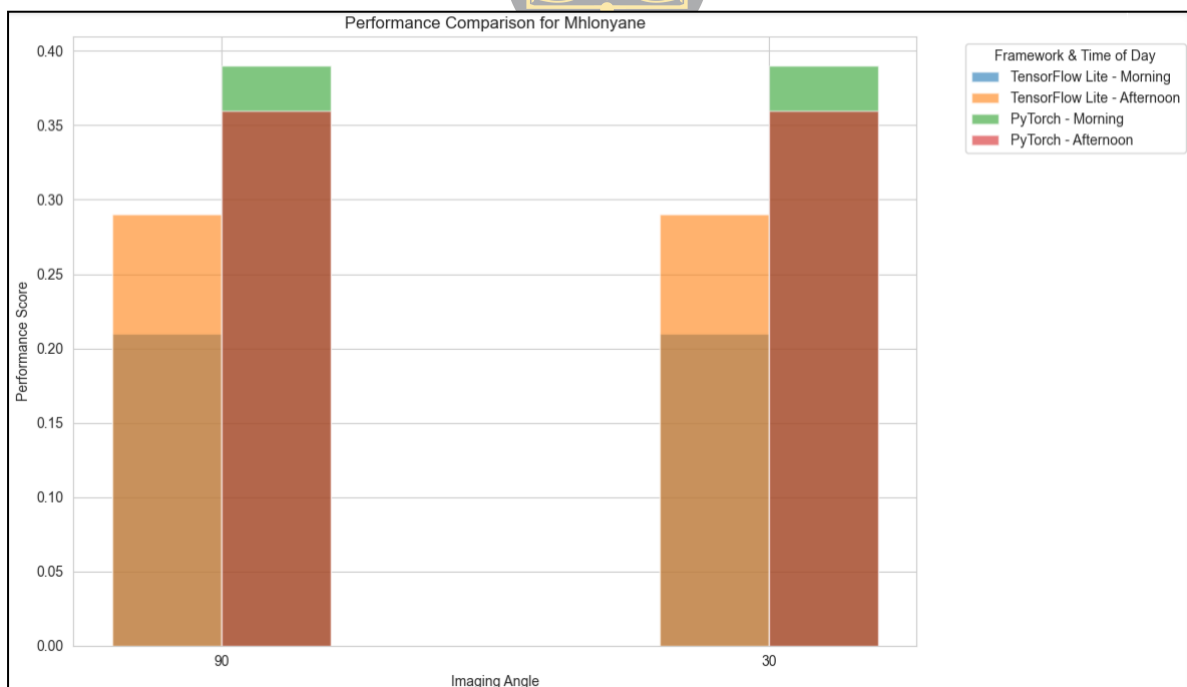


Figure 4-24: Underperformance comparison for Mhlonyane for TensorFlow and PyTorch.

Results demonstrate that the poor performance of both TensorFlow and PyTorch can be attributed to specific imaging angles and different times of the day, indicating that the models are sensitive to environmental conditions. One of the factors of poor performance demonstrated by the models was classifying the plants under low light and at a low angle (30°), which were conditions experienced during the morning. This was particularly evident in models performed with Cimamlilo and Mphepho. Overall, analysis of models for poor performance was significant because it revealed that as much as performance trends and correlations between the two models were similar, individual instances could ideally deviate, justifying complexities of plant species detection using imaging.

4.6 Summary

Chapter Four presents the implementation of the research methodology. As explained in Chapter Three, different methods were explored, and the most suitable methodology for mobile application and machine learning was selected. The approach to data collection was considered in connection with methods selected for two parts of the research (mobile application and machine learning). Code and application were viewed, giving a photographic representation of what methodologies were achieved in research through mobile application screenshots. The selection of a model for mobile applications is also presented, and the reasons behind the selected machine learning model are discussed. In summary, chapter four provided a view of how research was designed, conducted, and implemented. Chapter five will discuss the implementation findings of the current study in comparison to similar studies previously conducted.

CHAPTER FIVE

5: RESULTS AND DISCUSSIONS

5.1 Introduction

In chapter five, results from the implementation of the research are discussed. Results are analyzed and interpreted, and a comparison to related works is also presented. Research is then analyzed according to limitations and interpretations of results. The results will be discussed according to the research objectives.

5.2 Designing of the medicinal plant database

This objective was achieved by creating a medicinal plant library for all users of the Izazi application. The library consists of the five medicinal plants included in this study. Displayed in Figure 5-1 below is the medicinal plant library:

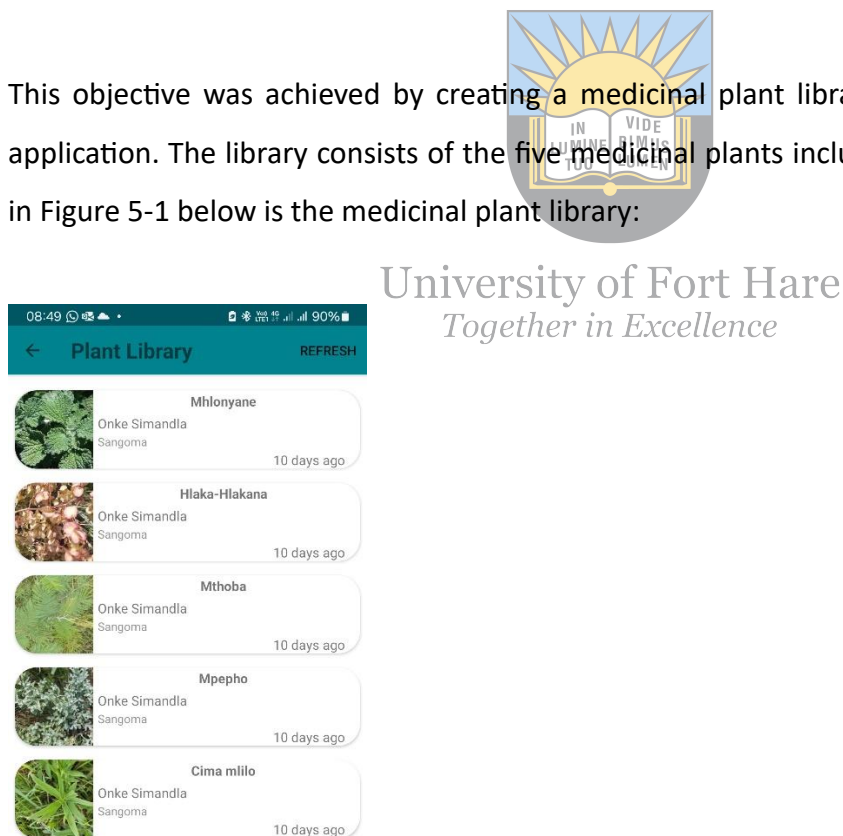


Figure 5-1: Medicinal plant library

The medicinal plant library displays each plant's uses. Plant uses are displayed in English and Xhosa, as shown in Figures 5-2. The language option makes it easier for mobile application users to understand plant uses in their most convenient language.

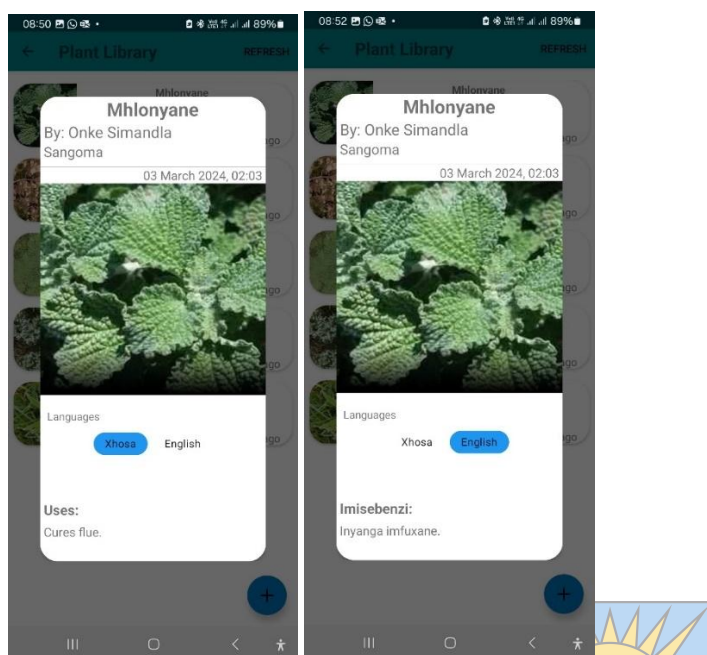


Figure 5-2: (a) Mhlonyane uses in English (b) Mhlonyane uses in Xhosa

The uses of the plants were also listed in the plant library in English and Xhosa. The uses assist the mobile application user to understand the impact the plant has. The choice of the two languages (English and Xhosa) was made as the study of this mobile application was focused on the Eastern Cape at the Hogsback Mountain range where English and Xhosa are the dominant languages.

5.3 Designing the mobile application interface

This objective was achieved by creating a mobile application that was comfortable for users to operate under the different conditions of cellular network availability. The login process was made easier for smartphones that have the fingerprint login capability. A registered mobile application user had the option of using password or fingerprint login. Figure 5-3 illustrates the login process of the mobile application.

