Selected Medicinal Plants Leaves Identification: A Computer Vision Approach

A thesis submitted in fulfillment of the requirements for a degree of

MASTER OF SCIENCE

in the

DEPARTMENT OF MATHEMATICS
RHODES UNIVERSITY

by

Avuya Deyi
Supervisor: Dr M. Atemkeng
Co-Supervisor: Dr A. Fadja, Dr X.S. Siwe Noundou

January 2023

Abstract

Identifying and classifying medicinal plants are valuable and essential skills during drug manufacturing because several active pharmaceutical ingredients (API) are sourced from medicinal plants. For many years, identifying and classifying medicinal plants have been exclusively done by experts in the domain, such as botanists, and herbarium curators. Recently, powerful computer vision technologies, using machine learning and deep convolutional neural networks, have been developed for classifying or identifying objects on images. A convolutional neural network is a deep learning architecture that outperforms previous advanced approaches in image classification and object detection based on its efficient features extraction on images.

In this thesis, we investigate different convolutional neural networks and machine learning algorithms for identifying and classifying leaves of three species of the genus *Brachylaena*. The three species considered are *Brachylaena discolor*, *Brachylaena ilicifolia* and *Brachylaena elliptica*. All three species are used medicinally by people in South Africa to treat diseases like diabetes. From 1259 labelled images of those plants species (at least 400 for each species) split into training, evaluation and test sets, we trained and evaluated different deep convolutional neural networks and machine learning models. The VGG model achieved the best results with 98.26% accuracy from cross-validation.

Keywords: Deep Learning, Convolutional Neural Networks, Machine Learning, Computer Vision, Medicinal Plants

Contents

List of Tables

List of Figures

List of Abbreviations

AI: Artificial Intelligence

ANN: Artificial Neural Networks

BP: Backprobagation

CNN: Convolutional Neural Network

DL: Deep Learning

GAN: Generative Adversarial Network

ML: Machine Learning

GD: Gradient Descent

KNN: K-Nearest Neighbour

SGD: Stochastic Gradient Descent

SVM: Support Vector Machine

CPC: Center for Plant Conservation

IUCN: International Union for Conservation of Nature

ROC: Receiver Operating Characteristics

AUC: Area Under the Curve

TP: True Positive

TN: True Negative

FN: False Negative

FP: False Positive

TPR: True Positive Rate

FPR: False Positive Rate

Acknowledgements

I thank all those who have helped and supported me to put the ideas in this thesis together. I would like to express my special gratitude to my supervisors Dr Marcellin Atemkeng, Dr Arnaud Nguembang Fadja, and Dr Xavier Siwe Noundou who worked with me tirelessly. I send gratitude to our previous and current HOD Prof. Denis Pollney and Dr Eric Andriantiana who gave me the opportunity to do this research. Any attempt at any level could not be satisfactorily completed without the support and guidance of Dr Eleonora Goosen who helped a great deal. My family has always trusted and supported me without exerting too much pressure on me to complete my studies; I say thank you!

Chapter 1

Introduction

1.1 Study Background

Organizations such as the International Union for Conservation of Nature (IUCN) (Luca, 2021) and the Center for Plant Conservation (CPC) (Durrington, 2019) are engaged in protecting endangered species. Forestry services, pharmaceutical laboratories, physicians, botanists, and traditional healers benefit significantly in their knowledge of identifying plant species (Begue et al., 2017). The identification of medicinal plants is critical and needs attention as misidentification can affect the consumers' health (Dileep & Pournami, 2019). Due to the effects of misidentification, experts like botanists are consulted for plant species identification and classification. This represents a huge issue to chemists, biochemists, microbiologists, and other researchers who do not have the skills and expertise for plant taxonomy. Therefore, there is a need to find or discover new methodologies of plant species identification and classification (Aguate et al., 2020).

There are many foodstuff and medicines made from plant species (Wagle et al., 2022), while some plant species are at risk of extinction. Some species are in demand and need to be used daily by people who are not experts in identifying plant species. Most plant identification and classification methods use features such as flowers, leaves, shape, color, and size (Lee et al., 2017). However, some of these features such as the shape change drastically as a plant grows. Furthermore, some features, such as flowers, only exist for a short lifetime. Leaves maintain the color, shape, and number of veins throughout a plant's lifetime and are easily accessible, which explains why most research uses them for classification (Wäldchen et al., 2018). One of the traditional ways to identify plants is visual observation (Wagle et al., 2022). According to Wäldchen et al. (2018) and Wagle et al. (2022) this method is exhaustive, costly, and time-consuming. Liu et al. (2017) use metabolite content to classify plants. A metabolite is a bioactive substance formed during the process of metabolism. It is different from each plant species (Liu et al., 2017). Lately, Artificial intelligence (AI) (Winston, 1984) methods have shown promising results of plant identification and classification.

AI is a field of science and technology that seeks to emulate the abilities of human beings in performing specific tasks (Winston, 1992). Winston (1984) says AI builds machines and computer systems that can perform functions commonly performed by humankind. The subfield of AI, machine learning (ML) (Balyen & Peto, 2019) and its subset known as deep learning (DL) (Balyen & Peto, 2019) have since received massive attention in the past decade. ML specialises in training machines to execute tasks typically carried out by humans (Mitchell et al., 2007). ML algorithms are constructed using existing training data, incorporating sets of procedures and mathematical logic statements, while DL is a subfield of ML inspired by the human brain (Martens et al., 2010) and known for its capability to deal with enormous large datasets (Erhan et al., 2010). ML and DL have been used successfully in diverse domains such as computer vision, biology, medical sciences, spacecraft engineering, finance, and entertainment (El Naqa & Murphy, 2015). Computer vision is a field of AI that is concerned about extracting information from images and videos (Jarvis, 1983). Hosch (2019) defines ML as a subfield of AI that uses mathematical and statistical algorithms to build models that learn from data and experience. Lately, DL is proving to be highly efficient in image classification (Balyen & Peto, 2019). DL models are built based on artificial neural networks (ANNs) and are made up of multiple layers connected with weights (Lee et al., 2020). The structure of an ANN is inspired by how the human brain works (Agatonovic-Kustrin & Beresford, 2000). DL models are classified as most successful in computer vision (Gallagher et al., 2020).

One of the most critical steps when building ML or DL models is to evaluate the capability of the employed algorithms (Sheik et al., 2007). Evaluation metrics are designed to explain the performance of a model and discriminate among the model results. The most used metrics are confusion matrix, AUC-ROC curves, root mean square error (RMSE), gin coefficient, F1 score, and cross-validation. These metrics provide feedback that is then used to improve the model's performance (Guo et al., 2008).

There has been rapid growth in the domain of computer vision since the breakthrough of applying DL to computer vision in the competition of ImageNet (Krizhevsky et al., 2012). In the paper by Krizhevsky et al. (2012), Neural Information Processing Systems (NIPS) 2012 outperformed the traditional models of ML by a margin of 10% (Krizhevsky et al., 2012). DL models have been used in image plant classification, and have shown success (Lee et al., 2017). AI algorithms, especially convolution neural networks (CNN) (Lee et al., 2020), have made computer vision extremely powerful than ever. Ghosal & Sarkar (2020) modelled a CNN architecture based on VGG-16 and obtained 92.46% accuracy. InceptionV3, Inception-ResNetV2, MobileNetV2, and EfficientNetB0 are built for 38 classes of diseases found in 14 different plant species (Hassan et al., 2021). These models returned acceptable results ranging from 97.02% to 99.56%. Nine layered deep CNN model was trained to classify 39 different classes of plant diseases, and it returned a classification accuracy of 96.46% (Geetharamani & Pandian, 2019). Pawara et al. (2017) emphasize using large datasets to train CNN models

since objects in real life appear in various forms. This research considers data augmentation techniques to expand the dataset. Data augmentation produces transformed images from the original dataset. As much as DL and CNN models proved to be very good when dealing with large complex datasets, traditional ML models still work, and in some cases such as having a small dataset or linearly related dataset, they proved to be better than DL models. Using traditional ML models for less complex datasets is recommended, since they are less computationally expensive and can run faster.

1.1.1 Problem Statement

Three species of the genus *Brachylaena* grow in the Makhanda district municipality. They are named *Brachylaena discolor*, *Brachylaena elliptica*, and *Brachylaena ilicifolia*. These medicinal plants are extensively used by chemists and botanists researchers around the area to extract chemicals and understand the make-up of the plants (Wagle et al., 2022). Locals also use the same medicinal plants to treat diseases like diabetes and renal conditions (Swain et al., 2012). All these people rely on experts with extensive knowledge to identify and classify these species. Their reliance on experts sometimes delays their projects (Machhour et al., 2020). This thesis provides the need to build an AI model for the three *Brachylaena* species classification and identification.

1.1.2 Objective and Contribution

In this thesis, we investigate the best model to classify selected medicinal plants. Two traditional ML models and two DL models are experimented. This work has the following contributions:

- Three medicinal plants belonging to the Brachylaena genus are investigated and explained thoroughly.
- The current work on classifying and identifying plants using AI is reviewed.
- It reviewed ML algorithms and DL models for image classification and identification.
- The results of the models explored provided acceptable results, with the VGG-16 models returning an accuracy of 98.26%.

1.1.3 Thesis Organisation

Chapter 1 discusses the necessity of using AI in different fields of science. This chapter then narrows this discussion to medicinal plant classification and identification using traditional ways and the late evolution to AI. The chapter briefly reviews the literature on plant image

classification and identification using traditional ML and DL models. The latest research review suggests that CNNs master the art of feature extraction for imagery data.

Chapter 2 gives a detailed review of related work to this thesis. Many researchers have used ML algorithms such as support Vector machines (SVM), K-nearest neighbours (KNN), and decision trees for image identification, but recently DL models, specifically CNNs, have been the most used. The chapter discusses the challenges of using AI, particularly ML and DL, for image identification. A detailed review of the three Brachylaena species is presented.

Chapter 3 details the most important concepts of this research, it has six subsections in this order: computer vision, supervised ML, classification, ML algorithms, DL and CNN, and data augmentation techniques.

Chapter 4 presents the experiments performed and the results obtained.

Chapter 5 concludes the thesis.

Chapter 2

Related Literature and Dataset

2.1 Related Work

Computer vision is increasingly becoming the most reliable solution to plant classification and identification as different types of powerful technologies are built for image capturing (Wäldchen & Mäder, 2018). Van Hieu & Hien (2020) conducted a study to classify 12000 plant species in Vietnam, and they used MobileNetV2, VGG16, ResnetV2, Inception ResnetV models. MobileNetV2 registered a relatively acceptable accuracy of 83.9% best performance with support vector machines (SVM) classifier out of the four models evaluated.

A recent study by Huixian (2020) uses four deep learning (DL) models to learn plant leaf features and classify species based on images in a dataset of 7 different species: KNN-based neighbourhood classification, a self-organising feature mapping algorithm known as Kohene network, backpropagation (BP) neural network, and SVM are trained and compared on a database of close to 200 images. The BP network returns the highest recognition accuracy of 92.48%, followed by the Kohonen classifier that obtained 86.78%.

In a study that sought to identify and classify plants from the BJFU100 dataset, an accuracy of 91.78% is obtained. The dataset has 10000 images. A 26 layered ResNet DL model proved to be acceptable with the BJFU100 dataset (Sun et al., 2017). The study uses the basic structure of ResNet and attempts to find the best architecture. These results are outstanding considering that the dataset has 100 classes.

In a study very similar to this thesis, Kho et al. (2017) build a model to classify three different types of *Ficus* species. The study is similar because it tries to build a model to classify or identify different plants of the same family. They have a relatively small dataset of 54 images. The paper evaluates the performances of ANNs and SVMs. It is found that the two models retain similar accuracies of 83.3% each. ANNs are known for improving with an increase in the training dataset (Kho et al., 2017). It is expected that with more data, the model will improve.

Jeon & Rhee (2017) used GoogleNet model for plant leaves' classification. It achieved approximately 94% accuracy. They described GoogleNet as a CNN model that extracts and learns feature points. GoogLeNet is famous for winning the 2014 competition where it was trained from a database of 1.2 million images (Szegedy et al., 2015). The model returned the smallest loss of 6%.

As these techniques of ML and DL become popular and successful, more ideas come up to solve different or similar real-life problems. The agricultural sector is among the sectors that have benefited from the positive developments in the AI industry. Several studies reveal that ML methods can be used to detect unhealthy plant leaves. Most recently, a nine-layer CNN model was developed to detect and classify infected plant leaves (Geetharamani & Pandian, 2019). As experimented by Geetharamani & Pandian (2019), the CNN model competed with SVM, logistic regression, decision tree, and KNN. It achieved the highest performance of 96.46%. This study supports that CNN models excel when there is a larger set of data. They used $556,363,900$ images for training and validation. Li et al. (2020) proposed a combination of shallow CNN and SVM classification algorithms as a good attempt for plant disease identification. They suggested that shallow CNN must be considered first before going for the deep CNN models. These shallow models provide simpler structures and less computational costs. They struggle with complex datasets and big data.

Most of the studies we reviewed, experimented using CNN, KNN, ANN, and/ or SVM for plant classification problems. CNN architectures comprise layers and millions of parameters (Pawara et al., 2017). Pawara et al. (2017) used CNN models, AlexNet and GoogleNet for plant classification. AlexNet is an eight-layered neural network where five layers are convolutional, three layers are pooling, and two fully-connected layers. GoogleNet architecture is deeper than that of AlexNet architecture. It introduces a new module known as inception (Shin et al., 2016). Unlike AlexNet, GoogleNet has fewer convolutional and pooling layers, two for both. GoogleNet also has nine inception layers. The role of the inception module is to summarise filters with different dimensions and sizes to one filter (Shin et al., 2016). Contrary to the CNN models, KNN is a non-parametric model. It has a simple structure but still effective in many cases (Guo et al., 2003). KNN is a supervised versatile ML algorithm. The algorithm considers KNNs to predict the class or the next value for the new input (Guo et al., 2003).

ANN is a feed-forward neural network since the learning process only in the forward direction in the network through the input nodes. It imitates how the brain of a human being functions (Bala & Kumar, 2017). Bala & Kumar (2017) further say that ANN is a group of multiple perceptrons or neurons at each layer. Contrary to CNN, ANN has three primary layers. The layers are listed as input, hidden, and output layers. Features are received in the input layer. It is then processed by learning weights in the hidden layer, and the output layer produces results. When ANNs solve an image identification problem, they transform image data from a 2-dimensional shape to a 1-dimensional vector. For this reason, ANN loses the ability to

capture spatial features. Contrary to ANNs, CNNs use image data without transforming it.

One of the commonly used ML algorithms for classification is the SVM. Yang et al. (2010) describe SVM as an algorithm that intends to generate a hyperplane that divides different classes in higher dimensional spaces. SVM seeks to generates optimal hyperplane in a repetitive manner, which reduces an error as much as possible. The main idea of SVM is to find an optimal hyperplane that best divides the dataset into different categories. One of its drawbacks is sensitiveness towards outliers or noise from the input data (Yang et al., 2010). SVM uses clustering to group common data points in an unlabelled dataset, and then hyperplane can be constructed.

Most studies that we have come across in this literature review suggest that deep CNNs are the future in solving complex image classification and identification problems. However, the traditional ML models still work perfectly in some cases where there is an insufficient dataset. A specific model is employed depending on the nature of a dataset and the problem. In this research, we are classifying three image leaves of the genus *Brachylaena*. Two traditional ML and DL models are experimented with and compared against each other. The ML models are SVM and decision trees, and the DL models are AlexNet and VGG-16.

2.2 Challenges of ML or DL Classification

Although the performance of ML and DL models are acceptable in image classification problems, challenges still impact model's performance. Some of the challenges tend to badly impact the very same life of human beings that they are trying to advance. Privacy and security are some of the many challenges of DL and ML (Liu et al., 2020).

2.2.1 Security and Privacy Concerns

ML and DL models have shown potential capabilities in solving several challenging problems, but recently they have displayed some security and privacy vulnerabilities (Xue et al., 2020). Xue et al. (2020) categorizes these vulnerabilities into five classes: "training set poisoning; adversarial example attacks; backdoors in the training set; model theft; recovery of sensitive training data." In training set poisoning, training data is maliciously manipulated to mislead the training process (Vorobeychik & Kantarcioglu, 2018). For example, a model used for malware detection can be fooled on the training set by swapping labels. Liu et al. (2020) published an article titled "Deep Learning: The Good, the Bad, and the Ugly." The article mentions the issue of misuse of sensitive data by the companies such as Facebook, Twitter, and Amazon, that host personal information for their customers. User private data can be mishandled if it lands in the wrong hands.

2.2.2 Insufficient Data

DL models need as much as possible training data to perform (Kim & Cho, 2021). DL and ML models adjust their weight for every input they receive because they want to capture most of the significant features of a particular class. However, the reality is that we can capture finite data with our systems (Marcus, 2018). Marcus (2018) emphasises that if we feed DL models with unlimited data and use powerful computing resources, we can build close to perfect models. Leung et al. (2019) mentions an example of self-driving cars which struggle driving at night because most of the dataset they learn from it is collected during the day. Though this issue of insufficient data is a torn when building ML and DL models, there are proposed solutions, but they also rely on existing datasets, see Section 3.6 for an intensive discussion.

2.2.3 Selective Bias

ML and DL models are not going to easily escape bias; they are built based on bias. When solving a problem there are too many models to choose from. Choosing one is based on the objectives, and bias plays a huge role. Choosing the best hyper-parameters requires bias, whether using random search or grid-search. The random search is biased because when it returns parameters that do not perform as expected on the dataset, it is executed again until it returns acceptable results. The dataset that is fed to the models to train is collected by people, sometimes it does not represent the entire domain of the source data.

2.2.4 Opacity of DL models

DL models are a black box, and sometimes it cannot be known how they have arrived to certain decisions. Sometimes these models easily execute a challenging task. For example, they would classify millions of images of 100 classes within 5 minutes, but a human being would take days to do so. Also, the world champion of chess is a DL model (Dickson, 2018). DL models diffuse the information in a way that is exceedingly hard to decipher, instead of storing the information in a friendly format for humans to learn (Castelvecchi, 2016). These DL models lack transparency. They are employed in many domains, and some domains require transparency, for example: deciding on a legal case. This issue threatens the growth of AI because the engineers too cannot explain some decisions taken by DL models. However, lately, researchers are campaigning for explainable AI, which targets to minimize this problem in the future (Xu et al., 2019; Gade et al., 2019; Holzinger et al., 2017; Hoffman et al., 2018; Rai, 2020).

2.2.5 Shallowness of DL and ML models

DL models are exceptional at mapping inputs and outputs, but they are not good at understanding the context of the dataset they are handling (Dickson, 2018). Ghozia et al. (2020) agreed with Dickson (2018) that DL algorithms capture patterns from data very well but cannot relate to the meaning. These models lack context awareness, and they cannot extract emotions and feelings from the data (Ghozia et al., 2020). Dickson (2018) further says that DL algorithms cannot naturally abstract concepts such as justice and democracy. However, there is an ongoing research to embed politics and laws in the AI models (Djeffal, 2019; Nandutu et al., 2021). The other issue with DL models they struggle when dealing with outliers (Marcus, 2018), a model built to recognise a human being would find it hard to recognise someone who has been badly injured such that they lost nose, arms, legs and eyes. A human being is likely to recognise that person from the first attempt.

2.3 Dataset: Selected Medicinal Plants

The genus *Brachylaena* consists of eight species. Three, namely *Brachylaena discolor* DC. (*B. discolor*), *Brachylaena elliptica* (Thunb.) Less (*B. elliptica*), and *Brachylaena ilicifolia* (Lam.) Phillips & Schweick (*B. ilicifolia*) grow in the Makhanda district municipality of the Eastern Cape Province of South Africa (Moll, 1983). This work seeks to classify and identify the three *Brachylaena* species that grow in Makhanda using ML and DL algorithms. Below we discuss the three types of *Brachylaena*.

