Convolutional Neural Networks for Robust Fynbos Leaf Classification

Enabling Trustworthy Machine Learning in Botanical Science

Presented by:
Jarushen Govender

Prepared for:
Associate Prof. Simon Winberg
Department of Electrical Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in fulfillment of the academic requirements for a Masters of Science degree in Electrical Engineering

August 11, 2024

Declaration

I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own. This thesis/dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor.

Signature:.....
Jarushen Govender

August 11, 2024

Acknowledgments

Firstly I would like to thank my supervisor, Dr Simon Winberg, for his excellent advice, guidance and enthusiasm during the project. I greatly appreciate the opportunity to carry out research and acknowledge the privilege of being allowed to do so.

I would like to thank my parents Inba and Saroj and my grandfather Da for always encouraging me to study further and supporting me throughout this project. You are missed greatly Da. I would also like to thank my siblings especially Rees for the constant encouragement.

Special mention needs to go to Pieter Wessels, Audrey Staniforth and the rest of the CVM Innovation Team at Nedbank. Studying with full time employment has proven to be one of the greatest challenges I have faced and I would not have been able to accomplish this without their constant support and guidance.

Special thanks to Yasteel Sitaram for daily phone calls and check-ins and contributing to making me the best version of myself.

I would also like to thank Harpreet Singh for his perpetual and unconditional support. You are the only person who truly understood what is like to go through this and the only one who made me believe I could get it done. I'd also like to extend thanks to the others; Shaun, Eric, Darren, Ethan, Chetan, Sayur and Reece.

Next I would like to thank Jash for his encouragement and supportive messages and for keeping me sane during crunch time. I'd also like to thank Tevin and Estrada for encouraging me to push through and finish, as well as the rest of the Park Run Crew for keeping me sane. I'd also like to thank Merusha Naidoo for the motivational talk she gave me last year which inspired the will to push through and finish.

Lastly I'd like to thank the support crew in JHB who have put up with me during the stressful periods; Nerusha, Vicardo, Thashlin, Kimaya, Sanusha, Vinolan, Lishaya, Tereisha, Adrian, Yushen and Keshani.

Abstract

The Fynbos Leaf Optical Recognition Application (or FLORA) is a novel machine-learning tool created for the purposes of aiding conservation efforts of the Cape Floral Region, and the species of plants known as Fynbos in particular. Known for their distinctive evolutionary features, the species maintains a revered position in the ecological heritage of South Africa.

This version of FLORA intends to make use of a Convolutional Neural Network trained on a dataset of collected leaf images, to correctly classify species of Fynbos using natural images as training data. The thesis intends to combat many of the pitfalls of using CNN technology such as working with small datasets and provides a novel approach for dealing with image quality issues and over-fitting that arise from working with limited data. This project also intends to be scalable, and to be able to grow and become more generalised as more training data are added.

The collected data involved manual sample collection using photography equipment and consists of 1,196 field images spread across 35 different species of plants. A part of thesis involved the creation of a novel Image Quality Assessment tool to remove low quality images that negatively influenced the predictive capability of the model.

The model evaluation process makes use of SHapley Additive exPlanations (SHAP), a tool for visualising model predictions, to contribute to the explian-ability of the model and to develop trust and confidence in machine-learning algorithms, with the ultimate aim of providing a tool to merge the fields of ecology, botany and electrical engineering.

Multiple models were trained and evaluated and the selected model for the project obtained a classification accuracy of 76% on the validation data, and an F1-score of 74%. This was an extremely positive result as the training data consisted of exclusively natural images and no feature engineering was performed. The model was then tuned to specific hyper-parameter values which yielded a small performance boost, and then tested on its ability to generalise. Five new classes were added to the training set and the model performance remained consistent, demonstrating robust generalisation. This project contributes knowledge to the growing field of image recognition, and provides a clear framework for model explain-ability which should benefit future research endeavors.

Contents

List of Figures

List of Tables

List of Code Listings

Nomenclature

η	Learning Rate
L	Loss Function
CNN	Convolutional Neural Network
FN	False Negatives
FP	False Positives
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
SHAP	SHapley Additive exPlanations
TN	True Negatives
Top-1	Top-1 Accuracy
Top-10	Top-10 Accuracy
Top-5	Top-5 Accuracy
TP	True Positives

Chapter 1

Introduction

While the mathematical foundations of convolutional neural networks (CNNs) have existed for decades, the ability to implement them marked a significant milestone in the meteoric rise of machine learning and computer vision [5]. These algorithms have become pivotal in the analysis, interpretation and categorisation of many forms of visual data, with unprecedented accuracy and efficiency. This thesis explores the applications of CNNs within the very specific context of classifying species of Fynbos (a diverse and widespread family of hardy plants endemic to South Africa) by images of their leaves. The work in this thesis expands upon previous iterations of the FLORA project, with the CNN approach expected to provide significant upgrades to the overall usability, bringing the project one step closer to becoming a viable tool for use in the ecological conservation efforts in the Western Cape.

Fynbos is a diverse and ecologically significant vegetation type indigenous to South Africa's Cape Floristic Region (CFR) predominantly located in the Western Cape of South Africa. Known, for it's rich biodiversity and unique plant and animal species, this region holds significant cultural, ecological and biodiversity value. Characterised by fine leaves, and diverse colouring and visually stunning flowers, Fynbos comprises roughly 9030 species, of which roughly 70% are endemic to the region [9]. The intricacy and variation present in the phenotypes of different Fynbos species makes identification a challenging task especially when using conventional image recognition methods. In the design of this project it is hoped to both provide a tool to advance botanical research causes and contribute to the overall conservation efforts of this unique and valuable region.

1.1 Background

The background will focus on the historical context of FLORA as a whole, and how this model aims to improve the design. It also focuses on the ecological value of the Fynbos biome to justify FLORA's use a conservation tool.

1.1.1 Project Page

This thesis posits a new version of the currently existing FLORA (Fynbos Leaf Optical Recognition Application) project. The FLORA project was started in 2011 to classify and identify Fynbos plants by using digital images of leaves. The projects are outlined below:

- FLORA-A: Originally designed in MATLAB by this project compared several machine learning algorithms and achieved very high accuracy metrics but was applied to small set of Fynbos species. [10]
- FLORA-B : The initial FLORA methodology was implemented into Python and base work began on transferring the model into a suitable web-app. [11].
- FLORA-C : This model was focused on implementing FLORA into C++ in order to improve the performance. [12].
- FLORA-D: GPU acceleration was trialed to boost performance of the application [12].
- FLORA-E: Key-point matching algorithms were implemented and the web-app design was furthered. The classes were also greatly expanded [13]

This version of FLORA will be dubbed FLORA-CNN due its usage of convolutional neural network architecture.

Analysing the timeline of FLORA from 2011 - 2023 within the context of the machine learning field, several significant developments in image recognition software including the advent of the ImageNet dataset and the consequent development of AlexNet [5]. In addition there have been multiple advancements and refinements within the neural network space itself, as well as further development of many of the algorithms originally attempted - for example Meta released a new form of the k nearest neighbour algorithm

used in FLORA-A, called approximate nearest neighbour which is reportedly much less computationally expensive. There have also been many studies involving applying deep learning to agriculture and botany with positive results and several large scale open-source datasets have been created. However, this does not include Fynbos and a part of this project intends to lay the foundations for large scale image indexing of the plants.

Previous iterations of FLORA focused on using leaf features such as colour, contour, texture and shape in order to classify them. Images were transformed in various ways (such as making them greyscale), and were binarised using contour lines. Using these methods, a level of accuracy of roughly 89% was achieved using a k-nearest neighbour algorithm [10].

Because Fynbos leaves are highly specialized to deal with the unique conditions of the environment that they are in, they display very distinct phenotypic characteristics [14]. This makes them excellent candidates for use in CNN image recognition which relies on layers to identify particular characteristics of images. It also creates challenges as many species share traits and external features.

With this iteration of FLORA, it is hoped it solve many of the limitations of earlier version while also providing a tool for end users that finally allows for an intersection of machine learning, ecology and botany.

1.1.2 Fynbos Plant Species

The Cape Floral Region (CFR) is a great source of heritage for South Africa and the Western Cape. This region consists of truly enormous amounts of biodiversity. The varied and nutrient poor soil of the mountainous region, coupled with the sometimes harsh Mediterranean climate, have created unique evolutionary opportunities for fauna and flora [14] [15]. Fynbos plants are characterized by distinct looking flowers, leaves, stems and roots - all adapted for the the environment - and the biome itself holds roughly 9000 different species of plant with around 70% being endemic [14]. Despite the adaptations, different species of Fynbos can often have startlingly different characteristics.

There are already numerous conservation efforts in action to preserve the biome with current threats emerging in the form of:

- Invasive alien species such as serotinous pines and hakeas which are able to out

(a) *Carpobrotus edulis*

(b) *Protea cynaroides*

(c) *Protea aurea*

Figure 1.1: Pictures representing the type of images collected for this project. The image of *Protea aurea* represents the type of natural image that this thesis is investigating

compete the Fynbos plants for resources, and are better adapted to recover from fires [16]

- Climate change is predicted to severely impact the area in terms of temperature increases which will effect the germination of Fynbos plants which rely on fires for seed dispersal. More intense and more frequent fires could wreck the delicate ecosystem [16].

Conserving the region requires active efforts through dedicated ecological restoration

and management initiatives. The Center for Invasive Biology at Stellenbosch University routinely organises efforts to remove alien species by manually identifying and removing the foreign organisms [16]. A tool like FLORA would be invaluable for use in these conservation operations, requiring little more than smart phone with an internet connection.

Of particular interest to FLORA, would be the eco-tourism niche. Providing a tool not just for botanists, but also for the general public to be able to engage with and celebrate this unique ecological niche, will provide untold value. Society is moving towards a greener future with environmentalism a hot topic in the current political climate. FLORA will allow for the fields of machine learning and botany to merge together, raising awareness and contributing to the conservation efforts of South Africa's ecological heritage.

The research can also contribute to the United Nations Sustainable Development Goal (SDG) 15: Life on Land. SDG 15 aims to protect, restore, and promote sustainable use of terrestrial ecosystems, manage forests sustainably, combat desertification, halt and reverse land degradation, and halt biodiversity loss. Enhanced classification tools can help in the early detection of invasive species, assessment of ecosystem health, and implementation of conservation strategies, thereby promoting the sustainable management and preservation of these ecosystems including the Fynbos biome [17].

1.2 Important Terminology

This thesis adopts a 'first mention - first define' approach but important terms, definitions and jargon will be presented here:

Convolutional Neural Network A variation of neural networks that involve the convolution function. Used widely in image recognition tasks.

Fynbos A type of vegetation native to Southern Africa defined by distinct leaf shape that is adapted to the harsh climate conditions of the area.

Front-end In this context the Front-end is the part of a computer application that end users interact with through an interface.

Back-end This is the part of an application responsible for data management and performing the core functions of the application. It is not visible to the user.

SHAP SHapley Additive exPlanations - This is machine learning framework for explaining

the output of any model by computing the contribution of each feature to the prediction in a consistent way.

Accuracy This measures the proportion of correct predictions to all predictions made by the model.

Sclerophyllous A type of vegetation characterised by large, thick leaves adapted to dry climates.

Backpropagation The key to neural network training which allows the network to adjust its weights based on the error between the output and expected result.

Gradient Descent An optimisation algorithm which is designed to minimise a function by moving in the direction of the steepest descent.

Neuron A computational unit that receives inputs, processes them and creates outputs that are fed into other neurons.

Transfer Learning The concept of using a pre-trained neural network to import weights. This allows the model to start with basic pre-learned features and adapt to new data.

1.3 Motivation and Objectives

The motivation for this project has been briefly touched on thus far but a thorough set of well-defined objectives was created and clarity for the goals ahead will be discussed in this section.

Motivation

The Fynbos biome is an incredibly valuable region in terms of ecological heritage, biodiversity, tourism and natural resources. Numerous conservation efforts exist to preserve the region and the need for standardized documentation and information about Fynbos species is important for these initiatives. Being able to identify a species in the field is a valuable tool for conservationists and botanists. FLORA hopes to provide a readily available tool that is easy to use, but it also hopes to create a large dataset of Fynbos leaves that can be easily extended over time by botanists and other scientists. This indexed database will hope to provide live information on various Fynbos species, their vulnerability, uses, distribution and threats. Using image metadata, it will also be possible to geo-locate

particular species. These uses will position FLORA as an important tool for awareness, tourism and conservation. In addition, with more images, the CNN model used will gain additional training data and will become more robust and capable.

While applications for leaf identification do exist, FLORA will hope to specialise the design for a CNN adapted to the unique requirements of Fynbos leaves and further the field of image recognition. It might be possible to employ transfer learning and adapt a currently existing model, but it would be ideal to create a specialised model for this purpose, and to further grow image recognition software capability with regard to irregular and complex objects.

Objectives

The objective of this project is to build a platform that is capable of storing and indexing images and labels of different species endemic to the CFR and then using this database to train a CNN that is capable of identifying and classifying species of Fynbos from images of their leaves/other features. The architecture of this model will be designed and tuned specifically to deal with the unique phenotypic characteristics of Fynbos leaves. The objectives will be split into primary objectives which will form the core goals of the thesis and secondary objectives which will be features that will improve the overall functionality of the application.

Primary Objectives

1. The creation of a labeled database of Fynbos species to use as training data. Since no current database exists that is readily available online, this dataset will be collected and organised manually. It will also undergo a data quality assessment in which a novel approach is designed to rate the overall suitability of images for training.

2. Conceptualise and design a basic architecture for a CNN that can identify Fynbos leaves that is capable of multi-class classification on the genus or family level. A transfer learning approach will also be used to adapt a currently existing model with the new images. These two models will form a basis of comparison for a more advanced design. These models need to be adapted to working on limited datasets of natural images - no feature engineered images will be used.

3. Optimise the best-performing model through additional functions including more

advanced data pre-processing, data augmentation and hyper-parameter tuning. In total there will be 5 different models created. The model chosen was Optimised Model B.

4. Identify a suitable tool for model explain-ability - or to be able to understand why the model made certain predictions. This thesis makes use of the SHAP tool to accomplish this.

Secondary Objectives

The secondary objective of this thesis is to design a web-portal for the FLORA front-end. The front-end needs to provide a user-friendly interface for end users to upload images of leaves that then get identified. The front-end must also provide a user friendly method for botanists or other domain experts to upload new labeled images of new species that get added to the dataset and which the model retrains on. This will allow the model to organically grow over time and will also allow for the improvement of the CNN as it will gain access to more training data.

Continuous dataset enlargement would both benefit conservation efforts, and the reliability and robustness of the model. This will increase the trust in machine learning within the context of biological sciences, and paves the way for further development and additional tools.

1.3.1 Problem Description

The FLORA project is based around the idea of using machine learning algorithms that have been trained on images of Fynbos leaves to identify species in real time using some form of visual data. Previous iterations of the project have achieved high levels of accuracy for very specific species. However they are not suited to real world use nor are the techniques easily scalable to larger datasets, nor can they autonomously grow.

The aim of FLORA-CNN is therefore to design a platform that can identify Fynbos leaves in real time with a high level of accuracy, but also to be able to learn new data and grow autonomously with minimal human oversight. The platform should allow botanists and machine learning engineers to grow the dataset and hence classification rate over time. The model also intends to be scalable to very large datasets and to cover many species of not only Fynbos, but all botanical aspects of the Cape Floral Kingdom.

1.3.2 Terms of Reference

The objectives outlined in 1.3 are translated into a list of requirements for the FLORA-CNN platform. These requirements are listed below:

R1 The platform must be accessible to end users through a user-friendly interface with all the mathematics and algorithms abstracted away. This section of the thesis involves development of the web-based application. Designing a full stack application greatly adds to the developmental overhead, so the decision was made to just design the front-end.

R2 The platform must be able to perform multi-class classification on leaves with reasonable levels of accuracy. This forms part of the primary objective of the thesis and the majority of time is spent building a model that can do this with a high level of accuracy.

R3 The platform must produce a list of classes and the probabilities of belonging to each class. This requirement feeds into the previous one, and the majority of time was spent on this section.

R4 The models predictions must be explainable to a human. This formed a core aspect of the thesis was accomplished using SHAP visualisations.

R5 The model must be easy to tune. Tuning the model took a considerable amount of time and formed a core experiment to choose the optimal model. Full hyper-parameter tuning requires manually adjusting every parameter, one at a time by training a new model each time. This thesis instead chose only the most important hyper-parameters and tuned them and compared results. This still resulted in multiple new models and considerable time investment.

R6 The image dataset must be properly labeled. This was a key factor to the inherent trust in the model. Input data validation took a significant portion of time and the entire modeling process cannot proceed without this.

R7 The image dataset must be extendable in a user-friendly way while also preserving quality. The model's generalisation is tested and a suitable way to upload images through a user-friendly interface is designed and presented.

R8 The data pre-processing methods need to be standardized and documented. This includes the Image Quality Assessment which formed a core part of the thesis and assisted with improving predicative capability.

With the system requirements defined, a list of functionalities required from the system is also defined. The functionality will form the basis of testing and results

F1 A training dataset of raw images of labeled Fynbos leaves.

F2 Image Quality Assessment to prevent poor quality data from being ingested.

F3 Data-Prepossessing for training dataset.

F4 Data Augmentation for training dataset.

F5 Multi-class classification performed by a baseline model used to establish benchmark results.

F6 Multi-class classification performed by a suitable transfer learning model.

F7 Multi-class classification performed by a more advanced model with a novel attention mechanism.

F8 Multi-class classification performed by a sophisticated model imbued with learnings from all previous designs.

F9 Return multi-class classification results including probability and accuracy metrics.

F10 System should be cloud deployable with suitable back-end and front-end technology.

F11 Expansion of dataset with new labeled data and autonomous ingestion and retraining.

The table below shows the sections of this thesis in which the above functionality and requirements were tested or demonstrated.

Sections	**Description**	**Functions Checked**	**Requirements Tested**
3.2	Dataset of Fynbos Images	F1, F2, F11	R1, R6, R7, R8
3.5	Image Quality Assessment	F1, F2, F11	R1, R7, R8
3.6, 3.7, 3.8	CNN Models Design	F5, F6, F7, F8, F9	R2, R4, R5
3.6, 3.7, 3.8	Data Handling	F3, F4, F11	R4, R5, R8
4.5	Web Application	F10,F11	R1, R2
5.1, 5.2, 5.3	Model Performance	F5, F6, F7, F8	R2, R3, R4, R5
5.6	Autonomous Dataset Expansion	F10, F11	R1, R6, R7, R8

Table 1.1: Tests conducted to evaluate system requirements and functionality.

1.3.3 Scope and Limitations

The scope of this thesis is limited to the design and application of CNN architecture for image recognition. Other algorithms will not be considered nor used as a form of comparison.

The following limitations apply:

- No large scale open-source dataset of Fynbos images exists currently based on leaves specifically. The database construction and image collection phases of this project therefore represent a significant portion of the time commitment. The 'Garbage In - Garbage Out' mindset is applied which resulted in significant time being devoted to data handling.
- Only CNN architecture is considered. No other image recognition algorithms are considered.
- The front end and back end commitment will only consider basic functionality and will exclude advanced HTML, CSS and Javascript as this will contribute to the development overhead. The full back-end functionality will not be implemented due to the excessive development overhead that will derail the rest of the project.
- Endangered species of Fynbos that are not easy to collect samples for will be omitted, with the functionality to add them being designed.
- While the main focus of the project remains the Fynbos, other species make up the CFR and samples collected for these species will be used in the training as the thesis hopes to contribute to the conservation of the biome as a whole. Additional classes will also make the model more robust.
- The collection dataset consists of > 1500 images in total which is relatively small for CNN training. This limitation will influence design choices which will optimise for low training data.
- Advanced hyper-parameter tuning techniques like Particle Swarm Optimisation, Bayesian Optimisation and Genetic Algorithms are discussed but not used in the thesis as they would greatly increase the resources required and do not meaningfully impact the goal of the thesis.
- Due to the a tiny dataset size, a test set is not explicitly kept aside.

1.3.4 Dissemination Plan

The plan to disseminate all research material is presented below:

1. All code used in this thesis including model architecture, saved models, the Image Quality Assessment, data pre-processing, SHAP plots, statistics and data visualisations and the front-end design is available on Github at `https://github.com/Jarushen/FLORA-CNN`. The list of required libraries is listed in the tutorial.
2. This thesis which includes the model design and experiments performed to obtain a full model will be made available and a suitable, well-indexed journal will be written to summarise it.
3. The thesis will be marked and thoroughly peer-reviewed and its suitability will be assessed.
4. The end stage would be to deploy the full web application of FLORA and begin growing the training set

1.3.5 Document Outline

Chapter 2 - Literature Review This presents a thorough literature review that will provide suitable context to expand on the introduction focusing on the Fynbos Biome, CNNs used for image recognition tasks and the context of CNN architecture and data handling as well as a look at current CNN applications in botany.

Chapter 3 - Methodology This section presents the proposed CNN architecture with justifications for usage. It will detail the stages of creating baseline models, accuracy metrics, and then additional model refinements to compare against. It will also discuss the numerous data strategies including the Image Quality Assessment.

Chapter 4 - FLORA System Design This section presents the various experiments that the models were tested on as well as the web application.

Chapter 5 - Results This section presents the results of each iteration of the project and provides empirical, quantitative and qualitative discussion of the results and discusses

options for improvement. This section will also contain a section on explain-ability using the SHAP tool for each model as well as the results of the experiments proposed.

Chapter 6 - Conclusion This section provides conclusions to the project and presents possible improvements and refinements for future developments.

Chapter 2

Literature Review

The primary objective of this literature review is to establish a foundational understanding of the current state of research with regards to the applications of convolutional neural networks (CNN's) within botanical image recognition, with a specific focus on the identification of Fynbos leaves.

2.1 Introduction

The introduction will briefly discuss the main components of the literature review.

2.1.1 Outline of the Review

The review will proceed as follows:

- The first section will focus on creating an understanding of the current state of the Fynbos biome and its ecological significance. The key to the whole project lies within the foundational knowledge of the ecology and is born out of respect for the CFR as a whole. This section will focus on the unique evolutionary traits that Fynbos plants have adopted to survive within their environment and explain what evolved features makes them suitable for image recognition algorithms [9]. A portion of this section will also discuss the ecological value of Fynbos and why a project like this will aid in conservation efforts.

- After establishing a solid contextual base of the ecology the review will move onto **developing an understanding of the technical foundation of a CNN application**. It will unpack the various layers including the input layer, the convolutional layers (or filters), the pooling layer, activation functions, the fully connected layers and the output layer. Concepts like **gradient descent** and **backpropagation** will also be explained and explored as well as any other terminology. The specific focus will of this chapter will be to apply the concepts to Fynbos images and extra notes pertaining to this need will be made where applicable.

  This section will also explore the application of CNNs to image recognition specifically, talking about the advantages and also some of the challenges that are experienced with these models. Additionally a model to use for transfer learning will be decided from this section.

- With a solid foundational base, the review will then explore some **current applications of CNNs in botanical research** with a focus on plant identification. A summary will be provided showing the results, focus of research and potential applications will be created of each example. The advantages and disadvantageous of these particular use cases will also be provided. This section will serve as starting point for the architectural design based on the results observed in other models.

- The next section will focus on contextualising existing research. To my knowledge the previous iterations of FLORA are the only existing research base for applying image recognition software to Fynbos leaves but there have been applications to other types of plants. Each previous iteration will be explored, and significant learnings will be extracted.

- The next section will focus on the data handling. Precise data handling is perhaps the most important step in optimizing the performance of CNNs and this section will hope to guide the more robust designs of FLORA-CNN by looking at any cutting edge data handling research that is available, with specific focus on botanical applications.

  The review will explore the importance of data quality, annotation and labeling, data augmentation, normalisation and standardisation, image resizing and reshaping and how the model will go about splitting data into test, train and validation sets. This section will be used extensively in the upgraded versions of the base model as well as the transfer learning model.

- Finally the review will wrap everything up by identifying gaps in current research methodologies and provide opportunities for further research. A key objective of this review is to identify the advances that have already been made with regards

to CNNs being used in botanical identification, particularly in environments which are as complex and diverse as the Cape Floral Region. It is equally important to understand any gaps or limitations in the current research methodology which may present opportunities for further innovation both in this thesis and in future.

2.1.2 Background on Fynbos Biome and Leaf Morphology

The Cape Floristic Region (CFR) is one of only six floral kingdoms in the world and compromises a land surface area of 90,000 square kilometers which as is less than 5% of the total African Continent [18]. However, this tiny region consists of an enormously wealthy diversity of plant species with an estimated 9030 different species of vascular plants, of which 8920 are flowering plants and of which 6500 are endemic to the area [18]. For comparison, the Amazon rain forest encompasses an area of 6.7 Million square kilometers and is estimated to host around 50,000 plant species [19]. Based on these numbers, the CFR roughly has 0.1 different species per square kilometer while the Amazon rainforest has roughly 0.007 different plant species per square kilometer. In other words, the plant biodiversity per square kilometre of the CFR is roughly 14 times higher than that of the Amazon Rainforest!

A key feature of the CFR is the varied terrain and unique climate. The sandstone derived soils that geographically define the area have a low nutrient density which has shaped the unique evolution of Fynbos and thus their distinct features [9]. The region is characterised by mountains composed almost entirely out of nutrient poor quartz, interspersed with more nutrient dense granite and limestone, which has likely contributed to the high biodiversity of the region [9], [18], [20], [3]. Due to the very specific conditions, plants in the CFR have some common specialisations with special mention of leaf structure:

- Sclerophyllous leaves – Leathery and long-lived leaves which reduce nutrient loss and provide a degree of water stress resistance. This feature is usually associated with leaves that are $< 10\,\mathrm{m}^2\,\mathrm{kg}^{-1}$ [18], [9].

Figure 2.1: An example of a sclerophyllous leaf belonging to *Protea neriifolia*

- Small leaf size – small leaf sizes are very distinct amongst Fynbos species [18]. Small leaves reduce the boundary layer thickness which facilitates heat loss when conditions are hot and dry. Small leaves promote heat loss in summer and promote water loss in wet periods which aids nutrient acquisition. Since these features would be highly beneficial for the region they live in, a remarkable amount of biodiversity is seen in this trait ranging from needle-like leaves to short and flat leaves such as those belonging to *Oscularia deltoides* seen below [18], [15].

Figure 2.2: An example of small leaf size belonging to *Oscularia deltoides*

- **Thick Cuticles and Waxy Surfaces** – The leaves of some Fynbos plants display thick cuticles and waxy surfaces to facilitate a reduction in water loss and to provide protection from sunlight and fire [18].

- **Phenotypic Plasticity** – Certain species Fynbos plants seem to exhibit a high degree of phenotypic plasticity in their leaf morphology which means that the same species might have different leaf characteristics depending on the specific environmental conditions it grows in. Such plasticity was observed in species like *Olea capensis* which showed large variation in cuticle properties between forest and Fynbos environments but not between different micro-habitats within these areas [18].

- **Nutrient Concentration** – Because of the nutrient poor soils discussed above, the Fynbos plants have evolved leaves that possess high nutrient concentrations particularly nitrogen and phosphorus which are needed to support growth [18].

- **Specialized Leaf Structures** – Some Fynbos species such as *Drosera capensis*, have evolved specialised leaf structures for capturing and digesting insects, an adaption seen in carnivorous plants. This strategy supplements nutrient intake in the harsh environment [21].

- **Fire Adaptation** – The leaves of many species are adapted to survive or to quickly regrow after fires, which commonly occur in this region. These features include protective leaf sheaths or the ability to sprout rapidly from underground structures post-fire [18]. Furthermore there seems to be evidence that certain Fynbos leaves are able to detect smoke which often initiates germination [9], [15]. This fire adaption also represents a problem in which Fynbos leaves can look phenotypically diverse during different times of the year [9]. This trait would pose difficulty to models training on images collected at particular times.

These adaptions are crucial for the plants to survive and thrive in the biome and are a result of complex interplay between the plants and their environment. This makes them a fascinating subject of study for ecologists and botanists [18]. The unique phenotypic characteristics of the leaves also makes them particularly suitable for use with image recognition technologies and algorithms that excel at feature recognition. The shared morphology also poses a challenge to algorithms as species might share traits resulting in mis-classification which poses a threat to any autonomous algorithm [15].

2.2 Conservation and Study Importance

Introduction The Cape Floral Kingdom is not only a matter of academic interest, but also a critical area for conservation, ecological tourism and heritage. This section will focus more on Fynbos through the lens of conservation, highlighting the unique challenges it faces and the imperative to preserve the ecological value of the area. This thesis hopes to generate a use-case to stimulate the ecological awareness of the area, and thus is vitally important to understand the context of fitting everything together.

Global Biodiversity Significance

The Fynbos biome itself, as noted earlier, holds a very unique position in global biodiversity. It represents roughly 9030 plant species of which 70% are endemic [9]. It is also one of the world's 25 most threatened hotspot – with a hotspot being defined as an area with a high concentration of plant and animal life that is under the threat of extinction should development continue at current pace. It is estimated that 1736 Fynbos plants are critically endangered, endangered or vulnerable [9]. The biodiversity as discussed earlier, is the result of the unique environmental conditions including the soil type and dry, mountainous terrain [18].

Over 59,2% of species fall into the ten largest families of which the largest are the Asteraceae and Fabaceae families (which together consist of 20% of the total species) [3]. Asteraceae, the largest family, are usually found in arid to semi-arid regions. The top 20 largest families consist of 77,4% of total species - 6989 species in total. The top 20 largest families are presented in the table below:

Family	**Number of Members**
Asteraceae	1036
Fabaceae	761
Iridaceae	677
Aizoaceae	659
Ericaceae	657
Scrophulariaceae	414
Proteaceae	329
Restionaceae	318
Rutaceae	273
Orchidaceae	227
Poaceae	207
Cyperaceae	206
Hyacinthaceae	191
Campanulaceae	183
Asphodelaceae	157
Geraniaceae	157
Polygalaceae	141
Rhamnaceae	137
Crassulaceae	134
Thymelaeaceae	125

Table 2.1: Top 20 Fynbos Species Count which represents 77.4% of all Fynbos adapted from [3]

Members of these families are almost all evergreen, sclerophyllous shrubs and often have summer flowering with small flowers and typically grow in sandstone-derived soils which are nutrient poor [18]. The shrubs are diverse but usually include species with sclerophyllous and mostly micro-phyllous leaves but the species do show variations. For example Proteaceae, a family common in the region, have broad sclerophyllous leaves whereas many of the Ericaceae shrubs have narrow and needle-like leaves [3].


Figure 2.3: The large sclerophyllous leaves can be clearly seen on *Protea neriifolia* (right) contrasted against the small needle like leaves of *Erica arborescen* (left)

This also introduces one of the limitations of this model, namely that it will be biased towards more commonly seen species as pictures of these species will be more readily available and more accessible. This represents a concern as one of the aims of this project is to aid with conservation efforts and endangered or rare species will end up neglected just by virtue of availability.

The suggestion here is to also potentially focus on the most well known families of Fynbos - namely Asteraceae, Ericaceae and Proteaceae - in the data collection process since over 77.4% of all species occur in the largest 20 families and these species are more well represented in collection sites [3]. This bias towards certain species in studies and media is noted [22]. The consequence of this decision will be neglecting endangered or rare species which goes against the philosophy of conservation, but FLORA-CNN intends to be a tool that gets refined over time. A good starting point would be to start with common species for the initial training set to allow the model to learn general Fynbos features as discussed in 2.1.2, and expand the set later to capture rarer species.

Threats to the Fynbos The Cape Floral Kingdom faces several threats that endanger the biodiversity of the region. Urban expansion, agricultural development and invasive alien species that are better adapted to certain conditions all contribute to habitat loss. Climate change additionally could potentially destabilize the delicate needs of the biome with the CFR predicted to become hotter and drier in coming years [16]. The West Coast forelands area for example is predicted to experience higher temperatures and lower rainfall [9]. This will have a severe impact on fires in the region which will affect the germination of Fynbos plants, particularly if those fires are more intense (defined as the rate at which energy is released during combustion) and occur during different seasons [23]. Lightning is a common cause of firs, and altering thunderstorm conditions

could increase the amount and severity of fires [22]. This would directly affect Fynbos germination cycles.

Direct revenue sources are also generated directly from the Fynbos through the harvesting of rooibos tea, wildflowers reeds and other traditional and commercial medicinal plants [16].

Conclusion

Contextualising the significance of the Fynbos biome resulted in the following learnings:

- A high quality image dataset is needed of high resolution and well-lit Fynbos leaves which should capture the leaves from different angles and in different lighting conditions. There should also be a variety within species to cover different possible phenotypes. This dataset should focus on the main families - and only species with distinct leaves - in terms of representation with the intention to refine this over time. In this way it is ensures that FLORA-CNN's value as a conservation tool is still preserved, while still leaving open the possibility to adapt it at later dates.
- Fynbos leaves might have unique textures, colours or other features that are important for the classification algorithm. The CNN should be designed to preserve these details, possibly through high-resolution input layers or architectures adept at capturing fine details. For instance, deeper networks with more layers might be needed to capture the complex patterns in Fynbos leaves. However, deeper networks require more data and computational resources.
- Fynbos leaves can vary in shape, size, color, and texture. The network should be robust to these variations, which can be achieved through a diverse training dataset and potentially by incorporating techniques like transfer learning.
- The data collection needs to minimise intra-class variability to prevent the model from being biased towards more common species. The class imbalance can also manifest if too many images from certain classes are deemed low quality (the images captured are cast in a shadow for example).
- Many species share evolutionary characteristics and can appear similar. This will result in mis-classifications by the model due to inter-class similarities, or the sharing of learned features which can cause mis-classification across multiple classes.

2.3 Fundamentals of Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision and image analysis, emerging as a cornerstone technology in machine learning. These networks are specifically designed to process and interpret visual information and have been used successfully in many aspects of society such as autonomous driving and facial recognition. In recent years there have been many applications to biological research including botanical research and medical imaging [24]. The key advantage of these models is their ability to automatically detect features in data [2].

The fundamental concept of the CNN is the **neuron**. A neuron in this context is determined by its **weights** and **bias**. Weights are the parameters that transform input data within the network's architecture. Each neuron in the network receives one or more inputs, and each of these inputs is multiplied by a weight. These weights represent the strength or influence of the input on the neuron's output [5].

The inputs are multiplied by the weights, summed together, and then a bias is added. The result of this sum is then passed through an activation function, which determines the output of the neuron. A bias is an additional parameter in the neural network that is added to the weighted sum of the inputs to a neuron. The bias allows the neuron to shift the activation function to the left or right, which is essential for best fitting the given data [25].

The activation function introduces non-linear properties to the network, enabling it to learn complex patterns. Common activation functions include the sigmoid, hyperbolic tangent, and ReLU (Rectified Linear Unit) [1]. A simplified view of neurons is presented below in **??**:

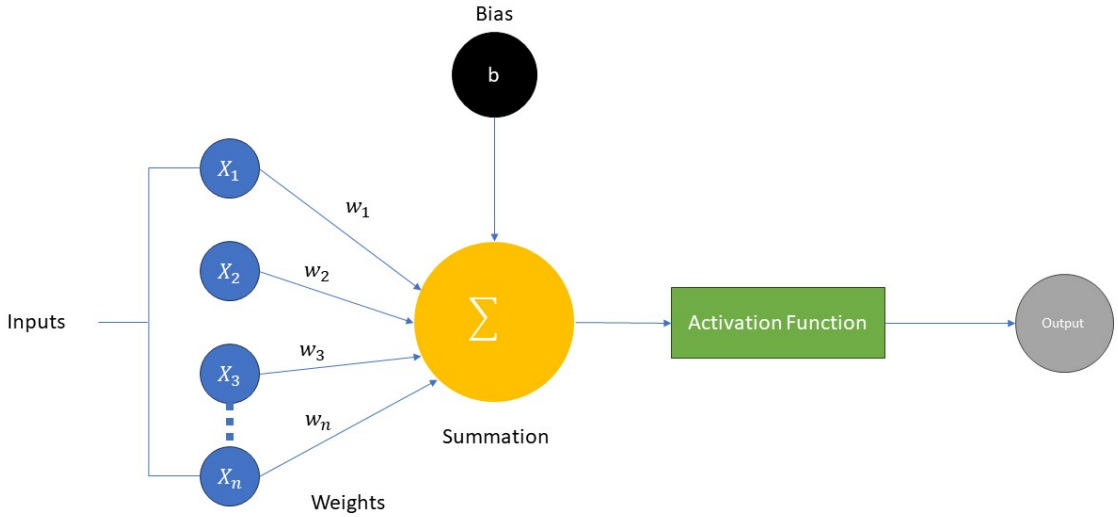


Figure 2.4: Example of neurons with weights and biases summed and being fed into an activation function

The full architecture typically consists of convolutional layers, pooling layers and fully connected layers all made up of neurons. Small, learnable filters (or kernels) then slide over input images to perform convolution operations which allows the capturing of spatial features and patterns in the data such as lines, edges, textures and more complex and deeper structures [24], [2].

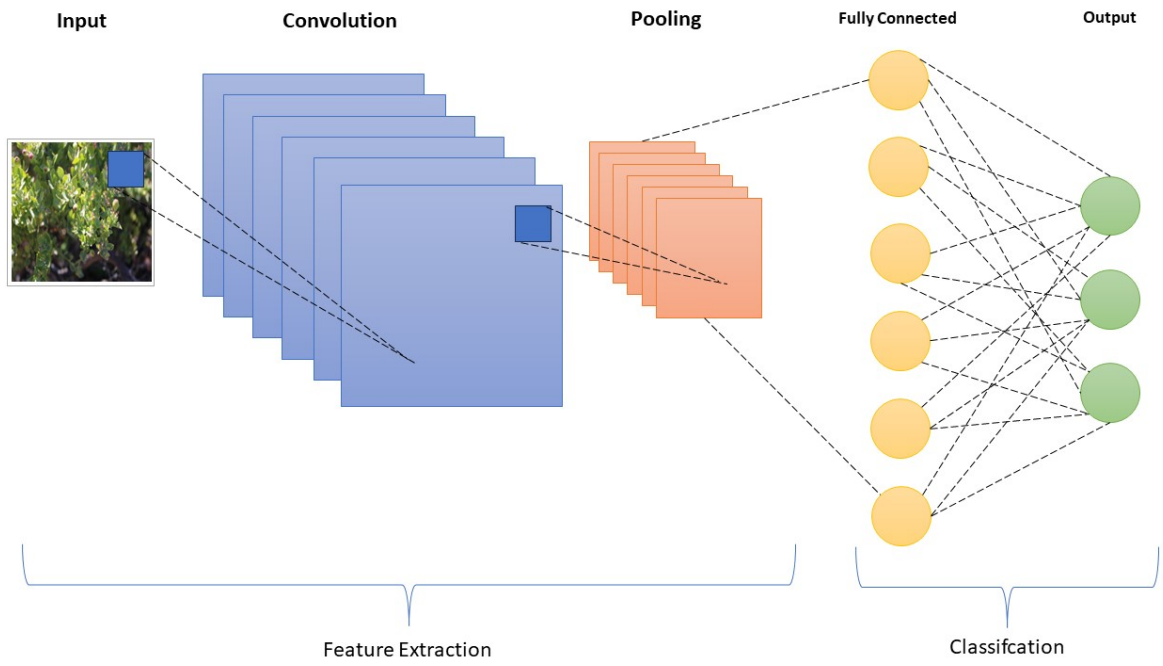


Figure 2.5: Example of Basic CNN Architecture

The convolutional layers are inter-spaced with pooling layers which reduce spatial dimensions of the data, decreasing computational requirements. As data passes through the layers (the feed forward process) it becomes increasingly more abstract and representative of features within the input image. The final stage of the CNN is typically composed of fully connected layers that use the extracted features from the earlier layers to perform classification [2].

Training CNNs involve the concepts of **backpropagation** and **gradient descent**. Backpropagation is a method used to train the neural network in which the error between the predicted output and actual output (the loss function) is sent back through the network which helps to adjust the weights of the neurons in order to minimize the error. Gradient descent is an optimisation algorithm used alongside backpropagation to retroactively update neuron weights by finding the gradient that minimised the loss function [26]. The main goal of training a neural network is to find the optimal set of weights that minimize the error in predicting the output [24].

Finally, there exists one output neuron for each object category in the output layer. The output of the classification part is the classification result and in this case will determine which class the leaf image belongs to [6].

Using these concepts, CNNs are able to automatically adapt and learn spatial hierarchies from input images and the need for manual feature extraction is greatly reduced. This

represents a significant benefit over the previous FLORA models such as the models created by [10] and [12]. This capability makes CNNs highly efficient and accurate for various image processing tasks, and it makes them useful for specific images such as Fynbos leaves. This review will expand upon these core concepts using relevant research where applicable to build a sound theoretical base for modeling a CNN [24]. One of the key challenges of this thesis is a limited dataset so the lens of this review will be that of applying a CNN to limited data.

2.3.1 Origins and Evolution

The history of computer vision is both fascinating and enlightening. In a 1959 paper titled 'Receptive Fields of single neurons in the cat's striate cortex' by D.H. Hubel and T.N. Wiesel , the researchers, were able to determine that neurons in the brain of an anaesthetised cat, fired when the cat first detected simple structures such as oriented edges [27]. This would form the basis for the field of computer vision, but the application would not be recognized until decades later. The architecture of deep convolutional neural networks is inspired idea of receptive fields existing within cells, whereby cells actually respond to summation of inputs from other local cells. Each cell is able to recognize a particular feature of the image (such as an edge or colour), and together, the brain is able to use these inputs to recognise an image.

It would be a lengthy process to name every single contributor to the field from that point until the present day, but other noteworthy researchers include Kunihiko Fukushima - inspired by Hubel and Wiesel - who, in 1979, developed the basis of the modern convolutional neural network in a model called Neocognitron. The artificial network involved self-organising simple and complex cells that were able to recognise patterns by using rectangular layers whose receptive fields had weight vectors. Neocognitron is used for Japanese handwriting recognition and can be considered the grandfather of modern convolutional neural networks [28]

Following this development in 1989 was Yann LeCun who applied the concept of backpropagation algorithms to Fukushima's architecture. LeCun eventually released LeNet-5 which was also used for character recognition and incorporates many of the fundamental characteristics of CNNs used today [29],[4]. This leads to the papers that form the basis of the algorithms used for this project which were primarily written with regards to the Imagenet Large Scale Visual Recognition Competition. Imagenet is a large dataset of images (over a million images representing over 1000 classes) which helps to prevent over-fitting when

training on it[5], [30].

There are several variations and improvements available such as VGG-16[31]. This literature review will provide a brief summary of **transfer learning** - the concept of using an already trained model and adapting it to new data - with the expectation to adopt a transfer learning as one of the model iterations [5].

2.3.2 Architecture of CNN's

It is great to laud the performance of the Imagenet results but without a clear understanding of how and why CNNs work, the development and improvement of models becomes a matter of trial and error [32]. CNNs are different to most image recognition approaches as they combine both feature extraction and classification. This section will build upon the concepts presented by 2.3. Each feature of this model will be reviewed for basic functionality, any new developments, applications to botanical research, optimization options, applications to small datasets and any other relevant information.

The goal of this design is going to be to create an architecture that is optimised for classifying images of leaves and also well adapted to working with limited datasets. Each part of the CNN will be expanded on in this section and relevant terminology explained. This section forms the theoretical basis of the thesis.

Input Layer

The process begins with an input image. When layers a mentioned, they refer to a series of neurons with similar functions. In this case the input image will be a picture of a leaf. The input layer (the leftmost layer) represents this image which can be thought of as a grid of pixels. Red, Green and Blue (RGB) images are used as inputs, so the input layer is said to have three channels ranging from a value of (0 to 255). A value of 0 means none of that colour is present while 255 means the maximum of that colour is present [33]. Two features in the forms of **brightness** and **hue** are stored in the value of each pixel [24]. It should be noted that the input for non-image based models will take a different shape.

The input layer requires a fixed size for the image. For this to be true, the FLORA images will need to go through some form of pre-processing which will form a large part of this thesis and will be reviewed in section 2.3.8 [8]. RGB images get treated as three

dimensional arrays with each dimension representing the red, green and blue components of the pixels in the image. This allows the CNN to extract features based on colour patterns [32]. An image from the collected dataset with it's RGB channels is presented below:

Figure 2.6: Example: *Protea cynaroides* from collected dataset

Figure 2.7: Red Channel

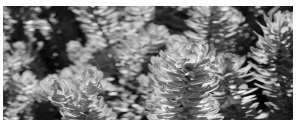


Figure 2.8: Green Channel

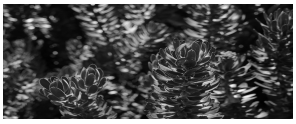


Figure 2.9: Blue Channel

The input layer is dependent on the image being fed into it. These images will need to undergo pre-processing in order to be meaningful for the model. There is a large range of pre-processing techniques available and they will be vital to the model. They will be reviewed in 2.3.8.

Convolutional Layer:

The **convolutional layer** is the core building block of the CNN and it represents the biggest contribution to computation. Convolution works by dividing the input image into small slices. This division is what helps in extracting features [2].

This layer essentially performs a dot product between two matrices. The first matrix is the input matrix from the input layer and the second matrix is a spatially smaller matrix called a kernel (or a filter) [5]. Filter and kernel will be used interchangeably in this thesis. The kernel scans over the image matrix and computes the dot product between the kernel values and the pixel values that it covers (the process of convolution, hence the name). This process creates what is known as a **feature map** that highlights the

presence of specific features in the image [33]. Mathematically the the whole process is represented by:

$$Conv(F, I))(x, y) = \sum_{m} \sum_{n} \sum_{d} F(m, n, d) \cdot I(x + m, y + n, d) \tag{2.1}$$

where:

- $Conv(F, I))(x, y)$ represents the convolution operation with $Conv$ being the convolution function
- I is the input image
- F is the kernel/filter
- $F(m, n, d)$ represents the kernel at position (m, n, d). The kernel is a small window that slides over the input image, m and n iterate over the spatial dimensions of the kernel and d refers the depth dimension which corresponds to the number of channels in the input image (e.g. RGB channels) which will be 3 in our case.
- $I(x + m, y + n, d)$ refers to the element (or more generally the number of layers in a feature) of the input image at position $(x + m, y + n, d)$. As the kernel slides over the image $(x + m, y + n)$ goes on to represent the position on the input image which corresponds to the filter element $F(m, n, d)$. The d informs us of which is channel being processed
- The convolution operation involves summing over the dimensions m, n and d which means the corresponding product of the kernel and input image over all spatial dimensions and depth, and then the summation of these products to get a single value at each position in the output feature map. This is the dot product.

Visualizing this process is presented below in **??** using an image from the dataset:

71	48	76	131	146	67	96	138	69	96
99	99	63	139	150	65	86	116	79	138
66	68	50	146	157	140	119	91	72	115
98	41	104	127	144	138	167	82	65	94
128	30	100	94	129	139	168	138	107	73
125	43	127	92	110	142	114	133	119	64
140	147	165	127	102	134	94	109	96	90
119	153	143	119	102	133	88	117	101	66
110	80	125	121	124	134	104	120	81	41
106	36	93	126	125	133	99	105	60	49

Figure 2.10: An image from the dataset downsized to 10x10 for demonstration purposes and its pixel representation on the right. In reality the resolution of the image is much larger and contains many more pixels.

The chosen kernel is then applied. This can be thought of as a representation of some type of feature that would exist in the image. This kernel then 'scans the image as shown below and computes the dot product as it goes. This will create another matrix (the feature map) with a list of dot products. Intuitively this shows that a high dot product will indicate the presence of a feature and a low dot product will indicate that it is not there.

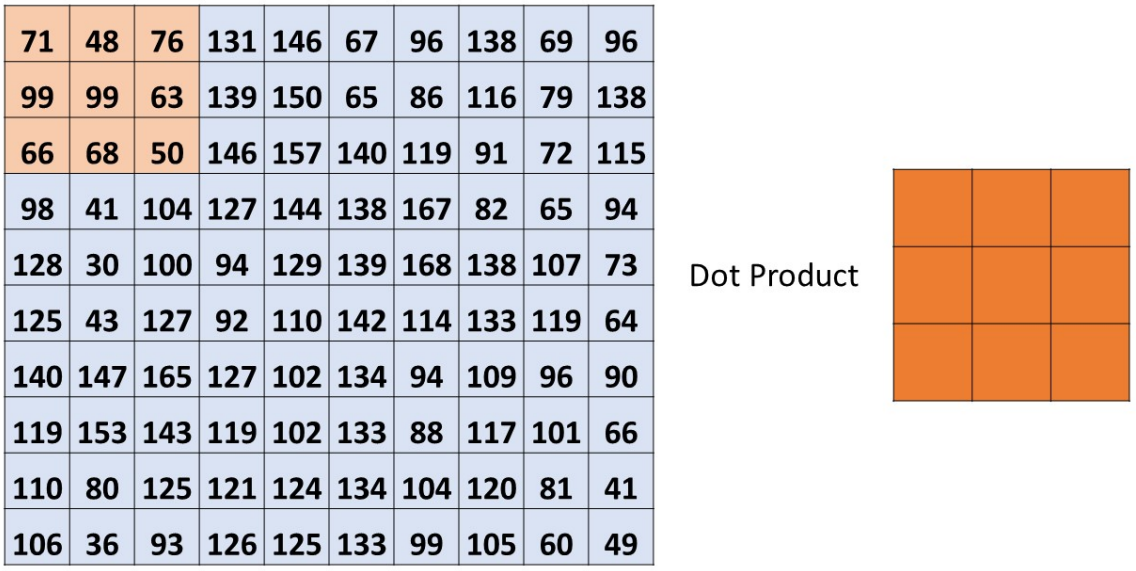


Figure 2.11: Example of kernel applied to receptive field

There are two additional hyper parameters in this layer that need to be explained: **stride** and **padding**.

Padding: One of the problems noted above is that values at the corners and borders do not have equal weighting as the values in the center of the matrix This will help giving equal weighting to features that occur in these regions of the image as is done by adding another row and column to all sides of the matrix with just the value zero in it [34]. **Stride** on the other hand refers to the number of pixels by which the kernel moves across the input image. All the above depictions assumed it had a value of 1 - where every pixel in the input image is involved in the convolution operation and which leads to a detailed feature map. Making it > 1 will result in the kernel skipping pixels as it moves across the input image which results in a smaller feature map and reduces the spatial resolution of the output. larger strides help to reduce the size of the output and the computational requirements [34].

Visualizing the kernels is challenging but can be done using a pre-fit model. The VGG-16 model presented in 2.3.4 and used at the ImageNet Large Scale Visual Recognition Challenge [35] can be applied to one of the collected images. The model is reasonable to use for visualisation as it has a uniform structure of serially ordered convolutional and pooling layers [35]. It has 16 learned layers. This visualisation can be performed using Python and the code is available in the Github repository.

Learned kernels are actually just weights, but because of the two-dimensional structure of filters the weight values have a spatial relations [5]. This means that each kernel can be plotted as a two-dimensional image and extract some meaning. The convolutional layers are designed in the VGG-16 model to use 3x3 kernels which are small and make them easier to interpret [6]. Something to be noted here with regards to architecture is that the depth of the kernel first needs to match the depth of the input to the kernel. For example with an input image that has three channels for red, green and blue, each kernel will have a depth of three.

After applying basic image pre-processing to scale down the image to the size the mdoel needs and normalising values to make them easier to output, the first six kernels from the first convolutional layer in the VGG-16 model can be shown:

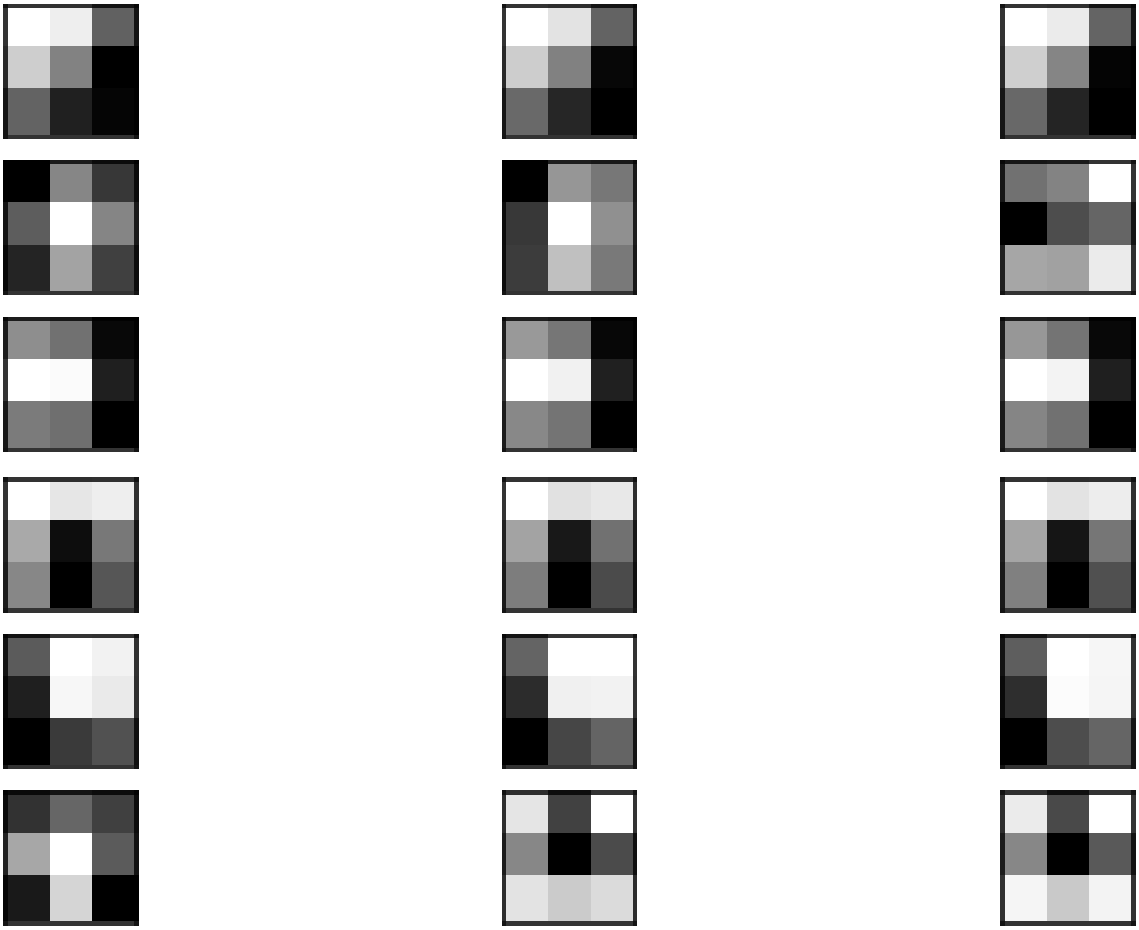


Figure 2.12: Filter Visualisation

The example creates a figure of 18 images, one row for each filter and one column for each input channel (Red, Green and Blue). It is seen that in some cases the kernel is actually the same across the channels, but in the others the filter differs. The dark squares indicate small weights and the white squares represent large weights.