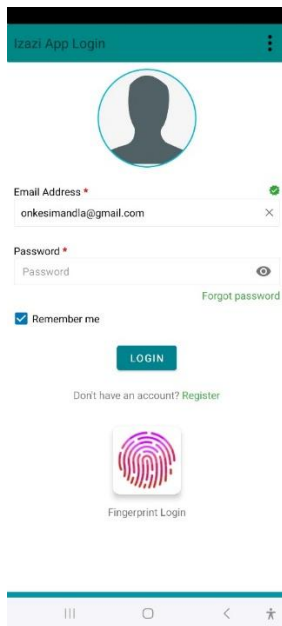
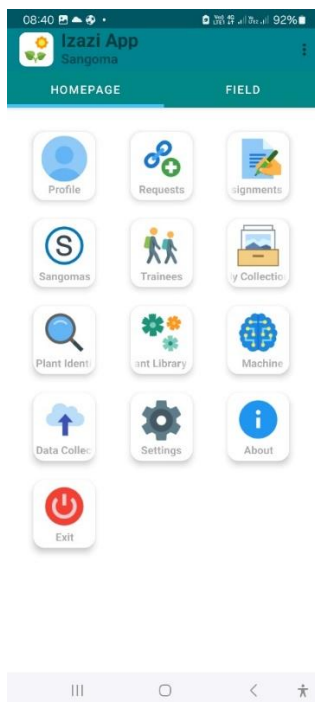


Figure 5-3: Login process for the mobile application

The mobile application was designed to take the user to the menu page after being successfully logged in. Figure 5-4 below shows the menu page of the Izazi mobile application.



University of Fort Hare
Together in Excellence

Figure 5-4: Izazi mobile application menu

Mobile application process operation was also tested to determine the response speed of certain mobile application processes. Results seek to display the operational efficiency of the

mobile application. Results are discussed in the following order: Random Access Memory (RAM) performance, Central Processing Unit (CPU) performance, and Application Security.

5.3.1 RAM performance

RAM performance results focus on mobile device memory performance when a mobile application is used. The aim was to see how memory-intensive the mobile application is. An application on a Samsung mobile device called Device Care (version 13.6.01.4) was used to measure results. Results are based on a comparison with other similar plant-identifying applications discussed in the study. Memory utilization of mobile applications and processes within the application is tested.

Table 5-1: Mobile application memory testing

Mobile Application	Trained machine learning algorithm hosting	Training data amount	Initial mobile device memory reading	Application launch memory reading difference	Plant identification launch memory reading difference	Plant identification memory reading difference
Izazi	On-device	27000	919 MB	86 MB	72 MB	252 MB
Pl@ntNet	On server	21920	919 MB	91 MB	162 MB	666 MB
LeafSnap	On server	5000	919 MB	188 MB	79 MB	269 MB

Memory in mobile devices is an essential resource as there is no possibility of extending it, unlike in workstations and servers, which makes memory in mobile devices a sensitive resource that needs to be looked after. A higher utilization of memory leads to slower performance of mobile devices. It is then crucial for a mobile application to be memory sensitive (not use too much memory), and the test shown in Table 5-1 addresses that aspect. The test was conducted on a Samsung A23 smartphone with Android version 13. All background applications were closed, ensuring that when an application being tested was opened, it was the only one using device resources on available memory. When no application was running on a mobile device, the memory reading was 919 MB for all the applications tested. Applications that were tested against the mobile application for this study were Pl@ntNet (Affouard et al., 2017) and LeafSnap (Barré et al., 2017).

The Izazi mobile application was tested first; memory utilization was tested when the application launch finished (the application launch included user login). Results showed an increase of 86 MB utilization in memory from the application, better than Pl@ntNet, which recorded 91 MB, and LeafSnap, recording 188MB, which means that the total available memory for the mobile device decreased by 86 MB when the Izazi mobile application was launched. The plant identification process was launched on the Izazi mobile application, and a further decrease of 72 MB in memory was observed, which is better than the 162 MB recorded from Pl@ntNet and 79 MB from LeafSnap. When the plant identification process was done on the three applications, the Izazi mobile application performed better with a 252 MB increase in memory utilization, followed by LeafSnap at 269 MB, and Pl@ntNet was recorded at 666MB.

Memory results make more sense when including hosting of machine learning algorithm and training data amount. Hosting of an algorithm refers to where the trained algorithm database was hosted (on a mobile device or server). Training data amount refers to plant images used to train machine learning algorithms. When looking at the three applications being tested, the Izazi application is the one application that has a trained machine algorithm hosted on a smartphone, with other applications hosting their algorithms on a server. The number of training images used for machine learning algorithms was the highest for the Izazi application, with 27000 images. Pl@ntnet mobile application is second with 21920 training datasets. LeafSnap used about 5000 images for algorithm training; the amount of 5000 was used as it is difficult to estimate the exact number (in the LeafSnap study, training images are referred to as about 5000) Barré et al. (2017) were referring to.

The number of training datasets means that a larger trained machine learning model affects mobile device memory when executing a specific mobile application process. The performance of the Izazi application in memory tests is impressive, with the largest dataset for the machine learning model. An advantage of the Izazi application over other applications is that it is tested with on-device hosting of a machine learning model. On-device hosting means having a machine learning algorithm database locally on a device. There is a direct link between the application plant identification process and the machine learning algorithm responsible for identification. They are making the Izazi application use less memory

compared to other applications. The difference between on-device hosting and server hosting of the machine learning model is that PI@ntNet and LeafSnap mobile applications must query a server that is hosting the trained model and wait for the server's response. PI@ntNet and LeafSnap have higher memory utilization than the Izazi application. Memory utilization for LeafSnap for plant identification was closer to the Izazi application. LeafSnap's machine learning algorithm had only 5000 images as a training dataset. This means there is far less plant identification information for the LeafSnap application, hence lower memory utilization than PI@ntNet.

5.3.2 CPU performance

When specific mobile application processes are executed, the performance of the mobile device CPU was tested. Phone Configuration Checker is an application used to test the CPU performance of smartphones. The processor of the Samsung A23 device used for this test was AArch64 Processor rev 0 (aarch64). The CPU architecture was 4 x 2,40GHz and 4 x 1,90GHz on a 64-bit CPU with 300 MHz to 1900 MHz frequency.

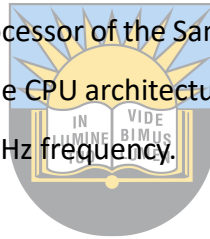


Table 5-2 Mobile application CPU testing

Mobile Application	Trained machine learning algorithm hosting	Training data amount	Initial mobile device CPU reading	Application launch CPU reading difference	Plant identification launch CPU reading difference	Plant identification CPU reading difference
Izazi	On-device	27000	44%	24%	9%	39%
PI@ntNet	On server	21920	44%	32%	13%	24%
LeafSnap	On server	5000	44%	41%	41%	31%

The same processes were used to test memory and CPU performance for consistency. CPU test aims to check the time a mobile application takes to execute a specific task. The initial CPU utilization reading on the smartphone for all three tested applications, as reflected in Table 5-2, was 44%. Results for the application launch of the Izazi mobile application show an increase of 24% in CPU utilization, where PI@ntNet recorded 32%, and LeafSnap CPU utilization for the same process was 41%. This means that the Izazi mobile application takes a shorter time to launch than other applications being tested. When opening the plant identification process, the Izazi application recorded a CPU increase of 9%, whereas PI@ntNet

recorded 13%, and LeafSnap had a 41% CPU utilization increase. When identifying a plant, the Izazi application got the highest CPU increase with 39%, where Pl@ntNet was 24%, and LeafSnap was 31%. The Izazi application had the highest CPU utilization when identifying a plant image because of the trained machine learning algorithm database hosted locally on a smartphone. The advantage of hosting an algorithm on a server is highlighted in these results as the smartphone sends a request to a server (the server is the one that is identifying) and waits for feedback. This means that during the process of plant identification, a mobile application needs to wait for the server to perform plant identification process and provide feedback to the mobile application. These results show a disadvantage of the machine learning algorithm on-device hosting. Izazi is the third application in CPU performance for identifying a plant image after Pl@ntNet and Leafsnap applications.

5.3.3 Application Security

The security of a mobile application was tested by testing the reliability of the fingerprint process. The test was done by checking if a user would be allowed to log in to an already existing profile with fingerprints already captured on the system if a wrong fingerprint was used. Results showed that the application successfully denied an unknown user trying to access the device using fingerprints, as displayed in Figure 5-5 below.

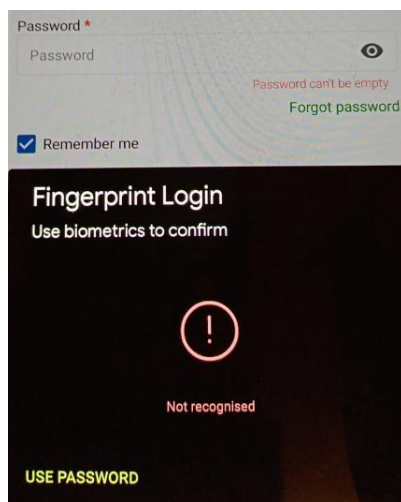
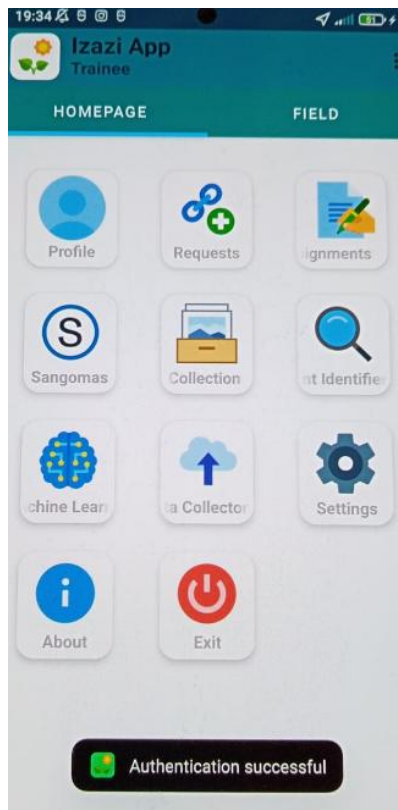


Figure 5-5: Fingerprint not recognized test

The login process for the Izazi mobile application is also possible offline, making it perfect for places with low or no coverage, as most medicinal plants are found in such areas. Logging

process tests were done with both online and offline mobile applications. The offline mode of the mobile application was accompanied by placing a mobile device in flight mode to ensure there was no internet connectivity on a smartphone. The login process still worked under flight mode, making the application secure and operational offline. Login details are downloaded to the device the first time the user logs into the application, enabling a mobile application to retrieve login data offline. Figure 5-6 below displays a login process to an application in an offline format.