2.3.1 The *Bracylaena discolor* DC.

Alternative names for this medicinal plant are wild silver oak (English) (Moll, 1983), wildevaalbos (Afrikaans)(Moll, 1983), umgqeba (isiXhosa) (Cocks & Dold, 2006), umpahla (isiXhosa (Dold & Cocks, 1999) and isiZulu (Venter & J, 2007)), Skead; umpatha (isiXhosa) Dold & Cocks (1999), and isiduli (isiXhosa) (Dold & Cocks, 1999), Bantu Cancer Registry Herbarium BCRH 1112 (Dold & Cocks, 1999) and mphahla (Northern Sotho) (Venter & J, 2007). It has been used by both African and European in South Africa to treat diabetes and renal conditions (Watt & Breyer-Brandwijk, 1962). Dutch settlers in the region used the ashes to make soap. The AmaZulu use it to treat intestinal parasites such as roundworms. The timber is used for wagon building, boat timber, fencing posts and pick handles (Watt & Breyer-Brandwijk, 1962). It is found as an evergreen shrub or a small tree, between 4 and 8 meters high on the margins of evergreen forests and in coastal woodland or bushes. The leaves are lanceolate to elliptic between 3.5 and 11.5 cm long. They are dark green on the top and pale white/grey with dense hairs at the bottom. The margins are irregularly or

obscurely jaggedly toothed (Moll, 1983). *B. discolor* leaves were collected at the Sunnyside Garden Centre in Cromwell street, $(-33.3170600°, 26.5353751°)$, Makhanda on 2 March 2020 and 15 October 2021. A voucher specimen, EDG29112021, has been deposited at the Selmar Schonland Herbarium, Albany Museum, Somerset street Makhanda, Eastern Cape, South Africa. See Figure 2.1.

Figure 2.1: *Bracylaena discolor*

2.3.2 The *Brachylaena elliptica* (Thunb.) DC.

Alternative names for this plant are bitter leaf (English) (Moll, 1983), bitterblaar (Afrikaans) (Moll, 1983) isiduli (isiXhosa) (Dold & Cocks, 1999) Skead; isagqeba, (isiXhosa) (Dold & Cocks, 1999). This plant is used medicinally by the amaZulu, amaXhosa, and people of European descent. It is known to treat diabetes successfully, but no clinical proof has been found by controlled observations (Dold & Cocks, 1999). The bitterness of *B. elliptica* has been ascribed to the presence of glucosides (Dold & Cocks, 1999). The AmaZulu use an infusion of decorticated roots to treat patients with breathing difficulties or as an emetic for side pain. An enema of a leaf infusion is a remedy for backache and biliousness. Wild animals often eat the leaves (Watt & Breyer-Brandwijk, 1962). It is a small shrub or tree that grows up to 4 m in height. It grows at the margins of evergreen forests, in semikarroid areas or coastal shrub (Moll, 1983). The leaves are evergreen, elliptic to ovate and lanceolate between 2 to 11 cm long and 0.5 to 3 cm wide. They are dark green above with sparse hairs at times. White felted hairs are present at the bottom. The margins are usually irregularly toothed, and they are often, but not always, two lobes near the apex of the leaf, creating the appearance of three lobes, as shown in Figure 2.2. The leaves from the same plant had two lobes near the apex, and others had no lobes near the apex. *B. elliptica* leaves were collected in Gowie's kloof $(-33, 293142°, 26, 512950°)$, Makhanda, Eastern Cape, South Africa, on 15

October 2021. Voucher specimen number EDG15102021, has been deposited at the Selmar Schonland Herbarium, Albany Museum, Somerset street Makhanda, Eastern Cape, South Africa.

Figure 2.2: *Brachylaena elliptica*

2.3.3 *The Brachylaena ilicifolia* (Lam.)

Alternative names for this plant are small bitter leaf (English) (Moll, 1983), fynbitterblaar (Afrikaans) (Moll, 1983) umgqeba (isiXhosa) (Cocks & Dold, 2006), and isiduli (isiXhosa) (Dold & Cocks, 1999).

B. ilicifolia is one of the 60 most traded medicinal plants (amayeza) in the Eastern Cape (Cocks & Dold, 2006). It is used in treating diabetes that had not been observed up to 1962 Watt & Breyer-Brandwijk (1962). Later publications confirmed the use of a leaf infusion to cure diabetes (Moll, 1983). Coughs, sore throat and asthma are treated by oral administration of a leaf infusion, and pimples of the mouth are treated by gargling with the infusion. Similarly, sheep with paratyphoid is treated with an infusion (Cocks & Dold, 2006). It is a small shrub or tree that grows up to 4 m in height (Venter & J, 2007). It grows in bush, on rocky hillsides and in scrub forest (Moll, 1983). The leaves are small, narrow and oblong. They are lanceolate to ovate 1 to 4.5 cm long and 0.2 to 1 cm wide. They are green on top without hairs and have pail green-white laves at the bottom. The entire margin has small teeth *B. ilicifolia* (Moll, 1983) leaves were collected on the Committee's Drift road $(-33,2306°, 26,2409°)$ in Makhanda Municipal district on 2 March 2020 and 15 October 2021. Voucher specimen numbers Carli Weyers Col no 1 and 2 at the Selmar Schonland Herbarium, Albany Museum, Somerset street Makhanda, Eastern Cape, South Africa.

Figure 2.3: *Brachylaena ilicifolia*

2.4 Dataset collection

When comparing the descriptions of the leaves and the photos above, it might not always be clear to which species a leaf might belong. The main aim of this project is to use AI to distinguish between the different species. We harvested the leaves of the three *Brachylaena species* from branches and placed them in labelled plastic bags. The plastic bags were marked with the relevant name of each species, namely *B. discolor*, *B. ilicifolia*, and *B. elliptica*. Figure 2.4 shows the leaves of the three *Brachylaena species*.

Figure 2.4: The three *Brachylaena species*

The size of a dataset is essential when building DL or ML models. Because the images were captured in a controlled environment, each leaf image was captured with a black background. An estimate of roughly 1000 images would be enough to train our models. We started to take pictures of each leaf on the second day using a Canon PowerShot SX610 HS Point and Shoot camera. We took 1259 images and is enough for training a successful models since they have good quality , where 401 images belong to *B. discolor*, 437 to *B. elliptica* and 421 to *B. ilicifolia*. Roughly half of our images are of the tops of the leaves. The other half of the images are of the bottoms of the leaves.

2.5 Conclusion

The purpose of this review was to lay a background on the existing literature of ML and DL models in solving image classification and identification problems. From this review, it is clear that CNNs are the art of image feature extraction. Most papers reviewed suggest that CNNs excel in computer vision problems. The review also aimed at understanding the importance of the three Brachylaena species. According to this review, Brachylaena species are important medicinal plants that are a source of many remedies to diseases such as sore throat, diabetes, intestinal parasites and asthma. The literature also suggest that Brachylaena plants are not limited to medicinal use as they have been used for producing soap, and they are also used for timber, building wagons, and fencing posts. The next chapter, Chapter 3, digs deep on the aspects of computer vision and the models used for this work.

Chapter 3

Computer Vision

In the past twenty years, computer vision has been among the most researched topics of DL and ML (Boryshchak, 2020). The research places a major focus on replicating or doing better than the capabilities of the human eye. Computer vision is the ability of computers to extract information from digital data such as images, videos, and other visual digital outputs (Meer, 2004). A video is an image in motion. In this chapter, the key concepts about computer vision on images, are described. This chapter explains in detail the ML and DL models used for this research. It also details some of the techniques used to evaluate the models.

3.1 Digital Image

In this section, a mathematical description of an image is given. The main features of an image are points, contrast, color, and edges. Very tiny dots that make up an image are called pixels. Let $\mathbf{x}_{i,j,k} \in \mathbb{R}^{r \times c \times 3}$ be a tensor that defines an image with three channels, where i, j and k runs from 1 to the number of rows r, number of columns c and number of channels 3, respectively.

3.1.1 Image Contrast

Contrast is defined as a variance in luminance or color that makes an object differentiable, and it carries information about the brightness of an image. Computer vision experts quantify contrast in many ways, Stone (2019) explains a commonly used technique for calculating image contrast. They looked at a standard deviation of the pixels in a region or the entire image. In their work, the contrast of an image is defined as:

$$C = \sqrt{\frac{1}{3rc} \sum_{i=1}^{r} \sum_{j=1}^{c} \sum_{k=1}^{3} (\mathbf{x}_{i,j,k} - \overline{\mathbf{x}})^2},$$

where $\overline{\mathbf{x}}$ is the mean of the intensity of the complete image defined as:

$$\overline{\mathbf{x}} = \frac{1}{3rc} \sum_{i=1}^{r} \sum_{j=1}^{c} \sum_{k=1}^{3} \mathbf{x}_{i,j,k}.$$

Large datasets with huge variances in-between data points might delay or confuse the model's training process. Data normalization is a solution to that problem. It produces new values that maintain the general distribution and ratios from the original dataset but are re-scaled to be in the range of $(0, 1)$. A well known normalization process is given by:

$$C_{i,j,k} = \frac{\mathbf{x}_{i,j,k} - \overline{\mathbf{x}}}{\sqrt{\epsilon + \frac{1}{3rc} \sum_{i=1}^{r} \sum_{j=1}^{c} \sum_{k=1}^{3} (\mathbf{x}_{i,j,k} - \overline{\mathbf{x}})^2}},$$

where ϵ is an insignificant constant added to avoid dividing by zero.

3.1.2 Edge Detection

Edge detection is among the highly weighted features in image classification, and identification (Lee et al., 1987). Although this feature is significant for computer vision since it gives a general shape of an object being examined, it is hard to capture from an image (Faugeras & FAUGERAS, 1993). One idea to detect edges would be to detect some sort of discontinuity from the image intensity function. However, this sometimes is not the best way due to difficulties with noise measurement close to and on edges. There are many techniques for edge detection. Bhardwaj & Mittal (2012) conducted a comprehensive study to compare different edge detectors. They start by discussing Roberts, Sobel and Prewitt edge detectors (Chaple et al., 2015; Cherri & Karim, 1989; Hoang & Nguyen, 2018). They then focus on Laplacian of Gaussian, Canny, and declivity edge detector.

For the purpose of this study which does not focus on edge detection, we discuss one edge detector, Color Edge Detection Using Euclidean Distance and Vector Angle as researched by (Nadernejad et al., 2008). Most edge detectors are designed for greyscale images (one channel). That makes them computational less expensive but reduces the ability to detect edges as they omit an essential feature: high color variation and low-intensity variation near edges that can be taken advantage of. The two main operators employed by Nadernejad et al. (2008) are euclidean distance and vector angle. Let $\mathbf{v}_1$ and $\mathbf{v}_2$ be RGB triplets such that the Euclidean Distance that separates two pixels is given by:

$$D(\mathbf{v}_1, \mathbf{v}_2) = ||\mathbf{v}_1 - \mathbf{v}_2||.$$

To determine the vector angle between any two pixels, we have:

$$\sin\theta = \left[1 - \left(\frac{\mathbf{v}_1^T\mathbf{v}_2}{||\mathbf{v}_1||.||\mathbf{v}_2||}\right)^2\right]^{\frac{1}{2}} (Nadernejadet\ al.,\ 2008).$$

The saturation based method is used to combine the Euclidean Distance and Vector Angle between two pixels:

$$C_{GV} = \rho(k_1, k_2)\sin\theta + (1 - \rho(k_1, k_2))||\mathbf{v}_1 - \mathbf{v}_2||,$$

$$\text{where} \quad \rho(k_1, k_2) = \sqrt{\alpha(k_1)\alpha(k_2)},$$

and k_1 and k_2 are the saturation values of each pixels. The sigmoid function $\alpha(k)$ uses "$slope$" and "$offset$" values which can be obtained experimentally.

$$\alpha(k) = \frac{1}{1 + e^{-slope(k-offset)}}$$

This method of detection takes a 3×3 of pixels encircling each pixel in the image. It then calculates the saturation-based combination of the euclidean distance and vector angle separating the center pixel and each of the eight pixels around it. The center pixel gets assigned the highest value. A threshold is determined, and run each pixel's results through to spot the edges and false edges. This is one of the many edge detectors in image classification and identification. Many image features are used in image classification, but we discuss the most commonly used ones. The models used for this project automatically capture the image features through the distribution of pixel values.

3.2 Supervised Machine Learning

ML has three main paradigms that inform a considerable part of building models. Those paradigms consist of supervised learning, unsupervised learning, and reinforcement learning, but not limited to only these. The main characteristic of supervised learning is that it uses annotated data for training (Cunningham et al., 2008). In this project, we are using a labelled dataset, therefore supervised ML models are used. Given a sample space X and Y, supervised learning seeks to determine the relationship $f : \mathbf{x} \rightarrow \mathbf{y}$ such that $\mathbf{D} = ((\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)) \in (\mathbf{x} \times \mathbf{y})^n$ is a dataset, where $\mathbf{x}_i$ is a vector and y_i is a scalar. The fundamental concept in training an ML model is minimising the error or loss that separates the predicted observation $f(\mathbf{x}_i)$ and the ground truth $y_i \in \mathbf{y}$.

Loss functions are a significant building block of ML and DL models. These functions use the predicted $f(\mathbf{x}_i)$ and the ground truth $y_i \in \mathbf{y}$ values to guide the training process. If

$\mathcal{L}(f(\mathbf{x}_i), y_i)$ is a loss function, and the value of $\mathcal{L}(f(\mathbf{x}_i), y_i)$ during training decreases, that in theory signals the improvement of the model. This study briefly discusses two loss functions mostly applied for classification problems: binary cross-entropy loss and multi-class cross-entropy loss. The standard weighted binary cross-entropy loss function can be expressed as:

$$\mathcal{L}_{wbce} = -\frac{1}{n}\sum_{i=1}^{n}\left[w \times y_i \times \log(f_\theta(\mathbf{x}_i)) + (1 - y_i) \times \log(1 - f_\theta(\mathbf{x}_i))\right],$$

where w is a weight associated with each class, n is a count of samples, and f_θ is a model with weight θ. The weights can be adjusted with the understanding of the data. It is common to give more weight to minority classes (Ruby & Yendapalli, 2020). The categorical cross-entropy loss is given as:

$$\mathcal{L}_{wcce} = -\frac{1}{n}\sum_{c=0}^{|\mathbf{c}|}\sum_{i=1}^{n} w_c \times y_i^c \times \log(f_\theta(\mathbf{x}_i, c)),$$

(Ruby & Yendapalli, 2020) where w_c is a weight associated with each class, $\mathbf{x}_i$ is an input training sample with a target y_i. Loss is the penalty for a bad prediction. That is, loss is a number indicating how bad the model's prediction was on a single example. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater.

Contrary to supervised learning, unsupervised learning uses an unlabelled dataset. Unsupervised learning needs no targets, and it receives input and learns the patterns to generate the targets. Unsupervised learning tries to find patterns in the input data, and this is pure unstructured noise (Barlow, 1989). In reinforcement learning, the algorithm accepts inputs, produces actions, and it is rewarded. The idea of the machine is to maximize rewards based on the current and future actions of the other machines. The latter two ML concepts are not of focus in this thesis, therefore, do not warrant detailed explanation. For an extensive discussion on unsupervised and reinforcement learning, we refer the reader to Gentleman & Carey (2008) and Szepesvári (2010), respectively.

3.3 Classification

This section discusses classification to introduce the two main classification categories, namely the binary and the multi-classification. Since this thesis is based on a multi-classification problem, multi-classification will be discussed in detail.

3.3.1 Binary Classification

Assume that $\mathbf{D}$ is the dataset; $\mathbf{D} = \{\mathbf{x}, \mathbf{y}\}$, where $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ is the inputs data and $\mathbf{y} = \{y_1, y_2, \cdots, y_n\}$ is the list of class labels associated to the input data and n is the count of data samples. We can write, $\mathbf{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i$ is an image and y_i is a label. The binary classifier tries to build a function:

$$f : \mathbf{x} \to \{0, 1\}$$

that takes an input and assign it a class of 0 or 1. The objective of binary classification in ML, is to train on the existing dataset to find the function f that maps the values of $\mathbf{x}$ to those of $\mathbf{y}$. The rule for binary classifier is defined as:

$$f(\mathbf{x}) = \begin{cases} 1, & \text{if } p(\mathbf{x}_i) \text{ satisfies the requirement.} \\ 0, & \text{otherwise.} \end{cases}$$

Here $p(\mathbf{x}_i)$ is the probability such that $f(\mathbf{x}_i)$ is 1, the positive outcome. In most cases if $p(\mathbf{x}_i) \geq 0.5$ we have $f(\mathbf{x}_i) = 1$, otherwise $f(\mathbf{x}_i) = 0$. But there are sensitive problems to the threshold where $p(\mathbf{x}_i) = 0.5$ is moved to accommodate sensitivity. For an example, if we were to build a classifier that seeks to classify individuals who have contracted COVID-19 (Arpaci et al., 2021), it would be best to allow a room for error in those who test positive while they are negative and allow no mistake that will classify someone as negative while they are positive. In this case it would be reasonable to set the threshold such that if $p(\mathbf{x}_i) \geq 0.35$, $f(\mathbf{x}_i) = 1$, otherwise $f(\mathbf{x}_i) = 0$.

3.3.2 Multi-Classification

The supervised multi-class classification algorithms try to construct a function to map inputs to at least three classes (Aly, 2005). The multi-class classification problems are solved by extending the binary classification approach for some algorithms (Voloshynovskiy et al., 2009). One goal of binary or multi-classifiers is to build a rule to predict $\mathbf{y}_i$ given $\mathbf{x}_i$ using the available dataset. There are two common algorithms to build a multi-class classifier from a binary classifier; the One-vs-All and the One-vs-One.

3.3.2.1 One-vs-All

For $|\mathbf{c}|$-class instances dataset, the One-vs-All generates $|\mathbf{c}|$-binary classifiers. In a One-vs-All techniques, each class has only one classifier. Most multi-classification work general applies this strategy as a default choice. The technique assigns the class 1 to a given class $c_i \in \mathbf{c}$ and treats all the $\mathbf{c} - c_i$ classes as 0. This is ran for all $c_i \in \mathbf{c}$, where $\mathbf{c}$ is a set of classes, if the size of $\mathbf{c}$ is $|\mathbf{c}|$ then the method builds $|\mathbf{c}|$ binary classifiers (Har-Peled et al., 2003).

These binary classifiers return the probability that the input is of label 0 or 1 (Rifkin, 2008). In other words, the algorithm returns the likelihood that the input is part of class c_i for all classes in $\mathbf{c}$ and the class with a probability score above the given threshold is chosen (Aly, 2005). When the threshold is not defined, the class with a bigger score will be chosen. An illustration of four classes problem is depicted in Figure 3.1 using the One-vs-All algorithm. From four classes, four binary classifiers are generated as shown on Figure 3.1.

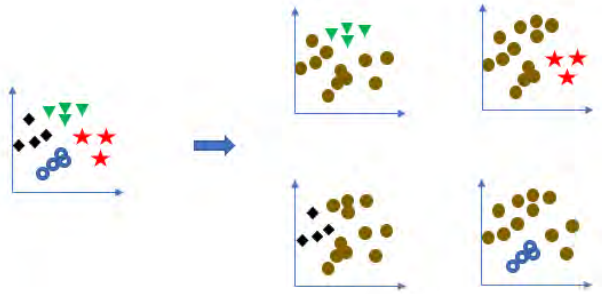


Figure 3.1: One vs All multi-classifier

3.3.2.2 One-Vs-One

For every pair of classes, the One-Vs-One algorithm generates one classifier. During prediction, it selects the class that receives majority of votes (Har-Peled et al., 2003). Suppose two classes have received equal votes. In that case, it selects the class with the most sum score as the label by adding up all the pair-wise classification confidence scores calculated by the underlying binary classifiers (Platt et al., 1999). This techniques generates $\frac{|\mathbf{c}|(|\mathbf{c}|-1)}{2}$ classifiers for $|\mathbf{c}|$ classes (Rifkin, 2008). Figure 3.2 displays the One-vs-One classifier. It is an example of classifying four categories, each class is compared to every class and it generates six models.