Intuitively this means that the kernels on the first row are detecting a gradient from light to dark (top left to bottom right).

It becomes difficult to perform visualation for a second layer as the scale becomes unmanageable. In the second layer, each of the 64 filters above will have 64 channels to match which will result in 4096 plots.

However something similar can be done to visualise feature maps. The idea behind this would be to try and understand what features of the input image are preserved in the feature maps with the expectation being that the feature maps close to the input would

detect smaller details and the the maps closer to the output would capture larger, more general features. Again, after some basic pre-processing, VGG16 can be modified to look at these.

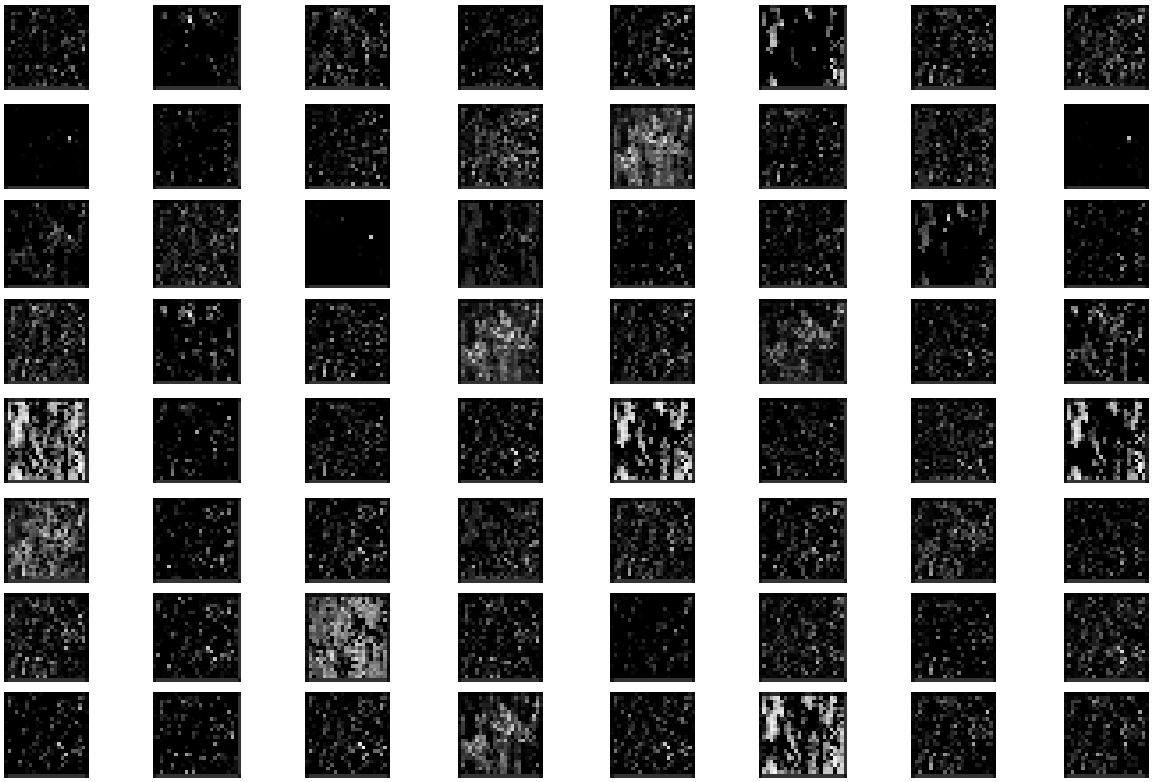


Figure 2.13: Feature Map Visualisation - First Convolutional Layer Output. Basic shapes are identified and extracted

It is seen that after applying the filters in the first convolutional layer, multiple versions of the Fynbos image from 2.3.2 are outputted with different features selected. This generally matches expectations. Progressing through the layers shows less and less detail.

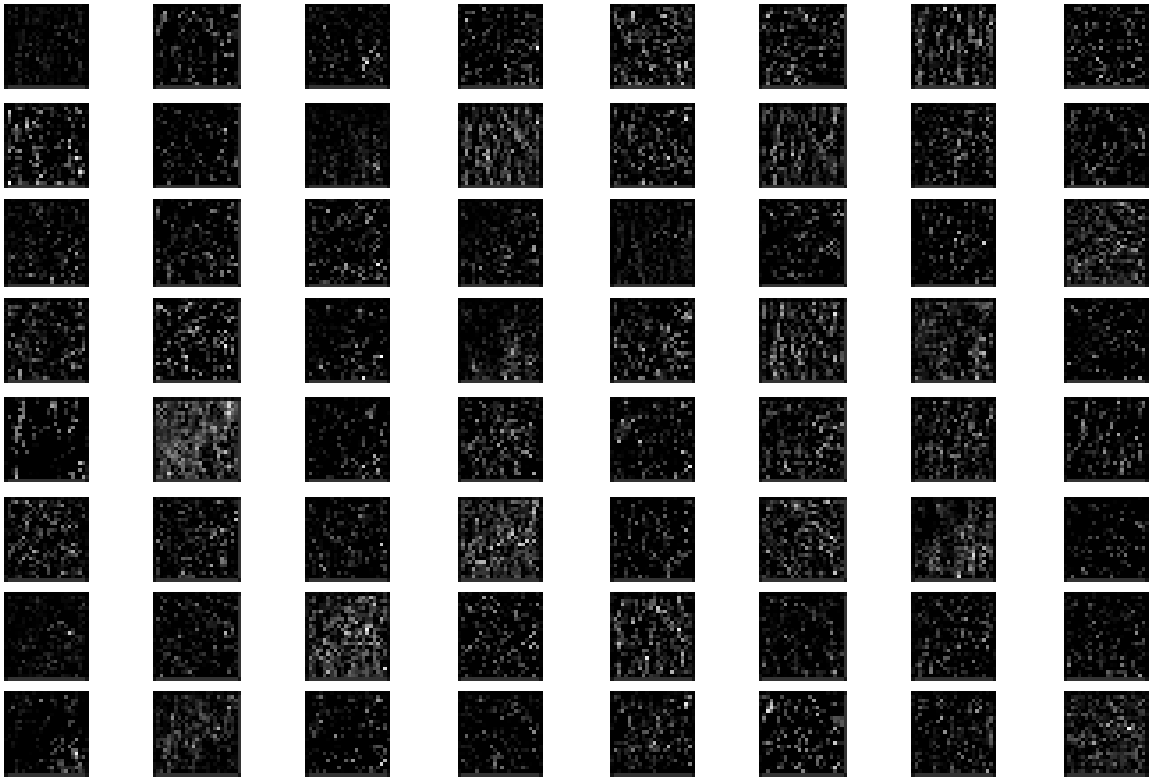


Figure 2.14: Feature Map Visualisation - Second Convolutional Layer Output. Simple shapes are seen but the maps are becoming more abstract

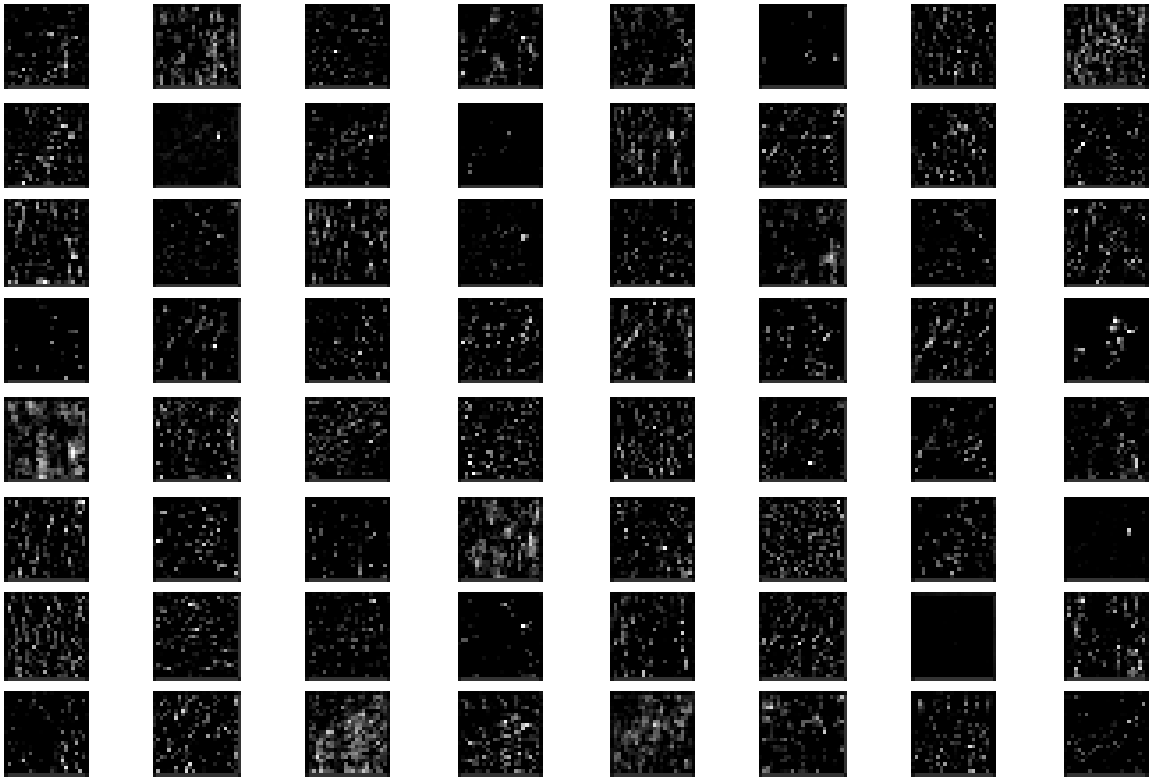


Figure 2.15: Feature Map Visualisation - Third Convolutional Layer Output. There is much more abstraction

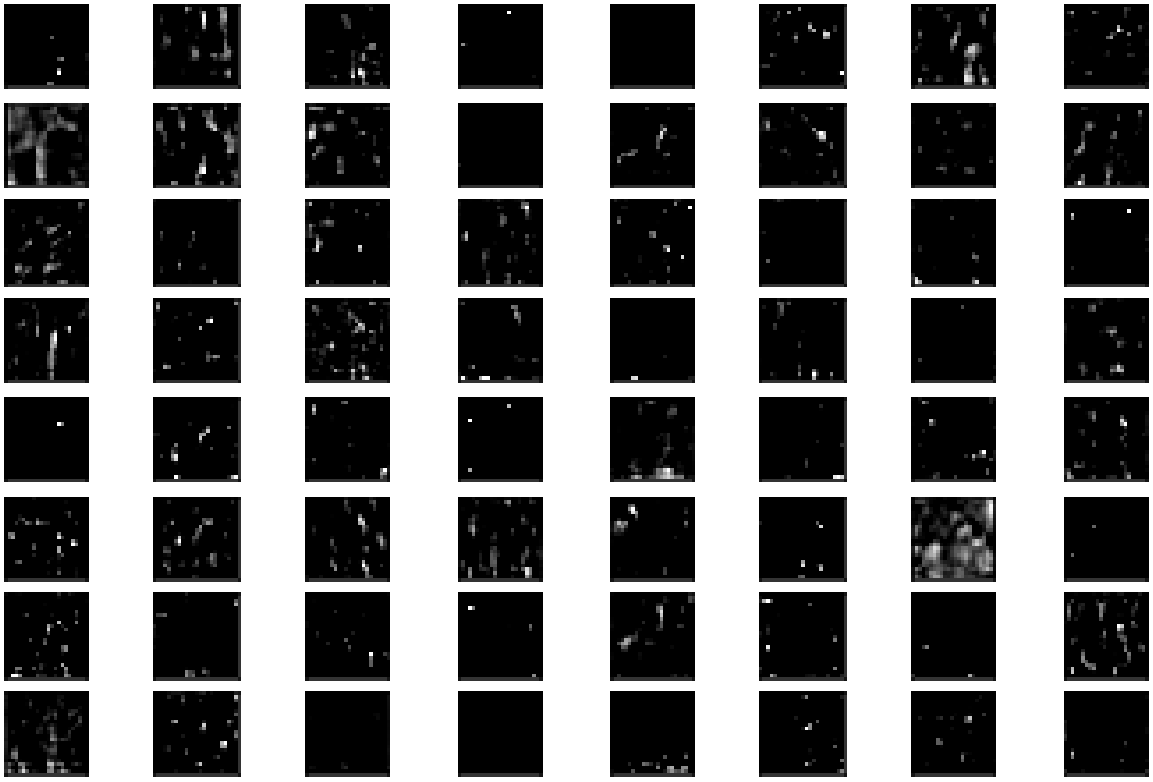


Figure 2.16: Feature Map Visualisation - Fourth Convolutional Layer Output. These maps are much more complex that the first one

The consequences of this will be important when designing the FLORA model but it can be sees that the first layer extracts simple features of the leaf, such as its shape. The fourth layer is extracting much more complex and abstract patterns that can't be interpreted. This is what was originally demonstrated for the LeNet-5 model discussed in 2.3.4 [4].

Activation Function The second component of CNN architecture to discuss is activation functions as they play a significant role in the architecture. Activation functions decide whether a neuron should fire or not, essentially deciding whether or not that neuron's input is important. They are used to introduce non-linearity- a key component of neural networks - without which the network will be unable to learn complex patters and relationships that might exist with in the data. In most applications of CNNs in literature, not much weight is placed on anything other than ReLU. Older models suffered from something known as the vanishing gradient problem and so were replaced with ReLU which protects against this [36] [4].

Activation functions enable **backpropagation** which is short for "backward propagation

of errors" [5]. This process allows the gradient of the loss function to pass through the layers which is essential for model training as it adjusts the weights based on the error. Backpropagation is the core concept of neural network training. It is based on the error rate of the previous epoch (an **epoch** is an iteration of model training).

Backpropagation works by sending the gradient (of the loss function) calculation "backwards" through the neural network [37]. The gradient of the final layer of weights is calculated first and the gradient of the first layer is calculated last with partial calculations getting reused as in the gradient calculation of the previous layer. Instead of naively calculating the gradient of each layer separately the backwards flow of the error allows for efficient computation of the gradient at each layer as there is a better starting point [24].

This is where activation functions come in. The chain rule is used in backpropagation in deep neural networks. Since there are many layers in large neural networks, should the gradient of any activation functions be less than 1, then the gradient at a layer connected to it but further from the outputs will be close to zero. If the gradient becomes close to zero for any layer, the process of backpropagation stops in the preceding layers. This is known as the **vanishing gradient problem** [1]. This kills the ability of the model to learn features. The most common types of activation functions are presented below:

- **ReLU (Rectified Linear Unit):**

$$\text{ReLU}(x) = \max(0, x) \tag{2.2}$$

  The range of ReLu is 0 to ∞ with gradient for positive and negative inputs being one and zero respectively.

  This is one of the most widely used activation functions. It performs a threshold operation where an input value less than zero gets set to zero, and any positive value is left unchanged. This function helps to mitigate the vanishing gradient problem which is a key concern for any deep learning [1]. This is the activation function that was used in the AlexNet model [5]. Several advancements have been proposed to deal with the negative values, including the Leaky Rectified Linear Unit (LReLU) which has an output range of $+\infty$ and $-\infty$. There is also Parametric ReLU which considers the slope for a negative input to be a trainable parameter [1]. It is also observed that networks with ReLU train several times faster than their equivalents with tanh functions [5].

  When using this function it is advisable to be mindful of the Dying ReLU Problem. If the weights in the network end up always leading to negative gradients, the ReLU

functions will only output zeroes . This problem needs to be especially monitored for in small datasets with complex layers [38].

- **Sigmoid/Logistic**

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

This function squashes values between 0 and 1 which makes it suitable for binary classification task and is commonly used in the output layer where the neural network has to predict the probability as an output. This function is susceptible to the vanishing gradient problem because it has a saturated output [1]. It has generally been replaced by ReLU in modern models, but several improvements have been proposed such as the Parametric Sigmoid Function and the Rectified Hyperbolic Secant. However both functions are still susceptible to the vanishing gradient problem [1].

- **Tanh (Hyperbolic Tangent):**

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.4}$$

Similar to sigmoid but with an output range from -1 to 1 instead. It's zero-centered, making it easier in some cases for the model to learn. The same drawbacks as the sigmoid exist [1].

- **Softmax:**

$$\text{Softmax}(O)_j = \frac{e^{O_j}}{\sum_k e^{O_k}} \tag{2.5}$$

Used in the output layer of a neural network for multi-class classification, it converts raw scores to probabilities. The Softmax function will be used on FLORA's output layers [1].

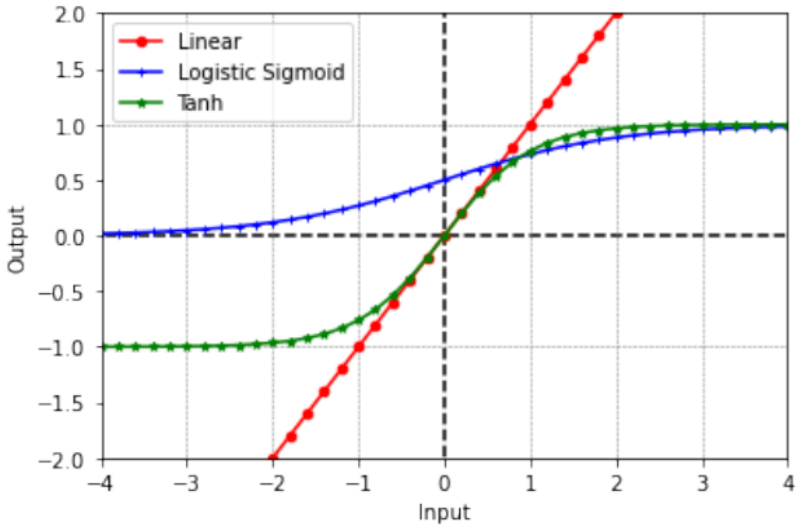


Figure 2.17: Linear, Sigmoid and Hyperbolic Tangent Activation Functions

In terms of implementation and the choice of activation function, it would make sense start off with ReLU in the convolutional layers and avoid the logistic or hyperbolic tangent functions as the models need to avoid the vanishing gradient problem. Performance can be attuned from there.

Pooling Layer

The convolutional layers generate feature maps. Once these features are extracted, their exact location becomes less important provided that the approximate position relative to other features is preserved [25]. In the pooling layer the spatial size of these maps is reduced. This is essential to reduce the number of parameters in the network and makes the network computationally efficient. Output feature maps are sensitive to the positioning of features in the input and one way to deal with this is to down sample the feature maps. Pooling layers provide an approach to sample the feature maps by summarising the presence of features in patches of the feature map [26] [37].

After non-linearity has been applied via the activation function, the pooling layer can be added. This addition of a pooling layer may be repeated one or more times in a given model and in the more complex models shown in 2.3.4, many, many pooling layers are needed to keep the computation manageable [5], [6].

A pooling operation - very similar to a kernel - gets applied ro reduce dimensionalality. In most cases this involves 2x2 pixels applied with a stride of 2 pixels [26].

A pooling layer applied to a feature map that is 10x10 (100 pixels) will result in an output pooled feature map of 5x5 (25 pixels) as an example, or a reduction by a factor of a 75

The pooling functions needs to be specified. The three main pooling functions will be considered:

- Average Pooling: Calculate the average value for each patch on the feature map. This method was used in earlier models (LeNet-5 for example [4]) but has generally fallen out of favor in most recent research dedicated to image recognition as Max Pooling just performs better at feature detection [5].

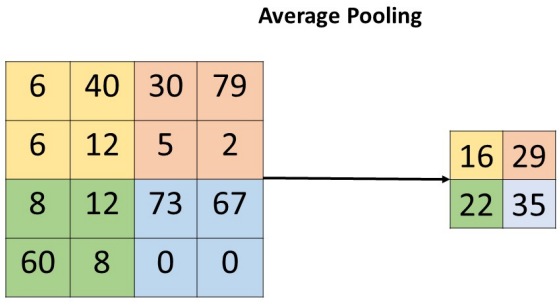


Figure 2.18: Average Pooling calculation. Notice reduced dimensionalality

- Maximum Pooling (or Max Pooling): Calculate the maximum value for each patch of the feature map.

$$MaxPooling(I)(x,y) = \max(I(x,y), I(x,y+1), I(x+1,y), I(x+1,y+1)) \quad (2.6)$$

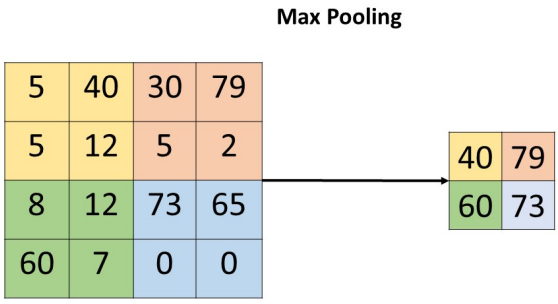


Figure 2.19: Maximum Pooling calculation. Notice reduced dimensionality.

- Global pooling: Global pooling down samples the entire feature map to a single value. This pooling function is used sub-parts of some deep models such as the Google Inception Model which places this pooling layer between the last fully connected layer and the output classification [8]. There is Global Average Pooling and Global Max Pooling. Global Pooling tends to focus on the most significant features and is vital in many newer models to reduce over-fitting which makes many models very viable [39]. Global average pooling used in this way eliminates the use of fully connected layers which significantly reduces computations and the number of learnable parameters.

After using the pooling function and creating the pooled feature maps, the result is a summarised version of the features that were detected in the input. This solves the problem of small changes in the locations of features having large impacts on the outputs. This capability added by pooling is called the model's invariance to local translation. Maximum Pooling is the selected method for baseline models as it is best used for image recognition tasks since it emphasises the locations of features [26]. However one of the core issues observed in the results was over-fitting so the more advanced models will attempt to make use of global average pooling [39].

The convolutional layer and the pooling layer form what is known as the textbfConvolutional Block. Simple architectures generally consist of at least three convolutional blocks [2]. With the more advanced models such as ResNet-50 containing 16 convolutional blocks with each block containing multiple layers [40]. This is explored more in 2.3.4.

Batch Normalisation

Batch normalisation is a technique used in neural networks to improve training efficiency and stability and to reduce over-fitting [41]. It is used to address the issues that arise with covariance shift within feature maps. The internal covariance shift refers to network layer inputs changing as the network's parameters are during the process of backpropagation. This shift can be problematic, especially in deep networks, where small changes in earlier layers can significantly affect the distributions in deeper layers. [25].

The process of batch normalisation (or whitening) involves the following steps adapted from [42]:

1. The mean and variance of the activation of each input layer need to be computed for each mini-batch.
2. The activations are normalized by subtracting the mean and dividing by the standard deviation. This ensures that each feature's distribution has zero mean and unit variance.
3. The normalized values are then scaled and shifted using two parameters, gamma and beta, which are learned during the training process. These parameters allow the network to undo the normalization if that is what the learned behavior requires.

Batch normalization typically occurs before the activation function in a network layer, although it can also be applied after the activation function in certain cases. This is usually dependent on the activation function used. for sigmoid and hyperbolic tangent functions, it is normally applied after the function, and for ReLU it is normally applied before the function [41].

Batch Normalisation has been shown to severely assist with model over-fitting, reduce computational demands and contribute to overall model performance [41]. It is usually more effective in deeper models, and it has been shown that it can both replace dropout and work in parallel with it to reduce over-fitting. It is a parameter that will be considered for the modeling process and experimented with. It has been shown to produce good results in terms of optimisation even with small batch sizes which is vital for FLORA-CNN [43].

Dropout Dropout is a regularization technique widely used in neural networks [36]. Its main purpose is to prevent over-fitting, which is a common problem in deep learning where the model performs well on training data but poorly on new, unseen data. This

will be a focus point for this thesis as one of the problems FLORA will encounter is over-fitting due to the limitations of the dataset.

During the training of a neural network, dropout randomly deactivates a subset of neurons in the network. This is done with a specified probability, commonly set to 0.5. This means that each neuron has a 50% chance of being excluded from a particular round of training. The key idea behind this is to prevent neurons from co-adapting too much, encouraging them to learn features independently and thus making the model more robust [36].

Dropout tends to be more effective in scenarios where the training data is limited, as these are situations where over-fitting is more likely to occur. In contrast, with very large datasets, the benefit of dropout might be less pronounced [36].

Fully Connected Layer

The fully connected layer is important for the model's ability to make predictions and classifications based on the previous layers.

The fully connected layer takes the high level-features like edges, lines or textures from the pooling layer and learns non-linear combinations of them, essentially synthesizing the information extracted from the convolutional and pooling layers [25].

In this layer, every neuron is connected to every neuron in the previous layer which contrasts with the convolutional layer where the connections are localised. The fully connected layer maps the learned features representation to the final output such as the classes in a classification task. In an image classification task, the fully connected layer would have as many layers as the number of classes with each neuron representing the likelihood of a specific class [44].

Before passing the output of the convolutional layers to the full connected layers, the data is usually 'flattened' into a single vector. This process converts the two-dimensional output of the previous layers into a format suitable for ingestion by the fully connected layers [44].

Output Layer

The output layer in a CNN represents the final output classifications of the network.

Typically, it is a fully connected layer, and the number of neurons in this layer corresponds to the number of classes in a classification task. For instance, in a network designed to classify images into different categories, each neuron in the output layer would represent a specific category. The multi-class classification is achieved through a soft max activation function set to the number of output classes in the data [37].

The output layer synthesizes all the features and patterns learned by the network through its convolutional and pooling layers, and presents them in a form that can be interpreted as specific classifications or predictions.

2.3.3 Applications in Image Recognition

With the advent of the iPhone in 2005 and the introduction of Big Data, numerous advances in the image detection field have been seen largely in part to the applicability of CNNs to be able to work with this data [45]. This has been present in many fields such as facial recognition, medical image analysis, remote sensing and handwriting recognition [46], [4]. In this section the advantages and disadvantages of using CNNs for image recognition will be discussed.

Advantages

- As seen in section 2.3.2, CNNs can automatically detect and learn important features without any human intervention. This is major advantage as feature engineering of input images was a major stumbling block for previous iterations of FLORA as it prevented the models from scaling [10]. Feature engineered models are also brittle and their performance greatly depends on the features selected.
- Because of the pooling functions, CNNS are not sensitive to image variations as discussed in section 2.3.2. This makes them effective for real world applications where such variations are expected [26].
- CNNs have consistently outperformed traditional image recognition methods in terms of accuracy, particularly in complex tasks like recognizing subtle features in medical images or distinguishing between similar objects. They have even achieved high accuracy in botanical image classification tasks [47].
- Transfer learning: CNNs trained on large datasets can be adapted or fine-tuned for specific tasks. This transfer learning ability saves time and resources as models do

not need to be built from scratch [48]. It also lowers the likelihood of over-fitting caused by training.

Disadvantages and Challenges

- CNNs require a large amount of labeled training data to perform well. In cases where data is scarce, this can be a limiting factor. This is a key consideration for this thesis and various data augmentation strategies will need to be investigated.
- Training CNNs is computationally intensive and may require high-end hardware in the form of GPUs. Cloud computing options can mitigate this somewhat as can transfer learning models [46].
- CNN models, like many deep learning models, are often considered "black boxes" due to their complexity, making it hard to interpret how they arrive at certain decisions. It is possible to apply explain-ablity frameworks like SHAP to address this [49].
- While CNNs are less prone to over-fitting due to their architecture, it's still a risk, especially when training on small datasets. This will be a problem that is encountered in this model and appropriate care needs to be taken to address this such as incorporating features like dropout and batch normalization [36], [42].

2.3.4 CNN Architecture Strategies

CNNs have evolved significantly since over time but attempts to classify these improvements in terms of parameter optimisation, regularisation or structural changes but the largest contribution to to CNN improved performance seems to come from restructuring the processing units and designing new blocks [25]. This section will look at common models that have proven very successful and will dissect what made them so well adapted.

LeNet-5

LeNet-5 was pioneering research performed in 1998 that showed the applications of CNNs to the task of handwriting recognition [4]. The architecture of the CNN is presented below:

A summary of the architecture is presented in the table below:

Table 2.2: LeNet-5 Architecture

Layer	**Feature Map**	**Size**	**Kernel Size**	**Stride**	**Activation**
Input	Image	1	32x32	-	-
1	Convolution	6	28x28	5x5	tanh
2	Average Pooling	6	14x14	2x2	tanh
3	Convolution	16	10x10	5x5	tanh
4	Average Pooling	16	5x5	2x2	tanh
5	Convolution	120	1x1	5x5	tanh
6	FC	-	84	-	tanh
Output	FC	-	10	-	softmax

The architecture of LeNet-5 was originally designed for digit recognition and offers a useful blueprint for feature extraction using convolutional layers. The original model was applied to 60,000 image samples, albeit of relatively simplistic images of handwritten digits [4]. For Fynbos leaf classification, the model can be fine-tuned to detect the key morphological features outlined in 2.1.2 such as leaf shape, leaf clustering or other Fynbos features. The initial layers will capture the basic textural and structural information shown in 2.13 which would be crucial for differentiating between leaf structures. The subsequent layers would combine the basic features to identify more complex patterns unique to Fynbos morphology such as the waxy or leathery textures as an example.

By today's standards LeNet-5 is a simple model with limited depth and will likely require modifications to capture the more intricate leaf details required for FLORA-CNN. A proposal to improve this model is to incorporate an **attention mechanism**, which were inspired by the selective focusing ability of LeNet-5 [50], [51]. These mechanisms can take the form of a 'Squeeze-and-Excite' function [52].

LeNet-5 also uses relatively small feature maps by the standards of modern models [4]. Based on the summary table in 2.2, the feature map size would be 28X28X6 from the first convolutional layer. While this design choice makes perfect sense for digit recognition, the higher resolution images from the collected samples have many more details, and larger features maps are needed to preserve spatial information in the model. Due to data limitations, techniques to overcome over-fitting such as dropout or batch normalisation would be required [36]. The activation functions used in LeNet-5 are hyperbolic tangent functions in the convolutions layers and a softmax function in the output layer. The softmax function was necessary for multi-class classification but the hyperbolic tangent

functions should be replaced by a ReLU functions or equivalent to avoid the vanishing gradient problem [51]. For Fynbos leaf images, it would make more sense to replace average pooling with max pooling as it works better preserve image features [26]. Should the model complexity increase it might be better to consider global average pooling instead [8].

The model architecture of LeNet-5 provides a valuable and explainable starting point, but FLORA-CNN would need various modifications based on more recent research. The main limiting factor is that LeNet-5 was not designed to work with complex, high resolution images. However it could still be adapted for leaves. It was shown that a classification accuracy of 89.58% was achieved on the task of leaf identification with minor adjustments to LeNet 5 including updating the activation functions to ReLU [53]. However the model was not tested on unseen data and is likely to be suffering from over-fitting.

AlexNet and Transfer Learning

AlexNet represented a major breakthrough for CNNs in 2012 when it won the ImageNet large Scale Visual Recognition Challenge. The architecture used demonstrated incredible ability to learn complex representations from much more complex image data than LeNet-5 [5]. For FLORA-CNN, AlexNet architecture could provide a suitable framework, albeit with some adaptations. AlexNet architecture is more complex than LeNet-5 but the summary table of its layers is presented below in 2.3:

Table 2.3: AlexNet Architecture

Layer	**Type**	**Feature Maps**	**Size**	**Kernel Size**	**Activation**
Input	Image	-	227x227x3	-	-
1	Convolution	96	55x55x96	11x11	ReLU
2	Max Pooling	96	27x27x96	3x3	-
3	Convolution	256	27x27x256	5x5	ReLU
4	Max Pooling	256	13x13x256	3x3	-
5	Convolution	384	13x13x384	3x3	ReLU
6	Convolution	384	13x13x384	3x3	ReLU
7	Convolution	256	13x13x256	3x3	ReLU
8	Max Pooling	256	6x6x256	3x3	-
9	Fully Connected	-	4096	-	ReLU
10	Fully Connected	-	4096	-	ReLU
11	Fully Connected	-	1000	-	ReLU
Output	Softmax	-	1000	-	-

AlexNet is characterised by deep architecture with multiple convolutional layers followed by max-pooling layers and fully connected layers [5]. It also makes use of **dropout regularisation** discussed in 2.3.2. The network is able to capture a variety of complex features which makes it suitable for FLORA-CNN. The Imagenet dataset also does contain labeled images of trees, leaves and crops [30].

The deep layered approach could be customised to capture leaf morphology with the convolutional layers being adapted to the leaf morphology from 2.1.2. However it would require adaptions. The original AlexNet was designed for images of dimension 227x227 pixels [5]. The input layer will need to be adjusted to accommodate the size and resolution of images in the collected dataset to ensure that the critical morphological details are preserved.

Because AlexNet was trained on 1.2 Million images [5], the direct applications to FLORA-CNN need to be looked at through the lens of the limited Fynbos dataset. FLORA-CNN needs to mitigate over-fitting and data augmentation needs to be performed. While the AlexNet results are spectacular, the considerations of the model were not specialising on one particular type of image. There were 1000 image classes, and in the output layer, need to be modified to the number of classes in the FLORA-CNN collected dataset.

AlexNet does however offer a suitable candidate for **transfer learning**. Transfer Learning refers to a technique where a model is initially developed for one particular task and is then later adapted for a secondary but related task [54]. It is often used for CNNs where a model is trained on a particular dataset and, then gets used on a new, smaller and more specialised dataset. The new model is initialised with weights that have been pre-trained on much larger and more diverse datasets such as the ImageNet dataset. Transfer models allow for faster convergence and better generalisation [54]. Transfer learning allows the model to use pre-learned features and adapt them to new datasets. For small datasets, this is valuable and has been demonstrated in 2.3.5.

Networks that have already learned to recognise leaf morphology, can apply this to distinguish between Fynbos leaves which will shorten the training time and grants the model superior feature-detection ability [54]. It would typically be required to fine tune parts of the architecture and careful tuning.

For FLORA-CNN it might be more useful to make use of transfer learning (such as using AlexNet) at least as a base model. Transfer learning approaches in botany have been used successfully in literature to achieve excellent results. Refer to 2.7 for the findings of these papers.

VGG-16

The VGG-16 network was developed in 2014 and was a logical evolution of AlexNet in the timeline of CNN development [6]. The model makes use of consecutively stacked 3x3 convolutional filters which offer offers the ability to capture increasingly complex features through the 16 weight layers. A summary of the architecture presented in the original study is presented below in 2.4:

Table 2.4: VGG-16 Architecture

Layer	**Type**	**Feature Maps**	**Size**	**Kernel Size**	**Activation**
Input	Image	-	224x224x3	-	-
1	Convolution	64	224x224x64	3x3	ReLU
2	Convolution	64	224x224x64	3x3	ReLU
3	Max Pooling	-	112x112x64	2x2	-
4	Convolution	128	112x112x128	3x3	ReLU
5	Convolution	128	112x112x128	3x3	ReLU
6	Max Pooling	-	56x56x128	2x2	-
7	Convolution	256	56x56x256	3x3	ReLU
8	Convolution	256	56x56x256	3x3	ReLU
9	Convolution	256	56x56x256	3x3	ReLU
10	Max Pooling	-	28x28x256	2x2	-
11	Convolution	512	28x28x512	3x3	ReLU
12	Convolution	512	28x28x512	3x3	ReLU
13	Convolution	512	28x28x512	3x3	ReLU
14	Max Pooling	-	14x14x512	2x2	-
15	Convolution	512	14x14x512	3x3	ReLU
16	Convolution	512	14x14x512	3x3	ReLU
17	Convolution	512	14x14x512	3x3	ReLU
18	Max Pooling	-	7x7x512	2x2	-
19	Fully Connected	-	4096	-	ReLU
20	Fully Connected	-	4096	-	ReLU
21	Fully Connected	-	1000	-	ReLU
Output	Softmax	-	1000	-	-

Adapting it to the needs of FLORA-CNN, VGG-16 can be used to capture more granular features which would be essential for differentiating leaf structures. It is a series of convolutional layers, each followed by a ReLU activation function[6]. The pair or triplets of convolutional layers are followed by max pooling layers which reduce spatial dimensions

[26]. This design ensures that more complex features are learned progressively as the model deepens. The uniform kernel sizes of 3x3 allow for optimisation of pattern recognition, which is particular useful for Fynbos characteristics.

VGG-16 was originally trained on the ImageNet dataset, it can be transferred to more specialised tasks and has been successfully implemented in plant disease detection [55]. It does however require serious fine tuning, typically in the last few fully connected layers so that it aligns with the new dataset which might be drastically different to the original data VGG-16 was trained on.

The structure of VGG-16 is relatively uniform and straightforward [6]. This simplicity does come with a serious computational cost as there are a large number of parameters, particularity in the fully connected layers. It has approximately 138 million parameters [6]. Practical application of this without access to super computers, will require careful consideration.

Res-Net

ResNet utilizes residual blocks with skip connections to allow the training of networks that are substantially deeper than previous models. This addresses the vanishing gradient problem and allows ResNet models to be made much deeper [7]. There are multiple versions of ResNet with different depths so a simplified summary is presented below in 2.5:

Table 2.5: Simplified ResNet-50 Architecture

Layer Block	**Output Size**	**Layer Type and Size**	**Repetitions**
Conv1	112x112	Conv, 7x7, 64, stride 2	1
Conv2_x	56x56	Conv, 3x3 max pool, stride 2 Residual block, 3x3, 64	3
Conv3_x	28x28	Residual block, 3x3, 128	4
Conv4_x	14x14	Residual block, 3x3, 256	6
Conv5_x	7x7	Residual block, 3x3, 512	3
Average Pool	1x1	Average pool, 7x7	1
Fully Connected	-	FC, 1000 (num classes)	1
Softmax	-	Softmax	1

There are different versions of ResNet but they all incorporate the skip functions which

allows the gradient to bypass one or more layers. This allows the signal to be preserved through many layers without attenuation [7]. While training deeper networks is generally correlated with a higher capacity to learn more detailed features and would be particularly beneficial for the needs of FLORA-CNN, deeper networks being applied to small datasets can also lead to significant over-fitting [48].

Implementing ResNet for leaf classification would also need careful consideration of the variants available - such as ResNet-50, ResNet-101, and ResNet-152 - which have different strengths and trade-offs. This choice would need to be made being aware of the dataset limitations and available computation ability. Additionally, the transfer learning approach, where a pre-trained ResNet model is fine-tuned on the Fynbos dataset, could expedite the training process and lead to improved generalization, given the similarity between the pre-training tasks and the target classification task [54]. Pre-training tasks are the original tasks or datasets used to train ResNet, before it is then adapted to a new, specific task through transfer learning. These pre-training tasks typically involve large-scale image classification on datasets like ImageNet, which contains millions of images across thousands of categories [7].

By training on such a large and diverse dataset, the ResNet model learns to recognize a wide variety of features, such as edges, textures, shapes, and object parts, that are common across different types of images. These learned features form the basis of the model's understanding of visual patterns which can then be used to classify more specific data [7].

ResNet transfer learning has successfully been adapted to leaf disease identification tasks [56]. The model made use a Leaky ReLU Activation Function to deal with neurons that have a negative output becoming inactive with standard ReLU activation functions. It is an interesting design choice and something to consider. The model demonstrated a high accuracy of 94.56% but was trained on 24,000 images from the Plant Village dataset [57]. ResNet is popular CNN model for transfer transfer learning applications for leaf disease identification as can be seen in 2.3.5 .

GoogLeNet

The inception model consists of 'Inception modules' which are able to perform multi-scale feature extraction using different sized convolutional filters within the same layer [8]. This allows the network to capture spatial hierarchies in image data at different resolutions.

Table 2.6: Simplified Inception-v1 (GoogLeNet) Architecture

Layer Block	**Output Size**	**Components**
Convolutional Layers	Varies	Multiple Conv layers with varying filter sizes
Inception Modules	Varies	Multiple parallel Conv layers (1x1, 3x3, 5x5) and pooling
Auxiliary Classifiers	Varies	Auxiliary softmax for deeper layers
Final Pooling	7x7	Average pooling
Fully Connected	-	FC, 1000 (num classes)
Softmax	-	Softmax

The Inception model is designed around the concept of being able to look at an image through multiple perspectives at the same time. It uses filters of varying sizes (1x1, 3x3, 5x5) to capture simpler features with smaller filters and more abstract features with larger filters [8]. This model's approach would be particular suited for the compelx features present in the collected dataset. The architecture of inception is deep and wide, but is able to discern subtle differences between species.

These characteristics would make it an excellent candidate for transfer learning. The pre-trained weights will allow for a more advanced starting point and customizing and freezing layers can be quite advantageous to the task at hand. However the complexity and computational requirements represent challenges. Over-fitting also becomes a problem when working with relatively limited training data.

Accuracy Metrics and Explain-ability The measurement and continued monitoring of how models perform is vital to the success of this project and of any large scale complex machine learning in general. In this section, various methods of model evaluation will be reviewed with relevance to FLORA-CNN. This section will also introduce an explain-ability framework for deep learning known as SHAP. This analysis will be used to iteratively improve the models in the methodology. Most accuracy metrics were originally envisioned for binary classification problems but FLORA-CNN represents a multi-class classification problem [58].

The most commonly occurring metrics found in literature are covered below:

Classification Accuracy (CA) - The accuracy is calculated as the ratio of the sum of True Positives (TP) and True Negatives (TN) to the total number of cases (TP, TN, False Positives (FP), and False Negatives (FN)) [58]. it can be represented by 2.7:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.7}$$

Accuracy is the key metric presented in many models applied to botany [47], but in cases such as leaf disease detection in [48], a very high accuracy ($> 99\%$) can be achieved because of the nature of the problem. This is unlikely to be possible for species differentiation involving multi-class classification, especially on limited training data. Accuracy is measured on the training set and validation set separately (training accuracy and validation accuracy respectively). Should the training accuracy be higher than the validation accuracy, the model will be considered over-fitted [31]. This is especially a problem for small datasets such as that of FLORA-CNN so while accuracy is an important metric, it is not the defining feature especially when the context of poor data is considered. This contrasts heavily against earlier versions of FLORA. As an example, if there are multiple classes with extremely poor quality of samples, the model will have a poor accuracy even if it is performing well on other species.

Precision - Precision measures the ratio of correctly predicted positive observations (TP) to the total predicted positives (TP and FP) [58]. In other words, it looks at how many positive predictions made by the model were actually positive. In the task of classifying Fynbos leaves, a high precision will generally be crucial as the goal is to minimise the number of incorrect classifications. In the context of rarer species this is even more important. It can be calculated by 2.8:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.8}$$

Recall/Sensitivity - Recall on the other hand measures the capability of the model to find all relevant cases in a model [58]. It measures the ratio of correctly predicted positive observations (TP) to all observations in the actual class (TP and FN). In practice, there's often a trade-off between precision and recall. Improving precision typically reduces recall and the inverse is also true. For the purposes of FLORA-CNN, where missing particular rare or ecologically important species of Fynbos, recall might be the more important variable. It is computed as follows in 2.9:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.9}$$

F1-Score - The F1-Score is the harmonic mean of Precision and Recall [58]. The F1

score might be used as a balanced metric to evaluate the overall performance of the CNN in classifying Fynbos leaves, considering both precision and recall. After accuracy it is primarily used in literature as an evaluation metric as seen in 2.3.5. For FLORA-CNN, F1 score will be the most important metric for determining model performance as it will infer how the model performs on certain classes in terms of classification. This will directly impact design choices in the model architecture. It is calculated as follows in 2.10:

$$\text{F1-Score} = 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \tag{2.10}$$

Confusion Matrix - The confusion matrix is a useful tool to visualise. It shows true positives, false positives, true negatives, and false negatives, and provides a more in depth look at where the model is making errors [58]. This will be vital for FLORA-CNN as it will provide reason for design choices and assist with explain-ability.

Top-k Accuracy - Top-k accuracy measures are other metrics used extensively in literature as can be seen in 2.3.5. In terms of classifying Fynbos leaves, this metric will be particularly valuable because the actual distinction between classes is challenging, especially when species share many traits [59], [48], [60], [53]. These metrics will be included in the evaluation as they show how the model deals with learning shared features.

Cross Entropy Loss - Loss is the most important measurement in a neural network This is a common loss function used in classification tasks [61]. The measures the performance of a classification model with output 0-1. The loss is expected to increase as the prediction probability diverges from the actual label [61]. For multi-class classification, it is often used in conjunction with the softmax activation function, where the combination is commonly referred to as "softmax loss" [**?**]. It assists the training process by improving model accuracy by minimizing the difference between the predicted and actual probability distributions. This will be an effective metric to analyse as the model trains so as to see the overall model performance. It is important to emphasise that this metric is used during the training process. It is represented by 2.11:

$$\text{Cross-Entropy Loss} = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{2.11}$$

where:

M is the total number of classes,

c is a class index, running over all classes $1, 2, \ldots, M$,

$y_{o,c}$ is a binary indicator (0 or 1) for whether class c is the correct classification for observation o,

$p_{o,c}$ is the predicted probability that observation o is of class c.

The goal in training the model is to minimize cross-entropy loss, which is achieved by adjusting the model parameters to produce predicted probabilities that are as close as possible to actual labels.

This combination of evaluations metrics will allow for a suitable framework of model evaluation. Accuracy will be used as the fundamental metric to evaluate how the model is performing, but precision and recall will be used to deep-dive specific classes to see how the model generalises. The key to FLORA-CNN is to provide a model that generalises well, so that new data can be ingested. Classifications reports and confusion matrices will dominate the discussion points

SHAP Analysis

In terms of model evaluation, CNNs often fall into the trap of becoming 'black boxes' where all of the mathematics is abstracted away and lost in a sea of incomprehensible numbers. It is vitally important that this not be true for FLORA-CNN as the iterative design will require explain-ability at each step as will ongoing monitoring of the model performance. A core objective of this thesis is also to build trust in model predictions Explain-ability refers to the ability to explain why a model made a certain prediction. There are several explain-ability tools but this project will focus on SHAP [49]. Based on Shapley Values, a concept from game theory, SHAP provides a suitable framework for understanding why certain results were obtained as predictions by measuring the contributions of each feature to the output.

For deep learning, there is a specific adaptation of SHAP which integrates SHAP with DeepLIFT (Layer-wise Relevance propagation). Deep SHAP essentially computes SHAP values by approximating the contributions of each feature in each layer of the neural network [49]. This is particularly useful for FLORA-CNN where understanding which parts of the leaf images most influence the prediction, is the most important aspect of design. Deep SHAP also allows for the analysis of feature contributions of different layers

of the CNN which can reveal how the different layers process and transform the input features [49].

For the CNN models used in this study, the SHAP plots are computed at the output layer of the model, specifically after the final convolutional or dense layers. The SHAP values reflect the contribution of the input features (i.e., the pixels in the image) to the final class probabilities output by the CNN [49].

SHAP analysis are model agnostic - meaning that they do not impact the predictive ability of the model. They are applied after training to determine what features contributed strongly to the to the prediction [49]. In the case of FLORA, the SHAP analysis will highlight clusters of pixels (the image features) which contributed strongly to a positive or negative prediction.

By providing clear explanations, Deep SHAP builds trust in the model's predictions and helps validate the model's behavior. It also provides insights to guide model improvements. However it is computationally expensive, and knowledge of CNN architecture will need to be adapted to interpret the results of the SHAP analysis. SHAP analysis are not often presented in literature, but for FLORA-CNN, it will be vital to the model's success and it is hoped to inspire the use of this tool going forward for deep learning.

While individual SHAP plots for deep networks may be less clear than shallow ones, aggregating SHAP values across multiple instances can provide insights into the general behavior of the model. This can help identify which types of features or image regions are generally important across a dataset [49].

2.3.5 CNNs in Botanical Research

Convolutional Neural Networks have increasingly been applied in botanical and agricultural research utilized for tasks such as species identification, disease detection, and phenotypic analysis [47]. Their ability to process and analyze complex image data makes them ideal for studying plant morphology and health [59]. Large amounts of image data have been collected through technologies like remote sensing in the field of agriculture and image analysis techniques have started to get applied at a large scale on things such as image identification/classification and anomaly detection [47].

Leaf Disease Detection Several studies focused on detecting diseases in plants either from their leaves or other features[48], [62], [63]. All three approaches made use of CNNs and several interesting design choices will be noted that have direct applicability to this project. A comparison of the papers is presented below:

Table 2.7: Summary of Results of Leaf Disease Detection Research

Description	**Dataset**	**Deep Learning Model**	**Evaluation**	**Reference**
Leaf Disease Detection	15 species of leaves and 4483 images collected from the internet. Data augmentation was performed to increase the dataset.	Transfer Learning: CaffeNet with adjustments to the fully connected layer. ReLU used exclusively.	99.99% (CA).	[48]
Detecting crop disease through image recognition	38 classes using 54,306 images from the open source PlantVillage dataset.	Transfer Learning: AlexNet, GoogleNet	0.9934 (F1).	[62]
Classify banana leaf diseases	6 classes using 3,700 images from the open source PlantVillage dataset with no data augmentation.	Transfer Learning: LeNet	0.9462 (F1).	[59]
Classify tomato leaf diseases	54,306 samples from 14 types of plants from the open source PlantVillage dataset with no data augmentation.	Transfer Learning: ResNet and VGG-16 with major modifications	99.42 (CA).	[63]
Plant disease detection model for edge computing devices	54,306 samples from 14 types of plants from the open source PlantVillage dataset.	Transfer Learning: MobileNetV3	99.50 (CA) after 154 epochs.	[52]

All five models used a transfer learning approach and achieved very high F1 scores. Three of the models were trained using the PlantVillage dataset which unfortunately does not

contain Fynbos images. The other paper presented used a collection of internet images and data augmentation to construct a dataset. Image pre-processing was performed to different degrees in all 4 experiments ranging from just resizing, to applying manual cropping of leaves and removal of images that were too small [48].

The outcome of this research suggests that an approach based on transfer learning for smaller datasets will work well [48], [59], [63]. It is also extensively shown that data augmentation contributed heavily to the model performance [48]. It was shown that the MobileNetV3 was suitable for classification, and was aided by the implementation of a Squeeze-and-Excite (SE) module that improved feature representation [52].

It is noted that the model was not tested on unseen data [64]. This could indicate that the model suffers from some form of over-fitting and this consideration will need to be taken forward into the Methodology. Another interesting consideration is that training epochs seemed to vary greatly from 30 in to 100,000 and the results achieved across all models seemed to be fairly similar with all 4 papers noting convergence after a small number. This indicates that the number of epochs on previously trained models might not be as important and is also something to consider for the Methodology. This will be an important factor as computation resources are limited and if meaningful results can be gleaned from > 30 epoch cycles then it will translate to superior performance. Models are trained until convergence and not based on any pre-determined number of epochs which may or may not be reached.

2.3.6 Image Recognition of Plant Species

Much of the work involving CNNs in botany and agriculture are focused on satellite images and plant disease (as seen by 2.3.5). There have however been studies performed on leaf identification from input images, and entire viable applications created. The findings are summarized and presented in the table below:

Table 2.8: Summary of CNNs used in Leaf Classification

Description	**Model Used**	**Results**	**Source**
Exploration of plant identification using leaf vein morphology. Focuses on differentiating species based on unique vein patterns.	CNNs with increasing layer complexity	96.9% Classification Accuracy for 5 layer model. Visualization techniques provided insights into relevant vein patterns.	[60]
Study on classification of weed seeds using a deep learning.	AlexNet and other Transfer Learning models.	92.50% Classification Accuracy and 92.65% F1-score with AlexNet.	[65]

It was shown that increasing the number of layers in a network correlates with an increase in model performance [60]. The 5 layer custom model in this study was able to achieve an accuracy of 96.9% using an image resolution of 100x100 [60]. A key difference of this model to FLORA however, is the presence of advanced feature engineering on the input images. The central patch of each leaf was cropped and the rest of the image was discarded in order to actually eliminate the use of leaf shape as a feature. FLORA-CNN is intended to learn features from various aspects of Fynbos and therefore the images will not be subject to feature engineering, and it is unlikely that a accuracy of $> 90\%$ can be achieved.

2.3.7 Specific Studies on Fynbos Leaves Identification Including Previous FLORA iterations

Existing Research The FLORA project was originally developed in 2011 [10]. The initial project involved manually capturing a picture of a Fynbos leaf using an Image Capturing Device (ICD) and then manually uploading this file to a Central Processing Server (CPS). The image was then matched to a database which contained a variety of features and thus classified accordingly [10].

The study compared the performance of using a Probabilistic Neural Network (PNN), General Regression Neural Network (GRNN) and k-Nearest Neighbour (k-NN) classier algorithms. Six digital morphological features were extracted from 90 images of Fynbos leaves. The Fynbos dataset incorporated nine species from the Proteaceae family [10]. The accuracy of the various methods were as follows:

- GRNN - 86.66% with variance 9.41%
- PNN - 91.1% with variance 9.07%
- k-NN - 89.74% with variance 7.45%

Subsequent iterations of FLORA have attempted to improve on this algorithm in terms of computation speed and adjustments to improve accuracy [11], [12]. FLORA-E demonstrated more sophisticated matching algorithms including key-point matching of leaves and hashing [13]. These papers all repeat the feature engineering discussed in 2.3.5 in which images of leaves are cropped and placed against a white background to enhance the features in question. FLORA-CNN intends to train on natural images on Fynbos, which will increase the generalisation of the model while decreasing the accuracy.

These iterations of FLORA provide valuable insights into the ongoing development of image classification. FLORA-CNN attempts to address the feature engineering issues that each of these versions faced by training the model on natural images rather than photographing leaves against a white background to emphasis the features.

2.3.8 Data Handling and Preprocessing for CNNs

Data handling is perhaps the most important process in working with any machine learning algorithm. This section explores the key strategies and best practices in preparing data for CNNs, underscoring their significance in enhancing model accuracy and efficiency. Due to the limitations of FLORA-CNN's dataset, these procedures are vitally important and this section will inform many of the design decisions in the methodology.