University of Fort Hare
Together in Excellence

Figure 5-6: Offline login

5.4 Developing an AI algorithm based on image recognition techniques to extract relevant features from the captured plant images.

Machine learning results are discussed, and the identification results of 5 different plant species are displayed. Results for plant image identification were divided into two sections: image angles test results and image test results according to weather. Results were used to test for any difference from different angles in the plant identification process. The weather was also tested for any difference in the plant identification process. The central hypothesis

of the study is to see if the mobile application can identify medicinal plants using a selected Pytorch machine-learning algorithm. It is also important to investigate plant identification deeper by testing two aspects (image angles and weather conditions) to see if these aspects have any influence on the identification process.

5.4.1 Image angles and weather conditions test results

The results discussed are from different angles (30 degrees, 60 degrees, and 90 degrees) used to capture plant images. The time of day (morning, midday, and afternoon) was tested in conjunction with the image angle. The weather was not considered in pictures taken as conditions were sunny, mostly making the comparison according to weather unbalanced. The mobile application Angle Meter was used to determine the angles of pictures to be taken.

Figure 5-7 below illustrates the Angle Meter application.

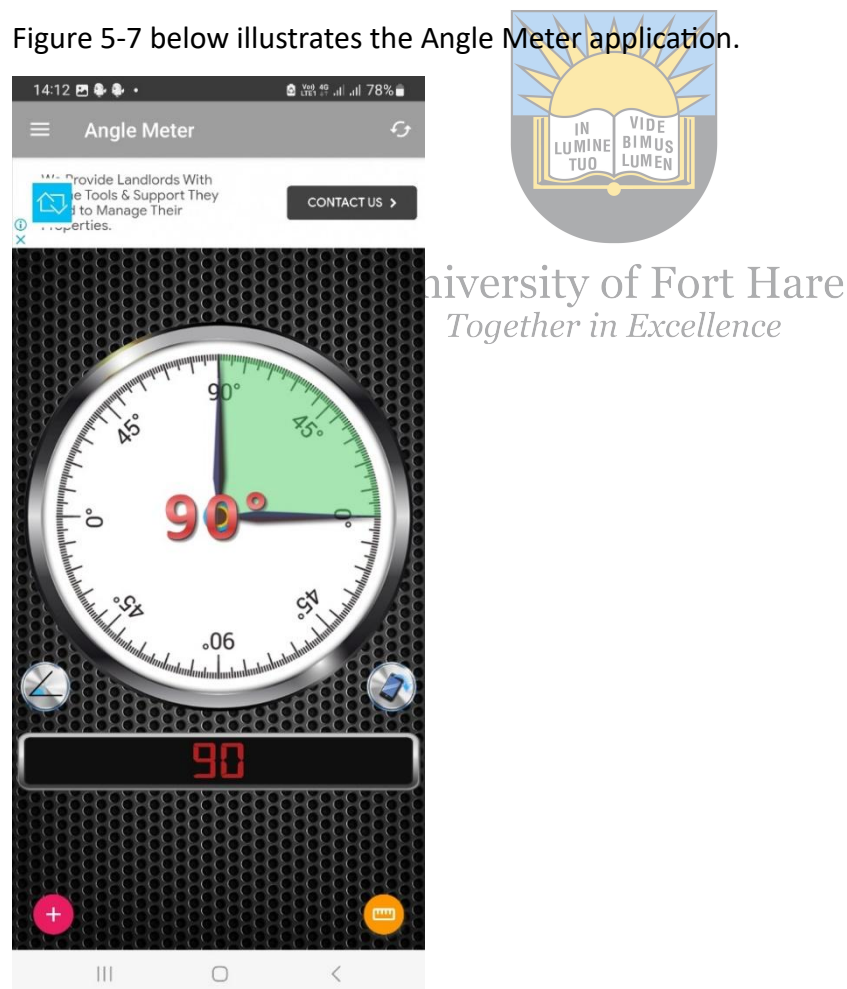


Figure 5-7: Angle meter application

To compare different image angles, on-field testing of mobile applications was done. Images were taken from each angle group at each time of the day measured. For the first plant species, Cimamlilo, 1 picture was taken for on-field testing from the 30-degree angle for the morning session; the same was done for the midday and afternoon sections. Several 30-degree images were tested for plant species Cimamlilo, which was 3 (1 per different time of the day and image angle). The same was repeated for 60-degree and 90-degree angles of plant images, meaning that nine pictures were tested per plant species and times of the day. In total, 45 plant images were included in the test. Displayed in Table 5.3 below are average test results from different plant species tested.

Table 5-3: Plant identification results

Plant name	Morning			Midday			Afternoon		
	30 degrees	60 degrees	90 degrees	30 degrees	60 degrees	90 degrees	30 degrees	60 degrees	90 degrees
Cimamlilo	90%	95%	96%	99%	99%	96%	92%	97%	98%
Mpepho	64%	54%	88%	51%	62%	83%	77%	88%	99%
Mthoba	88%	93%	98%	87%	91%	91%	89%	97%	97%
Hlaka-Hlakana	58%	44%	50%	78%	73%	91%	73%	75%	94%
Mhlonyane	77%	95%	94%	99%	98%	99%	89%	98%	99%
Total Average	75%	76%	85%	83%	84%	92%	84%	91%	98%

University of Fort Hare

Results are shown in Tables 5-3 for each plant species, and the angles were taken from one plant image of each angle for every time of the day. This means every result in Table 5-3 is from 1 plant identification result taken using the Izazi application (except for the Total Average values). Plant images were identified randomly on the field at Hogsback Mountains for on-field results purposes. An illustration of plant species Cimamlilo's first image of the morning on a 90-degree angle identification results is shown in Figure 5-8 below.



Figure 5-8: Cimamlilo identified by the Izazi mobile application

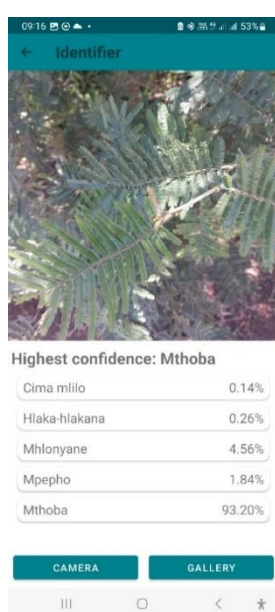
Four other plant images for the 90-degree angle are identified, as illustrated in Figure 5-8, and identification percentages are captured. The same process happened for all other angles and time of day slots. Results were arranged into tables, from which the identification results shown in tables 5-3 are taken. Shown below is Table 5-4, from which the identification results illustrated in Figure 5-6 are represented.

Table 5-4: Cimamlilo morning identification results

Plant name	Time of the day	Image angle	Identification results
Cimamlilo	Morning	30	89,72%
Cimamlilo	Morning	60	94,73%
Cimamlilo	Morning	90	95,52%

From understanding how identification results are put together, results are discussed. 90-degree angle was the highest in identification results average for all the plant species at all times of the day measured. The afternoon was the highest time of the day for identification results average for all measured angles. This means a user of the Izazi mobile application is more likely to have a higher success rate when taking a plant image at 90 degrees angle midday between 15:00 and 17:59. Results show an identification result average of 98% for the afternoon 90 degrees results, 91% for 60 degrees angle and 84% for 30 degrees. Results for 60-degree angles make the angle the second best in plant identification. A 30-degree angle is the least performing angle for the afternoon timeslot at an overall rate of 84%.

These results do not imply that the Izazi mobile application cannot identify a plant when an image is taken at a different time of day and angle. During the afternoon, the sun's position on Earth changes, making the angle at which the sun is shining on Earth change, as Patel (2017) discussed in a paper about Low-cost and robust solar tracking systems. Sun position becomes evident in the image below, taken in the morning at 60 60-degree angle for the plant species Mthoba. Results from images were one of the highest for all the 60-degree angle images taken in the morning at 93%. The first plant image taken in the morning at the 60-degree angle is displayed below in Figure 5-9.



University of Fort Hare
Together in Excellence

Figure 5-9: Mthoba plant image at a 60 degrees angle

Table 5-3 shows that the Izazi mobile application machine learning algorithm can identify plants at different timeslots and angles. The 90-degree angle in the afternoon image is most likely to have a higher identification ratio than other timeslots and angle combinations. Another interesting fact about results is that when we look at Hlaka-Hlakana results in Table 5-3 for a morning on the 60-degree angle, identification results are 44%. The reason behind these results is displayed in Figures 5-10 below.