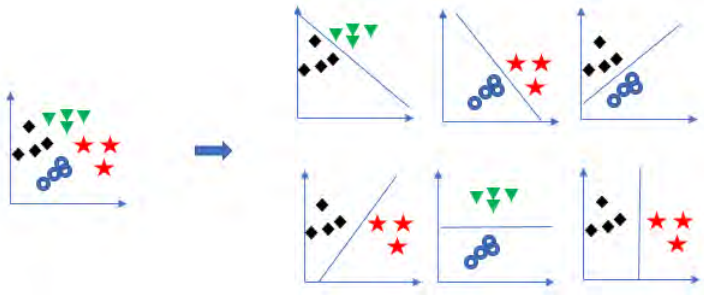


Figure 3.2: One vs one multi-classifier

AlexNet is a CNN model, and uses Multi-layer Feed-forward Neural Networks to provide a natural extension to the multi-class problem. It has $|\mathbf{c}|$ binary neurons in the output layer in place of the normal one neuron with binary output. In this study, we use the One-vs-all multi-classifier. We expect three neurons in the output layer since we are classifying three classes.

3.3.3 Cross-validation

Selection bias and over-fitting are common challenges of ML and DL models. The holdout technique splits the sample data into two independent parts, training and validation data. This technique is likely to suffer from selection bias since only one part of the data is used in training. Selection bias occurs when the data used for training or test does not represent most of the dataset's features of the population intended to be analyzed. It can lead to models performing poorly on large and unseen datasets.

The k-fold cross-validation method trains the model by splitting the dataset into k different approximately equal-sized parts. Each of the k-parts gets a chance to be used for validation whilst the other $k-1$ parts are combined to train the model. It then generates k models from an architecture of one model, which produces k possible results. The recommended choices of k are 5 and 10 (Moreno-Torres et al., 2012). The more folds made from the data sample, the more unbiased the result. The most unbiased result is likely to be generated when k is the count of data samples, which is computationally expensive.

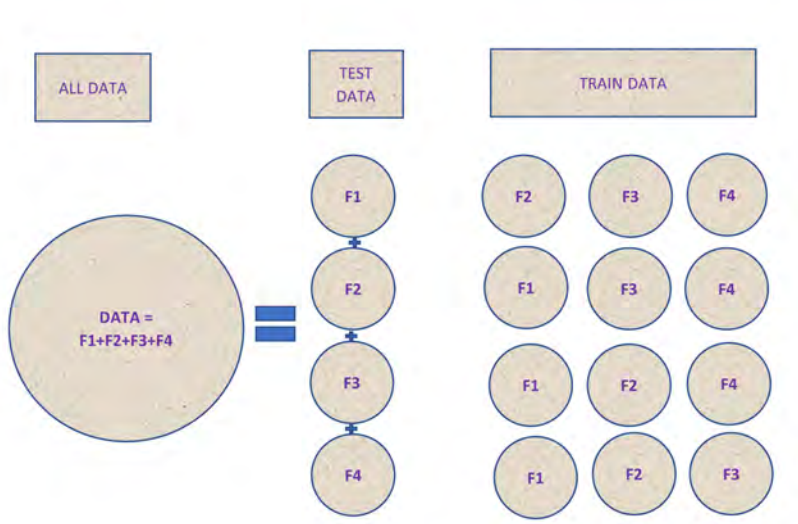


Figure 3.3: cross-validation (adapted from Berrar (2019))

Figure 3.3 presents a demonstration of how the cross-validation process is carried out. For example, the dataset is split into four folds. The folds are named F1, F2, F3, and F4. This split will produce four models. The first model will be trained using F2, F3, and F4. F1 is kept for the test. The same process as illustrated in Figure 3.3 is applied to generate the other three models. It generates four models, the results are aggregated to produce the final model score in classification problems. The results obtained from this process indicate how accurate the model generated from this data is. We used this technique to validate the results for the model we built.

3.3.4 ROC and AUC

The receiver operating characteristic (ROC) curve is a probability curve used to graph the true positive rate (TPR) against false positives rate (FPR). The area under the ROC curve, (AUC), shows the model's ability to capture all the categories. AUC indicates the degree of separability of the classes. The AUC value ranges between 0 and 1. The bigger the value of AUC, the better the model. In the context of classification in ML, positive refers to a desired outcome, and negative means unfavoured outcome. The following equations define how to compute the TPR and the FPR.

$$TPR = \frac{TP}{TP + FN},$$

where TP presents the count of true positives, it shows the count of actual positive examples classified as negative. TPR measures the sensitivity of the model to a particular class.

$$FPR = \frac{FP}{TN + FP},$$

where FP is the count of false positives, it returns the count of actual negative data points classified as positive. TN is the count of true negatives and presents the count of negative data points classified accurately. There is also a measure of specificity:

$$Specificity = \frac{TN}{TN + FP}.$$

The $Specificity$ is inversely proportional to sensitivity; it increases with the decrease in sensitivity, and vice versa. AUC and ROC are crucial metrics when assessing a classification model. For this thesis, we consider this work a multi-classification problem, and a one-vs-all approach is employed.

3.4 Machine Learning Algorithms

3.4.1 Decision Trees

Among the commonly applied data mining techniques are the decision trees (Shouman et al., 2011). Decision trees are supervised machine learning models that are commonly applied to solve classification and regression problems. Rokach & Maimon (2005) defined decision trees as classifiers expressed in a recursive partition of the current state. A decision tree starts from a root node with only outgoing edges, connecting to internal nodes as incoming edges. Each internal node and the root node splits into two or more sub-spaces that generates the subsequent nodes using the classification function $f(\mathbf{x}_i)$. Internal nodes are also known as chance nodes since they show multiple uncertain outcomes. The outcome can either be

another internal node or end node. End nodes display the outcome.

Let μ be the state or domain to which $\mathbf{x}_i$ belongs to. The function defines the following node values:

$$f(\mathbf{x}_i) = \begin{cases} f_1(\mathbf{x}_i), & \text{if} \quad A(\mathbf{x}_i) \quad \text{holds} \\ f_2(\mathbf{x}_i), & \text{else} \end{cases}$$

where f_1 and f_2 are either a constant or another classification function (Hegland, 2001). $A(\mathbf{x}_i)$ is a condition that defines the split. The complexity of a balanced tree is $O(\log(n))$ levels, where n is the count of data points. For an unbalanced decision tree, $O(nm\log(n))$ level is the average complexity of the tree building algorithm, where m is the count of attributes.

The decision tree is an optimal algorithm; each partition is greedily selected. The algorithm chooses a split with the maximum information gain. This brings us to the next subsection; Node impurity and information gain.

3.4.1.1 Node Impurity and Information Gain

Deciding on the best split for a problem is an essential step of decision trees due to their greedy nature. Decision trees partition the nodes in all available attributes and choose the split with the most homogeneous results which means a node dominated by one class. There are many algorithms to decide on the best split for an existing problem. For the scope of this research, we discuss the Gini impurity, Entropy, and Variance since they are most used and flexible for different problems. They are all used in classification problems except for the latter, which works for regression tasks (Tangirala, 2020).

For the Gini Impurity, let $\mathbf{D}$ be a training dataset such that $p_y(\mathbf{D})$ is a probability vector of the targets that is defined by:

$$p_y(\mathbf{D}) = \left(\frac{|\mathbf{D}_{y=0}|}{|\mathbf{D}|}, \frac{|\mathbf{D}_{y=1}|}{|\mathbf{D}|}, ..., \frac{|\mathbf{D}_{y=|\mathbf{c}|-1}|}{|\mathbf{D}|} \right),$$

where $|\mathbf{c}|$ is the count of classes, and $\frac{|g(\mathbf{x}_i)_{y=c_i}\mathbf{D}|}{|\mathbf{D}|}$ is the likelihood of class c_i in the selected node. The $Gini$ is given as:

$$Gini(y, \mathbf{D}) = 1 - \sum_{i=1}^{|\mathbf{c}|} (p_i)^2.$$

Gini lies between zero and one, as it is the probability, and the higher this value, the more will be the purity of the nodes. The criteria for choosing the main attribute $\mathbf{x}_i$ is defined by:

$$GiniGain(\mathbf{x}_i, \mathbf{D}) = Gini(y, \mathbf{D}) - \sum_{c=0}^{|\mathbf{c}|} \frac{|\mathbf{D}_{y_i=k}|}{|\mathbf{D}|} Gini(y, \mathbf{D}_{y_i=c_i}).$$

According to Shouman et al. (2011), the Entropy is defined as:

$$Entropy(y, \mathbf{D}) = -\sum_{k=1}^{|\mathbf{c}|} p_k \log(p_k).$$

Entropy is used to generate information gain as shown:

$$InfoGain(\mathbf{x}_i, \mathbf{D}) = Entropy(y, \mathbf{D}) - \sum_{i,k} \frac{|\mathbf{D}_{y_i=k}|}{|\mathbf{D}|} Entropy(y, \mathbf{D}_{y_i=k}).$$

The $InfoGain$ is always greater than or equal to zero. This is because by observing the input $\mathbf{x}_i$, you either gain information or not, but you do not lose information.

In classification problems, the gain is applied, but for regression models, we employ variance formulation as given in Equation 3.4.1:

$$\sigma = \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y_i}), \tag{3.4.1}$$

where n is the count of samples in $\mathbf{D}$, and $\bar{y_i}$ is the mean of the target values. The homogeneity of the node is found by calculating the variance. If a node is entirely homogeneous, then the variance is zero. The attribute candidate at a given node is that of the biggest Gini gain or information gain.

3.4.1.2 Pruning

The greedy nature of decision trees often leads to complex trees because they have captured noise in the dataset (Mohamed et al., 2012). Since decision trees fall among the most researched data mining methods, the literature covers many pruning methods to solve the issues of complex trees that lead to high energy consumption. Large trees risk having overfitting. Pruning can be mainly approached in two different techniques: the top-down and bottom-up approaches. The top-down approach starts pruning from the root node, and the bottom-up resumes the checks from the last nodes of a tree. We review two pruning methods, cost-complexity, and pessimistic pruning, as discussed by Rokach & Maimon (2005). At the end of pruning, some nodes have been reduced to leaves. The process seeks to reduce all the unnecessary nodes.

Cost-Complexity pruning is a widely used method for pruning decision trees. Let T_0 be the original decision tree before pruning. In the process of pruning a sequence $S_k = \{T_0, T_1, T_2, ..., T_k\}$ of trees is generated. The pruning starts with T_0 and T_{i+1} is generated from T_i by removing at least one sub-tree of T_i and replace with leaves (Quinlan, 1987). For all $S \in T$, where S is a sub-tree but S is different to $leaf$. To determine whether to keep a node or sub-tree, we

have:

$$\alpha = \frac{e(pruned(T,S),\mathbf{D}) - e(T,\mathbf{D})}{|leaves(T)| - |leaves(pruned(T,S))|},$$

where e stands for the count of miss-classified trees or sub-trees, and $|leaves(T)|$ is the number of leaves of a given tree T or sub-tree S (Rokach & Maimon, 2005). $pruned(T,S)$ denotes pruned trees where a node t has been replaced by the most dominant leaf, and α is evaluated for all $T \in S_n$ and the T_i with the minimum α is the most pruned tree.

Mansour (1997) says pessimistic pruning relies on the size of a training set in each sub-tree and the misclassification error. Pessimistic pruning employs a top-down approach to traverse the internal nodes. This procedure cuts out all of the descendants of the pruned node. As a consequence of that, the pruning becomes relatively faster (Maimon & Rokach, 2005). Mingers (1989) defines three main equations to assess if a node will be pruned or not. A continuity correction for a node is given by:

$$n'(t) = e(t) + \frac{1}{2},$$

where $e(t)$ is the count of misclassified examples at node t. A continuity correction for a sub-tree is defined as:

$$n'(T_t) = \sum e(i) + \frac{|leaves(T)|}{2},$$

where i covers the leaves of the sub-tree S. The standard error allowed is:

$$SE = \sqrt{\frac{n'(T_t)[N(t) - n'(T_t)]}{N(t)}},$$

where $N(t)$ is the count of the training set at a given node. If $n'(T_t) + SE \geq n'(t)$, the sub-tree is pruned, else it is kept as it is (Mingers, 1989). This method only requires one pass for all the nodes it reaches when traversing from the root node and does not always reach all the nodes because some might be pruned with sub-trees. Although decision trees are reliable, effective, and easy to understand, they tend to perform poor than other algorithms such as random forest or SVM when applied in the same data (Podgorelec et al., 2002). They are unstable, and minimal changes in the training dataset can lead to a big difference in the main structure of the tree.

3.4.2 Support Vector Machines

SVMs are classical ML techniques introduced in the early 1990s (Deka et al., 2014). SVMs tackle classification and regression prediction problems (Gunn et al., 1998). Image classification problems can be dealt with using SVMs (Noble, 2006). Noble (2006) defines SVMs as mathematical algorithms that define a maximal function $f : \mathbf{x} \rightarrow y$ for classification and prediction problems. Some datasets can easily be classified by a linear classifier (hyperplane), there are usually many hyperplanes that can separate a data of two or more classes, but

SVMs seek to find the optimal hyperplane (Manevitz & Yousef, 2001). SVMs solve binary and multi-class problems. In a multi-class problem, it converts the problem to binary as discussed in Section 3.3.2.

SVMs seek to find the function $f : \mathbb{R}^N \longrightarrow \{\pm 1\}$ given a training data:

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_n, y_n)\} \in \mathbb{R}^N \times \{\pm 1\}.$$

The function f defines a hyperplane that divides the two investigated classes to two different sides of the plane. It also maximises the least distance from the hyperplane to either of the two classes (Schuldt et al., 2004). The optimal hyperplane is the plane that separates the two classes by a maximum distance from both categories.

3.4.2.1 Linearly Separable Datasets

Assume that the general equation of hyperplane is $\mathbf{w}.\mathbf{x} = 0$, where $\mathbf{w}$ is the vector of weights (Gunn et al., 1998). The vector $\mathbf{w}$ forms a right angle with the surface of the hyperplane h. That implies its unit vector $\mathbf{u} = \frac{\mathbf{w}}{||\mathbf{w}||}$, and it has to be perpendicular to the h with magnitude 1. SVM returns a unique global solution for each problem (Mavroforakis & Theodoridis, 2006). In Figure 3.4, although all the three linear functions classify the classes 100% correct, the orange line is the optimal hyperplane to classify the two classes.

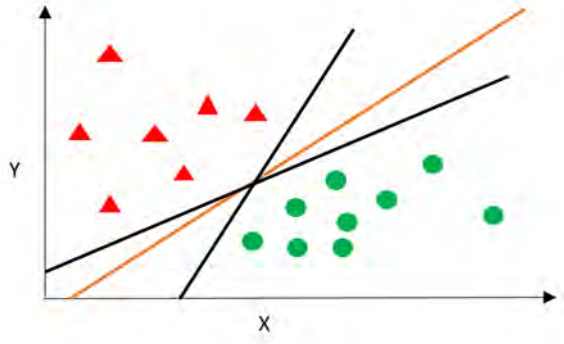


Figure 3.4: Linear problem

Data points that are situated very close to the hyperplane highly influence the location and adjustment of the hyperplane and are known as support vectors (Yu & Kim, 2012). Removing or shifting a support vector, shifts the position of the hyperplane. For every linearly separable training dataset, there is a pair $(\mathbf{w}, b)$ such that:

$$\mathbf{w}.\mathbf{x}_i + b \geq +1 \quad \text{if} \quad y_i = +1,$$

$$\mathbf{w}.\mathbf{x}_i + b \leq -1 \quad \text{if} \quad y_i = -1.$$

Let h_1 and h_2 be planes that pass through -1 and 1, respectively. If h_1 and h_2 are planes that pass on top of the support vectors, d^+ is the minimal distance from h_1 to h and d^- is the smallest distance between h_1 and h. The margin d is the maximum distance between h_1 and h_2, $d = d^- + d^+$ (Bennett & Bredensteiner, 2000). Refer to Figure 3.5.

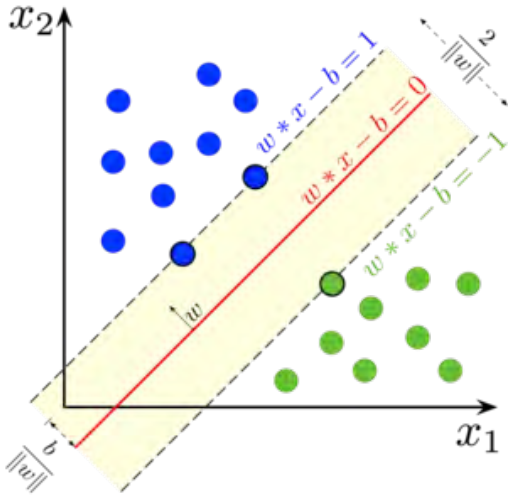


Figure 3.5: Hyperplane (Bennett & Campbell, 2000)

To find an optimal hyperplane, we use optimization techniques with Lagrange equation as detailed by Yu & Kim (2012), Gunn et al. (1998) and Schuldt et al. (2004). Let $\mathbf{x}_i$ be a point that belong to one of the two linearly separable classes being classified, c_1 and c_2. SVM tries to find $g(\mathbf{x}) = \mathbf{w}.\mathbf{x} + b$, a function that splits the two categories with a maximum margin. $g(\mathbf{x}) = 1$ when the closest point $\mathbf{x}_i$ belongs to c_1 and for $g(\mathbf{x}) = -1$, $\mathbf{x}_i$ belongs to c_2. Since the distance of any point to the hyperplane is equal $\frac{g(\mathbf{x})}{||\mathbf{w}||}$, we have:

$$\frac{1}{||\mathbf{w}||} + \frac{1}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}. \tag{3.4.2}$$

To optimise Equation 3.4.2 one has to minimise $\mathbf{w}$, and to do that one can introduce the objective quadratic function:

$$f = \frac{1}{2}||\mathbf{w}||^2, \quad \text{such that} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad 1 \leq i \leq n.$$

Then the Lagrange multiplier is to be optimised:

$$L(\mathbf{w}, b, a) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} a_i[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1)], \quad \forall i, a_i \geq 0,$$

where a_i is a Lagrange multiplier. At a minimum point of the function $L(\mathbf{w}, b, a)$ derivatives with respect to $\mathbf{w}$ and b are equal to zero (Chen et al., 2005):

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} a_i y_i \mathbf{x}_i = 0, \quad \text{and} \quad \frac{\partial L}{\partial b} = \sum_{i=1}^{n} a_i y_i = 0$$

$$\Longrightarrow$$

$$\mathbf{w} = \sum_{i=1}^{n} a_i y_i \mathbf{x}_i, \quad \text{and} \quad \sum_{i=1}^{n} a_i y_i = 0.$$

Then the classification solution is found to be:

$$x \rightarrow sign \sum_{i} a_i y_i (\mathbf{x}_i \cdot \mathbf{x})$$

and the dependence from $\mathbf{w}$ and b has been eliminated.

