An Investigation into Computer Vision Methods to Identify Material Other than Grapes in Harvested Red Wine Grape Loads

by
Riaan Kleyn

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Engineering Management) in the Faculty of Engineering at Stellenbosch University.
This thesis has also been presented at Reutlingen University in terms of a double-degree agreement.

Supervisor: Mr. Konrad Von Leipzig
Co-supervisor: Prof. Daniel Palm

December 2023

Declaration

By submitting this project electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December, 2023

Abstract

Mass wine production companies across the globe are provided with grapes from winegrowers that predominantly utilise mechanical harvesting machines to harvest wine grapes. Mechanical harvesting accelerates the rate at which grapes are harvested, allowing grapes to be delivered faster to meet the demands of wine cellars. The disadvantage of the mechanical harvesting method is the inclusion of material-other-than-grapes (MOG) in the harvested wine grape loads arriving at the cellar. The MOG degrades the quality of wine that can be produced while it is also costly to transport and discard and it can cause machine downtime.

This paper seeks to find an optimal computer vision method capable of detecting the amount of MOG within a wine grape load. A MOG detection method will encourage winegrowers to deliver MOG-free wine grape loads to avoid penalties which will indirectly enhance the quality of the wine to be produced. Traditional image segmentation methods were compared to deep learning segmentation methods based on images of wine grape loads that were captured at a wine cellar. The Mask R-CNN model with a ResNet-50 convolutional neural network backbone emerged as the optimal method for this study to determine the amount of MOG in an image of a wine grape load. Furthermore, a statistical analysis was conducted to determine how the MOG on the surface of a grape load relates to the mass of MOG within the corresponding grape load.

Opsomming

Massawynproduksie maatskappye regoor die wêreld word voorsien van druiwe van wynboere wat hoofsaaklik meganiese oesmasjiene gebruik om wyndruiwe te oes. Meganiese oesmetodes versnel die tempo waarteen druiwe geoes word sodat druiwe vinniger aan wynkelders gelewer kan word om aan hul vereistes te voldoen. Die nadeel van die meganiese oesmetode is die insluiting van materiaal anders as druiwe (MOG) in die geoesde wyndruif vragte wat by die kelder aankom. Die MOG verswak die kwaliteit van wyn wat geproduseer kan word, terwyl dit ook duur is om te vervoer en weg te gooi en dit kan veroorsaak dat die masjiene tot stilstand kom.

Hierdie navorsing poog om 'n optimale rekenaarvisiemetode te vind wat die hoeveelheid MOG binne 'n vrag wyndruiwe kan identifiseer. 'n MOG-identifiseringsmetode sal wynboere aanmoedig om MOG-vrye wyndruifvragte aan wynkelders te lewer om sodoende boetes te vermy wat indirek die kwaliteit van die wyn wat geproduseer kan word, sal verbeter. Tradisionele beeldsegmenteringsmetodes is vergelyk met diepleersegmenteringsmetodes gebaseer op beelde van wyndruifvragte wat by 'n wynkelder vasgevang is. Die Mask R-CNN model met 'n ResNet-50 konvolusionele neurale netwerk ruggraat het na vore gekom as die optimale metode vir hierdie studie om die hoeveelheid MOG in 'n beeld van 'n wyndruiflading te bepaal. Verder is 'n statistiese analise uitgevoer om te bepaal hoe die MOG op die oppervlak van 'n druiwelading verband hou met die massa van die MOG binne in die ooreenstemmende druiflading.

Acknowledgements

The author wishes to acknowledge the following people and institutions for their various contributions towards the completion of this work:

- My supervisor at Stellenbosch University, Mr. Konrad Von Leipzig, for his help and guidance throughout the project. It was a pleasure working with you.

- My co-supervisor at Reutlingen University, Prof. Daniel Palm for his valuable input during the colloquiums.

- My family for their motivation and continuous support.

Table of Contents

List of Acronyms

ANN: Artificial Neural Network

AWRI: Australian Wine Research Institute

CI: Confidence Interval

CLT: Central Limit Theorem

CNN: Convolutional Neural Network

CPU: Central Processing Unit

CVAT: Computer Vision Annotation Tool

FN: False Negative

FP: False Positive

GPU: Graphical Processing Unit

HSV: Hue Saturation Value

IoU: Intersection over Union

LReLU: Linear Rectifier Linear Units

MOG: Material-Other-than-Grapes

R-CNN: Region-based Convolutional Neural Network

ReLU: Rectifier Linear Units

RGB: Red Green Blue

ROI: Region of Interest

RPN: Region Proposal Network

RGB: Red Green Blue

SVM: Support Vector Machine

TN: True Negative

TP: True Positive

YOLO: You-Only-Look-Once

List of Figures

List of Tables

CHAPTER 1

Introduction

This chapter serves the purpose of presenting an overview of the research undertaken in this project. The establishment of a background lays the foundation upon which the research is predicated, subsequently leading to the formulation of a research problem. The chapter proceeds to outline the project objectives to address the research problem. The research contribution, research design, and research scope are subsequently discussed. Finally, this chapter concludes with the research methodology and organisation of the report.

1.1 Background

"Today, praise be to God, wine was pressed for the first time from Cape grapes." - Jan van Riebeeck [116]

The historical roots of South Africa's wine tradition date back to the year 1655 when the Cape's first governor, Jan van Riebeeck, laid the foundation for vine cultivation by planting a vineyard. Four years later, the first wine was successfully made from grapes harvested in the Cape, leading to more extensive vine planting in the region. Today, South Africa plays a pivotal role in the worldwide wine industry. [116].

According to a recent study, South Africa ranks as the eighth largest producer of wine globally. The country harvested 973.6 million litres from 2778 vineyards and 92005 hectares of vines in 2019 with 86% of that devoted to winemaking, 9.7% to distilling wine, 4% to brandy and 0.3%

to grape juice. The volume of wine exported was 319.8 million litres which is 32.8% of the total wine produced [120].

The wine industry's supply chain network typically consists of winegrowers who harvest and supply grapes to the winemakers working in the wine cellars [2]. Winemakers utilise grapes for the purpose of producing either wine or juice. Subsequently, the wine and juice may either be retailed in large quantities or undergo bottling procedures before being distributed to different markets in smaller quantities. Both the bulk and bottled wine can either be sold to the local market or the export market. Figure 1.1 illustrates a diagram of the supply chain for the local wine market subdivided into three distinct phases. The supply of raw materials, dry goods and harvested wine grape loads from winegrowers form part of the *source* process. Wine production in cellars and bottling forms part of the *make* process. The process of wine production entails grape crushing and pressing, followed by the transformation of sugar into alcohol through fermentation. Subsequently, the wine is subjected to clarification procedures, which aim to extract insoluble particles from the wine, and finally, it undergoes the ageing process. Following ageing or clarification, the wine is distributed to diverse markets, thereby forming part of the *delivery* process aimed at ultimately reaching the end consumer.

The wine supply chain of mass wine production companies is driven by the quantity of grapes winegrowers are able to supply to cellars. These winegrowers predominantly use mechanical harvesting machines as shown in figure 1.2 to harvest wine grapes. Mechanical harvesting accelerates the rate at which grapes are harvested, allowing grapes to be delivered faster to meet the demands of wine cellars. The disadvantage of mechanical harvesting is the inclusion of an unwanted byproduct known as MOG (Material-Other-than-Grapes) with the harvested wine grapes. MOG consists of leaves, canes, and wood, and it has a negative impact on wine quality and value by emitting an unpleasant bitter aroma and green characteristics. MOG is also expensive to transport, extract, and discard, and it can cause machine downtime, eventually leading to a bottleneck at the crushers as production systems slow down to accommodate grape destemming and sorting. Thus, MOG is not something that wine cellars want, and it certainly is not something that winemakers can use to make wine.

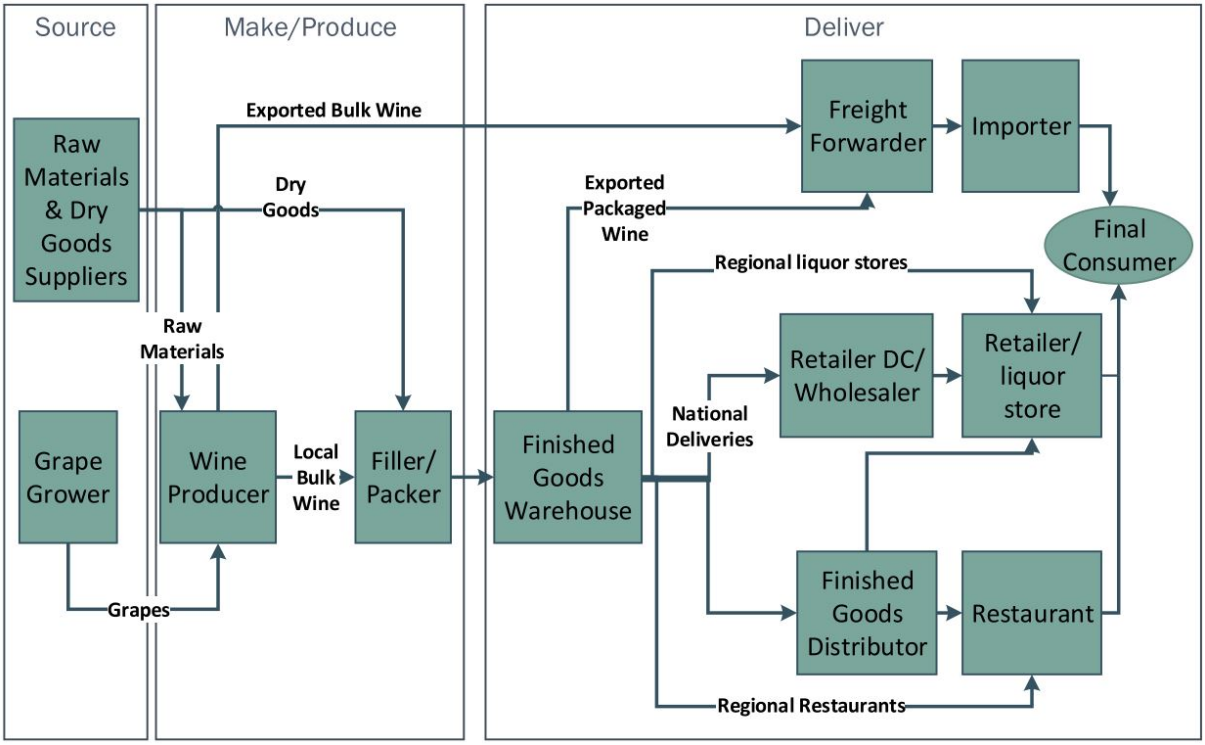

FIGURE 1.1: *South African local wine supply chain [54]*

FIGURE 1.2: *A mechanical harvesting machine in the vineyard [114].*

Wine grapes are harvested and collected in a steel container mounted on top of a trailer, which is usually drawn by a tractor or a truck. When the container is fully loaded with wine grapes, the grape loads are delivered to the wine cellars as soon as possible. Before the wine grapes may be unloaded into a crushing machine, the tractor with the wine grape load should first report to the weighbridge which is an intermediate stage between the source and make processes of the supply chain as illustrated in figure 1.3. The following tasks are performed at the weighbridge:

- A sample is collected to determine the sugar and pH levels,
- then the wine grape load is weighed.

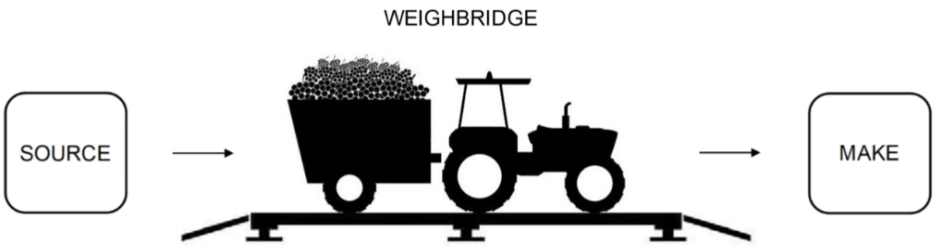

FIGURE 1.3: *Weighbridge between the source and make process.*

In cases where a winegrower consistently supplies grape loads with inadequate sugar and pH levels, it is recommended that they delay harvest until such time that their grape loads exhibit satisfactory sugar and pH levels. The weight of the grape load is used to determine how much the wine cellar should pay the winegrower for that load. The cultivar of the wine grapes will also influence the load price. One task that appears to have been omitted from the enlisted tasks is the evaluation of the grape loads' quality with regard to the existence of MOG. This aspect of quality inspection at not only the weighbridge but also at the crusher where the grapes are unloaded is the main focus of this project.

The quality of the product offered to the commercial marketplace is significantly contingent upon the quality of the grape loads delivered by the winegrowers which are subsequently utilised to produce wine or juice. Good quality grape loads with minimal MOG will ensure a good product is produced while simultaneously lowering the costs of production. Therefore, a thorough MOG inspection of the grape loads would mitigate the production of substandard wine and substantially improve the efficiency of the wine supply chain.

The present study was motivated by the author's inclination towards the wine industry as well as a need expressed by a local mass wine production company to investigate whether there is a cost-efficient method to determine the amount of MOG present in wine grape loads. A requirement is that the quality inspection is done without human intervention because this process can become time-consuming for humans and their judgement can be biased in an environment where time,

quality, and quantity are critical factors.

A singular previous study addressing the identification of MOG in grape loads was discovered in which the Australian Wine Research Institute (AWRI) researched digital approaches for evaluating grape loads upon their delivery to the wine cellar [18]. Hyperspectral imaging techniques were employed to distinguish MOG in grape loads that had been mechanically harvested. This study revealed that the occurrence of shadows during imaging of bulk mechanically harvested grape samples led to errors in material classification due to changes in spectral profiles. Furthermore, this study has demonstrated that optimum analysis may be achieved through two laboratory-based methods. The first involves spreading the grape samples evenly on a static tray and subsequently capturing them using a snapshot camera. Alternatively, the samples may be carefully positioned on a conveyor belt and imaged using a line scan camera. One significant limitation associated with this methodology pertains to its cost, as hyperspectral imaging cameras tend to be extremely expensive. Another limitation resides in the fact that repetitive sample collection and laboratory-based imaging procedures can be labour-intensive and time-consuming.

1.2 Problem Statement

The previous section highlighted the importance of improving the quality of wine grape loads to improve the quality of wine that can be produced, while simultaneously lowering production costs. The existence of MOG within wine grape harvests has a detrimental impact on the quality of the wine that can be produced while concurrently increasing production expenses and time requirements. Currently, the price wine cellars are paying for the wine grape loads does not take the presence of MOG into consideration. Therefore they are actually paying for something they do not want.

In addressing the aforementioned concern, wine cellars require a cost-efficient system that can determine the quantity of MOG present in a load of wine grapes without the need for human intervention. The research conducted by AWRI suggested the utilisation of costly hyperspectral imaging techniques, which normally are labour and time-intensive. Furthermore, they implemented a technique that involved the utilisation of a conveyor belt, resulting in increased costs. One approach worthy of exploration is the application of computer vision algorithms utilising

a standard camera in normal conditions at the wine cellar's weighbridge and crushing machine. The utilisation of a camera positioned at the crusher subsequent to the unloading of grape loads is to serve the purpose of verifying the presence of MOG detected at the weighbridge. Therefore winegrowers are unable to merely remove the MOG from the surface of the grape load before it arrives at the weighbridge. This technology has the potential to effectively eliminate human intervention, ensuring impartial assessments of MOG present in wine grape loads. The implementation of such a MOG identification system would encourage winegrowers to make diligent efforts to deliver wine grape loads free from any MOG, consequently enhancing the quality of wine that can be produced. It will also facilitate the wine cellars to procure grapes at justifiable prices while also enabling winegrowers to receive fair compensation for their grape produce.

1.3 Project Objectives

The main objective of this project is to implement an automated and cost-efficient MOG identification system based on computer vision methods that can determine the amount of MOG within a wine grape load. To achieve this objective, the following sub-objectives have been defined:

I Conduct a literature review related to computer vision:

(a) Machine learning algorithms for image classification.

(b) Neural networks in deep learning for object detection.

(c) Image segmentation methods.

(d) Object detection and segmentation performance evaluation methods.

II Preparation of a machine-harvested grape load image data set.

III Evaluation and selection of an optimal computer vision method to detect MOG in images of wine grape loads.

IV Analyses of MOG present on the surface of the grape load related to total MOG within the grape load using statistical techniques.

V Summarise findings and recommend areas for further research based on the findings of this

report.

1.4 Research Contribution

The research conducted in this study will determine appropriate computer vision algorithms to automatically detect material other than grapes on the surface of a wine grape load. It will also determine whether a linear relationship exists between the amount of MOG on the surface of the wine grape load and the total amount of MOG within the wine grape load. This research could be useful to wine cellars to gain more knowledge about the grape loads they are buying which will indirectly add value to the product being sold to the market while simultaneously reducing the costs of production and improving the efficiency of their supply chain.

1.5 Research Design

The research design specifies the procedures for collecting, analyzing, and interpreting data in research studies [125]. The Saunders research onion, which is a well-known approach to describe the different elements in the research design, is used in this project [100]. Figure 1.4 describes the research onion for this project specifically. The research onion consists of six layers with each layer representing a more detailed step of the research process starting from the outer layer.

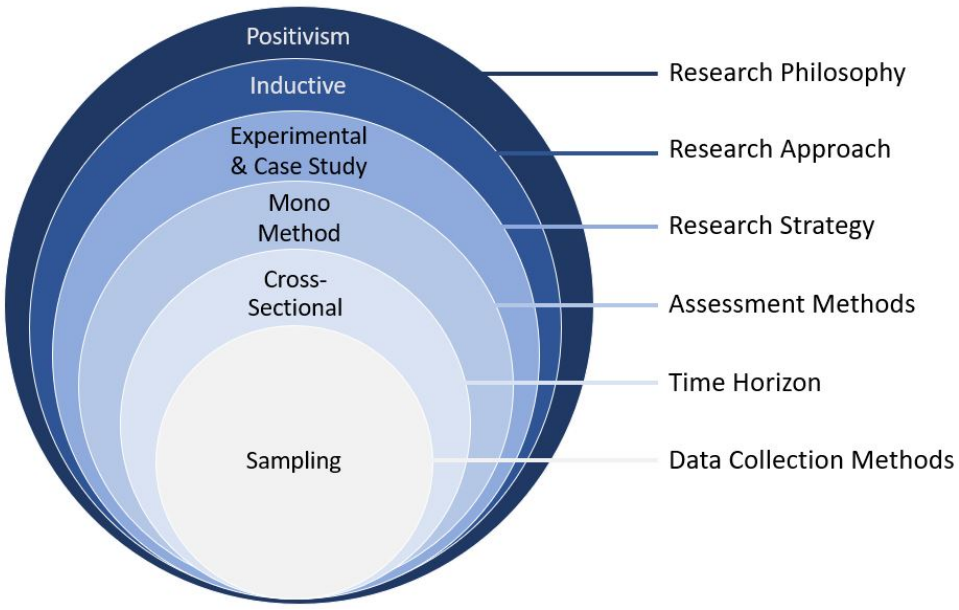

FIGURE 1.4: *The research design of this project based on Saunders' research onion*

The first layer represents the research philosophy. This layer depicts how the researcher views the world which has an influence on how the researcher will execute tasks [100]. In this project, the *positivism* research philosophy is used. The data is structured and measurable, implying that the research cannot be influenced by the author's values. MOG detection in wine grape loads is based on specific patterns and characteristics that can be measured. The second layer represents the researcher's research approach, which in this project is an *inductive* approach [99]. The author reviewed previous computer vision methods used to solve similar problems. Based on this knowledge, new methods were developed. The methods were tested, and conclusions were drawn based on the results of the analyses. The third layer of the research onion represents the research strategy. This project employs an *experimental* strategy. Experiments with various parameters were carried out to obtain information about the new methods. The fourth layer describes the assessment method used in the project. A *mono-method* is used in this project since only quantitative data is collected. Images that can be represented as matrices are collected and fed into the computer vision algorithms. The fifth layer simply specifies the time horizon for collecting data. A *cross-sectional* time horizon is used because data is only collected during the harvesting season. The core of the research onion describes data collection methods. A *Sampling* technique is used in this project to collect a quantitative data set of images. A standard camera is used to capture various images of wine grape loads.

1.6 Research Scope

The focus of this project is that the research is mainly applicable to mass wine production companies that have various wine growers using mechanical harvesting methods that supply them with wine grapes. This study investigated how effectively MOG within a wine grape load can be detected with existing computer vision algorithms. This investigation is restricted to a data set comprising of red wine grape loads. This study makes an assumption that the distribution of the volume of wine grapes and MOG in a wine grape load is uniformly distributed, owing to the consistent harvesting method employed throughout the load. Therefore, the surface of the wine grape load is a representation of the load's entire volume.

1.7 Research Methodology and Report Organisation

Apart from this introductory chapter, this report consists of five additional chapters. The remainder of the report is organised as follows:

In Chapter 2 objective I is fulfilled which documents an in-depth literature review of machine learning algorithms for image classification; the structure of neural networks in deep learning, the importance of activation functions in these structures and how convolutional neural networks are used for object detection; popular techniques used for image segmentation; and performance evaluation metrics of object detection and image segmentation algorithms.

Chapter 3 which fulfils objective II is a description of how the objects of interest within the images are annotated for algorithm training preparation.

Objective III is fulfilled in Chapter 4 where experiments with computer vision algorithms are executed and evaluated to select the best-performing algorithm. These experiments included image segmentation and deep learning methods.

Chapter 5 fulfils objective IV where a statistical analysis is executed to determine how the MOG on the surface of a grape load corresponds with the total MOG within a grape load.

Finally, objective V will be achieved in Chapter 7 by making recommendations for further research based on the findings in the report. This chapter also serves as a conclusion to the project.

1.8 Conclusion

This chapter introduced the study by providing background information related to the research being carried out. A problem statement is formulated followed by project objectives to address the problem. The research contribution and research design in the mould of the Saunders research onion are then discussed followed by the scope of this research. This chapter concludes with the research methodology and organisation of the report. The following chapter contains an in-depth literature review of computer vision techniques.

CHAPTER 2

Literature Review

This chapter reviews the pertinent computer vision literature, which serves as the theoretical foundation for this research. Computer vision consists of advanced techniques for image processing that produce consistent results and it has great potential to still improve. It has already been applied to various heterogeneous domains, including the agriculture domain [115]. Image acquisition and image processing are the two basic stages of a computer vision system. This chapter reviews the literature on image processing. Chapter 3 discusses the image acquisition process and chapter 4 describes how image processing was performed in this project. This chapter investigates the following computer vision knowledge areas: machine learning classification techniques, object detection with neural networks in deep learning and the role of various activation functions in a neural network structure, techniques for image segmentation, and finally performance evaluation metrics for object detection and image segmentation processes.

2.1 Machine Learning

Machine learning, as defined by Arthur Samuel - who was known for his checkers playing program - is a discipline that enables computers (machines) to learn without the need for explicit programming [70]. Machine learning is how the computer learns from analysing and studying training data to perform a task. The training data used to train the machine consists of examples which are described by a set of features. The trained machine learning model can be utilised to predict, classify or cluster new examples from the testing data based on the experience obtained

from the training process [68]. Figure 2.1 illustrates a typical machine learning procedure where training data is fed into the machine learning algorithm and then it can make predictions from testing data.

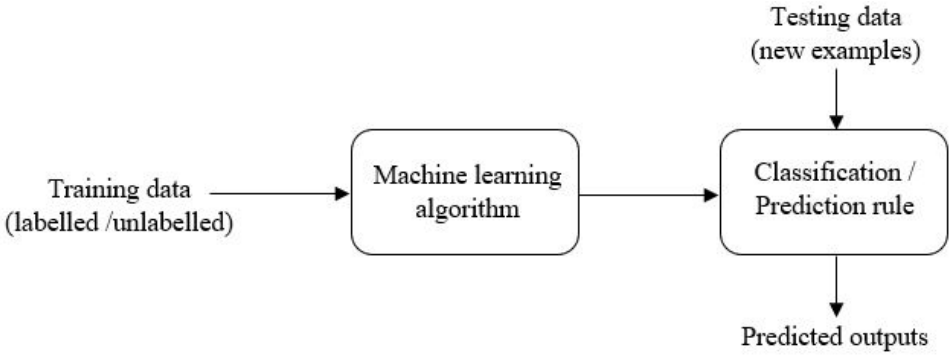

FIGURE 2.1: *A typical machine learning procedure [68].*

The major machine learning algorithms can be categorised into three distinct forms of learning namely supervised learning, unsupervised learning, and reinforcement learning. Figure 2.2 shows a breakdown of these machine learning types with machine learning algorithms categorised by the tasks they solve. A short summary of each learning type is given below.

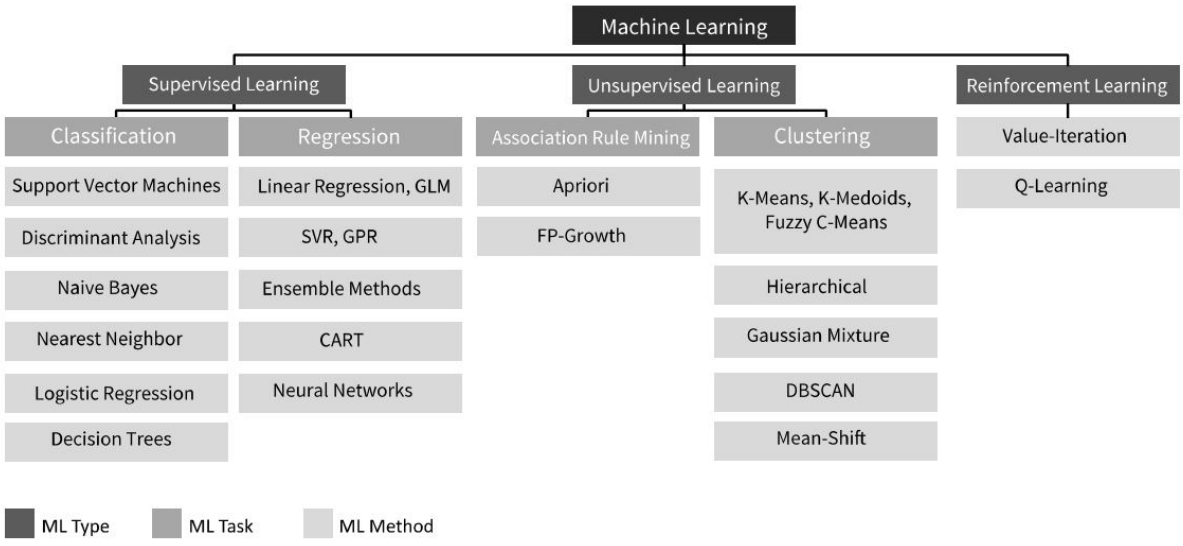

FIGURE 2.2: *A breakdown of major machine learning types and machine learning methods categorised by the tasks they solve [119].*

Supervised Learning

The most frequently employed category of machine learning is the supervised learning form.[73].