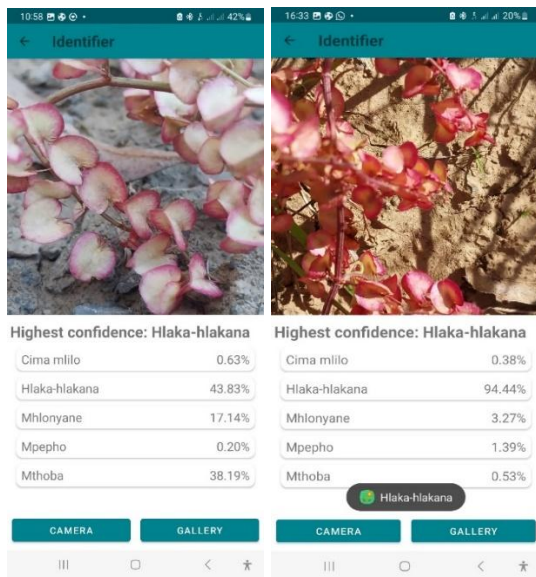


Figure 5-10: (a) Hlaka-Hlakana on the shadows (b) Hlaka-Hlakana under the direct sunlight

Looking closely at figure 5-10, there is clear evidence of shadows in the Figure 5-10 (a) image while the figure 5-10 (b) image displays plants under direct sunlight. The reason was that the Hlaka-Hlakana plant was found next to giant trees. Shadows from trees covered Hlaka-Hlakana in the morning, which might have affected its identification. The point of shadows affecting plant identification was not tested in this research. In the afternoon, shadows from trees did not cover the Hlaka-Hlakana plant, giving mobile applications enough chance to identify images. Conditions under which dataset images were taken become crucial as the algorithm returns to those images to try and find similarities. The identification result of 44% shows that tree shadows on those plants found next to trees are an important aspect of dataset collection when data is collected from their natural habitat without removing it.

Another interesting fact about results was for the plant Mpepho. Looking at Figure 5-11 below, identification results for Mpepho in the morning and midday were low on average compared to the afternoon results.

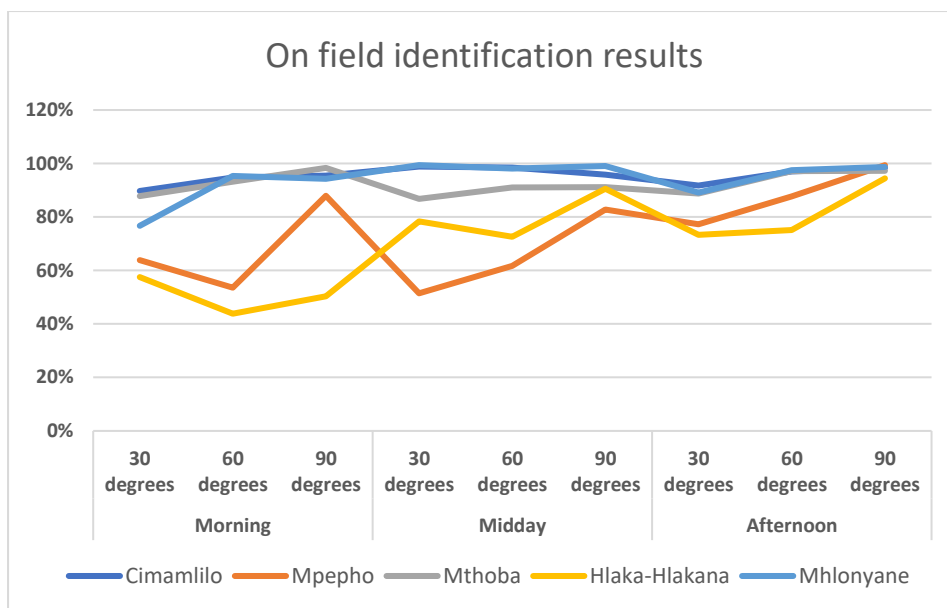


Figure 5-11: On-field identification results

The difference in identification results for Mpepho can be explained by looking at Figure 5-12 below. Figure 5-12 (a) shows the Mpepho plant at a different growth stage than the image in Figure 5-12 (b).

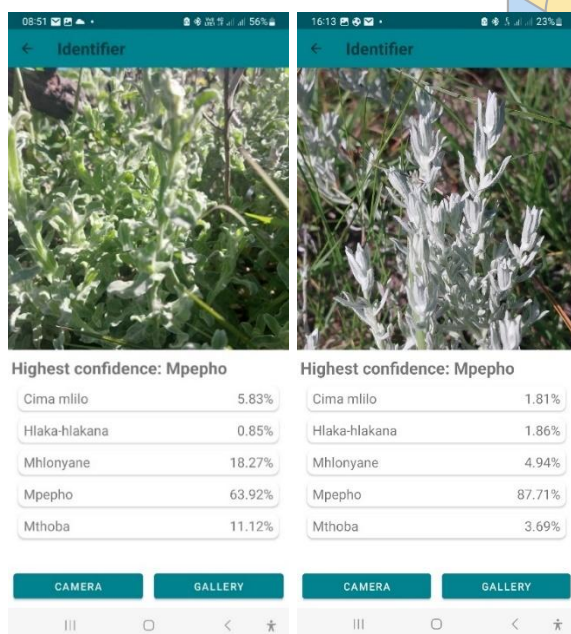


Figure 5-12: (a) Mpepho (b) Mpepho on different growth stage as (a)

Images show different growth stages of the same Mpepho plant. There is a difference in how the two images look, even though they are from the same plant. The machine learning algorithm was trained with both types of images showing different growth stages included.

The plant identification process was able to identify the Mpepho plant on both occasions, but the identification process might stand a better chance with a collection of different plant growth phases separated as individual plants. An example is Mpepho early growth stage images collected separately, and a machine learning algorithm trained with only this data set (the same should be done for Mpepho late growth stage).

The overall performance of the iZazi mobile application in plant identification is displayed in Figure 5-13 below. The most favorable time of the day for plant identification for the Izazi mobile application on all medicinal plants tested is the Afternoon (from 15:00 to 17:59). The preferred angle for most successful results is the 90° angle. The most prosperous time of the day and angle produced average results of 98% for all the medicinal plants tested.

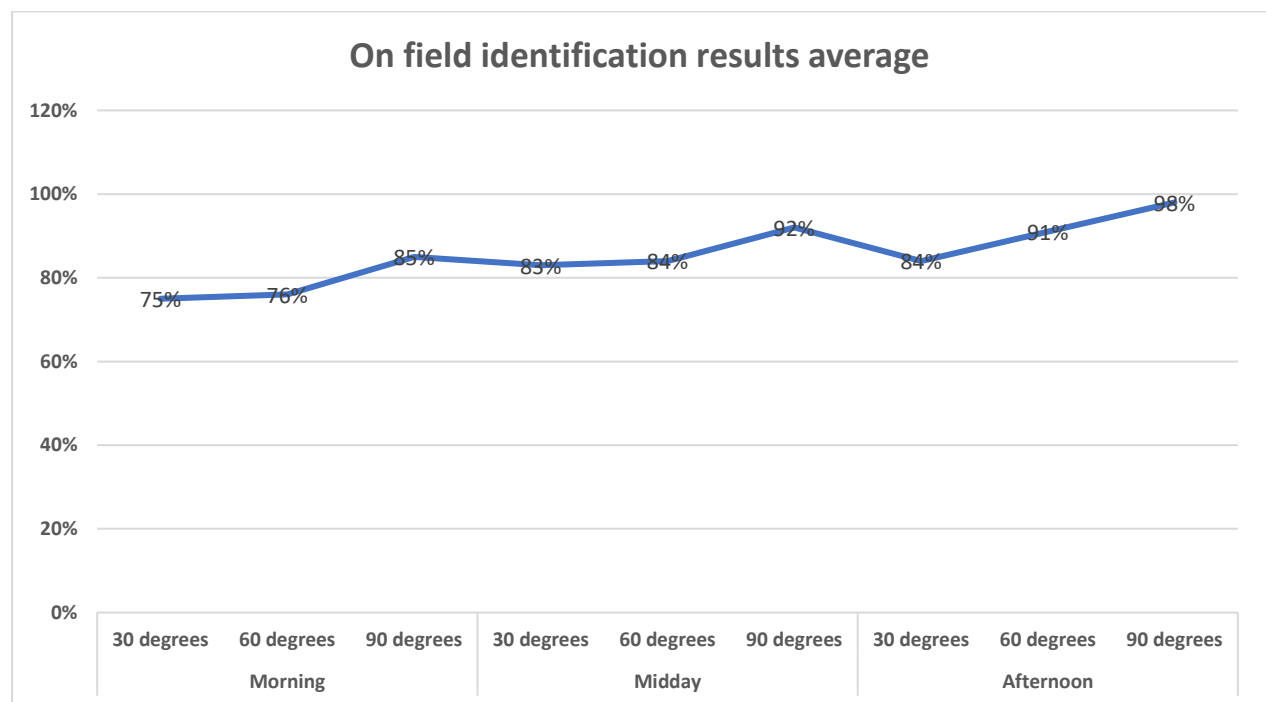


Figure 5-13: On-field identification results average

5.5 Evaluation of CNN

The performance of TensorFlow and PyTorch was successfully evaluated with PyTorch outperforming TensorFlow. Figure 4-20 to Figure 4-25 display the results between PyTorch and TensorFlow where PyTorch achieved the highest identification of 85% on the most favorable angle of 90° where TensorFlow achieved 71%. Figure 4-20 displays performance results based

on plant species where TensorFlow and PyTorch were closely matched with PyTorch outperforming TensorFlow.

5.6 Summary

In Chapter Five, results from the implementation of research methodology and design are discussed. A mobile application was successfully implemented with offline capabilities that were key to the traditional healing community, as some parts where these medicinal plants grow are remote with very limited to no signal. The security aspect of mobile applications was covered by introducing the capability of logging in by fingerprints. The machine learning section of the mobile application worked well with an average of 98% identification results (98% average taken from 5 identification results of plant images at 90 degrees angle in the afternoon) for the most successful identification method. Implementing the Izazi mobile application also introduced challenges in plant identification and dataset collection on shadow cover, which shows the depth of the research topic and the lengths that still need to be covered.