3.4.2.2 Linearly Non-separable Datasets

Sometimes dataset patterns are not linearly separable, new linearly separable data patterns are derived from the original datasets through transformations to new space, usually higher dimensional spaces (Inoue & Abe, 2001). To enable the optimal hyperplane in a non-linearly separable dataset, Schuldt et al. (2004) suggest an introduction of a penalty function that uses the variable $e \geq 0$. The penalty function:

$$P(e) = \sum_{i}^{n} e_i, \quad 0 < i \leq n.$$

This leads to a modified version of the Lagrange duality and its constraints. The model now intends to reduce or minimise the miss-classifications (Rennie & Rifkin, 2001):

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - e_i.$$

The vector $\mathbf{w}$ determines the generalised optimal separating hyperplane. The main aim now is to minimise the function:

$$L(\mathbf{w}, e) = \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i}^{n} e_i, \tag{3.4.3}$$

where C is a parameter that seeks to direct the relative weighting of two goals of optimising $||\mathbf{w}||^2$ (Inoue & Abe, 2001). In theory, reducing the value of C will lead to more data points being misclassified, and be treated as outliers.

Solving Equation 3.4.3 requires the use of Lagrange multiplier technique since it has two constraints $\mathbf{w}$ and b. The technique requires that you translate into the Lagrange dual form and solve with one constraint rather :

$$L(\mathbf{w}, b, e, a, r) = \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i}^{n} e_i - \sum_{i=1}^{n} a_i [y_i(\mathbf{w} \cdot \mathbf{x} + b) - 1 + e_i] - \sum_{i=1}^{n} r_i e_i, \tag{3.4.4}$$

where a_i and r_i are Lagrange multipliers, and they both greater than or equal to zero. As Schuldt et al. (2004) explained, the dual problem follows:

$$\max_a(M(a,r)) = max_{a,r}\left[min_{\mathbf{w},b,e}L(\mathbf{w},b,e,a,r)\right], \tag{3.4.5}$$

$$M(a,r) = min_{\mathbf{w},b,e}L(\mathbf{w},b,a,r,e).$$

The minimum of $L(\mathbf{w},b,a,r,e)$ is found by evaluating partial derivatives of L with respect to $\mathbf{w}, b$ and e , and equate them to zero.

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \implies \mathbf{w} = \sum_{i=1}^{n} a_i y_i$$

$$\frac{\partial L}{\partial b} = 0 \implies \sum_{i=1}^{n} a_i y_i = 0$$

$$\frac{\partial L}{\partial e} = 0 \implies a_i + r_i = C.$$

From the above partial derivatives, and the Equations 3.4.4 and 3.4.5, we can get the dual form (Chen et al., 2005):

$$\max_a W(a) = \max_a -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) + \sum_{k=1}^{n} a_k. \tag{3.4.6}$$

According to Santosa (2009), the dual problem in Equation 3.4.6 is solved by:

$$q = arg\min_a \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) - \sum_{k=1}^{n} a_k, \tag{3.4.7}$$

with the restrictions that

$$0 \le a_i \le C, \quad \text{and} \quad \sum_{j=1}^{n} a_j y_j = 0.$$

3.4.2.3 Non-Linearly Separable Dataset

Real-life problems can be complex such that the data is not linearly separable. The common way to solve non-linearly separable data is to map the data onto spaces of higher dimension and employ the linear classifier to the spaces. The linear classifier uses a dot product between data point vectors, and that idea can be expanded to a non-linear classifier using a dot product of transformed vectors (Fu et al., 2010). Let $\Phi : \mathbf{x} \to \Phi(\mathbf{x})$ be a transformation from lower to upper dimensions. Computing the dot product, $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, where $\mathcal{K}$ is a kernel function is defined by dot product in some expanded feature space.

Determining a Kernel function $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$ is a necessary process when training an SVM classifier (Awad & Khanna, 2015). There are many different kernels which perform differently in non-identical classification problems. Techniques such as cross-validation are used when evaluating the best kernel and hyper-parameters for a classification problem (Ng, 2000). This thesis discusses the three commonly used kernels for real-valued data, and they are sigmoid, polynomial, and radial basis kernel. We first consider the sigmoid kernel given as:

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = tanh(a\mathbf{x}_i^T \cdot \mathbf{x}_j + r),$$

where $tanh$ is the tangent hyperbolic. The function takes two parameters a and r, where a scales the input data, and r controls the threshold of mapping (Lin & Lin, 2003). The polynomial kernel is the most popular in image processing:

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \cdot \mathbf{x}_j + r)^p,$$

where p is the degree of the polynomial, and $r \geq 0$ is a parameter that balances off considerable differences in the degree of the polynomials. Radial Basis Function (RBF) kernel is the most generalised form of kernelisation. Due to its similarity to the Gaussian distribution, it is one of the widely used kernels. The RBF kernel is preferred because it is localized and has a finite response along the complete x-axis. Computing the similarities and how close are two samples from the dataset one use the RBF kernel which is represented as feature vectors in some input space as give by:

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right),$$

where σ is the variance and $||\mathbf{x}_i - \mathbf{x}_j||^2$ is the Euclidean Distance that separates two points. In Figure 3.6, we present a demo of the RBF kernel. The main idea of kernel functions is to

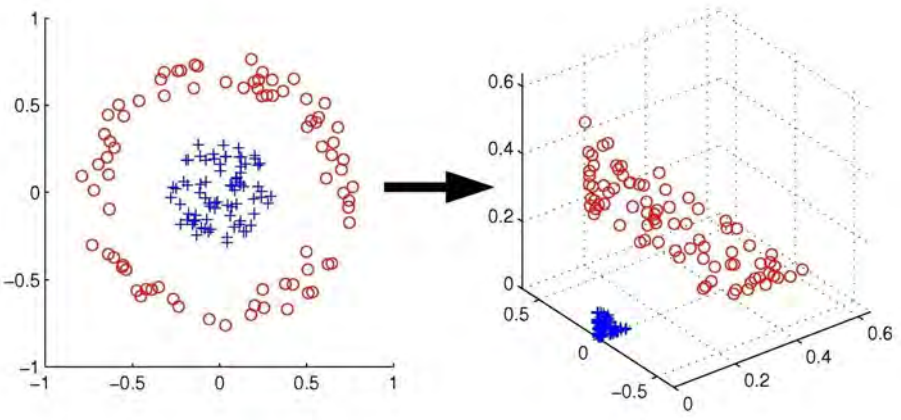


Figure 3.6: RBF kernel

transform input data to the required form for classification.

The solution for non-linearly separable datasets is similar to that of linearly separable data

points except for the modification of bounds of Lagrange multipliers. The Laplace training Equation 3.4.7 changes to this after applying the kernel function:

$$\begin{aligned} L(a)_{max} &= \sum_{i=1} a_i - \frac{1}{2} \sum_i \sum_j a_i a_j y_i y_j, \quad \text{s.t} \quad a_i \geq 0, \quad i = 1, ..., m \\ &= \sum_{i=0} a_i y_i = 0, \end{aligned}$$

which then generates a modified classification problem, that relies on the kernel function (Bridgelall, 2017):

$$\mathbf{x} \rightarrow sign \left(\sum_{i=1} a_i y_i \mathcal{K}(\mathbf{x}_i, \mathbf{x}) \right).$$

3.5 Convolutional Neural Networks

ML and DL are fields of AI concerned with sourcing knowledge and information from data using statistical and mathematical algorithms. They are known for automatically identifying data patterns and then using them to predict unseen input data or to guide decision-makers under uncertain circumstances (Murphy, 2012). Recently, CNN has shown an exceptional ability to extract patterns from visual data, such as images, videos, and more. CNNs are considered pioneers when applied to classification problems and have shown unmatched capabilities when applied to image classification and object detection (Szegedy et al., 2013). They are a type of ANN that specialises in processing visual data (Yamashita et al., 2018). In the past two decades, CNNs have shown exceptional results when dealing with imagery data (Li et al., 2014). CNNs have been successfully applied across various fields such as medicine (Choi, 2018), games (Sutskever & Nair, 2008), weather services (Khaki et al., 2020), finance (Selvin et al., 2017), face-recognition (Coşkun et al., 2017), security (Li et al., 2019), and diagnosis of deceases (Rathod et al., 2018). They are popular for extracting patterns and features from image data. As mentioned by Yang et al. (2021) convolutional layers have the ability to extract features such as corners, objects, edges, and textures. The structure of CNN is made up of three main layers, namely: Convolution layer, Pooling layer, and Fully-connected (FC) layer.

3.5.1 Convolutional Layer

A convolutional layer is a core component of CNNs. This layer receives kernel filters and the input image as parameters (Singh et al., 2020b). Filters are convolved over the entire image, and they have to be smaller than the input image. Filters convolve with the input image to produce feature maps. The feature maps are produced over the feed-forward process that computes dot product from the filters and the pixels of input image. Figure 3.7 illustrates

the convolutional process. The input of 3×3 is given to the model and a filter or kernel of 2×2 is convolved to capture patterns and features from the input data. The process produces an output of 2×2. This output will be passed to the following layer, which could be a fully-connected layer. Convolutional layers are not highly dense connected, some input

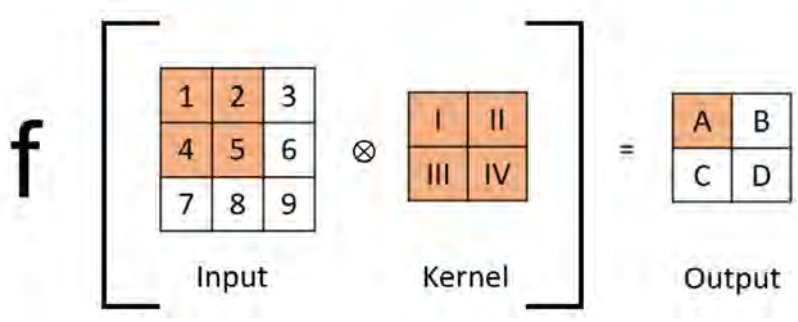


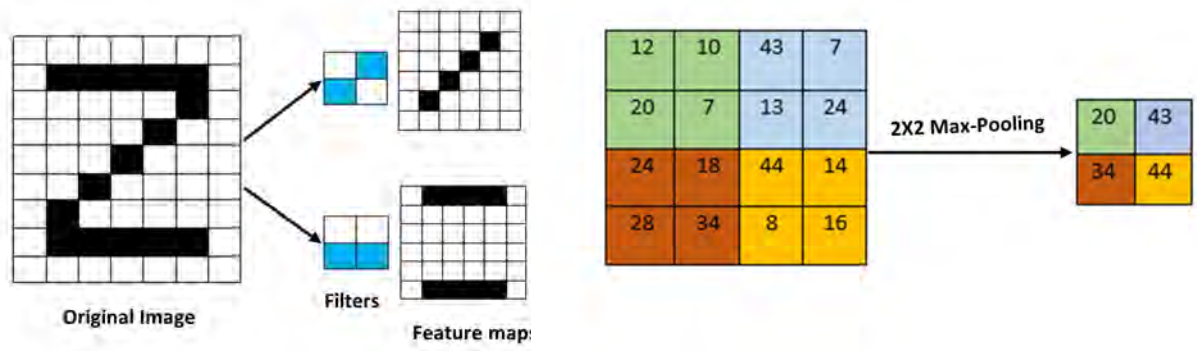
Figure 3.7: Convolutional Layer (Unzueta, 2021)

nodes are disconnected to all the output nodes and it is the same for some output nodes. Because nodes in the convolutional layers have less number of connections that translate to small number of weights per layer and that helps when employed to high dimensional input data such as videos. That is what gives CNNs the ability to be exceptional when analysing imagery data.

3.5.2 Pooling Layer

Pooling layers are normally situated between two consecutive convolutional layers. Immediately after the activation function (e.g., ReLu) has been operated on the feature maps output, pooling is applied. Pooling layers are responsible for down-sampling feature maps and reducing the number of parameters for computational benefits (Ke et al., 2018). The pooling process also helps reduce the model's sensitivity to the precise location of the structures or features of the image (Zeiler & Fergus, 2013). Maximum and average pooling are the two main techniques commonly applied by pooling layers (Ke et al., 2018). Average pooling summarizes the patch feature presence, whereas maximum pooling captures the most dominant feature (See Figure 3.8b). After each pooling layer, new feature maps are produced with less size than the previous ones (Singh et al., 2020a), as shown in Figure 3.8a.

Zeiler & Fergus (2013) say the pooling process aggregates the information with small local regions of the image. Both the techniques have some disadvantages. Average pooling tends to struggle when dealing with feature maps with minimal and high values in the selected region because averaging them does not entirely represent the similar values of the feature map. Whereas maximum pooling over-fits if the maximum value in the region is an outlier. Pooling methods are not limited to the two discussed above. Yu et al. (2014) proposed mixed pooling which combines the two pooling methods and it lessens the loss of information.



(a) Convolution operation. **(b)** Pooling operation

Figure 3.8: Convolution and pooling operations.

3.5.3 Fully-Connected Layer

A fully-Connected layer is a mapping from $\mathbb{R}_m$ to $\mathbb{R}_n$, where $n < m$. Each output dimension depends on each input dimension, implying that every output vector has an input vector that guides or defines it. In Figure 3.9, the two hidden layers are fully-Connected because every node in the input layer has a connection with all inputs in the hidden layer 1. For hidden layer 2, every input from layer 1 has connections with all the nodes in layer 2.

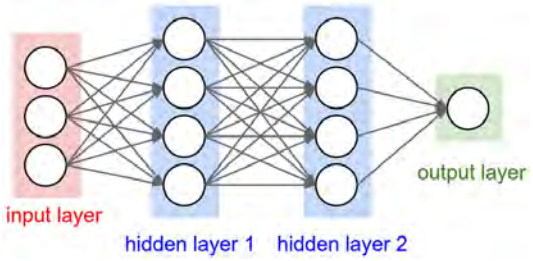


Figure 3.9: Fully-Connected Layer (Li, 2022)

Fully-Connected layer receives input from the feature analysis and applies weights to deduce the class or label. It returns the probability of each class. Fully-Connected layers converts (flattens) the outcomes of the previous layers to a vector that will be an input for the following layer. The general role of these layers is to compile the output data from the preceding layers to decide on the output as shown in Figure 3.10.

Figure 3.10 illustrates flattening happening in the last fully-Connected layer. An input of 1×9 vector and a matrix of 9×4 weights is being transformed using a dot product and activation function. The output is a 1×4 vector.

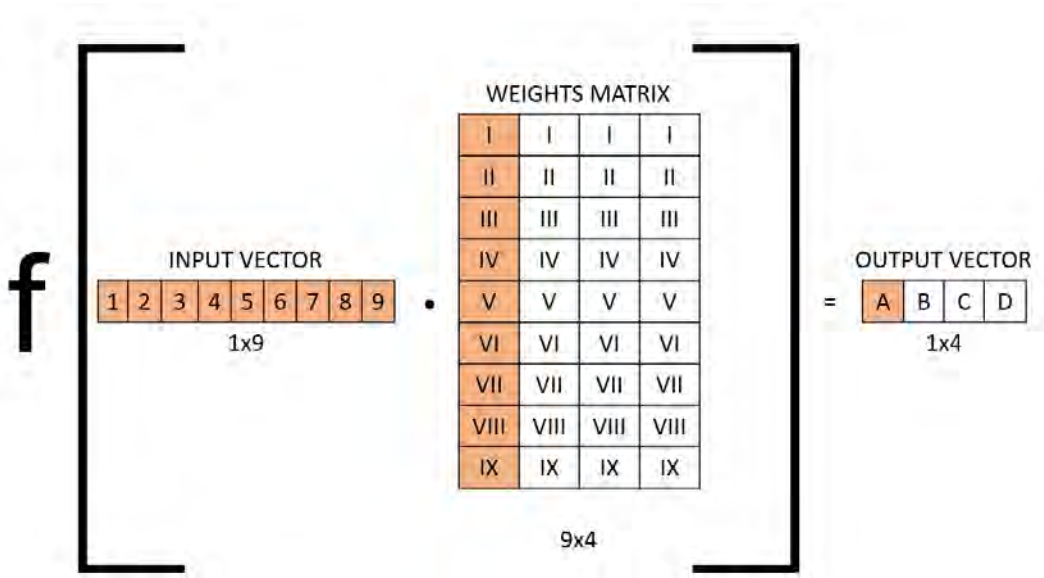


Figure 3.10: Flatten the input (Dot Product) (Unzueta, 2021)

3.5.4 Activation functions

Activation functions assist the neural network to eliminate irrelevant information and keep the vital information for use in classification or prediction (Pratiwi et al., 2020).

The results from the convolved kernels is assigned to the non-linear activation function, which not only assist in learning abstractions or significant information but also sets non-linearity in the new feature space (Khan et al., 2020). Activation functions carry a role to activate neuron based on the information that it carries. As mentioned by Sharma et al. (2017) activation functions derive output from input values fed to a layer (or a node). There are several activation functions, we discuss the ReLu (Rectified Linear Unit), sigmoid (Han & Moraga, 1995), *tanh* (Abdelouahab et al., 2017), and softmax (Kouretas & Paliouras, 2019) functions.

ReLu or rectified activation function is a commonly used activation function (Schmidt-Hieber, 2020) given as:

$$f(\mathbf{x}) = max\{0, \mathbf{x}\},$$

where $\mathbf{x}$ is an input to the neuron, $f(\mathbf{x})$ replaces all negative feature values by zero and return the positive values as they are. The output of the ReLu function is the input of the next convolutional layer. ReLu allows faster and efficient training of deep neural architectures on large and complicated datasets as compared to sigmoid and *tanh* function. Figure 3.11 shows the ReLu function.

The sigmoid activation function is defined by:

$$f(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{x}}}, 0 < f(\mathbf{x}) < 1,$$

where $\mathbf{x}$ represents an output from the convolutional kernels. The sigmoid function is s-shaped

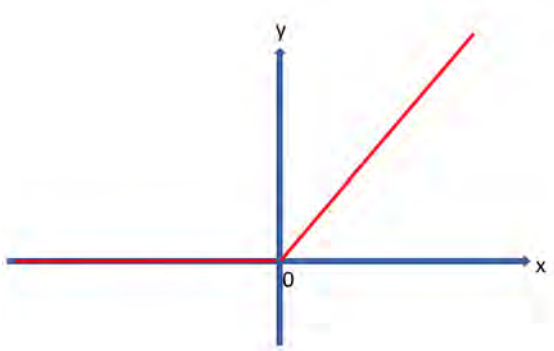


Figure 3.11: ReLU activation function

and is differentiable everywhere it is defined (Sharma et al., 2017). The sigmoid activation function is commonly applied to problems that are expected to produce probability as an output. Its probabilistic interpretation gives it an advantage over other activation functions, which only classify without presenting the probabilities. However, it suffers from being "stuck" in local minimums during training. This is because strongly-negative input values return values near zero. This behavior causes slow weight updates.

Tanh activation function, which is also known as the hyperbolic tangent function, is defined by dividing the hyperbolic *sine* with the hyperbolic *cosine* function.

$$tanh(\mathbf{x}) = \frac{\sinh(\mathbf{x})}{\cosh(\mathbf{x})} = \frac{e^{\mathbf{x}} - e^{-\mathbf{x}}}{e^{\mathbf{x}} + e^{-\mathbf{x}}}.$$

The output of *tanh* ranges between -1 and 1, which minimises the chances of a network being "stuck" during training. *Tanh* is likely to converge faster than the sigmoid function, although the two functions can have a challenge of vanishing gradient and sometimes exploding gradient problems, see Figure 3.12. *Tanh* and sigmoid functions are favoured for binary classification problems.