These machine learning methods are trained by feeding labelled training data into the learning algorithm. The knowledge of both the input and output allows the machine to identify and extract patterns and features from the training data set that are associated with the output. The machine will then be able to apply its newly gained knowledge of patterns and features to not previously seen data sets to make predictions or classifications [70, 119].

Unsupervised Learning

The unsupervised learning method is also called "learning without a teacher" since no correct answers are provided in this method [119]. As in supervised learning, the unsupervised learning method also uses training data to train the algorithm, however, the data is not labelled, and thus the output is not known. The method discovers knowledge of hidden patterns and features from the unlabelled data and then applies this knowledge to new input data to categorise or cluster it by common patterns and features [98].

Reinforcement Learning

The reinforcement method does not know the optimal solution from the beginning of the learning phase and it is therefore determined by an iterative approach. This method increases its performance by rewarding sensible approaches whereas wrong steps are punished. [119]. Hence, the reinforcement method can autonomously find its own optimal solutions [113].

In this project, labelled data will be needed as it will be important for the machine to predict specific MOG objects. Therefore for the rest of this project, the focus will be on supervised learning. In the following subsections, a comprehensive overview of five prevalent supervised learning techniques is presented. These include support vector machines, naive Bayes, nearest neighbour, decision trees, and neural networks.

2.1.1 Support Vector Machine (SVM)

The support vector machine is a classification technique that creates boundaries, in the form of margins on either side of a hyperplane, to segregate data into two distinct classes within a decision boundary as illustrated in figure 2.3 [61]. The goal of SVM is to determine an optimal separating hyperplane to maximise the margin between the nearest data points, referred to as

support vectors, thereby minimising the classification error [62, 70]. There are many possible ways to construct a hyperplane between data classes, but only the optimal hyperplane will maximise the margin. Upon identification of the optimal hyperplane, the solution is presented as a linear combination of the support vectors, with all other data points being disregarded. The complexity of an SVM model is therefore not influenced by the varying number of features present in the training data. This is due to the SVM algorithm's tendency to identify and select only a limited number of support vectors. [61]. Therefore SVMs are capable of addressing learning tasks in which the quantity of features exceeds that of the training points.

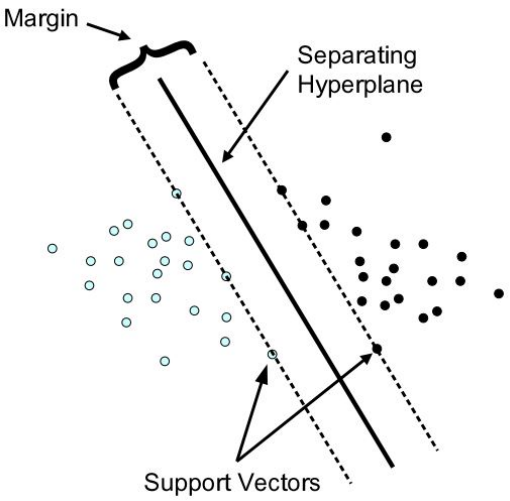

FIGURE 2.3: *Linear separable Support Vector Machine (SVM) [77].*

The majority of real-world problems involve data which is not linearly separable and for which a hyperplane does not exist. These problems can become computationally expensive, thus a *kernel* trick is used to separate objects from different classes [62]. A kernel trick is a mathematical technique that transforms data from the initial input space to a feature space with a higher dimension, resulting in linear separability of the data. A linear separation in the feature space then represents a non-linear separation in the initial input space [61].

2.1.2 Naive Bayes Classifier

The Naive Bayes classification algorithm simplifies the learning process by making a key assumption: that features are conditionally independent within a given class. [95]. The classifier

is based on Bayes' conditional probability theorem. [93]:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}, \tag{2.1}$$

where $P(H|E)$ is the posterior probability of the hypothesis given that the evidence is true; $P(E|H)$ is the probability of the evidence given that the hypothesis is true; $P(H)$ is the prior probability that the hypothesis is true; and $P(E)$ is the prior probability that the evidence is true.

The Naive Bayes Classifier work as follows [67, 70]:

Input : Training data set T
Output : A class to which a testing data set belongs

1. T is a set of samples where each sample is denoted by a n-dimensional vector $\mathbf{X} = x_1, x_2, ..., x_n$ of n features $F_1, F_2, ..., F_n$ respectively. Each sample belongs to one of the k potential classes $C_1, C_2, ..., C_n$.

2. The classifier will predict the class to which a given sample vector $\mathbf{X}$ belongs, based on the greatest posterior probability, under the condition of the given sample. That is, according to the Bayes theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})} \tag{2.2}$$

   $\mathbf{X}$ will be classified as C_i subject to $\mathrm{P}(C_i|\mathbf{X}) > \mathrm{P}(C_j|\mathbf{X})$ for $1 \leq j \leq k$ and $j \neq i$.

3. It is known that $P(\mathbf{X})$ is identical for all the classes, thus only $P(\mathbf{X}|C_i)\mathrm{P}(C_i)$ needs to be maximised. If the class prior probabilities $\mathrm{P}(C_i)$ are not known, it is assumed that they are equally likely, that is $P(C_1) = P(C_2) = ... = P(C_k)$ and therefore only $P(\mathbf{X}|C_i)$ will need to be maximised. It is also possible to estimate the class prior probabilities by $\mathrm{P}(C_i) = \mathrm{freq}(C_i, T)/|T|$.

4. Given the potential abundance of features contained in data sets, it can become computa-

tionally expensive to compute $P(\mathbf{X}|C_i)$. This computation is simplified by assuming that the class label of a sample and feature values are conditionally independent of one another. The assumption can be mathematically expressed as

$$P(\mathbf{X}|C_i) \approx \prod_{k=1}^{n} P(x_k|C_i) \tag{2.3}$$

where the probabilities $P(x_1|C_i)$, $P(x_2|C_i)$,..., $P(x_n|C_i)$ are estimated from the training data set.

5. Finally, $P(\mathbf{X}|C_i)\mathrm{P}(C_i)$ is assessed for each class C_i for $1 \leq i \leq k$. The classifier will then predict that whichever class C_i that maximises $P(\mathbf{X}|C_i)$ is the class to which sample $\mathbf{X}$ belong.

2.1.3 k-Nearest Neighbour

The k-Nearest Neighbour algorithm is a classification algorithm that operates on the assumption that instances exhibiting similar properties within a given data set are likely to be located in close proximity to one another. Each of the data points in close proximity will be tagged with the same classification label. If there are data points that are not yet labelled, it will be determined from the label of its k nearest classified neighbours [61, 69]. The instances with the shortest Euclidean distance from the query instance are typically considered to be the nearest neighbours. [58].

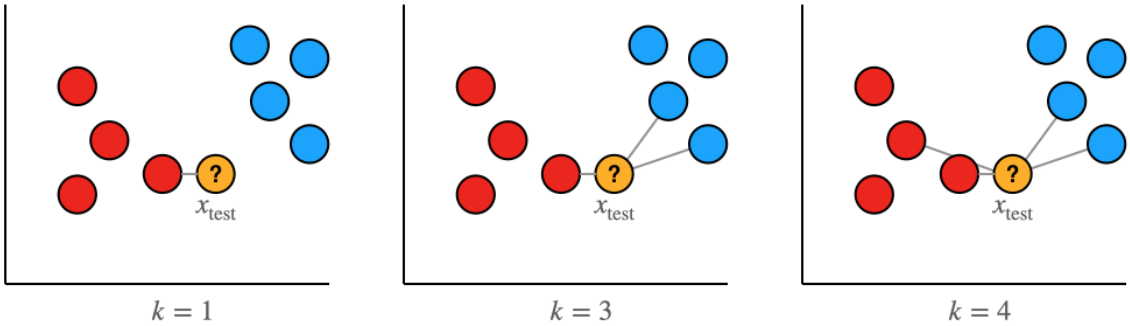

FIGURE 2.4: *An illustration of the k-nearest neighbour algorithm with different k-values [124].*

k Represents the number of nearest neighbours the algorithm will look at for classification and the choice of k can have a significant effect on how well the algorithm performs [108]. Figure 2.4 shows that the query instance (x_{test}) will be classified as part of the red class if k=1 because

the algorithm only looks for one nearest neighbour which is red in this case. If k=3, the query instance will be classified as part of the blue class because two of the three nearest neighbours are blue. If k=4, the query instance will not be properly defined as there are an equal amount of blue and red nearest neighbours. This problem, however, can easily be solved by always choosing an unequal value for k. Another error may occur when noisy features are present within the data set, leading to inaccurate classifications. For this problem, larger values of k would be preferred. Furthermore, there is no proper way to choose a value for k, other than utilising cross-validation or similar computationally expensive methods [42].

2.1.4 Decision Tree

The decision tree method does classification tasks by sorting instances in a data set based on feature values. A hierarchical tree-like learning model is constructed by a series of IF-THEN conditions derived from the training data set to sort the instances and predict the output class [31, 79]. The structure of the decision tree as shown in figure 2.5 consists of three node types including the *root node*, located at the apex of the decision tree; the *decision node*, which indicates a decision to be made on each feature; and the *leaf node*, which is the final classification node [92]. In the analysis of a given data set, classification of all instances commences at the root node and proceeds through successive decision nodes until they are ultimately sorted and classified at the leaf node.

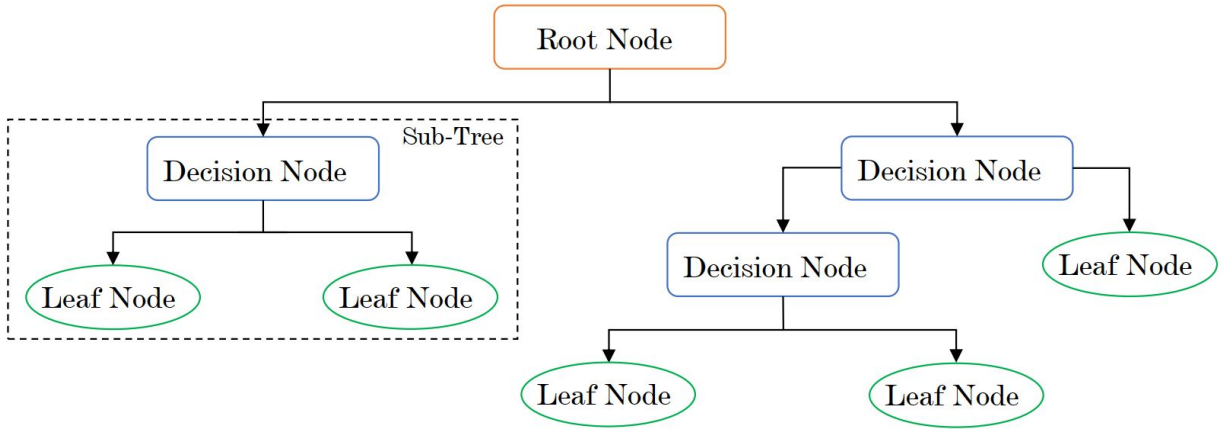

FIGURE 2.5: *Structure of the decision tree algorithm [45].*

2.1.5 Neural Networks

Neural networks, also known as Artificial Neural Networks (ANN) draw inspiration from the operations of neurons in the human brain, intimating complex tasks such as decision-making, pattern recognition, learning and reasoning [75]. The human brain remarkably consists of an average of 86 billion neurons that transmit and process information throughout the human body [50]. An ANN consists of a similar, but simplified model of the biological neural network structure with processing units (neurons) that are linked to each other and organised in a certain topology.

Neural networks form part of the study of deep learning which is thoroughly discussed in the following section.

2.2 Deep Learning

Deep learning is a subcategory of machine learning within the broader scope of artificial intelligence. Deep learning is a method that learns representations of data through non-linear computational models consisting of multiple processing layers. At every layer, features are extracted from the input image data, and representations of the data become more abstract which enables the model to learn complex functions. Critical features are amplified as the data progresses through the layers while irrelevant features are suppressed [65].

Features are extracted from images based on their pixel intensities. An input image can be represented as either a single-channel matrix or a three-channel matrix of pixels and each pixel within the input image has an intensity value ranging from 0 to 255 [84]. A greyscale image is a single-channel matrix where a pixel value of 0 is black and 255 is white with shades of grey in between. An image in the Red Green Blue (RGB) colour space is a three-channel matrix that defines a combination of red, green and blue intensities ranging from 0 to 255 for each pixel. A colour space refers to a predetermined range of potential colours and intensity values that can be presented on a visual plane [49]. The RGB colour space is presented in figure 2.6 along with

the coordinates for the saturated colours of pure red, green, blue and white.

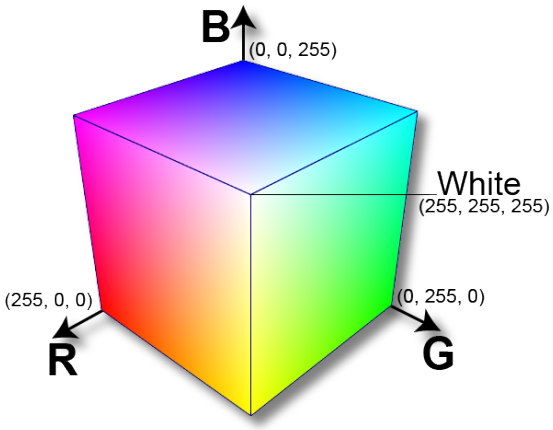

FIGURE 2.6: *A visual representation of the RGB colour space.*

Deep Learning uses a backpropagation algorithm to discover complex structures within substantial data sets. Backpropagation uses the chain rule of derivatives to compute the gradient of an objective function (error function) with reference to the internal parameters (weights) of a multiple-layer neural network. The weights are adjusted according to the objective function if the output of the neurons is different from the expected output. This process starts from the last layer and works in reverse through the network to the first layer. For every iteration of this process, the weights are updated to enhance the accuracy of the neural network's output [20, 65, 85].

Furthermore, deep learning is the study of artificial neural networks inspired by biological neuron networks. The neurons are where the computation happens and they are arranged in multiple layers as follows:

1. An *input layer* (i) that receives raw data.

2. One or more *hidden layers* (h_n) that execute nonlinear transformations of the input values.

3. An *output layer* (o) that produces a final prediction.

Figure 2.7 is an illustration of a typical artificial neural network architecture showing how neurons are interconnected and arranged in each layer to form fully connected layers *i.e.* every neuron

in the previous layer is connected to every neuron in the adjacent layer. The outputs of neurons are fed as inputs to the neurons in the following layers and the total neurons in the output layer is equivalent to the number of classes to be predicted.

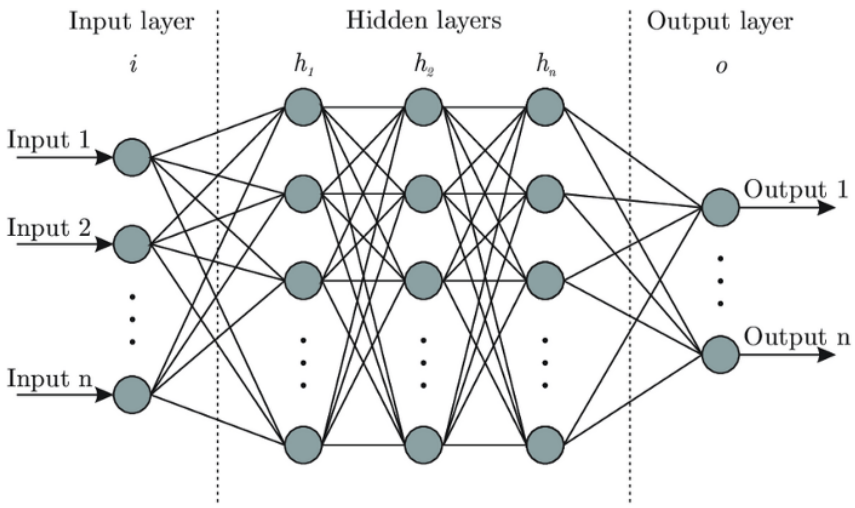

FIGURE 2.7: *A typical artificial neural network architecture [16].*

The general structure of a single neuron in the ANN is based on the interaction of the following variables [118]:

- Inputs: x_1, x_2,...,x_m
- Weights: w_1, w_2,...,w_m
- Bias: b
- Activation Function: $\varphi(.)$
- Output: y = $\varphi(.)$

Figure 2.8 is an illustration of what a single neuron in an artificial neural network might look like and it shows how each neuron's output is dependent on the weighted sum of the inputs, $\mathbf{X}$=$[x_1, x_2, ..., x_m]$ and the weights, $\mathbf{W}$=$[w_1, w_2, ..., w_m]$ assigned to them and a bias, b that either amplifies or dampens the inputs. The sum is then transmitted through an activation function, φ(x) to determine whether and to what degree the neuron's signal should advance through the ANN to affect the final prediction. When the neuron has been activated, the signal will advance [82]. The final output of the artificial neuron can be mathematically expressed as:

$$y = \varphi(\sum_{j=1}^{m} w_j x_j + b) \tag{2.4}$$

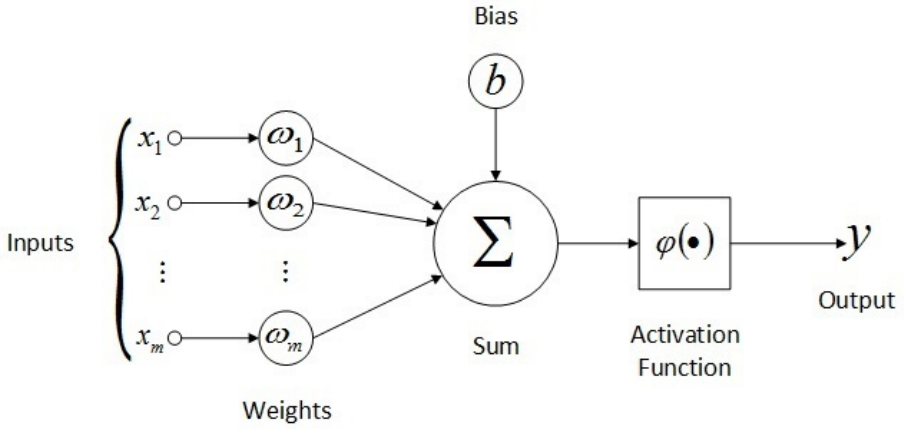

FIGURE 2.8: *A diagram of what one neuron in an artificial neural network might look like [118].*

2.2.1 Activation Functions

The activation function is a very important element of the neuron and when applied to neural networks, it gives them the ability to learn, represent and process complex data and make complex non-linear mappings between inputs and outputs. The activation function outputs are fed as inputs into the neurons of the following layers. A key feature of the activation function is that it should be differentiable in order to enable backpropagation in the neural network to compute the errors and update the weights and biases accordingly to minimise the errors.

Some activation functions including the binary step- and linear activation function as well as prevalent non-linear activation functions such as the sigmoid-, hyperbolic tangent-, rectifier linear units-, leaky rectifier linear units- and softmax activation function, are discussed below.

2.2.1.1 Binary Step Function (Threshold Function)

The binary step function is an activation function typically utilised for binary classification. The neuron will only be activated if the input value is greater than some threshold value and consequently a value of 1 will then be assigned to the output. Otherwise, the output will be

deactivated and have a value of 0 assigned to it and as a result, the output will not be fed as an input into the following layer. The drawback of this activation function is that it can not be used for multiple-class classification - a neuron can only belong to one class. Also, this activation function has a zero gradient, *i.e.* the derivative of its output is equal to zero which can cause difficulties in the backpropagation algorithm [106]. Figure 2.9a is a visual representation of the binary step function and in mathematical terms, the function can be defined as:

$$\varphi(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.5}$$

2.2.1.2 Linear Function

A linear activation function is a straight-line function that is directly proportional to the weighted sum of the input. As opposed to the binary step function, the linear activation function can be utilised for multiple class classification. The drawback however is that the derivative of the function is a constant value, meaning that all updates made in backpropagation are made with the same factor *i.e.* the neural network would not improve the error. A linear activation is therefore not able to deal with complex tasks, but rather simpler tasks where interpretation may be required [97, 106]. The linear activation function is illustrated in figure 2.9b and the mathematical formula for the function is:

$$\varphi(x) = cx, \tag{2.6}$$

where c is a constant parameter of the function's gradient.

2.2.1.3 Sigmoid Function

The sigmoid function which is also known as the logistic activation function takes a probabilistic approach towards decision making [23, 63]. This function has a smooth S-shape and is asymmetric about zero with outputs ranging from 0 to 1 [106]. It is one of the most used activation functions and it has proven to be very effective, but the problem with this function is that when very high or very low inputs are converted to a range between 0 and 1, the derivatives become very small and the gradient becomes almost non-existent. Hence this problem is called the *vanishing*

gradient problem and it will result in a neural network not being able to learn further [97]. Figure 2.9c shows the graph for the sigmoid function and mathematically it can be defined as:

$$\varphi(x) = \frac{1}{1 + e^{-x}} \tag{2.7}$$

The hyperbolic tangent (tanh) function exhibits certain similarities with the sigmoid function. However, it distinguishes itself from the sigmoid function by presenting symmetry about zero, and its output values span the range of -1 to 1 for a given neuron.

2.2.1.4 Hyperbolic Tangent Function

The hyperbolic tangent (tanh) function bears similarity to the sigmoid function but is different in that it is symmetric about zero and the neuron outputs range from -1 to 1, thus the hyperbolic tangent function has a steeper gradient [106]. The tanh function's symmetric output around zero leads to faster convergence which speeds up the learning process, however, the hyperbolic tangent function also has the vanishing gradient problem [8]. The hyperbolic tangent function is shown in figure 2.9d and the mathematical formula for this function is as follows:

$$\varphi(x) = 2sigmoid(2x) - 1 = \frac{2}{1 + e^{-2x}} - 1 \tag{2.8}$$

According to neuroscience research, only one to four percent of the brain's neurons can be activated at the same time, but neural networks with sigmoid or tanh activation functions have about half of their neurons activated simultaneously. The redundancy of activated neurons is not consistent with neuroscience research which is a good indication as to why both the sigmoid and tanh activation functions are computationally expensive [32].

2.2.1.5 Rectifier Linear Units Function

The Rectifier Linear Units (ReLU) activation function is a solution to the problems of the sigmoid and tanh functions. The activation function's outputs range from zero to infinity where all negative inputs are converted to zero, and when a neuron's output is zero, it will not be activated. Thus, the ReLU activation function does not activate all the neurons at the same

time which creates a sparse matrix in the hidden layer of the neural network which improves computation efficiency. This activation function however is not flawless and has a problem called the *dying ReLU problem*. The dying ReLU problem occurs when some weights and biases in the neural network are not updated anymore during the backpropagation steps due to the zero gradient. Regardless of what the inputs are to these neurons, they will always be zero *i.e.* the neurons will never be activated and will effectively die [9, 32, 106]. The ReLU function is shown in figure 2.9e and it can be mathematically defined as:

$$\varphi(x) = max(0, x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.9}$$

2.2.1.6 Leaky Rectifier Linear Units Function

The Leaky ReLU (LReLU) activation function solves the dying ReLU problem by introducing a slight constant gradient below zero that enables backpropagation. Instead of converting all negative values to zero as with the ReLU activation function, the LReLU activation function converts them to very small negative output values which ensures that there are no neurons that can not be activated [32, 63]. The graph for the LReLU activation function is illustrated in figure 2.9f and compared to the graph of the ReLU activation function, it can be seen that they are very similar, except for the negative gradient that was introduced to the LReLU function. The LReLU function can be mathematically defined as:

$$\varphi(x) = max(0, x) = \begin{cases} x, & x \geq 0 \\ c, & x < 0 \end{cases} \tag{2.10}$$

where c is a small constant value like 0.01 for example.

More variations of the ReLU activation function exist where performance improvements are made, but they also contain their own flaws. For example, the parametric rectifier linear unit (PReLU) activation function is created to replace the constant parameter in the LReLU activation function with an optimal parameter that is learned from the data [32]. The downside of determining an

optimal parameter is that it can become computationally expensive, thus some of the advantages of the ReLU activation function are fading.

2.2.1.7 Softmax Activation Function

The goal of the softmax activation function is to convert a vector of real input values into a vector of real output values ranging from 0 to 1 that sum to 1 so that the values can be interpreted as probabilities in multiple classification problems. This activation function is predominantly used in the last layer *i.e.* the output layer of a neural network [63]. The graph for the softmax activation function is similar to the graph of the sigmoid function and the mathematical formula for the softmax activation function is defined as follows:

$$\varphi(\mathbf{X})_i = \frac{e^{-x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{2.11}$$

where $\mathbf{X}$ is a vector of real input values, i is the i-th value in the vector and K is the number of classes in the multiple class classifier.