University of Fort Hare
Together in Excellence

CHAPTER SIX

6. CONCLUSION AND RECOMMENDATIONS

6.1 Introduction

In chapter six, an overview of the research is given, and objectives and implementation based on objectives are discussed. Chapter Six also covers further extensions from research that can form future studies.

6.2 Research Objectives Analysis

Objective 1: Designing an intuitive and user-friendly mobile application interface for capturing plant images and submitting them for identification. The research aims to guide mobile application design successfully. Literature review assisted with finding the best methods when approaching a design for a mobile application. The mobile application design was successfully created using design science research methodology—similar research assisted in guiding some design decisions like machine learning algorithm training. Figure 4.1 shows the system design achieved. A mobile application was developed using Python programming language on a Dell workstation based on system design. The mobile application was developed and deployed to an Android smartphone on a Samsung A9 and Samsung A23 smartphones, respectively. The machine learning model was integrated into the mobile application and implemented successfully on the smartphone.

Objective 2: Develop an AI algorithm based on image recognition techniques to extract relevant features from the captured plant images. AI algorithm development was successfully done in research. As discussed in Chapter Four, image recognition techniques for extracting plant features were applied under the machine learning section. The AI algorithm can successfully identify medicinal plants researched, as revealed by results in chapter five.

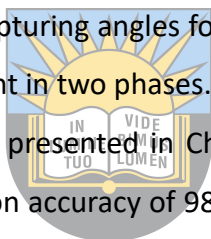
Objective 3: Create a comprehensive database of medicinal plant species, including information on their properties, traditional uses, and potential medical applications.

A database containing medicinal plant species and their uses has been successfully created. As displayed in Chapter 5, a user can see the uses of these medicinal plants.

Objective 4: Training the AI algorithm using machine learning techniques to recognize and classify different medicinal plant species based on the extracted image features.

AI algorithms were trained using data sets collected from natural habitats. Images from data sets were presented to AI algorithms for training.

Objective 5: Evaluating performance and accuracy of the developed mobile application through rigour testing, comparing its results with manual identification and validation by experts. The rigour testing was done by testing one image per each time of the day section, and one image per different image capturing angles for each plant species. Testing was done during mobile application development in two phases. First, testing for choosing a suitable AI algorithm was successfully done and presented in Chapter 4. Second, on-field testing was successfully done with an identification accuracy of 98% from the most successful angle and time of the day, as presented in Chapter 5.



University of Fort Hare
Together in Excellence

6.3 Recommendations and Conclusion

Recommendations from this study can be made to address challenges faced when identifying Mpepho morning plant images. Plant images of a matured Mpepho plant (which looks different from the same plant in a different growth phase) were used to identify against a model primarily trained by plant images of a different growth phase. Other studies can focus on identifying plants in their different growth phases, where datasets of the same plant can be grouped according to how different medicinal plants look in their growth phases. Another part of the study that can be furthered is to create a system for the traditional healing community that will focus on the official curriculum of a system for the conventional healing journey. This means there needs to be a system that outlines the journey that will be taken by iThwasa, with all the different phases incorporated into the traditional healing process.

Another study that can be done from this study is to create a system that will host a database of medicinal plants known by their traditional names. It is challenging to find Hlaka-Hlakana online as all created databases have scientific names unknown by most of the conventional healing community.

The study about medicinal plant identification using artificial intelligence did introduce technological advances to the traditional healing community. Doing more studies in the indigenous knowledge systems will assist with using technology to improve the African indigenous systems.



University of Fort Hare
Together in Excellence

References:

- Abdollahi, J. (2022a) 'Identification of medicinal plants in ardabil using deep learning: identification of medicinal plants using deep learning', in *2022 27th International Computer Conference, Computer Society of Iran (CSICC)*. IEEE, pp. 1–6.
- Adade Williams, P., Sikutshwa, L. and Shackleton, S. (2020) 'Acknowledging indigenous and local knowledge to facilitate collaboration in landscape approaches♦Lessons from a systematic review', *Land*, 9(9), p. 331.
- Anand, U. *et al.* (2019) 'A comprehensive review on medicinal plants as antimicrobial therapeutics: potential avenues of biocompatible drug discovery', *Metabolites*, 9(11), p. 258.
- Antonelli, A. *et al.* (2020) *State of the World♦s Plants and Fungi*. PhD Thesis. Royal Botanic Gardens (Kew); Sfumato Foundation.
- Barbedo, J.G.A. (2019) 'Plant disease identification from individual lesions and spots using deep learning', *Biosystems Engineering*, 180, pp. 96–107.
- Bhardwaj, S., Verma, R. and Gupta, J. (2018) 'Challenges and future prospects of herbal medicine', *International research in medical and health sciences*, 1(1), pp. 12–15.
- Bi, Y., Xue, B. and Zhang, M. (2020) 'Genetic programming with a new representation to automatically learn features and evolve ensembles for image classification', *IEEE transactions on cybernetics*, 51(4), pp. 1769–1783.
- Bock, C.H. *et al.* (2020) 'From visual estimates to fully automated sensor-based measurements of plant disease severity: status and challenges for improving accuracy', *Phytopathology Research*, 2(1), pp. 1–30.
- Boho, D. *et al.* (2020) 'Flora Capture: a citizen science application for collecting structured plant observations', *BMC bioinformatics*, 21(1), pp. 1–11.
- Castiglioni, I. *et al.* (2021) 'AI applications to medical images: From machine learning to deep learning', *Physica Medica*, 83, pp. 9–24.
- Chalmers, D., MacKenzie, N.G. and Carter, S. (2021) 'Artificial intelligence and entrepreneurship: Implications for venture creation in the fourth industrial revolution', *Entrepreneurship Theory and Practice*, 45(5), pp. 1028–1053.
- Chattu, V.K. (2021) 'A review of artificial intelligence, big data, and blockchain technology applications in medicine and global health', *Big Data and Cognitive Computing*, 5(3), p. 41.
- Demie, G., Negash, M. and Awas, T. (2018) 'Ethnobotanical study of medicinal plants used by indigenous people in and around Dirre Sheikh Hussein heritage site of South-eastern Ethiopia', *Journal of Ethnopharmacology*, 220, pp. 87–93.

- Dhillon, A. and Verma, G.K. (2020) 'Convolutional neural network: a review of models, methodologies and applications to object detection', *Progress in Artificial Intelligence*, 9(2), pp. 85–112.
- Giger, M.L. (2018) 'Machine learning in medical imaging', *Journal of the American College of Radiology*, 15(3), pp. 512–520.
- Jamshidi-Kia, F., Lorigooini, Z. and Amini-Khoei, H. (2017) 'Medicinal plants: Past history and future perspective', *Journal of herbmed pharmacology*, 7(1), pp. 1–7.
- Jha, K. *et al.* (2019) 'A comprehensive review on automation in agriculture using artificial intelligence', *Artificial Intelligence in Agriculture*, 2, pp. 1–12.
- Lee, S.H. *et al.* (2017) 'How deep learning extracts and learns leaf features for plant classification', *Pattern Recognition*, 71, pp. 1–13.
- Liu, B.-Y. *et al.* (2022) 'Two-stage convolutional neural networks for diagnosing the severity of alternaria leaf blotch disease of the apple tree', *Remote Sensing*, 14(11), p. 2519.
- Muneer, A. and Fati, S.M. (2020) 'Efficient and automated herbs classification approach based on shape and texture features using deep learning', *IEEE Access*, 8, pp. 196747–196764.
- Noor, F. *et al.* (2022) 'Network pharmacology approach for medicinal plants: Review and assessment', *Pharmaceuticals*, 15(5), p. 572.
- Nowak, M.M. *et al.* (2020) 'Mobile GIS applications for environmental field surveys: A state of the art', *Global Ecology and Conservation*, 23, p. e01089.
- Raji, I.D. *et al.* (2020) 'Closing the AI accountability gap: Defining an end-to-end framework for internal algorithmic auditing', in *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pp. 33–44.
- Salma, U., Kundu, S. and Mandal, N. (2017) 'Artificial polyploidy in medicinal plants: advancement in the last two decades and impending prospects', *Journal of crop science and biotechnology*, 20, pp. 9–19.
- Shivaprakash, K.N. *et al.* (2022) 'Potential for Artificial Intelligence (AI) and Machine Learning (ML) applications in biodiversity conservation, managing forests, and related services in India', *Sustainability*, 14(12), p. 7154.
- Singh, K.K. (2018) 'An artificial intelligence and cloud based collaborative platform for plant disease identification, tracking and forecasting for farmers', in *2018 IEEE international conference on cloud computing in emerging markets (CCEM)*. IEEE, pp. 49–56.
- Souza, K.E.S. de *et al.* (2022) 'An evaluation framework for user experience using eye tracking, mouse tracking, keyboard input, and artificial intelligence: A case study', *International Journal of Human–Computer Interaction*, 38(7), pp. 646–660.
- Strasser, B. and Haklay, M. (2018) 'Citizen science: Expertise, democracy, and public participation'.