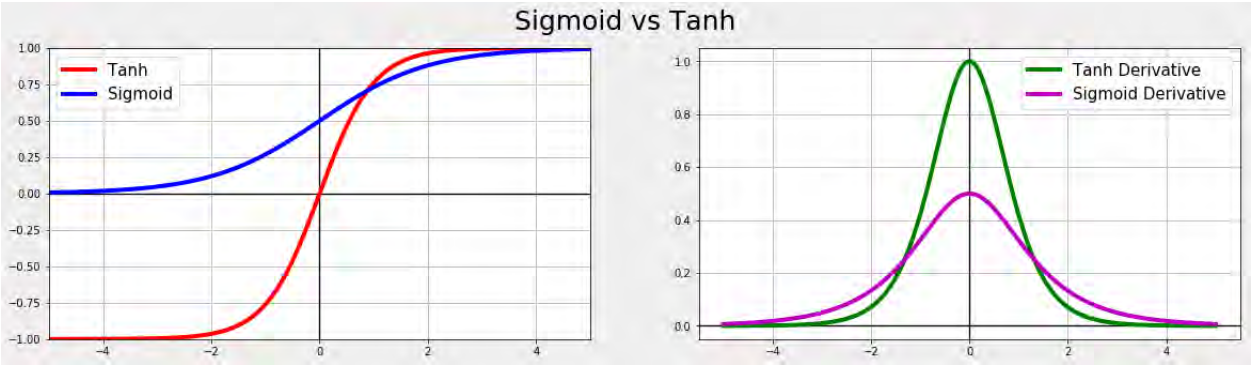


Figure 3.12: Sigmoid and *Tanh* activation functions (Wilson, 2019)

The softmax activation function is applied for multi-classification problems (Adem et al., 2019). According to Adem et al. (2019) "The softmax activation function is used to represent a probability distribution over a discrete variable with n possibilities." The softmax layer is

calculated as:

$$S(y_i|\mathbf{x}_i) = \frac{e^{y_i}}{\sum_{j=1}^{k} e^{y_j}}, \ j = 1, 2, ..., k,$$

for each layer. It is a linear classifier based on probability. The sum of all the probabilities generated by a layer is 1, see Figure 3.13. The variable with the maximum probability is

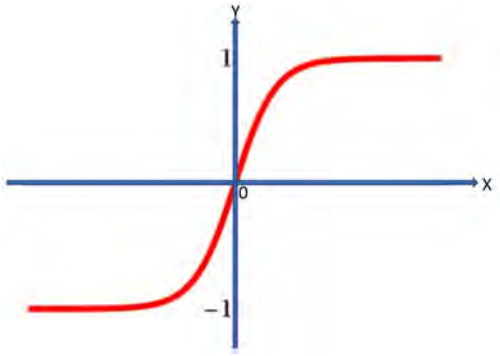


Figure 3.13: Softmax activation function

chosen in the output layer. There are other activation functions, and they play a significant role in the training process and the network's ability to learn, by adjusting neurons' output. For more intensive discussion on other activation functions, we refer the reader to Ding et al. (2018).

3.5.5 Feed-forward and Back-propagation

In the 1960s, the algorithm known as BP was invented. It seeks to generalise optimization methods for performing automatic differentiation of complex composite functions (Baydin et al., 2018). The algorithm received popularity through the work of Rumelhart et al. (1986). It requires a differentiable activation function. Dong et al. (2022) say the BP algorithm is a repetition of three main steps: "inference, backward propagation of errors, and weight update." Inference is also known as forward propagation, it is when a neural network receives input data and propagates it through the neurons until the output layer as a prediction. Backward propagation start with random weights, then adjust them to minimise error so that the neural network learns the relationship between the attributes and targets. In the process, the error signal is fed back to enhance the weights of the network.

BP uses gradient descent to minimise the difference between the targets and the actual network outputs. Gradient descent is commonly used in ML and DL to minimise cost or loss function. It is an iterative optimisation algorithm that finds a local minimum or maximum of a given function (Ruder, 2016). Depending on the nature of the dataset, gradient descent applies different optimization algorithms. For this study we review momentum, Adam, and Nadam. Among the weaknesses of BP is poor rate of convergence as pointed out by Dong et al. (2022). Gradient descent relies on the size of a learning rate, for very small learning

rates, the training fails to converge. For large learning rates, the model is likely to under-fit. Momentum suggest a method of moving average of gradients. It is an extension of gradient descent algorithm, by accumulating the gradient of the past steps to determine the direction to go. The gradient descent processes are executed repeatedly until convergence, given as:

$$v_t = \beta v_{t-1} + (1-\beta)\nabla_w \mathcal{L}(w_{t-1}, \mathbf{x}, y),$$

$$w_{t+1} = w_t - \alpha v_t,$$

where v_t is a current gradient, β is a decay rate, ∇_w indicates a derivative with respect to w, $\mathcal{L}$ is a loss function, and α is a learning rate. One of the mostly used optimisers for image classification is Adam optimizer which is an extended version of the stochastic gradient descent algorithm (Ruder, 2016). The Adam optimizer seeks to keep the information about the decomposing past gradients. Adam optimizer relies on:

$$v_t = \beta_1 v_{t-1} + (1-\beta_1)\nabla_w \mathcal{L}(w_{t-1}, \mathbf{x}, y),$$

$$m_t = \beta_2 m_{t-1} + (1-\beta_2)\nabla_w(\nabla_w \mathcal{L}(w_{t-1}, \mathbf{x}, y)),$$

where v_t and m_t are gradients of the first and second moment found using momentum updating method. The Adam optimizer uses $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ and $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$ to reduce bias and avoid vanishing gradients. The following is used to update the gradient.

$$\theta_{t+1} = \theta_t - \frac{n}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t.$$

An improved version of Adam is known as Nadam (Ruder, 2016). It is a combination of Adam and Nesterov accelerated gradient (Tato & Nkambou, 2018). Nadam optimizer exploit the process by gathering information from the moving gradients, the current gradients and the previous one to direct the training. Nadam is used for difficult gradients to calculate due to high curvatures or noisy data. Nadam updates the parameters as shown below:

$$\theta_{t+1} = \theta_t - \frac{n}{\sqrt{\hat{v}_t} + \epsilon}\left(\beta_1 \hat{m}_t + \frac{(1-\beta_1)\nabla_w \mathcal{L}(w_t, \mathbf{x}, y)}{1-\beta_1^t}\right).$$

Optimizers are an essential building blocks of ML and DL models. They are used to update the attributes of a neural network such as weights and biases to minimise the loss function.

3.5.6 AlexNet Architecture

There are many CNN models to use when dealing with image classification. We employ the well-known AlexNet model as the basis for this project. AlexNet made a breakthrough performance of a CNN model in computer vision. This took place back in 2012, in an imageNet

competition (Krizhevsky et al., 2012). This was an unprecedented model performance. It produced a margin of approximately 10% from the model on the second position. The AlexNet model looks very similar to the LeNet model. LeNet is the earliest convolutional network structure proposed by LeCun et al. (2015b) in 1998. The paper uses BP and feed-forward neural networks to classify handwritten digits (MNIST). The input size of the AlexNet model is 27x27x3. It is an eight-layered network, five of which are convolution layers, and two are fully-Connected, followed by one softmax layer (output), all shown in Figure 3.14. It uses ReLU activation, and three convolution layers have max-pooling done after the convolution (Krizhevsky et al., 2012). AlexNet uses Adam (Adaptive Moment Estimation) optimizer and a dropout rate of 0.05.

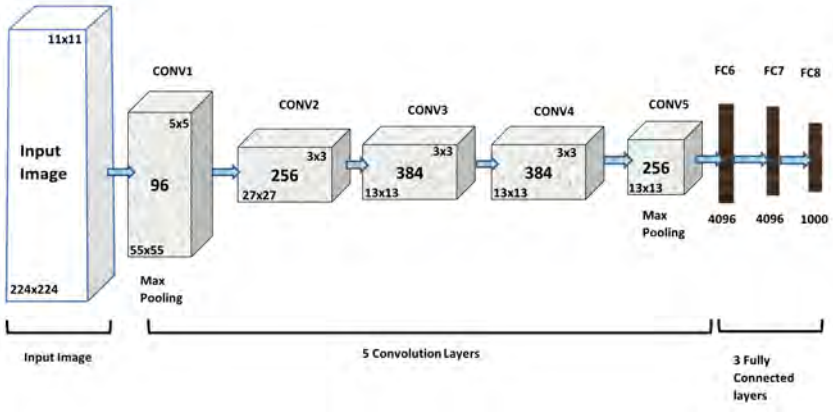


Figure 3.14: AlexNet Architecture Krizhevsky et al. (2012)

3.5.7 VGG-16 Architecture

In this subsection, we look at the standard VGG-16 network architecture. VGG-16 is one of the models of DL that has produced excellent and acceptable performance in image identification and classification. A recent study carried by Sitaula & Hossain (2021) uses VGG-16 network architecture to build a successful model for COVID-19 chest X-ray image classification. Lately, more researchers has shown more interest in using VGG-16 for image classification, see Zhang et al. (2020), Islam et al. (2019), Chitic et al. (2020) and Tamuly et al. (2019). Figure 3.15 displays a VGG-16 basic architecture with 16 layers.

Two pairs and three trios of convolutional layers, followed by a trio of dense layers are all separated by one pooling layer. VGG-16 can be seen as an improvement of AlexNet since it replaces the large kernel-sized filters 11×11 and 5×5 by 2×2 and 3×3 kernel sizes, respectively (Qassim et al., 2018). The input size of the VGG-16 model is $224 \times 224 \times 64$. Guan et al. (2019) says VGG-16 was built to be very deep CNN because it had to classify 1000 in a dataset of more 6 million. VGG was among the top performers in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in the year 2014 (Tammina, 2019), and

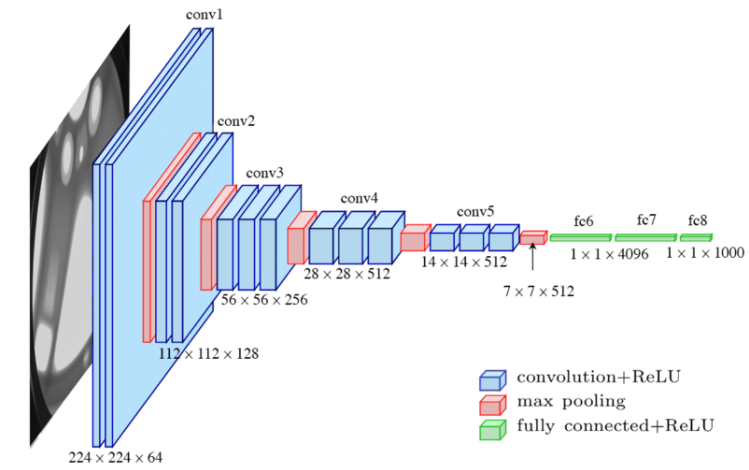


Figure 3.15: VGG-16 Architecture Ferguson et al. (2017)

outperformed the AlexNet model (Guan et al., 2019). VGG achieves improved performance by using numerous weight layers, made possible through the use of small-size convolution filters (Wei, 2019).

3.6 Data Augmentation Techniques

Data shortage is one of the challenges that face engineers when building ML and DL models. There are many techniques of data augmentation which are solutions. In this thesis, we review image transformation and generative adversary networks (GANs) techniques.

3.6.1 Transformation of Images

CNN models train in large datasets to obtain more skilled models (Keshari et al., 2018). Data augmentation is one of the standard methods to increase dataset. This technique creates more data by modifying the copies of existing data. It allows transformed images to be produced from the original images (DeVries & Taylor, 2017). DeVries & Taylor (2017) described dataset augmentation as the process of synthetically expanding the dataset by applying a wide range of transformations to generate slightly different data points from the original dataset. The transformations can include translations, rotations, and isometries. Figure 3.16 shows one image that is augmented to generate six different images.

This technique is efficient and computationally affordable compared to GANs. Section 3.6.2 discusses GANs in detail.

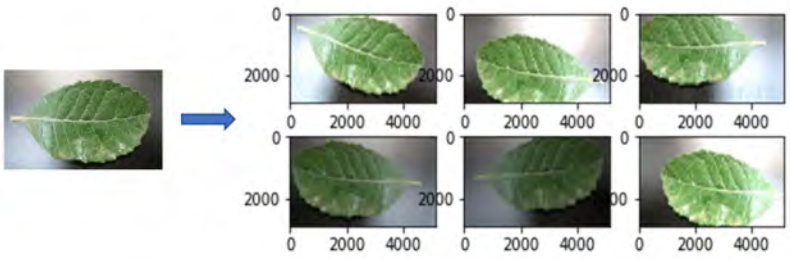


Figure 3.16: Images augmented from 1 image

3.6.2 Generative Adversarial Networks

As mentioned in the above sections, DL models train well in large and high-quality datasets. This section discusses an efficient technique to increase datasets and improve resolution, Generative Adversary Networks (GANs) (Goodfellow et al., 2020). GANs are widely used to generate realistic images, videos, and voices. Goodfellow et al. (2014) introduced GANs in 2014. GANs are made up of two competing models: a generator $\mathcal{G}$ and a discriminator $\mathcal{D}$ (Cao & Guo, 2021). The two models have different roles where the $\mathcal{G}$ model randomly generates a latent space $\mathbf{z}$ and sends it to the $\mathcal{D}$ model that classifies it as authentic or unreal data. The generator starts by generating datasets that are very far from the ones in the training set, and as the model trains, it learns and get skilled, and the model produces improved results. When model $\mathcal{G}$ has learned very well, the $\mathcal{D}$ cannot distinguish that the data from the $\mathcal{G}$ model is fake. Then, the $\mathcal{G}$ model has been well trained to produce realistic data points. Figure 3.17 shows GANs model used to generate plant leaf images.

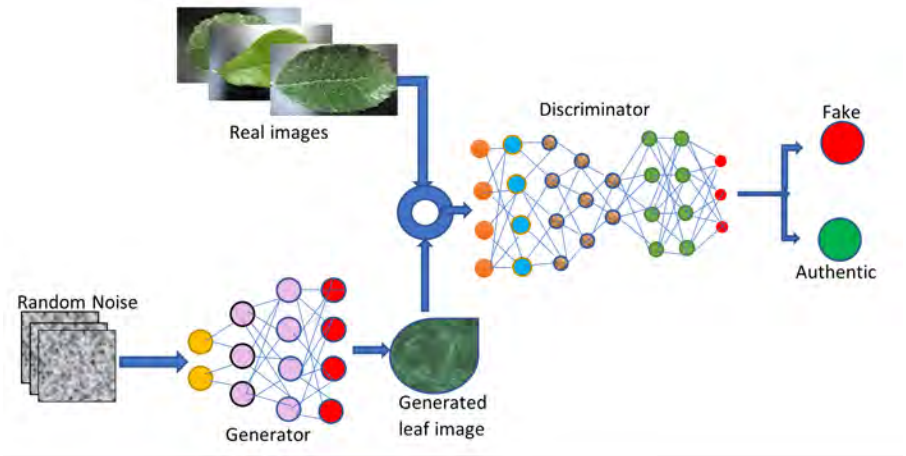


Figure 3.17: GANs model

The generator takes an array with random numbers and returns an image. The discriminator receives a pool of authentic images alongside with a generated image. Neural network of the discriminator is fed with both real and unreal images, it returns a probability that the image belongs to the real dataset. Two feedback loops of the discriminator and generator keeps on updating weights in the models, with the discriminator relying on authentic dataset and the generator learning from the discriminator. When the model has learned well, the

discriminator cannot classify between the authentic and unreal images.

The adversarial modeling framework seeks to learn the generator's distribution ρ_1 over data $\mathbf{x}$, the prior noise is defined by the distribution ρ_2. $\mathcal{D}(\mathbf{x})$ and $\mathcal{G}$ are simultaneously trained, where $\mathcal{D}(\mathbf{x})$ is trained to maximise the probability of correctly matching label to both authentic training examples and generative samples from $\mathcal{G}$, and $\mathcal{G}$ is trained to minimise $log(1 - \mathcal{D}(\mathcal{G}(\mathbf{x})))$. According to Feng et al. (2020), GANs models are optimised by finding the Nash equilibrium between $\mathcal{G}$ and $\mathcal{D}$ of the function $\mathcal{V}(\mathcal{D}, \mathcal{G})$:

$$\underset{\mathcal{G}}{Inf}\underset{\mathcal{D}}{Sup}\mathcal{V}(\mathcal{D}, \mathcal{G}) = \mathbb{E}_{\mathbf{x}\ \rho_1}[log\mathcal{D}(\mathbf{x})] + \mathbb{E}_{z\ \rho_2}[log(1 - \mathcal{D}(\mathcal{G}(\mathbf{x}))],$$

where $\underset{\mathcal{G}}{Inf}$ and $\underset{\mathcal{D}}{Sup}$ are infimum and supremum, respectively.

GANs are difficult to train, and they are prone to failure. Behind that, there are many reasons. Wang (2020) points out three known failure modes of GANs. He mentions vanishing gradients, model collapse, and failure to converge. A skilful discriminator model leads to vanishing gradients, this model have:

$$\mathcal{D}(\mathbf{z}) = 1, \forall \mathbf{x} \in \rho_1$$

and

$$\mathcal{D}(\mathcal{G}(\mathbf{z})) = 0, \forall \mathbf{z} \in \rho_2.$$

The loss function is squashed to 0, and the gradient tends to zero for every data point. This scenario creates a perfect discriminator model, which causes a lack of quality feedback for the generator to learn and improve (Wiatrak et al., 2019). In dealing with vanishing gradients, Ian et al. (2014) suggest a non-saturating loss, though it has its shortcomings such as instability during training.

Mode collapse occurs when a generator returns the same outputs for multiple different inputs. This poses an issue since one of the plausible things about GANs is that diverse output lying within a particular distribution learned by the generator. It is hard to avoid this challenge because it is rooted in the greediness of the generator to fool the discriminator (Wiatrak et al., 2019). Sometimes the discriminator is trapped in a local minimum, and the generator takes that advantage of non-improving discriminator Wang (2020).

When the generator and discriminator loss oscillate and struggle to reach equilibrium, there will be no convergence in the model. Recently, researchers have been trying to find ways to improve the issue of hardly converging GANs.

3.7 Conclusion

Chapter 3 presents a detailed explanation of the models and tools used to accomplish the results presented in Chapter 4. The chapter starts by introducing fundamental concepts of computer vision, such as feature recognition. We looked at features such as image contrast and edge detection. The matrices used to evaluate the capability of models to identify and classify the Brachylaena species are discussed in this chapter. We have reviewed two data augmentation techniques that help in increasing datasets. The next chapter presents experiments and results.

Chapter 4

Experiments and Results

In our experiments, we have used four models of which two are traditional ML models and the other two are DL models. The traditional ML models are SVM and decision trees and DL models are AlexNet and VGG-16. In order to perform experiments, we trained different configurations of the four model architectures on a dataset of 1259 images. The dataset is split as follows: 66% is used for training, 14% for validation and 20% for testing the trained models .

In this section, the results obtained from training different models are presented. The testing accuracy and loss were used to assess the trained models. ROC and AUC curves are other methods used to show the ability of the best models to classify each species. To validate and support the results, we used the k-fold cross-validation method.

4.1 Machine Learning Algorithms Results

4.1.1 SVM Results

This section presents results obtained from SVMs models. A pre-trained SVM model from keras applications is applied. The grid-search is used to find the parameters that optimises and fine tunes the model. Table 4.1 displays parameters and their arguments for grid-search. The grid-search evaluated 108 classifiers with different parameters. The parameters and the

Table 4.1: Grid-search parameters for SVM

Parameter	**Value**
Kernel	linear, poly, rbf, sigmoid
Degree	2, 3, 4
Gamma	1, 0.01,0.001
C	0.1, 1, 10

arguments of the best classifier found by performing grid-search are as follows: the kernel is

set to polynomial with a degree of 4, gamma is set to 1, and C is set to 0.1. The classifier returned an accuracy of 72.45%. Confusion matrix is used to reveal the model's ability to categorise each species, see Figure 4.1. The model cannot recognise *B. ilicifolia* species, it misclassified 77% of the images belonging to *B. ilicifolia*, of the 77% images, 66% were misclassified as *B. discolor*. Contrary to that, the model did very well to recognise *B. discolor* and *B. elliptica*. There were less than 1% images of *B. discolor* that were misclassified, and only 8% of *B. elliptica* images that were misclassified.