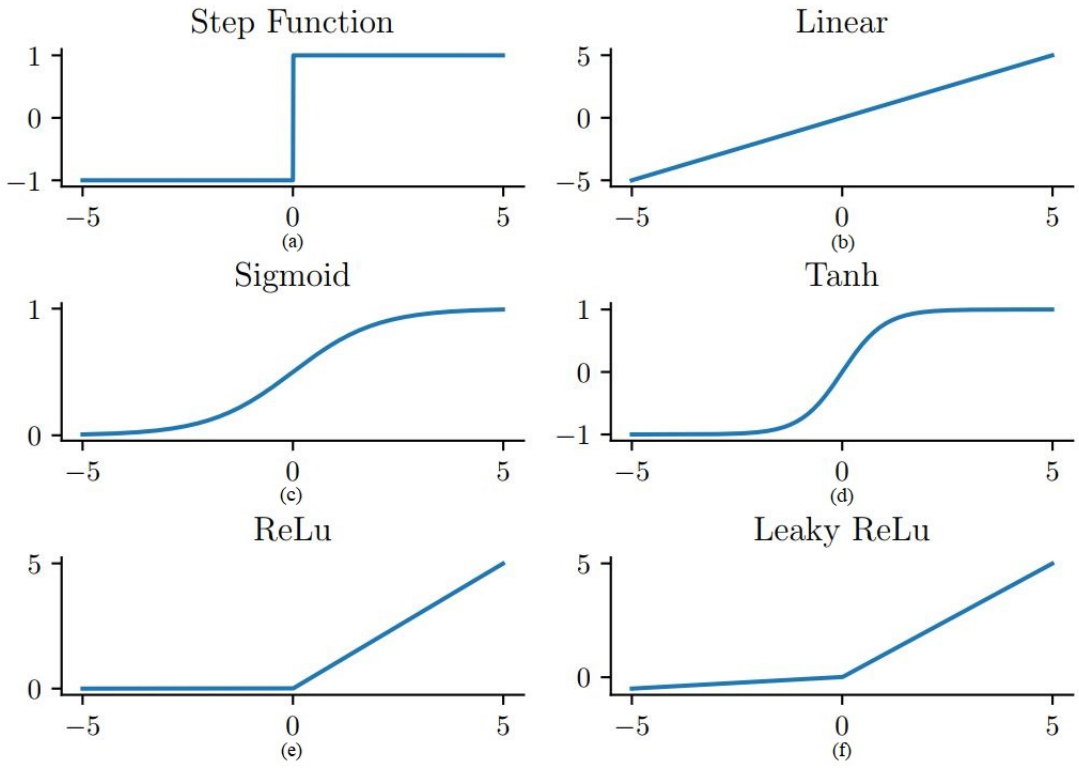

FIGURE 2.9: *Graphs of the most popular activation functions used in an artificial neural network [35].*

There are various types of deep learning methods that made significant improvements to state-of-the-art computer vision applications such as object detection, object recognition, image classification and many other domains. Among others, these methods include Convolutional Neural Networks, Recurrent Neural Networks, Restricted Boltzmann Machines, Sparse Coding and Autoencoder [43]. The Convolutional Neural Network (CNN) is a supervised deep learning method and it is discussed in the following subsection. The other deep learning methods are all unsupervised learning types that fall beyond the scope of this project and they will not be further discussed.

2.2.2 Convolutional Neural Networks

The Convolutional Neural Network is a method of deep learning that is designed to solve computer vision problems such as image classification, object detection and image segmentation by recognising patterns within images [83, 87]. A CNN is similar to a traditional ANN in the way it is made up of multiple layers of neurons that are being updated through learning. The structure of a CNN however, is made up of three different types of layers whereas an ANN only has one type of layer. The three main layers in a CNN are convolutional layers, pooling layers and fully connected layers [83]. Figure 2.10 shows the general architecture of a CNN and it can be seen that the fully connected layer is basically a traditional ANN and that a convolutional layer and pooling layer are added in front of the ANN to form a CNN. Additionally, the CNN also contains an activation layer and flattening layer that exist in between the main layers. The layers of a CNN are discussed below.

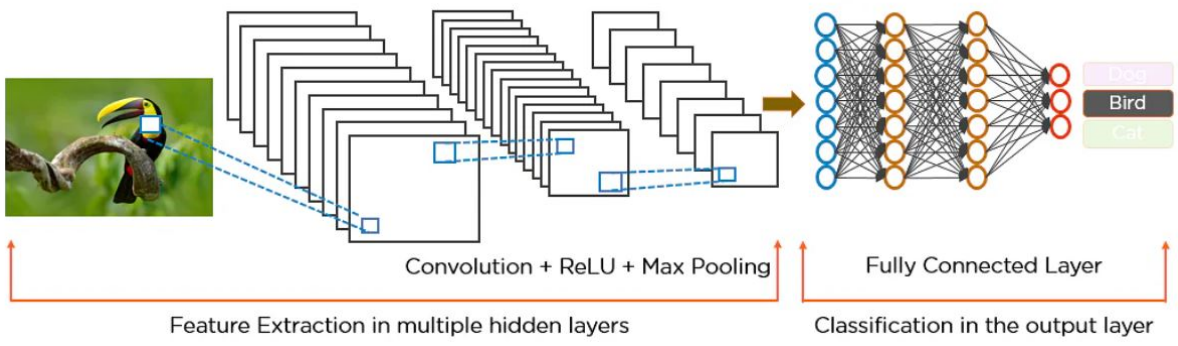

FIGURE 2.10: *A general architecture of a convolutional neural network[17].*

2.2.2.1 Convolutional layer

Convolutional layers are at the beginning of the CNN and they consist of locally connected neurons with shared input weights that make use of kernels (filters) to extract important features from input images. The neurons each cover a smaller area of the input image's spatial dimensionality known as the receptive field [20]. Images are fed into the network as multidimensional arrays of pixels that have values between 0 and 255. These images can be either greyscale or RGB images with array dimensions of h x w x d (height, width, and depth), where depth is the number of colour channels, therefore the greyscale and RGB images have depths of one and three respectively. Kernels have smaller array dimensions, or at least smaller than that of the input image, but with the same depth as the input image [28]. As the name of the layer suggests, the kernels *convolve* across the image by computing the dot product of the kernel values with the pixel values of the receptive field as the kernel shifts across the image. As a result, a two-dimensional feature map will be produced which in turn is locally connected to the neurons in the following layer. A feature map will be produced for every kernel that is applied to the input image, thus, if n kernels are applied, n feature maps will be produced and stacked as an output to the convolutional layer [20, 83]. This convolution process is illustrated in figure 2.11.

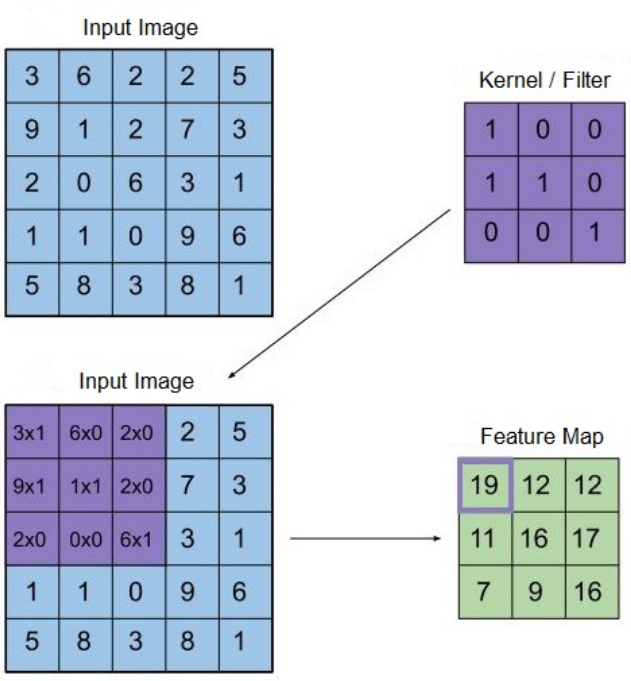

FIGURE 2.11: *An illustration of how a kernel convolves across an input image to produce a feature map as an output. [20].*

Every additional convolution layer is able to recognise more complex features by combining previous feature maps. The first features that will be extracted from input images are simple features such as edges and different shapes (circles, rectangles, squares). The convolutional layers closer to the end are able to detect more complex features such as the structure of human or animal faces. The downside, however, is that every additional layer adds more complexity to the computations of the CNN [20, 66]. The computation complexity of the CNN can be reduced by optimising the output size of the convolutional layers. It is possible to optimise the output size by altering the following three hyperparameters: depth, stride and zero-padding [83].

The depth of the output *i.e.* the number of stacked feature maps is simply equal to the number of kernels in the convolutional layer. The number of kernels needed to convolve across an image is dependent on the size of the kernel. For example, a 2 x 2 kernel covers a smaller receptive field than a 3 x 3 kernel, therefore more 2 x 2 kernels than 3 x 3 kernels will be needed to cover the entire spatial size of an image, assuming the stride (which is explained below) remains the same. Reducing the number of kernels used in the network will reduce its complexity, but also reduce the CNN's ability to recognise patterns. Thus, the CNN will benefit from choosing an optimal number of kernels [83].

The stride is the number of positions the kernel moves in the horizontal direction from left to right, or in the vertical direction from top to bottom to convolve with a new receptive field in the image. If a value of 1 is assigned to the stride, the receptive fields will significantly overlap and a large feature map will be produced. A greater stride number will reduce the amount of overlapping receptive fields and produce a smaller feature map [4, 83].

Zero-padding is the process of adding zeros around the border of the image's array of pixels so that the size of the feature map can be managed. This process will help to ensure that the information that is on the border of an image does not disappear. Consequently, the output of the convolutional layer is not reduced too quickly and the CNN can learn more effectively [4, 83].

These three parameters all have an impact on the output size of the convolutional layer and in turn the complexity of the CNN. The output size can be determined by the following formulas:

$$\text{Output Width} = \frac{W - K_w + 2P}{S_w} + 1 \tag{2.12}$$

$$\text{Output Height} = \frac{H - K_h + 2P}{S_h} + 1 \tag{2.13}$$

where W is the input width, H is the input height, K_w is the kernel width, K_h is the kernel height, P is the number of zero-padding layers, S_w is the horizontal stride and S_h is the vertical stride [78].

2.2.2.2 Activation Layer

Following the convolutional layer, there is an activation layer that applies an activation function to the output of the convolutional layer. The ReLU activation function is the most commonly used activation function to ensure non-linearity and it converts all the negative values in the feature map to zero [17].

2.2.2.3 Pooling Layer

The goal of a pooling layer is to reduce the onward computation complexity in the CNN by reducing the resolution of a feature map before it is fed into the following convolutional layer [4]. The pooling layer limits overfitting, which is a state that occurs when the algorithm learns irrelevant and noisy data that are fitted closely to the training data, leading to ineffective learning [22]. The two main pooling methods are maximum pooling and average pooling as shown in figure 2.12. For both methods, the first step is to split the spatial size of the feature map into smaller regions. Maximum pooling takes the maximum pixel value in each smaller region and discards the rest of the values. Average pooling computes the average pixel value in each smaller region. The resulting values of these methods form updated and smaller feature maps [87].

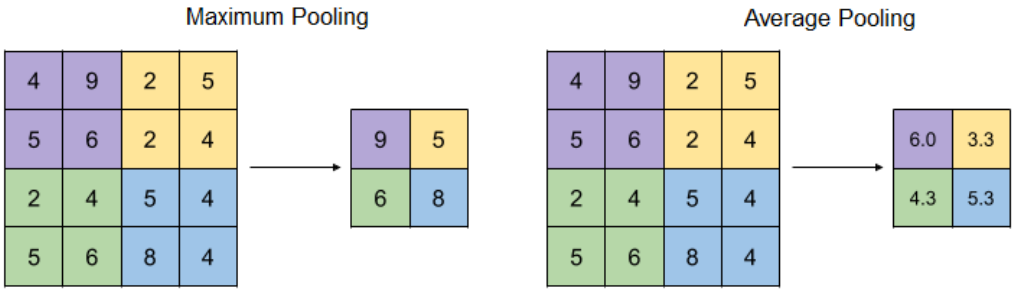

FIGURE 2.12: *Maximum Pooling and Average Pooling [88].*

2.2.2.4 Flattening Layer

A flattening layer exists between the final pooling layer and the fully connected layers. This layer simply converts the data of a two-dimensional feature map into a one-dimensional array so that it can be fed as an input into the first fully connected layer as shown in figure 2.13 [76].

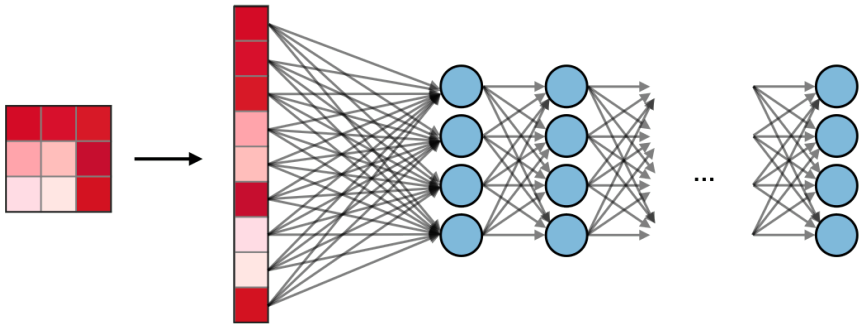

FIGURE 2.13: *Flattening of a feature map [6].*

2.2.2.5 Fully Connected Layer

The final layers of the CNN are fully connected layers that undertake the shape of a traditional ANN. All the neurons of one layer are fully connected to all the neurons in the adjacent layers [83]. The output layer of the CNN also forms part of the fully connected layer. The neurons in the output layer utilise the softmax activation function to make classifications based on the features detected by the convolutional layers.

2.2.3 Object Detection with Deep Learning

Object detection is a process that determines to which class an object belongs (object classification) as well as estimating where the object is located within the image (object localisation). A

bounding box is drawn around the object of interest to indicate the location of the object [86]. Object detection differs from image classification since object detection can identify multiple objects belonging to different classes in a single image, whereas image classification only makes a classification based on an image as an entirety.

Deep learning based object detection can be divided into three stages: (i) informative region selection, (ii) feature extraction and (iii) classification. Informative region selection searches the image for locations where objects may appear. These locations are also known as regions of interest (ROI). Feature extraction obtains a vector of features at different regions of the image that represent discriminative semantic information of the region. In the classification step, the region classifiers learn to label regions. Support vector machines are commonly used for this classification [121]. The manner in which these stages are approached depends on whether a two-stage detector framework or a one-stage detector framework is used. Two-stage detectors first use informative region selection and then extract features from each of those regions, followed by classification of the region. One-stage detectors predict object categories at each location on the feature map without using the cascaded region classification step [121].

A Region-based convolutional neural network (R-CNN) proposed by Girshick *et al.* is an example of a two-stage detector framework [39]. R-CNN makes use of a selective search algorithm to generate about 2000 region proposals within an image. These regions are bounded and features are extracted from these regions with a CNN. Based on the extracted features, a category-specific linear SVM classifies the object within the region proposals. It is possible that a region containing an object will have multiple overlapping bounding boxes. Such regions are adjusted with bounding box regression and filtered with non-maximum suppression to obtain the final bounding box location for an object [126].

The You-Only-Look-Once (YOLO) model is an example of a one-stage detector framework that makes predictions for both categories and boundary boxes of objects. The YOLO framework divides an image into an S x S grid as shown in figure 2.14. From each cell in the grid, features are extracted to make a prediction on the object centred in that cell. The bounding boxes in each grid cell are then predicted with their corresponding confidence scores. Adjacent bounding boxes that predict the same object are merged together to predict a final bounding box [28, 126].

Figure 2.14 shows how the adjacent grid cells around the dog and bicycle respectively are merged together in the class probability map.

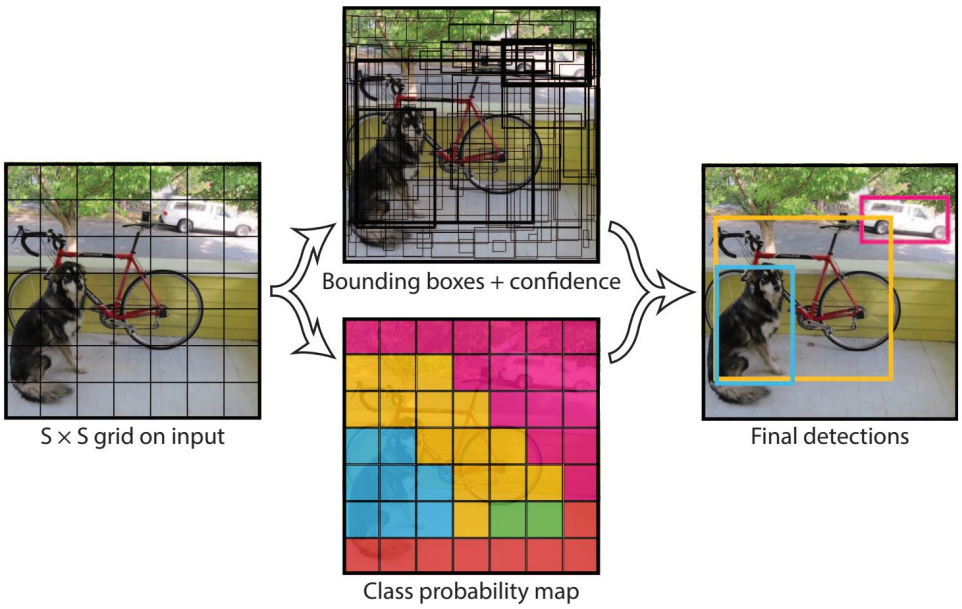

FIGURE 2.14: *The basic idea of how a YOLO framework operates [94].*

2.3 Image Segmentation

It is often in practice that not all parts within an image are of interest, but only some areas that hold certain features. Therefore, a process called image segmentation is used to segment (divide) images into multiple segments that contain similar features and characteristics. Image segmentation is a method that correctly classifies the pixels within an image through a decision-oriented application such that the pixels within the same segment exhibit similar features and the pixels between the segments show a high contrast. From the segmented areas, a binary image (an image made up of pixels that can only have one of two distinct colours) called a mask is created that isolates the objects of interest from the background of the image. The goal of image segmentation is to simplify the representation of an image for easier further analysis [28, 110].

There are two broad category types in which image segmentation can be performed, namely semantic segmentation and instance segmentation. Semantic segmentation assigns a different colour to every class, and every pixel is assigned the colour corresponding to the class it belongs. Thus, objects that belong to the same class will have the same colour. Instance segmentation is

different from semantic segmentation in that it separates objects from the same class *i.e.* every object will have a different colour assigned to it, even if some objects belong to the same class [7, 81]. Figure 2.15 clearly shows the difference between semantic segmentation and instance segmentation as explained.

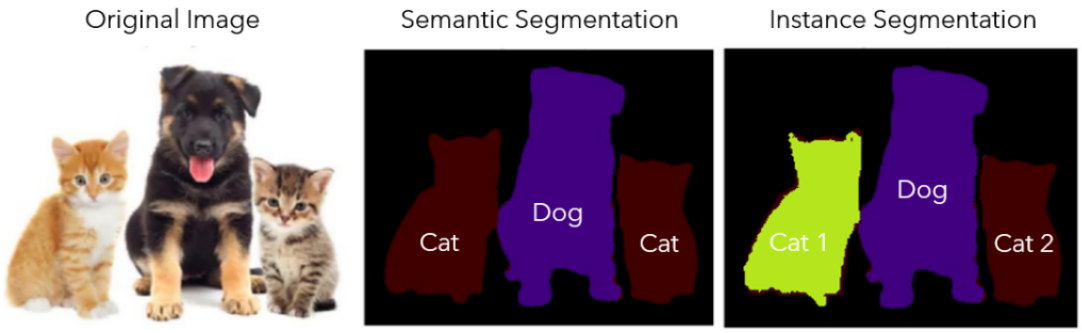

FIGURE 2.15: *Semantic Segmentation vs Instance Segmentation[12].*

Various popular image segmentation techniques are discussed below including edge-detection, threshold, region-based, watershed and clustering techniques.

2.3.1 Threshold Techniques

The threshold segmentation technique divides the pixels of an image into multiple regions which separate objects from the background by introducing a threshold value, T. Any arbitrary pixel with an intensity level greater than some threshold value is considered to be part of some object, otherwise, the pixel is considered to be part of the background. This technique is simple and fast and it is especially useful in images that have objects with lighter intensity levels than the background [27]. The drawback, however, is that thresholding does not take the spatial features into account, thus it can be sensitive to noise [30]. The three basic types of thresholding techniques are global thresholding, variable thresholding and multiple thresholding.

2.3.1.1 Global Thresholding

Global thresholding makes use of a constant thresholding value for the entire image, thus the image will only be segmented into two regions [57]. For a given input image $f(x,y)$ and threshold value T, the binary output of an image $g(x,y)$ can be produced from the following formula:

$$g(x,y) = \begin{cases} 1, & f(x,y) > T \\ 0, & f(x,y) \leq T \end{cases} \tag{2.14}$$

In this formula a 1 will be assigned to a pixel that belongs to an object and a 0 will be assigned to a pixel that belongs to the background. The problem with this technique is that it might deliver inaccurate results when the background enlightenment is not even [109].

2.3.1.2 Variable Thresholding

The variable thresholding technique consists of a threshold value that varies across the image to possibly solve the irregular enlightenment problem. This technique can be further broken down into two types namely local thresholding and adaptive thresholding [57, 109].

In *local threshold* the threshold value at a certain point *(x,y)* in an image depends on the pixel intensities of the neighbourhood.

In *adaptive threshold* the threshold value is a function of x and y.

2.3.1.3 Multiple Thresholding

The output of a multiple thresholding technique relies on multiple predefined threshold values like T_0 and T_1 to group pixels into different regions [57]. With such threshold values, the output can be computed as:

$$g(x,y) = \begin{cases} a, & f(x,y) > T_1 \\ b, & f(x,y) \leq T_1 \\ c, & f(x,y) \leq T_0 \end{cases} \tag{2.15}$$

where a, b, and c are some values that are assigned to pixel values, corresponding to the region in which they are grouped. The output can be in the form of a greyscale image where a, b, and c will have different intensity values to separate the regions from each other.

2.3.2 Edge Detection Techniques

Edge detection is a segmentation technique that detects pixels or edges with rapid changes in intensity between the different regions of an image. In other words, this technique makes use of discontinuities on the object boundaries within an image to detect edges. The discontinuities between the edges are then closed to form the required boundaries. The advantage of viewing an image in the form of its edges is the significantly lower quantity of data that needs to be processed compared to its original form, while the important information about the shapes of objects is retained [91, 109]. In literature, there are many techniques for edge detection segmentation, with some of the most popular techniques discussed below [14, 71].

2.3.2.1 Sobel Operator

The Sobel Operator can be thought of as a discrete differential technique that approximates the gradient of an image intensity function. This technique detects edges based on the first derivative. A pair of 3 x 3 matrices that represent transverse (equation 2.16) and longitudinal (equation 2.17) kernels, convolve across the image to detect edges with horizontal and vertical orientations respectively.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.16}$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.17}$$

At every pixel, the horizontal and vertical gradient approximations can be combined to determine the magnitude of the gradient using equation 2.18, and the orientation of the gradient using equation 2.19.

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.18}$$

$$\theta = arctan(\frac{G_y}{G_x}) \tag{2.19}$$

2.3.2.2 Roberts Edge Detection

The Roberts edge detection technique is similar to the Sobel edge detection technique, however, this technique is designed to detect edges with approximately 45° orientations. This operation is performed by a pair of 2 x 2 kernels as expressed in equation 2.20 and 2.21 that separately convolve across the image.

$$G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.20}$$

$$G_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{2.21}$$

As with the Sobel edge detection technique, G_x and G_y can be combined to determine the magnitude and the orientation of the gradient using equations 2.18 and 2.19 respectively.

2.3.2.3 Canny Edge Detection

The canny edge detection technique is a very well-known edge detector and by some, it is known as the optimal edge detection technique. The goal of this technique is to improve the performance of existing edge-detecting techniques according to a criteria list. The first criterion is to minimise the error rate *i.e.* edges should not be missed and no false edges should be detected. The second criterion is that the difference between the detected and the actual distance between edges should be minimised. The third and last criterion is that each edge should only be detected once. Based on this list of criteria, the canny edge detection technique is executed in the following steps [14,

71]:

Step 1: Filter out the noise that may be present in the image before applying the edge detector. A Gaussian filter is used to execute this task. A Gaussian filter is a smoothing operator that convolves across an image to *blur* images and removes any noise [36].

Step 2 and Step 3: These steps follow the same process as the Sobel or Roberts edge detection techniques. Kernels convolve across the image to find edges and the appropriate formulas are used to determine the magnitude (step 2) and orientation (step 3) of the edges at each pixel.

Step 4: When the orientation of the edge has been determined, relate the direction to a degree that can be traced. Relate the direction to either horizontal (0 degrees), positive diagonal (45 degrees), vertical (90 degrees), or negative diagonal (135 degrees).

Step 5: Apply non-maximum suppression to the image to trace along the detected edges. The local maximum edges will be regarded as true edges while pixel values that are not regarded as edges will be suppressed (set to zero).

Step 6: Apply hysteresis to avoid edge contours breaking up due to edges that occur above or below the threshold. Hysteresis is a double thresholding method consisting of a high and a low threshold. Pixels greater than the higher threshold are marked as edges. Any pixels that are connected to these edge pixels with intensities greater than the lower threshold are also marked as edges.

2.3.3 Region-based Technique

The region-based segmentation technique assigns every pixel in an image to a specific region that refers to objects within the image. The region of an image is a subset of analogous pixels based on some predefined criterion like their grey intensity level or texture. The two techniques based on region-based segmentation are region growing and region splitting and merging and they do well at handling noise[55, 56, 109].

2.3.3.1 Region Growing

Region growing groups pixels into regions by first defining an initial seed point. Then all the pixels that are connected to the seed point with similar criteria are added to the growing seed region. This process will continue until every pixel has been assigned to a region [30]. The region growing technique follows a simple five-step procedure [27]:

1. Choose an initial group of seed pixels in the original image.
2. Decide on predefined criteria for similarities.
3. Set up a stopping rule, for example, the maximum size a region can grow or region shape.
4. Grow regions by adding neighbouring pixels that conform to the criteria for similarities to a seed point.
5. Stop the region growing for a particular region when the stopping rule is met.

2.3.3.2 Region Splitting and Merging

The region splitting and merging technique follows a top-down approach. Instead of choosing an initial seed point, this technique splits the original image into multiple unconnected regions such that each separate region is more homogeneous to the entire image [27, 109]. The quad tree method as illustrated in figure 2.16 is used to split each region into four sub-regions. The adjacent similar regions are then merged again to execute reasonable segmentation of the original image. The procedure of the region splitting and merging technique works as follows [27, 57]:

Let R represent the entire original image and let H be some predicate.