- Süntar, I. (2020) 'Importance of ethnopharmacological studies in drug discovery: role of medicinal plants', *Phytochemistry Reviews*, 19(5), pp. 1199–1209.
- Tian, H. *et al.* (2020) 'Computer vision technology in agricultural automation❖A review', *Information Processing in Agriculture*, 7(1), pp. 1–19.
- Wäldchen, J. *et al.* (2018) 'Automated plant species identification❖Trends and future directions', *PLoS computational biology*, 14(4), p. e1005993.
- Wäldchen, J. and Mäder, P. (2018) 'Machine learning for image based species identification', *Methods in Ecology and Evolution*, 9(11), pp. 2216–2225.
- Abu Alfeilat, H.A. *et al.* (2019) 'Effects of distance measure choice on k-nearest neighbor classifier performance: a review', *Big data*, 7(4), pp. 221–248.
- Affouard, A. *et al.* (2017) 'PI @ ntNet app in the era of deep learning To cite this version : HAL Id : hal-01629195'.
- Agbokou, K. and Mensah, Y. (2023) 'Varying bandwidth parameter method on Kernel Gini index estimation', 50(1), pp. 29–41. doi:10.52846/ami.v50i1.1602.
- Al-Saqqah, S., Sawalha, S. and Abdelnabi, H. (2020) 'Agile software development: Methodologies and trends', *International Journal of Interactive Mobile Technologies*, 14(11), pp. 246–270. doi:10.3991/ijim.v14i11.13269.
- Alwalah, M., Alshhry, S. and Elbadri, L. (2019) 'Android Vs IOS performance', (December), pp. 0–1.
- Alzubi, J., Nayyar, A. and Kumar, A. (2018) 'Machine Learning from Theory to Algorithms: An Overview', *Journal of Physics: Conference Series*, 1142(1). doi:10.1088/1742-6596/1142/1/012012.
- Anticipatory, A. and Improvement, P. (2020) 'A Survey on Different Data Mining & Machine Learning Methods for Credit Card Fraud Detection', (March). doi:10.13140/RG.2.2.22116.73608.
- Archana, K.S. *et al.* (2022) 'A novel method to improve computational and classification performance of rice plant disease identification', *Journal of Supercomputing*, 78(6), pp. 8925–8945. doi:10.1007/s11227-021-04245-x.
- Aroral, H.K. (2021) 'Waterfall Process Operations in the Fast-paced World: Project Management Exploratory Analysis', *International Journal of Applied Business and Management Studies*, 6(1), p. 2021.
- Bagherzadeh, J. and Asil, H. (2019) 'A review of various semi-supervised learning models with a deep learning and memory approach', *Iran Journal of Computer Science*, 2, pp. 65–80.
- Balaji, S., Nathani, K. and Santhakumar, R. (2019) 'IoT technology, applications and challenges: a contemporary survey', *Wireless personal communications*, 108, pp. 363–388.
- Banerjee, M. *et al.* (2019) 'Tree-based analysis: A practical approach to create clinical decision-making tools', *Circulation: Cardiovascular Quality and Outcomes*, 12(5), pp. 1–11.

doi:10.1161/CIRCOUTCOMES.118.004879.

- Bansal, M., Goyal, A. and Choudhary, A. (2022) 'A comparative analysis of K-Nearest Neighbor, Genetic, Support Vector Machine, Decision Tree, and Long Short Term Memory algorithms in machine learning', *Decision Analytics Journal*, 3(May), p. 100071. doi:10.1016/j.dajour.2022.100071.
- Barré, P. *et al.* (2017) 'LeafNet: A computer vision system for automatic plant species identification', *Ecological Informatics*, 40, pp. 50–56. doi:10.1016/j.ecoinf.2017.05.005.
- Batta, M. (2018) 'Machine Learning Algorithms - A Review', *International Journal of Science and Research (IJSR)*, 18(8), pp. 381–386. doi:10.21275/ART20203995.
- Cao, W., Liu, Q. and He, Z. (2020) 'Review of Pavement Defect Detection Methods', *IEEE Access*, 8, pp. 14531–14544. doi:10.1109/aACCESS.2020.2966881.
- Carvalho, D. V., Pereira, E.M. and Cardoso, J.S. (2019) 'Machine learning interpretability: A survey on methods and metrics', *Electronics (Switzerland)*, 8(8), pp. 1–34. doi:10.3390/electronics8080832.
- Cervantes, J. *et al.* (2020) 'A comprehensive survey on support vector machine classification: Applications, challenges and trends', *Neurocomputing*, 408, pp. 189–215.
- Chandra, M.A. and Bedi, S.S. (2021) 'Survey on SVM and their application in image classification', *International Journal of Information Technology*, 13, pp. 1–11.
- Cui, Y. and Qu, F. (2021) 'Experimental discussion on fire image recognition based on deep learning', *Journal of Physics: Conference Series*, 2066(1). doi:10.1088/1742-6596/2066/1/012071.
- Cunningham, P. and Delany, S.J. (2021) 'K-Nearest Neighbour Classifiers-A Tutorial', *ACM Computing Surveys*, 54(6). doi:10.1145/3459665.
- Datilo, P.M., Ismail, Z. and Dare, J. (2019) 'A Review of Epidemic Forecasting Using Artificial Neural Networks', 6(3), pp. 132–143. doi:10.15171/ijer.2019.24.
- Deng, X. *et al.* (2019) 'Feature selection for text classification: A review', *Multimedia Tools and Applications*, 78, pp. 3797–3816.
- Díez-Sanmartín, C. and Cabezuelo, A.S. (2020) 'Application of artificial intelligence techniques to predict survival in kidney transplantation: A review', *Journal of Clinical Medicine*, 9(2). doi:10.3390/jcm9020572.
- Dogiparthi, H. (2019) 'Rapid Application Development Harish Dogiparthi University of the Cumberlands Dept. of Information technology', pp. 1–5. Available at: https://www.researchgate.net/publication/336601939_Rapid_Application_Development.
- Dutta, D. (2022) 'Edtech with Perspective of Business Analysis', *Interantional Journal of Scientific Research in Engineering and Management*, 06(03), pp. 0–7. doi:10.55041/ijsrem12044.
- Eshete, M.A. and Molla, E.L. (2021) 'Cultural significance of medicinal plants in healing human ailments

- among Guji semi-pastoralist people, Suro Barguda District, Ethiopia', *Journal of Ethnobiology and Ethnomedicine*, 17(1), pp. 1–18. doi:10.1186/s13002-021-00487-4.
- Feng, K. *et al.* (2022) 'An integration model for converter molten steel end temperature prediction based on Bayesian formula', *steel research international*, 93(2), p. 2100433.
- Gajowniczek, K. and Ząbkowski, T. (2021) 'Interactive decision tree learning and decision rule extraction based on the imbtreetentropy and imbtreetauc packages', *Processes*, 9(7). doi:10.3390/pr9071107.
- Ganesan, V. (2022) 'Machine Learning in Mobile Applications', (March). doi:10.47760/ijcsmc.2022.v11i02.013.
- Gechman, M. (2019) 'Software Development Methodologies', *Project Management of Large Software-Intensive Systems*, (August 2013), pp. 49–66. doi:10.1201/9780429027932-4.
- Gemaque, R.N. *et al.* (2020) 'An overview of unsupervised drift detection methods', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(6), pp. 1–18. doi:10.1002/widm.1381.
- Ghosh, S., Dasgupta, A. and Swetapadma, A. (2019) 'A study on support vector machine based linear and non-linear pattern classification', in *2019 International Conference on Intelligent Sustainable Systems (ICISS)*, pp. 24–28.
- Gogul, I. and Kumar, V.S. (2017) 'Flower species recognition system using convolution neural networks and transfer learning', *2017 4th International Conference on Signal Processing, Communication and Networking, ICSCN 2017* [Preprint], (March) doi:10.1109/ICSCN.2017.8085675.
- González-Pérez, A. *et al.* (2022) 'Using mobile devices as scientific measurement instruments: Reliable android task scheduling', *Pervasive and Mobile Computing*, 81, p. 101550. doi:10.1016/j.pmcj.2022.101550.
- Hamzah, M.A. and Othman, S.H. (2021) 'A Review of Support Vector Machine-based Intrusion Detection System for Wireless Sensor Network with Different Kernel Functions', 11(1), pp. 59–67.
- Hong, Y., Lin, J. and Chen, W. (2019) 'Simulation of thermal field in mass concrete structures with cooling pipes by the localized radial basis function collocation method', *International Journal of Heat and Mass Transfer*, 129, pp. 449–459.
- Howes, M.J.R. *et al.* (2020) 'Molecules from nature: Reconciling biodiversity conservation and global healthcare imperatives for sustainable use of medicinal plants and fungi', *Plants People Planet*, 2(5), pp. 463–481. doi:10.1002/ppp3.10138.
- Ibrahim, F.A. and Shiba, O.A. (2019) 'Data Mining : WEKA Software (an Overview)', *Journal of Pure & Applied Sciences*, 18(3), pp. 54–58. Available at: www.Suj.sebhau.edu.ly.