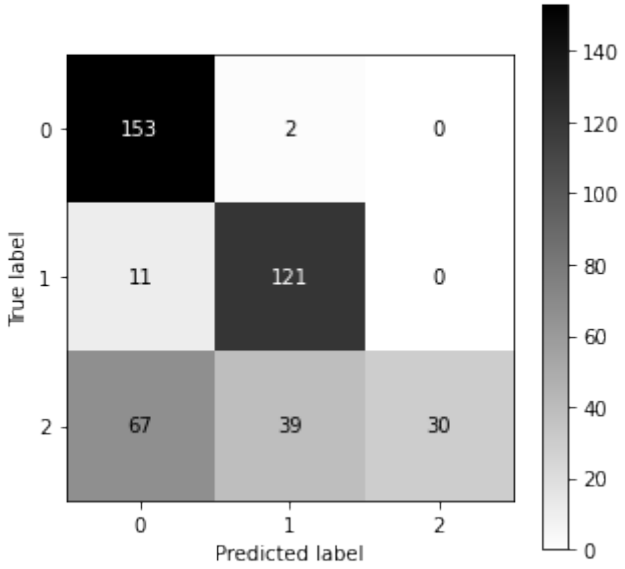


Figure 4.1: Best SVM model matrix with a degree of 16

As anticipated from the above results, the recall and f1-score for *B. ilicifolia* are 22% and 36%, respectively. Confusion matrix summary is shown on Table 4.2

Table 4.2: Summary of confusion matrix (SVM)

	Precision	**Recall**	**F1-Score**
B. discolor	66%	99%	79%
B. elliptica	75%	92%	82%
B. ilicifolia	100%	22%	36%
Accuracy			72%
Macro average	80%	71%	66%
Weighted average	80%	72%	66%

The data is split into 5 folds for cross-validation as explained in Chapter 3. Cross-validation returns an average accuracy of 70.46% from 5 models with standard deviation of 4%. The results are reported in Table 4.3.

4.1.2 Decision Trees Results

In this section, decision trees results are presented. Decision trees are known for over-fitting. We have used a pre-trained decision tree model from the Keras applications.

Table 4.3: Cross-validation results (SVM)

Model	**Test Accuracy**
Model-CV-0	73.91%
Model-CV-1	73.52%
Model-CV-2	62.84%
Model-CV-3	70.00%
Model-CV-4	72.05%
Average	70.46% ± 4.05

4.1.2.1 Unpruned Decision Tree results

The unaltered pre-trained model has its most crucial default parameters set as: criterion is set as Gini, the splitter is set to best, maximum depth of the tree is set to None, and the minimum number of samples needed to be at a leaf node is set to 1. The model registers 84.87% accuracy on the test set. It has produced more than 85 nodes, see Appendix 1. In the following subsection the model is pruned to reduce number of nodes and sub-trees.

4.1.2.2 Pruned Decision Tree Using Grid-search

The model is pruned through grid-search, it is fed multiple values for each of the parameter as shown in Table 4.4 . The grid-search examined 48 models of different parameters. The

Table 4.4: Grid-search parameters for decision tree

Parameter	Value
criteion	2, 3, 5, 10, 15, 30
min samples leaf	5, 10, 20, 50
man samples leaf	gini, entropy

combination of parameters that returned the best results are as follows: criterion uses gini, max depth is set to 5, min samples leaf is set to 10, and random state is set to 42. The number of nodes is reduced to 35 nodes from over 80 nodes of the model graph. Find the graph in Appendix 2 for substantial visualisation. Although the number of nodes and sub-trees were reduced, the accuracy was maintained at 84% on the test set.

Most misclassification came from *B. discolor*, 19% images belonging to *B. discolor* were misclassified to *B. ilicifolia* and *B. elliptica*, roughly 66% of those were misclassified as *B. ilicifolia*. This can be noted on Figure 4.2. The *B. ilicifolia* and *B. elliptica* had fewer misclassification as compared to other species, only 13% and 11% images, respectively, were misclassified.

The model seems to have learned well the features for classifying *B. elliptica* and *B. ilicifolia* but struggles with the *B. discolor*. Table 4.5 displays a summary of confusion matrix.

To verify and support the results generated by the best decision tree model, we used cross-validation method. Results from cross-validation are shown in Table 4.6. Cross-validation

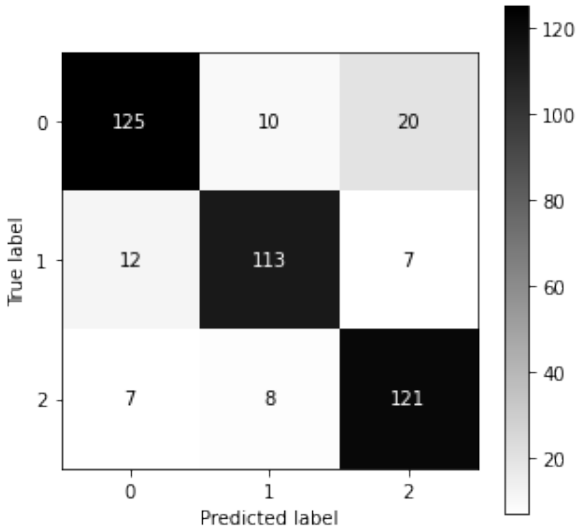


Figure 4.2: One vs one multi-classifier

Table 4.5: Confusion matrix summary.

	Precision	**Recall**	**F1-Score**
B. discolor	85%	85%	85%
B. elliptica	81%	83%	82%
B. ilicifolia	88%	86%	87%
Accuracy			85%
Macro average	85%	85%	85%
Weighted average	85%	85%	85%

Table 4.6: Cross-validation results (Decision Trees)

Model	**Test Accuracy**
Model-CV-0	87.35%
Model-CV-1	82.61%
Model-CV-2	85.77%
Model-CV-3	83.79%
Model-CV-4	82.61%
Average	84.43% ± 2

returns an accuracy of 84.4% with a confidence interval of ±2%. Since the accuracy given by the pruned model lies within the interval range of accuracy obtained by cross-validation, we are confident that it reflects the model's ability to classify the three Brachylaena species. These are convincing results, but ML and DL models will always provide an opportunity for improvement. The best model's accuracy lies within the given interval. In the following section, two CNN models are trained and tested on the same data.

4.2 Deep Learning Results

We trained and tested different configurations of the AlexNet and VGG-16 architecture on a dataset of 1259 images. The dataset was split as follows: 66% was used for training, 14% for validation and 20% for testing the trained models. The original AlexNet architecture was first trained and led to an accuracy of approximately 44%. Different hyper-parameters, including learning rate, were investigated to improve the accuracy. The experiments showed that the values 0.001 and 0.0001 returned efficient results considering the accuracy obtained and the time taken to run the models. The learning rate values initially evaluated are 0.1, 0.01, 0.001, 0.0001, and 0.00001. It was observed that the smaller the values, the models tend to take much time to process but improve significantly in accuracy. Initially, the number of filters in each layer were modified as follows: The filters in the first layer were kept the same, i.e. 48. In the second layer, the number of filters were changed from 128 to 20. The successive layers whose initial filters were 192, and were modified to 20 and 30, respectively. In layer seven, the number increased from 30 to 128. The modified architecture was trained using Adam and Nadam Optimisers with early-stopping. Note that a model trained for a long time could over-fit. Moreover, training a model for a short time does not guarantee a return of a well- skilled model as it might underfit. As a result, the early-stopping strategy was used to monitor when to stop the training. The early-stopping strategy monitored the validation loss throughout the training process. In our experiment, the training process ended when the validation accuracy did not improve over four epochs. We set the maximum number of epochs at 20 and the training stopped at epoch 10. Other hyper-parameters such as batch size and learning rate were also investigated.

For VGG-16 model, a pre-trained model as detailed in Section 3.5.7 is used. The procedure followed for AlexNet was also followed for VGG-16 and the experiments suggested to use 6 epochs for training the configurations.

4.2.1 AlexNet Results and VGG-16 Results

The testing accuracy and loss were used to assess the trained models. ROC and AUC curves are other methods used to show the ability of the best models to classify each species. To validate and support the results, we used the k-fold cross-validation. Twelve models; referred to as Model i, $i = 1, \cdots, 12$ were trained. The number of Batch sizes explored is 32, 64, and 128, and the learning rates were 0.001 and 0.0001. Tables 4.7 and 4.8 show the results of the models trained using the Adam optimiser.

The results show that all the models trained with Adam optimizer could extract relevant features necessary to classify any plants belonging to the three categories. The test accuracy ranges between 95% and 98%. The best two models from the two different architectures were chosen based on the accuracy of each model. In the AlexNet models, Model 2 achieved the

Table 4.7: AlexNet models trained using the Adam optimiser and number of epochs 10.

Optimiser = Adam						
Model Name	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
Batch Size	32	64	128	32	64	128
Learning Rate	0.001	0.001	0.001	0.0001	0.0001	0.0001
Test loss	11.47%	6.29%	10.60%	10.18%	11.70%	18.59%
Test accuracy	95.92%	97.14%	95.51%	95.18%	95.92%	95.10%

Table 4.8: VGG-16 models trained using the Adam optimiser and number of epochs 6.

Optimiser = Adam						
Model Name	Model 7	Model 8	Model 9	Model 10	Model 11	Model 12
Batch Size	32	64	128	32	64	128
Learning Rate	0.001	0.001	0.001	0.0001	0.0001	0.0001
Test loss	11.47%	6.29%	28.46%	10.18%	27.05%	18.59%
Test accuracy	95.92%	97.14%	97.55%	97.59%	95.92%	95.10%

highest accuracy. For VGG-16 models, Model 10 had the highest accuracy. Model 2 used a batch size of 64 and learning rate of 1×10^{-3} whereas Model 10 used a batch size of 32 and a learning rate of 1×10^{-4}. Using the Nadam optimiser with the same hyper-parameters, the test accuracy ranges between 93% and 98.50% for AlexNet models, and Model 15 registered the highest accuracy of 98.36% (see Table 4.9). The accuracy for VGG-16 model ranges between 96.5% and 97.96%, and Model 24 has the best accuracy of 97.96%. On average, models from the VGG-16 architecture using Nadam optimiser seem to perform better than most of the models, this can be seen in Table 4.10. The best two models from both architectures are found from Nadam optimiser, it is Model 15 with 98.36% and Model 24 with 97.96%.

In Table 4.11, the confusion matrices of Model 15 (AlexNet) and Model 24 (VGG-16) are shown.

The summary of the confusion matrix model is displayed in Table 4.12. The table shows accuracy, F1-Score, recall, precision and specificity.

Figures 4.3b and 4.3a display the ROC and AUC for Model 15 and Model 24, respectively. It is clear that both models are able to classify and identify all the three species. *Discolor* is the one that the models seem to capture more accurately than other species.

More than half of the trained models provided an accuracy above 95%. The reliability and efficiency of this performance were validated by performing a k-fold cross-validation method. The k-fold cross-validation method also reduced the selection bias issue described in Section 3.3.3. We randomly split the data into 5 equal folds. All the hyperparameters shared by Model 24 were kept constant for k-fold cross-validation. As per the experiments in the above sections, batch sizes did not significantly impact the performance of the models. Models trained with the Nadam optimiser were, on average, more accurate than those trained with the Adam optimiser. Hence we use the Nadam optimiser for k-fold cross-validation.

Table 4.9: AlexNet models trained using the Nadam optimiser and number of epochs 10.

Optimiser = Nadam						
Model Name	Model 13	Model 14	Model 15	Model 16	Model 17	Model 18
Batch Size	32	64	128	32	64	128
Learning Rate	0.001	0.001	0.001	0.0001	0.0001	0.0001
Test loss	13.62%	9.70%	9.31%	14.48%	11.14%	17.57%
Test accuracy	93.88%	96.73%	98.36%	93.87%	96.73%	94.69%

Table 4.10: VGG-16 models trained using the Nadam optimiser and number of epochs 6.

Optimiser = Nadam						
Model Name	Model 19	Model 20	Model 21	Model 22	Model 23	Model 24
Batch Size	32	64	128	32	64	128
Learning Rate	0.001	0.001	0.001	0.0001	0.0001	0.0001
Test loss	40.19%	20.18%	30.3%	54.14%	13.94%	19.85%
Test accuracy	97.55%	97.59%	97.14%	96.73%	97.55%	97.96%

From all the results shown above, Model 24 outperformed all the other models discussed in almost all the matrices examined. To evaluate the credibility of this model, we used the cross-validation method. The data is split into 5 approximately balanced equal folds for testing and training. The results of the five models trained are displayed in Table 4.13. Cross-validation process returns an average of 98.26% which is 0.3% away from Model 24. The average loss of the cross-validation is 14.99% and Model 24 loss is 19.85%. From these results in the mentioned table, we verified that Model 24 is generic enough to classify the three species of *Brachylaena*. Our best model, Model 24 used a batch size of 128, a learning rate of 1×10^{-4}, Nadam optimiser, and it is VGG-16.

4.3 Discussions

The dataset used for this thesis was collected from the field for this research. It has never been used before. As a result, we have found no study in the literature that seeks to classify or identify the three species of *Brachylaena* using ML or DL models. However, there exists

Table 4.11: Confusion Matrix

Confusion Matrix of the two models						
Metrics	**VGG-16**			**AlexNet**		
	Discolor	**Elliptica**	**Ilicifolia**	**Discolor**	**Elliptica**	**Ilicifolia**
TP	80	67	68	72	57	64
FP	16	4	10	36	5	11
TN	162	165	163	165	165	163
FN	3	13	14	11	23	18

Table 4.12: Confusion Matrix Summary

Confusion Matrix of the two models					
Classifier	Accuracy	F1-Score	Recall	Precision	Specificity
AlexNet (Model 15)	79%	79%	79%	81%	90.46%
VGG-16 (Model 24)	88%	88%	88%	88%	94.23%

Table 4.13: Cross-validation Results (VGG-16)

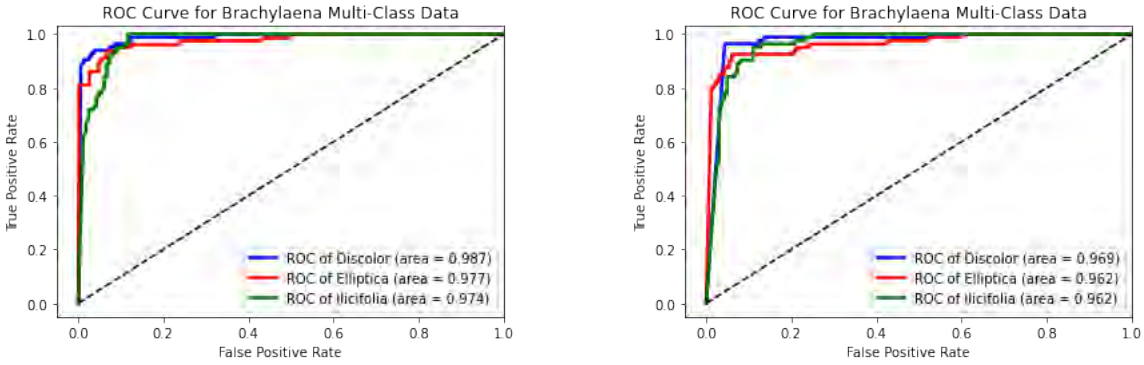
Model	**Test Accuracy**	**Test Loss**
$Model - CV_0$	95.28%	31.28%
$Model - CV_1$	98.81%	20.12%
$Model - CV_2$	99.60%	1.15%
$Model - CV_3$	99.60%	1.55%
$Model - CV_4$	98.03%	20.86%
Average	98.26%	14.99%

research that comes across studies of different domains about *Brachylaena* species, and most are researched outside the field of ML, see Wijianto et al. (2020), and Nyanchama (2021). Fundisi et al. (2021) included one species of Brachylaena among many plant species that were classified in an article by Fundisi et al. (2021).

Since the dataset used in this study has never been used by any research before, the validity and reliability of the results obtained from this research is not supported by citing existing research but by using the validation methods of ML model performances. The confidence of the results of this thesis relies on the robust cross-validation method applied. It was discussed in detail under section 3.3.3. A. Ramezan et al. (2019) define Cross-validation as a methodology that make use of training and accuracy assessment samples repeatedly and thus inherent confidence on the results.

The models evaluated in this thesis are pre-trained except for the AlexNet model. The two traditional ML models performed poorly compared to the DL models. A study carried out by Gunning & Aha (2019) concerning the explainability of AI supports such results. Gunning & Aha (2019) say DL models are complex and not easy to explain, but they perform much better than traditional models when applied to large and complex datasets. It is expected that CNNs would outperform most other models when dealing with imagery data, refer to section 3.5. The performance is from the highest to the lowest; DL models, decision trees, and SVMs.

According to the results observed from the previous section, the *B. discolor* images are more differentiable by the models, see Figure 4.3. The best model by SVM has misclassified two images of *B. discolor*, one misclassified as *B. ilicifolia* and the other one as *B. elliptica*. Seven images of *B. elliptica* are misclassified, three of which are falsely predicted to be *B. discolor* and the rest to be *B. ilicifolia*. Eight images of *B. ilicifolia* are misclassified, and six are



(a) ROC and AUC (Model 24)

(b) ROC and AUC (Model 15)

Figure 4.3: ROC and AUC plots.

incorrectly predicted as *B. discolor*.

The best model of decision trees has the most misclassifications(21) between *B. discolor* and *B. ilicifolia*. This can also be observed from the models of AlexNet and VGG-16. Taking a closer look at the images, it can be seen that *B. elliptica* images have three leaf-end sharp teeth, whereas the other species' leaves have more of a round leaf-end, and that confirms the associativity found by the models.

All the results from the two DL models are satisfactory. In comparison, results obtained by SVMs are not satisfying. The poorest model retained 70.46%, and the best model obtained 98.26%. This research supports that DL models perform better when compared to traditional ML models for complex problems. The traditional models average 77.45% and the CNNs models to 98.31%. DL models took much time to train the models.

4.4 Conclusion

Chapter 4 details the experiments conducted and the results achieved in this work. The chapter starts by presenting the dataset split for training and testing. It then presents the results achieved by the decision trees and SVMs. Both models were evaluated with a finite number of parameters through grid-search. The chapter then proceeds to lay down all the hyper-parameters investigated for the two DL models and gives the ratio of dataset-split with experiments. The previous section, 4.3, discusses the results obtained from the experiments. The two DL models proved more accurate in classifying the three *Brachylaena* species. The summary of the results is shown in Table 4.14.

Table 4.14: Results Summary

	SVMs	**Decision Trees**	**AlexNet**	**VGG-16**
Test Accuracy	70.46%	84.43%	98.36%	98.26%

All the results shown in Table 4.14 were validated with the k-fold cross-validation technique.

The next chapter concludes the thesis and points to future work.

Chapter 5

Conclusion

Misuse, overdosing, or errors in the use of herbal medicines can cause devastating effects on our health. Apart from that, misidentifying medicinal plants is a considerable threat to the medicinal industry. The work conducted in this thesis seeks to classify and identify three *Brachylaena* species known as *B.discolor*, *B.elliptica*, and *B.ilicifolia* that grow up in Makhanda. We reviewed the current work on medicinal plants classification and identification using AI

The models used for this work are decision trees, SVM, AlexNet, and VGG-16. The two DL models returned acceptable results averaging 98% accuracy. The results were rigorously validated by k-fold cross-validation. Most studies we reviewed directed us to use CNN models for image classification problems, and this study asserts that. In this study, the VGG-16 (pre-trained) performed very well in this problem with a classification test accuracy of 98.26% with a confidence interval of ±2.16%. The excellent performance proves the efficiency of CNN models in classifying the three species in the genus *Brachylaena*.