Data Augmentation Data augmentation is a technique used to increase the diversity of the training set by applying random but realistic transformations to the training images. Almost all examples featured in the literature use some form of data augmentation. For the classification of Fynbos leaves, data augmentation helps the model to generalise better, making it more robust to variations in new, unseen data which is a key requirement of FLORA-CNN. In deeper networks with large training sets, the data augmentations tend to be less extreme [5]. For models with small datasets, the process was much more thorough with a dataset of 900 images being boosted up to over 4000 in [48] by making use of data augmentation methods. A brief summary of all methods applied in literature are presented below:

Table 2.9: Data Augmentation Techniques for Fynbos Leaves Classification

Augmentation Technique	**Justification**
Rotation	Random rotations of images help the model to recognize leaves in various orientations, thus simulating the varied orientations in which leaves might be photographed in real-world scenarios. Rotations were used in the original AlexNet model [5].
Horizontal Flip	Flipping images horizontally mirrors the leaves, increasing the dataset's diversity without compromising the leaves' structural integrity and features. This method was used to increase the leaf disease dataset significantly [48].
Zoom	Random zooming in on images helps the model focus on different parts of the leaves. This assists with feature extraction. A modification of this was performed in [66] to boost the training dataset, but only in high resolution images.
Width Shift	Data augmentation involving width shifts, height shifts, and shear transformations allowed for better model generalization [67].
Height Shift	Similar to width shifting, height shifting ensures the model can recognize leaves that are not perfectly centered vertically [67].
Colour Alterations	Grey scale images showed a decreased F1 score compared to colour images for leaf detection, indicating that colour is a useful feature to preserve [62].
Shear Transformation	Shearing alters the shape of the leaves in the image, which helps the model learn to identify leaves even when they are presented in slightly distorted forms. This was also present in [48].

Model performance was measured with increasing data transformations and it was shown that the error rate of the models with rotation, width-shift, height shift and perspective transformations far outperformed models without any transformations [67]. It was also noted that there is a maximum number of augmentations before the model performance begins to drop again. particularly for perspective shifts.

It is noted that data transformations need to be label preserving and not computationally expensive [5]. The AlexNet model made use of horizontal reflections and altering the intensities of RGB channels [5]. Deeper models such Inception employs random flipping, colour augmentation, random cropping and random scaling, although not all at the same time [8].

Given that these deep models trained on millions of images use extensive data augmentation, FLORA-CNN will also need to incorporate some of these into the model training to artificially increase the size of the dataset.

Normalization and Standardization

Normalization and standardization are two fundamental data pre-processing techniques in machine learning, especially for image data used in CNNs. These techniques adjust the pixel values of images to make the neural network training more efficient by limiting the number of features it can learn and preventing it from learning noise.

Table 2.10: Normalization and Standardization Techniques for Image Pre-processing

Pre-processing Technique	**Justification**
Normalisation	Normalisation involves scaling the pixel values of images to a range of 0 to 1. This is achieved by dividing each pixel value by 255 (the maximum pixel value) [68].
Standardisation	Standardisation involves transforming the pixel values so that they have a mean of 0 and a standard deviation of 1. This technique is necessary to align the scales of different features (pixel values in this case), which is particularly important for models sensitive to the scale of input data, like CNNs [68].

Image Resizing to 150x150 is generally performed depending on the model [5]. Resizing all images in the dataset to 150x150 pixels is necessary for maintaining consistency in the input data. This size is a balance between retaining sufficient detail for the CNN to learn features and keeping the computational requirements manageable. Different models such ResNet, use different sized images (224x224) which provides more learnable features at the expense of training time and computational resources [7]. For FLORA-CNN it seems there needs to be a balance struck in terms of image resolution as the model aims to maximise the number of features learned.

Data Quality Measuring data quality is crucial for working with small datasets. FLORA-CNN aims to provide a generalised model that can grow with time. It is for this reason that assessing the quality of training data is a key concern of the project.

Image sharpness refers to the blurriness of an image and can be measured by the Laplacian Filter which detects sudden transitions and edges [69]. The filter itself is a 3x3 matrix and when convolved with an image, calculates the second derivative of the image at each pixel. This is the measure of the rate of change of intensity, which can be expected to be higher at the edges. The kernel is displayed below:

$$L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.12}$$

The Laplacian filter can be used to quantitatively rate how sharp an image is, and sharper images can be assumed to provide better feature detection by the CNN. A similar approach is presented to detect the background and foreground in videos with some additional data pre-processing applied [70]. The Sobel Filters are presented below:

Sobel Operator for Horizontal Edges (Gx):

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.13}$$

Sobel Operator for Vertical Edges (Gy):

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.14}$$

The Sobel operator uses the above two kernels to detect edges in images. The consistency of the background can then be identified by analysis of the edges. Uniform backgrounds (such as the sky in FLORA images) will have fewer edges whereas Fynbos plant features such as leaves or flowers will have more edges [71].

It has also been shown that colour has a noticeable effect on CNN performance [62]. As a result colour can be a source of image quality. One of the metrics optimised was colour balance which is measured as the standard deviation of each R, G and B channel of an image [72]. This measures the spread of colour values in the image around the average. If the standard deviations are similar, it suggests that the image is more natural, whereas imbalanced images will have widely differing standard deviations. [72], [73].

Another key factor of image quality is the concept of noise and the calculation of it. One common method is to calculate the standard deviation of pixel intensities in a region of the image that should be uniform in color and brightness. This standard deviation gives a quantitative measure of the variation in pixel values, which, in a uniform area, should ideally be minimal [73]. This study provides extreme detail on different ways to measure different types of noise in images [73].

2.3.9 Gaps in Current Research

Based on the literature review, several gaps are noticed in the literature and FLORA-CNN will attempt to fill some of these gaps:

- There is a dearth of image recognition software being applied to Fynbos leaves. FLORA remains the only application thus far. This represents a significant opportunity for the further development of this tool.
- Models discussed in 2.3.5 rarely engage in data visualisation and model explainability. The CNN applications in botany and agriculture have not focused on understanding what features the models are able to identify. This represents a

significant gap in the literature as it enforces black box models and prevents the fine tuning of models for this particular application. Most models have used transfer learning approaches and have not customised CNN architecture to leaf or plant features. FLORA-CNN will make use of SHAP to address this.

- Models discussed in 2.3.5 are hyper-optimised for accuracy and often perform poorly on unseen data [48]. This suggests that the generalisation of models is not priority. Generalisation itself is a poorly defined term with regards to model evaluation as it is often not a concern when training on large datasets of images. FLORA-CNN proposes a novel approach to solving this. It is also suggested that results tested on field images yielded poor results for leaf disease [52].

- The literature discussed models trained on large datasets. There are instances of CNNs being trained on small datasets such as [74], but FLORA-CNN hopes to provide a novel approach to dealing with small datasets for training. As a result, over-fitting is often ignored or not assessed in literature but will be a primary concern for FLORA.

- Image quality is often not a concern in the literature as very few researchers collected their own images. Image quality represents a threat to model performance and FLORA-CNN hopes to address this based on the discussion in 2.3.8.

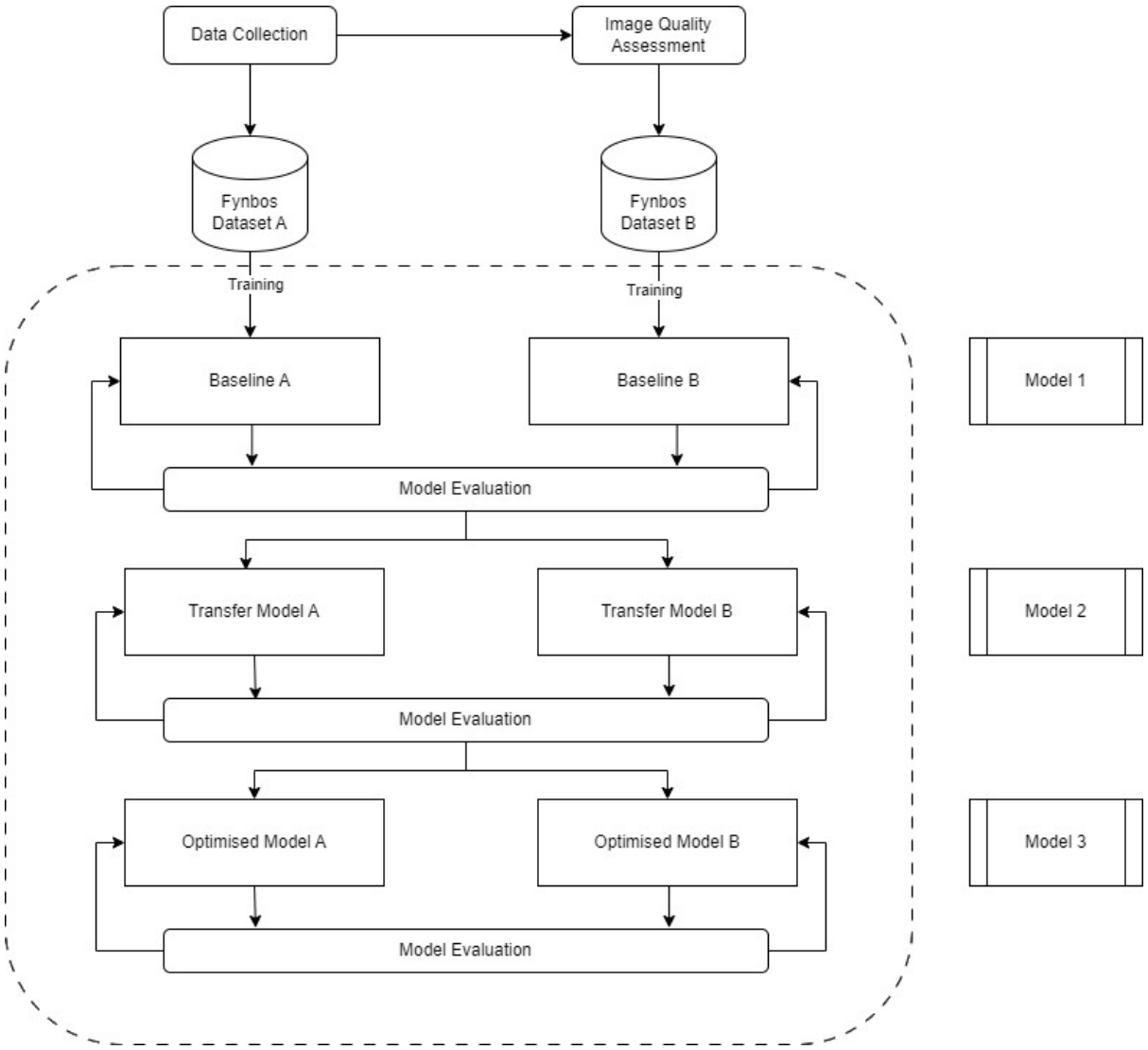
Chapter 3

Methodology

3.1 Research Design and Approach

This thesis adopts a phase based design founded on the engineering process. The core objective of this implementation is to incrementally enhance the capability of the CNN to accurately classify Fynbos species, maintaining performance and generalisation so that the model can improve over time as more training data are collected. Each phase or iteration focuses on specific aspects of the model's development and refinement, systematically building on the learnings and the outcomes of previous models. The rationale behind this approach is that the design of most deep learning models is not a linear process and deciding on a final design will require multiple rounds of experimentation, tuning and validation. The proposed phased methodology is presented below in **??**:

Figure 3.1: Project Phases



3.1.1 Explanation of Figure 3.1: Project Phases

reffig:methodology outlines the sequential phases of the project through data collection to model evaluation. The process is divided into multiple stages, where each stage builds upon the previous one, intended to improve the model's performance. This process results in three different models each trained on two different datasets, which results in a total of six models.

3.1.1.1 Data Collection and Image Quality Assessment

The project begins with data collection, where images of Fynbos leaves are gathered from Kirstenbosch Botanical Garden in Cape Town. These images are then passed through an image quality assessment to ensure that only high-quality images are included in the datasets. The images are organized into two distinct datasets: Fynbos Dataset A and Fynbos Dataset B. These datasets represent different subsets of the collected data, which may be used to evaluate the robustness and generalizability of the models.

3.1.1.2 Baseline Model Training

The first phase of model training involves the creation of baseline models, referred to as Baseline A and Baseline B. These models are trained on Fynbos Dataset A and Fynbos Dataset B, respectively. The performance of these baseline models is then evaluated to establish a reference point for future improvements.

- **Models Developed:** Baseline Model A and Baseline Model B.

3.1.1.3 Transfer Learning

The project moves into the transfer learning phase. In this phase, pre-trained models (such as ResNet) are fine-tuned on Fynbos Dataset A and Dataset B, resulting in Transfer Model A and Transfer Model B. These transfer models leverage the knowledge learned from previous tasks (such as image classification on a larger dataset like ImageNet) to improve performance on the Fynbos leaf classification task. The transfer models are then evaluated to assess the impact of transfer learning on model accuracy and generalisation.

- **Models Developed:** Transfer Model A and Transfer Model B.

3.1.1.4 Model Optimization

The final phase involves optimizing the transfer models. Through hyper-parameter tuning and other optimization techniques, the models are refined to maximize their classification

accuracy. The optimized models, Optimised Model A and Optimised Model B, are trained and then subjected to a final round of evaluation.

- **Models Developed:** Optimised Model A and Optimised Model B.

3.1.1.5 Final Models

The output of the entire process is six models: Baseline Model A, Baseline Model B, Transfer Model A, Transfer Model B, Optimised Model A, and Optimised Model B. These models represent different approaches and stages in the project, with each model offering insights into the impact of different techniques on Fynbos leaf classification. The models are categorized as follows:

- **Model 1:** Baseline Model A
- **Model 2:** Baseline Model B
- **Model 3:** Transfer Model A
- **Model 4:** Transfer Model B
- **Model 5:** Optimised Model A
- **Model 6:** Optimised Model B

Each of these models was developed, trained, and evaluated to determine the best-performing approach for FLORA. This ensures that the explain-ability of the model is always kept in mind, which ultimately builds better trust in machine learning.

3.2 Data Collection

Perhaps key to the entire model, the CNN is completely dependent on the data that it is trained on. There is no open-source dataset of Fynbos leaves that is suitable for FLORA, so the first phase of the project was manual data collection. This phase of the project was significant and was undertaken in multiple phases at different times of day in Kirstenbosch Botanical Gardens. The initial batch contained 328 samples which was deemed too little

and provided very over-fitted models. The site was revisited and a second batch added 868 samples for a total of 1,196 samples spread across 35 species. Earlier iterations of FLORA demonstrated were able to perform with much smaller datasets as they used leaf images with considerable feature engineering - notably placing the leaf images against a white background [10], [12]. These were considered shallow classifier models which attain high accuracy when trained using small datasets. This removed any background noise, irrelevant features, inconsistent lighting and other unfavorable conditions. This approach is similarly followed in the literature presented in 2.3.5. FLORA-CNN uses natural images and as a result has a different set of challenges, but is able to classify images taken in the field.

Botanical gardens were preferred for specimen collection as it was minimally invasive and the best way to ensure correct labeling of species by making use of garden labels. Images without correct labels are useless to the modeling process and have to be discarded. Images were taken at various angles, at various times of day and with varying backgrounds to mimic real life conditions. Care was taken to avoid shots of the sky as cloud patterns and other features would negatively impact feature detection specific to Fynbos leaves. Devices used for image capture include:

- Canon Camera EOS 2000D 24 MP DSLR Camera with default settings
- Samsung S22 Ultra phone camera with default settings
- Iphone 11 Pro camera with default settings

The different devices were used to mimic pictures that would be taken in the real world. In order to mitigate bias, there was a randomness applied to the photography with no species being favored over another. After collection the dataset is as follows:

ID	Species	Family	Image Count	Fynbos	Endemic to CFR
0	Aloe arborescens	Asphodelaceae	35	Yes	Yes
1	Arctotis stoechadifolia	Asteraceae	23	Yes	Yes
2	Aristea capitata	Iridaceae	21	Yes	Yes
3	Baloskion tetraphyllum	Restionaceae	23	Yes	No
4	Carpobrotus chilensis	Aizoaceae	27	Yes	No
5	Carpobrotus edulis	Aizoaceae	25	Yes	No
6	Cotyledon orbiculata	Crassulaceae	48	No	Yes
7	Curio talinoides	Asteraceae	42	No	Yes
8	Erica arborescens	Ericaceae	30	Yes	Yes
9	Erica cinerea	Ericaceae	37	Yes	Yes
10	Erica discolor	Ericaceae	33	Yes	Yes
11	Erica duthieae	Ericaceae	16	Yes	Yes
12	Erica perspicua	Ericaceae	26	Yes	Yes
13	Gazania rigens	Asteraceae	28	No	No
14	Grevillea banksii	Proteaceae	41	No	No
15	Helichrysum petiolare	Asteraceae	57	Yes	Yes
16	Leucadendron argenteum	Proteaceae	53	Yes	Yes
17	Leucadendron laureolum	Proteaceae	37	Yes	Yes
18	Leucadendron salignum	Proteaceae	39	Yes	Yes
19	Leucospermum cordifolium	Proteaceae	54	Yes	Yes
20	Leucospermum oleifolium	Proteaceae	51	Yes	Yes
21	Lithospermum ruderale	Boraginaceae	11	No	No
22	Melianthus major	Melianthaceae	52	No	No
23	Agathosma serpyllacea	Rutaceae	18	Yes	Yes
24	Mutisia orbignyana	Asteraceae	13	No	No
25	Oscularia deltoides	Aizoaceae	60	Yes	Yes
26	Osyris lanceolata	Santalaceae	17	No	Yes
27	Ozothamnus leptophyllus	Asteraceae	19	No	No
28	Pelargonium crispum	Geraniaceae	30	No	Yes
29	Protea aurea	Proteaceae	34	Yes	Yes
30	Protea cynaroides	Proteaceae	75	Yes	Yes
31	Protea neriifolia	Proteaceae	50	Yes	Yes
32	Protea repens	Proteaceae	41	Yes	Yes
33	Schoenoplectus californicus	Cyperaceae	10	No	No
34	Strelitzia reginae	Strelitziaceae	55	No	No

Table 3.1: Number of images per species, their respective families, Fynbos and Endemic classification

The dataset consists of 35 classes with 1,196 images which represents a fairly small dataset. However it has been shown that such a small dataset can be used with data augmentation techniques to achieve F1 scores of 99% albeit for leaf disease detection instead of species classification [48]. A leaf classification study done with >1000 images was performed with great results using relatively simple CNN architecture ranging from three to six layers deep although the input images were again heavily engineered to show only important features and natural images were not used as training data [60].

As can be seen 11 species were collected that are not classified as Fynbos and 8 of these are not endemic to the CFR. However the collected species often have similar characteristics to Fynbos due to the evolutionary requirements to live in the region [9]. *Cotyledon orbiculata*, for example has Fynbos-like leaves. There are some species included that are actually not native to South Africa such as *Schoenoplectus californicus*. A solution to this is proposed in 3.5. The species distribution per family represented in the table below:

Family	**Number of Species Classes**
Proteaceae	7
Ericaceae	6
Asteraceae	5
Aizoaceae	4
Iridaceae	3
Crassulaceae	2
Asphodelaceae	2
Geraniaceae	2
Melianthaceae	1
Restionaceae	1
Boraginaceae	1
Santalaceae	1
Strelitziaceae	1
Cyperaceae	1
Rutaceae	1

Table 3.2: The number of species classes within each family, sorted from highest to lowest

As can be seen there is a bias towards Proteaceae, Ericaceae and Asteraceae which was expected from the literature review in 2.1.2. However the collected species still show great variation in leaf features, which is required for the kernels to capture feature variety and improve the generalisation of the model as discussed in 2.3.2.

Some limitations and possible problems arise with the dataset as is such as:

- The dataset is sparse for a typical neural network and this could cause over-fitting as was seen in many of the leaf disease tasks in 2.3.5. The entire design process needs to be built around this in order to obtain a suitable model. However, with time the dataset is expected to grow, in which case the model will then have to be adjusted as it would have been optimised for a thin dataset. Therefore the model design needs to sacrifice optimised accuracy for generalisation. Being able to learn features from many different images and classify many different classes reasonably is preferred to a model that can predict a few classes with high accuracy. The model is tested for its ability to generalise in 4.1.

- This phase ended up being the most limiting aspect of the entire project as manual data collection is time consuming and challenging to do without a domain expert. It also happens to be the most important phase of the project as the data quality has a a significant impact on model performance (as can be seen in 5.1, in particular the higher validation accuracy for Baseline Model B after low quality images were removed) and ultimately guides the design choices. Future iterations of FLORA should focus on the dataset improvements.

- There exists a bias towards plants that are available in botanical gardens. This represents an issue of bias as these environments are not truly reflective of the natural CFR. However for the purposes of modeling, this is acceptable since other species can be added and trained later if the model is suitably generalised.

- A key aspect of the raw dataset is that images contain backgrounds which are often flowers or other leaves. For CNNs this is actually preferred, but it might have impacts on performance due to the limited dataset. Since the images are fairly high resolution it presents several trade-offs in design since it is often good practice to resize images to 150x150 for CNN ingestion, but doing so might hide image features [5]. A possible solution to this is presented in the 3.5.

- The dataset is made up of images taken with inconsistent lighting conditions, angles and resolutions. This was done to mimic real world data. This dataset will possibly perform better on real world data as a result but data augmentation techniques still need to be applied, and the model might struggle to learn features that would exist in leaf images taken against white backgrounds, such as leaf shape [10].

- Data labeling is crucial to the modeling process and that is why botanical gardens were favored. There were clear plaques and signs denoting each species at both sites and labeling of each species was conducted systematically. There is still the

possibility of human error as no botany expert was involved especially with species that look very similar. However this error, if it exists, is expected to be small. Future versions of the dataset should be peer reviewed for accuracy.

- There is class imbalance with a bias towards Proteaceae, Ericaceae and Asteraceae classes.

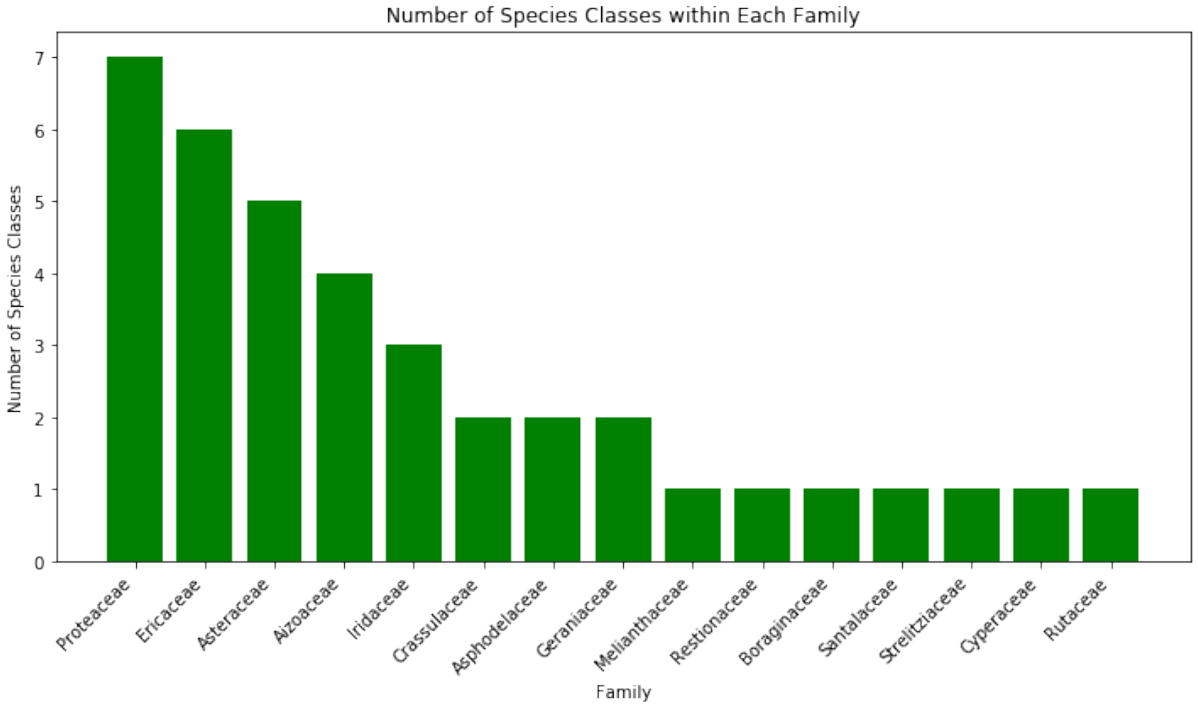


Figure 3.2: Histogram showing number of families and classes belonging to them in the dataset

While the dataset is small, this should not represent a significant problem. A solution to the imbalanced dataset is suggested in 3.5.

- The images of leaves were taken within the same month on different trips. Fynbos species are usually variable with seasons [18], so capturing them at roughly the same time should ensure that there is not much feature variability between images belonging to the same classes. As the dataset grows, this will become a concern.

- As far as the collection methodology goes, samples were collected randomly with visually striking leaves selected from various different locations within the botanical garden. This strategy ensured correct labeling but it also resulted in superfluous species being collected and wasted images. In future iterations it is suggested to apply a more systematic approach and involve a domain expert.

- Due to the low overall size of the dataset, there is significant room for data augmentation to artificially expand the dataset, but this needs to be performed within limits so

as not to cause over-fitting as observed in 2.3.5.

This dataset will be versioned as **Fynbos Dataset A - Version 1** and includes all the captured images that were able to be labeled.

3.3 Test Set

In the process of evaluation of machine learning models, data is typically divided into three subsets: training, validation, and test sets. The training set is used to fit the model, the validation set is employed for tuning hyper-parameters and model selection, and the test set is reserved for a final evaluation of the model's performance on unseen data.

A test set plays a crucial role in providing an unbiased assessment of a model's ability to genralise to new data. After training and validating a model, the test set serves as a proxy for real-world data that the model has not encountered before.

The dataset used in this study consists of a small set of Fynbos images. Dividing the dataset further into a training, validation, and test set would have resulted in much smaller subsets for training and validation, potentially compromising the model's ability to learn and generalize, and introducing bias due to the class imbalance.

Given the constraints, the decision was made to maximize the use of available data for training and validation purposes. By doing so, the study aimed to develop a more robust model with the limited data available. The validation set was employed to simulate the role of a test set by using it to tune hype-rparameters and assess the model's performance.

Although not originally set-aside, a post-hoc test set could be created from new data. This test set could be used to confirm the validation results and further assess the model's performance on unseen data. This will be encouraged for future developments.

3.4 Exploratory Data Analysis

This section serves to provide the first level of iterative design. The collected samples will be investigated and analysed. The main aim of this stage is to enhance **Fynbos Dataset A - Version 1**, and to prepare it for data pre-processing.

Visual Inspection Before a quantitative analysis is performed a visual inspection of **Fynbos Dataset A - Version 1** was conducted. There are no obvious anomalies such as corrupted files. There are however some images of plaques with poor representation of leaves that were included in the dataset. There were roughly 4 detected, it is proposed to remove these as they are irrelevant. There do not appear to be any major issues with regards to unclear or unfocused images, although there are highly irregular lighting conditions in many classes such as *Protea cynaroides*. However several images have noticeable background information including skies and mountains which could have an impact on model performance.

Figure 3.3: Images with noticeable background noise and sky features

For now, these images will be included as the dataset is already limited and results in 5.1 (particularly the SHAP plot in 5.1.4) do show that features for certain species such as *Leucadendron salignum* are clearly learnable. There are also several images that appear to contain multiple species. These will also be kept for now, but represent a threat to model performance.

Data Distribution Analysis

A histogram is generated to show the distribution of image counts per species to show which species are better represented in the dataset:

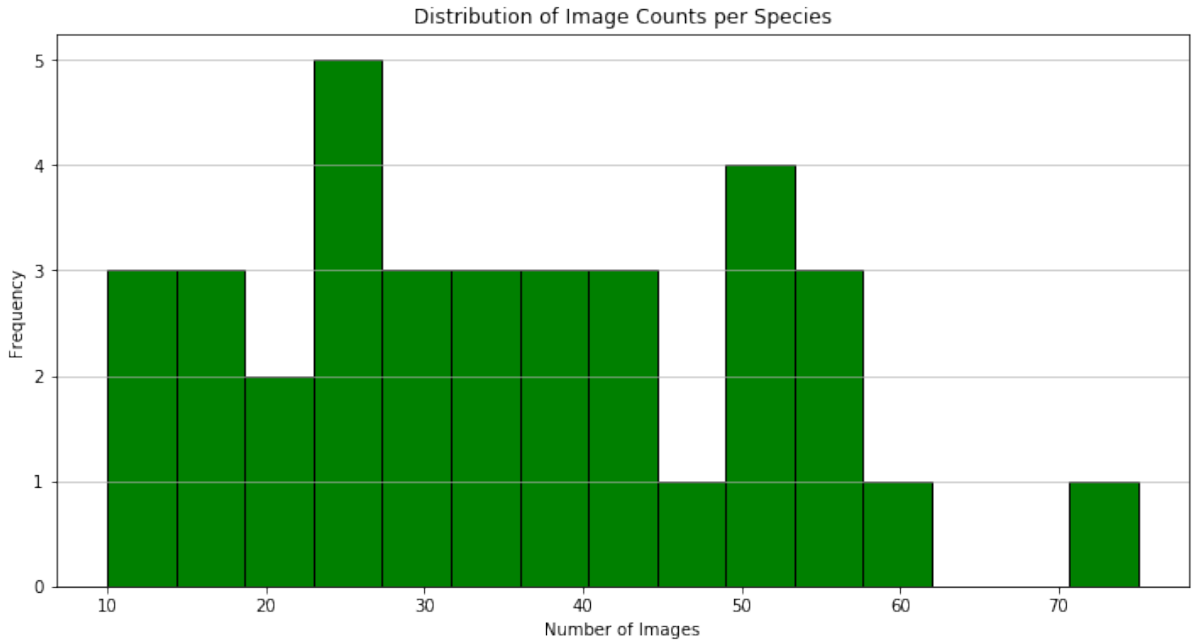


Figure 3.4: Histogram showing distribution of image counts per species

A **wide spread** is observed in the histogram denoting noticeable class **imbalance**. Some species have significantly more images than others suggesting a collection bias in the dataset, which was expected.

File Integrity Check The integrity of the sample images in **Fynbos Dataset A - Version 1** was assessed using a custom script written in Python designed to loop through the filing structure of the dataset and identify any corrupted files. This script was executed on the entire dataset, which comprised 35 different species, each stored in separate sub-directories. The script checked each image file for potential corruption, ensuring that the data used for model training was of high quality and reliability. The script can be found in Appendix C.

The results of the file integrity check were positive. Across all 35 species represented in the dataset, encompassing a total of 1,196 images, the script did not identify any corrupted files. This outcome indicates a high level of data integrity in the dataset, as shown in the table below:

Table 3.3: Summary of File Integrity Check Results

Species	**Corrupted Files**
Total Species Checked	35
Total Images Checked	1,196
Corrupted Files Found	0

This zero-incidence of file corruption contributes positively to the dataset's reliability, ensuring that the subsequent phases of data pre-processing and model training are not negatively impacted by file integrity issues.

3.5 Data Quality Assessment

In light of results obtained from the Baseline in 5.1 particularly the SHAP analysis in 5.1.4 where it was observed that species such as *Arista capitata* presented little to no learnable features for the model, several adjustments need to be proposed. It was noted that several classes seemed completely incomprehensible to the model and no features could be detected. This problem can be clearly seen in B.4 for the species *Aloe arborescens* and in B.2 for the species *Erica cinerea* where shadows on the images caused the model to focus on background features instead of the leaf shape.

Considering that parameters in 3.6 and 3.7 were explicitly designed to detect features in these classes, the problem appears to be that of data quality rather than a failing in the model especially because other classes like *Leucadendron salignum* did show clear features and achieved a suitable F1-score. It is therefore proposed to remove the poor quality samples from the dataset with the intention of increasing the generalisation of the model. The goal of FLORA-CNN has always been to provide a scalable platform for community research and not to hyper optimise the model on the collected dataset, acknowledging the limitations of the collection process presented in 3.2.

Removing poor quality images is a subjective task and it is preferred to adopt a quantitative approach rather than a judgment based approach. The majority of models reviewed in literature in 2.3.5 were either trained on much larger datasets (>20000 images) or used data augmentation to artificially boost the class sample sizes. The data augmentation approach often improved results, but caused severe over-fitting and the respective models performed poorly on unseen data [48]. These outcomes are unsuitable for FLORA-CNN and most approaches focus on collecting more data to address training issues, instead of

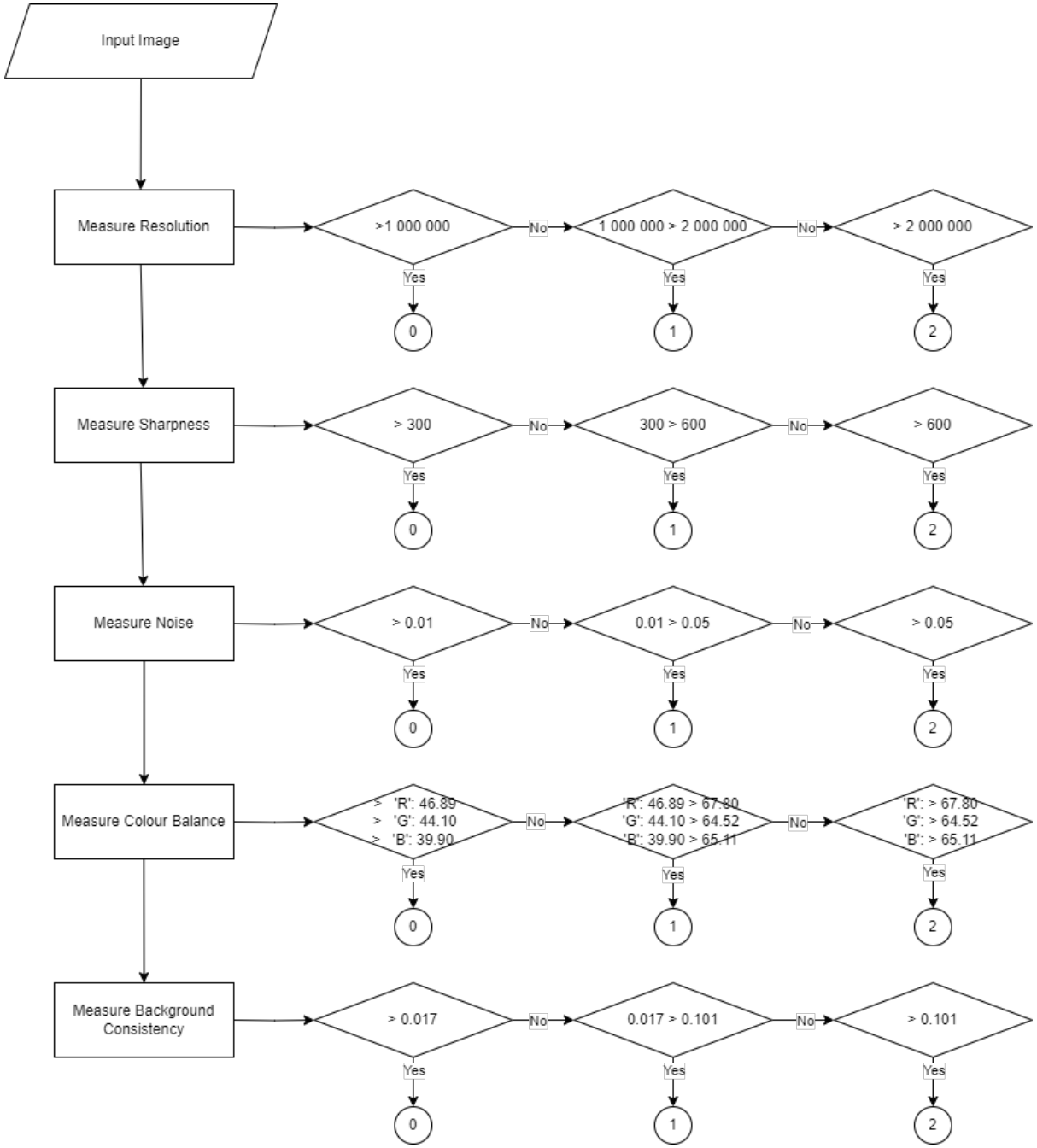
assessing existing data.

In order to quantitatively assess the data quality of Fynbos Dataset A, a novel approach to data quality is proposed taking into account image quality metrics discussed in 2.3.8. Common metrics used to assess data are:

- **Resolution and Size:** Ensuring each image has a minimum resolution and size which is necessary for CNN models. When using transfer learning, models have specific input requirements in terms of image resolution.
- **Sharpness:** Blurry images can lead to poor classification. Sharpness can be quantified using metrics like the Laplacian variance.
- **Noise:** High levels of noise can obscure details. Noise can be assessed using signal-to-noise ratio (SNR) or by calculating the standard deviation of greyscale images [73].
- **Colour Balance:** Especially important if the classification depends on color features which are common amongst the collected dataset images.
- **Background Consistency:** Assessing whether the background is uniform or cluttered, which might affect feature extraction.

Some of these metrics can be quantitatively measured for each image in Fynbos Dataset A and the results populated using the methods learned in 2.3.8. Based on the results a scoring system can be created, the scores tallied for each image and a class average can be computed. This class average can then be used to justify the exclusion of certain classes that are not contributing to the prediction, and which are in fact making the prediction worse by training the model on superfluous features. A flow chart representing this process is shown below:

Figure 3.5: Image Quality Assessment Process



A python function is created to measure these metrics for each image in Fynbos Dataset A a snippet of the code and can be found in Appendix C in C.1.

Methods such as Structural Similarity Index Metric (SSIM) provide a means to quantify image quality by comparing luminance, contrast, and structural similarity between images. However, more advanced IQA techniques including machine learning based ones, which involves blind image quality estimation via distortion aggravation, represents a significant

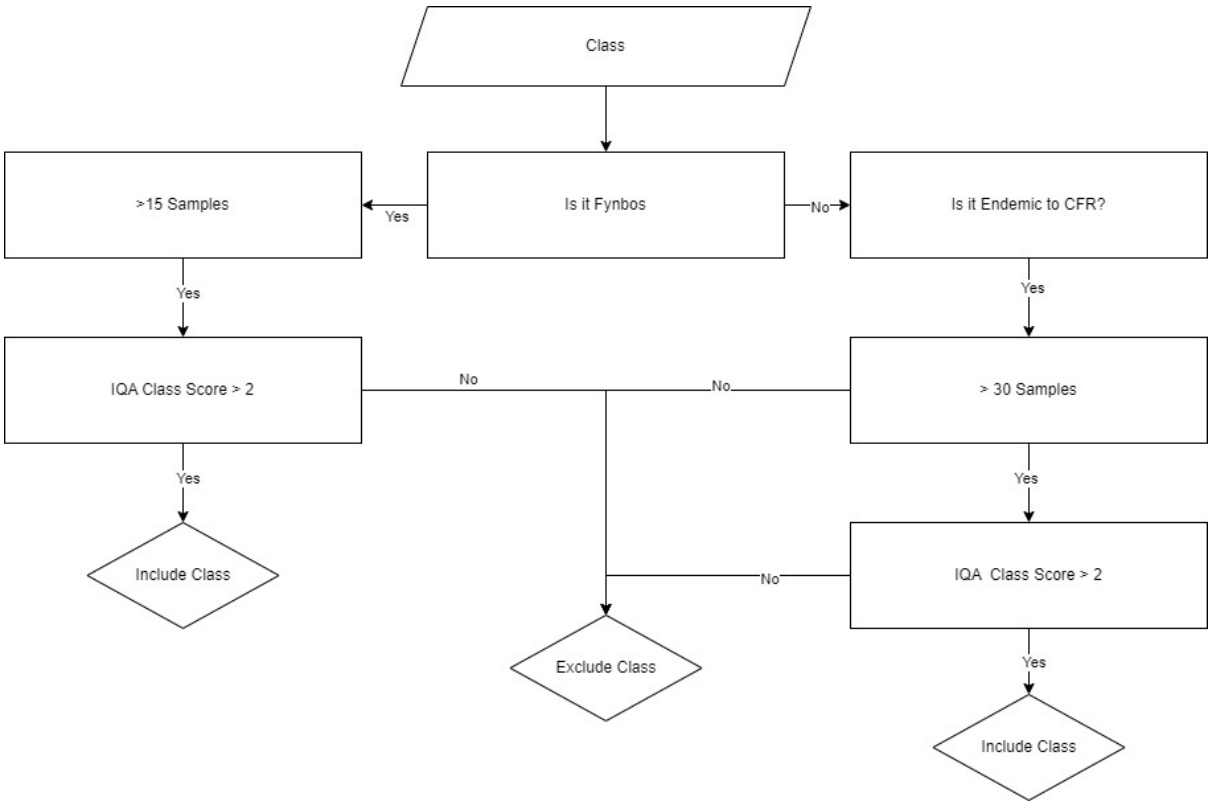
advancement in the field. This approach makes use of machine learning to predict image quality without requiring reference images, offering a robust mechanism for assessing and enhancing image data quality. By integrating such sophisticated IQA metrics, the training dataset in this project could be signifcantly improved. [75]. However this falls out of the scope of this thesis, but will be considered for future iterations.

Once these metrics are assessed, the dataset values are aggregated and threshold values are created to initiate a scorecard. Each metric is measured on a scale from 0-2 for a total score of 10 per image. This approach is a novel approach for CNN training data and hopes to solve the problem of poor image quality for small datasets. Threshold values are estimated by looking at images that score poorly in the dataset and comparing against the SHAP results, but are kept within reasonable limits to maintain generalisation since the datasets are expected to grow with time. The scoring can be found in C.2 in Appendix C.

Proposed Improvement - Fynbos Dataset B

Based on learnings thus far from the literature review and the first iteration of results from the Baseline Model on Dataset A (5.1), and results of the Data Quality Assessment Tool, a decision tree has been crafted to deal with several issues such as the noted data imbalance and the clear over-fitting that occurs in 5.1. The accuracy will always be poor if classes are included with which the model cannot find any type of pattern or feature. Until the sample size can be grown for these classes they are contributing very little to overall model, and in fact impacting the prediction negatively as observed from the SHAP plots where high negative values are attributed to background noise pixels. The result of this decision tree will be **Fynbos Dataset B**. The process is presented below:

Figure 3.6: Decision Tree to create Dataset B



The decision is validated by the assertions from 2.1.2 where the importance of the CFR was discussed. This study is designed for plants in the CFR region and mainly focused on Fynbos. As a result all Fynbos with less than 15 image samples are excluded as the model just does not seem able to learn features. This limit is increased to 30 samples for classes that are not Fynbos but are endemic to the CFR. Classes that are not Fynbos and not endemic to the CFR will be excluded with the exception of *Strelitzia reginae* as it demonstrates sclerophyllous leaves and is endemic to Southern Africa.

The number of images and image quality is not contributing to feature detection and it would be better to exclude these classes entirely. Applying this decision tree results in a secondary dataset **Fynbos Dataset B - Version 1**. All models will be trained on both datasets and the results compared. An overview is provided below in Table 3.4:

Table 3.4: Fynbos Dataset B

Species	**Image Count**
Aloe arborescens	35
Aristea capitata	21
Baloskion tetraphyllum	23
Carpobrotus edulis	25
Cotyledon orbiculata	48
Curio talinoides	42
Erica arborescens	30
Erica cinerea	37
Erica discolor	33
Helichrysum petiolare	57
Leucadendron argenteum	53
Leucadendron laureolum	37
Leucadendron salignum	39
Melianthus major	52
Mutisia orbignyana	13
Oscularia deltoides	60
Osyris lanceolata	17
Ozothamnus leptophyllus	19
Pelargonium crispum	30
Protea cynaroides	75
Protea neriifolia	50
Protea repens	41
Strelitzia reginae	55

This dataset contains 869 images spread across 23 classes for a roughly 29% decrease in size from Fynbos Dataset A. This decision is validated by the results in 5.1, where the over-fitting is severely reduced in the Baseline Model run on Dataset B.

3.6 Design 1: Baseline Model

This section outlines the design philosophy and expectations for the Baseline Model - the first of the three.

Model Architecture

The model architecture designed for the Baseline Model is based on understanding the theoretical foundations of CNNs presented in 2.3. As the first model, this model is designed to be straightforward, easy to implement and to explain. Its performance is not expected to be optimised but it has been designed so that the model can be explained and iterations and improvements can be easily applied based on feedback from the results. It is expected for the model to perform better than random guessing. Random guessing would represent a model that cannot learn any features and extract any meaning with only a random probability of successful prediction.

The initial hyper-parameter choice will be based on literature and informed estimations. Due to the small dataset and simple architecture, over-fitting is expected as with other small dataset models in the literature shown in 2.3.5, but by using SHAP, it will be possible to unpack what is causing this and the learnings will be taken forward into the next model.

The model architecture is presented below and justified followed by the hyper parameter choice and justification. Observed results from 5.2 shows that the number of layers often correlates to the validation accuracy [60]. However, it was decided to use just three layers in this model as there are no pre-trained weights. Without transfer learning, the model complexity would negatively contribute to the result [48].

The architecture is visualised below in **??** using the visualkeras library in Python. Note the three convolutional layers in yellow.

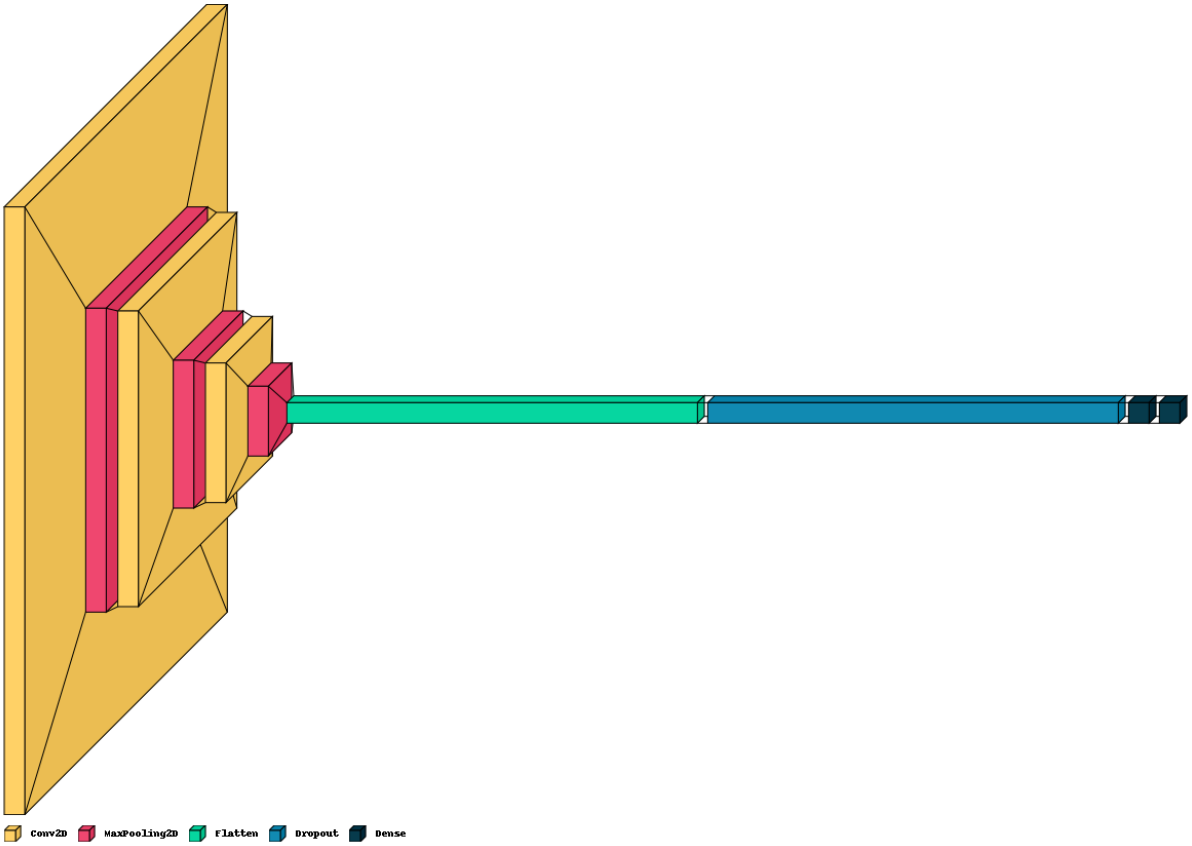


Figure 3.7: Baseline Model Architecture visualised using visualkeras library

The full architecture and justifications are presented below:

Table 3.5: CNN Architecture for Fynbos Leaves Classification

Layer Type	**Configuration**	**Justification**
Conv2D	32 filters, 3x3 kernel, 1x1 stride	First convolutional layer to capture basic features like edges and textures of leaves. Small kernel sizes of 3x3 are intended to capture finer details in the leaves as discussed in 2.3.2.
MaxPooling2D	2x2 pool size, 2x2 stride	Reduces spatial dimensions, aiding in feature extraction and reducing computational load. Max Pooling is chosen over average pooling as it is superior for image feature recognition as discussed in 2.3.2.
Continued on next page		

Layer Type	**Configuration**	**Justification**
Conv2D	64 filters, 3x3 kernel, 1x1 stride	Increases complexity to capture more detailed features. The model is moving onto more complex features in the leaves now such as clusters or veins.
MaxPooling2D	2x2 pool size, 2x2 stride	Same as above.
Conv2D	128 filters, 3x3 kernel, 1x1 stride	Further increases the network's ability to recognize complex patterns.
MaxPooling2D	2x2 pool size, 2x2 stride	Same as above.
Flatten	-	Flattens the output for the dense layer processing.
Dropout	0.5 dropout probability	Set to 50% to effectively prevent over-fitting by randomly turning off half of the neurons in the dropout layers during training. This helps the model generalize better to unseen data which is vital for reducing over-fitting in the small dataset. This was discussed in 2.3.2 and is the only regularisation applied to the Baseline Model.
Dense	128 units	Dense layer for learning non-linear combinations of features.
Dense	Number of output units	Fully connected layer where the number of units equals the number of classes in the dataset.
Softmax	-	Applies the softmax activation function to convert the outputs into probabilities for multi-class classification.

This architecture should be sufficiently geared to capture patterns present in the samples and it is expected that it would perform better than random guessing on the data. The Python implementation can be found in C.3.

Hyper-Parameter Choice

The selected initial parameters are justified in this section. For the Baseline Models, the parameters are based on the learnings from 2.3. This will lead to less robust results but will allow for explain-ability and assist with tuning more advanced models.

Table 3.6: Hyperparameters of the CNN Model

Parameter	**Value**	**Justification**
Learning Rate	0.0001	The learning rate optimizes the gradient descent through the network. Due to the small number of data samples and high complexity in leaf features, a smaller learning rate than the standard 0.001 is selected.
Optimizer	Adam	The Adam optimizer is chosen for its efficiency and because it allows for adaptive learning rates.
Loss Function	Categorical Crossentropy	Suitable for multi-class classification problems. It measures the difference between predictions and observed labels, as discussed in 2.3.4.
Batch Size	32	A commonly used batch size that offers a good trade-off between training speed and stability of the learning process [60]. The batch size also depends on GPU memory limitations.
Epochs	50	The number of epochs is set to allow sufficient time for the network to converge without over-fitting. This number was chosen for the Baseline Model as there are no pre-trained weights, and it will take longer for the network to converge. The choice of epochs was discussed in 2.3.5.
Dropout Rate	0.5	Set to 50% to effectively prevent over-fitting by randomly turning off half of the neurons in the dropout layers during training [36]. This helps the model generalize better to unseen data, which is key for the small dataset.

The hyper-parameter choice for the Baseline Model should provide an adequate baseline for the model to work with and it is expected that some Fynbos features will be learned

by the model.

Data Handling

Data handling represents a key aspect of modeling. The limited dataset size of the collected samples represents a challenge. However, several data augmentation strategies can be implemented. The choice of these methods is based on learnings from 2.3.8 and they are presented and justified in Table 3.7 below. For the Baseline Model, minimal transformations were applied and the only pre-processing step was to change the input image size to 150x150. This size strikes a balance between providing images that are not computationally expensive, while still retaining enough detail for accurate feature maps to be created. The full data augmentation strategy is presented below in 3.7.

Parameter	**Value**	**Justification**
Rescale	1/255	Normalize pixel values which is standard for CNNs.
Rotation Range	20 degrees	Introduce rotational invariance. Natural images would expect to have some type of rotation.
Width Shift Range	20%	Horizontal shifts to reduce bias towards central pixels.
Height Shift Range	20%	Vertical shifts to reduce bias towards central pixels.
Shear Range	0.2	A simulation of perspective transformations.
Zoom Range	0.2	This parameter artificially adjusts feature sizes.
Horizontal Flip	True	Applies horizontal symmetry.
Fill Mode	Nearest	Fills in new pixels after transformations.
Validation Split	20%	80/20 Training/Validation Split is standard practice.
Target Size	150x150 pixels	Input image size is selected to be 150x150.
Class Mode	Categorical	This is for multi-class classification.
Shuffle (Training)	True	This creates a random distribution of batches aiding in model generalisation.
Shuffle (Validation)	False	The validation set needs to maintain the order for consistent results.

Table 3.7: Data Augmentation and Preprocessing Parameters of the Baseline Model

Some data augmentation is applied to boost number of images in the datasets, but it is applied randomly to the training set. The overall transformations are chosen to maintain the features of natural images, and prevent unnecessary over-fitting.

3.7 Design 2: Transfer Learning Model

The second model is designed around the use of transfer learning. This is used to include a pre-trained model which is then adapted to the specific classification needs of the Fynbos images.

Model Architecture

The Baseline Model results in 5.1 demonstrated several key factors for consideration in the design of the second model. A flaw in the model is the large amount of over-fitting that occurred evidenced by the training and validation accuracy and loss graphs not tracking each other in 5.1 and 5.2. However it can also be seen in the SHAP analysis for *Aloe arborescens* for example that the model is performing poorly in feature detection. It cannot detect complex shapes. This is affirmed by the long time it takes for the validation accuracy to increase to its maximum level. This suggests that the model is not deep enough to be able to learn the more complex shapes that Fynbos leaves in the dataset assume.