- Iheanacho, V.U. (2022) 'Christianity and medical science: A historical quest for remedy in time of epidemics', *Acta Theologica*, 42(1), pp. 39–58. doi:10.18820/23099089/actat.v42i1.4.
- Inbavel, M. and Prakasam, P.S. (2022) 'Analysis on Hidden Layer in Deep Artificial Neural Network for Classification of Sentiments', in *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 1646–1653.
- Jiménez, V., Afonso, P. and Fernandes, G. (2020) 'Using agile project management in the design and implementation of activity-based costing systems', *Sustainability (Switzerland)*, 12(24), pp. 1–23. doi:10.3390/su122410352.
- Khan, R.A. *et al.* (2022) 'Security Assurance Model of Software Development for Global Software Development Vendors', *IEEE Access*, 10(May), pp. 58458–58487. doi:10.1109/ACCESS.2022.3178301.
- Khatib, M. El *et al.* (2021) 'The impact of blockchain on project management', *ICIC Express Letters*, 15(5), pp. 467–474. doi:10.24507/icicel.15.05.467.
- Kramer, M. (2018) 'Best Practices in Systems Development Lifecycle: An Analyses Based on the Waterfall Model', *Review of Business & Finance Studies*, 9(1), pp. 77–84.
- Kumar, R., Maheshwary, P. and Malche, T. (2019) 'Inside Agile Family Software Development Methodologies', *International Journal of Computer Sciences and Engineering*, 7(6), pp. 650–660. doi:10.26438/ijcse/v7i6.650660.
- Kurniawan, F.F., Shidiq, F.R. and Sutoyo, E. (2020) 'WeCare Project: Development of Web-based Platform for Online Psychological Consultation using Scrum Framework', *Bulletin of Computer Science and Electrical Engineering*, 1(1), pp. 33–41. doi:10.25008/bcsee.v1i1.6.
- De Laat, P.B. (2018) 'Algorithmic Decision-Making Based on Machine Learning from Big Data: Can Transparency Restore Accountability?', *Philosophy and Technology*, 31(4), pp. 525–541. doi:10.1007/s13347-017-0293-z.
- Lee, C.S., Cheang, P.Y.S. and Moslehpour, M. (2022) 'Predictive Analytics in Business Analytics: Decision Tree', *Advances in Decision Sciences*, 26(1), pp. 1–29. doi:10.47654/V26Y2022I1P1-30.
- Malhotra, K. and Singh, A.P. (2021) 'Implementation of decision tree algorithm on fpga devices', *IAES International Journal of Artificial Intelligence*, 10(1), pp. 131–138. doi:10.11591/ijai.v10.i1.pp131-138.
- Marcus, G. (2018) 'Deep Learning: A Critical Appraisal', pp. 1–27. Available at: <http://arxiv.org/abs/1801.00631>.
- Margotti, F. (2023) 'Linear Functional Strategy and the Approximate Inverse for Nonlinear Ill-Posed Problems', *Numerical Functional Analysis and Optimization*, pp. 1–34.
- Maulany, R. *et al.* (2021) 'Design of learning applications using the Rapid Application Development

- method', *IOP Conference Series: Materials Science and Engineering*, 1098(2), p. 022090. doi:10.1088/1757-899x/1098/2/022090.
- Mesia Dhas, J.T. and Shiny Angel, T. (2022) 'APPLICATIONS', (April).
- Mienye, I.D., Sun, Y. and Wang, Z. (2019) 'Prediction performance of improved decision tree-based algorithms: A review', *Procedia Manufacturing*, 35, pp. 698–703. doi:10.1016/j.promfg.2019.06.011.
- Mostajer Kheirkhah, F. and Asghari, H. (2019) 'Plant leaf classification using GIST texture features', *IET Computer Vision*, 13(4), pp. 395–403. doi:10.1049/iet-cvi.2018.5028.
- Naheed, N. *et al.* (2020) 'Importance of features selection, attributes selection, challenges and future directions for medical imaging data: A review', *CMES - Computer Modeling in Engineering and Sciences*, 125(1), pp. 315–344. doi:10.32604/cmes.2020.011380.
- Najmi, A. *et al.* (2022) 'Modern Approaches in the Discovery and Development of Plant-Based Natural Products and Their Analogues as Potential Therapeutic Agents', *Molecules*, 27(2). doi:10.3390/molecules27020349.
- 'Obesity Reviews - 2020 - Miller - Systematic review and meta-analyses of the relationship between short sleep and incidence.pdf' (no date).
- Ojo, J.F., Olanrewaju, R.O. and Folorunso, S. (2017) 'BAYES', (June).
- Papoutsis, G. (2022) 'Creation of an Android Security Training Lab', pp. 1–24. Available at: <https://dione.lib.unipi.gr/xmlui/handle/unipi/14658%0Ahttps://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/14658/Papoutsis%20MTE1927.pdf?sequence=1>.
- Patel, I. and Patel, S. (2019) 'Flower identification and classification using computer vision and machine learning techniques', *International Journal of Engineering and Advanced Technology*, 8(6), pp. 277–285. doi:10.35940/ijeat.E7555.088619.
- Pieroni, A. *et al.* (2020) 'Taming the pandemic? The importance of homemade plant-based foods and beverages as community responses to COVID-19', *Journal of Ethnobiology and Ethnomedicine*, 16(1), pp. 1–9. doi:10.1186/s13002-020-00426-9.
- Prasandika, B. *et al.* (2021) 'Ayurvedic (Herbal) plant identification system Aayu mobile application - Implementation Ayurvedic (Herbal) plant identification system Aayu mobile application - Implementation', (May), pp. 0–42. doi:10.13140/RG.2.2.35700.48009.
- Rama Devi, R. and Abualkibash, M. (2019) 'Intrusion Detection System Classification Using Different Machine Learning Algorithms on KDD-99 and NSL-KDD Datasets - A Review Paper', *International Journal of Computer Science and Information Technology*, 11(03), pp. 65–80. doi:10.5121/ijcsit.2019.11306.
- Rashidi, H.H. *et al.* (2021) 'Machine learning in health care and laboratory medicine: General overview

- of supervised learning and Auto-ML', *International Journal of Laboratory Hematology*, 43(S1), pp. 15–22. doi:10.1111/ijlh.13537.
- Sadgrove, N.J. (2020) 'Southern Africa as a "cradle of incense" in wider African aromatherapy', *Scientific African*, 9, p. e00502. doi:10.1016/j.sciaf.2020.e00502.
- Saeed, S. et al. (2019) 'Analysis of software development methodologies', *International Journal of Computing and Digital Systems*, 8(5), pp. 445–460. doi:10.12785/ijcds/080502.
- Salameh, A. and Bass, J.M. (2022) 'An architecture governance approach for Agile development by tailoring the Spotify model', *AI and Society*, 37(2), pp. 761–780. doi:10.1007/s00146-021-01240-x.
- Santos, I. et al. (2021) 'Artificial neural networks and deep learning in the visual arts: A review', *Neural Computing and Applications*, 33, pp. 121–157.
- Saranya, T. et al. (2020) 'Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review', *Procedia Computer Science*, 171(2019), pp. 1251–1260. doi:10.1016/j.procs.2020.04.133.
- Sarkar, A. et al. (2019) 'Android application development: A brief overview of android platforms and evolution of security systems', in *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pp. 73–79.
- Sekarrini, C.E. et al. (2022) 'Euclidean Distance Modeling of Musi River in Controlling the Dengue Epidemic Transmission in Palembang City', *Open Access Macedonian Journal of Medical Sciences*, 10(G), pp. 422–429. doi:10.3889/oamjms.2022.9125.
- Senarath, U.S. (2021) 'Waterfall Methodology, Prototyping and Agile Development', (June). doi:10.13140/RG.2.2.17918.72001.
- Shah, T.M., Nasika, D.P.B. and Otterpohl, R. (2021) 'Plant and weed identifier robot as an agroecological tool using artificial neural networks for image identification', *Agriculture (Switzerland)*, 11(3). doi:10.3390/agriculture11030222.
- Shao, L., Zhao, Y. and Liu, Y. (2021) 'Organic synaptic transistors: The evolutionary path from memory cells to the application of artificial neural networks', *Advanced Functional Materials*, 31(28), p. 2101951.
- Shi, Y. et al. (2021) 'Recent Advances in the Prediction of Fouling in Membrane Bioreactors'.
- da Silva, L.E.B., Elnabarawy, I. and Wunsch II, D.C. (2019) 'A survey of adaptive resonance theory neural network models for engineering applications', *Neural Networks*, 120, pp. 167–203.
- Skrbo, A. et al. (2022) 'Development of Pharmacy in Bosnia and Herzegovina in the Past - an Overview', *International Journal on Biomedicine and Healthcare*, 10(2), p. 120.

- doi:10.5455/ijbh.2022.10.120-133.
- Sujana, A. *et al.* (2021) 'Data Processing and Plant Identification Based on Computer', *Journal of Physics: Conference Series*, 1933(1). doi:10.1088/1742-6596/1933/1/012051.
- Sun, Y. *et al.* (2017) 'Deep Learning for Plant Identification in Natural Environment', *Computational Intelligence and Neuroscience*, 2017. doi:10.1155/2017/7361042.
- Świechowski, M. *et al.* (2023) *Monte Carlo Tree Search: a review of recent modifications and applications*, *Artificial Intelligence Review*. Springer Netherlands. doi:10.1007/s10462-022-10228-y.
- Takawira, O. and Mwamba, J.W.M. (2020) 'Determinants of Sovereign Credit Ratings: an Application of the Naïve Bayes Classifier', *Eurasian Journal of Economics and Finance*, 8(4), pp. 279–299. doi:10.15604/ejef.2020.08.04.008.
- Tembo, N. *et al.* (2021) 'Ethnobotanical survey of medicinal plants used for cervical cancer management in Zomba District, Malawi', *Scientific African*, 13, p. e00941. doi:10.1016/j.sciaf.2021.e00941.
- Tesfahuneygn, G. and Gebreegziabher, G. (2019) 'Medicinal Plants Used in Traditional Medicine by Ethiopians A Review Article Journal of Respiratory Medicine and Lung Disease', *J Respir Med Lung Dis*, 2(4), pp. 18–21.
- Thompson, P. and Brink, W. (2020) 'Image identification of protea species with attributes and subgenus scaling', *Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020*, pp. 2094–2102. doi:10.1109/WACV45572.2020.9093641.
- Wäldchen, J. and Mäder, P. (2018) 'Machine learning for image based species identification', *Methods in Ecology and Evolution*, 9(11), pp. 2216–2225. doi:10.1111/2041-210X.13075.
- Wang, C.C. *et al.* (2020) 'Implementation a Context-Aware Plant Ecology Mobile Learning System', *SAGE Open*, 10(2). doi:10.1177/2158244020920701.
- Wickramasinghe, I. and Kalutarage, H. (2021) 'Naive Bayes: applications, variations and vulnerabilities: a review of literature with code snippets for implementation', *Soft Computing*, 25(3), pp. 2277–2293.
- Van Wyk, A.S. and Prinsloo, G. (2018) 'Medicinal plant harvesting, sustainability and cultivation in South Africa', *Biological Conservation*, 227, pp. 335–342.
- Xue, J.R. *et al.* (2019) 'Automated Chinese medicinal plants classification based on machine learning using leaf morpho-colorimetry, fractal dimension and visible/near infrared spectroscopy', *International Journal of Agricultural and Biological Engineering*, 12(2), pp. 123–131. doi:10.25165/j.ijabe.20191202.4637.
- Yen, D.C. and Davis, W.S. (2019) 'Rapid application development (RAD)', in *The Information System*

Consultant's Handbook. CRC Press, pp. 247–252.