5.1 Limitations of work

As mentioned in Section 4.3, this dataset was collected from the field, and we could not find existing studies to compare the performance of the models. The lack of sufficient amounts of domain-specific data was one of the limitations. It would be great if the data were collected from different places and seasons since plants change with seasons. This leads to classifiers being trained on the dataset with limited information. The machines used to execute the models did not have sufficient capability to examine many hyperparameters simultaneously.

5.2 Future work

Future research should consider compressing the VGG-16 model. The idea is to produce a high-speed and less computationally expensive model that could be deployed on a mo-

bile device. Neural networks models are generally over-parameterized, and there is much redundancy, especially in DL models (Han et al., 2015). A large number of parameters and connections lead to the overuse of memory and computational power. Computational and memory problems become a burden when the model is embedded in a mobile application Han et al. (2015). Hence, in the future, we will consider compressing the AlexNet model. One can also try to use shallow DL models. Choudhary et al. (2020) backs the findings by Han et al. (2015) that DL models tend to run faster and need less energy and memory when compressed. However, it does affect the model's accuracy, and it has a negative impact in most cases. Chu et al. (2003) and Markovsky & Usevich (2012) investigated the use of low-rank approximation for compression, which can be another avenue for future research. The dataset used for this research has good quality. The images have high resolution, and we use three channels dataset. This presents another opportunity to improve speed and reduce the memory needed. The dataset can be converted to grayscale images and do more data pre-processing to save memory.

References

A. Ramezan, C., A. Warner, T., & E. Maxwell, A. (2019). Evaluation of sampling and cross-validation tuning strategies for regional-scale machine learning classification. *Remote Sensing*, 11(2), 185.

Abdelouahab, K., Pelcat, M., & Berry, F. (2017). Why tanh is a hardware friendly activation function for CNNs. In *Proceedings of the 11th international conference on distributed smart cameras* (pp. 199–201).

Adem, K., Kiliçarslan, S., & Cömert, O. (2019). Classification and diagnosis of cervical cancer with stacked autoencoder and softmax classification. *Expert Systems with Applications*, 115, 557–564.

Agatonovic-Kustrin, S. & Beresford, R. (2000). Basic concepts of artificial neural network (ann) modeling and its application in pharmaceutical research. *Journal of pharmaceutical and biomedical analysis*, 22(5), 717–727.

Aguate, M., Tesfahun, A., & Aschale, A. (2020). Medicinal plant part identification and classification using deep learning based on multi label categories.

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)* (pp. 1–6).: IEEE.

Aly, M. (2005). Survey on multiclass classification methods. *Neural Netw*, 19, 1–9.

Arandelovic, O. (2021). AI, democracy, and the importance of asking the right questions. *AI & Ethics Journal*.

Arpaci, I., Huang, S., Al-Emran, M., Al-Kabi, M. N., & Peng, M. (2021). Predicting the covid-19 infection with fourteen clinical features using machine learning classification algorithms. *Multimedia Tools and Applications*, 80(8), 11943–11957.

Awad, M. & Khanna, R. (2015). Support vector machines for classification. In *Efficient Learning Machines* (pp. 39–66). Springer.

Bala, R. & Kumar, D. (2017). Classification using ANN: A review. *International Journal of Computational Intelligence Research*, 13(7), 1811–1820.

Balyen, L. & Peto, T. (2019). Promising artificial intelligence-machine learning-deep learning algorithms in ophthalmology. *The Asia-Pacific Journal of Ophthalmology*, 8(3), 264–272.

Barlow, H. B. (1989). Unsupervised learning. *Neural Computation*, 1(3), 295–311.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18, 1–43.

Begue, A., Kowlessur, V., Mahomoodally, F., Singh, U., & Pudaruth, S. (2017). Automatic recognition of medicinal plants using machine learning techniques. *International Journal of Advanced Computer Science and Applications*, 8(4), 166–175.

Bennett, K. P. & Bredensteiner, E. J. (2000). Duality and geometry in SVM classifiers. In *ICML* (pp. 57–64).: Citeseer.

Bennett, K. P. & Campbell, C. (2000). Support vector machines: hype or hallelujah? *ACM SIGKDD explorations newsletter*, 2(2), 1–13.

Berrar, D. (2019). Cross-validation.

Bhardwaj, S. & Mittal, A. (2012). A survey on various edge detector techniques. *Procedia Technology*, 4, 220–226.

Boryshchak, Y. (2020). Mathematical perspective of machine learning. *arXiv e-prints*, (pp. arXiv–2007).

Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade* (pp. 421–436). Springer.

Breyer-Brandwijk, M. G. et al. (1962). The medicinal and poisonous plants of southern and eastern africa being an account of their medicinal and other uses, chemical composition, pharmacological effects and toxicology in man and animal. *The Medicinal and Poisonous Plants of Southern and Eastern Africa being an Account of their Medicinal and other Uses, Chemical Composition, Pharmacological Effects and Toxicology in Man and Animal.*, (Edn 2).

Bridgelall, R. (2017). Introduction to support vector machines. *Lecture Notes*, (pp. 1–18).

Cao, H. & Guo, X. (2021). *Generative Adversarial Network: Some Analytical Perspectives*. Technical report, arXiv. org.

Castelvecchi, D. (2016). Can we open the black box of AI? *Nature News*, 538(7623), 20.

Chaple, G. N., Daruwala, R., & Gofane, M. S. (2015). Comparisions of robert, prewitt, sobel operator based edge detection methods for real time uses on fpga. In *2015 International Conference on Technologies for Sustainable Development (ICTSD)* (pp. 1–4).: IEEE.

Chen, P.-H., Lin, C.-J., & Schölkopf, B. (2005). A tutorial on ν-support vector machines. *Applied Stochastic Models in Business and Industry*, 21(2), 111–136.

Cherri, A. K. & Karim, M. A. (1989). Optical symbolic substitution: edge detection using prewitt, sobel, and roberts operators. *Applied optics*, 28(21), 4644–4648.

Chitic, R., Bernard, N., & Leprévost, F. (2020). A proof of concept to deceive humans and machines at image classification with evolutionary algorithms. In *Asian Conference on Intelligent Information and Database Systems* (pp. 467–480).: Springer.

Choi, H. (2018). Deep learning in nuclear medicine and molecular imaging: current perspectives and future directions. *Nuclear medicine and molecular imaging*, 52(2), 109–118.

Choudhary, T., Mishra, V., Goswami, A., & Sarangapani, J. (2020). A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review*, 53(7), 5113–5155.

Chu, M. T., Funderlic, R. E., & Plemmons, R. J. (2003). Structured low rank approximation. *Linear Algebra and its Applications*, 366, 157–172.

Cocks, M. L. & Dold, A. P. (2006). Cultural significance of biodiversity: the role of medicinal plants in urban african cultural practices in the eastern cape, south africa. *Journal of Ethnobiology*, 26(1), 60–81.

Coşkun, M., Uçar, A., Yildirim, Ö., & Demir, Y. (2017). Face recognition based on convolutional neural network. In *2017 International Conference on Modern Electrical and Energy Systems (MEES)* (pp. 376–379).: IEEE.

Cunningham, P., Cord, M., & Delany, S. J. (2008). Supervised learning. In *Machine learning Techniques for Multimedia* (pp. 21–49). Springer.

Deka, P. C. et al. (2014). Support vector machine applications in the field of hydrology: a review. *Applied soft computing*, 19, 372–386.

DeVries, T. & Taylor, G. W. (2017). Dataset augmentation in feature space. *arXiv e-prints*, (pp. arXiv–1702).

Dickson, B. (2018). TechTalks the limits and challenges of deep learning. `https://bdtechtalks.com/2018/02/27/limits-challenges-deep-learning-gary-marcus/`. Accessed: 2022-01-28.

Dileep, M. & Pournami, P. (2019). Ayurleaf: A deep learning approach for classification of medicinal plants. In *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)* (pp. 321–325).: IEEE.

Ding, B., Qian, H., & Zhou, J. (2018). Activation functions and their characteristics in deep neural networks. In *2018 Chinese control and decision conference (CCDC)* (pp. 1836–1841).: IEEE.

Djeffal, C. (2019). AI, democracy and the law. *AI Critique/ Volume*, (pp. 255).

Dold, A. & Cocks, M. (1999). Preliminary list of xhosa plant names from eastern cape, south africa. *Bothalia*, 29(2), 267–292.

Dong, S., Chen, Y., Fan, Z., Chen, K., Qin, M., Zeng, M., Lu, X., Zhou, G., Gao, X., & Liu, J.-M. (2022). A backpropagation with gradient accumulation algorithm capable of tolerating memristor non-idealities for training memristive neural networks. *Neurocomputing*, 494, 89–103.

dos Santos Tanaka, F. H. K. & Aranha, C. (2019). Data augmentation using gans. *Proceedings of Machine Learning Research XXX*, 1, 16.

Druzhkov, P. & Kustikova, V. (2016). A survey of deep learning methods and software tools for image classification and object detection. *Pattern Recognition and Image Analysis*, 26(1), 9–15.

Durrington, B. (2019). Cpc, center for plant conservation. https://saveplants.org/. Accessed: 2022-02-03.

El Naqa, I. & Murphy, M. J. (2015). What is machine learning? In *machine learning in radiation oncology* (pp. 3–11). Springer.

Erhan, D., Courville, A., Bengio, Y., & Vincent, P. (2010). Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 201–208).: JMLR Workshop and Conference Proceedings.

Fadja, A. N., Lamma, E., Riguzzi, F., et al. (2018). Vision inspection with neural networks. In *RiCeRcA@ AI* IA*.

Faugeras, O. & FAUGERAS, O. A. (1993). *Three-dimensional computer vision: a geometric viewpoint*. MIT press.

Feng, J., Feng, X., Chen, J., Cao, X., Zhang, X., Jiao, L., & Yu, T. (2020). Generative adversarial networks based on collaborative learning and attention mechanism for hyperspectral image classification. *Remote Sensing*, 12(7), 1149.

Ferguson, M., Ak, R., Lee, Y.-T. T., & Law, K. H. (2017). Automatic localization of casting defects with convolutional neural networks. In *2017 IEEE international conference on big data (big data)* (pp. 1726–1735).: IEEE.

Fu, Z., Robles-Kelly, A., & Zhou, J. (2010). Mixing linear SVMs for nonlinear classification. *IEEE Transactions on Neural Networks*, 21(12), 1963–1975.

Fundisi, E., Tesfamichael, S. G., & Ahmed, F. (2021). A bi-seasonal classification of woody plant species using sentinel-2a and spot-6 in a localised species-rich savanna environment. *Geocarto International*, (pp. 1–22).

Gade, K., Geyik, S. C., Kenthapadi, K., Mithal, V., & Taly, A. (2019). Explainable AI in industry. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 3203–3204).

Gallagher, B., Rever, M., Loveland, D., Mundhenk, T. N., Beauchamp, B., Robertson, E., Jaman, G. G., Hiszpanski, A. M., & Han, T. Y.-J. (2020). Predicting compressive strength of consolidated molecular solids using computer vision and deep learning. *Materials & Design*, 190, 108541.

Geetharamani, G. & Pandian, A. (2019). Identification of plant leaf diseases using a nine-layer deep convolutional neural network. *Computers & Electrical Engineering*, 76, 323–338.

Gentleman, R. & Carey, V. J. (2008). Unsupervised machine learning. In *Bioconductor case studies* (pp. 137–157). Springer.

Ghosal, S. & Sarkar, K. (2020). Rice leaf diseases classification using CNN with transfer learning. In *2020 IEEE Calcutta Conference (CALCON)* (pp. 230–236).: IEEE.

Ghozia, A., Attiya, G., Adly, E., & El-Fishawy, N. (2020). Intelligence is beyond learning: A context-aware artificial intelligent system for video understanding. *Computational Intelligence and Neuroscience*, 2020.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11), 139–144.

Guan, Q., Wang, Y., Ping, B., Li, D., Du, J., Qin, Y., Lu, H., Wan, X., & Xiang, J. (2019). Deep convolutional neural network vgg-16 model for differential diagnosing of papillary thyroid carcinomas in cytological images: a pilot study. *Journal of Cancer*, 10(20), 4876.

Gunn, S. R. et al. (1998). Support vector machines for classification and regression. *ISIS Technical Report*, 14(1), 5–16.

Gunning, D. & Aha, D. (2019). Darpa's explainable artificial intelligence (xai) program. *AI magazine*, 40(2), 44–58.

Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). KNN model-based approach in classification. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (pp. 986–996).: Springer.

Guo, X., Yin, Y., Dong, C., Yang, G., & Zhou, G. (2008). On the class imbalance problem. In *2008 Fourth international conference on natural computation*, volume 4 (pp. 192–201).: IEEE.

Han, J. & Moraga, C. (1995). The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International workshop on artificial neural networks* (pp. 195–201).: Springer.

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems*, 28.

Har-Peled, S., Roth, D., & Zimak, D. (2003). Constraint classification for multiclass classification and ranking. *Advances in Neural Information Processing Systems*, (pp. 809–816).

Hassan, S. M., Maji, A. K., Jasiński, M., Leonowicz, Z., & Jasińska, E. (2021). Identification of plant-leaf diseases using CNN and transfer-learning approach. *Electronics*, 10(12), 1388.

Hegland, M. (2001). Data mining techniques. *Acta Numerica*, 10, 313–355.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Klambauer, G., & Hochreiter, S. (2017). GANs trained by a two time-scale update rule converge to a nash equilibrium.

Hoang, N.-D. & Nguyen, Q.-L. (2018). Metaheuristic optimized edge detection for recognition of concrete wall cracks: a comparative study on the performances of roberts, prewitt, canny, and sobel algorithms. *Advances in Civil Engineering*, 2018.

Hoffman, R. R., Mueller, S. T., Klein, G., & Litman, J. (2018). Metrics for explainable AI: Challenges and prospects. *arXiv preprint arXiv:1812.04608*.

Holzinger, A., Biemann, C., Pattichis, C. S., & Kell, D. B. (2017). What do we need to build explainable AI systems for the medical domain? *arXiv preprint arXiv:1712.09923*.

Hosch, W. L. (2019). Machine learning. https://www.britannica.com/technology/machine-learning/. Accessed: 2022-02-04.

Huixian, J. (2020). The analysis of plants image recognition based on deep learning and artificial neural network. *IEEE Access*, 8, 68828–68841.

Hussain, M., Bird, J. J., & Faria, D. R. (2018). A study on CNN transfer learning for image classification. In *UK Workshop on computational Intelligence* (pp. 191–202).: Springer.

Ian, G., Pouget-Abadie, J., Mirza, M., Xu, B., & Warde-Farley, D. (2014). Generative adversarial nets. *In Advances in Neural Information Processing Systems*.

IndianTechWarrior (2021). Machine learning, fully connected layers in convolutional neural networks. https://indiantechwarrior.com/fully-connected-layers-in-convolutional-neural-networks/. Accessed: 2022-06-14.

Inoue, T. & Abe, S. (2001). Fuzzy support vector machines for pattern classification. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, volume 2 (pp. 1449–1454).: IEEE.

Islam, S., Khan, S. I. A., Abedin, M. M., Habibullah, K. M., & Das, A. K. (2019). Bird species classification from an image using vgg-16 network. In *Proceedings of the 2019 7th International Conference on Computer and Communications Management* (pp. 38–42).

Jarvis, R. A. (1983). A perspective on range finding techniques for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2), 122–139.

Jeon, W.-S. & Rhee, S.-Y. (2017). Plant leaf recognition using a convolution neural network. *International Journal of Fuzzy Logic and Intelligent Systems*, 17(1), 26–34.

Kandel, I., Castelli, M., & Popovič, A. (2020). Comparative study of first order optimizers for image classification using convolutional neural networks on histopathology images. *Journal of Imaging*, 6(9), 92.

Ke, Q., Liu, J., Bennamoun, M., An, S., Sohel, F., & Boussaid, F. (2018). Chapter 5 - computer vision for human–machine interaction. In M. Leo & G. M. Farinella (Eds.), *Computer Vision for Assistive Healthcare*, Computer Vision and Pattern Recognition (pp. 127–145). Academic Press.

Keshari, R., Vatsa, M., Singh, R., & Noore, A. (2018). Learning structure and strength of CNN filters for small sample size training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 9349–9358).

Khaki, S., Wang, L., & Archontoulis, S. V. (2020). A CNN-RNN framework for crop yield prediction. *Frontiers in Plant Science*, 10, 1750.

Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial intelligence review*, 53(8), 5455–5516.

Khirirat, S., Feyzmahdavian, H. R., & Johansson, M. (2017). Mini-batch gradient descent: Faster convergence under data sparsity. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)* (pp. 2880–2887).: IEEE.

Kho, S. J., Manickam, S., Malek, S., Mosleh, M., & Dhillon, S. K. (2017). Automated plant identification using artificial neural network and support vector machine. *Frontiers in Life Science*, 10(1), 98–107.

Kim, J.-Y. & Cho, S.-B. (2021). Deep CNN transferred from vae and gan for classifying irritating noise in automobile. *Neurocomputing*, 452, 395–403.

Kouretas, I. & Paliouras, V. (2019). Simplified hardware implementation of the softmax activation function. In *2019 8th international conference on modern circuits and systems technologies (MOCAST)* (pp. 1–4).: IEEE.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 1097–1105.

LeCun, Y., Bengio, Y., & Hinton, G. (2015a). Deep learning. *Nature*, 521(7553), 436–444.

LeCun, Y. et al. (2015b). Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 20(5), 14.

Lee, D., Lee, S.-J., & Seo, Y.-J. (2020). Application of recent developments in deep learning to ann-based automatic berthing systems. *Int. j. Eng. Technol. Innov*, 10(1), 75–90.

Lee, J., Choi, H., Park, D., Chung, Y., Kim, H.-Y., & Yoon, S. (2016). Fault detection and diagnosis of railway point machines by sound analysis. *Sensors*, 16(4), 549.

Lee, J., Haralick, R., & Shapiro, L. (1987). Morphologic edge detection. *IEEE Journal on Robotics and Automation*, 3(2), 142–156.

Lee, S. H., Chan, C. S., Mayo, S. J., & Remagnino, P. (2017). How deep learning extracts and learns leaf features for plant classification. *Pattern Recognition*, 71, 1–13.

Leung, H. K., Chen, X.-Z., Yu, C.-W., Liang, H.-Y., Wu, J.-Y., & Chen, Y.-L. (2019). A deep-learning-based vehicle detection approach for insufficient and nighttime illumination conditions. *Applied Sciences*, 9(22), 4769.

Li, D., Deng, L., Gupta, B. B., Wang, H., & Choi, C. (2019). A novel CNN based security guaranteed image watermarking generation scenario for smart city applications. *Information Sciences*, 479, 432–447.

Li, F.-F. (2022). Machine learning, deep learning for computer vision. https://cs231n.github.io/convolutional-networks/. Accessed: 2022-06-14.

Li, Q., Cai, W., Wang, X., Zhou, Y., Feng, D. D., & Chen, M. (2014). Medical image classification with convolutional neural network. In *2014 13th international conference on control automation robotics & vision (ICARCV)* (pp. 844–848).: IEEE.

Li, Y., Nie, J., & Chao, X. (2020). Do we really need deep CNN for plant diseases identification? *Computers and Electronics in Agriculture*, 178, 105803.

Lin, H.-T. & Lin, C.-J. (2003). A study on sigmoid kernels for SVM and the training of non-PSD kernels by smo-type methods. *Neural Comput*, 3(1-32), 16.

Liu, B. (2011). Supervised learning. In *Web Data Mining* (pp. 63–132). Springer.

Liu, K., Abdullah, A. A., Huang, M., Nishioka, T., Altaf-Ul-Amin, M., & Kanaya, S. (2017). Novel approach to classify plants based on metabolite-content similarity. *BioMed research international*, 2017.

Liu, S. & Deng, W. (2015). Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)* (pp. 730–734).: IEEE.