Therefore the design philosophy of this iteration of the model is to use transfer learning (discussed in 2.3.4) to use a pre-trained model, and adapt it to the Fynbos datasets. This will mean that the model has already been trained to detect features on other images, and it will be modifying these learnings to be able to use for detection of Fynbos leaf features. This will solve the training time problem, as well as the simple feature problem evident in the Baseline Model evaluation. The design is presented below in Table 3.8. The chosen Transfer Learning model is the ResNet-50 model based on its previous application in plant disease classification tasks and for multi-class plant classification [56], [63]. However the model will be adapted and customised for the specific needs of this project by unfreezing the last 30 layers of the model, adding another global pooling layer, two more dense layers a dropout layer. The modifications and general architecture are presented below.

Table 3.8: CNN Architecture for Transfer Learning Model

Layer Type	**Configuration**	**Justification**
ResNet-50 (Partial Freeze)	Last 30 layers unfrozen	The ResNet-50 model, pre-trained on ImageNet which does contain other leaf images [30], is used as a base to leverage learned features from a larger dataset. The last 30 layers of the pre-trained model are unfrozen to allow adaptation to specific features of Fynbos leaves while retaining the already learned generic features in the earlier layers. This approach is beneficial for feature extraction in complex images [7].
Global Average Pooling 2D	-	Global Pooling discussed in 2.3.2 is used to reduce the spatial dimensions of the output. It helps in reducing the number of parameters, which decreases the computational load and mitigates the risk of over-fitting which was a problem observed in the results of the Baseline Models.
Dense	512 units, ReLU activation	This layer increases the model's learning capacity. The ReLU activation function is used to introduce non-linearity and helps to avoid the vanishing gradient problem. This helps the model learn more complex shapes.
Dropout	0.3	Set to 30% to prevent over-fitting. The dropout is adjusted down from the Baseline Model. Over-fitting is still expected to be a problem, but with the increase in model complexity, the dropout can be reduced slightly.
Dense (Output Layer)	Equals number of classes, Softmax activation	The final dense layer used for multi-class classification. The number of units corresponds to the number of Fynbos species (which differs between Dataset A and Dataset B).

The unfreezing of the 30 layers and the additional 4 layers are to allow the model the

capability to generalise to new data. Since it has been shown that the Fynbos images contain complex features, this modeling choice is intended to help the model use its pre-existing learned features and adapt them to the Fynbos characteristics. Performance is still not expected to be optimal, but an improvement on the Baseline models is expected.

Since this contributes 54 layers (50 in ResNet-50 plus the four additional layers), it is unrealistic to visualise the network architecture in this report but the code to create the above architecture is presented below:

The code used to implement the architecture can be found in C.4.

The parameter choice is justified in the table below in 3.9 for this model:

Parameter Choice

Parameter	**Value**	**Justification**
Learning Rate	0.0001	The learning rate is preserved from the Baseline Model. With the pre-trained weights from ResNet-50, it is expected that the training time will be much shorter and the accuracy and loss will converge much sooner.
Optimizer	Adam	The Adam optimiser is suited for working with deep networks applied to small datasets.
Loss Function	Categorical Crossentropy	This function is appropriate for multi-class classification.
Batch Size	32	This is the same as the Baseline Model to ensure efficient computation
Epochs	10	A lower number of epochs, as the model starts from a pre-trained state and requires less training to converge. this choice is validated by convergence after roughly 6-8 epochs seen in 5.12

Parameter	Value	Justification
Dropout Rate	0.3	A lower dropout rate is used in comparison to the Baseline Models, as the model is less prone to over-fitting due to the use of pre-trained weights and the added regularization from transfer learning. This value is a conservative estimate.
Callbacks	Early Stopping (patience=5), Reduce LR on Plateau (factor=0.2, patience=2, min_lr=0.00001)	Early stopping helps to mitigate over-fitting by terminating training if the validation loss stops improving. Reduce LR on Plateau changes the learning rate based on model performance automatically which assists with convergence and prevents the need to tune this parameter. These functions are built into Tensorflow and assist with training optimisation [76].

Table 3.9: Hyper-parameters of Transfer Learning Model

Functionally the hyper-parameters are again kept relatively general to provide a good comparison to the Baseline Model. This model is intended to test the effectiveness of transfer learning. The dropout of 0.3 is the only source of over-fitting mitigation, although it is expected that over-fitting will reduce due to the complexity increase in learned features.

Data Handling

The data augmentation applied to the model is largely based on learnings from 2.3.8. The transformations are kept to minimal values and are applied randomly to images in the dataset. The same pre-processing function is applied to both Dataset A and Dataset B before ingestion into the model.

Table 3.10: Data Augmentation and Preprocessing Parameters

Parameter	Value	Description
Rescale	224x224	This is the optimal Resnet input size.

Continued on next page

Table 3.10 continued from previous page

Parameter	Value	Description
Rotation Range	10 degrees	Kept to a minimum as it is unlikely that images will be photographed at extreme angles.
Width Shift Range	10%	Kept minimal to augment the dataset without too much distortion. The SHAP analysis of the Baseline Models showed a large amount feature detection took place at the edges of images.
Height Shift Range	10%	Kept minimal to augment the dataset without too much distortion. The SHAP analysis of the Baseline Models showed a large amount feature detection took place at the edges of images.
Shear Range	0.2	This is also kept subtle to provide augmentation without too much distortion.
Zoom Range	0.2	Also kept minimal to prevent distortion.
Horizontal Flip	True	Augment images for horizontal asymmetry.
Fill Mode	Nearest	Fill in new pixels after transformations.
Validation Split	20%	80/20 Train/Test Split is good practice.
Target Size	224x224	Standardise input image size for input layer. ResNet-50 can also accept 150x150 and 299x299 inputs [7].
Class Mode	Categorical	This is necessary for multi-class classification.

Continued on next page

Table 3.10 continued from previous page

Parameter	Value	Description
Shuffle (Training)	True	This ensures the random distribution of batches.
Shuffle (Validation)	False	The validation consistency needs to remain constant so this is set to false.

The data augmentation is kept more or less consistent with the Baseline Models with the main difference being the change in input image size. Larger images will cause the model to take longer to train, but will provide more detailed learnable features.

3.8 Design 3: Optimised Model

This section represents the third model designed using learnings from the previous and introduces a Squeeze-and-Excite Module to increase model performance.

Model Architecture The results obtained for the Transfer Learning models highlighted noticeable improvement over the Baseline models, but there was still significant over-fitting for both datasets. The model still struggled with several failings, noticeably several mis-classifications represented by the off-diagonal elements in the confusion matrices. The macro-adjusted accuracy for Transfer Model B was 69% which is not yet suitable for a viable model. The SHAP analysis revealed that the model still struggles with inconsistent lighting conditions and *Erica cinerea* in particular continued to provide background noise that confused the learnings. In addition the complexity of the features learned seen in the SHAP analysis are not ideal. This can be observed in the SHAP analysis in 5.19, where despite scoring highly, the features only make up small portions of the pixels that contribute to positive predictions.

For these reasons, a more optimised approach is suggested to mitigate many of these weaknesses. The first new addition is Squeeze-And-Excite (SE) Block which is able to adaptively re-calibrate channel-wise feature responses [52]. This builds on the attention focused strategies discussed by [25]. This strategy enhances the representational power of the network by emphasizing informative features and suppressing less useful ones such as the noise in the background, or dark patches caused by leaf shape that were prominent

in the SHAP plots. The Squeeze-and-Excite module was featured several times in the literature in 2.3.4.

The SE block works by creating an additional block composed of two parts:

- The 'Squeeze' which makes use of Global Pooling (discussed in 2.3.2) across the spatial dimensions (the width and height) of the input feature map to create a feature descriptor that contains the global spatial information for each channel [52].
- The 'Excitation' phase where the block learns a set of channel-wise lengths by passing the above feature descriptor through a two layer full connected network. This network consists of a layer that reduces dimensionality, a ReLU activation, and a layer that increases dimesnionality. The output from this layer is a set of scalar coefficients for each channel which are obtained with a sigmoid activation function (discussed in 2.3.2) which ensures that the coefficients are in the range [0,1]. These values represent the importance of each channel's features[52].
- The last step is to scale the original feature maps by these learned co-coefficients. This process allows the network to emphasise important features. This operation is the channel-wise multiplication between the original feature map and the learned scalars.

The SE block aims to enhance the ability of the network to detect important features and suppressing the background noise and lesser features seen in the SHAP analysis. This model also introduces batch normalization to reduce covariate shift (discussed in 2.3.2) to reduce the over-fitting and improve model generalisation.

The block is implemented in Python can be found in C.5. Again the best way to visualise the architecture is the below table due to the large number of layers:

Table 3.11: Enhanced CNN Architecture with Squeeze-and-Excite Blocks

Layer Type	**Configuration**	**Justification**
ResNet50 Base	Input size: 224x224x3	The increased input size allows the model to capture more detailed features.

Layer Type	**Configuration**	**Justification**
Squeeze-and-Excite Block	Ratio: 16	Squeeze-and-Excite blocks enhance the representational power of the model, allowing it to concentrate on more important features and ignore less important ones.
Batch Normalization	-	Applied after each Conv2D layer. These layers are introduced to mitigate the over-fitting seen in the Baseline Models and Transfer Models.
Activation	ReLU	ReLU activation is used after batch normalization to introduce non-linearity, which allows the model to learn more complex features in the data.
Partial Freeze	Last 30 layers unfrozen	Fine-tuning the last 30 layers to adapt the pre-trained features for specific recognition of Fynbos leaves. The earlier layers are kept frozen to retain generic features learned from ImageNet.
Global Average Pooling 2D	-	Reduces the spatial dimensions, condensing the feature maps into a single vector per map, reducing the number of parameters and computation in the network as discussed in 2.3.2.
Dense	1024 units, ReLU activation	A large dense layer to learn complex combinations of features extracted by the convolutional base and Squeeze-and-Excite blocks.
Dropout	0.5	A dropout rate of 50% to prevent over-fitting, especially important given the increased complexity of the model. This value is increased from 0.3 in the previous model.
Dense	512 units, ReLU activation	An additional dense layer to further refine the features for the final classification task.
Output Dense	Units equal to number of classes, Softmax activation	The final layer for classification with units equal to the number of Fynbos leaf classes, using softmax to output probability scores for each class in the datasets.

The CNN architecture has been optimised based on learnings from the first two models. The increase in dropout to 0.5 and the introduction of batch normalisation layers should address some of the over-fitting seen in 5.12.

The Squeeze-and-Excitation block is designed to allow the model to learn more complex features, and to ignore irrelevant features that it has been learning so far. This should assist with the poor performing species seen the SHAP plots of the Transfer Models including addressing the brightness issues with *Erica cinerea* seen in 5.23.

The architecture code implementation for this model can be found in C.6.

This model represents a significant modification to the ResNet-50 Model designed to adapt to classifying Fynbos leaves. An increase in performance is expected in terms of accuracy, lower loss, higher F1-scores (showing fewer mis-classifications) and the SHAP plots are expected to show larger and more complex images.

Parameter Choice The hyper-parameters for this model and presented and justified in this section in the table below:

Parameter	**Value**	**Justification**
Learning Rate	0.0001	The initial learning rate is set low because of the increased complexity of the model.
Optimizer	Adam	Continues with the Adam optimizer for its adaptability and efficiency.
Loss Function	Categorical Crossentropy	Appropriate for multi-class classification tasks.
Epochs	15	Increased from the Transfer models, reflecting a balance between sufficient training for convergence and avoiding over-fitting, especially given the model's increased complexity.
Callbacks	Early Stopping (patience=5), Reduce LR on Plateau (factor=0.2, patience=2, min_lr=0.0001)	Early stopping prevents over-fitting by ceasing training if the validation loss stops improving, while Reduce LR on Plateau adjusts the learning rate adaptively during training [76].

Parameter	Value	Justification

Table 3.12: Hyper-parameters of Optimised Models

The hyper-parameter choices are based on learnings from previous models and should provide a suitably generalised model. The most significant parameters will be tuned in 5.5.

Data Handling Due to the increase in model complexity, most of these parameters are reduced. The reduction is designed to reduce over-fitting, particularly because the SE block should negate the need for many of the transformations. The data augmentation parameters are presented below:

Table 3.13: Data Augmentation and Preprocessing Parameters

Parameter	Value	Description
Rescale	224x224	ResNet allows for this image size to be used as inputs and it strikes a balance between computation time and detail [7].
Rotation Range	5 degrees	Reduced from the previous model to preserve the natural orientation of images
Width Shift Range	5%	Reduced to maintain framing
Height Shift Range	5%	Reduced to maintain framing
Shear Range	0.05	Reduced but still provides subtle transformations
Zoom Range	0.05	Reduced but still provides some minor scaling transformations
Horizontal Flip	True	Maintained as true to allow flipping
Fill Mode	Nearest	Fills in new pixels after transformations
Validation Split	20%	The 80/20 training/validation split is standard practice

Continued on next page

Table 3.13 continued from previous page

Parameter	Value	Description
Class Mode	Categorical	For multi-class classification
Shuffle (Training)	True	Ensure random distribution of batches
Shuffle (Validation)	False	Maintain order for validation consistency

The reduction in data augmentation is offset by increase in model complexity. This should provide a suitable balance, while still allowing for some minor augmentation.

3.9 Model Evaluation

In order to validate and observe the performance of each model, a series of evaluation metrics are derived from 2.3.4 which each offer insights into different aspects of the model's predicate abilities. These metrics are presented below:

- **Accuracy (Training and Validation)** is the primary metric and it is a measure of the proportion of correctly identified instances from the total number of predictions. Both training (the model's accuracy on the training set) and validation (the model's accuracy on the validation set) are plotted and observed throughout the training process. The analysis was focused on identifying the training epoch at which over-fitting or convergence began, which are shown by plateaus in the plots.
- **Loss (Training and Validation)** is the discrepancy between the predicted and actual labels. A decreasing trend between training and validation loss shows that the model is minimizing this error and learning while an diverging trend shows over-fitting.
- **Top-5 and Top-10 Accuracy (Training and Validation)** are metrics that assess the models ability to make the correct prediction within its top 5 or top 10 predictions. This is a more lenient metric, but is useful for showing that the model is learning features unique to Fynbos species. A high top-5 or top-10 accuracy, compared to top-1 accuracy, indicates the model's effectiveness in narrowing down the possible classes to a the most similar classes.

- **Classification Reports** are detailed reports that show precision, recall and F1-score for each class in the dataset. This offers insights into high and low performing classes and informs modeling decisions.
- **Confusion Matrices** were used to visualise the actual predictions of the models. This visualisation is useful for highlighting the examples of mis-classifications across all classes and determining patterns that emerge in model behavior.

SHAP Analysis

In order to interpret each model's ability to make predictions, SHAP values are used. This builds on the literature reviewed in 2.3.4. This approach enables a detailed explanation and visualisation of the type of features (e.g. shape, colour, leaf texture, leaf shape) learned and their influence on the final prediction. This offers human-interpretable insights into what the model is focusing on and guides design choices to change what this focus is on. A SHAP plot from the analysis is shown below:

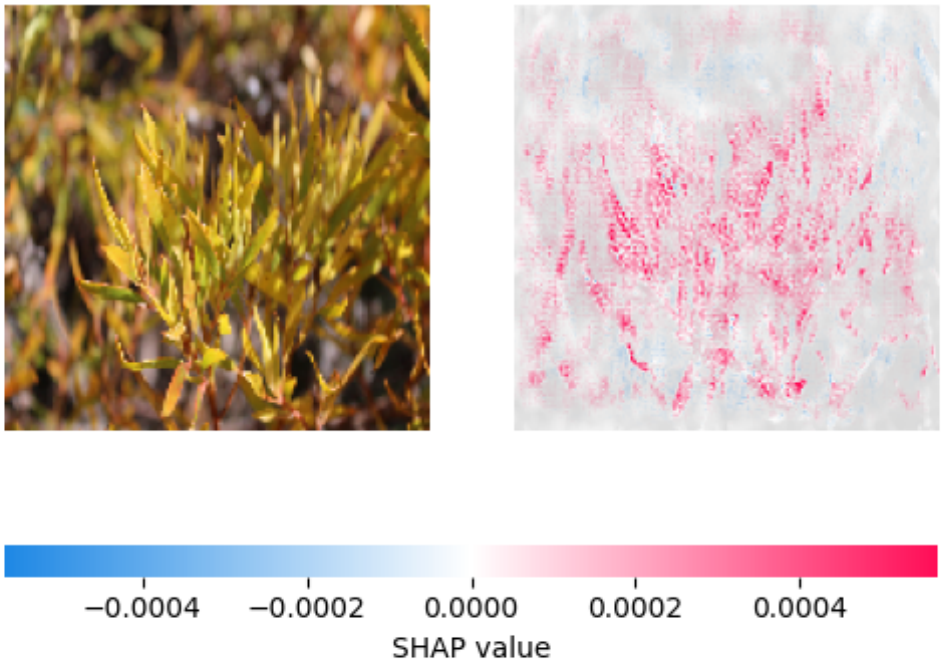


Figure 3.8: SHAP Plot of *Leucadendron salignum*

The SHAP plot is essentially a heat-map of SHAP values. High positive SHAP values are denoted by a pink colouration. These features contribute to the model making a positive prediction. The large negative SHAP values are denoted by blue pixels and these pixels contribute to negative prediction.

Using this colouration it possible to observe which pixels in each image contributed he positive and negative predictions. The SHAP plots provide observations of the type of features that the models are learning. More distinct and larger features are denoted by large and bright pink distributions. In contrast, patches of blue pixels also show features that the model is learning to avoid (in this case it appears to be background features). Images with SHAP values close to zero denoted by grey, show areas that the model is ignoring.

With these visualisations, the black-box aspect of neural networks can be removed as human observers can now see what the model is focusing on during predictions. Remember

SHAP is agnostic to the model's prediction itself [49]. This analysis goes a long way to building trust in these models.

The code for the SHAP plots can be found in C.7.

Chapter 4

FLORA Design

This chapter discusses the experimental setup of the project and how models are evaluated and explained. This lays the foundation of the iterative design and results from the earlier models are designed to provide insights and results that can be applied to later designs, optimising performance and providing a framework for sound engineering principles. This section aims to provide the bridging point between the learnings from the literature review, the methodology and the observed results, demonstrating the learning feedback.

4.1 Overview

FLORA stands for Fynbos Leaf-based Online (or Optical) Recognition Application. FLORA is intended to be a web-based application which is capable of ingesting an image of a Fynbos species and classifying it in real time. The web based architecture has undergone several iterations since inception but the overall design remains similar. FLORA up until this point has been limited to whatever species it was already been trained on. FLORA-CNN provides the first iteration where labeled images can be fed in and the model can automatically retrain itself to either start detecting new species or improve the classification rate of species already in the dataset.

FLORA has some system requirements:

- Internet Access
- An internet browser

The FLORA proposal is split into 4 main components: the trained CNN model discussed in 3.6 , the database of Fynbos leaves discussed in 3.2, the back-end and the front-end discussed in 4.5. The model needs to output a multi-class classification report of the image fed to it. The front-end system is responsible for supplying a graphical interface for the user to upload images as well as for a domain expert to input labeled images of new species. The back-end and database are responsible for all image processing, model training and model classification.

The development of a full application falls out of the scope of this project as the full back-end would add considerable developmental overhead. For this reason, only a comprehensive-front end module was designed.

4.2 Experiment Setup

As outlined in the methodology there are three models designed and two datasets. Each model will be run on each dataset resulting in 6 models. For the testing setup, the Python is used exclusively with Jupyter Notebooks to conduct testing. All code for this project can be found on Github at:

`https://github.com/Jarushen/FLORA-CNN`

In conducting computation experiments for this thesis, Google Colab was selected as the primary environment due to its access to advanced GPUs and TPUs via cloud computing which significantly reduced the training time of the graphics heavy models used in this thesis. Google Colab's integration with Google Drive allowed for effective data management as Fynbos Dataset A and Fynbos Dataset B are currently stored on Google Drive.

This environment unfortunately introduces a set of limitations for the project, including inconsistent access to computing resources, unpredictable hardware allocation, constraints on session time, which introduce significant variability in model training times. This variability is out of the users control and renders training time an unreliable metric for evaluating model performance. As the FLORA application transitions to its own dedicated servers, the evaluation of training time will become relevant. Training times are stored, but due to the above reasons are not considered.

Ultimately training times will scale with dataset size, and they do not represent a

significant hurdle to the project as the training will be compiled in the background.

All Python libraries used are outlined in the Appendix.

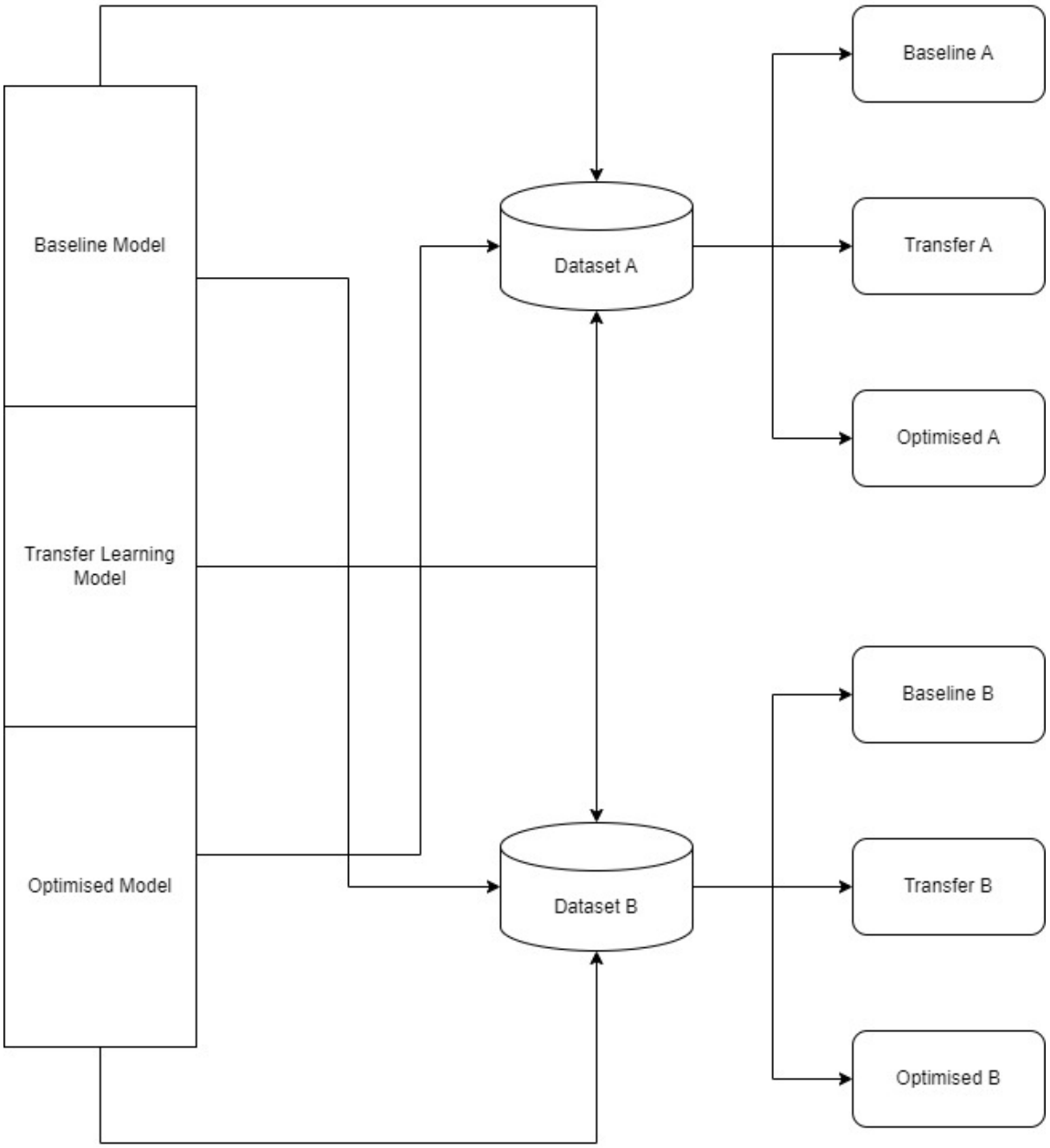


Figure 4.1: Experiment Setup

This will result in 6 sets of results to be analysed. At this point model parameters will be set from literature, experimentation and informed estimation. These models will be assessed on their training and validation accuracy, training and validation loss, top-k accuracy and F1-scores. Classification reports for each model and confusion matrices will

be created in order to show which classes perform well and which struggle for each model. This will inform which class to focus on for the SHAP analysis which will inform what features each model is detecting.

The best performing class from the six will then be used for Experiments 1 and 2 to follow.

4.3 Experiment 1: Hyper-parameter Tuning

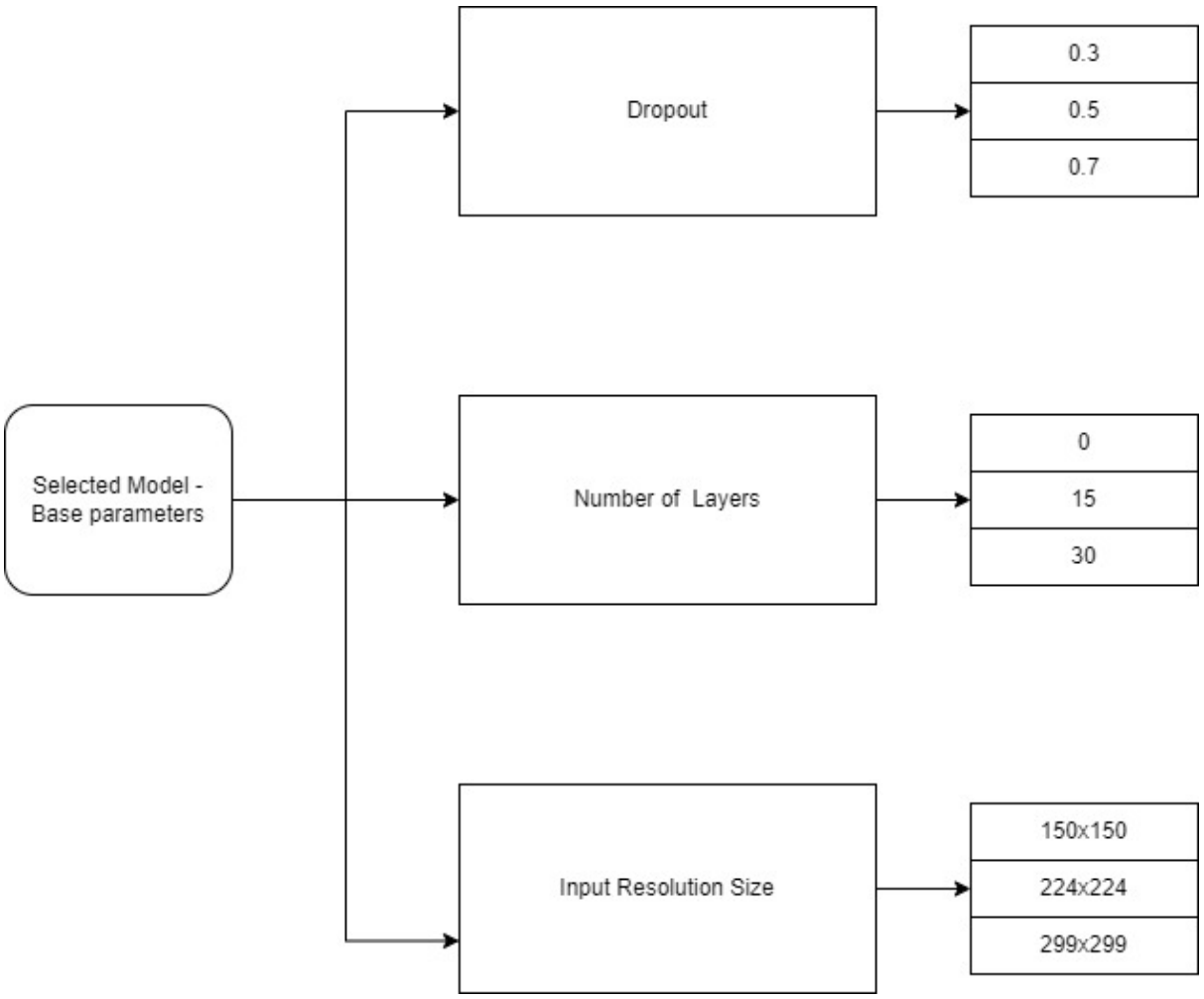
For the Baseline and Transfer Learning models presented in Section 4.1, parameters are selected from literature and experimentation. However for the Optimised Model, important parameters will be tuned in a systematic way. A choice of the three most important parameters will be made based on literature in Section 2.3 and based on experimentation. The model designed in Section 3.8 will be tuned by manually changing each parameter and leaving all others the same.

These parameters are selected to be:

- Dropout - due to the consistent presence of over-fitting in the results seen noticeably in 5.1, different parameters of dropout will be selected ranging from 0.3 to 0.7. Dropout was reviewed in 2.3.2. Higher dropouts might lead to worse model performance over time.
- Input image size - It has been determined through experimentation that the models often struggle to learn features from certain images as can be seen in B.2. It is possible that the input image resolution is too low resulting in feature loss. Therefore three resolutions were tested: 150x150, 224x244 and 299x299. ResNet-50 is able to accept these three sizes [7].
- Number of layers - It is observed by accuracy linearly increases with the number of layers in a network. The number of unfrozen layers will be set to 0, 15 and 30 [60].

The process flow is presented below for this experiment:

Figure 4.2: Experiment 1: Hyper-parameter tuning



Each parameter will be trained separately with control values in the other parameters. This prevents the unnecessary overhead of training 54 different models and will result in 12 additional models which will be graded in terms validation accuracy, validation loss, F1-Score, Top-5 Accuracy and Top-10 Accuracy and the best parameter choices will be selected.

The hyper-parameters chosen were determined through a combination of empirical testing and using grid search and random search. However it is acknoweldged that more sophisticated optimisation techniques could potentially yield greater performance. Techniques like Particle Swarm Optimization (PSO), Bayesian Optimization, and Genetic Algorithms (GA) are advanced methods that can efficiently navigate the hyper-parameter space.

PSO mimics the social behavior of birds flocking or fish schooling to find the most optimal solution, while Bayesian Optimization uses probabilistic models to make intelligent guesses

about which hyper-parameters might actually perform best, and then iterates this process. Genetic Algorithms employ mechanisms inspired by biological evolution, such as mutation, crossover, and selection, to evolve a set of hyper-parameters towards better performance [77].

These techniques could significantly enhance the model's predictive ability, by potentially finding a more optimal set of hyper-paramters than those derived from simpler methods. However they fall out of the scope of this study, but should be considered for future refinements.

4.4 Experiment 2: Model Generalisation

In order to test and define model generalisation, additional classes will need to be collected. These classes will not form part of Fynbos Dataset A or B, but will be instead used for this experiment. The chosen model from 4.2 will then be trained on a new dataset with all classes from Fynbos Dataset B merged with the 5 new classes. The model will then be evaluated and compared against the model without the five new classes. Based on the experimental results in 5.1, 5.2. 5.3, the Image Quality Assessment from 3.5 was invaluable in improving model performance. It will therefore be required for the 5 new classes to pass the criteria of the assessment before being used as training data. In order to avoid bias being introduced, the class size of each of these five classes is limited to 15. Images that fail the assessment were discarded, and if one of the classes had less than 15 images, it was also discarded.

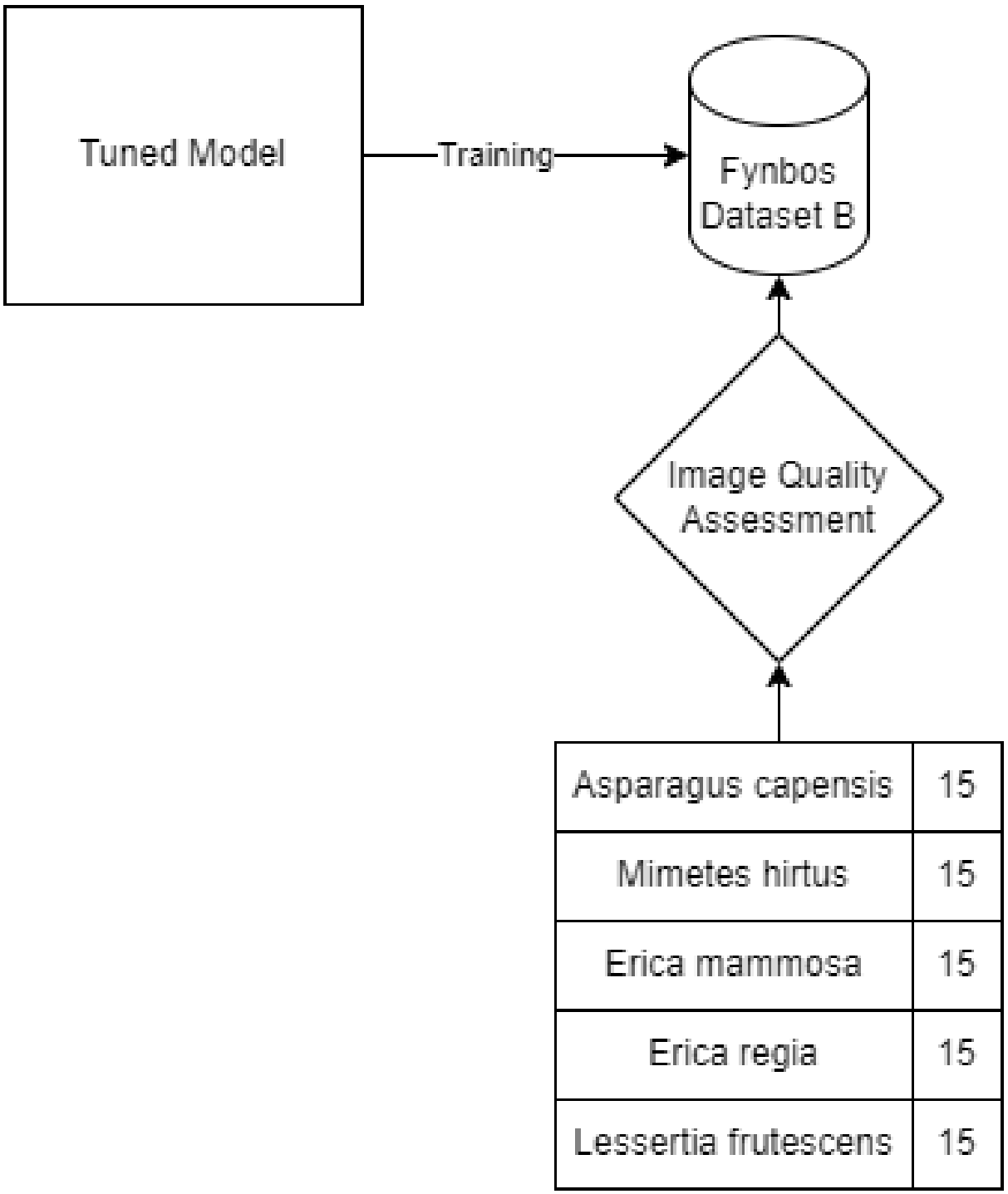


Figure 4.3: Experiment 1: Model Generalisation

The new species will not be added to Fynbos Dataset A or B. They are strictly for this experiment. The new model will be evaluated on validation accuracy, F1 Score, Loss and Top-k accuracy. Should the model be sufficiently generalised, the model performance should not drastically change during these tests. This experiment is vital to building trust in FLORA as an application as it should be expected to to perform robustly when trained on new data.

4.5 FLORA System Design

The eventual aim of FLORA is to be utilized as a classification tool in the field. Based on the experimental results in this project a suitable model will be chosen and then housed in a web-based application. As part of this project a suitable front-end was designed and is presented in this section.

The key development goal for the front end was to create an intuitive and responsive user interface. In order to achieve this, Angular, a platform framework for building applications using HTML, CSS and Typescript was chosen due to its comprehensive set of features and tools, and because of its scalability.

1. FLORA is designed to have two separate access points. The first access point will be the public access to the application where a simple drag and drop allows an image to be uploaded. This image is then resized and sent to the back-end where the chosen model will perform classification. A multi-class classification report is then returned to the user. This report will show the Top-1 Accuracy, the top-5 accuracy and corresponding class images. This can assist the user in determining whether or not the model is predicting the correct class. The uploaded image will not be uploaded into the Database and used for model training as the image will not be labeled and even a high accuracy is not fool-proof.

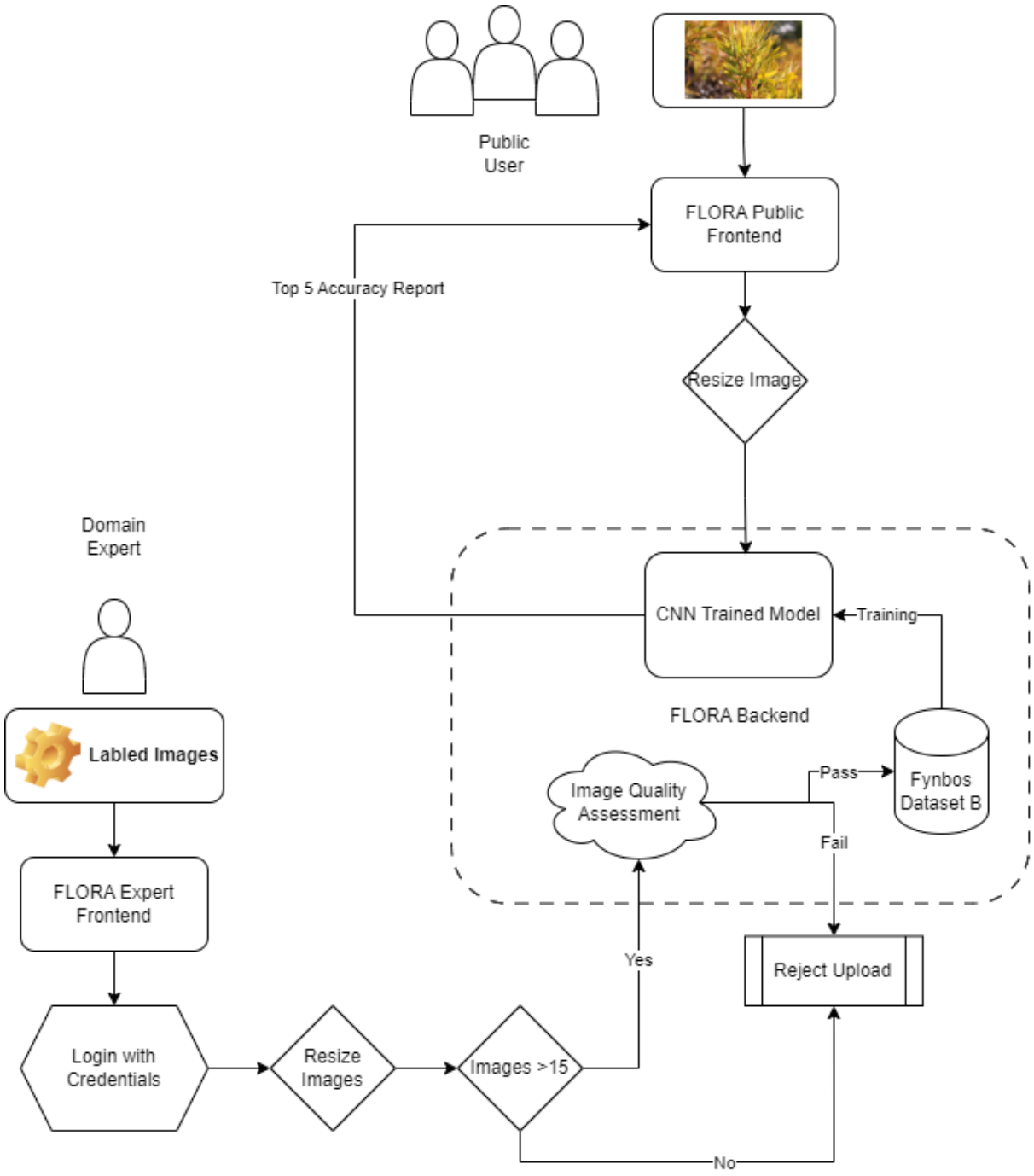
2. The second user interface will be for registered domain experts to upload new training data. The users will receive credentials that allows access to this portal, and will be required to upload at least 15 images of a Fynbos species and provide a label. Building trust in machine learning models starts with shared domain knowledge and this project demonstrates the importance of high quality, well-labeled data for use in training. Provided these uploaded images pass the Image Quality Assessment, they can be added to the training data and FLORA becomes capable of recognising a new species. This would need to be performed for all 9000 Fynbos species and continuous monitoring and adjustment of system performance will need to be carried out.

3. In this way, FLORA can grow organically. The experimental results showed suitable model generalisation to new species, and validated the Image Quality Assessment. This project provides a suitable starting point to eventually cover all Fynbos species, and eventually the entire CFR.

4. After experimentation it was determined that 15 images was suitable for the model to start learning basic features. Any fewer, and the data quality would pollute the

results with background noise. 15 images allows for suitable feature learning, and any less results in worse overall model performance.

Figure 4.4: FLORA Application Design



The key aspect to this model is the ability for domain experts to increase the training data with new labeled images. In theory, the model will be able to automatically retrain whenever new data are added and a live dashboard to monitor the model performance

will need to be created to monitor for irregularities.

4.5.1 Front-end

The application was structured around a series of Angular components that correspond to each stage of the front-end. These components include a home page with a drag and drop function, a classification report generated for the image uploaded, a separate (protected) user-interface for domain experts to upload labeled training data, validation of this upload and an Image Quality Assessment Score calculation. These features are presented below in a series of screenshots.

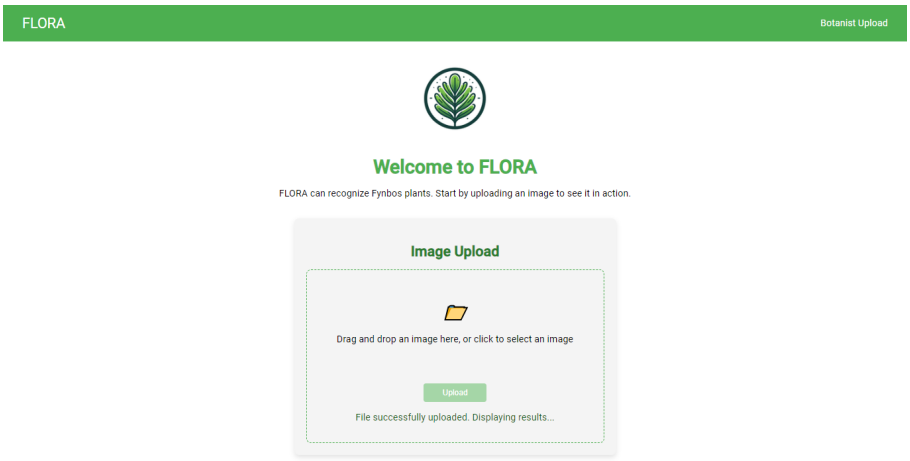


Figure 4.5: FLORA Homepage

The FLORA homepage is designed to be minimalist and clean. The user simply needs to drag and drop or manually upload an image into the box on the screen.

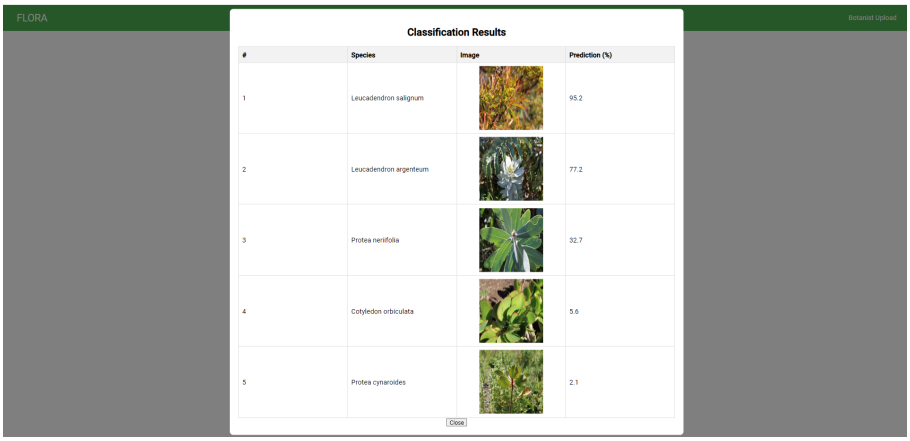


Figure 4.6: FLORA Classification Report

Once the user uploads an image, the application resizes the image in the background, and passes it to the model where classification is performed. The top 5 classes by classification accuracy are returned with the class name and classification accuracy in descending order.

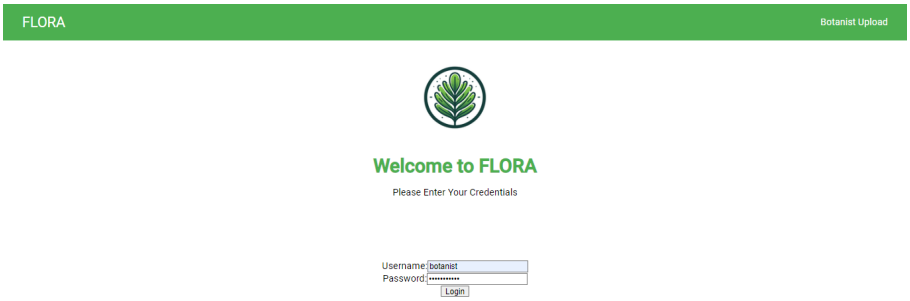


Figure 4.7: Domain Expert Login Page

The alternative access point for FLORA is locked behind a lock in screen. Users with access will be able to add images to the training set, so this needs to be locked away from all users.

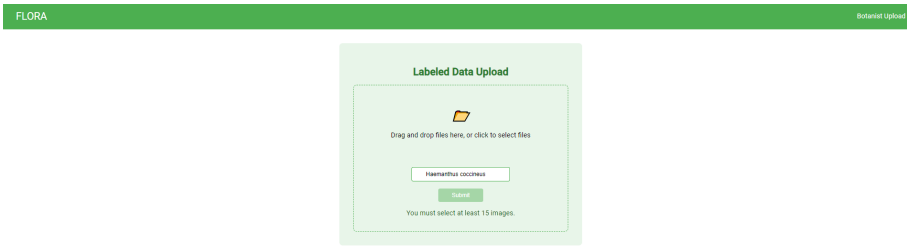


Figure 4.8: Image Count Check for Labeled Input

The input to the new labeled classes has requirements including the number of images being uploaded.

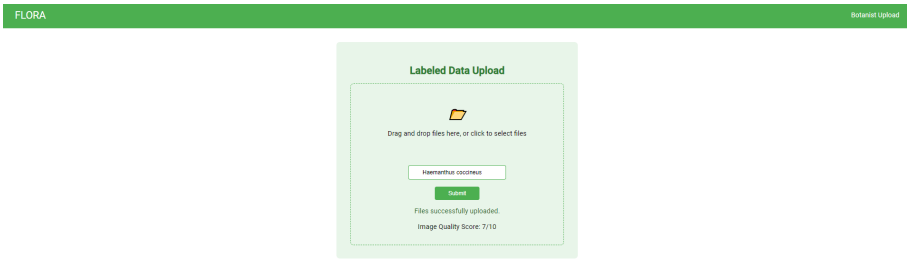


Figure 4.9: Image Quality Assessment Score after successful labeled data upload

Finally the new class is passed through the image quality assessment in the back-end and the score is returned. This score needs to exceed the threshold of 2 to be included into the training data.

This process suitably describes the main front-end components involved in FLORA. As stated earlier, the full application would require interaction between these components, a database of images for training and the selected model from this project all interacting together.

The overall front-end design scales to mobile phones because of Angular's built in auto-scaling parameters which were built into the CSS files. The mobile version of FLORA is visualised using Google Chrome's Developer Tools which allows for the application across viewing alternative devices. Note this is not a separate mobile application, rather it is how a smart phone would access FLORA via a browser The screenshot is presented

below:

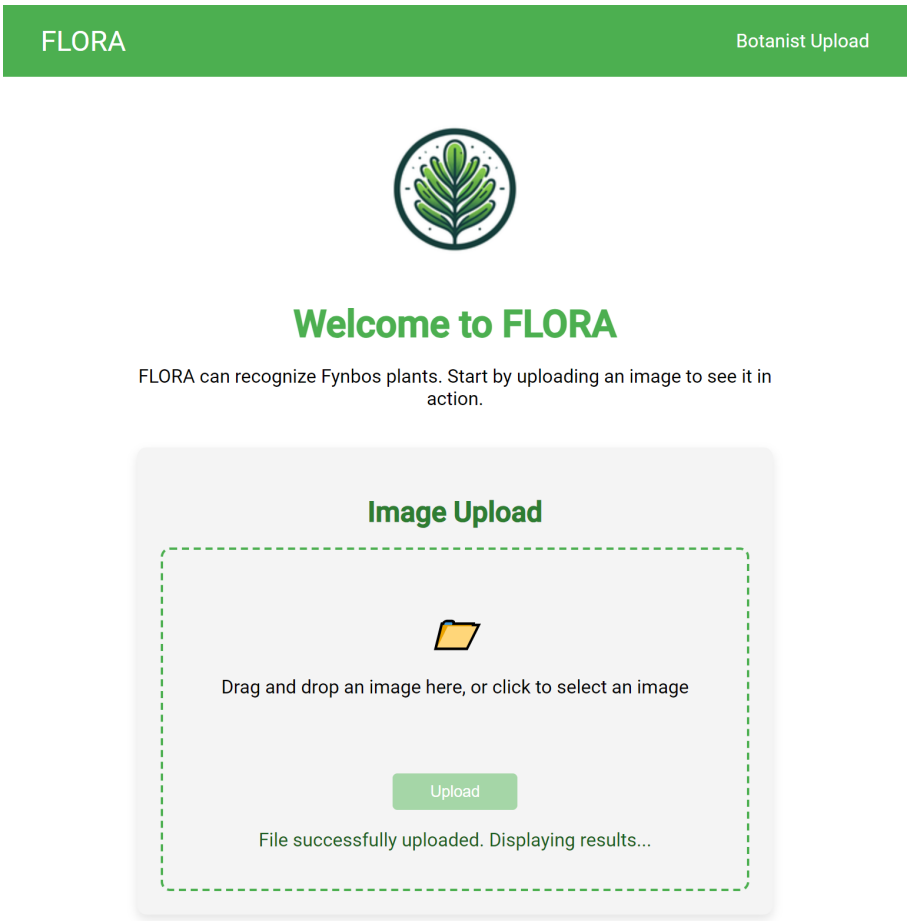


Figure 4.10: FLORA mobile version

The code for the application is included in the Github repository.

Back-end Design

Although the development of back-end functionality is beyond the scope of this project, the following considerations are outlined for potential future work:

- **Server-Side Deployment:** The trained CNN model can be deployed on a cloud-based server, such as Amazon Web Services (AWS) or Google Cloud, to handle image classification requests. This approach would enable scalability and real-time processing, ensuring that the system can manage a large number of requests efficiently.
- **Database Integration:** A back-end database, such as PostgreSQL or MongoDB, could be employed to store user data, classification results, and a library of reference images. This would facilitate the management and retrieval of data, enhancing the overall functionality and user experience of the application.
- **API Development:** The development of RESTful APIs would allow communication between the front-end and back-end components of the application. These APIs would enable the transfer of image data to the server for classification and the subsequent retrieval of results.
- **Security and Authentication:** Implementing security protocols, such as OAuth, is crucial to protect user data and control access to the model. Ensuring that the application adheres to best practices in security would be essential for maintaining user trust and safeguarding sensitive information.

These back-end functionalities, while not developed within the scope of this project, are essential for the full deployment and operation of the application in a real-world setting. Future work should focus on integrating these components to create a fully functional implementation of FLORA.

Chapter 5

Results

The results are divided into three sections that encompass the experimental design procedure used to create the final model. The results are designed to be iterative, with results from the earlier models influencing design choices in the later models. For convenience, this section is split up as follows:

- Each of the six models presented in 4.2 are evaluated. Their accuracy, loss, top-k accuracy, F1-scores, classification reports and confusion matrices are presented and discussed as well as the SHAP analysis for relevant classes. This section encompasses 5.1, 5.2 and 5.3 and consists of the bulk of this section.
- The best performing model from six presented will then undergo hyper-parameter tuning following the experimental procedure in 4.2.
- The tuned model will then follow the experimental procedure in 4.2 to assess its ability to generalise to new data.

5.1 Baseline Model

The Baseline Model results for Baseline A and Baseline B are presented here.

5.1.1 Model Evaluation

Accuracy - Baseline A and Baseline B

The first metric to look at it is accuracy. Accuracy as a whole is a good starting point when it comes to model evaluation, but it does not provide the full picture as to model performance. Accuracy on the training set is measured and compared against accuracy of the validation set, for both Datasets A and B and the results are presented below:

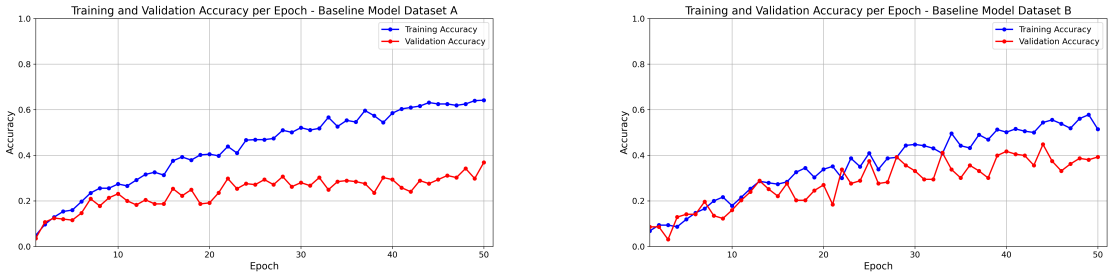


Figure 5.1: Training and validation accuracy of the Baseline Model over a series of 50 epochs during the training process for Dataset A (a) and Dataset B (b)

Baseline A trained on the raw, uncleaned images in Dataset A, has a smooth training accuracy curve which consistently increases over time showing that the model is capable of learning features in the data, albeit very slowly. The validation accuracy is lower than the training accuracy and the gap between the two measures widens over time which is a strong indication that the model is **over-fitting**. It is likely learning background details and noise in the training set which negatively impacts its performance on the validation set and this can actually be observed in the SHAP analysis.