Zerual, A. *et al.* (2019) 'Using a Fine-Tuning Method for a Deep Authentication in Mobile Cloud Computing Based on Tensorflow Lite Framework', *Proceedings - ICNAS 2019: 4th International Conference on Networking and Advanced Systems* [Preprint], (December 2021). doi:10.1109/ICNAS.2019.8807440.

Zhu, J. *et al.* (2020) 'Automatically processing IFC clipping representation for BIM and GIS integration at the process level', *Applied Sciences (Switzerland)*, 10(6). doi:10.3390/app10062009.



University of Fort Hare
Together in Excellence

APPENDIX A

Plant identifier activity is presented on appendix A. Should there be a need for more code, a request should be submitted, and the required code will be shared.

```
package com.izazi;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.TypedValue;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.widget.TextViewCompat;
import androidx.recyclerview.widget.DefaultItemAnimator;
import androidx.recyclerview.widget.LinearLayoutManager;

import com.github.dhaval2404.imagepicker.ImagePicker;
import com.izazi.databinding.ActivityIdentifierBinding;
import com.izazi.utils.SoftMax;
import com.mikepenz.fastadapter.commons.adapters.FastItemAdapter;
import com.pixplicity.easyprefs.library.Prefs;
import com.yarolegovich.lovelydialog.LovelyStandardDialog;

import org.pytorch.IValue;
import org.pytorch.Module;
import org.pytorch.Tensor;
import org.pytorch.torchvision.TensorImageUtils;

import java.io.File;
import java.io.IOException;
import java.nio.FloatBuffer;
import java.util.Arrays;
import java.util.List;
import java.util.Locale;

public class IdentifierActivity2 extends AppCompatActivity {

    ActivityIdentifierBinding binding;

    Module module;

    List<String> IMAGENET_CLASSES = Arrays.asList("Cima mlilo", "Hlaka-hlakana", "Mhlonyane", "Mpepho", "Mthoba");

    private static final int INPUT_TENSOR_WIDTH = 224;
    private static final int INPUT_TENSOR_HEIGHT = 224;

    float[] mean = {0.485f, 0.456f, 0.406f};
    float[] std = {0.229f, 0.224f, 0.225f};
    private FloatBuffer mInputTensorBuffer;
    private Tensor mInputTensor;

    Uri picUri = null;

    FastItemAdapter<ResultsAdapterItem> itemFastItemAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityIdentifierBinding.inflate(getLayoutInflater());
        View view = binding.getRoot();
```



University of Fort Hare
Together in Excellence


```

setContentView(view);
setSupportActionBar(binding.toolbar);

getSupportActionBar().setDisplayHomeAsUpEnabled(true);
getSupportActionBar().setDisplayShowHomeEnabled(true);

binding.toolbar.setNavigationOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        onBackPressed();
    }
});

itemFastItemAdapter = new FastItemAdapter<>();
itemFastItemAdapter.setHasStableIds(true);

binding.resultsRecyclerview.setLayoutManager(new LinearLayoutManager(this));
binding.resultsRecyclerview.setHasFixedSize(true);
binding.resultsRecyclerview.setItemAnimator(new DefaultItemAnimator());

binding.resultsRecyclerview.setAdapter(itemFastItemAdapter);

binding.buttonCamera.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        ImagePicker.Companion.with(IdentifierActivity2.this)
            //Crop image(Optional), Check Customization for more option
            .cameraOnly()
            .crop(INPUT_TENSOR_WIDTH, INPUT_TENSOR_WIDTH)
            //SaveDir(getExternalFilesDir("Izasi"))
            //compress(1024) //Final image size will be less than 1 MB(Optional)
            //maxResultSize(640, 480) //Final image resolution will be less than 1080 x 1080(Optional)
            .start();
    }
});

binding.buttonGallery.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ImagePicker.with(IdentifierActivity2.this)
            //Crop image(Optional), Check Customization for more option
            .galleryOnly()
            .crop(INPUT_TENSOR_WIDTH, INPUT_TENSOR_WIDTH)
            //compress(1024) //Final image size will be less than 1 MB(Optional)
            //maxResultSize(640, 480) //Final image resolution will be less than 1080 x 1080(Optional)
            .start();
    }
});

if(Prefs.getString("model", "").equals("")){
    new LovelyStandardDialog(IdentifierActivity2.this, LovelyStandardDialog.ButtonLayout.HORIZONTAL)
        //setTopColorRes(R.color.indigo)
        .setButtonsColorRes(R.color.accent)
        //setIcon(R.drawable.ic_star_border_white_36dp)
        //setTitle(R.string.rate_title)
        .configureView(v1 -> {
            View topView = v1.findViewById(com.yarolegovich.lovelydialog.R.id.id_color_area);
            topView.setVisibility(View.GONE);

            TextView messageText = v1.findViewById(com.yarolegovich.lovelydialog.R.id.id_message);
            TextViewCompat.setTextAppearance(messageText, android.R.style.TextAppearance_Large);
            messageText.setTextColor(getResources().getColor(com.mikepenz.materialdrawer.R.color.md_light_primary_text));
            messageText.setAlpha(0.6f);
            messageText.setTextSize(TypedValue.COMPLEX_UNIT_SP, 14);

        })
        //setMessage("Changes were made!")
        .setTitle("No model selected!")
        .setPositiveButton("Select Model", new View.OnClickListener() {
            @Override

```



University of Fort Hare
Togeltho in Enkhabeni


```

        public void onClick(View v) {
            Intent = new Intent(IdentifierActivity2.this, MachineLearningActivity.class);
            startActivity(intent);
        }
    })
    .setNegativeButton("Cancel", new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            IdentifierActivity2.this.finish();
        }
    })
    .show();
} else {

    try {
        File model = new File(getExternalFilesDir("ML"), Prefs.getString("model", ""));
        module = Module.load(model.getCanonicalPath());

        mInputTensorBuffer = Tensor.allocateFloatBuffer(3 * INPUT_TENSOR_WIDTH * INPUT_TENSOR_HEIGHT);
        mInputTensor = Tensor.fromBlob(mInputTensorBuffer, new long[]{1, 3, INPUT_TENSOR_HEIGHT, INPUT_TENSOR_WIDTH});
        module.forward(IValue.from(mInputTensor)).toTensor();
    } catch (Exception e) {
        showToast("Error while loading model!");
        IdentifierActivity2.this.finish();
    }
}

}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    // check if the request code is same as what is passed here it is 2
    if (resultCode == Activity.RESULT_OK) {
        // File file = ImagePicker.Companion.getFile(data);
        // String filePath = ImagePicker.Companion.getFilePath(data);

        picUri = data.getData();
        try {
            Bitmap bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), picUri);
            binding.imagePlant.setImageBitmap(bitmap);

            bitmap = Bitmap.createScaledBitmap(bitmap, INPUT_TENSOR_WIDTH, INPUT_TENSOR_WIDTH, false);

            Tensor inputTensor = TensorImageUtils.bitmapToFloat32Tensor(bitmap, this.mean, this.std);

            Tensor outputTensor = module.forward(IValue.from(inputTensor)).toTensor();
            float[] scores = new SoftMax(outputTensor.getDataAsFloatArray()).getValue();

            itemFastItemAdapter.clear();

            int classIndex = argMax(scores);

            for (int i = 0; i < scores.length; i++) {
                ResultsAdapterItem itemAdapter2 = new ResultsAdapterItem(this);

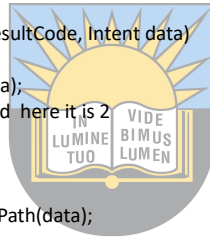
                float c = scores[i] * 100;
                String confidence = String.format(Locale.US, "%.2f", c);
                String className = IMAGENET_CLASSES.get(i);

                itemAdapter2.setResult(capitalizeString(className));
                itemAdapter2.setConfidence(confidence + "%");
                itemFastItemAdapter.add(itemAdapter2);
            }

            String className = IMAGENET_CLASSES.get(classIndex);

            binding.textViewResults.setText("Highest confidence: " + className);

```



University of Fort Hare

Together We Rise

```

        showToast(className);

    } catch (IOException e) {
        e.printStackTrace();
        showToast("Something went wrong!");
    }
}

public int argMax(float[] inputs){

    int maxIndex = -1;
    float maxvalue = 0.0f;

    for (int i = 0; i < inputs.length; i++){

        if(inputs[i] > maxvalue) {

            maxIndex = i;
            maxvalue = inputs[i];
        }

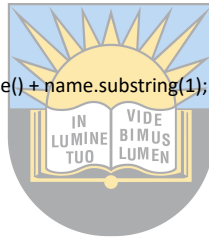
    }

    return maxIndex;
}

private String capitalizeString(String name){
    String capitalizedString="";
    if(!name.trim().equals("")){
        capitalizedString = name.substring(0,1).toUpperCase() + name.substring(1);
    }
    return capitalizedString;
}

void showToast(String s){
    Toast.makeText(IdentifierActivity2.this,
        s, Toast.LENGTH_LONG).show();
}
}

```



University of Fort Hare
Together in Excellence