1. If H(R) = False, then the original image is divided into four separate regions with the quad tree method. If H is not true for any of those sub-regions *i.e.* if $H(R_i)$ = False for any value of i, then that region is split into another four regions. This process will continue until splitting is not possible anymore.
2. Any adjacent regions R_i and R_j for which $H(R_i \cup R_j)$ = True, are merged.
3. Stop when merging is not possible anymore.

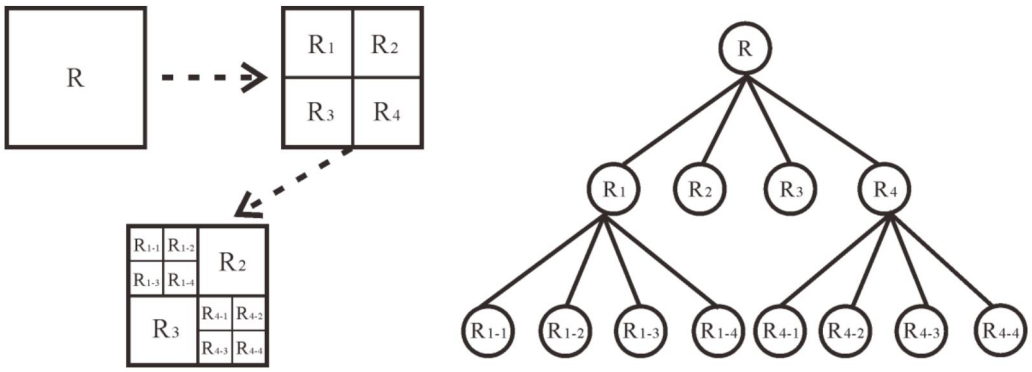

FIGURE 2.16: *An illustration of the quad tree method[52].*

2.3.4 Watershed Technique

The watershed segmentation technique follows the idea of a geographical analogy and interpretation. An image is interpreted as a topographical landscape with ridges and valleys. The elevation between the ridges and valleys is typically defined by the pixel intensity values, hence a three-dimensional representation of an image with catchment basins is created. For each regional minima, a basin exists that is made up of all points whose trajectory of steepest descent ends at that minima as shown in figure 2.17. A "water source" at each of the regional minima will start to fill the basin and the area where the water of adjacent basins meet serves as boundaries (watersheds) between regions. This process continues until all basins are surrounded by watersheds.[28, 89]. The watershed segmentation technique is known to perform well and extract meaningful regions and boundaries, but it can sometimes have the problem of over-segmentation due to noise in the image[46].

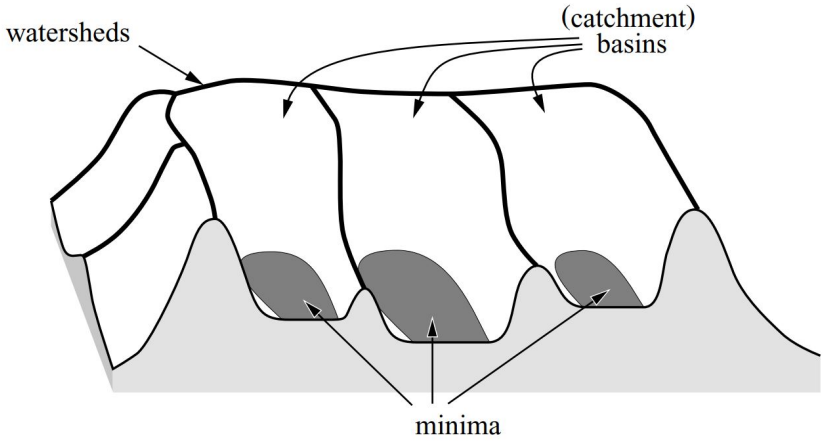

FIGURE 2.17: *A topographical interpretation of an image with watersheds, basins and minima [46].*

2.3.5 Clustering Technique

The clustering technique is an unsupervised learning task for image segmentation. This technique segments an image into multiple clusters so that pixels in the same cluster have more similar properties than those in other clusters. There are two categories of the clustering technique: the hierarchical method and the partition-based method. A hierarchical method is based on the structure of a tree where the roots refer to the database and the internal nodes refer to the clusters. The partition-based method utilises optimisation techniques to iteratively minimise an objective function that divides pixels into non-overlapping clusters [30, 57, 105]. Within these two categories, there are a further two basic types of clustering namely hard clustering and soft clustering that are discussed below.

2.3.5.1 Hard Clustering

Hard clustering techniques divide an image into multiple clusters and it limits each pixel to belong to one and one cluster only. These types of techniques assign membership values of either 1 or 0 to pixels, meaning that they can either belong to a certain cluster or not. The K-means algorithm is a well-known technique of hard clustering. The algorithm works on the basic concept of grouping n pixels into K clusters according to the distance between the pixels, where $K < n$. The algorithm also focuses on maximising intra-cluster similarity and minimising inter-cluster similarity which in turn means minimising the intra-cluster distance and maximising the inter-cluster distance as shown in figure 2.18 [57, 27]. The K-means algorithm can be expressed as follows [27, 110]:

1. Initially choose an arbitrary number of K clusters to be formed.
2. Randomly select K pixels with different intensities to act as centroids.
3. Compute the Euclidean distance between every pixel and every centroid. A pixel will be assigned to the closest clustering centroid.
4. When there are no more pixels to be assigned to clusters, the initial clustering is completed.
5. Calculate the mean of every cluster and let this mean be the updated centroid of the cluster.
6. Repeat steps 3 to 5 until the centroids no longer change or when the iteration limit has been reached.

7. The image is appropriately grouped into K clusters.

The advantage of the K-means algorithm is its simplicity, its efficiency and its scalability to various sizes of data sets. The drawback however is that there is no explicit selection criteria to determine the optimal number of clusters to be used by the algorithm and it can be difficult to estimate a suitable number of clusters [110].

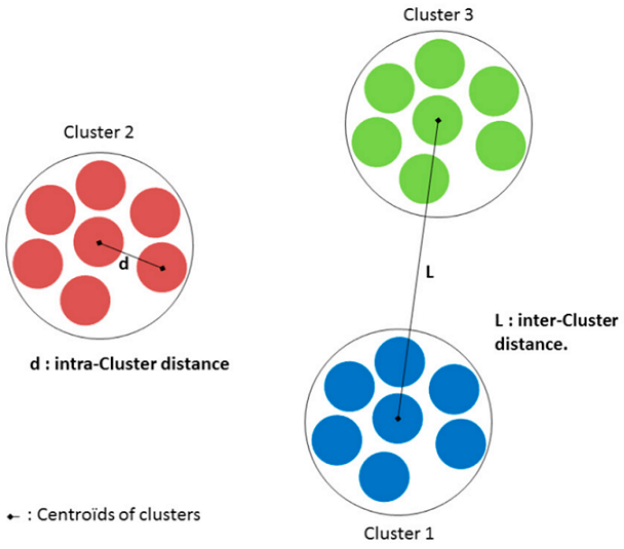

FIGURE 2.18: *Maximised inter-cluster and minimised intra-cluster distances in a k-means clustering process [21].*

2.3.5.2 Soft Clustering

Soft clustering is a type of clustering that has a more natural approach. In the real world, the exact grouping of clusters is not realistic since noise is present, therefore, soft clustering will be useful in circumstances where precise grouping is not strictly needed. The fuzzy c-means clustering technique is a popular technique of soft clustering. The fuzzy c-means technique groups pixels into clusters based on partial membership which allows pixels to belong to multiple clusters. The degree to which each pixel belongs to a cluster is described by a membership value between 0 and 1. The fuzzy c-means technique is more flexible compared to the K-means technique [57].

Table 2.1 is a summarised comparison of the advantages and disadvantages of all the segmentation

techniques that are discussed above.

TABLE 2.1: *A comparison of the various segmentation techniques [57].*

Segmentation Technique	**Description**	**Advantages**	**Disadvantages**
Threshold Technique	Based on a threshold to divide pixels into different groups.	No previous information of the image is required.	Spatial information is not considered.
Edge Detection Technique	Based on the detection of discontinuity on object boundaries.	Works well with images with good contrast between objects and it reduces the data quantity that needs to be processed.	It can sometimes detect false edges.
Region Based Technique	Based on dividing an image into homogeneous regions.	Good at handling the presence of noise.	Time and memory consuming.
Watershed Technique	Based on geographical analogy and interpretation.	Stable and meaningful results with continuous boundaries.	Complex gradient calculations and over-segmentation can occur due to noise.
Clustering Technique	Based on segmenting an image into homogeneous clusters.	Soft clustering can be applied to real world problems due to partial membership assigned to pixels.	The membership function is complex.

2.4 Performance Evaluation Metrics

The accuracy and effectiveness of image segmentation and object detection methods are assessed by performance evaluation metrics. They provide a quantitative measure of how well the method's algorithm is performing. The evaluation metrics described below are the most commonly used metrics as they give a good overall performance review of object detection and segmentation methods. A confusion matrix as in table 2.2 that provides insight into the number

of correctly predicted image pixels needs to be created before the metrics can be calculated. A positive pixel belongs to the object of interest and a negative pixel does not. True positive (TP) and true negative (TN) are the pixels that are correctly classified as positive and negative pixels respectively, whereas false positive (FP) and false negative (FN) are incorrectly classified as positive and negative pixels respectively. The values of TP, TN, FP and FN are used to calculate the performance evaluation metrics [103].

TABLE 2.2: *A confusion matrix for the ground-truth and predicted segments in an image.*

		Ground-truth	
		Positive	Negative
Prediction	Positive	**True Positive (TP)**	**False Positive (FP)**
	Negative	**False Negative (FN)**	**True Negative (TN)**

2.4.1 Accuracy

The accuracy metric represents the percentage of correctly classified pixels within an image and it can be mathematically described as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \times 100 \tag{2.22}$$

A higher percentage indicates a better performance. This metric however can give a misperception of the method's performance if the data set is imbalanced. For example, if 90 % of the pixels in an image are negative pixels and the metric achieves a score of 90 % accuracy, it can mean that only the negative pixels have been correctly classified and no positive pixels have been detected. This issue is called class imbalance, therefore more metrics are used to evaluate the segmentation method [25].

2.4.2 Precision

Precision is an indication of how precisely the segmentation model segments the object of interest. It is the fraction of pixels that are correctly classified as positive pixels [26]. The equation to

calculate precision is as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.23}$$

Precision ranges from 0 to 1. A higher precision means more true positives and fewer false positives, therefore the closer precision is to one, the better the segmentation method.

2.4.3 Recall

Recall is a metric that determines the fraction of the object of interest that is correctly classified as true positives. Mathematically recall can be described as:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.24}$$

Recall range from 0 to 1. A higher recall value means that a greater fraction of the object of interest has been detected i.e. more true positives and less false negatives. Therefore the closer recall is to 1, the better the segmentation method [26].

Interpretations of Precision and Recall:

- High precision but low recall: The objects that are detected are mostly correct, but most of the ground-truth objects have not been detected.
- High recall but low precision: Most ground-truth objects have been detected, but many false positives are also detected.
- High precision and high recall: Most ground-truths are accurately detected.

2.4.4 F1 Score

Precision and recall on their own are not sufficient to provide a reliable evaluation of the segmentation model. The F1 Score is an evaluation metric that incorporates both precision and recall to compute the harmonic mean between them as follows:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.25}$$

$$\text{i.e. F1 Score} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{2.26}$$

The F1-score range from 0 to 1 and it is only high when both precision and recall are high. Therefore, the higher the F1-score, the better the performance of the segmentation method [111].

2.4.5 Intersection over Union

Intersection over Union (IoU) measures the area of overlap between the predicted bounding box and the ground-truth bounding box. IoU is computed as the area of intersection between the predicted and ground-truth bounding boxes divided by their union as shown in figure 2.19. Equation 2.27 shows how to calculate the IoU metric.

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{2.27}$$

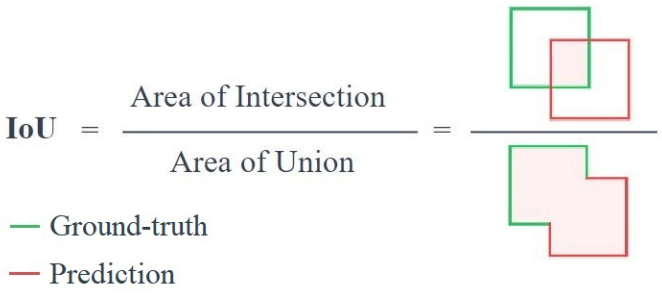

FIGURE 2.19: *A visual representation of the Intersection Over Union calculation [26].*

IoU range from 0 to 1. As shown in figure 2.20, an IoU of 0 means that the bounding boxes do not overlap whereas an IoU of 1 means that they completely overlap. Therefore, the greater the

IoU, the better the prediction of the segmentation method [11].

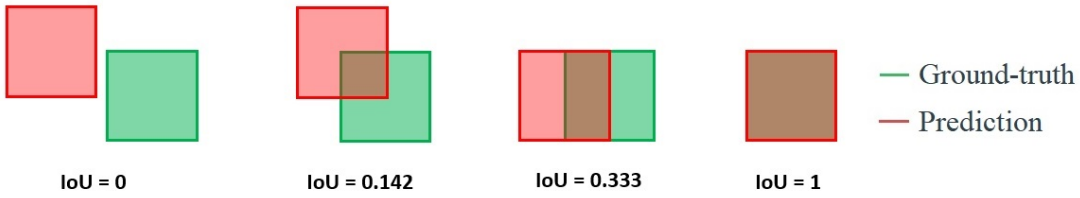

FIGURE 2.20: *Examples of different IoU values [11].*

2.5 Conclusion

This chapter provided an in-depth literature review related to computer vision techniques. The first section distinguishes between the supervised learning, unsupervised learning and reinforcement learning types in machine learning followed by a discussion of some popular supervised learning classification algorithms. Deep learning, which is a subsection of machine learning, was discussed in the following section. Deep learning makes use of artificial neural networks to detect objects within images. The third section provides a review of image segmentation methods which divide images into multiple segments that exhibit similar features. Finally, the fourth section reviews the metrics that evaluate the performance of object detection and image segmentation algorithms.

CHAPTER 3

Data Collection and Preparation

This chapter describes the data collection process and the preparation of the data before it was utilised for image processing and algorithm learning to detect and segment material other than grapes within a digital image.

3.1 Collection of Images

Images of machine-harvested grape loads were collected throughout the data-gathering process. During the time of this research, there were no open-source data set of machine-harvested grape load photographs that have been made accessible for research. Therefore, as a critical initial contribution to this study, it was required to create an image data set of machine-harvested wine grape loads from scratch.

The aforementioned photographs were obtained towards the latter stages of the harvest period in the year 2022, utilising an iPhone 7 camera equipped with a 12-megapixel sensor and f/1.8 aperture. Subsequently, the photographs were saved in the JPEG file format with a 1512 x 2016 resolution. The photographs were acquired outdoors in natural illumination at a winery, specifically at their weighbridge and crusher station. The data was acquired through the secure fixation of the iPhone onto a selfie stick, by suspending the device vertically above the grape loads in order to capture images. Challenges were encountered at the weighbridge due to shadows initially cast onto the grape loads by surrounding trees. Undesirable shadows on image data may affect the performance of image processing algorithms. The identified issue was alleviated by

positioning a board above the iPhone to obviate the impact of shadows cast by the surrounding trees.

3.2 Annotations

Within the context of supervised learning, it is necessary for data to be annotated or labelled prior to its utilisation within learning algorithms. In this research project, the objective was to perform image segmentation to separate the objects of interest from the background to determine the pixel area that is constituted by MOG in an image, rather than merely detecting the MOG. It was therefore deemed necessary to perform pixel-wise polygon annotations that precisely encompass the objects of interest instead of only drawing bounding boxes around the objects of interest as with most object detection tasks. However, the process of annotating all the images with polygons necessitated substantially more effort than bounding boxes would have. The annotation process was executed by utilising an online open-source image annotation tool, CVAT (Computer Vision Annotation Tool) to attain accurate and reliable ground truth pixels. The choice of this particular tool was based on its user-friendly interface and accessibility without any cost. The data annotated with identifying markers were stored in JSON format.

Material other than grapes includes leaves, petioles, canes, wood materials as well as insects and mice and other possible foreign objects like strainer wire. However, some of the before-mentioned items like mice and foreign objects appear very seldom in grape loads, On the contrary, no foreign objects and only two mice were identified during the data collection process. Therefore it was not possible to create a data set of those items. Petioles form part of a leaf structure. During the initial stages of the annotation process, the annotation of petioles posed a formidable task due to the orientation and occlusion of leaves. Considering the close association of petioles with leaves, it was deemed appropriate to include them in the class of leaves. As a result materials other than grapes were categorised into three distinct classes, namely Leaf, Cane, and Wood.

Figures 3.1, 3.2 and 3.3 depict zoomed-in representations of an annotated leaf in green, cane in yellow and piece of wood in red respectively.

FIGURE 3.1: *Example of an annotated leaf at the bottom left.*

FIGURE 3.2: *Example of an annotated cane at the top left.*

FIGURE 3.3: *Example of an annotated wood piece at the top right.*

3.3 Image Augmentation

The concept of image augmentation entails utilising image processing techniques to artificially expand a data set i.e. to create new and transformed data from the existing data set. [96, 101]. Many state-of-the-art deep learning models require a vast amount of annotated data to perform well, therefore, image augmentation proves to be a helpful technique when confronted with a restricted data set.

The following image augmentation techniques were utilised to expand the data set:

3.3.1 Image Flipping

In this technique, the images were flipped along both the vertical and horizontal axes as illustrated in figure 3.4 which expanded the data set to four times the size of the original data set.

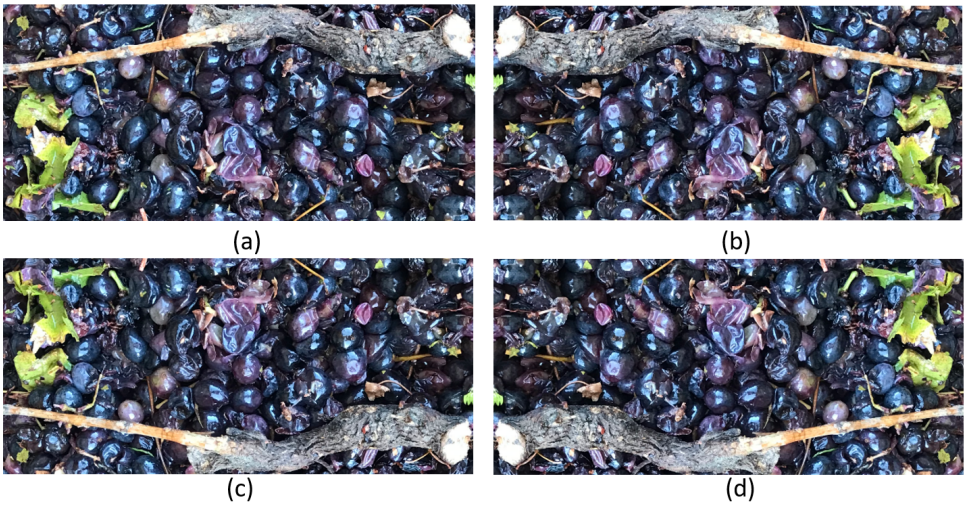

FIGURE 3.4: *(a) Original image, (b) flipped along the vertical axis, (c) flipped along the horizontal axis, (d) flipped along both the vertical and horizontal axis.*

3.3.2 Random Image Rotation

The rotation technique expands the data set by slightly rotating the images at different angles. Figure 3.5 shows how new images were created by rotating the original image at 15° in the clockwise and anti-clockwise directions.

FIGURE 3.5: *Rotation of the original image at 15° clockwise and anti-clockwise.*

The implementation of these augmentation techniques was deemed appropriate for the given data set, as MOG objects can appear in various orientations. Consequently, the data set was augmented without any anomalous images.

CHAPTER 4

Evaluation of Image Segmentation Methods

The evaluation and selection of image segmentation methods is an indispensable element of this project. This chapter aims to evaluate traditional and deep learning-based image segmentation methods that are appropriate to detect and segment MOG within a wine grape load image. It is important to not only detect MOG in an image but also do a pixel-wise segmentation to obtain masks of the objects of interest because the goal is to determine the pixel area in an image that is covered by MOG.

4.1 Traditional Image Segmentation Methods

In this section, some of the traditional image segmentation methods as described in Chapter 2.3 are explored and implemented in Python and OpenCV (a Python library enabling the execution of image processing and computer vision operations) for the use of this project.

4.1.1 Development of Image Segmentation Process

The global thresholding method is the most basic and least computationally expensive method for image segmentation. This method works well to separate objects from the background if there is a considerable difference in pixel intensity values between the objects and the background. Figure 4.1 is an example of a photo of wine grapes with leaves and sticks and in the rest of this section this image will be referred to as the "original image". It can be seen that the MOG objects are visually lighter than the grapes, therefore global thresholding was used as the initial

thresholding technique to identify MOG from the grapes. First, the images were converted to greyscale and then global thresholding was applied to the images for manually selected threshold values of 65, 127 and 190 to visualise the performance of low, mid and high threshold values.

FIGURE 4.1: *Example of grapes with MOG*

Figure 4.2 shows the masks in red that were created on top of the original image after the global thresholding technique was applied for the different threshold values. It can be seen that for a threshold value of 65, the image was segmented into too many different regions, for a threshold value of 190 the image was segmented into too few regions and for a threshold value of 127 some areas that should have been segmented were not segmented and other areas that should not have been segmented were segmented.

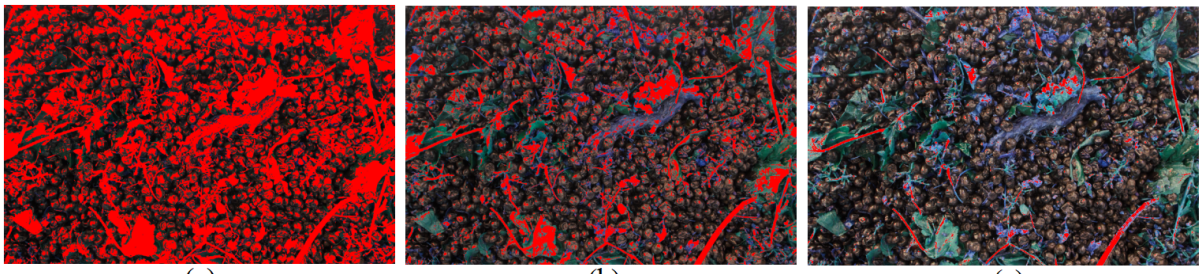

FIGURE 4.2: *Results for the global thresholding method; (a) Threshold value of 65; (b) Threshold value of 127; (c) Threshold value of 190.*

Otsu's threshold method, a method proposed by Nobuyuki Otsu in 1979 was used in an attempt to improve on the segmentation. This method assumes that an image contains two classes of pixels that are distributed along a bi-modal histogram; then it automatically searches for an optimal threshold so that the combined intra-class variance is minimal [13]. This method's ability to automatically find an optimal threshold is advantageous because it can adapt to varying pixel intensities of photos taken at different times of the day.

Otsu's threshold method was applied to figure 4.1 with the optimal threshold computed as 89. The resulting mask is shown in figure 4.3. It can be seen that this method also did a poor job of correctly segmenting the image since there are many false-positive pixels present on the grapes due to light reflecting on the wet surfaces of the grapes. The reason for this poor segmentation is that Otsu's threshold method operated entirely on the pixel intensity of the greyscale image and the distribution of pixel intensities as shown in the intensity histogram in figure 4.4 does not create a bi-modal histogram i.e. there is no valley between peaks where an ideal threshold value can be selected.

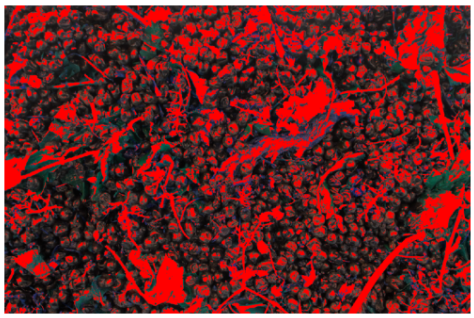

FIGURE 4.3: *Otsu Thresholding of figure 4.1*

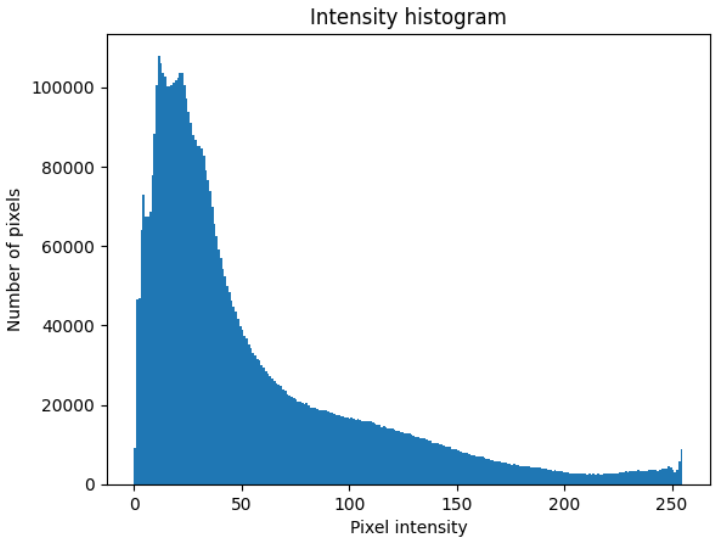

FIGURE 4.4: *Intensity Histogram of figure 4.1*

The observation was made that merely applying thresholding on greyscale images would not be adequate, as the presence of light can be deceptive and thereby distort the resulting seg-

mentation mask. Consequently, a decision was made to investigate the impact that the Hue Saturation Value (HSV) colour space would have on subsequent image segmentation. HSV offers enhanced exploitation of the image's colour information in comparison to RGB and it conveys how colours are perceived in terms of light. Figure 4.5 is an illustration of the HSV colour space. It demonstrates that hue represents the colour and is found in its purest form on the perimeter of the cone's base, saturation reflects the dominance of hue in the colour and a high saturation indicates a high dominance, and value represents the intensity of light. [64]. The RGB images were converted to the HSV colour space and figure 4.6 represents the HSV colour space of the original image in figure 4.1.