Baseline B on the other hand shows training accuracy steadily increases over time, which is a good sign that the model is learning from the data. However there appear to be large fluctuations which indicates there is large degree of variance in learning from one epoch to another. This is possibly due to the lower size of the dataset, as it is observed that the fluctuations for Dataset A are not as drastic. But the training accuracy is higher for Baseline B which indicates that the the model is better able to learn features from the cleaned dataset.

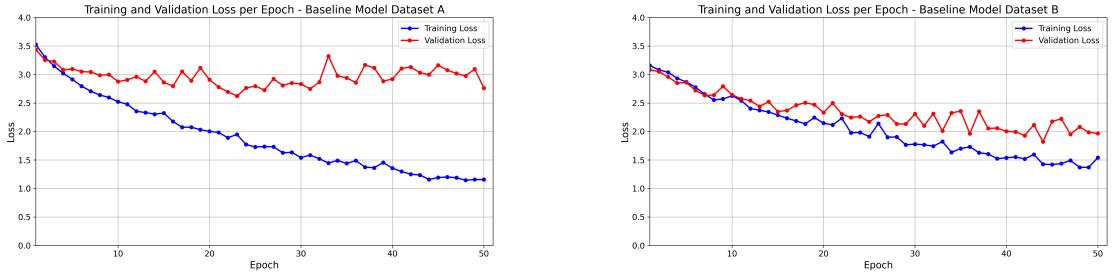
Both sets of results display over-fitting but it appears more pronounced with Baseline A. This is most likely due to the image background noise and overall lower quality of image which the model may be memorizing. The cleaned data in Dataset B seems to offer a more stable learning process although the accuracy is still not high when compared to

results achieved in literature. It appears that the data cleaning performed in 3.5, cleaned some of the noise but the essential features for classification might not be as prominent, or the model might not be deep enough to learn the more advanced features in the leaves.

Going forward, the over-fitting needs to be addressed. Techniques discussed in 2.3 like batch normalisation will need to be implemented for future models. Additionally, more violent data augmentation could help the model generalise better by providing more variety during training. Dropout was set to 0.5 in this model, but might need to be adjusted to improve the over-fitting. Despite there only being three layers in this network, the model shows that it is capable of learning.

Loss Graphs - Baseline A and Baseline B

The Loss Graphs provide insight into how well the model is doing it terms of minimising the loss function described in 2.3.2, which is the primary objective of the model. The loss for each Baseline model is shown below in Figure 5.2



(a) Baseline A

(b) Baseline A

Figure 5.2: Training and validation loss of the Baseline over a series of 60 epochs

For Baseline A the training loss shows a downward trend over time, indicating that the model is improving its prediction over time. It continues to decrease until the final epoch which suggests that over-fitting seen in Figure 5.1 is due to the dataset limitation and not the model architecture. The red line which depicts validation loss, should ideally decrease alongside the training loss. This would indicate good **generalisation** for the model. However the validation loss is noticeably higher than the training loss and it plateaus after a small, initial decrease. This does indicate a divergence between training and validation performance which is a sign of over-fitting. The model is likely memorising irrelevant features in training data which was also demonstrated in 5.1, and is then unable to apply the learnings to unseen data

For Baseline B however, the training loss trends downwards which shows that the model is able to learn from the training data. There are fluctuations similar to 5.1, which suggests that the learning is not stable. This indicates that the dataset might be too small and certain features are more pronounced than others. The validation loss is more volatile than the training loss and is generally higher. It does not show a clear downward trend over time - decreasing initially and then plateauing around the 30th epoch. It has several upward and downward fluctuations which indicates that the model's ability to generalise is poor. Baseline B is also over-fitting although not as severely for Baseline A, as indicated by the lower training loss compared to the validation loss. The fluctuations indicate that the model might be sensitive to certain features or noise in the validation set, or that the cleaning process has removed some variability that the model needs to generalize well.

Top 5 and Top 10 Accuracy The Top 5 and Top 10 Accuracy metrics are useful when dealing with classification tasks where the correct answer might not be the top choice of the model, but it's still within the top few predictions. This is very valuable due to the high number of classes and general class imbalance seen in 3.2.

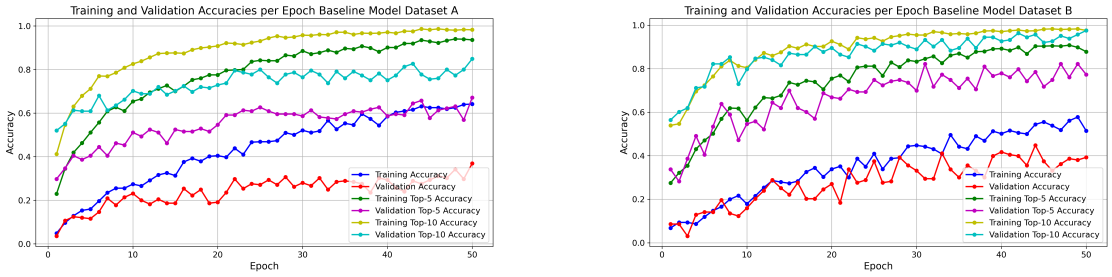


Figure 5.3: Top 5 Accuracy and Top 10 Accuracy and for the Baseline Model for training and validation over a series of 50 epochs during the training process for Dataset A and Dataset B

For Baseline A and Baseline B, the Top-5 Accuracy (green) is higher that the standard training accuracy which is expected as it is easier to be correct within 5 guesses than it is to be in 1. It shows a steady increase over time which is good for model performance as it means that the model is getting better at learning features. The Top-5 Validation accuracy (purple) fluctuates a bit but is higher than the standard validation accuracy. Again a gap between Top-5 Training Accuracy and Top-5 Validation Accuracy is seen which suggest over-fitting, but much less so than when considering only the top prediction. The Top-10 Accuracy likewise shows a similar trend, but again the gap between the validation and training lines show over-fitting. baseline B does show much greater fluctuations which indicates instability in the detected features.

The Top-5 and Top-10 accuracy metrics are considerably higher than the standard accuracy for both models with which indicates that while the model often gets its first prediction wrong, it often has the correct class within its top few predictions. This is a positive indication that the model is capable of learning despite its limited depth.

5.1.2 Baseline Model Classification Reports

The Classification Report for Dataset A is presented below in Table 5.1. The various metrics are explained in 2.3.4. Support refers to the the number of samples that appeared in the validation set for that particular class.

Table 5.1: Baseline Model A Classification Report

Species	Precision	Recall	F1-score	Support
Agathosma serpyllacea	0.00	0.00	0.00	3
Aloe arborescens	0.25	0.33	0.29	6
Arctotis stoechadifolia	1.00	0.25	0.40	4
Aristea capitata	0.50	0.25	0.33	4
Baloskion tetraphyllum	0.40	0.50	0.44	4
Carpobrotus chilensis	0.00	0.00	0.00	5
Carpobrotus edulis	0.00	0.00	0.00	4
Cotyledon orbiculata	0.25	0.11	0.15	9
Curio talinoides	0.62	0.62	0.62	8
Erica arborescens	0.57	0.80	0.67	5
Erica cinerea	0.00	0.00	0.00	7
Erica discolor	0.57	0.67	0.62	6
Erica duthieae	0.00	0.00	0.00	3
Erica perspicua	0.00	0.00	0.00	5
Gazania rigens	0.50	0.20	0.29	5
Grevillea banksii	0.00	0.00	0.00	8
Helichrysum petiolare	0.70	0.64	0.67	11
Leucadendron argenteum	0.30	0.60	0.40	10
Leucadendron laureolum	0.00	0.00	0.00	7
Leucadendron salignum	0.88	1.00	0.93	7
Leucospermum cordifolium	0.00	0.00	0.00	10
Leucospermum oleifolium	0.00	0.00	0.00	10
Lithospermum ruderale	0.10	0.50	0.17	2
Melianthus major	0.12	0.20	0.15	10
Mutisia orbignyana	0.20	1.00	0.33	2
Oscularia deltoides	0.56	0.82	0.67	11
Osyris lanceolata	0.33	0.33	0.33	3
Ozothamnus leptophyllus	0.50	0.67	0.57	3
Pelargonium crispum	0.30	0.60	0.40	5
Protea aurea	0.00	0.00	0.00	6
Protea cynaroides	0.15	0.29	0.20	14
Protea neriifolia	0.00	0.00	0.00	9
Protea repens	0.31	0.50	0.38	8
Schoenoplectus californicus	0.33	1.00	0.50	1
Strelitzia reginae	0.67	0.40	0.50	10
Accuracy			0.33	225
Macro avg	0.29	0.35	0.29	225
Weighted avg	0.29	0.33	0.29	225

Analysing the report for Baseline A first, several species like *Agathosma serpyllacea*, *Carpobrotus chilensis*, *Erica duthieae* and *Leucospermum cordifolium* have a precision, recall, and F1-score of 0.00, meaning the model is unable to correctly identify any samples of these species. This is possibly due to data quality but is also because there are too few samples for those classes. This is a problem of data imbalance. There are 12 classes with a zero for their F1-score. These classes need to be investigated in the SHAP analysis as they could be victims of poor data quality in which the three layered model (shown in Section 3.6) lacks the depth to identify features contained within. *Leucospermum*

oleifolium for example is well represented in the dataset and has suitable morphological features which should be detected by the model.

With just three layers in the network - coupled with no regularisation techniques apart from dropout - the uncleaned data could be having a signification impact on the poor model outcome. The model is likely memorizing irrelevant background features and struggling with actual leaf features. Another flaw could be that the image resolution resize of 150x150 is just too small for complex Fynbos images and this parameter should be adjusted as features are being lost.

Some species like *Leucadendron salignum* (F1-Score of 70) and *Osyris lanceolata* (F1-Score of 0.86) show relatively high F1-scores, which suggests that the model is capable of identifying features, learning and classifying those classes. These features will be highlighted by the SHAP analysis in 5.1.4.

The classification report for Baseline B is presented below in Table 5.2:

Table 5.2: Classification Report for Baseline Model B

Species	**Precision**	**Recall**	**F1-score**	**Support**
Aloe arborescens	0.50	0.33	0.40	6
Aristea capitata	0.25	0.25	0.25	4
Baloskion tetraphyllum	0.50	0.25	0.33	4
Carpobrotus edulis	0.20	0.25	0.22	4
Cotyledon orbiculata	0.25	0.11	0.15	9
Curio talinoides	0.67	0.75	0.71	8
Erica arborescens	0.33	1.00	0.50	5
Erica cinerea	1.00	0.43	0.60	7
Erica discolor	0.46	1.00	0.63	6
Helichrysum petiolare	0.75	0.27	0.40	11
Leucadendron argenteum	0.53	1.00	0.69	10
Leucadendron laureolum	0.50	0.14	0.22	7
Leucadendron salignum	0.78	1.00	0.88	7
Melianthus major	0.50	0.20	0.29	10
Mutisia orbignyana	0.29	1.00	0.44	2
Oscularia deltoides	0.60	0.82	0.69	11
Osyris lanceolata	0.75	1.00	0.86	3
Ozothamnus leptophyllus	0.40	0.67	0.50	3
Pelargonium crispum	0.00	0.00	0.00	5
Protea cynaroides	0.40	0.29	0.33	14
Protea neriifolia	0.00	0.00	0.00	9
Protea repens	0.11	0.12	0.12	8
Strelitzia reginae	0.62	0.50	0.56	10
Accuracy			0.46	163
Macro avg	0.45	0.49	0.42	163
Weighted avg	0.47	0.46	0.42	163

For Baseline B the model displays a noticeably more heterogeneous performance which is expected as the model is trained on the cleaned dataset. Species like *Oscularia deltoides*, *Osyris lanceolata*, and *Leucadendron salignum* show high precision, recall and F1-scores now whereas for Baseline A, *Oscularia deltoides* and *Leucadendron laureolum* had F1-scores of 0.67 and zero respectively. *Oscularia deltoides* now shows a high precision of 0.79 and a recall of 1.00 which equates to an F1-score of 0.88. *Osyris lanceolata* similarly has a high precision and recall (0.75 and 1.00 respectively) resulting in an F1-score of 0.86. These high values suggest strong predictive capability of the model for those particular classes. This indicates that the model has actually been able to learn distinguishing features of these species and is not learning irrelevant background features as it was for Baseline A.

Conversely, two species have F1-scores of zero - *Protea neriifolia* and *Pelargonium crispum*

There are no true positive predictions which highlights potential issues like a lack of distinctive features, class imbalance or insufficient training samples. The confusion matrix will reveal more information on these classes.

The overall accuracy of the model is 0.32, which while an increase from Baseline A, shows severe room for improvement. The class support varies significantly from 2-14 instances across the different species which affects the reliability of precision and recall for classes with very few samples such as *Osyris lanceolata*. It has a high precision and recall, but it only has a 3 for class support which means the model could very well be randomly predicting them.

Classes with high performance will be analysed in the SHAP analysis to decipher which key features are contributing to the model's success, and which could be potentially leveraged to boost the performance in under-performing classes.

5.1.3 Confusion Matrix

The confusion matrices are presented below for both Baseline A and Baseline B. Confusion matrices assist with unpacking the classification report and showing which classes were mis-classified as others. This assists with identifying model weaknesses. The confusion matrix for Baseline A is presented below in 5.4:

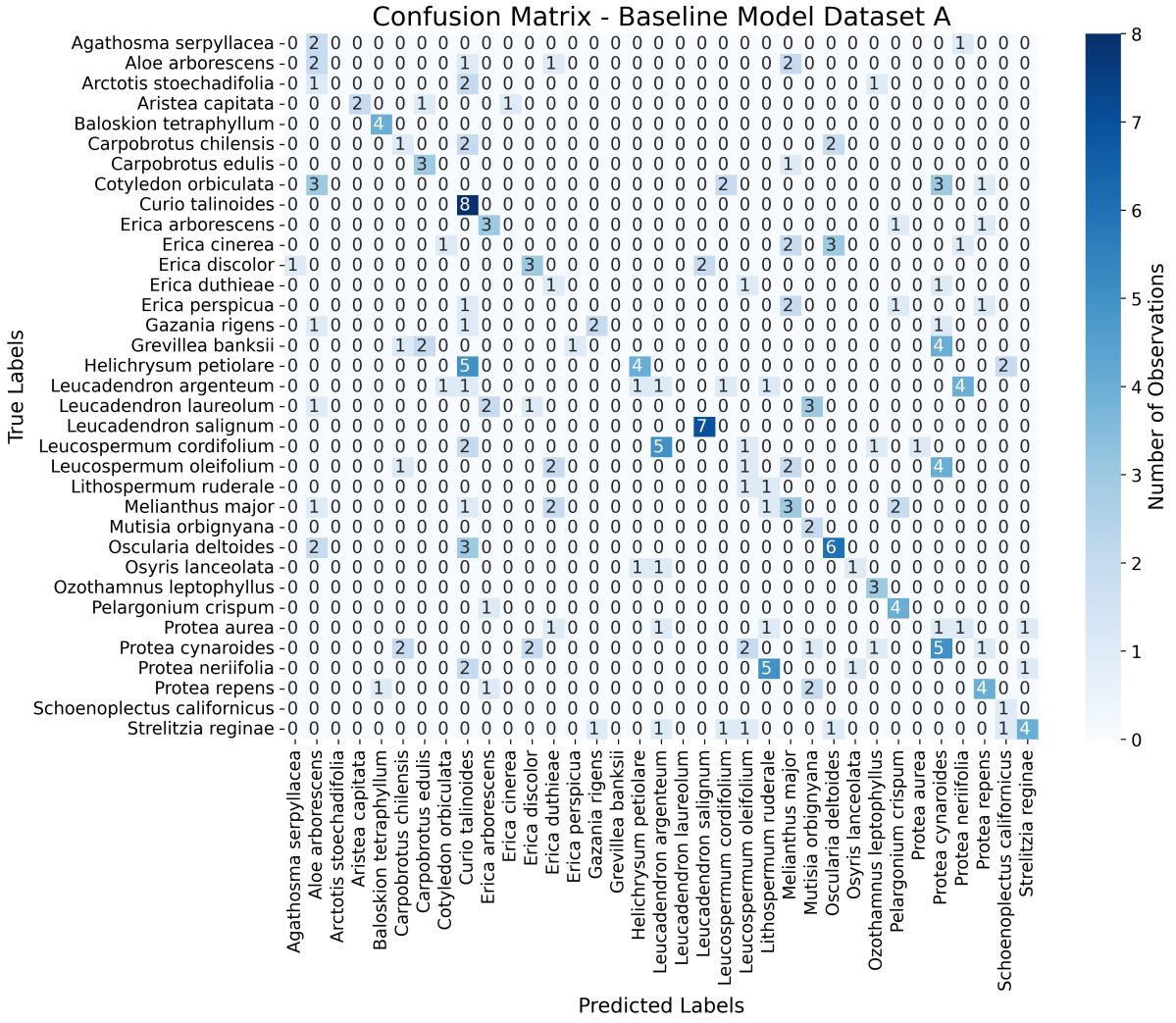


Figure 5.4: Confusion Matrix for Baseline A

The diagonal for Baseline A shows a substantial number of zeros across the majority of species, indicating numerous instances where the model failed to correctly identify the true class. There are some notable exceptions where the model has performed reasonably well, such as for *Leucadendron salignum*, with 7 true positives out of 9 actual instances. However, the confusion matrix also reveals several mis-classifications, with certain species such as *Erica discolor* and *Erica cinerea* having a relatively higher number of false positives, as they were often predicted incorrectly when other species were the true labels. This indicates that these classes have features that are common to other classes, something that was expected from 2.1.2 where it was discussed that Fynbos species are often similar to one another. The model as is, might be too simple to capture the detailed nuances between species and a deeper model will need to be built. The diagonal line representing the True Positives for this model is shown below:

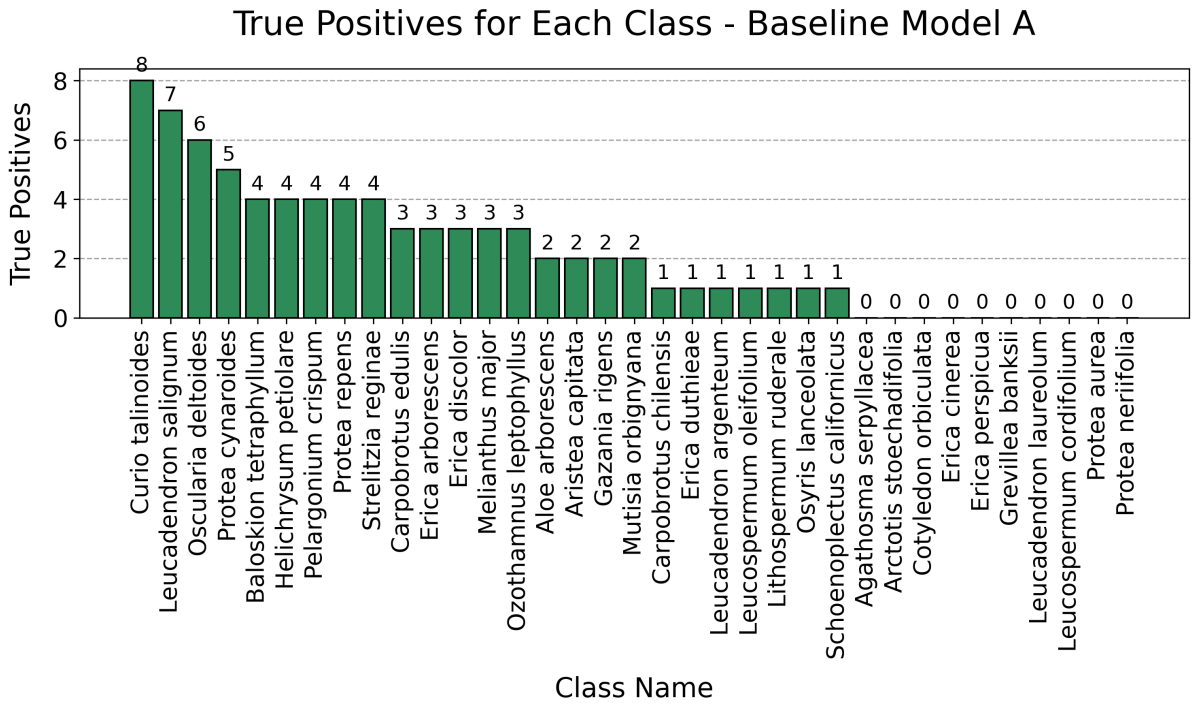


Figure 5.5: True Positives for Baseline A per class

Overall there is significant variability across classes with 10 classes having zero True Positive predictions and seven classes having just one. Many of these under-performing classes ultimately get removed by the Image Quality Assessment and as a result a noticeable improvement can be seen in Baseline B's confusion matrix shown below in Figure 5.6:

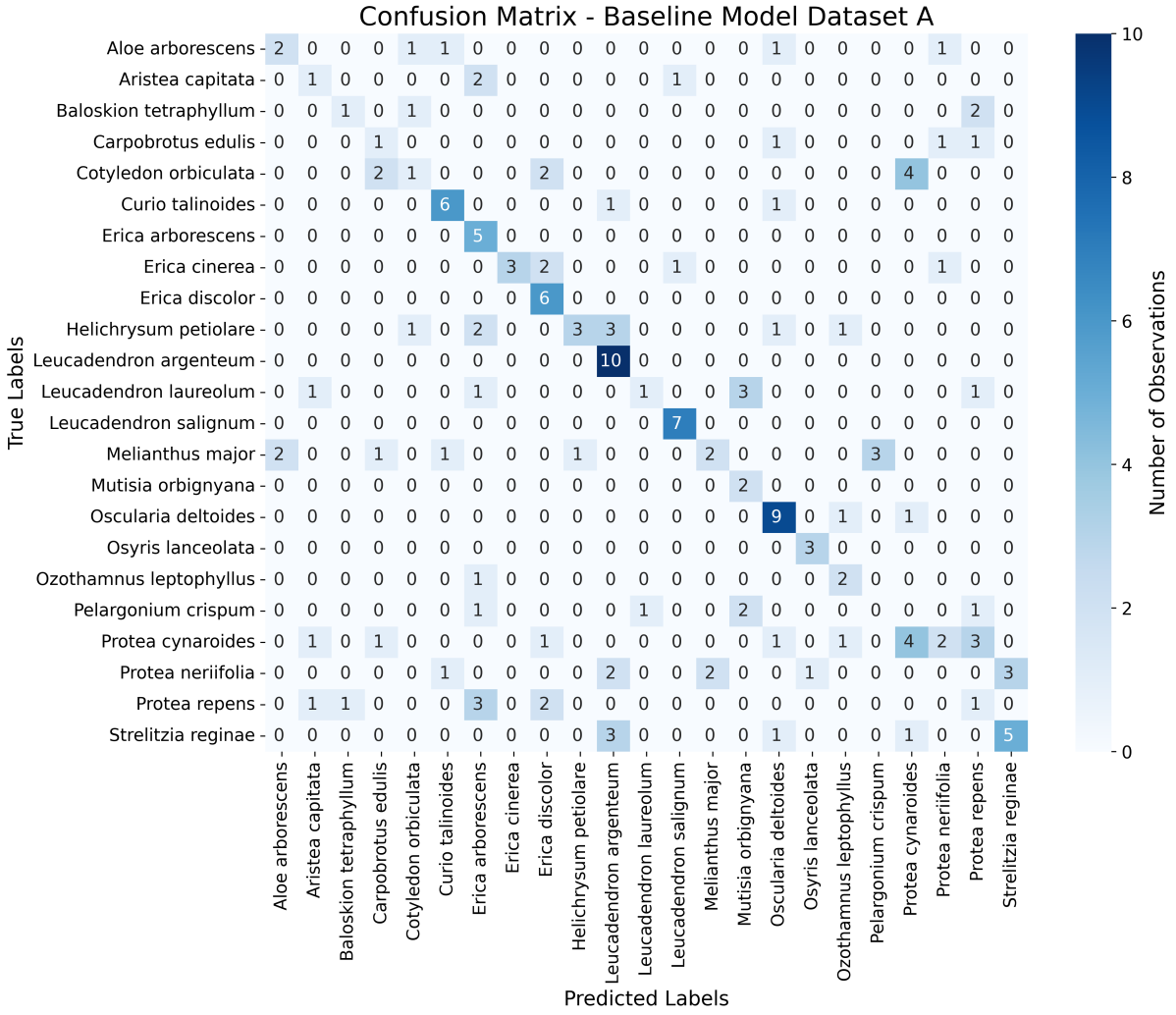


Figure 5.6: Confusion Matrix for Baseline B

Baseline B's confusion matrix shows a slightly different pattern. It also displays a significant number of zeros, but there are species for which the model has predicted more consistently correct classifications. For example, *Oscularia deltoides* and *Leucadendron salignum* exhibit a stronger diagonal presence, suggesting that the model has a better grasp of the features that define these species. However, similar to Baseline A, there are still many mis-classifications present, as well as instances where the model has failed to recognize the correct species entirely. The True Positives distribution is presented below:

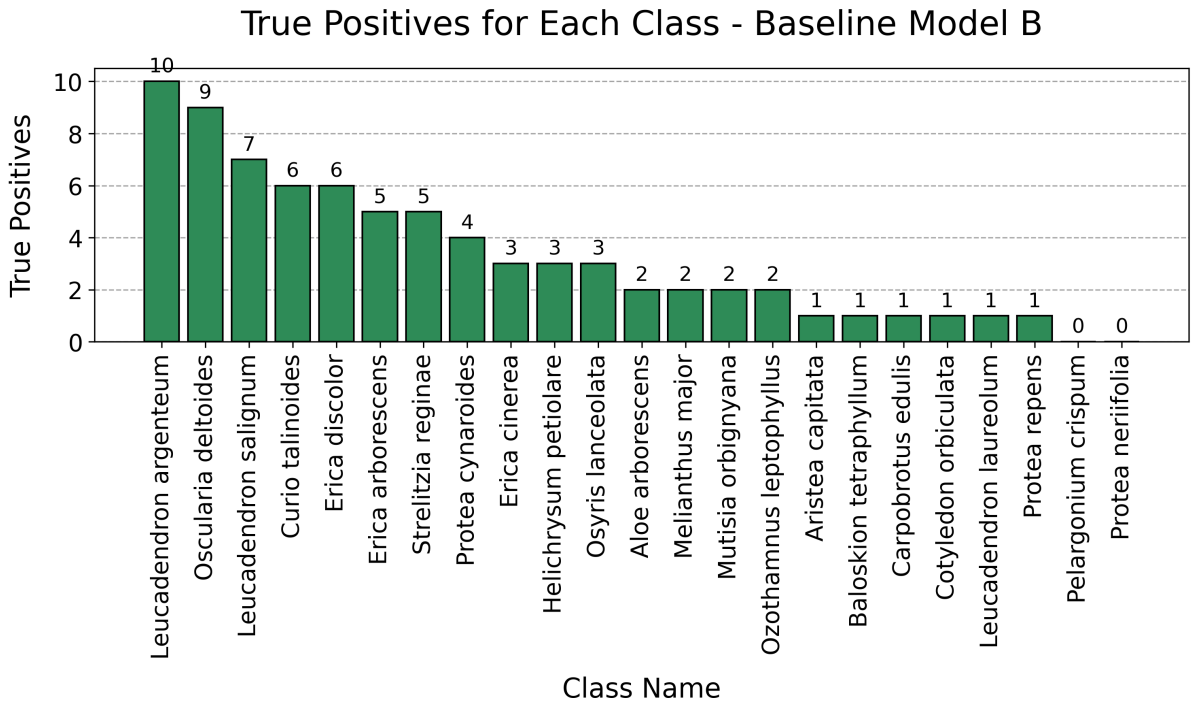


Figure 5.7: True Positives for Baseline B per class

Unlike in Baseline A, there are now only two classes that have no True Positives. The model still overall has fairly weak prediction capability, but is at least able to identify one sample from the majority of species.

Comparative Analysis:

Both models show signs of difficulty in classification with notable amounts of either mis-classified or unclassified instances. Baseline B does show slightly improved model performance with more true positive predictions, which is consistent with the higher overall accuracy reported in the classification report for Baseline B. The true positives for Baseline B are more spread out across the different species than in Baseline A, which indicates better model generalisation when trained on cleaned data.

Both matrices demonstrate cases of mis-classification but the patterns diverge. Baseline B has certain species like *Oscularia deltoides* which have a high number of true positives and fewer false positives and false negatives compared to Baseline A, where the correct predictions are lower and more spread out.

Some species, such as *Aloe arborescens*, appear to be difficult for the model to classify in both datasets, as indicated by the lack of true positives. These species will need to be

analysed in the SHAP analysis in 5.1.4.

5.1.4 Baseline Model SHAP Analysis

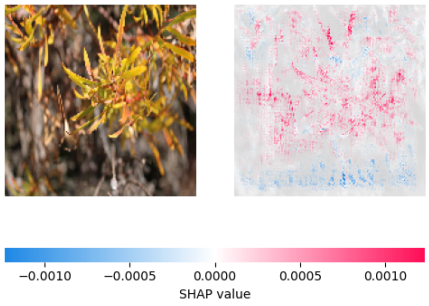
SHAP (SHapley Additive exPlanations) values decompose a prediction to show the impact each feature has on the final prediction. In the context of FLORA, SHAP values can highlight the regions the leaf images that significantly contribute to the model's prediction. This allows for a visualisation of the high performing and low performing classes, which is invaluable to the iterative design. the SHAP analysis is model agnostic and has no impact on the predictive ability of the model as discussed in 2.3.4.

For better readability, the Baseline A SHAP plots are moved to the appendix in B.

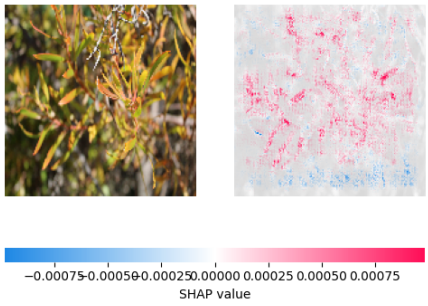
Baseline B

High performing Classes

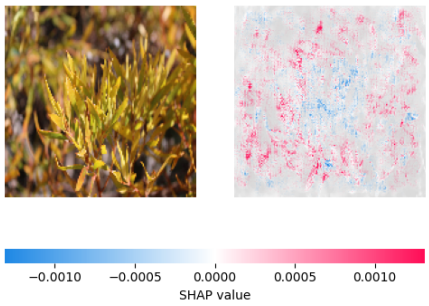
Just like in Baseline A, *Leucadendron salignum* is again a high performing class. For Baseline B, it has just one true negative (mis-classified as *Erica discolor*) and has an F1 score of 0.88. The SHAP plots are presented below:



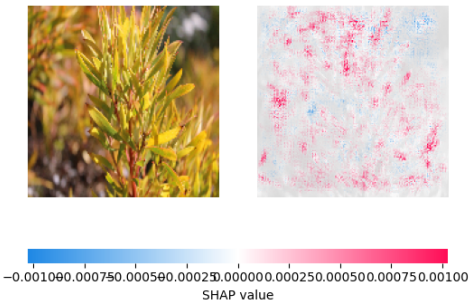
(a) *Leucadendron salignum* SHAP i



(b) *Leucadendron salignum* SHAP ii



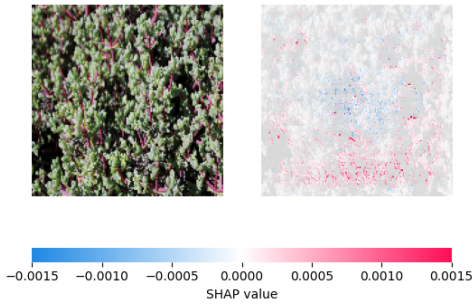
(c) *Leucadendron salignum* SHAP iii



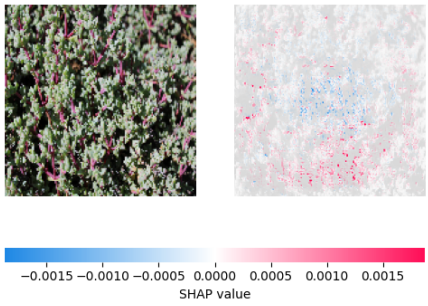
(d) *Leucadendron salignum* SHAP iv

Figure 5.8: *Leucadendron salignum* leaves subset SHAP Values from Baseline B

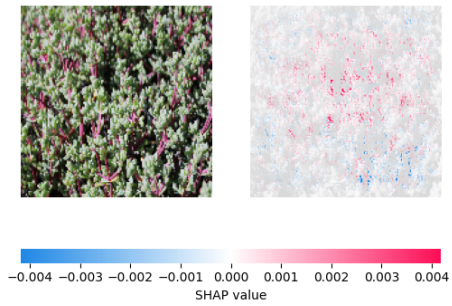
Visually the plots are similar to B.1 with distinctive leaf features being extracted and learned by the model. The plots show more intensity for Baseline B, indicating that the model is extracting clearer features. Again the colour seems to be preserved as brighter portions of the plant show higher SHAP values, meaning they contribute more to positive predictions. Contrasting this against B.1, it can be observed that the features in this SHAP plot are more focused and less spread out. In this model, the individual leaf shape is being learned contrasted against the leaf positions in Baseline A. This is likely an indication that there is less background noise in this dataset, and hence less noise learned during training. This was expected after the Image Quality Assessment and validates the use of it.



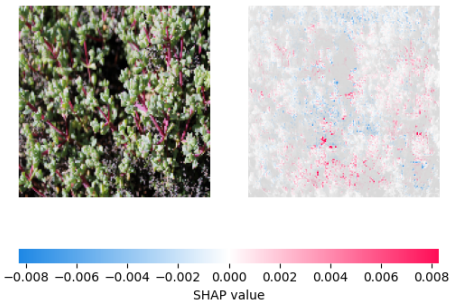
(a) *Oscularia deltoides* SHAP i



(b) *Oscularia deltoides* SHAP ii



(c) *Oscularia deltoides* SHAP iii



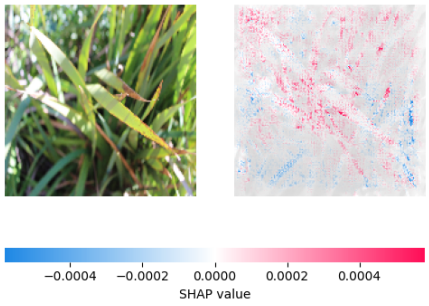
(d) *Oscularia deltoides* SHAP iv

Figure 5.9: *Oscularia deltoides* leaves subset SHAP Values from Baseline B

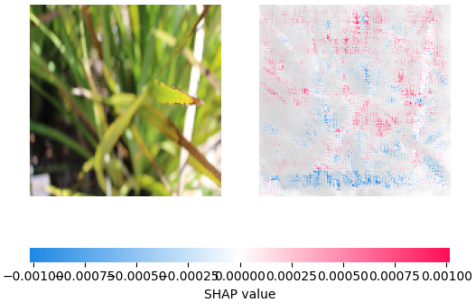
In Baseline B, *Oscularia deltoides* showed much improved performance with 11 true positives and an F1 score of 0.69 although there were still 7 mis-classifications. Looking at the SHAP plots above in 5.9, the model shows that it is actually learning both leaf and stem features, with the SHAP plot of Image iv displaying this with the intense pink colouring. The model appears to be capturing the clustering of the leaves, albeit on a simplistic level and this clustering appears to be common to several species, hence the mis-classification. The model is performing well on this species, and the differentiation should increase with deeper models, as the more complex shapes of the leaves will be identified. A possible problem here is also the low image resolution makes the individual features less defined which can be adjusted in later models.

Poor Performing Classes Baseline B is also performing poorly on*Aristea capitata*. There are 1 True Positive for this class in the confusion matrix and 2 False Positives. It

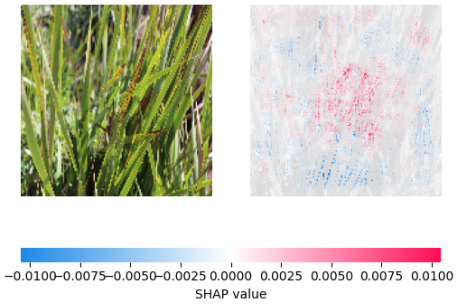
has an F1 Score of 0.25. The SHAP plots are presented below:



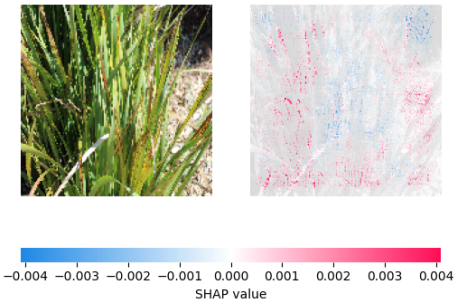
(a) *Aristea capitata* SHAP i



(b) *Aristea capitata* SHAP ii



(c) *Aristea capitata* SHAP iii



(d) *Aristea capitata* SHAP iv

Figure 5.10: *Aristea capitata* leaves subset SHAP Values from Baseline B

Looking at the SHAP plots it can be seen that Baseline B is struggling to really identify any features in this class. The SHAP plots show more intensity than the ones in Baseline A, particularity around the leaf structure in SHAP i, but the model is still learning background noise as evidenced in SHAP ii. This class is likely performing poorly because of lower image quality, so the threshold score in the Image Quality Index is perhaps too low at the moment.

5.1.5 Discussion

The Baseline Models overall performed fairly poorly. While the validation accuracy achieved by Baseline B sits at around 46% which is better than random guessing, it is

still far from suitable. The training of the models showed that the simplistic architecture of the models cause very slow learning times, although the learning does not plateau in Baseline B, speaking to the ability of the model to continuously learn from the training set. Top-5 and Top-10 accuracy are noticeably better sitting at 81% and 96% respectively. The model usually gets its prediction correct within 5 guesses and almost always within 10 guesses. This does show that the model is learning to differentiate between classes despite the poor top-1 accuracy. Despite all of this, there is clear over-fitting present. In this model the only guard against over-fitting was including a dropout layer set to 0.5

The improved performance in Baseline B from Baseline A validates the use of the Image Quality Assessment. The uncleaned dataset increases the presence of background noise in many of the images. The SHAP analysis of *Erica discolor* in B.2 shows this problem as the SHAP plot reveals that background features are contributing to the prediction. This is a common theme in Baseline A and directly explains much of the poor performance. By directly removing images with too much background noise, the number of True Positives in Baseline B, significantly improved. The model does however show that it is capable of learning fairly complex leaf features as shown by *Leucadendron salignum* in 5.8. However this does not appear to be the standard for other species. *Leucadendron salignum* likely has very favorable features for this particular model, including good lighting condition and distinct colour. The lighting conditions have significant impact on model performance with the irregular lighting in B.4 negatively influencing classification.

Mis-classification was fairly common for these models with both confusion matrices showing significant off-diagonal elements. Mis-classification is expected when model parameters are not optimised. In Baseline B, there is significant improvement in terms of the diagonal elements, showing better model performance.

5.2 Model 2: Transfer Learning Model Results

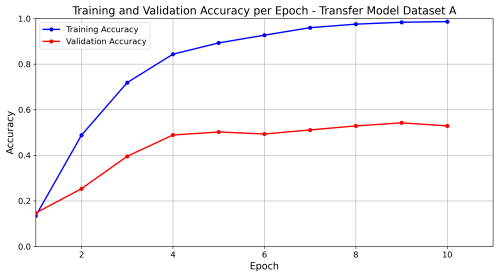
To address the limitations of the Baseline Model - the inability to detect complex features, long training time, the over-fitting and the focus on background noise - a second model was proposed in 3.7. This model makes use of Transfer Learning, using pre-trained weights, to create a more robust model. This model uses the ResNet-50 architecture which was trained on the ImageNet dataset [7], [30]. The expectation would be that the model's pre-training has already taught it simple to complex generic shapes, and by unfreezing and training the last 30 layers of the model on the Fynbos datasets, that this model would be adapted to now classifying the Fynbos species. The results are presented

in the same way as that of the Baseline models in 5.1, showing two models, A and B (trained on Dataset A and B respectively). The number of epochs is reduced to 10, with the expectation that the models will converge faster due to the pre-training.

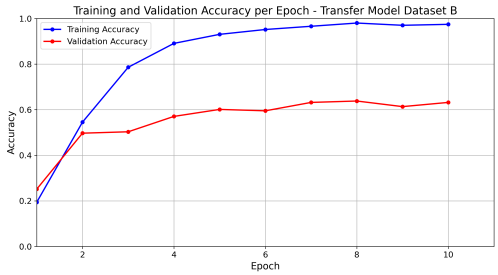
5.2.1 Performance Results

Accuracy

The accuracy graphs immediately show significantly more stability than the Baseline Models in 5.1. The accuracy converges fairly quickly in both models and 10 epochs proved more than adequate. The validation and training accuracy per epoch is presented below:



(a) Transfer Model A



(b) Transfer Model B

Figure 5.11: Training and validation accuracy of the Transfer Model over a series of 10 epochs during the training process for Dataset A and Dataset B

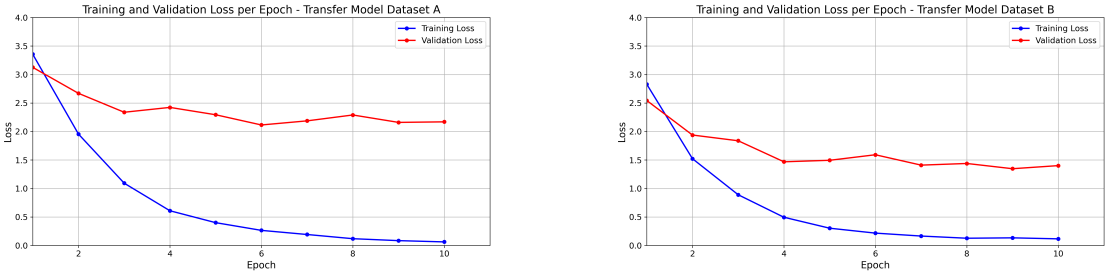
Transfer Model A and Transfer Model B demonstrated rapid improvement in training accuracy and reached near 90% by the fourth epoch, plateauing close to 99% by epoch 10. This is noticeably higher than either Baseline A or Baseline B which both seemed to plateau around the 60% mark after 50 epochs seen in 5.1. This suggests that the features which ResNet-50 has already learned from other datasets (the basic shapes at least), are well suited to the task of Fynbos feature classification. However the validation accuracy plateaued at a lower level (around 50%) starting from the fifth epoch which persisted until the tenth epoch. The gap between the training and validation accuracy is significant and again shows clear over-fitting, continuing the observation from the Baseline Models. The raw dataset trained on by Transfer Model A could have introduced complexities that the model could not generalise beyond the training set, even with the pre-set weights.

Transfer Model B showed a similar pattern with regards to validation accuracy, with the value plateauing at around 60% instead from the fourth epoch onward. However there is

a rapid increase in validation accuracy from the first to second epochs, suggesting that the model was able to generalise better and capture features in the cleaned dataset more easily. This again validates what was seen in the SHAP analysis from the Baseline Models. The gap between training accuracy and validation accuracy again shows significant overfitting. Both models show clear memorisation.

Loss

The loss graphs for the two models again shows much better stability than those seen in the Baseline Model. There is again a noted increase in the model trained on the cleaned dataset.



(a) Transfer Model A (b) Transfer Model A

Figure 5.12: Training and validation loss of the Transfer Model over a series of 10 epochs

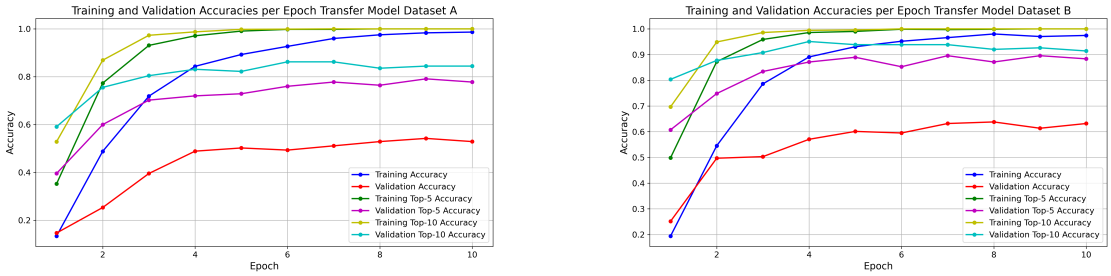
In the case of Transfer Model A, the training loss shows a steep decline which continues throughout the training process. It approaches an asymptote by the last epoch which is indicative that the model is growing in confidence in its training data predictions. It would likely continue to decrease were the epochs increased. This is expected with transfer learning models as they are capable of learning more complex shapes. The validation loss starts high, decreases until the third epoch and then follows a plateau from the fourth epoch onward. It does slightly improve in the last few epochs.

For Transfer Model B, the initial training loss decreases sharply from the first to second epoch mirroring the sharp ascent of validation accuracy in 5.11. it then gradually stabilises around the sixth epoch, which suggests that the model has reached convergence in terms of learning from the training data.

Both models exhibit a similar pattern where training loss continues to decrease and validation loss plateaus. Transfer Model B achieves a lower validation loss plateau (demonstrating superior performance), which suggests that the Image Quality Assessment

in 3.5 is beneficial to the model's ability to generalise.

The divergence between training loss and validation loss is an indicator of over-fitting again. Although both models exhibit this, it is more pronounced in Transfer Model A. The inability of the validation loss to track the training loss suggests that the model's improvements are not translated well to unseen data, implying poor generalisation.



(a) Transfer Model A (b) Transfer Model B

Figure 5.13: Top 5 Accuracy and Top 10 Accuracy for the Transfer Learning Model for training and validation over a series of 10 epochs

Top 5 and Top 10 Accuracy The Top-5 and Top-10 accuracy metrics are significantly higher than the top-1 accuracy for both training and validation. They both peak at around the sixth epoch in Transfer Model A. This indicates that while the model may not always be certain, it usually obtains the correct classification within its top few guesses which is a demonstration that model is actually learning features from the data. The high top-10 accuracy also suggests that the model has developed a broad understanding of general leaf shape of the Asteraceae, Ericaceae and Proteaceae species due to the deeper number of layers, but it might struggle with finer distinctions or features from other families.

Transfer Model B follows a similar pattern, but does show marginally higher validation scores. The gap between the training and test Top-k accuracy metrics again demonstrates that over-fitting is occurring.

The trend across both models shows that the model is more successful when evaluated using top-5 and top-10 accuracy metrics, which is expected as the scores are much less stringent than the top-1 accuracy. In both models, the validation accuracy does not reach the same level as training accuracy which indicates that over-fitting is still taking place. However he gap between training and validation is noticeably lower for Transfer Model B, indicating better model generalisation.

5.2.2 Transfer Model Classification Reports

The classification reports are presented for both models. There is a very significant improvement in F1-scores compared to the Baseline Models. The transfer learning approach as resulted in the increase in performance of numerous classes.

Table 5.3: Transfer Model Dataset A Classification Report

Species	**Precision**	**Recall**	**F1-score**	**Support**
Agathosma serpyllacea	1.00	0.67	0.80	3
Aloe arborescens	0.56	0.83	0.67	6
Arctotis stoechadifolia	1.00	0.75	0.86	4
Aristea capitata	0.75	0.75	0.75	4
Baloskion tetraphyllum	1.00	0.50	0.67	4
Carpobrotus chilensis	1.00	0.40	0.57	5
Carpobrotus edulis	1.00	1.00	1.00	4
Cotyledon orbiculata	0.60	0.33	0.43	9
Curio talinoides	1.00	0.88	0.93	8
Erica arborescens	0.40	0.80	0.53	5
Erica cinerea	0.00	0.00	0.00	7
Erica discolor	0.36	0.83	0.50	6
Erica duthieae	0.20	0.33	0.25	3
Erica perspicua	0.11	0.20	0.14	5
Gazania rigens	1.00	0.60	0.75	5
Grevillea banksii	0.67	0.25	0.36	8
Helichrysum petiolare	0.89	0.73	0.80	11
Leucadendron argenteum	0.40	0.20	0.27	10
Leucadendron laureolum	0.50	0.14	0.22	7
Leucadendron salignum	0.78	1.00	0.88	7
Leucospermum cordifolium	0.29	0.20	0.24	10
Leucospermum oleifolium	0.40	0.20	0.27	10
Lithospermum ruderale	0.50	0.50	0.50	2
Melianthus major	0.64	0.90	0.75	10
Mutisia orbignyana	0.40	1.00	0.57	2
Oscularia deltoides	0.79	1.00	0.88	11
Osyris lanceolata	0.50	1.00	0.67	3
Ozothamnus leptophyllus	0.75	1.00	0.86	3
Pelargonium crispum	0.44	0.80	0.57	5
Protea aurea	0.00	0.00	0.00	6
Protea cynaroides	0.47	0.64	0.55	14
Protea neriifolia	0.50	0.22	0.31	9
Protea repens	0.70	0.88	0.78	8
Schoenoplectus californicus	1.00	1.00	1.00	1
Strelitzia reginae	1.00	0.90	0.95	10
Accuracy			0.58	225
Macro Avg	0.62	0.61	0.58	225
Weighted Avg	0.60	0.58	0.56	225

The overall weighted accuracy for Transfer Model A is 0.58 which indicates that 58% of the predictions were correct across all classes. This a a notable improvement over

the performance of Baseline A which had an accuracy of 0.24. The macro-average precision and recall are almost the same at 0.62 and 0.61 respectively and the F1-score is 0.58. These numbers indicate a balance between precision and recall across all species. Species such as *Carpobrotus edulis*, *Curio talinoides*, *Helichrysum petiolare*, *Leucadendron salignum*, and *Strelitzia reginae* have high F1-scores which show a balance between precision and recall, showing noticeable improvement over Baseline A. The more sophisticated architecture is much better at classifying more species.

Certain species such as *Erica cinerea* and *Protea aurea* have very low scores across all metrics indicating that the model has difficulty correctly identifying these classes. Species that have a precision or recall of 1.00 but lower F1-scores like *Agathosma serpyllacea* and *Arctotis stoechadifolia* indicate that while the model is confident in its predictions when it does make them, it is not consistently able to recognise all instances of these particular species.

There is a very noticeable improvement in F1-score over Baseline A with there now being multiple high performing species. *Leucadendron salignum* has gone from having the best F1-score in the Baseline Models, to now being the joint third best.

The results for the model when trained on Dataset B are presented below in 5.3:

Table 5.4: Transfer Learning Model Dataset B Classification Report

Species	Precision	Recall	F1-score	Support
Aloe arborescens	1.00	0.67	0.80	6
Aristea capitata	0.80	1.00	0.89	4
Baloskion tetraphyllum	1.00	0.50	0.67	4
Carpobrotus edulis	0.57	1.00	0.73	4
Cotyledon orbiculata	0.50	0.11	0.18	9
Curio talinoides	1.00	0.88	0.93	8
Erica arborescens	0.62	1.00	0.77	5
Erica cinerea	0.00	0.00	0.00	7
Erica discolor	0.56	0.83	0.67	6
Helichrysum petiolare	1.00	0.91	0.95	11
Leucadendron argenteum	1.00	0.30	0.46	10
Leucadendron laureolum	0.25	0.14	0.18	7
Leucadendron salignum	0.70	1.00	0.82	7
Melianthus major	0.91	1.00	0.95	10
Mutisia orbignyana	0.67	1.00	0.80	2
Oscularia deltoides	1.00	1.00	1.00	11
Osyris lanceolata	0.75	1.00	0.86	3
Ozothamnus leptophyllus	0.38	1.00	0.55	3
Pelargonium crispum	0.36	0.80	0.50	5
Protea cynaroides	0.33	0.36	0.34	14
Protea neriifolia	0.43	0.67	0.52	9
Protea repens	1.00	0.75	0.86	8
Strelitzia reginae	1.00	0.90	0.95	10
Accuracy			0.69	163
Macro Avg	0.69	0.73	0.67	163
Weighted Avg	0.71	0.69	0.66	163

The overall accuracy for Transfer Model B is 0.69 which is better than the 0.58 of Transfer Model A. This indicates that 69% of model predictions are correct which is also a large improvement over the Baseline Models. The macro-average precision is 0.69, the recall is 0.73 and the F1-score is 0.67 which are all higher scores than that achieved for Transfer Model A, validating the Image Quality Assessment.

High performing species include *Oscularia deltoides* which has a perfect score across precision, recall and F1-score which indicates that the model has been able to learn the features of this class and consistently identifies this species correctly.

Leucadendron laureolum and *Protea cynaroides* have F1-scores of 0.18 and 0.34 respectively which highlights the models difficulty in classifying them, and will need to be analysed in the SHAP analysis to understand why this arose.

The overall improved scores for Transfer Model B can be attributed to the Image Quality Assessment which has allowed the model to learn and generalise better than when using

the raw images in Dataset A.

While the Transfer Learning approach has yielded significantly better results for the uncleaned data than the Baseline Model, the difference between Transfer Model A and Transfer Model B is still significant. This demonstrates that despite the increase in model complexity, it is still unable to consistently deal with the uncleaned data and is likely over-fitting on background noise.

For the SHAP analysis it would be beneficial to look at species which have:

- Low F1-scores like *Erica cinerea* and *Protea cynaroides* in both models to understand the feature contribution that lead to the poor model performance.
- Species with high precision and low recall or low precision and high recall such as *Leucadendron laureolum* in Transfer Model B to identify any features that are causing imbalanced predictions.
- Species that have high F1-scores such as *Curio talinoides* and *Oscularia deltoides* in Transfer Model B in order to understand the features that are contributing positively to the model's predictions in order to apply these insights to the model architecture.

5.2.3 Confusion Matrix

The confusion matrices for both models are presented below. There is a much more defined diagonal element in both models than there was in the Baseline Models, although Baseline A still shows significant mis-classification.