Liu, X., Xie, L., Wang, Y., Zou, J., Xiong, J., Ying, Z., & Vasilakos, A. V. (2020). Privacy and security issues in deep learning: A survey. *IEEE Access*, 9, 4566–4593.

Luca, Z. (2021). Iucn, international union for conservation of nature. https://www.iucn.org/. Accessed: 2022-02-03.

Ma, S., Zhou, H., & Zha, H. (2021). Mean field game gan. *arXiv e-prints*, (pp. arXiv–2103).

Machhour, A., Zouhri, A., El Mallahi, M., Lakhliai, Z., Tahiri, A., & Chenouni, D. (2020). Plants classification using neural shifted legendre-fourier moments. In *Advances in Smart Technologies Applications and Case Studies: Selected Papers from the First International Conference on Smart Information and Communication Technologies, SmartICT 2019, September 26-28, 2019, Saidia, Morocco* (pp. 149–153).: Springer.

Mahesh, B. (2020). Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9, 381–386.

Maimon, O. & Rokach, L. (2005). Decomposition methodology for knowledge discovery and data mining. In *Data Mining and Knowledge Discovery Handbook* (pp. 981–1003). Springer.

Manevitz, L. M. & Yousef, M. (2001). One-class SVMs for document classification. *Journal of Machine Learning Research*, 2(Dec), 139–154.

Mansour, Y. (1997). Pessimistic decision tree pruning based on tree size. In *Machine Learning -International Workshop then Conference-* (pp. 195–201).: Citeseer.

Marcus, G. (2018). Deep learning: A critical appraisal. *arXiv e-prints*, (pp. arXiv–1801).

Markovsky, I. & Usevich, K. (2012). *Low rank approximation*, volume 139. Springer.

Martens, J. et al. (2010). Deep learning via hessian-free optimization. In *ICML*, volume 27 (pp. 735–742).

Mavroforakis, M. E. & Theodoridis, S. (2006). A geometric approach to support vector machine (SVM) classification. *IEEE transactions on neural networks*, 17(3), 671–682.

Meer, P. (2004). Robust techniques for computer vision. *Emerging topics in computer vision*, (pp. 107–190).

Michie, D., Spiegelhalter, D. J., Taylor, C., et al. (1994). Machine learning. *Neural and Statistical Classification*, 13(1994), 1–298.

Mingers, J. (1989). An empirical comparison of pruning methods for decision tree induction. *Machine learning*, 4(2), 227–243.

Mitchell, T. M. et al. (2007). *Machine learning*, volume 1. McGraw-hill New York.

Mohamed, W. N. H. W., Salleh, M. N. M., & Omar, A. H. (2012). A comparative study of reduced error pruning method in decision tree algorithms. In *2012 IEEE International conference on control system, computing and engineering* (pp. 392–397).: IEEE.

Moll, E. J. (1983). Trees of southern africa.

Moreno-Torres, J. G., Sáez, J. A., & Herrera, F. (2012). Study on the impact of partition-induced dataset shift on k-fold cross-validation. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8), 1304–1312.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Nadernejad, E., Sharifzadeh, S., & Hassanpour, H. (2008). Edge detection techniques: Evaluations and comparisons. *Applied Mathematical Sciences*, 2(31), 1507–1520.

Nandutu, I., Atemkeng, M., & Okouma, P. (2021). Integrating AI ethics in wildlife conservation AI systems in south africa: A review, challenges, and future research agenda. *AI & SOCIETY*, (pp. 1–13).

Ng, A. (2000). Cs229 lecture notes. *CS229 Lecture Notes*, 1(1), 1–3.

Noble, W. S. (2006). What is a support vector machine? *Nature biotechnology*, 24(12), 1565–1567.

Nyanchama, O. R. N. (2021). *Assessing the Abundance and Nesting Behaviour of Golden-Rumped Elephant-Shrews (GRES)(Rhynchocyon chrysopygus) In Arabuko-Sokoke Forest, Kilifi Country, Kenya*. PhD thesis, Pwani University.

Pawara, P., Okafor, E., Schomaker, L., & Wiering, M. (2017). Data augmentation for plant classification. In *International Conference on Advanced Concepts for Intelligent Vision Systems* (pp. 615–626).: Springer.

Phansalkar, V. & Sastry, P. (1994). Analysis of the back-propagation algorithm with momentum. *IEEE Transactions on Neural Networks*, 5(3), 505–506.

PK, F. A. (1984). What is artificial intelligence? *"Success is no accident. It is hard work, perseverance, learning, studying, sacrifice and most of all, love of what you are doing or learning to do"*., (pp.6̃5).

Platt, J. C., Cristianini, N., Shawe-Taylor, J., et al. (1999). Large margin dags for multiclass classification. In *nips*, volume 12 (pp. 547–553).

Podgorelec, V., Kokol, P., Stiglic, B., & Rozman, I. (2002). Decision trees: an overview and their use in medicine. *Journal of Medical Systems*, 26(5), 445–463.

Pratiwi, H., Windarto, A. P., Susliansyah, S., Aria, R. R., Susilowati, S., Rahayu, L. K., Fitriani, Y., Merdekawati, A., & Rahadjeng, I. R. (2020). Sigmoid activation function in selecting the best model of artificial neural networks. In *Journal of Physics: Conference Series*, volume 1471 (pp. 012010).: IOP Publishing.

Pujari, A. K. (2001). *Data mining techniques*. Universities press.

Qassim, H., Verma, A., & Feinzimer, D. (2018). Compressed residual-VGG16 CNN model for big data places image recognition. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 169–175).: IEEE.

Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3), 221–234.

Rai, A. (2020). Explainable AI: From black box to glass box. *Journal of the Academy of Marketing Science*, 48(1), 137–141.

Rathod, J., Waghmode, V., Sodha, A., & Bhavathankar, P. (2018). Diagnosis of skin diseases using convolutional neural networks. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)* (pp. 1048–1051).: IEEE.

Rennie, J. D. & Rifkin, R. (2001). Improving multiclass text classification with the support vector machine.

Rifkin, R. (2008). Multiclass classification. *Lecture Slides. February*.

Rokach, L. & Maimon, O. (2005). Decision trees. In *Data mining and Knowledge Discovery Handbook* (pp. 165–192). Springer.

Ruby, U. & Yendapalli, V. (2020). Binary cross entropy with deep learning technique for image classification. *Int. J. Adv. Trends Comput. Sci. Eng*, 9(10).

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088), 533–536.

Santosa, B. (2009). Application of the cross-entropy method to dual lagrange support vector machine. In *International Conference on Advanced Data Mining And Applications* (pp. 595–602).: Springer.

Schmidt-Hieber, J. (2020). Nonparametric regression using deep neural networks with relu activation function. *The Annals of Statistics*, 48(4), 1875–1897.

Schuldt, C., Laptev, I., & Caputo, B. (2004). Recognizing human actions: a local SVM approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3 (pp. 32–36).: IEEE.

scikit-learn developers (2021). TechTalks multiclass and multioutput algorithms. `https://scikit-learn.org/stable/modules/multiclass.html`. Accessed: 2021-08-28.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E., Menon, V. K., & Soman, K. (2017). Stock price prediction using LSTM, RNN and CNN-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)* (pp. 1643–1647).: IEEE.

Shaikh, N. A., Mallah, G. A., Karaş, İ. R., & Akay, A. E. (2018). Using mobile image recognition system for nonwood species identification in the field. *Journal of Innovative Science and Engineering*, 2(2), 40–50.

Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. *towards data science*, 6(12), 310–316.

Sheik, S., MOHAMMED, R., Teeparthi, K., & Raghuvamsi, Y. (2007). Prediction of sensitization degree for conventional nickel based and high nitrogen austenitic stainless steels: A machine learning approach. *Available at SSRN 4382762*.

Shin, H.-C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., & Summers, R. M. (2016). Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging*, 35(5), 1285–1298.

Shouman, M., Turner, T., & Stocker, R. (2011). Using decision tree for diagnosing heart disease patients. In *Proceedings of the Ninth Australasian Data Mining Conference*, volume 121 (pp. 23–30).

Singh, P., Raj, P., & Namboodiri, V. P. (2020a). Eds pooling layer. *Image and Vision Computing*, 98, 103–923.

Singh, S. A., Meitei, T. G., & Majumder, S. (2020b). Short pcg classification based on deep learning. In *Deep Learning Techniques for Biomedical and Health Informatics* (pp. 141–164). Elsevier.

Sitaula, C. & Hossain, M. B. (2021). Attention-based vgg-16 model for covid-19 chest x-ray image classification. *Applied Intelligence*, 51(5), 2850–2863.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 1929–1958.

Stone, J. V. (2019). *Artificial intelligence engines: A tutorial introduction to the mathematics of Deep Learning*. Sebtel Press.

Sun, Y., Liu, Y., Wang, G., & Zhang, H. (2017). Deep learning for plant identification in natural environment. *Computational Intelligence and Neuroscience*, 2017.

Suthaharan, S. (2016). Support vector machine. In *Machine Learning Models and Algorithms for Big Data Classification* (pp. 207–235). Springer.

Sutskever, I. & Nair, V. (2008). Mimicking go experts with convolutional neural networks. In *International Conference on Artificial Neural Networks* (pp. 101–110).: Springer.

Swain, M., Dash, S. K., Dash, S., & Mohapatra, A. (2012). An approach for iris plant classification using neural network. *International Journal on Soft Computing*, 3(1), 79.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–9).

Szegedy, C., Toshev, A., & Erhan, D. (2013). Deep neural networks for object detection. *Advances in neural information processing systems*, 26.

Szepesvári, C. (2010). Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1), 1–103.

Tammina, S. (2019). Transfer learning using vgg-16 with deep convolutional neural network for classifying images. *International Journal of Scientific and Research Publications (IJSRP)*, 9(10), 143–150.

Tamuly, S., Jyotsna, C., & Amudha, J. (2019). Deep learning model for image classification. In *International Conference On Computational Vision and Bio Inspired Computing* (pp. 312–320).: Springer.

Tangirala, S. (2020). Evaluating the impact of gini index and information gain on classification using decision tree classifier algorithm. *International Journal of Advanced Computer Science and Applications*, 11(2), 612–619.

Tato, A. & Nkambou, R. (2018). Improving adam optimizer.

Unzueta, D. (2021). Deep learning fundamentals, convolutional layers vs fully connected layers. https://towardsdatascience.com/convolutional-layers-vs-fully-connected-layers-364f05ab460b. Accessed: 2022-06-14.

Van Hieu, N. & Hien, N. L. H. (2020). Automatic plant image identification of vietnamese species using deep learning models. *arXiv e-prints*, (pp. arXiv–2005).

Venter, F. & J, V. (2007). *Benut ons Inheemse Bome*, volume 2. Briza.

Voloshynovskiy, S., Koval, O., Beekhof, F., & Holotyak, T. (2009). Multiclass classification based on binary classifiers: On coding matrix design, reliability and maximum number of classes. In *2009 IEEE International Workshop on Machine Learning for Signal Processing* (pp. 1–6).: IEEE.

Vorobeychik, Y. & Kantarcioglu, M. (2018). Adversarial machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3), 1–169.

Wagle, S. A., Harikrishnan, R., Ali, S. H. M., & Faseehuddin, M. (2022). Classification of plant leaves using new compact convolutional neural network models. *Plants*, 11(1), 24.

Wäldchen, J. & Mäder, P. (2018). Plant species identification using computer vision techniques: A systematic literature review. *Archives of Computational Methods in Engineering*, 25(2), 507–543.

Wäldchen, J., Rzanny, M., Seeland, M., & Mäder, P. (2018). Automated plant species identification—trends and future directions. *PLoS Computational Biology*, 14(4), e1005993.

Wang, Y. (2020). A mathematical introduction to generative adversarial nets (gan). *arXiv e-prints*, (pp. arXiv–2009).

Watt, J. & Breyer-Brandwijk, M. (1962). Medicinal and poisonous plants of southern and eastern africa (edinburgh: E. & s. livingstone).

Wei, J. (2019). Vgg neural networks: The next step after alexnet. `https://towardsdatascience.com/vgg-neural-networks-the-next-step-after-alexnet-3f91fa9ffe2c`. Accessed: 2022-10-25.

Wiatrak, M., Albrecht, S. V., & Nystrom, A. (2019). Stabilizing generative adversarial networks: A survey. *arXiv e-prints*, (pp. arXiv–1910).

Wijianto, W., Dwi, C., & Nurul, Q. (2020). (peer review+ similarity+ document) how to improve employee performance at the forest service.

Wilson, A. (2019). Sigmoid and tanh activation functions. `https://a-i-dan.github.io/math_nn`. Accessed: 2022-10-25.

Winston, P. H. (1984). *Artificial intelligence*. Addison-Wesley Longman Publishing Co., Inc.

Winston, P. H. (1992). *Artificial intelligence*. Addison-Wesley Longman Publishing Co., Inc.

Wittman, T. (2010). An introduction to mathematical image processing ias, park city mathematics institute, utah undergraduate summer school 2010.

Xu, F., Uszkoreit, H., Du, Y., Fan, W., Zhao, D., & Zhu, J. (2019). Explainable AI: A brief survey on history, research areas, approaches and challenges. In *CCF international conference on natural language processing and Chinese computing* (pp. 563–574).: Springer.

Xue, M., Yuan, C., Wu, H., Zhang, Y., & Liu, W. (2020). Machine learning security: Threats, countermeasures, and evaluations. *IEEE Access*, 8, 74720–74742.

Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9(4), 611–629.

Yang, B., Guo, H., & Cao, E. (2021). Design of cyber-physical-social systems with forensic-awareness based on deep learning. In *Advances in Computers*, volume 120 (pp. 39–79). Elsevier.

Yang, X., Zhang, G., Lu, J., & Ma, J. (2010). A kernel fuzzy c-means clustering-based fuzzy support vector machine algorithm for classification problems with outliers or noises. *IEEE Transactions on Fuzzy Systems*, 19(1), 105–115.

Yu, D., Wang, H., Chen, P., & Wei, Z. (2014). Mixed pooling for convolutional neural networks. In *International conference on rough sets and knowledge technology* (pp. 364–375).: Springer.

Yu, H. & Kim, S. (2012). SVM tutorial-classification, regression and ranking. *Handbook of Natural Computing*, 1, 479–506.

Yue, S., Li, P., & Hao, P. (2003). SVM classification: Its contents and challenges. *Applied Mathematics-A Journal of Chinese Universities*, 18(3), 332–342.

Zeiler, M. D. & Fergus, R. (2013). Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*.

Zhang, H., Hong, X.-g., & Zhu, L. (2021). Detecting small objects in thermal images using single-shot detector. *Automatic Control and Computer Sciences*, 55(2), 202–211.

Zhang, Q., Bai, C., Liu, Z., Yang, L. T., Yu, H., Zhao, J., & Yuan, H. (2020). A gpu-based residual network for medical image classification in smart medicine. *Information Sciences*, 536, 91–100.

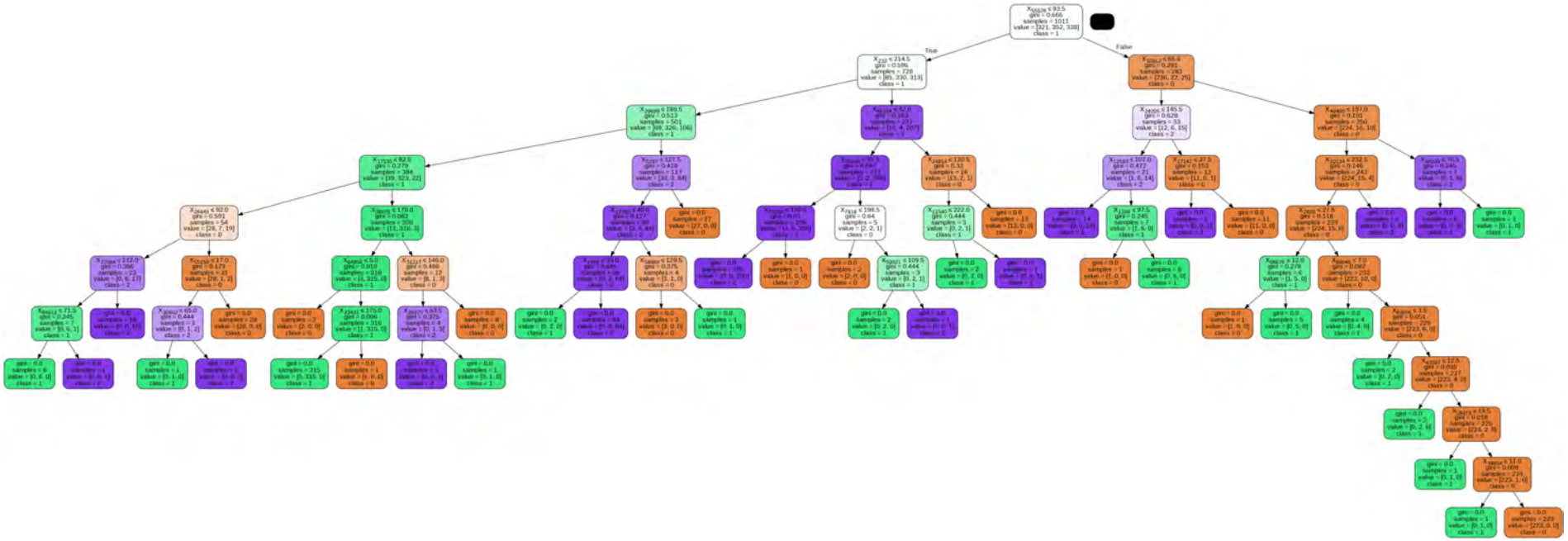


Figure 1: Unpruned Decision Tree Graph

Appendix 1

Appendix 2

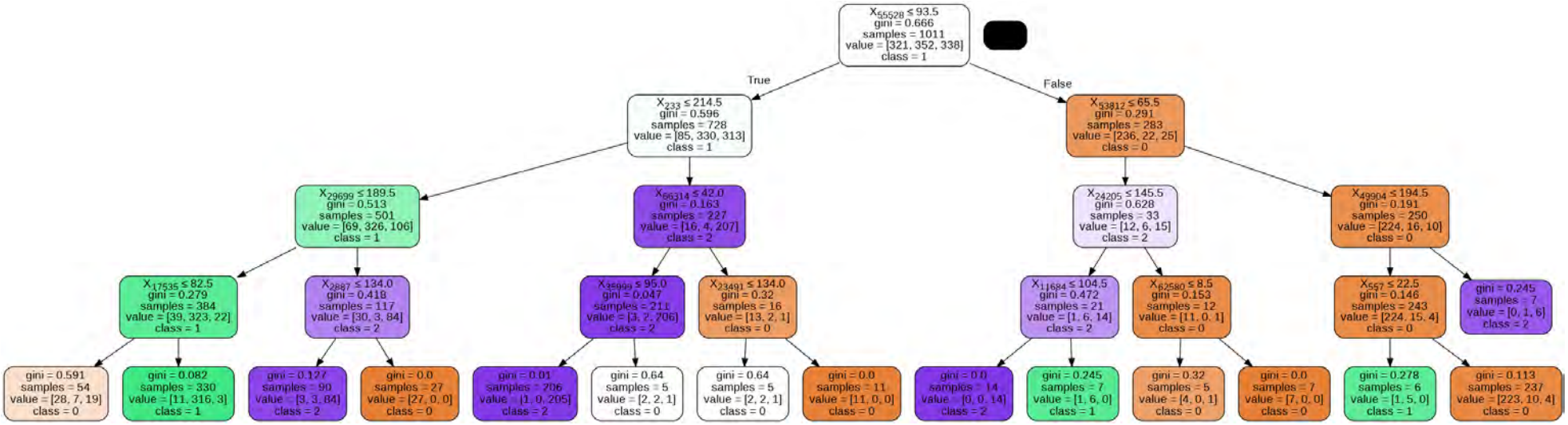


Figure 2: Pruned Decision Tree Graph