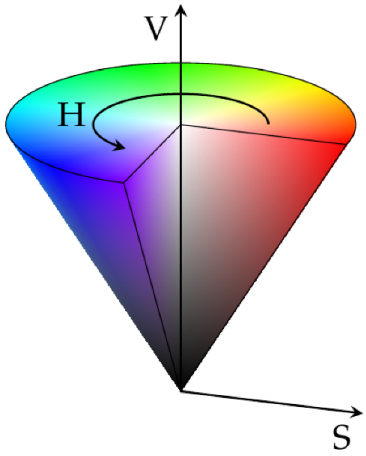

FIGURE 4.5: *Hue(H) Saturation(S) Value(V) colour space illustration [59]*

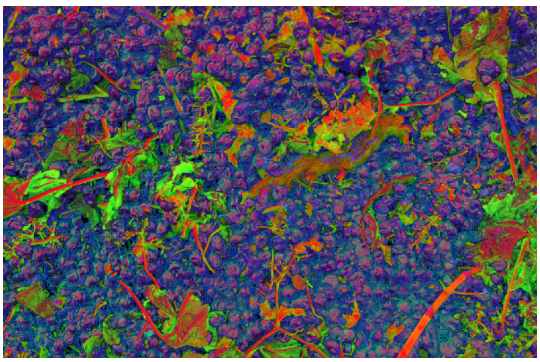

FIGURE 4.6: *Original image in HSV colour space.*

The hue, saturation, and value channels were then extracted from 4.6 and they are presented in 4.7 along with their pixel histograms in figure 4.8. Otsu's thresholding was applied to these channels and individual segmentation masks were created on them as shown in figure 4.9. Visual inspection reveals that the mask created on the hue channel outperforms the masks created on

the saturation and value channels, and it is a significant improvement over the mask created on the greyscale image, as there are far fewer false-positive pixels and most of the objects of interest have been segmented. The reason for this improved mask is that Otsu's threshold method was completely applied to the colours of the image and light did not have an effect. It is also noticeable that the hue pixel histogram in figure 4.8 exhibits a feature whereby it creates a valley that separates two peaks. This feature renders the hue channel conducive for the segmentation purposes of separating the MOG from the background. Consequently, the masks generated from the hue channels using Otsu's threshold method were utilised for additional segmentation.

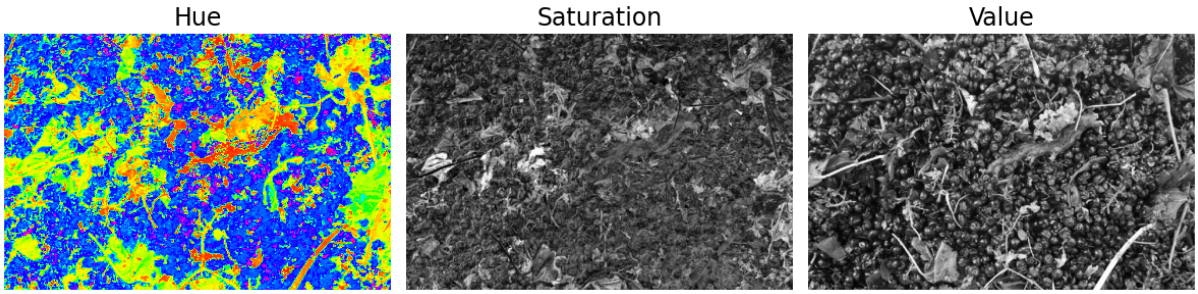

FIGURE 4.7: *Hue, saturation and value channels of figure 4.6.*

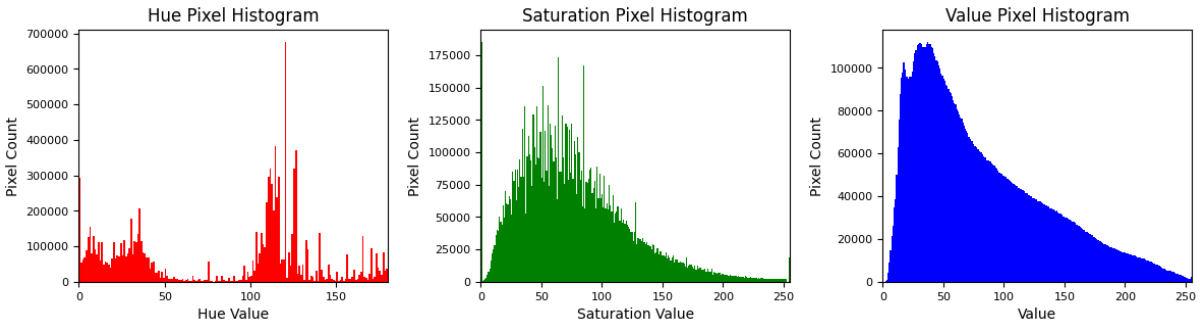

FIGURE 4.8: *Hue, saturation and value pixel histograms.*

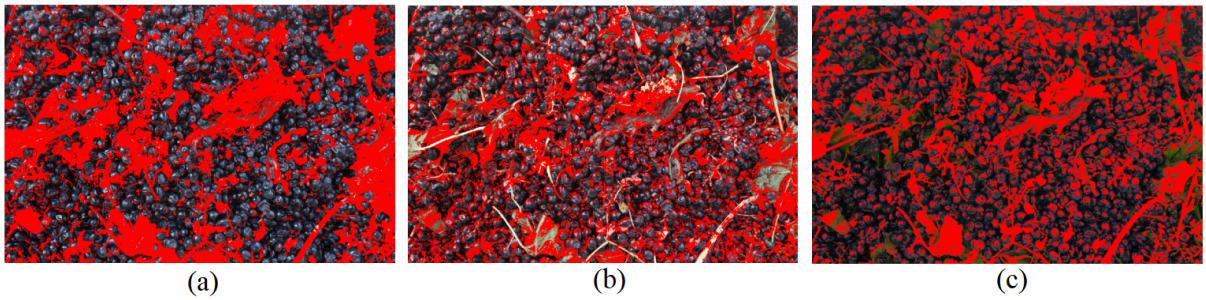

FIGURE 4.9: *Otsu thresholding on hue, saturation and value channels.*

Morphological operations were then executed on the masks generated from the hue channels to minimise the false-positive and false-negative pixels on the masks. Morphological image processing applies a kernel with a set of non-linear operations as set out in table 4.1 to a binary

image to remove imperfections [34]. These operations produce new binary images of the same size, but the pixel values will depend on a comparison of the adjacent pixels in the original image. [74].

Through a process of experimentation with different morphological operations for numerous iterations, it was visually observed that the masks could be enhanced by applying the following combination of morphological operations: First, the masks should be eroded with a 15 x 15 kernel size for one iteration to remove some unwanted pixels. After erosion, a 2 x 2 dilation kernel should be applied for 25 iterations to fill the holes in the mask. This process produced improved and smoother masks with fewer false-positive and more true-positive pixels.

TABLE 4.1: *Morphological image processing operations [74]*

Morphological Operations		
Fundamental Operations	Erosion	A pixel is set to 0 if any adjacent pixels have the value of 0 i.e. floating pixels are removed and shapes appear smaller.
	Dilation	A pixel is set to 1 if any adjacent pixels have the value of 1 i.e. holes in objects are filled and shapes appear bigger.
Compound Operations	Opening	The image is first eroded, then dilated.
	Closing	The image is first dilated, then eroded.

Following morphological operations on the masks containing the objects of interest, i.e. the MOG, additional thresholding was used to separate the leaves, canes and wood in the hue channel. For this segmentation, the multi-thresholding technique was used and three hue ranges were set to indicate where thresholding in the hue range should occur. These ranges were chosen by comparing the hue colours of the leaves canes and wood in a set of diverse images with the values that the hue colours represent and then computing the mean range of hue values for each object. It was determined that the hue values for wood range from 0 to 8, for canes from 9 to 17 and for leaves it ranges from 18 to 73.

The original mask was further segmented into the three regions mentioned above and the resulting masks for leaves, canes and wood can be seen in figure 4.10. It can be seen that the green leaves were quite well segmented, but in the image where the wood and canes were segmented, there are

also areas segmented that do not contain wood or cane objects. These areas are occupied mainly by dry leaves which have a similar colour to wood and canes. These dried leaves create unwanted noise and in order to remove some of the noise and improve the masks, morphological operations were applied to the individual masks. Again by experimentation with different morphological operations for numerous iterations, it was decided to erode the wood and canes mask with a 2 x 2 kernel for two iterations and to apply the closing operator to the leaves mask with a 2 x 2 kernel for five iterations. After the morphological operations were applied the final segments for the wood and leaves were produced.

FIGURE 4.10: *Segment masks for (a) wood, (b) canes and (c) leaves.*

The flowchart in figure 4.11 depicts the proposed segmentation process using traditional segmentation methods.

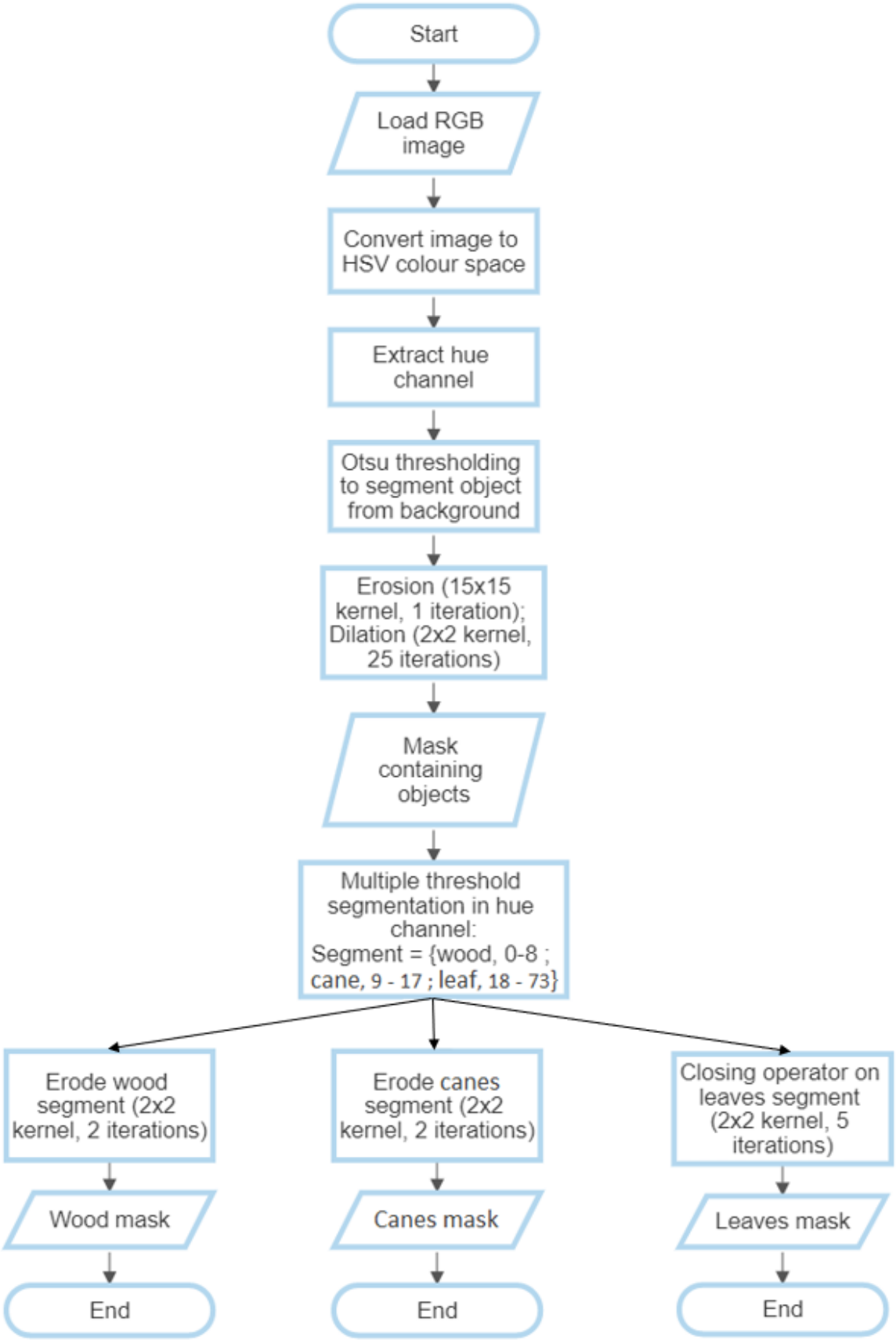

FIGURE 4.11: *Flowchart for proposed image segmentation process.*

4.1.2 Evaluation of Proposed Image Segmentation Process

To evaluate the performance of the segmentation process depicted in figure 4.11, with respect to the metrics outlined in Chapter 2.4, the grape load image data set as described in Chapter 3 was employed to compare the segmented pixels against their ground truth counterparts. The segmentation performance of each class was evaluated to obtain separate scores as shown in table 4.2. The overall segmentation performance for all classes is documented in table 4.3.

TABLE 4.2: *Performance evaluation of each class of proposed image segmentation process.*

Metric	Score (%)		
	Wood	Cane	Leaf
Accuracy	93.59	93.58	86.88
Precision	27.94	12.29	67.14
Recall	55.93	57.90	75.65
F1 Score	37.26	20.27	71.14
IoU	22.90	11.28	55.21

TABLE 4.3: *Overall performance evaluation of image segmentation process.*

Metric	Score (%)
Accuracy	91.35
Precision	35.79
Recall	63.16
F1 Score	42.89
IoU	29.80

It can be seen in table 4.3 that the overall accuracy score is quite high, however, this metric only measures the correctly classified pixels and does not take the pixels that were incorrectly classified into account. Therefore precision, recall, F1 score and IoU were also computed with Recall and F1 score being the most valuable metrics. Recall because it is important for the wine cellar to know what fraction of the MOG objects have been detected, but also F1 score that considers pixels that have been falsely detected as MOG objects which is important for the sake of the wine grower. The metrics of concern in table 4.3 are precision, F1 score and IoU as they are all below 50 %. By looking at table 4.2 it is clear that the wood and cane classes cause the low scores of the concerning metrics. By considering the equations used to determine these scores, it is evident that the root of the poor scores is that too many pixels were falsely detected as wood or canes. As a result, the performance of the proposed segmentation process with

traditional methods is not satisfactory. Consequently, deep learning algorithms were explored in the following section.

4.2 Deep Learning Based Image Segmentation Methods

Deep learning has the ability to learn representations of data by extracting characteristics from images throughout the layers of a deep neural network. Prior to the emergence of deep learning-based image segmentation, there has been extensive exploration and utilisation of deep learning methodologies for object detection techniques. Prominent deep learning techniques for object detection comprise the R-CNN and YOLO methods as explained in chapter 2.2.2 as well as refined versions that derived from these methods, such as Fast R-CNN, Faster R-CNN and numerous YOLO variations. Advancements in image analysis with deep learning have prompted certain researchers to investigate the segmentation of objects in images utilising deep learning methods. For example, it has been found in literary analysis that S. Aich *et al.* counted leaves by segmenting them from a simple background using a deep learning framework consisting of a deconvolutional neural network to perform initial segmentation followed by a convolutional neural network to count the leaves [3]. H. Scharr *et al.* undertook a study to analyse four distinct deep learning techniques for the purpose of segmenting leaves within a given image, but these methods lacked accuracy with images consisting of younger and overlapping leaves [102]. A shallow CNN with a Canny edge detector proposed by J. Bell *et al.* was effective in identifying occluding leaves with a simple background, however, a limitation identified in this study is that the CNNs were trained on the same data as the test data [15]. K. Simonyan *et al.* investigated what effect the depth of a CNN has on comprehensive image recognition accuracy and it was found that more layers in the CNN provide better performance [107]. K. He *et al.* proposed the latest state-of-the-art framework for object instance segmentation called Mask Region Convolutional Neural Network (Mask R-CNN). Mask R-CNN is an enhanced version of its predecessors namely R-CNN, Fast R-CNN and Faster R-CNN owing to the incorporation of an additional branch that enables it to not only predict a bounding box of an object but also segment regions containing objects at the pixel level, however, it does incur a computational cost [47]. Figure 4.12 shows how the R-CNN model evolved into the Mask R-CNN model.

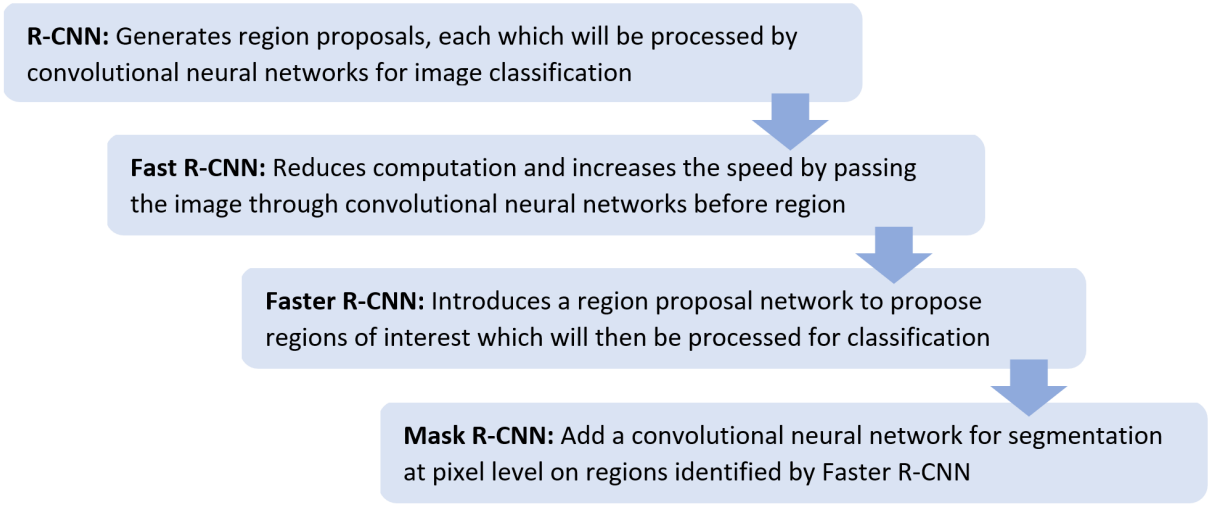

FIGURE 4.12: *The evolution of Mask R-CNN [19].*

Despite its sophisticated approach, the Mask R-CNN model does not introduce entirely new building blocks and concepts, as these elements have previously proven to be successful [33]. The Mask R-CNN model can identify and segment multiple objects in a single image with various shapes and sizes which makes it relevant to detect leaves and other impurities that can have many different appearances in grape loads. The goal of the computer vision model in this project is that it should perform accurate pixel-wise segmentation to separate MOG from the grapes, therefore the Mask R-CNN model is explored for the remainder of this section and its performance on the MOG data set is evaluated.

4.2.1 Mask R-CNN Framework

The Mask R-CNN framework broadly consists of two connected parts as shown in the flowchart in figure 4.13.

Part 1: Part one is dedicated to extracting features from images to propose possible bounding box regions. First, the image is fed into the backbone which acts as the main CNN of the model to extract relevant features from the images to produce feature maps. The Region Proposal Network (RPN) performs the task indicated by its name which involves generating proposals for regions that potentially encompass objects within the feature maps [53, 122].

Part 2: Part two classifies and predicts the coordinates (box regression) of the bounding boxes in the image whilst simultaneously generating masks at the pixel level of the objects. The ROI

(Region of Interest) pooling layer converts and aligns the regions generated by the RPN to the same sizes while maintaining the dimensional properties. In the left branch of the framework in figure 4.13 the aligned regions are passed through fully connected layers to classify the objects in the regions while the box regressor refines the obtained regions for a more closely fitted bounding box. In the right branch, the regions from the ROI layer that contains objects are passed through a CNN to create segmentation masks [122].

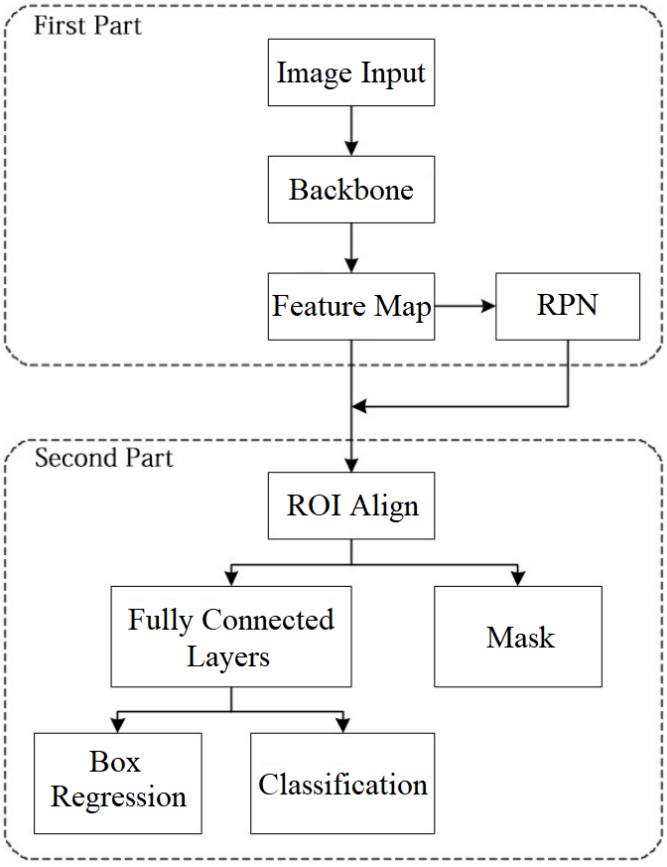

FIGURE 4.13: *Mask R-CNN flowchart [122]*

4.2.2 Backbone Selection

The backbone of the Mask R-CNN architecture is a conventional CNN that extracts features from the input images. The initial layers of the CNN detect low-level features such as edges or corners. The final layers of the CNN detect and classify higher-level features, such as the leaves, canes and wood [41]. The selection of a backbone architecture affects the accuracy and speed of the model.

During a literature search to identify a backbone suitable for this study, three models were found to be relevant namely ResNet-101, ResNet-50 and MobileNet-V2. ResNet which is short for residual network and MobileNet are certain types of CNNs introduced by K. He *et al.* and Google researchers respectively [48, 90]. K. Yang *et al.* employed the Mask R-CNN model with the ResNet-101 architecture as the backbone to segment and classify fifteen different leaf species within images with complex backgrounds while X. Yang *et al.* used an identical model to segment and classify plant leaves within images [122, 123]. V. Giuffrida *et al.* employed a model of the same family as the aforementioned, the ResNet-50 model to segment and count rosette-shaped leaves in images [40]. In another paper, E. Badeka *et al.* detected vine trunks in images of vineyards implementing the MobileNet-V2 model [10].

To select the optimal backbone model, the ResNet-101, ResNet-50, and MobileNet-V2 models pre-trained on the ImageNet data set were compared according to the following criteria [72]:

1. **Top-1 error:** This error arises when the class predicted by the model with the greatest confidence does not align with the actual class. A lower top-1 error indicates a model with better accuracy.
2. **Inference time on the GPU (Graphical Processing Unit):** The time it takes the model to run data points through the model and to make a prediction. A lower inference time means the model can make faster predictions which indicates a better model i.e. the lower the inference time, the better is the model.
3. **Model size:** This is the space in terms of megabytes the pre-trained model occupies within a system. It is ideal to have a model that does not occupy a lot of space. This criterion is especially important when the model is to be used on a mobile device, however, in this project the model is not intended to be used on a mobile device and therefore it is regarded as the least important criterion compared to the criteria above.

In accordance with the criteria outlined above, figure 4.14 depicts a bubble chart that illustrates information pertaining to multiple pre-trained models. The X-axis represents the top-1 error as a percentage, the Y-axis represents inference time on the GPU in milliseconds and the bubble size represents the size of the model [72]. The ResNet-101, ResNet-50, and MobileNet-V2 models

which are of interest in this study, are delineated by orange squares.

In the comparison between the ResNet-50 and ResNet-101 models, the ResNet-50 model is smaller in size and its inference time is approximately half that of the ResNet-101 model inference time when executed on a GPU. Furthermore, the ResNet-101 model only offers a marginal advantage of approximately 1 % in terms of its top-1 error. As a result, the ResNet-50 model has been regarded as the better model.

The inference times for the ResNet-50 and MobileNet-V2 models are similar. The MobileNet-V2 model is smaller, however, the ResNet-50 model has an approximately 4.5 % better performance on the top-1 error. Since the MOG detection system is not intended to be used on a mobile device the top-1 error is more important than the size of the model. Consequently, the ResNet-50 model emerges as the optimal architecture to be used as the backbone of the Mask R-CNN model within this study.