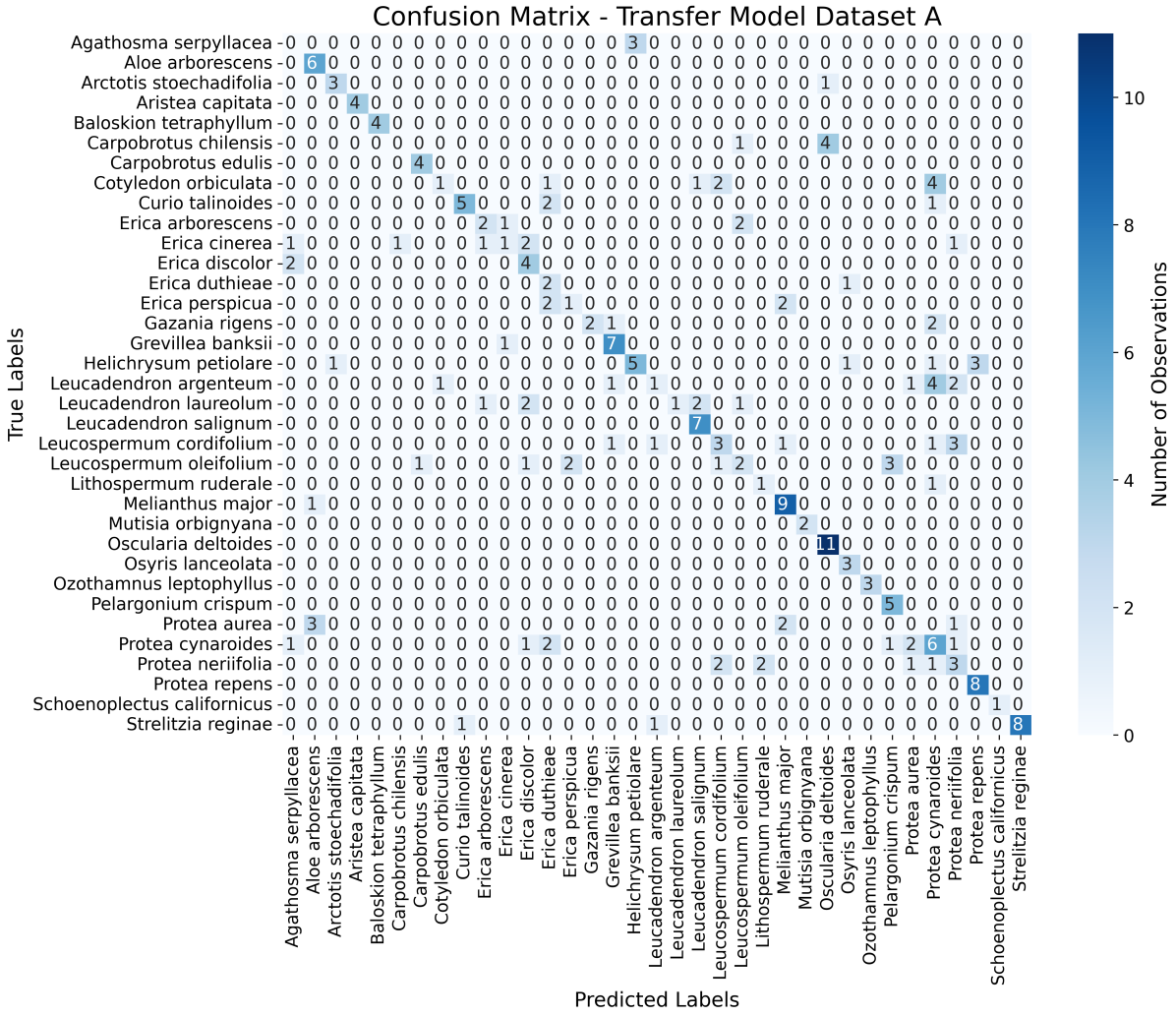


Figure 5.14: Confusion Matrix for Transfer Learning Model on Dataset A

The confusion matrix for Transfer Model A displays a much better diagonal shape than the Baseline Models. the transfer learning approach in design is clearly superior to building a model from scratch. Certain species like *Grevillea banksii* and *Erica discolor* show a higher number of correct classifications (8 for both) which indicates some success in recognising these species. There is however a significant mis-classification spread across the matrix. For example, *Protea cynaroides* has been confused with other species multiple times, indicating potential difficulties that the model faces in differentiating one species from another when the shapes of the plants are similar. The diagonal elements (showing the True Positives) are plotted below:

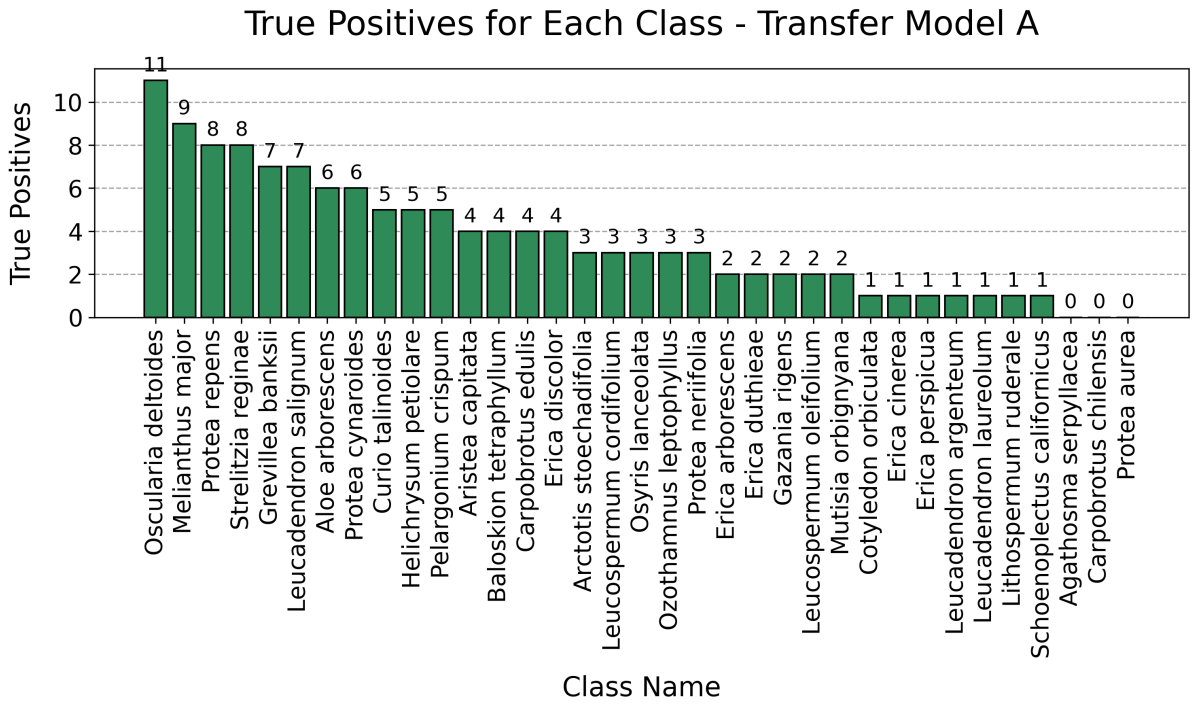


Figure 5.15: True Positives for Transfer Model A per class

The distribution of True Positives is significantly improved over the Baseline A results shown in 5.5. There are still three classes with zero True Positives, but overall the model was able to predict at least one sample correctly in the rest of the classes. This shows a much better model generalisation than the Baseline models, which is expected as the deeper network is learning able to learn more complex features.

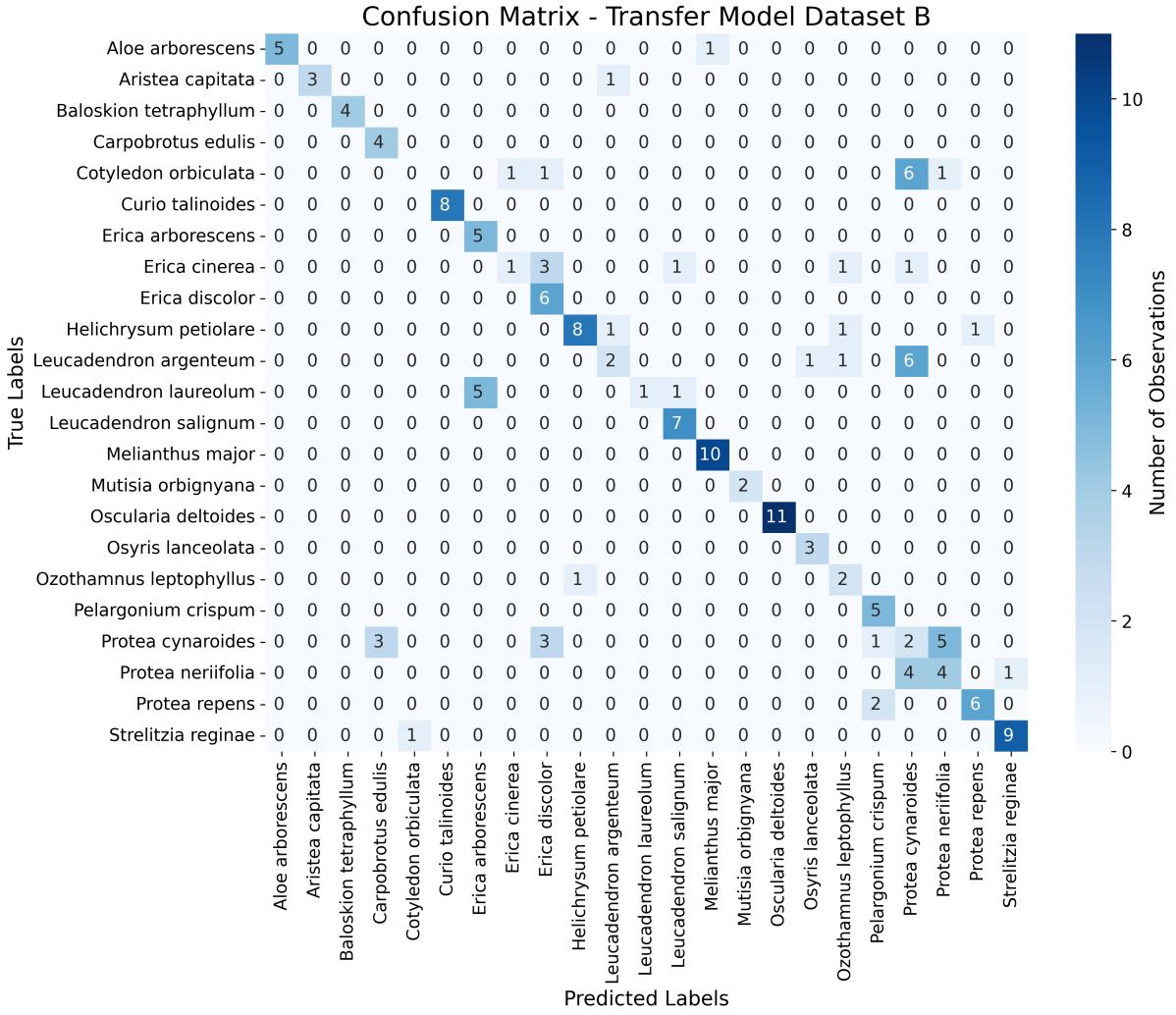


Figure 5.16: Confusion Matrix for Transfer Learning Model on Dataset B

The confusion matrix for Transfer Model B displays a certain level of diagonal concentration at a level much better than the previous models, which indicates that the model is mostly able to obtain correct classifications. But there is still a noticeable amount of off-diagonals which represent mis-classifications.

Certain species like *Leucadendron salignum* and *Oscularia deltoides* show a relatively high number of correct predictions at 7 and 11 respectively which suggests that the model has been able to learn and distinguish features from these species. *Oscularia deltoides* in particular has no mis-classifications in this model showing that its features are distinctive enough.

There is noticeable mis-classification in species like *Erica discolor* and *Protea repens* which are confused with several other species. This indicates that these species are phenotypically similar and share characteristics, or at least that the model learns shared

features between them. The species with the most off-diagonals is *Protea neriifolia* which has 4 True Positives, but 18 True Negatives distributed across four other species. *Protea neriifolia* likely shares features with other classes this mis-classification rate is consistent with Transfer Model A. The diagonals are again plotted below:

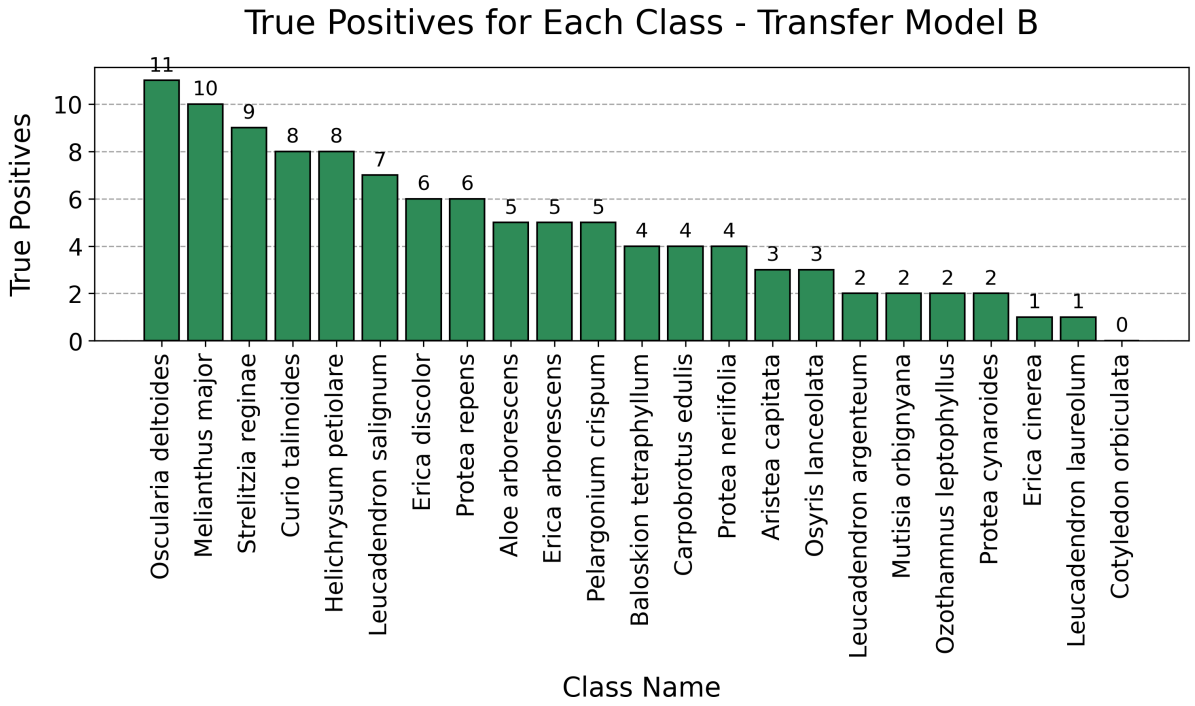


Figure 5.17: True Positives for Transfer Model A per class

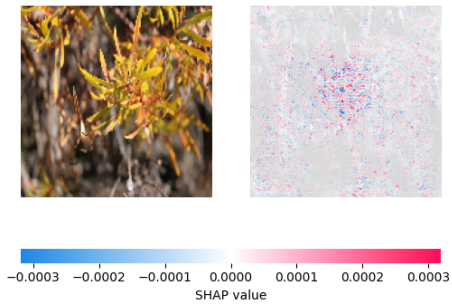
The number of True Positives skews much higher for this model than for the earlier models. There is only 1 class with zero True Positives and 2 classes with only 1 True Positive. The best performing species is now *Oscularia deltoides*.

5.2.4 SHAP Analysis

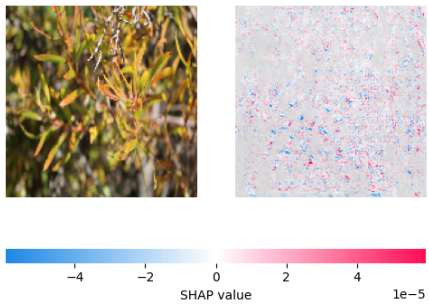
Due to a significantly deeper model, the SHAP analysis will be able to dictate model performance at much more granular level, and the more complex features it is able to identify will be be visualised. Again for the sake of readability, the SHAP plots for Transfer Model A are moved to the appendix.

Transfer Model B

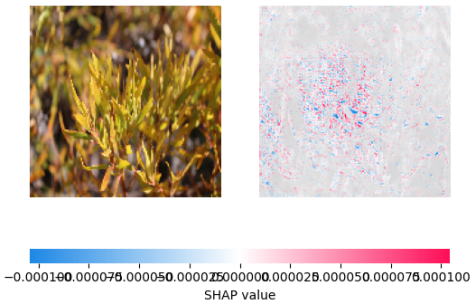
High Performing Classes *Leucadendron salignum* has shown consistent high performance in the previous three models, and that is continued for Transfer Model B. It has an F1 score of 0.82, but it is not the best performing class for this model. It shows 7 True Positives and 2 False Positives in the confusion matrix out of 9 observations. Transfer Model B can mostly classify this class, although there is still some confusion. The SHAP plots below also show diverse features in the pixels. The model is not learning particular patterns, rather it is learning multiple features contained within these images. This shows strong generalisation for this class.



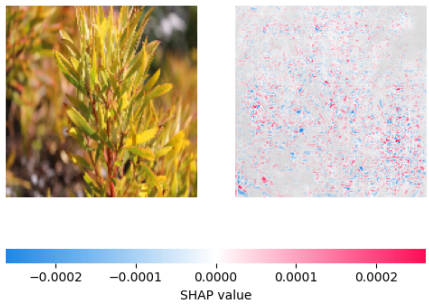
(a) *Leucadendron salignum* SHAP i



(b) *Leucadendron salignum* SHAP ii



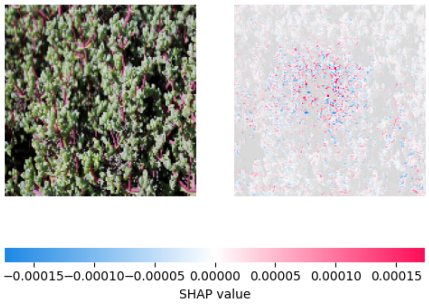
(c) *Leucadendron salignum* SHAP iii



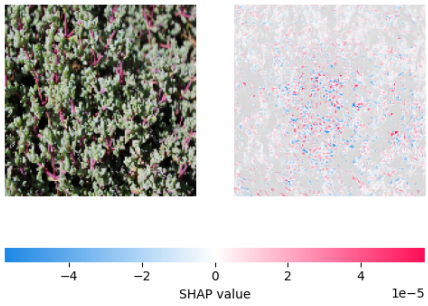
(d) *Leucadendron salignum* SHAP iv

Figure 5.18: *Leucadendron salignum* leaves subset SHAP Values from Transfer Model B

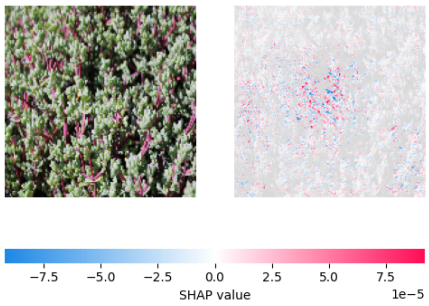
Oscularia deltoides performs exceptionally well in Transfer Model B with a perfect F1-score of 1. it had 11 True Positives out of 11 samples. This class was also higher performer in the Baseline B model. The models are adept at learning the distinct leaf clustering seen in all four of the SHAP plots. This particular species would be difficult to isolate leaves for and natural images such as SHAP i-iv below would be the most likely images to be trained on.



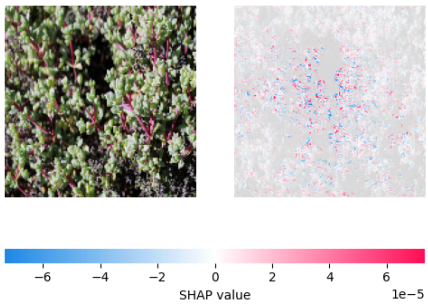
(a) *Oscularia deltoides* SHAP i



(b) *Oscularia deltoides* SHAP ii



(c) *Oscularia deltoides* SHAP iii



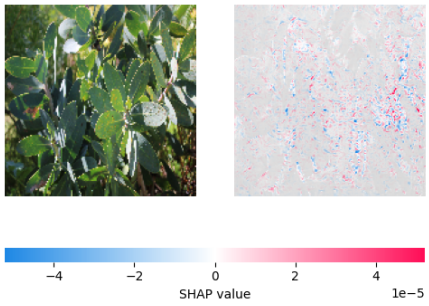
(d) *Oscularia deltoides* SHAP iv

Figure 5.19: *Oscularia deltoides* leaves subset SHAP Values from Transfer Model B

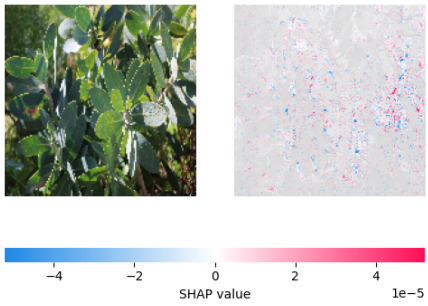
Leaf shape and also stems appear to contribute to the positive prediction. Darker patches of the images contribute to the negative prediction as seen by the blue patches. This again speaks to the models difficulty in dealing with variable lighting conditions.

Mis-Classified Classes

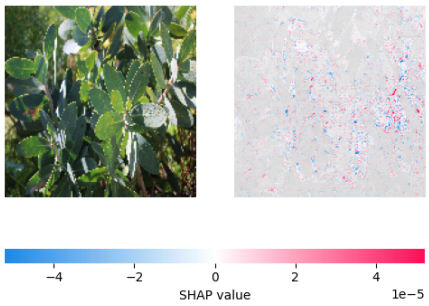
In Transfer Model B *Protea neriifolia* has an F1-score of 0.52 with a precision of 0.43 and a recall of 0.67. This model is able to learn some features, but is struggling with the classification. The confusion matrix shows that it has 4 Positives, but it is incorrectly classified as *Protea cynaroides* 5 times and once as *Cotyledon orbiculata*.



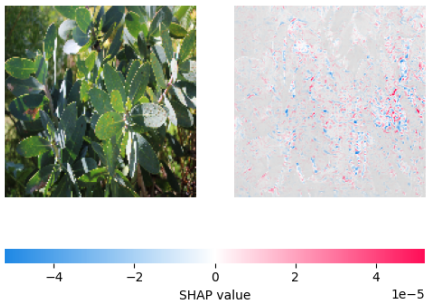
(a) *Protea neriifolia* SHAP i



(b) *Protea neriifolia* SHAP ii



(c) *Protea neriifolia* SHAP iii

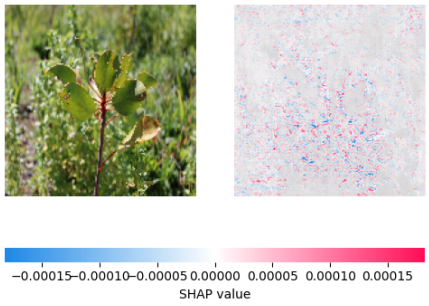


(d) *Protea neriifolia* SHAP iv

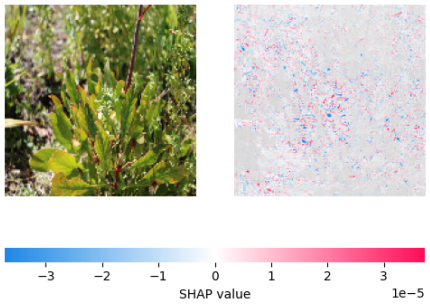
Figure 5.20: *Protea neriifolia* leaves subset SHAP Values from Transfer Model B

The SHAP plots in all four images indicate a similar pixel pattern. This is clear evidence that the model has learned irrelevant features from another class and is applying them to this one. A diversity on the shape of the pixels in the SHAP plot would show good generalisation, but the opposite is observed here. The model is attempting to force features onto this class, likely because the images between the classes contain similar features (actual plant features or similar background noise features).

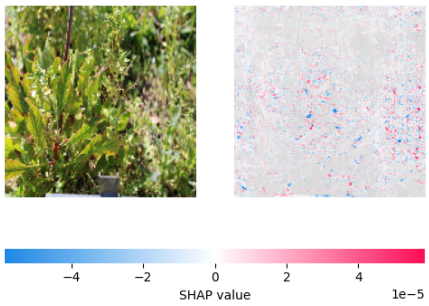
Similarly *Protea cynaroides* has an F1-score of 0.34 with a precision of 0.33 and a recall of 0.36. While the model is learning some basic features, it is likely learning a lot of noise which is causing these mis-classifications. The confusion matrix shows that it has 5 True Positives, but it is incorrectly classified as *Protea cynaroides* 4 times and 6 times as *Cotyledon orbiculata*, a further 6 times as *Leucadendron argenteum* and once as *Erica cinerea*. The features learned from the class are likely generalised features or noise that is present in all these different classes.



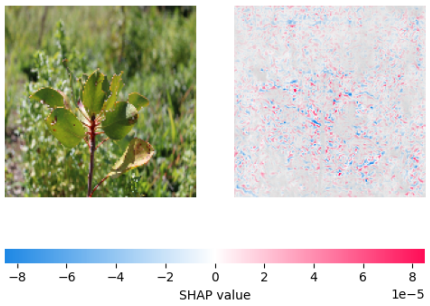
(a) *Protea cynaroides* SHAP i



(b) *Protea cynaroides* SHAP ii



(c) *Protea cynaroides* SHAP iii



(d) *Protea cynaroides* SHAP iv

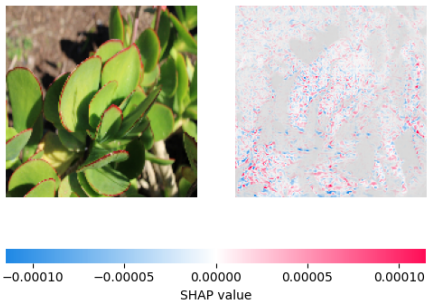
Figure 5.21: *Protea cynaroides* leaves subset SHAP Values from Transfer Model B

The SHAP plots show more diverse shapes for this class, but the features that contribute to the prediction are not focused on the actual plant and are instead clustered around background features or other leaves contained within the image. This class is likely confusing the model by introducing these generalised features. The Image Quality Assessment does not full deal with the image quality problems that have arisen in this class.

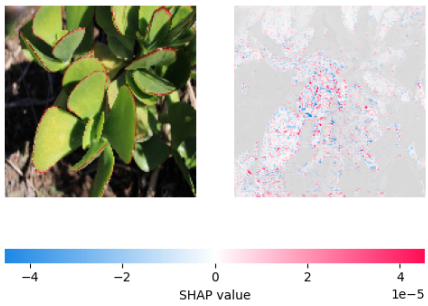
Low Performing Classes

A particularly poorly performing class that is somewhat unexpected is *Cotyledon orbiculata* which has distinctive leaf shapes. It has an F1-score of 0.18. The precision is 0.50 however, while the recall is 0.11. There are no True Positives in the confusion matrix and 1 False Positive (mis-classified as *Strelitzia reginae* once. The SHAP plots below in 5.22 show that leaf features are being learned, but these features seem to focus on the waxy surfaces

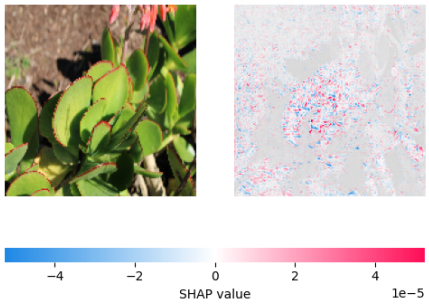
discussed in 2.1.2 and not the actual leaf shape. This can be seen in SHAP ii. SHAP i and SHAP iii are very similar images to a human observer, but the model learns very different features. There is also noticeable contribution from the edges of the image. This shows that the Width transformation in the data augmentation of this model is having an impact on prediction, although for this class it is creating confusion. The edge pixels in SHAP ii are pink showing a positive impact on prediction, but there does not appear to be any features captured there.



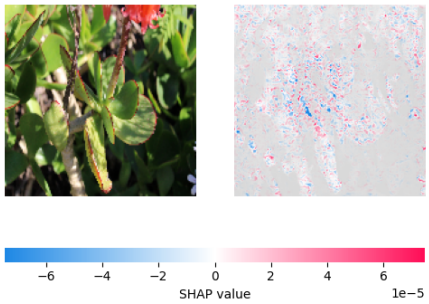
(a) *Cotyledon orbiculata* SHAP i



(b) *Cotyledon orbiculata* SHAP ii



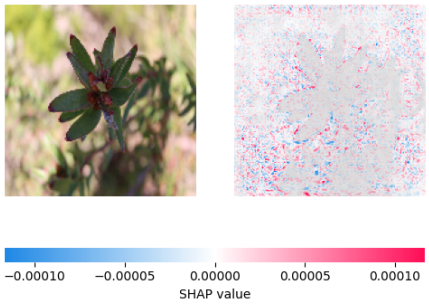
(c) *Cotyledon orbiculata* SHAP iii



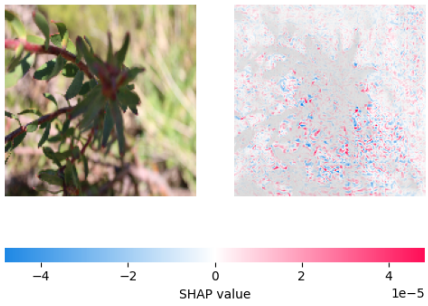
(d) *Cotyledon orbiculata* SHAP iv

Figure 5.22: *Cotyledon orbiculata* leaves subset SHAP Values from Transfer Model B

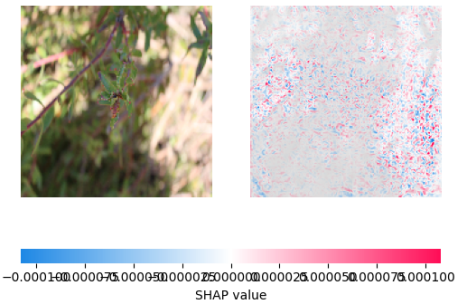
Erica cinerea continues to score poorly. It has an F1-score of zero (and is the only species in this model to do so). It is mis-classified a few times in the confusion matrix and the problems presented in the earlier models are exacerbated here. The shadows cast on the images are still not being considered by the model and the features learned in the background noise are considerably worse. This class alone is introducing unnecessary features into the learning process and the model still cannot handle inconsistent lighting.



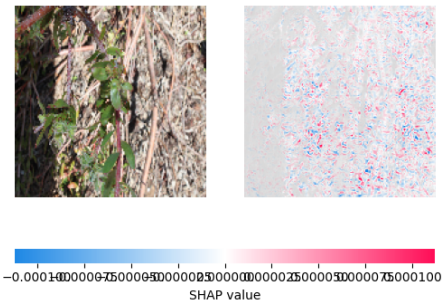
(a) *Cotyledon orbiculata* SHAP i



(b) *Cotyledon orbiculata* SHAP ii



(c) *Cotyledon orbiculata* SHAP iii



(d) *Cotyledon orbiculata* SHAP iv

Figure 5.23: *Erica cinerea* leaves subset SHAP Values from Transfer Model B

The *Erica cinerea* class has been a poor performing class across all models so far and it has some important considerations for the next development.

5.2.5 Discussion

Applying Transfer Learning to the model had a considerable positive impact on model performance. The models converged much faster and ten training epochs was more than suitable to see the models reach peak performance. Validation accuracy was also higher than the Baseline Models and demonstrably more stable. It converged to slightly above 60% for Transfer Model B which is a large and impressive leap over the Baseline Model performance. There is however still considerable over-fitting, more so in Transfer Model A. This indicates that the pre-trained weights (and thus pre-learned features) were instrumental in the improved model performance.

One of the flaws of training CNNs on small datasets is attempting a train the network on limited features and expecting it to be able to generalise. This process works up to a limit as seen in in the Baseline Model results, but the SHAP analysis for the Baseline Models showed significant features learned on background noise in images. The transfer learning approach overcomes this problem but introducing pre-learned features which the the model then adapts to with the unfrozen layer parameter.

The transfer learning model used a dropout of 0.3 and no other forms for regularisation which is likely why there are such high levels of over-fitting. It was originally assumed that the more complex model would deal with noise better but the poor data quality in Dataset A has limited model performance.

The model shows that is capable of learning more complex features which can be seen in 5.19. The leaf clusters and stems contribute highly to the prediction, but the model is still not learning advanced patterns. The poor performance shown in 5.22 in particular shows evidence of this as the species does have the large leaves with waxy surfaces described in2.1.2. The model is limited and needs further enhancements to differentiate between the leaves and plant features and background noise.

The problematic class of *Protea cynaroides* still contributes greatly to the mis-classification rate of the model. The problem of poor image quality persists from the Baseline Model shown in 5.21. The model cannot deal with variable lighting conditions and shadows and a solution to this will need to be designed. This also shows a weakness in the Image Quality Assessment which at the moment does not consider image lighting as quality metric.

5.3 Model 3: Optimised Model Results

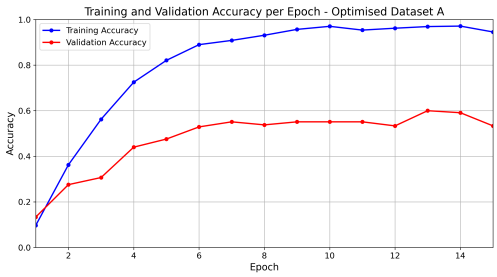
The Optimised model produces the best results so far, particular Optimised Model B which is trained on the clean dataset. The improvements to the model have yielded positive results. The training versus accuracy graphs are presented below in Figure 5.24.

This model employed the SE module to enhance important features. It also added batch normalisation to reduce the over-fitting from the previous two models and increased the dropout factor to 0.5. Due to the increased complexity in this model, the number of training epochs was increased from 10 to 15 to allow more time to converge.

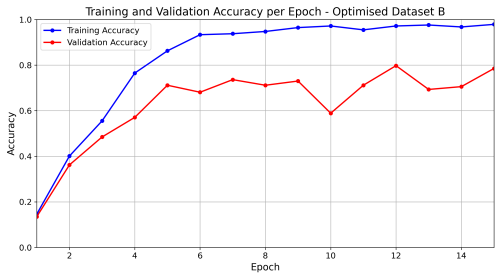
5.3.1 Performance Results

Accuracy The accuracy achieved in this model is a noted improvement over the previous validating the Squeeze-and-Excite addition. The training and validation accuracy metrics are shown in Figure 5.24.

Figure 5.24: Training and validation accuracy of the Optimised Model over a series of 15 epochs



(a) Optimised Model A



(b) Optimised Model B

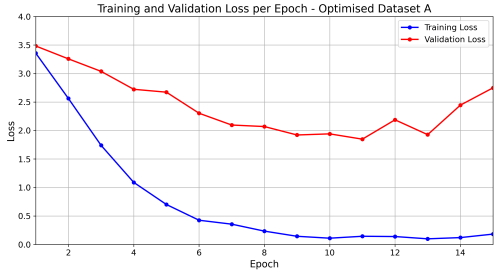
There is a clear distinction in learning dynamics between the two models. For Optimised Model A the training accuracy displays a consistent ascent which nears 99% by the final epoch. The validation accuracy also improved, but at a diminishing pace, which again shows the over-fitting pitfalls experienced by the earlier models. The validation accuracy plateaus at around 60% which is again and indicator of over-fitting and the model is memorising features in the training data and not generalising from. The gap between the training and validation accuracy is sign of poor model generalisation.

Optimised Model B on the other hand, displays a similar pattern for training accuracy where it approaches 100% towards the final epochs, but the validation accuracy is significantly improved and appears to plateua around the 75% range. The difference between the training and validation sets still shows over-fitting but it is much less pronounced. This shows that the model is able to learn from the cleaned dataset much better than it can from raw, unprocessed images.

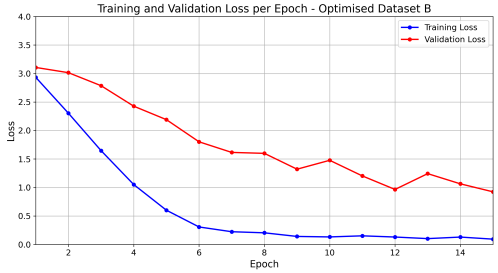
Loss

The loss graphs for these models also show an improvement over the previous models. The validation loss fr both Optimised Model A and B shows a consistent downward trend, although Optimised Model A's loss does plateau.

Figure 5.25: Training and validation loss of Model 3 over a series of 15 epochs



(a) Optimised Model A



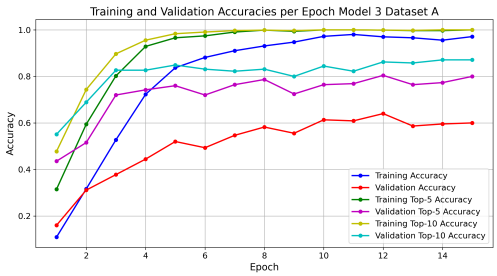
(b) Optimised Model B

For Optimised Model A, the training loss shows a downward trend with a somewhat steep initial reduction in magnitude that ends around the 5th epoch. This implies that the model has a more difficult learning process, likely due to the images being in their raw format. The validation loss for this model displays some fluctuations and it eventually plateaus, implying some challenges for the model to generalise and suggesting that there are limitations to how much it can learn from this dataset.

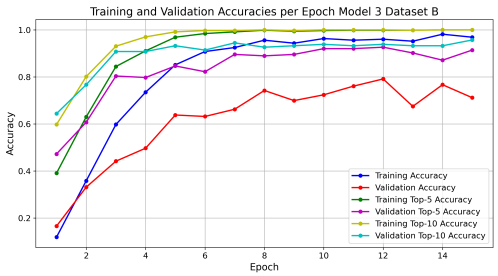
Conversely for Optimised Model B, the training loss shows a more gradual descent and it stablises at around 0.2 at around the the eight epoch. The validation loss mirrors this decline and shows that it continues beyond the 10th epoch unlike in Optimised Model A. This suggests that the model continues to learn features in the cleaned dataset long after it has been unable to extract anything from dataset A.

There is still over-fitting in both models but it is much less pronounced in Optimised Model B which shows a greater capacity to generalise.

Top 5 and Top 10 Accuracy



(a) Optimised Model A



(b) Optimised Model B

Figure 5.26: Top 5 Accuracy and Top 10 Accuracy for Model 3 for training and validation over a series of 15

Both top-5 and top-10 accuracy metrics were again evaluated for these models. The top-10 training accuracy for both models reaches 100% by the 8th epoch. For Optimised Model A, a gap exists between training and validation accuracy, again hinting at over-fitting. Despite this, the model maintained a stable validation accuracy which indicates reasonable generalisation.

The top-5 and top-10 accuracy rates for Optimised Model B were significantly higher, for both training and validation, which indicates that the model may occasionally falter in making the exact correct classification, the correct classification often resides within its top few predictions.

5.3.2 Classification Report

The classification reports for both optimised models are presented in this section. For the first time, there is only two classes with a F1-scores of zero, meaning at least one support was correctly classified in every other class. This demonstrates the improved feature learning ability of the model with the addition of the Squeeze-and-Excite block. The classification report for Optimised Model A is presented below:

Table 5.5: Classification Report for Optimised Model A

Species	**Precision**	**Recall**	**F1-score**	**Support**
Agathosma serpyllacea	0.29	0.67	0.40	3
Aloe arborescens	0.55	1.00	0.71	6
Arctotis stoechadifolia	0.67	0.50	0.57	4
Aristea capitata	0.36	1.00	0.53	4
Baloskion tetraphyllum	0.50	0.25	0.33	4
Carpobrotus chilensis	0.00	0.00	0.00	5
Carpobrotus edulis	0.80	1.00	0.89	4
Cotyledon orbiculata	1.00	0.22	0.36	9
Curio talinoides	0.80	1.00	0.89	8
Erica arborescens	0.75	0.60	0.67	5
Erica cinerea	0.67	0.29	0.40	7
Erica discolor	0.38	0.50	0.43	6
Erica duthieae	0.22	0.67	0.33	3
Erica perspicua	0.50	0.20	0.29	5
Gazania rigens	0.40	0.40	0.40	5
Grevillea banksii	0.70	0.88	0.78	8
Helichrysum petiolare	0.83	0.45	0.59	11
Leucadendron argenteum	0.33	0.10	0.15	10
Leucadendron laureolum	1.00	0.14	0.25	7
Leucadendron salignum	1.00	0.86	0.92	7
Leucospermum cordifolium	0.38	0.50	0.43	10
Leucospermum oleifolium	0.67	0.20	0.31	10
Lithospermum ruderale	1.00	1.00	1.00	2
Melianthus major	0.67	1.00	0.80	10
Mutisia orbignyana	0.50	1.00	0.67	2
Oscularia deltoides	0.71	0.91	0.80	11
Osyris lanceolata	0.50	1.00	0.67	3
Ozothamnus leptophyllus	0.75	1.00	0.86	3
Pelargonium crispum	0.71	1.00	0.83	5
Protea aurea	0.00	0.00	0.00	6
Protea cynaroides	0.43	0.64	0.51	14
Protea neriifolia	0.83	0.56	0.67	9
Protea repens	0.89	1.00	0.94	8
Schoenoplectus californicus	1.00	1.00	1.00	1
Strelitzia reginae	1.00	0.90	0.95	10
Accuracy			0.60	225
Macro Avg	0.62	0.64	0.58	225
Weighted Avg	0.64	0.60	0.57	225

Optimised Model A shows a wide range of performance across the different species. The precision, recall and F1-scores vary significantly. Species like *Lithospermum ruderale* and *Schoenoplectus californicus* actually get perfectly classified now while others like *Carpobrotus chilensis* and *Protea aurea* do not get classified at all.

The overall accuracy of the model stands at 60% with a macro average F1-score of 0.58 and weighted average F1-score of 0.57. This indicates moderate performance and marked improvement on other models trained in Dataset A. The classification report for Optimised Model B is presented below:

Table 5.6: Classification Report for Optimised Model B

Species	**Precision**	**Recall**	**F1-Score**	**Support**
Aloe arborescens	1.00	1.00	1.00	6
Aristea capitata	1.00	1.00	1.00	4
Baloskion tetraphyllum	0.80	1.00	0.89	4
Carpobrotus edulis	1.00	1.00	1.00	4
Cotyledon orbiculata	1.00	0.22	0.36	9
Curio talinoides	1.00	1.00	1.00	8
Erica arborescens	0.75	0.60	0.67	5
Erica cinerea	1.00	0.29	0.44	7
Erica discolor	1.00	0.83	0.91	6
Helichrysum petiolare	1.00	0.73	0.84	11
Leucadendron argenteum	0.00	0.00	0.00	10
Leucadendron laureolum	0.50	0.14	0.22	7
Leucadendron salignum	0.88	1.00	0.93	7
Melianthus major	1.00	1.00	1.00	10
Mutisia orbignyana	0.22	1.00	0.36	2
Oscularia deltoides	1.00	0.91	0.95	11
Osyris lanceolata	1.00	0.67	0.80	3
Ozothamnus leptophyllus	1.00	1.00	1.00	3
Pelargonium crispum	0.83	1.00	0.91	5
Protea cynaroides	0.38	0.93	0.54	14
Protea neriifolia	0.67	0.89	0.76	9
Protea repens	1.00	0.88	0.93	8
Strelitzia reginae	0.91	1.00	0.95	10
Accuracy			0.76	163
Macro Avg	0.82	0.79	0.76	163
Weighted Avg	0.81	0.76	0.74	163

For Optimised Model B, the performance metrics are the best seen so far across all six models, with an overall accuracy of 76% which is excellent. There is a higher macro and weighted average F1-score of 0.76 and 0.74 respectively. There is notable perfect classification on several species including *Lithospermum ruderal*, *Schoenoplectus*

californicus which achieved F1-scores of 1. Conversely, *Carpobrotus chilensis* and *Protea aurea* achieved F1-scores of 0. The disparity in performance can be explained by looking at the SHAP analysis.

5.3.3 Confusion Matrix

The confusion matrices for the optimised models show the greatest diagonal element so far with much fewer mis-classifications. They are still present but in much smaller quantities.

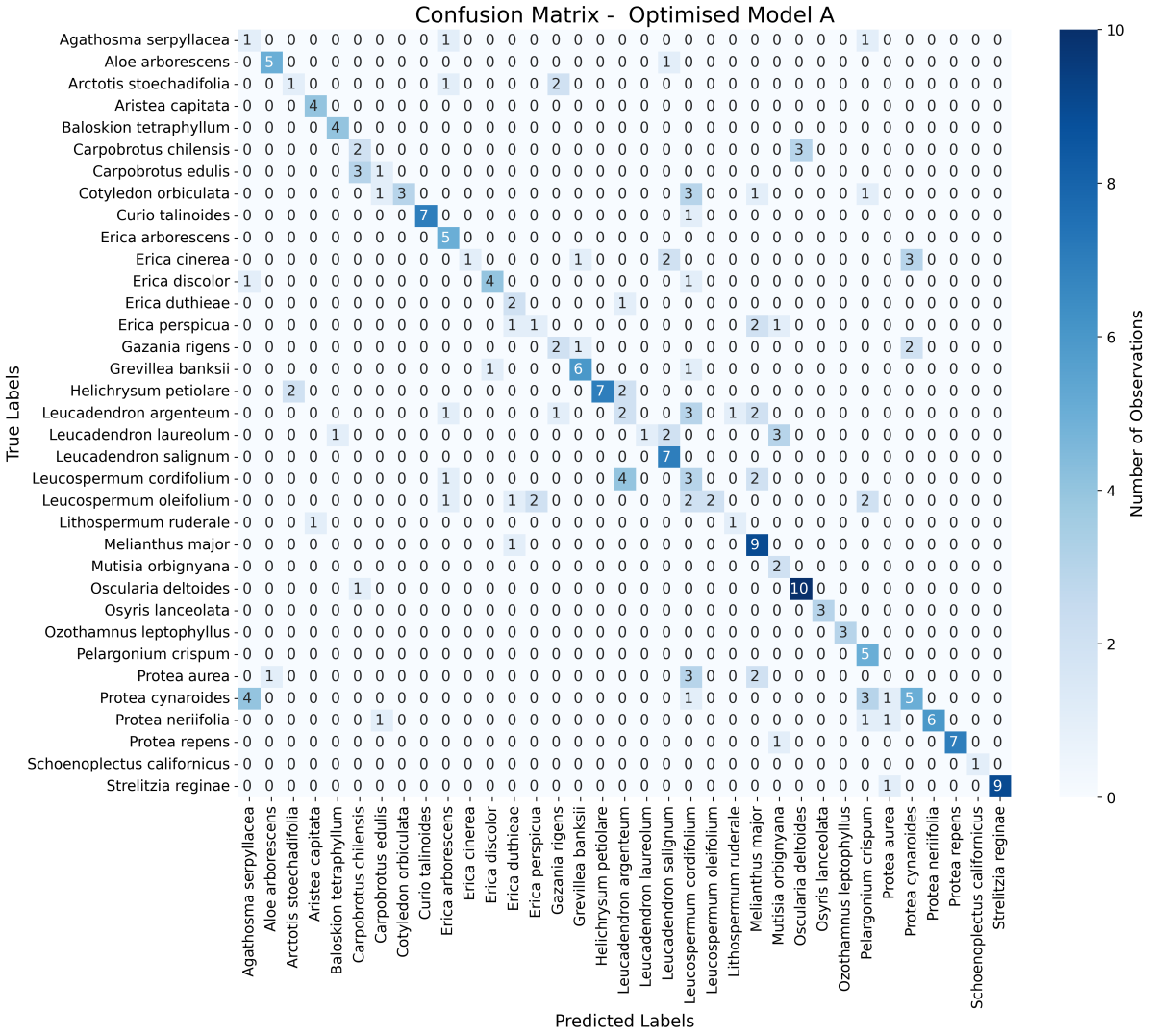


Figure 5.27: Confusion Matrix for Optimised Model A

In Optimised model A, species such as *Aloe arborescens* and *Cotyledon orbiculata* have a strong diagonal presence which indicates a high number of true positives and successful

prediction. There are still several off0diagonal elements that are non-zero which are indicative of mis-classifications. *Erica ciliaris* for example is often confused with *Leucadendron salignum*. The species with the highest confusion)off-diagonal elements) should be prioritised in the SHAP analysis.

It should also be noted that *Protea cynaroides* shows significantly improved performance in this model with fewer mis-classifications and 9 True Positives. This also means that the model is not learning the generalised noise in this images like it was previously. A bar graph showing the True Positives per class is shown below:

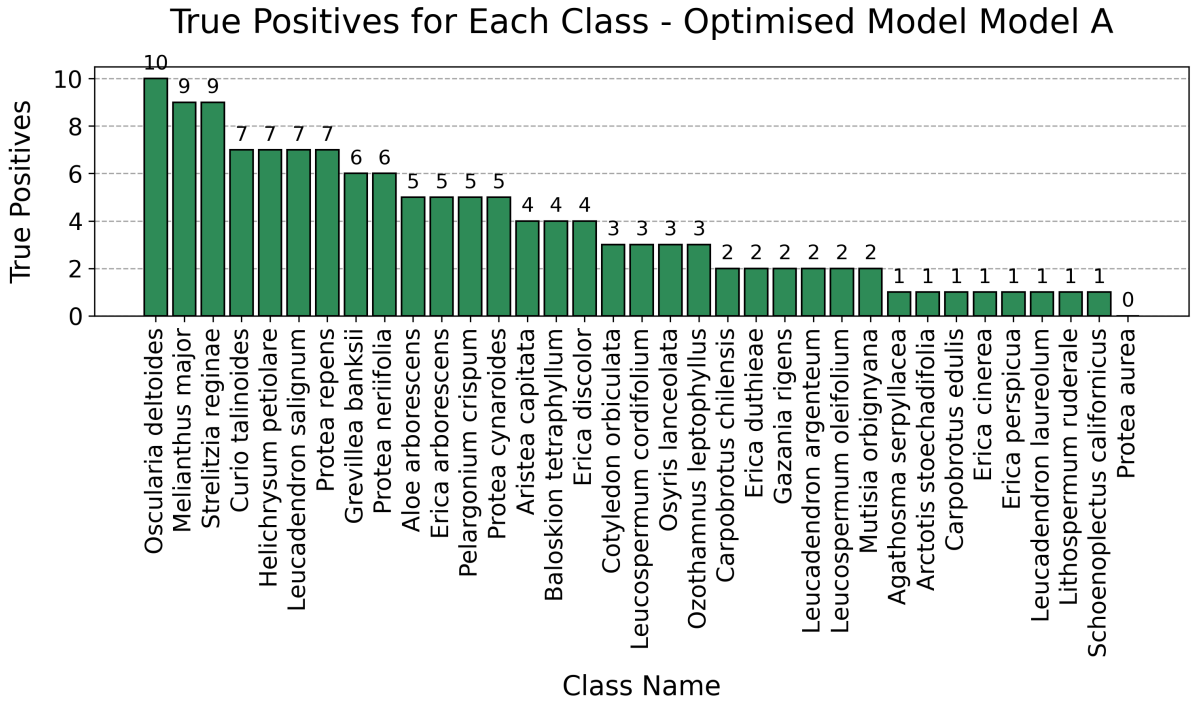


Figure 5.28: True Positives for Optimised Model A per class

It can be seen in 5.28 that there more classes with high True Positives than in previous models. There are still several classes with 1 or less True Positives.

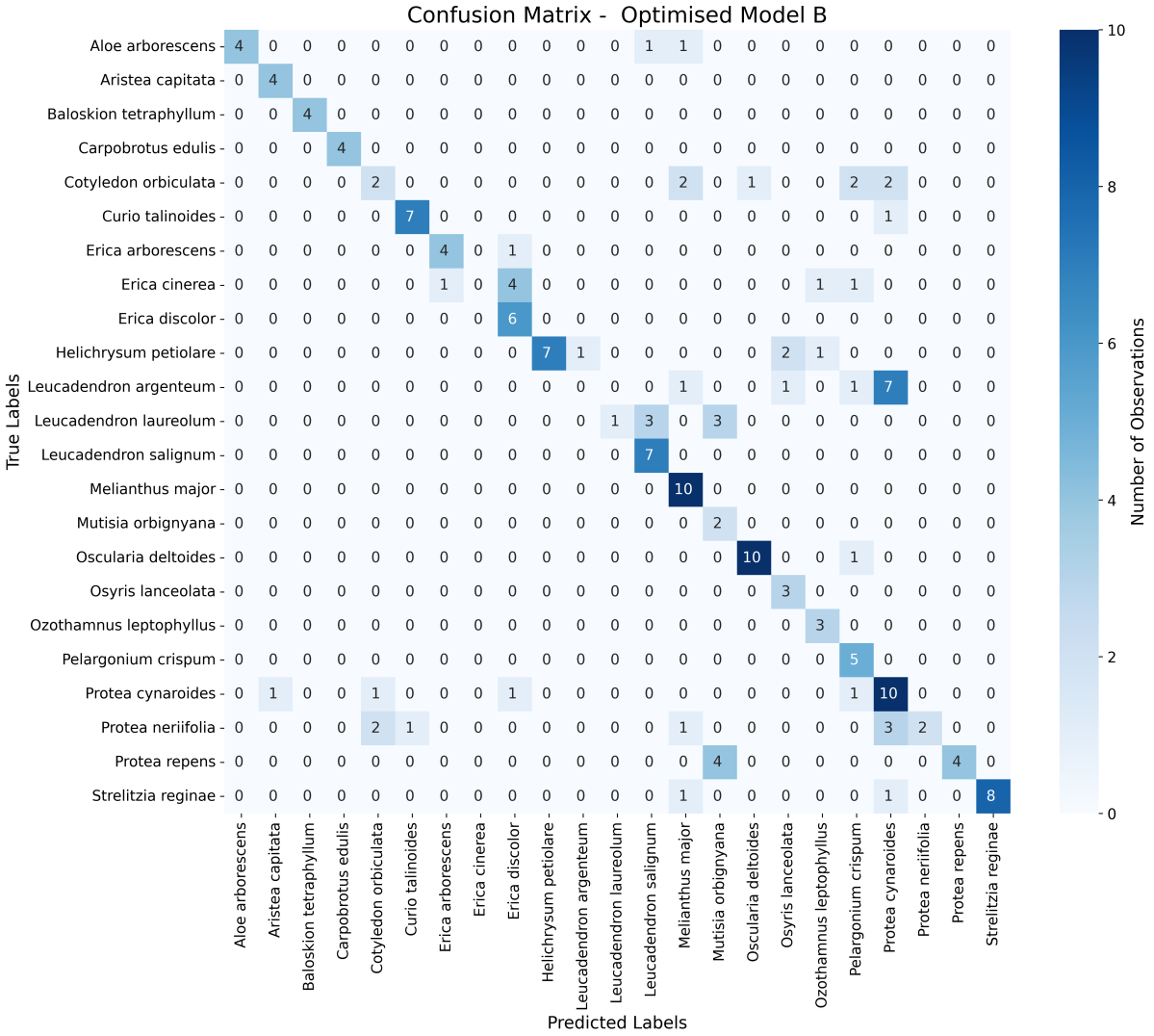


Figure 5.29: Confusion Matrix for Optimised Model B

The confusion matrix for Optimised Model B shows a similar pattern with some species like *Leucadendron salignum* and *Protea cynaroides* being classified correctly most of the time. There are still cases of mis-classification with species such as *Leucadendron laureolum* being mis-classified with *Leucadendron salignum* suggesting that the model may be leveraging similar features to make its predictions for these species.

The diagonal element in the matrix is plotted into a bar graph below in 5.30

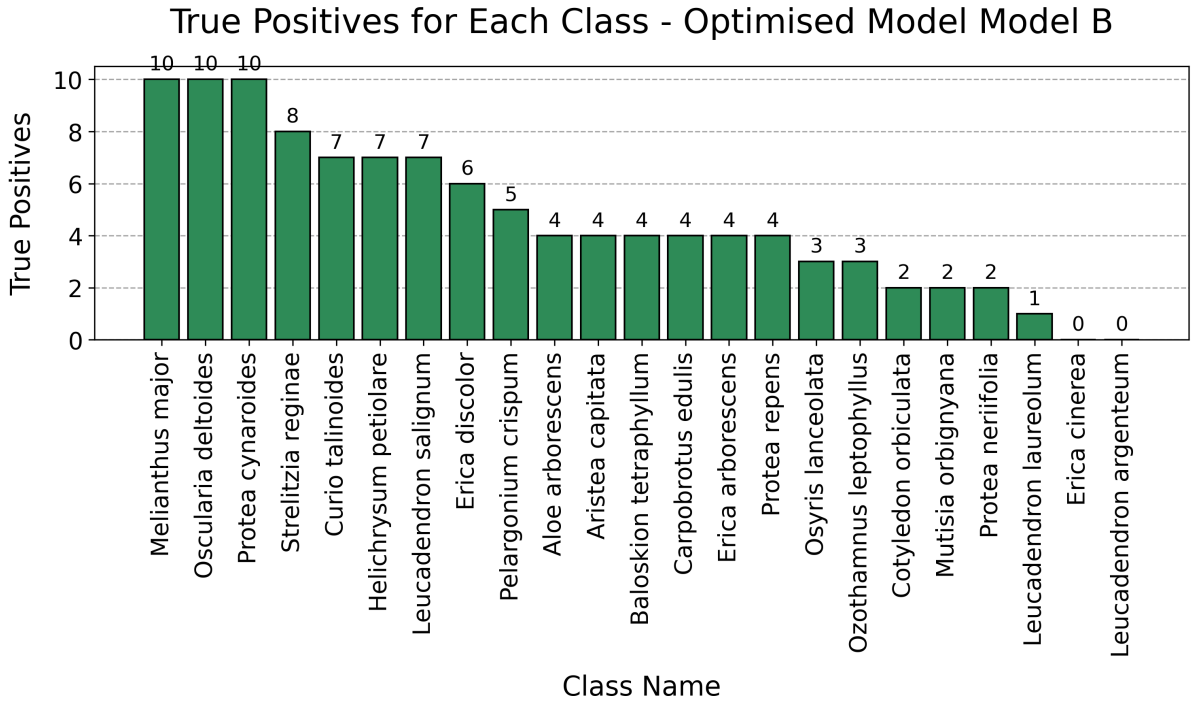


Figure 5.30: True Positives for Optimised Model B per class

There are two classes that have zero true positives. Interestingly these classes did perform relatively well in previous models.

5.3.4 SHAP Analysis

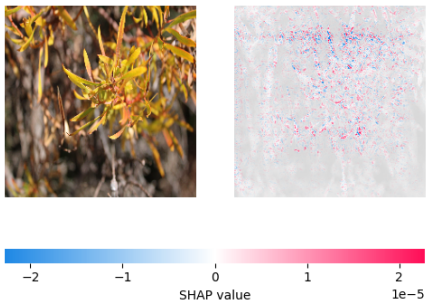
For the Optimised Model, the SHAP plots for Optimised Model A are presented in this section because they show unusual behavior from the model that needs to be explained.

Optimised Model A

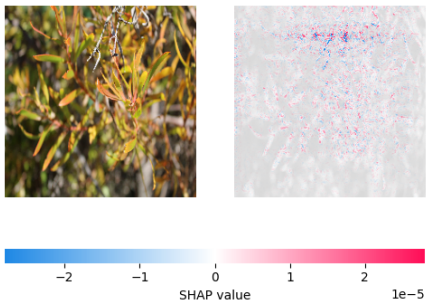
High Performing Classes

Leucadendron salignum has an F1-score of 0.92 with a precision of 1 and recall of 0.86 and has 6 True Positives in the confusion matrix and no false positives. The SHAP analysis for this class reveals a similar shaped feature in each plot i-iv. The pixels that contribute to the positive prediction appear to be resolving into a particular shape. This SHAP plot helps visualise the concept of over-fitting, where the model is fitting the images to

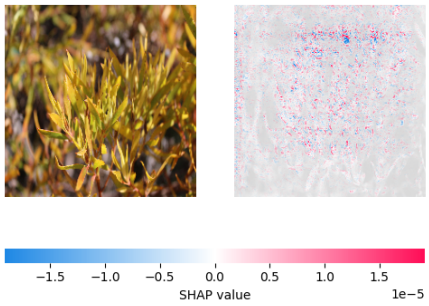
a generalised feature it has learned. This feature may not actually be representative of *Leucadendron salignum* taken from other angles or in different lighting conditions. This suggests that despite the improved model performance for Optimised Model A, it is still not suitable for use as a generalised model.



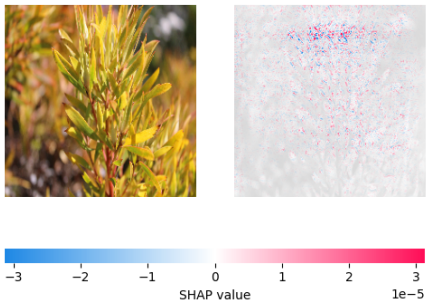
(a) *Leucadendron salignum* SHAP i



(b) *Leucadendron salignum* SHAP ii



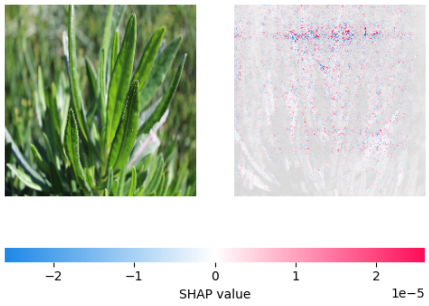
(c) *Leucadendron salignum* SHAP iii



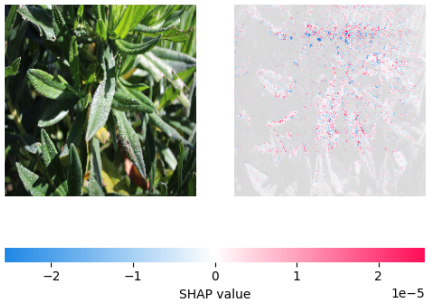
(d) *Leucadendron salignum* SHAP iv

Figure 5.31: *Leucadendron salignum* leaves subset SHAP Values from Optimised Model A

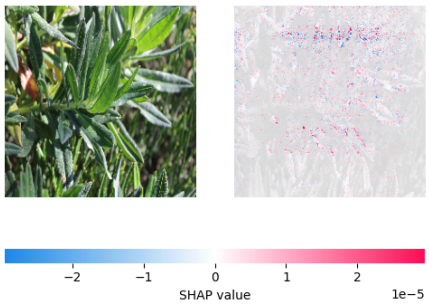
Lithospermum ruderale has 2 true positives and an F1-score of 1. However the support is only 2 which implies that the model could be randomly guessing the predictions correctly. The SHAP plots again show the generalised feature seen in 5.31, showing the over-fitting present in this model. The feature contributing to the prediction appears to be simialr across different species.



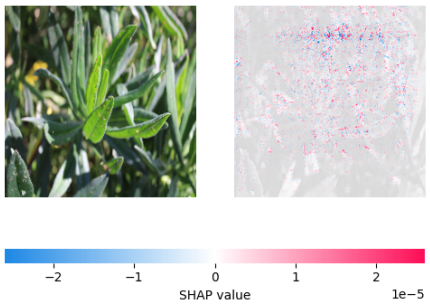
(a) *Lithospermum ruderale* SHAP i



(b) *Lithospermum ruderale* SHAP ii



(c) *Lithospermum ruderale* SHAP iii

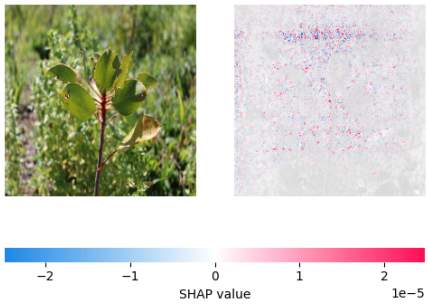


(d) *Lithospermum ruderale* SHAP iv

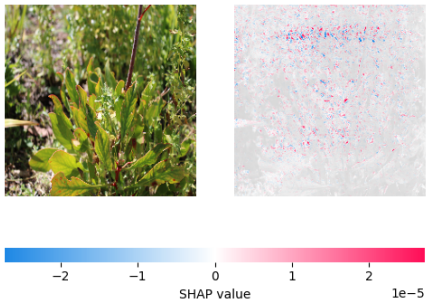
Figure 5.32: *Lithospermum ruderale* leaves subset SHAP Values from Optimised Model A

Mis-Classified Classes

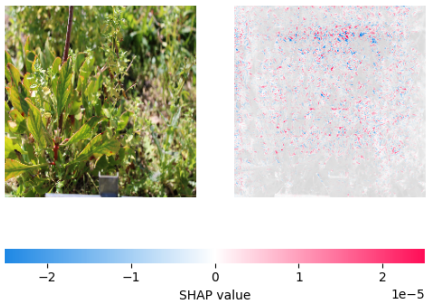
Protea cynaroides has an F1-score of 0.51 and a precision of 0.43 and a recall of 0.64. It had 9 true positives in the confusion matrix but 11 mis-classifications as various other classes. The same feature seen other classes for this model is manifesting in all the SHAP plots.



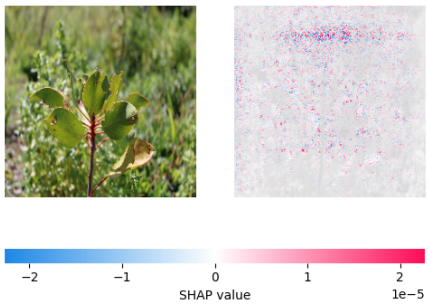
(a) *Protea cynaroides* SHAP i



(b) *Protea cynaroides* SHAP ii



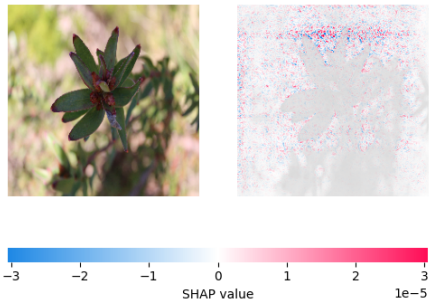
(c) *Protea cynaroides* SHAP iii



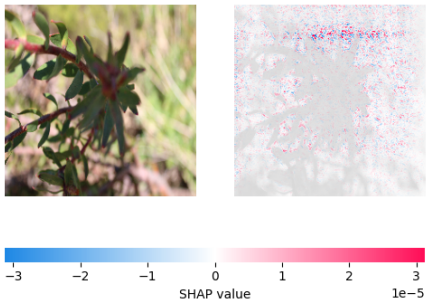
(d) *Protea cynaroides* SHAP iv

Figure 5.33: *Protea cynaroides* leaves subset SHAP Values from Transfer Model B

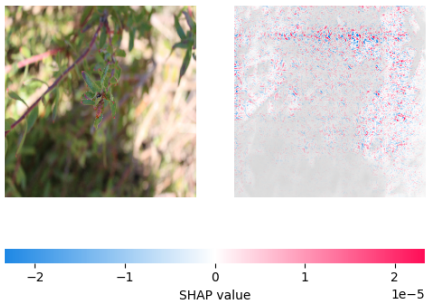
Erica cinerea had two true positives and several mis-classifications, but it had and F1-score of 0.4 and a precision of 0.67 and a recall of 0.29. The over-fitting theme is continued where a generalised shape can be seen in SHAP plots that does not mirror the shape of the species at all. This model is also not handling the inconsistent lighting conditions in this class.



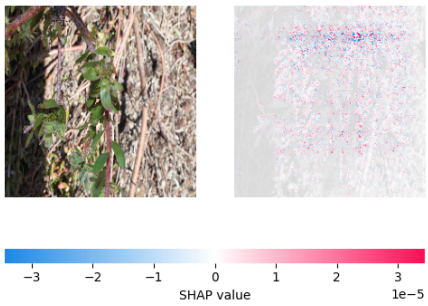
(a) *Erica cinerea* SHAP i



(b) *Erica cinerea* SHAP ii



(c) *Erica cinerea* SHAP iii

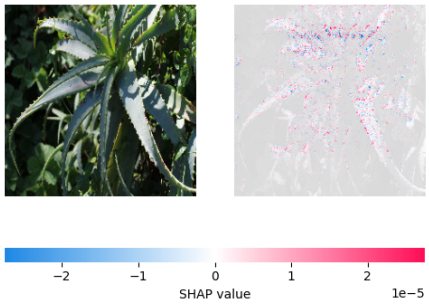


(d) *Erica cinerea* SHAP iv

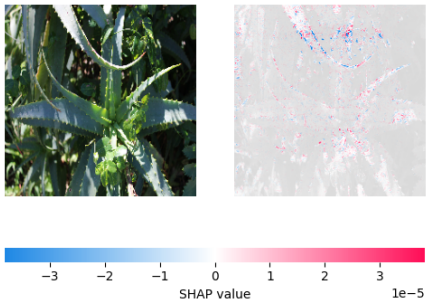
Figure 5.34: *Erica cinerea* leaves subset SHAP Values from Optimised Model A

Low Performing Classes

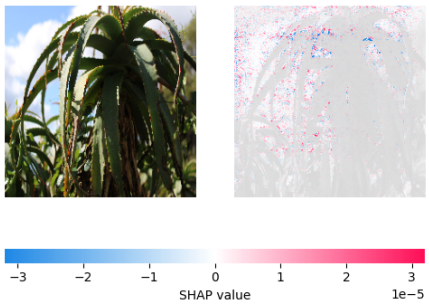
Protea aurea had an F1-score of zero. It was mis-classified 3 times but never correctly predicted. The problems from earlier still exist with this class in that the model is unable to identify features because of poor-lighting conditions. The Squeeze-and-Excite block has allowed the model to not focus on the background noise as can be seen in all for SHAP plots, but Optimised Model A is clearly over-fitting on features from other classes which has been a common theme for this model.



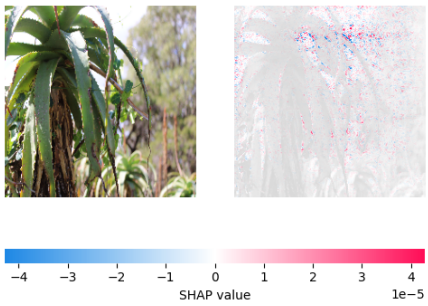
(a) *Protea aurea* SHAP i



(b) *Protea aurea* SHAP ii



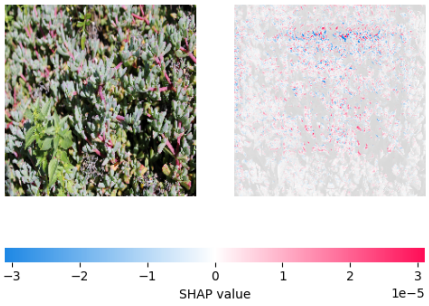
(c) *Protea aurea* SHAP iii



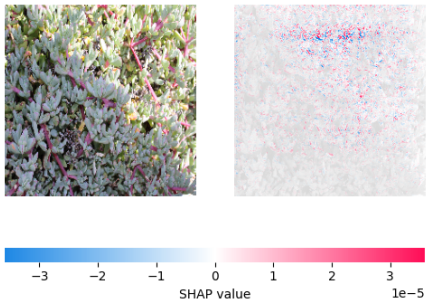
(d) *Protea aurea* SHAP iv

Figure 5.35: *Protea aurea* leaves subset SHAP Values from Optimised Model A

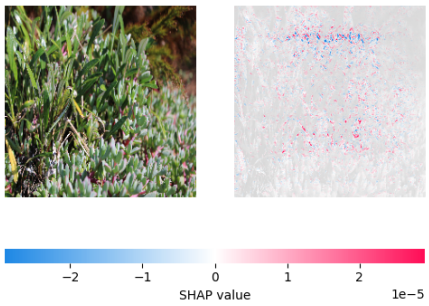
Carpobrotus chilensis no true positives and a zero for F1-score. The over-fitting problem continues here where the model is recognising a general shape and forcing each image to a prediction.



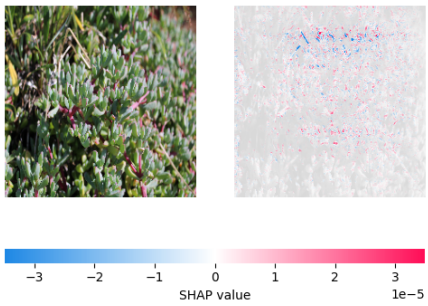
(a) *Carpobrotus chilensis* SHAP i



(b) *Carpobrotus chilensis* SHAP ii



(c) *Carpobrotus chilensis* SHAP iii



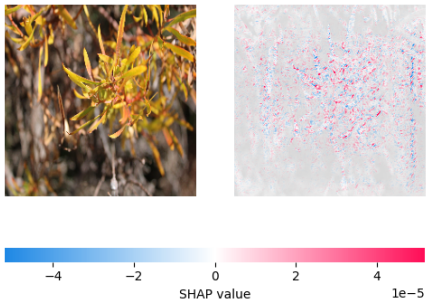
(d) *Carpobrotus chilensis* SHAP iv

Figure 5.36: *Carpobrotus chilensis* leaves subset SHAP Values from Optimised Model A

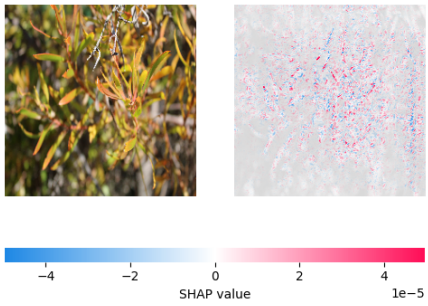
Optimised Model B

High Performing Classes

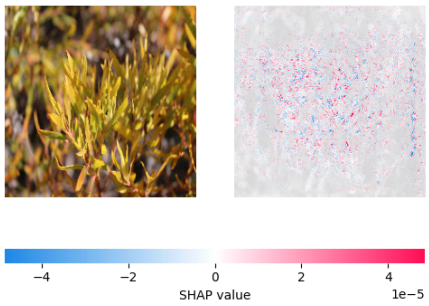
Consistently throughout all 6 models *Leucadendron salignum* has emerged as a top performing class and it achieves an F1-score of 0.93 for this model. It actually has one False Positive when it was incorrectly classified as *Leucadendron argenteum*. The SHAP analysis is shown in 5.37. The over-fitting problem present in Optimised Model A is now gone and the model is learning distinctive features from this class. As can be seen in all four SHAP plots, the features contributing to the prediction are both larger and more spread out as compared to the SHAP plot for the Transfer Models, showing that Optimised Model B is capable of learning significantly more advanced features.



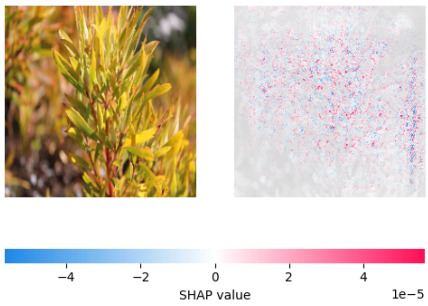
(a) *Leucadendron salignum* SHAP i



(b) *Leucadendron salignum* SHAP ii



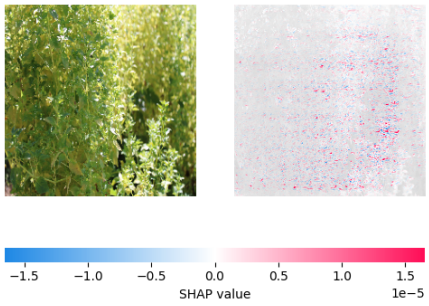
(c) *Leucadendron salignum* SHAP iii



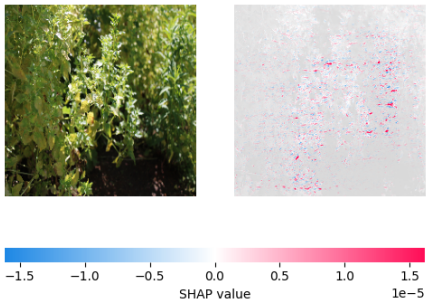
(d) *Leucadendron salignum* SHAP iv

Figure 5.37: *Leucadendron salignum* leaves subset SHAP Values from Transfer Model B

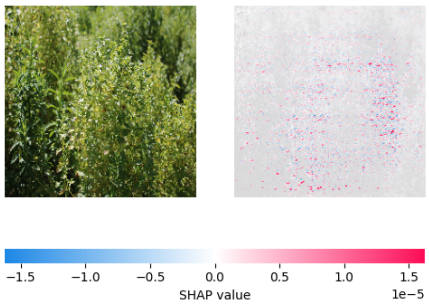
Protea repens achieved a high F1-score of 0.93 with a precision of 1 and a recall of 0.88. Looking at the SHAP plots shows that the model is learning image features which have leaf edges. There is some background noise in SHAP iv which is contributing to the prediction and possibly causing mis-classifications in other classes.



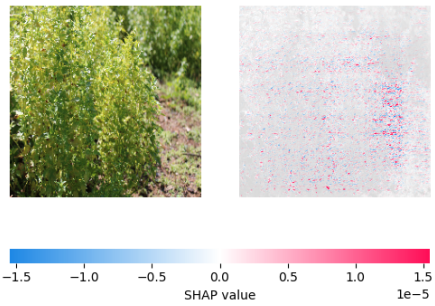
(a) *Protea repens* SHAP i



(b) *Protea repens* SHAP ii



(c) *Protea repens* SHAP iii

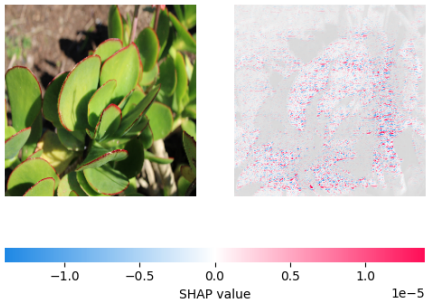


(d) *Protea repens* SHAP iv

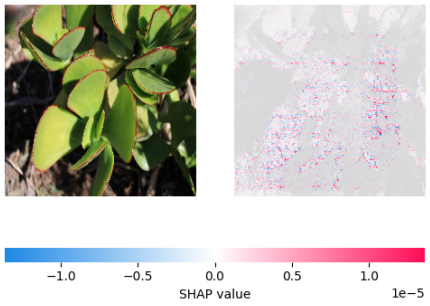
Figure 5.38: *Protea repens* leaves subset SHAP Values from Transfer Model B

Mis-Classified Classes

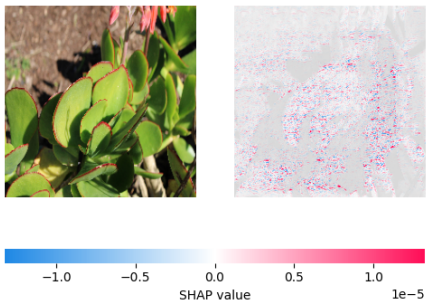
Cotyledon orbiculata was a poor performing class with an F1-score of 0.36, but the precision was 1.00 and the recall was 0.22. It had only 2 True Positives in the confusion matrix and was mis-classified 6 times, 4 of which were with *Protea cynaroides*. Looking at the SHAP plots in 5.39, the over-fitting problem from Optimised Model A is clearly gone. This model is learning complex features in the images, but it can be seen the model is not truly understanding the leaf shape as a feature. The SHAP values are highest on lighter colourations of the waxy surfaces. The negative SHAP values are found on the darker patches. This again points to the importance of dealing with lighting in the training data.



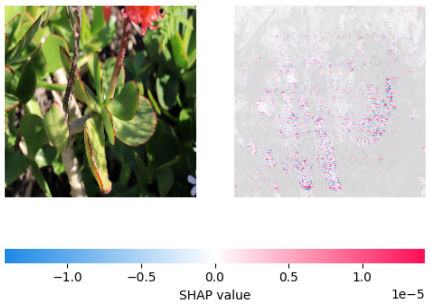
(a) *Cotyledon orbiculata* SHAP i



(b) *Cotyledon orbiculata* SHAP ii



(c) *Cotyledon orbiculatam* SHAP iii

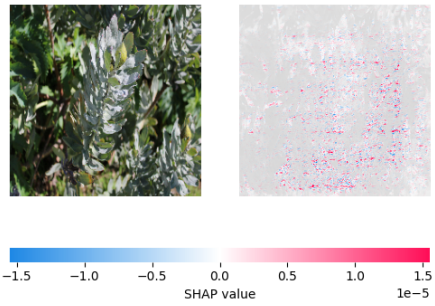


(d) *Cotyledon orbiculata* SHAP iv

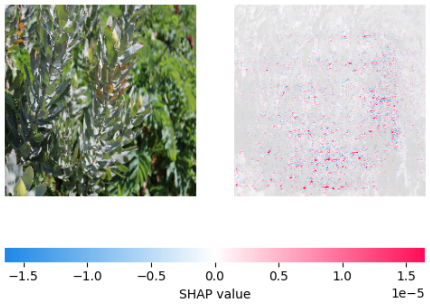
Figure 5.39: *Cotyledon orbiculata* leaves subset SHAP Values from Transfer Model B

Low Performing Classes

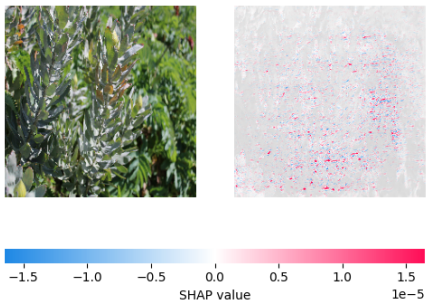
Strangely *Leucadendron argenteum* was poorly classified in this model. it is the only class that achieved a 0 F1-score for Optimised Model B. It actually received an F1-score of 0.15 for Optimised Model A. It has been mis-classified 10 times, with all ten being assigned to *Protea cynaroides*. This mis-classification emerged only in this model. The SHAP plots in 5.40 show that the contributions to the prediction are not the actual leaf or plant. The positive SHAP values are clustered around background features, apart from SHAP iv where a leaf is clearly defined.



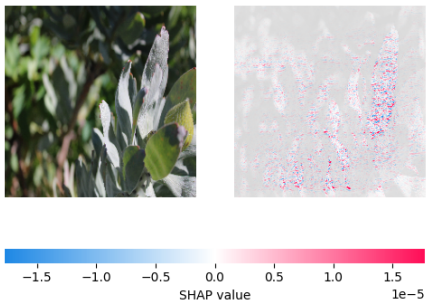
(a) *Leucadendron argenteum* SHAP i



(b) *Leucadendron argenteum* SHAP ii



(c) *Leucadendron argenteum* SHAP iii



(d) *Leucadendron argenteum* SHAP iv

Figure 5.40: *Leucadendron argenteum* leaves subset SHAP Values from Transfer Model B

5.3.5 Discussion

The Optimised Models show much better performance. The introduction of batch normalisation and increase in dropout severely limited the over-fitting in Optimised Model B. The SE block has demonstrably improved the complexity of the features seen in the SHAP plots of Optimised B.

However, Optimised Model A shows an alarming trend in the SHAP analysis. The model has learned very general features and then forced most of the prediction to adhere this feature. Despite the high validation accuracy, this model seems severely over-fitted and is possibly the least reliable of all models as it has not demonstrated an ability to learn nuance features. It also shows that it has learned significant amounts of background

noise. The SE block in this case is acting against model performance. This suggests that the SE block does not perform well when there is excessive background noise in the training data, and in fact makes the model perform worse.

Optimised Model B however does not have this problem. it shows the best results of all models, with minimal over-fitting and a validation accuracy of 76%. This model still struggled with certain classes, but the Squeeze-and-Excite block showed suitable improvements for dealing with classes that had shadows. *Erica cinerea* still had a low F1-score of 0.44 and no true positives.

Regardless of any model improvements, the biggest influence on model performance is image quality of the training set. Optimised Model A showed that more specialised architecture can lead to negative performance if the training data has too much noise and irregularities. Before models can be fine-tuned, the quality of the training data has to be assessed and improved.

5.4 Model Comparison

The previous sections showed the model evaluation procedure for 6 separate models. There were 3 models each trained on two separate datasets. Due to the small dataset size disturbances in the validation set can have large impacts on the results so for this summary, a validation is set randomly from 20% of the data, and every model is run on the same validation set to ensure consistent results. this is why the validation accuracy shown below is not the same as the weighted average accuracy metrics shown in the classification reports. The results of all six are summarised below:

Table 5.7: Model Performance on Datasets A and B

Model	**Accuracy**	**Loss**	**F1-score**	**Top 5 Acc.**	**Top 10 Acc.**
		Dataset A			
Baseline Model	0.337778	3.178614	0.293444	0.617778	0.786667
Transfer Learning Model	0.528889	2.092186	0.533157	0.791111	0.848889
Optimised Model	0.690476	1.154175	0.651683	0.886905	0.952381
		Dataset B			
Baseline Model	0.466258	1.807953	0.419244	0.815951	0.969325
Transfer Learning Model	0.650307	1.308513	0.611181	0.883436	0.938650
Optimised Model	0.766871	1.041047	0.744890	0.883436	0.920245

5.4.1 Discussion

Table 5.7 shows that the Optimised Model outperforms the other two models in all metrics. The Image Quality Assessment evidently improves model performance, demonstrating the importance of high quality data for training CNNs.

The accuracy measures what proportion of correct predictions from the total number of evaluated models. The Optimised Model shows the best results for both datasets. with 69.05% on Dataset A and 76.69% on Dataset B. The optimised model also shows the lowest loss across both datasets which is the measure of the difference between predictions an actual labels. The lowest loss thus indicates the models predictions are closest to the actual values, although it is noted that the loss is much lower in the Dataset B models in general.

Due to the imbalance in the datasets, F1-score is a useful metric at evaluation how well precision and recall are balanced. Again the optimised model shows the best F1-scores at 65.17% and 74.49% respectively. The Optimised Model also wins the Top-K accuracy metrics, but it is noted that the Top-10 accuracy is actually higher for Dataset A.

All models show improved performance when trained on Dataset B compared to Dataset A which indicates that the Image Quality Assessment was effective at removing low quality data, and made it easier for the models to learn features and make more accurate predictions.

While these results are expected, it is interesting to note that the Top 5 and Top 10 accuracy metrics also maintain high performance. This actually suggests that the IQA did not actually impede the models' ability to generalise to less confident predictions. This was observed in the SHAP analysis for the Baseline Model A, where certain classes showed distinct and somewhat complex shapes as features despite the limitations of the model

5.5 Experiment 2: Hyper-Parameter Tuning

Experiment 2 was performed to quantitatively decide on the most suitable hyper-parameters for the model selected from the design process. Each parameter below was measured while all other parameters were kept constant. This process was repeated 9 times (per experiment) and the results are presented below:

5.5.1 Experimental Results

Table 5.8: Model Performance Metrics Across Different Hyperparameters

Configuration	Accuracy	F1 Score	Loss	Top 5 Accuracy	Top 10 Accuracy
Dropout Rate					
Dropout 0.3	0.723926	0.655200	1.261531	0.865031	0.901841
Dropout 0.5	0.766871	0.744890	1.041047	0.883436	0.920245
Dropout 0.7	0.717791	0.689902	1.159770	0.914110	0.944785
Number of Unfrozen Layers					
0 Layers	0.662577	0.671930	1.279569	0.852761	0.963190
15 Layers	0.687117	0.583604	1.129590	0.907975	0.969325
30 Layers	0.766871	0.744890	1.041047	0.883436	0.920245
Input Image Size					
150x150	0.680982	0.645163	1.174687	0.926380	0.957055
224x224	0.766871	0.744890	1.041047	0.883436	0.920245
299x299	0.773006	0.753822	0.933734	0.901841	0.938650

Overall the best hyper-parameters are presented below:

Table 5.9: Summary of Best Hyper-parameters

Hyperparameter	Setting	Accuracy	F1 Score	Loss	Top 5 Accuracy	Top 10 Accuracy
Dropout	0.5	0.766871	0.744890	1.041047	0.883436	0.920245
Unfrozen Layers	30	0.766871	0.744890	1.041047	0.883436	0.920245
Input Image Size	299x299	0.773006	0.753822	0.933734	0.901841	0.938650

5.5.2 Discussion

It appears as though Dropout has a maximal limit when applied to models before performance starts to suffer. Too little dropout appears to actually be better than too much. This is expected as too much dropout will undoubtedly lead to weaker performance.

Interestingly the number of unfrozen layers in the transfer learning models appears to have a linear impact on model performance. The more layers unfrozen, the better the model performs. This is somewhat expected for Fynbos classification. By increasing the

number of trainable layers, the features that the transfer learning model already knows are refined and enhanced [60].

In terms of the input image size, the largest size that ResNet-50 accepts is 299x299 (explained in [7]) and this shows the best performance, albeit it is not a significant upgrade on 224x224. The increase in training time and computational requirements is the trade-off.

The best performing hyper-parameters will then be combined and used in Experiment 3.

5.6 Experiment 3: Model Generalisation

The final experiment measures the ability of the tuned model to generalise to new data. This is vital for FLORA-CNN as the model needs to be trained on images of 9000 species of Fynbos eventually without having the performance affected too much. It would be assumed that as the training set grows in size, the parameters will have to be monitored and adjusted to maintain or improve performance. Five new species were added to the training set. Each class contains 15 images to avoid bias and each class had to score a minimum of 2 on the IQA before it could be ingested. Further detail on the experiment can be found in 4.2.

5.6.1 Experimental Results

The results of the generalised model are presented below starting with a look at classification versus training accuracy in 5.41.

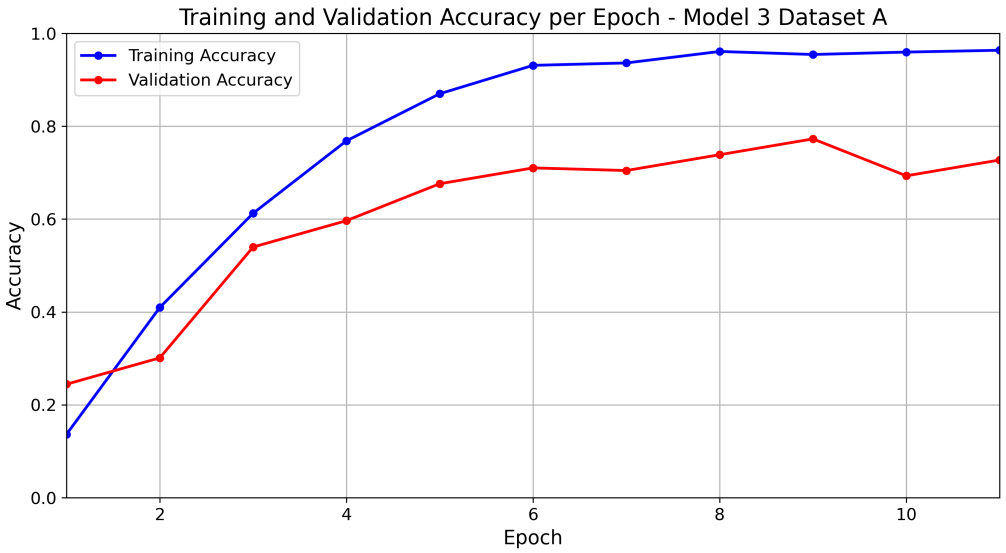


Figure 5.41: Training and Validation Accuracy per Epoch for Model 3 Dataset A

The performance of the model is similar to that of Optimum Model B, but is improved by the additional hyper-parameter tuning performed in the previous experiment. The validation accuracy converges around epoch 6 at around 75% but it does show some minor instability afterwards, likely due to the new species. The over-fitting in this model is the lowest achieved out of all prior models with the validation accuracy tracking the training accuracy suitably. There is still room for improvement, but it is expected that the larger the dataset becomes, the lower the over-fitting becomes. The model is suitably generalised considering it has been trained on natural images. The loss graphs shown in 5.42 also show a similar pattern to Optimised Model B. The training loss approaches zero and converges at around 0.2 by epoch 8 and the validation loss continues to decrease, albeit more slowly than the training accuracy.

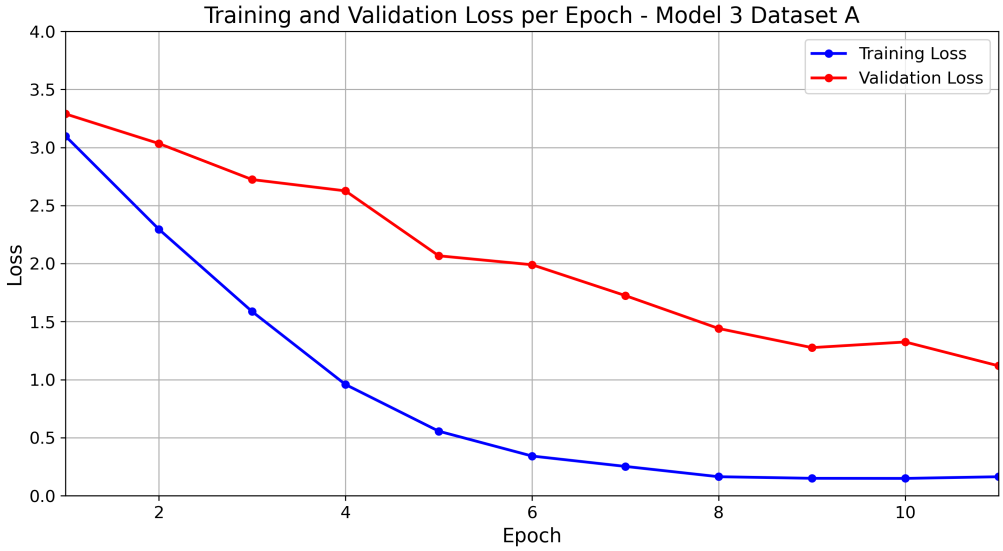


Figure 5.42: Loss

The model still has over-fitting despite the additions of optimised dropout and batch normalisation discussed in 2.3.2. However the over-fitting is noticeably reduced in this model. It appears as though there is a maximum amount of over-fitting that can be realistically expected when training models on small datasets, particularity datasets of natural images. The top-k accuracy graphs are shown below in 5.43.

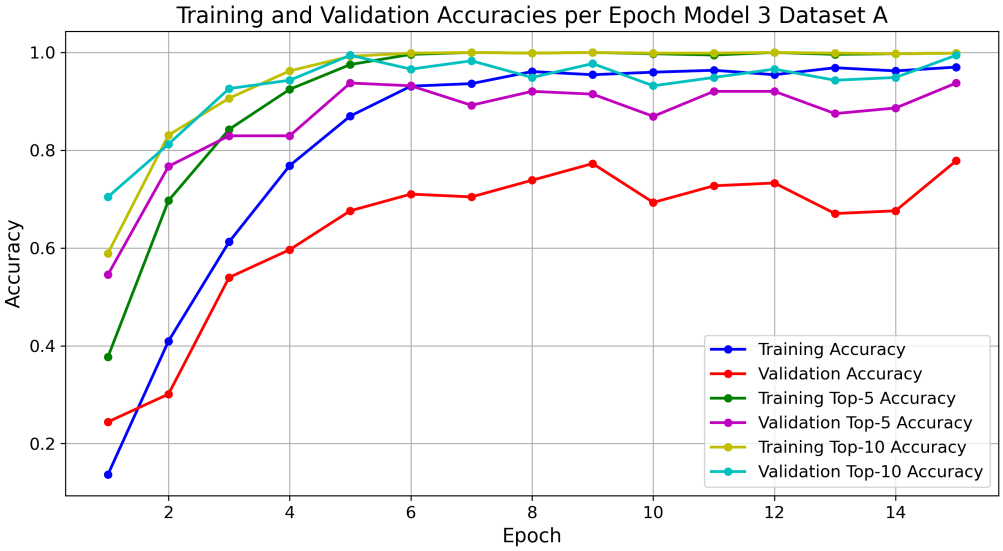


Figure 5.43: Top K accuracy

Both top-5 and top-10 validation accuracy metrics achieve scores greater than 90% by the final epoch. The model seems to almost always be able to get the correct prediction

within the first five guesses.

The classification report is presented below showing the new classes:

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Aloe arborescens	1.00	0.83	0.91	6
Aristea capitata	1.00	1.00	1.00	4
Asparagus capensis	1.00	1.00	1.00	3
Baloskion tetraphyllum	1.00	1.00	1.00	4
Carpobrotus edulis	1.00	1.00	1.00	4
Cotyledon orbiculata	1.00	0.11	0.20	9
Curio talinoides	1.00	1.00	1.00	8
Erica arborescens	0.83	1.00	0.91	5
Erica cinerea	1.00	0.14	0.25	7
Erica discolor	1.00	0.83	0.91	6
Erica mammosa	1.00	0.50	0.67	2
Erica regia	0.67	1.00	0.80	2
Helichrysum petiolare	0.92	1.00	0.96	11
Lessertia frutescens	1.00	0.67	0.80	3
Leucadendron argenteum	1.00	0.10	0.18	10
Leucadendron laureolum	1.00	0.14	0.25	7
Leucadendron salignum	0.64	1.00	0.78	7
Melianthus major	0.91	1.00	0.95	10
Mimetes hirtus	1.00	1.00	1.00	3
Mutisia orbignyana	0.40	1.00	0.57	2
Oscularia deltoides	1.00	1.00	1.00	11
Osyris lanceolata	1.00	1.00	1.00	3
Ozothamnus leptophyllus	1.00	1.00	1.00	3
Pelargonium crispum	0.20	1.00	0.33	5
Protea cynaroides	0.59	0.71	0.65	14
Protea neriifolia	1.00	0.89	0.94	9
Protea repens	1.00	1.00	1.00	8
Strelitzia reginae	0.90	0.90	0.90	10
Accuracy			0.78	176
Macro Avg	0.89	0.82	0.78	176
Weighted Avg	0.90	0.78	0.76	176

The macro weighted accuracy is 78% which shows suitable performance. Each of the

new species *Asparagus capensis*, *Erica mammosa*, *Erica regia*, *Lessertia frutescens* and *Mimetes hirtus* achieve F1-scores greater or equal to 0.67 with *Asparagus capensis* and *Mimetes hirtus* achieving perfect F1-scores of 1.00. This demonstrates that the model is able to adapt to new species and to learn new features.

There is possible bias introduced in this step as the new images were not collected as randomly as the original datasets and the learnings from the modeling was applied in order to get better quality images.

The confusion matrix is presented below in 5.44:

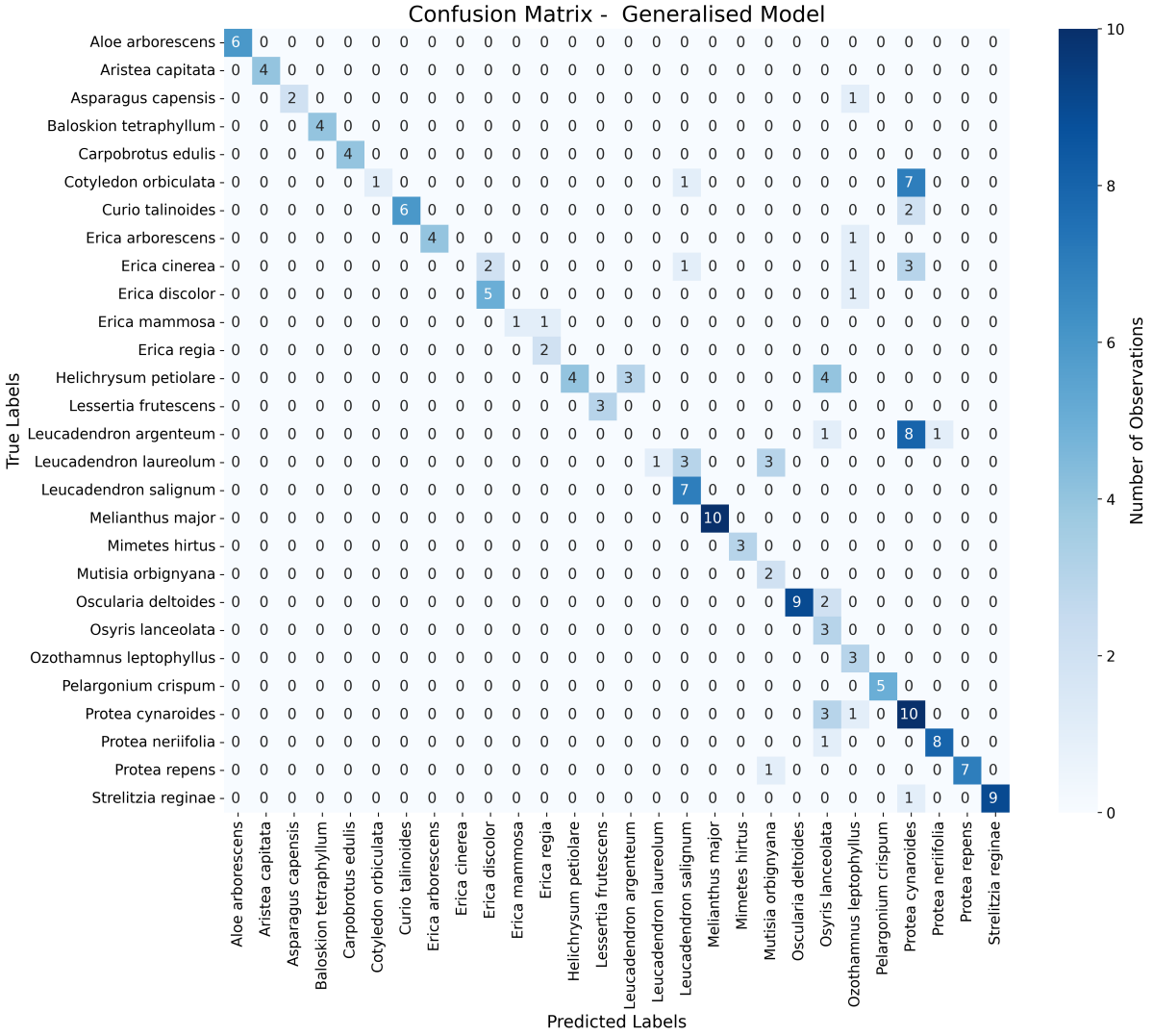


Figure 5.44: Confusion Matrix

The confusion matrix shows a strong diagonal element with multiple species with no misclassifications. The original offenders are still present with *Protea cynaroides* showing multiple False Positives. Of the new classes *Erica mammosa* and *Erica regia* were mis-

classified as each other on one occasion each.

5.6.2 Discussion

This model represents the best designed model in terms of predicative capability. It shows suitable generalisation to 5 new classes as the model performance more or less stayed the same. The problems arising within the model have come from the original offenders with *Protea cynaroides* proving especially difficult to classify due to the inconsistent lighting conditions in the training images. With the classification accuracy hovering around 78% after hyper-parameter tuning, the model shows suitable generalisation and the ability to learn the features that differentiate classes.

This model demonstrates its suitability to be expanded into a research tool, although the importance of optimal data cannot be stressed enough. The IQA needs to be expanded to detect shadows and poor lighting, and all new species being uploaded needs to meet a minimum standard of image quality before the model should be allowed to train on it.

Chapter 6

Conclusions and Future Work

6.1 Conclusion

This thesis presented a successful solution to the classification of Fynbos images in the form of a suitably designed Convolutional Neural Network that is able to ingest natural or field images of different Fynbos species, learn features from these images and then use this knowledge to perform multi-class classification when presented with new data. The F1-score of tuned model reaches 75.38% and the validation accuracy sits at 77.30%. In the context of working with natural images, these results show a suitable and viable model for use in the FLORA application. While previous iterations of FLORA were able to acheive 86.66%, 91.1% and 89.74% for GRNN, PNN and k-NN algorithms respectively.

The learnings focus on training a CNN with a small dataset of natural images, which diverges significantly from most models built in literature which use large datasets of images and heavy feature engineering. The core learning of this project was the importance of data quality for use in neural networks. After initial experimentation, it was found that the original dataset contained images that had considerable background noise and other irrelevant features, and these contributed to poor predictive capability. To solve this a novel tool called the Image Quality Assessment was produced which rated each image on features such as resolution, sharpness, colour balance, background noise and edge features. Using this tool to remove classes which had too many low quality images significantly improved model performance and severely reduced over-fitting.

Three models were designed - a simple baseline model, a model using transfer learning from the ResNet-50 pre-trained model and a third model which combined learnings and

observations from the previous two - and each one was trained on the cleaned and uncleaned dataset resulting in six total models. The results of the six models are presented in 5.4 and it was determined that the Optimised Model trained on the cleaned dataset performed the best.

Models trained on the cleaned data all showed less over-fitting, higher validation accuracy measures and higher F1-scores with the Optimised Model showing a 7,6% increase in validation accuracy when trained on the cleaned dataset and an 9.3% increase in F1-score. More importantly the SHAP plots for the model trained on uncleaned data showed severe over-fitting and irrelevant learning of background features which were noticeably removed in the model trained on the cleaned dataset.

After the model was selected, three of the most influential hyper-parameters were chosen in 5.5 - Dropout, the number of unfrozen layers and the input image size - and the model was tuned to find the best configuration. It was found that a dropout of 0.5, the number of unfrozen layers being set to 30 (out of 54) and increasing the input image size to the maximum of 299x299 resulted in the best performing model with a marginal boost in accuracy and F1-score. This model was then tested on its ability to generalise to new data whereby 5 new classes with previously unseen images were added to the cleaned dataset and the model was retrained. The final model was able to ingest the new images, and validation accuracy actually marginally improved by around 1% and the F1-score improved by around 2%. The model showed excellent generalisation ability to new data.

A core objective for this thesis was to ensure high model explain-ability - or the ability to explain why a model made a certain prediction. To facilitate this, SHAP plots were generated for classifications of interest. The SHAP plots provided visual representations of the pixels in each image and positively and negatively contributed to the prediction. This process allows the modeler and reader to understand the features that each model actually learns from the and to observe the impact on these features by changing model complexity. The SHAP plots were key for several design choices including the Squeeze-And-Excite block used in the Optimised Models. Through the SHAP plots it was observed that Optimised Model A was over-fitting to background noise because of the increase in model complexity. When this noise was removed for Optimised Model B, the SE block contributed to a significant increase in the complexity of learned features.

As a result of these observations, the four primary objectives are achieved, namely:

- The creation of a labeled training dataset of Fynbos images manually collected from the field.

- The design of a suitable CNN that can train on this dataset and perform multi-class classification.
- The optimisation of the model through hyper-parameter tuning
- Successful implementation of SHAP to visualize model predictions.

As part of the secondary objectives, a front-end was designed for FLORA using the Angular framework. FLORA-CNN should eventually represent the first live use case of the FLORA application.

6.2 Future Work

Based on the learnings of this project, the following recommendations are advised for future work:

- Grow the dataset both by increasing samples available in the current classes and by introducing new classes. The model showed excellent generalisation and should be able to eventually be trained on all 9000 species of Fynbos with some minor changes.
- Involve a domain expert. A key aspect of this project's continued success will be human validation of data. A suitably trained botanical expert can contribute greatly to enhancing the trust of the model.
- The models in this thesis were optimised on leaf shape, but Fynbos also contain distinctive flowers, stems and roots. The SHAP plots showed that even with this optimisation for leaves, flowers and stems often formed part of the prediction. It is suggested to build separate models to focus on each component and combine them in the prediction.
- Build a monitoring and evaluation tool to see how the model reacts to growing training data.
- Focus on automatic CNN architecture search and hyper parameter tuning to overcome the trial and error approach in finding a suitable model.
- Enabling the CNN to work with higher resolution images. It can be seen from Table 5.8 that recognition accuracy improved with increased input image resolution. The

accompanying increase in computational cost can be mitigated elsewhere in the model.

- Develop or integrate additional tools (such as LIME) to be used in addition to SHAP to aid with explain-ability.
- Build a robust back-end system for the application.
- Develop strategies for using unclear and blurry pictures such as what the end users would be expected to capture.

Bibliography

[1] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," 2022.

[2] A. Upreti, "Convolutional neural network (cnn). a comprehensive overview," 08 2022.

[3] P. Goldblatt and J. Manning, "Plant diversity of the cape region of southern africa," *Annals of the Missouri Botanical Garden*, vol. 89, 11 2001.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1106–1114.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. [Online]. Available: https://ar5iv.org/abs/1409.1556

[7] K. He *et al.*, "Deep residual learning for image recognition," 2015.

[8] C. Szegedy *et al.*, "Going deeper with convolutions," *arXiv preprint arXiv:1409.4842*, 2014. [Online]. Available: https://ar5iv.org/abs/1409.4842

[9] A. Rebelo *et al.*, *Fynbos Biome*. Department of Environmental Affairs, Jan. 2005, pp. 52–219.

[10] S. Katz, "Fynbos leaf-based online recognition application," in *2013 Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*, 2011.

[11] M. A. Ramone, "Flora – fynbos leaf online recognition application," University of Cape Town, Tech. Rep., 2013.

[12] K. Naidoo and S. Winberg, "Gpu acceleration of the fynbos leaf-based online recognition application," vol. 29, 2015.

[13] R. Makumborenga, "Enhancement of the fynbos leaf optical recognition application (flora-e)," Master's thesis.