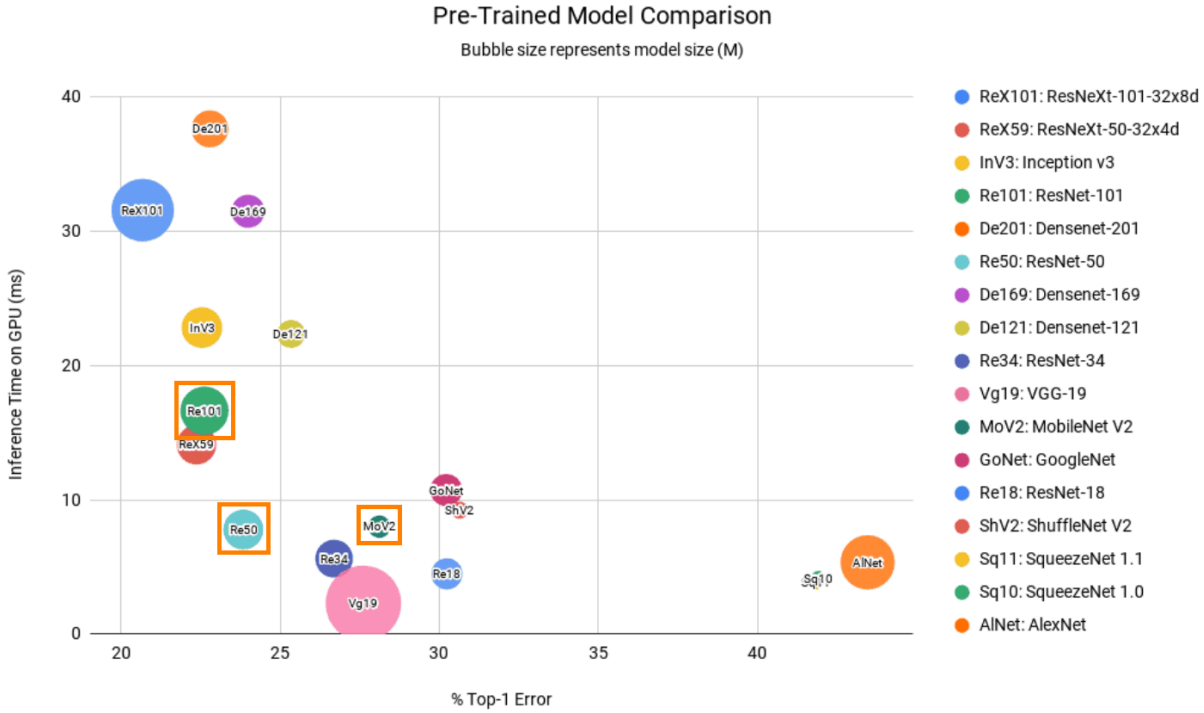

FIGURE 4.14: *Bubble chart comparison of pre-trained models [72].*

The architecture of the ResNet-50 model is shown in figure 4.15 and it consists of the following layers in this order [80]:

1. An input layer that receives images with an input size of 224 x 224 x 3.
2. A convolutional layer with 64 7 x 7 kernels.
3. A 3 x 3 maximum pooling layer.
4. The layers in the yellow block of figure 4.15 are repeated three times: A convolutional layer with 64 1 x 1 kernels, a convolutional layer with 64 3 x 3 kernels and a convolutional layer with 256 1 x 1 kernels.
5. The layers in the blue block are repeated four times: A convolutional layer with 128 1 x 1 kernels, a convolutional layer with 128 3 x 3 kernels and a convolutional layer with 512 1 x 1 kernels.
6. The layers in the green block are repeated six times: A convolutional layer with 256 1 x 1 kernels, a convolutional layer with 256 3 x 3 kernels and a convolutional layer with 1024 1 x 1 kernels.
7. The layers in the purple block are repeated three times: A convolutional layer with 512 1 x 1 kernels, a convolutional layer with 512 3 x 3 kernels and a convolutional layer with 2048 1 x 1 kernels.
8. An average pooling layer.
9. A flattening layer.
10. Fully connected layers of 1000 neurons with a softmax activation function in the output layer.

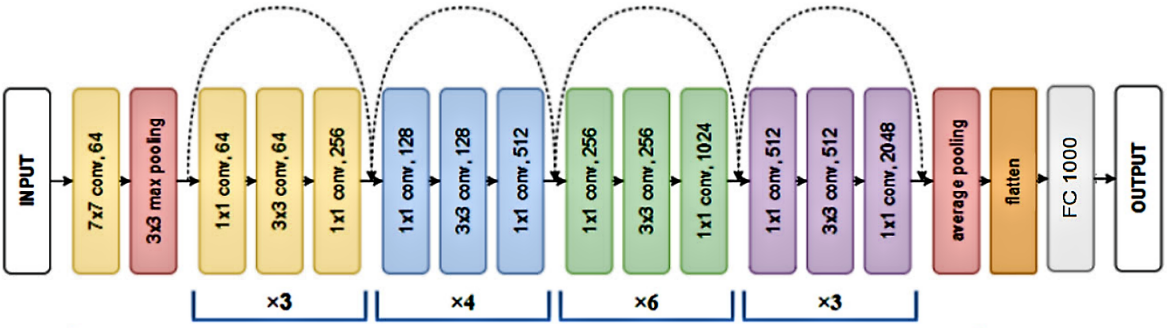

FIGURE 4.15: *ResNet-50 model architecture [1].*

4.2.3 Mask R-CNN Training Hyperparameters

Several hyperparameters are involved to control the Mask R-CNN learning process. These parameters are explained below.

4.2.3.1 Model Initialisation

The Mask R-CNN model should be initialised with weights before it can be trained. Neural networks are often initialised with arbitrary values assigned to weights [60]. However, in this project transfer learning was used to initialise the Mask R-CNN model with pre-trained weights that were trained on the extensive ImageNet data set. Transfer learning is a machine learning technique that transfers the knowledge gained from a previous training task with a large data set to a new task with a smaller data set [104]. Initialising the model with transfer learning improves training performance, especially when limited data is available.

4.2.3.2 Batch Size

The training data set is partitioned into multiple smaller batches, which are dependent upon the predetermined size of the batch. The batch size specifies the number of images that are fed through the model before the weights of the model are updated. The utilisation of large batch sizes results in an increased demand for the system's memory and can potentially enhance the computational efficiency of the process. Conversely, decreased batch sizes offer the benefits of lower memory requirements and allow for more frequent weight updates. The choice of batch size can have a significant effect on the training accuracy and generalisation [117]. The optimal batch size for the training process of the Mask R-CNN model was selected based on the machine's hardware constraints as outlined in the subsequent section.

4.2.3.3 Optimiser

An optimiser enhances the performance of a deep learning model by reducing the total loss and increasing its accuracy. The optimiser updates the weights of the model in an optimal manner with the objective of minimising the loss function. The weights are updated in accordance with the gradients that were computed through the backpropagation process [44]. The ADAM optimiser which is currently the most popular optimiser to train neural networks was used in the

ResNet-50 backbone [112].

4.2.3.4 Learning Rate

The learning rate denotes the extent of modification to the weights of the model. High learning rates can accelerate the convergence of a model, however, if it is too high it can also cause the model to overshoot or oscillate. Learning rates that are too low may cause the model to get stuck in a local minima. During the course of training, adjustments to the learning rate are made to find an optimal learning rate for a good convergence yet with a reasonable training length [29]. An initial learning rate of 0.00001 was selected.

4.2.3.5 Epochs

Epochs refer to the frequency of iterations during which the entire training data set is propagated through the neural network model for the purpose of updating the internal weights. The training performance of a model tends to improve as the number of epochs increases until it reaches the convergence point. However, beyond this point, the model starts to learn irrelevant information resulting in difficulty to generalise to new unseen data [51].

4.2.3.6 Loss Function

The loss function serves the purpose of quantifying the total error between the anticipated output and actual output of a neural network pertaining to a specific data set [24]. The loss function is employed for the purpose of determining whether the neural network's performance is improving across sequential epochs. If a decrease in loss is observed while the number of epochs increases, it indicates that the neural network is continuing to improve its training performance. If the loss function starts to oscillate around a singular point while the number of epochs increases, then the neural network cannot improve its training performance any further [60].

4.2.4 Mask R-CNN Training and Evaluation

The specifications of the hardware used for the training process are first discussed followed by a discussion of the training and evaluation process.

4.2.4.1 Hardware Specifications

The training of the Mask R-CNN model was initially executed on a Lenovo G50-70 laptop with the following specifications:

- Intel Core i5-4210U CPU (Central Processing Unit) @ 2.4GHz
- 12GB DDR3 Ram
- Intel HD Graphics 4400

However, a shortcoming of the laptop was the absence of a built-in GPU and it was soon realised that training solely on a CPU would result in a time-consuming process, taking several hours to days to complete, provided the training process did not encounter any failures. Training on a GPU can accelerate the training process significantly. Therefore, an exploration of cloud-based machine learning platforms emerged as a viable alternative given that the acquisition of a costly laptop equipped with an appropriate GPU was deemed impractical. There are several cloud-based machine learning platforms that offer high-performance GPUs, including Amazon Web Services, Google Cloud, Google Colaboratory (Google Colab) and Microsoft Azure, among others. The Google Colab platform was chosen as the preferred option due to being freely available for use with the latest version of Python and libraries pre-installed. In contrast, the other cloud-based platforms typically operate on a rental basis, charging users on an hourly basis.

The specifications of the virtual machine underlying the Google Colab platform are as follows:

- Intel Xeon CPU @ 2.20GHz
- 13GB Ram
- Tesla K80 GPU
- 12 GB DDR5 VRAM

4.2.4.2 Training and Evaluation

The grape load image data set, as described in Chapter 3, was partitioned into training, validation, and testing sets using an 80:10:10 ratio. All images were resized to 224 x 224 which is the size required for the input layer of the ResNet-50. The Mask R-CNN model was initially trained

for 500 epochs with a batch size of 16 which was the maximum batch size that the system's memory (ram) could accommodate.

Figure 4.16 presents the graphical representation of the training and validation loss curves across a span of 500 epochs while the results for the training and validation accuracy at various intervals during the course of training are documented in table 4.4. From the graph, it is apparent that the training and validation loss curves are approaching convergence from around 150 epochs and are fully converged at 300 epochs with loss values of about 0.2. In table 4.4 it can be seen that the rise in training and validation accuracy slowed down from 100 epochs. At the 300th epoch, the training accuracy reached 94 % while a validation accuracy of 93 % was attained. However, a decline in the validation accuracy was observed at the 400th epoch, dropping to 92 %. This phenomenon is referred to as overfitting, whereby the training accuracy consistently improves while the validation accuracy begins to decline due to the incorporation of noise into the learned model. The model was subsequently retrained for a limited duration of 300 epochs in order to attain its peak validation accuracy of 94 %.

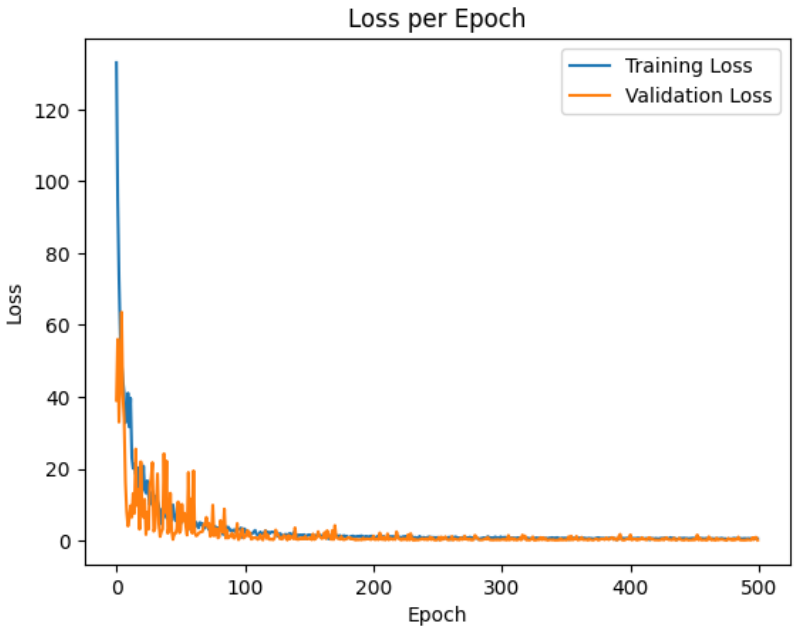

FIGURE 4.16: *Training and validation loss per epoch.*

The masks generated by the Mask R-CNN model on the testing data set were evaluated using the metrics specified in Chapter 2.4. Table 4.5 displays the mean scores attained from the testing data

TABLE 4.4: *Training and validation accuracy.*

Epochs	Training Accuracy	Validation Accuracy
5	49 %	45 %
25	76 %	71 %
50	86 %	83 %
100	91 %	87 %
150	92 %	90 %
200	93 %	91 %
250	93 %	93 %
300	94 %	93 %
400	94 %	92 %
500	94 %	92 %

set compared to the segmentation performance of the traditional image segmentation method described in Chapter 4.1.

TABLE 4.5: *Mask R-CNN vs Traditional image segmentation.*

Metric	Segmentation Method Score (%)	
	Mask R-CNN	Traditional
Accuracy	93.38	91.35
Precision	85.21	35.79
Recall	86.35	63.16
F1 Score	85.78	42.89
IoU	75.09	29.8

The superiority of the Mask R-CNN model over the traditional image segmentation method is clearly demonstrated in table 4.5, as it outperforms the latter comprehensively in all aspects. The recall and F1 scores which are the most valuable metrics for this study are both greater than 85 %. These results indicate that the model possesses the capability to accurately identify the majority of pixels associated with the MOG within a digital image, with a minimal occurrence of misclassification where pixels are falsely categorized as part of MOG. Consequently, the Mask R-CNN model will be able to reliably detect and segment MOG within an image of a wine grape load.

4.3 Conclusion

This chapter evaluated the performance of both traditional and deep learning approaches for image segmentation on a data set comprising images of machine-harvested wine grape loads. The traditional method for image segmentation involved the utilisation of thresholding techniques

on images that were first transformed into the HSV colour space. Morphological operations were implemented to enhance the masks generated from the segmentation process. For the deep learning based method it was identified that the Mask R-CNN model with a ResNet-50 CNN acting as the backbone is a suitable model for this study. The performance of both the traditional and deep learning based image segmentation methods was evaluated according to the metrics outlined in Chapter 2.4 with recall and F1 score being the most valuable metrics. The chapter concluded with the demonstration that the Mask R-CNN model vastly outperformed the traditional image segmentation method and that it will be able to reliably detect and segment MOG within an image of a wine grape load.

CHAPTER 5

Statistical Analysis of MOG in Grape Load

The objective of this chapter was to investigate how the quantity of MOG on the surface of a grape load relates to the total MOG present within a harvested grape load in terms of mass by conducting a statistical analysis on samples collected from grape loads. The samples were used to represent the grape load population that grows with every harvesting season. The first step was to determine whether a linear relationship exists between the MOG on the surface of a grape load and the mass of MOG in the corresponding grape load. If a linear relationship exists, the second step was to determine the amount of mass that the MOG on the surface is associated with.

5.1 Collection of Sample Data

A commonly accepted guideline among statisticians is that a sample size of thirty or greater is deemed sufficient to offer a valid representation of a given population [37]. Therefore for this study, a total of thirty random grape load samples were collected from various grape loads arriving at the cellar from different farmers by scooping the grapes together with the MOG into containers. The container's dimensions measured 44 cm x 30 cm x 20 cm, which is an adequate size to ensure a representative sample of the grape load. First, a top-down photograph of the container was acquired. Subsequent to the measurement of the mass of the container, a process of assorting the leaves, canes, and wood was carried out, where each component was individually weighed. The mass percentage of the leaves, canes, and wood in relation to the total mass of

the sample was obtained. Afterwards, ground truth pixels of the MOG present in each image of the sample data were obtained with the CVAT annotation tool. The percentage of pixels within the images designated to leaves, canes and wood together with the mass percentages were documented in table 5.1. The average mass and pixel percentage of each MOG component in the sample are also indicated in the table.

TABLE 5.1: *Mass and pixel percentages of a grape load sample.*

Sample	Leaf mass %	Cane mass %	Wood mass%	Leaf pixel %	Cane pixel %	Wood pixel %
1	0.53	0.30	0.17	13.32	0.68	0.29
2	0.61	0.72	0.52	7.99	1.82	1.28
3	0.33	0.21	0.24	6.74	0.67	0.38
4	0.64	0.58	0.49	20.45	1.41	0.81
5	0.33	0.26	0.17	13.66	0.86	0.30
6	0.84	0.37	0.65	20.10	1.22	1.18
7	1.22	1.30	1.14	23.40	2.13	3.00
8	0.48	0.35	0.24	10.13	0.94	0.40
9	1.08	1.15	0.98	20.01	1.86	2.19
10	0.75	0.82	0.79	11.38	2.10	2.10
11	1.15	0.37	1.09	24.10	1.87	4.69
12	0.68	0.47	0.58	24.14	1.78	2.54
13	1.43	0.84	1.38	30.88	2.41	6.31
14	1.01	0.75	0.24	21.65	2.14	0.49
15	1.15	0.91	0.95	25.05	2.42	1.30
16	1.23	0.99	0.62	31.15	2.27	2.84
17	0.63	0.50	0.62	13.52	1.21	1.21
18	1.37	0.80	0.75	34.54	2.54	3.65
19	1.34	0.93	1.09	28.44	2.69	2.11
20	0.81	0.33	0.45	16.53	1.43	0.81
21	0.69	0.21	0.24	16.71	0.95	0.37
22	0.49	0.42	0.00	17.06	1.14	0.00
23	0.84	0.54	0.86	27.53	2.06	3.35
24	0.66	0.28	0.10	13.13	1.16	0.00
25	1.09	0.50	0.34	27.76	2.00	2.03
26	0.94	0.65	0.61	16.61	1.58	1.38
27	1.31	0.76	1.35	27.49	2.14	5.50
28	0.51	0.39	0.11	9.74	0.89	0.00
29	0.90	1.14	1.09	14.77	2.37	2.91
30	0.54	0.31	0.35	20.75	1.51	1.73
Average	0.85	0.61	0.60	19.62	1.67	1.84

5.2 Significance of the Correlation Coefficient between Mass and Pixel Percentage

This section first determines whether there exists a correlation between the mass and pixel percentages within the sample. If a correlation exists within the sample, it is determined whether

it can be utilised to model the relationship of the population. Finally, the mass percentage of MOG associated with one pixel percentage was determined.

5.2.1 Sample Data Correlation

The data presented in Table 5.1 were utilised to generate scatter plots depicting the relationship between mass and pixel percentages of the leaf, cane and wood components in figures 5.1, 5.2 and 5.3 respectively. It is evident that a positive correlation exists among the three scatter plots, as indicated by the positive slopes of their respective trend lines. Therefore, as the mass percentage increases, there is a corresponding increase in the pixel percentage.

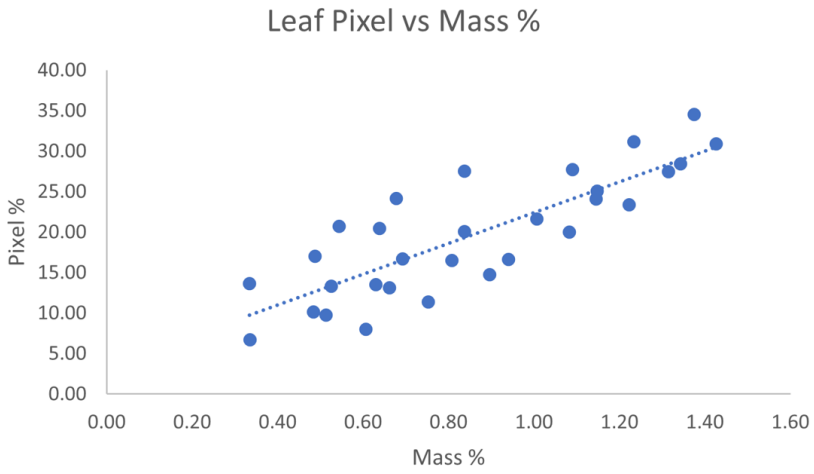

FIGURE 5.1: *Leaf pixel vs mass % scatter plot.*

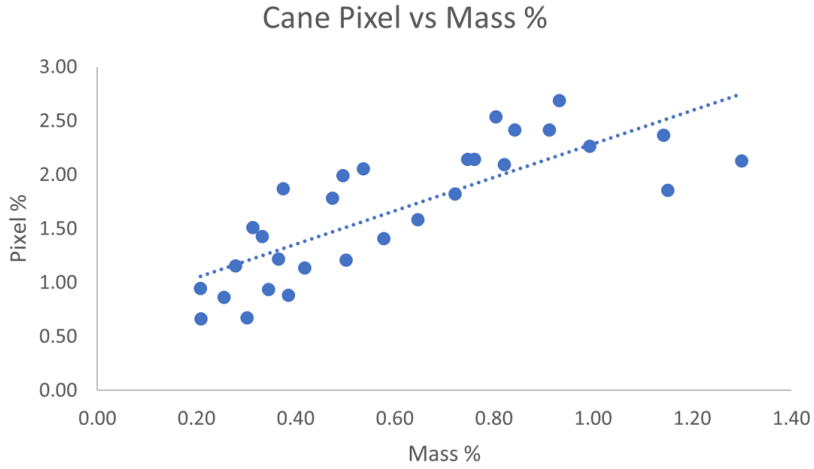

FIGURE 5.2: *Cane pixel vs mass % scatter plot.*

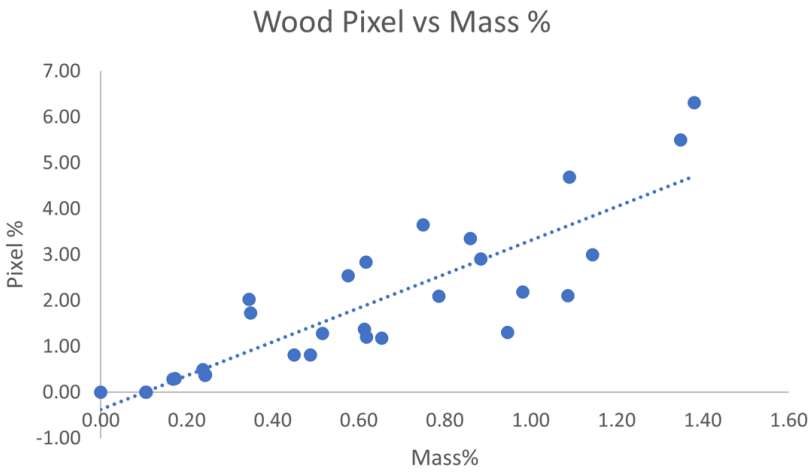

FIGURE 5.3: *Wood pixel vs mass % scatter plot.*

The correlation coefficient, which quantifies the magnitude and direction of the linear correlation between the x and y variables in each scatter plot, can be computed as follows:

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} \tag{5.1}$$

where:

- r is the correlation coefficient of the sample data set.
- x_i are the mass percentages in the samples.
- $\bar{x}$ is the mean of the mass percentages.
- y_i are the pixel percentages in the samples.
- $\bar{y}$ is the mean of the pixel percentages.

This formula provides a correlation coefficient value within the range of -1 to 1. A negative value signifies a negative correlation, a value of zero indicates a lack of correlation, whereas a positive value signifies a positive correlation. As the value increases, the correlation in either a positive or negative direction becomes progressively stronger.

In order to determine whether the linear relationship between the mass percentages and pixel percentages observed within the sample data set is appropriate to model the relationship of the population, it is necessary to conduct hypothesis tests to establish the statistical significance of the correlation coefficients.

5.2.2 Hypothesis Tests

A hypothesis test to test the significance of the correlation coefficient is performed in the following four steps [5]:

Step 1: State the null and alternative hypotheses. The null hypothesis in every test will be $H_0 : \rho = 0$ where ρ is the correlation coefficient for the population. This null hypothesis states that the correlation coefficient for the population is close to zero i.e. a significant linear relationship between the mass and pixel percentages does not exist in the population. The alternative hypothesis in every test will be $H_a : \rho > 0$ which states that the population correlation coefficient is significantly greater than zero and there exists a significant linear relationship between the mass and pixel percentages in the population.

Step 2: Determine the critical value with the sample size (n) and level of significance (α) using Pearson's Correlation Coefficient table for a one-tailed test in Appendix A.

Step 3: Calculate r for the sample data and compare it to the critical value. If $|r|$ is greater than the critical value, then reject H_0, otherwise fail to reject H_0.

Step 4: Interpret the outcome.

Hypothesis tests were conducted with a significance level of $\alpha = 0.01$ to determine the existence of a linear relationship between the percentage of ground truth leaf, cane and wood pixels in an image of a grape load and the mass percentage of leaves, canes and wood in the corresponding grape load.

5.2.2.1 Leaf Hypothesis Test

1. $H_0 : \rho = 0$, $H_a : \rho > 0$
2. $n = 30$, $\alpha = .01$ i.e. critical value $= 0.4226$

3. $r = 0.8268 >$ critical value i.e. reject H_0
4. There is sufficient evidence at a 99 % confidence level that there exists a linear relationship between the percentage of ground truth leaf pixels in an image of a grape load and the mass percentage of leaves in the corresponding grape load.

5.2.2.2 Cane Hypothesis Test

1. $H_0 : \rho = 0$, $H_a : \rho > 0$
2. $n = 30$, $\alpha = .01$ i.e. critical value $= 0.4226$
3. $r = 0.7960 >$ critical value i.e. reject H_0
4. There is sufficient evidence at a 99 % confidence level that there exists a linear relationship between the percentage of ground truth cane pixels in an image of a grape load and the mass percentage of canes in the corresponding grape load.

5.2.2.3 Wood Hypothesis Test

1. $H_0 : \rho = 0$, $H_a : \rho > 0$
2. $n = 30$, $\alpha = .01$ i.e. critical value $= 0.4226$
3. $r = 0.8610 >$ critical value i.e. reject H_0
4. There is sufficient evidence at a 99 % confidence level that there exists a linear relationship between the percentage of ground truth wood pixels in an image of a grape load and the mass percentage of wood in the corresponding grape load.

The hypothesis tests that were conducted above provide sufficient evidence that there is a significant correlation coefficient between the mass percentages and pixel percentages within the sample data set and that it can be utilised to model the relationship of the population. Therefore an increase in ground truth pixel percentage designated to MOG within a digital image indicates an increase in MOG mass percentage within the corresponding grape load.

5.2.3 Pixel-to-Mass Percentage Ratio Confidence Intervals

The leaf, cane and wood pixel-to-mass % ratios of the sample data set are documented in table 5.2. Each of the ratios in the table was determined by dividing the mass percentage of a sample

by the corresponding pixel percentage. The average and standard deviation (s) of each ratio was also calculated as shown in the table. The zero ratios in the "wood pixel-to-mass % ratio" column were excluded from the average and standard deviation calculations because it is impossible to have 1 % of ground-truth wood pixels in an image, but 0 % wood within the grape load.

TABLE 5.2: *Pixel-to-mass % ratios of leaves, canes and wood in a grape load sample.*

Sample	Leaf pixel-to-mass % ratio	Cane pixel-to-mass % ratio	Wood pixel-to-mass % ratio
1	0.0395	0.4472	0.5796
2	0.0759	0.3959	0.4012
3	0.0497	0.3131	0.6412
4	0.0312	0.4102	0.6009
5	0.0245	0.2958	0.5820
6	0.0417	0.2991	0.5533
7	0.0523	0.6103	0.3813
8	0.0477	0.3681	0.6078
9	0.0541	0.6196	0.4481
10	0.0661	0.3915	0.3751
11	0.0475	0.2003	0.2321
12	0.0280	0.2658	0.2267
13	0.0462	0.3486	0.2186
14	0.0465	0.3485	0.4836
15	0.0458	0.3773	0.7261
16	0.0396	0.4380	0.2169
17	0.0466	0.4146	0.5115
18	0.0398	0.3167	0.2051
19	0.0472	0.3467	0.5139
20	0.0489	0.2327	0.5552
21	0.0414	0.2189	0.6530
22	0.0286	0.3676	0.0000
23	0.0304	0.2612	0.2566
24	0.0504	0.2405	0.0000
25	0.0393	0.2483	0.1695
26	0.0566	0.4081	0.4440
27	0.0478	0.3552	0.2450
28	0.0526	0.4350	0.0000
29	0.0606	0.4825	0.3743
30	0.0262	0.2072	0.2012
Average	0.0451	0.3555	0.4224 (excluding 0 values)
s	0.0117	0.1046	0.1698 (excluding 0 values)

The interval of ratio values for the population can be estimated based on the sample data. To do this a confidence interval (CI) has to be estimated that represents the range of values for the population within a certain level of confidence.

The confidence interval can be estimated by the following formulas:

$$\text{CI} = \bar{x} \pm Z \cdot \frac{s}{\sqrt{n}} \tag{5.2}$$

$$\text{CI} = \bar{x} \pm t \cdot \frac{s}{\sqrt{n}} \tag{5.3}$$

where:

- $\bar{x}$ is the mean of the ratios in the sample.
- Z is the critical z-value
- t is the critical t-value.

The CI in equation 5.2 is determined based on the presupposition that the sample data adheres to an approximately normal distribution. The central limit theorem (CLT) posits that the mean of a sample approximates a normal distribution as the size of the sample increases. Furthermore, the CLT states that the sample should consist of a minimum of thirty instances for the CLT to hold [38]. Therefore equation 5.2 is utilised to estimate a CI in cases where the sample size is equal to or exceeds thirty. However, if the sample consists of less than thirty instances, equation 5.3 is utilised to estimate the confidence interval as it accounts for the uncertainty in smaller samples due to increased variability.

Both sample data for the leaf and cane pixel-to-mass % ratios contain a total of thirty values, while the sample data for the wood pixel-to-mass % ratios comprises only 27 values due to the exclusion of values equating to zero. Therefore equation 5.2 was used to estimate the CI for the leaf and cane pixel-to-mass % ratios while the CI for the wood pixel-to-mass % was estimated with equation 5.3. The confidence intervals were estimated with a 95 % level of confidence. By utilising the normal cumulative distribution function table and the student's t percentage points table in Appendix A, the Z and t critical values were determined as 1.96 and 2.056 respectively. The confidence intervals for the leaf, cane and wood pixel-to-mass percentages are shown in table 5.3.

TABLE 5.3: *Confidence intervals for the leaf, cane and wood pixel-to-mass percentages.*

Confidence Intervals for Pixel-to-Mass %			
	Leaf	Cane	Wood
Mean	0.0451	0.3555	0.4224
Lower bound	0.0409	0.3181	0.3552
Upper Bound	0.0493	0.3929	0.4895

Interpretation of the confidence intervals:

- For every 1 % of ground-truth leaf pixels within a grape load image, the leaves within the corresponding grape load weigh between 0.0409 % and 0.0493 % with an average of 0.0451 % of the grape load's total mass for 95 % of the time.

- For every 1 % of ground-truth cane pixels within a grape load image, the canes within the corresponding grape load weigh between 0.3181 % and 0.3929 % with an average of 0.3555 % of the grape load's total mass for 95 % of the time.

- For every 1 % of ground-truth wood pixels within a grape load image, the wood within the corresponding grape load weighs between 0.3552 % and 0.4895 % with an average of 0.4224 % of the grape load's total mass for 95 % of the time.

5.3 Example Usage of MOG Pixel-to-Mass % Confidence Intervals

The confidence intervals above can be used to determine the amount of MOG within a wine grape load given that the total mass of the grape load and pixel percentage of each MOG component is known. For example, table 5.4 presents the MOG mass for a wine grape load with a total mass of five tonnes and pixel percentages of 19.62 %, 1.67 % and 1.84 % for leaf, cane and wood respectively which are the average pixel percentages within the images of the grape load samples documented in table 5.1.

From table 5.4 it can be seen that the mean mass of MOG within a five-tonnes grape load with

TABLE 5.4: *MOG mass for a grape load mass of five tonnes and pixel percentages of 19.62 %, 1.67 % and 1.84 % for leaf, cane and wood respectively.*

MOG Mass (kg)				
	Leaf	Cane	Wood	Total
Mean	44.24	29.68	38.86	112.79
Lower bound	40.12	26.56	32.68	99.36
Upper Bound	48.36	32.81	45.03	126.20

the aforementioned pixel percentages is 112.79 kg which is 2.26 % of the grape load's total mass. At first, 2.26 % might sound like an insignificant percentage of MOG within the load, however, it will actually have a significant negative impact for the following reasons:

1. MOG emits an unpleasant bitter aroma and green characteristics onto the wine grapes when they make contact with each other. Therefore, even though there is just a small percentage of MOG within the grape load, it makes contact with a lot of wine grapes throughout the load, creating an unwanted aroma.

2. MOG causes bottlenecks as production systems slow down to remove it and 112.79 kg of MOG to be removed is certainly not insignificant. It can also lead to possible machine breakages if large pieces of wood are present, even if it is a small percentage of wood.

3. MOG is costly to transport and discard. Considering that 2.26 % of MOG within a five-tonnes grape load weighs about 113 kg, then ten five-tonnes grape loads quickly add up to 1130 kg of MOG which suddenly is quite a heavy pile of MOG that needs to be transported and discarded. Of course, there are much more than ten wine grape loads that arrive at the cellar per day and many of them weigh more than five tonnes.

This example demonstrated how the MOG pixel-to-mass % confidence intervals are intended to be used while it also highlights the impact of MOG, even if it is just a small percentage of the entire grape load. Winegrowers do not take responsibility for the MOG that is present in the grape loads which reiterates the importance for wine cellars to check for MOG when the grape load arrives at the weighbridge.

CHAPTER 6

MOG Identification System Overview, Verification and Validation

This chapter serves the purpose of first providing an overview of the MOG identification system drawing from the previous chapters followed by a verification and validation of the system.

6.1 MOG Identification System Overview

The introductory chapter stipulated that the MOG identification system must satisfy the criteria of automated MOG identification within a wine grape load while simultaneously being cost-efficient. It was also highlighted that a previous study conducted by the AWRI recommended the utilisation of hyperspectral imaging methods to identify MOG in wine grape loads. Two techniques were proposed, one of which necessitated the even dispersion of grape samples on a stationary tray and subsequent capture of images by a snapshot camera, while the other technique employed a line scan camera along with samples placed on a conveyor belt [18]. However, in addition to the laborious and time-consuming limitations of these techniques, hyperspectral cameras tend to be extremely expensive as explicated in section 6.1.2. Hence, in this study, computer vision algorithms were explored to identify MOG within wine grape load images captured with a standard camera.

6.1.1 MOG Identification System Process Flow

The process of the MOG identification system is activated upon the arrival of a tractor that is transporting a harvested load of wine grapes to the weighbridge at a wine cellar and concludes with the successful unloading of the grapes into the crusher. The operations at the weighbridge and crusher occur after the grapes have been harvested from a vineyard and prior to the wine-making process as indicated by the blue boundary in figure 6.1. At the weighbridge, the usual tasks such as weighing the load and measuring the pH and sugar ratings will occur, but additionally, a top-down photograph of the grape load will be captured by a camera positioned above the weighbridge. Subsequently, the tractor will proceed towards the crusher where the grapes are unloaded from the steel container into the crusher where another top-down photograph of the grapes before being crushed will be obtained. It is important to ensure that there are no surrounding objects that cast shadows onto the grape loads at the weighbridge and crusher during the image-capturing process as it will affect the performance of image processing algorithms. The images are then transmitted to a computer where the Mask R-CNN model as described in Chapter 4 will determine the percentage of MOG present within the images. The percentage of MOG determined within the image obtained at the weighbridge will be compared to that of the image obtained at the crusher to verify the percentage of MOG present in the wine grape load. Subsequently, a justifiable price that the wine cellar will pay for the grape load will be determined.

The verification of the MOG percentages between the weighbridge and crusher will prevent winegrowers from removing MOG from the surface of the grape load before it arrives at the weighbridge which may provide a false indication of the MOG within the grape load. The MOG percentages within the two images are expected to be the same due to the consistent mechanical harvesting method employed throughout the load, therefore it will be noticeable if the MOG on the surface has been removed. In circumstances where it is clear that the MOG on the surface has been removed, the wine cellar may contemplate imposing a penalty upon the winegrower.

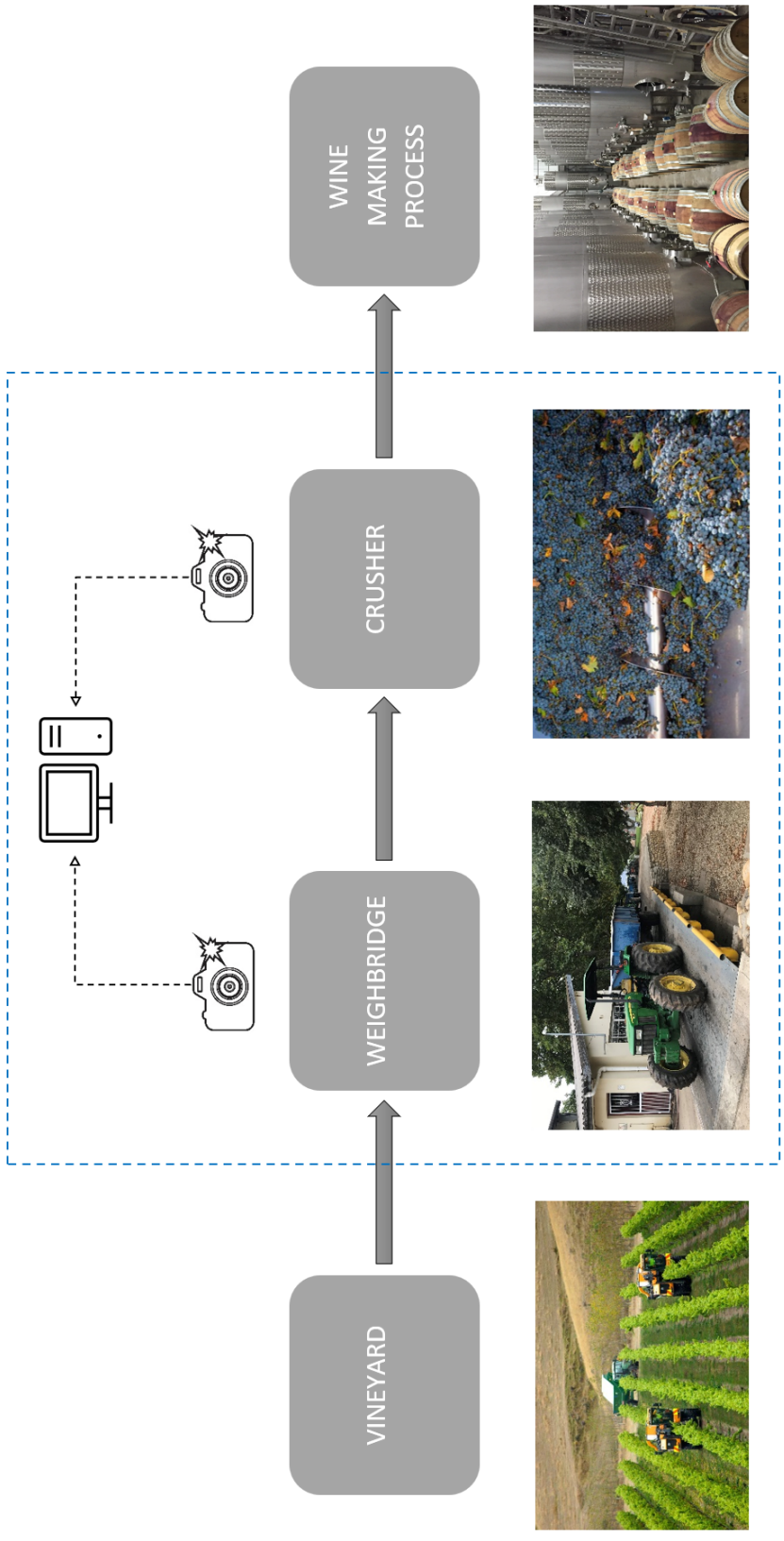

FIGURE 6.1: *Flow diagram of MOG identification system within the broader wine making process.*

6.1.2 Hyperspectral Camera vs Standard Camera Cost

The AWRI proposed two different techniques with two different hyperspectral cameras namely a snapshot and line scan camera. At the time of this research, the most inexpensive snapshot and line scan cameras that could be found cost approximately R360 000 and R225 000 respectively. The iPhone that was used in this project had a 12-megapixel camera, however, the primary reason smartphones went from 8-megapixel to 12-megapixel cameras was so they could utilise 4k video recording which is not necessary for the MOG identification system. Therefore an 8-megapixel camera is deemed sufficient which has the benefits of being more affordable than a 12-megapixel camera and it also uses less data meaning faster processing times while maintaining to capture high quality photos. It is also deemed necessary for the camera to be robust to rainy and windy weather conditions given that it will be affixed to an outdoor setting. The most cost-efficient camera found that meets the above criteria at the time of research is the "Hikvision 8MP Dome Camera" which costs approximately R4 000. It is clear that this camera is much more affordable than the hyperspectral cameras which makes the computer vision method with a standard camera the most cost-efficient MOG identification system.

6.2 MOG Identification System Verification

This section briefly discusses the verification of the computer vision model and the MOG pixel-to-mass % confidence intervals.

The optimal algorithm that was selected to identify and segment MOG within a wine grape load was the Mask R-CNN model with a ResNet-50 backbone. This model was evaluated with a test data set which consisted of image data the model had not seen before. The MOG-identified pixels were compared to the ground truth MOG pixels and it was confirmed that the model is able to accurately identify the majority of pixels associated with MOG within a digital image.

The MOG pixel-to-mass confidence intervals in table 5.3 were tested using the average pixel and mass percentage values of the grape load samples documented in table 5.1. The example usage of MOG pixel-to-mass % confidence intervals in Chapter 5.3 indicated that the mass of MOG for a five-tonnes grape load with the pixel percentages obtained from table 5.1 should weigh between

99.36 kg and 126.20 kg. By using the average mass percentages in table 5.1, it is calculated that the mass of MOG for a five-tonne grape load with similar will be 103 kg, which falls within the predicted mass interval.

Through these tests, the MOG identification system was verified.

6.3 MOG Identification System Validation

This section presents a validation of the MOG identification system with regard to its practicality from the standpoint of a professional. A semi-structured interview was conducted with a professional in the mass wine production industry. During the interview, the MOG identification system was presented, highlighting the operation thereof, the accuracy of the computer vision model, the statistical analysis behind the MOG pixel-to-mass % confidence intervals and how it compares to the costs of the hyperspectral imaging techniques proposed by the AWRI. Subsequently, the industry professional provided some feedback on the practicality of the system.

It was affirmed that the presented MOG identification system can be a viable system wine cellars can use to encourage wine growers to deliver better quality wine grape loads. The presence of MOG in wine grape loads becomes an increasing problem and since winegrowers are not penalised for the MOG, they do not make the effort to fine-tune their mechanical harvesting machines to harvest better quality wine grape loads. With this ongoing problem, wine cellars can not produce cost-efficient wine, therefore there is a great interest in such a MOG identification system at a wine cellar.

Cost efficiency is an important factor in making the MOG identification system affordable to wine cellars. By comparing the costs of the cameras of the system in this project to the costs of the cameras of the hyperspectral imaging techniques proposed by the AWRI, it was agreed that the system in this project is a more cost-efficient solution to identify MOG within wine grape loads.

It was stated that the placement of cameras at both the weighbridge and the crusher to double-check the presence of MOG within the wine grape loads would be very beneficial to strengthen the credibility of the system. It was further stated that the performance of the computer vision

model to identify MOG is adequate, however, it was also mentioned that if the performance is below 100 %, there is always room for improvement. Furthermore, the thorough statistical analysis behind the MOG pixel-to-mass % confidence intervals and interpretations thereof was well accepted.

Overall, it is believed that the MOG identification system presented in this project could add immense value to encourage winegrowers to deliver better quality wine grape loads with less MOG. It will improve the wine that can be produced while simultaneously improving the efficiency of the wine supply chain. Moreover, the implementation of this system is being considered to be tested for future harvesting seasons.

CHAPTER 7

Conclusion

This concluding chapter contains a discussion on whether the objectives of the project have been fulfilled, subsequently providing a summary of the project and offering recommendations for future research.

7.1 Fulfilment of Project Objectives

The objectives outlined in Chapter 1.3 were restated in the present section, accompanied by a discussion on how the objectives were fulfilled.

Objective I: Conduct a literature review related to computer vision.

This objective was fulfilled by conducting a thorough literature review on the following topics: machine learning and deep learning techniques pertaining to image classification and object detection, image segmentation methods and finally metrics to evaluate the performance of object detection and segmentation techniques. This review serves as the theoretical foundation for this research project.

Objective II: Preparation of a machine-harvested grape load image data set.

This objective was fulfilled by gathering images of machine-harvested grape loads at a wine cellar and then annotating the leaf, cane and wood objects within the image. The annotations were

performed by drawing polygons that precisely encompass the objects of interest. Furthermore, the data set was expanded with image augmentation techniques including flipping and rotation.

Objective III: Evaluation and selection of an optimal computer vision method to detect MOG in images of wine grape loads.

The stated objective was successfully fulfilled. Initially, traditional image segmentation techniques were explored, which subsequently led to the proposition of an approach involving the utilisation of thresholding methodologies on images that were transformed into the HSV colour space. A deep learning approach, namely Mask R-CNN, was subsequently applied for image segmentation. The ResNet-50 convolutional neural network was selected as the optimal backbone architecture based on specific criteria. The performance of both the traditional and deep learning based image segmentation methods was evaluated according to the metrics outlined in Chapter 2.4 with recall and F1 score being the most valuable metrics. The Mask R-CNN model demonstrated superior performance compared to the traditional segmentation approach and was deemed the most suitable method for detecting MOG in images of wine grape loads.

Objective IV: Analyses of MOG present on the surface of the grape load related to total MOG within the grape load using statistical techniques.

The stated objective was successfully fulfilled. A statistical analysis conducted on samples collected from grape loads provided sufficient evidence to prove that there exists a linear relationship between the quantity of MOG on the surface of a grape load and the mass of MOG within the corresponding grape load. Furthermore, confidence intervals for the mass percentage of MOG in a grape load associated with one pixel percentage of MOG in an image of the corresponding grape load were determined.

Objective V: Summarise findings and recommend areas for further research based on the findings in this report.

This objective is fulfilled in the subsequent sections.

7.2 Summary

The Mask R-CNN model, incorporating a ResNet-50 backbone, was selected as the optimal computer vision method to detect and segment MOG within an image of a wine grape load. The recall score, which represents the fraction of ground-truth MOG pixels in an image that was classified as MOG, was measured at 86.35 %. The precision score, signifying the fraction of MOG classified pixels that were correctly classified as MOG, achieved a score of 85.21 %. Furthermore, the F1 score, representing the harmonic mean between recall and precision, was calculated to be 85.78 %. The high F1 score indicates that the Mask R-CNN model is able to reliably detect and segment MOG within an image of a wine grape load.

A statistical analysis was conducted to determine the mass percentage that corresponds to each pixel percentage of the MOG components within a grape load image. It was determined that for every 1 % of leaf, cane and wood pixels in an image, the average corresponding mass are 0.0451 %, 0.3555 % and 0.4224 % respectively of the grape load's total mass. This information can enable a wine cellar to determine the mass of the MOG present in a grape load after the MOG on the surface has been quantified and the load's total mass has been obtained. As a result, the wine cellar can determine fair prices for the grape loads that take the presence of MOG into consideration.

7.3 Recommendations for Further Research

The following recommendations are made to expand on the work in this research study:

1. An additional data augmentation technique that adjusts the brightness of the images to artificially expand the data set can be implemented. This can be a useful technique since in reality some days are sunny and some days are cloudy which will change the light intensities on the grape loads. This augmentation technique on the data set might further improve the performance of the Mask R-CNN model.

2. The performance of the Mask R-CNN model can be implemented and evaluated on a data set with white grape loads. It is however expected that some fine-tuning will be necessary

since the leaves and white grapes which are actually green are very similar in colour.

3. Another recommendation for further research is to consider implementing the Mask R-CNN model on a mobile device, meaning that photos are taken with the device and evaluated directly on the device. However, this would require a new algorithm with a smaller size that requires less memory space but it might be worthy of exploration if it is something that is desired in the future.

References

[1] Abunasser B, Al-Hiealy M, Zaqout I & Abu-Naser S, 2023, *Convolution Neural Network for Breast Cancer Detection and Classification Using Deep Learning*, Asian Pac J Cancer Prev, **24(2)**, February, pp. 531–544.

[2] Agenbag GA, 2018, *A framework for capacity planning and capacity relocation in the wine supply network*, MA thesis, Stellenbosch University, Stellenbosch, South Africa.

[3] Aich S & Stavness I, 2017, *Leaf Counting with Deep Convolutional and Deconvolutional Networks*.

[4] Albawi S, Mohammed TA & Al-Zawi S, 2017, *Understanding of a convolutional neural network*, Proceedings of the 2017 international conference on engineering and technology (ICET), pp. 1–6.

[5] Amber, 2023, *Hypothesis testing for correlation (2.5.2): Edexcel A level maths: Statistics revision notes 2018*, [Online; accessed April 7, 2023], url: https://www.savemyexams.co.uk/a-level/maths_statistics/edexcel/18/revision-notes/2-data-presentation--interpretation/2-5-further-correlation--regression-a-level-only/2-5-2-hypothesis-testing-for-correlation/.

[6] Amidi A & Amidi S, no date, *Convolutional Neural Networks cheatsheet*, Available from https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks.

[7] *An Analysis of Object Detection, Image Segmentation, Image Classification*, Accessed: 2022-09-25, url: https://www.devfi.com/an-analysis-of-object-detection-image-segmentation-image-classification.

[8] Antoniadis P, 2022, *Activation Functions: Sigmoid vs Tanh*, Available from https://www.baeldung.com/cs/sigmoid-vs-tanh-functions.

[9] Apicella A, Donnarumma F, Isgrò F & Prevete R, 2021, *A survey on modern trainable activation functions*, Neural Networks, **138**, pp. 14–32.

[10] Badeka E, Kalampokas T, Vrochidou E, Tziridis K, Papakostas G, Pachidis T & Kaburlasos V, 2021, *Vision-based Vineyard Trunk Detection and its Integration into a Grapes Harvesting Robot*, International Journal of Mechanical Engineering and Robotics Research, **10**, July, pp. 374–385.

[11] Baeldung, 2023, *Intersection over Union for Object Detection*, [Online; accessed May 10, 2023], url: https://www.baeldung.com/cs/object-detection-intersection-vs-union.

[12] Bandyopadhyay H, 2022, *The Definitive Guide to Instance Segmentation [+V7 Tutorial]*, Available from https://www.v7labs.com/blog/instance-segmentation-guide.

[13] Bangare SL, Dubal A, Bangare PS & Patil S, 2015, *Reviewing Otsu's method for image thresholding*, International Journal of Applied Engineering Research, **10(9)**, pp. 21777–21783.

[14] Bansal B, Saini JS, Bansal V & Kaur G, 2012, *Comparison of various edge detection techniques*, Journal of Information and Operations Management, **3(1)**, pp. 103.

[15] Bell J & Dee HM, 2019, *Leaf segmentation through the classification of edges*, arXiv preprint arXiv:1904.03124.

[16] Bhardwaj A, no date, *What are neural networks? An introduction to machine learning algorithms*, Available from https://iaviral.medium.com/what-are-neural-networks-an-introduction-to-machine-learning-algorithms-6b73383c9089.

[17] Biswal A, 2022, *Convolutional Neural Network Tutorial*, Available from https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network#:~:text=Flattening%20is%20used%20to%20convert,layer%20to%20classify%20the%20image..

[18] Bob D, WenWen J, Simon N, Eric W & Paul P, 2019, *Digital solutions for grape quality measurement at the weighbridge*, (Unpublished) Technical Report, The Australian Wine Research Institute.

[19] Bobba R, 2019, *Taming the hyper-parameters of mask RCNN*, [Online; accessed May 5, 2023], URL: https://medium.com/analytics-vidhya/taming-the-hyper-parameters-of-mask-rcnn-3742cb3f0e1b.

[20] Boman J, 2020, *A deep learning approach to defect detection with limited data availability*, MA thesis, UMEÅ UNIVERSITY.

[21] Bouzenad AE, El Mountassir M, Yaacoubi S, Dahmene F, Koabaz M, Buchheit L & Ke W, 2019, *A Semi-Supervised Based K-Means Algorithm for Optimal Guided Waves Structural Health Monitoring: A Case Study*, Inventions, **4**, March, pp. 17.

[22] Chadalawada SK, 2020, *Real Time Object Detection and Recognition Using Deep Learning Methods*, MA thesis, Blekinge Institute of Technology.

[23] Csáji BC, 2001, *Approximation with artificial neural networks*, Faculty of Sciences, Etvs Lornd University, Hungary, **24(48)**, pp. 7.

[24] Cui N, 2018, *Applying Gradient Descent in Convolutional Neural Networks*, Journal of Physics: Conference Series, **1004**, April, pp. 012027.

[25] D E, 2019, *Accuracy, Recall amp; Precision*, [Online; accessed May 10, 2023], URL: https://medium.com/@erika.dauria/accuracy-recall-precision-80a5b6cbd28d.

[26] Dao E, 2022, *Evaluating object detection models*, [Online; accessed May 10, 2023], URL: https://erikdao.com/machine-learning/evaluating-object-detection-models/.

[27] Dass R & Devi S, 2012, *Image Segmentation Techniques*.

[28] Deacon Q, 2022, *Development of a smart trap for the surveillance of invasive fruit flies using internet of things and artificial intelligence*, MA thesis, Stellenbosch University.

[29] Deepchecks, 2022, *Learning Rate in Machine Learning*, [Online; accessed May 27, 2023], URL: https://deepchecks.com/glossary/learning-rate-in-machine-learning/.