[14] [South African National Biodiversity Institute], "Fynbos biodiversity under threat," [SANBI], 2024. [Online]. Available: https://www.sanbi.org/wp-content/uploads/2018/03/sustaininglifeinthefynbos.pdf

[15] T. Kraaij and B. van Wilgen, "Drivers, ecology, and management of fire in fynbos," in *Fynbos: Ecology, Evolution, and Conservation of a Megadiverse Region*. Oxford University Press, 09 2014. [Online]. Available: https://doi.org/10.1093/acprof:oso/9780199679584.003.0003

[16] P. Holmes *et al.*, "Guidelines for restoring lowland sand fynbos ecosystems," 2022.

[17] Sustainable Development Goals United Nations, "Sdg interactions climate sei2023.010," 2023. [Online]. Available: https://sdgs.un.org/sites/default/files/2023-03/sdg-interactions-climate-sei2023.010_0.pdf

[18] N. Allsopp *et al.*, *Fynbos: ecology, evolution, and conservation of a megadiverse region*, 1st ed. Oxford University Press, 01 2014, pp. 248–272.

[19] D. Cardoso *et al.*, "Amazon plant diversity revealed by a taxonomically verified species list," *Proceedings of the National Academy of Sciences*, vol. 114, no. 40, pp. 10 695–10 700, 2017. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1706756114

[20] A. M. Streich and K. A. Todd. (2014, Jan) Classification and naming of plants. Nebraska Extension Publications. EC1272. [Online]. Available: https://extension.unl.edu/ClassificationandNamingofPlantsec1272.pdf

[21] M. Richards *et al.*, "Soil factors and competition as determinants of the distribution of six fynbos proteaceae species," *Oikos*, vol. 79, no. 2, pp. 394–406, 1997, accessed 13 Jan. 2024. [Online]. Available: https://doi.org/10.2307/3546024

[22] F. Schurr *et al.*, "Fynbos proteaceae as model organisms for biodiversity research and conservation," *South African Journal of Science*, vol. 108, pp. 12–16, 01 2012. [Online]. Available: http://www.scielo.org.za/scielo.php?script=sci_arttext&pid=S0038-23532012000600005&nrm=iso

[23] S. Higgens *et al.*, "An ecological economic simulation model of mountain fynbos ecosystems: Dynamics, valuation and management," *Ecological Economics*, vol. 22,

no. 2, pp. 155–169, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921800997005752

[24] M. Taye, "Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions," *Computation*, vol. 11, p. 52, 03 2023.

[25] A. Khan *et al.*, "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, vol. 53, no. 8, Apr. 2020. [Online]. Available: http://dx.doi.org/10.1007/s10462-020-09825-6

[26] J. Brownlee. (2019) Pooling layers for convolutional neural networks. https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/. Accessed: [August 2022]. [Online]. Available: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/

[27] D. Hubel and T. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *J Physiol*, vol. 148, no. 3, pp. 574–591, Oct 1959.

[28] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[29] Y. LeCun, "A theoretical framework for back-propagation," *Unknown*, Aug 2001. [Online]. Available: https://www.researchgate.net/publication/2360531_A_Theoretical_Framework_for_Back-Propagation

[30] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," in *Int J Comput Vis 115,*, 2015, p. 211–252.

[31] V. H. Phung and E. J. Rhee, "A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets," *Applied Sciences*, vol. 9, p. 4500, 10 2019.

[32] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Lecture Notes in Computer Science*, vol. 8689, 2013.

[33] Z. J. Wang *et al.*, "Cnn explainer: Learning convolutional neural networks with interactive visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, 2021. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2020.3030418

[34] J. Brownlee. (2019) A gentle introduction to padding and stride for convolutional neural networks. https://machinelearningmastery.

com/padding-and-stride-for-convolutional-neural-networks/. Accessed: 2024-01-19. [Online]. Available: https://machinelearningmastery.com/padding-and-stride-for-convolutional-neural-networks/

[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[36] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 06 2014.

[37] Z. M. Chng. (2021) Using activation functions in neural networks. Accessed: 2023-04-04. [Online]. Available: https://machinelearningmastery.com/using-activation-functions-in-neural-networks/

[38] L. Lu, "Dying relu and initialization: Theory and numerical examples," *Communications in Computational Physics*, vol. 28, no. 5, pp. 1671–1706, Jun. 2020. [Online]. Available: http://dx.doi.org/10.4208/cicp.OA-2020-0165

[39] M. Lin, Q. Chen, and S. Yan, "Network in network," in *International Conference on Learning Representations*, 2014.

[40] K. He *et al.*, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 770–778.

[41] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning*, vol. 37, 2015, p. 448–456.

[42] J. Brownlee. A gentle introduction to batch normalization for deep neural networks. https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/. [Accessed 10-02-2024]. [Online]. Available: https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/

[43] M. Kolarik, R. Burget, and K. Riha, "Comparing normalization methods for limited batch size segmentation neural networks," in *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*. IEEE, Jul. 2020. [Online]. Available: http://dx.doi.org/10.1109/TSP49548.2020.9163397

[44] J. Gu *et al.*, "Recent advances in convolutional neural networks," in *Pattern Recognition*, 2017, pp. 354–377.

[45] N. Sharma, V. Jain, and A. Mishra, "An analysis of convolutional neural networks for image classification," *Procedia Computer Science*, vol. 132, pp. 377–384, 01 2018.

[46] R. Chauhan, K. Ghanshala, and R. Joshi, "Convolutional neural network (cnn) for image detection and recognition," in *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, 12 2018, pp. 278–282.

[47] A. Kamilaris and F. X. Prenafeta-Boldú, "A review of the use of convolutional neural networks in agriculture," *The Journal of Agricultural Science*, vol. 156, no. 3, p. 312–322, 2018.

[48] S. Sladojevic *et al.*, "Deep neural networks based recognition of plant diseases by leaf image classification," *Computational Intelligence and Neuroscience*, vol. 2016, pp. 1–11, 06 2016.

[49] S. M. Lundberg and S. I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 4765–4774. [Online]. Available: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[50] D. Bhatt *et al.*, "Cnn variants for computer vision: History, architecture, application, challenges and future scope," *Electronics*, vol. 10, p. 2470, 10 2021.

[51] Z. Baozhou *et al.*, "An attention module for convolutional neural networks," vol. 12891, 2021.

[52] A. T. Khan *et al.*, "Plant disease detection model for edge computing devices," *Frontiers in Plant Science*, vol. 14, 2023, publisher Copyright: Copyright © 2023 Khan, Jensen, Khan and Li.

[53] A. Pearline and S. Kumar, "Plant species recognition using modified lenet-5 cnn architecture," in *National conference on Recent Trends in Instrumentation and Control (RTIC-2019)*, 08 2020.

[54] M. Hussain, J. Bird, and D. Faria, "A study on cnn transfer learning for image classification," in *Advances in Intelligent Systems and Computing*, vol. 840, 06 2018.

[55] B. Jangid and R. Sharma, "Rice disease detection using deep learning vgg-16 model and flask," Ph.D. dissertation, 03 2023.

[56] S. Padshetty and Ambika, "Leaky relu-resnet for plant leaf disease detection: A deep learning approach," *Engineering Proceedings*, vol. 59, no. 1, 2023. [Online]. Available: https://www.mdpi.com/2673-4591/59/1/39

[57] D. P. Hughes and M. Salathe, "An open access repository of images on plant health to enable the development of mobile disease diagnostics," 2016. [Online]. Available: https://arxiv.org/abs/1511.08060

[58] M. Hossin and M. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, pp. 1–11, 03 2015.

[59] J. Amara, B. Bouaziz, and A. Algergawy, "A deep learning-based approach for banana leaf diseases classification," pp. 79–88, 01 2017.

[60] G. Grinblat *et al.*, "Deep learning for plant identification using vein morphological patterns," *Computers and Electronics in Agriculture*, vol. 127, pp. 418–424, 09 2016.

[61] A. Mao, M. Mohri, and Y. Zhong, "Cross-entropy loss functions: Theoretical analysis and applications," in *ICML'23: Proceedings of the 40th International Conference on Machine Learning*, no. 992, 2023, p. 23803–23828.

[62] S. P. Mohanty, D. Hughes, and M. Salathe, "Using deep learning for image-based plant disease detection," vol. 7, p. 1419, 2016.

[63] M. Nawaz *et al.*, "A robust deep learning approach for tomato plant leaf disease localization and classification," *Scientific Reports*, vol. 12, 11 2022.

[64] J. Vankara *et al.*, "Plant disease prognosis using spatial-exploitation-based deep-learning models," *Engineering Proceedings*, vol. 59, no. 1, 2023. [Online]. Available: https://www.mdpi.com/2673-4591/59/1/137

[65] X. Wang and C. Cheng, "Weed seeds classification based on pcanet deep learning baseline," in *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conferenc*, 12 2015, pp. 408–415.

[66] K. H. Liu *et al.*, "Plant species classification based on hyperspectral imaging via a lightweight convolutional neural network model," *Frontiers in Plant Science*, vol. 13, 04 2022.

[67] Y. Mitani, Y. Fujita, and Y. Hamamoto, "Augmentation on cnns for handwritten digit classification in a small training sample size situation," *Journal of Physics Conference Series*, vol. 1922, p. 012007, 05 2021.

[68] J. Brownlee. How to use data scaling improve deep learning model stability and performance. https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/. [Accessed 31-01-2024]. [Online]. Available: https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/

[69] L. Moraru *et al.*, "Chapter 9 - dempster-shafer fusion for effective retinal vessels' diameter measurement," in *Soft Computing Based Medical Image Analysis*, N. Dey, A. S. Ashour, F. Shi, and V. E. Balas, Eds. Academic Press, 2018, pp. 149–160. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128130872000087

[70] U. Kumar and S. Bharathi, "Visual background extractor with improved sobel operator for moving object detection," *International Journal of Engineering and Technology(UAE)*, vol. 7, pp. 1298–1302, 07 2018.

[71] O. Vincent and O. Folorunso, "A descriptive algorithm for sobel image edge detection," 01 2009.

[72] E. Tsikata *et al.*, "Automated brightness and contrast adjustment of color fundus photographs for the grading of age-related macular degeneration," vol. 6, p. 3, 03 2017.

[73] R. C. Gonzalez and R. E. Woods, *Digital image processing.* Upper Saddle River, N.J.: Prentice Hall, 2008. [Online]. Available: http://www.amazon.com/Digital-Image-Processing-3rd-Edition/dp/013168728X

[74] M. Segal Rozenhaimer *et al.*, "Cloud detection algorithm for multi-modal satellite imagery using convolutional neural-networks (cnn)," *Remote Sensing of Environment*, vol. 237, p. 111446, 02 2020.

[75] X. Min, G. Zhai, K. Gu, X. Liu, and X. Yang, "Blind image quality estimation via distortion aggravation," *IEEE Transactions on Broadcasting*, vol. 64, no. 2, pp. 508–517, 2018.

[76] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2016.

[77] M. Feurer and F. Hutter, "Hyperparameter optimization," in *Automated Machine Learning.* Springer, Cham, 2019, pp. 3–33.

Appendix A

Appendix A - Image Quality Assessment

Table A.1: An extract of 55 Results From The Image Quality Assessment

Class_Name	Sharpness	Noise	Color_Balance	Background_Consistency	Sharpness_Score	Noise_Score	Color_Balance_Score	Background_Consistency_Score	Resolution_Score	Total_Score
Agathosma serpyllacea	24.69406524364	0.2803775844968553	[72.46993714 72.76679604 62.64113518]	0.014511183133435305	0	2	2	0	2	6
Agathosma serpyllacea	25.501727352971137	0.26737092133267254	[70.41496594 69.13530105 60.44483579]	0.0137459472697702	0	2	2	0	2	6
Agathosma serpyllacea	17.606730637690912	0.2776086199379974	[73.27585015 71.63162857 63.50830009]	0.010373778620792312	0	2	2	0	2	6
Agathosma serpyllacea	12.465135957848705	0.26324871599814287	[69.55726145 68.59228097 54.89339933]	0.006771097736281417	0	2	2	0	2	6
Agathosma serpyllacea	42.9701665951909	0.23517706142714814	[64.25270475 62.29356202 42.23264448]	0.019223694572512426	0	1	1	1	2	5
Agathosma serpyllacea	24.640947053317838	0.20619365129193876	[56.64172154 54.27248194 40.70330483]	0.011030931418546835	0	1	1	0	2	4
Agathosma serpyllacea	17.2973478874972	0.2080228452240078	[57.02243777 54.77071061 37.78696086]	0.008457898111555874	0	1	1	0	2	4
Agathosma serpyllacea	16.83074460988417	0.20347124144135514	[56.4467369 53.91554041 32.77957517]	0.007966661873369031	0	1	1	0	2	4
Agathosma serpyllacea	13.028417916065045	0.2135873657988578	[59.66419104 55.42517705 41.97509333]	0.006292810761318494	0	1	1	0	2	4
Agathosma serpyllacea	21.081602065289484	0.17479015024536207	[50.58284466 45.44337374 39.03651502]	0.010467415098208532	0	1	1	0	2	4
Agathosma serpyllacea	29.09570633627001	0.18380310758820934	[51.59206163 47.93942105 45.64210967]	0.011725726809179833	0	1	1	0	2	4
Agathosma serpyllacea	20.80017597081173	0.2134434614438161	[57.24708183 57.19238389 35.89363459]	0.009358857405511628	0	1	1	0	2	4
Agathosma serpyllacea	14.813775622251065	0.214352246394287	[57.51976433 57.18995885 38.61344575]	0.007640137000995918	0	1	1	0	2	4
Agathosma serpyllacea	18.848454603464557	0.2425796310356644	[64.09930893 64.04964489 43.72398199]	0.008237508026857486	0	1	1	0	2	4
Agathosma serpyllacea	18.52520030896946	0.1891451903104963	[52.14623129 50.10076164 41.32962355]	0.00975038539177748	0	1	1	0	2	4
Agathosma serpyllacea	58.45815124924552	0.24944142926316412	[66.09212795 64.23142296 61.44157524]	0.020031252091061217	0	1	1	1	2	5
Agathosma serpyllacea	19.988871551203356	0.23084746656831048	[60.08599641 58.67483082 60.19333555]	0.010848839452567772	0	1	1	0	2	4
Aloe arborescens	5386.8535799743695	0.1427721417290694	[47.95661318 37.18634893 36.60355577]	0.0676246863107276	2	0	0	1	0	3
Aloe arborescens	18.746061633934346	0.27369177318988597	[72.3658362 71.45505562 53.92675896]	0.009000346618229111	0	2	2	0	2	6
Aloe arborescens	18.7050623888837	0.2610066768005128	[68.97219097 67.94730316 55.23572832]	0.009265193759174397	0	2	2	0	2	6
Aloe arborescens	22.956001943467964	0.23977108557145627	[66.49971321 61.79333224 53.97510321]	0.010058680907483374	0	1	1	0	2	4
Aloe arborescens	33.79982505438574	0.30027598184167403	[77.44949029 78.13864286 65.26476316]	0.013685747385990004	0	2	2	0	2	6
Aloe arborescens	18.380939791566334	0.26217314937498815	[68.05874427 68.56666758 54.73372396]	0.011601318297404127	0	2	2	0	2	6
Aloe arborescens	31.551758426061106	0.24816235331233094	[64.70108483 64.70477758 54.39779792]	0.01485507452196737	0	1	1	0	2	4
Aloe arborescens	15.35537924988081	0.2616201045708841	[69.95774766 67.66764659 58.86659761]	0.007824853066341045	0	2	2	0	2	6
Aloe arborescens	11406.240253762999	0.2641892316631731	[70.99872199 68.69203625 55.2181301]	0.10669162976959758	2	2	2	2	0	8
Aloe arborescens	3346.706382941939	0.20116546335242066	[57.38519742 49.17318559 59.71472727]	0.06292510329212164	2	1	1	1	0	5
Aloe arborescens	2285.735682109606	0.17124521783258667	[43.12471942 46.97243527 25.22707808]	0.06685801302873592	2	1	0	1	0	4
Aloe arborescens	6991.494229527512	0.3129715900616301	[81.16191447 79.47310469 87.82383271]	0.09830273543024914	2	2	2	1	0	7
Aloe arborescens	2167.2775039789362	0.23996899959152798	[64.88798975 62.62501096 51.20138659]	0.06745431704426384	2	1	1	1	0	5
Aloe arborescens	6734.417304585888	0.22656815729604363	[64.73119995 59.41834148 60.14019618]	0.11833226497683834	2	1	1	2	0	6
Aloe arborescens	10527.366911195788	0.1941004570691168	[51.57241149 51.03636656 45.63331861]	0.09682801022670759	2	1	1	1	0	5
Aloe arborescens	10577.359764603258	0.2749733507869215	[71.14771849 71.21317229 58.2139038]	0.14084014985985735	2	2	2	2	0	8
Aloe arborescens	2199.4040312470092	0.187358744542719	[55.95884176 47.89946362 44.02284148]	0.06300078782592919	2	1	1	1	0	5
Aloe arborescens	5997.94876964816	0.277238365989298	[73.1566058 72.94090548 49.85422869]	0.08909046956425562	2	2	2	1	0	7
Aloe arborescens	11937.635286345518	0.25745960492626613	[78.36096907 66.96351788 54.24236124]	0.14745244071774405	2	2	2	2	0	8
Aloe arborescens	1802.9959180647752	0.17214951876014253	[49.16043822 42.62471547 53.80995216]	0.069377662154485	2	1	1	1	0	5
Aloe arborescens	1800.9668476826068	0.23932688614566552	[59.7537564 62.23794786 57.34705026]	0.10094500646706046	2	1	1	1	0	5
Aloe arborescens	1239.8599741594514	0.24944541365604048	[66.5534801 63.92102953 60.4244236]	0.0617452450146172	2	1	1	1	0	5
Aloe arborescens	5535.271679120106	0.23597858658628879	[57.63475691 61.9818757 53.16772527]	0.09496643051351811	2	1	1	1	0	5
Aloe arborescens	5200.644888145669	0.20135236774065102	[52.32874904 51.93028339 49.47151185]	0.08813584434480567	2	1	1	1	0	5
Aloe arborescens	5231.612661462304	0.23563709519857956	[59.07170853 63.23853337 49.08933]	0.0907025514336043	2	1	1	1	0	5
Aloe arborescens	5607.712550714335	0.15796806848532183	[38.91507958 42.74703522 31.44225739]	0.09709219437067655	2	0	0	1	0	3
Aloe arborescens	1344.46187731104	0.19989474004263957	[45.80837898 53.29569444 48.77040981]	0.0643172352894796	2	1	1	1	0	5
Aloe arborescens	6350.912398234132	0.1713790304303028	[42.77507552 44.20002122 44.99543455]	0.079965218710713	2	1	1	1	0	5
Aloe arborescens	2369.3430167208844	0.19456656878886075	[53.95387 48.46497027 56.01932052]	0.07887324614046236	2	1	1	1	0	5
Aloe arborescens	7254.3446000486365	0.20736689202977607	[50.90880613 54.25728634 51.76921326]	0.12002541996621426	2	1	1	2	0	6
Aloe arborescens	4557.521383795394	0.1910944609456642	[50.30756792 50.02599168 42.62408945]	0.10533127000632304	2	1	1	2	0	6
Aloe arborescens	4802.663773507004	0.21925966658456453	[59.36279658 56.05729132 48.90551143]	0.09242474700081647	2	1	1	1	0	5
Aloe arborescens	7126.262796101638	0.2730390934097869	[76.23797623 71.67290153 42.24580392]	0.11962904265340939	2	2	2	2	0	8
Aloe arborescens	5180.84479438049	0.19216583538966345	[48.62865928 51.01281601 39.6750284]	0.10118270647769954	2	1	1	2	0	6
Arctotis stoechadifolia	10887.73408322418	0.24105250869566522	[62.1094075 61.36256641 62.76341833]	0.1281273942733678	2	1	1	2	0	6
Arctotis stoechadifolia	6374.364524998796	0.2328763994867218	[60.19495189 59.3040176 59.93890898]	0.1387155532674172	2	1	1	2	0	6
Arctotis stoechadifolia	5022.5801132593315	0.24397719564444906	[60.38814426 63.60669617 60.73948826]	0.11947318896872705	2	1	1	2	0	6
Arctotis stoechadifolia	18103.211423171437	0.19037391922270996	[47.10782847 49.49129033 47.94322303]	0.13146518577555408	2	1	1	2	0	6

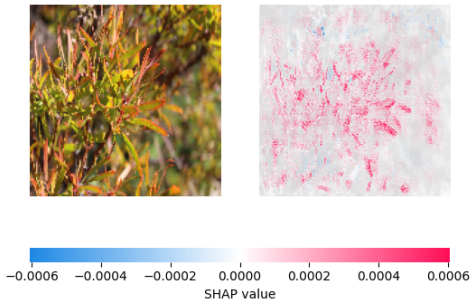
Appendix B

Appendix B - Additional SHAP Plots

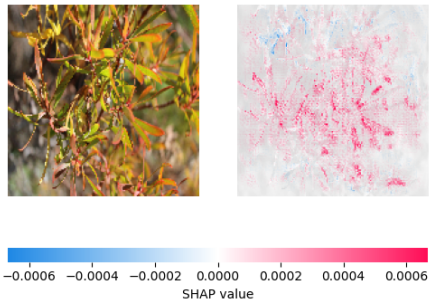
Baseline A

High Performing Classes

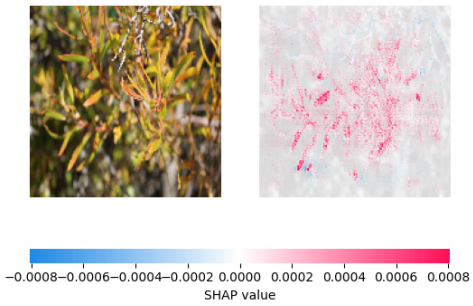
Given the the F1-Score of *Leucadendron salignum* being 0.70 for Baseline A, a SHAP Analysis was performed from images in this class with the intention of visualising what features the model was recognising. These SHAP visualisations show distinct regions in *Leucadendron salignum* leaves that positively contribute to the model performance. These regions are highlighted in pink and it can be observed that the regions cluster around leaves and the shapes of leaves. The fourth image also contains noticeable background and the model still manages to identify the leaves as features. The needle-like shapes of the leaves discussed in 2.1.2 are the clear distinction as well as the very unique brown colouring. This suggests that the model has identified and learned features of this class extremely well and this is expressed by the high F1-Score. Classes that have similar distinct features should be expected to return similar F1-Scores.



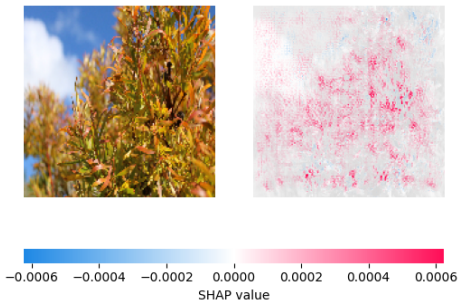
(a) *Leucadendron salignum* SHAP i



(b) *Leucadendron salignum* SHAP ii



(c) *Leucadendron salignum* SHAP iii



(d) *Leucadendron salignum* SHAP iv

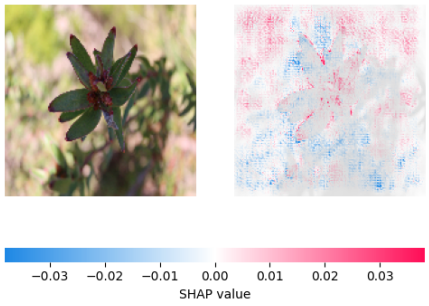
Figure B.1: *Leucadendron salignum* leaves subset SHAP Values from Baseline A

This shows that even the relatively simple architecture of the Baseline Model is capable of identifying features in leaves. The needle shape of the leaves and colour are features that should be observed and enhanced by the data augmentation going forward as these seem to be important to the model prediction. Conversely the model seems to be able to capture larger features (like clusters of leaves in all four images) quite well here. This would suggest that an architecture that favors capturing these larger features would be preferable.

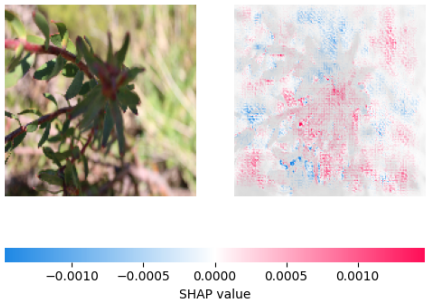
Mis-classified Classes Classes

Given the the F1-Score of *Erica cinerea* being 0.36 for Dataset A this class was deemed to be poorly performing but it also showed a high mis-classification rate in the confusion matrix. The precision of 0.5 and recall of 0.29 indicate that the model is correct 50%

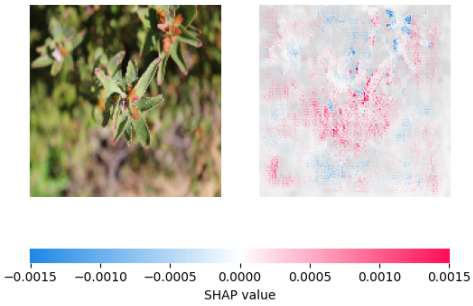
of the time when it predicts a positive outcome for this class and that it only identifies 29% of all the actual positive instances in the dataset. Based on the confusion matrix for Baseline A, it appears as though *Erica cinerea* and and *Erica Discolor* are often mis-classified. *Erica cinerea* in general appears to be mis-classified often. For these two classes The SHAP analysis for both classes is presented below:



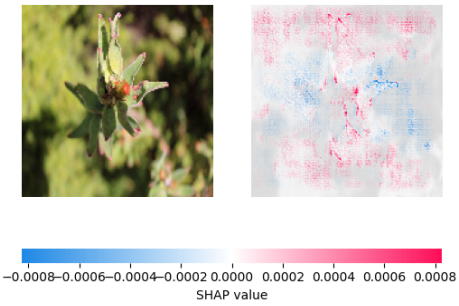
(a) *Erica cinerea* SHAP i



(b) *Erica cinerea* SHAP ii



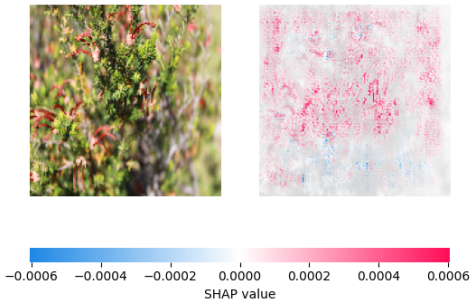
(c) *Erica cinerea* SHAP iii



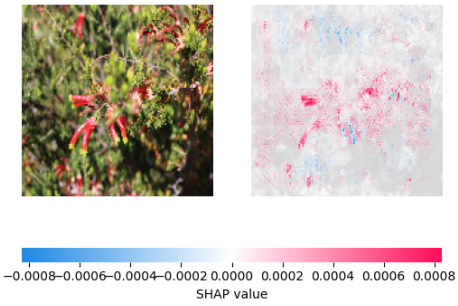
(d) *Erica cinerea* SHAP iv

Figure B.2: *Erica cinerea* leaves subset SHAP Values from Baseline A

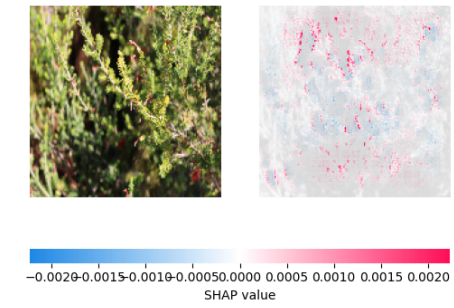
The problem can be clearly observed emerging in the subset of images presented in B.2. The leaves belonging to *Erica cinerea* were captured in shadowy conditions which is causing the model to focus on other features in the image. This can be clearly observed in SHAP i in B.2 where the model ignores the leaf and plant and learns background features as evidenced by the pink pixels. These background features are generic and poorly defined and appear to indicate clusters of leaves, which match the more distinctive features of *Erica discolor* shown in B.3.



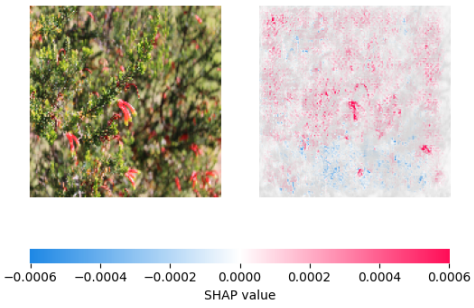
Erica discolor SHAP A



Erica discolor SHAP B



Erica discolor SHAP C

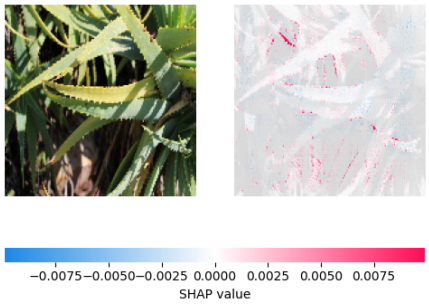


Erica discolor SHAP D

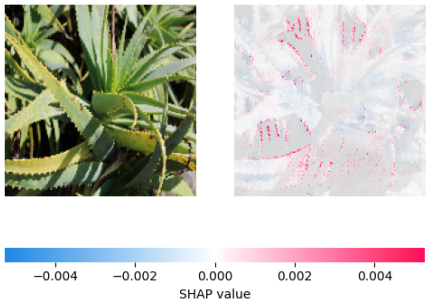
Figure B.3: *Erica discolor* leaves subset SHAP Values from Baseline A

The issue of the model training on background noise is addressed in 3.5 where classes that contain images with too much background are identified using the Sobel index and removed from the dataset. In Baseline B, this issue does not occur with just 1 mis-classification of these two classes, down from 5 in baseline A. The mis-classification problem is down to low data quality (in this cases the shadows cast on the leaves). It also shows that the brightness and colour of the plants are important features for CNNs.

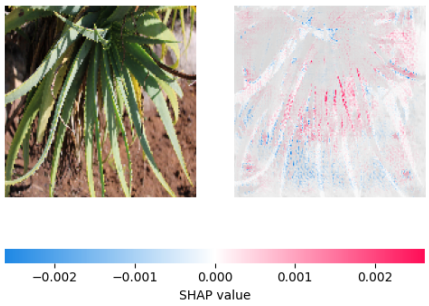
Poor Performing Classes Looking at classes with low F1-scores, *Aloe arborescens* and *Arista capitata* jump to view with F1-Scores of 0.12 and 0.14 respectively. These values indicate that the model is learning some basic features from these classes but is struggling to learn enough detail to make accurate predictions on unseen data. The SHAP analysis is shown below:



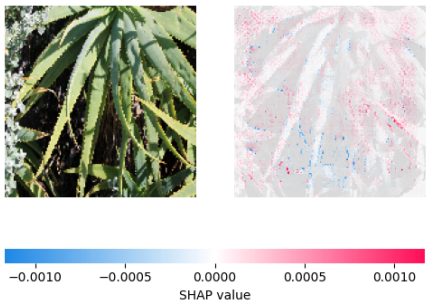
(a) *Aloe arborescens* SHAP i



(b) *Aloe arborescens* SHAP ii



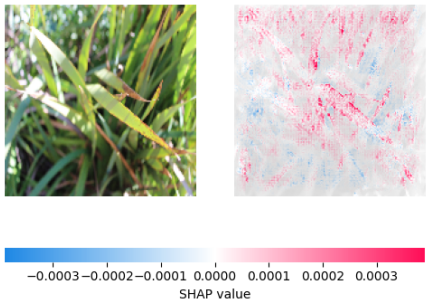
(c) *Aloe arborescens* SHAP iii



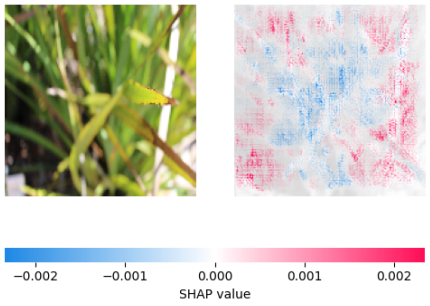
(d) *Aloe arborescens* SHAP iv

Figure B.4: *Aloe arborescens* leaves subset SHAP Values from Baseline A

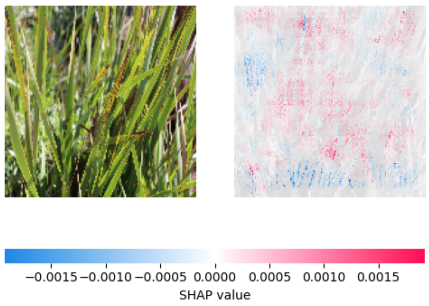
Aloe arborescens shown in B.4 is proving a challenge for the model to learn features from. On first glance, the large sclerophyllous leaves with jagged edges seem visually striking and distinctive. However, the SHAP analysis shows that the model is again focusing on background noise. The reasoning for this appears to be the variable lighting conditions caused by the leaves themselves. Figure iii in B.4 shows this problem which appears to be an issue with network depth. The CNN archtiecture from the Baseline is not able to construct complex enough shapes in the deepest layer. This suggests that for this class, the model architecture is inadequate as deeper networks will be able to pick up the leaf shapes instead of just small parts of them. More complex shapes could possibly also account for the shadows which will exist in natural images of thesis species regardless, simply due to the morphological features.



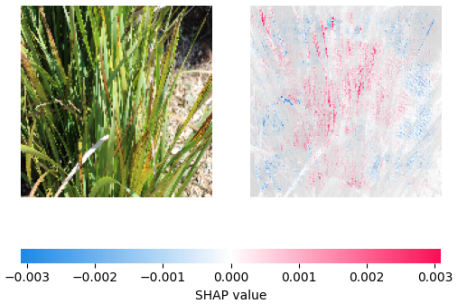
(a) *Arista capitata* SHAP i



(b) *Arista capitata* SHAP ii



(c) *Arista capitata* SHAP iii



(d) *Arista capitata* SHAP iv

Figure B.5: *Arista capitata* leaves subset SHAP Values from Baseline A

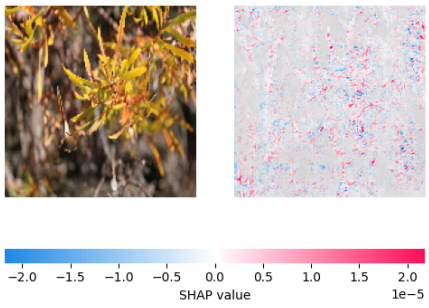
The SHAP analysis of *Arista capitata* shown in B.5 is a more complicated example. The SHAP plots show that the model is learning the general features of the class, apart from SHAP ii which seems to have learned background noise. Given the F-1 score from the classifciation report, it seems as though the model is over-generalising. While it is learning features, they do not appear to be as defined as they should be. This is also likely a problem of the model not being deep enough. It can make out the simpler patterns in the image, but is struggling to put them together into more complex shapes.

Transfer Model A

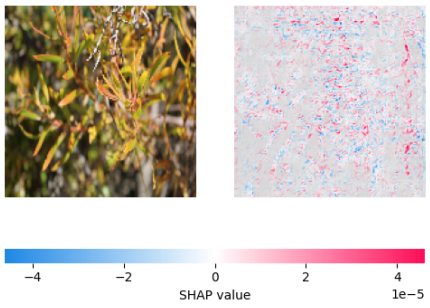
High Performing Species

Similarly to the Baseline Models, *Leucadendron salignum* is again a high performing class

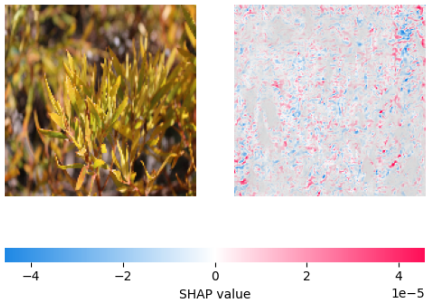
with an F1-score of 0.88 and perfect recall. This shows that the more advanced model is still comfortably learning features from this class and using them to make accuracte predictions. The SHAP plots in B.6 show a continuation of what was observed in the Baseline Models for this class. Despite the presence of background noise (in SHAP i for example), the positive SHAP values are congregated around the leaf shapes. The model is able to learn more complex features than the Baseline Models which focused on particular leaves. Colour still plays an important role in classification.



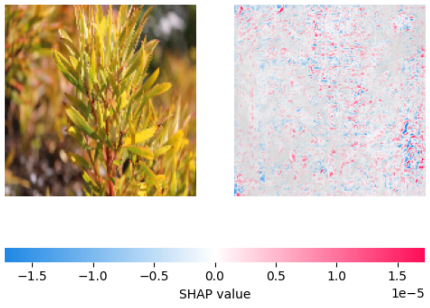
(a) *Leucadendron salignum* SHAP i



(b) *Leucadendron salignum* SHAP ii



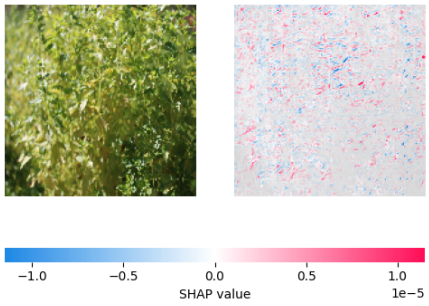
(c) *Leucadendron salignum* SHAP iii



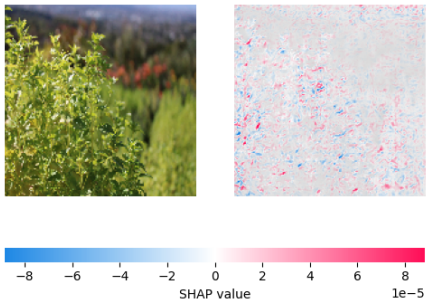
(d) *Leucadendron salignums* SHAP iv

Figure B.6: *Leucadendron salignum* leaves subset SHAP Values from Transfer Model A

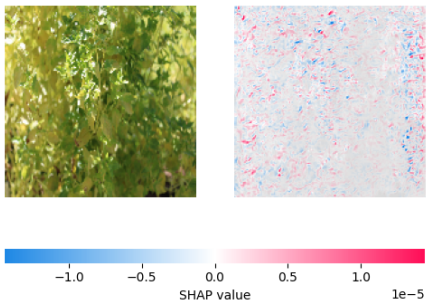
In Transfer Model A, *Protea repens* also performs fairly well with an F1-score of 0.78 and 8 True Positives in the confusion matrix. The SHAP plot in B.7 shows that the model is learning features centered around the leaf clusters and shapes, but the objects learned are not always clear. SHAP ii shows some background features contributing to the prediction, likely a result of Transfer Model A learning noise in the uncleaned dataset.



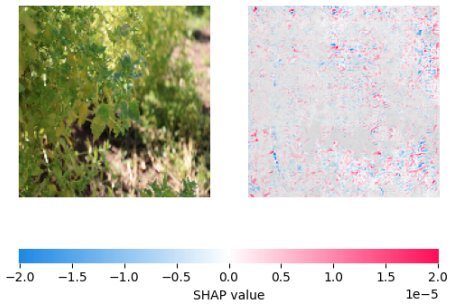
(a) *Protea repenss* SHAP i



(b) *Protea repenss* SHAP ii



(c) *Protea repenssm* SHAP iii

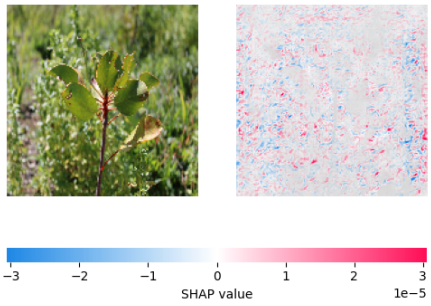


(d) *Protea repenss* SHAP iv

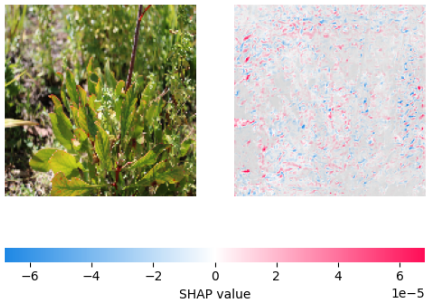
Figure B.7: *Protea repenssm* leaves subset SHAP Values from Transfer Model A

Mis-Classified Species

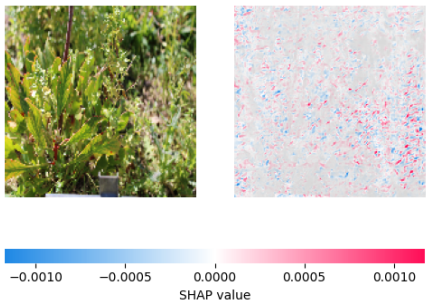
Protea cynaroides has an F1-score of 0.55 which is fairly average when compared across classes. However it is often mis-classified. The SHAP plots below reveal that there is significant noise in the images and the backgrounds distort the leaf patterns (as observed in SHAP ii and SHAP iii below). The model is learning irrelevant features from this class, which is causing mis-classifications in other classes such as *Cotyledon orbiculata*. The overall performance also speaks to the learnings from 2.1.2 where it was determined that different species often share physical similarities. Despite this, the model only struggles with a small group of classes with shared features.



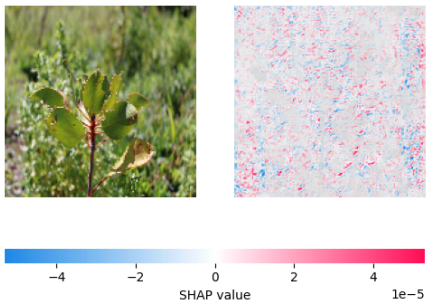
(a) *Protea cynaroides* SHAP i



(b) *Protea cynaroides* SHAP ii



(c) *Protea cynaroides* SHAP iii

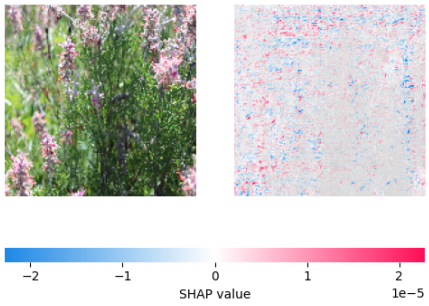


(d) *Protea cynaroides* SHAP iv

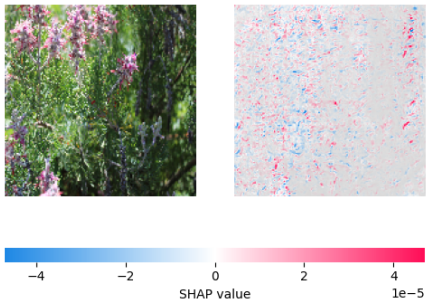
Figure B.8: *Protea cynaroides* leaves subset SHAP Values from Transfer Model A

Low Performing Species

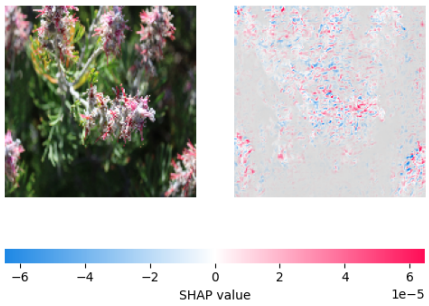
Grevillea banksii performed poorly in Baseline A and is now marginally improved with an F1-score of 0.36. This species is not Fynbos, nor is it endemic to the region. The model is unable to learn features in this class as they are not shared by the Fynbos. This class gets removed by the Data Quality Assessment and the SHAP plots below in B.9 show that the features learned by the model for this class are not being generalised well. The shapes are erratic and the model seems unable to genralise the flowers or stems (noticeable in SHAP iii and SHAP iv).



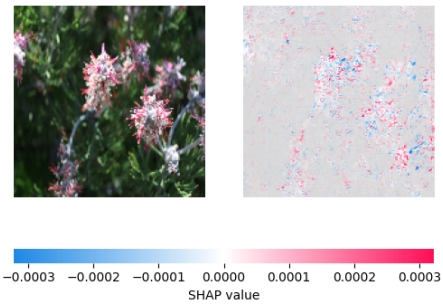
(a) *Grevillea banksii* SHAP i



(b) *Grevillea banksii* SHAP ii



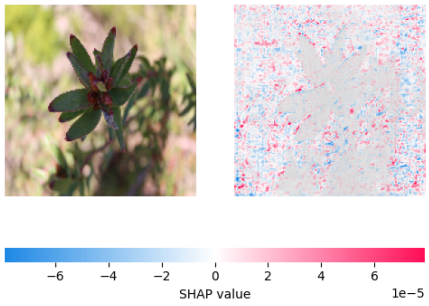
(c) *Grevillea banksiis* SHAP iii



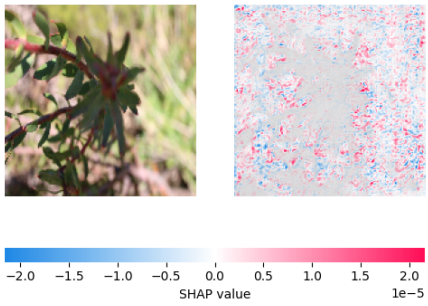
(d) *Grevillea banksii* SHAP iv

Figure B.9: *Grevillea banksiis* leaves subset SHAP Values from Transfer Model A

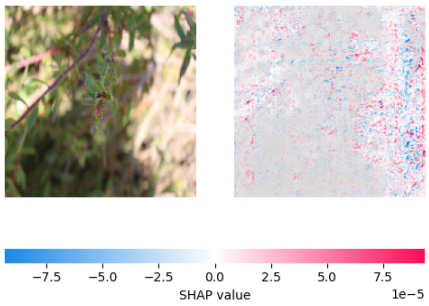
Eric cinerea had an F1 score of zero in both models. It was mis-classifed 5 times in Transfer Model A and 7 times in Transfer Model B. The reason behind this is evident from the SHAP analysis below in B.10. The shadows that the leaves cast on one another in the training data, distorts the learning. This is the same problem that was faced by Baseline A and is a function of Image Quality. The current Image Quality Assessment does not check for shadows or poor lighting. This problem also emerged for the class *Aloe*



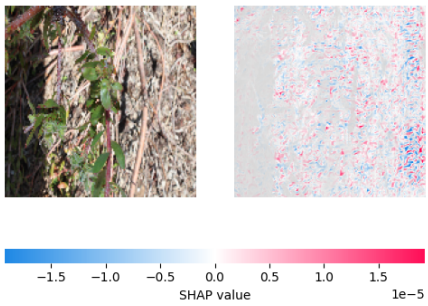
(a) *Erica cinerea* SHAP i



(b) *Erica cinerea* SHAP ii



(c) *Erica cinerea* SHAP iii



(d) *Erica cinerea* SHAP iv

Figure B.10: *Erica cinerea* leaves subset SHAP Values from Transfer Model A

SHAP i and SHAP iii are clear indications of the poor lighting conditions influence on the model. The SHAP plots show that the model actually ignores almost any pixels on the actual leaf and focuses on the brighter but less focused background.

Appendix C

Appendix C - Code Segments

```
    def assess_image_quality(image_path):
    image = imread(image_path)
    gray_image = rgb2gray(image)

    # Sharpness: Laplacian variance
    sharpness = cv2.Laplacian(image, cv2.CV_64F).var()

    # Noise: Standard deviation of greyscale image
    noise = np.std(gray_image)

    # Colour Balance: Standard deviation of color channels
    colour_balance = np.std(image, axis=(0, 1))

    # Background Consistency Using Sobel: Edge detection area
    edge_sobel = sobel(gray_image)
    background_consistency = np.mean(edge_sobel)

    return sharpness, noise, colour_balance, background_consistency

```

Listing C.1: Image Quality Assessment

Figure C.1: Python function for assessing image quality by measuring sharpness, noise, colour balance, and background consistency.

```
    # Define thresholds
    sharpness_threshold = (300, 600)  #  (low, high)
    noise_threshold = (0.01, 0.05)    #  (low, high)
    color_balance_threshold = (50, 100)  #  (low, high) for each channel
    background_consistency_threshold = (0.1, 0.3)  #  (low, high)
    resolution_threshold = (1000000, 2000000)  #  (low, high) in pixels

    def score_metric(value, low, high):
    if value < low:
    return 0
    elif value < high:
    return 1
    else:
    return 2

```

Listing C.2: Image Quality Assessment Thresholds

Figure C.2: Thresholds for image quality assessment metrics and a function to score each metric.

```
    model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D(2, 2),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),

    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),

    Flatten(),
    Dropout(0.5),

    Dense(128, activation='relu'),
    Dense(len(train_generator.class_indices), activation='softmax')
    ])

```

Listing C.3: CNN Model Architecture

Figure C.3: Convolutional Neural Network (CNN) model architecture used for classifying Fynbos leaves.

```
    # Load the base model, pre-trained on ImageNet
    base_model = ResNet50(weights='imagenet', include_top=False,
input_shape=(150, 150, 3))

    for layer in base_model.layers[-30:]:  # Unfreeze the last 30 layers
    layer.trainable = True

    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(512, activation='relu')(x)  # Adjusted number of neurons
    x = Dropout(0.3)(x)  # Adjusted dropout rate
    predictions = Dense(len(train_generator.class_indices), activation='
softmax')(x)
    model = Model(inputs=base_model.input, outputs=predictions)

```

Listing C.4: Transfer Model Architecture

Figure C.4: Transfer learning model architecture using ResNet50 pre-trained on ImageNet for classifying Fynbos leaves.

```
    def squeeze_excite_block(input, ratio=16):
    init = input
    channel_axis = -1
    filters = init.shape[channel_axis]
    se_shape = (1, 1, filters)

    se = GlobalAveragePooling2D()(init)
    se = Reshape(se_shape)(se)
    se = Dense(filters // ratio, activation='relu', use_bias=False)(se)
    se = Dense(filters, activation='sigmoid', use_bias=False)(se)

    x = Multiply()([init, se])
    return x

```

Listing C.5: Squeeze and Excite Block

Figure C.5: Implementation of the Squeeze and Excite block for enhancing feature maps.

```
    # Load the ResNet50 model
    input_tensor = Input(shape=(224, 224, 3))
    base_model = ResNet50(weights='imagenet', include_top=False,
    input_tensor=input_tensor)

    # Add Squeeze-and-Excite blocks with Batch Normalization
    x = base_model.output
    for layer in base_model.layers:
    if isinstance(layer, Conv2D):
    layer = BatchNormalization()(layer.output)
    layer = Activation('relu')(layer)
    x = squeeze_excite_block(layer)

    # Fine-tuning the last N layers
    N = 30  # Number of layers to unfreeze
    for layer in base_model.layers[-N:]:
    layer.trainable = True

    # Adding custom top layers
    x = GlobalAveragePooling2D()(x)
    x = Dense(1024, activation='relu')(x)
    x = Dropout(0.5)(x) # dropout set to 0.5
    x = Dense(512, activation='relu')(x)
    predictions = Dense(len(train_generator.class_indices), activation='
    softmax')(x)

```

Listing C.6: Optimized Model Architecture

```
    # Directory of the class to plot
    class_dir = '../Protea cynaroides'  # Replace with your class directory

    # Directory to save the SHAP plots
    save_dir = '../SHAP/Protea cynaroides'  # Replace with your desired
    save directory
    if not os.path.exists(save_dir):
    os.makedirs(save_dir)

    # Initialize SHAP GradientExplainer
    explainer = shap.GradientExplainer(model, np.zeros((1, 224, 224, 3)))

    # Iterate through each image in the specified class directory
    for img_file in os.listdir(class_dir):
    img_path = os.path.join(class_dir, img_file)
    preprocessed_image = load_preprocess_image(img_path, (224, 224))

    # Predict the class of the image
    predictions = model.predict(preprocessed_image)
    predicted_class = np.argmax(predictions, axis=1)[0]

    shap_values = explainer.shap_values(preprocessed_image)
    shap_values_for_predicted_class = shap_values[predicted_class][0]

    # Plot SHAP values
    plt.figure()
    shap.image_plot(shap_values_for_predicted_class, preprocessed_image[0],
    show=False)

    # Save the plot in the specified directory
    plt.savefig(os.path.join(save_dir, f'shap_output_{img_file}
    _protea_cynaroides.png'))
    plt.close()  # Close the plot to free memory

    print("All SHAP analysis plots saved.")

```

Listing C.7: SHAP Plot Code

```
import os
import cv2
import numpy as np
import pandas as pd
from skimage.io import imread
from skimage.color import rgb2gray
from skimage.filters import sobel
```

```
8
9  def assess_image_quality(image_path):
10 image = imread(image_path)
11 gray_image = rgb2gray(image)
12
13 # Sharpness: Laplacian variance
14 sharpness = cv2.Laplacian(image, cv2.CV_64F).var()
15
16 # Noise: Standard deviation of greyscale image
17 noise = np.std(gray_image)
18
19 # Colour Balance: Standard deviation of color channels
20 color_balance = np.std(image, axis=(0, 1))
21
22 # Background Consistency Using Sobel: Edge detection area
23 edge_sobel = sobel(gray_image)
24 background_consistency = np.mean(edge_sobel)
25
26 return sharpness, noise, color_balance, background_consistency
27
28 # Initialize a DataFrame to store the results
29 columns = ['Class_Name', 'Image_Path', 'Sharpness', 'Noise', '
    Color_Balance', 'Background_Consistency']
30 results_df = pd.DataFrame(columns=columns)
31
32 dataset_path = 'C:/Users/Jarushen/Desktop/Masters Thesis/Images/
    Fynbos_Dataset_A'
33 image_extensions = ['.png', '.jpg', '.jpeg', '.bmp', '.tiff', '.jfif']
34
35 for class_folder in os.listdir(dataset_path):
36 folder_path = os.path.join(dataset_path, class_folder)
37 if os.path.isdir(folder_path):  # Check if it's a directory
38 for image_name in os.listdir(folder_path):
39 if any(image_name.lower().endswith(ext) for ext in image_extensions):
40 image_path = os.path.join(folder_path, image_name)
41 sharpness, noise, color_balance, background_consistency =
    assess_image_quality(image_path)
42
43 # Append the results to the DataFrame
44 results_df = results_df.append({
45   'Class_Name': class_folder,
46   'Image_Path': image_path,
47   'Sharpness': sharpness,
48   'Noise': noise,
49   'Color_Balance': color_balance,
50   'Background_Consistency': background_consistency
51 }, ignore_index=True)
```

```
52
53 # Optionally, save the DataFrame to a CSV file
54 results_df.to_csv(r"C:\Users\Jarushen\Desktop\Masters Thesis\Report_Final
    \images\Results\Data_Quality_Assessment\image_quality_assessment_results
    .csv", index=False)
55
56 # Print the DataFrame
57 print(results_df)
```

Listing C.8: Image Quality Assessment Script