[30] Dhanachandra N & Chanu YJ, 2017, *A survey on image segmentation methods using clustering techniques*, European Journal of Engineering and Technology Research, **2(1)**, pp. 15–20.

[31] Dhasaradhan K, Jaichandran R, Shunmuganathan K, Usha Kiruthika S & Rajaprakash S, "Hybrid machine learning model using decision tree and support vector machine for diabetes identification", in: *Data Engineering and Intelligent Computing*, Springer, 2021, pp. 293–305.

[32] Ding B, Qian H & Zhou J, 2018, *Activation functions and their characteristics in deep neural networks*, 2018 Chinese Control And Decision Conference (CCDC), pp. 1836–1841.

[33] Doğru A, Bouarfa S, Arizar R & Aydogan R, 2020, *Using Convolutional Neural Networks to Automate Aircraft Maintenance Visual Inspection*, Aerospace, **7**, December.

[34] Efford ND, 2000, *Digital Image Processing: A Practical Introduction Using Java*, Proceedings of the.

[35] Engelbrecht JA, 2021, *Machine Learning for Semantic Segmentation of Biomedical Imaging Data: Ensemble 2D UNets for CT and PET Images*, MA thesis, Stellenbosch University.

[36] Fisher R, Perkins S, Walker A & Wolfart E, 2003, *Gaussian Smoothing*, Available from https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm.

[37] Frost J, 2022, *Central limit theorem explained*, [Online; accessed April 15, 2023], url: https://statisticsbyjim.com/basics/central-limit-theorem/.

[38] Ganti A, 2023, *Central limit theorem (CLT): Definition and key characteristics*, [Online; accessed April 7, 2023], url: https://www.investopedia.com/terms/c/central_limit_theorem.asp.

[39] Girshick R, Donahue J, Darrell T & Malik J, 2014, *Rich feature hierarchies for accurate object detection and semantic segmentation*, Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580–587.

[40] Giuffrida V, Doerner P & Tsaftaris S, 2018, *Pheno-Deep Counter: a unified and versatile deep learning architecture for leaf counting*, The Plant Journal, **96**, August.

[41] Gonzalez S, Arellano C & Tapia JE, 2019, *Deepblueberry: Quantification of Blueberries in the Wild Using Instance Segmentation*, IEEE Access, **7**, pp. 105776–105788.

[42] Guo G, Wang H, Bell D, Bi Y & Greer K, 2003, *KNN Model-Based Approach in Classification*, Proceedings of the, pp. 986–996.

[43] Guo Y, Liu Y, Oerlemans A, Lao S, Wu S & Lew MS, 2016, *Deep learning for visual understanding: A review*, Neurocomputing, **187**, pp. 27–48.

[44] Gupta A, 2021, *A comprehensive guide on Optimizers in deep learning*, [Online; accessed May 12, 2023], url: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/.

[45] Hafeez M, Rashid M, Tariq H, Abideen Z, Alotaibi S & Sinky M, 2021, *Performance Improvement of Decision Tree: A Robust Classifier Using Tabu Search Algorithm*, Applied Sciences, **11**, July, pp. 6728.

[46] Hahn HK, 2005, *Morphological Volumetry: Theory, Concepts, and Application to Quantitative Medical Imaging*, PhD thesis, University of Bremen.

[47] He K, Gkioxari G, Dollár P & Girshick R, 2020, *Mask R-CNN*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **42(2)**, pp. 386–397.

[48] He K, Zhang X, Ren S & Sun J, 2016, *Deep Residual Learning for Image Recognition*, Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778.

[49] Heckmann C, 2021, *What is Color Space — Mastering Color in Post in Photo amp; Film*, [Online; accessed October 12, 2022], url: https://www.studiobinder.com/blog/what-is-color-space-definition/.

[50] Herculano-Houzel S, 2012, *The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost*, Available from https://www.pnas.org/doi/10.1073/pnas.1201895109#:~:text=Remarkably%2C%20at%20an%20average%20of,as%20other%20primates%20(26).

[51] IBM, 2023, *What is overfitting?*, [Online; accessed May 27, 2023], url: https://www.ibm.com/topics/overfitting.

[52] Jiang D, Zeng Z, Zhou S, Guan Y, Lin T & Lu P, 2020, *Three-Dimensional Magnetic Inversion Based on an Adaptive Quadtree Data Compression*, Applied Sciences, **10(21)**, pp. 7636.

[53] Jiao L & Zhao J, 2019, *A Survey on the New Generation of Deep Learning in Image Processing*, IEEE Access, **7**, pp. 172231–172263.

[54] Jooste C, 2016, *Performance Measurement Framework for the South African Wine Supply Chain: An Investigation of Packaged Products in the Local Market*, MA thesis, University of Stellenbosch.

[55] Kaganami HG & Beiji Z, 2009, *Region-based segmentation versus edge detection*, Proceedings of the 2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 1217–1221.

[56] Kang W.-X, Yang Q.-Q & Liang R.-P, 2009, *The comparative research on image segmentation algorithms*, Proceedings of the 2009 First international workshop on education technology and computer science, pp. 703–707.

[57] Kaur D & Kaur Y, 2014, *Various image segmentation techniques: a review*, International Journal of Computer Science and Mobile Computing, **3(5)**, pp. 809–814.

[58] Khan A, Baharudin B, Lee LH & Khan K, 2010, *A review of machine learning algorithms for text-documents classification*, Journal of advances in information technology, **1(1)**, pp. 4–20.

[59] Khawla BS, Othmani M & Kherallah M, 2021, *A novel approach for human skin detection using convolutional neural network*, The Visual Computer, **38**, April.

[60] Kleyn E, 2020, *Measuring techniques to determine hydrophobicity of polymer insulators*, MA thesis, Stellenbosch University, Stellenbosch, South Africa.

[61] Kotsiantis SB, Zaharakis ID & Pintelas PE, 2006, *Machine learning: a review of classification and combining techniques*, Artificial Intelligence Review, **26(3)**, pp. 159–190.

[62] Kumar S, Khan Z & Jain A, 2012, *A review of content based image classification using machine learning approach*, International Journal of Advanced Computer Research, **2(3)**, pp. 55–60.

[63] Kumawat D, 2019, *7 Types of Activation Functions in Neural Network.*, Available from https://www.analyticssteps.com/blogs/7-types-activation-functions-neural-network.

[64] Learn, 2023, *HUE, VALUE, SATURATION*, [Online; accessed May 19, 2023], url: https://learn.leighcotnoir.com/artspeak/elements-color/hue-value-saturation/.

[65] LeCun Y, Bengio Y & Hinton G, 2015, *Deep learning*, nature, **521(7553)**, pp. 436–444.

[66] Lee T, 2018, *How computers got shockingly good at recognizing images*, Available from https://arstechnica.com/science/2018/12/how-computers-got-shockingly-good-at-recognizing-images/.

[67] Leung KM, 2007, *Naive bayesian classifier*, Polytechnic University Department of Computer Science/Finance and Risk Engineering, **2007**, pp. 123–156.

[68] Liakos KG, Busato P, Moshou D, Pearson S & Bochtis D, 2018, *Machine learning in agriculture: A review*, Sensors, **18(8)**, pp. 2674.

[69] Lorena AC, Jacintho LF, Siqueira MF, De Giovanni R, Lohmann LG, De Carvalho AC & Yamamoto M, 2011, *Comparing machine learning classifiers in potential distribution modelling*, Expert Systems with Applications, **38(5)**, pp. 5268–5275.

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

[71] Maini R & Aggarwal H, 2009, *Study and comparison of various image edge detection techniques*, International journal of image processing (IJIP), **3(1)**, pp. 1–11.

[72] Mallick S, 2023, *Pre trained models for image classification - pytorch for Beginners*, [Online; accessed May 28, 2023], URL: https://learnopencv.com/pytorch-for-beginners-image-classification-using-pre-trained-models/.

[73] Marsland S, 2011, *Machine learning: an algorithmic perspective*, Chapman and Hall/CRC.

[74] MathWorks, 2023, *Types of Morphological Operations*, [Online; accessed May 18, 2023], URL: https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html.

[75] McCulloch WS & Pitts W, 1943, *A logical calculus of the ideas immanent in nervous activity*, The bulletin of mathematical biophysics, **5(4)**, pp. 115–133.

[76] McCullum N, no date, *The Flattening and Full Connection Steps of Convolutional Neural Networks*, Available from https://www.nickmccullum.com/python-deep-learning/flattening-full-connection/.

[77] Meyer D & Wien F, 2015, *Support vector machines*, The Interface to libsvm in package e1071, **28**, pp. 20.

[78] Mikulski B, 2019, *Understanding layer size in Convolutional Neural Networks*, Available from https://www.mikulskibartosz.name/understanding-layer-size-in-convolutional-neural-networks/.

[79] Miorelli R, Kulakovskyi A, Chapuis B, D'almeida O & Mesnil O, 2021, *Supervised learning strategy for classification and regression tasks applied to aeronautical structural health monitoring problems*, Ultrasonics, **113**, pp. 106372.

[80] Mukherjee S, 2022, *The annotated resnet-50*, [Online; accessed May 28, 2023], url: https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758.

[81] Murali N, 2021, *Image Classification vs Semantic Segmentation vs Instance Segmentation*, Available from https://nirmalamurali.medium.com/image-classification-vs-semantic-segmentation-vs-instance-segmentation-625c33a08d50.

[82] Nicholson C, no date, *A Beginner's Guide to Neural Networks and Deep Learning*, Available from http://wiki.pathmind.com/neural-network.

[83] O'Shea K & Nash R, 2015, *An introduction to convolutional neural networks*, arXiv preprint arXiv:1511.08458.

[84] Ongsulee P, 2017, *Artificial intelligence, machine learning and deep learning*, Proceedings of the 2017 15th international conference on ICT and knowledge engineering (ICT&KE), pp. 1–6.

[85] Panthula GA, 2018, *Object Detection Techniques Using Convolutional Neural Networks*, PhD thesis, The Florida State University.

[86] Pathak AR, Pandey M & Rautaray S, 2018, *Application of Deep Learning for Object Detection*, Procedia Computer Science, **132**, pp. 1706–1717.

[87] Perez H, Tah JH & Mosavi A, 2019, *Deep learning for detecting building defects using convolutional neural networks*, Sensors, **19(16)**, pp. 3556.

[88] Pokhrel S, 2019, *Beginners Guide to Convolutional Neural Networks*, Available from https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d.

[89] Preim B & Botha C, 2014, *Image Analysis for Medical Visualization*, Proceedings of the Visual Computing for Medicine (Second Edition), Boston, pp. 111–175.

[90] Pujara A, 2023, *Image classification with MobileNet*, [Online; accessed May 27, 2023], url: https://builtin.com/machine-learning/mobilenet.

[91] R M, 2011, *Edge Detection Techniques For Image Segmentation*, International journal of computer science and information technology, **3**, December, pp. 259–267.

[92] Rahmani AM, Yousefpoor E, Yousefpoor MS, Mehmood Z, Haider A, Hosseinzadeh M & Ali Naqvi R, 2021, *Machine learning (ML) in medicine: review, applications, and challenges*, Mathematics, **9(22)**, pp. 2970.

[93] Ray S, 2019, *A quick review of machine learning algorithms*, Proceedings of the 2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon), pp. 35–39.

[94] Redmon J, Divvala S, Girshick R & Farhadi A, 2016, *You Only Look Once: Unified, Real-Time Object Detection*, Proceedings of the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[95] Rish I, 2001, *An empirical study of the naive Bayes classifier*, Proceedings of the 22$^{\text{th}}$ IJCAI 2001 workshop on empirical methods in artificial intelligence, pp. 41–46.

[96] Sajid H, 2022, *What is image data augmentation?*, [Online; accessed October 7, 2022], url: https://www.picsellia.com/post/image-data-augmentation.

[97] Samson H, 2015, *Getting to know Activation Functions in Neural Networks.*, Available from https://towardsdatascience.com/getting-to-know-activation-functions-in-neural-networks-125405b67428#:~:text=Binary%20step%20function%20is%20a,said%20threshold%20neuron%20is%20deactivated.

[98] Saravanan R & Sujatha P, 2018, *A State of Art Techniques on Machine Learning Algorithms: A Perspective of Supervised Learning Approaches in Data Classification*, Proceedings of the 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 945–949.

[99] Saunders M, Lewis P, Thornhill A & Bristow A, 2019, *"Research Methods for Business Students" Chapter 4: Understanding research philosophy and approaches to theory development*, pp. 128–171 in.

[100] Saunders M & Tosey P, 2012, *The layers of research design.*, Rapport (Winter 2012/2013), pp. 58–59.

[101] Saxena S, 2021, *Image augmentation techniques for training deep learning models*, [Online; accessed October 7, 2022], URL: https://www.analyticsvidhya.com/blog/2021/03/image-augmentation-techniques-for-training-deep-learning-models/.

[102] Scharr H, Minervini M, French AP, Klukas C, Kramer DM, Liu X, Luengo I, Pape J.-M, Polder G & Vukadinovic D, 2016, *Leaf segmentation in plant phenotyping: a collation study*, Machine vision and applications, **27**, pp. 585–606.

[103] Shah D, 2022, *Mean average precision (MAP) explained: Everything you need to know*, [Online; accessed May 10, 2023], URL: https://www.v7labs.com/blog/mean-average-precision.

[104] Sharma P, 2021, *Understanding transfer learning for deep learning*, [Online; accessed May 27, 2023], URL: https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning/.

[105] Sharma P & Suji J, 2016, *A review on image segmentation with its clustering techniques*, International Journal of Signal Processing, Image Processing and Pattern Recognition, **9(5)**, pp. 209–218.

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

[107] Simonyan K & Zisserman A, 2015, *Very Deep Convolutional Networks for Large-Scale Image Recognition.*

[108] Singh A, Thakur N & Sharma A, 2016, *A review of supervised machine learning algorithms*, Proceedings of the 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 1310–1315.

[109] Sonawane M & Dhawale C, 2015, *A brief survey on image segmentation methods*, International Journal of Computer Applications, **975**, pp. 8887.

[110] Song Y & Yan H, 2017, *Image segmentation techniques overview*, Proceedings of the 2017 Asia Modelling Symposium (AMS), pp. 103–107.

[111] Srivastava T, 2019, *12 important model evaluation metrics for Machine Learning Everyone should know (updated 2023)*, [Online; accessed May 10 2023], url: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/.

[112] Sun R, 2020, *Optimization for Deep Learning: An Overview*, J. Oper. Res. Soc. China, **8**, pp. 249–294.

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

[114] TJ Mullinax GFG, 2015, *New grape harvester leaves MOG in the field*, [Online; accessed October 2, 2022], url: https://www.goodfruit.com/new-grape-harvester-leaves-mog-in-the-field/.

[115] Tripathi MK & Maktedar DD, 2020, *A role of computer vision in fruits and vegetables among various horticulture products of agriculture fields: A survey*, Information Processing in Agriculture, **7(2)**, pp. 183–203.

[116] van Zyl D, 1987, *Vineyards and Wine and History in South Africa*, Vineyards and Wine and History, pp. 33.

[117] Varikuti M, 2022, *What is the effect of batch size on model learning?*, [Online; accessed May 18, 2023], url: https://pub.towardsai.net/what-is-the-effect-of-batch-size-on-model-learning-196414284add.

[118] Vera R & Ossandón S, 2014, *On the Prediction of Atmospheric Corrosion of Metals and Alloys in Chile Using Artificial Neural Networks*, International journal of electrochemical science, **9**, September, pp. 7131–7151.

[119] Wenzel H, Smit D & Sardesai S, 2019, *A literature review on machine learning in supply chain management*, Proceedings of the Artificial Intelligence and Digital Transformation in Supply Chain Management: Innovative Approaches for Supply Chains. Proceedings of the Hamburg International Conference of Logistics (HICL), Vol. 27, pp. 413–441.

[120] *Wines of South Africa: SA industry Statistics*, no date, Available from https://www.wosa.co.za/The-Industry/Statistics/World-Statistics/.

[121] Wu X, Sahoo D & Hoi SC, 2020, *Recent advances in deep learning for object detection*, Neurocomputing, **396**, pp. 39–64.

[122] Yang K, Zhong W & Li F, 2020, *Leaf Segmentation and Classification with a Complicated Background Using Deep Learning*, Agronomy, **10(11)**.

[123] Yang X, Chen A, Zhou G, Wang J, Chen W, Gao Y & Jiang R, 2020, *Instance Segmentation and Classification Method for Plant Leaf Images Based on ISC-MRCNN and APS-DCCNN*, IEEE Access, **8**, pp. 151555–151573.

[124] yousefnami, 2021, *Why Does Increasing k Decrease Variance in kNN? Intuition, proof, and empirical results*, Available from https://towardsdatascience.com/why-does-increasing-k-decrease-variance-in-knn-9ed6de2f5061.

[125] Yu CH, 2009, *Designing and conducting mixed methods research.*, Organizational Research Methods, **12(4)**, pp. 801–804.

[126] Zhao Z.-Q, Zheng P, Xu S.-t & Wu X, 2019, *Object detection with deep learning: A review*, IEEE transactions on neural networks and learning systems, **30(11)**, pp. 3212–3232.

APPENDIX A

Statistical Tables

The critical Pearson's correlation coefficient value in table A.1 is determined by using the sample size to identify the row and α to identify the column of the critical value.

The critical Z-value in table A.2 is determined by searching for the Z-value that corresponds with the probability of 1-(α level/2).

The critical t-value in table A.3 is determined by first calculating the degrees of freedom (ν = n-1) to identify the row in which the desired t-value is and then matching it with the column which is equal to 1-(α /2).

TABLE A.1: *Pearson's correlation coefficient critical values table*

Pearson's Correlation Coefficient Critical Values Table						
	Level of Significance α (one-tailed test)					
n	0.1	0.05	0.025	0.01	0.005	0.0005
3	0.9511	0.9877	0.9969	0.9995	0.9999	1.0000
4	0.8000	0.9000	0.9500	0.9800	0.9900	0.9990
5	0.6870	0.8054	0.8783	0.9343	0.9587	0.9911
6	0.6084	0.7293	0.8114	0.8822	0.9172	0.9741
7	0.5509	0.6694	0.7545	0.8329	0.8745	0.9509
8	0.5067	0.6215	0.7067	0.7887	0.8343	0.9249
9	0.4716	0.5822	0.6664	0.7498	0.7977	0.8983
10	0.4428	0.5494	0.6319	0.7155	0.7646	0.8721
11	0.4187	0.5214	0.6021	0.6851	0.7348	0.8470
12	0.3981	0.4973	0.5760	0.6581	0.7079	0.8233
13	0.3802	0.4762	0.5529	0.6339	0.6835	0.8010
14	0.3646	0.4575	0.5324	0.6120	0.6614	0.7800
15	0.3507	0.4409	0.5140	0.5923	0.6411	0.7604
16	0.3383	0.4259	0.4973	0.5742	0.6226	0.7419
17	0.3271	0.4124	0.4821	0.5577	0.6055	0.7247
18	0.3170	0.4000	0.4683	0.5425	0.5897	0.7084
19	0.3077	0.3887	0.4555	0.5285	0.5751	0.6932
20	0.2992	0.3783	0.4438	0.5155	0.5614	0.6788
21	0.2914	0.3687	0.4329	0.5034	0.5487	0.6652
22	0.2841	0.3598	0.4227	0.4921	0.5368	0.6524
23	0.2774	0.3515	0.4132	0.4815	0.5256	0.6402
24	0.2711	0.3438	0.4044	0.4716	0.5151	0.6287
25	0.2653	0.3365	0.3961	0.4622	0.5052	0.6178
26	0.2598	0.3297	0.3882	0.4534	0.4958	0.6074
27	0.2546	0.3233	0.3809	0.4451	0.4869	0.5974
28	0.2497	0.3172	0.3739	0.4372	0.4785	0.5880
29	0.2451	0.3115	0.3673	0.4297	0.4705	0.5790
30	0.2407	0.3061	0.3610	0.4226	0.4629	0.5703
31	0.2366	0.3009	0.3550	0.4158	0.4556	0.5620
32	0.2327	0.2960	0.3494	0.4093	0.4487	0.5541

TABLE A.2: *The normal cumulative distribution function table.*

NORMAL CUMULATIVE DISTRIBUTION FUNCTION										
Z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7703	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767
2.0	0.9772	0.9778	0.9783	0.9788	0.9793	0.9798	0.9803	0.9808	0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838	0.9842	0.9846	0.9850	0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875	0.9878	0.9881	0.9884	0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904	0.9906	0.9909	0.9911	0.9913	0.9916
2.4	0.9918	0.9920	0.9922	0.9925	0.9927	0.9929	0.9931	0.9932	0.9934	0.9936
2.5	0.9938	0.9940	0.9941	0.9943	0.9945	0.9946	0.9948	0.9949	0.9951	0.9952
2.6	0.9953	0.9955	0.9956	0.9957	0.9959	0.9960	0.9961	0.9962	0.9963	0.9964
2.7	0.9965	0.9966	0.9967	0.9968	0.9969	0.9970	0.9971	0.9972	0.9973	0.9974
2.8	0.9974	0.9975	0.9976	0.9977	0.9977	0.9978	0.9979	0.9979	0.9980	0.9981
2.9	0.9981	0.9982	0.9982	0.9983	0.9984	0.9984	0.9985	0.9985	0.9986	0.9986
3.0	0.9987	0.9987	0.9987	0.9988	0.9988	0.9989	0.9989	0.9989	0.9990	0.9990
3.1	0.9990	0.9991	0.9991	0.9991	0.9992	0.9992	0.9992	0.9992	0.9993	0.9993
3.2	0.9993	0.9993	0.9994	0.9994	0.9994	0.9994	0.9994	0.9995	0.9995	0.9995
3.3	0.9995	0.9995	0.9995	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9997
3.4	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9998
3.5	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998
3.6	0.9998	0.9998	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
3.7	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
3.8	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
3.9	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

TABLE A.3: *The student's t percentage points table.*

STUDENT'S t PERCENTAGE POINTS											
ν	60.0%	66.7%	75.0%	80.0%	87.5%	90.0%	95.0%	97.5%	99.0%	99.5%	99.9%
1	0.325	0.577	1.000	1.376	2.414	3.078	6.314	12.706	31.821	63.657	318.31
2	0.289	0.500	0.816	1.061	1.604	1.886	2.920	4.303	6.965	9.925	22.327
3	0.277	0.476	0.765	0.978	1.423	1.638	2.353	3.182	4.541	5.841	10.215
4	0.271	0.464	0.741	0.941	1.344	1.533	2.132	2.776	3.747	4.604	7.173
5	0.267	0.457	0.727	0.920	1.301	1.476	2.015	2.571	3.365	4.032	5.893
6	0.265	0.453	0.718	0.906	1.273	1.440	1.943	2.447	3.143	3.707	5.208
7	0.263	0.449	0.711	0.896	1.254	1.415	1.895	2.365	2.998	3.499	4.785
8	0.262	0.447	0.706	0.889	1.240	1.397	1.860	2.306	2.896	3.355	4.501
9	0.261	0.445	0.703	0.883	1.230	1.383	1.833	2.262	2.821	3.250	4.297
10	0.260	0.444	0.700	0.879	1.221	1.372	1.812	2.228	2.764	3.169	4.144
11	0.260	0.443	0.697	0.876	1.214	1.363	1.796	2.201	2.718	3.106	4.025
12	0.259	0.442	0.695	0.873	1.209	1.356	1.782	2.179	2.681	3.055	3.930
13	0.259	0.441	0.694	0.870	1.204	1.350	1.771	2.160	2.650	3.012	3.852
14	0.258	0.440	0.692	0.868	1.200	1.345	1.761	2.145	2.624	2.977	3.787
15	0.258	0.439	0.691	0.866	1.197	1.341	1.753	2.131	2.602	2.947	3.733
16	0.258	0.439	0.690	0.865	1.194	1.337	1.746	2.120	2.583	2.921	3.686
17	0.257	0.438	0.689	0.863	1.191	1.333	1.740	2.110	2.567	2.898	3.646
18	0.257	0.438	0.688	0.862	1.189	1.330	1.734	2.101	2.552	2.878	3.610
19	0.257	0.438	0.688	0.861	1.187	1.328	1.729	2.093	2.539	2.861	3.579
20	0.257	0.437	0.687	0.860	1.185	1.325	1.725	2.086	2.528	2.845	3.552
21	0.257	0.437	0.686	0.859	1.183	1.323	1.721	2.080	2.518	2.831	3.527
22	0.256	0.437	0.686	0.858	1.182	1.321	1.717	2.074	2.508	2.819	3.505
23	0.256	0.436	0.685	0.858	1.180	1.319	1.714	2.069	2.500	2.807	3.485
24	0.256	0.436	0.685	0.857	1.179	1.318	1.711	2.064	2.492	2.797	3.467
25	0.256	0.436	0.684	0.856	1.178	1.316	1.708	2.060	2.485	2.787	3.450
26	0.256	0.436	0.684	0.856	1.177	1.315	1.706	2.056	2.479	2.779	3.435
27	0.256	0.435	0.684	0.855	1.176	1.314	1.703	2.052	2.473	2.771	3.421
28	0.256	0.435	0.683	0.855	1.175	1.313	1.701	2.048	2.467	2.763	3.408
29	0.256	0.435	0.683	0.854	1.174	1.311	1.699	2.045	2.462	2.756	3.396
30	0.256	0.435	0.683	0.854	1.173	1.310	1.697	2.042	2.457	2.750	3.385
35	0.255	0.434	0.682	0.852	1.170	1.306	1.690	2.030	2.438	2.724	3.340
40	0.255	0.434	0.681	0.851	1.167	1.303	1.684	2.021	2.423	2.704	3.307
45	0.255	0.434	0.680	0.850	1.165	1.301	1.679	2.014	2.412	2.690	3.281
50	0.255	0.433	0.679	0.849	1.164	1.299	1.676	2.009	2.403	2.678	3.261
55	0.255	0.433	0.679	0.848	1.163	1.297	1.673	2.004	2.396	2.668	3.245
60	0.254	0.433	0.679	0.848	1.162	1.296	1.671	2.000	2.390	2.660	3.232
∞	0.253	0.431	0.674	0.842	1.150	1.282	1.645	1.960	2.326	2.576	3.090